CCD3 Array Controller

Technical Manual Rev. 0.1

Introduction

This document describes the low-level command interface of the Niels Bohr Institute CCD3, a 3rd generation CCD array controller and associated software package. The document is intended as a working reference for systems integrators and engineers working with integration, interfacing and commissioning.

The reader is assumed to have a sound knowledge of the internal workings of the CCD3 camera system and charge coupled detector technology in general terms.

For introduction, general information or system description, the reader is referred to the documentation available on http://ccd3.not.iac.es/controller/

Nomenclature

Examples or text referring to actual screen display are showed in `courier` font face.

Sections marked with a grey color are topics currently under development, planned for future releases or designated exclusively for development or debugging purposes. They should therefore not be relied on in actual implementation and/or are not for use in production environments.

Disclaimer

The Niels Bohr Institute is committed to continuously supply state of the art CCD camera technology. As such, the CCD3 array controller and associated software are subject to a continuing development effort. Therefore, as a work in progress, all information disclosed here is by definition subject to change, but should be backward compatible.

For latest version of this document, please refer to the online document repository at http://ccd3.not.iac.es

Requirements and dependencies

In general, the CCD3 Array Controller system delivery includes a suitable control and acquisition computer, which is configured to customer specifications and tested according to the agreed performance. If for some reason the software needs to be reinstalled or the hardware replaced, please refer to this chapter for details.

The control and acquisition software may possibly also function satisfactorily in less powerful configurations, but this has not been verified and is not supported.

Hardware

- Intel x86-64 based CPU. Multicore versions[1] are recommended; the acquisition software scales well on more CPU cores.
- 1 available PCI[2] half length expansion slot.

Software

- X86_64 GNU/Linux, kernel 2.6.32-27 or newer. Recommended distribution is Ubuntu 10.04 Lucid.
- g++-4.4.3
- xpa-2.1.12
- cfitsio v.3.21
- libncurses5-dev
- ivy-c v.3.11.4
- libpcre3-dev v8.10
- libglib2.0-dev
- tcl8.5-dev
- xutils-dev
- libmysqlclient-dev
- kernel-headers matching running kernel
- mysql-server v5.1
- Gnome desktop

Recommended utilities

- phpmyadmin
- Midnight Commander
- SAOImage DS9
- IDL

[1] Intel Core i5, i7 especially
[2] http://www.pcisig.com/home

Installation

The CCD3 Array Controller acquisition software “ccd3comm” and associated modules are distributed in a compressed tarball “ccd3-X.X.XX.tar.gz”. The files are extracted using a command like: `tar xvzf ccd3-X.X.XX.tar.gz.` Suggested location of the source files is `/usr/src`.

As per *de facto* GNU/Linux standard, the build procedure is started with the command `make all,` which will compile and install all required modules; likewise, the install procedure, which should be executed with administrative privileges, is started with: `sudo make install`.

File locations

The files will be installed in the following locations:

- `/usr/local/ccd3/*`
  contains all binaries, configuration files, startup scripts.

- `/usr/local/bin/*`
  symbolic links to binaries.

- `/etc/ccd3/*`
  symbolic links to configuration files.

- `/etc/sysctl.d/`
  symbolic link to kernel configuration file

- `/etc/init.d/*`
  symbolic links to startup scripts

- `/lib/modules/[kernel version]/kernel/drivers/ccd3/ccd3.ko`
  ccd3 kernel module

- `/etc/udev/rules.d/ccd3.rules`
  udev rules for the ccd3 kernel driver

Modified files:

- `/etc/modules`
  ccd3 kernel driver added to module load sequence.

Permissions

The device files `/dev/ccd3ctl` and `/dev/ccd3` are by default owned by the root user, with the permissions set as `666`, i.e. readable and writable by everyone. If the permissions are altered, it must be ensured that the user running the ccd3comm module has read and write permissions to the device files.

All files created by the ccd3comm are owned by the user currently running ccd3comm. All files used by the ccd3comm, e.g. scripts and batch files, must be readable, and scripts must be executable.

Configuration

ccd3comm.conf

The ccd3comm.conf configuration file is an editable text file, specifying general settings for the ccd3comm module. The application will search for this file in the /etc/ccd3 directory, but an alternative configuration file can be specified by the “-c” command line switch.

`[Application]`
`; General application settings`	

`Ansi = true`	Use of ansi codes on/off in terminal.
`Uid = root`	User id when running application as daemon.
`Batchfile = on_start`	Batch file to be executed upon initialization.
`Scriptdir = /etc/ccd3/scripts`	Location of scripts and batch files.
`Mirror_x = false`	If set, output image will be mirrored across x-axis.
`Mirror_y = false`	If set, output image will be mirrored across y-axis.
`Rotate = 0`	Will rotate output image accordingly; valid values are 0, 90,180, 270.

`[log]`
`; Specifies log settings`	

`Logfile = syslog`	Logging destination. Either a file or the special values “syslog” which will log using the syslog.
`Syslog_ident = ccd3`	Source identification when using syslog logging.
`Syslog_facility = LOG_LOCAL5`	Facility when using syslog.

`[file]`
`; Contains settings associated with file handling`	

`Combine = false`	If set, data will be descrambled and organized into one complete image. Otherwise each amplifier output is saved into separate extensions.
`Allow_close = false`	If set, the output file is closed when the data transfer and processing has been completed. Otherwise an explicit “`fileclose`” command is required to close the file.
`Prefix = ccd3_`	Prefix prepended to any autogenerated filenames.

`Autosave = true`	If set, all image data are saved regardless of whether a filename has been specified. If no filename has been specified, the file is saved to an autogenerated filename.

`Namestyle = default`	Specifies the style of autogenerated filenames. Valid values are [default, not, “string”].
	Default = [prefix][nnnn].fits, n = continous counter.
	Not = NOT style, see NOT documentation.
	“string” = [prefix][string][nnnn].fits, n = continous counter.

`Output_dir = /images`	Directory for output images.

`[detector_layout]`
`; Specifies physical layout of the detectors, location and orientation`	

`Layout = --:--`	
`Orientation = horizontal`	

`[header_keywords]`
`; This section contains static keywords written to the main header`
`; of the .fits file.`	

`ORIGIN = PLEASE SET /etc/ccd3comm.conf`	Optional but suggested keyword.
`OBSERVAT = PLEASE SET /etc/ccd3comm.conf`	Optional but suggested keyword.
`TELESCOP = PLEASE SET /etc/ccd3comm.conf`	Optional but suggested keyword.
`INSTRUME = PLEASE SET /etc/ccd3comm.conf`	Optional but suggested keyword.
`DETNAME = PLEASE SET /etc/ccd3comm.conf`	Optional but suggested keyword.
`COMMENT = PLEASE SET /etc/ccd3comm.conf`	Optional but suggested keyword.
`CREATOR = CCD3COMM`	Optional but suggested keyword.

`[extension_keywords]`
`; This section contains static keywords written to the extensions headers`
`; of the .fits file.`	

`BUNIT = count`	Optional but suggested keyword.
`CCDNAME = PLEASE SET /etc/ccd3comm.conf`	Optional but suggested keyword.

`[ivy_message_bus]`
`; This section specifies how ccd3comm communicates on the ivy message bus`[3]*`.`*	

`name = ccd3comm`	Ivy identification string, should be unique on the ivy message bus.
`Enable = true`	Enable/disable ivy message bus communication.
`[ccd3_event] = [ivy_message]`	General form of ivy message specification. A few examples are shown below.
`on_start = application.start`	Sent on application initialization.
`on_stop = application.stop`	Sent on application shutdown.
`on_error = application.error`	Sent on general application error handling.
`on_reset = application.reset`	Sent on hardware reset.

[3] *See section section “Ivy bus” for details*

`on_command = con.command`	Sent on console command.
`on_response = con.response`	Sent on console response.
`on_exposure_start = exposure.start`	Sent on start of exposure.
`On_exposure_progress = exposure.progress`	Sent during exposure.
`On_exposure_end = exposure.end`	Sent on finalization of exposure.

Architecture

The main interface to the CCD3 controller is a fiber optic cable, connecting the CCD3 controller and PCI interfacing board. The communication is divided in two logical channels: a character based command channel and a high speed data channel for bulk data. Command and status are exchanged on the command channel in clear text mode[4], while image data are transmitted from the controller to the acquisition card on the data channel formatted as 32 bit unsigned integers.

Kernel module

The hardware communication is implemented in a kernel module, performing actual I/O and presenting a well defined interface to the user mode module(s). The interface is populated in the device file system, where communication is possible using file read/write or memory mapping techniques.

Device files:

```
/dev/ccd3ctl
/dev/ccd3
```

Character channel

The character channel is implemented on the `/dev/ccd3ctl` device. Commands can be issued e.g. using `fwrite()` and status read by `fread()`. Protocol is clear text ascii characters.

Data channel

The data channel is implemented on the `/dev/ccd3` device. This is a read-only device where 32 bit unsigned image data can be read. Any attempt to write to this device will fail, and results are undefined.

Concurrency

Concurrency is possible, but no means of resource protection or synchronization is implemented, and responsibility for timing is thus left to the user applications.

[4] For protocol information see document http://ccd3.not.iac.es/controller/FE-SW/current_Ctrl_Comm.pdf

Interfacing

Command line

Multiple possibilities exist to loop into the command chain. At the user interface level, the application is constructed as a console program reading from stdin and writing to stdout, and thus ordinary piping and redirection can be used to communicate from other programs or shell scripts, e.g.

```
echo xsiz 1024 | ccd3comm
```

or if the ccd3comm is run as a daemon, using ivyprobe to send, e.g.

```
ivyprobe ‘(.*)’
ccd3comm.con.command rotate 90
```

At the driver level, the /dev/ccd3ctl can be utilized for reading status and writing commands, and the /dev/ccd3 can be used for reading image data.

Examples:

```
echo @sint >>  /dev/ccd3ctl
echo @xsiz
```

wait for exposure to complete…

```
cat /dev/ccd3 >> ~/myimage.raw
```

in order to read out raw non-descrambled and unprocessed image data.

Using this approach there is no mean to monitor the current status of controller. It is recommended to use the ccd3comm console interactively or alternatively run the ccd3comm as a daemon and use the ivy bus for controlling the acquisition. These two modes are also the ones primarily targeted for production systems.

Ivy bus

Comprehensive information regarding runtime state and control is broadcast on the ivy bus. Selected ivy commands are also available to control the application and controller. In general, all commands handled by the interpreter are transmitted as:

```
ccd3comm.con.command [command]
```

and resulting reply is transmitted as:

```
ccd3comm.con.reply [reply to command]
```

The actual ivy message transmitted or received can be customized in the /etc/ccd3/ccd3comm.conf configuration file. If multiple controllers are running on the same TCP/IP network, the sender name can be changed in the configuration file in order to not create conflicts between multiple camera systems.

Also, it is suggested that the systems integrator investigates to use ivy message bus communication, using the ivyprobe utility:

```
ivyprobe ‘(.*)’
```

for listing all communication, or:

```
Ivyprobe ‘(ccd3comm.*)’
```

for listing all communication from the “ccd3comm” node on the ivy network.

Example output from ivyprobe:

```
root@cam2:~# ivyprobe '(.*)'
Broadcasting on network 127.255.255.255, port 2010
ccd3comm connected from localhost
ccd3comm sent  'ccd3comm.application.start'
ccd3comm sent  'ccd3comm.cam.reply !drv reset'
ccd3comm sent  'ccd3comm.cam.query ?drv sync'
ccd3comm sent  'ccd3comm.cam.reply !drv sync 1'
ccd3comm sent  'ccd3comm.cam.query ?deen'
ccd3comm sent  'ccd3comm.cam.reply !deen 3'
ccd3comm sent  'ccd3comm.cam.query ?xsiz'
ccd3comm sent  'ccd3comm.cam.reply !xsiz 2148'
ccd3comm sent  'ccd3comm.cam.query ?ysiz'
ccd3comm sent  'ccd3comm.cam.reply !ysiz 4102'
ccd3comm sent  'ccd3comm.cam.query ?xbeg'
ccd3comm sent  'ccd3comm.cam.reply !xbeg 1'
ccd3comm sent  'ccd3comm.cam.query ?ybeg'
ccd3comm sent  'ccd3comm.cam.reply !ybeg 1'
ccd3comm sent  'ccd3comm.cam.query ?xbin'
ccd3comm sent  'ccd3comm.cam.query ?xsiz'
ccd3comm sent  'ccd3comm.cam.reply !xbin 1'
ccd3comm sent  'ccd3comm.cam.reply !xsiz 2148'
ccd3comm sent  'ccd3comm.cam.query ?ybin'
ccd3comm sent  'ccd3comm.cam.query ?ysiz'
ccd3comm sent  'ccd3comm.cam.reply !ybin 1'
ccd3comm sent  'ccd3comm.cam.reply !ysiz 4102'
ccd3comm sent  'ccd3comm.cam.query ?time'
ccd3comm sent  'ccd3comm.cam.reply !time 1443'
ccd3comm sent  'ccd3comm.cam.query ?tmpw'
ccd3comm sent  'ccd3comm.cam.reply !tmpw   30.00'
ccd3comm sent  'ccd3comm.cam.query ?tmpa'
ccd3comm sent  'ccd3comm.cam.reply !tmpa    0.00'
ccd3comm sent  'ccd3comm.cam.query ?tmpl'
ccd3comm sent  'ccd3comm.cam.reply !tmpl    0.00'
ccd3comm sent  'ccd3comm.cam.query ?pres'
ccd3comm sent  'ccd3comm.cam.reply !pres 0.00e+00'
ccd3comm sent  'ccd3comm.cam.query ?shut'
ccd3comm sent  'ccd3comm.cam.reply !shut 1'
ccd3comm sent  'ccd3comm.cam.query ?stat'
ccd3comm sent  'ccd3comm.cam.reply !stat 4'
ccd3comm sent  'ccd3comm.cam.query ?rexp'
ccd3comm sent  'ccd3comm.cam.reply !rexp 1'
ccd3comm sent  'ccd3comm.cam.query ?deav'
ccd3comm sent  'ccd3comm.cam.reply !deav 3'
```

External scripting

Customized scripts and batch files should be located in the /etc/ccd3/scripts/ directory, or according to the script directory setting in the /etc/ccd3/ccd3comm.conf configuration file.

For any commands that are not prefixed by a ‘@’ or ‘?’ character and are not contained in the set of defined application commands, the ccd3comm application searches through the defined script directory for files of that specific name. If a file is found and readable, it is considered a script or batch file.

The application determines the type of file by examining the content of the file. A script file will always start with a sequence like “#!/bin/bash” - specifying the system interpreter - and if this sequence is present, the file is passed to a shell and executed. Otherwise the file is considered a batch file, and the content will be executed by the ccd3comm command interpreter, as if the commands were typed directly at the console prompt.

Batch files are ordinary text files containing any command that would be valid on the ccd3comm console command line. The file should be readable, but not executable. For an example of a batch file, refer to page 21, section “Example batchfile”.

Script files are shell scripts, which can be executed by the shell available on the current system. Script files should be readable and executable. For an example of a script file, refer to page 22, section “Generating batch files in shell scripts”.

Communication structure

The CCD3 Array controller system comprises the CCD controller, interfacing to the detector, an optical fiber based communication network and an acquisition application running on a X86_64 GNU/Linux compatible operating system.

Commands, requests, replies

The camera can be controlled by issuing text commands on the logical text channel of the network, and status information can be obtained in a similar manner. The communication is connection oriented, in the sense that a command or question is always followed by a reply from the camera. The general format is structured as per Table 1.

Table 1 Controller communication structure

Function	***Description***
`@[token] {arg 1 [.. arg n]}`	*A controller command is preceded by the '@' character (ascii 64) and terminated by either a CR (ascii 13) or LF character (ascii 10).* *A command may be followed by one or more arguments, specific for the command.* *The controller is requested to perform some specific action and reply with a "!token" answer, possibly containing one or more arguments e.g. indicating a state or scalar value.* *Example setting the horizontal dimension of the image geometry:* *@xsiz 1024* *!xsiz 1024*
`?[token]`	*A controller question is preceded by the '?' character (ascii 63) and terminated by either a CR (ascii 13) or LF character (ascii 10).* *The controller is requested to reply with a "!token" answer, indicating the result of the query.* *Exampling requesting the horizontal dimension of the image geometry:* *?xsiz* *!xsiz 1024*

`![token] [arg]`	*A controller reply is preceded by the '!' character (ascii 33) and terminated by either a CR (ascii 13) or LF character (ascii 10).* *The reply may be followed by one or more arguments indicating the result or a state.*
`#[token] {on [period]| off}`	*Status solicitation commands are preceded by the '#' character (ascii 35) and terminated by either a CR (ascii 13) or LF character (ascii 10).* *The command may be followed by an "on" keyword, turning solicitation mode on, or an "`off`" keyword, turning solicitation mode off.* *The "on" argument may be followed by an optional period specified in milliseconds, by which the status is repeated. If no period is specified, the status message is repeated every 1000 millisecond. Upon a change in state, the status is also repeated.*

Direct commands or queries for the controller, prepended by a '`@`' or '`?`' character, are piped directly to the controller via the communication network text channel and subsequently interpreted and processed by the CCD3 controller. All other commands are handled by the command interpreter of the acquisition application. An application command may or may not result in one or more direct controller commands, e.g. issuing an "`exposure`" command will open a file, query the controller for the current metrics and generate a "`sint`" command for the CCD3 controller. Issuing a "`file [filename]`" command will store the filename internal to the application, but no controller communication will occur as a result of the command.

Controller Direct Commands

Direct commands are command given directly to the controller i.e. not interpreted by the user application. Commands given to the controller are case insensitive.

Exposure control **(all times in msecs)**

Command	**Request**	**Reply**	***Description***
`@sint`	-	`!sint`	*Start integration. Internal command; can't be issued directly – DON'T USE.*
`@time n`	`?time`	`!time %8d`	*Initial integration time in msec (n>1)*
`@timr n`	`?timr`	`!timr %8d`	*Residual exposure time in msecs (n>1)*
-	`?tima`	`!tima %8d`	*Actual elapsed time in msec.*
`@timw n`	-	`!timw %8d`	*new total wanted time in msec (n>1)*
`@sdly n`	`?sdly`	`!sdly %8d`	*Shutter delay in msec (n>1)*
-	`?stat`	`!stat %5d`	*status[31..16] : all zeroes* *status[15..08] : status from ctrl-sequencer* *status[07..00] : status from ctrl-program* *[0] : sequencer start/stop* *[1] : -* *[2] : shutter enable* *[3] : reset sequencer* *[8] : sequencer prompts for a start timing* *[9] : sequencer prompts for a PSU-sync* *[14,12]=0 idle* *[14,12]=1 integrating* *[14,12]=2 readout* *[14,12]=3 clear* *[14,12]=4 shutter delay*
`@imod n,m`	`?imod n`	`!imod %1d %1d`	*Integration mode:* *n=0 => shutter* *n=1 => clear before exposure* *n=2 => readout after exposure* *m=0 => off; m=1 => on;*
`@brek`		*`!brek`*	*Hard break of integration or readout: Abort integration and/or readout without any saving and closes shutter.* *If the integration is to be terminated with a normal file save, use the timr or timw commands.*

Table 2 Exposure control

Readout format

Command	Request	Reply	Action

@fres	-	!fres	*Format reset :
			xtot and ytot to hard-programmed values
			xsiz=xtot; ysix=ytot (no windowing)
			xbeg=ybeg=xbin=ybin=1 (no binning)*

@xtot n	?xtot	!xtot%5d	*Total x size (for engineering only)*
@ytot n	?ytot	!ytot%5d	*Total y size (for engineering only)*
@xsiz n	?xsiz	!xsiz%5d	*x size for window*
@ysiz n	?ysiz	!ysiz%5d	*y size for window*
@xbeg n	?xbeg	!xbeg%5d	*x coordinate of lower left corner of window (First pixel is 1)*
@ybeg n	?ybeg	!ybeg%5d	*y coordinate of lower left corner of window (First pixel is 1)*
@xbin n	?xbin	!xbin=%4d, Tpix=%4d => %4dkpix/s	*x binning Recalculates: xsiz = (xsiz*xbin_old) div xbin_new;*
@ybin n	?ybin	!ybin%4d	*y binning, recalculates: ysiz = (ysiz*ybin_old) div ybin_new;*
@rden m,n	?rden [m]	!rden %1d %1d	*Readout amplifier control[5]; 0: Off, 1: Left; 2: Right, 3: Dual. This command is handled on a per detector basis, e.g. set dual amplifiers on detector zero: @read 0 3*
-	?reav [m]	!reav %1d	*Number of available readout amplifiers on detector number m*
-	?deav	!deav %1d	*Number of available detectors*
@deen n	*?deen*	*!deen %1d*	*Number of enabled detectors[6]*

Table 3 Readout format

[5] *As of writing, windowing is supported only in read 1 mode (left readout*)

[6] *Please note that the geometric settings (eg. xsiz, ysiz), is handled on a per detector base.*

Readout timing

Command	Request	Reply	Action
@tsam n	?tsam	!tsam %4d,Tpix=%4d => %4dkpix/s	Clamp and sample times in clocks
@tspw n	?tspw	!tspw %4d,Tpix=%4d => %4dkpix/s	Serial pulse width in clocks
@tsol n	?tsol	!tsol %4d,Tpix=%4d => %4dkpix/s	Serial pulse overlap in clocks
@tsnd n	?tsnd	!tsnd %4d,Tpix=%4d => %4dkpix/s	Serial neutral delay in clocks
@tstr n	?tstr	!tstr %4d,Tpix=%4d => %4dkpix/s	Serial rise/fall times in clocks
@tres	-	!tres %4d, Tpix=%4d => %4dkpix/s	Reset all timing

Table 4 Readout timing

Gain and offset

Command	Request	Reply	Action
@gain n m	?gain n	!gain %2d %7.3f	Individual Digital gain m in channel n
@zero n m	?zero n	!zero %2d %8d	Digital zero m in channel n
@offs n m	?offs n	!offs %2d %7.0f	Analog offset m in channel n
@cdsg n	?cdsg	!cdsg %8d	Fundamental cds-gain (n is integer)

Table 5 Gain and offset

Bias voltage

Command	Request	Reply	Action
@vbha n m	?vbha n	!vbha %2d %7.3f	Set HA high voltage channel n to m volts 5.0<=m<=24.0; Usually used for OD *1
@vbhb n m	?vbhb n	!vbhb %2d %7.3f	Set HB high voltage channel n to m volts 5.0<=m<=24.0; Usually used for RD *1
@vbhc n m	?vbhc n	!vbhc %2d %7.3f	Set HC high voltage channel n to m volts 5.0<=m<=24.0; *1
@vbla n m	?vbla n	!vbla %2d %7.3f	Set LA low voltage channel n to m volts -4.0<=m<=+4.0; Usually used for OG1 *1
@vblb n m	?vblb n	!vblb %2d %7.3f	Set LB low voltage channel n to m volts -4.0<=m<=+4.0; Usually used for OG2 *1
@vbod n m	?vbod n	!vbha %2d %7.3f	Same as vbha (for backward compatibility)
@vbrd n m	?vbrd n	!vbhb %2d %7.3f	Same as vbhb (for backward compatibility)
@vbdx n m	?vbdx n	!vbhc %2d %7.3f	Same as vbhc (for backward compatibility)
@vbog n m	?vbog n	!vbla %2d %7.3f	Same as vbla (for backward compatibility)
@vbgx n m	?vbgx n	!vblb %2d %7.3f	Same as vblb (for backward compatibility)

Table 6 Bias voltage

Miscellaneous

Command	**Request**	**Reply**	***Description***
`@rest`	-	`!rest`	*Reset OptoRing*
-	`?pixc`	`!pixc %8d %8d`	*Pixel counter (for test purpose)* *new_pixcnt,(new_pixcnt-old_pixcnt)*
	`?temp n`		

Table 7 Miscellaneous

Application Commands

Command	***Description***
`cam [n]`	*Select camera device indexed by n, n = [1-8].*
`file [filename]`	*Set output file to [`filename`]. The filename is reset after a successful data offload sequence. The filename needs to be a valid UNIX filesystem name.* *If a value of "`auto`" isset, the filename will be auto-generated according to the `namestyle` setting in the configuration file.* *If [filename] is omitted a currently open file is closed. If no files are open, the command is ignored.*
`show [n]`	*Show image saved at position [n], where n needs to be a previously allocated and stored image.*
`eval [expression]`	*Perform calculation given by [`expression`]. Syntax TBD.*
`v | verbose [n]`	*Set application verbosity to level [n], where 0 = less informative and 4 = debugging information.*
`Reset`	*Reset hardware, buffers and internal states.*
`clear`	*Clear the console.*
`expose`	*Initiate image integration and readout.*
`abort`	*Abort current integration and/or transfer.*
`xpaset [arg]`	*Set xpa command [`arg`] to DS9 preview display. Not valid when preview is not enabled.*
`xpaget [arg]`	*Get xpa response from DS9 preview display. Not valid when preview is not enabled.*
`examine [width][height]`	*List data values in the area of the size = width x height, positioned by the mouse cursor.*
`rms [width][height]`	*Calculate the rms (root mean square) value of the data in the area of size width x height, positioned by the mouse cursor.*
`ivy [text string]`	*Send the message [`text string`] on the ivy communication bus.*
`keyword [name][value][comment]`	*For the next exposure, save a fits header keyword given by `[name][value][comment]`. Comments are optional.*
`batch [batchfile]`	*Execute commands stored in the file [`batchfile`] sequentially. [`batchfile`] needs to be a text file containing valid camera and/or application commands. [`batchfile`] may contain a valid path to the file, with respect to the current execution directory.*
`execute [program]`	*Execute an external binary given by [`program`]. [`program`] needs to specify a valid UNIX executable binary and may contain a valid path to the file with respect to the current execution directory. Command control is returned when the program returns. Possible output is displayed on stdout.*
`run [program]`	*Execute an external binary give by [`program`]. [`program`] needs to specify a valid UNIX executable binary and may contain a valid path to the file, with respect to the current execution direction. Command control is returned immediately; thus this command may be a source of stale processes in case of errors. USE WITH CAUTION.*
`help`	*Display help on application commands on console.*
`quit | q`	*Quit application.*

Table 8 Application commands

Command line options

CCD3COMM

A number of command line options are available when executing the ccd3comm application.

Option		Description
-n	--noansi	Turn off ansi codes.
-b [filename]	--batch	Execute commands in file [filename].
-d	--daemon	Release console and go to background.
-i [device index]	--index	Use CCD3 device [device index] Index on the pci bus, try “lspci”, default = 0.
-a [camera index]	--camera	Use CCD3 camera [camera index] Valid range = [1..8], default = 8.
-c [config file]	--config	User configuration file [config file].
-v [level]	--verbose	Set verbosity level. Valid range = [0..4] or “silent”, “fatal”, “error”, “normal” and “debug”
-V	--version	Display version information and exit.
-h	--help	Display command line help and exit.

Table 9 Command line options

CCD3DB

A number of command line options are available when executing the ccd3db application.

Option		Description
-d	--daemon	Release console and go to background.
-c [config file]	--config	User configuration file [config file].
-v [level]	--verbose	Set verbosity level. Valid range = [0..4] or “silent”, “fatal”, “error”, “normal” and “debug”.
-V	--version	Display version information and exit.
-h	--help	Display command line help and exit.

Example batchfile

Example of a batch file creating 4 images suitable for noise calculations. This sequence can be loaded from the application console by issuing the command “`[filename]`” on the ccd3comm console, or in the shell by starting the application with the command “`ccd3comm –b [filename]`”

```
@imod 0
@xsiz 2148
@ysiz 4102
@tsam 10
@time 5
file dmy.fits
sint
file bias1.fits
sint
file bias2.fits
sint
@imod 1
@time 300
file flat1.fits
sint
file flat2.fits
sint
q
```

Generating batch files in shell scripts

For exposure sequences iterating on one or more parameters and possible subsequent post-processing, it is convenient to use a shell script for generating the batch file(s) and performing processing.

The following listing shows an example on how to use a shell script to generate batch file commands to iterate on exposure times from 100 milliseconds to 4 seconds and clamp/sampling times from 20 to 360 clocks. Using this approach, it is possible to iterate over as many parameters as desired and in any combination, e.g. in order to find local maximum or minimums in some characteristic, say noise or linearity.

```
#!/bin/sh

TMPFILE="./tmpbatch"
OUTFILE="./data.txt"
CCD3COMM="/usr/local/bin/ccd3comm"
IDL="/usr/local/bin/idl"
IDL_SCRIPT="./tilegain.pro"
XBEG=1350
YBEG=3080
XSIZ=500
YSIZ=500


B1FILE=bias1.fits
B2FILE=bias2.fits
F1FILE=flat1.fits
F2FILE=flat2.fits

rm -f $OUTFILE

# iterate 30x times on TIME
TIME_FROM=100
TIMETO=4100
TIME_INCREMENT=100


#iterate 34x times on TSAM
TSAM_FROM=20
TSAM_TO=360
TSAM_INCREMENT=10

#total measurements , 30x34 = 1020

TSAM=$TSAM_FROM

while [ $TSAM -lt $TSAM_TO ];
do
    echo @tsam $TSAM >  $TMPFILE
    TIME=$TIME_FROM

    while [ $TIME -lt $TIMETO ];
    do
        echo "********** Doing tsam=$TSAM, time=$TIME **********"
        rm -f $B1FILE
        rm -f $B2FILE
        rm -f $F1FILE
        rm -f $F2FILE

        echo @xbeg $XBEG        >> $TMPFILE
        echo @ybeg $YBEG        >> $TMPFILE
        echo @xsiz $XSIZ        >> $TMPFILE
        echo @ysiz $YSIZ        >> $TMPFILE
        echo @imod 0            >> $TMPFILE

        echo @imod 0            >> $TMPFILE

        echo @time 5            >> $TMPFILE

        echo file $B1FILE       >> $TMPFILE
        echo sint               >> $TMPFILE

        echo file $B2FILE       >> $TMPFILE
        echo sint               >> $TMPFILE

        echo @imod 1            >> $TMPFILE
        echo @time $TIME        >> $TMPFILE

        echo file $F1FILE       >> $TMPFILE
        echo sint               >> $TMPFILE

        echo file $F2FILE       >> $TMPFILE
        echo sint               >> $TMPFILE
        echo q                  >> $TMPFILE

        $CCD3COMM -b $TMPFILE

        echo $TSAM $TIME $(echo .run $IDL_SCRIPT | $IDL) >> $OUTFILE
        echo "********** Done tsam=$TSAM, time=$TIME **********"

        rm -f $TMPFILE
        let TIME=TIME+$TIME_INCREMENT
    done

    let TSAM=TSAM+$TSAM_INCREMENT
done

exit 0
```

List of tables