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

VLT Software

CCD Detectors Control Software

User Manual

Doc.No. VLT-MAN-ESO-17240-0672

Issue 1.6

Date 25/09/98

Prepared	P.Duhoux	25/09/98	
	Name	Date	Signature
Approved	A.Longinotti		
	Name	Date	Signature
Released	G.Raffi		
	Name	Date	Signature

Change Record

Issue/Rev.	Date	Section/Page affected	Reason/Initiation/Document/Remarks
1.0	28/09/95	All	First issue
1.1	13/03/96	All (see change bars)	Updated to CCD sw release FEB96
1.2	31/07/96	2.2	Added. Explanation on parallelism
		2.3.3	Added values for loop in exposure status
		2.4	Added. Explanation of image processing
		2.10	Command WAIT added Commands OBJBRGT, OBJCLOS, OBJECTS removed
		2.14, 2.15	Examples of usage modified
		3.1.5	Template for image processing user function
		4.2.1	Sun platform supported
		4.6.1	CCDDXF, CCDOSLX_SERVER added INTROOT removed
		4.6.5	LCU configuration simplified
1.3	31/10/96	2.4	New image processing functionality
		2.12	ccdIpStatus_uifClass modified
		2.13.2	New setup keywords implemented
		2.13.4	ipCenVal re-introduced in the public OLDB
		4.6.1	INS_SETUPPATH removed INS_USER added (SPR 960300)
		4.6.6	Scan System configuration added (SPR 960543)
1.4	25/04/97	1.4	Added reference to TCCDS Systems User man.
		2.4.1	WS image processing added (SPR 960204)
		2.10	New command STANDAL
		2.11	C library **libccdObj.a** added in replacement of commands OBJBRGT, OBJCLOS, OBJECTS
		2.12	new GUI classes added
		2.13.2	DET.EXP.TIMEREP added. Some keywords moved to DET.WIN<i> prefix
		2.13.3.1	World Coordinates Display supported
		2.13.3.2	More hierarchical keywords added
		2.13.4	OLDB Attributes for World Coord display added
		3.1	Manual pages for ccdObjXXX functions added
		3.3	manual pages for ccdMidasBgStart/Stop added manual page for ccdDcsWcs.sh added
		4.2.1	Hw requirements updated (SPR 970090)
		4.6.1	CCDOSLX_SERVER not used any more

Change Record

Issue/Rev.	Date	Section/Page affected	Reason/Initiation/Document/Remarks
1.5	19/11/97	1.2	Scope explicitely restricted to ACE systems
		2.2	Usage of exposure_2 removed: not supported, although still in OLDB for compatibility
		2.3.2	Multi-step exposures supported for TCCDS
		2.5	Operational state INITIALISED removed (SPR 970045)
		2.11	ccdGetIndexFromId not documented any more, although still available
		2.13.4	Various attributes added in public part of OLDB
		2.14	Various setup parameters added (SPR 970016, 970017, 970083, 970148, 970158)
		2.17	FITS logs added
1.6	25/09/98	2.6.2	New description of Image Processing on LCU
		2.10	Improved description of StartAg command
		2.14	Updated description of setup keywords
		2.16.2	Updated LCU public database attributes
		3.1	Added IP man-pages
		3.5 - 3.9	Updated files

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

The CCD control software is going to be used by several categories of users. The information related to this software package has been spread over several documents, in order to give each user an easier access only to the information he needs.

1. If you want to **install** from scratch the CCD software, read this manual, section *Installation Guide*.

   Pre-requisites:

   a. Experience with the installation of other VLT sw packages.

   b. Knowledge of UNIX and VxWorks operating systems

   c. Familiarity with the VLT sw standard environment.

2. If you want to **configure** the CCD camera you intend to use, read [16].

   Pre-requisites:

   a. What at point 1. must already have been done.

   b. Knowledge of the CCD camera characteristics.

3. If you want to **operate** the CCD camera as simple stand-alone instrument, read [14]:

   a. What at point 2. must already have been done.

   b. Some experience with CCD cameras operations.

4. If you want to **interact programmatically** with the CCD sw, read this manual, section *User's Guide*.

   Pre-requisites:

   a. What at point 2. must already have been done.

   b. Experience with programming within the VLT sw environment.

5. If you have to **develop software to control special hardware** for a CCD camera sub-system (e.g. shutter), read [17]

   Pre-requisites:

   a. What at point 2. must already have been done.

   b. Experience with programming within the VLT sw environment, and in particular with LCU common software.

This document is the User Manual of the CCD DCS Software.
It is intended to provide people, who implement applications using the CCD DCS Software programmatically, with all the necessary information.

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

In addition to the **Introduction**, this manual contains three major chapters:

User's Guide: it contains information about:

1. Definitions: exposure id, type and status, operational mode and state.
1. Image processing
2. Startup/Shut-down
3. Include files
4. Command Interface to the CCD DCS software
5. Function Interface (C library) to the CCD DCS software
6. GUI panel classes for applications using the CCD DCS software.
7. Data Dictionary
8. Setup parameters and files
9. Image files. Interface with Observation Software: who does what and when.
10. Public section of the CCD DCS on-line database.
11. Configuration and operational logs
12. Example of usage of the CCD DCS Software from applications using scientific or technical CCD cameras.

Reference: manual pages of functions available to external software, as well as scripts and processes interacting with external sw and their Command Definition Tables, include files and examples of files of various type to be used as templates.

Installation Guide: It contains a step-by-step description of the procedure to be followed to install the software.

Error Messages: reference to files containing information about errors produced by the CCD software.

The following table presents the complete list of documents available about the CCD software, together with a summary of the contents and the category of users who may be interested in reading them.

Document #	Title	Contents	Users
VLT-MAN-ESO-17240-0672	CCD Software - User Manual	• Software installation • Programmatic interface	• Responsible for VLT sw installation • Software developers
VLT-MAN-ESO-17240-0917	CCD Cameras - User Manual - Software part	CCD stand-alone	• Responsible for CCD cameras and chips tests. • Software developers
VLT-MAN-ESO-17240-0918	CCD Cameras - Maintenance Manual - Software part	• Camera configuration • Periodical maintenance • Trouble-shooting	• Responsible for camera installation. • Software developers

Document #	Title	Contents	Users
VLT-MAN-ESO-17240-0919	CCD Software - Device Control Libraries - User Manual	Integration of special hardware in the CCD sw (e.g. FORS shutter).	• Software developers (only if special hw used).

1.2 Scope

The present document is intended to be used for both scientific and technical CCD cameras based on the ACE controller.
For CCD cameras using the FIERA controller, refer to [28].

1.3 Applicable Documents

The following documents, of the exact issue shown, form a part of this document to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this document, the contents of this document shall be considered as a superseding requirement.

[1] VLT-PRO-ESO-10000-0228, 1.0 10/03/93 ---- VLT Software Programming Standards

[2] VLT-SPE-ESO-17212-0001, 2.0 12/04/95 ---- VLT Instrumentation Sw Specification

[3] VLT-SPE-ESO-17240-0227, 1.0 08/04/93 ---- CCD Detectors Control Software Specification

[4] VLT-SPE-ESO-17240-0385, 2.1 15/07/96 ---- INS Common Software Specification

[5] GEN-SPE-ESO-19400-0794, 1.1 25/11/97 ---- ESO Data Interface Control Document

[6] VLT-ICD-ESO-17240-19400, 2.6 23/05/97 ---- ICD between VLT Control sw and Archive

1.4 Reference Documents

The following documents contain additional information and are referenced in the text.

[7] VLT-MAN-ESO-17200-0642, 1.8 31/05/97 ---- VLT Common Software Installation Manual

[8] VLT-MAN-ESO-17200-0888, 1.0 17/08/95 ---- VLT Common Software Overview

[9] VLT-MAN-SBI-17210-0001, OCT98 ---- LCU Common Software User Manual

[10] VLT-MAN-ESO-17210-0619, 1.7 30/04/97 ---- Central Control Software User Manual

[11] VLT-MAN-ESO-17240-0816, 1.0 28/09/95 ---- CCD DCS Sw Maintenance Manual WS part

[12] VLT-MAN-ESO-17240-0932, 1.0 28/09/95 ---- CCD Stand-alone Sw Maint. Manual WS part

[13] VLT-MAN-ESO-17240-0817, 1.0 28/09/95 ---- CCD Sw Maintenance Manual LCU part

[14] VLT-MAN-ESO-17240-0917, 1.5 19/11/97 ---- CCD Cameras User Manual, Sw part

[15] VLT-MAN-DJO-11700-0001, 1.0 03/05/96 ---- Technical CCD Systems User Manual

[16] VLT-MAN-ESO-17240-0918, 1.4 19/11/97 ---- CCD Cameras Maintenance Manual, Sw part

[17] VLT-MAN-ESO-17240-0919, 1.0 28/09/95 ---- CCD Software DCL User Manual

[18] VLT-MAN-ESO-13600-0789, 2.0 in prep. ---- B016 Driver for VxWorks. User manual

[19] VLT-MAN-ESO-13600-0527, 2.0 10/03/95 ---- INMOS SW on VxWorks. User manual

[20] VLT-MAN-ESO-13600-0686, 1.2 20/10/95 ---- LCU-Transputer Interface Sw User Manual

[21] VLT-MAN-ESO-17210-0707, 1.5 30/04/97 ---- On Line Database Loader User Manual

[22] VLT-MAN-ESO-17210-0690, 3.1 05/05/97 ---- Graphical User Interface User Manual

[23]VLT-MAN-ESO-17240-0637,2.3 30/10/96 ---- INS Common Sw - dxf User Manual
[24]VLT-MAN-ESO-17240-0726,1.7 07/05/97 ---- INS Common Sw - slx User Manual
[25]VLT-MAN-ESO-17240-0853,1.2 07/05/96 ---- INS Common Sw - oslx User Manual
[26]VLT-MAN-ESO-17240-0866,2.4 01/10/96 ---- INS Common Sw - rtd User Manual
[27]VLT-MAN-ESO-17240-0725,1.3 07/05/96 ---- INS Common Sw - pco User Manual
[28]VLT-MAN-ESO-13640-1388,1.0 13/11/97 ---- FIERA sw User Manual

1.5 Abbreviations and Acronyms

The following abbreviations and acronyms are used in this document:

ACE	Array Control Electronics
CCD	Charge-Coupled Device
CCS	Central Control Software
CDT	Command Definition Table
CPU	Central Processing Unit
DCL	Device Control Library
DCS	Detector Control Software
ESO	European Southern Observatory
FDDI	Fiber Distributed Data Interface
FIERA	Fast Imager Electronic Readout Assembly
FITS	Flexible Image Transport Format
HW	Hardware
INS	Instrumentation Software Package
I/O	Input/Output
LAN	Local Area Network
LCC	LCU Common Software
LCU	Local Control Unit
MIDAS	Munich Image Data Analysis System
N/A	Not Applicable
SCCD	Scientific CCD
SW	Software
TBC	To Be Clarified
TBD	To Be Defined
TCCD	Technical CCD
TCS	Telescope Control Software
TIM	Time Interface Module
TRS	Time Reference System
UIF	(Portable) User Interface (Toolkit)
VLT	Very Large Telescope

VME	Versa Module Eurocard
WAN	Wide Area Network
WS	Workstation

1.6 Glossary

No special definition is introduced in this manual

1.7 Stylistic Conventions

The following styles are used:

bold in the text, for commands, file names, etc. as they have to be typed.

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

`teletype` for examples.

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

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

1.8 Naming conventions

This implementation follows the naming conventions as outlined in [2].

1.9 Problem Reporting/Change Request

The form described in [7] shall be used.

2 USER'S GUIDE

This part of the document provides a description of the programmatic interface of the CCD software. It gives software engineers, writing applications using a ACE based CCD camera, information on what they have to implement in their code to be able to interact with the CCD software.

For people having no experience with the CCD software yet, it is highly recommendable, although not necessary, before starting to read this section, first to have a look at the interactive usage of the CCD software as stand-alone instrument (see [14]) and possibly try it out; it can help to get a better idea of the way how the CCD software works, as well as of the functionality currently implemented.

2.1 Overview

The CCD Detectors Control Software is distributed over three hardware platforms: WS, LCU and ACE transputer network.

The current standard configuration (see [14]) foresees that all sub-systems (readout, shutter, telemetry and temperature) are controlled by the ACE software.
The present document assumes that the standard configuration is used.
On the other hand, the design of the CCD LCU software is such that it allows to replace the standard control software of one or more sub-systems with other software developed for special hardware (e.g. a special shutter). For more information on this subject, see [17]

For a description of the hardware and software needed to run the CCD software, see section 4.2.

A program accesses the functionality provided by the CCD software in the following way:

1. It retrieves information about configuration and status of the CCD camera through functions available in the library *libccd* (see section 2.11). Whenever the information wanted cannot be retrieved by means of library function calls, a direct access to the public part of the on-line database is possible (see section 2.16).
2. It asks the CCD software for services by mean of commands using the CCS message system (see section 2.10).
3. It can use the CCS event system to be informed about changes of status of the CCD system (see section 2.19.1).

Note: In the examples shown in the next sections we take the following setting (see section 4.6.1):

```
setenv CCDNAME myccd
setenv RTAPENV wmyccd
setenv CCDLENV lmyccd
```

2.2 Parallelism

One of the main requirements for the CCD software is to optimize observation time, performing as much as possible operations in parallel.
For this reason, parts of image already been read-out are transferred to the Workstation and saved in FITS file and/or displayed (depending from the setup), while the next image portion is being read-out.; this parallelism of read-out and image transfer guarantees that the time interval between the end of the read-out and the completion of image display or FITS file on disk is minimized.

2.3 Exposure definitions

2.3.1 CCD Exposure Id

In order to be able to uniquely identify an exposure among those already finished and the running one, an Identification number is associated to each exposure.

The Id used by the CCD sw for this purpose is an integer sequential number (vltINT32) returned by the CCD DCS software as reply parameter to the command START.
Applications must pass this Id as parameter to all commands dealing with a running exposure (e.g. ABORT, see section 3.2.1).
Special values (see *ccd.h*) can be used; applications are encouraged to use them whenever possible:

1. **`ccdEXP_NEXT`** refers to the next exposure to be started (typically for command SETUP).
2. **`ccdEXP_LAST`** refers to the last exposure started and possibly still running.

The Id defined here does NOT replace the exposure Id as defined in the INS Common Software document [4]. It is just a different Id, which has to be used in the communication between the CCD sw and external sw (e.g. INS OS). In particular:

1. The OS expoId follows the rules in [4] and is generated by OS. It is used to communicate between OS and the software on top of it (GUI panel, Sequencer etc.). Applications using only TCCDS only can just ignore this point.
2. The CCD expoId is a different variable and is generated by CCD at START time. It is used only for the internal communication between CCD sw and higher level sw (e.g. INS OS or TCS AG sw).

   In particular the following points should be considered:

   a. Command SETUP, parameter *expoId*:

      i. Use **`ccdEXP_NEXT`** (-1) for an exposure not started yet (command START not sent yet).

      ii. Use **`ccdEXP_LAST`** (0), or the value returned by the CCD sw to the START command, for a running exposure (e.g. to change the exposure time).

   b. Command START.

      No *expoId* has to be passed as parameter. On the contrary, the reply contains the *expoId* value of the exposure just started.

   c. Other commands related to a running exposure (ABORT etc.), parameter *expoId*:

      Use **`ccdEXP_LAST`** (0), or the value returned by the CCD sw to the START command.

2.3.2 Exposure types

The CCD Detector Control Software distinguishes among three different types of exposure (see also [3]).

1. **Dark**. It has the following characteristics:

   a. It consists of one single integration

   b. The CCD is always wiped before starting a new exposure.

   c. During the integration the shutter is kept closed. For cameras without shutter (e.g. TCCDS), only dark with 0 integration time (bias) are meaningful and possible.

   d. After the integration, the CCD is read out. The read-out mode is determined by setup parameters.
2. **Normal**. It has the following characteristics:
   a. It consists of one single integration
   b. The CCD is normally wiped before starting a new exposure, unless the time interval between the expected start of the new exposure and the last read-out or wipe is shorter than the value of the database attribute *exposures.wipeTimeTol*.
   c. During the integration the shutter, if present, is kept open.
   d. After the integration, the CCD is read out. The read-out mode is determined by setup parameters.
3. **Multi-step** *(in the present release supported only for systems without shutter)*. It has the following characteristics*:*
   a. It consists of many integrations. They may have the same or different duration.
   b. The CCD is normally wiped before starting a new exposure, unless the time interval between the expected start of the new exposure and the last read-out or wipe is shorter than the value of the database attribute *exposures.wipeTimeTol*.
   c. During the integration the shutter, if present, is kept open.
   d. After each integration, excluding the last one, the exposure is automatically paused (the shutter, if present, is closed)
   e. During pauses between consecutive integrations, rows may be shifted on chip, according to the value of the setup parameter **`DET.READ.SHIFT`** (see also **`ccdREADSHIFT`** in file *ccd.h*).
   f. For systems with shutter, the next integration is started with a CONT command. For systems without shutter (e.g. TCCDS) the integration is started as soon the rows shift operation has been completed.
   g. After the integration, the CCD is read out. The read-out mode is determined by setup parameters.

It is responsibility of higher level software (e.g. Observation Software) to translate the various supported exposure types as defined in [5] into one of the types known by CCD DCS for the setup parameter **`DET.EXP.TYPE`**. It is recommended to use variables of type *ccdEXPTYPE* or the macros associated to the corresponding strings (e.g. **`ccdEXP_DARK_STR`**), defined in *ccd.h*, for all operations dealing with the CCD DCS exposure type.

2.3.3 Exposure status

An attribute in the on-line database (*exposures:exposure_1.expStatus,* see also section 2.16) is dedicated to the storage of the status of a running exposure (or the end status of the last finished exposure if no new exposure has been started yet). It is a bit-field value and the meaning of each bit is (less significant bit is indicated as 1):

1. NOT ACTIVE
2. PENDING. Wiping the chip before integrating.
3. INTEGRATING.
4. PAUSED
5. READ-OUT ACTIVE

6. PROCESSING IMAGE DATA
7. TRANSFERRING IMAGE DATA
8. COMPLETED SUCCESSFULLY
9. COMPLETED WITH ERROR
10. ABORTED
11. FINITE LOOP OF REPEATED EXPOSURES ACTIVE
12. INFINITE LOOP OF REPEATED EXPOSURES ACTIVE

Note that, in case of a loop of repeated exposures, the status does NOT change during the whole loop. Applications interested in the status of each single exposure should not use the repeated exposure feature.
The exposure status returned by the CCD sw in the reply to some commands (e.g. STATUS and WAIT) is coded in the same way. Additionally, for those exposures already completed, whose final status is not available any more (e.g. because a new exposure has been meanwhile started), the returned status is **`ccdEXP_DONE`** (see *ccd.h*)
See also file *ccd.h* for all macros and associated values (e.g. **`ccdEXP_INACTIVE`**).

2.4 Operational modes and simulation

A detailed description of the operational modes implemented and their meaning is given in [14].

1. The operational mode can be set:
   a. Programmatically.
      - Write directly in the Workstation on-line database (see section 2.16).
      - Run Workstation script *ccdDcsDbSave.sh* (see section 3.3). Needed only if LCU used.
   b. Interactively through a GUI panel (see [14] or manual page of *ccdOpMode* for more):
      $ccdOpMode "cwp=<alias>$CCDNAME" &

The current operational mode can be retrieved through the command *STATUS* (see 2.10).

It is recommended to use variables of type *ccdOPMODE*, defined in *ccd.h*, for all operations dealing with the CCD software operational mode

Note: any change to the current operational mode have effect only after the CCD software is brought to *OFF* state (Shutdown, see also 2.5).

2.5 Operational states

The list of operational states implemented in the CCD software is given in [14]. In Table 1 are shown the commands needed to do any change in the operational state of a CCD camera.

Table 1 CCD DCS state transition commands

From \ To	OFF	LOADED	STAND-BY	ON-LINE
OFF	---	ccdDcsStart.sh	---	---
LOADED	OFF	---	STANDBY	ONLINE
STAND-BY	OFF	---	STANDBY	ONLINE
ON-LINE	OFF	---	STANDBY	ONLINE

See section 2.10 for more information about commands available.

The current operational state can be retrieved through the command *STATUS* (see 2.10).

It is recommended to use variables of type *ccdSTATE*, defined in *ccd.h*, for all operations dealing with the CCD software operational state.

2.6 Image processing

This section applies mainly to applications dealing with TCCDS.

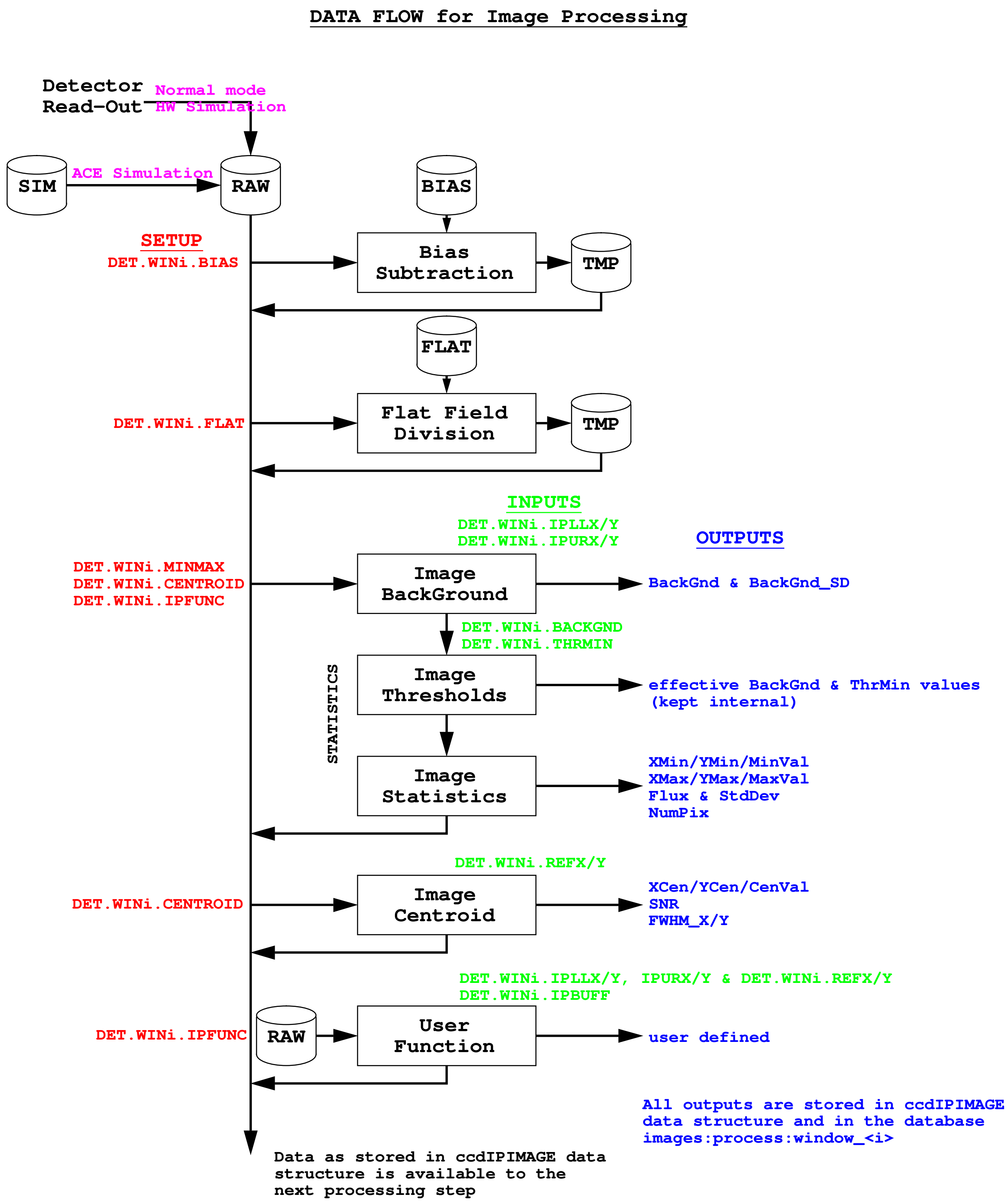
2.6.1 Image processing on Workstation

The CCD sw provides a C library, called *libccdObj* (see also section 2.11), containing functions which, using on-line Midas (see also [27]), determine the location and intensity of objects within a frame contained in a FITS file. The functions available allow to:

1. Determine and return the location and intensity of **all objects** detected in a frame.
2. Determine and return the location and intensity of the **brightest** detected **object** in a frame.
3. Determine and return the location and intensity of the detected **object** in a frame, **closest to a reference point.**

2.6.2 Real-time image processing on LCU

The CCD sw provides facilities to perform real-time image processing on the CCD LCU while the image data are being read-out and before they are transferred to Workstation.



Currently these facilities consist of:

1. Bias subtraction (see section 2.14 on setup parameter **DET.WIN<i>.BIAS**). In order to perform this operation a bias frame must have been stored before in the LCU memory; this can be done in two ways:
   a. Execute a bias exposure with the command BIAS (see section 2.10).

   b. At startup, the CCD sw looks for a file in the image directory with name *bias_<CCDNAME>.fits* and loads it automatically in the LCU memory.

2. Flat field division (see section 2.14 on setup parameter **DET.WIN<i>.FLATF**). In order to perform this operation a flat-field frame must have been stored before in the LCU memory; this can be done in two ways:
   a. Execute a flat field exposure with the command FLAT (see section 2.10).
   b. At startup, the CCD sw looks for a file in the image directory with name *flat_<CCDNAME>.fits* and loads it automatically in the LCU memory.
3. Computation of statistics over image (see section 2.14 on setup parameter **DET.WIN<i>.MINMAX**).

   The results are stored in the LCU on-line database (see also section 2.16) in the same sequential order as below:
   a. ***images:process:window<index>.ipXMin***. X coordinate of pixel with minimum intensity.
   b. ***images:process:window<index>.ipYMin***. Y coordinate of pixel with minimum intensity.
   c. ***images:process:window<index>.ipMinVal***. Minimum intensity over frame.
   d. ***images:process:window<index>.ipXMax***. X coordinate of pixel with maximum intensity
   e. ***images:process:window<index>.ipYMax***. Y coordinate of pixel with maximum intensity.
   f. ***images:process:window<index>.ipMaxVal***. Maximum intensity over frame.
   g. ***images:process:window<index>.ipFlux***. Average pixel intensity.
   h. ***images:process:window<index>.ipStdDev***. Pixel intensity standard deviation.
   i. ***images:process:window<index>.ipBGnd***. Background pixel intensity.
   j. ***images:process:window<index>.ipBGnd_SD***. Background pixel intensity std dev..
   k. ***images:process:window<index>.ipNumPix***. Number of pixels with intensity > *ThrMin*.
4. Calculation of centroiding over image; it implies step 3. 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 set to 0), and finally computes the centre of gravity of the resulting image. The way how the background level and threshold are determined is discussed in section 2.14 (setup parameter **DET.WIN<i>.CENTROID**).

   The results are stored in the LCU on-line database (see also section 2.16) in the same sequential order as below:
   a. ***images:process:window<index>.ipSNR***. Estimated Signal-to-Noise Ratio.
   b. ***images:process:window<index>.ipFWHM_X***. Estimated object FWHM along X-axis.
   c. ***images:process:window<index>.ipFWHM_Y***. Estimated object FWHM along Y-axis.
5. Hook to user function implementing a special application-dependant algorithm (see section 2.14 on setup parameters **DET.WIN<i>.IPFUNC** and **DET.WIN<i>.IPBUFF**). It implies step 3. This function must comply with the prototype defined in *ccd.h* (see **ccdIPUSERFUNC**). A template is also provided with the tape: *<TAPEROOT>/CCD/ccdip/src/ccdipUsrFuncTemplate.*

Steps 1. to 5. can be individually enabled/disabled. If more than one is enabled, then they are executed, for each window being read-out, sequentially in the same order as above (user function always last). Example: the setup defines that two windows have to be read-out, on the first one statistics and user function have to be called, on the second one statistics, centroiding and user function. The sequencial order is then: statistics on window 1, user function on window 1, statistics on window 2, centroiding on window 2, user function on window 2.

The results of the centroiding calculation for each window are stored in the LCU on-line database

after both steps 4. and 5., if enabled, have been performed. They are stored in the same sequential order as below:

a. ***images:process:window<index>.ipXCen***. Error vector along X-axis.

b. ***images:process:window<index>.ipYCen***. Error vector along Y-axis.

c. ***images:process:window<index>.ipCenVal***. Intensity of the centroid nearest pixel.

If no object is detected (e.g. all pixels are below the threshold), all attributes are set to 0.
It is responsibility of the user function, whenever enabled, to return in the *ccdIPIMAGE* strcucture (see *ccd.h*) the computed centroiding results, if supposed to do centroiding, otherwise to leave these values unchanged, if supposed to do something else.

2.7 Startup

The CCD DCS software provides a startup script.

Type from the Workstation shell (see also manual page at section 3.3):

```
$ccdDcsStart.sh
```

Note 1: As from the NOV97 release, the script *ccdDcsWarmStart.sh* has become obsolete and it still available for backwards compatibility reasons only. Application sw is invited to use *ccdDcsStart.sh* only.

Note 2: The script *ccdDcsStart.sh* should not be confused with *ccdStart.sh* (see also [14]):

- *ccdDcsStart.sh* starts the DCS control sw both on WS and LCU.
- *ccdStart.sh* starts only the CCD stand-alone GUI panel: no control process is started.

2.7.1 Startup verification

To verify if the CCD sw startup was successful:

1. The operational state (see section 2.5) should be 2 (*ccdLOADED*). Check from the UNIX shell:
   $ dbRead "<alias>$CCDNAME.opState"
   INT32/UINT32 value = 2
2. The following processes must be running in the WS environment (use option *Rtap Perf. Monitor* from *ccsei* to check)
   a. ccdconCI_*myccd*
   b. ccditWs_*myccd*
3. The following processes must be running in the LCU environment (use *lqsPrintLocalTbl* from the VxWorks shell to check):
   a. ccdcon_*myccd*
   b. ccdit_*myccd*
   c. ccdrdt_*myccd*
   d. ccdip_*myccd*
   e. ccdsht_*myccd* (only if shutter declared available; not available for TCCDs)
   f. ccdtel_*myccd* (only if telemetry control declared available)
   g. ccdtmp_*myccd* (only if temperature control declared available; not available for SCCDs)

2.8 Shutdown

The CCD DCS software provides a shutdown script.

Type from the Workstation shell (see also manual page at section 3.3):

```
$ccdDcsStop.sh
```

2.9 Include files

Applications dealing with CCD DCS software have to include the file *ccd.h* (see also 3.5).

2.10 Command Interface

The only CCD process accessible via CCS commands from external software is ccdconCI_*$CCDNAME*, running on the Workstation (e.g. if **CCDNAME** is set to *myccd*, the process name is *ccdconCI_myccd*)**.**
The command interface between CCD software and external software is represented therefore by the Command Definition Table of that process.
Here a simple list of all the commands implemented is given. For more details, see Command Definition Table at section 3.2.1

1. **Abort.** Abort last exposure started or specified one.
2. **Bias.** Execute one bias exposure and save it for later usage.
3. **Cont.** Continue an exposure paused by the user or automatically if multi-step.
4. **Display.** Start exposure and display image.
5. **Dump.** Dump last image read or the specified one from VME memory. If no image available, read the chip(s) and then transfer to WS.
6. **End.** Stop current integration and read chip immediately.
7. **Exit.** Same as *Off*
8. **Flat.** Execute one normal exposure and save it as flat field image for later usage.
9. **GrabImage.** Save on disk file in FITS format the image currently being displayed.
10. **Init.** Initialise CCD sw and/or HW. It brings the system to INITIALISED state.
11. **Kill.** Kill the CCD main task.
12. **Off.** Bring the system to OFF state. Terminate the whole CCD software
13. **Online.** Bring the system to ON-LINE state.
14. **Pause.** Pause current integration.
15. **Setup.** Define or just check setup values for next exposure or change values for the specified running exposure. See also sections 2.14 and 3.7.1.
16. **Standal.** CCD sw is used in stand-alone. All FITS information is merged in the image file and optionally the archive is informed of new images produced.
17. **Standby.** The whole CCD system is brought to STAND-BY state.
18. **Start.** Start new exposure (possibly repeated). Setup parameters can be specified as well. One single reply is returned as soon the exposure has started.
19. **StartAg.** Start autoguiding. An infinite loop is started using the current setup overloaded with the dedicated setup file *ccdCmdStartAg.det*. Each iteration consists of:
    a. Exposure

   b. Computation of error vector (using **DET.WIN1.CENTROID** = **ccdCEN_THRESHOLD_STR**, **DET.WIN1.BACKGND** = **ccdBCKGND_FLUX_WINDOW_SELF** and **DET.WIN1.THRMIN** = **ccdTHRMIN_3SIGMA_WINDOW_SELF**).
   c. Results are stored in the LCU database.

20. **StartLoop.** Start an infinite loop of repeated exposures using the current setup overloaded with the dedicated setup file *ccdCmdStartLp.det*.
21. **StartTelemetry.** Start monitoring of telemetry values
22. **StartTemperature.** Start monitoring of temperature values
23. **StartWipe.** Start periodical wiping of the chip(s). Setup parameters can be specified as well.
24. **Status.** Check and update current status. Return information in the reply.
25. **Stop**. Stop in an ordered way an on-going loop of repeated exposures.
26. **StopTelemetry.** Stop monitoring of telemetry values
27. **StopTemperature.** Stop monitoring of temperature values
28. **StopWipe.** Stop periodical wiping of the chip(s).
29. **Version.** It returns the current version of the CCD sw.
30. **Wait.** Wait for the specified exposure to complete.
31. **StopWait.** Stop loop and Wait for the specified exposure to complete.

External applications are recommended to use the macros for commands and parameters as specified in *ccd.h* (e.g. command **ccdCMD_SETUP**, parameter **ccdEXP_TYPE**)

2.11 C libraries

The library *libccd* provides routines for external software:

1. ***ccdGetConf***. Get the current camera configuration (see also types ccdCAMERA and ccdCONFIG in *ccd.h*). See manual page in section 3.1
2. ***ccdGetCIName***. Get name of the CCD process to send commands to (see also section 2.10). See manual page in section 3.1
3. ***ccdCheckSetup***. Check a complete setup and return computed values (see types ccdSETUP and ccdSETUPRES in *ccd.h*). See manual page in section 3.1

The library *libccdObj* provides object detection functions. It uses on-line Midas and all functions implemented assume that a background Midas session is running (see also manual pages of the scripts *ccdMidasBgStart.sh* and *ccdMidasBgStop.sh* in section 3.3 on how to start/stop a MIdas background session). The test program *ccdTestObj* and script *ccdTestMidasObj.sh*, distributed with the tape and located in the directory *CCD/ccd/ws/test*, can be used as example.
The functions implemented are:

1. ***ccdObjAll***. Detects objects in a frame from a FITS file and returns their location and intensity. See manual page in section 3.1
2. ***ccdObjBright***. Detects objects in a frame from a FITS file and returns location and intensity of the brightest one. See manual page in section 3.1
3. ***ccdObjClose***. Detects objects in a frame from a FITS file and returns location and intensity of the closest one to a reference point. See manual page in section 3.1
4. ***ccdObjDisplay***. Displays objects in a frame with Midas display. See manual page in section 3.1

2.12 Graphical User Interface CCD classes

It is foreseen to provide applications using the CCD software with a set of classes, produced with the CCS panel editor, to be instantiated in bigger application panels.
The classes provided are available in the public library ***libccdGuiPublic.tcl*** (which has to be registered to any panel using it, see [22]):

1. ***ccdExpStatus_uifClass***. It displays the status of an exposure. See also Fig.2 and manual page at section 3.4

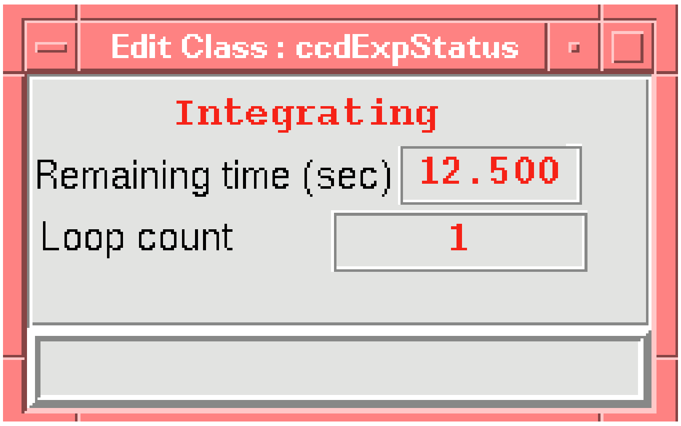


Fig.1 Exposure status GUI class

2. ***ccdExpSetup_uifClass.*** It displays the main setup parameters of the most recently started exposure. See also Fig.2 and manual page at section 3.4

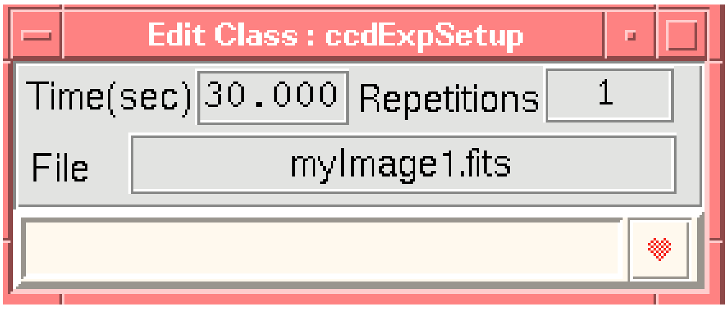


Fig.2 Exposure setup GUI class

3. ***ccdReadoutSetup_uifClass.*** It displays the read-out setup parameters of the most recently started exposure. See also Fig.2 and manual page at section 3.4

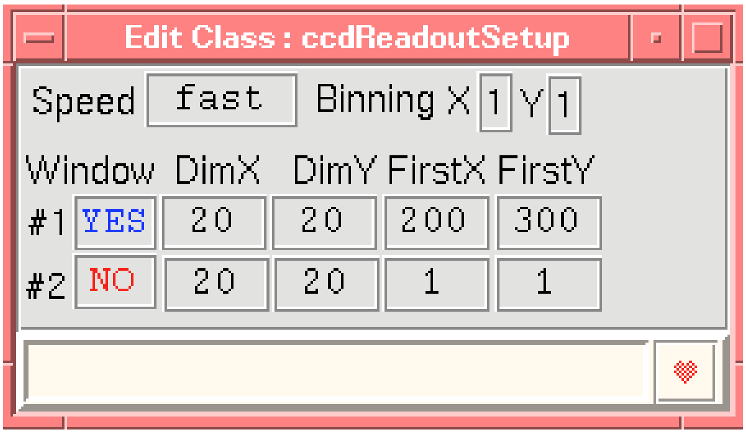


Fig.3 Read-out setup GUI class

4. ***ccdIpStatus_uifClass***. It displays the status of real-time image processing on LCU. See also Fig.4 and manual page at section 3.4.

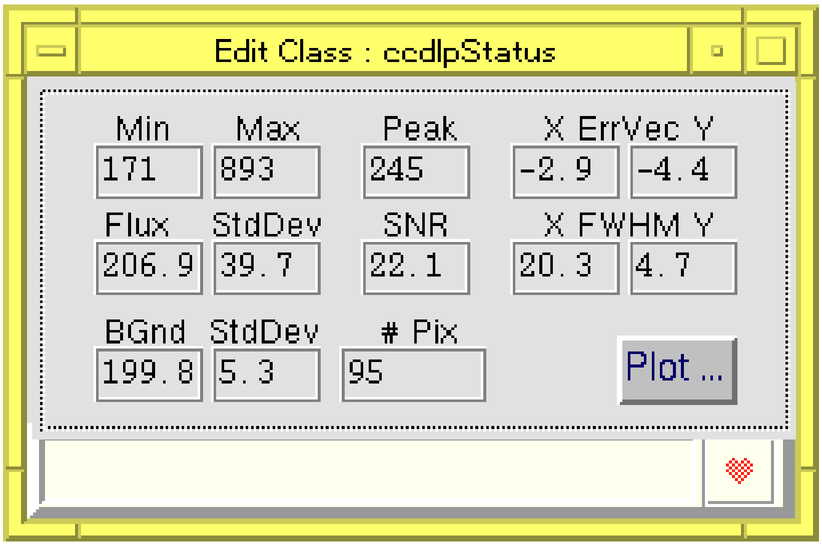


Fig.4 Real-time image processing status GUI class

2.13 Data Interface Dictionary

The CCD software needs a Data Interface Dictionary file to be able to operate with setup keywords and files and to create proper headers in FITS files. The name of the dictionary is specified through the environment variable **CCDDID** (see section 4.6.1). The default dictionary (if **CCDDID** is not defined) is **CCDDCS** (full name *ESO-VLT-DIC.CCDDCS*)

2.14 Setup parameters and files

Among all kinds of setup files defined in [4], the CCD sw handles only detector setup files. They are normally created and modified through GUI panels and can be used as input for exposure setup definition (setup file name is passed as parameter to the SETUP command as well as single setup keywords).

The keywords currently implemented, and documented in the CCD DCS Data Dictionary (ESO-VLT-DIC.CCDDCS) are listed in the setup file shown in section 3.7.1 (it represents a complete setup file for CCD DCS). Information about the parameters for the SETUP command are also given in the CDT of ccdconCI (section 3.2.1) and in the include files *ccd.h and ccdDbPublic.h* (section 3.5)

2.14.1 Exposure

1. The parameter **`DET.EXP.TYPE`** defines the type of exposure and can take the following string values (see section 2.3.2 and *ccd.h*):
   a. **`ccdEXP_NORMAL_STR`**
   b. **`ccdEXP_DARK_STR`**. Possible only for cameras with shutter (SCCDS only)
   c. **`ccdEXP_MULTI_STR`**. *Currently implemented for TCCDS only*
2. The parameter **`DET.EXP.NREP`** indicates how many times the defined exposure must be executed.

   The special value **`ccdREPEAT_FOREVER`** (see *ccd.h*) defines an infinite loop of repeated exposures.

   Applications interested in knowing the intermediate status of an exposure must set it to 1 (no repetition).
3. The parameter **`DET.FRAM.TYPE`** indicates the type of frame resulting from the next exposure. Possible values are **`ccdFRAME_NORMAL`**, **`ccdFRAME_BIAS`**, **`ccdFRAME_FLAT`**. The LCU CCD sw keeps in memory at any time the last acquired image of each type.

2.14.2 Times

1. The parameter **`DET.EXP.TIMEREP`** defines the period (in secs) between consecutive exposures for repeated exposures. A value of 0 means no delay between exposure (next started as soon previous one finishes).

   It is ignored in case of single exposures (**`DET.EXP.NREP`** set to 1)
2. The parameter **`DET.WIN1.UIT<i>`** specifies the (sub-)integration time in seconds. If **`DET.EXP.TYPE`** is not **`ccdEXP_MULTI_STR`**, only **`DET.WIN1.UIT1`** must be used.
3. The parameter **`DET.EXP.WIPETIM`** indicates if a wipe is wished or not for exposures in a loop. The wipe for the first exposure is always done.

   Meaning:

   n < 0 never wipe

   n = 0 always wipe (default)

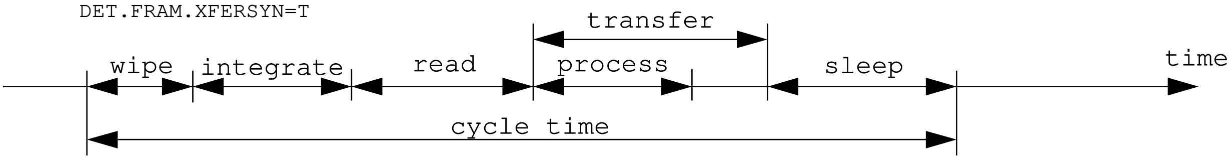
   n > 0 wipe if (actual time - last readout time) > n (in msec)

   For frame transfer chips (TCCDS), if no wipe is performed, readout and next integration overlap.

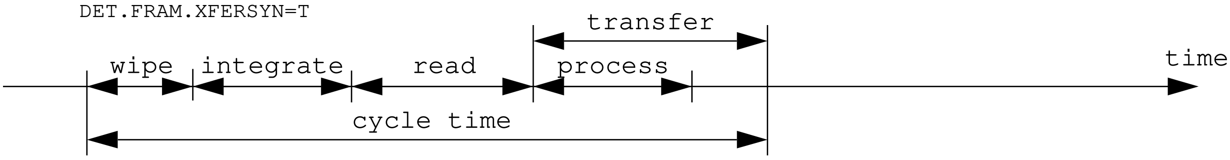
   Note: If **`DET.EXP.WIPETIM`** is negative, depending on the setup, it may result in a discrepancy between the requested exposure time **`DET.WIN1.UIT1`** and the effective exposure time if the exposure cycle time is larger than the exposure time: the effective exposure time is obtained as the maximum of (cycle time, exposure time) where the cycle time is the sum of exposure time + readout + process + transfer times.

The following diagrams illustrate some timing cases:

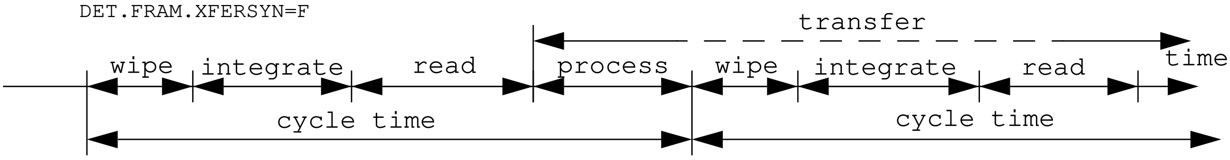
```
Case1: DET.EXP.WIPETIM=0
       DET.EXP.TIMEREP > wipe + integrate + read + transfer
       DET.FRAM.XFERSYN=T
```




```
Case2: DET.EXP.WIPETIM=0
       DET.EXP.TIMEREP=0
       DET.FRAM.XFERSYN=T
```

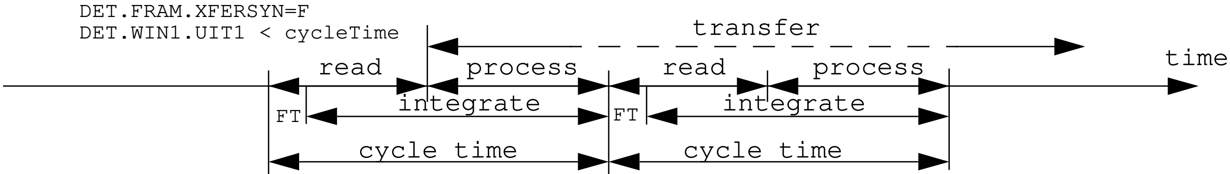



```
Case3: DET.EXP.WIPETIM=0
       DET.EXP.TIMEREP=0
       DET.FRAM.XFERSYN=F
```



In this case, the image transfer is performed asynchronously.

```
Case4: DET.EXP.WIPETIM<0
       DET.EXP.TIMEREP=0
       DET.FRAM.XFERSYN=F
       DET.WIN1.UIT1 < cycleTime
```



In this case, the effective integration time may be larger than requested.

They apply for Technical CCD on small windows so that the transfer starts at readout completion. In the last case, the integration of the next exposure starts at the end of the Frame Transfer (FT) performed before the readout of the current one, it terminates with the start of the next readout FT. The asynchronous data transfer is performed by the CPU when time is available; thus it may end a couple of exposures later depending on the LCU load.

2.14.3 Readout

1. The parameter **`DET.READ.CLKIND`** is an integer number (range 1-10). It determines which read-out mode among those supported has to be used. This choice determines automatically the read-out speed and the number and location of on-chip outputs used (see also panel *ccdConfig* to find the correspondence between this index and the characteristics of the associated read-out).

   Normally in case of TCCDs this parameter must be set to 1. For a limited set of TCCD systems (only of type C, i.e. large format), the value should be better set to 2 (slow read-out mode). The list of TCCD heads for which a value 2 is recommended is available in the CCD section of the Release Notes accompanying every VLT sw release.

2. The parameter **`DET.OUT1.GAININD`** defines the amplifier gain to be used during read-out. Each read-out mode has a default gain associated. For normal observations the default value should be used and therefore **this parameter should NOT be set**. However for laboratory tests a gain

different from the default one may be wished; in this case the value of parameter **DET.OUT1.GAININD** has to be specified **after** the definition of **DET.READ.CLKIND** (see interpretation of parameters to command SETUP in [4]) in order to overwrite the default value. The allowed range for SCCDS is 0-7, the allowed values for TCCDS are 1,2,4,8.

3. The parameters **DET.WIN1.BINX** and **DET.WIN1.BINY** define the binning to be applied to the read-out.

   The same binning factor applies to all defined windows.

4. The parameter **DET.WIN<i>.ST** is of logical type (T/F) and indicates if a windowed readout is wanted. Currently the system supports a maximum of two windows (**DET.WIN1.ST** and **DET.WIN2.ST**). There are however limitations to be taken into account:

   a. For TCCDS systems, parallel windowing and binning is not supported. Therefore, whenever **DET.WIN1.BINX** or **DET.WIN1.BINY** is greater than 1 (binning), **DET.WIN<i>.ST** must be F (and whenever **DET.WIN<i>.ST** is T, **DET.WIN1.BINX** and **DET.WIN1.BINY** must be 1).

   b. For SCCDS systems, windowing is supported ONLY if the whole chip read-out is performed through one single output, i.e. for some values, varying from system to system, of **DET.READ.CLKIND**, windowing may not be supported. For the system in use, the maximum number of windows allowed for each supported read-out mode is shown in the configuration panel (see [16]).

      In case of two windows read-out, they must be either completely aligned vertically (**DET.WIN2.STRY**=**DET.WIN1.STRY** and **DET.WIN2.NY**=**DET.WIN1.NY**) or completely misaligned (**DET.WIN2.STRX** > **DET.WIN1.STRY**+**DET.WIN1.NY** or **DET.WIN1.STRY** > **DET.WIN2.STRY**+**DET.WIN2.NY**).

5. The parameters **DET.WIN<i>.STRX**, **DET.WIN<i>.STRY**, **DET.WIN<i>.NX** and **DET.WIN<i>.NY** define the location (lower left corner; first pixel in the chip has coordinate 1,1) and size of a window.

   They are ignored if **DET.WIN1.ST** is F.

6. The parameter **DET.READ.SIMIMG** is ignored if the operational mode is not **ccdSIM_LCU** (ACE simulated at LCU level). It defines the file containing the simulated image to be loaded in the LCU in place of the standard synthetic image generation. The path is supposed to be relative to the directory where images are normally stored (see OLDB attribute *images.imageDirectory*, section 2.16. Useful for testing centroiding algorithms on real images.

2.14.4 Processing

The following parameters are described in the order they are used as shown in the data flow diagram on page 20.

1. The parameter **DET.WIN<i>.BIAS** is of logical type (T/F) and indicates if a bias subtraction has to be applied to the data for the specified window. It presumes that an image of type Bias is stored in the pool of images on the LCU. See section 2.6 for more information.
2. The parameter **DET.WIN<i>.FLATF** is of logical type (T/F) and indicates if a flat field division has to be applied to the data for the specified window. It presumes that an image of type Flat is stored in the pool of images on the LCU. See section 2.6 for more information.
3. The parameter **DET.WIN<i>.MINMAX** is of logical type (T/F) and indicates if statistics (e.g. minimum/maximum, average) has to be computed on the data for the specified window.

4. The parameters **DET.WIN<i>.IPLLX/Y**, **DET.WIN<i>.IPURX/Y** define the offsets with respect to the defined window (see **DET.WIN<i>.STRX** etc.), identifying the sub-window on which the image processing, such as centroiding, has to be performed. IPLLX and IPLLY define respectively the horizontal and vertical offset with from the lower left corner, IPURX and IPURY the same with respect to the upper right corner. As for TCCDS the first column as result of the read-out is always brighter than the rest, it is recommended to exclude it from the image processing; in this case it should be IPLLX = 1, IPLLY = IPURX = IPURY = 0.
5. The parameter **DET.WIN<i>.CENTROID** indicates the kind of centroiding algorithm wanted for the specified window (see also section 2.6.2). The following values are supported (see *ccd.h*):
   a. **ccdCEN_NONE_STR**: no centroid is performed
   b. **ccdCEN_THRESHOLD_STR***:* threshold based algorithm.
      the background level is defined by the value of parameter **DET.WIN<i>.BACKGND**
      the threshold level is defined by the value of parameter **DET.WIN<i>.THRMIN**

   This parameter must always preceed **DET.WIN<i>.BACKGND** and **DET.WIN<i>.THRMIN** ; if they are not defined by the user, they are defaulted to resp. **ccdBCKGND_FLUX_WINDOW_SELF** and **ccdTHRMIN_3SIGMA_WINDOW_SELF**. Thus attention must be observed in the setting precedence.
6. The parameter **DET.WIN<i>.BACKGND** defines the background level to be used in the centroiding algorithm when parameter **DET.WIN<i>.CENTROID** has value **ccdCEN_THRESHOLD_STR**. It can have any positive value or the following negative values with special meaning (see *ccd.h*):
   a. **ccdBCKGND_FLUX_WINDOW_SELF** (-1): background level is set to the average value over the related window ***images:process:window_<i>.ipBGnd***.
   b. **ccdBCKGND_FLUX_WINDOW_PREV** (-11): background level is set to the average value over window1 (attribute ***images:process:window_1.ipBGnd***). Meaningful only if set for **DET.WIN2.BACKGND** and two windows are defined)
7. The parameter **DET.WIN<i>.THRMIN** defines the threshold level to be used in the centroiding algorithm when parameter **DET.WIN<i>.CENTROID** has value **ccdCEN_THRESHOLD_STR**. It can have any positive value or the following negative values with special meaning (see *ccd.h*):
   a. -10 < N < 0: threshold level is set to (-N) *σ ;
      where σ = standard deviation over the background level of the related window (attribute ***images:process:window_<i>.ipBGnd_SD***)
      The computation using 3 * σ is coded as **ccdTHRMIN_3SIGMA_WINDOW_SELF**
      The computation using 5 * σ is coded as **ccdTHRMIN_5SIGMA_WINDOW_SELF**
   b. -20 < N < -10: threshold level is set to (-N-10) *σ;
      where σ = standard deviation over the pixel intensity on window1 (attribute ***images:process:window_1.ipStdDev***). Meaningful only if set for **DET.WIN2.THRMIN** and two windows are defined.
      The computation using 3 * σ is coded as **ccdTHRMIN_3SIGMA_WINDOW_PREV**
      The computation using 5 * σ is coded as **ccdTHRMIN_5SIGMA_WINDOW_PREV**
8. The parameters **DET.WIN<i>.REFX** and **DET.WIN<i>.REFY** define the reference point for the computation of the error vector in the centroiding algorithm. These values may be modified during an infinite loop of exposures.

   They are ignored if **DET.WIN<i>.CENTROID** is **ccdCEN_NONE_STR**.

9. The parameter **DET.WIN<i>.IPFUNC** indicates the name of the user function the CCD LCU sw has to call as last step of the image processing for the specified window. If set to **ccdIP_NO_USER_FUN**, no user function is called. See also section 2.6.
10. The parameter **DET.WIN<i>.IPBUFF** indicates the name of a global variable whose address has to be passed to the user function specified by **DET.WIN<i>.IPFUNC**. If set to **ccdIP_NO_USER_BUF**, a NULL pointer is passed to the user function. See also section 2.6.

    It is ignored if **DET.WIN<i>.IPFUNC** is set to **ccdIP_NO_USER_FUN**.

2.14.5 Transfer

1. The parameters **DET.DISPLAY** specifies the frame Id for the VLT Real-Time display facility (see also [26]). The value **ccdNO_DISPLAY** (see *ccd.h*) indicates that images do not need to be displayed.
2. The parameter **DET.FRAM.FITSMTD** indicates if and how an image has to be saved on disk in FITS format. Currently only **ccdDISK_NONE** and **ccdDISK_UNCOMPRESS** are implemented (see *ccd.h*)
3. The parameter **DET.FRAM.FITSUNC** specifies the name of the file. It must have suffix ***.fits.*** The maximum allowed length is **ccdMAXLENFILE** (see *ccd.h).*

   This parameter is ignored if **DET.FRAM.FITSMTD** is set to **ccdDISK_NONE**.
4. The parameter **DET.FRAM.SAMPLE** indicates how frequently images have to be transferred to the Workstation during a loop of repeated exposures. A value of 1 means that all images have to be transferred. A value *N* higher than 1 means that only one image out of *N* has to be transferred. Useful to reduce network load if no high refresh rate is needed on Workstation.

   It is ignored for single not repeated exposures (**DET.EXP.NREP** set to 1) or repeated exposures where images are not needed on Workstation (**DET.FRAM.FITSMTD** set to **ccdDISK_NONE** and **DET.FRAM.DISPLAY** set to **ccdNO_DISPLAY**).
5. The parameter **DET.FRAM.XFERSYN** is of logical type (T/F) and indicates if the image transfer shall occur synchronously. For single exposures, this flag has little effect: the last exposure in a loop is always transfered synchronously; therefore this parameter is ignored for single exposures. If set to T, the system waits for the image transfer to WS to be completed before running the next exposure in a loop, otherwise it starts the next exposure as soon the readout and image processing are ready.

   It is recommended to set this keyword always to T, except special applications requiring high performances at LCU level and using the image transfer only for real-time display of the image.

2.14.6 Multiple Exposure

A multiple exposure is a sequence of integrations separated by a effective shift of the object on the detector. Currently, this feature is only available on Technical CCD. The shift is obtained by moving a given number of lines from the illuminated part to the cache of the CCD.
The exposure type **DET.EXP.TYPE** must be set to **ccdEXP_MULTI_STR**. The following parameters are ignored if **DET.EXP.TYPE** is not **ccdEXP_MULTI_STR**.

1. The parameter **DET.WIN1.NDIT** specifies the number of sub-integrations to be performed. The maximum value allowed by the CCD sw is **ccdMAXINTEGR** (=10).

2. The parameter **DET.WIN1.ASUIT1** is of logical type (T/F). If set to T, all sub-integrations have the same time, as specified in **DET.WIN1.UIT1**, if F, each sub-integration time is defined as in **DET.WIN1.UIT<i>**.

   The parameter **DET.WIN1.ASUIT** is obsolete. It has been kept for backwards compatibility.
3. The parameter **DET.READ.SHIFTYP** determines which type of on-chip rows shift has to be applied between sub-integrations.

   Possible values are :
   a. **ccdLINE_SHIFT_ALT_STR**. Shift alternatively up and down the amount of rows specified in **DET.READ.SHIFT1** (va-et-vient mode, see [3]).
   b. **ccdLINE_SHIFT_IDEM_STR**. Shift always in the same direction the amount of rows specified in **DET.READ.SHIFT1**.
   c. **ccdLINE_SHIFT_LIST_STR**. Shift the amount of rows specified in **DET.READ.SHIFT<i>**.
4. The parameter **DET.READ.SHIFT<i>** specifies the amount of rows to be shifted after a sub-integration. The sign indicates the direction: if positive, rows are shifted downwards, if negative upwards. The latter is possible only when there are at least two on-chip outputs, one at the bottom and one at the top. In case of TCCDS (one output only at the bottom) only positive values are valid.

Applications handling CCD setup parameters are requested to use the macros defined in *ccd.h* (e.g. ccdEXP_TYPE).

2.15 Image data

Image data are provided by the CCD DCS software in two ways:

- Raw-data for Real-time display.
- FITS files

The orientation of the image is independent from the clock pattern used (see request from Fors VLT-LET-VIC-13110-0030): independently from the outputs involved in the readout, the image is displayed and/or stored on file always with the same orientation. This strategy has of course its price, namely that for some clock pattern the whole chip must be read-out before displaying and storing on file the very first pixel; in other words, the overall performance of the system depends heavily on the clock pattern used, as in some cases image readout and image storage must be performed sequentially, in other cases they can be done in parallel (this point does not concern technical CCDs, because only one clock pattern is provided and performances are optimized for that one).
The CCD sw does not *know* the orientation of the camera with respect to the sky. It *thinks* exclusively in terms of physical coordinates on the chip.
Location (1,1) corresponds to the first pixel value stored in a FITS file.

2.15.1 Raw-data for real-time display

The CCD software provides **raw data** for the VLT sw **real-time display** utility whenever the setup parameter **DET.DISPLAY** is set to a value higher than **ccdNO_DISPLAY**. The setting of this parameter determines the frame where the image will be displayed (currently the utility *rtd d*efines frame Id 0 for the big frame and 4 for the rapid frame). The CCD software supports up to 10 different frames Id during the same session (10 different images displayed on different frames).
The mechanism to deliver raw data is the same as defined in [26].
Raw-data are written in shared memory as they come out from the ACE, namely with full

resolution (16-bits unsigned integer). No reduction (e.g. to 8-bits) is done by the CCD software.

In addition to the display of the raw-data, the CCD sw supports also the **display of World Coordinates** through *rtd*. One point in the CCD branch of the OLDB is dedicated to this feature. It contains two different kind of attributes:

1. Attributes having static values, i.e. they do not change from one exposure to another. They can be set once and forever through the script *ccdDcsWcs.sh* (see section 3.3).
2. Attributes having values, which may change from one exposure to another. They are part of the public part of the CCD database (see also section 2.16). It is responsibility of applications using CCDs to enter in these attributes the right values at the right time.

2.15.2 FITS files

Images, as result of exposures, are written on WS disk in FITS format (see [4]), whenever the setup parameter **`DET.FRAM.FITSMTD`** is other than **`ccdDISK_NONE`** (see *ccdDISKSAVE* in *ccd.h*).

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

Independently from the read-out mode used, the complete physical image is stored in one single FITS file. For multiple windows (currently a maximum of two not overlapping windows is supported), they are also stored in one FITS file with IMAGE extension (see [5]).

The option to get in a second file image data compressed (e.g. to be transferred to a remote station) is not implemented yet (**`ccdDISK_COMPRESS`** is treated as **`ccdDISK_NONE`**). Only uncompressed data files are produced.

Apart from the image raw data, the CCD software is also responsible to provide 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 directly by the CCD sw, such that, whatever happens to higher level software (e.g. OS), a readable FITS file is saved, although with basic information only. The number of 80-characters lines reserved by the CCD sw at the beginning of the file is defined by the macro **`ccdDCSHEADLINES`** in *ccd.h*. See also the example given in section 3.8.1
- **Hierarchical keywords**. They are not strictly needed to interpret the pixel values and normally do not appear at the beginning of the FITS header. Since the CCD sw cannot know at which position in the FITS header they must be written, they are written into a separate file with the same name as the image file and extension defined by the macro **`ccdFITSHIERSUFF`** in *ccd.h*; it is responsibility of the higher level software (e.g. OS for an instrument) to read this file and merge the information contained with the other information collected from the various equipment (e.g. instrument, telescope). See also the example given in section 3.8.2

CCD DCS first writes in these files, then sets the exposure status to **`ccdEXP_COMPLETED`**

The responsibility to build image FITS files is shared among DCS and OS (or TCS in case of technical CCDs). The following rules are set by the CCD DCS software:

1. OS sets in the CCD database, before the CCD DCS software is started, the name of the directory where image files must be saved (see 2.16 and *ccdConfig* in [16] for an alternative interactive way).
2. OS creates the image file(s) and reserves enough space for the complete FITS header.

3. OS passes to CCD DCS the name of the file as setup parameter before an exposure is started (see **ccdCMD_SETUP** and **ccdFILE_UNC** in *ccd.h*).
4. When read-out is started, CCD DCS looks for the file specified by OS (see point 3.) and opens it for writing.

   Note 1: Normally the file should be there and have the size for the complete FITS header (see point 2.). If it is not there, CCD DCS creates it. If it is there, but not accessible (e.g. write protected), CCD DCS tries to create a new one with the same name, followed by a sequential integer index, starting from 1 (see [6]), incrementing the index until it succeeds.

   Note 2: In case the number of FITS file to be produced is more than one (e.g. setup parameter exposure repetition factor, see **ccdREPEAT_DEF** in *ccd.h*, is set to 3), CCD DCS assumes that all files will have the same name, followed by a sequential integer index, starting from 1. Example:

   ccdREPEAT_DEF is set to 3.

   ccdFILE_UNC is set to *myImage.fits*

   CCD DCS will look for files *myImage.fits* (first exposure), *myImage.0.fits* (second exposure) and *myImage.1.fits* (third exposure)

   See also [5] for file naming rules.

   If the exposure repetition factor**DET.EXP.NREP** is set to **ccdREPEAT_FOREVER**, the file is overwritten and contains the last image.
5. CCD DCS writes the basic standard keywords at the beginning of the file and then moves up to the end of the file (space reserved by OS for the rest of the header).
6. CCD DCS writes the complete image.
7. If multiple window read-out has been performed, CCD DCS writes the first extension header after the first frame data, reserves as much space as in the main header and then writes the second frame data.
8. CCD DCS writes the hierarchical keywords in a separate file, as described above.
9. CCD DCS sets the exposure status to **ccdEXP_COMPLETED**.
10. OS retrieves from the CCD database (see section 2.16) the name of the file containing the image. Normally it is the same as the setup values passed by OS, but CCD DCS may have been obliged to change it (see point 4.before).
11. OS collects hierarchical information from all sub-systems, moves **ccdDCSHEADLINES** from the beginning of the file and writes them. If multiple window read-out has been performed, the same operation has to be done for all extension headers.

    OS knows how much space it has reserved for the header and is therefore the only responsible to avoid that the image data section is overwritten with header information.

Note 1: tools and functions for handling image FITS files are provided by the INS common sw (*slx*, see [24], and *oslx*, see [25]).
Note 2: in order to get the proper absolute time information in the FITS header (in particular the keyword **MJD-OBS**) and logs, the CCD LCU has to be properly configured: the *tim* module has to be loaded if the LCU is connected to the Time Reference System, otherwise the *ntp* module has to be loaded (see also section 4.6.7).

2.16 Public part of the CCD database

Some attributes of the CCD branch of the on-line database are made public for direct read/write operations from external software. Attributes which are accessed indirectly from external software through commands, functions or panels provided by the CCD software (e.g. all setup attributes set with the command SETUP, configuration attributes read with the function *ccdGetConf* and set with the panel *ccdConfig*) are not considered public (no direct call to *dbRead/dbWrite* CCS functions).

When accessing CCD database attributes with direct CCS db calls, **applications are requested to use the macros defined in *ccdDbPublic.h***: in this way, any change in name or location of the attribute would require just a new compilation.

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

2.16.1 Public part of the CCD Workstation database

The attributes marked with **(*)** require that the Scan System is working between CCD LCU and WS.

1. **Read/Write attributes**
    a. ***opMode*** (dbINT32) Camera operational mode (**`ccdDB_CON_OPMODE`**)
    b. ***images.imageDirectory*** (dbBYTES128) Directory where images are stored (**`ccdDB_CON_IMGPATH`**).
    c. ***wcs.ra*** (dbDOUBLE) Center right ascension in degrees for World Coordinates display (**`ccdDB_WCS_RA`**)
    d. ***wcs.dec*** (dbDOUBLE) Center declination in degrees for World Coordinates display (**`ccdDB_WCS_DEC`**)
2. **Read only attributes**
    a. ***exposures:exposure_1.expStatus (*)*** (dbINT32) Current exposure status (**`ccdDB_STA_EXPOSURE1`**).
    b. ***exposures:exposure_1.expId*** (dbINT32) ID of exposure. (**`ccdDB_STA_EXPID1`**)
    c. ***exposures:exposure_1:transfer.fileNameUnComp*** (dbBYTES32) Name of FITS file where uncompressed image data are written. *(**`ccdDB_EXP_FILEUNC1`**).*
    d. ***opState*** **(*)** (dbINT32) Camera operational state. (**`ccdDB_STA_SYSTEM`**)
3. **Read only attributes for GUI panels.**

    As already mentioned in section 2.12, the CCD software provides applications with CCD GUI classes to be incorporated in their own panels. It is **strongly recommended** to use these classes in application panels. For all cases where this is not possible, the following attributes are made public for usage within GUI panels;.
    a. ***images:transfer.percent*** (dbINT32) Percentage of image transferred to WS (**`ccdDB_IT_STA_PERC`**)
    b. ***images:transfer.last*** (dbINT32) Last line transferred to WS (**`ccdDB_IT_STA_LINE`**)
    c. ***shutter.status (*)*** (dbINT32) Shutter status (**`ccdDB_IT_STA_SHTSTATUS`**)
    d. ***exposures:exposure_1.timeRem*** (dbDOUBLE) Remaining integration time. (**`ccdDB_STA_TIMEREM1`**)
    e. ***failureWs*** (dbINT32) Stack Id for asynchronous errors occurring on WS (e.g. the disk is full while transferring the image result of an exposure) (**`ccdDB_STA_FAILURE_WS`**).

f. ***failureLcu (*)*** (dbINT32) Stack Id for asynchronous errors occurring on LCU (e.g. the image read-out fails) (**ccdDB_STA_FAILURE_LCU**).

g. ***temperature.message (*)*** (dbBYTES256) Message showing a summary of the status of temperature sensors (**ccdDB_STA_TMPMESSAGE**)

h. ***telemetry.message (*)*** (dbBYTES256) Message showing a summary of the status of telemetry sensors (**ccdDB_STA_TELMESSAGE**)

2.16.2 Public part of the CCD LCU database

1. **Read/Write attributes**

   None

2. **Read only attributes**

   a. ***exposures:exposure_1.expStatus*** (dbINT32) Exposure status (**ccdDB_STA_EXPOSURE1**).

   b. ***exposures:exposure_1.expId*** (dbINT32) ID of exposure. (**ccdDB_STA_EXPID1**)

   c. ***images:process:window_1.ipXCen*** (dbDOUBLE) Error vector (x-component) from centroiding calculation. Same for *window_2* (**ccdDB_IP<index>_XCEN**).

   d. ***images:process:window_1.ipYCen*** (dbDOUBLE) Error vector (y-component) from centroiding calculation. Same for *window_2* (**ccdDB_IP<index>_YCEN**).

   e. ***images:process:window_1.ipCenVal*** (dbDOUBLE) Value of the pixel closest to the computed centroid position. Same for *window_2* (**ccdDB_IP<index>_CENVAL**).

   f. ***images:process:window_1.ipXMin*** (dbINT32) Horizontal position of pixel with minimum intensity. Same for *window_2* (**ccdDB_IP<index>_XMIN**).

   g. ***images:process:window_1.ipYMin*** (dbINT32) Vertical position of pixel with minimum intensity. Same for *window_2* (**ccdDB_IP<index>_YMIN**).

   h. ***images:process:window_1.ipMinVal*** (dbDOUBLE) Minimum intensity. Same for *window_2* (**ccdDB_IP<index>_MINVAL**).

   i. ***images:process:window_1.ipXMax*** (dbINT32) Horizontal position of pixel with maximum intensity. Same for *window_2* (**ccdDB_IP<index>_XMAX**).

   j. ***images:process:window_1.ipYMax*** (dbINT32) Vertical position of pixel with maximum intensity. Same for *window_2* (**ccdDB_IP<index>_YMAX**).

   k. ***images:process:window_1.ipMaxVal*** (dbDOUBLE) Maximum intensity. Same for *window_2* (**ccdDB_IP<index>_MAXVAL**).

   l. ***images:process:window_1.ipFlux*** (dbDOUBLE) Average pixel intensity value. Same for *window_2* (**ccdDB_IP<index>_FLUX**).

   m. ***images:process:window_1.ipStdDev*** (dbDOUBLE) Pixels intensity standard deviation. Same for *window_2* (**ccdDB_IP<index>_STDDEV**).

   n. ***images:process:window_1.ipNumPix*** (dbINT32) Total number of pixels with intensity higher than 0 after background subtraction and thresholding. Same for *window_2* (**ccdDB_IP<index>_NUMPIX**).

   o. ***images:process:window_1.ipSNR*** (dbDOUBLE) Estimated Signal-to-Noise Ratio. Same for *window_2* (**ccdDB_IP<index>_SNR**).

   p. ***images:process:window_1.ipBGnd*** (dbDOUBLE) Background intensity estimation. Same for *window_2* (**ccdDB_IP<index>_BGND**).

q. ***images:process:window_1.ipBGnd_SD*** (dbDOUBLE) Std Dev. on background intensity estimation. Same for ***window_2*** (**`ccdDB_IP<index>_BGND_SD`**).

r. ***images:process:window_1.ipFWHM_X*** (dbDOUBLE) Object FWHM estimation along X-axis. Same for ***window_2*** (**`ccdDB_IP<index>_FWHM_X`**).

s. ***images:process:window_1.ipFWHM_Y*** (dbDOUBLE) Object FWHM estimation along Y-axis. Same for ***window_2*** (**`ccdDB_IP<index>_FWHM_Y`**).

2.17 Configuration and operational logs

The CCD sw logs the main operations it performs, using the VLT CCS Logging System, in FITS format. Examples of logs produced by the CCD sw are given in section 3.9
According to the syntax specified in [5], the last part of the log message must contain a source mask, in order to identify the sub-system the log is coming from. As CCD cameras are used in many different sub-systems, this mask cannot be set by the CCD sw itself, but must be defined and set by the sub-system using it. A script, to be executed once when configuring the CCD sw (see sections 4.6.6 and 4.6.10) and called *ccdDcsSetLogMask.sh* (see also section 3.3) is dedicated to this purpose. Examples:

1. For SCCD:
```
$ ccdDcsSetLogMask.sh $CCDNAME $RTAPENV "UVER"
```
2. For TCCD:
```
$ ccdDcsSetLogMask.sh $CCDNAME $RTAPENV "UT1AGA"
```

2.18 Examples for Technical CCD Camera

2.18.1 Usage of CCD technical cameras

The following example is a Sequencer script available on tape (file <TAPE>/CCD/ccdcon/ws/test/ccdconTestTec.tcl).
In the following we consider the case of cameras used for auto-guiding (AG)

```
#--------------------------------------------------------------------------
# Basic initialization of this application
# (register to CCS, fill in gvar array with definitions needed etc.)
ccdconTestInit Tec

#--------------------------------------------------------------------------
# Set operational mode to no simulation
# (write in WS OLDB and run ccdDcsDbSave.sh)
ccdSetOpMode $gvar(no_sim)

puts "--------------------------------------------------------------------------"
puts "                        Cold startup"
puts "--------------------------------------------------------------------------"
# (execute ccdDcsStart.sh script)
ccdCstart

puts "--------------------------------------------------------------------------"
puts "               Bring the CCD system to ONLINE state"
puts "--------------------------------------------------------------------------"
# STANDBY (optional)
# ONLINE
ccdOnLine
```

```
puts "-------------------------------------------------------------------------"
puts "      Load the complete setup file defining all application defaults"
puts "-------------------------------------------------------------------------"
# Important, otherwise CCD sw defaults would be used
ccdSendCmd SETUP "-expoId $gvar(ccdEXP_NEXT) -file $gvar(setupFileDefaults)"
# Make sure that chip is clean
startWipe
sleep 3
stopWipe
set defTime 0.1; # default integration time for this demo

# Here telescope, guide probe etc. should be positioned
puts ""
puts "-------------------------------------------------------------------------"
puts "                          Star acquisition"
puts "          setup: Full frame, display and save in file"
puts "-------------------------------------------------------------------------"
askUserAck
puts ""
# define SETUP for field acquisition
ccdSendCmd SETUP "-expoId $gvar(ccdEXP_NEXT) -function \
                 $gvar(SETUP_EXP_TYPE) $gvar(ccdEXP_NORMAL_STR) \
                 $gvar(SETUP_TIME_DEF) $defTime \
                 $gvar(SETUP_READ_WINBX_1) 1 \
                 $gvar(SETUP_READ_WINBY_1) 1 \
                 $gvar(SETUP_READ_WINEN_1) F \
                 $gvar(SETUP_READ_WINEN_2) F \
                 $gvar(SETUP_EXP_REPEAT) 1 \
                 $gvar(SETUP_DISPLAY) $gvar(ccdDISPLAY_MAINFRAME) \
                 $gvar(SETUP_TRAN_DISK) $gvar(ccdDISK_UNCOMPRESS) \
                 $gvar(SETUP_TRAN_UCOM) xccdconTestTecFull.fits \
                 $gvar(SETUP_TRAN_RATE) 1"

# START exposure
set expoId [ccdSendCmd START]
# WAIT for exposure completion
ccdSendCmd WAIT "-expoId $expoId -waitMode Single"
send_to_rtd autocut
puts "Look for brightest star in field"
ccdObjBright "R,3"
# Define brightest star as reference point and window around it
ccdDefineWindowBright

puts ""
puts "-------------------------------------------------------------------------"
puts "                          Guiding loop"
puts "     setup: window 20x20, centroiding, display and no save in file"
puts "-------------------------------------------------------------------------"
askUserAck
puts ""
# Define SETUP for rapid guide loop
ccdSendCmd SETUP "-expoId $gvar(ccdEXP_NEXT) -function \
                 $gvar(SETUP_EXP_TYPE) $gvar(ccdEXP_NORMAL_STR) \
                 $gvar(SETUP_TIME_DEF) $defTime \
                 $gvar(SETUP_EXP_REPEAT) $gvar(ccdREPEAT_FOREVER) \
                 $gvar(SETUP_READ_WINBX_1) 1 \
                 $gvar(SETUP_READ_WINBY_1) 1 \
```

```
                 $gvar(SETUP_READ_WINEN_1) T \
                 $gvar(SETUP_READ_WINSX_1) $gvar(xStart) \
                 $gvar(SETUP_READ_WINSY_1) $gvar(yStart) \
                 $gvar(SETUP_READ_WINNX_1) $gvar(xSize) \
                 $gvar(SETUP_READ_WINNY_1) $gvar(ySize) \
                 $gvar(SETUP_READ_WINEN_2) F \
                 $gvar(SETUP_PROC_EXTR_1) T \
                 $gvar(SETUP_PROC_CENT_1) $gvar(ccdCENTROID_STANDARD_STR) \
                 $gvar(SETUP_PROC_UFCT_1) $gvar(ipUserFunction) \
                 $gvar(SETUP_PROC_UBUF_1) $gvar(ipUserBuffer) \
                 $gvar(SETUP_PROC_XREF_1) $gvar(objBrgX) \
                 $gvar(SETUP_PROC_YREF_1) $gvar(objBrgY) \
                 $gvar(SETUP_DISPLAY) $gvar(ccdDISPLAY_MAINFRAME) \
                 $gvar(SETUP_TRAN_DISK) $gvar(ccdDISK_NONE)"

# START loop
set expoId [ccdSendCmd STARTAG]
send_to_rtd autocut
# Ask user to acknowledge to stop the loop
askUserAck
# STOP loop
ccdSendCmd STOP
# WAIT for loop completion
ccdSendCmd WAIT "-expoId $expoId -waitMode Global"


puts "-------------------------------------------------------------------------"
puts "                             Shutdown camera"
puts "-------------------------------------------------------------------------"
# (execute ccdDcsStop.sh script)
ccdShutdown

#-------------------------------------------------------------------
# Terminate application
ccdconTestExit
```

2.18.2 Full Frame setup

The following setup is typically used by the Active Optics.

```
DET.EXP.TYPE      "Normal  ";   # Type of exposure as known to the CCD sw
DET.EXP.NREP                1; # Single exposure
DET.READ.CLKIND             1; # Index of clock pattern used
DET.OUT1.GAININD            8; # Gain index for output
DET.WIN1.UIT1          30.000; # user defined subintegration time
DET.WIN1.BINX               1; # Binning factor in X
DET.WIN1.BINY               1; # Binning factor along Y
DET.DISPLAY                -1; # no real-time image display
DET.FRAM.FITSMTD            0; # no data save on disk
DET.FRAM.SAMPLE             1; # Image sampling on workstation
DET.FRAM.XFERSYN            T; # Image transfer synchronous
DET.FRAM.TYPE     "Normal  ";   # Type of frame
DET.WIN1.ST               "F"; # Full Frame
DET.WIN1.BIAS             "F"; # no bias frame subtraction performed
DET.WIN1.FLATF            "F"; # no flat field division performed
DET.WIN1.MINMAX           "F"; # no image extrema search performed
```

```
DET.WIN1.CENTROID "none    ";  # no centroiding calculation
DET.WIN1.IPFUNC   "None    ";  # no User Function
DET.WIN2.ST               "F"; # Full Frame
DET.WIN2.BIAS             "F"; # no bias frame subtraction performed
DET.WIN2.FLATF            "F"; # no flat field division performed
DET.WIN2.MINMAX           "F"; # no image extrema search performed
DET.WIN2.CENTROID "none    ";  # no centroiding calculation
DET.WIN2.IPFUNC   "None    ";  # no User Function
```

2.18.3 1 Window setup

The following setup is typically used by the TCS Autoguider/Field Stabilization.

```
DET.EXP.TYPE     "Normal  ";   # Exposure type
DET.WIN1.UIT1           0.010; # user defined subintegration time
DET.EXP.NREP                0; # infinite loop
DET.EXP.WIPETIM             0; # wipe before each exposure
DET.READ.CLKIND             1; # Index of clock pattern used
DET.OUT1.GAININD            8; # Gain index for output
DET.FRAM.FITSMTD            0; # No Data storage
DET.DISPLAY                 0; # real-time image display frame ID
DET.WIN1.BINX               1; # Binning factor along X
DET.WIN1.BINY               1; # Binning factor along Y
DET.FRAM.SAMPLE             1; # Image sampling on workstation
DET.FRAM.TYPE    "Normal  ";   # Type of frame
DET.FRAM.XFERSYN          "F"; # Asynchronous transfer
DET.EXP.TIMEREP      1.000000; # Time between two repeated exposures
DET.WIN1.ST               "T"; # One Window
DET.WIN1.STRX             209; # Lower left pixel in X
DET.WIN1.NX                25; # # of pixels along X
DET.WIN1.STRY             134; # Lower left pixel in Y
DET.WIN1.NY                25; # # of pixels along Y
DET.WIN1.CENTROID "threshold"; # type of centroiding calculation
DET.WIN1.BACKGND    -1.000000; # sky background
DET.WIN1.THRMIN     -3.000000; # lower threshold
DET.WIN1.REFX         221.773; # X position of reference
DET.WIN1.REFY         146.590; # Y position of reference
DET.WIN1.MINMAX           "F"; # image extrema search performed
DET.WIN1.IPFUNC  "None    ";   # user defined function name
DET.WIN1.IPBUFF  "None    ";   # buffer variable in user function
DET.WIN1.BIAS             "F"; # bias frame subtraction performed
DET.WIN1.FLATF            "F"; # flat field division performed
DET.WIN1.IPLLX              1; # X offset from bottom left for proc. sub-window
DET.WIN1.IPURX              1; # X offset from top right for proc. sub-window
DET.WIN2.ST               "F"; # Window 1 only
```

2.18.4 2 Window setup

The following setup is typically used by the Astronomical Site Monitor in Seeing mode (DIMM).

```
DET.EXP.TYPE      "Normal  ";   # Exposure type
DET.EXP.NREP               800; # number of repeated exposures
DET.READ.CLKIND              1; # Index of clock pattern used
DET.OUT1.GAININD             8; # Gain index for output
DET.FRAM.FITSMTD             0; # Data storage method
DET.DISPLAY                  0; # real-time image display frame ID
DET.WIN1.BINX                1; # Binning factor along X
DET.WIN1.BINY                1; # Binning factor along Y
DET.FRAM.SAMPLE             20; # Image sampling on workstation
DET.FRAM.TYPE     "Normal  ";   # Type of frame
DET.EXP.TIMEREP           0.00; # Time between two repeated exposures
DET.FRAM.XFERSYN           "F"; # asynchronous image transfer
DET.EXP.WIPETIM              0; # wipe before starting exposure
DET.WIN1.ST                "T"; # If T, window enabled
DET.WIN1.STRX              158; # Lower left pixel in X
DET.WIN1.NX                 31; # # of pixels along X
DET.WIN1.STRY              125; # Lower left pixel in Y
DET.WIN1.NY                 30; # # of pixels along Y
DET.WIN1.CENTROID "threshold"; # type of centroiding calculation
DET.WIN1.REFX           184.00; # X position of reference
DET.WIN1.REFY           144.00; # Y position of reference
DET.WIN1.MINMAX            "T"; # image extrema search performed
DET.WIN1.IPFUNC   "asmdimUSR";  # user defined function name
DET.WIN1.IPBUFF   "None    ";   # buffer variable in user function
DET.WIN1.BIAS              "F"; # bias frame subtraction performed
DET.WIN1.FLATF             "F"; # flat field division performed
DET.WIN1.BACKGND     -1.000000; # sky background
DET.WIN1.THRMIN      -3.000000; # lower threshold
DET.WIN1.IPLLX               1; # X offset from bottom left
DET.WIN1.IPLLY               0; # Y offset from bottom left
DET.WIN1.IPURX               0; # X offset from top right
DET.WIN1.IPURY               0; # Y offset from top right
DET.WIN2.ST                "T"; # If T, window enabled
DET.WIN2.STRX              200; # Lower left pixel in X
DET.WIN2.NX                 31; # # of pixels along X
DET.WIN2.STRY              125; # Lower left pixel in Y
DET.WIN2.NY                 30; # # of pixels along Y
DET.WIN2.CENTROID "threshold"; # type of centroiding calculation
DET.WIN2.REFX           212.00; # X position of reference
DET.WIN2.REFY           144.00; # Y position of reference
DET.WIN2.MINMAX            "T"; # image extrema search performed
DET.WIN2.IPFUNC   "asmdimUSR";  # user defined function name
DET.WIN2.IPBUFF   "None    ";   # buffer variable in user function
DET.WIN2.BIAS              "F"; # bias frame subtraction performed
DET.WIN2.FLATF             "F"; # flat field division performed
DET.WIN2.BACKGND     -1.000000; # sky background
DET.WIN2.THRMIN      -3.000000; # lower threshold
DET.WIN2.IPLLX               1; # X offset from bottom left
DET.WIN2.IPLLY               0; # Y offset from bottom left
DET.WIN2.IPURX               0; # X offset from top right
DET.WIN2.IPURY               0; # Y offset from top right
DET.WIN1.NDIT                1; # # of subintegrations
DET.WIN1.ASUIT1            "T"; # All UITi as UIT1 else as defined in UITi
DET.WIN1.UIT1            0.010; # user defined subintegration time

# --- oOo ---
```

2.18.5 Multiple exposure setup

The following setup is typically used by the Astronomical Site Monitor in WaveFront Coherence mode.

```
DET.EXP.TYPE     "Multiple";   # Exposure type
DET.EXP.NREP              400; # number of repeated exposures
DET.READ.CLKIND             1; # Index of clock pattern used
DET.OUT1.GAININD            8; # Gain index for output
DET.FRAM.FITSMTD            0; # Data storage method
DET.DISPLAY                 0; # real-time image display frame ID
DET.WIN1.BINX               1; # Binning factor along X
DET.WIN1.BINY               1; # Binning factor along Y
DET.FRAM.SAMPLE            10; # Image sampling on workstation
DET.FRAM.TYPE    "Normal  ";   # Type of frame
DET.EXP.TIMEREP          0.00; # Time between two repeated exposures
DET.FRAM.XFERSYN          "F"; # If T, image transfer occurs synchronously
DET.EXP.WIPETIM             0; # wipe before starting exposure in a loop
DET.WIN1.ST               "T"; # If T, window enabled
DET.WIN1.STRX              94; # Lower left pixel in X
DET.WIN1.NX                32; # # of pixels along X
DET.WIN1.STRY               1; # Lower left pixel in Y
DET.WIN1.NY               290; # # of pixels along Y
DET.WIN1.CENTROID "none    ";  # type of centroiding calculation
DET.WIN1.REFX          128.00; # X position of reference
DET.WIN1.REFY          207.00; # Y position of reference
DET.WIN1.MINMAX           "F"; # image extrema search performed
DET.WIN1.IPFUNC  "ccdipWCE";   # user defined function name
DET.WIN1.IPBUFF  "None    ";   # buffer variable in user function
DET.WIN1.BIAS             "F"; # bias frame subtraction performed
DET.WIN1.FLATF            "F"; # flat field division performed
DET.WIN1.BACKGND    -1.000000; # sky background
DET.WIN1.THRMIN     -3.000000; # lower threshold
DET.WIN1.IPLLX              1; # X offset from bottom left for proc. sub-window
DET.WIN1.IPLLY              0; # Y offset from bottom left for proc. sub-window
DET.WIN1.IPURX              0; # X offset from top right for proc. sub-window
DET.WIN1.IPURY              0; # Y offset from top right for proc. sub-window
DET.WIN2.ST               "T"; # If T, window enabled
DET.WIN2.STRX             132; # Lower left pixel in X
DET.WIN2.NX                32; # # of pixels along X
DET.WIN2.STRY               1; # Lower left pixel in Y
DET.WIN2.NY               290; # # of pixels along Y
DET.WIN2.CENTROID "none    ";  # type of centroiding calculation
DET.WIN2.REFX          128.00; # X position of reference
DET.WIN2.REFY          207.00; # Y position of reference
DET.WIN2.MINMAX           "F"; # image extrema search performed
DET.WIN2.IPFUNC  "ccdipWCE";   # user defined function name
DET.WIN2.IPBUFF  "None    ";   # buffer variable in user function
DET.WIN2.BIAS             "F"; # bias frame subtraction performed
DET.WIN2.FLATF            "F"; # flat field division performed
DET.WIN2.BACKGND    -1.000000; # sky background
DET.WIN2.THRMIN     -3.000000; # lower threshold
```

```
DET.WIN2.IPLLX              1; # X offset from bottom left for proc. sub-window
DET.WIN2.IPLLY              0; # Y offset from bottom left for proc. sub-window
DET.WIN2.IPURX              0; # X offset from top right for proc. sub-window
DET.WIN2.IPURY              0; # Y offset from top right for proc. sub-window
DET.WIN1.NDIT              10; # # of subintegrations
DET.WIN1.ASUIT1           "T"; # All UITi as UIT1 else as defined in UITi
DET.READ.SHIFTYP "idem    ";   # Line shift type
DET.READ.SHIFT1            29; # Lines shifted between integration
DET.WIN1.UIT1           0.001; # user defined subintegration time

# --- oOo ---
```

2.19 Examples for Scientific CCD Camera

2.19.1 Usage of CCD scientific cameras

The following example is a Sequencer script available on tape (file <TAPE>/CCD/ccdcon/ws/test/ccdconTestSci.tcl).
In the following we consider the case of an instrument not handling parallel exposures with the same camera (see 2.2)

```
#------------------------------------------------------------------------------
# Basic initialization of this application
# (register to CCS, fill in gvar array with definitions needed etc.)
ccdconTestInit Sci

#------------------------------------------------------------------------------
# Set operational mode to no simulation
# (write in WS OLDB and run ccdDcsDbSave.sh)
ccdSetOpMode $gvar(no_sim)

puts "-----------------------------------------------------------------------"
puts "                        Cold startup"
puts "-----------------------------------------------------------------------"
# (execute ccdDcsStart.sh script)
ccdCstart

puts "-----------------------------------------------------------------------"
puts "               Bring the CCD system to ONLINE state"
puts "-----------------------------------------------------------------------"
# STANDBY (optional)
# ONLINE
ccdOnLine

puts "-----------------------------------------------------------------------"
puts "      Load the complete setup file defining all application defaults"
puts "-----------------------------------------------------------------------"
# Important, otherwise CCD sw defaults would be used
ccdSendCmd SETUP "-expoId $gvar(ccdEXP_NEXT) -file
$gvar(setupFileDefaults)"

set defTime 5.0; # default integration time for this demo

puts "-------------------------------------------------------------------------"
```

```
puts "Attach to exp. status change event"
set evtId [ccdExpStatusAttach]
puts "-----------------------------------------------------------------------"

puts ""
puts "-----------------------------------------------------------------------"
puts "                       First observation"
puts "             setup: Full frame, display and save in file"
puts "-----------------------------------------------------------------------"
askUserAck
puts ""
# Make sure that FITS file does not exist and create space for header
set fileName xccdconTestSciFull.fits
fitsCreate $fileName
# Send SETUP to TCS, ICS, DCS
# Here only to DCS for simplicity
ccdSendCmd SETUP "-expoId $gvar(ccdEXP_NEXT) -function \
                  $gvar(SETUP_EXP_TYPE) $gvar(ccdEXP_NORMAL_STR) \
                  $gvar(SETUP_TIME_DEF) $defTime \
                  $gvar(SETUP_READ_WINBX_1) 1 \
                  $gvar(SETUP_READ_WINBY_1) 1 \
                  $gvar(SETUP_READ_WINEN_1) F \
                  $gvar(SETUP_READ_WINEN_2) F \
                  $gvar(SETUP_EXP_REPEAT) 1 \
                  $gvar(SETUP_DISPLAY) $gvar(ccdDISPLAY_MAINFRAME) \
                  $gvar(SETUP_TRAN_DISK) $gvar(ccdDISK_UNCOMPRESS) \
                  $gvar(SETUP_TRAN_UCOM) $fileName \
                  $gvar(SETUP_TRAN_RATE) 1"
# START exposure
set expoId [ccdSendCmd START]
# Wait for integration started
ccdExpStatusWait $gvar(ccdEXP_INTEGRATING)
# Wait for integration completion
ccdExpStatusWait $gvar(ccdEXP_READING) "<"
# Collect FITS information from ICS, TCS etc.
fitsCreateIns $fileName
fitsCreateTel $fileName
# Now I can start next SETUP to TCS, ICS and DCS
# Here only to DCS for simplicity
set fileNameNext xccdconTestSciBin.fits
ccdSendCmd SETUP "-expoId $gvar(ccdEXP_NEXT) -function \
                 $gvar(SETUP_READ_WINBX_1) 2 \
                 $gvar(SETUP_READ_WINBY_1) 2 \
                 $gvar(SETUP_TRAN_UCOM) $fileNameNext"
# Wait for exposure completion
send_to_rtd autocut
ccdExpStatusWait $gvar(ccdEXP_COMPLETED) "<"
send_to_rtd autocut
# Merge DCS, TCS and ICS information in the image file
fitsMerge $fileName
# Inform Archive
archive $fileName

puts ""
puts "-----------------------------------------------------------------------"
puts "                      Second observation"
```

```
puts "          setup: Binned frame, display and save in file"
puts "-----------------------------------------------------------------------"
askUserAck
puts ""
set fileName $fileNameNext
# Make sure that FITS file does not exist and create space for header
fitsCreate $fileName
# SETUP to TCS, ICS, DCS already sent before
# START exposure
set expoId [ccdSendCmd START]
# Wait for integration started
ccdExpStatusWait $gvar(ccdEXP_INTEGRATING)
# Wait for integration completion
ccdExpStatusWait $gvar(ccdEXP_READING) "<"
# Collect FITS information from ICS, TCS etc.
fitsCreateIns $fileName
fitsCreateTel $fileName
# Now I can start next SETUP to TCS, ICS and DCS
# Here only to DCS for simplicity
set fileNameNext xccdconTestSciWin.fits
ccdSendCmd SETUP "-expoId $gvar(ccdEXP_NEXT) -function \
                 $gvar(SETUP_READ_WINBX_1) 1 \
                 $gvar(SETUP_READ_WINBY_1) 1 \
                 $gvar(SETUP_READ_WINEN_1) T \
                 $gvar(SETUP_READ_WINNX_1) 100 \
                 $gvar(SETUP_READ_WINNY_1) 100 \
                 $gvar(SETUP_READ_WINSX_1) 1 \
                 $gvar(SETUP_READ_WINSY_1) 1 \
                 $gvar(SETUP_READ_WINEN_2) T \
                 $gvar(SETUP_READ_WINNX_2) 100 \
                 $gvar(SETUP_READ_WINNY_2) 100 \
                 $gvar(SETUP_READ_WINSX_2) 150 \
                 $gvar(SETUP_READ_WINSY_2) 1 \
                 $gvar(SETUP_TRAN_UCOM) $fileNameNext"
# Wait for exposure completion
send_to_rtd autocut
ccdExpStatusWait $gvar(ccdEXP_COMPLETED) "<"
send_to_rtd autocut
# Merge DCS information in the image file
fitsMerge $fileName
# Inform Archive
archive $fileName

puts ""
puts "-----------------------------------------------------------------------"
puts "                        Last observation"
puts "         setup: 2 Windows frame, display and save in file"
puts "-----------------------------------------------------------------------"
askUserAck
puts ""
set fileName $fileNameNext
# Make sure that FITS file does not exist and create space for header
fitsCreate $fileName
# SETUP to TCS, ICS, DCS already sent before
# START exposure
set expoId [ccdSendCmd START]
# Wait for integration started
ccdExpStatusWait $gvar(ccdEXP_INTEGRATING)
```

```
# Wait for integration completion
ccdExpStatusWait $gvar(ccdEXP_READING) "<"
# Collect FITS information from ICS, TCS etc.
fitsCreateIns $fileName
fitsCreateTel $fileName
# Wait for exposure completion
send_to_rtd autocut
ccdExpStatusWait $gvar(ccdEXP_COMPLETED) "<"
send_to_rtd autocut
# Merge DCS information in the image file
fitsMerge $fileName
# Inform Archive
archive $fileName

puts "------------------------------------------------------------------------"
puts "                             Shutdown camera"
puts "------------------------------------------------------------------------"
# (execute ccdDcsStop.sh script)
ccdShutdown

#-------------------------------------------------------------------
# Terminate application
ccdconTestExit
```

3 REFERENCE

This chapter provides a detailed description of the CCD software programmatic interface, namely functions, programs (including Command Definition Tables), scripts and include files, together with various template files.

3.1 Functions

Functions provided by the CCD software are grouped in the library *libccd.a*, available on Workstation only.

The functions provided by the CCD software to external software are:

1. **`ccdCheckSetup`**. It checks the validity of a complete setup and returns also some computed values, such as the estimated read-out time.
2. **`ccdGetCIName`**. It returns the name of the CCD Command Interface process.
3. **`ccdGetConf`**. It returns all configuration values for the camera used.

The LCU Image Processing User Functions are located in the binary file *ccdipUSR* (automatically loaded at boot time):

4. **`ccdipAG`** : centroid calculation (IRAF-based algorithm)

   This function recenters the processing window on the centroid of the image
5. **`ccdipCV`** : centroid validation (multi-criterium check)

   Thie function checks various image parameters against limits
6. **`ccdipIQE`** : centroid & image quality estimation (SkyCat-based algorithm)

   This function estimates the sky background and computes the second-order moment of the intensity distribution, a 2D Gaussian fit delivers centroid and FWHM
7. **`ccdipWCE`** : wavefront coherence estimation (Astronomical Site Monitor only)

   This function processes the stripes obtained during a multiple exposure. It performs the centroid calculation on each stripe.

The function **ccdipUsrFuncTemplate** is a template for LCU real-time image processing functions.

In addition, the library *libccdObj.a* provides functions specialized in the detection of objects within images. They are:

1. **`ccdObjAll.`** Detects objects in a frame from a FITS file and returns their location and intensity.
2. **`ccdObjBright`**. Detects objects in a frame from a FITS file and returns location and intensity of the brightest one.
3. **`ccdObjClose`**. Detects objects in a frame from a FITS file and returns location and intensity of the closest one to a reference point.
4. **`ccdObjDisplay`**. Displays objects in a frame with Midas display.

3.1.1 ccdCheckSetup(3)

NAME

ccdCheckSetup, ccdCheckSetupWindow - Check a setup for a CCD camera

SYNOPSIS

```
#include "ccd.h"

ccsCOMPL_STAT ccdCheckSetup (
                             ccdCONFIG   *config,
                             ccdSETUP    *setup,
                             ccdSETUPRES *results,
                             ccsERROR    *error
                            )

ccsCOMPL_STAT ccdCheckSetupWindow (
                                   ccdCONFIG   *config,
                                   ccdSETUP    *setup,
                                   ccdSETUPRES *results,
                                   ccsERROR    *error
                                  )
```

DESCRIPTION

These functions perform a static check of the defined setup parameters against the configuration. This check does NOT do any dynamic check against the current state of the camera.
The check is stopped after the first error detection and the function returns with FAILURE.

ccdCheckSetup checks a complete setup. It also returns values derived from the setup parameters.

ccdCheckSetupWindow checks only the part of the setup concerning the windowed readout.

```
IN    config     information about camera configuration (see ccdGetConf)
INOUT setup      structure containing exposure setup (see ccd.h)
OUT   results    structure containing check results (see ccd.h)
OUT   error      error structure
```

RETURN VALUES

SUCCESS if everything OK
FAILURE if error occurs

EXAMPLES

```
#include "ccd.h"
....
ccdCAMERA   camera = {"ccdFors", ccsLOCAL_ENV};
ccdCONFIG   config;
ccdSETUP    setup;
ccdSETUPRES results;
ccsERROR    error;
....
if (ccdGetConf(&camera, &config, &error) == FAILURE)
    {
    ... handle failure
    }
```

```
        ....
           fill in values in setup structure
        ....
        if (ccdCheckSetup(&config, &setup, &results, &error) == FAILURE)
            {
            ... handle failure
            }
```

SEE ALSO

ccdGetConf

\- - - - - -
Last change: 08/10/97-13:47

3.1.2 ccdGetCIName(3)

NAME

```
ccdGetCIName - Get name of Command Interface Process for a CCD camera
```

SYNOPSIS

```
#include "ccd.h"
void ccdGetCIName(ccdCAMERANAME camera,
                  ccsPROCNAME procName)
```

DESCRIPTION

```
This function returns the name of the only CCD process communicating
with external sw through the CCS message system to control a specific
CCD camera.
IN   camera       name of the camera used
OUT  procName     name of the Command Interface process for camera used
```

RETURN VALUES

```
None
```

EXAMPLES

```
#include "ccd.h"
....
ccdCAMERA camera = {"ccdFors", ccsLOCAL_ENV};
ccsPROCESS ccdProcess;
....
ccdGetCIName(camera.name, ccdProcess);
... ccdProcess is now set to "ccdconCI_ccdFors"
```

```
- - - - - -
Last change:  08/10/97-13:47
```

3.1.3 ccdGetConf(3)

NAME

```
ccdGetConf - Get information about a CCD camera configuration
```

SYNOPSIS

```
#include "ccd.h"
ccsCOMPL_STAT ccdGetConf(ccdCAMERA *camera,
                         ccdCONFIG *config,
                         ccsERROR  *error)
```

DESCRIPTION

```
This function returns information concerning the configuration of a
CCD camera, which might be needed by software packages using the CCD sw.
IN  camera       structure containing the following elements:
       name      name of the camera used
       envName   name of the CCS environment where CCD software runs
                 It can be set to ccsLOCAL_ENV for local env.
OUT config       information about camera configuration
OUT error        error structure
```

RETURN VALUES

```
SUCCESS if everything OK
FAILURE if error occurs
```

EXAMPLES

```
 #include "ccd.h"
 ....
 ccdCAMERA camera = {"ccdFors", ccsLOCAL_ENV};
 ccdCONFIG config;
 ccsERROR  error;
 ....
 if (ccdGetConf(&camera, &config, &error) == FAILURE)
     {
     ... handle failure
     }
```

```
- - - - - -
Last change:  08/10/97-13:47
```

3.1.4 ccdipAG(3)

NAME

```
ccdipAG - Auto Guiding (User Hook Function)
```

SYNOPSIS

```
#include "ccd.h"
#include "ccdipAG.h"

ccsCOMPL_STAT ccdipAG (ccdIPIMAGE *img,ccdipAGBUF *agb,ccsERROR *err)
```

DESCRIPTION

```
This function determines the Error Vector to the object contained
in the image in the vicinity of the Reference Point:
- error vector

<img>   IN    image to process
<agb>   INOUT AG data
<err>   OUT   error structure

The debug flag ccdDEBUG_IP_USER (see ccdInternal.h) may be set to trace
```

SEE ALSO

```
ccdipIQE
```

```
- - - - - -
Last change:  22/09/98-08:26
```

3.1.5 ccdipAG.h

```
/*******************************************************************************
* E.S.O. - VLT project
*
* "@(#) $Id: ccdipAG.h,v 1.56 1998/09/17 08:53:15 vltsccm Exp $"
*
* who       when      what
* --------  --------  ----------------------------------------------
* pduhoux  27/05/98  created
*/

/*
 ************************************************************************
 * Auto Guiding : data structure & function prototype
 *------------------------------------------------------------------------
 */

#ifndef CCDIP_AG_H
#define CCDIP_AG_H

#ifdef __cplusplus
extern "C" {
#endif

typedef struct
    {
    vltDOUBLE xErrVec,yErrVec;               /* error vector                 */
    } ccdipAGBUF;

ccsCOMPL_STAT ccdipAG (ccdIPIMAGE *img, ccdipAGBUF *agb, ccsERROR *err);

#ifdef __cplusplus
}
#endif

#endif /*!CCDIP_AG_H*/
```

3.1.6 ccdipCV(3)

NAME

```
ccdipCV - Centroid Validation
```

SYNOPSIS

```
#include "ccd.h"
#include "ccdipCV.h"

ccsCOMPL_STAT ccdipCV (ccdIPIMAGE *img,ccdipCVBUF *cvb,ccsERROR *err)
```

DESCRIPTION

```
This function checks the validity of the Centroid with respect
to the given validation criteria.

<img>   IN    image to process
<cvb>   INOUT Centroid Validation limits
<err>   OUT   error structure

If any of the validation criterium is set to zero, the check against
this criterium is skipped.

<cvb->minNumPix>     : minimum number of valid pixels
<cvb->minSNR>        : minimum SNR
<cvb->maxFWHM>       : maximum image size in pixels
<cvb->maxElongation> : maximum image elongation in pixels (> 1)
<cvb->maxOffset>     : maximum error vector length in pixels

Whenever a criterium is not satisfied, the Intensity of the
centroid is set to ZERO.
```

RETURN VALUES

```
SUCCESS always
```

SEE ALSO

```
ccdipCentroid
```

```
- - - - - -
Last change:  22/09/98-08:26
```

3.1.7 ccdipCV.h

```
/*******************************************************************************
* E.S.O. - VLT project
*
* "@(#) $Id: ccdipCV.h,v 1.56 1998/09/17 08:53:16 vltsccm Exp $"
*
* who       when      what
* --------  --------  ----------------------------------------------
* pduhoux   17/09/98  created
*/

/*
 ************************************************************************
 * Centroid Validation : data structure & function prototype
 *------------------------------------------------------------------------
 */

#ifndef CCDIP_CV_H
#define CCDIP_CV_H

#ifdef __cplusplus
extern "C" {
#endif

typedef struct
    {
    vltINT32   minNumPix;
    vltDOUBLE  minSNR;
    vltDOUBLE  maxFWHM;
    vltDOUBLE  maxElongation;
    vltDOUBLE  maxOffset;
    } ccdipCVBUF;

ccsCOMPL_STAT ccdipCV (ccdIPIMAGE *img, ccdipCVBUF *cvb, ccsERROR *err);

#ifdef __cplusplus
}
#endif

#endif /*!CCDIP_CV_H*/
```

3.1.8 ccdipIQE(3)

NAME

ccdipIQE - Image Quality Estimation (User Hook Function)

SYNOPSIS

```
#include "ccd.h"
#include "ccdipIQE.h"

ccsCOMPL_STAT ccdipIQE  (ccdIPIMAGE *img,ccdipIQEBUF *iqeb,ccsERROR *err)
```

DESCRIPTION

```
This function determines the Image Quality for the object contained
in the image in the vicinity of the Reference Point:
- position
- FWHM on both axis
- orientation angle
- peak intensity
- backGroung intensity
Each value is given with Standard Deviation

The function ccdipIQE  works on the 2D image

<img>   IN    image to process
<iqeb>  INOUT IQE data
<err>   OUT   error structure
```

SEE ALSO

ccdipAG

\- - - - - -
Last change: 22/09/98-08:26

3.1.9 ccdipIQE.h

```
/*******************************************************************************
* E.S.O. - VLT project
*
* "@(#) $Id: ccdipIQE.h,v 1.56 1998/09/17 08:53:15 vltsccm Exp $"
*
* who       when      what
* --------  --------  ----------------------------------------------
* pduhoux  27/05/98  created
*/

/*
 ************************************************************************
 * Image Quality Estimation : data structure & function prototype
 *------------------------------------------------------------------------
 */

#ifndef CCDIP_IQE_H
#define CCDIP_IQE_H

#ifdef __cplusplus
extern "C" {
#endif

typedef struct
    {
    vltDOUBLE xMPos,xMPos_SD;              /* mean X position       */
    vltDOUBLE yMPos,yMPos_SD;              /* mean Y position       */
    vltDOUBLE xFWHM,xFWHM_SD;              /* gauss fit X FWHM      */
    vltDOUBLE yFWHM,yFWHM_SD;              /* gauss fit Y FWHM      */
    vltDOUBLE ObjAngle,ObjAngle_SD;        /* orientation [degrees] */
    vltDOUBLE ObjPeak,ObjPeak_SD;          /* peak intensity        */
    vltDOUBLE BGnd,BGnd_SD;                /* back ground level     */
    } ccdipIQEBUF;

ccsCOMPL_STAT ccdipIQE  (ccdIPIMAGE *img, ccdipIQEBUF *iqeb, ccsERROR *err);

#ifdef __cplusplus
}
#endif

#endif /*!CCDIP_IQE_H*/
```

3.1.10 ccdipWCE(3)

NAME

```
ccdipWCE - Compute the centroids on multi-step window
```

SYNOPSIS

```
#include "ccd.h"
#include "ccdipWCE.h"

ccsCOMPL_STAT ccdipWCE (ccdIPIMAGE *img,ccdipWCEBUF *wceb,ccsERROR *err)
```

DESCRIPTION

```
This function computes for each stripe in the window:
   - the position of the center of mass (centroid / center of gravity)
   - the number of valid pixels in the stripe
   - raw intensity/pixel + sigma
   - sky background estimation/pixel + sigma
   - SNR & FWHM estimations

<img>        IN     pointer to image description
<wceb>       INOUT  WCE data
<err>        OUT    error stack (unused)
```

FILES

```
None
```

ENVIRONMENT

```
None
```

RETURN VALUES

```
SUCCESS   always
FAILURE   if the image data type should differ from vltINT8, vltINT16
          or vltFLOAT. This exceptional case is probably an advice of
          memory corruption, or an indication that the application may
          have modified the <img> structure.
```

CAUTIONS

```
compile with the "OPTIMIZE=9" option, in order to have better
   performances.
```

SEE ALSO

```
ccdipBackground(l), ccdipStatistic(l), ccdipCentroid(l)
```

```
- - - - - -
Last change:  22/09/98-08:26
```

3.1.11 ccdipWCE.h

```
/*******************************************************************************
* E.S.O. - VLT project
*
* "@(#) $Id: ccdipWCE.h,v 1.56 1998/09/17 08:53:15 vltsccm Exp $"
*
* who       when      what
* --------  --------  ----------------------------------------------
* pduhoux  27/05/98  created
*/

/*
 ************************************************************************
 * Wavefront Coherence Estimation : data structure & function prototype
 *-----------------------------------------------------------------------
 */

#ifndef CCDIP_WCE_H
#define CCDIP_WCE_H

#ifdef __cplusplus
extern "C" {
#endif

typedef struct
    {
    /*
     * This structure contains the results of the coherence mode centroid
     * computation for one stripe:
     */
    vltDOUBLE    flux;               /* Raw Average ADC counts / pixel      */
    vltDOUBLE    sigma;              /* Standard-Deviation on the flux      */
    vltDOUBLE    BckGnd,BckGnd_SD;   /* Back Ground estimation + Std Dev    */
    vltDOUBLE    xFWHM,yFWHM;        /* X & Y FWHM on object                */
    vltDOUBLE    SNR;                /* Signal-to-Noise Ratio               */
    vltINT32     numPix;             /* Number of valid pixels in image     */
    vltDOUBLE    cenVal,xCen,yCen;   /* Location of Centroid relative       */
                                     /* to stripe LL corner                 */
    } ccdipSTRIPE;

typedef struct
    {
    /*
     * This structure contains the results of the coherence mode centroid
     * computation:
     *   - numStripe : number of stripes
     *   - lenStripe : height of stripes
     *   - expTimeDef: expTime / stripe
     *   - expTime   : effective expTime / stripe
     *   - numValPix : number of valid pixels in each stripe
     *                 (pix - bckGnd) > thrMin
     *                 When more than 1 spot, forced to -1
     *   - xCen,yCen : coordinates of spot centroid relative to LL corner
     *                 of the stripe.
     */
    ccsTIMEVAL   utcStart;                /* UTC of first stripe            */
    vltINT32     numStripe;               /* Number of stripes              */
```

```
    vltINT32     lenStripe;                /* Height of stripes             */
    vltDOUBLE    lapTime;                  /* Effective Time between stripe  */
    vltDOUBLE    expTime;                  /* Effective expTime/stripe       */
    ccdipSTRIPE  stripe[2][ccdMAXINTEGR];  /* Centroid / stripe / window     */
    } ccdipWCEBUF;

ccsCOMPL_STAT ccdipWCE  (ccdIPIMAGE *img, ccdipWCEBUF *iqeb, ccsERROR *err);

#ifdef __cplusplus
}
#endif

#endif /*!CCDIP_WCE_H*/
```

3.1.12 ccdipUsrFuncTemplate(3)

NAME

```
ccdipUsrFuncTemplate - Template for User-Defined Image Processing Function
```

SYNOPSIS

```
#include "ccd.h"

ccsCOMPL_STAT ccdipUsrFuncTemplate (
                                   ccdIPIMAGE *image,
                                   void       *usrBuffer,
                                   ccsERROR   *error
                                  )
```

DESCRIPTION

```
This function logs the content of the <usrBuffer>, expected as a char ptr

<image>      INOUT pointer to image description structure (see ccd.h).
<usrBuffer>  INOUT user defined I/O buffer
<error>       OUT  error stack

The debug flag ccdDEBUG_IP_USER (see ccdInternal.h) may be set to trace
the function progress.
```

\- - - - - -
Last change: 08/10/97-13:49

3.1.13 ccdObjAll(3)

NAME

ccdObjAll - Get all objects detected in an image

SYNOPSIS

```
#include "ccd.h"
ccsCOMPL_STAT ccdObjAll(char *file,
                        const ccdOBJMETHOD *method,
                        vltINT32  *objectsN,
                        ccdOBJECT **objects,
                        ccsERROR *error)
```

DESCRIPTION

```
This function uses the ccdMidasObj.prg on-line Midas procedure to
determine the position of stars in an image stored in a FITS file.
This function assumes that a Midas session is already running in
background and the pco process is active in the current CCS environment.
The memory space pointed by the parameter "objects" is allocated by this
function. It is released with the next call.

IN   file         name of the FITS file contianing image
IN   method       contains parameters needed to find the objects
       procId     ID number (between 0 and 99) of the running Midas
       threshAbs  if ccsTRUE, threshVal is interpreted as an absolute
                              threshold value
                  if ccsFALSE, the threshold is computed as threshVal *
                               image standard deviation
       threshVal  threshold value. The meaning depends on the value of
                  threshAbs
OUT  objectsN     number of objects detected
OUT  objects      array of objectsN elements, each consisting of:
       xPos       location of object along x axis
       yPos       location of object along y axis
       intensity  intensity of object
OUT  error        error structure
```

FILES

```
The following working files are created by the Midas procedure used.
midasObj0001.bdf    image in bdf format for Midas
midasObj0001SKY.tbl table containing intermediate sky parameters
midasObj0001CAT.tbl table containing final object parameters
```

ENVIRONMENT

HOST it must be defined to specify the local machine

RETURN VALUES

SUCCESS if everything OK
FAILURE if some error has been detected

EXAMPLES

```
#include "ccd.h"
....
char imageFile[256];
ccdOBJMETHOD method;
vltINT32 objectsN;
```

```
ccdOBJECT *objects;
ccsERROR error;
....
strcpy(imageFile, "myFile.fits");
method.procId = 50;
method.threshAbs = ccsFALSE;
method.threshVal = 3;
if (ccdObjAll(imageFile, &method, &objectsN, &objects, &error) != SUCCESS)
    {
    printf("Error. Exit\n");
    return(EXIT_FAILURE);
    }
for (i=0 ; i<objectsN; i++)
    {
    printf("Object #%3d  position X %6.2f  Y %6.2f  intensity %6.2f\n",
           i, objects[i].xPos, objects[i].yPos,
           objects[i].intensity);
    }
....
```

SEE ALSO

```
ccdObjBright, ccdObjClose
ccdObjDisplay
ccdMidasBgStart.sh
ccdMidasObj.prg
Midas ROMAFIT package (Midas Manual Volume B)
ccdTestObj for an example of application using it
```

```
- - - - - -
Last change:  08/10/97-13:47
```

3.1.14 ccdObjBright(3)

NAME

```
ccdObjBright - Get the brightest object detected in an image
```

SYNOPSIS

```
#include "ccd.h"
ccsCOMPL_STAT ccdObjBright(char *file,
                           const ccdOBJMETHOD *method,
                           ccdOBJECT *object,
                           ccsERROR *error)
```

DESCRIPTION

```
This function uses the ccdMidasObj.prg on-line Midas procedure to
determine the position of stars in an image stored in a FITS file and
returns position and intensity of the brightest one.
This function assumes that a Midas session is already running in
background and the pco process is active in the current CCS environment.

IN   file         name of the FITS file contianing image
IN   method       contains parameters needed to find the objects
       procId     ID number (between 0 and 99) of the running Midas
       threshAbs  if ccsTRUE, threshVal is interpreted as an absolute
                             threshold value
                  if ccsFALSE, the threshold is computed as threshVal *
                              image standard deviation
       threshVal  threshold value. The meaning depends on the value of
                  threshAbs
OUT  object       info about brightest object, consisting of:
       xPos       location of object along x axis (-1 if no object found)
       yPos       location of object along y axis (-1 if no object found)
       intensity  intensity of object (-1 if no object found)
OUT  error        error structure
```

FILES

```
The following working files are created by the Midas procedure used.
midasObj0001.bdf    image in bdf format for Midas
midasObj0001SKY.tbl table containing intermediate sky parameters
midasObj0001CAT.tbl table containing final object parameters
```

ENVIRONMENT

```
HOST it must be defined to specify the local machine
```

RETURN VALUES

```
SUCCESS if everything OK
FAILURE if some error has been detected
```

EXAMPLES

```
 #include "ccd.h"
 ....
 char imageFile[256];
 ccdOBJMETHOD method;
 ccdOBJECT object;
 ccsERROR error;
 ....
 strcpy(imageFile, "myFile.fits");
```

```
method.procId = 50;
method.threshAbs = ccsFALSE;
method.threshVal = 3;
if (ccdObjBright(imageFile, &method, &object, &error) != SUCCESS)
    {
    printf("Error. Exit\n");
    return(EXIT_FAILURE);
    }
printf("Brightest Object position X %6.2f  Y %6.2f  intensity %6.2f\n",
       object.xPos, object.yPos, object.intensity);
....
```

SEE ALSO

```
ccdObjAll, ccdObjClose
ccdMidasObj.prg
Midas ROMAFIT package (Midas Manual Volume B)
ccdTestObj for an example of application using it
```

```
- - - - - -
Last change:  08/10/97-13:47
```

3.1.15 ccdObjClose(3)

NAME

```
ccdObjClose - Get the closest object to a reference point in an image
```

SYNOPSIS

```
#include "ccd.h"
ccsCOMPL_STAT ccdObjClose(char *file,
                           const ccdOBJMETHOD *method,
                           ccdOBJECT *reference,
                           ccdOBJECT *object,
                           ccsERROR *error)
```

DESCRIPTION

```
This function uses the ccdMidasObj.prg on-line Midas procedure to
determine the position of stars in an image stored in a FITS file and
returns position and intensity of the one closest to the specified
reference position
This function assumes that a Midas session is already running in
background and the pco process is active in the current CCS environment.

IN   file         name of the FITS file contianing image
IN   method       contains parameters needed to find the objects
       procId     ID number (between 0 and 99) of the running Midas
       threshAbs  if ccsTRUE, threshVal is interpreted as an absolute
                              threshold value
                  if ccsFALSE, the threshold is computed as threshVal *
                               image standard deviation
       threshVal  threshold value. The meaning depends on the value of
                  threshAbs
OUT  reference    info about reference position, consisting of:
       xPos       position along x axis
       yPos       position along y axis
       intensity  not used
OUT  object       info about closest object, consisting of:
       xPos       location of object along x axis (-1 if no object found)
       yPos       location of object along y axis (-1 if no object found)
       intensity  intensity of object (-1 if no object found)
OUT  error        error structure
```

FILES

```
The following working files are created by the Midas procedure used.
midasObj0001.bdf    image in bdf format for Midas
midasObj0001SKY.tbl table containing intermediate sky parameters
midasObj0001CAT.tbl table containing final object parameters
```

ENVIRONMENT

```
HOST it must be defined to specify the local machine
```

RETURN VALUES

```
SUCCESS if everything OK
FAILURE if some error has been detected
```

EXAMPLES

```
 #include "ccd.h"
 ....
```

```
char imageFile[256];
ccdOBJMETHOD method;
ccdOBJECT object;
ccsERROR error;
....
strcpy(imageFile, "myFile.fits");
method.procId = 50;
method.threshAbs = ccsFALSE;
method.threshVal = 3;
reference.xPos = 200;
reference.yPos = 150;
if (ccdObjClose(imageFile, &method, &reference, &object, &error)
    != SUCCESS)
    {
    printf("Error. Exit\n");
    return(EXIT_FAILURE);
    }
printf("Object reference position X %6.2f  Y %6.2f\n",
       reference.xPos, reference.yPos);
printf("Object closest   position X %6.2f  Y %6.2f  intensity %6.2f\n",
       object.xPos, object.yPos, object.intensity);
....
```

SEE ALSO

```
ccdObjAll, ccdObjClose
ccdMidasObj.prg
Midas ROMAFIT package (Midas Manual Volume B)
ccdTestObj for an example of application using it
```

```
- - - - - -
Last change:  08/10/97-13:47
```

3.1.16 ccdObjDisplay(3)

NAME

```
ccdObjDisplay - Display the image analysed with all objects detected
```

SYNOPSIS

```
#include "ccd.h"
ccsCOMPL_STAT ccdObjDisplay(char *file,
                            vltUINT8 procId,
                            ccsERROR *error)
```

DESCRIPTION

```
This function uses the ccdDisplayMidasObj.prg on-line Midas procedure to
display the image analysed with ccdObjAll and show with circles the
location of the objects found.

IN   file         name of the FITS file contianing image
IN   procId     ID number (between 0 and 99) of the running Midas
OUT  error        error structure
```

FILES

```
The following working files are assumed to be present.
midasObj0001.bdf    image in bdf format for Midas
midasObj0001SKY.tbl table containing intermediate sky parameters
midasObj0001CAT.tbl table containing final object parameters
```

ENVIRONMENT

```
HOST it must be defined to specify the local machine
```

RETURN VALUES

```
SUCCESS if everything OK
FAILURE if some error has been detected
```

EXAMPLES

```
#include "ccd.h"
....
char imageFile[256];
ccsERROR error;
....
strcpy(imageFile, "myFile.fits");
if (ccdObjDisplay(imageFile, 50, &error) != SUCCESS)
    {
    printf("Error. Exit\n");
    return(EXIT_FAILURE);
    }
....
```

SEE ALSO

```
ccdObjAll
ccdDisplayMidasObj.prg
Midas ROMAFIT package (Midas Manual Volume B)
ccdTestObj for an example of application using it
```

```
- - - - - -
Last change:  08/10/97-13:47
```

3.2 Programs

The name of all CCD processes within their respective environment is built as <*program name*>*_$CCDNAME*.

The only program representing the Command Interface to external application software is *ccdconCI*, which runs on Workstation. The corresponding process for the camera *myccd* is therefore *ccdconCI_myccd*.

The script *ccdDcsCdt.h*, called within *ccdDcsStart.sh*, creates in $INTROOT/CDT (or $VLTROOT/CDT) the CDTs for the CCD camera processes as symbolic links to the CDT of the respective program (e.g. *ccdconCI_myccd.cdt* is created as symbolic link to *ccdconCI.cdt*).

3.2.1 Command Definition Table for program ccdconCI

```
//*************************************************************************
// E.S.O. - VLT project
//
// "@(#) $Id: ccdconCI.cdt,v 2.2 1998/09/10 14:24:39 vltsccm Exp $"
//
// Command definition table for program ccdconCI (main task WS CCD software).
//
// who            when      what
// -------------  --------  --------------------------------------------
// A.Longinotti   14/11/94  First version
// C.Cumani       05/04/95  keywords reordered to follow FEB95 rel.
// A.Longinotti   04/08/95  Updated according to INS common sw specs 2.0
// A.Longinotti   01/02/96  Uncommented PAR_MAX_REPETITION for SETUP
// A.Longinotti   04/06/96  OBJECTS, OBJBRGT, OBJCLOS removed (SPR960204)
//                          expoId removed for CONT (SPR960292)
// A.Longinotti   12/07/96  added WAIT
// A.Longinotti   13/12/96  added STANDAL
// A.Longinotti   16/12/96  added parameter archive to STANDAL
// A.Longinotti   17/01/97  default for param. 'at' set to "now" (SPR960680)
// P.Duhoux       13/02/97  added param <literal> to STATUS & WAIT commands
//                          for reply format specification
// P.Duhoux       14/02/97  added cmd ONLINE = OPERATE (SPR970045)
// P.Duhoux       12/03/97  added cmd STARTTL/STOPTL and STARTTM/STOPTM for
//                          Telemetry & Temperature monitoring
// P.Duhoux       12/03/97  added cmd OFF = PDOWN
// A.Longinotti   18/04/97  Improved description of DUMP (SPR960680)
// A.Longinotti   23/05/97  Added test command DEBUG
// P.Duhoux       04/03/98  Improved description of STOP (SPR980004)
// P.Duhoux       12/08/98  Improved description of BIAS,DISPLAY,FLAT,
//                          STARTAG & STARTLP (SPR980415)
// P.Duhoux       12/08/98  Added command STPWAIT (SPR980004)
//------------------------------------------------------------------------

//==========================================================================

PUBLIC_COMMANDS

//==========================================================================
```

```
// Standard application commands (see CCS User Manual)
//============================================================================

//============================================================================

COMMAND=  EXIT
SYNONYMS=
FORMAT= A
PARAMETERS=
REPLY_FORMAT= A
HELP_TEXT =
Terminate the whole CCD sw
One reply only at end of execution
@

//============================================================================
// Standard INS commands for DCS (see INS common sw 2.0 04/05/95)
//============================================================================

COMMAND=  ABORT
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        0
REPLY_FORMAT = A
HELP_TEXT=
Abort exposure with ID <expoId> (default last started exposure).
Image data are lost.
One reply only at end of execution

----> Currently parameter expoId ignored. Assumed always 0 <----
@

//============================================================================

COMMAND=  CONT
SYNONYMS= continue
FORMAT= A
PARAMETERS=
    PAR_NAME=           at
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "now"
REPLY_FORMAT = A
HELP_TEXT=
Continue a paused exposure at a given optional time. Default is 'now'
The time parameter <at> shall comply ISO8601, ie YYYY-MM-DDThh:mm:ss.uuu
Available only for CCD equipped with a shutter.
One reply only at end of execution

@

//=====================================================================

COMMAND=  DUMP
```

```
SYNONYMS=
FORMAT= A
PARAMETERS=
REPLY_FORMAT = A
HELP_TEXT=
Dump the last acquired image to disk (same directory where all other images
are stored, see UM, file name as from last SETUP command submitted).
Should be used only for recovery purposes, after DCS got stuck or did not
succeed in obtaining the image.
One reply only at end of execution
@

//=====================================================================

COMMAND=  END
SYNONYMS=
FORMAT= A
PARAMETERS=
REPLY_FORMAT = A
HELP_TEXT=
End the current exposure(s) and read out the data.
One reply only at end of execution
@

//=====================================================================

COMMAND=  INIT
SYNONYMS= initialize, initialization
FORMAT= A
PARAMETERS=
    PAR_NAME=           function
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
//    PAR_DEF_VAL=        "all"
    PAR_MAX_REPETITION= 999
REPLY_FORMAT = A
HELP_TEXT=
Initialize the functions contained in the list of arguments.
One reply only at end of execution

----> Currently parameter function ignored. Assumed always "all" <----
@

//=====================================================================

COMMAND=  OFF
SYNONYMS= off
FORMAT= A
PARAMETERS=
REPLY_FORMAT = A
HELP_TEXT=
Bring the system to operational state LOADED
One reply only at end of execution
@

//=====================================================================

COMMAND=  OPERATE
```

```
SYNONYMS= online
FORMAT= A
PARAMETERS=
REPLY_FORMAT = A
HELP_TEXT=
Bring the system to operational state OPERATING=ONLINE
One reply only at end of execution
@

//=====================================================================

COMMAND=  ONLINE
SYNONYMS= online
FORMAT= A
PARAMETERS=
REPLY_FORMAT = A
HELP_TEXT=
Bring the system to operational state OPERATING=ONLINE
One reply only at end of execution
@

//=====================================================================

COMMAND=  PAUSE
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           at
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "now"
REPLY_FORMAT = A
HELP_TEXT=
Pause an exposure at given optional time. Default is 'now'.
The time parameter <at> shall comply ISO8601, ie YYYY-MM-DDThh:mm:ss.uuu
Available only for CCD equipped with a shutter.
One reply only at end of execution

@

//=====================================================================

COMMAND=  PDOWN
SYNONYMS= off
FORMAT= A
PARAMETERS=
REPLY_FORMAT = A
HELP_TEXT=
Bring the system to operational state LOADED
One reply only at end of execution
@

//=====================================================================

COMMAND=  PXQUERY
SYNONYMS=
FORMAT= A
PARAMETERS=
```

```
    PAR_NAME=           window
    PAR_TYPE=           INTEGER
    PAR_OPTIONAL=       NO
    PAR_MAX_REPETITION= 4
REPLY_FORMAT = B
HELP_TEXT=
Query the raw value of one or more pixels, given the detector coordinates
of the window.
One reply only at end of execution

----> WS simulation only ! <----
@

//=====================================================================

COMMAND=  SELFTST
SYNONYMS= selftest
FORMAT= A
PARAMETERS=
    PAR_NAME=           function
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
//    PAR_DEF_VAL=        "all"
    PAR_MAX_REPETITION= 999
REPLY_FORMAT = A
HELP_TEXT=
Execute a self-test (sw and hw) of the function(s) specified
One reply only at end of execution

----> WS simulation only ! <----
----> Currently parameter function ignored. Assumed always "all" <----
@

//=====================================================================

COMMAND=  SETUP
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        -1

    PAR_NAME=           file
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_MAX_REPETITION= 999

    PAR_NAME=           function
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_MAX_REPETITION= 999

    PAR_NAME=           noMove
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES
```

```
    PAR_NAME=           check
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES
REPLY_FORMAT = A
HELP_TEXT=
Set-up the functions as listed.
Parameter expoId:  -1 = next exposure to start; 0 = last exposure started;
or current expoId when exposure is running.
One reply only at end of execution

----> Currently parameter noMove ignored. Always FALSE considered <----
----> Currently parameter check  ignored. Always FALSE considered <----

The full list of setup keywords can be found in the file :
    $INS_ROOT/SYSTEM/COMMON/SETUPFILES/DET/ccdSetupComplete.det

@

//=====================================================================

COMMAND=  STANDBY
SYNONYMS=
FORMAT= A
PARAMETERS=
REPLY_FORMAT = A
HELP_TEXT=
Bring the system to operational state STAND-BY
One reply only at end of execution
@

//=====================================================================

COMMAND=  START
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           at
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "now"
REPLY_FORMAT = A
REPLY_PARAMETERS=
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        11

HELP_TEXT=
Start an exposure at given optional time. Default is 'now'.
The time parameter <at> shall comply ISO8601, ie YYYY-MM-DDThh:mm:ss.uuu
One reply only when exposure started.
@

//=====================================================================

COMMAND=  STATUS
SYNONYMS=
FORMAT= A
PARAMETERS=
```

```
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        0

    PAR_NAME=           function
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_MAX_REPETITION= 999

    PAR_NAME=           literal
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES
REPLY_FORMAT = A
REPLY_PARAMETERS=
    PAR_NAME=           expStatus
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0

    PAR_NAME=           funcStatus
    PAR_TYPE=           STRING
    PAR_DEF_VAL=        "DET.MODE 1 DET.STATE 0 DET.SHUT 0 DET.TELE 0 DET.TEMP 0"
DISPLAY_FORMAT= "Exposure status %d. Function(s) status %s"
HELP_TEXT=
Get the status of the functions in the list of arguments.
Parameter function; currently implemented:
                    DET.MODE
                    DET.STATE
                    DET.SHUT
NOT IMPLEMENTED YET: The last parameter <literal> specifies the format of the enu-
merated types:
omitted : numeric [default]
set     : literal [numerical values coded as strings]
@

//============================================================================
// Commands defined only by CCD sw
//============================================================================

//============================================================================

COMMAND= BIAS
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           at
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "now"
REPLY_FORMAT= A
REPLY_PARAMETERS=
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0
HELP_TEXT =
Start a bias exposure using the actual setup with overloaded keywords
at given optional time and save in LCU memory as Bias image.
```

```
The current setup is overloaded with the setup 'ccdCmdBias.det':
    DET.EXP.TYPE         "Dark";    # Exposure type is DARK/BIAS
    DET.EXP.NREP         1;         # Single exposure
    DET.FRAM.TYPE        "Bias";    # Frame type is BIAS
    DET.WIN1.UIT1            0.000; # Integration time is ZERO
    DET.WIN1.BIAS        F;         # No bias correction
    DET.WIN1.FLATF       F;         # No flat field correction
    DET.WIN1.MINMAX      F;         # No search for extrema
    DET.WIN1.IPFUNC      "None";    # No User-Defined IP function
    DET.WIN1.IPBUFF      "None";    # No User-Defined data buffer
    DET.WIN1.CENTROID    "none";    # No centroiding calculation
    DET.WIN2.BIAS        F;         # No bias correction
    DET.WIN2.FLATF       F;         # No flat field correction
    DET.WIN2.MINMAX      F;         # No search for extrema
    DET.WIN2.IPFUNC      "None";    # No User-Defined IP function
    DET.WIN2.IPBUFF      "None";    # No User-Defined data buffer
    DET.WIN2.CENTROID    "none";    # No centroiding calculation

The time parameter <at> shall comply ISO8601, ie YYYY-MM-DDThh:mm:ss.uuu
One reply only when exposure started

@

//=====================================================================

COMMAND=  DISPLAY
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           at
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "now"
REPLY_FORMAT = A
REPLY_PARAMETERS=
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0
HELP_TEXT=
Start an exposure using the actual setup with overloaded keywords and
display the image on main RTD frame (frame ID 0).

The current setup is overloaded with the setup 'ccdCmdFlat.det':
    DET.EXP.NREP         1;         # Single exposure
    DET.FRAM.FITSMTD     0;         # Image data is not saved on disk
    DET.DISPLAY          0;         # Image data is display on RTD main frame

The time parameter <at> shall comply ISO8601, ie YYYY-MM-DDThh:mm:ss.uuu
One reply only when exposure started

----> Currently parameter "at" ignored. Assumed always "now"  <----
@

//=====================================================================

COMMAND=  ENHANCE
SYNONYMS=
FORMAT= A
```

```
PARAMETERS=
REPLY_FORMAT = A
HELP_TEXT=
Do a bias subtraction and flat field division to last acquired image

----> WS simulation only ! <----
@

//=============================================================================

COMMAND= FLAT
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           at
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "now"
REPLY_FORMAT= A
REPLY_PARAMETERS=
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0
HELP_TEXT =
Start one normal exposure using the actual setup with overloaded keywords
and save it in LCU memory as Flat Field image.

The current setup is overloaded with the setup 'ccdCmdFlat.det':
    DET.EXP.TYPE         "Normal";  # Exposure type is Normal
    DET.EXP.NREP         1;         # Single exposure
    DET.FRAM.TYPE        "FF";      # Frame type is FLAT
    DET.WIN1.BIAS        F;         # No bias correction
    DET.WIN1.FLATF       F;         # No flat field correction
    DET.WIN1.MINMAX      F;         # No search for extrema
    DET.WIN1.IPFUNC      "None";    # No User-Defined IP function
    DET.WIN1.IPBUFF      "None";    # No User-Defined data buffer
    DET.WIN1.CENTROID    "none";    # No centroiding calculation
    DET.WIN2.BIAS        F;         # No bias correction
    DET.WIN2.FLATF       F;         # No flat field correction
    DET.WIN2.MINMAX      F;         # No search for extrema
    DET.WIN2.IPFUNC      "None";    # No User-Defined IP function
    DET.WIN2.IPBUFF      "None";    # No User-Defined data buffer
    DET.WIN2.CENTROID    "none";    # No centroiding calculation

The time parameter <at> shall comply ISO8601, ie YYYY-MM-DDThh:mm:ss.uuu
One reply only when exposure started

@

//=============================================================================

COMMAND= GRAB
SYNONYMS= GrabImage
FORMAT= A
PARAMETERS=
    PAR_NAME=           frameId
    PAR_TYPE=           INTEGER
    PAR_OPTIONAL=       YES
```

```
    PAR_DEF_VAL=        0

    PAR_NAME=           file
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "grab.fits"
REPLY_FORMAT= A
HELP_TEXT =
Save the image currently being displayed on disk file in FITS format.
One reply only at end of operation
@

//============================================================================

COMMAND= STANDAL
SYNONYMS= StandAlone
FORMAT= A
PARAMETERS=
    PAR_NAME=           on
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES

    PAR_NAME=           archive
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES
REPLY_FORMAT= A
HELP_TEXT =
It sets a flag, indicating if the CCD camera is used as stand-alone instrument
or not (default not). If option 'on' is selected, FITS HIERARCH info
is not written in a separate file, but in the image file.
If option 'archive' is selected, Archive is informed when a new file is
completed.
One reply only when commands completed.
@

//============================================================================

COMMAND= STARTAG
SYNONYMS= autoguide
FORMAT= A
PARAMETERS=
    PAR_NAME=           at
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "now"
REPLY_FORMAT= A
HELP_TEXT =
Start an infinite loop of repeated exposures using the actual setup
with overloaded keywords at given optional time.

The current setup is overloaded with the setup 'ccdCmdStartAg.det':
    DET.EXP.TYPE         "Normal";  # Exposure type is Normal
    DET.EXP.NREP          0;        # Infinite loop of exposures
    DET.FRAM.FITSMTD      0;        # Image data is not saved on disk
    DET.WIN1.CENTROID  "threshold"; # Centroiding calculation method
    DET.WIN1.BACKGND     -1;        # Estimate sky background on window corners
    DET.WIN1.THRMIN      -3;        # Threshold at 3 sigma of sky
    DET.FRAM.XFERSYN      F;        # Asynchronous image transfer
```

```
The time parameter <at> shall comply ISO8601, ie YYYY-MM-DDThh:mm:ss.uuu

An infinite loop is started, consisting of:
- Exposure using the actual setup with overloaded keywords
- Computation of error vector
- Storage of results in the LCU database.
One reply only when exposure started

@

//============================================================================

COMMAND= STARTLP
SYNONYMS= startloop
FORMAT= A
PARAMETERS=
    PAR_NAME=           at
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
    PAR_DEF_VAL=        "now"
REPLY_FORMAT= A
HELP_TEXT =
Start an infinite loop of repeated exposures using the actual setup
with overloaded keywords at given optional time.

The current setup is overloaded with the setup 'ccdCmdStartLp.det':
    DET.EXP.NREP         0;         # Infinite loop of exposures
    DET.FRAM.FITSMTD     0;         # Image data is not saved on disk

The time parameter <at> shall comply ISO8601, ie YYYY-MM-DDThh:mm:ss.uuu

An infinite loop is started, consisting of:
- Exposure using the actual setup with overloaded keywords
One reply only when exposure started

@

//============================================================================

COMMAND = STARTWP
SYNONYMS = startwipe
FORMAT =                A
PARAMETERS =
   PAR_NAME = periodic
   PAR_UNIT =
   PAR_TYPE = INTEGER
   PAR_RANGE =
   PAR_DEF_VAL = 0
REPLY_FORMAT =          A
HELP_TEXT =
Wipe chip(s) once or periodically. On each wipe cycle the chips are
wiped as often as specified in DB attribute readout.nWipes without delay
in between. In case of periodic wiping the period is defined in DB attribute
readout.wipePeriod.
Parameter periodic: 1 = periodic wipe; 0 = one-shot only
Two replies: one as soon first wipe has started, the other when
it has finished.
```

```
@

//===========================================================================

COMMAND = STOP
SYNONYMS =
FORMAT =                A
PARAMETERS =
REPLY_FORMAT =          A
HELP_TEXT =
Stop in an orderly way any on-going activity.
This command has little effect on single exposures.
For exposure loop, the STOP takes effect at completion of the last
started exposure. Thus, when STOP returns the last reply, it does not imply
that the loop is completed. This check is achieved with the WAIT command.
The recommended sequence is STOP + WAIT 0,Global (+ END)

One reply only at end of execution
@

//===========================================================================

COMMAND = STOPWP
SYNONYMS = stopwipe
FORMAT =                A
PARAMETERS =
REPLY_FORMAT =          A
HELP_TEXT =
Stop in an orderly way (let last iteration complete) a periodical wipe
One reply only at end of execution
@

//===========================================================================

COMMAND = WAIT
SYNONYMS =
FORMAT =                A
PARAMETERS =
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0

    PAR_NAME=           waitMode
    PAR_TYPE=           STRING
    PAR_DEF_VAL=        "Single"

    PAR_NAME=           literal
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES
REPLY_FORMAT =          A
REPLY_PARAMETERS=
    PAR_NAME=           expStatus
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0
// Two replies, one at beginning, the other at end of execution
HELP_TEXT =
Wait for exposure completion and returns exposure status
```

```
Parameter expoId indicates the ID of exposure to wait for (see START).
0 means: last started exposure.

Parameter waitMode can have values:
"Single": wait until the current repetition is completed
"Global": wait until all repetitions are completed.
It makes sense only if the setup parameter DET.EXP.NREP is > 1.

NOT IMPLEMETED YET: The last parameter <literal> specifies the format of the reply:
omitted: the reply is numeric [default]
set    : the reply is literal [numerical value coded as string]
@

//============================================================================

COMMAND = STPWAIT
SYNONYMS = stpwait
FORMAT =                A
PARAMETERS =
    PAR_NAME=           expoId
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0

    PAR_NAME=           literal
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES
REPLY_FORMAT =          A
REPLY_PARAMETERS=
    PAR_NAME=           expStatus
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0
// Two replies, one at beginning, the other at end of execution
HELP_TEXT =
Stop exposure loop & Wait for exposure completion and returns exposure status.
This command has little effect on single exposures.
For exposure loop, the STPWAIT takes effect at completion of the last
started exposure. STPWAIT returns the last reply when the last exposure is
completed. Equivalent to the sequence STOP + WAIT 0,Global

Parameter expoId indicates the ID of exposure to wait for (see START).
0 means: last started exposure.

NOT IMPLEMETED YET: The last parameter <literal> specifies the format of the reply:
omitted: the reply is numeric [default]
set    : the reply is literal [numerical value coded as string]
@

//============================================================================

COMMAND= STARTTL
SYNONYMS = telemon
FORMAT =                A
PARAMETERS =
  PAR_NAME = period
  PAR_TYPE = INTEGER
  PAR_DEF_VAL = 0
REPLY_FORMAT =          A
HELP_TEXT =
```

```
Start monitoring of telemetry values
One reply only at end of execution

@

//=============================================================================

COMMAND= STOPTL
SYNONYMS = telstop
FORMAT =                A
PARAMETERS =
REPLY_FORMAT =          A
HELP_TEXT =
Stop monitoring of telemetry values
One reply only at end of execution

@

//=============================================================================

COMMAND= STARTTM
SYNONYMS = tempmon
FORMAT =                A
PARAMETERS =
  PAR_NAME = period
  PAR_TYPE = INTEGER
  PAR_DEF_VAL = 0
REPLY_FORMAT =          A
HELP_TEXT =
Start monitoring of temperature values
One reply only at end of execution

@

//=============================================================================

COMMAND= STOPTM
SYNONYMS = tmpstop
FORMAT =                A
PARAMETERS =
REPLY_FORMAT =          A
HELP_TEXT =
Stop monitoring of temperature values
One reply only at end of execution

@

//=============================================================================

MAINTENANCE_COMMANDS

//=============================================================================
// Standard application commands (see CCS User Manual)
//=============================================================================

//=============================================================================

COMMAND= BREAK
```

```
SYNONYMS=
FORMAT= A
PARAMETERS=
REPLY_FORMAT= A
HELP_TEXT=
Break execution of current command.
@

//============================================================================

COMMAND= KILL
SYNONYMS=
FORMAT= A
PARAMETERS=
REPLY_FORMAT= A
HELP_TEXT=
Kill this task
@

//============================================================================
// Standard INS commands for DCS (see INS common sw 2.0 04/05/95)
//============================================================================

//=====================================================================

COMMAND=  DISABLE
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           function
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
//AL950804 Not allowed with PAR_MAX_REPETITION
//   PAR_DEF_VAL=        "all"
    PAR_MAX_REPETITION= 999
REPLY_FORMAT = A
HELP_TEXT=
Disable access to one or more CCD functions (e.g. shutter)
One reply only at end of execution

----> WS simulation only ! <----
@

//=====================================================================

COMMAND=  ENABLE
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           function
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
//    PAR_DEF_VAL=        "all"
    PAR_MAX_REPETITION= 999
REPLY_FORMAT = A
// One reply only at end of execution
HELP_TEXT=
Enables the access to the functions contained in the list of arguments.
```

```
----> WS simulation only ! <----
@

//=====================================================================

COMMAND=  SIM
SYNONYMS= simulate, simulation
FORMAT= A
PARAMETERS=
    PAR_NAME=           function
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
//    PAR_DEF_VAL=        "all"
    PAR_MAX_REPETITION= 999
REPLY_FORMAT = A
HELP_TEXT=
Put the functions contained in the list of arguments into simulation mode.
One reply only at end of execution

----> Not implemented yet <----
@

//=====================================================================

COMMAND=  STOPSIM
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           function
    PAR_TYPE=           STRING
    PAR_OPTIONAL=       YES
//    PAR_DEF_VAL=        "all"
    PAR_MAX_REPETITION= 999
REPLY_FORMAT = A
HELP_TEXT=
Stop the simulation for the functions contained in the list of arguments.
One reply only at end of execution

----> Not implemented yet <----
@

//=====================================================================

COMMAND=  VERBOSE
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           on
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES

    PAR_NAME=           off
    PAR_TYPE=           LOGICAL
    PAR_OPTIONAL=       YES
REPLY_FORMAT = A
HELP_TEXT=
Set verbose mode on/off. If no parameter passed, return current verbose mode
```

```
One reply only at end of execution

----> WS simulation only ! <----
@

//============================================================================
// Commands defined only by CCD sw
//============================================================================

//============================================================================

COMMAND= PROCESS
SYNONYMS=
FORMAT= A
REPLY_FORMAT= A
HELP_TEXT =
Make pre-processing of last image read.
One reply only at end of execution

----> WS simulation only ! <----
@

//============================================================================
//============================================================================

COMMAND= VERSION
SYNONYMS=
FORMAT= A
REPLY_FORMAT= A
REPLY_PARAMETERS=
    PAR_NAME=           ccdVersion
    PAR_TYPE=           STRING
    PAR_DEF_VAL=        "CCD SW simulated"
HELP_TEXT =
Return current version of the CCD sw
One reply only at end of execution
@

//============================================================================




TEST_COMMANDS

//============================================================================
// Commands defined only by CCD sw
//============================================================================

//============================================================================

COMMAND= CONFIG
SYNONYMS= configure
FORMAT= A
REPLY_FORMAT= A
// One reply only at end of execution
HELP_TEXT =
Read again configuration of the system: a change has been done
```

```
----> Not implemented yet <----
@

//==============================================================================

COMMAND= DEBUG
SYNONYMS=
FORMAT= A
PARAMETERS=
    PAR_NAME=           level
    PAR_TYPE=           INTEGER
    PAR_DEF_VAL=        0
REPLY_FORMAT= A
HELP_TEXT =
Define debugging level.
One reply only at end of execution
@

//==============================================================================

//
// _oOo_
```

3.3 Scripts

The scripts available on Workstation are:

1. **ccdInstall.sh.** Installation of files needed by the CCD software for the camera used.
2. **ccdDcsStart.sh**. Cold Startup script.
3. **ccdDcsWarmStart.sh**. Warm Startup script.
4. **ccdDcsStop.sh**. Shutdown script.
5. **ccdDcsDbSave.sh**. Save current CCD configuration.
6. **ccdDcsScan.sh**. Define scan table for CCD DCS part
7. **ccdosScan.sh**. Define scan table for CCD stand-alone part.
8. **ccdMidasBgStart.sh**. Start a Midas background session.
9. **ccdMidasBgStop.sh**. Stop a Midas background session.
10. **ccdDcsWcs.sh**. Set values in the CCD OLDB for World Coordinates display.
11. **ccdDcsSetLogMask.sh**. Set source mask for FITS logs.

3.3.1 ccdInstall.sh(1)

NAME

```
ccdInstall.sh - Install CCD files in INS_ROOT
```

SYNOPSIS

```
ccdInstall.sh <DbFile> [<ins_root_dir>]
```

DESCRIPTION

```
This shell script installs all files needed to run a CCD system in the
root directory for the instrument it belongs to. Additionally, it
stores in the on-line database the values contained in configuration
files.

<DbFile> Name of the .dbcfg file in $VLTROOT/config containing DB
         init values.
<ins_root_dir> root directory for the instrument the CCD belongs to
               Default: $INS_ROOT (env. variable)
```

FILES

```
$VLTROOT/config/ccd*.dbcfg
Files containing on-line database initialisation values for CCD systems
known.
For technical CCDs there is one file for each system available. They
normally do not need to be changed by the user. The name is
ccdTec<nn>.dbcfg whereby <nn> is the number of camera head.
Example: head number 21 ---> ccdTec21.dbcfg
For scientific CCDs there is one file for each system to be delivered.
They normally have to be checked and possibly modified by experts of
the specific CCD system. The name is ccdSci<id>.dbcfg whereby <id> is
a symbolic name of the system.
Example: Fors CCD ---> ccdSciFors.dbcfg
```

ENVIRONMENT

```
INTROOT  integration root directory
CCDNAME  CCD camera name
RTAPENV  CCD Rtap environment name
INS_ROOT default instrument root directory
```

RETURN VALUES

```
0 if SUCCESS
1 if FAILURE
```

EXAMPLES

```
> ccdInstall.sh ccdTec21.dbcfg $INS_ROOT
Install all what needed for technical CCD head 21
> ccdInstall.sh ccdSciFors.dbcfg
Install all what needed for Fors scientific CCD
```

SEE ALSO

```
INS common sw specification
```

```
- - - - - -
Last change:  08/10/97-13:52
```

3.3.2 ccdDcsStart.sh(1)

NAME

```
ccdDcsStart.sh - startup of CCD DCS
```

SYNOPSIS

```
ccdDcsStart.sh [<camera>] [<WS env.>] [<LCU env.>] [<INS root>] [kill]
```

DESCRIPTION

```
This shell script performs a startup of CCD DCS.
No CCD stand-alone module is started.
To be run once at startup only!
It performs the following steps:
  1 - Load database configuration values on WS
  2 - Run warm startup (ccdDcsWarmStart.sh)

<camera>    camera name, also root point in DB (default env. var. CCDNAME)
<WS env.>   name of workstation environment (default env. var. RTAPENV)
            If value is 0, no action on WS part of CCD sw is taken
<LCU env.>  name of LCU environment (default env. var. CCDLENV)
            If value is 0, no action on LCU part of CCD sw is taken
<INS root>  INS_ROOT environment variable
kill        kill all already running processes before starting
```

ENVIRONMENT

```
INTROOT  integration root directory
CCDNAME  default for camera name (e.g. ccdFors)
RTAPENV  default for WS local environment (e.g. wte13)
CCDLENV  default for LCU environment (e.g lte25)
INS_ROOT default root directory for instrument data
```

RETURN VALUES

```
0 if SUCCESS
1 if FAILURE
```

CAUTIONS

```
If this script is run twice or more, the system may not behave properly
any more. In this case better to shutdown the whole system and restart it
again.
```

EXAMPLES

```
> ccdDcsStart.sh ccdFors wte13 lte25
      Start the CCD DCS sw both at WS and LCU level
      for camera "ccdFors", WS environment wte13, LCU environment lte25

> ccdDcsStart.sh ccdFors wte13 0
      Start the CCD DCS sw at WS level only
      for camera "ccdFors", WS environment wte13

> ccdDcsStart.sh ccdFors wte13 0 $INS_ROOT kill
      Kill and restart the CCD DCS sw at WS level only
      for camera "ccdFors", WS environment wte13

> ccdDcsStart.sh ccdFors 0 lte25
      Start the CCD DCS sw at LCU level only
      for camera "ccdFors", LCU environment lte25
```

SEE ALSO

```
ccdDcsStop.sh ccdDcsWarmStart.sh
```

```
- - - - - -
Last change:  21/09/98-11:41
```

3.3.3 ccdDcsStop.sh(1)

NAME

```
ccdDcsStop.sh - shut-down CCD DCS
```

SYNOPSIS

```
ccdDcsStop.sh [<camera>] [<WS env.>] [<LCU env.>] [kill]
```

DESCRIPTION

```
This shell script performs a shut-down of CCD DCS.
If doess the following steps:
  1 - Verify if the main process is running.
  2 - Try to terminate CCD processes in a 'soft' way (command EXIT)
  3 - Try to terminate CCD processes in a 'hard' way (command KILL) (opt.)
  4 - Disable scanning of data from LCU to WS

<camera>   camera name, also root point in DB (default env. var. CCDNAME)
<WS env.>  name of workstation environment (default env. var. RTAPENV).
           If value is 0, no action on WS part of CCD sw is taken
<LCU env.> name of LCU environment (default env. var. CCDLENV)
           If value is 0, no action on LCU part of CCD sw is taken
kill       kill all processes
```

ENVIRONMENT

```
CCDNAME  default for camera name (e.g. ccdFors)
RTAPENV  default for WS local environment (e.g. wte13)
CCDLENV  default for LCU environment (e.g lte25)
```

RETURN VALUES

```
0 if SUCCESS
1 if FAILURE
```

CAUTIONS

```
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

```
> ccdDcsStop.sh ccdFors wte13 lte25
      Terminate in a 'soft' way the CCD sw both at WS and LCU level
      for camera "ccdFors", WS environment wte13, LCU environment lte25

> ccdDcsStop.sh ccdFors wte13 lte25 kill
      Terminate in a 'hard' way the CCD sw both at WS and LCU level
      for camera "ccdFors", WS environment wte13, LCU environment lte25

> ccdDcsStop.sh ccdFors wte13 0 kill
      Terminate in a 'hard' way the CCD sw at WS level only
      for camera "ccdFors", WS environment wte13

> ccdDcsStop.sh ccdFors 0 lte25
      Terminate in a 'soft' way the CCD sw at LCU level only
      for camera "ccdFors", LCU environment lte25
```

SEE ALSO

```
ccdDcsStart.sh, ccdDcsKill.sh
```

```
- - - - - -
Last change:  21/09/98-11:41
```

3.3.4 ccdDcsDbSave.sh(l)

NAME

```
ccdDcsDbSave.sh - Save CCD database values into backup file
```

SYNOPSIS

```
ccdDcsDbSave.sh [<camera>] [<env>] [<file>]
```

DESCRIPTION

```
This shell script reads some values from the CCD WS database and
stores them into <file> using the utility dbBackup

<camera> camera name, also root point in DB (default env. var. CCDNAME)
<env>    environment where to read from (default env. var. RTAPENV)
<file>   name of backup file
         (default INS_ROOT/SYSTEM/COMMON/CONFIGFILES/<camera>.dbcfg)
```

FILES

```
ccdConfig.inp     input file for utility dbBackup
```

ENVIRONMENT

```
CCDNAME   default for camera name (e.g. ccdFors)
RTAPENV   default environment (e.g.wte16)
INS_ROOT  default for instrument root directory
```

RETURN VALUES

```
0 if SUCCESS
1 if FAILURE
```

EXAMPLES

```
ccdDcsDbSave.sh ccdFors wte16 $INS_ROOT/SYSTEM/COMMON/CONFIGFILES/ccdFors.dbcfg
```

SEE ALSO

```
dbBackup
```

```
- - - - - -
Last change:  21/09/98-11:41
```

3.3.5 ccdDcsScan.sh(1)

NAME

```
ccdDcsScan.sh - configure the DB values to be scanned from LCU for CCD DCS
```

SYNOPSIS

```
ccdDcsScan.sh [<camera>] [<WS env.>] [<LCU env.>] [<option>]
```

DESCRIPTION

```
This shell script configures the Scan System to retrieve DB values
for the CCD LCU to WS.

<camera>   camera name, also root point in DB (default env. var. CCDNAME)
<WS env.>  name of workstation environment (default env. var. RTAPENV)
<LCU env.> name of LCU environment (default env. var. CCDLENV)
<option>   a    add entries to scan list
           c    clean entries from scan list
           d    disable scanning from the LCU
           e    enable scanning from the LCU
           default: a
```

ENVIRONMENT

```
CCDNAME  default for camera name (e.g. ccdFors)
RTAPENV  default for WS local environment (e.g. wte13)
CCDLENV  default for LCU environment (e.g lte25)
```

RETURN VALUES

```
0 if SUCCESS
1 if FAILURE
```

CAUTIONS

```
1 - It is assumed here that CCD branch in the LCU database is attached
    directly to the root point (not a sub-point of something else).
2 - If this script is run twice or more, the system may not behave properly
    any more. In this case better to shutdown the whole system and restart
    it again.
```

EXAMPLES

```
ccdDcsScan.sh ccdFors wte13 lte25
     Add entries for LCU lte25 and WS wte13 camera ccdFors
ccdDcsScan.sh ccdFors wte13 lte25 c
     Clean entries for LCU lte25 and WS wte13 camera ccdFors
ccdDcsScan.sh 0 wte13 lte25 d
     Disable scanning from LCU lte25 to WS wte13
ccdDcsScan.sh 0 wte13 lte25 e
     Enable scanning from LCU lte25 to WS wte13
```

```
- - - - - -
Last change:  21/09/98-11:41
```

3.3.6 ccdosScan.sh(1)

NAME

```
ccdosScan.sh - configure the DB values to be scanned from LCU for CCD OS
```

SYNOPSIS

```
ccdosScan.sh [<camera>] [<WS env.>] [<LCU env.>] [<option>]
```

DESCRIPTION

```
This shell script configures the Scan System to retrieve DB values
from the CCD LCU to WS (stand-alone part only).

<camera>   camera name, also root point in DB (default env. var. CCDNAME)
<WS env.>  name of workstation environment (default env. var. RTAPENV)
<LCU env.> name of LCU environment (default env. var. CCDLENV)
<option>   a    add entries to scan list
           c    clean entries from scan list
           d    disable scanning from the LCU
           e    enable scanning from the LCU
           default: a
```

ENVIRONMENT

```
CCDNAME  default for camera name (e.g. ccdFors)
RTAPENV  default for WS local environment (e.g. wte13)
CCDLENV  default for LCU environment (e.g lte25)
```

RETURN VALUES

```
0 if SUCCESS
1 if FAILURE
```

CAUTIONS

```
1 - It is assumed here that CCD branch in the LCU database is attached
    directly to the root point (not a sub-point of something else).
2 - If this script is run twice or more, the system may not behave properly
    any more. In this case better to shutdown the whole system and restart
    it again.
```

EXAMPLES

```
ccdosScan.sh ccdFors wte13 lte25
     Add entries for LCU lte25 and WS wte13 camera ccdFors
ccdosScan.sh ccdFors wte13 lte25 c
     Clean entries for LCU lte25 and WS wte13 camera ccdFors
ccdosScan.sh ccdFors wte13 lte25 e
     Enable entries for LCU lte25 and WS wte13 camera ccdFors
```

SEE ALSO

```
ccdDcsScan.sh
```

```
- - - - - -
Last change:  21/09/98-11:45
```

3.3.7 ccdMidasBgStart.sh(1)

NAME

```
ccdMidasBgStart.sh - start a Midas background session and pco
```

SYNOPSIS

```
ccdMidasBgStart.sh [<unit ID>]
```

DESCRIPTION

```
This shell script starts a Midas session in background mode
and the VLT pco program to communicate with it.
<unit ID> is the Midas unit (00 to 99) to be started. Default : 50
```

ENVIRONMENT

```
RTAPENV current local environment used
HOST    name of host computer
```

EXAMPLES

```
> ccdMidasBgStart.sh
      Start pco and Background Midas session with default unit 50.

> ccdMidasBgStart.sh 51
      Start pco and Background Midas session with unit 51.
```

SEE ALSO

```
ccdMidasBgStop.sh
```

```
- - - - - -
Last change:  08/10/97-13:47
```

3.3.8 ccdMidasBgStop.sh(1)

NAME

```
ccdMidasBgStop.sh - stop a Midas background session
```

SYNOPSIS

```
ccdMidasBgStop.sh [<unit ID>]
```

DESCRIPTION

```
This shell script stops a Midas session running in background mode.
<unit ID> is the Midas unit (00 to 99) to be stopped. Default : 50
```

ENVIRONMENT

```
HOST    name of host computer
```

EXAMPLES

```
> ccdMidasBgStop.sh
      Stop Background Midas session default unit 50.

> ccdMidasBgStopt.sh 51
      Stop Background Midas session unit 51.
```

SEE ALSO

```
ccdMidasBgStart.sh
```

BUGS

```
Currently there are problems with resource releasing when one tries
to terminate a Midas background session (SPR 960690).
This script just brings back the background MIdas session to
foreground.
```

```
- - - - - -
Last change:  08/10/97-13:47
```

3.3.9 ccdDcsWcs.sh(1)

NAME

```
ccdDcsWcs.sh - Set World Coordinate values in OLDB for CCD camera
```

SYNOPSIS

```
ccdDcsWcs.sh <camera> <WS env.> <xrefpix> <yrefpix>
             <secpix> <rotate> <equinox> <epoch> <proj>
```

DESCRIPTION

```
This shell script writes in the WS CCD OLDB the values of attributes
related to the display of World Coordinates with rtd.
It is meant as a utility to applications having to configure the
WCS parameters for the camera used.

<camera>     camera name, also root point in DB
<WS env.>    name of workstation environment
<xrefpix>    X Coordinate of Reference Pixel
<yrefpix>    Y Coordinate of Reference Pixel
<secpix>     Number of arcseconds per pixel
<rotate>     Rotation angle (clockwise positive) in degrees
<equinox>    Equinox of coordinates, 1950 and 2000 supported
<epoch>      Epoch of coordinates, used for FK4/FK5 conversion
<proj>       Projection
```

RETURN VALUES

```
0 if SUCCESS
1 if FAILURE
```

EXAMPLES

```
> ccdDcsWcs.sh ccdTEC wtccds 200.0 145.0 0.3 0.0 1950 0.0 PIXEL
      Reference pixel [200.0;145.0]
      Each pixel covers 0.3 arcseconds
      Rotation angle is 0.0 degrees
      Equinox of ccordinates set to 1950
      Epoch of ccordinates set to 0.0
      Projection set to PIXEL
```

SEE ALSO

```
rtdImageEvent.h for more information about parameters meaning and
                possible values
```

```
- - - - - -
Last change:  21/09/98-11:41
```

3.3.10 ccdDcsSetLogMask.sh(1)

NAME

```
ccdDcsSetLogMask.sh - Set FITS logging mask in OLDB for CCD camera
```

SYNOPSIS

```
ccdDcsSetLogMask.sh <camera> <WS env.> <logMask>
```

DESCRIPTION

```
This shell script writes in the WS CCD OLDB the value of the attribute
related to the FITS logging of main info.
An update of the $INS_ROOT/SYSTEM/COMMON/CONFIGFILES/<camera>.dbcfg file
is performed. As well a snap of the WS CCD OLDB is made.
It is meant as a utility to applications having to configure the
logMask parameter for the camera used.

<camera>     camera name, also root point in DB
<WS env.>    name of workstation environment
<logMask>    Mask for FITS logging
```

RETURN VALUES

```
0 if SUCCESS
1 if FAILURE
```

EXAMPLES

```
> ccdDcsSetLogMask.sh ccdAGA wut1tcs UT1AGA
      FITS Logs [UT1AGA]
```

```
- - - - - -
Last change:  21/09/98-11:41
```

3.4 GUI classes

The following GUI classes, built with the VLT panel editor (see [22]) for being used within bigger application panels, are included in the library libccdGuiPublic.tcl:

1. **ccdExpStatus_uifClass.** It displays the status of an exposure.
2. **ccdIpStatus_uifClass.** It displays the status of the real-time image processing on LCU.
3. **ccdExpSetup_uifClass.** It displays the main setup definitions of the running exposure.
4. **ccdReadoutSetup_uifClass.** It displays the readout setup definitions of the running exposure

3.4.1 ccdExpStatus_uifClass(1)

NAME

```
ccdExpStatus_uifClass - GUI class for CCD exposure status display

Application panels who intend to use this class, must register to the
library libccdGuiPublic.tcl.
All database attributes defined within this class are relative to the
root point of the camera used. Therefore, when importing an instance of
this class in a panel, the Database Current Working Point must be set
accordingly. For example, if the camera used is called "myccd", then
the CWP has to be <alias>myccd.

Note 1: The class shows the contents of the Workstation on-line database.
        The CCS Scan System must be enabled and working in order to
        retrieve the correct values from the CCD LCU.

Note 2: The class works only with exposures under the point
        exposures:exposure_1
```

APPLICATION AREA

```
This class displays only output fields.
In the order from top to button, left to right:

"Exposure status"
    String showing the current exposure status (updated on change).

"Remaining time (sec)"
    Remaining integration time (sec) (updated on polling).

"Loop count"
    Current repetition of the defined exposure (updated on polling).
```

BUGS

```
The attributes updated with a polling mechanism have a fixed polling rate.
The change of values, although faster in the database, might be shown on
the panel with some delay (up to 2 seconds).
```

```
- - - - - -
Last change:  08/10/97-14:56
```

3.4.2 ccdIpStatus_uifClass(1)

NAME

```
ccdIpStatus_uifClass - GUI class for CCD image processing status display

Application panels who intend to use this class, must register to the
library libccdGuiPublic.tcl.
All database attributes defined within this class are relative to the
point images:process relative to the root point of the camera used.
Therefore, when importing an instance of this class in a panel, the
Database Current Working Point must be set accordingly.
For example, if the camera used is called "myccd", then
the CWP has to be <alias>myccd:images:process.

Note: The class shows the contents of the Workstation on-line database.
      The CCS Scan System must be enabled and working in order to
      retrieve the correct values from the CCD LCU.
```

APPLICATION AREA

```
This class displays only output fields.
In the order from top to button, left to right:

"Min"
    Minimum pixel value over the image (updated on polling).

"Max"
    Maximum pixel value over the image (updated on polling).

"Average"
    Average pixel value over the image (updated on polling).

"Sigma"
    Standard deviation of pixel values over the image (updated on polling).

"# of pixels"
    Number of pixels above threshold (updated on polling).

"X-Err"
    X component of the error vector computed by the centroiding
    algorythm on LCU (updated on polling).

"Y-Err"
    Y component of the error vector computed by the centroiding
    algorythm on LCU (updated on polling).

"Intensity"
    Intensity of the pixel closest to the centroid position computed
    on LCU. If set to 0, it indicates that no object has been detected.
    (updated on polling).
```

BUGS

```
The attributes updated with a polling mechanism have a fixed polling rate.
The change of values, although faster in the database, might be shown on
the panel with some delay (up to 2 seconds).
```

```
- - - - - -
Last change:  08/10/97-13:55
```

3.4.3 ccdExpSetup_uifClass(1)

NAME

```
ccdExpSetup_uifClass - GUI class for CCD exposure setup display

Application panels who intend to use this class, must register to the
library libccdGuiPublic.tcl.
All database attributes defined within this class are relative to the
root point of the camera used.
Therefore, when importing an instance of this class in a panel, the
Database Current Working Point must be set accordingly.
For example, if the camera used is called "myccd", then
the CWP has to be <alias>myccd.

Note: The class shows the contents of the Workstation on-line database.
      The CCS Scan System must be enabled and working in order to
      retrieve the correct values from the CCD LCU.
```

APPLICATION AREA

```
This class displays only output fields.
In the order from top to button, left to right:

"Time(sec)"
    Exposure time in seconds as defined by the user

"Repetitions"
    Number of times the same exposure has to be repeated. A value of 0
    means that the exposure has to be repeated forever until a STOP
    command is issued

"File"
    File where the last image has been saved.
```

```
- - - - - -
Last change:  08/10/97-13:55
```

3.4.4 ccdReadoutSetup_uifClass(1)

NAME

```
ccdReadoutSetup_uifClass - GUI class for CCD readout setup display

Application panels who intend to use this class, must register to the
library libccdGuiPublic.tcl.
All database attributes defined within this class are relative to the
root point of the camera used.
Therefore, when importing an instance of this class in a panel, the
Database Current Working Point must be set accordingly.
For example, if the camera used is called "myccd", then
the CWP has to be <alias>myccd.

Note: The class shows the contents of the Workstation on-line database.
      The CCS Scan System must be enabled and working in order to
      retrieve the correct values from the CCD LCU.
```

APPLICATION AREA

```
This class displays only output fields.
In the order from top to button, left to right:

"Speed"
    Readout speed as defined by the user

"Binning"
    "X"
    Horizontal binning factor as defined by the user
    "Y"
    Vertical binning factor as defined by the user
"Window"
    "#"
    Window index followed by field indicating if the window is enabled
    (YES) or not (NO)
    "DimX"
    Horizontal window size
    "DimY"
    Vertical window size
    "FirstX"
    Horizontal coordinate of window lower left corner
    "FirstY"
    Vertical coordinate of window lower left corner
```

```
- - - - - -
Last change:  08/10/97-13:55
```

3.5 Include files

All definitions needed by applications using the CCD software are contained in the file ***ccd.h***.
It includes the files ***ccdDbPublic.h***, which contains definitions of public database points/attributes, ***ccdErrors.h***, which contains the CCD error codes, and ***ccdObj.h***, which contains the definitions used by the library *libccdObj.a*.
These files only are considered public and must be used by external applications.

Other include files are installed as well, since they are needed by more than one CCD software module; nevertheless they are considered as private files to the CCD software: external applications are not allowed to use information contained in them, and therefore they are not documented in this section.

3.5.1 ccd.h

```
/*************************************************************************
* E.S.O. - VLT project
*
* "@(#) $Id: ccd.h,v 2.4 1998/09/18 08:57:18 vltsccm Exp $"
*
* who       when      what
* --------  --------  ----------------------------------------------
* alongino  24/07/95  created
* alongino  17/08/95  ccdCAMERANAME_LEN moved from ccdInternal.h
* alongino  18/08/95  SETUP parameters moved from ccdInternal.h and modified
* alongino  23/08/95  added ccdSETUPRES and ccdCheckSetup
* alongino  24/08/95  added ccdWINDOW and ccdSETUP
*                     modified macro for commands (added CMD in front)
* alongino  31/08/95  added ccdFITSHIERSUFF
* alongino  04/09/95  ccdGetIndexFromId moved from ccdInternal.h
* alongino  29/09/95  ccdDatabase.h replaced with ccdDbPublic.h
* alongino  30/01/96  added definitions for command parameters
* pduhoux   07/03/96  removed 'const' from prototypes
* pduhoux   24/05/96  added WAIT modes
* alongino  10/06/96  moved from module ccdcon to ccd
* alongino  09/07/96  added ccdIP_NO_USER_FUN and ccdIP_NO_USER_BUF
* alongino  12/07/96  added ccdCMD_WAIT
* pduhoux   22/07/96  added doTransfer/doProcess flags in ccdSETUPRES
* alongino  09/10/96  DET.FRAM.TYPE DET.FRAM.BIAS DET.FRAM.FLATF implemented
* pduhoux   11/11/96  added setup parameters to ccdIPIMAGE data structure
* alongino  15/11/96  added ccdObj.h
* alongino  13/12/96  added ccdCMD_STAND_ALONE
* alongino  16/12/96  added parameters for ccdCMD_STAND_ALONE
* pduhoux   17/01/97  added macro ccdMAXLENFILE = 31 (SPR960683)
* pduhoux   12/02/97  added setup macros ccdTIME_START/REP/DEF
* pduhoux   04/03/97  added ccdPROCESS data structure
*                     replaced IP setup by array for multiple window IP
* pduhoux   12/03/97  VERSION_DATE is MAY97
* pduhoux   12/03/97  added cmd STARTTL/STOPTL and STARTTM/STOPTM
*                     moved cmd OFF/ONLINE from ccdInternal.h
* pduhoux   13/03/97  removed setup keywords for former one window processing
* pduhoux   14/03/97  added ccdKEY_TMPSTAT & ccdKEY_TELSTAT for STATUS cmd
* pduhoux   16/04/97  added <gainIndex> in ccdCONFCLOCK
* pduhoux   02/06/97  new data structure ccdIPIMAGE
```

```
*                       enhanced ccdSETUP/ccdPROCESS data structures
* pduhoux   30/06/97  added <timeDefIdem>, <linesToShiftType> and
*                     <linesToShiftMulti[]> in ccdSETUP data structure
*                     added setup macros ccdEXP_STEP_DEF, ccdTIME_DEF_IDEM,
*                     ccdTIME_DEF1 to ccdTIME_DEF10, ccdREAD_SHIFT_TYPE,
*                     ccdREAD_SHIFT1 to ccdREAD_SHIFT10
* pduhoux   30/06/97  added <elAdu[]> in ccdCONFCLOCK
* pduhoux   26/07/97  added ccdTHRMIN_5SIGMA_WINDOW_SELF/PREV
*                     added setup macro ccdREAD_IMAGE_SIM
*                     added field <fileNameSimu> in ccdSETUP
* pduhoux   04/08/97  generalized support to ccdTHRMIN & ccdBACKGND
* pduhoux   16/09/97  removed timeStart from ccdSETUP (unused)
* pduhoux   10/11/97  moved ccdVERSION & ccdVERSION_DATE to
*                     ccdint/include/ccdintDefines.h
* alongino  09/05/98  ccdIPEXPOSURE enhanced with more time info
* alongino  12/05/98  added ccdDO_WIPE and ccdXFER_SYNC
*                     wipeTime and itSync in ccdSETUP
* pduhoux   25/05/98  added some fields in ccdIPEXPOSURE
* pduhoux   12/08/98  added command ccdCMD_STOP_WAIT (STPWAIT)
*                     renamed DET.WINi.ASUIT into DET.WINi.ASUIT1
*/
/************************************************************************
*  This file contains all definitions used by external modules using the
*  CCD sw
************************************************************************/

#ifndef CCD_H
#define CCD_H

#ifdef __cplusplus
extern "C" {
#endif

/*
 * CCS & LCC header files
 */
#include "CCS.h"
#include "cai.h"

/*
 * CCD header files
 */
#include "ccdDbPublic.h"
#include "ccdErrors.h"
#include "ccdObj.h"

/* CCD module name */
#define ccdMODULE       "ccd"

/*
 * Constants
 */
#define ccdENV_LENV       "CCDLENV"   /* name of env. var. for LCU env.      */
#define ccdENV_DID        "CCDDID"    /* name of env. var. for Dictionary    */
#define ccdDID            "CCDDCS"    /* name of default CCD Data Dictionary */
#define ccdCAT            slxDETECTOR /* name of FITS category               */
#define ccdEXP_NEXT       -1          /* ID for next exposure to start       */
#define ccdEXP_LAST       0           /* ID for last started exposure        */
```

```
#define ccdCAMERANAME_LEN 8           /* maximum length of camera name       */
#define ccdMAXLENFILE     31          /* maximum length of FITS file names   */
#define ccdMAXBINX        8           /* maximum binning allowed along x     */
#define ccdMAXBINY        8           /* maximum binning allowed along y     */
#define ccdFITSHIERSUFF   "det"       /* suffix for FITS HIER DET file       */
#define ccdDCSHEADLINES   21      /* lines written by CCD DCS in FITS header */
#define ccdDCSXTNDLINES   22      /* lines written by CCD DCS in FITS header */
                                  /* XTENSION part                           */

/* commands implemented */
#define ccdCMD_ABORT       "ABORT"
#define ccdCMD_BIAS        "BIAS"
#define ccdCMD_CONTINUE    "CONT"
#define ccdCMD_DISPLAY     "DISPLAY"
#define ccdCMD_DUMP        "DUMP"
#define ccdCMD_END         "END"
#define ccdCMD_EXIT        "EXIT"
#define ccdCMD_FLAT        "FLAT"
#define ccdCMD_GRAB        "GRAB"
#define ccdCMD_INIT        "INIT"
#define ccdCMD_OFF         "OFF"
#define ccdCMD_ONLINE      "ONLINE"
#define ccdCMD_OPERATE     "OPERATE"
#define ccdCMD_PAUSE       "PAUSE"
#define ccdCMD_PDOWN       "PDOWN"
#define ccdCMD_SETUP       "SETUP"
#define ccdCMD_STAND_ALONE "STANDAL"
#define ccdCMD_STANDBY     "STANDBY"
#define ccdCMD_START       "START"
#define ccdCMD_STARTAG     "STARTAG"
#define ccdCMD_STARTLP     "STARTLP"
#define ccdCMD_START_WIPE  "STARTWP"
#define ccdCMD_STARTTL     "STARTTL"
#define ccdCMD_STARTTM     "STARTTM"
#define ccdCMD_STATUS      "STATUS"
#define ccdCMD_STOP        "STOP"
#define ccdCMD_STOP_WAIT   "STPWAIT"
#define ccdCMD_STOP_WIPE   "STOPWP"
#define ccdCMD_STOPTL      "STOPTL"
#define ccdCMD_STOPTM      "STOPTM"
#define ccdCMD_VERS        "VERSION"
#define ccdCMD_WAIT        "WAIT"

/* commands implemented in WS simulation only (not supported by LCU yet ) */
#define ccdCMD_DISABLE     "DISABLE"
#define ccdCMD_ENABLE      "ENABLE"
#define ccdCMD_ENHANCE     "ENHANCE"
#define ccdCMD_PROCESS     "PROCESS"
#define ccdCMD_PXQUERY     "PXQUERY"
#define ccdCMD_SELFTST     "SELFTST"
#define ccdCMD_VERBOSE     "VERBOSE"

/* commands not yet implemented */
#define ccdCMD_CONFIG      "CONFIG"
#define ccdCMD_SIM         "SIM"
#define ccdCMD_STOPSIM     "STOPSIM"

/* parameters */
```

```
/* command ccdCMD_STATUS */
#define ccdPAR_STATUS_EXPOID   "expoId"          /* exposure ID parameter name  */
#define ccdPAR_STATUS_FUNCTION "function"        /* function parameter name     */

/* paremeters currently used for ccdPAR_STATUS_FUNCTION */
#define ccdALL                "all"
#define ccdGLOBAL             "global"
#define ccdKEY_OPMODE         "DET.MODE"         /* see also ccdOPMODE          */
#define ccdKEY_OPSTATE        "DET.STATE"        /* see also ccdSTATE           */
#define ccdKEY_SHTSTAT        "DET.SHUT"         /* see also ccdSHT_STATUS      */
#define ccdKEY_TMPSTAT        "DET.TEMP"         /* Temperature values          */
#define ccdKEY_TELSTAT        "DET.TELE"         /* Telemetry values            */

/* reply parameters for ccdCMD_STATUS */
#define ccdPAR_STATUS_EXPSTATUS  "expStatus"     /* exposure status par. name   */
                                                 /* see also ccdEXP_xxx         */
#define ccdPAR_STATUS_FUNCSTATUS "funcStatus"    /* function status par. name   */

/* command ccdCMD_SETUP */
#define ccdPAR_SETUP_EXPOID   "expoId"           /* exposure ID parameter name  */
#define ccdPAR_SETUP_FILE     "file"             /* setup file parameter name   */
#define ccdPAR_SETUP_FUNCTION "function"         /* setup function name         */
#define ccdPAR_SETUP_CHECK    "check"            /* setup check required        */

/* command ccdCMD_START */
#define ccdPAR_START_AT       "at"               /* start at UTC                */

/* paremeters currently used for ccdPAR_SETUP_FUNCTION */
#define ccdEXP_TYPE           "DET.EXP.TYPE"     /* exposure type               */
                                                 /* see also ccdEXPTYPE         */
#define ccdEXP_STEP_DEF       "DET.WIN1.NDIT"    /* Number of steps for MULTI   */

#define ccdTIME_DEF_IDEM      "DET.WIN1.ASUIT1" /* integration time as UIT1    */
#define ccdTIME_DEF_BASE      "DET.WIN1.UIT"
#define ccdTIME_DEFi          "DET.WIN1.UIT%i"   /* integration time            */
#define ccdTIME_DEF           "DET.WIN1.UIT1"    /* integration time single     */

#define ccdREPEAT_DEF         "DET.EXP.NREP"     /* exp. repetition count       */
                                                 /* see also ccdREPEAT_FOREVER  */
#define ccdDO_WIPE            "DET.EXP.WIPETIM"  /* wipe at the begin of exp.   */
#define ccdTIME_REP           "DET.EXP.TIMEREP"  /* Interval between 2 exposures*/
#define ccdFILE_UNC           "DET.FRAM.FITSUNC" /* file for uncompr. image data*/
#define ccdDISK_SAVE          "DET.FRAM.FITSMTD" /* disk save method            */
#define ccdXFER_SYNC          "DET.FRAM.XFERSYN" /* image transfer synchronous  */
                                                 /* see also ccdDISKSAVE        */
#define ccdSAMPLE             "DET.FRAM.SAMPLE"  /* transfer sampling rate      */
#define ccdDISPLAY            "DET.DISPLAY"      /* real-time display frame Id  */
                                                 /* see also ccdNO_DISPLAY      */
#define ccdREAD_CLOCK         "DET.READ.CLKIND"  /* clock index                 */
#define ccdREAD_GAIN          "DET.OUT1.GAININD" /* readout gain                */
#define ccdREAD_FRAME_TYPE    "DET.FRAM.TYPE"    /* frame type                  */
                                                 /* see also ccdFRAME_xxx       */
#define ccdREAD_SHIFT_TYPE    "DET.READ.SHIFTYP" /* line shift type for MULTI  */
#define ccdREAD_SHIFT_BASE    "DET.READ.SHIFT"   /* line shift                  */
#define ccdREAD_SHIFTi        "DET.READ.SHIFT%i" /* line shift step #i          */

#define ccdREAD_IMAGE_SIM     "DET.READ.SIMIMG"  /* image to load for ACE sim   */
```

```
/*===========*/
/* WINDOW #1 */
/*===========*/
#define ccdREAD_WIN1_BIN_X    "DET.WIN1.BINX"    /* readout x binning            */
#define ccdREAD_WIN1_BIN_Y    "DET.WIN1.BINY"    /* readout y binning            */
#define ccdREAD_WIN1_ENAB     "DET.WIN1.ST"      /* enabled flag                 */
#define ccdREAD_WIN1_FIR_X    "DET.WIN1.STRX"    /* first x pixel                */
#define ccdREAD_WIN1_DIM_X    "DET.WIN1.NX"      /* x dimensions                 */
#define ccdREAD_WIN1_FIR_Y    "DET.WIN1.STRY"    /* first y pixel                */
#define ccdREAD_WIN1_DIM_Y    "DET.WIN1.NY"      /* y dimensions                 */
#define ccdPROC_WIN1_MINMAX   "DET.WIN1.MINMAX"  /* search image extrema         */
#define ccdPROC_WIN1_BIAS     "DET.WIN1.BIAS"    /* bias subtraction wanted      */
#define ccdPROC_WIN1_FLAT     "DET.WIN1.FLATF"   /* flat field division wanted   */
#define ccdPROC_WIN1_CENTROID "DET.WIN1.CENTROID"/* type of centroiding          */
                                                 /* see also ccdCENTROID_xxx     */
#define ccdPROC_WIN1_REF_X    "DET.WIN1.REFX"    /* X ref. star for centroiding */
#define ccdPROC_WIN1_REF_Y    "DET.WIN1.REFY"    /* Y ref. star for centroiding */
#define ccdPROC_WIN1_USERFUNC "DET.WIN1.IPFUNC"  /* User Function name           */
#define ccdPROC_WIN1_USERBUFF "DET.WIN1.IPBUFF"  /* User Buffer name             */
#define ccdPROC_WIN1_THRMIN   "DET.WIN1.THRMIN"  /* threshold for min intensity */
#define ccdPROC_WIN1_THRMAX   "DET.WIN1.THRMAX"  /* threshold for max intensity */
#define ccdPROC_WIN1_BACKGND  "DET.WIN1.BACKGND" /* sky background               */
#define ccdPROC_WIN1_IPLLX    "DET.WIN1.IPLLX"   /* proc subWin offsets          */
#define ccdPROC_WIN1_IPLLY    "DET.WIN1.IPLLY"   /* proc subWin offsets          */
#define ccdPROC_WIN1_IPURX    "DET.WIN1.IPURX"   /* proc subWin offsets          */
#define ccdPROC_WIN1_IPURY    "DET.WIN1.IPURY"   /* proc subWin offsets          */

/*===========*/
/* WINDOW #2 */
/*===========*/
#define ccdREAD_WIN2_BIN_X    "DET.WIN2.BINX"    /* readout x binning            */
#define ccdREAD_WIN2_BIN_Y    "DET.WIN2.BINY"    /* readout y binning            */
#define ccdREAD_WIN2_ENAB     "DET.WIN2.ST"      /* enabled flag                 */
#define ccdREAD_WIN2_FIR_X    "DET.WIN2.STRX"    /* first x pixel                */
#define ccdREAD_WIN2_DIM_X    "DET.WIN2.NX"      /* x dimensions                 */
#define ccdREAD_WIN2_FIR_Y    "DET.WIN2.STRY"    /* first y pixel                */
#define ccdREAD_WIN2_DIM_Y    "DET.WIN2.NY"      /* y dimensions                 */
#define ccdPROC_WIN2_MINMAX   "DET.WIN2.MINMAX"  /* search image extrema         */
#define ccdPROC_WIN2_BIAS     "DET.WIN2.BIAS"    /* bias subtraction wanted      */
#define ccdPROC_WIN2_FLAT     "DET.WIN2.FLATF"   /* flat field division wanted   */
#define ccdPROC_WIN2_CENTROID "DET.WIN2.CENTROID"/* type of centroiding          */
                                                 /* see also ccdCENTROID_xxx     */
#define ccdPROC_WIN2_REF_X    "DET.WIN2.REFX"    /* X ref. star for centroiding */
#define ccdPROC_WIN2_REF_Y    "DET.WIN2.REFY"    /* Y ref. star for centroiding */
#define ccdPROC_WIN2_USERFUNC "DET.WIN2.IPFUNC"  /* User Function name           */
#define ccdPROC_WIN2_USERBUFF "DET.WIN2.IPBUFF"  /* User Buffer name             */
#define ccdPROC_WIN2_THRMIN   "DET.WIN2.THRMIN"  /* threshold for min intensity */
#define ccdPROC_WIN2_THRMAX   "DET.WIN2.THRMAX"  /* threshold for max intensity */
#define ccdPROC_WIN2_BACKGND  "DET.WIN2.BACKGND" /* sky background               */
#define ccdPROC_WIN2_IPLLX    "DET.WIN2.IPLLX"   /* proc subWin offsets          */
#define ccdPROC_WIN2_IPLLY    "DET.WIN2.IPLLY"   /* proc subWin offsets          */
#define ccdPROC_WIN2_IPURX    "DET.WIN2.IPURX"   /* proc subWin offsets          */
#define ccdPROC_WIN2_IPURY    "DET.WIN2.IPURY"   /* proc subWin offsets          */

/* paremeters foreseen but not used (the list is not finalized) */
#define ccdREADPIX            "DET.READ.NO"      /* pixel readout times (skip)   */
#define ccdMPPMODE            "DET.READ.MODE"    /* mpp mode                     */
#define ccdFILE_COM           "DET.FRAM.FITSCMP" /* file for compr. image data   */
```

```
/* command ccdCMD_START */
#define ccdPAR_START_EXPOID   "expoId"           /* exposure ID parameter name  */

/* command ccdCMD_VERS */
#define ccdPAR_VERS_CCDVERS   "ccdVersion"       /* version of CCD software     */

/* parameters for ccdCMD_STAND_ALONE */
#define ccdPAR_STANDAL_ON     "on"              /* stand-alone flag          */
#define ccdPAR_STANDAL_ARC    "archive"         /* archive images            */

/*
 * exposure status (ccdPAR_STATUS_EXPSTATUS). Bit-field
 */
#define ccdEXP_NONE             0
#define ccdEXP_INACTIVE         1
#define ccdEXP_PENDING          2
#define ccdEXP_INTEGRATING      4
#define ccdEXP_PAUSED           8
#define ccdEXP_READING         16
#define ccdEXP_PROCESSING      32
#define ccdEXP_TRANSFERING     64
#define ccdEXP_COMPLETED      128
#define ccdEXP_FAILED         256
#define ccdEXP_ABORTED        512
#define ccdEXP_LOOP_FINITE   1024
#define ccdEXP_LOOP_INFINITE 2048

#define ccdEXP_DONE \
       (ccdEXP_COMPLETED | ccdEXP_FAILED | ccdEXP_ABORTED)
#define ccdEXP_RUNNING \
       (ccdEXP_PENDING | ccdEXP_INTEGRATING | ccdEXP_PAUSED | \
        ccdEXP_READING | ccdEXP_PROCESSING | ccdEXP_TRANSFERING )
#define ccdEXP_LOOP \
       (ccdEXP_LOOP_FINITE | ccdEXP_LOOP_INFINITE)

/*
 * Infinite loop (ccdREPEAT_DEF)
 */
#define ccdREPEAT_FOREVER 0

/*
 * Display types
 */
#define ccdNO_DISPLAY   -1

/*
 * Wait modes
 */
#define ccdWAIT_SINGLE 0
#define ccdWAIT_GLOBAL 1

#define ccdWAIT_SINGLE_STR "Single"
#define ccdWAIT_GLOBAL_STR "Global"

/* ccdEXP_TYPE */
typedef enum
    {
```

```
    ccdEXP_NORMAL = 1,
    ccdEXP_DARK,
    ccdEXP_MULTI
    } ccdEXPTYPE;
#define ccdEXP_NORMAL_STR "Normal"
#define ccdEXP_DARK_STR   "Dark"
#define ccdEXP_MULTI_STR  "Multiple"   /* not implemented yet */

typedef char ccdCAMERANAME[ccdCAMERANAME_LEN + 1];   /* name of CCD camera  */

/* operational modes */
typedef enum                      /* possible operational modes           */
    {
    ccdNOT_AVAILABLE = 1,         /* CCD sw is not available              */
    ccdNORMAL,                    /* CCD sw and hw available              */
    ccdSIM_WS,                    /* LCU CCD sw is simulated at WS level  */
    ccdSIM_LCU,                   /* CCD hw is simulated at LCU level     */
    ccdSIM_ACE                    /* CCD hw is simulated at ACE level     */
    } ccdOPMODE;

/*
 * operational states
 */
typedef enum {
    ccdUNKNOWN   = ccsSTATE_UNK,     /* unknown                               */
    ccdOFF       = ccsSTATE_OFF,     /* terminating                           */
    ccdLOADED    = ccsSTATE_LOADED,  /* need to be initialised                */
    ccdSTANDBY   = ccsSTATE_STANDBY, /* stand-by                              */
    ccdONLINE    = ccsSTATE_ONLINE,  /* ready for exposures                   */
} ccdSTATE;

#define ccdOPERATING ccsSTATE_ONLINE

/*
 * shutter status definition
 */
typedef enum
    {
    ccdSHT_ERROR = -1,
    ccdSHT_CLOSED,
    ccdSHT_OPENED,
    ccdSHT_CLOSING,
    ccdSHT_OPENING
    } ccdSHT_STATUS;

/* Various ways of saving an image in disk file(s) */
typedef enum
    {
    ccdDISK_NONE = 0,
    ccdDISK_COMPRESS,              /* not implemented yet */
    ccdDISK_UNCOMPRESS,
    ccdDISK_BOTH                   /* not implemented yet */
    } ccdDISKSAVE;

#define ccdDISK_NONE_STR       "None"
#define ccdDISK_COMPRESS_STR   "Compress"
#define ccdDISK_UNCOMPRESS_STR "Uncompress"
#define ccdDISK_BOTH_STR       "Both"
```

```
/*
 * Frame type (for storage in LCU memory, see setup param. ccdREAD_FRAME_TYPE)
 */
#define ccdFRAME_NORMAL "Normal"          /* normal frame: temporary storage */
#define ccdFRAME_BIAS   "Bias"    /* reference bias frame: permanent storage */
#define ccdFRAME_DARK   "Dark"              /* dark frame: permanent storage */
#define ccdFRAME_FLAT   "FF" /* reference flatfield frame: permanent storage */

typedef enum
    {
    ccdFRM_NORMAL = 0,
    ccdFRM_BIAS,
    ccdFRM_DARK,
    ccdFRM_FLAT
    } ccdFRMTYPE;

/*
 * Multi-Step Integration time
 */
#define ccdEXPTIME_IDEM ccsTRUE  /* use DET.WIN1.UIT1 for all steps */
#define ccdEXPTIME_LIST ccsFALSE /* use DET.WIN1.UITi as defined    */

/*
 * Line Shift type
 */
    /* Setup values for Line Shift type */
typedef enum
    {
    ccdLINE_SHIFT_ALT = -1,      /* use DET.READ.SHIFT1 alt. pos & neg*/
                                 /* Alternate exposure mode           */
    ccdLINE_SHIFT_IDEM = 0,      /* use DET.READ.SHIFT1 for all steps */
                                 /* Focus & Sky Coherence modes       */
    ccdLINE_SHIFT_LIST = 1       /* use DET.READ.SHIFTi as defined    */
    } ccdLINESHIFT_TYPE;

#define ccdLINE_SHIFT_ALT_STR  "alternate"
#define ccdLINE_SHIFT_IDEM_STR "idem"
#define ccdLINE_SHIFT_LIST_STR "list"

/*
 * Centroiding type (see setup param. ccdIP_CENTROID)
 */
typedef enum
    {
    ccdCENTROID_NONE = 0,
    ccdCENTROID_STANDARD,
    ccdCENTROID_THRESHOLD
    } ccdCENTYPE;

#define ccdCEN_NONE_STR      "none"       /* no centroiding calculation      */
#define ccdCEN_STANDARD_STR  "standard"   /* standard centroiding calc.      */
#define ccdCEN_THRESHOLD_STR "threshold"  /* standard centroiding calc.      */

    /*
     * Maximum Threshold :
     * ===================
     * - Ignored in the current implementation
```

```
     */

    /*
     * Minimum Threshold :
     * ===================
     * => value >= 0 : use this value
     * => value within the integer range [-1;-9] :
     *    the threshold min is preset to N StdDev of Intensity distribution
     *    over this window
     * => value within the integer range [-11;-19] :
     *    the threshold min is preset to N StdDev of Intensity distribution
     *    over the first window
     *    This setting is only valid for the second window
     */
    /* The following macros are standard settings */
#define ccdTHRMIN_3SIGMA_WINDOW_SELF  -3
#define ccdTHRMIN_3SIGMA_WINDOW_PREV -13
#define ccdTHRMIN_5SIGMA_WINDOW_SELF  -5
#define ccdTHRMIN_5SIGMA_WINDOW_PREV -15

    /*
     * BackGround :
     * ============
     * => value within the >= 0 : use this value
     * => value set to -1 :
     *    the BackGround is preset to the Average Intensity
     *    over this window
     * => value set to -11 :
     *    the BackGround is preset to the Average Intensity
     *    over the first window
     *   This setting is only valid for the second window
     */
    /* The following macros are standard settings */
#define ccdBCKGND_FLUX_WINDOW_SELF    -1
#define ccdBCKGND_FLUX_WINDOW_PREV   -11

/* real-time image processing user function definitions */
#define ccdIP_NO_USER_FUN   "None"        /* no user function to be called   */
#define ccdIP_NO_USER_BUF   "None"        /* user function has no buffer     */

/*
 * IMAGE PROCESSING DATA STRUCTURES
 * --------------------------------
 * The coordinates are, unless specified, absolute to the lower left corner
 * of the detector. This pixel has the coordinates (1,1).
 */
typedef struct
    {
    /*
     * The exposure structure provides general information on the
     * exposure.
     */
    ccdEXPTYPE   expType;
    vltINT32     stepsDef;            /* number of steps (MULTI)            */
    ccsTIMEVAL   wipeStart;           /* actual wipe start time (UTC)       */
    vltDOUBLE    wipeTime;            /* actual wipe duration               */
    ccsTIMEVAL   expStart;            /* actual integration start time (UTC) */
    vltDOUBLE    expTime;             /* Effective exposure time            */
```

```
    vltDOUBLE    expTimeMulti[ccdMAXINTEGR];
    vltINT32     shiftLine[ccdMAXINTEGR];
    ccsTIMEVAL   shiftStart[ccdMAXINTEGR]; /* Line shift start time (UTC)    */
    vltDOUBLE    shiftTime;           /* Effective line shift time           */
    ccsTIMEVAL   readStart;           /* actual readout start time (UTC)     */
    vltDOUBLE    readTime;            /* actual readout duration             */
    ccsTIMEVAL   procStart;           /* actual image proc. start time (UTC) */
    } ccdIPEXPOSURE;

/* data types */
typedef enum
    {
    ccdIMAGE_CHAR = 0,               /* 8 bit unsigned                     */
    ccdIMAGE_INT,                    /* 16 bit signed                      */
    ccdIMAGE_FLOAT                   /* 32 bit floating point              */
    } ccdIMAGE_REP;

typedef struct
    {
    /*
     * The window structure describes the current window passed to the
     * IP function. Besides the information on the window position on
     * the detector and its dimensions, available are also the window
     * index (starting from 0 as first window defined), the pixel
     * representation (integer 8 or 16 bits, or 32-bit floating point)
     * which is required for proper casting of the data pointer.
     */
    vltINT32     winNum;              /* Window index                        */
    ccdIMAGE_REP pixelRep;            /* Pixel value representation          */
    vltINT32     xPos,yPos;           /* Location of lower left corner       */
    vltINT32     xDim,yDim;           /* Window dimension                    */
    void        *data;                /* Pointer to image data               */
    } ccdIPWINDOW;

typedef struct
    {
    /*
     * The setup structure contains the setup parameters used by the
     * image processing functions.
     */
    vltINT32     llx,lly;             /* Offset to Lower Left  corner        */
    vltINT32     urx,ury;             /* Offset to Upper Right corner        */
    vltDOUBLE    xRef,yRef;           /* Reference point                     */
    vltDOUBLE    minThr,maxThr;       /* Thresholds                          */
    vltDOUBLE    bckGnd;              /* Estimation of Sky Back Ground       */
    } ccdIPSETUP;

typedef struct
    {
    /*
     * The result structure is filled in according to the setup
     * The first part (STATISTICS) is automatically updated
     * whenever the centroid calculation or a user-function is
     * invoked.
     * Since the two advanced image processing functions
     * ccdipAG(3) and ccdipIQE(3) have the same interface (ccdIPUSERFUNC),
     * it is possible to design a more complex function complying the
     * interface and invoking these functions in the processing sequence.
```

```
     * However, the designer has to considere the computation times which
     * are involved (for 20x20 window) :
     *       ccdipAG  : <20ms
     *       ccdipIQE : 800ms
     */
        /* STATISTICS : ccdipStatistic() */
    vltDOUBLE    max;                /* First Maximum Intensity            */
    vltDOUBLE    xMax,yMax;          /* Location                           */
    vltDOUBLE    min;                /* First Minimum Intensity            */
    vltDOUBLE    xMin,yMin;          /* Location                           */
    vltDOUBLE    flux;               /* Raw Average ADC counts / pixel     */
    vltDOUBLE    sigma;              /* Standard-Deviation on the flux     */
    vltINT32     numSpot;            /* Number of spots in image           */
    vltINT32     numPix;             /* Number of valid pixels in image    */
        /* CENTROID : ccdipCentroid() or ccdipAG() or ccdipWCE() */
    vltDOUBLE    cen;                /* Centroid Intensity (closest pixel)  */
    vltDOUBLE    xCen,yCen;          /* Location                           */
    vltDOUBLE    xErr,yErr;          /* Error vector (relative to Ref Point)*/
    vltDOUBLE    SNR;                /* Signal-to-Noise Ratio              */
    vltDOUBLE    xFWHM,yFWHM;        /* FWHM estimation                    */
        /* ANALYSIS : ccdipIQE() */
    vltDOUBLE    xFWHM_SD,yFWHM_SD;  /* FWHM Std Dev                       */
    vltDOUBLE    ObjAngle,ObjAngle_SD;/* Object position angle  + Std Dev  */
        /* BACKGROUND : ccdipBackground() */
    vltDOUBLE    BckGnd,BckGnd_SD;   /* Back Ground estimation + Std Dev   */
    } ccdIPRESULT;

typedef struct
    {
    ccdIPEXPOSURE exposure;
    ccdIPWINDOW   window;
    ccdIPSETUP    setup[ccdMAXWINDOW];
    ccdIPRESULT   result[ccdMAXWINDOW];
    } ccdIPIMAGE;

/* Function prototype for LCU real-time image processing user function */
typedef ccsCOMPL_STAT (* ccdIPUSERFUNC)(ccdIPIMAGE *,void *,ccsERROR *);

/* info about chip configuration */
typedef struct
    {
    vltLOGICAL available;            /* available                        */
    vltINT32   xLocation;            /* x location in mosaic             */
    vltINT32   yLocation;            /* y location in mosaic             */
    } ccdCONFCHIP;

/* info about chip configuration */
typedef struct
    {
    vltLOGICAL available;            /* available                        */
    vltINT32   chipIndex;            /* index of chip it belongs to      */
    vltINT32   xLocation;            /* x location in mosaic             */
    vltINT32   yLocation;            /* y location in mosaic             */
    vltINT32   leftToRight;          /* horizontal readout direction     */
    vltINT32   downToUp;             /* vertical readout direction       */
    vltINT32   xPrescan;             /* prescan pixels along x           */
    vltINT32   yPrescan;             /* prescan pixels along y           */
    vltINT32   xPix;                 /* active pixels along x            */
```

```
    vltINT32   yPix;                 /* active pixels along y              */
    vltINT32   xOverscan;            /* overscan pixels along x            */
    vltINT32   yOverscan;            /* overscan pixels along y            */
    } ccdCONFOUTPUT;

/* info about clock pattern */
typedef struct
    {
    vltBYTES16 description;          /* description of clock pattern used  */
    vltINT32   firstOutput;          /* output generating first pixel value*/
    vltINT32   gainIndex;            /* default gain index                 */
    vltINT32   nWindows;             /* maximum number of windows supported*/
    vltINT32   adcTime;              /* usec needed to digitize one pixel  */
    vltINT32   lineShiftMode;        /* supported line shift mode          */
    vltLOGICAL outEnab[ccdMAXOUTPUT];/* outputs enabled                    */
    vltDOUBLE  elAdu[ccdMAXOUTPUT];
    } ccdCONFCLOCK;

/* info about camera shutter          */
typedef struct
    {
    vltLOGICAL available;            /* available                          */
    } ccdCONFSHUTTER;

/* info about camera configuration    */
typedef struct
    {
    ccdCAMERANAME name;              /* camera name                        */
    ccsENVNAME    envName;           /* environment name                   */
    } ccdCAMERA;

/* info about camera configuration  */
typedef struct
    {
    vltLOGICAL     frameTransfer;       /* frame transfer CCD                 */
    vltLOGICAL     skipper;             /* skipper CCD                        */
    vltLOGICAL     mpp;                 /* mpp mode supported                 */
    vltINT32       xPixels;             /* Number of columns                  */
    vltINT32       yPixels;             /* Number of rows                     */
    vltINT32       bitsPixel;           /* ADC resolution                     */
    vltINT32       numOutputs;          /* Number of outputs                  */
    ccdCONFCHIP    chip[ccdMAXCHIPS];   /* single chip description            */
    ccdCONFOUTPUT  output[ccdMAXOUTPUT];/* single output description          */
    ccdCONFCLOCK   clock[ccdMAXCLOCKS]; /* single clock pattern information */
    ccdCONFSHUTTER shutter;             /* shutter information                */
    } ccdCONFIG;

/* window readout definition */
typedef struct
    {
    vltLOGICAL enabled;             /* this window is enabled for readout  */
    vltLOGICAL idem;                /* same binning as Window #1           */
    vltINT32   xBinning;            /* horizontal binning factor           */
    vltINT32   yBinning;            /* vertical binning factor             */
    vltINT32   xFirst;              /* first column                        */
    vltINT32   xDim;                /* number of columns                   */
    vltINT32   yFirst;              /* first row                           */
    vltINT32   yDim;                /* number of rows                      */
```

```
    } ccdWINDOW;

/* window process definition */
typedef struct
    {
    vltLOGICAL   minMax;            /* do MinMax                         */
    vltLOGICAL   bias;              /* do Bias correction                */
    vltLOGICAL   dark;              /* do Dark correction                */
    vltLOGICAL   flat;              /* do Flat Field correction          */
    vltINT32     averageN;          /* do average over N frames          */
    vltINT32     centroiding;       /* do Centroid calculation (mode)    */
    vltDOUBLE    centrRefX;         /* Reference Point X                 */
    vltDOUBLE    centrRefY;         /* Reference Point Y                 */
    vltDOUBLE    centrThreshMin;    /* Centroid Threshold Min            */
    vltDOUBLE    centrThreshMax;    /* Centroid Threshold Max            */
    vltDOUBLE    backGround;        /* Sky BackGround                    */
    vltBYTES32   userFuncName;      /* User-Defined function name        */
    vltBYTES32   userBuffName;      /* User-Defined buffer address       */
    vltINT32     procLLX,procLLY;   /* Lower Left & Upper Right corners  */
    vltINT32     procURX,procURY;   /* of Processing window (offset from */
                                    /* physical window)                  */
    } ccdPROCESS;

/* exposure setup parameters */
typedef struct
    {
    ccdEXPTYPE   expType;        /* exposure type                           */
    vltINT32     stepsDef;       /* number of steps (MULTI)                 */
    vltINT32     repeatDef;      /* repetition number (0 is forever)        */
    vltINT32     wipeTime;       /* wipe wanted (>= 0) or not (<0)          */
    ccdFRMTYPE   frameType;      /* frame type                              */
    vltLOGICAL   timeDefIdem;    /* exposure time UITi as UIT1 ?            */
    vltDOUBLE    timeDef;        /* exposure time (step 1)                  */
    vltDOUBLE    timeDefMulti[ccdMAXINTEGR];
                                 /* exposure time (steps 2+ for MULTI)      */
    vltDOUBLE    timeRepeat;     /* sec between repeated exposures          */
    vltLOGICAL   fitsInfoCollect;/* Collect info for FITS header            */
    vltLOGICAL   tempExp;        /* Store tmp. periodically during exposure */
    vltINT32     tempSecs;       /* sec between temperature checks          */
    vltLOGICAL   teleExp;        /* Store tel. periodically during exposure */
    vltINT32     teleSecs;       /* sec between telemetry checks            */
        /* IT  : */
    ccdDISKSAVE  diskSave;             /* disk save method                  */
    vltBYTES32   fileNameUnComp;       /* uncompressed FITS file name       */
    vltBYTES32   fileNameComp;         /* compressed FITS file name         */
    vltINT32     pixRepr;              /* # bit for representation          */
    vltINT32     bitShift;             /* # bit shift for RTD               */
    vltINT32     compression;          /* compression type                  */
    vltINT32     sampling;             /* 1 image transferred out of N read */
    vltLOGICAL   itSync;               /* image transfer synchronous        */
        /* RTD : */
    vltINT32     frameId;              /* real-time display frame Id        */
        /* IP  : */
    ccdPROCESS   process[ccdMAXWINDOW];/* single window setup               */
        /* RDT : */
    vltINT32     linesToShiftType;     /* how lines shall be shifted        */
    vltINT32     linesToShift;         /* # lines to be shifted step 1      */
    vltINT32     linesToShiftMulti[ccdMAXINTEGR];
```

```
                                      /* # lines to be shifted steps 2+    */
    vltINT32     readPixelN;          /* # times the same pixel is read    */
    vltLOGICAL   mpp;                 /* MPP mode flag                     */
    vltINT32     clockIndex;          /* readout clock to use              */
    vltINT32     gainIndex;           /* readout gain index                */
    vltBYTES32   fileNameSimu;        /* simulated image (ccdSIM_LCU)      */
    ccdWINDOW    window[ccdMAXWINDOW]; /*  single window setup             */
    } ccdSETUP;

/* results of setup check */
typedef struct
    {
    vltLOGICAL changed;             /* some setup values have been changed*/
    vltLOGICAL doTransfer;          /* Image transfer required            */
    vltLOGICAL doProcess;           /* Image processing required          */
    vltDOUBLE  readTime;            /* estimated readout time (sec)       */
    vltDOUBLE  wipeTime;            /* estimated wiping  time (sec)       */
    } ccdSETUPRES;

/*
 * Macros
 */

/*
 * function prototypes
 */
/* Get information about a CCD camera configuration */
ccsCOMPL_STAT ccdGetConf    ( ccdCAMERA *camera, ccdCONFIG *config,
                              ccsERROR  *error);

/* Check a complete exposure setup */
ccsCOMPL_STAT ccdCheckSetup ( ccdCONFIG *config, ccdSETUP *setup,
                              ccdSETUPRES *results, ccsERROR *error);

/* Check the setup for a windowed readout */
ccsCOMPL_STAT ccdCheckSetupWindow ( ccdCONFIG *config, ccdSETUP *setup,
                                    ccdSETUPRES *results, ccsERROR *error);

/* Get information about a CCD camera configuration */
void ccdGetCIName           ( ccdCAMERANAME camera, ccsPROCNAME procName );

/* Extract index of DB point describing an exposure from the exposure ID     */
vltINT32 ccdGetIndexFromId  ( vltINT32 expId );

#ifdef __cplusplus
}
#endif

#endif /* ! CCD_H */
```

3.5.2 ccdDbPublic.h

```
/************************************************************************
* E.S.O. - VLT project
*
* "@(#) $Id: ccdDbPublic.h,v 2.4 1998/09/18 08:57:19 vltsccm Exp $"
*
* who       when      what
* --------  --------  ----------------------------------------------
* alongino  31/08/94  created
* alongino  19/05/95  modified while rearranging database for dbLoader
* alongino  18/07/95  ccdMAXEXP added
* alongino  20/07/95  ccdMAXOUTCHIP added
* alongino  29/09/95  changed name from ccdDbDimensions.h to ccdDbPublic.h
*                     added macros for public part of the database
* pduhoux   30/01/96  added macro ccdDB_EXP_FILECOM
* alongino  21/02/96  added macros define ccdDB_IP_XCEN ccdDB_IP_YCEN
*                     ccdDB_IP_CENVAL
* pduhoux   28/05/96  added macros ccdDB_STA_FAILURE_LCU/WS
* alongino  18/07/96  added macro ccdDB_STA_EXPID
* P.Duhoux  21/01/97  moved ccdDB_WCS_RA & DEC for WorldCoord attributes
*                     from ccdDatabase.h
* P.Duhoux  27/02/97  Added macros for images:process_2 attributes
*                     ccdDB_IP2_
* P.Duhoux  04/03/97  Renamed point <process> into <process:window_1>
* alongino  26/05/97  ccdDB_IP_USERBUFFER removed (SPR 970177)
*                     same for ccdDB_IP_OBJNUMBER and ccdDB_IP_OBJECTS
* P.Duhoux  02/06/97  added ccdDB_IP_FLUX & ccdDB_IP_NUMPIX
* P.Duhoux  21/07/97  added ccdDB_IP_SNR
* P.Duhoux  12/08/97  added ccdDB_STA_INITDONE
* P.Duhoux  09/10/97  added ccdDB_STA_XXX1 for full path to exposure_1 attr.
*                     added ccdDB_IPxxx for extrema
*                     added ccdDB_STA_TELMESSAGE & ccdDB_STA_TMPMESSAGE
*/

/************************************************************************
*  This file contains all definitions for CCD database public attributes
************************************************************************/

#ifndef CCDDBPUBLIC_H
#define CCDDBPUBLIC_H

/*
 * Definition of database vector and table dimensions. Keep consistant !
 * NOTE: This file is used by dbLoader as well. Nothing else than define
 *       instructions or comment lines are possible here.
 */
/* These constants are actually not used directly in the database definition
 * files, but the database itself supports configuration for these values
 * only. To increase capabilities, modifications in the class definition
 * files are needed as well, in particular ccdDETECTOR and ccdEXPOSURES
 */
#define ccdMAXCHIPS   1       /* maximum number of chips in one mosaic      */
#define ccdMAXWINDOW  2       /* maximum number of windows for one readout  */
#define ccdMAXCLOCKS  10      /* maximum number of clock patterns           */
```

```
#define ccdMAXEXP     2       /* maximum number of expositions              */

/* These constants are used directly in the database definition files */
#define ccdMAXLINKS   15      /* maximum number of transputer links to ACE  */
#define ccdMAXLENMSG  32      /* maximum length of message to/from ACE      */
#define ccdMAXGAIN    8       /* maximum number of possible gains           */
#define ccdMAXOUTCHIP 4       /* maximum number of outputs in one chip      */
#define ccdMAXOUTPUT  4       /* maximum number of outputs in mosaic        */
#define ccdMAXOBJECTS 100     /* maximum number of objects in image proc.   */
#define ccdMAXINTEGR  10      /* maximum number of integrations in one exp. */
#define ccdMAXIMAGES  100     /* maximum number of images in memory         */
#define ccdMAXTELSRC  100     /* maximum number of telemetry sources        */
#define ccdMAXTMPSRC  10      /* maximum number of temperature sensors      */

/* Public part of the CCD database */

/* operational mode */
#define ccdDB_CON_OPMODE      ".opMode"

/* system state */
#define ccdDB_STA_SYSTEM      ".opState"

/* system errors */
#define ccdDB_STA_FAILURE_WS  ".failureWs"
#define ccdDB_STA_FAILURE_LCU ".failureLcu"

/* default image file directory  */
#define ccdDB_CON_IMGPATH     "images.imageDirectory"

/* running exposures prefix  */
#define ccdDB_EXP_POINT       "exposures:exposure"
#define ccdDB_EXP_POINT1      "exposures:exposure_1"

/* exposure status - under ccdDB_EXP_POINT_<expIndex> */
#define ccdDB_STA_EXPOSURE    "expStatus"
#define ccdDB_STA_EXPOSURE1   "exposures:exposure_1.expStatus"

/* exposure ID - under ccdDB_EXP_POINT_<expIndex> */
#define ccdDB_STA_EXPID       "expId"
#define ccdDB_STA_EXPID1      "exposures:exposure_1.expId"

/* name of uncompressed FITS file */
#define ccdDB_EXP_FILEUNC     "transfer.fileNameUnComp"
#define ccdDB_EXP_FILEUNC1    "exposures:exposure_1:transfer.fileNameUnComp"

/* Shared Memory Id associated to the Frame (with rtdServer) */
#define ccdDB_EXP_SHMID       "display.shmId"
#define ccdDB_EXP_SHMID1      "exposures:exposure_1:display.shmId"

/* Extrema */
#define ccdDB_IP_XMIN         "images:process:window_1.ipXMin"
#define ccdDB_IP1_XMIN        ccdDB_IP_XMIN
#define ccdDB_IP2_XMIN        "images:process:window_2.ipXMin"

#define ccdDB_IP_YMIN         "images:process:window_1.ipYMin"
#define ccdDB_IP1_YMIN        ccdDB_IP_YMIN
#define ccdDB_IP2_YMIN        "images:process:window_2.ipYMin"
```

```
#define ccdDB_IP_MINVAL       "images:process:window_1.ipMinVal"
#define ccdDB_IP1_MINVAL      ccdDB_IP_MINVAL
#define ccdDB_IP2_MINVAL      "images:process:window_2.ipMinVal"

#define ccdDB_IP_XMAX         "images:process:window_1.ipXMax"
#define ccdDB_IP1_XMAX        ccdDB_IP_XMAX
#define ccdDB_IP2_XMAX        "images:process:window_2.ipXMax"

#define ccdDB_IP_YMAX         "images:process:window_1.ipYMax"
#define ccdDB_IP1_YMAX        ccdDB_IP_YMAX
#define ccdDB_IP2_YMAX        "images:process:window_2.ipYMax"

#define ccdDB_IP_MAXVAL       "images:process:window_1.ipMaxVal"
#define ccdDB_IP1_MAXVAL      ccdDB_IP_MAXVAL
#define ccdDB_IP2_MAXVAL      "images:process:window_2.ipMaxVal"

/* Centroiding Error vector */
#define ccdDB_IP_XCEN         "images:process:window_1.ipXCen"
#define ccdDB_IP1_XCEN        ccdDB_IP_XCEN
#define ccdDB_IP2_XCEN        "images:process:window_2.ipXCen"

#define ccdDB_IP_YCEN         "images:process:window_1.ipYCen"
#define ccdDB_IP1_YCEN        ccdDB_IP_YCEN
#define ccdDB_IP2_YCEN        "images:process:window_2.ipYCen"

/* Intensity of image center, as computed from centroiding algorythm */
#define ccdDB_IP_CENVAL       "images:process:window_1.ipCenVal"
#define ccdDB_IP1_CENVAL      ccdDB_IP_CENVAL
#define ccdDB_IP2_CENVAL      "images:process:window_2.ipCenVal"

/* Average flux / pixel */
#define ccdDB_IP_FLUX         "images:process:window_1.ipFlux"
#define ccdDB_IP1_FLUX        ccdDB_IP_FLUX
#define ccdDB_IP2_FLUX        "images:process:window_2.ipFlux"

/* Standard Deviation of Pixel Intensity */
#define ccdDB_IP_STDDEV       "images:process:window_1.ipStdDev"
#define ccdDB_IP1_STDDEV      ccdDB_IP_STDDEV
#define ccdDB_IP2_STDDEV      "images:process:window_2.ipStdDev"

/* Signal-to-Noise Ratio */
#define ccdDB_IP_SNR          "images:process:window_1.ipSNR"
#define ccdDB_IP1_SNR         ccdDB_IP_SNR
#define ccdDB_IP2_SNR         "images:process:window_2.ipSNR"

/* number of pixels between the thresholds */
#define ccdDB_IP_NUMPIX       "images:process:window_1.ipNumPix"
#define ccdDB_IP1_NUMPIX      ccdDB_IP_NUMPIX
#define ccdDB_IP2_NUMPIX      "images:process:window_2.ipNumPix"

/* Background estimation */
#define ccdDB_IP_BGND         "images:process:window_1.ipBGnd"
#define ccdDB_IP1_BGND        ccdDB_IP_BGND
#define ccdDB_IP2_BGND        "images:process:window_2.ipBGnd"

/* StdDev on Background */
#define ccdDB_IP_BGND_SD      "images:process:window_1.ipBGnd_SD"
#define ccdDB_IP1_BGND_SD     ccdDB_IP_BGND_SD
```

```
#define ccdDB_IP2_BGND_SD     "images:process:window_2.ipBGnd_SD"

/* Object FWHM X estimation */
#define ccdDB_IP_FWHM_X       "images:process:window_1.ipFWHM_X"
#define ccdDB_IP1_FWHM_X      ccdDB_IP_FWHM_X
#define ccdDB_IP2_FWHM_X      "images:process:window_2.ipFWHM_X"

/* Object FWHM Y estimation */
#define ccdDB_IP_FWHM_Y       "images:process:window_1.ipFWHM_Y"
#define ccdDB_IP1_FWHM_Y      ccdDB_IP_FWHM_Y
#define ccdDB_IP2_FWHM_Y      "images:process:window_2.ipFWHM_Y"

/* attributes for WCS handling */
#define ccdDB_WCS_RA          "wcs.ra"
#define ccdDB_WCS_DEC         "wcs.dec"

/*
 * Attributes supposed to be used ONLY within GUI panels
 */

/* remaining exposure time for each step - under ccdDB_EXP_POINT_<expIndex> */
#define ccdDB_STA_TIMEREM     "timeRem"
#define ccdDB_STA_TIMEREM1    "exposures:exposure_1.timeRem"

/* % of image transf. */
#define ccdDB_IT_STA_PERC     "images:transfer.percent"

/* last line transf. */
#define ccdDB_IT_STA_LINE     "images:transfer.last"

/* shutter status */
#define ccdDB_STA_SHTSTATUS   "shutter.status"

/* telemetry status message */
#define ccdDB_STA_TELMESSAGE  "telemetry.message"

/* temperature status message */
#define ccdDB_STA_TMPMESSAGE  "temperature.message"

#endif /* ! CCDDBPUBLIC_H */
```

3.5.3 ccdErrors.h

```
/*******************************************************
# "@(#) $Id: ccdErrors.h,v 2.4 1998/09/18 08:57:18 vltsccm Exp $"
#
# Error Include File    Created on  Sep 10 15:44:33 1998
#
# This file has been generated by a utility
#
#  !!!!!!!!!!!  DO NOT MANUALLY EDIT THIS FILE  !!!!!!!!!!!
#
********************************************************/
#ifndef ccdERRORS_H
#define ccdERRORS_H


#define ccdERR_BUSY                   1
#define ccdERR_CMD_EXECUTE            2
#define ccdERR_CMD_IGNORED            3
#define ccdERR_CMD_UNKNOWN            4
#define ccdERR_CMD_WARNING            5
#define ccdERR_DB_ROOT                6
#define ccdERR_DB_CONSISTENCY         7
#define ccdERR_DB_READ                8
#define ccdERR_DB_WRITE               9
#define ccdERR_ENV_UNKNOWN            10
#define ccdERR_INVALID_ARGUMENT       11
#define ccdERR_NOT_IMPLEMENTED        12
#define ccdERR_OPEN_FILE              13
#define ccdERR_OPMODE                 14
#define ccdERR_OPSTATE                15
#define ccdERR_SETUP                  16
#define ccdERR_STATUS                 17
#define ccdERR_TIMEOUT                18
#define ccdERR_PARAM_INVALID          19
#define ccdERR_PARAM_RANGE            20
#define ccdERR_CCS_CALL               101
#define ccdERR_CD_SET                 102
#define ccdERR_CONFIGURATION          103
#define ccdERR_CMD_LIST_FULL          104
#define ccdERR_CMD_LIST_NOT_FOUND     105
#define ccdERR_ENV                    106
#define ccdERR_ENV_GET                107
#define ccdERR_ENV_VAR                108
#define ccdERR_ENV_IT                 109
#define ccdERR_FITS_FILE              110
#define ccdERR_FITS_KEYWORD           111
#define ccdERR_IMAGE                  112
#define ccdERR_IMAGE_INFO             113
#define ccdERR_MSG_ADD                114
#define ccdERR_MSG_ADD_CMD            115
#define ccdERR_MSG_COMM               116
#define ccdERR_MSG_NOT_FOUND          117
#define ccdERR_MSG_RCV                118
#define ccdERR_MSG_RPLY               119
```

```
#define ccdERR_NOT_INIT               120
#define ccdERR_NOT_ONLINE             121
#define ccdERR_PIXELS                 122
#define ccdERR_RPLY                   123
#define ccdERR_RPLY_EXEC              124
#define ccdERR_RPLY_LENGTH            125
#define ccdERR_RPLY_SEND              126
#define ccdERR_RPLY_UNEXP             127
#define ccdERR_RPLY_UNEXP_LAST        128
#define ccdERR_SETUP_FILE             129
#define ccdERR_SETUP_PAR              130
#define ccdERR_SHARED_MEMORY          131
#define ccdERR_STATE_EXEC             132
#define ccdERR_TIME                   133
#define ccdERR_NO_DISPLAY             134
#define ccdERR_TIMES_NOT_ALIGNED      135
#define ccdERR_TASK_SPAWN             201
#define ccdERR_TASK_EXISTS            202
#define ccdERR_TASK_NAME              203
#define ccdERR_NO_MEMORY              204
#define ccdERR_FIND_SYMBOL            205
#define ccdERR_VX_CALL                206
#define ccdERR_SEMAPHORE              207
#define ccdERR_ENVIRONMENT            208
#define ccdERR_ACE_CALL               209
#define ccdERR_DCL_CALL               211
#define ccdERR_LCC_CALL               212
#define ccdERR_INT_CALL               213
#define ccdERR_USR_CALL               214
#define ccdERR_INIT_MODULE            215
#define ccdERR_MSG_TYPE               218
#define ccdERR_FIND_FILE              219
#define ccdERR_READ_FILE              220
#define ccdERR_SEND_DATA              221
#define ccdERR_FORMAT                 226
#define ccdERR_REPLY                  227
#define ccdERR_STARTUP                228
#define ccdERR_TIMER                  229
#define ccdERR_SEND_COMMAND           230
#define ccdERR_IMAGE_LIST             231
#define ccdERR_ID                     232
#define ccdERR_IMAGE_MISMATCH         233
#define ccdERR_TELEMETRY              234
#define ccdERR_TEMPERATURE            235

#endif
```

3.5.4 ccdObj.h

```
/*************************************************************************
* E.S.O. - VLT project
*
* "@(#) $Id: ccdObj.h,v 1.76 1997/08/28 11:10:20 vltsccm Exp $"
*
* who          when      what
* -----------  --------  ----------------------------------------------
* A.Longinotti 12/11/96  created
* A.Longinotti 28/04/97  removed pco.h. Problems on Solaris
*/

/*************************************************************************
*  This file contains all definitions used by the library ccdObj
*************************************************************************/

#ifndef CCDOBJ_H
#define CCDOBJ_H

#ifdef __cplusplus
extern "C" {
#endif

/*
 * CCS & LCC header files
 */
#include "CCS.h"

/*
 * CCD header files
 */

/*
 * Constants
 */

/*
 * Types definitions
 */
typedef struct                 /* parameters needed to find objects        */
    {
    vltUINT8 procId;           /* Midas unit ID (between 0 and 99)         */
    vltLOGICAL threshAbs;      /* meaning of threshVal                     */
    vltINT32 threshVal;        /* intensity threshold                      */
    } ccdOBJMETHOD;

typedef struct                 /* object information                       */
    {
    vltFLOAT xPos;             /* position along x axis (pixel coord.)     */
    vltFLOAT yPos;             /* position along y axis (pixel coord.)     */
    vltFLOAT intensity;        /* intensity                                */
    } ccdOBJECT;

/*
 * function prototypes
```

```
 */
/* Get all objects detected in an image */
ccsCOMPL_STAT ccdObjAll(
    char *file,                /* name of FITS file containing image    */
    const ccdOBJMETHOD *method, /* parameters needed for the computation */
    vltINT32  *objectsN,       /* number of objects found               */
    ccdOBJECT **objects,       /* info about objects found              */
    ccsERROR *error);          /* error structure                       */

/* Get the brightest object in an image */
ccsCOMPL_STAT ccdObjBright(
    char *file,                /* name of FITS file containing image    */
    const ccdOBJMETHOD *method, /* parameters needed for the computation */
    ccdOBJECT *object,         /* info about object found               */
    ccsERROR *error);          /* error structure                       */

/* Get the closest object to a reference point in an image */
ccsCOMPL_STAT ccdObjClose(
    char *file,                /* name of FITS file containing image    */
    const ccdOBJMETHOD *method, /* parameters needed for the computation */
    ccdOBJECT *reference,      /* reference position                    */
    ccdOBJECT *object,         /* info about object found               */
    ccsERROR *error);          /* error structure                       */

/* Display the image analysed with all objects detected */
ccsCOMPL_STAT ccdObjDisplay(
    char *file,                /* name of FITS file containing image    */
    vltUINT8 procId,           /* Midas unit ID                         */
    ccsERROR *error);          /* error structure                       */

#ifdef __cplusplus
}
#endif

#endif /* ! CCDOBJ_H */
```

3.6 Database

This section contains the manual pages for:

1. Database Class definition for CCD DCS
2. Database Class definition for a stand-alone CCD camera
3. Configuration File for CCD database branch for a stand-alone camera

It also contains examples (to be used as templates) of:

4. DATABASE.db file
5. USER.db file

3.6.1 ccdCAMERADCS

```
//*********************************************************************
//* E.S.O. - VLT project
//*
//* "@(#) $Id: ccdCAMERADCS.class,v 2.4 1998/09/18 08:56:58 vltsccm Exp $"
//*
//* who           when      what
//* --------      --------  ----------------------------------------------
//* A.Longinotti 18/05/95  created
//* P.Duhoux     21/01/97  added point 'wcs' for World Coordinates
//* P.Duhoux     16/09/97  added attr 'logMask' for FITS logging
//* P.Duhoux     11/11/97  added attr 'installed' for FITS DET.DATE
//*

//***********************************************************************
//*   NAME
//*   ccdCAMERADCS - Common class definitions for DCS of a complete CCD camera
//*
//*   SYNOPSIS
//*       #include <ccdCAMERADCS.class>
//*
//*   PARENT CLASS
//*       None
//*
//*   DESCRIPTION
//*       This class defines the attributes for any CCD camera DCS part only
//*       The camera cannot be used in stand-alone
//*
//*   CAUTIONS
//*       none
//*
//*   EXAMPLES
//*
//*   SEE ALSO
//*       ccdCAMERA.class
//*
//*   BUGS
//*
//*------------------------------------------------------------------------
```

```
#ifndef ccdCAMERADCS_CLASS
#define ccdCAMERADCS_CLASS

#include "ccdLCUWSAPPLICATION.class"
#include "ccdEXPOSURES.class"
#include "ccdIMAGES.class"
#include "ccdDETECTOR.class"
#include "ccdREADOUT.class"
#include "ccdSHUTTER.class"
#include "ccdTEMPERATURE.class"
#include "ccdTELEMETRY.class"
#include "ccdCRYOSTAT.class"
#include "ccdACE.class"
#include "ccdWORLDCOORD.class"

CLASS ccdLCUWSAPPLICATION ccdCAMERADCS
BEGIN
    ATTRIBUTE Bytes16        description      // Location, purpose, etc.
    ATTRIBUTE Bytes16        id               // unique identification code
    ATTRIBUTE Bytes16        date             // commissioning date
    ATTRIBUTE Bytes32        installed        // installation date
    ATTRIBUTE Bytes8         logMask          // Fits Log Mask
    ATTRIBUTE ccdEXPOSURES   exposures        // exposures info
    ATTRIBUTE ccdIMAGES      images           // images info
    ATTRIBUTE ccdDETECTOR    detector         // detector data
    ATTRIBUTE ccdREADOUT     readout          // readout sub-system
    ATTRIBUTE ccdSHUTTER     shutter          // shutter sub-system
    ATTRIBUTE ccdTEMPERATURE temperature      // temperature sub-system
    ATTRIBUTE ccdTELEMETRY   telemetry        // telemetry sub-system
    ATTRIBUTE ccdCRYOSTAT    cryostat         // cryostat characteristics
    ATTRIBUTE ccdACE         ace              // ACE characteristics
    ATTRIBUTE ccdWORLDCOORD  wcs              // World Coordinates
END

#endif /*!ccdCAMERADCS_CLASS*/
// *___oOo___*
```

3.6.2 ccdCAMERA

```
//*******************************************************************
//* E.S.O. - VLT project
//*
//* "@(#) $Id: ccdCAMERA.class,v 2.4 1998/09/18 08:56:57 vltsccm Exp $"
//*
//* who          when      what
//* --------     --------  ----------------------------------------------
//* A.Longinotti 18/05/95  created
//*

//*********************************************************************
//*   NAME
//*   ccdCAMERA - Common class definitions for a CCD stand-alone camera
//*
//*   SYNOPSIS
//*       #include <ccdCAMERA.class>
//*
//*   PARENT CLASS
//*       None
//*
//*   DESCRIPTION
//*       This class defines the attributes for any CCD camera system
//*       stand-alone
//*
//*   CAUTIONS
//*       none
//*
//*   EXAMPLES
//*
//*   SEE ALSO
//*
//*   BUGS
//*
//*----------------------------------------------------------------------
#ifndef ccdCAMERA_CLASS
#define ccdCAMERA_CLASS

#include "ccdCAMERADCS.class"
#include "ccdOBSERVATIONS.class"

CLASS ccdCAMERADCS ccdCAMERA
BEGIN
    ATTRIBUTE ccdOBSERVATIONS observations     // observations info
END

#endif /*!ccdCAMERA_CLASS*/
// *___oOo___*
```

3.6.3 ccd.db(l)

NAME

```
ccd.db - Template of Branch Configuration File for a CCD camera
```

DESCRIPTION

```
This template can be used to generate a complete branch for a CCD
camera.
The user should just include this file in his .db file and define
the following macros:
CCDNAME  name of the camera. It becomes the alias of the CCD DB root
CCDROOT  absolute path of the CCD DB root
```

EXAMPLES

```
#define CCDNAME ccdtest                \\ camera name
#define CCDROOT :Appl_data:ccdtest     \\ CCD DB root
#include "ccd.db"
```

```
- - - - - -
Last change:  21/09/98-11:47
```

3.6.4 Example for DATABASE.db file

```
//*****************************************************************************
//* E.S.O. - VLT project
//*
//* "@(#) $Id: DATABASE.db,v 1.60 1997/11/12 14:12:08 vltsccm Exp $"
//*
//* who       when      what
//* --------  --------  ----------------------------------------------
//* alongino  11/04/97  created as templated for CCD stand-alone system
//*

//************************************************************************
// VLT COMMON SOFTWARE
//************************************************************************

// Loads classes definition from standard file
#include "CCS.db"

//
// USER.db contains necessary points for CCS, but requiring editing to match
// installed configuration.
#include "USER.db"


//************************************************************************
// APPLICATION SOFTWARE
//************************************************************************

//
// Load branches needed by your application

// Replace "myccd" with actual camera name
#define CCDNAME myccd
#define CCDROOT :myccd
#include "ccd.db"
// If you want a second CCD:
// 1) Replace "myccd2" with actual camera name
// 2) Uncomment next lines
//#undef CCDNAME
//#undef CCDROOT
//#define CCDNAME myccd2
//#define CCDROOT :myccd2
//#include "ccd.db"

// ___oOo___
```

3.6.5 Example for USER.db file

```
//*****************************************************************************
//* E.S.O. - VLT project
//*
//* "@(#) $Id: USER.db,v 1.60 1997/11/12 14:12:08 vltsccm Exp $"
//*
//* who       when      what
//* --------  --------  -----------------------------------------------
//* alongino  11/04/97  created as template for CCD stand-alone system
//*
//**************************************************************************

// Loads classes definition from standard file
#include "CCS.db"

//
// **************************************************************************
// Points needed by LCC, but requiring editing to match user installed config.
// **************************************************************************
//

//
//**************************************************************************
//   SCAN SYSTEM points
//**************************************************************************
//

//
// Replace hereafter "wmyccd" by the name of the Workstation environment
// the LCU shall report to.
POINT "<VLT scan dev>" "LAN:wmyccd"
BEGIN
        ALIAS   "wmyccd"
END

// ___oOo___
```

3.7 Setup files

This section contains an example of a complete CCD DCS setup file.

3.7.1 ccdSetupComplete.det

```
PAF.HDR.START;                    # Start of PAF Header
PAF.TYPE         "Detector Setup"; # Type of PAF
PAF.ID           "        ";      # ID for PAF
PAF.NAME         "        ";      # Name of PAF
PAF.DESC         "        ";      # Short description of PAF
PAF.CRTE.NAME    "        ";      # Name of creator
PAF.CRTE.DAYTIM  "        ";      # 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
DET.EXP.TYPE     "Normal  ";      # Exposure type
DET.EXP.NREP                  1;  # number of repeated exposures
DET.READ.CLKIND               1;  # Index of clock pattern used
DET.OUT1.GAININD              8;  # Gain index for output
DET.FRAM.FITSMTD              2;  # Data storage method
DET.FRAM.FITSUNC "myImage.fits";  # File name for uncompressed data
DET.DISPLAY                   0;  # real-time image display frame ID
DET.WIN1.BINX                 1;  # Binning factor along X
DET.WIN1.BINY                 1;  # Binning factor along Y
DET.FRAM.SAMPLE               1;  # Image sampling on workstation
DET.FRAM.TYPE    "Normal  ";      # Type of frame
DET.EXP.TIMEREP            0.00;  # Time between two repeated exposures
DET.READ.SIMIMG  "        ";      # Simulation file name
DET.FRAM.XFERSYN            "T";  # If T, image transfer occurs synchronously
DET.EXP.WIPETIM               0;  # wipe before starting exposure in a loop
DET.WIN1.ST                 "F";  # If T, window enabled
DET.WIN1.STRX               200;  # Lower left pixel in X
DET.WIN1.NX                  20;  # # of pixels along X
DET.WIN1.STRY               300;  # Lower left pixel in Y
DET.WIN1.NY                  20;  # # of pixels along Y
DET.WIN1.CENTROID "none    ";     # type of centroiding calculation
DET.WIN1.REFX             50.00;  # X position of reference
DET.WIN1.REFY            100.00;  # Y position of reference
DET.WIN1.MINMAX             "F";  # image extrema search performed
DET.WIN1.IPFUNC  "None    ";      # user defined function name
DET.WIN1.IPBUFF  "None    ";      # buffer variable in user function
DET.WIN1.BIAS               "F";  # bias frame subtraction performed
DET.WIN1.FLATF              "F";  # flat field division performed
DET.WIN1.BACKGND       0.000000;  # sky background
DET.WIN1.THRMIN        0.000000;  # lower threshold
DET.WIN1.IPLLX                0;  # X offset from bottom left for proc. sub-window
DET.WIN1.IPLLY                0;  # Y offset from bottom left for proc. sub-window
DET.WIN1.IPURX                0;  # X offset from top right for proc. sub-window
DET.WIN1.IPURY                0;  # Y offset from top right for proc. sub-window
DET.WIN2.ST                 "F";  # If T, window enabled
DET.WIN2.STRX                 1;  # Lower left pixel in X
DET.WIN2.NX                  20;  # # of pixels along X
DET.WIN2.STRY                 1;  # Lower left pixel in Y
DET.WIN2.NY                  20;  # # of pixels along Y
```

```
DET.WIN2.CENTROID "none    ";  # type of centroiding calculation
DET.WIN2.REFX             1.00; # X position of reference
DET.WIN2.REFY             1.00; # Y position of reference
DET.WIN2.MINMAX            "F"; # image extrema search performed
DET.WIN2.IPFUNC  "None    ";   # user defined function name
DET.WIN2.IPBUFF  "None    ";   # buffer variable in user function
DET.WIN2.BIAS              "F"; # bias frame subtraction performed
DET.WIN2.FLATF             "F"; # flat field division performed
DET.WIN2.BACKGND     0.000000; # sky background
DET.WIN2.THRMIN      0.000000; # lower threshold
DET.WIN2.IPLLX               0; # X offset from bottom left for proc. sub-window
DET.WIN2.IPLLY               0; # Y offset from bottom left for proc. sub-window
DET.WIN2.IPURX               0; # X offset from top right for proc. sub-window
DET.WIN2.IPURY               0; # Y offset from top right for proc. sub-window
DET.WIN1.NDIT                1; # # of subintegrations
DET.WIN1.ASUIT1            "T"; # All UITi as UIT1 else as defined in UITi
DET.READ.SHIFTYP "list    ";   # Line shift type
DET.READ.SHIFT1              0; # Lines shifted between integration
DET.WIN1.UIT1            1.000; # user defined subintegration time

# --- oOo ---
```

3.8 Image files

This section contains examples of:

1. FITS standard keywords written by CCD DCS in the FITS header
2. FITS hierarchical keywords written by CCD DCS in a separate file.

3.8.1 Example of FITS header standard keywords written by CCD DCS

```
SIMPLE  =                    T          / Standard FITS format (NOST-100.0)
BITPIX  =                   16          / # of bits storing pix values
NAXIS   =                    2          / # of axes in frame
NAXIS1  =                  592          / # pixels/axis
NAXIS2  =                  578          / # pixels/axis
ORIGIN  = 'ESO     '                    / European Southern Observatory
DATE    = '1997-11-19T10:18:39.535'     / UT date when this file was written
CRVAL1  =                  1.0          / value of X ref pixel
CRPIX1  =                  1.0          / X Ref. pixel of center of rotation
CDELT1  =                  1.0          / Binning factor
CTYPE1  = 'PIXEL   '                    / Pixel coordinate system
CRVAL2  =                  1.0          / value of X ref pixel
CRPIX2  =                  1.0          / X Ref. pixel of center of rotation
CDELT2  =                  1.0          / Binning factor
CTYPE2  = 'PIXEL   '                    / Pixel coordinate system
BSCALE  =                  1.0          / pixel=FITS*BSCALE+BZERO
BZERO   =                  0.0          / pixel=FITS*BSCALE+BZERO
MJD-OBS =       50771.42960247          / MJD start (1997-11-19T10:18:37.653)
DATE-OBS= '1997-11-19T10:18:37.653'     / Date of observation
EXPTIME =               1.0004          / Total integration time
```

3.8.2 Example of FITS hierarchical keywords written by CCD DCS

```
HIERARCH ESO DET ID           = 'CCD ACE - Rev 2.5' / Detector system Id
HIERARCH ESO DET NAME         = 'ccdUvSR - tccd_ace$20' / Name of system
HIERARCH ESO DET DATE         = '1997-11-19T10:16:23.000' / Installation date
HIERARCH ESO DET DID          = 'ESO-VLT-DIC.CCDDCS - Rev 2.5' / Ddictionary
HIERARCH ESO DET BITS         =           12 / Bits per pixel readout
HIERARCH ESO DET SOFW MODE    = 'Normal  '   / CCD sw operational mode
HIERARCH ESO DET CHIP FRMXFER=             T / chip is of frame transfer type
HIERARCH ESO DET CHIP1 ID     = 'tccd$58 '   / Detector chip identification
HIERARCH ESO DET CHIP1 NAME   = 'djo_LARGE'  / Detector chip name
HIERARCH ESO DET CHIP1 X      =            1 / X location in array
HIERARCH ESO DET CHIP1 Y      =            1 / Y location in array
HIERARCH ESO DET CHIP1 NX     =          592 / # of pixels along X
HIERARCH ESO DET CHIP1 NY     =          578 / # of pixels along Y
HIERARCH ESO DET CHIP1 PSZX   =         22.0 / Size of pixel in X
HIERARCH ESO DET CHIP1 PSZY   =         22.0 / Size of pixel in Y
HIERARCH ESO DET EXP NO       =           11 / Unique exposure ID number
HIERARCH ESO DET EXP TYPE     = 'Normal  '   / Exposure type
HIERARCH ESO DET EXP DUMDIT   =            3 / # of dummy readouts
HIERARCH ESO DET EXP RDTTIME =         0.916 / image readout time
HIERARCH ESO DET EXP XFERTIM =         1.032 / image transfer time
HIERARCH ESO DET READ MODE    = 'normal  '   / Readout method
HIERARCH ESO DET READ CLOCK   = 'acetecL.clk' / Readout clock pattern used
HIERARCH ESO DET READ SPEED   = 'fast    '   / Readout speed
```

```
HIERARCH ESO DET READ PIXTIME=          2.0 / Pixel digitalization time [us]
HIERARCH ESO DET OUT1 NAME   = 'A       '   / Description of output
HIERARCH ESO DET OUT1 CHIP   =            1 / index of chip it belongs to
HIERARCH ESO DET OUT1 X      =            1 / X location of output
HIERARCH ESO DET OUT1 Y      =            1 / Y location of output
HIERARCH ESO DET OUT1 NX     =          592 / valid pixels along X
HIERARCH ESO DET OUT1 NY     =          578 / valid pixels along Y
HIERARCH ESO DET OUT1 PRSCX  =            0 / Prescan region in X
HIERARCH ESO DET OUT1 OVSCX  =            0 / Overscan region in X
HIERARCH ESO DET OUT1 CONAD  =         5.00 / Conversion from electrons to ADUs
HIERARCH ESO DET OUT1 GAININD=            8 / Gain index for output
HIERARCH ESO DET FRAM ID     =            1 / Image sequencial number
HIERARCH ESO DET FRAM TYPE   = 'Normal  '   / Type of frame
HIERARCH ESO DET WINDOWS     =            1 / # of windows readout
HIERARCH ESO DET WIN1 STRX   =            1 / Lower left pixel in X
HIERARCH ESO DET WIN1 STRY   =            1 / Lower left pixel in Y
HIERARCH ESO DET WIN1 NX     =          592 / # of pixels along X
HIERARCH ESO DET WIN1 NY     =          578 / # of pixels along Y
HIERARCH ESO DET WIN1 BINX   =            1 / Binning factor along X
HIERARCH ESO DET WIN1 BINY   =            1 / Binning factor along Y
HIERARCH ESO DET WIN1 NDIT   =            1 / # of subintegrations
HIERARCH ESO DET WIN1 UIT1   =        1.000 / user defined subintegration time
HIERARCH ESO DET WIN1 DIT1   =       1.0004 / actual subintegration time
HIERARCH ESO DET WIN1 DKTM   =       1.0004 / Dark current time
HIERARCH ESO DET TMP1 ID     = 'House   '   / ID of temperature sensor
HIERARCH ESO DET TMP1 START  =       295.95 / Temperature at exposure start
HIERARCH ESO DET TMP1 END    =       295.98 / Temperature at exposure completion
HIERARCH ESO DET TMP2 ID     = 'Chip    '   / ID of temperature sensor
HIERARCH ESO DET TMP2 START  =       229.07 / Temperature at exposure start
HIERARCH ESO DET TMP2 END    =       229.00 / Temperature at exposure completion
```

3.9 Log files

This section contains examples of:

1. Configuration and operational logs in FITS format from a scientific CCD camera
2. Configuration and operational logs in FITS format from a technical CCD camera

3.9.1 Example of FITS log from a Scientific CCD

```
16:17:55>-START DET SOFW / CCD control sw started [SUSI]
16:17:55> DET SOFW ID = $Revision: 1.84 $ / CCD control program name-version [SUSI]
16:17:55> DET SOFW MODE = ACE simulated / CCD sw operational mode [SUSI]
16:17:55> DET STATE = LOADED / CCD sw state [SUSI]
16:17:55>/UNFORSEEN: LCU & WS times are not aligned [SUSI]
16:17:55>/RECOVERY: LCU time set to 1997-11-05T16:17:55 [SUSI]
16:17:55>/Time Accuracy = 10 ms [SUSI]
16:18:28>-CHANGE DET STATE / CCD sw state change [SUSI]
16:18:28> DET SHUT TYPE = motorDriven / type of shutter [SUSI]
16:18:28> DET SHUT ID = Susi / Shutter unique identifier [SUSI]
16:18:28> DET TELE NO = 72 / # of sources active [SUSI]
16:18:28> DET TELE TOLE = 0.10 / Telemetry variation tolerance [SUSI]
16:18:28> DET TLM1 NAME = Bias_1 / Description of telemetry param. [SUSI]
16:18:28> DET TLM1 ID = AGND / ID of telemetry sensor [SUSI]
16:18:28> DET TLM2 NAME = Bias_2 / Description of telemetry param. [SUSI]
16:18:28> DET TLM2 ID = TENVOLT / ID of telemetry sensor [SUSI]
.....
16:18:28> DET TLM71 NAME = Clock_24_L / Description of telemetry param. [SUSI]
16:18:28> DET TLM71 ID = CLK23_LOW / ID of telemetry sensor [SUSI]
16:18:28> DET TLM72 NAME = Clock_24_H / Description of telemetry param. [SUSI]
16:18:28> DET TLM72 ID = CLK23_HIGH / ID of telemetry sensor [SUSI]
16:18:28>/No Bias Frame found [SUSI]
16:18:28>/No Flat Field found [SUSI]
16:18:28> DET STATE = STANDBY / CCD sw state [SUSI]
16:18:28>-CHANGE DET STATE / CCD sw state change [SUSI]
16:18:29>/World Coordinates: Enabled [SUSI]
16:18:29> DET STATE = ONLINE / CCD sw state [SUSI]
.....
16:20:51>-START DET EXP / Start exposure [SUSI]
16:20:51> DET EXP NO = 211 / Unique exposure ID number [SUSI]
16:20:51>-START DET CHIP.WIPE / CCD start wipe action [SUSI]
16:20:51> DET CHIP.WIPE STATUS = DONE / CCD wipe completed [SUSI]
16:20:51>-OPEN DET SHUT / CCD shutter open [SUSI]
16:20:52>-PAUSE DET EXP / Pause exposure [SUSI]
16:20:52>-CLOSE DET SHUT / CCD shutter closed [SUSI]
16:20:57>-RESUME DET EXP / Continue exposure [SUSI]
16:20:57>-OPEN DET SHUT / CCD shutter open [SUSI]
16:21:27>-CLOSE DET SHUT / CCD shutter closed [SUSI]
16:21:27>-START DET CHIP.READ / start readout of CCD chip [SUSI]
16:21:27>-START DET TRANS / CCD data transfer started [SUSI]
16:21:28> DET CHIP.READ STATUS = DONE / CCD chip readout completed [SUSI]
16:21:28> DET TRANS STATUS = DONE / CCD data transfer completed [SUSI]
16:21:28> DET EXP STATUS = DONE / Exposure completed [SUSI]
.....
11:38:24>-CHANGE DET STATE / CCD sw state change [SUSI]
11:38:26> DET STATE = OFF / CCD sw state [SUSI]
```

3.9.2 Example of FITS log from a Technical CCD

```
11:07:27>-START DET SOFW / CCD control sw started [UT1AG]
11:07:27> DET SOFW ID = CCD Rev 2.5 / CCD control program name-version [UT1AG]
11:07:27> DET SOFW MODE = Normal / CCD sw operational mode [UT1AG]
11:07:27> DET STATE = LOADED / CCD sw state [UT1AG]
11:07:20>/Time Accuracy = given by TIM [UT1AG]
11:07:30>-CHANGE DET STATE / CCD sw state change [UT1AG]
11:07:31>/ACE program 'acetec.btl' loaded from /INTROOT [UT1AG]
11:07:31>/DSP program 'acetecL.clk' loaded from /INTROOT [UT1AG]
11:07:34> DET TELE NO = 48 / # of sources active [UT1AG]
11:07:34> DET TELE TOLE = 0.10 / Telemetry variation tolerance [UT1AG]
11:07:34> DET TLM1 NAME = Bias_1 / Description of telemetry param. [UT1AG]
11:07:34> DET TLM1 ID = UDC0 / ID of telemetry sensor [UT1AG]
11:07:34> DET TLM2 NAME = Bias_2 / Description of telemetry param. [UT1AG]
11:07:34> DET TLM2 ID = UDC1_PD / ID of telemetry sensor [UT1AG]
.....
11:07:34> DET TLM47 NAME = Clock_16_L / Description of telemetry param. [UT1AG]
11:07:34> DET TLM47 ID = CLK15_L_RG / ID of telemetry sensor [UT1AG]
11:07:34> DET TLM48 NAME = Clock_16_H / Description of telemetry param. [UT1AG]
11:07:34> DET TLM48 ID = CLK15_H_RG / ID of telemetry sensor [UT1AG]
11:07:34> DET TEMP NO = 2 / # of sensors active [UT1AG]
11:07:34> DET TEMP TOLE = 0.50 / Temperature variation tolerance [UT1AG]
11:07:34> DET TMP1 NAME = House_1 / Description of temperature sensor [UT1AG]
11:07:34> DET TMP1 ID = House / ID of temperature sensor [UT1AG]
11:07:34> DET TMP2 NAME = Chip_1 / Description of temperature sensor [UT1AG]
11:07:34> DET TMP2 ID = Chip / ID of temperature sensor [UT1AG]
11:07:34>/No Bias Frame found [UT1AG]
11:07:34>/No Flat Field found [UT1AG]
11:07:34> DET STATE = STANDBY / CCD sw state [UT1AG]
11:07:35>-CHANGE DET STATE / CCD sw state change [UT1AG]
11:07:35> DET TEMP INT = 60.0 / temperature reading period [UT1AG]
11:07:35>-START DET TEMP / Temperature monitoring is active [UT1AG]
11:07:35> DET TMP1 PARM = 281.22 / Current value temperature sensor [UT1AG]
11:07:35> DET TMP2 PARM = 232.18 / Current value temperature sensor [UT1AG]
11:07:35>/World Coordinates: Enabled [UT1AG]
11:07:35> DET STATE = ONLINE / CCD sw state [UT1AG]
.....
11:18:37>-START DET EXP / Start exposure [UT1AG]
11:18:37> DET EXP NO = 11 / Unique exposure ID number [UT1AG]
11:18:37> DET TMP1 START = 265.40 / Temperature at exposure start [UT1AG]
11:18:37> DET TMP2 START = 235.19 / Temperature at exposure start [UT1AG]
11:18:37>-START DET CHIP WIPE / CCD start wipe action [UT1AG]
11:18:37> DET CHIP WIPE STATUS = DONE / CCD wipe completed [UT1AG]
11:18:38>-START DET CHIP READ / start readout of CCD chip [UT1AG]
11:18:38>-START DET TRANS / CCD data transfer started [UT1AG]
11:18:39> DET CHIP READ STATUS = DONE / CCD chip readout completed [UT1AG]
11:18:40> DET TMP1 END = 263.39 / Temperature at exposure completion [UT1AG]
11:18:40> DET TMP2 END = 228.01 / Temperature at exposure completion [UT1AG]
11:18:40> DET TRANS STATUS = DONE / CCD data transfer completed [UT1AG]
11:18:40> DET EXP STATUS = DONE / Exposure completed [UT1AG]
.....
11:22:51> DET EXP NO = 141 / Unique exposure ID number [UT1AG]
11:22:51> DET EXP REPEAT = infinite / number of repeated exposures [UT1AG]
11:22:51>-START DET CHIP WIPE / CCD start wipe action [UT1AG]
11:22:51> DET CHIP WIPE STATUS = DONE / CCD wipe completed [UT1AG]
11:23:01>-STOP DET EXP / Stop integration / loop [UT1AG]
11:23:01> DET EXP NREP = 428 / number of repeated exposures [UT1AG]
11:23:01> DET EXP STATUS = DONE / Exposure completed [UT1AG]
.....
11:43:57>/UNFORSEEN: TMP1 (Chip_1) : 238.4 K - Out of range 200.0 to 238.0 K [UT1AG]
11:44:54> DET TMP1 PARM = 269.91 / Current value temperature sensor [UT1AG]
.....
```

```
11:46:15>-CHANGE DET STATE / CCD sw state change [UT1AG]
11:46:15>-STOP DET TEMP / Temperature monitoring has been stopped [UT1AG]
11:46:16> DET STATE = OFF / CCD sw state [UT1AG]
```

3.10 Configuration files

This section contains examples of files needed to configure the environment where the CCD software is supposed to run.
On LCU:

1. Example of bootScript

3.10.1 Example for LCU bootScript file

```
#! VxWorks
#******************************************************************************
# E.S.O. - VLT project
#
# "@(#) $Id: bootScript,v 1.60 1997/11/12 14:12:09 vltsccm Exp $"
#
# THIS IS A GENERATED FILE - DO NOT EDIT (unless you know how)
#
# Created by
#   User: pduhoux
#   Host: te49
#   Date: Fri Nov  7 08:31:58 MEZ 1997
#   Tool: vccConfigLcu
#   Proc: vccConfigLcuGenerateBootScript 2.7
#

#:::   VCCDATA   :::
# vLcuEnv:      lte53
# vLcuHost:     te53
# vLcuIPaddr:   134.171.12.189
# vLcuHost0:    te53
# vLcuIPaddr0:  134.171.12.189
# vLcuHost1:
# vLcuIPaddr1:
# vLcuHost2:
# vLcuIPaddr2:
# vLcuTCPport:  2160
# vLcuCpu:      MC68040
# vLcuType:     68k
# vLcuBsp:      mv167
# vBwsEnv:      wccd
# vBwsHost:     te49
# vBwsIPaddr:   134.171.12.184
# vBwsTCPport:  2301
# vBwsCpu:      hppa
# vBwsType:     hp9700
# vGateway:
# BOOTROOT:     /diskd/vlt/data
# BOOTHOME:     /diskd/vlt/data/ENVIRONMENTS/lte53
# VXROOT:       /diskb/vw5.3/target
# vVltHost:     te49
# vVltIPaddr:   134.171.12.184
# VLTROOT:      /diskd/vlt/NOV97
# vIntHost:     te49
# vIntIPaddr:   134.171.12.184
# INTROOT:      /diskb/introot/pduhoux/introot
```

```
# vModHost:     te49
# vModIPaddr:   134.171.12.184
# MODROOT:
# vSysmodlist:  lcudrv lculog lqs tim lcc cai scan b016
# vUsrmodlist:  ntp ccd
# vDevlist:     {? /tim}
#

#:::   ROOT   :::
putenv "BOOTROOT=/diskd/vlt/data"
putenv "BOOTHOME=/diskd/vlt/data/ENVIRONMENTS/lte53"
putenv "VLTROOT=/VLTROOT"
putenv "INTROOT=/INTROOT"
#putenv "MODROOT=/MODROOT"

#:::   NETWORK   :::
nfsIdSet 138 /* vx */
nfsAuthUnixSet "te49",138,300 /* vx,vlt */
nfsMount "te49","/diskd/vlt/data",0
hostAdd "te49","134.171.12.184"
nfsAuthUnixSet "te49",138,300 /* vx,vlt */
nfsMount "te49","/diskd/vlt/NOV97","/VLTROOT"
hostAdd "te49","134.171.12.184"
nfsAuthUnixSet "te49",138,300 /* vx,vlt */
nfsMount "te49","/diskb/introot/pduhoux/introot","/INTROOT"

#:::   LCUBOOT   :::
cd getenv("VLTROOT")
ld 0,1,"vw/bin/MC68040/lcuboot"
lcubootLogInit

#:::   ENVIRONMENT   :::
putenv "CPU=MC68040"
putenv "LOCALENV=lte53"
putenv "LOCALTCPPORT=2160"
putenv "HOSTENV=wccd"
putenv "HOSTTCPPORT=2301"
putenv "LOCALHOST=te53"
putenv "LOCALIPADDRESS=134.171.12.189"
#putenv "LOCALHOST_1="
#putenv "LOCALIPADDRESS_1="
#putenv "LOCALHOST_2="
#putenv "LOCALIPADDRESS_2="
lcubootAutoEnvInit /* set rest of variables automatically */

#:::   MODULES   :::
lcubootAutoLoad 1,"lcudrv",0
lcubootAutoLoad 1,"lculog",0
lcubootAutoLoad 1,"lqs",0
# auto-install with defered loading for driver: tim
lcubootAutoLoad 1,"lcc",0
lcubootAutoLoad 1,"cai",0
lcubootAutoLoad 1,"scan",0
lcubootAutoLoad 1,"b016",0
lcubootAutoLoad 1,"ntp",0
lcubootAutoLoad 1,"ccd",0
# Initialize System-Modules:
lcubootAutoCdBoot 1,"lcudrv.boot"
```

```
< lcudrv.boot
lcubootAutoCdBoot 1,"lculog.boot"
< lculog.boot
lcubootAutoCdBoot 1,"lqs.boot"
< lqs.boot
lcubootAutoCdBoot 1,"tim.boot"
< tim.boot
lcubootAutoCdBoot 1,"lcc.boot"
< lcc.boot
lcubootAutoCdBoot 1,"cai.boot"
< cai.boot
lcubootAutoCdBoot 1,"scan.boot"
< scan.boot
lcubootAutoCdBoot 1,"b016.boot"
< b016.boot

#:::   DEVICES   :::
#lcubootAutoDevCheck "/tim",-1

#:::   USER   :::
# Initialize User-Modules:
lcubootAutoCdBoot 1,"ntp.boot"
< ntp.boot
lcubootAutoCdBoot 1,"ccd.boot"
< ccd.boot
# Execute User-Script:
cd getenv("BOOTHOME")
#< userScript

#:::   TERMINATION   :::
taskDelay 100
lcubootAutoSpawn 1,"tLccResumeInit",80,0x18,40000,"lccResumeInit"
lcubootAutoExec 1,"lccWaitInit"
lcubootLogFinish

# ___END_OF_BOOT_SCRIPT___
```

3.11 Error message files

3.11.1 ccd_ERRORS

```
#
# Version Id. is specified at the end of the file
#
# Error Definition File    Created on  Oct  9 12:41:59 1997
#
# This file has been generated by a utility
#
#  !!!!!!!!!!!  DO NOT MANUALLY EDIT THIS FILE  !!!!!!!!!!!
#
############################################################



1 W {}
ccdERR_BUSY : Cannot process command '%.16s' - Process already executing
ccdERR_BUSY.hlp
2 S {}
ccdERR_CMD_EXECUTE : Failed executing command '%.16s' - %.80s
ccdERR_CMD_EXECUTE.hlp
3 W {}
ccdERR_CMD_IGNORED : Command '%.16s' ignored - %.80s
ccdERR_CMD_IGNORED.hlp
4 S {}
ccdERR_CMD_UNKNOWN : Unknown command '%.16s'
ccdERR_CMD_UNKNOWN.hlp
5 W {}
ccdERR_CMD_WARNING : Command '%.16s' returned warning - %.80s
ccdERR_CMD_WARNING.hlp
6 F {}
ccdERR_DB_ROOT : Wrong root database point '%.80s'
ccdERR_DB_ROOT.hlp
7 F {}
ccdERR_DB_CONSISTENCY : Expected %.80s - Check module version with system manager
ccdERR_DB_CONSISTENCY.hlp
8 F {}
ccdERR_DB_READ : Failed to read database attribute '%.80s'
ccdERR_DB_READ.hlp
9 F {}
ccdERR_DB_WRITE : Failed to write database attribute '%.80s'
ccdERR_DB_WRITE.hlp
10 F {}
ccdERR_ENV_UNKNOWN : Unknown environment variable '%.80s'
ccdERR_ENV_UNKNOWN.hlp
11 S {}
ccdERR_INVALID_ARGUMENT : Invalid argument '%.80s' - %.80s
ccdERR_INVALID_ARGUMENT.hlp
12 F {}
ccdERR_NOT_IMPLEMENTED : Feature '%.80s' not implemented
ccdERR_NOT_IMPLEMENTED.hlp
13 F {}
ccdERR_OPEN_FILE : Cannot open file '%.80s' - %.80s
ccdERR_OPEN_FILE.hlp
```

```
14 F {}
ccdERR_OPMODE : Illegal operational mode '%d' - %.80s
ccdERR_OPMODE.hlp
15 S {}
ccdERR_OPSTATE : Illegal operational state '%.20s' - %.80s
ccdERR_OPSTATE.hlp
16 F {}
ccdERR_SETUP : Inconsistent setup - Check parameters
ccdERR_SETUP.hlp
17 S {}
ccdERR_STATUS : Illegal %.20s status - %.80s
ccdERR_STATUS.hlp
18 S {}
ccdERR_TIMEOUT : Timeout waiting for %.80s
ccdERR_TIMEOUT.hlp
19 S {}
ccdERR_PARAM_INVALID : Invalid parameter '%.80s' - %80s
ccdERR_PARAM_INVALID.hlp
20 S {}
ccdERR_PARAM_RANGE : Parameter out of range '%.80s' = %.80s - %.80s
ccdERR_PARAM_RANGE.hlp
101 F {}
ccdERR_CCS_CALL : CCS service call failed : %.80s
ccdERR_CCS_CALL.hlp
102 F {}
ccdERR_CD_SET : Error setting current working directory to %.256s
ccdERR_CD_SET.hlp
103 S {}
ccdERR_CONFIGURATION : %.80s
ccdERR_CONFIGURATION.hlp
104 F {}
ccdERR_CMD_LIST_FULL : List of commands waiting for termination of image transfer
full.
ccdERR_CMD_LIST_FULL.hlp
105 F {}
ccdERR_CMD_LIST_NOT_FOUND : %.20s not found in the list of commands waiting for
termination of image transfer.
ccdERR_CMD_LIST_NOT_FOUND.hlp
106 F {}
ccdERR_ENV : Environment not completely defined or not properly set.
ccdERR_ENV.hlp
107 F {}
ccdERR_ENV_GET : Failed to get environment information for program %20s
ccdERR_ENV_GET.hlp
108 F {}
ccdERR_ENV_VAR : Environment variable %.20s not defined. Needed to proceed.
ccdERR_ENV_VAR.hlp
109 F {}
ccdERR_ENV_IT : Failed to get environment information for image transfer sub-sys-
tem.
ccdERR_ENV_IT.hlp
110 S {}
ccdERR_FITS_FILE : Error during operations on FITS file %.256s
ccdERR_FITS_FILE.hlp
111 S {}
ccdERR_FITS_KEYWORD : Error with FITS keyword %.20s
ccdERR_FITS_KEYWORD.hlp
112 S {}
```

```
ccdERR_IMAGE : Error while transfering next image
ccdERR_IMAGE.hlp
113 S {}
ccdERR_IMAGE_INFO : Error retrieving image information
ccdERR_IMAGE_INFO.hlp
114 F {}
ccdERR_MSG_ADD : Error trying to allocate memory space for a new message.
ccdERR_MSG_ADD.hlp
115 F {}
ccdERR_MSG_ADD_CMD : Failed adding new command %.16s from process %d environment
%.20s to the list of pending messages
ccdERR_MSG_ADD_CMD.hlp
116 F {}
ccdERR_MSG_COMM : Failed in sending command %.16s to process %.20s environment
%.20s
ccdERR_MSG_COMM.hlp
117 F {}
ccdERR_MSG_NOT_FOUND : Message %.256s of type %d, ID %d from environment %.20s not
found in pending messages list
ccdERR_MSG_NOT_FOUND.hlp
118 S {}
ccdERR_MSG_RCV : Failed while getting new message
ccdERR_MSG_RCV.hlp
119 S {}
ccdERR_MSG_RPLY : Failed sending reply to command %.16s to process number %d en-
vironment %.20s
ccdERR_MSG_RPLY.hlp
120 F {}
ccdERR_NOT_INIT : System not initialised (see command INIT). Command %.16s cannot
be executed.
ccdERR_NOT_INIT.hlp
121 S {}
ccdERR_NOT_ONLINE : System not online (see command ONLINE). Command %.16s cannot
be executed
ccdERR_NOT_ONLINE.hlp
122 S {}
ccdERR_PIXELS : Wrong amount of data received from LCU: %d pixels instead of %d
ccdERR_PIXELS.hlp
123 S {}
ccdERR_RPLY : Error returned or timeout in reply from process %d environment %.20s
to command %.16s
ccdERR_RPLY.hlp
124 F {}
ccdERR_RPLY_EXEC : Failed executing action after reply to command %.16s sent by
process %d environment %.20s
ccdERR_RPLY_EXEC.hlp
125 F {}
ccdERR_RPLY_LENGTH : Unexpected length of reply message=%d. Expected %d bytes
ccdERR_RPLY_LENGTH.hlp
126 F {}
ccdERR_RPLY_SEND : Error trying to send a reply to command originator
ccdERR_RPLY_SEND.hlp
127 S {}
ccdERR_RPLY_UNEXP : Unexpected reply to command %.16s from process %d environment
%.20s
ccdERR_RPLY_UNEXP.hlp
128 S {}
ccdERR_RPLY_UNEXP_LAST : Unexpected last reply to command %.16s from process %d
```

```
environment %.20s
ccdERR_RPLY_UNEXP_LAST.hlp
129 F {}
ccdERR_SETUP_FILE : Error during %.80s operation with setup file %.256s
ccdERR_SETUP_FILE.hlp
130 F {}
ccdERR_SETUP_PAR : Missing parameter value for setup parameter %.80s
ccdERR_SETUP_PAR.hlp
131 S {}
ccdERR_SHARED_MEMORY : Failed during operation with shared memory %.20s
ccdERR_SHARED_MEMORY.hlp
132 F {}
ccdERR_STATE_EXEC : Failed executing action after reaching new state
ccdERR_STATE_EXEC.hlp
133 W {}
ccdERR_TIME : %.80s
ccdERR_TIME.hlp
134 S {}
ccdERR_NO_DISPLAY : No image currently displayed on frame ID %d
noHelp
135 S {}
ccdERR_TIMES_NOT_ALIGNED : LCU & WS times are not aligned - %.80s
ccdERR_TIMES_NOT_ALIGNED.hlp
201 S {}
ccdERR_TASK_SPAWN : Failed to spawn task '%.20s'
ccdERR_TASK_SPAWN.hlp
202 S {}
ccdERR_TASK_EXISTS : Task '%.20s' already exists
ccdERR_TASK_EXISTS.hlp
203 S {}
ccdERR_TASK_NAME : Cannot retrieve camera name from task '%.20s'
ccdERR_TASK_NAME.hlp
204 S {}
ccdERR_NO_MEMORY : Failed to allocate memory for %.80s
ccdERR_NO_MEMORY.hlp
205 S {}
ccdERR_FIND_SYMBOL : Cannot find symbol '%.20s' in global symbol table
ccdERR_FIND_SYMBOL.hlp
206 S {}
ccdERR_VX_CALL : Call to VxWorks function '%.20s' failed - %.80s
ccdERR_VX_CALL.hlp
207 S {}
ccdERR_SEMAPHORE : Failed to take semaphore '%.20s' - %.80s
ccdERR_SEMAPHORE.hlp
208 S {}
ccdERR_ENVIRONMENT : Expected valid environment variable '%.20s'
ccdERR_ENVIRONMENT.hlp
209 S {}
ccdERR_ACE_CALL : ACE call failed : %.80s
ccdERR_ACE_CALL.hlp
211 S {}
ccdERR_DCL_CALL : DCL call failed : %.80s
ccdERR_DCL_CALL.hlp
212 S {}
ccdERR_LCC_CALL : LCC service call failed : %.80s
ccdERR_LCC_CALL.hlp
213 S {}
ccdERR_INT_CALL : Failed to %.256s
```

```
ccdERR_INT_CALL.hlp
214 S {}
ccdERR_USR_CALL : Failed to process user function %.80s
ccdERR_USR_CALL.hlp
215 S {}
ccdERR_INIT_MODULE : Failed to initialize module '%.20s' - %.80s
ccdERR_INIT_MODULE.hlp
218 S {}
ccdERR_MSG_TYPE : %.80s
ccdERR_MSG_TYPE.hlp
219 S {}
ccdERR_FIND_FILE : Cannot find file '%.80s'
ccdERR_FIND_FILE.hlp
220 W {}
ccdERR_READ_FILE : Failed to read from file '%.80s'
ccdERR_READ_FILE.hlp
221 S {}
ccdERR_SEND_DATA : %.80s
ccdERR_SEND_DATA.hlp
226 W {}
ccdERR_FORMAT : Invalid command parameter format - %.80s
ccdERR_FORMAT.hlp
227 W {}
ccdERR_REPLY : Error reply received on %.80s command to %.80s
ccdERR_REPLY.hlp
228 W {}
ccdERR_STARTUP : %.80s
ccdERR_STARTUP.hlp
229 W {}
ccdERR_TIMER : %.80s
ccdERR_TIMER.hlp
230 S {}
ccdERR_SEND_COMMAND : Failed to send %.20s command to %.80s
ccdERR_SEND_COMMAND.hlp
231 W {}
ccdERR_IMAGE_LIST : %.80s
ccdERR_IMAGE_LIST.hlp
232 S {}
ccdERR_ID : Invalid %.20s ID = %.80s
ccdERR_ID.hlp
233 W {}
ccdERR_IMAGE_MISMATCH : %.80s
noHelp
234 W {}
ccdERR_TELEMETRY : %.256s
noHelp
235 W {}
ccdERR_TEMPERATURE : %.256s
noHelp
#
# "@(#) $Id: ccd_ERRORS,v 1.88 1997/11/12 15:04:11 vltsccm Exp $"
```

4 INSTALLATION GUIDE

The CCD sw is delivered as part of the VLT sw distribution kit.

This section describes step by step all what is needed to make the CCD sw ready to run and the test procedures to verify the correctness of the installation

4.1 GENERAL

The VLT CCD Control Software currently includes:

- the LCU-Transputer interface (LCU and ACE part).
- the CCD Software (WS, LCU and ACE parts for science and technical CCDs).

and all the relevant documentation in both printed and PostScript file format.

The distribution kit contains all the sources needed to regenerate the software and can be installed with or without the real camera hardware (ACE in simulation).

4.1.1 Copyright

See [7].

4.2 SUPPORTED CONFIGURATION

4.2.1 Hardware

The following components are needed by ACE based CCD systems, in addition to the VLT standard ones, listed in [7]:

WS

- one or more big hard-disk(s) for image storage.

LCU (not needed if LCU sw is simulated):

- a 19" VME bus chassis with power supply and **32-bit backplane**
- CPU board (currently MVME167), equipped with enough RAM to accommodate the whole software and at least one full image (minimum 8 MBytes; for scientific cameras: 32 Mbytes or more is recommended)
- Transputer Link Interface Board (INMOS B016). This board uses 24-bits addresses and therefore it must be connected to the CPU board over the backplane also on the P2 connector. This board is not needed if ACE sw is simulated (see section 2.4).
- ESO Transputer Link Adapter Board. For TCCDS systems it is connected to the Transputer Link Interface Board through a flat cable over P2 (cable delivered with the system). This board is not needed if ACE sw is simulated.
- (optional) ESO Time Interface Module (TIM) to connect to the Time Reference System (TRS). This board makes special usage of the P2 connector. It must therefore be plugged in the VME chassis in a slot where P2 is not connected to the backplane.

ACE (not needed if ACE sw is simulated)

- Complete ACE box

Note: More information about the hardware configuration of TCCDS systems can be found in [15].

4.2.2 Software

UNIX Operating System (see [7] for the types and versions supported).

VLT Common Software - OCT98, installed including RTAP and VxWorks according to [7])

(optional) VLT Real Time Display, installed according to [26].

(optional) INMOS D4305 package.
This package is needed to compile and generate the transputer part of the CCD sw, running on ACE. For all those configurations where this package is not available, no compilation takes place and executable files are directly installed. The installation procedure assumes that the INMOS package is available if the environment variable ISEARCH is defined.

(optional) Perimos DTM560 package.
This package is needed to compile and generate the DSP part of the CCD sw, running on ACE. For all those configurations where this package is not available, no compilation takes place and executable files are directly installed. The installation procedure assumes that the Perimos package is available if the environment variable DSP_ROOT is defined.

4.3 CONTENTS

The VLT CCD Control Software is delivered as part of the VLT sw release. See [7] for more information about the contents.

4.4 PROBLEM REPORTING/CHANGE REQUEST

See [7] for how to report problems/errors or suggested changes in software or documentation.

4.5 INSTALLATION

See [7].

4.6 CONFIGURATION

After successful installation of the software, the following steps will take you through the configuration of a complete CCD system. A CCD system requires a WS environment, an LCU and an ACE. Although existing environments can be used, we suggest to perform the configuration using new environments. Once you are familiar with that, the integration of the CCD software to an existing project is a straight forward process.

This section assumes that you master the process of creating/configuring environments, including the directory structure and available tools. (If not, please have a look to the configuration and verification section of [7]).

4.6.1 Environment variables.

Some environment variables are used on Workstation by the CCD sw and therefore must be defined; some of them, **marked with (*), are optional**: if they are not defined, the specified default value is taken.

HOST

name of the local Workstation. It has to be set to the value returned by the command *hostname*:

```
$ setenv HOST `hostname`
```

RTAPENV

name of the workstation CCS environment where CCD sw runs. Example:

```
$ setenv RTAPENV wmyccd
```

INS_ROOT

name of the root directory where data dictionaries, setup and configuration files are stored/retrieved (see [4] for the full directory structure). At least the root directory must exist, all the sub-directories needed by the CCD software, if not existing, will be created and populated in the next step (4.6.4).

Note that, although INS_ROOT has been defined only for instrumentation sw, it has to be defined and must exist also for technical CCD cameras.

CCDNAME

name of the CCD camera (**max. 7 characters**). Examples:

```
$ setenv CCDNAME myccd          right
$ setenv CCDNAME myccdred       wrong! (8 characters)
```

The alias name of the root point for the database CCD branch (see also point 4.6.3) **must be the same as the value of the environment variable CCDNAME**.

CCDLENV

name of the LCU environment where CCD sw runs. Example:

```
$ setenv CCDLENV lmyccd
```

CCDDXF (*)

name of the WS host receiving CCD image data (see [23]). If not defined, default is the contents of the environment variable HOST (same LAN used for messages and data). Example:

```
$ setenv CCDDXF texxAtm
```

INS_HOST (*)

name of the Workstation hosting the disk containing the INS_ROOT directory. If not defined, default is the local Workstation (see HOST above). Example:

```
$ setenv INS_HOST $HOST
```

INS_USER (*)

INS_ROOT branch for setup files search (see [4] for more details). If not defined, default is `SYSTEM`. Example:

```
$ setenv INS_USER SYSTEM
```

CCDDID (*)

Name of Data Interface Dictionary to be used (excluded prefix ESO-VLT-DIC.). If not defined, default is CCDDCS. Example:

```
$ setenv CCDDID FORS
```

The Data Interface Dictionary file must be stored in $INS_ROOT/SYSTEM/Dictionary.

The default Dictionary file (ESO-VLT-DIC.CCDDCS) is in $VLTROOT/config.

In the following sections we assume as example that the following definitions are done (all optional variables are defaulted):

```
$ setenv CCDNAME myccd
$ setenv RTAPENV wmyccd
$ setenv HOST `hostname`
$ setenv CCDLENV lmyccd
$ setenv INS_ROOT /diskc/myName/insroot
```

4.6.2 Single CCD stand-alone camera

Whenever the CCD camera is going to be used as a simple stand-alone instrument, and therefore WS and LCU environments can be configured to support the CCD sw only, the script *ccdEnvFromTeplate.sh* can be used. It creates and configures WS and LCU environments for one single CCD camera, taking the values of environment variables described in section 4.6.1.

```
$ ccdEnvFromTemplate.sh
```

In this case, the first part of sections 4.6.3 and 4.6.7 can be skipped (just perform the verification indicated in sub-sections 4.6.3.1 and 4.6.7.1).

4.6.3 WS Environment

1. Create a new environment as in [7]. Remember, the environment shall be named as in `$RTAPENV`. Do not forget to properly configure `/etc/services`, `/etc/$RTAPROOT/RtapEnvList (or $RTAPROOT/etc/RtapEnvList on older Rtap releases)`, etc.
   a. start *ccsei*
      ```
      $ ccsei &
      ```
   b. select "*CCS Environment Setup.*". The utility *vccEnv* is started.
   c. enter in the *vccEnv* panel the name of the environment to create
   d. Press the button *Create*
   e. Keep the panel *vccEnv* open: you will need in the next steps as well.
2. Configure the database
   a. set to the environment dbl directory:
      ```
      $ cd $VLTDATA/ENVIRONMENTS/wmyccd/dbl
      ```
   b. Add the CCD to the database:
      i. edit DATABASE.db file (see also example in section 3.6).
      ii. move to application definitions section.
      iii. Add the following lines (see also manual page of *ccd.db* in section 3.6):
      ```
      #define CCDNAME myccd                // camera name
      #define CCDROOT :Appl_data:myccd     // CCD branch root point
      #include "ccd.db"                    // build CCD branch
      ```
   c. add a dedicated point in the scan system configuration:

i. edit USER.db file (see also example in section 3.6)
ii. move to the scan system section.
iii. Add the following lines (instead of lmyccd use your CCDLENV!)

```
POINT "<VLT scan dev>" "ccs_config:scan config:LAN:lmyccd"
BEGIN
    Alias lmyccd
END
```

d. regenerate the (RTAP) database files:

```
$ make clean db
```

e. verify the correctness of the generated RTAP files:

```
$cd ../DB/Appl_data
$ls
```

...if the CCD branch has been successfully created, a subdirectory *myccd* exists

```
$cd ../ccs_config/scan\ config/LAN
```

...if the CCD scan point has been successfully created, a subdirectory *lmyccd* exists.

3. generate the environment and then shutdown it:
   a. Press the button *Init* in the *vccEnv* panel

4. Start the environment:
   a. Press the button *Start* in the *vccEnv* panel
   b. Close the panel *vccEnv*: not needed any more (*File*->*Quit*)
   c. Close the panel *ccsei*: not needed any more (*File*->*Quit*)

4.6.3.1 WS Environment Verification

1. If an LCU is used, edit the file *$VLT_LOG_ROOT/logLCU.config* to define the log node for the LCU. Example:

```
lmyccd     wmyccd
```

2. Verify (with *RtapDbConfig*) that point `Appl_data` contains a sub-point *myccd* (alias as well), having the structure shown in Fig.5 (in this example, the CCD branch root point is *ccdtest* instead of *myccd*):

```
                       |-ace
                       |-cryostat
                       |
                       |                      |-chip_1
                       |                      |-output_1
                       |-detector------------ |-output_2
                       |                      |-output_3
                       |                      |-output_4
                       |
                       |                                             |-display
                       |                                             |
                       |                                             |                      |-window_1
                       |                                             |-process------------- |-window_2
                       |                                             |
                       |                      |-exposure_1---------- |                      |-window_1
                       |                      |                      |-readout------------- |-window_2
                       |                      |                      |-telemetry
                       |                      |                      |-temperature
                       |                      |                      |-transfer
                       |                      |
                       |                      |                      |-display
                       |                      |                      |
                       |                      |                      |                      |-window_1
                       |                      |                      |-process------------- |-window_2
                       |-exposures----------- |                      |
                       |                      |-exposure_2---------- |                      |-window_1
                       |                      |                      |-readout------------- |-window_2
                       |                      |                      |-telemetry
                       |                      |                      |-temperature
                       |                      |                      |-transfer
                       |                      |
|-ccdUvSR------------- |                      |                      |-display
                       |                      |                      |
                       |                      |                      |                      |-window_1
                       |                      |                      |-process------------- |-window_2
                       |                      |-exposure_next------- |
                       |                                             |                      |-window_1
                       |                                             |-readout------------- |-window_2
                       |                                             |-transfer
                       |
                       |                                             |-plot---------------- |-plot_table
                       |                      |-process------------- |-window_1
                       |-images-------------- |                      |-window_2
                       |                      |-transfer
                       |-observations
                       |
                       |                      |-clock_1
                       |                      |-clock_10
                       |                      |-clock_2
                       |                      |-clock_3
                       |                      |-clock_4
                       |-readout------------- |-clock_5
                       |                      |-clock_6
                       |                      |-clock_7
                       |                      |-clock_8
                       |                      |-clock_9
                       |-shutter
                       |
                       |-telemetry----------- |-plot---------------- |-plot_table
                       |
                       |-temperature--------- |-plot---------------- |-plot_table
                       |-wcs
```

Fig.5 CCD database branch structure

3. Verify that the needed processes are running in the CCS environment (use *RtapPerfMon* for verification):

```
Proc     Name              pid  prio  %cpu    qid rx #msg bytes debug select
   1 RtapScheduler       25043   NRT    .    3676  .    .     .    .      ..
   2 RtapMonitor         25044   NRT    .    2780  .    .     .    .      ..
   3 RtapQServer         25045   NRT    .  216986  .    .     .    .      ..
   4 RtapEventTrig       25047   NRT    .   35187  .    .     .    .      ..
   5 RtapEventConfg      25048   NRT    .   33088  .    .     .    .      ..
   6 RtapTimeKeeper      25049   NRT    .   69989  .    .     .    .      ..
   9 RtapDbCfServer      25053   NRT    .    1891  .    .     .    .      ..
  10 RtapScanMngr        25055   NRT    .    1293  .    .     .    .      ..
  11 ccsScan             25078   NRT    .     400  .    .     .    .      ..
  12 logManager          25054   NRT    .   13892  .    .     .    .      ..
  13 msgServer           25057   NRT    .     695  .    .     .    .      ..
  14 cmdManager          25058   NRT    1     696  .    .     .    .      ..
  15 RtapASServer        25059   NRT    .     397  .    .     .    .      ..
  16 RtapASLogger        25060   NRT    .     398  .    .     .    .      ..
  17 RtapDHMngr          25077   NRT    .     399  .    .     .    .      ..
  20 envsKill            25056   NRT    .    5494  .    .     .    .      ..
  58 RtapMQDBM           25052   NRT    .   10890  .    .     .    .      ..
```

4.6.4 Files needed for CCD operations

Some files containing information needed to operate a CCD camera must be installed in the proper subdirectory of `INS_ROOT`; to do that, the CCD sw provides a UNIX shell script (*ccdInstall.sh*, see manual page in section 3.3).
Before running this script, **it is strongly recommended to save the contents of the current INS_ROOT in a backup directory**: the script will overwrite some files, in particular the files defining the CCD database configuration values.
The script needs as parameter the name of the file containing information about the specific camera used; there is one file for each CCD system; to list them all (see [7] for VLTROOT or INTROOT):

```
$ls $VLTROOT/config/ccd*.dbcfg
```

The choice of the right file for the system to be used should be straightforward if the following criteria are followed

- For TCCDS systems the file name is *ccdTec<n>.dbcfg (e.g. ccdTec21.dbcfg)*. The integer number <n> correspond to the number of the CCD camera head used (this number can be read on a side face of the CCD camera head. Example (from UNIX shell):

  ```
  $ccdInstall.sh ccdTec21.dbcfg
  ```

  The template file *ccdTecTemplate.dbcfg* is available as extrema ratio if no appropriate file is found.

- For SCCDS the choice, due to the limited number of delivered system, should be even easier. The files have prefix *ccdSci<instrument{Arm]> (e.g. ccdSciFors.dbcfg)*. Example (from UNIX shell):

  ```
  $ccdInstall.sh ccdSciFors.dbcfg
  ```

  The template file *ccdSciTemplate.dbcfg* is available as extrema ratio if no appropriate file is found.

For verification, do the following:

1. change working directory to $INS_ROOT

   `cd $INS_ROOT`

2. Show contents of sub-directories:

   `ls -R`

   At least the following directories must exist (see [4] for the meaning):

   $INS_ROOT/SYSTEM/DETDATA

   $INS_ROOT/SYSTEM/Dictionary

   `ESO-VLT-DIC.CCDDCS`

   $INS_ROOT/SYSTEM/COMMON/CONFIGFILES

   At least the following files must exist:

   `ccdDcsFixed.dbcfg`

   `ccdSciTemplate.dbcfg (*)`

   `ccdTecTemplate.dbcfg (*)`

   `<CCDNAME>.dbcfg`

   $INS_ROOT/SYSTEM/COMMON/SETUPFILES

   $INS_ROOT/SYSTEM/COMMON/SETUPFILES/REF

   $INS_ROOT/SYSTEM/COMMON/SETUPFILES/DET

   At least the following files must exist:

   `ccdCmdBias.det`

   `ccdCmdDisplay.det`

   `ccdCmdFlat.det`

   `ccdCmdStartAg.det`

   `ccdCmdStartLp.det`

   `ccdSetupComplete.det(*)`

   All files and directories listed above must be present to be able to run the system. They must not be deleted nor modified. The only exceptions are template files, marked with (*), which might be useful to the user, but are not strictly needed, and the file *<CCDNAME>.dbcfg*, which might be modified (not deleted!) to accommodate modifications in the camera configuration (see chapter “Configuration” in [16]).

Note: if the file `<CCDNAME>.dbcfg` is already existing, it is first saved as `<CCDNAME>.dbcfg`.BAK and then overwritten. The old information, if changed by the user, must then be re-entered (see chapter “Configuration” in [16]).

4.6.5 Real Time Image and World Coordinates Display

CCD Software can work with or without the Real Time Display. If you have RTD installed, check that the process rtdServer is already running:

```
$ ps -aef | grep rtdServer | grep -v grep
```

If this is not the case, start it:

```
$ rtdServer &
```

If It is intended to show the actual World Coordinates of the objects being displayed, some parameters have to be entered into the CCD sw. As already said in section 2.15.1, a script is used for this purpose: *ccdDcsWcs.sh* (see section 3.3). Example:

```
$ccdDcsWcs.sh $CCDNAME $RTAPENV 1.0 1.0 0.5 0.0 1950 0.0 PIXEL
```

4.6.6 Setting source mask for FITS logs

As already said in section 2.17, the source mask for FITS log has to be defined, using the script *ccdDcsSetLogMask.sh* (see also section 3.3). Example:

```
$ ccdDcsSetLogMask.sh $CCDNAME $RTAPENV "UT1AGA"
```

4.6.7 LCU Environment

After successful configuration of the workstation CCD software, one can already proceed to the CCD camera configuration operations (see chapter "Configuration" in [16]) and then run and operate a CCD camera in simulating the LCU Software (see chapter "Getting started" in [14]).
If you do not have an LCU skip to the next session. For all other operational modes, the LCU environment must be configured as well.

1. Create a new environment as in [7]. Remember, the environment shall be named as in `$CCDLENV`. Do not forget to properly configure `/etc/services`, `/etc/$RTAPROOT/RtapEnvList (or $RTAPROOT/etc/RtapEnvList on older Rtap releases)`, the LCU booting parameters, the communication parameters, etc.
   a. start *ccsei*

      ```
      $ ccsei &
      ```

   b. select "*CCS Environment Setup..*". The utility *vccEnv* is started.
   c. enter in the *vccEnv* panel the name of the LCU environment to create. Check if the WS host and boot environment are set as wanted; if needed, change these entries.
   d. Press the button *Create.*
   e. Press the button *Config*. The utility *vccConfigLcu* is started.
   f. Enter in the *vccConfigLcu* panel the LCU boot parameters. In particular:
      i. In the *Modules Configuration* area enter, **in the same order as below:**

         System Mod:

         ```
         lcudrv, lculog, lqs, tim, lcc, ntp, cai, scan, b016
         ```

         User Mod:

         ```
         ccd
         ```

      ii. in the *Processes* area enter (optional, not strictly needed):

         ```
         lccServer, msgServer, rdbServer
         ```

      Fig.6 shows an example of *vccConfigLcu* panel.

   g. Press the button *Write files* to save the defined configuration
   h. Press the button *Configure LCU* to store the boot parameters in the LCU CPU board.
   i. Press the button *Reboot LCU* to verify that the defined configuration is OK and the LCU boots properly.
   j. Close the *vccConfigLcu* panel (*File* -> *Quit*)

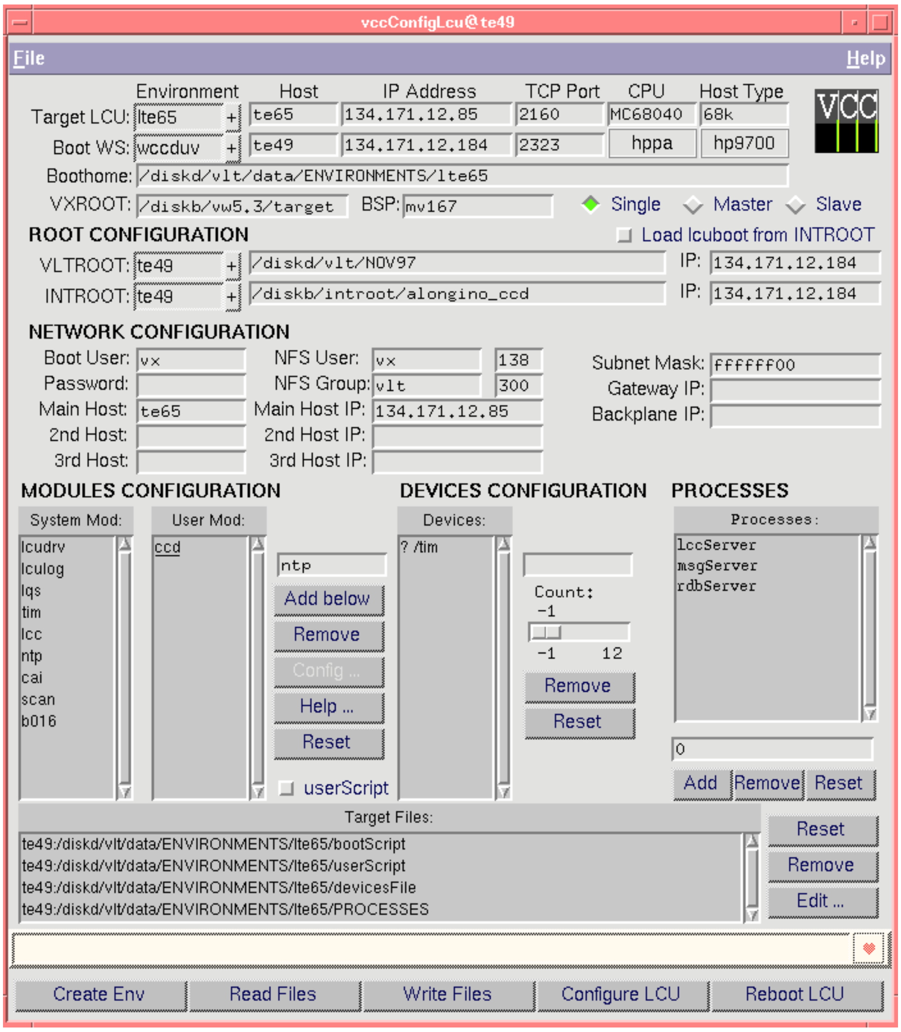


Fig.6 CCD LCU configuration

k. Close the *ccsei* panel (*File* -> *Quit*)

2. Configure the database

   a. go to the LCU environment dbl directory

      `$ cd $VLTDATA/ENVIRONMENTS/lmyccd/dbl`

   b. Add the CCD to the database:

      i. edit DATABASE.db file (see also example in section 3.6)

      ii. move to application definitions section.

      iii. Add the following lines (see also manual page of ccd.db in section 3.6):

```
#define CCDNAME myccd            // camera name
#define CCDROOT :myccd           // CCD branch root point
#include "ccd.db"                // build CCD branch
```

      Please note that the definition of the CCD branch root point CCDROOT is different with respect to the WS environment (*:Appl_data:myccd* see 4.6.3)

   c. add a dedicated point in the scan system configuration:

      i. edit USER.db file (see also example in section 3.6)

      ii. move to the scan system section.

iii. Add the following lines:

```
#define RTAPENV wmyccd      // CCD WS environment
POINT "<VLT scan dev>" LAN:RTAPENV // build CCD scan point
BEGIN
    ALIAS RTAPENV
END
```

d. regenerate the (LCU) database files:

```
$ make clean db
```

e. verify the correctness of the generated RTAP files:

```
$cd ../DB
$ls
```

...if the CCD branch has been successfully created, a subdirectory *myccd* exists

```
$ls LAN
```

...if the CCD scan point has been successfully created, a subdirectory *wmyccd* exists.

3. No special definition in the *userScript* is needed by the CCD sw. Note however that the directory specified on the Workstation in the environment variable **INS_ROOT** (see 4.6.1) **must be NFS mounted** on the target LCU and **the path string** on both WS and LCU **must be the same**. Examples:
   a. on CCD WS: INS_ROOT = /diskc/no_backup/vltccd/insroot
      on CCD LCU /diskc is mounted to CCD WS /diskc -------> OK, no further action.
   b. on CCD WS: INS_ROOT = /tmp_mnt/notCcdWs/diskc/no_backup/vltccd/insroot
      on CCD LCU /diskc is mounted to CCD WS /diskc -------> add line in *userScript*:
      `nfsMount ("notCcdWs","/diskc","/tmp_mnt/notCcdWs/diskc")`
4. Reboot the LCU.

4.6.7.1 LCU Environment Verification

1. Verify that the registered processes are:

```
$rlogin <lcuname>
-> lqsPrintLocalTbl
PID      Process-Name    ReadCId WriteCId ......
--- ------------------ -------- -------- ------
  1          msgServer 32783408       22 ......
  2                lqs 17              0 ......
  3            lccTime 32891072       21 ......
  4          lccLogger 32760284       24 ......
  5           lccEvent 32757788       25 ......
```

```
6           lccDevice 31240840       23 ......
7           lccServer 30690912       26 ......
8           rdbServer 30580372       28 ......
```

2. Verify using the browsing facility of the *ccseiDb* or *lccei* utility (see manual page) that the CCD branch and scan point have been added:
   a. Select point *@lmyccd:myccd*
   b. Read attribute *opStateLcu*.
   c. Select point *@lmyccd:LAN* (scan system). A sub-point *wmyccd* must exist

4.6.8 Scan System configuration

Before starting to use the CCD sw WS and LCU parts, one has to configure in the OLDB the table of LCU attributed which have to be updated on WS through the CCS Scan System.
To do this, the CCD sw provides a UNIX script, called *ccdDcsScan.sh* (see also manual page in 3.3).
Type from the UNIX shell:

```
ccdDcsScan.sh $CCDNAME $RTAPENV $CCDLENV
```

If the stand-alone panel is intended to be used, also the attributes needed by its Graphical User Interface must be included in the scan table. In this case, the script *ccdosScan.sh* must be run in addition to the previous one (see also manual page in 3.3):

```
ccdosScan.sh $CCDNAME $RTAPENV $CCDLENV
```

Note that, in order them to succeed, both WS and LCU environments must be properly configured and active.

4.6.9 ACE Verification

If one intends to use the CCD sw simulating the ACE Software (e.g. no ACE available, see also section 2.4), this section must be skipped.
If a ACE box is available and connected, the following verification test must be performed on the LCU console or from the host ($> rlogin <lcu>) to verify the proper connection of the ACE box to the LCU:

1. l<lcu> -> cd "/VLTROOT/vw/bin/MC68040"
2. l<lcu> -> acecom0
3. The following message should appear:

```
 ---> Test LCU-ACE connection <---

Loading transputer process ...
Attaching channel ...
Sending message to transputer ...
Reply is correct !

 ---> Test has been successful !!!

value = 2 = 0x2
```

4.6.10 Example

The following is an example of a sequence of UNIX shell commands needed to configure the CCD sw for usage. It is assumed here that the CCS environment, including OLDB, has been created and is active, as well as *rtdServer*. These commands are typically included in a CCD configuration script.

```
$ setenv HOST `hostname`
$ setenv RTAPENV wmyccd
$ setenv INS_ROOT /diskc/myName/insroot
$ setenv CCDNAME myccd
$ setenv CCDLENV lmyccd
$ ccdInstall.sh ccdTec58.dbcfg
$ ccdDcsWcs.sh $CCDNAME $RTAPENV 1.0 1.0 0.5 0.0 1950 0.0 PIXEL
$ ccdDcsSetLogMask.sh $CCDNAME $RTAPENV "UT1AGA"
$ ccdDcsScan.sh $CCDNAME $RTAPENV $CCDLENV
```

4.7 CCD CAMERA VERIFICATION

To verify the installation and configuration, execute the following steps:

1. start *ccdVerifyGui* and push the button *Check*: the correctness of WS, LCU and ACE configuration, both sw and hw, including cabling, is checked (see also “Complete self-test” in [16]):
   $ ccdVerifyGui
2. get familiar with the user interface:
   execute the “Simple Demo Session” as described in “Getting started” in [14].
3. run the WS processes:
   execute the “LCU Simulated Session” as described in “Getting started” in [14].
4. run the WS and LCU processes:
   execute the “ACE Simulated Session” as described in “Getting started” in [14].

4.8 USE THE CCD SOFTWARE

After successful completion of all the steps described in this chapter, the CCD software and the environment are ready for usage.
It is suggested to follow the steps described in “Running the system” in [14].

5 ERROR DEFINITIONS

The CCD software uses the standard mechanism defined and provided by the CCS error system to log and return error information to external applications, both at function and command level. Errors are defined for both WS and LCU in the following files:

1. **ccdErrors.h** error codes
2. **ccd_ERRORS** error messages

See sections 3.5 and 3.11 for more information.

___oOo___