NIR (GEIRS Manual) (v 6.351)

<table>
<thead>
<tr>
<th>Name</th>
<th>Centre</th>
<th>Date</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepared:</td>
<td>R. J. Mathar</td>
<td>MPIA</td>
<td>December 17, 2019</td>
</tr>
</tbody>
</table>

Revised:

Approved:

Authorised:

---

### Document change record

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Section / paragraph / page</th>
<th>Change description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.266</td>
<td>23 Sep 2013</td>
<td>All</td>
<td>first version</td>
</tr>
<tr>
<td>6.351</td>
<td>December 17, 2019</td>
<td>All</td>
<td>GEIRS SVN major version trunk-r789M-33</td>
</tr>
</tbody>
</table>
Contents

1 OVERVIEW 8

1.1 Design ......................................................... 8
1.2 Interfaces ...................................................... 9
1.3 Operation ...................................................... 9
1.4 Acronyms ....................................................... 10
1.5 References ..................................................... 13

2 INSTALLATION 15

2.1 External Software .............................................. 15
    2.1.1 Compilers ............................................... 15
    2.1.2 boost .................................................... 16
    2.1.3 gsl ....................................................... 17
    2.1.4 libtiff .................................................. 17
    2.1.5 TwiceAsNice ............................................. 17
    2.1.6 Terminal Library ......................................... 17
    2.1.7 xterm Library ........................................... 17
    2.1.8 Other .................................................... 18
    2.1.9 Plx ....................................................... 19

2.2 GEIRS Compilation ........................................... 21
    2.2.1 Obtaining the Source Code and Patterns ................... 21
    2.2.2 Compilation ............................................... 22

2.3 De-Installation ............................................... 23

2.4 Configuration of the Operating System ....................... 24
    2.4.1 Shared Memory .......................................... 24
    2.4.2 Subnet .................................................. 25
    2.4.3 journaling ............................................... 25
    2.4.4 Shutdown ................................................ 25

2.5 User Configuration ........................................... 26
    2.5.1 Directory Layout ........................................ 26
    2.5.2 Path ..................................................... 27
    2.5.3 Standard Scripts ....................................... 27
5.4.2 Syntax Checker ........................................... 86
5.4.3 Total Integration Time .................................. 87
5.4.4 Macro Generators ....................................... 88
5.5 Shell Commands ........................................... 90
5.6 Windows ................................................... 100
  5.6.1 Window Classifications and Nomenclature .......... 100
  5.6.2 srre Readout Mode .................................. 101
5.7 Tutorial .................................................. 115
  5.7.1 read, sync, save ..................................... 116
  5.7.2 itime, ctype ......................................... 116
  5.7.3 crep, set savepath, next ............................ 116
  5.7.4 save multiple times, sample-up-the-ramp .......... 117

6 FITS OUTPUT ............................................. 118
  6.1 Illustrative Example ..................................... 118
  6.2 Online Keyword Modification ............................ 121
    6.2.1 PANIC Temperatures and Pressures ................. 121
    6.2.2 File-based Subscriptions ............................ 121
    6.2.3 fits Command ....................................... 123
  6.3 Optional Cleanup ...................................... 123
  6.4 GEIRS Core Keywords ................................... 124
  6.5 Image Location ......................................... 131
  6.6 Image Construction With srre(e) ....................... 134
  6.7 Single Frame Dumps .................................... 135

7 EXPOSURE TIME ......................................... 141
  7.1 Nomenclature .......................................... 141
  7.2 Lir with idle break .................................... 141
  7.3 frr with idle break .................................... 142
  7.4 mer with idle break .................................... 142
  7.5 sfr with idle break .................................... 142
  7.6 Hardware Windowing .................................... 142
  7.7 Higher resolutions ..................................... 144
7.7.1 Readout times across the detector surface .......................... 144
7.7.2 Chopped illumination ...................................................... 145
7.8 Bright Sources, High Speed .............................................. 147
7.9 Time between Reset and First Read .................................... 150

8 COORDINATE SYSTEMS .................................................. 151
8.1 Exposure Start .............................................................. 151

9 TROUBLE-SHOOTING ....................................................... 152
9.1 ROE Interface .............................................................. 152
9.2 Software ................................................................. 156
9.3 Operating System ......................................................... 161
9.4 Motor Interface .......................................................... 161
9.5 External Software ......................................................... 162
9.6 Recent Changes .......................................................... 164

A BEYOND GEIRS ............................................................... 166
A.1 Installment of a new ROE IP address .................................. 166
   A.1.1 Using RS232 .......................................................... 166
   A.1.2 Using ethernet ...................................................... 167
A.2 Image Rotation ........................................................... 168
A.3 Remote Sound ............................................................ 169
A.4 Network Time ............................................................ 173
A.5 X11 ......................................................................... 173
   A.5.1 Forwarding .......................................................... 173
   A.5.2 Tunneling ............................................................ 174
   A.5.3 vnc client .......................................................... 175
   A.5.4 NX client .......................................................... 176
   A.5.5 x2go ............................................................... 177
   A.5.6 Fonts ............................................................... 177
A.6 FITS ....................................................................... 178
   A.6.1 Chopping MEF .................................................... 178
   A.6.2 ds9loop ........................................................... 178
A.6.3 fits2csv ......................................................... 178
A.6.4 FTOOLS ....................................................... 181
A.6.5 ds9 ................................................................. 183
A.6.6 siril ............................................................... 183
A.6.7 SkyMaker ......................................................... 184
A.7 SVN installation ............................................... 184
1 OVERVIEW

1.1 Design

The Generic Infrared Software (GEIRS) is a software layer which

- assembles parameter lists and commands received from its own graphical interface or other supervisor software,
- translates these into the firmware language ("patterns") of the MPIA readout electronics (ROE)
- initializes the readout cycles
- and accumulates the frames received from the ADC’s of the electronics as FITS files or screen images.

GEIRS is

- neither a data pipeline or data reduction tool for an type of infrared images or detectors,
- nor a FITS display tool.

The generic attribute of the name illustrates that the core part of the software has been adapted to generations of the MPIA electronics which controlled various infrared detector chips in the past 20 years. In consequence, the command library is a superset of functionality released for a set of cameras in the past, and currently operating or under commissioning for

1. LUCI1 and LUCI2 at the Large Binocular Telescope under CentOS 7,
2. PANIC upgrade to Hawaii-4RG at the MPIA under openSUSE 15.1 or higher,
3. CARMENES on the Calar Alto under openSUSE 13.1 [1],
4. LN at the Large Binocular Telescope under CentOS 7.
5. a test camera with the older PANIC mosaic at the AIP, presumably under openSUSE 15.1 or higher,
6. the NTEimg and NTEspec upgrades at the NOT, presumably under Ubuntu 19.04 or nigher.

It also is used as a data acquisition and display tool in an experimental setup for Sidecar development. The development platform is openSUSE Leap 15.1 currently.

The software comprises pieces of instrument and telescope control software, as will become obvious and will be discussed at the subsection affected. Graphical user interfaces slavishly reflect—following established paradigms of good software practise—underlying batch processing capabilities, so some of the buttons or menus are either dead-ended, wiped out or set to invariable constants.

This document summarizes
the system setup (installation, compilation);

- the graphical user interface for the standalone setup, that is, the system running without supervision or interference by any camera control software [2]. This might be the least important part during production (after commissioning);

- the command interface;

A recent version of this document is in this PDF, the subversion system of the source code, and the GEIRS/version/doc subdirectory of the source code on the computers where GEIRS is installed. It describes the GEIRS release with the version imprinted on the footers of the man-pages in Section 5.5. Where instrument teams decided not to upgrade GEIRS any longer, one should not consult this documentation but the documentation of the applicable release.¹

The software is currently developed under openSUSE Leap 15.1 with gcc version 7.4, perl 5 (version 26) and PLX SDK 8.00. It does not contain parts constrained by (re)licensing: there is no IDL, Matlab, Mathematica, NAG or others.

1.2 Interfaces

The document complements the documents on the camera control software [2], the FITS format [3], ROE [4], first-stage pipeline [5], ICS [6], readout patterns [7], installation and pattern generator [8, 9].

Note that GEIRS is just a detector control system, usually governed by some higher instrument control software. That supervisor software may at any time modify, add or delete files or programs such that the information in this manual may appear to be invalid. In case of doubt, try to contact someone or to find some manual which describes these modifications for the particular instrument.

1.3 Operation

GEIRS is installed by adding drivers of the PLX board at standard places to the Operating System, configuring the allowable shared memory parameters, retrieving the source code from a SVN repository or the MPIA public ftp server, and compiling the source code with the GNU C/C++ and Java compilers.

GEIRS is started with a one-line command to the Operating System with an option to start with or without interactive GUI support. The configuration of essentially permanent parameters (TCP interfaces to the ROE, the location of files concerning patterns, sound control, etc.) is done in the very same startup-script. This needs of the order of ten seconds.² There is no “initialization sequence” because essentially all parameters concerning exposures are forwarded later.

Health of the GEIRS command interface and shared memory manager may then and at any latter time be checked by querying parameters with the status command. More tests by scanning the log files for prototypical answers from the ROE are possible if initialization tests are needed.

¹With the exception of Linc-Nirvana, MPIA has no control over instrument groups’ decisions to work with any particular GEIRS release...
²most of which is spent to upload default patterns to the ROE via the internet.
The standard operation of generating the images (that is, generating the FITS files) is to send a
sequence of commands to the GEIRS “shell.” There are configurational commands that specify
ROE parameters like integration times, integration/readout types, repetition factors, location and
size of windows in the geometry, and names of the FITS files. After such preparational step, the two
commands **read** (start ADC conversion and data transfer between ROE and the host computer),
and **save** (convert RAM-data to FITS file(s)) define the fundamental cycle of generating the images.
The configuration may be changed after each read-save cycle. This allows the higher level control
software to examine (the quality of) the FITS images before starting another exposure with the
same or modified parameters.

To simplify operations, any sub-sequence of these commands may be packed into macros (ASCII
files in a subdirectory) which are callable by a single command.

GEIRS is shut down by sending a **quit** command to the command interpreter.\(^3\) This leaves the
ROE in its most recently selected idle-mode (until powered off). Instruments specific aspects will
probably be bundled in a set of macro files related to scenarios like calibration/flat-fielding and/or
star magnitudes once the details of the windowing and timing patterns are fixed.

### 1.4 Acronyms

**ADC** analog-to-digit conversion

**ADU** analog-to-digital unit

**AIP** Leibniz-Institut für Astrophysik Potsdam [https://www.aip.de](https://www.aip.de)

**API** Application Programmer Interface


**CAHA** Calar Alto Astronomical Observatory [http://www.caha.es](http://www.caha.es)

**CARMENES** Calar Alto High-Resolution Search for M Dwarfs with Exoearths with
Near-infrared and Optical Echelle Spectrographs [carmenes.caha.es](http://carmenes.caha.es)

**ccw** counter clock wise

**CPU** Central Processing Unit

**DAC** digit-to-analog converter

**DCS** Detector Control System

**DEC** declination coordinate of the ICRF

**DICOM** Digital Imaging and Communications in Medicine

[https://www.dicomstandard.org/](https://www.dicomstandard.org/)

**DMA** Direct Memory Access

\(^3\)The various ways are to click the **shutdown** button in the controls GUI, to type in **quit** in the GEIRS shell, or
to use **quit** as the argument to the **geirsCmd** or to the **cmd_** Linux executables.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>Domain Name Service</td>
</tr>
<tr>
<td>EPICS</td>
<td><a href="http://www.aps.anl.gov/epics">www.aps.anl.gov/epics</a></td>
</tr>
<tr>
<td>FPGA</td>
<td>Field programmable gate array</td>
</tr>
<tr>
<td>FWHM</td>
<td>Full width at Half Maximum</td>
</tr>
<tr>
<td>GEIRS</td>
<td>Generic Infrared Software</td>
</tr>
<tr>
<td>GNU</td>
<td><a href="http://www.gnu.org">www.gnu.org</a></td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HDU</td>
<td>header-data unit (of FITS)</td>
</tr>
<tr>
<td>HEASARC</td>
<td>High Energy Astrophysics Science Archive Research Center <a href="https://heasarc.gsfc.nasa.gov/">https://heasarc.gsfc.nasa.gov/</a></td>
</tr>
<tr>
<td>ICS</td>
<td>Instrument Control Software</td>
</tr>
<tr>
<td>IDL</td>
<td>Interactive Data Language <a href="http://www.uni-giessen.de/hrz/software/idl/">http://www.uni-giessen.de/hrz/software/idl/</a></td>
</tr>
<tr>
<td>IIF</td>
<td>Instrument Interface of the LBT <a href="http://wiki.lbto.org/twiki/bin/view/SoftwareProducts/TCSsoftware">http://wiki.lbto.org/twiki/bin/view/SoftwareProducts/TCSsoftware</a></td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>LBT</td>
<td>Large Binocular Telescope <a href="http://www.lbto.org/">http://www.lbto.org/</a></td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>LINC-NIRVANA</td>
<td>LBT Interferometric Camera and Near-Infrared / Visible Adaptive Interferometer for Astronomy</td>
</tr>
<tr>
<td>LN</td>
<td>liquid nitrogen</td>
</tr>
<tr>
<td>LN</td>
<td>LINC-NIRVANA</td>
</tr>
<tr>
<td>LSB</td>
<td>Least significant bit</td>
</tr>
<tr>
<td>LTCS</td>
<td>Linc-Nirvana Telescope Control System</td>
</tr>
</tbody>
</table>
LUCI  LBT NIR spectroscopic Utility with Camera and Integral-Field Unit for Extragalactic Research http://www.mpe.mpg.de/ir/lucifer
MEF  Multi-extension FITS
MER  Multi-endpoint Read
MIDAS  Munich Image Data Analysis System
        http://www.eso.org/sci/software/esomidas/
        ftp://ftp.eso.org/pub/midaspub/
MPIA  Max-Planck Institut für Astronomie, Heidelberg http://www.mpia.de
NIR  near infrared
NOT  Nordic Optical Telescope http://www.not.iac.es/
NTE  NOT Transit Explorer http://www.not.iac.es/
PANIC  Panoramic Near-Infrared Camera https://panic.iaa.es
PCI  Peripheral Component Interconnect
PCIE  Peripheral Component Interconnect Express
        http://en.wikipedia.org/wiki/PCI_Express
PCI-X  Peripheral Component Interconnect eXtended
PDF  Portable Document Format
PLX  PLX Technology,
PSF  point spread function
RA  Right Ascension
RAM  Random Access Memory
RGB  Red-Green-Blue
RoCon  Readout Controller
ROE  Readout Electronics
SVN Subversion http://subversion.apache.org

TCP Transmission Control Protocol

TCS Telescope Control System

URI Universal Resource Identifier

UT Universal Time

UTC Universal Time Coordinated

WCS World Coordinate System http://atnf.csiro.au/people/mcalabre/WCS/

1.5 References

References


[4] U. Mall, C. Storz, CARMENES - NIR channel – Readout electronics and software, FDR-04C2A. E: in section 2.6.2 the factor 0.5 of the voltage divider is wrong. The actual value for the CARMENES racks is 0.699. (30 Jan. 2013).


2 INSTALLATION

Sections 2.1 and 2.4–2.5 discuss the setup for a first-time GEIRS installation or aspects related to upgrades of the operating system. Section 2.2 describes the installation and compilation of the GEIRS tar ball. The unxz, cd and INSTALL commands is all that is needed to upgrade to another GEIRS version!

2.1 External Software

2.1.1 Compilers

In case the person to install the operating system did not have have software development in mind and just went on with the standard distribution, various developer packages will probably be missing.

2.1.1.1 c++ The GNU C++ compiler is not distributed with the default layout of openSUSE. If

which g++

reveals that this is the case, post-install the packages with

zypper in gcc gccfortran gcc-c++ cpp
and the equivalent `yum` under CentOS or `dnf` under Fedora.


```bash
yum install centos-release-scl
yum install devtoolset-7
scl enable devtoolset-7 bash
```

### 2.1.1.2 flex

The flex compiler is not distributed with the default layout of openSUSE 13.1. If

which flex

reveals that this is the case, use

```bash
zypper in flex # openSUSE
apt install flex # Ubuntu
```

to post-install it.

### 2.1.1.3 Java

The Java compiler is not distributed with the default layout of openSUSE 15.1 or Ubuntu 19.04. If

which javac

reveals that this is the case, use

```bash
zypper in java-1_8_0-openjdk-devel # openSUSE
apt install openjdk-11-jdk # Ubuntu
```

to post-install it.

### 2.1.1.4 readline

The readline library is not distributed with the default layout of openSUSE 13.1. If the GEIRS installation does not find the header files, it compiles and installs its own copy of the library in its local directory; this is a waste of time. So it is recommended, if `/usr/include/readline/readline.h` is missing, to post-install the package with

```bash
zypper in readline-devel # openSUSE
yum install readline-devel # CentOS
apt install libreadline-dev # Ubuntu
```

### 2.1.2 boost

GEIRS uses the regex package of the boost library. If the library is not found under openSUSE it suffices to run `/sbin/yast2` the Software management submenue, to search for boost and to install the subpackage:
zypper install libboost_regex1_66_0-devel # openSUSE
yum install boost boost-devel # CentOS
apt install libboost-regex-dev # Ubuntu

2.1.3 gsl

GEIRS uses the Gnu Scientific Library for some parts of the CARMENES pipeline. If /usr/include/gsl is missing, you need to install the developer’s version, gsl-devel:

zypper in gsl-devel gsl # openSUSE
yum install gsl-devel $ CentOS

If the library is not installed, GEIRS will not provide the functions that need the library.

2.1.4 libtiff

GEIRS uses the libtiff Library for its program fits2tiff. If the library is missing, you need to install

zypper in libtiff tiff # openSUSE
apt-get install libtiff-tools # Ubuntu

otherwise GEIRS will not provide the fits2tiff command.

2.1.5 TwiceAsNice

This section is only relevant to Linc-Nirvana. If the environment variable INSROOT is set at compile time and the header file Ice/Ice.h is found, the GEIRS installation assumes that TwiceAsNice is available [10] and additional LN programs are compiled. In practise this means that GEIRS should be compiled after compiling TwiceAsNice.

2.1.6 Terminal Library

GEIRS uses texinfo which needs a terminal library. If it does not find any, it will compile its own local copy of ncurses, which is a waste of time. To avoid this, install at least one suitable package with

zypper in ncurses-devel # openSuse
yum install ncurses-devel.x86_64 # CentOS
apt install texinfo # Ubuntu

2.1.7 xterm Library

If
which xterm

indicates that xterm is not available (as apparently under newer CentOS), get it with

```
zypper in xterm-devel # openSuse
yum install xterm # CentOS
apt install xterm # Ubuntu
```

Otherwise GEIRS will try to use konsole instead, with limited flexibility. If the include file X11/Xlib.h is missing,

```
zypper in xorg-x11-devel # openSuse
```

Under Ubuntu apparently only gnome-terminal is installed by default which seems to be tricky to start without patching some settings related to the locale/language settings. For that reason gnome-terminal is not used by GEIRS and either apt install xterm or apt install konsole are required on Ubuntu.

2.1.8 Other

2.1.8.1 gnuplot. If the executable gnuplot is not found when GEIRS is compiled, all associated graphing functionality will be disabled. The recommendation is: if

which gnuplot

does not find the executable, install the package

```
zypper install gnuplot # openSUSE
yum install gnuplot # CentOS
```

2.1.8.2 Within GEIRS. Further external packages (cfitsio, CCfits, texinfo, sofa, xercesc, and parallel) in the GEIRS/branch/extern subdirectory are compiled later with the main source code. If the compilation of cfitsio does not succeed because no acceptable Fortran compiler is found, this may mean that /usr/bin/gfortran is missing. Use

```
zypper search fortran
zypper install gcc-fortran gcc48-fortran
```

to install the packages, or the equivalent yum on CentOS.

If the linker complains about missing libcui libraries, you need to install the dependency packages

```
zypper in libcui-devel libcui60_2 libcui60_2-1dedata # openSuse
```
2.1.9 Plx

Section 2.1.9 can be ignored if the software is installed on computers without OPTPCI boards, that is, computers that run GEIRS only in simulation mode.

The Linux driver for the PCI bus delivered by the manufacturer (PLX) of the main chip on the OPTPCI board (which is designed by MPIA) is expected to be installed in /usr/src, which needs root privileges. If these header files and driver libraries are not found at GEIRS compile time, the software will always run in ROE software simulation.

The following instructions are a summary of the documentation found in the directory Documentation/PLX/Linux_Release_Notes.htm of the driver. You are strongly advised to recompile the driver each time a kernel update was installed in /usr/src—which happens a few times per year under a well-maintained operating system.

Details may differ. In particular, the version will change as time progresses. The symbolic link installed below ensures that the header files are always found in /usr/src/Plx/Linux/PlxSdk/Include and that admin/plxload finds the driver to install. We build only the drivers for the two PLX chips that have been in use by the MPIA electronics: 8311 (newer, PCIe, OPTPCI-e, the relevant one for LUCI1/2, LN, PANIC, NTE and CARMENES) and 9656 (older, PCI-X, OPTPCI, still on duty on some MPIA computers). The manufacturer’s imprint on the fattest chip onboard the OPTPCI shows immediately which of the two types is in use.

The PLX drivers are currently not under SVN control. This is third party software and distribution of the complete SDK package is explicitly not covered by the license.

1. If this follows a fresh installation of the operating system, the kernel drivers in the directory /usr/src/linux-?.?.? (openSUSE) or /usr/src/kernels/ (CentOS) may be missing. This will lead to complaints of the form

   make: *** /lib/modules/3.11.6-4-desktop/build: No such file or directory. Stop.
   make: *** [BuildDriver] Error 2

   when the PLX driver is installed further down. This is the case if the following test does not find the build directory of the Linux distribution of the current system:

   unamer='uname -r'
   cd /lib/modules/${unamer}/build
   ls -l include

   This usually means that openSUSE was installed without the “developer” version of the kernel—which is one of the options while installing the OS but not included by default. This is basically cured by running /sbin/yast2, selecting the Software Management, the Repositories, and post-installing the kernel-desktop-* packages. On a freshly installed CentOS 7 the error message was triggered by an incorrect symbolic link to a non-existing build directory in /lib/modules/3.10.0-123.6.3.el7.x86_64, which had to be repaired. On a freshly installed Ubuntu 19.04 this obstacle does not appear.

2. We start from the Linux version distributed by PLX, log into the machine as root, and copy the BroadcomLinux_v*.tar.gz or PLX_SDK_LINUX_v*.zip file into /usr/src. Only installations with major number > 7.1 are supported. Then move into the GEIRS source directory and call
.

./INSTALL.plx

to compile the PLX driver.

3. To load the driver each time the computer is (re)booted plxload* is copied in the “old-fashioned” way to /etc/init.d/, then

chkconfig --add plxload8311
chkconfig --list

These steps are not needed and actually fail if no PLX device (read: no OPTPCI board) is found on the local bus system. Caveat: if this is automatism is not added, each invocation of GEIRS or any of the tests involving the OPTPCI board (i.e., everything beyond running GEIRS with ROE in simulation) needs to call either the wrapper script

plxstartup

or

/sbin/service plxload8311 restart

at least once (which needs root privileges). plxstartup tries to load two different device drivers for OPTPCI-X and OPTPCI-e boards, but only one type of boards is used for any type of computer, so the command will usually emit an error

Install: Plx9656

Load module......... ERROR: Load error or no supported devices found

This error should be ignored, because it refers to the type of board that is not applicable to the particular computer.

4. A simple check of successful loading of the driver is that

lsmod | fgrep -i Plx

contains the Plx8311 entry and that

/sbin/service --status-all | fgrep -i plx

contains a line which mentions loaded active (openSUSE) or loaded (CentOS) or [ + ] plx (Ubuntu).

Call

/sbin/lspci -v | grep -E 'Plx(8311|9656)' # CentOS openSUSE
lspci -v | grep -E 'Plx(8311|9656)' # Ubuntu

so see which boards are plugged into the computer.

If you have root permissions,
cat /proc/vmallocinfo | fgrep Plx

should show three lines for each OPTPCI board plugged into the computer. Starting yast2, moving into the Security center and hardening menu, selecting the Configure of Enable basic system services should also indicate the Plx drivers enabled. If lsmod does not show the driver, scan the system logs:

journalctl | fgrep plxload

Note that the chkconfig activates driver loading at computer run level changes; you won’t see the driver in the services until the next reboot or a manual interaction as in the previous bullet.

Each time the driver is recompiled, all GEIRS versions need to be recompiled—starting with the oldest—because they are linked with the binaries in the /usr/src directory, Section 2.2.2. If zypper up has installed a new Linux kernel, the steps are (as root)

chkconfig --del plxload8311 # remove (temporarily) to avoid boot propblems with old driver
reboot now # ensure that new kernel is active
INSTALL.plx # recompile PLX driver
chkconfig --add plxload8311 # add new driver to forthcoming reboots

2.2 GEIRS Compilation

2.2.1 Obtaining the Source Code and Patterns

• With subversion (SVN), the current (read: potentially unreliable) source is extracted with a script like

export CAMHOME=${HOME}/GEIRS
mkdir -p $CAMHOME
cd $CAMHOME ; svn checkout https://svn.mpia.de/gulli/geirs/src/trunk trunk

If the KWallet system asks annoying additional questions, you might disable it entirely by using the KDE application menu, System→KWalletManager→Settings→Configure KWallet.

There is no public read access to this repository. Requests to obtain rights on the repository need to be directed to Florian Briegel at the MPIA. The standard way of distributing the source code is that the GEIRS maintainer (currently the same as the author of this manual) obtains full access to the computer on which GEIRS is run, and installs the software there.

• If otherwise the source code is available in a compressed tar ball, move this into the CAMHOME subdirectory of the observer (Linux account) who will start and run GEIRS and eventually generate the FITS files with the data. This tar ball is the same for all instruments supported.

4The step that dives into the extern directory of the GEIRS source code can be skipped to save some time, because none of the external packages links with the PLX driver. The configure, make and make install steps in the top source need to be redone.
by GEIRS. If this is a first installation for an account, configure the environment as explained in Section 2.5, and re-login to activate these changes. Prepare for the compilation by unbundling it:

```bash
cd $CAMHOME
unxz -c *_r*.tar.xz | tar x
```

The `MACROS` and `scripts` directories are not under SVN and cannot be obtained that way (and do not need to be obtained that way).

### 2.2.2 Compilation

There is only installation support based on the GNU autotools. This works as described in the file `$CAMHOME/branch/INSTALL` in the source code, which is particularly designed to be executed. This is in general the only thing that needs to be done to upgrade the GEIRS version. If this is a first installation for an account, configure the environment as explained in Section 2.5, and re-login to activate these changes. Compile the source code:

```bash
cd $CAMHOME/... # move into the new _r*M-* source directory to be compiled
./INSTALL
```

If you are an inexperienced user or are installing GEIRS on unsupported Linux flavors, save the output so the installation process can be inspected later on:

```bash
./INSTALL |& tee INSTALL.log
```

This is all done under a generic non-privileged Unix/Linux account. The `INSTALL` script will ask with a `sudo(1)` command for permissions to modify two binaries just compiled. For test environments where GEIRS runs the data acquisition in simulation mode this is superfluous (and the `INSTALL` request may be cancelled with `CTRL-C`). For production code at the telescope, however, it is recommended to set the permissions to stabilize the real-time behaviour of the data acquisition.\(^5\)

There will be `Error 2 (ignored)` and `failed` messages related to packages mentioned in Section 2.1.8 which GEIRS will not install if equivalent packages are found in system libraries. Which system libraries are found depends on the operating systems, and even more on the attitudes of the individual system administrators to deal with software upgrades in general and the recommendations of Section 2.1 in particular. So `failed` messages are generally good because they indicate that GEIRS skipped (failed) compilation of packages because the system administrator maintained the standard libraries.

A second `./INSTALL` may run faster than the first because usually the libraries that were compiled in the first run are not recompiled.

This needs of the order of ten minutes. (This means there is no reason to cheat the installation by copying binaries or setting links or symbolic links between various Unix/Linux accounts.)

\(^5\)These `root` permissions can of course also be set by someone else in the `bin` subdirectory after the `INSTALL`.
To recompile a package, remove the entire _r*M-* versioned source directory, and call the unxz on the *.xz and the ./INSTALL again. So after any changes to system libraries, upgrades of the compiler and so on, we recommend to run the entire ./INSTALL, not just a make -f Makefile distclean ; make -f Makefile install in the source directory.

Starting from GEIRS version 759 or newer,

cd ${CAMHOME}/...._r*M-..
make distclean
./INSTALL

should have the same effect. Note that make -f Makefile install in the source directory would only recompile GEIRS but not the external packages.

This tar ball and the compilation step is the same for all instruments supported by GEIRS. Note that many links to the scripts directory are not installed by this step of the compilation/installation, but at the time when GEIRS is started. The simple reason is that the scripts that are available should be those depending on the GEIRS version that is run, not on the most recently compiled version. The decision on which instrument is started/configured is not done at compile time but later at startup time.

The installation should not be upgraded while GEIRS is running, because some files at common places will be replaced by the versions of the release that is compiled—for the same reason as the one mentioned in Sect. 4.1.

Compile GEIRS separately for each user. Never (!) cross-link or copy binaries from one account to another. The source code uses static variables and these would be shared if the binaries would be run by the different accounts at the same time (leading to interference effects between the concurrent GEIRS sessions).

The subdirectories admin and devel are not compiled with a standard installation.

By design, there are GEIRS features that depend on whether the source code is compiled on a computer with a MPIA IP address or not, for example

- The standard logging level is reduced outside MPIA;
- Default IP addresses change;
- Support of handling temperatures and pressures is reduced outside MPIA for instruments other than PANIC;
- Standard sets of operators (Figure 3) change.

If the account is set up properly (Section 2.5), you should be able to start GEIRS as indicated at the beginning of Section 3 — at least putting all components in simulation mode—and to get some images by pressing on the Read button of the controls GUI, Figure 5.

2.3 De-Installation

Any single GEIRS version suffices to run the instrument.
As with any other software old bugs are removed and occasionally new bugs appear as new versions are developed.

To de-install a GEIRS version remove the entire subdirectory of $CAMHOME with the subversioned name, which will be of the format trunk-r*. If you never want to see it again also remove the associated compressed tar ball. There are no GEIRS specific remnants in the standard system’s directories like /usr. Versions that are removed disappear from the options for the start_*_* and geirs_start startup methods.

This cleanup is recommended for all versions that have never been used for real-data acquisition at a telescope—to save disk space. This cleanup is almost mandatory each time the kernel of the operating system and the PLX driver have been upgraded—to avoid that operators start the old binaries that link to incompatible new PLX libraries.

2.4 Configuration of the Operating System

2.4.1 Shared Memory

The following paragraph is only of interest if the GEIRS computer is also running competitive programs that use shared memory for their databases and similar purposes.

Under openSUSE or CentOS, the available amount of shared memory is indicated by

```
cat /proc/sys/kernel/shmall
```
or

```
/sbin/sysctl -a | fgrep shm
```
or

```
ipcs -lm
```

As root, this may be momentarily changed by (sysctl(8))

```
sysctl -w kernel.shmall=...
```

To allow this configuration to persist through rebooting the computer, it is recommended to modify /etc/sysctl.conf like

```
kernel.shmall = ...
kernel.shmmax = ...
```

shmax is the maximum memory of a single allocatable chunk of shared memory in bytes, and shmall is the total allocatable shared memory in units of pages (where a page is typically 4096 bytes as indicated by the output of getconf PAGE_SIZE or the number of shmni generated above).

A full frame of a $2 \times 2$ chip comprises $4 \times 1024^2 = 4,194,304$ pixels, which amount to $2 \times 4,194,304 = 8,388,608$ bytes with a 16-bit ADC (LUCI,LINC-NIRVANA,NTE) or $4 \times 8,388,608 = 33,554,432$ bytes for a mosaic of 4 chips or a single Hawaii-4RG (PANIC,AIP) or $2 \times 8,388,608 = 16,777,216$ bytes for a mosaic of 2 chips (CARMENES).

---

6This is not recommended for versions that have actually been run in production because one might want to roll back and recompile if for instance the operating system and the drivers or the compiler have been updated.
The minimum requirements for the allocatable shared memory is roughly twice these numbers, because the software uses a scheme of two alternating buffers. These values may be taken from the `shmmanager:wanted` lines in the standard output created during startup (Section 3).

A guideline of the shared memory for production where GEIRS runs at most two instruments on the computer at the same time would be half of the total memory available on the machine. These numbers are obtained with

```
cat /proc/meminfo
free
```

under openSUSE or CentOS. The effect is basically a cap on the number of frames that can be swallowed at one time, so it puts limits on the “length” of the sample-up-the ramp modes, on the repetition factors of most modes and the number of pairs of Fowler modes.

### 2.4.2 Subnet

This subsection is obviously not GEIRS specific but a generic hint to configuration of the host workstation.

If the rack of the ROE electronics are given IP addresses on local networks, the file `/etc/sysconfig/network/ifcfg-eth0` (typically for openSUSE) on the GEIRS workstation needs to be augmented with the additional subnet(s) and mask(s) by lines of the format\(^7\)

```
IPADDR_ir2='192.*.*.*/*'
# LABEL_...='...' 
```

Details depend on how the GEIRS workstation is known to the subnet. This is tested by powering the devices up and `ping`ing the devices from the GEIRS workstation (`ping(1)`). On behalf of GEIRS there is no need to add a nameserver for these devices; working with the 4-byte numerical addresses in the startup-script suffices.

If such entries are missing, GEIRS cannot communicate via Ethernet with these devices.

### 2.4.3 journaling

It is recommended to enable access of GEIRS to the system journaling as detailed in Section 9.2 because GEIRS uses the `syslog(3)` to log informal and error messages. This is not strictly needed for a well-debugged GEIRS version, and the casual GEIRS user will not know what to do with that information. But the installation is necessary to work with the error and debug monitors of the controls GUI.

### 2.4.4 Shutdown

To terminate all GEIRS sessions gracefully when the computer is shut down normally,\(^8\) it is recommended to install the `admin/geirsStop` script in the manner of the PLX load script of Section

\(^7\)For PANIC at CAHA this is 192.168.70.1

\(^8\) `not` with `shutdown now` or from power outages...
2.1 into /etc/init.d. (For efficiency, the list of commands in that script may be adapted to the instruments of the computer.)

```bash
cp admin/geirsStop /etc/init.d
chkconfig --level 016 geirsStop off
chkconfig --level 235 geirsStop on
```

To remove this recognition of GEIRS for start/stop use

```bash
chkconfig --del geirsStop
```

2.5 User Configuration

2.5.1 Directory Layout

The standard directory layout of the GEIRS installation in the observers file system is a directory named GEIRS with subdirectories INFO, MACROS, OBJECTS, log and scripts and a selection of GEIRS versions which have file names that start with trunk and end with a SVN revision number and perhaps a subrevision number.

GEIRS

- INFO/
- MACROS/
- log/
- scripts/
- trunk_r694/
- trunk_r779M-50/

Each of the GEIRS versions contains a bundle of C/C++/perl/Java source files and binaries, and directories for the documentation and so on, after the step of Section 2.2.1 is finished:

GEIRS/trunk-r779M-50

- admin/
- bin/
- caha/
- de/
- devel/
- doc/
- share/
- test/
- *.cxx
- *.h
- *.pl
- Makefile.am
- INSTALL
- configure.ac
Some of the files in such a version are linked back to the `scripts` directory either when the version is compiled or when GEIRS is started. This concept keeps the mandatory executables at a single place (the `scripts` directory) for the benefit of a simple `PATH` variable, but also keeps them synchronized with the operators decision to launch a particular version.

### 2.5.2 Path

It is well advised to add `${CAMHOME}/scripts` to the path at the standard location; this would be

```bash
export CAMHOME=$HOME/GEIRS
export PATH=${CAMHOME}/scripts:${PATH}
export MANPATH=${CAMHOME}/man:${MANPATH}
```

in `$HOME/.bash_login` or `$HOME/.bash_profile` (but not both) for the bash(1), for example. Unfortunately there are users who let the environment ignore that setting because they chose their shells not to be login shells—as revealed by the `shopt` command. In these cases the `PATH` must be set in `$HOME/.bashrc` with constructions like

```bash
if [[ $BASH_SUBSHELL -eq 0 ]]; then
    export CAMHOME=$HOME/GEIRS;
    export PATH=${CAMHOME}/scripts:${PATH};
    export MANPATH=${CAMHOME}/man:${MANPATH}
fi
```

### 2.5.3 Standard Scripts

If a certain class of users should better not start some of the instruments, delete the associated symbolic link in the `scripts` directory of the user's GEIRS installation; this removes the command from the set of executables of the Linux/Unix account because it disappears from the search list of the `PATH`.

The file `GENERIC` is not just a startup script but a configuration script that defines many of the variables listed in Section 3.2. These defaults must be edited at least at two places:

1. If a ROE is to be used such that it is not simulated, `CAMPORT` must be changed to the address of the ROE. Once the instrument is run in a stable environment, the default address is known and ought to be compiled into the scripts of the SVN repository. For transient setups, one may also set the environment variable in the Linux shell before starting GEIRS, see Section 3.2.

2. The `CAMROE_REV` must be set to the existing pattern directory. This must be done even if the software is used in ROE simulation mode. The default is to use the pattern directory of the active GEIRS version.

---

9One reason is that the application launcher of openSUSE ignores the files `.Xresources` or `.xinitrc` where one would set the `Xterm*.loginShell` variable. A simple way to improve this is to add the `-ls` option to the `System->Terminal->Xterm` command when editing the openSUSE application launcher with a right-click, and to add that `xterm` to the Panel.
2.5.4 Hooked Scripts

GEIRS has 4 points of the command loop where programs (scripts in some interpreter or binaries, executables in the Linux sense) are executed. This serves to adapt GEIRS on a per-instrument basis to requirements that are not actually in the realm of a detector controller, and allows to synchronize the detector readout with other mechanisms of the telescope or instrument. These executables are in the `scripts` subdirectory and re-installed at startup with the script of the current GEIRS version. The executable may put itself into a background program to run asynchronously with GEIRS; GEIRS waits until the executable returns.

- **QueueAFiles** is called when the `save` command is executed and before the FITS files are created. The usual action here is to assemble the files with the complementary FITS keywords in the associated file.

- **QueueEFiles** is called when the `read` command is received, and before the detector readout actually starts. This may be used to adjust some optics of the instrument before the exposure. Linc-Nirvana uses this to move the derotator. **QueueEFiles** is actually only called when the 3rd bit (0=LSB) of the `GEIRS_FLAGS` integer in the shared memory data base is set, so it can be changed through supervisor programs with the `put` command (see Section 5.3). This bit can also be toggled with the `-Q` check box in the controls window (Section 4.3.1).

- **QueueFiles** is called when the `save` command has been completed. It might be used to display the new FITS file with ds9(1), trigger some action related to data archival, or start some data pipeline.

- **QueueZFiles** is called when the `quit` command is received. It may be used to add symbolic links in the data directory such that the FITS files are available under standard names of the observatory, or to extract some database from these FITS files.

Such a script may be not doing anything: a 2-liner like

```bash
#!/usr/bin/env bash
exit 0
```

or a 1-liner like

```bash
#!/usr/bin/env python

—made executable with `chmod(1)`—would implement that.
```

2.5.5 Shared Memory

Whereas the setup in Section 2.4 allows some maximum of the memory (real and virtual) to be dedicated to shared memory blocks by any applications on the computer, GEIRS needs also to be configured to request some (or all) of this when started. This is done by editing the size of the variable `CAMSHMSZ` in `$CAMHOME/scripts/Generic`, likely by setting it to some default of approximately 2048 depending on the name of the workstation. Typically this will be the integer obtained from
The main effect of this number is to limit the number of frames that can be held in memory for the standard non-continuous readout modes before releasing that space at the time of a save.

The GENERIC file uses defaults which are slightly dependent on the name of the workstation on which GEIRS is run. For LUCI there is a deliberate further divisor of 2 assuming that the two LUCI instruments may be run on the same computer.

2.5.6 Disk Allocation

There is no automated removal of administrative files by the software. Users need to look into the $CAMHOME/DATA directory, the $CAMTMP and in particular in $CAMHOME/log for obsolete and large log files left behind.

The amount of space required by various log-files depends in particular on the value assigned to LOG_LEVEL in configure.ac in the source directory. That default level depends on whether the source code is compiled on a computer with MPIA IP address or elsewhere.

Some files grow without bounds, so it is useful to split them into subfiles in regular intervals (with crontab(1) for example) one time per day when the instrument is not used. A shell script to automate this is proposed in GEIRS/<branch>/admin/glogRotate.sh and installed with INSTALL if missing. If

1. glogRotate.sh is copied to $CAMLOG—where CAMLOG is usually $CAMHOME/log—,
2. this is made executable with chmod +x glogRotate.sh, and if
3. the associated entry as proposed in glogRotate.sh is added with crontab -e into the schedule of the usual account that runs GEIRS,

this infinite growth of files is limited by the daily growth.

Since GEIRS version 769M-27, the logs of the main program are written with syslog(3) to the journal files, no longer to $CAMLOG. Only (i) the PANIC temperature logs, (ii) the lists of the commands received by the command manager and (iii) the messages exchanged with the ROE remain in CAMLOG.

2.5.7 info

The info file camera.info is available which is basically supported by adding also

```
export $CAMHOME=$HOME/GEIRS # assumes default directory layout
export INFOPATH=${INFOPATH}:${geirs_build}
```

into the $HOME/.bash_login such that
info camera

of info(1) will also find the help file of Section 5.3.

2.5.8 Sound Configuration

GEIRS generates sound by playing the audio files in $CAMHOME/<branch>/admin/*.au at certain events unless

1. the sound level within GEIRS is set to zero in the Options submenue in Figure 5 or with the sound command (Section 5.3).
2. the sound is muted with the sound/mixer application on the user’s desktop,
3. GEIRS runs on a remote computer and sound is not forwarded to the user’s desktop (Section A.3),
4. the environment variable CAMAUDIOPLAY was not set (in the startup scripts).

History shows that the people who install GEIRS usually fail to test and install their (remote) sound configuration on the GEIRS workstation, so the sound volume is initially switched to zero for new users to avoid any followup problems. If the setup is not installed properly and sound is switched on (measured according to the criteria listed above), it will likely happen that at the first time a sound is configured to be played, the system call to play that sound will crash, which will trigger a followup error because this will attempt to play crash.au, which will not succeed and eventually turn into a recursive endless cascade of sound errors.

The sounds may be changed by replacing the audio files in the GEIRS file system in that directory.

<table>
<thead>
<tr>
<th>Sound File</th>
<th>triggered by...</th>
</tr>
</thead>
<tbody>
<tr>
<td>doorbell.au</td>
<td>readout finished</td>
</tr>
<tr>
<td>cuckoo.au</td>
<td>macro finished</td>
</tr>
<tr>
<td>bong.au</td>
<td>backup or the ‘shift-and-add’ calculation finished</td>
</tr>
<tr>
<td>crash.au</td>
<td>general error</td>
</tr>
<tr>
<td>fastbusy.au</td>
<td>warning (at changing user level to engineer or if near saturation)</td>
</tr>
<tr>
<td>whistle.au</td>
<td>save completed</td>
</tr>
<tr>
<td>sorrydave.au</td>
<td>unrecognized command</td>
</tr>
<tr>
<td>touchtone.0.au</td>
<td>disk full</td>
</tr>
</tbody>
</table>

The executables charged with the sound creation are weakly configurable with the two CAMAUDIO environment variables of Section 3.2.

10Those problems can be re-introduced if software-engineers just copy GEIRS from one user account to the other; this practise is very bad and entirely discouraged.
3 INVOCATION

3.1 From workstation or remotely

Call the $CAMHOME/scripts/start_*_new that matches the instrument name, which is $CAMHOME/scripts/-start_carmenes_new for CARMENES. The full path name is not needed, of course, if the environment has been set up as proposed in Section 2.5.

This will create directories and files like $HOME/tmp and $HOME/DATA and $HOME/*_.log if these do not exist. To relocate source, data and logging directories, edit the associated environment variables in $CAMHOME/scripts/Generic or set them before starting GEIRS.

The principal ways to control the electronics via GEIRS are

1. Interactive manipulation of parameters and exposures with the GUI;
2. Interactive submission of commands with a text interface to the GEIRS “shell” (Figure 9). This interface is richer than the set of GUI buttons because many commands do not have a perfectly equivalent button.
3. Commands sent from the computer on which GEIRS is running from the UNIX/Linux shell with

   cmd_carmenes cmd arguments [; cmd arguments...]
   snd_carmenes [-s server[:port]] [-p port] cmd arguments [; cmd arguments...]

   or

   cmd_carmenes_new cmd arguments [; cmd arguments...]
   snd_carmenes_new [-s server[:port]] [-p port] cmd arguments [; cmd arguments...]

   or

   cmd_carmenes_old cmd arguments [; cmd arguments...]
   snd_carmenes_old [-s server[:port]] [-p port] cmd arguments [; cmd arguments...]

   or

   geirs_cmdClient [-s server[:port]] [-p port ] [-v] [-fi|fc] cmd arguments [; cmd arguments...]

The difference between using or not using the _new and _old suffixes is that the start script sets the CAMBIN environment variable to different subdirectories of CAMHOME so one can conviently keep a set of different GEIRS versions in the CAMHOME subdirectory.

The cmd_ versions connect to the shared memory database of a GEIRS command interpreter running on the local machine; no TCP socket is used—as one may guess from the absence of the corresponding command line options. To this effect it uses the shared memory socket created by the same user in $CAMTMP when GEIRS was started; this basically avoids interferences if multiple users are running multiple GEIRS instances on the same computer. For the Luci instruments the standard installation in Section 2.5.3 will create indexed versions cmd-_luci1 and cmd_luci2 of the command, and this may lead to confusion: because cmd_ looks up in the user’s ~/tmp/shmsocket to which port to connect, the index of either cmd_luci1 or cmd_luci2 does not select the instrument. The instrument is the instrument the Linux/Unix user calling the cmd_ actually started most recently.
The `snd` interfaces and `geirs_cmdClient` are essentially the same, where `snd` calls `geirs_cmdClient` which is based on TCP sockets. `snd` are shell scripts and supposedly a little slower, but they offer a slightly finer control of which shell variables and GEIRS versions are used while executing a command.

4. Commands sent from a remote computer from the UNIX/Linux shell with

```
geirsCmd [-t timeoutSeconds] [-s server[:port]] [-p port] cmd arguments ; cmd arguments...
```

The standard port is 8501 for `geirsCmd` and taken from the port entry in the user’s shared memory socket on the `server` for `geirs_cmdClient`.

Using another port—for example for running multiple instances on the same computer—is supported by starting the `cmdClient` in `GENERIC` either with the switch `-s server:port` or with the switch `-p port` or modifying the `CAMSERVERPORT` before starting.

`geirsCmd` uses a TCP socket interface which “represents” the same set of commands as the other interfaces. On the GEIRS computer, the sockets are managed by the `cmdServer`, which is started by either one of the `start*` commands or checking the `-cmd` option in the engineering GUI (Figure 4). `geirsCmd` is indeed just a wrapper which uses that socket interface to submit commands to the `cmdServer`.

The `snd` versions and the `geirsCmd` both use a socket interface for the command and answer. `snd` needs an active (=started) GEIRS sessions on the local computer to hook into and uses the port number registered with the shared memory socket at GEIRS startup as a default, whereas `geirsCmd` can contact a GEIRS session running on any remote computer reachable via the network.

5. Any other fundamental socket connection. A `telnet(1)` example looks like

```
mathar@mathar:~> telnet irws2 8501
Trying 149.217.42.24...
Connected to irws2.
Escape character is '^]'.
status
GEIRS_reply_2.0 694
itime: 2.7399310505
cycle-type: lir
cycle-repeat: 1
coadds: 1
cctime: 5.4812006566
last-filename: <unknown_not_yet_saved>
next-filename: trash_0001
autosave: off
...
error: NONE
version: carmenes@irws2: trunk-r737M-7 (May 20 2015, 17:48:39) (SINGLE) (/home/carmenes/GEIRS/trunk-r737M-7/bin, Carmenes_r9M)
status itime
GEIRS_reply_2.0 20
itime: 2.7399310505
```
The replies contain a header line starting with `GEIRS_reply` (a version number, a blank, and the number of bytes in the main body, including any line feeds), plus one or more lines in the main body.

If you wish to talk to GEIRS via that socket interface, be aware on fundamental Linux design issues, in particular the timeout parameters shown with

```
cd /proc/sys/net/ipv4
cat tcp_keepalive_time
cat tcp_keepalive_probes
cat tcp_keepalive_intvl
cat tcp_retries2
```

If your client interface does not get answers from GEIRS, your client may have been idle too long, and this is not an error of the GEIRS server, see RFC 1122.

Brackets indicate that switches and/or multiple command-argument lists are optional. Quotation marks around the command lists are usually required to avoid that the shell of the operating system splits the lists.

The `server` argument is either a simple name of the workstation on which GEIRS is running (if supported by a DNS) or a plain `tcp://x.y.z.w` IP specification.

If GEIRS has been started without opening the GUIs, inserting `quit` for `cmd` above is the recommended way of shutting GEIRS down.

Note that at GEIRS startup a single (one and only one) command port is activated to which the server listens. The `snd` and `geirsCmd` methods open and close their (client) ports for the duration of their isolated commands. This ensures (to some degree) proper sequentialization of commands and answers. The variety of other possible socket connections to that port will become very confused if a mix of these access methods is used. A standard indicator of that murky situation is that commands do not receive replies because the port is kept open by another client. In short: do not open the port if it is already used by another client.

### 3.2 Environment Variables

The configuration if GEIRS is steered primarily by setting environment variables (in the standard Unix/Linux sense of the shell) during the startup phase and later on by communication of the subprocesses via a shared memory data base.

The fundamental values of environment variables may have been set outside GEIRS with the standard mechanisms
• during login (the files .bashrc, .bash_login in the home directory and equivalent locations),
• with the export command.

A refined set of variables is then established in a second step within either

• the start.*, snd.* or cmd.* scripts or
• the geirs_start GUI.

In a third level, the shared memory manager starts with an internal set of default values, and overrides these with values set during the second step. As a side effect of that procedure, changing these fundamental parameters channeled trough environment variables requires a GEIRS shutdown and restart.

The following shell environment variables may be set in the start.* scripts to configure defaults of the behavior of the software:

CAMAUDIOMIX The name of the mixer of the audio files, for example aumix. If the variable is not set, no mixer will be used.

CAMAUDIOPLAY The name and options of the executable that plays the sound files, for example paplay, aplay -d 5 -N -q, auplay or audioplay. This specifies the full command stripped off its final parameter (the file name), such that attaching the name of the sound file and redirecting the standard output is a valid system call. See also [11].

CAMBIN The name of the subdirectory of $CAMHOME with the compiled code. This is the bin subdirectory of a subversion branch name, like ~/GEIRS/trunk_r713M/bin. Whereas the variable CAMHOME usually remains fixed for the operator, CAMBIN is chosen as one of these subdirectories when GEIRS is started; this allows switching between different releases of the software.

CAMBROWSER Full path name to a HTML browser. Only used if the online help is called with the button as in Section 5.3 or for the air mass plotting in the GUI of Figure 15.

CAMDATAPORT IP port number of the data server that submits data to the real-time display. The startup script defines the standard port and echoes its value to the standard output. After GEIRS is started one can send

  get DATAIPPORT

to the GEIRS server to ask what its current port is.

CAMDPORTS The number of PCIe channels and fibers set up for the transfer of the ADC data from the ROE. This is 1 for all cameras with a single chip (LINC-NIRVANA, LUCI and NTE), 2 for PANIC, AIP and for CARMENES. The basic advantage of using two channels (which at the same time implies using both fibers of the connection from the ROE to the computer) is that the data transfer is more stable.\textsuperscript{11}

\textsuperscript{11} Related to the existance of a 128 kB FIFO on the OPTPCI at the end of each channel/fiber that feeds into the PLX. At a standard readout frame period of 1.3 seconds, the net 16-bit data stream from the ROE to the computer
CAMERA  The master configuration label, which is either Nirvana, Panic, Carmenes, Luci2, Luci1, NTEimg, NTEspec, Aip or SIDECAR. Other names are not supported and obsolete.

CAMHOME  The top level directory of GEIRS. It contains at least one INFO subdirectory and one log subdirectory.

CAMICEPORT  IP port number of the ICE server. Only relevant if GEIRS is integrated in the LN operation, and that server may be queried by other servers. For LN any change of that number must be reflected in the listing for the geirs-svr in the lnsw/config/alias-lbt.cfg and lnsw/config/alias-lbto.cfg files so the clients will find that server.

CAMINFO  A subdirectory for configuration purposes, typically $CAMHOME/INFO. It also contains bad pixel masks, and gnuplot command sequences.

Cammotserdelay  Delay between transmission of individual bytes on serial lines connected directly (through a line connected to the GEIRS computer) to motors.

CAMPORT  IP port of the ROE as a string of the tcp://xxx.xxx.xx.xx:4000 format. Empty or not set if there is no ROE rack such that this interface will be used in software simulation. The modification of this address on the ROE side via its interfaces is described in [12, Sec. 4.1.2][13] and Section A.1. The port number (4000) cannot be changed—there is no scenario where one would have to change it.¹²

Wherever GEIRS is run, it must be able to connect to the ROE that controls the detector via the Internet; for testing purposes only, a control through the RS232 serial interface is possible (Section 9.1). The fiber pair from that ROE must lead back to the expected OPTPCIe board without swapping the two fiber heads. The fiber connection does not use any sort of network protocol but bare 16-bit data, so it cannot work through any type of hubs, routers or switches; it must be direct in the physical layer in that sense, permitting only patch panels, ST connectors and so on to cross between laboratories. Note that the DATAIMPORT1 assignments are dynamic: if any OPTPCIe board is removed from the computer, the remaining one is always addressed as /dev/plx00.

If a spare ROE rack is available, there are various options to swap it in:

1. remove the old ROE (switch off, at least disconnect from the Ethernet to avoid duplicate use of the IP address), modify the IP address of the spare to match the default IP address as instructed in Section A.1, put the spare into the network,

2. or modify the CAMPORT shell environment variable of the account that starts GEIRS to match the new ROE’s IP address before starting GEIRS, for example

   export CAMPORT="tcp://192.168.0.34:4000"
   start_luci1_new

The export command can be inserted into the ~/.bashrc or ~/.bash_login of the account. This is the recommended variant because it needs the least amount of human interaction and is easily reverted;

¹² unless communication to the ROE is mediated by an interface similar to Figure 25.
Figure 1: The ROE sends the digitized pixel data of one of the two detector chips through one and the digitized pixel data of the other detector chip through the other fiber of a fiber pair. Each of the two CARMENES computers may receive data from any of the ROE’s if GEIRS is configured with the CAMPORT variable to talk to the ROE that generates the data and if the fiber that streams the digitized data ends up at the correct OPTPCIe board configured with the DATAINPORT1 and DATAINPORT2 variables. The two CARMENES ROE’s have the same IP address, which means they must not be used at the same time on the same subnet.

3. edit the IP-address in the GENERIC script by an ASCII editor before starting GEIRS, if this is a permanent change, make sure that the GEIRS maintainer also modifies the SVN source code so upcoming GEIRS versions know about this...

4. start each time with geirs_start and edit the CAMPORT entry before continuing.

Replacement of the ROE rack always requires shutting down and re-starting GEIRS.

CAMROE_REV The name of a subdirectory of $CAMBIN/../.pttrns with the patterns to be applied. If the variable is not set, a default is used which is equivalent to the name of the camera, either Panic, Carmenes, Luci2, Nirvana, Luci1 or Aip. There may be more than one of these subdirectories to allow switching between different pattern versions. Examples: Panic or Panic_r74 or Panic_r76 for PANIC. Carmenes or Carmenes_r5 for CARMENES. Nirvana or Nirvana_r98 for LINC-NIRVANA. Luci1_r19M or Luci2_r20 for LUCI.

CAMSHMSZ Shared memory (in MBytes) reserved for use by GEIRS, see Section 2.5.5. This is roughly aligned with the total available RAM of the host computer via

```
setenv CAMSHMSZ `cat /proc/meminfo | fgrep MemTotal | awk '{printf "%d",$2/2048}'`
```

in scripts/GENERIC. The divisor is basically 1024 (to convert KiB to MiB) multiplied by some rather arbitrary small factor of the order of 1 or 2. It might be adjusted if concurrent
data acquisitions (more than one GEIRS session) are run by multiple users or for multiple ROEs at the same time. This sets an upper limit of the number of frames and images that can be acquired without intermediate save operations.

**CAMSERVERPORT** IP port number of the command server. The startup script defines the standard port and echoes its value to the standard output. After GEIRS startup one can test with a command in the style of

```
nc -v -z server port
```

from the Unix/Linux shell whether GEIRS is actually using that port. One can send

```
get CMDIPPORT
```

to the GEIRS server to ask what its current port is—this may not be useful because to submit the get to the correct server implies that one already knows the port...).

**CAMSERIALDELAY** Delay between transmission of individual bytes on serial lines. Usually irrelevant because commands are send to the ROE via Ethernet.

**CAMSERIALEOL**

**CAMSERIALEOL_RD** Number of end-of-line characters for serial communication with the ROE (reading). Usually irrelevant because commands are send to the ROE via Ethernet.

**CAMSERIALEOL_WR** Number of end-of-line characters for serial communication with the ROE (writing). Usually irrelevant because commands are send to the ROE via Ethernet.

**CAMSERIALSPEED** Baud rate of serial communications with the ROE. Usually irrelevant because commands are send to the ROE via Ethernet.

**CAMTMP** The name of the directory for temporary files. If not set explicitly, set to $TMPDIR, $TMP or $HOME/tmp in that order, depending on whether the enviroment variables TMPDIR or TMP are set.

**CAMWWW** The full path name of the HTML help file for use as in Figure 6.

**CAM_CHIPGAPX** Size of the vertical gap between the Hawaii2 RG chips in the mosaic in units of pixels. If not set, a default of 167 is assumed. Only relevant for the AIP detector hardware.

**CAM_CHIPGAPY** Size of the horizontal gap between the Hawaii2 RG chips in the mosaic in units of pixels. If not set, a default of 167 is assumed. The two chip gaps are used to span a WCS coordinate system across all four chips in PANIC’s MEF headers. Only relevant for the AIP detector hardware.

**CAM_DETROT90** A number from 0 up to 3 (inclusive) to trigger rotations of the detector image by a multiple of 90 degrees to the right. (The fact that these rotations are clockwise is a consequence of GEIRS using a left-handed X11-type coordinate system acting on some internal index tables.) Defining a value of zero is equivalent to not setting the variable at all such that GEIRS falls back to the default of a non-rotated output. This effects both, the views within the engineering GUI’s described in this manuscript as well as the pixel distribution in the FITS files. The default for CARMENES is either 1 or 3 to generate a view where the two detector chips are aligned left-right (not up-down). The design of the detector mount allows to re-install the detector rotated by 180°, so whether the standard is 1 or 3 depends (i) on
the actual way of detector installation and (ii) on the various opinions of interested parties
to flip the wavelength and order axes.

The CARMENES 2-chip layout is special insofar the switch of DET_ROT90 from 1 to 3 is
similar to a swap of the fiber heads (at least if the default of CAMDPORTS=2 is used). This
cannot be used for a lazy correction of a wrong fiber connection, though, because features
like the reset windowing (Section 5.6.1) operate on a dedicatedly enumerated order of the two
Hawai chips: wrong fiber connections let the reset windows appear in the wrong corners of
the display and FITS files.

**CAM_DETXYFLIP** If set to 1, this commands a left/right reflection of the images along the
vertical axis. If set to 2, this commands a up/down reflection of the images along the hori-
zontal axis. If not set or set to zero, there is no flip. If set to 3, the two flips are combined
and replaced by a rotation of 180 degrees.

In combination with the previous keyword, this supports eight orientations of detector images—
the basic mean to obtain a (rough) standard image orientation along N and E in the images
(Sect. A.2). Rotations and reflections are not commutative: the rotation will be executed
first.

The combined action of **CAM_DETXYFLIP** and **CAM_DETROT90** on the default orientation of the
chip—as displayed in the manufacturer’s manuals—is shown in Figures 2–4. A posteriori
these two integer values can be read from the FITS header of the data files.

Note that swap of the two fibers that transport the data from the ROE rack to the GEIRS
computer (on any of the two sides) cannot be replaced or undone by any combination of the
**CAM_DETROT90** and/or **CAM_DETXYFLIP** keywords.

**CAM_HINVDIR** The bits in this non-negative integer value indicate the left-right directions in
the horizontal scanner. For the HAWAII-2RG the least-significant 8 bits are relevant, for
the HAWAII-4RG the least-significant 2 bits are relevant (equivalent to HINVDIR[8..9] after
left-shift), and for the HAWAII-2 the value is irrelevant. More specifically, the relevant bits
for HAWAII-2RG depend on the number of channels (see the Teledyne manuals for details):

- 32 channels: all 8 bits are relevant.
- 4 channels: bits 0, 3, 4 and 7 are relevant.
- 1 channel: only bit 0 is relevant.

If not set, a value of 0 is assumed, meaning all channels are read left-to-right.
**CAM_VINVDIR** This is an integer value of either 0 or 1 indicating the top-bottom direction of the vertical scanner for HAWAII-2RG or HAWAII-4RG types. For the HAWAII-2 the value is irrelevant. If not set, a value of 0 is assumed, meaning all channels are read top-to-bottom.

**CAM_IDSTR** A string generally used in frames of GUIs. Useful if one switches between two similar instruments both run by GEIRS at potentially the same time, like LUCI or NTE.

**CAM_MAX_EDTBUFSIZE** Defines the size of a single buffer in the ring buffer in units of kilobytes.

**CAM_NADC36** Number of ADC36 boards in the ROE rack. By default this is 4 for AIP, 2 for CARMENES and PANIC, and 1 for the other configurations.

**CAM_NDET** Number of infrared chips, and—with the exception of AIP and CARMENES—always 1. If the parameter is set to 1 for CARMENES, the GEIRS software will treat the entire readout system as if only the SCA1 detector were present, triggering only the ADCs on one of the two ROE boards, receiving data only through one of the two fibers, showing only a $2048 \times 2048$ image and so on.

**CAM_NQCHAN** Number of output ports of each detector chip. By default this is 64 for Hawaii-4RG configurations, and 32 for the Hawaii-2 and Hawaii-2RG cases. There is preliminary experimental support for 32 for Hawaii-4RG, 16 for Hawaii-2RG and 4 for Hawaii-2.

**CAM_NORTH** North direction in the images in the FITS files measured in degrees ccw from $+x$. If not set, a default of 90 is used. The number is used to construct/predict a WCS coordinate system across all four chips in PANIC’s MEF headers.

**CONTRLX, CONTRLY** Horizontal and vertical X11 coordinate of the preferred startup position of the Controls GUI. Here X11 means that the upper left corner of the screen is at (0,0).

**DATAINPORT1, DATAINPORT2** Pseudo-device name in the Unices sense used by GEIRS to indicate on which of the OPTPCI board(s) and which fiber the 16-bit pixel data arrive. Almost always plx-00 and plx-01 unless more than one OPTPCI board are plugged into the computer. The first (left) of the two digits enumerates the OPTPCI boards on the GEIRS workstation starting at 0. The second (right) of the two digits enumerates the two fibers/DMA channels, 0 or 1. (The physical layer of the data/fiber connections from the ROE to the computer comes always with fiber pairs.) For instruments with only one fiber/DMA channel (Luci, Linc-Nirvana, PANIC or CARMENES with CAM_NDET=1, NTE), the second (right) number is always 0, and DATAINPORT1=/dev/plx?0. For instruments with two fiber/DMA channels (AIP with CAM_NDET=4, PANIC with CAM_NDET=1, and CARMENES with CAM_NDET=2), DATAINPORT1=/dev/plx?0 and DATAINPORT2=/dev/plx?1. The software does not support feeding the two fibers of one instrument into two different OPTPCI boards, so the first (left) of the two digits of DATAINPORT1 and DATAINPORT2, represented by the question mark above, needs to be the same. If the startup scripts detects that the first (left) of the digits is larger than what is supported by the number of OPTPCI boards currently plugged into the computer, it patches the DATAINPORT variables to match that reality.\(^{14}\)

The two digits of this pseudo-device name are not related to the MPIA serial number on a small sticker on the board.

---

\(^{14}\)This is currently only relevant if LUCI is started on lucix.luci which only has one board.
**DISPLYX, DISPLYY** Horizontal and vertical X11 coordinate of the preferred startup position of the Realtime Display.

**MOTPORT** Ports for direct communication with the motors (filter wheels etc.). This is a comma-separated list of values, one per MoCon board under GEIRS control. The parameter should be left blank if GEIRS does not control motors. This means it is only relevant to PANIC, which addresses the four filter wheels and the cold stop shutter through the first in this address list.\(^\text{15}\)

**TELESCOPE** The label of the observatory, which is used to set the geographic coordinates and to convert from equatorial to topocentric coordinates. Only a few fixed strings are supported: LBT, CA3.5m, CA2.2m, NOT, Lab, GENERIC and some obsolete others.

**TEMPORT** Port for direct communication with the temperature and pressure sensors. This is only relevant as a default for the crontab job (i.e., the executable `pantempress` that reads PANIC temperatures and pressures if the command line option `-i` is missing and if the default IP address of CAHA is not to be used. Only relevant to PANIC.

**TMOUT** If the variable is set and larger than zero, it indicates that GEIRS should shut down if it is idle for that many seconds, which means if no read command is received for that duration. Note that this is deliberately the same variable as in the bash(1).

This list is mentioned for documentation purposes. Not all combinations of cameras and variables are supported or meaningful. In case of doubt it is recommended not to set a variable.

These variables are set in the startup script and exported, so they are defined in the child subprocesses; they are not exported “up” to the calling operator’s shell—there is no mechanism in Unices for such modification in the other direction.

Editing the actual startup script is not recommended because any new GEIRS version will overwrite `scripts/Generic` with its current version. If long-term changes are required, contact the GEIRS maintainer to have these added to `GENERIC`, and use exported shell variables in the meantime.

The generic strategy in the `GENERIC` script is to honor (not to change) variables which are already set when the script is called. This allows users with lesser knowledge of shell scripting to configure/set the variables at other places, for example immediately before calling the script or in the standard files like `.bashrc` or `.bash_login`. Another use of this feature is that one can call GEIRS versions that are older than the most recent three ones or one can invoke pattern versions that are older than the most recent one. Here is an example in the case of LN started from a bash(1) shell:

```
export CAMBIN=${HOME}/GEIRS/trunk-r784M-17/bin
start_nirvana_old
```

A further aspect is that one can run GEIRS sessions in parallel on the same computer by different Unix/Linux accounts without interference, if the communication channels from the observer tool to the GEIRS server and from the GEIRS server to the ROE are kept separate, and if the computer is equipped with at least as many OPTPCI boards as active (=non-simulated) ROE’s:

```
export CAMSERVERPORT=10501
```

\(^{15}\)At MPIA, the address is found with `nslookup elotest`. 
export CAMPORT=tcp://192.168.0.14:4000
export DATAINPORT1=/dev/plx00
start_luci2_new
export CAMSERVERPORT=9501
export CAMPORT=tcp://192.168.0.24:4000
export DATAINPORT1=/dev/plx10
start_lucil_new

(Note that this is just an example. Variables will differ for the real instrument depending on hardware configurations!)

In summary: all major parameters are equipped with defaults (which depend on the instrument). If the defaults do not represent the current hardware configuration—because someone changed ROE IP addresses, re-plugged fibers and so on—the GEIRS parameters should be changed either with the Linux shell export commands as illustrated above before calling the start script or by modifying them through the startup GUI (Section 4.2.)

The parameters of the GEIRS server are a combination of

1. exported shell environment variables;
2. modifications of the environment variables by the engineering GUI in Figure 4;
3. modifications of the environment variables by the startup scripts;
4. modifications of the availability of subsystems (simulation) defined in the GUI in Figure 3;
5. defaults stored in the CAMTMP directory at a previous shutdown.

3.3 Postprocessing

An infinitely rich interface to post-processing the data, starting pipelines or archival systems is offered by the script or executable located in QueueFiles on the GEIRS computer. (The file QueueFiles may be anywhere in the $PATH but is usually in $CAMHOME/scripts/QueueFiles.) It is called at the very end of every save command (but not at the end of saving the intermediate frames configured by the sfdump command). It receives two parameters, the file name of the file created by that save command, and a number indicating the number of files expected to be created by that save command. (The latter offers some means to postpone actions in that script for example if GEIRS constructs a series of files with one window per file.) These two parameters are available in the script as $1 and $2 in the common Unix/Linux shells, or in the argv vector of higher programming languages if one would replace the shell script by any binaries.

The features of that architecture are:

- At the point in time when QueueFiles is called, the FITS files are already closed. So instead of polling the status of the crep counter or any similar status variable, or polling the file system for any new files that arrive, it is safer and less disruptive to trigger pipeline actions by adding them to the script.
• The `save` command is finished when `QueueFiles` terminates. If foreground commands in `QueueFiles` hang, `save` does not terminate—which might lead to the wrong conclusion that GEIRS hangs whereas it actually waits.

• As already said, `QueueFiles` is called synchronously with the `save`. Within this script, however, further actions may be pushed into background processes such that they are effectively becoming asynchronous to the GEIRS processing.

• The `sync` and `sync save` command wait on the `save` command, so the delay depends implicitly on the timing chosen within the `QueueFiles`.

• The `QueueFiles` must be a valid script and of course be executable as usual in the Unix/Linux sense. It may be empty—aside from comments etc.—if there is nothing to be done.

• There is only one `QueueFiles`. If instrument pipelines or monitors need variable actions depending on other than the two variables forwarded as command line arguments, they either need to edit/move/remove the `QueueFiles` dynamically—cautiously synchronized with the `save`—, or gather more information from the shell or user environment and use standard branching/switching statements of the shell.

Examples of actions in the `QueueFiles` are ds9 calls (Section 4.3.3) or examination of test files with the script in `test/QueueFiles` of the source directory. PANIC uses this file to add CAHA ambient data to the place where forthcoming `save` processes pick up additional FITS information. This interface is a specialized (by time and place of the invocation) call to the operating system. The `system` command (Section 5) to the shell offers the more flexible and general interface.

### 3.4 Concurrent Sessions

*Section 3.4 is mainly of interest for LUCI in binocular mode and potentially for the NTE cameras.*

Multiple GEIRS sessions may in principle be run at the same time on a single computer. In that and many other respects a GEIRS session is not a server but a user program. Because

1. each session maintains the user’s shared memory contents in a socket (special) file,
2. each session’s command interpreter listens to a specific socket (port) fixed at the start of the session,
3. each session connects to one ROE represented by a network address and an OPTPCI board (unless in simulation)
4. each session grabs by default almost all of the available memory for its image storage (unless this is LUCI which requests only half of it)

there are some constraints to that guideline as follows

1. A Linux user can only run one GEIRS session at a time.

---

16 Each session is typically represented by five programs `geirs_shmmanager`, `geirs_cmdServer`, `geirs_control`, `geirs_disp` and `geirs_dataServer` on the computer; see the output of `ps -elf geirs` in the Linux shell.
2. Hardware is not shareable. Therefore the maximum number of sessions not run in simulation is limited to the number of independent pairs of ROE’s and OPTPCI boards. So each Linux user can only use a ROE and an OPTPCI board that is not already in use by another session.

3. Users starting sessions of the same instrument on the same computer need to change their command server port away from the default port (from the second user on).

Users ignoring these constraints will observe strange and undocumented cross-talks and interferences between commands and images as a result.

Note that each session’s command server listens to all commands that appear at its port. There is no protection by any type of firewall or password or user id, so every Linux user may send commands to any GEIRS session. This is for example needed because the prototypical observer never quits a session and every other user that needs to restart GEIRS for that instrument needs to send the quit to that abandoned session to shut it down properly before restarting it.
4 GRAPHICAL USER INTERFACE (GUI)

The software handles all infrared cameras at Calar Alto. Therefore the observer, once having used
one system, will easily feel at home with the other cameras. Changes are introduced only due to
different hardware. The aspects of GEIRS working as

1. a telescope control interface,
2. a motor control interface,
3. a temperature/pressure monitoring system

are partially disabled or virtualized in the Carmenes configuration.

4.1 Start-up (Standard)

It is useful to check with

```
ps -C geirs_shmmanager
ps -elf | fgrep geirs
```

whether someone else is already running GEIRS on the machine. Then the command

```
```

respectively

```
```

or for the most recent version of the software

```
```

starts GEIRS. If no command line option is used, four of them are implicitly activated. If the
-iwin option was present (explicitly or implicitly), it commences with the start-up screen of Figure
3. The controls and/or the image GUI will be opened depending on the presence of the options
-gui and/or -disp. The command server is started depending on the presence of the option -cmd.
The -gui option works only if the command server is either started here or already running. The
data server is started depending on the presence of the option -data. The real-time display requires
that the data server is run.

Error messages of the “Command not found” class indicate that the software may not have been
compiled, installed or simply not integrated into the PATH of the operating system.

The start commands refuse to start GEIRS if the associated TCP port is already in use.

The startup script may replace some files at common places (like in the scripts or INFO directories)
by versions that depend on the GEIRS version that just has been called. It generally does this by
managing symbolic links. The only reason for this breaking of the rules of versioning is that some
other softwares (drivers that access GEIRS from the outside) expect to find them at fixed locations
in the directories.
In the associated shell script, a set of configuration decisions have already been made. Most of the screen shots of this manuscript show the result of setting `CAMFONT` to `helvetica` in `scripts/Generic`, for example.

The startup script shows the remaining disk file capacity on the initial FITS file directory. The guideline is that readout electronics, detectors and fiber channels inbound via the OPTPCI boards are `not` shareable resources. The number of GEIRS instances running in simulation is not limited (apart from details mentioned elsewhere), but the number of GEIRS instances handling any real ROE or OPTPCI board at a time must never be larger than one. To that purpose, the startup script runs once `geirs_cleanup` with a test flag, which detects GEIRS processes already running by this or other users on this computer (see Section 5.5). On a system similar to LUCI or NTE with two GEIRS instances possibly running in parallel, don’t be alarmed if some GEIRS linux processes pop up here, because this may be the handler of the other arm of the telescope! In the standard case of running GEIRS for PANIC, CARMENES or LN with a telescope, GEIRS processes should not appear in the list—anything else means that either

1. local policies of properly shutting down GEIRS have not been communicated well between observers, or

2. observers erroneously believe that closing some of the main GUIs terminates GEIRS, or

3. the previous shutdown of GEIRS did not run smoothly. In that case running `geirs_cleanup`—without the `-t` option—may be useful to clean up these residuals, before trying again to start GEIRS.

Some parameters may be edited in Figure 3 at this time:

- **OBSERVER** Enter your name as observer. This will (i) appear in the FITS files and (ii) toggle allowances for some commands reserved for engineering purposes. (See Section 5).

- **OPTICS** This is fixed here, because the optical elements are not changing properties as far as GEIRS is concerned.

- **CAM_NDET** The number of detector infrared chips is fixed here.

- **DATAINPORT(s)** Defines through which bus of the operating system the software expects data. Operation through as many different PCIe boards as the computer hardware allows interfacing to a set of different ROE electronic boxes. Details depend on the slot assignment on the host computer. The names `/dev/plx-??` are used for historical reasons. They do not correspond to UNIX/Linux devices in the file system (which appear as `/dev/plx/Plx*` if installed as described above). The first placeholder in the name is 0 or larger if more than one OPTPCI board is installed. The second placeholder is 0, and may be also 1 if the ADC data from the ROE are also sent in parallel via the second data port.

- **CAMPORT** Selecting the empty string will start the software in a simulation mode for detector data. Otherwise it is the TCP socket and port for the internet communication with the ROE.

If the data generator of the OPTPCI board in the computer will be used for test purposes described in [8], but if no ROE rack is available or if this rack is switched off, some fake address of a non-responding computer should be inserted here. This allows to set up some
half-way simulation where the `rotype dgen` command followed by a `read` lets the OPTPCI feed data into GEIRS which are reduced and displayed as if they were streaming in through the fibers.

In simulation mode, GEIRS produces fake images and FITS files by placing spots at randomized positions across all detector chips in the field mimicking a seeing close to one arcsecond. It does not try to communicate with the ROE via the network or to receive image data through the fibers. The positions are randomly selected for each of the images; they are not drawn from any star catalog. The time stamps produced in the simulation mode are rough software simulations; they have much larger variances than the time stamps of modes that are fed with data via the OPTPCI boards.

- **MOTPORT** Absent, because GEIRS does not control CARMENES motors.
- **TEMPORT** Irrelevant, because temperatures are neither controlled nor monitored by GEIRS for this instrument.
- **TELESCOPE** This entry is absent.
- **Telescope Access** This is always false, because GEIRS does not communicate with the telescope controls. The virtual pointing and catalog operations described further down are nevertheless enabled.

The GUI in Figure 3 allows essentially to move subsystems into simulation mode. If you are not satisfied with some of the selectable parameters, you need to start from scratch, either with Figure 4 or by explicitly setting the shell variables before using the start-commands of Section 4.1.

The GUI in Figure 3 uses a countdown of 20 seconds, where the remaining time is indicated in the label of the `all` button. If no button is pressed to change the configuration within that time interval, it will continue to start GEIRS with the currently selected set of parameters.\footnote{That countdown was added in response to the fact that some people seem to start GEIRS without ever pressing one of the three buttons at the bottom, so we may end up with some of these GUI's hanging around for indefinite periods of time.}
Figure 3: Startup screen to start GEIRS. Which of these layouts appears depends on the instrument.
After you press all in Figure 3, the subsystems (most noticeably the ROE) are initialized and the GEIRS window of Figure 5 will appear. At that time all (recent) instrument patterns send commands to the ROE which switch most of the ROE’s LED’s off. The LED’s of the network card of the ROE cannot manipulated by these software means (and must be taped to shield their light).

The button OK compares the current parameters of the command server with the parameters proposed in the GUI and skips the initialization if the two sets are the same.

Actually both the “Controls” window (Figure 5) and the main display window (Figure 12) may be suppressed by removing the -gui and the -disp options, respectively, from the call of the shell in the $CAMHOME/scripts/GENERIC script. These changes in the configuration are available if the instrument is run in a stable production mode where the pipeline investigates the FITS files that are produced, such that the quick look at the frames is not needed or replaced by the more common ds9 viewer.

If some subsystems of GEIRS, like the ROE, the Motors or the Telescope are set to the simulation state in Figure 3, some parts of the GUIs described in this manual display yellow diagonal crosses or yellow backgrounds in menus to provide a visual warning that the corresponding section of the action or information is in some state of software emulation/simulation.

### 4.2 Start-up (Engineering)

Alternatively there is an engineering GUI called by

geirs_start

which pops up similar to Figure 4. This allows experienced users to edit many parameters on a finer level without editing the GENERIC script, but at a higher risk of starting GEIRS with modes that are not supported.

The entries with a white background can be fully edited (after left-mouse-click into the GUI or through selection of fixed entries by clicking on the down-triangle); the entries with a gray background can be changed to a limited degree by choosing from a pre-selected set with the down-triangle. Down-triangles turn gray if the selection is fixed (not editable).

The program scans (pings) a list of fixed ROE IP addresses and puts those that seem to be online into the selector for the CAMPORT. It puts subdirectories of CAMHOME that look like compiled GEIRS versions into the CAMBIN selector. If the Continue/Start button is pressed, the program sets some of the environment variables mentioned in Section 3; labels in the GUI and environment variables correspond to each other. Then it calls the shell script scripts/GENERIC with the options set in the third but last line of Figure 4. See Section 5.5 for the meaning of the GENERIC options. The principal rationale for having this GUI is that one can

1. mix hybrid instrument configurations as they frequently occur in the MPIA development process.

2. switch temporarily to a configuration without editing the GENERIC script, to narrow down connectivity problems (Section 9).

The major drawback of starting with this GUI is that none of the confirming messages do appear on standard output as they do with the start* scripts mentioned above.
4.3 The GUI’s windows

4.3.1 Camera control window

The control window of Figure 5 is the interactive interface to the camera.

In the top row three pull-down menus provide further options:

- **File Menu**

  - **Init/reboot ROE** reboots the read-out electronics, which means, sends a set of standard readout and idle patterns to the ROE. This will transmit roughly 2,000 “words” to the two FPGA chips on the ROE.\(^\text{18}\) Accounting for a few milliseconds per “word” that is transferred via the Ethernet to the ROE, this will need up to 10 or 20 seconds,\(^\text{18}\) The lowest level of these has a maximum of 1024 “words” and the second level a maximum of 512. Not all of them are used, depending on the complexity of the patterns.
Figure 5: The camera control window with its drop-down menus. The menus can be reached by clicking on the buttons or with \texttt{<Alt>F}, \texttt{<Alt>M} or \texttt{<Alt>O}. Most submenus can be called pressing \texttt{<Ctrl>} and a letter.

depending on Ethernet speed. (You may watch that progress with the \texttt{Modules\to ROE Log Monitor} menu.) It is futile to attempt a readout during that intermediate period.

- \textbf{Help} Opens a web browser which shows a HTML version of the command list, similar to Figure 6, equivalent to the contents of Section 5.3. This will fail if the environment variables \texttt{CAMWWW} and/or \texttt{CAMBROWSER} of Section 3.2 are not configured correctly.

- \textbf{Shutdown GEIRS} will close GUI’s related to the session and terminate the command server, shared memory manager and ICE server (if applicable). It is equivalent to the \texttt{quit} command (Section 5.3). This is a swift and recommended way of terminating GEIRS. \textit{Just closing the window does not shut down GEIRS!}

The background of the menu is yellow if the ROE is simulated, which means that all the images are faked in software and not actually generated by interaction with a ROE rack.

- Modules Menu The modules menu starts the different modules, each of which has its own description section.

  - \textbf{Display}: Toggles the status of the image display, Figure 12, i.e., starts it if not shown and closes it if shown.
  
  - \textbf{Telescope} Telescope control. \textit{Only available for PANIC}.
  
  - \textbf{TempControl} \textit{Only available for PANIC}. Displays a graph with the pressure and various temperatures inside the dewar This button is only present if the \texttt{CAMWOTPCTRL} is not set in the environment (that is, in the shell script to start the instrument). The display is passive in the sense that they show a scan of lines in a special format taken from a log file that is typically fed by a \texttt{cron(5)} job which reads the sensors. GEIRS does not need to be online to store these. The plot may even be displayed with

\begin{verbatim}
  cd GEIRS/INFO ; xterm -e gnuplot tmp_gp.panic
\end{verbatim}

if GEIRS is not started.

Irrelevant in the case of LBT instruments or CARMENES which have dedicated sub-systems to deal with these house keeping data.

- \textbf{New InstrShell} Opens a instrument shell window similar to Figure 9.

- \textbf{DebugLog-Mon}. Opens a debug log monitor
Figure 6: The web browser called by the Help button in Figure 5.

- **ErrorLog-Mon.** Opens an error log monitor
- **ROE-Log-Mon.** Opens a log monitor similar to Figure 10 showing a history of command exchange with the ROE.
- **Cmd-Log-Mon.** Opens a log monitor similar to Figure 11.

- **Options Menu**
  - **Sound** calls up a sound menu like in Figure 7, where a specific sound file can be associated with a variety of different events (such as telescope moves, completion of a read ...). To “activate” sounds played by GEIRS,
    1. the sound system must be configured as in Appendix A.3 such that it is forwarded over the network from the GEIRS workstation to the operator’s computer,
    2. the volume must be set to a value larger than zero,
    3. the sound flag for Sound On must be checked
    4. the volumes on the operator’s workstation must not be muted by the means of the operating system on that workstation.
  - **Savepath** and **Macropath** are directories that tell GEIRS where to save FITS data and where to look for macro files.
    - **Macropath**, the default search path for GEIRS macros, is usually set to the MACROS subdirectory in $CAMHOME.
    A default for the **CAMPATH** is proposed which is derived from the current value of the directory by replacing the lowest component with the instrument name and an ISO time...
Figure 7: Popup after Selecting Options→Sound in the Controls GUI of Figure 5. The events concerning telescope, wheels or temperatures are only triggered by the PANIC version of GEIRS, so selecting a sound file here for other instruments is futile.

stamp of the current date. Pressing cancel keeps the current value—which is shown in the title bar of the GUI. Editing the path name and pressing Save or carriage-return accepts the new directory (and creates it if needed).

At the time when GEIRS is shut down, the values are stored in the file geirs.xml in the $CAMTMP directory, and retrieved from there at the next startup.

– Logfile specifies where the log file is kept.

Below the drop-down menus various fields display the status of the camera and allow the setup to be changed:

• First row: Idle Loop setup

  – Idle This parameter defines whether the transition from the idle mode to the read mode is done
    * abruptly (break, with a sort of immediate termination or break of the idle cycle) or
    * whether the currently running idle cycle is completed before the read starts (wait,
reaching first a type of break point at the end of the idle cycle before switching to the read mode).

Using **break** has the advantage of starting the reading with the least possible overhead, but it usually leads to visible edge effects in the next frames because the clocking through the detector was interrupted at some position along the “slow” direction. For this reason this parameter defaults to **wait** for all instruments. There is an intermediate type called **auto** which is equivalent to **wait** for integration times shorter than some configurable threshold and to **break** for longer integration times. The associated command is **idlemode** in Section 5.3.

- **Idle Type** The idle mode is the (usually periodic) pattern of voltages applied to the detector lines (reading and resetting) while the ROE’s ADC’s are switched off such that no data are actually transferred via the fibers to the workstation. The resets avoid detector saturation. GEIRS supports four choices:

  1. **ReadWoConv** (Read with conversion) Reads and resets the same timing pattern as in the current read mode, including ADC conversion (although the workstation ignores this because it has not switched the data transfers on). The cycle time of these idle cycles is the same as the main mode, including the prolongations by any integration times; this aspect plays a major role if the **Idle** button has been switched to **wait**.

  2. **Lir** (Line interlaced read) A cyclic repetition of the read-reset-read pattern at the minimum integration time (which means, the integration time implied by clocking once through the detector at the current pixel time).

  3. **Rlr** (Reset level read) Resets then clocks through the detector line by line. There is a single read of each pixel in this idle pattern, so this is basically clocking once through the chips in half the time relative to the **Lir** idle mode.

  4. **Reset** (Reset only) Executes a series of resets.¹⁹ No reads are involved and therefore these idle mode cycles are the quickest available.

With the exception of PANIC the default is **Lir** for all instruments. The idle patterns are unaware of any of the three possible subwindow sets of the current read mode (Section 5.6.1), which means timing and resets in the idle cycles are equivalent to full frame handling of all chips. The associated command is **idlemode** in Section 5.3. Details of the idle patterns are discussed in [8].

- **Second and third row: Read mode/pattern setup**

  - **Read Mode** The different read modes available are described in detail elsewhere [9]. For standard broad band observing this should normally be left at the initial default of the instrument (which is **lir** for LN). The GUI sends a **ctype** command of Section 5 to the command/interpreter shell.

  - **n** is the number of reads and resets executed in the current read cycle. This is only editable for the multi-correlated modes.

  - **IT(s)** is the integration time in seconds. The detector is clocked with a rate of 100 kHz, resulting in a minimum integration time of

\[
\frac{2048 \times 2048 \text{ pixels}}{32 \text{ channels}} \times \frac{2 \text{ frames}}{100 \text{ kHz}} = 2.7 \text{ sec}
\]  

¹⁹full frame or line by line, I cannot tell... RJM 2015-08-03
for single or multiple Hawaii-2 and Hawaii-2RG detectors in full-frame mode that reads two frames, this accumulates 2.7 sec like in Figure 5. For Hawaii-4RG detectors read out by two MPIA ROE boards this is

\[
\frac{4096 \times 4096 \text{ pixels}}{64 \text{ channels}} \times \frac{2 \text{ frames}}{100 \text{ kHz}} = 5.2 \text{ sec.}
\]

The impact on LN detector saturation is discussed elsewhere [14].

- **prd** The pixel read time in nanoseconds. The standard is 10 µs equivalent to 100 kHz. See the `roe` command in Section 5.3 and also Section 7.8.

- **pskp** The pixel skip time in nanoseconds.

- **lskp** The line skip time in nanoseconds.

- **ems** The electronic multisampling factor.

- **Fourth row: Subwins** There is one button **On/Off** to switch between full-frame mode and subwindow mode. The button does not respond if no active subwindows exist.

The other button opens a GUI similar to Figure 8 with options to edit the index and the four parameters of the subwindows. Each row in the GUI represents one software window.

![Figure 8: Subwindow selections GUI opened with the Subwin-Selections window of Fig. 5.](image)

Click on a checkmark to remove a window from the set, and click on the empty square of a new line to start adding another window. The five integer numbers per line have the same meaning as the arguments of the `subwin` command (Section 5.3): (i) an index ≥ 1, distinct for each window, (ii) the \( x \) and \( y \) pixel coordinate of the lower left corner of the window in the range from 1 up to a multiple of 2048 depending on the number of chips in the detector, and (iii) the width and height (≥ 1) of the subwindow again in units of pixels. The two buttons at the bottom either activate the set of windows by using a chain of `subwin` commands, or leave the subwindow coordinates as they are; If the **Set** is pressed, the windows that are not check-marked in the GUI are forgotten by GEIRS—meaning to re-active them you will have to type them in with another round of editing. Editing entries in the GUI does not have any effect until the **Set** button is clicked.

**Set** and **Cancel** close the GUI. The **Subwin-Selections** and **OnOff** button indicate which configuration is left behind and effects subsequent read’s.
• Fifth row: Read

- **Read** The read button executes a read using the current exposure time and number of repeats. On completion of a read, the images are not saved unless autosave is selected under the save option. The button turns green while an exposure is executed; but it is yellow—as a warning—if the entire startup simulates the ROE in software.

- **-Q** If this flag is activated, the `scripts/QueueEFiles` script is executed before the exposure is started. For most instruments that script does nothing, so the status of this flag is irrelevant.

  For LN this script actually moves the derotation stage of the detector to the start position remembered from the previous exposure (in an attempt to keep the instruments infrared background constant on the detector surface).

- **Repeat** is the number of images $N$ with the specified exposure time $T$ which will be taken each time a read is executed (read-cycle). The total exposure time will then be $N \times T$ seconds. The maximum number of images depends on the computer shared memory set up in Section 2.4 and the setting of `CAMSHMSZ` in `scripts/Generic`.

- The current progress of the reads is displayed to the right of the Read button. The format shows two numbers separated by a colon, the current frame number and the current image number.

- **Endless** may be pressed to start an endless loop of reads. The images are read out with the current integration time and readout mode and dumped to the display. They are not saved unless the autosave option has been activated via the GUI or autosave command (Section 5.3). This is useful for positioning the telescope. Pressing the button again lets the button return to a gray background and back to the one-time action of the read and save buttons.

  The endless mode still includes the Repeat factor of the pattern blocks, which means for example that in a lir mode with Repeat set to 5, the natural $2\frac{1}{2}$ seconds gap after each 5 reads is observed.

- **Abort** Kills the read process—immediately, without regard of the current position of the address registers in the detector—and returns to the idle mode.

• Sixth and seventh row: Save

- **Save** The save button saves the most recent image(s) obtained using the currently defined save options. It turns green while files are saved to disk. At the end of a readout it turns blue to indicate that the current data have not yet been saved.

- **Save-Options** The check marks define the default way in which to save images. The file name to be created next is defaulted. The range of frames to be saved follows in the next line of options. The main choices are whether
  
  * to save individual exposures as separate disk files, equivalent not to activating any of the push buttons;
  * -i/integrated to integrate them (add them up arithmetically) and save only a single image;
  * -l/FITS-cube to store the individual frames as layers following the 3-dimensional FITS cube standard;
* -M/MEF to add the -M option to the `save` command and end up with the multi-extension FITS format, were images and subwindows are stored as FITS extensions, one extension per window

* -z/FITS compr. to use the “internal” tile compression registered as a convention of the FITS standard [15, 16]. The current implementation allows this only if also the MEF is activated.

* -S/single frms to add the -S option to the save command, which puts the individual frames into the FITS files, not the pre-correlated/preprocessed images.

* -a/auto-save to save the data automatically (without waiting for a request through a `save` via command shell or GUI)

* -s/immed.-save to save the data as soon as reading a frame is completed. (The difference to the auto-save is not waiting for macro termination and even starting the disk transfer before saving the previous frame has finished—used for the `diffint`.)

Note that the save options are overridden by any options specified in observing macros. For example `save -f 2 -i` in a macro will integrate from image 2 to the end of the series, and save only a single file, even if the save options specify saving images separately. Turning on auto-save will execute a save after every read, without clicking on the save button.

Filename The name of the next file to be saved by pressing the `Save` button at the beginning of this line or by issuing a `save` command from a script. One can either specify a name or a root. In the latter case the filename is the root plus a four digit integer, which will be automatically incremented by one each time a save is executed. By specifying the root, the system looks for the highest free filename. If a filename ends with a number this number will be increased.

Clicking on the name with the current FITS file allows to change the name for the next `save` command.

The two fields to the right of the FITS file name define the range of the first and last frame or image to be included in the output. Whether the count means frames or images depends on the readout mode and whether the -S option was selected further above. The two indices are generally $\geq 1$, but values of $-1$ are supported to indicate that the smallest respectively largest range of the images in the buffer should be saved to disk.

• Last row: Macros

Macro Specifies a macro (file with a list of GEIRS shell commands) to be executed by the macro parser. If the filename has the (recommended) suffix `.mac`, the filename may be specified without the `.mac` extension. The macro file must be in the MACROS directory specified under the macro path in the options menu (see above) or otherwise be specified by the full path name. Please refer to Section 5 for the macro syntax and commands. Specification of the macro just provides the file name; the macro is not started yet but with the button right to the entry field.

Start, Pause, and Quit Macro control the execution of observing macros, reads and running programs. Note, that if a pause or abort is issued, the macro will continue executing until the current command is completed! Check in the command window to be sure that the pause is in effect. Clicking again on Pause will continue executing the macro after the pause.
While the macro runs, the Start button turns green and the field right from it indicates which line in the macro file is currently executed.

If the GUI of Figure 5 disappeared, it can be reconstructed with the control command to the GEIRS shell (Section 5) or using the equivalent forwarding with cmd* or snd* (Section 3.1) from the Linux shell.

### 4.3.2 Command Shell and Log Monitors

The Modules → New InstrShell menu starts the interactive command shell interpreter of Figure 9.

The appearance of the Command Shell and logging windows (sizes, colors,...) is defaulted as for X-terminals as set at the standard places in the file system, $HOME.Xdefaults, $HOME/app-defaults etc.

After the prompt, the GEIRS command shell expects commands from the list reproduced in Section 5, and the terminal echoes the responses. The commands send from this window and the commands created by pushing buttons in Figure 5 are received by the same command manager and effect only one single set of state variables. Both channels may be used at the same time.

Two additional log monitors may be opened with the Modules menu, illustrated in Figures 10–11. These are passive displays: they filter lines from the $CAMHOME/log/*.log files; the logging parameters and amount of information that is stored in these files does not depend on whether the associated GUI is open or not. (The logging information does depend on the LOG_LEVEL definition in the GNUmakefile while compiling and further on the adjustments by any log commands send to the GEIRS shell.)

To retrieve the debugging logs use journalctl(1) with GEIRS as the identifier, for example

```
uid=$(id -u)
journalctl _UID=$uid SYSLOG_IDENTIFIER=GEIRS -p crit..warning
```

to obtain the logs from the critical up to the warning level.

The monitor of the ROE logs, Figure 10, tracks log/roe*.log, and shows a time stamp, the user name on the host machine, the camera name, and two kinds of lines:

1. Entry and exit from one of the functions that accumulate (compute) the duration of patterns and loops over patterns,

2. Patterns submitted to the ROE. The tout shows the timeout (in seconds) for waiting for an answer.

The monitor of the command logs, Figure 11, tracks log/cmd*.log. The inter flags that the line was generated by a shell script assembled by the command shell with sh -c, and the following i, c or s means the caller was the interactive gui, a command, or the shell, respectively.
4.3.3 Real-time Display

4.3.3.1 Introduction The display tool, Figure 12 works similar to ds9 or fv tools with some display options and similar statistics. The GEIRS display, however, is completely unaware of world coordinate systems standards. Some online data processing techniques are available. These interactive operations (magnifying, scrolling forward and backward through the frames, setting ADU cut levels, . . .) affect only the displayed data but do not manipulate the raw data that have been or will be saved to disk.

The command interpreter started with the Module $\rightarrow$ New Instrument Shell menu of Figure 5.

```
Figure 9: The command interpreter started with the Module $\rightarrow$ New Instrument Shell menu of Figure 5.

4.3.3 Real-time Display

4.3.3.1 Introduction The display tool, Figure 12 works similar to ds9 or fv tools with some display options and similar statistics. The GEIRS display, however, is completely unaware of world coordinate systems standards. Some online data processing techniques are available. These interactive operations (magnifying, scrolling forward and backward through the frames, setting ADU cut levels, . . .) affect only the displayed data but do not manipulate the raw data that have been or will be saved to disk.

xpa is compiled for example by installing the heatools (Section A.6.4). If xpa available, users can send a duplicate of each new FITS file that is generated by the save command to an online ds9 application by adding two lines like

```
Xpaset='type xpaset | awk '{print $3}''
cat $1 | $Xpaset ds9 fits
```
Figure 10: The monitor opened with the Module→RoeLog-Mon menu of Figure 5.

Figure 11: The monitor opened with the Module→CmdLog-Mon menu of Figure 5.

to the QueueFiles shell script (Sec. 3.3). As an alternative to using the type command one may use the full path of xpsaset or make sure by symbolic links that the path contains the executable. Note that ds9 sometimes needs to read ds9-64 depending on how this was compiled. With that setup, opening the GUI in Figure 12 may be superfluous.

The main difference against saving the image as a FITS file and then calling these standard displays is that one can address any picture in the current memory buffer rather quickly by its index. It is also easier to navigate through pictures if windowing was used, because GEIRS does not glue a set of subwindows while composing FITS files. (geirs2Panic has been written to merge these frames after they have been stored as FITS files.)

The real-time display polls the shared-memory database (with a combination of get INT_STOP_SEC, get READBUF and get NIMAGE) each $1\frac{1}{2}$ seconds to check whether a new image is available.\footnote{That interval can be changed by modifying the 1500 milliseconds of the pollTmr in de/m*/m*/g*/DisplayGUI.java.} It then requests the recent image from the data server and adds it to its circular buffer.
Figure 12: Current Exposure Display. Right after GEIRS startup this shows the white-on-blue logo of the MPIA.
The pixel display that covers most of the area is the most recent detector image. One or two scroll bars appear if the pixels of the detector(s) don’t fit into the operators window depending on the zoom factor.

The menu on the right hand side of Figure 12 has a number of fields, which are described from top to bottom in the subsequent paragraphs.

Hoovering with the mouse for approximately 2 seconds shows a short description of what these fields mean. To keep the space consumption of the fields low, labels have been abandoned because these “tool tips” provide the same functionality.

There are some standard types of fields:

- Down-triangles decorate scroll-down menus. They either offer a fixed set of options (with a scroll bar if the list of options is long), or offer selections that can in addition be edited. The latter is a frequent standard for numerical fields.
- Fields with gray background are outputs for information only.
- Fields with variable background are either buttons that toggle a state or open another menu.

4.3.3.2 Thumb Nail Pixel Image. This is the detector image downscaled by a fixed factor 9 (if the detector area is 2048 × 2048) or 19 (if the detector area is 4096 × 4096), independent of the scale factor of the main display.

The rectangular frame indicates which portion of the detector(s) is currently visible in the main display. Clicking with the left mouse in this small image is equivalent to centering the main (big) image around that point, i.e., could as well be achieved by moving the scroll bars in the main image.

4.3.3.3 Zoom factor. There is a selector with fractions ranging from approximately 1/32 to 4/1 that specify a zoom factor. Each detector pixel is replaced by that many pixels in the operator’s display. One may either click on the down-triangle to modify the factor or press − or + while the focus is in the pixel image to decrease or increase the factor.

The fractions from 1/32 to 1/2 are not binning groups of pixels but simply skip a fraction of rows and columns (sub-sampling) for the sake of speed. So note that for small fractions (large denominators) some of the stars—of any magnitude—may virtually disappear in the viewer if the detector pixel scale does not well resolve the Strehl width.

However, there is a flag to activate a software binning of pixel tiles to the right of the scale factor, which will replace each pixel in the GUI by the arithmetic average of a square tile in the neighbourhood if the zoom factor is < 1.21 This is computationally more expensive and may lead to false impressions of background noise, but avoids that stars just disappear because they are not hit by a finitely subsampled grid of rows and columns.

If the zoom factor is > 1, each detector pixel value is expanded in a small square tile by copying its value to the display pixels. There is no interpolation then akin to some smoothing image processing known to other software packages.

21It does not just add the pixel values in the tiles because that would require adjustment of the cut levels each time the zoom factor is changed.
4.3.3.4 Color. There is a selector for the color lookup table. Some of the color tables are from the Part 6 of the DICOM book, and the others from ImageJ. All color tables—with the exception of the Grayscale—quantize the pixel values with 256 different RGB colors.

The display uses a linear map for the translation of ADU’s to brightness by default (i.e., after GEIRS has been started). A \( \gamma \)-correction with a power law scaling is available by setting the DISP_GAMMA value of the shared memory database to some different value in the range \( 0 \leq \gamma \leq 100 \) with the put command (Section 5.3), for example

\[
\text{put DISP\_GAMMA 1.4}
\]

The default after GEIRS startup is \( \gamma = 1 \).

4.3.3.5 Image Stack. The image display maintains a finite circular buffer of images and frames, so one can step backwards through a limited number of previous images or frames. The choice current image lets the viewer collect new images arriving on the workstation, and current frame lets the viewer collect new frames on the workstation. This means

- The image display does not have arithmetic functions to split images into frames or to combine frames to images. It adds either new images or new frames to its stack, but not both at the same time. It is just a viewer for pixel arrays, not a calculator nor a data pipeline.

- The image display does not change if one toggles from current image to current frame or vice versa. One needs at least one further read to let the image display become aware of new data.

One of the small negative indices selects an earlier one in that circular buffer, so \(-1\) means the penultimate image and so forth. The negative labels are just indicating that one steps backwards—similar to the semantics of negative indices in Python or Maple lists.

If one tries to select an index which does not yet exist, the GUI corrects that index to an existing one. If you have clicked for example three times on the Read with a default repetition factor of 1 after starting the GUI, and select \(-8\) here, the GUI knows only 3 images whereas \(-8\) requires at least 9, so the GUI will change that selection to \(-2\), the oldest image it knows about.

The circular buffer of image is private to the GUI. It does not depend on how many images fit into the main GEIRS buffer in shared memory or on how many images are created in a single read cycle.

Selecting anything else but current means that the main display freezes an earlier image, but still active read-outs will enter the GUI’s buffer of images and old images are discarded. If the index in the stack is re-selected afterwards without halting the readout, the negative lables will not refer to the same images.

As a warning to the user, selection of anything but current paints the button in blue to indicate that one must select current to return to the life display.

4.3.3.6 Sky Background. There is a file name chooser for a FITS file that must contain a sky (background) image with the same full frame resolution as the current instrument. Clicking
on the button allows to select a file that exists on the local file system. To disable sky subtraction, click on the **Cancel** option in the file chooser, not on the **Save** button. The button shows the filename without the `.fits` suffix. If the name is empty or is not a compliant FITS file, no sky image subtraction occurs. **Compliant** means:

- The FITS file must contain a `EXPTIME` keyword in the primary header for an exposure time (in seconds) such its image can be linearly rescaled to the exposure time of the current display.
- The FITS file must contain an image or a cube in the primary or secondary header data unit with the same horizontal and vertical `NAXIS` dimensions as in the current full frame scenario. If the FITS file contains an at least 2-dimensional image in the primary HDU, it reads this, otherwise it tries to look for an at least 2-dimensional image in the next HDU, but it does not look into latter HDU. Even if the current exposures use subwindows, that file with the sky/background image must have the full frame dimension, including all detector chips.

If the sky image is in a cube, all slices in the cube are added up, and its effective exposure time is set to the `EXPTIME` of the primary header multiplied by the number of slices, in compliance with Section 6.4.

- The image in the FITS file must be of `BITPIX` type 16, 32, -32 or -64, the usual short, integer and floating point types.

If the selected file is not compliant, the software will clear the file name in the GUI and resume the mode without sky subtraction. The simplest way of creating a compliant file is to save a sky exposure in Figure 5 with `-i` checked and all other options unchecked.

If that sky subtraction is activated, the derived data like cut levels, FWHM estimators, and horizontal and vertical cuts through the images are all derived from the differential/subtracted total pixel brightnesses.

The task of subtracting two FITS images is usually left to more advanced programs. If the **FTOOLS** are installed (see Sections A.6.1A.6.4), save the first image to disk, for example the file `tst1.fits`, save the second image to disk, for example the file `tst2.fits`. Let the images be in the extension named WIN1, for example, then the difference is created with the **SUB** operator of **farith**

```
farith tst1.fits[WIN1] tst2.fits[WIN1] tst3.fits SUB
```

on the Linux shell.

### 4.3.3.7 Brightness Cut Levels (Parameters).

There is a cut levels selector which offers either **fixed** or a range of numbers are not adapted to the current image but stay fixed as new images arrive, and the cut levels may be edited by changing the lower and upper limit in the two fields underneath. If **fixed**, one may enter a pair of numbers where the left number (lower limit) is larger than the right number (upper limit) to revert the levels; then stars appear dark on a brighter (sky) background.

### 4.3.3.8 Brightness Cut Levels (Values).

There are two numbers that are computed pixel cut levels based on the methodology selected higher up. These cannot be edited unless **fixed** is selected. If editable, these entries also accept floating point notation like 2.5e3 (for 2500). The internal handling of these cut levels is quantized in integers: using accuracies better than 1 is futile.
4.3.3.9 **Hot Pixel Coordinate.** There are two integer numbers which represent the x and y coordinate of a “hot pixel” in the detector frame in FITS coordinates, so (1, 1) is the lower left pixel. The numbers can be changed by typing in other coordinates, or by clicking with the mouse at a place in the main image (not in the thumbnail image) or by using the 4 cursor keys. To disable their definition, insert a negative number. If enabled, that point is marked with a cross in the display.

4.3.3.10 **Horizontal/vertical Slice.** In the next row the horiz. and vert. buttons open plots that show the pixel values along two straight lines that cut horizontally and/or vertically through the image at the “hot pixel” marked by the center of the cross (Figure 13). The plots are updated if new images arrive or if the hot pixel is moved. The titles of the plots indicate the FITS x and y coordinates of the common crossing of the two cuts. The buttons are green while the plots are active. The two graphs should be closed by clicking again on the buttons; their color will switch back to gray if de-activated.

The width of the horizontal or vertical pixel interval that is shown in the slices becomes smaller or wider as a function of the zoom factor of the main display. At zoom factors < 1, the entire detector is scanned, at zoom factors > 1, the scans show smaller sections, which means higher resolutions.

![Figure 13: Examples of cut plots activated with the horiz. or vert. buttons of Fig. 12.](image)

---

22They are *not* cutting through the location where the additional FWHM centroid may appear.
4.3.3.11 **Radius of Interest.** The next field defines a radius in units of detector pixels. If chosen positive, it is also inradius of the square box painted around the “hot pixel” in the pixel image; if negative, the radius is undefined, no square box is drawn, and the FWHM computation is de-activated. In the example shown the value is 146, which means the box covers $293 \times 293$ detector pixels. (Detector pixels means that the apparent size of the box changes if the zoom factor is changed, whereas the size on the detector does not change.) The initial value is computed from a FWHM of 0.9 arcsec, the imager’s pixel scale and setting the boxes half edge to $2\sigma$.

4.3.3.12 **Hot Pixel Value.** The next single integer number is the pixel value at the place of the “hot pixel.” It cannot be edited.

4.3.3.13 **FWHM Estimate.** The next row contains two buttons and a number related to FWHM guesses. The row does not exist for CARMENES and the spectroscopic camera of NTE because the width is computed with a 2-dimensional isotropic fit to the pixel values which does not make sense if one of the axes in the image is not a direction on the sky.

The FWHM button can be pushed to activate FWHM monitoring in new images that arrive. FWHM is green while active and gray while not. Note that

1. the FWHM fit does not make any sense for the LUCI spectroscopy. But GEIRS does not have any clue whether it is currently used for imaging or spectroscopy in that case.

2. values are wrong if GEIRS does not know the correct pixel scale, in particular for instruments with variable pixel scales if started with the wrong resolution in Figure 3 and/or not getting updates. So in particular for LUCI it is recommended to check that the pixel scale reprinted in the title of the GUI is correct.

The history of these FWHM estimates contains up to 128 measurements; older values are forgotten if that stack size limit is reached. This ensures that the graph with the plotted values does not get too crowded.

The algorithm searches for a bright center in the square frame defined by the region of interest further up, and fits a Gaussian (background plus amplitude with variable width) there. The computation is disabled if the radius is negative. The fitted full-width-at-half-maximum printed to the right of the button in units of arcseconds.

A cursor with three green rotor blade lines is inserted into the pixel image where the algorithm located the center of the Gaussian. This gives a visual feedback to check that the centroid search ran wild. The algorithm considers an area equivalent to the black box around the cross for fitting; for optimum performance and quality, that box should roughly cover the center of the star and $2\sigma$ of the expected Gaussian around it.

The computational load may be massive; do not activate the button unless needed. The algorithm starts by computing at 5 points (center, East, North, West, South) around the center median values of pixels, selecting the largest of these 5 points, and continuing this recursively on a shrunk subarea taking the largest of the 5 points as the new center. The points in a virtual square box around that centroid are then binned by distance from the center, and a low-resolution fit to these binned data is generated using the first few components of a 1-dimensional Fourier transform to reduce noise. In the smooth approximation by that Fourier transform the zero of the second derivative
(equivalent to the 1σ distance from the center) is searched, and multiplied by $2\sqrt{2\ln 2}$ and the pixel scale to create the FWHM.

The algorithm does not fit a Gaussian if the amplitude appears to be less than $10^{-3}$ of the background. In these cases it sets the standard deviation of the Gaussian to the square box edge length as some sort of lower estimate, and the green cursor in the pixel display disappears.

The `plot` button pops up an auxiliary window similar to Figure 14, which shows a horizontal time axis and FWHM values on the vertical axis. FWHM values above 2 arcsec/px are not included (for the benefit of a supporting automated scaling of the vertical axis). The time axis is the time when the snapshot of the image was drawn from the GEIRS data server. Note that this time may be off by a full integration time in comparison to the end of the exposure that actually contributed to the image! If one skips backwards through the image stack as mentioned above, and if the FWHM button is active, this will add points to the FWHM stack at these times of the past.

The button is green while that plot is displayed. Pressing the `plot` button again removes the plot. The two buttons are slightly correlated: One can push FWHM to collect values without plotting them; but plot shows only those measurements that have been collected.

### 4.3.3.14 Main Display

The main display (with up to two scroll bars) shows a (optionally zoomed) version of the pixels.

A mouse click in this display moves the hot pixel location to that pixel.

A drag with the left mouse (press-hold-release) selects a rectangular subsection of the pixels, copies these values into a temporary FITS file, and opens a gnuplot display with a histogram of the pixel values rendered with `fitsImg2Asc(1)`, see Section 5.5.

### 4.3.4 Telescope control window

*Section 4.3.4 has no relevance to instruments besides PANIC.*

Virtual control of the telescope, such as moving to an absolute position or offsetting from the current position, is done on the telescope control panel. The basic information from the telescope,
such as airmass, UT, and current telescope position is also displayed here. This GUI panel should start automatically when the GUI is first initialized. If not, you can call it up from the camera control window (Fig. 5) in the menu Modules → Telescope.

GEIRS keeps some basic set of telescope parameters for the displays and for inclusion in FITS header keywords. This set of values is not necessarily up-to-date, because GEIRS reads the parameters from the EPICS interface only if it itself has forwarded one of the telescope commands and if that action terminated successfully. GEIRS does not poll telescope parameters, which means any change of pointing coordinates or focus offset and so on by any commands that bypass GEIRS will not be reflected correctly in GEIRS GUI’s or FITS files up to the next telescope command.

The three buttons in the display implement the three subcommands of the telescope command, see Section 5.3.

The air mass as a function of time for the current sky coordinate is plotted by a call to a web page on swarthmore.edu with a submenu of the File menu. This requires a reasonable setting of the CAMBROWSER environment variable of Section 3.2.23.

4.3.4.1 Moving to an absolute position An absolute position can be entered directly in the RA and Dec windows. The position can be sent to the telescope by clicking on the preset tel. button. The RA and Dec windows also display the current telescope position after each offset.

4.3.4.2 Relative offsets An offset in arcseconds and a direction on the sky in degrees can be entered and send to the telescope with the move tel. button. The angle is measured in sky coordinates relative to the $\delta$-coordinate such that 0 refers to a change along $+\delta$ and 90 to a change along $+\alpha$. Both parameters, the throw and the angle, are signed values. Flipping the sign of the distance is equivalent to adding or subtracting 180 to/from the direction.

GEIRS keeps track of the total vectorial sum of these offsets. A throw of 0 arcseconds (that otherwise would not make sense) can be used to reset both components of that cumulative sum to zero.

---

23 which is for example displayed when GEIRS is started…
4.3.4.3 **Focus**  
A request to change the focus position by moving M2 is triggered by the `move tel. focus` button. The value in the input field is a relative amount in units of $\mu$m. The total value in units of mm is shown two lines higher up.

4.4 **Taking data**

The windows introduced thus far are the environment in which one takes data manually (including the use of GEIRS macros, see Section 5). This is useful for tests or special calibrations.

4.4.1 **Setting up the camera for an exposure**

Before you start, make sure you have selected the proper paths for your data etc., see Figure 5 at upper right. You should also set the root name of the files to be stored on disk, which is also done in the camera control window. The instrument is completely setup in the camera control window. Here you select the read-out mode and the exposure times, to name the most important.

4.4.2 **Taking exposures**

An exposure is taken by pressing the `Read` button (below centre in the camera control window). Although this exposes the image, it is only read into the memory of the instrument computer. There you can use it to take a look at it on the real-time display, measure background level, seeing etc. there. If you decide to keep the image, you also may modify the format of the data (e.g. as a FITS cube, individual images, stacked images). Once set you save the data by pressing the `Save` button. Due to the double buffering, an image may be saved while the next one is already being taken. (For CARMENES, that alternating buffering scheme has been disabled to halve the RAM requirements. This works because CARMENES is only used in a batch type of environment where `read` and `save` are used only in sequential order without temporal overlap.)

4.5 **Saving data**

The data are stored on one of the disks of the instrument computer under the path you have specified under `SavePath` in the Options Menu of the camera control window, Fig. 5. The initial default is `$HOME/DATA` set at start-up time in Section 4.1. The files are stored as FITS files and are not write protected in the standard sense of the file system (!). Each `save`—either explicit or implicit with the `autosave` or `sfdump` mechanisms—creates FITS files which are cached by Linuces. This incrementally reduces the amount of free RAM displayed by commands like `top(1)` or `free(1)`. (People may erroneously interpret this as memory leaks or some sort of defect within GEIRS; there is even one instrument where the local system administrator restarts GEIRS periodically because he believes that memory is “lost.”) This effect is obviously very pronounced on computers that produce astronomical images, and in particular where GEIRS in its standard configuration can generate FITS files half as large as the total computer’s RAM in a single exposure. The caching mechanism basically does not harm but is often useless, because data reduction is rarely done on the GEIRS computer so the speed-up of reading the cached FITS files instead of their disk copies is never felt. An obvious exception here is the CARMENES first stage pipeline which uses these FITS files right at the end of each exposure.
5 COMMAND INTERFACE

5.1 Double buffering

It takes a some amount of time to transfer the data from the camera and save it to the hard-drive on the workstation. To reclaim some of this otherwise lost time, GEIRS has been configured with two image buffers. Thus, a new image can be read out while the previous image is being written to disk. To implement this feature, the commands should be written as in the examples (Section 5.7), with a sync tele after the telescope offset and save commands. The GUI will then only wait until the telescope move is completed before starting the next read (the save command may still be in progress).

5.2 Parser

5.2.1 Syntax

Commands and their arguments are usually submitted one per line, separated by line feeds. If two or more commands are to be send at once, they need to be separated by a semicolon. This makes for example sense for the commands that are almost always followed by the sync, for example:

```
save -M ; sync
```

Note that this format generates only a single answer from the interface, not separate answers from the individual commands in the list.

There is one command, save, which uses commas to bundle groups of options.

Note that command options cannot be squeezed into short forms and cannot be swapped with non-optional arguments nor be clumped without spaces, as some Unices allow in their shells or some higher programming languages support with some getopt(3) libraries. Example:

```
save -zC # wrong syntax !
save -z -C # valid syntax
```

As a guideline, trailing arguments or options in commands are silently ignored.

5.2.2 Timing

The GEIRS command interpreter does not have a command stack; so one cannot type ahead an arbitrary number of commands assuming that they will be executed in order. Proper timing is achieved if and only if each command waits for the reply from GEIRS before the next command is submitted. There is no reason to implement convoluted timeout data bases on the client side: GEIRS has its own internal timeout values for the various tasks. The reply will carry an error message if GEIRS has run into one of these timeouts. It would be an even worse design of the client to set some arbitrary constant timeout on the client side.

GEIRS maintains a busy state after it received many of the commands. To relax these requirements, GEIRS actually puts a single (!) command on hold if GEIRS currently is in its busy state, and
waits for up to 5 seconds for the removal of the busy flag, i.e., for the termination of the previous command. This means in practice that the client side can type ahead one single command if the previous command is handled within 5 seconds.

5.3 Command List

In this section a complete list of commands is given. The order is lexicographically, not by functionality. These commands and syntax can be used in macros or typed directly into the command window or submitted with the interfaces of Section 3.1. Use with caution some commands are better left out of macros! For example, `quit` will exit a macro at the point it occurs, no further instructions in the macro will be executed. Also, if interactive is on, and `ls`, `dir`, or `history` are used in a macro, the macro could stop executing and wait for a carriage return.

The subsequent pages are a PDF reproduction of the “help” page generated by texinfo in various formats. The intend is to demonstrate to reviewers that this information is indeed available, not to provide a reference that is anyway accessible with the online software. [For this reason, four pages of the PDF document have been packed on a single page of the manual; this also helps to realize that they carry their own internal pagination.]

The options to read this informations are:

1. the File→Help button in the controls menu, Figure 5, if the full path name of a browser has been set in environment variable `CAMBROWSER` in the startup file `scripts/Generic`. This is the same as calling

   ```
   setenv CAMBIN=${CAMHOME}/<branch>/share
   firefox ${CAMBIN}/camera.html
   ```

   offline.

2. the info(1) command

   ```
   info -f $CAMHOME/<branch>/share/info/camera.info
   ```

   opening the screen similar to Figure 16. This may be simplified as described in Section 2.5.7.

3. as a PDF document

   ```
   cd $CAMHOME/<branch>
   texi2dvi --clean --pdf --expand camera.texi
   evince camera.pdf
   ```

4. the help command entered in the command shell.

This is a generic account of the command interface, and again many of these do not apply to CARMENES, in particular the commands that interface with the telescope or motor and other controllers. The commands are either in the category `type:USER` or `type:ENG` or `type:SUPER`; the commands in the latter two categories are rejected unless one is using the instrument under one of the engineering observer ID’s or the observer ID `master`. (The observer ID is configured in the top field in the GUI of Figure 3.)
Chapter 72 [test], page 43

Causes sends a command to the

Chapter 51 [read], page 26

Aborts the execution of

Chapter 39 [macro], page 20

In consequence the

Overview

Interface to the command server of GEIRS, the Generic Infrared Detector Software of MPIA

1 abort

type: USER

syntax: abort [-r [-k [#]]] [-m] [-s] [-t] [-a] [-b]

Aborts the execution of read and/or macros.

• -r: abort read only. See Chapter 51 [read], page 26. Causes a send to the command server of GEIRS, the Generic Infrared Detector Software of MPIA.

• -w: shorten the initial wait time of the sync command and abort the read command. Note that the meaning of W is not the same as W of the sync command. It only means that the optional additional time delay that is an argument of the sync command is cut to zero and that the sync starts to wait on the termination of its processes without any artificial further delays. In particular this means that you cannot prematurely abort a read by an abort if a the sync has already been sent to the interpreter. See Chapter 66 [sync], page 38.

• -t: abort test only. See Chapter 72 [test], page 43.

• -k [#]: kill the read after waiting for # seconds (default is 2s). This tries first a “smooth” kill via a catchable signal, then enforces the kill.

The default, if no options are used, is to abort everything except save.

If the abort command has been finished, the number of frames that have been received on the workstation may be too small to create images based on that number of frames, depending on read mode (correlation type, sub-sampling frequency, ...) and time between starting the read and the abort.

In consequence the save may in general fail after abort. A typical example of this situation is a “telescope” mode with only two reads, where any abort results in only a single remaining frame (the rest frame), from which no useful information (i.e., image) can be extracted.

Special note to CARMEDES programmers: Note that a command sequence like read sync cannot be shortened by sending abort, because sync includes a sync read which enters a wait state that waits until all the images of the regular integration time have arrived. This means that effectively the abort would be recognized after the sync returns, and at that time there is nothing left to abort. This means if you are not sure at some time whether you would like to wait for the regular pairing of the full integration time or perhaps use a abort later, do not send a sync until you really mean to.

Note that a PLX Appliance 5xK will generally be emitted by the PLX driver because the abort cancels a wait status for further frames (if the abort occurs before the readout.
bias

maxvolt() function in the source code. If terminated regularly when the integration time is exhausted. This is a fairly common scenario for CARMENES, where the first one must be one of yes, on/no, off.

If called without parameters, the current status of autosave is returned.

Note: On problems with the audio driver the sound may be switched off in the Options menu of the controls.

4 autosave

type: USER

type: USER

syntax: autosave [yes/on/no/off] [sr|dr|ar|sr2|dr2|ar2|sr3|dr3|ar3]| [p|f] [filename|devname] . . .

Warning: autosave launches a background process and should be followed by a sync when used in a macro or when called externally.

6 cassoff

type: USER

type: USER

Sets the zero-point 'angle' as the cassegrain angle for the NSEW (North-South, East-West) orientation.

The command is only relevant for PANIC. The arithmetic difference of the cassegrain angle that is provided by the telescope (TECS) and that offset angle is stored as a FITS header keyword. So the sign convention and units must be compatible to those of the casap command, whatever that actually might be.

Note that this value is not used to modify the WCS values of the FITS header: to rotate the WCS matrix set the CAM\_NORTH environment variable, measured in degrees, to an angle, which differs from the default, before starting GEIRS. (The implicit default of CAM\_NORTH = 90; measured counter-clock-wise from the horizontal x* coordinate in the FITS images. This means to instruct GEIRS that north of PANIC images is not up but slightly more to the left, use a value of CAM\_NORTH that is slightly larger than 90.)

7 caspos

type: USER

Sets 'angle' to be the actual cassegrain angle. The default value of the angle when GEIRS starts is the one obtained from the EPICS interface of the telescope. The Cassegrain angle of the telescope is apparently measured in degrees with some undocumented offset and sign conventions.

This CASPOS FITS header line reports the difference of this cassegrain angle minus cassoff. If called without parameter, the actual cassegrain-angle relative to NSEW will be returned. The command is only relevant for PANIC.

8 cd

type: USER

Changes the directory for save operations, reminiscent of the UNIX cd(1).

The command checks the capacity of the filesystem of the new directory. If the capacity is below some value, the command issues a warning.

If the current default filename for the save operations was given as basename ending not with a digit [see Chapter 41 (next), page 21], and that directory already contains files with that basename, the number part in the default filename will be increased if this is necessary to avoid name conflicts. If there is no such file in the new directory, the default filename stays as it is.

Warning: If used without an argument, the new directory is set to the home directory of the user. The directory of the 'save-path' and the free disk space in that directory are returned.
mode, the integration time may be increased by GEIRS if the number of reads does not fit into the integration time ... merely a means of type saving because it allows to ignore all the trailing lines in a macro script from some point on.

17 delay
type: ENG
syntax: delay [crep #]
Set 'delay crep x.x' before crep read. The final argument is a floating point number in milliseconds (exact to three fractional digits, i.e., microseconds).

18 dir
type: USER
syntax: dir [filenames]
Executes 'ls' in the current directory. The output stops after each page; to proceed with the next page, enter: <Enter>; to abort the output, enter: <q>.

19 display
type: USER
syntax: display
Opens a GUI with the display of the detector readouts, some detector engineering capabilities (data visualization) and some kind of statistics of ADC variances.

20 engstatus
type: ENG
syntax: engstatus
Requests and returns the engineering status from the camera.

21 exit
type: ENG
syntax: exit [macro]
Syntomatic to the quit command. See Chapter 50 [quit], page 26. Shuts down GEIRS, its GUI's and servers of the camera software.

If the command is used in a macro and the argument macro is added, the effect is just to exit (leave) the macro at that point, but without shutting down the other parts of GEIRS. This is merely a means of type saving because it allows to ignore all the trailing lines in a macro script from some point on.
Chapter 11 [control], page 5

22 filter
cARMENES-AIV04B-NIR-DCS-MAN01

type: USER
syntax: filter [position]

Where position is one of the filter macro names defined in $CMANDIR/macros.instr and the suffix instr is actually panic because this is the only instrument where (this version of) GEIRS stores wheels.

The macro in this file define the position of all wheels following:

- `*` means leave this wheel wherever it is.

Syntax: In a macros-file, comments are started with either the semicolon (:) or with the sharp (#) and extend to the rest of the line in which they occur. Empty lines are ignored. Each other line is converted to uppercase letters for further use. In each line, a name (label) characterizing the compound filter set and the individual wheel positions are separated by any amount of white space (blanks). If there are more names than wheels in the instrument, the trailing names are ignored.

Each position (other than the star and the dash mentioned above) refers to a name in $CMANDIR/macros.instr and to a name in a file $CMANDIR/wheel[5-].instr, a set of files that enumerate wheels starting at index 0, again with the instrument’s name as the suffix.

Without arguments filter shows all available filter macros and the current one.

filter starts as background process and should be followed by a sync when used in a macro. The sync filter is generally insufficient here because the recomputation of the focus effect on the telescope may cause GEIRS to emit a slave telx proc command which also should be waited on.

23 fits

type: USER

23.1 specified header card

syntax: fits_header_card

Adds the FITS header card fits_header_card to a volatile local stack in the shared memory manager. This will be added in the FITS files created with the saves command of the future.

Generally cards that have keywords that already exist in the stack are replaced. If the card starts with COMMENT or HISTORY, however, the new card will be added without replacement.

Header cards with the HIERARCH convention must be at least 2 levels deep. If they appear to have only a single key, the HIERARCH prefix will be dropped:

- fits HIERARCH HUGO SANDERS = w / this will become a simple HIERARCH HUGO SANDERS = 'w'
- fits HIERARCH HUGO SANDERS = 1 / this will become a simple HIERARCH HUGO SANDERS = '1'

Attempts to add cards with organizational FITS information like BUNIT or END or CHECKSUM will be rejected. It’s believed that this could interfere with the file layout decision that are in the hands of GEIRS.

25 gui

type: USER

syntax: gui [x server] [if font]

Starts the graphical user interface (GUI) for the camera. For the description of the options, see Chapter 11 [control], page 5.

26 help

type: USER

syntax: help

Prints the list of commands allowed to the current class of the user.

syntax help command

prints information about the specified ’command’, where

- ‘syntax’ describes the (curious) parameters and switches.
- ‘type’
  - USER: normal user command
  - ENG: engineering command, not needed for standard operations
  - SUPER: system safety critical commands. A password is required to use such a command. The observer’s name is used as the password.

Parameters in ‘[]’ are optional. List of exclusive values are enclosed in ‘|’.

27 history

type: USER

27.1 history

syntax: history

syntax !? 

Print the GEIRS shell command history.

Chapter 24: get

The stack of cards runs in addition to the mechanism to add FITS keywords with the CMANDIR/macros.instr • files. All information in the local stack will be lost when GEIRS is terminated.

23.2 purge stack of information

syntax: fits clr

If the argument is the simple 3-letter word clr, all keywords and comments that have been added earlier with the fits command are deleted. This is like starting from a fresh empty local header card stack, equivalent to the situation when GEIRS is started.

24 get

type: USER

syntax: get varname/element [ .. ]

Reads one or more variables of the shared memory info database. When the ‘varname’ is an array, the entire array is listed. Alternatively, specifying an array-element in [index] reads only that single array element.

Warning: If the varname is shortened, only the first match in some internal table is returned.

In the instrument shell, a TAB in the command line will autocomplete or list the available arguments.

Examples:

- get CAMRA
- get ITIME
- get WHEELS
- get AIRMASS
- get MTYPE
- get READSYS
- get CAMBUSY
- get CPAR1
- get CNEP
- get CAMPATH
- get CTIME, RUT
- get CPAD
- get CTYP
- get DETUN
- get DEFXE
- get FS_FCAP
- get FRAMEERING
- get HAWK
- get LSTFILE
- get MACKS
- get MAXIMAGES
- get VIMAGE
- get WLAN
- get OBSERVER

Chapter 28: idlemode

27.2 previous

syntax ??

Repeats the last GEIRS shell command.

27.3 previous search

syntax !abc

Repeats the last GEIRS shell command that starts with ‘abc’.

28 idlemode

type: USER

syntax: idlemode [action] [threshold]

Selects the detector’s idlemode. The usual default is ‘wait’ with ‘Lir’, but that depends on the instrument/camera.

Without parameters it returns the current idle mode, which shows how the read would terminate the idle mode and what pattern the ROE runs on the detector while the idle mode is active.

This command is rejected while the camera is busy (i.e., while readout or wheel motions are in progress) unless it is only a query of the current parameters. Hence a previous call to sync may be needed in non-interactive modes, for example in macros.

Parameter action

- `break` interrupts clocking of the idle mode to start the next read immediately
- `wait` completes full idle cycles and transits seamlessly from idle clocking to clocking of the readout-pattern.
- `auto` uses an integration time threshold to switch between the ‘break’ and ‘wait’ mode.

If the action is set to `auto` and a number of a threshold follows, the threshold should be a floating point value representing an integration time. If the integration time is shorter than the threshold, the idle mode `wait` is used, otherwise the idle mode `break`.

If the parameter action is set to `default`, the default idle mode of the instrument is set.

The parameter `threshold` sets the idle type. The available choices depend on the camera. Valid idle type are:

- `default` sets the instrument default idletype
- `ReadOutMode` uses the current read-out-mode without transfer to the workstation (this was the default with previous software releases)
- `Restart` fast-reset cycles
- `Lir` reset-level-read cycles
- `Lir` interleaved read-reset-read cycles
Chapter 34: lamp

29.1 camera


Initialize the camera. Valid camera names and optics are defined in $CAMHOME/src/cameratypes.h. Camera names are not case sensitive and one of (Panic, Nirvana, Luci1, Luci2, Camenes, Aip, NTEmsg and NTExpec).

If no name is given, the current settings are used and checked.

Examples:

- init camera -r [...] re-initializes the camera settings and the specified options.
- init camera [...] initializes the camera implementing/setting the defaults.

Without the option -r, all options which are not set with this init command are set to default values of this camera.

With the option -r, all current settings of the camera remain as they are, unless they are overwritten with another option of this command.

- -r: #chans is the number of ADC-channels used to read a single detector chip. To get the total number of ADCs of the ROE multiply by the number of chips. For all MPAI systems in use, this is 32 for the Hawaii-2 and Hawaii-2RG. For the Hawaii-4RG this can be 64 (2x32) boards in the ROE, or 32.
- -o: optics = [wide,high,veryhigh]
- -s: status = [offline,online] (access to ROE hardware)
- -m: motors = [offline,camera,direct]
- -t: temperature-controller = [offline,camera,direct]
- -r: init camera -r does re-init but also re-setting all important last camera settings.

Init without parameters returns the states of the instrument parts.TBD

Chapter 32: time

30 inwindow

Syntax: type: USER

Syntax: inwindow

Open a window to setup the camera/telescope configuration. If you leave the window using the OK-button, the camera, the telescope and the wheels will be initialized if their setup was changed. The all-button forces a complete new initialization whether or not anything was changed.

31 interactive

Syntax: type: ENG

Syntax: interactive [on|off,reset]

If you use the interactive mode, the outputs in the shell are blocked after 10 lines, until you enter "<RET>". Default is "on". (All shell outputs are blocking if you use interactive=yes, and you may loose messages in the shell output ring buffer, if you set interactive=no.)

32 itime

Syntax: type: USER

Syntax: itime [min [sec]] [-f number] [-s status]

Set the integration time to time seconds. Without any argument it prints the actual integration-time status.

This command is rejected while the camera is busy (i.e., while readout or wheel motions are in progress) unless it is only a query of the current parameters. Hence a previous call to sync may be needed in non-interactive modes, for example in macros. See Chapter 66 [macro], page 38.

If the option -f or -stutter is added, the value is additionally printed to the associated output stream - only useful if called as cmd:exit itime -stutter

The options -f and -stutter are setting adjusting-factor and offset, which are used (until the value(s) are set to 0.0) according to formula: used itime = -m: [multiple adjustment ′-’offset = 0.0313 sets adding of constant offset of 0.0313 = 0.020 sets adjusting itime to a multiple of 0.020 seconds]

Rule: adjusted itime always >= given itime.

Exception: (adjusted time value < minimal integration time) will always set the minimal integration time.

Note: These values can be configured by the user via the environment variables CAMTIME_MUL and CAMTIME_PLUS, else the defaults are set to ‘no adjustments’, but may always be changed via this itime command from the user.

33 kill

Syntax: type: USER

Syntax: kill name | [name | [name]]

If name is one of the set {display, catchup, asleep, gui, control, telop, temporal, shuntoff, inwine} then a software-terminate flag is set to the named process. All other name result in syntax errors.

However, setting this flag does not necessarily mean that the process is able to recognize it since the mechanism works passively (sets a flag).

If name is one of the set {read, save, shell, tele, wheel, filter, host, aperture, optics} then first a ‘soft-kill’ signal is sent to the process. If after timeout (default 10 seconds) the process is still alive, a ‘kill’ signal is sent to the process.

The option -s # following the process name overrules the default timeout to wait for the process to terminate. The units of the parameter are seconds.

Additionally, PIP-entries and serial line flags are cleared, and maybe some other flags that used a reset.

Note: If name is (macro), it does not terminate the macro process, but reports only values of the macro status. If no macro process is active, it cleans the macro status.

Warning: kill read should hardly ever been used in favor of abort.

34 lamp

Syntax: type: USER

Syntax: lamp ALLOFF

Syntax: lamp L-[1|2|3|4|5] OFF

Syntax: lamp L5 ON [1|2|3|...9]

Syntax: lamp ALLOFF

Syntax: lamp L {1|2|3|4|5}OFF

Syntax: lamp L {1|2|3|4}ON

Syntax: lamp L5 ON {1|2|3|...9}

The command is only available if GEIRS is started for PANIC.

Controls the calibration lamps by executing the lamps.sh script with the syntax of the common rflat CAHA command. The rflat is executed on a remote computer.

All lamps can be switched off at once with the argument ALLOFF. Lamps 1 to 5 can be individually switched on or off. Lamp 5 can be switched on a specific power which is indicated by small integer numbers in the range 1 to 9.

Examples:

- lamp ALLOFF
- lamp L3 OFF
- lamp L4 ON
- lamp L5 ON 3

35 last

Syntax: type: USER

Syntax: last [file]

Returns the filename of the most recent image that was saved and stores the filename into ‘/dev/last’. (Relative path names are interpreted relative to the GEIRS start directory. This is considered a bug and may change in the future.)

Without the parameter, the filename is added to the file ‘/dev/last File’ in the directory $CAMHOME (which usually is ‘/tmp’).

36 load

Syntax: type: USER

Syntax: load filename [{n} [incr]]

The command treats the FITS data in the file as single frames, not images, to be pushed onto the buffer of frames as if they were just read. So to see that new frame in the online buffer, you need a reset.

If option nincr is given this value is added to the filename-numbering for loading the next FITS file.

The command has only been used to load a sky/background image that is subtracted from the master image in the display.

The command treats the FITS data in the file as single frames, not images, to be pushed onto the buffer of frames as if they were just read. So to see that new frame in the online buffer, you need a reset.

The command treats the FITS data in the file as single frames, not images, to be pushed onto the buffer of frames as if they were just read. So to see that new frame in the online buffer, you need a reset.

The command treats the FITS data in the file as single frames, not images, to be pushed onto the buffer of frames as if they were just read. So to see that new frame in the online buffer, you need a reset.

The command treats the FITS data in the file as single frames, not images, to be pushed onto the buffer of frames as if they were just read. So to see that new frame in the online buffer, you need a reset.
Since the shared memory frame-buffers are unsigned short integers, the image will not be correct if the FITS file is encoded with a different BITPIX value. This basically means that most of the FITS files saved by GEIRS should not be read that way to avoid misinterpretation, because they have been stored as correlated output with 32 bits per pixel, not as single frames."

If none of the n_files is added to the filename.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.

The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory. The command may fail if someone else created the directory but did not give sufficient rights to the Unix group or others to switch to that directory.
The automated numbering scheme of FITS files: A file name with an alphabetic letter at the end (basename) will be extended by a pattern with 4 digits. Basically a single next creates a name space for up to 9999 FITS files. During each save, GEIRS scans the current output (save) directory for file which matches the pattern of the filename followed by four digits and the extensions .fits, .dump, and increases the largest 4-digit number found by 1 to create the default file name of the FITS file.

```plaintext
next hugo...read ...
    save ... # no filename, creates hugo_0001.fits if no hugo_????_fits present
read ...
    save ... # no filename, creates hugo_0002.fits, because hugo_0001.fits present
read ...
    save ... # last file name is saved.
next: # creates file lastfits.fits
read ...
    save ... # no filename, creates hugo_0003.fits, because hugo_0002.fits present
```

The naming scheme is preserved during quit (shutdown) and restart operations because GEIRS stores the active filenames in CAMBUILD/pttrns/CAMFILENAME during quit and reads it from there at startup. Option `+t` (with or without a file name) tells GEIRS that the next save command should not use the default file name, but a temporary test file name. After the next save command the default file name is automatically reactivated, also if there was an error or problem with the save command. (Multiple sets of options in a single save command are treated as a single save command. This may lead to cases where the save cannot succeed if that implies using the same FITS file name multiple times.)

If option `+t` is given without a filename, the special name ‘test’ is used, else it uses the given filename. Attention: The test-filename is not used, if the next save command is given with a filename; it is only used if save is given without a filename.

To deactiviate the previously commanded temporary test filename, you might either just call `next #` without filename argument, or `next filename #`, where filename will be handled like above, or `next filename #`, where filename will be handled like above.

next tests if the ‘filename’ already exists in the current path and issues a warning if this is the case. (The next save will then fail, if the file already exists in the current path, unless an option for overwriting (Dangerous!) is given.)

If next is used without argument, the command returns the next default and next test file names, where the one which would be used at the next save command is marked as ‘next’. (The ‘test-filename’ shows you also the starting string of the saved files, which are not queued to automatic storing to tape, etc.)

### 42 object

<table>
<thead>
<tr>
<th>type</th>
<th>USER</th>
</tr>
</thead>
</table>

### 46 pipe

<table>
<thead>
<tr>
<th>type</th>
<th>SUPER</th>
</tr>
</thead>
</table>

```plaintext
pipe ...
    pipe 33 903 0 0x1f
    pipe 33 911 0 0x0
    pipe 33 508 0
    pipe 33 911 0 0x1
    pipe 33 903 0 0x1
```

Send `command` and `parameters` directly to the camera electronics. In the simple format, no interpretation or limit checking is performed.

- `-i` (int) just send command but do not wait for any answer.
- `-l` (list) interprets the `command` and optionally any of the further parameters as the name of files with a command list. The list files are attached here without their instrument suffixes. The search path is the CAMBUILD/pttrns subdirectory. In this format with the `-list`, the usual expansion of lists in the files happens: removal of comments, expansion of the multipliers, substitution of variables and so on. See the constructor manual for details.
- `-t` (timeout) followed by an integer increases the timeout for the communication to the ROE to that number of seconds.

To turn off the front LED's of the 3 ROE boards that are under software control, for example, use the three commands:

```plaintext
pipe 33 509 0
pipe 33 911 0 0x0
pipe 33 903 0 0x0
```

or to turn them on use:

```plaintext
pipe 33 506 0
pipe 33 911 0 0x1
pipe 33 903 0 0x1
```

In newer pattern versions, there are files ledoff * and ledon *, so to the same effect we may use:

```plaintext
pipe -list ledoff
pipe -list ledon
```

### 47 ptime

<table>
<thead>
<tr>
<th>type</th>
<th>ENG</th>
</tr>
</thead>
</table>

```plaintext
ptime [?]
```

Sets the base time for the ptime time (which is $\text{ptime}$ in the roe interface).

- for observers $\text{ptime}$ [default = slow] @ sets the configured base-times for $\text{ptime}$
- for engineers $\text{ptime}$ @val $\neq$ value >= 0 as base-time

### 48 put

<table>
<thead>
<tr>
<th>type</th>
<th>USER</th>
</tr>
</thead>
</table>

```plaintext
put [[-i,-f,-d,-s]] offset value
```

Write ‘value’ at ‘offset’ into the shared-memory infopage (database).

- `-i` ‘value’ is an (int) (default)
- `-f` ‘value’ is a (float)
- `-d` ‘value’ is a (double)
- `-s` ‘value’ is a (char*)

### 48.2 named

```plaintext
put varname[element] value [varname2[element2] value [...]]
```

A set of variables held in the shared memory data base may be put (set). The names have to match the names in the data base in full; abbreviating names is not supported.

When varname is an array, all array elements are set to value. In this case it is almost always better to address a single element of the array with the [element] index.

In the instrument shell, a TAB will autocomplete or list the available varnames.

See Chapter 24 [put], page 11.

To keep GEIRS informed about LUCI changes of the pixel scale, one would use for example:

```plaintext
put PIXSCALE 0.016
```

where the numerical value is in arcseconds per pixel.

### 49 pwnd

<table>
<thead>
<tr>
<th>type</th>
<th>USER</th>
</tr>
</thead>
</table>

```plaintext
pwnd
```

Prints the current directory for the saving operation (UNIX style) and the free space in MBs.
50 quit

type: USER

51 repeat

type: USER

54 rotype

type: USER

55 rtime

type: ENG

Example:

53 roe

type: USER

56 saad

type: ENG

Example:

57 save

type: USER
Chapter 65: sfdump

58.1 savepath

Syntax: set savepath [pathname]

Echo or set the directory (path) for saving files.

If the directory does not exist, it is created.

- `-a` append the string of the date-format to pathname
- `-u` append the string of the date-format
- `-1` starts with a slash, it is interpreted as a full computer, otherwise as a file relative to $HOME/tmp
- `-s` create pathname as subdirectory of the current savepath
- `-C` Add CHECKSUM keywords to the header-data units. This is not yet implemented

The effect of defining the new directory is seen in all subsequent commands that are executed relative to the save path, for example `cd`, `cd .`.

The command may fail if the account running GEIRS does not have sufficient rights to create this directory or any of its parent directories.

The command may fail if someone else created the directory but did not give sufficient rights to the GEIRS processes (i.e., to the account that starts GEIRS) to switch to that directory.

The command will return a warning if the account running GEIRS is not the owner of the directory and the group of the account does not have write access to the directory. This will lead to problems as soon as GEIRS tries to create the FITS files in the directory.

If GEIRS is shut down smoothly with quit as it should, the directory is stored in the file CAMMP/CAMPATH such that the next GEIRS session reads it from this file to provide the new path. Note that this mechanism of renaming the path name of the previous session does not notice if the path name contains some indications of a formatted date. So if the path name is /usr2.2013.1110 then for example during a session in Nov. 10th, GEIRS is shut down and restarted a month later, the path name still is/usr2.2013.1110 in December.

Note that for CARMENES the command savepath must not be placed after the save and before the read, but before the read. Otherwise the single frame FITS files are created in the wrong (old) directory and will not be found by save, such that also the first-stage pipeline will not find them.

58.2 macropath

Syntax: set macropath [pathname]

Echo or set the directory path for macros.

If the command is used without argument, it just returns the current name of the configuration file. This is an empty string if the dumping is not active.

59 sfdump

Type: USER

Syntax: sfdump [pathname | -c]

Specifies a configuration file with instructions to dump a set of windows of each single-frame to a directory while in any multi-correlated or double correlated read mode. The command merely configures the action. GEIRS actually dumps the files if it notices that it runs in one of the applicable read modes, that at least one window is defined in the file, and that new frames from the ROE arrive.

If the command is used without argument, it just returns the current name of the configuration file. This is an empty string if the dumping is not active.
65 *Subwin*

**Type/Status**

- **User**
- **Status**
  - `user` (e.g., `subwin user 99 on`)
  - `status` (e.g., `status`) for status information

**Example**: `subwin user 99 on` enables a software window with the ID 99.

**Syntax**

- `subwin [SW|HW] [clear] [on|off] [SW|HW] #wid [xlstart ylstart xsize ysize]` for hardware and software subwindows.

**Description**

- Enables or disables software and hardware subwindows.
- `clear` removes all subwindow definitions.
- `on|off` enables/disables the subwindow.
- `SW|HW` specifies the hardware or software.
- `#wid` is the window ID.
- `xlstart ylstart xsize ysize` define the subwindow's coordinates and size.

**Validity**

- Only one of the three listed alternatives is allowed.
- Without options, `status` returns the instrument's specific status list of file `status_user.cfi` if this file does not exist, it returns all possible status information of GEIRS (like `status`).

- **Options**
  - `clear` returns all available parameters of GEIRS.
  - `on|off` returns all statuses listed in file. The instrument's extension, e.g., `user`, is appended to the name if `off` is used.
  - `status <command>` returns parameter set defined in `$CAMINFO/status_cfg.<instru>`.
  - `status <command>` returns all available status information of GEIRS.
  - `status -f file` returns all statusses listed in file. The instrument's extension, e.g., `.lucifer`, is appended to the name if no dot `.` appears in the name.
  - `status -f file` returns only that specific status information.

**Examples**

- `status` returns parameter set defined in `$CAMINFO/status_user.cfi`.
- `status -a` returns all available status information of GEIRS.
- `status -f file` returns all statusses listed in file. The instrument's extension, e.g., `.lucifer`, is appended to the name if no dot `.` appears in the name.

**Clears, enables/disables, and sets the software (SW) and/or hardware (HW) subwindows and translates them to pattern windows.**

The union of the hardware windows are the data sent from the ROE to the GEIRS workstations via the fibers. Hardware means that the patterns run on the firmware of the ROE; determine which group of lines of pixels are either skipped or converted while reading the detector; one of the major side effects of skipping regions is that the shortest integration time becomes shorter. Pattern windows are the sub-regions of the hardware windows repeated in each of the 32 readout channels on each detector.

The GEIRS software on the workstation can in addition cut through regions of those hardware windows it received; this post-processing we call SW windowing. To set SW windowing, the software windows may be sligherly larger than the parameters xsize and ysize, which may overlap. The software windows with different #wid indices may overlap.

The order of the non-numerical parameters (clear, on, off, SW, auto, HW) can be swapped: the on, off or clear may also be placed after the SW, auto or HW.

The two sizes up such that the product is a multiple of 8 pixels, therefore a multiple of 16 bytes, such that the total over all 32 readout channels of each chip is a multiple of 512 bytes.

**Recommended command sequences**

- `subwin clear SW Clear all SW windowing definitions`,
- `subwin clear HW Clear all HW windowing definitions`,
- `subwin clear SW Clear all SW windowing definitions`.

**Important**

- Just setting the windows coordinates does not activate windowing. An explicit `subwin on` is still needed.
- Removing a single subwindow from the list of known subwindows is not possible. It is only possible to deactivate all of them. Still the deactivation needs to be followed by a `subwin auto on`.

**AND**

- If subwindowing is switched on, each subwindow command needs to recalculate the whole subwin-logic. Therefore it is always a good idea to execute first `subwin off` before changing subwin properties.
66 sync
  type: USER
  syntax: sync [read] [tele] [abs] [save] [test] [none] [all] [macro] [# time]
  syntax: sync -e

  Waits until the background processes named by the arguments have terminated.
  The model of the command execution means that these background processes reply
  with an early response to their command. These processes read, talk, and so on
  are in some sort of common group because they need some time until they finish.
  After starting any of these processes, commands like status and get can be used to
  monitor how far which of these processes have proceeded. The sync finally is actually
  waiting until these processes have finished (in some cases triggered by individual
  terminals), and responds which the information collected by the processes during
  their execution as parallel background processes.

  Think of the sync as blocking/delaying all follow-up commands (even short)! until sync
  itself returns. In practice this means do not send a sync if you may wish to abort the read
  at some time in the future.

  It returns the last errors of the background processes. If no name or all are specified, these
  are all errors, otherwise the errors of the process specified by the command. This allows to
  watch immediately the error of a background process.

  At each start of a background process it clears its last error.

  To clear all last errors of any background process use sync -e.

  sync -e [# time] waits like sync all [# time] but clears on return all previous errors
  of the background processes.

  @ time: int/float-value as last argument:
  sync waits at least @ time seconds, before checking on any process to synchronize with.
  This is a mean to ensure that even on a busy system a just scheduled command has indeed
  started (which may need some time).

  If the argument none is present, it does not sync with processes, even if process names are
  in the argument list.

  If no process parameter is given, sync waits for the termination of all five background
  processes listed above and currently running in the system, but not on the macro process.

  Without the @ time specification the sync waits at least 2 seconds. The signature #
  indicates that this duration may be specified in a floating point format.

  Examples
  sync -s [# time] waits like sync all [# time] but clears on return all previous errors
  of the background processes.

  sync 1.5 - synchronizes with all background processes after
  waiting a default time.

  sync:USER 1.5 - synchronizes with all background processes after
  waiting a default time.

  The telescope interface may return the following error codes:

  • 40 Incorrect alpha value.
  • 41 Incorrect delta value.
  • 42 Incorrect epoch.
  • 43 Position not reached.
  • 44 Telescope keeps on moving.
  • 45 Timeout when moving the telescope.

67 system
  type: USER
  syntax: system [ ]

  Executes any system command, where cmd might be any combination of arguments. On
  problems with special characters surround the cmd with the character ".

  sync abs -f 2 -i # save next data of 2nd read
  sync none # on all processes except the sync process.
  sync all # on all processes including the macro process.
  sync none # on all processes except the sync process.
  sync all # on all processes including the macro process.

  Type: USER

  68 tdebug
  type: USER
  syntax: tdebug [text [anytext [anytext]]]

  Writes an entry in the format '2004-05-28 11:23:41.3794 ZD account (logentry)
  alltext' to the log stream, where it can be retrieved via journal(1).

  Alltext (limited to roughly 2048-8192 chars) is the concatenation of all the arguments.

  Examples
  tdebug -s [# time] waits like sync all [# time] but clears on return all previous errors
  of the background processes.

  tdebug 1.5 - synchronizes with all background processes after
  waiting a default time.

  The telescope interface may return the following error codes:

  • 40 Incorrect alpha value.
  • 41 Incorrect delta value.
  • 42 Incorrect epoch.
  • 43 Position not reached.
  • 44 Telescope keeps on moving.
  • 45 Timeout when moving the telescope.

  The telescope interface may return the following error codes:

  • 50 Incorrect value in the delta offset
  • 51 Incorrect value in the delta offset
  • 52 Alpha and delta positions not reached
  • 53 Alpha position not reached
  • 54 Delta position not reached
  • 55 Timeout while moving to position

  tals is a "background" process and should have a sync after it.

69.2 relative
  syntax tele[scope] [relative] [int] or [dalpha ddelta]

  Moves the telescope by dalpha and ddelta arc-seconds. The numerical value of dalpha
  is supposed to include the factor cos(delta) of the current position. (It is removed by
  GEIRS.

  The telescope interface may return the following error codes:

  • 50 Incorrect value in the delta offset
  • 51 Incorrect value in the delta offset
  • 52 Alpha and delta positions not reached
  • 53 Alpha position not reached
  • 54 Delta position not reached
  • 55 Timeout while moving to position

  tals is a "background" process and should have a sync after it.

69.3 focus
  syntax tele[scope] [focus] [ ]

  Moves the telescope focus by # units (i.e. micron) by sending tele_command t_focus to the
telefocus interface.

  Note: that it is impossible (due to some intrinsics of the telefocus interface in the CAHA
  scripting) to move to a focus position that has a negative value on the absolute focus scale.

  Example: If the focus position is at 5 units before the move request, and if the argument
  focus to this command is -7, the desired final focus position would be -2, and that negative
  value cannot be accomplished.

  The telescope interface may return the following error codes:

  • 30 Incorrect value for the relative focus motion.

  control software uses the equinox to correct for some Earth polar motions; the author of this
  manual here has no opinion on this.

  The telescope interface may return the following error codes:

  • 40 Incorrect alpha value.
  • 41 Incorrect delta value.
  • 42 Incorrect epoch.
  • 43 Position not reached.
  • 44 Telescope keeps on moving.
  • 45 Timeout when moving the telescope.

  The telescope interface may return the following error codes:

  • 50 Incorrect value in the delta offset
  • 51 Incorrect value in the delta offset
  • 52 Alpha and delta positions not reached
  • 53 Alpha position not reached
  • 54 Delta position not reached
  • 55 Timeout while moving to position

  tals is a "background" process and should have a sync after it.
Chapter 71: tempplot

• 31 Position not reached.
• 32 Timeout while moving to focus.
At the final stage of each motor motion (individually or in groups via the filter), the telescope focus is changed from within the motor procedure (unless disabled or the sum of the focus corrections of the previous and new filters are too small and so on.) It is therefore not recommended to issue a *tel* focus while motors are still in motion.

69.4 query

- tel[ecpoe] position

Reports the telescope coordinates (alpha, delta, hour angle and air mass) by sending the command tel[ecpoe] request to the telescope.

69.5 extended query

- tel[ecpoe] get[all positions]

Requests tel ec and tels focus combined.

69.6 TECs

- tel[ecpoe] name

Return telescope name and TECs status read from SW database, which means it might not be up-to-date to the current use. The following command series returns a more reliable status-to-date status information:

```shell
'tel get; sync tels 0.5; status tels [get]'
```

The tel command in this form without argument and the status tels do not need a sync, as they are only reading a status and do not call a *tel* function.

### 70 tempHistory

- type: USER

- syntax: temp file ['[x time1 time2'] [t time] ['y temp1 temp2'] ['d xserver']]

Same syntax and actions as for `temp`: see Chapter 71 [tempplot], page 42.

- 71 tempplot

- type: USER

- syntax: temp file ['[x time1 time2'] [t time] ['y temp1 temp2'] ['d d xserver']]

Creates a X11 window plotting temperatures from the log file (that was created by tempcon). Only relevant to some Calar Alto instruments that have log files in the GEIRS format. The horizontal axis is minutes, the vertical temperatures Kelvin.

- x: time1/time2 = begin/end time on the horizontal axis.
- y: time = begin time on the horizontal axis.
- d: display on which the window is opened (e.g. xserver)

#### output##

- 2004.31 3.947 0.121 4

which shows for each ADC-channel the mean, standard deviation and pixel count. The window will not be closed when the software is shut-down with the quit command.

### 72 Test

- type: ENG

- syntax: test [std,mean,var] [-q \[n rt n2\] | [-r n1 n2]]

Computes pixel statistics and augments the result to the file chin{test}.log either in $HOME/TEST (usually ~)/temp or in the current directory:

- std: prints averages and deviations over all pixels in all images of each channel and the same for the full image (with additional stdv of channels-stdv). This is the default option if neither n1 nor n2 are used.
- mod: prints the mean of all channels of each image.
- var: prints the median of all pixel-averages as a function of time, and the median of all pixel-variances as a function of time. (Note: this throws an error if less than 2 images are available!)

Default: the log file shows results channel-by-channel. The channel order follows the default orientation of each detector, independent of the user's flips or rotations. That means the channel enumeration is usually not trivially related to the display orientation.

#### Options:

- -q: for 'test var': de-activates median of variances independent of median of pixels (averages take it from the average-pixel). Default: variance is taken as independent median value.
- -r n1 n2: use images n1 through n2 (e.g. 'test var -2 11')
- -r n1: use images n1 through the last (e.g. 'test var -1 2')
- -s: use the software subwindows for the tests if activated. Instead of the default statistics looking at the quadrants, the statistics is done by subwindow.
- -x: # use only quadrant or output-channel or SW-subwindow number '#', where the numbering starts at 1 (e.g. 'test var -1 1'). This option is only available with the var parameter.

Warning: the combination -a -x is not allowed.

If the -a option is not used, all HW-read data are accumulated to get the statistics. With the -a option, statistics are calculated in SW-subwindows, ignoring in which HW channels these are located.

The defaulted output of the command 'std' for PYRAMIR (4 channels) for example is:

```
Output Type: temperature
Parameters:

  stdmean & stdv & n ( 4 outputs, 10 images, \n  ctepy trayous, camera Pyramir, since 1.000000, 
  ctime 1.000078, FULL-frame 1, mpsel 1048256)
```

#### output##

- 2004.20 3.839 2621440

```
...:
...:
...:
...:
```

#### output##

- 2004.32 3.921 2621440

### 76 Version

- type: USER

- syntax: version [\#position-name]

Returns the version string of the GEIRS software. This includes the user account and workstation name, the revision number of GEIRS, the time it was compiled, and the location of the binary files.

The '#p' means that merely the current patterns directory is returned.

### 77 Wheel

- type: USER

#### 77.1 Basic Use

- syntax: wheel [\#position-name]

Only relevant to some Calar Alto instruments that control motorized wheels by GEIRS.

Move wheel number '0' to the named position or return the status information. The '0' is the wheel number from 0 up to n (inclusive), as shown by the answer of the command *wheel* used without arguments. Examples:

- wheel
  - Returns overview of all wheels;
  - wheel 2
  - Reads and displays current wheel-positions.
  - wheel 2 williamson45
  - Moves wheel2 to the William45 position.

If the wheel number is replaced by the string *aperture*, the command addresses the first wheel that is in the *aperture* in the INFO/wheel.* file. For PANO this is actually the cold-stop wheel.

wheel becomes a background process and should be followed by a sync if called from within a macro.

#### 77.2 Focus

- syntax: focus [on,off,new]

focus wheels on/off/new controls the relative focus adjustment for the selected combination of elements. Example:

- 'wheel focus off' - deactivates the focus correction of all filter wheels for the subsequent filter wheel command, until it is reactivated.

Example:

- 'wheel focus on' - (re)activates the focus correction for the subsequent filter wheel commands, which are tagged
Chapter 77: wheel

84 for CHKFOCUS-correction in the wheelN.<instrument> configuration files.

Example:

'wheel focus new' updates the relative focus correction information to the current wheel positions, for all filters which are tagged via CHKFOCUS correction in the wheelN.<instrument> configuration files. Note that this call does not change the on/off state!

Focus correction is always done relative to the last filter combination which was saved at the last filter-correction action. Application note: Focus settings beyond the wheel focus control through the program will remain correct and will lead to correct relative focus corrections, as long as neither wheel/filter exchanges nor manual focus-changes occur while the GEIRS state is 'wheel focus off':

- To enable the correction of the relative wheel focus, after wheel changes and manual focus settings had been done in 'off state', use wheel focus new to discard the previous information on the relative focus correction that was remembered by the server, and to update it with the current focus.
- initialisation of wheels does not change focus, but activates the focus correction for the next wheel usage. (At initialisation time the focus correction is correct.)

77.3 relative
syntax: wheel [ #wheel relative #offsetsteps]

  wheel 2 rel -25 Moves wheel2 25 steps backwards.

77.4 init
syntax: wheel init

77.5 warminit
syntax: wheel initwarm

77.6 dialog
syntax wheel dialog [on,off]

The syntax with dialog on or dialog off enables or disables warning and error GUIs. Dialogs are usually shut off if GEIRS is driven by an external handler and there is no operator that could click on the buttons.

77.7 rdb
syntax: wheel rdb
wheel rdb re-reads the wheel and wheel-macro database files.

77.8 aperture
syntax: wheel aperture

Yields a list of wheels in the aperture class. For PANIC this is the cold stop wheel.

77.9 optics
syntax: wheel optics

Yields a list of wheels in the optics class. For PANIC this list is empty.

77.10 filter
syntax: wheel filter

Provides a list of filter macro positions.

Index

(Index is nonexistent)
Figure 16: Example of the window appearing if info camera is called from the Linux shell.
5.4 Macros

5.4.1 Aim and Configuration

Macro files are prepared to carry out specific, normally reoccurring, tasks in the spirit of batch processing. The macro utility is sequentially oriented; each line in the macro file contains a command of the set of Section 5.3 for every action normally assembled by using the camera GUI or typing commands into the GEIRS shell.

Empty lines in the macro file are ignored/skipped. The part of lines starting at a hash (#) up to the end of the line is chopped—and serves to add comments to the macro files. The maximum line length in the macro files is 256 bytes.

The syntax does not provide conditional and loop capabilities beyond the repeat command of the GEIRS shell itself. In that respect it does not extend the command interface.

Macros can be nested 5 levels deep, so the macro command may appear in a macro file. The most economic way to loop through a set of fixed commands a fixed number of times is to write this set into a macro file, then to call this macro from another “higher level” macro as many times as wished. In any way, these techniques are based on working with copy-n-paste on the ASCII files of the macros.

Every macro command may be issued with the prefix cmd_carmenes from a UNIX/Linux shell or with $cmd_carmenes from MIDAS.

Macro files are started from the camera control window (lower part, see Figure 5) or with the macro command to the instrument shell. As a matter of orderly book-keeping, it is recommended to use the file suffix .mac for all macro files. GEIRS searches first for the macro file with the exact name provided by the user, and then searches in addition (as a fallback) for that exact name augmented by .mac. So one may lazily use the file name without suffix in the GUI of Figure 5 and after the macro command if file names in the directories do have the .mac suffix.

The “macro path” plays the role of a search path for these *.mac files. It is set/changed with the third pull-down menu of Figure 5 or the associated set macropath GEIRS shell command, and saved across GEIRS shutdown/startup cycles in the file $CAMTMP/CAMMACROS. If a macro file is not found in that directory defined by the search path, GEIRS also searches thereafter through $CAMHOME/MACROS by default. If users store their macros in that MACROS subdirectory anyway, the “macro path” is not that relevant.

The macro files support DOS-style end-of-line markers of the composite carriage-return and line-feed bytes. In that respect one can copy these files from older Microsoft operating systems without using dos2unix(1). UTF-16 encoding of the newer Microsoft OS’s is not supported and supposed to be converted by tools like recode(1) before feeding them into GEIRS.

5.4.2 Syntax Checker

A basic syntax checker for a macro file is called with

geirs_MChk macrofilename.mac

which tests many (but not all) lines in the macro file for syntactical correctness. geirs_MChk
prints the lines that appear to be suspicious to standard output. It checks only the most common
commands that appear in macros. Commands like status, ls and other commands that produce
detailed output or open windows that needs interpretation by some listening program and do not
make much sense in macros are also reported. Numerical parameter ranges are only checked by
order of magnitude, or even not at all.

Checking all macros in a subdirectory is done with a loop in some bash shell similar to

```
cd $CAMHOME/MACROS
for f in *.mac ; do
  echo $f"..."
  $CAMBIN/geirs_MChk $f
done
```

The main benefit of using the checker is that typographic errors may be detected early, just after
editing the macro file. The GEIRS macro interpreter reads one macro line at a time and executes
it. If the total real time of executing the macro is long, errors in its late parts may lead to much
delayed abortion of the macro. A syntax checker adds some safety and time savings in that type
of scenario.

### 5.4.3 Total Integration Time

The total integration time in a macro is a sum over all products of the crep arguments and the
itime arguments that are active at the read. It can be calculated by calling

```
geirs_MItime.pl [-q] macrofilename.mac
```

Using the -q option gives a more quiet output, where the partial sums are not printed. The
macrofilename.mac is either a full path name or the name in the current working directory. If that
file is not found and the CAMHOME environment variable is set, the program tries to locate the
file also in the directory $CAMHOME/MACROS.

This scanner looks for lines of the format

```
itime seconds
crep count
read
quit
exit
repeat count read
macro othermacrofile
repeat count macro othermacrofile
```

and accumulates the sum over the products. If the itime argument is zero, it is replaced by (an
estimate of) 1.3 seconds.
5.4.4 Macro Generators

Lengthy macros can essentially be created by any other high level language with loop control. We provide some examples based on languages that are available on Unices.

5.4.4.1 Shell  Here is an example of a bash-shell executable with a double loop which generates 18 read-save cycles—three different values of the ems parameter and six different subframe coordinates. The bash-script would be put in a file like tst.sh, and generate the macro with chmod +x tst.sh; tst.sh > tst.mac:

```bash
#!/bin/bash
for e in 1 2 4 ; do
  echo "roe" ems $e ;
  for w in 0 1 2 3 4 5 ; do
    echo "subwin auto 1 " $(( w * 128)) $((w * 128)) 128 128 ;
    echo "read" ;
    echo "sync" ;
    echo "save -i -f 2" ;
    echo "subwin clear" ;
  done ;
done
```

5.4.4.2 awk  Another example of a double loop put into a file tst.awk and then generating a macro calling awk as awk -F tst.awk > tst.mac:

```awk
BEGIN {
  emsarr[1] = 1 ;
  emsarr[2] = 2 ;
  emsarr[3] = 4 ;
  wxy[1] = 0 ;
  wxy[2] = 2;
  wxy[3] = 3;
  wxy[4] = 4;
  wxy[5] = 5 ;
  for (e in emsarr ) {
    printf("roe ems %d\n",emsarr[e]) ;
    for ( w in wxy ) {
      printf("subwin auto 1 %d %d 128 128\n", wxy[w]*128,wxy[w]*128) ;
      printf("read\n sync\n save -i -f 2\n subwin clear\n") ;
    }
  }
}
```

5.4.4.3 m4  A third variant is to save some typing by expansion of m4 macros. If a file tst.m4 contains

```m4
#define a m4 macro expo with a roe-subwin-read-sync-save-sync atomic operation
define(expo,
  # interpret the first argument as an ems parameter
  roe ems $1
  # interpret the second and third parameter as the lower left coordinates
```

```m4
```
# of a window divided by 128
subwin 'auto 1 eval('2' * 128) eval('3' * 128) 128 128'
read
sync
save
sync
subwin clear
)

# run one exposure with ems=1, then one with ems=2 and another with ems=1
expo(1,1,1)
expo(2,2,2)
expo(1,3,4)

then m4 mloop.m4 > tst.mac generates a file with three exposures.

The same “macro generator” variants could be worked out in many other programming languages.

5.4.4.4 Driver Loops

An alternative is to drive the instrument through the cmd_extension interfaces of the scripts directory (here: cmd_carmenes or cmd_carmenes_new for example) from other programs/interpreters (bash, perl, python, tcl, MIDAS,...). Macros are not needed in such case.

A python script would do this by its os.system calls. An example with three outer loops over a variable e which feeds the ems setting and five inner loops over a variable w which implements a marching square subwindow might look as follows:

```python
import os
for e in [1,2,4]:
    os.system('cmd_nirvana_new roe ems '+str(e))
for w in [1,2,3,4,5]:
    os.system('cmd_nirvana_new subwin SW 1 ' + str(w*128) + ' ' + str(w*128) + ' 128 128' )
    os.system('cmd_nirvana_new subwin on auto' )
    os.system('cmd_nirvana_new read' )
    os.system('cmd_nirvana_new sync' )
    os.system('cmd_nirvana_new save -i' )
    os.system('cmd_nirvana_new sync' )
    os.system('cmd_nirvana_new subwin clear’)
    os.system('cmd_nirvana_new subwin off’)
```

In the more familiar bash shell an example might look like

```bash
#!/bin/bash
for (( j = 1 ; $j <= 10 ; j++ )) ; do
    echo starting exposure $j ;
    snd_panic_new read ;
    snd_panic_new sync ;
    snd_panic_new save ;
    sleep 10 ;
    snd_panic_new sync ;
    echo done exposure $j ;
done
```
5.5 Shell Commands

After installation of the manual pages (Section 2.5.2), the following documents of programs in the Linux shell are available by calling man(1), of which we show the first pages:
NAME
dfits(1)

SYNOPSIS
dfits

OPTIONS

DESCRIPTION

The standard output contains lines in the Skymaker format, a 100 followed by the two FITS pixel locations
of the file names.

Instead of a floating point number in degrees, the RA may also be provided by the standard two-colon
hex-format, 4D0 MM SS ss.

Instead of a floating point number in degrees, the DEC may also be provided by the standard two-colon
hex-format, +-DD:MM:SS.ss.

The prerequisites of running the program are the regions of interest of the 2MASS catalogue in the standard
catalogue style of Skymaker.

The command interprets all arguments that start with a dash as ds9(1) options, and all others as directories.

OPTIONS

DESCRIPTION

The command interprets all arguments that start with a dash as ds9(1) options, and all others as directories.

The user must close (or exit) the ds9 GUI to move on to the next FITS file.

EXAMPLES
de9open.
de9open.data/Path

delkoop-multframe.
de9open/mosaic/image/blah-dicarmenes/DATA/2015-02/*
NAME  
fits2csv(1)  General Commands Manual  fits2csv(1)

SYNOPSIS  

DESCRIPTION  
Ffits2csv converts FITS files to TIFF 6.0 files. The conversion program is available in the readout2fits package (available on geirs.readout2). The program reads the FITS header data following directions from a configuration file.

The first command line argument is the file name of an existing FITS file that is to be modified, i.e., input file. If the option is not used, the TIFF file with the same name as the input file will be created.

The second argument and optionally further arguments are ASCII file templates structured very similar to the TIFF files used with http://heasarc.gsfc.nasa.gov/fitsio/fitsio.html and http://heasarc.gsfc.nasa.gov/STS/ab/htmlf.fits/index.html.

TEMPLATE FILE SYNTAX
Each of these may contain empty lines and comment lines (starting with #) that have no effect.

The following options are available when creating the output file.

-l means that the image of the TIFF file is flipped upside-down before producing the output file. If both -X and -Y are specified, the output image is flipped twice.

-X means that the image of the FITS file is flipped left-right before producing the output file. If both -X and -Y are specified, the output image is flipped two times.

-bins means that the size of the output image will have bins number of pixels in an image times (where 4 is the number of bits in a pixel, with 8 representing the output image number of pixels in a TIFF file).

Another optional argument -t: The second argument and optionally further arguments are ASCII files structured very similar to the TIFF files used with http://heasarc.gsfc.nasa.gov/fitsio/fitsio.html and http://heasarc.gsfc.nasa.gov/STS/ab/htmlf.fits/index.html.

The TIFF files to be scanned.

The output file contains the same values, which is the coordinate of the pixel, the y ordinate of the pixel, and the image value in the pixel at that point. This ASCII file has been targeted for use with a gnuplot(1) session e.g.:

gnuplot gnuplot> set xlabel 'x pixel' gnuplot> set ylabel 'y pixel'

gnuplot> set grid gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplot> plot gnuplo
**NAME**
fitsort

**SYNOPSIS**
fitsort [flags] [fitsfits]

**OPTIONS**
-d means that fitsort does not print the header line with the FITS keywords.

**DESCRIPTION**
The standard input of fitsort must be the output of the dfits() command. The program prints a spreadsheet table with sub-separated columns which shows for each of the FITS files (row by row) lines with the values of the specified keywords listed side-by-side.

This answers quickly a question like what have been the integration times for the exposures in the FITS file?

**EXAMPLES**
dfits *.fits | fitsort ITIME RA

dfits *.fits | fitsort ITIME RA

dfits *.fits | fitsort ITIME RA

dfits *.fits | fitsort ITIME RA

---

**NAME**
fitsort

**SYNOPSIS**
fitsort [-d]

**DESCRIPTION**
This is the start of the first stage pipeline which calls in succession

- pipFits_ls (to collect the names of the raw FITS frames that are the main input with some Fowler-type of selection criterion),
- pipFits_nonl (to apply a nonlinearity correction to the raw frames that are selected),
- pipFits_ols to merge the corrected frames into a single image.

The command line argument must be the full path name of one of the raw FITS frames of the two LIR or many SRI(E) reads of the finished exposure. This is usually done by putting the script into the QueueFiles script of GEIRS and forwarding the argument received by the QueueFiles. The pipeline creates one additional raw-frame FITS file in the same directory with a name of the form car*_P.fits if successful.

This will fail for any of the following reasons:
- The nonlinearity calibration file is not found (see below under ENVIRONMENT)
- The command line argument is not readable FITS file
- The exposure did not create at least 2 FITS frames.

Logs are created in the file SCAMLOG/QueueFiles.log, which usually is the same as SCAMHOME/log/QueueFiles.log. This contains information on timing and on the files used in the various steps. No logs are created if the script is not called (-), for example if the GEIRS save command does not succeed.

Note that this pipeline script is a bash(1) script, so editing the parameters that are used for the sub-steps is a trivial matter of editing that ASCII file.

**ENVIRONMENT**
SCAMHOME/seq contain scripts used by the pipeline. Therefore this should be in the PATH variable.

SCAMTMP must contain the file pipFits_ls which is the specification of the nonlinearity correction coefficients for the full frame. The subdirectory SCAMTMP/pip will be used for scratch files and cleaned mercilessly by the pipeline script according to its own needs.

**EXAMPLES**
gpts_carmen_pipe *DATA/2015-08-11/CAR-20150811T10h10m07s-gpts-carmen-fits

The control GUI is usually either started by default when GEIRS is started or sometimes not started at all if any client takes full control of the exposures. This means the `geirs_control` command is mainly useful to operators who have closed the GUI with the X-button of the window manager and want to get it back.

For a detailed description of the layout and use consult the GEIRS manual.

Options submenu

Allows to specify the directories that contain:

- the Start path where the FITS files or raw dumps will end up.
- a Macro path where the software will search for macro files.

Some fine control of the files with the sounds (volume, on/off) is available in the Sounds submenu.

DESCRIPTION

The server responds to queries of clients to obtain data frames. It is started by default in the startup script because the real-time display actually receives its pixel data by polling the data server. Headless GEIRS sessions that do not open the real-time display probably don't need to start the server. The server is a thread-less server, which means it listens for a new request (command) on the port, sends back an answer (usually a data frame), and closes the port, ready for another request.

Commands

The server knows three commands, containing just a few letters:

- d (short for dimension): lists the server return the number of horizontal pixels and vertical pixels of the full frame and the count of available images in the real buffer, separated by blanks, in ASCII format.
- f (short for frame): Returns the data server.
- t (short for time): Terminates the data server.

The image is returned in binary format, 4 bytes for each pixel, in the endianess of the GEIRS workstation. The image is returned in binary format, 4 bytes for each pixel, in the endianess of the GEIRS workstation.

The environment variable `CAMBIN` must be set and point to a valid GEIRS directory of compiled binaries.

Some fine control of the files with the sounds (volume, on/off) is available in the Sounds submenu.

SYNOPSIS

```
geirs_control [options] [instrument]
```

- `options`:
  - `-c` Triggers a (explicit) de-compaction (de-duplication) of the memory tables.

DESCRIPTION

The program executes effectively as:

```
sync; echo 3 > /proc/sys/vm/drop_caches
```

when the free amount of memory drops below the threshold (which sets the Linux OS clear the current cache in the virtual memory).

This is called by GEIRS at the start of every read that involves the PLX driver. The current threshold is obtained by sending the command:

```
ps aux | grep GEARC | grep -v EDI | awk '{print $3}'
```

to the GEIRS interpreter. It can be changed using the `put` command of the GEIRS interpreter while GEIRS is running, for example:

```
put GEARDCACHE 1024
```

to use 1024 MiB in the future. Note that all values defined by `put` are forgotten when GEIRS is shut down, so the effect of such a modification would only last for the current GEIRS session.

The command will fail with an error message if the associated permissions of the last lines in the INSTALL have not been set.

NAME

geirs_control

SYNOPSIS

```
geirs_control [options] [instrument]
```

OPTIONS

- `-c` Triggers a (explicit) de-compaction (de-duplication) of the memory tables.

DESCRIPTION

The program executes effectively as:

```
sync; echo 3 > /proc/sys/vm/drop_caches
```

when the free amount of memory drops below the threshold (which sets the Linux OS clear the current cache in the virtual memory).

This is called by GEIRS at the start of every read that involves the PLX driver. The current threshold is obtained by sending the command:

```
ps aux | grep GEARC | grep -v EDI | awk '{print $3}'
```

to the GEIRS interpreter. It can be changed using the `put` command of the GEIRS interpreter while GEIRS is running, for example:

```
put GEARDCACHE 1024
```

to use 1024 MiB in the future. Note that all values defined by `put` are forgotten when GEIRS is shut down, so the effect of such a modification would only last for the current GEIRS session.

The command will fail with an error message if the associated permissions of the last lines in the INSTALL have not been set.
NAME
geirs_lamp.sh

SYNOPSIS  CARMENES-AIV04B-NIR-DCS-MAN01
gears_lamp.sh [L1] (0) [OFF]
gears_lamp.sh L1 (0) [OFF]
gears_lamp.sh L3 (0) [OFF]
gears_lamp.sh L4 (0) [OFF]
gears_lamp.sh L5 (0) [OFF]
gears_lamp.sh STATUS

description
The command is called by using the lamp command in the GEIRS shell.
It executes a rflats command for switching calibrations lamp on or off followed by a rflats status
command which
• leaves a FITS line at a standard place searched by GEIRS for add-on FITS lines, and
• echoes a string suitable for storage in the GEIRS online data base.

Warning: no timeout is currently implemented. If the shell hangs for any reason, this will cause an indefinite
pausing of GEIRS (because there is no timeout currently enabled on the GEIRS side.)

This file has no use beyond GEIRS implementing PANIC at the Calar Alto.

ENVIRONMENT VARIABLES
The variable TELESCOPE determines which of the CAHA computers is consulted to execute the rflats
command.

FILES
The output of the command is registered in SCAM/MHomeDataAdd/panic_1. If CAMHOME is not
defined, it is replaced by SHOME/.

EXAMPLES
gears_lamp.sh L5 ON 4

gears_patterns

NAME
gears_patterns

SYNOPSIS  gearss_patterns instrument

description
The patterns of the readout electronics are controlled by separate svn repositories than the main code, so the
name of the directory that keeps the patterns slowly changes in time as new features are build into the
readout.

The command helps to find the version (the directory) that will be automatically selected by GEIRS at

The command requires at least that the SCAMHOME environment variable is set correctly and that the directory layout is a standard one as used for all installations as described in the

EXAMPLES
gears_patterns luci1
gears_patterns luci2
gears_patterns carmenes
gears_patterns Nirvana
gears_patterns Panic

gears_redump.pl

NAME
gears_redump.pl

SYNOPSIS  gearss_redump.pl [-t] instrument

OPTIONS
The option -t suppresses the output of the time stamps at which the command was forwarded to the ROE,
which provides a slightly more readable table.

DESCRIPTION
The command is a debugging aid for the readout pattern developers in the develop subdirectory of the source code.
It prints the most recent contents of the entries downlinked to the ROE to standard output.

It does this by filtering the ROE log file for commands of the 7xx and 5xx family, not by actually reading
the content of some online ROE. Therefore the command can also be used if the ROE is run in simulation.

Note that the script is not generally placed into the scripts directory of CAMHOME; therefore using a full
path name or changing into the develop subdirectory is required to call it.

ENVIRONMENT VARIABLES
The variable CAMHOME must point to the top directory of the installation. The log file to be scanned is in
SCAMHOME/log/

EXIT VALUE
Always 0 (success)

EXAMPLES
gears_redump Luci1
gears_redump -t Luci2
gears_redump Panic
gears_redump Nirvana
gears_redump Carmanes

roelog=$(ls -1 $CAMHOME/log/*roe_Luc* | tail -1)
gears_redump.pl $roelog

gears_quitXterm.sh

NAME
gears_quitXterm.sh

SYNOPSIS  gearss_quitXterm.sh

DESCRIPTION
The script is called for example when a quit command is received by the GEIRS command (shell)
interpreter.

It executes a

It does this by filtering the ROE log file for commands of the 7xx and 5xx family, not by actually reading
the content of some online ROE. Therefore the command can also be used if the ROE is run in simulation.

Note that the script is not generally placed into the scripts directory of CAMHOME; therefore using a full
path name or changing into the develop subdirectory is required to call it.

ENVIRONMENT VARIABLES
The variable CAMHOME must point to the top directory of the installation. The log file to be scanned is in
SCAMHOME/log/

EXIT VALUE
Always 0 (success)

EXAMPLES
gears_quitXterm Luci1
gears_quitXterm -t Luci2
gears_quitXterm Panic
gears_quitXterm Nirvana
gears_quitXterm Carmanes

roelog=$(ls -1 $CAMHOME/log/*roe_Luc* | tail -1)
gears_quitXterm.pl $roelog
NAME  
geirs_sndwinJ — open a sound configuration GUI

SYNOPSIS  
geirs_sndwinJ

OPTIONS  
The command line argument is the name of the currently active instrument, like Nirvana, Luc1, Panic and so on. If the environment variable CAMERAV is set, the command line argument is not needed.

DESCRIPTION  
The GUI allows to enable or disable sounds for the GEIRS processing, to set the volume, and to assign sound files to the admin subdirectory to the set of distinct events that may trigger sounds. The SCAMTMP/CAMSOUDS file saves the configuration when GEIRS is shut down and is read to initialize it when GEIRS is restarted.

EXAMPLES  
rm tmp.fits ; geirs_srreConfig -f psf.fits -N 18 -w 20 -h 20 -v -o tmp.fits

NAME  
geirs_srreConfig — configure the srre mode’s reset windows of upcoming GEIRS reads

SYNOPSIS  

OPTIONS  
-R leads to more verbose output of the actions

-p defines the name of the ptms sub-directory with the patterns

-s defines the name of the text file with the quasi-FITS syntax that provides the window coordinates

-f defines the name of an existing FITS file with an image in the primary header

-N defines the number of reset windows to be created

-w defines the width of each reset window in units of pixels

-h defines the height of each reset window in units of pixels

-o defines the FITS output file name that should not show the reset window pixels as zeros, but (in reverse) all the other pixels as zeros.

-M specifies that the dark regions are defined by sorting with respect to median, not with respect to integral sum of the pixel values. Note that this will slow down the operation considerably, but is more tolerant with respect to hot or cold bad pixels.

DESCRIPTION  
See the GEIRS user manual.

In the first syntax, the existing configfile is read and updates the files in the pattern directory that concern the reset window placement.

In the second syntax, the current reset windows in the pattern directory are dumped to standard output.

In the third syntax, the input FITS file is scanned for bright regions, which are used to define the places of the reset windows. This proposal of reset windows is written in configuration file format to the standard output.

The general application of the third syntax is to compute the three parameters (two offsets for translation and one angle) of an image relative to some other image of the same pixel scale in the field of view by localization of a small set of bright stars. The main problem here is that a simple moving/scanning window that assembles the integrated flux (sum of all pixels) does not work under realistic conditions with bad pixels: so the option SM allows to sort the regions by maximizing the median, not the sum of pixels in a window. The main objective of the program is to avoid full-sort algorithms of pixels in all possible positions of the window by some biased partial-sort algorithm to gain considerable speed.

NAME  
geirs_tempplot — plot recent PANIC pressures and temperatures

SYNOPSIS  
geirs_tempplot

OPTIONS  
The call does not have any arguments or options.

DESCRIPTION  
This opens a gnuplot(1) window and shows the approximately three last days of temperatures and pressures that have been collected with the cronjob job installed with regular GEIRS installations of PANIC.

ENVIRONMENT  
The environment variable CAMLOG should be the full path name of the directory with the panictemp log file. If the variable is not set, the log file must be in CAMHOME/log.
**NAME**

glogRotate.sh

**SYNOPSIS**

```
glogRotate.sh [-S][<year>][<date>][<time>]
```

**OPTIONS**

- `-S`
  - Specifies the GEIRS log file directory.
- `<year>`, `<date>`, `<time>`
  - Specifies the date and time of the log file.

**DESCRIPTION**

This program is used to rotate the log files generated by GEIRS. It is usually called from a crontab entry with a syntax like

```
10 11 * * * export CAMHOME=${HOME}/GEIRS ; cd $CAMHOME/log ; ./glogRotate.sh
```

**EXAMPLE**

```
$ CAMINFO=infile.txt ./glogRotate.sh
```

---

**NAME**

pipFits_bad

**SYNOPSIS**

```
pipFits_bad [-R] <fits-in> <fits-out>
```

**OPTIONS**

- `-R`
  - Specifies an existing FITS file which is the name of the instrument.

**DESCRIPTION**

This program converts a bad pixel mask from ASCII to FITS format.

**EXAMPLE**

```
rm diff.fits ; pipFits_bad -R ~/GEIRS/INFO/badpixels.carmenes diff.fits
```
NAME
pipFits_flat(1)  
General Commands Manual

SYNOPSIS
pipFits_flat [-v] [ -s MinPercent | -a ] [ infile1.fits infile2.fits ... ]

OPTIONS
-a indicates that the name of the output file should be derived from the name of the last input file in the argument list by adding a _N in front of the fits.
-s specifies that the first percent of pixel data and the last percent of pixel data should be used for the fit. If the option is absent, the default value is 20.
-v specifies that the fit is to a polynomial of the form a0 + a1*x + a2*x^2 + a3*x*y + a4*y^2 with 6 unknown coefficients.

DESCRIPTION
The program performs a polynomial fit to a sequence of FITS images. For each pixel in the input files, a polynomial fit is performed. The fit function is subtracted from each pixel across the chip independent of its original value. The resulting image has a flat (nearly homogeneous) field (flux) effectively removed. It is written to a fits file (which must exist when the program is called).

Note that using a large value of the -s option (larger than 50) will effectively disrupt photometry in the fits file. This is because that the coefficient of the constant term of the fit is caching most of the average flux in the original image.

Note that using large values of -s or -a is not recommended, because the result will show all the known ringing artifacts of polynomial fits.

EXAMPLES
rm fits ; pipFits_flat -v -s 10-90 -p l Linrey_No_xi0087.fits fits

NAME
pipFits_noise(1)  
General Commands Manual

SYNOPSIS
pipFits_noise [-v] [-m] [-s MinPercent] [ infile1.fits infile2.fits ... ]

OPTIONS
-a indicates that the name of the output file should be derived from the name of the last input file in the argument list by adding a _N in front of the fits.
-m specifies that not the noise but the mean of the input files should be computed. This is useful to estimate dark currents of the input files.
-s specifies that the first percent of pixel data and the last percent of pixel data should be used for the fit. If the option is absent, the default value is 20.
-v specifies that the set of pixel data that is printed is the complement of the full set with the common start time stamp, i.e., the files that are not taken into account.

DESCRIPTION
The opposite extreme is estimated from the header of the file fitsin.fits and all files in the same directory created with the same read are listed in a single line.

In detail, the fitsin.fits file is scanned for its START_INT header line. All files in the same directory created with the same start time and with a BTFPIX=16 value in the first extension header (reducing GEIRS raw FITS frames) are listed in a single line.

If the option -M is used and sufficiently small, only the first Npair s and the last Npair s of the files are printed, and the file list contains an even number of files. The program therefore selects pairs of files out of a larger set in the manner of multi-end-point (Fourier) sampling of nondestructive infrared detector readout.

The value of the option has been taken into gete_spectral_ppp after 10, which means a maximum of 20 raw frames will be considered for any further processing for nonlinearity and merge into a single image.

EXAMPLES
rm fits ; pipFits_ls /DATA/rrr0009.fits

NAME
pipFits_nonlnonlinearity(1)  
General Commands Manual

SYNOPSIS
pipFits_nonlnonlinearity fit to a sequence of FITS images

OPTIONS
-a indicates that the name of the output file should be derived from the name of the last input file in the argument list by adding a _N in front of the fits.
-1 specifies that the first percent of pixel data and the last percent of pixel data should be used for the fit. If the option is absent, the default value is 20.
-v specifies that the set of pixel data that is printed is the complement of the full set with the common start time stamp, i.e., the files that are not taken into account.

DESCRIPTION
The opposite extreme is estimated from the header of the file fitsin.fits and all files in the same directory created with the same read are listed in a single line.

In detail, the fitsin.fits file is scanned for its START_INT header line. All files in the same directory created with the same start time and with a BTFPIX=16 value in the first extension header (reducing GEIRS raw FITS frames) are listed in a single line.

If the option -M is used and sufficiently small, only the first Npair s and the last Npair s of the files are printed, and the file list contains an even number of files. The program therefore selects pairs of files out of a larger set in the manner of multi-end-point (Fourier) sampling of nondestructive infrared detector readout.

The value of the option has been taken into gete_spectral_ppp after 10, which means a maximum of 20 raw frames will be considered for any further processing for nonlinearity and merge into a single image.

EXAMPLES
rm fits ; pipFits_ls /DATA/rrr0009.fits

NAME
pipFits_lscollect(1)  
General Commands Manual

SYNOPSIS
pipFits_lscollect FITS file names associated with a single exposure

OPTIONS
-a indicates that the name of the output file should be derived from the name of the last input file in the argument list by adding a _N in front of the fits.
-m specifies that not the noise but the mean of the input files should be computed. This is useful to estimate dark currents of the input files.
-s specifies that the first percent of pixel data and the last percent of pixel data should be used for the fit. If the option is absent, the default value is 20.
-v specifies that the set of pixel data that is printed is the complement of the full set with the common start time stamp, i.e., the files that are not taken into account.

DESCRIPTION
The opposite extreme is estimated from the header of the file fitsin.fits and all files in the same directory created with the same read are listed in a single line.

In detail, the fitsin.fits file is scanned for its START_INT header line. All files in the same directory created with the same start time and with a BTFPIX=16 value in the first extension header (reducing GEIRS raw FITS frames) are listed in a single line.

If the option -M is used and sufficiently small, only the first Npair s and the last Npair s of the files are printed, and the file list contains an even number of files. The program therefore selects pairs of files out of a larger set in the manner of multi-end-point (Fourier) sampling of nondestructive infrared detector readout.

The value of the option has been taken into gete_spectral_ppp after 10, which means a maximum of 20 raw frames will be considered for any further processing for nonlinearity and merge into a single image.

EXAMPLES
rm fits ; pipFits_ls /DATA/rrr0009.fits

NAME
pipFits_nonlnonlinearity(1)  
General Commands Manual

SYNOPSIS
pipFits_nonlnonlinearity fit to a sequence of FITS images

OPTIONS
-a indicates that the name of the output file should be derived from the name of the last input file in the argument list by adding a _N in front of the fits.
-1 specifies that the first percent of pixel data and the last percent of pixel data should be used for the fit. If the option is absent, the default value is 20.
-v specifies that the set of pixel data that is printed is the complement of the full set with the common start time stamp, i.e., the files that are not taken into account.

DESCRIPTION
The opposite extreme is estimated from the header of the file fitsin.fits and all files in the same directory created with the same read are listed in a single line.

In detail, the fitsin.fits file is scanned for its START_INT header line. All files in the same directory created with the same start time and with a BTFPIX=16 value in the first extension header (reducing GEIRS raw FITS frames) are listed in a single line.

If the option -M is used and sufficiently small, only the first Npair s and the last Npair s of the files are printed, and the file list contains an even number of files. The program therefore selects pairs of files out of a larger set in the manner of multi-end-point (Fourier) sampling of nondestructive infrared detector readout.

The value of the option has been taken into gete_spectral_ppp after 10, which means a maximum of 20 raw frames will be considered for any further processing for nonlinearity and merge into a single image.

EXAMPLES
rm fits ; pipFits_ls /DATA/rrr0009.fits

NAME
pipFits_nonlnonlinearity(1)  
General Commands Manual

SYNOPSIS
pipFits_nonlnonlinearity fit to a sequence of FITS images

OPTIONS
-a indicates that the name of the output file should be derived from the name of the last input file in the argument list by adding a _N in front of the fits.
-1 specifies that the first percent of pixel data and the last percent of pixel data should be used for the fit. If the option is absent, the default value is 20.
-v specifies that the set of pixel data that is printed is the complement of the full set with the common start time stamp, i.e., the files that are not taken into account.

DESCRIPTION
The opposite extreme is estimated from the header of the file fitsin.fits and all files in the same directory created with the same read are listed in a single line.

In detail, the fitsin.fits file is scanned for its START_INT header line. All files in the same directory created with the same start time and with a BTFPIX=16 value in the first extension header (reducing GEIRS raw FITS frames) are listed in a single line.

If the option -M is used and sufficiently small, only the first Npair s and the last Npair s of the files are printed, and the file list contains an even number of files. The program therefore selects pairs of files out of a larger set in the manner of multi-end-point (Fourier) sampling of nondestructive infrared detector readout.

The value of the option has been taken into gete_spectral_ppp after 10, which means a maximum of 20 raw frames will be considered for any further processing for nonlinearity and merge into a single image.

EXAMPLES
rm fits ; pipFits_ls /DATA/rrr0009.fits

NAME
pipFits_nonlnonlinearity(1)  
General Commands Manual

SYNOPSIS
pipFits_nonlnonlinearity fit to a sequence of FITS images

OPTIONS
-a indicates that the name of the output file should be derived from the name of the last input file in the argument list by adding a _N in front of the fits.
-1 specifies that the first percent of pixel data and the last percent of pixel data should be used for the fit. If the option is absent, the default value is 20.
-v specifies that the set of pixel data that is printed is the complement of the full set with the common start time stamp, i.e., the files that are not taken into account.

DESCRIPTION
The opposite extreme is estimated from the header of the file fitsin.fits and all files in the same directory created with the same read are listed in a single line.

In detail, the fitsin.fits file is scanned for its START_INT header line. All files in the same directory created with the same start time and with a BTFPIX=16 value in the first extension header (reducing GEIRS raw FITS frames) are listed in a single line.

If the option -M is used and sufficiently small, only the first Npair s and the last Npair s of the files are printed, and the file list contains an even number of files. The program therefore selects pairs of files out of a larger set in the manner of multi-end-point (Fourier) sampling of nondestructive infrared detector readout.

The value of the option has been taken into gete_spectral_ppp after 10, which means a maximum of 20 raw frames will be considered for any further processing for nonlinearity and merge into a single image.

EXAMPLES
rm fits ; pipFits_ls /DATA/rrr0009.fits
NAME
pipFits_olsOrdinary least squares fit through a set of images

SYNOPSIS
pipFits_ols [-F] infile1.fits infile2.fits ...

OPTIONS
-a indicates that the name of the output file should be derived from the name of the last input file in the argument list by adding a _P in front of the .fits
-F indicates that the FITS files should be ordered not by looking at the STOP_INT value in the primary header but at the FRAMENUM value.

DESCRIPTION
For each pixel in the input files an ordinary least squares fit is constructed using the time axis as the abscissa and the ADU file as the ordinate. Multiplying the slope with the integration time for that pixel and putting this value into an image, a single output file in a BITPIX=32 format is generated. The time stamps of the reads are taken from the STOP_INT keywords on the primary headers. The grid of these time stamps may be irregular; the function can fit data that are taken in clumps with Fowler-type selections, for example. (The function takes care of wrapping around time stamps if the exposure crossed UTC midnight where STOP_INT receives a kink of minus 86400 seconds.)

This fitted slope is meant to have an error (noise) which is roughly inverse proportional to the square root of the number of data on the time axis (i.e., of the number of input files).

Note that the minimum number of input files is 2. This means the program would work as well for data generated with the GEIRS h2 readout pattern. (In this case, the fit becomes degenerate and is the exact linear interpolation between the two data points.)

Most of the FITS keywords in the output file are copied from the last input file, which is supposed to be the last one created in the bunch and therefore to have the most complete history of keywords. The keywords CREATOR, BSCALE, EXPTIME and so on are modified in the integrated image.

There is one case where sorting the input files by the STOP_INT values in the primary header does not work. This happens if the frames have been created with the same STOP_INT value, which puts the same STOP_INT time into each file’s header. The linear fit for this arrangement would use the same abscissa for all pixel data, mathematically speaking this is equivalent to an attempt to fit a slope to a vertical data set and the user will see errors in the library that say that the data are linearly dependent. In this case, one can use the option -F to tell the program to arrange the files using the FRAMENUM value in the primary headers as the abscissa coordinate. The option -F is not needed for the frames generated as frame dumps while GEIRS runs, because these get individual STOP_INT data in their primary FITS headers.
5.6 Windows

5.6.1 Window Classifications and Nomenclature

GEIRS uses three basic types of windowing for a variety of different purposes:

1. Sets of sub-areas of the full frame detector images which are read from the detector and saved to the FITS files. The geometry is configured by the \texttt{subwin} commands to the command interpreter (Section 5.3). The underlying actions are that only sub-areas of the detector are read out, followed by some clipping of the resulting information by the GEIRS software. (What is created by the detector and readout hardware is called hardware windows and what is left in by the further reduction within GEIRS called software windows.) \textit{This is what is usually meant by an infrared astronomer talking about subwindows!} This appears to be implausible: instruments with bigger and bigger cameras are assembled, why would one discard some of the information in the images? The dominant reasons are that

- one can increase the frequency of image generation (Section 7.6), if the object quivers on some fast time scales, and/or
- reduce the disk space consumption of the FITS data by discarding large empty areas of the detector that are of no interest.

In summary, there is \textit{no} scenario where using this type of subwindows makes sense for CARMENES, so the \texttt{subwin} command should be ignored:

- The information of the dispersed spectra and the modes is well distributed over both detectors, and the traces of the spectra are curved. There are no vast rectangular non-illuminated windows left that could be used as subwindows.
- There is no need to achieve integration times smaller than the standard 2.7 seconds of a full-frame read; for this fiber-fed spectroscopic instrument integration times are fundamentally longer.

2. Resetting some areas of the frames after each \texttt{read} while the (otherwise non-destructive) reads of multi-correlated readout modes are ongoing. This is only supported by Hawaii-2 RG detectors in conjunction with some of the MPIA ROE’s (actually those of LUCI and CARMENES but not of PANIC) and will be called the subwindow reset mode. The interface deals with two aspects: First telling GEIRS where these windows are located in the FITS coordinate system in the detector plane, second telling GEIRS that the subsequent readouts should use the mode (command \texttt{ctype srre}, Section 5.6.2).

In a vague sense this results in some opposite of the windows in the first item: the selected areas remain dark(er) than the rest of the images, whereas in the bullet above only the areas inside the windows remain visible.

The main objective of this mode is to subdue brightly illuminated parts on the detector. One can prolong the integration time such that the (nondestructive) readout values of most of the pixels increases, but at the same time the pixels in the reset windows are often reset and do not saturate as they would otherwise. Overall this helps to increase the accessible contrast, and is typically used for spectroscopic modes (read: LUCI and CARMENES) with a small number of bright lines that can be sacrificed for the benefit of the others.
3. Saving some areas of the frames into scratch files while the non-destructive reads of multi-correlated readout modes are ongoing. This is implemented in GEIRS as a “software trigger” and shall be called the guide mode. This is configured with the `sfdump` command (Section 6.7).

GEIRS started for CARMENES uses this to create snapshots of each read during the multi-correlated non-destructive reads in preparation of its pipeline step that reduced these frames to a single image.

The general setup is that any mix of these three window clipping features with three different sets of windows is active/enabled. The current GEIRS patterns however do not support concurrent operation with windows of the first two types, which means the corresponding pattern is not implemented.\(^\text{24}\)

GEIRS does not provide what is commonly called the guide mode in the literature—where sub-regions of the detector are read out at a higher frequency than the full frame by interrupting the full frame readout a few times, reading the sub-area, and resuming the full frame readout—.

### 5.6.2 srre Readout Mode

Section 5.6.2 is of no relevance to PANIC or LINC-NIRVANA because either the detector or the ROE does not support this mode.

#### 5.6.2.1 Principle of Operation

On some MPIA readout electronics that control Hawaii2-RG detectors\(^\text{17}\)—that is actually only CARMENES and LUCI now—the `srre` readout mode has been introduced. It is characterized by reading frames of the detector “non-destructively” while the detector is integrating, and resetting some of the pixels after each of these reads. This readout mode is activated with the `ctype srre` command (Section 5.3) and has the same global behaviour as the `srr` timing. The parameter of the `ctype srre` has the same meaning as for the `srr`; it is the number of reads and therefore also the number of resets distributed over the integration time at the end of the “ramp.” If the integration time is 120 seconds, and the command is `ctype srre 7`, for example, every 20 seconds a frame is read and every 20 seconds the pixels inside the reset windows are reset.\(^\text{25}\)

The difference between the `srr` and the `srre` (with resets) is that after each readout a finite subset of the pixels (called reset windows here) on the detector is reset. Consequences of this extended mode are that

\begin{itemize}
  \item these reset windows never accumulate more light than equivalent to the time between two readouts, whereas the other pixels have much longer integration times that linearly rise from frame to frame. This points at the principal application of the mode: protection against pixel saturation, plus the beneficial side effects of less cross-talk and less persistence between exposures.
\end{itemize}

\(^{24}\)As noted under 1. above, this is not useful for CARMENES. For LUCI it would make sense if the orientation of the slits would be predicted, but other factors like not considering disk space as a cost factor and not considering file transfer times across networks.

\(^{25}\)Note the simple arithmetics: \(N = 7\) reads corresponds to \(N - 1 = 6\) intervals.
• in the standard linear fit of ADC value as a function of frame number through the samples within GEIRS that combine all the frame samples to a single image when calling `save`, the brightness of pixels inside the rectangles of the reset windows is essentially zero (because this is the slope through a time series of pixels that appear in each frame with approximately the same ADC values). An equivalent set of rather dark rectangular shapes of the reset windows is also visible if the frames are saved individually with `save -S...` or online with the `sfdump` configuration.

• The (minimum) integration time of the exposure increases roughly linear to the number of reset windows, needed for downloading and executing the resets sequentially. This prolongation is negligible in practise.

![CARMENES exposure](image.png)

Figure 17: Example of a CARMENES exposure with 74 reset windows on the left and 68 reset windows on the right detector chip, each 102 × 102 pixels. This is the fifth frame in a ramp of five. Note that for dark exposures like this the pixels inside the reset windows are brighter than those outside due to the reset anomaly. This effect is ignored in the method, because the main effect is that the pixels inside the reset windows stay at that constant level and don’t saturate—as explained in the main text.

### 5.6.2.2 Reads Parameter

The number of samples along the “ramp” is an integer

\[ N \geq 2 \] (3)

and a free parameter which is to be specified by the operator with the `ctype` command. There are some technical constraints, however, which set limits on \( N \), and some interrelations with other parameters of the exposure:
• With the standard full-frame readout and at the standard pixel time of 10 µs (command \texttt{ptime}), reading once the detectors in the \texttt{srr(e)} modes needs slightly less than 1.4 seconds, a hard limit to the full-frame sampling frequency. Supposed the integration time \( I \) as specified with the \texttt{itime} command (Section 5.3) is set from the usual considerations on fluxes, readout-noise and so on, this trivially leads to

\[ N - 1 \leq \frac{I}{1.4s}. \]  

(4)

A maximum of \( I \) in spectroscopic modes is defined by the allowable shift of the radial velocity (i.e., line wandering on the detector) due to Earth rotation-nutation, due to Earth ecliptic motion, changes in air mass and so on while integrating.

• The parameter \( N \) defines the number of frames that will be stored on the workstation which runs GEIRS. There is a finite amount of RAM \( R \) and an alternating buffer scheme in GEIRS which leads to a maximum amount of available memory of \( R/2 \) for a single exposure started with the \texttt{read}. [In fact this is set with the \texttt{CAMSHMSZ} parameter at startup (Section 3.2).] Let \( N_{p} = 1 \) or \( N_{d} = 2 \) be the number of chips in the camera for LUCI or CARMENES, respectively. Each frame demands \( 2 \times N_{d} \times 2048^2 \) bytes in memory, and the obvious constraint is

\[ N \leq \frac{R}{2 \times 2 \times N_{d} \times 2048^2}. \]  

(5)

Note that this number needs in addition to be divided by the cycle repetition parameter.
Figure 19: Zoomed view of an example of a CARMENES exposure with $14 \times 16$ reset windows on the right detector chip, This is the fifth frame in a ramp of five.

(crep in Section 5.3), if exposures are scheduled to follow immediately on each other.\textsuperscript{26} For the CARMENES workstation we have $R \approx 32$ GB, and each raw frame needs $2 \times 2 \times 2048^2$ B = 16 MB. So a maximum of $32,000/2/16 \approx 1,000$ frames can be stored

$N \leq 1000$. \hfill (6)

Note that this is just a guess. The actual upper limit is usually smaller because GEIRS is hardly ever configured to require the entire RAM of the computer for the purpose of its own buffers. Because GEIRS automatically reduces a number of samples to the maximum supported by the configuration (see the \texttt{ctype} in Section 5.3), it is trivial to figure out that upper limit as follows: Either

1. Send a \texttt{ctype} request with much larger number to the shell and read the result

   \begin{verbatim}
   linux> snd_carmenes_new ctype srr 10000
   Attention: Reads per cycle reduced from 10000 to 804 to fit into RAM buffer
   \end{verbatim}

2. or select the \texttt{srr} or \texttt{srre} mode and enter a much larger into the \texttt{#Reads} field in the controls GUI (Figure 5) and observe how that number is reduced within a second or two to the actual maximum.

- The fundamental idea of the \texttt{srre} mode is to clamp bright pixel regions. The parameter $N$ defines not only the number of reads along the ramp; because the number of resets equals

\textsuperscript{26}This is not relevant for the standard CARMENES operation because the \texttt{abort} command would terminate the entire sequence of exposures. So \texttt{crep} is almost always 1 here.
the number of reads, it also defines the number of resets along the ramp. Let $I_s$ denote some estimated maximum integration time that can be tolerated for saturation and memory effects in the reset regions, then

$$N - 1 \geq \frac{I}{I_s}.$$  

(7)

- Monitoring variations in flux, supposedly variable sky transmission due to cloud coverage, cosmics and so on proposes to set a maximum time difference between samples of the order of $T_c \approx 1$ minute. On that ground

$$N - 1 \geq \frac{I}{T_c}.$$  

(8)

- The parameter $N$ is implemented as some sort of delay between two scans of the ROE through the detector. From the point of view of the software on the workstation it leads to an arrival of $N$ frames (less if aborted) at regular time intervals $I/(N - 1)$ during the ramp. This gives a strict constraint on the FITS data files that can be created, because data that did not arrive on the workstation cannot be saved. There is an explicit and an implicit method of saving the frames (which means, generating FITS files):

  - The command `save` generates a single FITS file by calculating a least squares linear fit through (almost) all $N$ frames of each pixel. The command has a parameter `-S` which allows also to save individually each of the $N$ raw frames, and the command may be repeated to generate both, the “correlated” image and the set of raw frames (Section 5.3.). Note that the parameter $N$ impacts both (i) the time that is needed for the `save` due to calculating the fits, and (ii) the disk space that is required for the `save -S`. If one
would save for example all CARMENES raw frames obtained at the minimum period of the aforementioned 1.4 s, equivalent to a data rate of 16 MB /1.4 s ≈ 11 MB/s, the CARMENES disk space of 180 GB would be exhausted after 180,000/11 s ≈ 16,000 s ≈ 4.5 hours.

Note that the command `save` has a functionality to trigger any type of pipeline code that may deal with the FITS files (not the raw frames!) in more detail than just fitting a straight line through the time. [With the current GEIRS code this “hook” is already used to start the First Stage Pipeline [5].]

Because saving the probably large number of “fast” \( N \) frames is usually not needed and has some disadvantages detailed above, there is an online GEIRS mechanism (command `sfdump` in Section 5.3 and Section 6.7) which stores the frames on disk while the exposure continues. The configuration options explained in Section 6.7 allow to subsample the raw frames, i.e., to store only each second or each third etc. frame. This helps to avoid the time and disk space overhead mentioned above, but does not support irregular frame subset picks.

Figure 21 illustrates how the integration time and the parameter \( N \) fix a time \( I/(N - 1) \) between the raw frames that are stored in the computer’s RAM, and how a subset of these frames is dumped into FITS files for reduction by a pipeline [5] or online monitoring.

Figure 21: Upper plot: \( N \) raw frames at intervals \( I/(N - 1) \) in the computer’s RAM. Lower plot: \( N_F \) FITS files generated from raw frames sub-sampled with `sfdump`, here with a sub-sampling factor of \( s = 3 \) in eq. (9).

### 5.6.2.3 Correlated Image

The construction of a correlated image from the set of the frames is the same for `srr` and `srre`: An optional number \( N_d \) of first frames out of the \( N \) frames that have been read is ignored/dropped. For each pixel the standard linear least squares fit is generated individually through the \( N - N_d \) frames that have not been dropped. (Such a fit needs at least two
points on the time axis to draw a line, because one cannot fit a line through a single point to get a slope. Accordingly, if the number of frames that would remain is \( N - N_d < 2 \), these frames are not actually dropped but used to define the fit.\(^{27}\) The slope of that fit is multiplied by the number of time slots along the ramp, which is \( N - 1 \), to calculate the count equivalent to the full integration time along the ramp. This number is stored in the FITS file for that pixel.\(^{27}\) (There is a small speciality for CARMENES: if the pixel is inside one of the \texttt{srr} windows, not that number derived from the fit’s slope is stored in the FITS file but a zero. This serves as an indicator to any followup software that these regions inside the reset windows should not be interpreted as fits. The values in the online display, on the other hand, are not zeroed; the regions inside the reset windows of the glasses assembly in the GEIRS GUI in Figure 19, for example, are not entirely black, but some residual noisy speckles are still seen inside.)

The number of dropped frames is by default \( N_d = 1 \) with the current release of the software. It can be changed online with the \texttt{use} command; \texttt{use \texttt{srr skip 0}} for example would set \( N_d = 0 \) and hence incorporate all \( N \) frames in the fit for all subsequent exposures. \texttt{status use} shows the current parameters for all readout modes. The choice to ignore the first frame (the frame just after the reset) to define the ramp is a matter of experience with the frames for most of the detectors at the current mix of idle and read modes. Broadly speaking the reset frame is often too bright, even brighter than the second frame, although it represents a state of essentially zero integration time: there is some sort of memory persisting through the line resets. Since the primary application of the \texttt{srr(e)} modes comes with long integration times and values of \( N \) typically of the order of tens, ignoring one “bad” out of the these frames is basically no loss integration efficiency.\(^{28}\)

The raw 16-bit sequential frames are storing the pixels data as they are (no further interpretation or nulling). This gives a pipeline (smart enough to deal with the noise and the shifting effective integration time as discussed in Section 7.7) opportunity to extract line shape information even at these places within the reset windows.

5.6.2.4 Configuration  The number of these reset windows is limited to 128 per chip, which is a limit resulting from the number of reserved registers in the RoCon firmware (not the \texttt{H2-RG}). There is in addition an effective maximum of the total number of reset windows (i) on both chips of CARMENES of currently 137, and (ii) on the single chip of LUCI of currently 83, which are limits set by some “line length” of 256 words in the RoCon firmware and in the layout of the patterns. The current maximum is therefore set to 63 per chip if the source code is compiled outside the MPIA, but will not be more than 128 in the future.

The configuration of the number and location of these reset windows is done with GEIRS by modifying the readout pattern files associated with the \texttt{srr(e)} mode in the \texttt{pttrns} subdirectory of the instrument currently in use. It is the operator’s responsibility to

- define the pattern subdirectory that will be used. These are typically names like \texttt{Carmenes-r6}, \texttt{Luci2-r42} and so on combining an instrument name and svn revision number. Because the information of the directory name to be used is actually hidden inside the startup script, and this is not scanned easily, the current procedure demands explicit knowledge of that directory’s name.

\(^{27}\)Actually the raw number is multiplied by \( N \) and the \texttt{BScale} keyword in the associated header is set to \( 1/N \) to compensate for that. This sort of administration improves the resolution of the integer data representation.

\(^{28}\)We plan to drop the first pair for the Fowler-Type of interpretations somewhen in the future for the same reason.
• fill an ASCII file with the srre configuration (windows and auxiliary parameters) prior to
the next call of a read in srre mode if this is different from the previous exposure. The
set of windows in this file replaces any previously defined set of windows; old windows are
forgotten. GEIRS does not remember the previous setup; in practice only the headers of old
FITS files reveal old window sets via the RESWN keywords (Section 6). In that sense the new
file contains a complete set for the next exposures. (There is no interface for an incremental
replacement, deletion or increment of individual windows.)

• transform that ASCII file to five associated pattern files in the aforementioned pttrns direc-
tory with a call to geirs_srreConfig prior to calling the read. Note that the next read in
the srre mode will then trigger an upload of a new pattern to the ROE and therefore need
roughly 10 to 20 seconds (depending on network latencies, number of windows and so on)
before the actual read process starts.

Alternatively, one can append the configuration file name to the argument list of the ctype
srre (after the number of reads) each time it has been changed. This generates the pattern
files and loads them to the ROE. 29

The configuration file looks like a FITS template file and contains lines of the following format:

• WIN[idx] = '[xstrt:xend,ystrt:yend]' A set of 1-based reset window specifications in the stan-
dard FITS syntax with ranges along the horizontal and vertical axis in the user’s standard
view of the images (i.e., including any optional modifications introduced by the CAM_DETROT90
and CAM_DETXYFLIP, Section 3.2). 1-based means that the index of the pixel in the lower left
corner of the coordinate system is at (x, y) = (1, 1), as in FITS. The upper limits of the
number for xend and yend in the coordinates are multiples of 2048, depending on how many
chips are in the detector, and for non-square configurations like CARMENES again depending
on CAM_DETROT90 and CAM_DETXYFLIP. Ill-formatted specifications, like those where the quo-
tation marks are missing or the xend is smaller than xstart or yend is smaller than ystrt or
the entire window is outside the pixel coordinates of the chips, will be silently dropped.

If a window stretches across more than one chip, it will only be recorded for the chip with the
smaller x and y FITS coordinates—which in fact means that for CARMENES a window
definition with xstrt ≤ 2048 and xend ≥ 2049 will define only a window on SCA2.

GEIRS will also reduce the windows to fit into the active 2040 × 2040 inner region of the
chips; reset pixels covering the reference pixels are filtered by the software.

The letters after the WIN (shown as idx above) should be non-negative integer numbers, and
each idx should occur only once (outside comments) in the configuration file. There may be
holes in the index list. (So you might insert a COMMENT in front of the WIN to disable that
window and do not need to edit the indices in the other lines in the configuration file.) You
can fill these indices with zeros for readability: keywords like WIN00, WIN0100 or WIN8 are
alright. Leading zeros in the idx are ignored, so WIN09 and WIN9 refer to the same window
and override each other if they are in the same file.

The numbers of reset windows on the different CARMENES chips may differ. For example
there may be 4 windows WIN01, WIN08, WIN12 and WIN13 on either SCA1 or SCA2, and
for example 6 windows WIN02, WIN03, WIN07, WIN11, WIN10 and WIN20 on the other.

29This additional parameter makes possibly sense for LUCI where resolutions and masks are frequently changed.
For CARMENES this is not supposed to happen because the window locations would change rarely, after earth-quakes
or after exchange of the calibration sources.
The FITS-style comments in the lines are *not* copied to the corresponding FITS header keywords in the images—at least not by GEIRS.

- **DETROT90 = [integer]** The same integer as used inside the startup script to initiate image rotations. If no such line exists in the configuration file, the default is taken from the shell environment variable `CAM_DETROT90` of the user who calls `geirs_srreConfig`. If this is also not set, the default is 1 for CARMENES and 1 for both LUCI’s.

- **DETYXFLI = [integer]** The same integer as used inside the startup script to initiate image rotations. If no such line exists in the configuration file, the default is taken from the shell environment variable `CAM_DETYXFLIP` of the user who calls `geirs_srreConfig`. If this is also not set, the default is 2 for CARMENES and 1 for both LUCI’s.

- **NDET = [integer]** Number of chips in the detector. If such a line is missing, the default is 2 for CARMENES and 1 for both LUCI’s. This keyword supports tests where the software is not run with the full number of boards or chips; for the same reason the `NDET` environment variable may be set in the startup script and selected in the GUI of Figure 3.

- **LINRES = [bool]** If true, the implementation uses line resets for the reset windows. If false, the implementation uses resets in the global window mode. If this is not set (which is recommended), the default is true.

- **KEYWOOD = blabla** Any keyword like this one which is not in the list shown above is ignored.

- **COMMENT blabla** Lines to be ignored and merely serving as comments to the configuration. There may be more than one of these comment lines.

- **# blabbla** Lines starting with the hash are also ignored. This is a lazy version of COMMENT.

- ----blabla Lines started with 8 spaces are also treated as comments.

All lines of these formats may be extended by a slash and further comments, which will be ignored by the parser build into `geirs_srreConfig`.

The keywords in the template header lines are converted to upper case before being checked. The interface is case-insensitive with respect to the keywords. This means for example that `Win81=...`, `wIn81=..` and `WIN81=...` are all specifying the same window; if that type of multiple re-definition happens in the configuration file, the coordinates in the latest lines (down in the file) survive.

The main differences between these FITS template files and real FITS header files are

1. FITS header lines are exactly 80 bytes long, whereas FITS template lines may be longer or shorter

2. FITS header lines are *not* terminated by line feeds or carriage returns, whereas FITS template lines *must* be terminated by line feeds

3. In the template files, the equal sign separating keyword and value is optional.

4. FITS header lines contain mandatory keywords, whereas that category does not exist in the template lines.
Examples of these files with names like `srreMask*` are in the `GEIRS/version/test` subdirectory of the GEIRS distribution.

The syntax of this configuration file is the same as the format of the configuration file of the `sfdump` command to the GEIRS shell (Section 5.3). Both files contain (i) a set of rectangular window geometries in the full-frame coordinate system, (ii) a small set of other keyword-value pairs and (iii) comments. Because the `sfump` and the `geirs_srreConfig` parsers ignore keywords that are not on their individual parameter lists, one may use a single, merged common configuration file at both places if one wishes to reset a set of windows after each `srre` read and to dump exactly the same set of windows after each read for monitoring purposes.

`geirs_srreConfig` is an executable in the Linux binaries, not a command of the GEIRS shell (!). It can actually be used even if GEIRS is not running, and it generally does not know which of the instruments supported by GEIRS (see Section 1.1) will be started by the Linux user. The syntax is

```
geirs_srreConfig -i configfile -p infodir
```

to translate an existing `configfile` to the five pattern files

1. `infodir/multi_win_res_coordinates.instru`,
2. `infodir/multi_win_res_init.instru`,
3. `infodir/multi_win_res_lay1.instru`,
4. `infodir/multi_win_res_lay2.instru`, and
5. `infodir/multi_win_res_pat.instru`

in the directory `infodir`. These five files are replaced/overwritten. *Never call this command before the current readout is finished and GEIRS has written the FITS files.*

Caution: while GEIRS is running there is one active pattern subdirectory selected at startup time—by default the subdirectory with the highest version number (see `CAMROE_REV` in Section 3.2). If the `infodir` parameter provided here is different, you will see no effect on the window coordinates in subsequent readouts, because the pattern files have been updated in a directory which is not used by the active GEIRS session. If GEIRS is actually running, one might ask it for its current pattern directory and feed this into the option:

```
30
pdir=$(geirsCmd version -p)
geirs_srreConfig -i configfile -p ${pdir}
```

If GEIRS is not running, and your environment variables are correctly configured, the current directory is also available via the `geirs_patterns` command, for example

```
pdir=$(geirs_patterns luci1)  
geirs_srreConfig -i configfile -p ${pdir}
```

---

30This option is new since trunk-r752
There is a limit set to the number of windows within the software to ignore windows that would not fit into some layers of the detector FPGA of the ROE. `geirs_srreConfig` ignores the abundant ones (i.e., drops those that are late in the file) and says something like *imposing a ... limit ...* if it does this.

If `configfile` does not start with a slash, the full path name is `$CAMTMP/configfile` if the environment variable `$CAMTMP` is set, otherwise `$TMPDIR/configfile` and then `$TMP/configfile` if either `$TMPDIR` or `$TMP` are set, and eventually just `configfile` (praying that this makes sense relative to the current working directory of the caller).

If `infodir` does not start with a slash, the full path name is `$CAMINFO/infodir` if the environment variable `$CAMINFO` is set, otherwise `$CAMHOME/INFO/infodir` if `$CAMHOME` is set, then `$HOME/GEIRS/INFO/infodir` if `$HOME` is set, and eventually just `infodir` (praying that this makes sense relative to the current working directory of the caller).

The maintenance of the `srre` configuration is quasi static:

1. As seen above, the configuration is represented by an existing set of files in the (active) pattern directory in the computer’s file system. As long as nobody changes these files by either calling `geirs_srreConfig` or running the `ctype srre` command with a file argument or switching to a different version of the pattern directory or editing the files by any other method, the places and size of the reset windows remain frozen. Any read with the `srre` mode uses the windows defined through these pattern files at that time. There are differences regarding which `srre` windows are defined when GEIRS starts up:
   - For GEIRS versions up to 751M-14, one of the test patterns was loaded
   - For GEIRS versions from 751M-18 on, shutting down GEIRS saves the current pattern in the `$CAMTMP` directory and reloads it at the next startup.

   This persistence was introduced when it became obvious that the CARMENES NIR software often did not configure the reset windows before using the `srre` mode.

2. The requirement to change these windows depends on (i) drifts in the optical setup of the instrument that may cause slow wandering of the spectral lines, (ii) on the necessity to subdue different line sets as a function of the different calibration lamps, (iii) modifications of the parameters for rotations and flips at GEIRS startup. All that is definitely not in the scope of the software manual.

3. The reset frequency is tight to the readout frequency and a consequence of the integration time and number of readouts of the ramp. Changing integration times or the number of readouts with the commands send to GEIRS does not require changing these pattern files. [Indeed the configuration file does not have timing parameters.]

5.6.2.5 Example From a driver’s point of view, the scheme is

```bash
# create contents of srre.cfg by any means (shell, other programs,..., support routines)
echo "WIN1 = '[100:100,200:200]'" > $CAMTMP/srre.cfg
echo "WIN2 = '[700:710,200:200]'" >> $CAMTMP/srre.cfg
echo "DETROT90 = 2" >> $CAMTMP/srre.cfg
echo "DETXYFLI = 1" >> $CAMTMP/srre.cfg
...```

and then use either

```plaintext
# update the pattern files in the pattern subdirectory
geirs_srreConfig -i $CAMTMP/srre.cfg -p $CAMINFO/Carmenes_r9
cambuild=$(geirs_build)
geirs_srreConfig -i $CAMTMP/srre.cfg -p $cambuild/pttrns/Carmenes

# start exposure in srre mode
snd_carmenes_new ctype srre 10
snd_carmenes_new read
snd_carmenes_new sync
snd_carmenes_new save
```

or

```plaintext
# configure and start exposure in srre mode
snd_carmenes_new ctype srre 10 $CAMTMP/srre.cfg
snd_carmenes_new read
snd_carmenes_new sync
snd_carmenes_new save
```

5.6.2.6 Programming Model  The following facts should be considered for software that uses GEIRS:

- The ROE keeps a single set of reset window (coordinates) at a time. Switching to another set of windows (with one of the two methods described above) costs typically 15 seconds, because this implies constructing a new pattern and downloading it to the ROE.

- GEIRS optimizes downloading sets of reset windows as follows to minimize the aforementioned overhead:

  - a set of new windows is only downloaded if the source file on the computer has a modification date that is newer than the previous download time. If GEIRS starts up it downloads a default srre pattern — at the same time memorizing that as the previous download time.

  - a set of new windows is only downloaded if the next read is in srre mode.

  - the download is triggered with the read and with the ctype srre command. In particular the read will effectively start later (by the download time) if it observes that the current readout mode is srre and that the file in the operating system is newer than the recent download time.

This conditions are all to be met at the same time to trigger a substitution of the reset window set.

- The ROE skips the actions related to the srre resets if the readmode mode is any other mode, like srr or lir. The ROE does not need to replace the reset windows in that case but has means to skip in a sort of subroutine manner the loop over the window resets. The set of reset window coordinates that resides on the ROE remains there in that case but stays idle/dormant until the readout mode is switched back to srre—which may trigger a replacement (new download) according to the conditions shown above.
In a typical use scenario of a spectroscopic camera, the reset window coordinates rarely need modification. So one needs to trigger a download of the reset window set only once (by replacing the configuration file with a newer file or at least using `touch(1)` to give it the appearance to be newer), and then one can efficiently switch these resets on and off by switching between the `srre` and `srr` modes. In that case there are no intermediate downloads, because the one in the ROE is always up-to-date with the time stamp of the file in the operating system.

If the reset window masks need a change for the next exposure, there is always that penalty of up to 15 seconds.\(^{31}\)

In summary: it is useless and a waste of time to create and to maintain `srre` configuration files with zero windows.

### 5.6.2.7 Support Routines

There is also an option to extract the brightest regions from a FITS image with the syntax

```
```

that reads the FITS image in the file of the \(-f\) option, employs a set of windows each as many pixels wide and high as specified by the \(-w\) and \(-h\) options, and extracts the brightest regions by a count delimited by the \(-N\) option, and dumps the coordinates of these windows to the standard output.

This call expects the image to be in the primary HDU of `fitsfile`; use the recipe of Section A.6.1 to prepare that format from other files.

The idea is that one can create the mask file for the reset windows in that semi-automated way in an environment where prediction of the bright spots is difficult because the optics configuration changes often and in hyperconvex parameter spaces. This primarily aims to deal with variable slit positions for LUCI, but clearly not spectral line positions for CARMENES. The program can also be used for semi-automated location of bright regions in some kind of simple astrometry for the other instruments,\(^{32}\) if the `width` and `height` are chosen to match the typical FWHM of the PSF.

The option \(-v\) increases verbosity and lets the program report also the average ADU’s in the computed subwindows. If the options `width` and `height` are missing, they default to 20. If one of the two width or height options is present and the other absent, the missing value will be set to the existing, resulting in square windows. If the option \(-N\) is missing, a default of 10 is substituted.

The option \(-o\) followed by the name of a FITS file (which must not yet exist, which means you need to remove it beforehand if the intent is to replace it) creates the `fitsofile` with a copy of the image in `fitsfile`, but with the regions of the windows wiped out by setting the values to zero inside the bright regions that are detected. This is basically a debugging option but may also be useful to remove bright regions in FITS images for example in search of ghosts. One may set in addition the \(-r\) flag which reverses/complements the set of pixels in `fitsofile`, which means, `fitsofile` shows only the pixels of `fitsfile` that are inside the bright regions.

The option \(-M\) uses not the integrated flux in rectangular regions but the median to sort them along brightness. This will slow down the calculation tremendously—the `wincnt` needs to be kept

\(^{31}\) the timing depends on the load on the network that connects the workstation with the ROE, the number of reset windows and so on.

\(^{32}\) The disadvantage of the program to that purpose is that this will preferentially flag all the hot pixel regions because no support for bad pixel masks exists in the current version.
small—but has the advantage of sidelining hot or cold pixels to some degree.

Note that the coordinates may be off by factors of 2048 if single-chip images are evaluated in that way and used to configure multi-chip detectors like CARMENES. If a DETSEC specification is found in fitsfile, it will be used to shift the coordinates; DETROT90 and/or DETXYFLI keywords in fitsfile will also be evaluated.

Also note that geirs_srreConfig -f ... just prepares the configuration file. It does not construct the pattern files that act on the forthcoming exposures. Therefore, in practise, a semiautomated application of the reset windows will always call pairs of geirs_srreConfig, the first with -f analysing a previous image, the second with -i and -p installing the new patterns. For CARMENES and for spectroscopy in general, there will at most be a handful of probably pre-selected reset window sets, because the location of bright spots on the detector depends only on a few parameters of the optical setup (the choice of calibration lamps, the option to rotate the entire detector by 180°,...)

In almost all cases the fitsfile will contain a full image, which means, not an image with darkened areas of the data by production with a previous srre. (There may be rare circumstances where deriving the reset window set recursively makes sense, starting with a full image, patching it with a finite cover of reset windows, deriving from that image the bright areas and patching this again...)

On a side note, this way of extracting the brightest pieces of an image could also be used to generate the configuration files of the sfdump command.

This invocation can only scan images in the primary HDU of the fitsfile;\(^{33}\) if the image is in FITS extensions, it may be copied to a temporary file with that format through the ftcopy command of the heatools in the style of

\[\text{ftcopy 'origfile.fits[SCA1_1]' tmp.fits copyall=no}\]

or using the pipFits_zech(1) program with its -P option to merge the images in the extensions into a single image in the primary HDU, for example

\[\text{pipFits_zech -P car_measured.fits carTmp.fits}\]

\[\text{geirs_srreConfig -f carTmp.fits -N 70 -w 20 -h 40 -v > srreMask.cfg}\]

Note that this does not upload any reset windows to anywhere, it just helps to get a first draft of the reset window coordinates into a file (here srreMask.cfg) of the correct format.

5.6.2.8 Disabling As a support for intermediate ROE versions that may not have firmware support of the reset window patterns, GEIRS runs through a set of decisions to consider the srre type supported or unsupported. If supported, the srre appears in the Read Mode submenue in Figure 5.

1. srre is not supported on Hawaii-2 detectors and not supported for PANIC.

2. srre is supported in all other cases unless all of the following is correct

\(^{33}\)this may change in the future
- The file $SCAMTMP/ip-address exists, where the IP-address is the currently agitated readout-electronics.
- There is a line in that file that sets the keyword CANSRRE to the value F. Note that this uses the FITS syntax for boolean values; in particular the F is not enclosed with quote marks.

5.6.2.9 Common SRRE Errors  The most common errors using the srre mode encountered while GEIRS is used by external software or human operators are:

1. The syntax of the configuration file is wrong, for example the quotation marks are missing in the coordinate specification.

2. Warnings and errors reported by geirs_srreConfig are ignored.

3. geirs_srreConfig writes files that are read-protected from the account that runs GEIRS.

4. The files in the INFO directory are write-protected from the account that runs GEIRS, so ctype srre cannot update them.

5. The pattern directory specified through the -p option of the geirs_srreConfig is not the one used by the current GEIRS version. The result of such error is that GEIRS will not register that the configuration changed and will keep the old one.

6. The driver does not wait for the reply after the subsequent call of ctype srre or read. That means the driver ignores that ctype or read—whichever comes next—will need typically 10 to 20 seconds to complete. See Section 5.2.2.

7. geirs_srreConfig is called while read is executing, trying to reconfigure GEIRS during an exposure.

8. Too many windows are configured and the errors from the next ctype srre or read concerning the unavailable FPGA registers are ignored.

9. There is a misconception that calling geirs_srreConfig ensures proper configuration. In fact geirs_srreConfig just prepares some files on the Linux workstation; configuration of the ROE happens later during the next read or ctype srre.

10. The subwin command is used in conjunction with ctype srre. This is not supported with the current GEIRS version.

5.7 Tutorial

Basically GEIRS is commanded by a base set of about 10 commands: the read-save pairs and parameters that define integration time, number of repetitions of the readout cycle and the place of the FITS files.
5.7.1 read, sync, save

If GEIRS has just been started up, some default values for the readout mode, integration time, output directory and FITS file name have already been set up. Here is the probably shortest command sequence to generate a single FITS file, which reads out the detector once if no crep as used earlier, waits until the frame data have arrived on the workstation, and saves the data (i.e., creates the FITS file):

```plaintext
read
sync
save
```

5.7.2 itime, ctype

The basic properties of the exposures are the integration time set with the `itime` and the readout mode (cycle type) set with the `ctype` command prior to one or more reads. The parameters do generally only need to be re-send if they should change; GEIRS remembers the current parameter set and applies it until parameters are modified. An exposure with a single-frame-read of 5 seconds (which is not saved) followed by an exposure of 5 seconds in the line-interlaced-read mode—which is saved in a FITS file—and then an exposure of 10 seconds in the sample-up-the-ramp mode with the default of 2 reads—which is saved in a FITS file—are induced by

```plaintext
ctype sfr
itime 5
read
sync
ctype lir
read
sync
save
sync
itime 10
ctype srr
read
sync
save
sync
```

5.7.3 crep, set savepath, next

The cycle repetition `crep` parameter triggers that the subsequent read commands are not creating a single image by reading the detector once (the default) but do this as often as the parameter says. The save path is the directory where new FITS files are created, and the `next` specifies a base name for creating indexed FITS files in the future.

The following sets the read mode to fowler pairs with 4 frames combined into a single image. The integration time (time between associated frames) is set to 5 seconds, and these quad-frames are
read 6 times. The resulting 6 images are stored in the files /dataA/2015-04-01/hah_0001.fits to /dataA/2015-04-01/hah_0006.fits (if the directory exists or permissions allow to generate the directory):

```
cctype mer 4
itime 5
crep 6
read
sync
set savepath /dataA/2015-04-01
next hah_
save
sync
```

### 5.7.4 save multiple times, sample-up-the-ramp

The srr mode is used with an argument which sets the number of reads along a non-destructive read. The integration time which is set independently then defines implicitly the duration between two reads. In infrared astronomy, usually all frames along the time axis are also saved (for a later independent correction for nonlinearities, dark currents and so on). A total integration time of 60 seconds with 13 reads (therefore 60/12 = 5 seconds between each read pair) saved into a file srr60_0001.fits with the linearly fitted image and the single frames saved into srr60_0002.fits up to srr60_0014.fits is executed by the sequence

```
cctype srr 13
itime 60
read
sync
next srr60_
save
sync
save -S
sync
```

The explicit -S will not be needed by future GEIRS versions, because by default a sfdump command will be activated which always saves all individual CARMENES frames in preparation of the pipeline. If the srr 13 at the start of this sequence is replaced by srre 13, the currently active set of reset windows is applied to each of the frames. So the 12 of the frames in srr60_0003.fits to srr60_0014.fits have visible dark patterns where the reset windows have been placed. The first frame, srr60_0002.fits, is the one read immediately after the initial reset and does not have such an imprinted pattern.
6 FITS OUTPUT

6.1 Illustrative Example

The primary FITS header generated by the stand-alone GEIRS is illustrated by the following example (extracted with dfits):

```
SIMPLE = T
BITPIX = 16
NAXIS = 2
NAXIS1 = 2048
NAXIS2 = 2048
EXTEND = T / FITS dataset may contain extensions
COMMENT FITS (Flexible Image Transport System) format is defined in 'Astronomy and Astrophysics', volume 376, page 359; bibcode: 2001A&A...376..359H
BSSCALE = 1.
BZERO = 32768. / [adu] real = bzero + bscale*value
BUNIT = 'adu ' / [adu]
MJD-OBS = 56610.398151 / [d] Modified julian date (TT) of DATE-OBS
DATE-OBS = '2013-11-14T09:33:20.2482' / [d] UTC date of end of first frame read
DATE = '2013-11-14T09:40:59.0409' / [d] UT-date of file creation
UT = 34400.248236 / [s] 09:33:20.2482 UTC at EOread
LST = 46667.9276 / [s] local sidereal time: 12:57:47.928 (EOread)
ORIGIN = 'Centro Astronomic Hispano Aleman (CAHA)'
OBSERVER= 'master '
TELESCOP= 'CA-2.2 '
FRATIO = 'F/08 ' / [1]
OBIGEO-B= 37.223037 / [deg] telescope geograph. latit.
OBIGEO-H= 2168. / [m] above sea level
LAMPSTS = ' ' / calib. lamp
INSTRUME= 'Panic '
CAMERA = 'HgCdTe (4096x4096) IR-Camera (4 H2RGs)'
PIXSCALE= 0.45 / [arcsec/px]
ELECGAIN= 2.01 / [ct] electrons/DN
ENOISE = 12. / [ct] electrons/read
ROVER = 'MPIA IR-ROelectronic Vers. 3' / Version det. electronics
WPOS = 5 / [ct] number of GEIRS wheels
W1POS = 'COLDSTOP22'
W2POS = 'KS '
W3POS = 'OPEN '
W4POS = 'OPEN '
W5POS = 'OPEN '
FILTER = 'NO ' / filter macro name of filter combinations
FILTERS = 'OPEN ' / combination of all filters used (single OPEN)
STRT_INT= 33398.779494 / [s] '09:16:38.7795' start integration (UT)
STOP_INT= 34400.185113 / [s] '09:33:20.1851' stop integration (UT)
DEC = 42.714232 / [deg] Dec.: +42:42:51
EQUINOX = 2000. / [a] Julian Epoch
OBSEPOCH= 2013.866673 / [a] Julian Epoch
AIRMASS = 1.051181 / [1] airmass
HA = 337.594738 / [deg] H.A. '22:30:22.74'
```
T_FOCUS = 30. / [mm] telescope focus
CASSPOS = 0. / [deg] cassegrain position rel. to NSEW
OBJECT = 'no object'
FILENAME = 'Illum_srr30_300s_0214.fits'
DITH_NO = 0 / [ct] dither step
EXPO_NO = 235 / [ct] exposure/read counter
TPLNAME = ' ' / macro/template name
TIMER0 = 67145 / [ms]
TIMER1 = 932855 / [ms]
TIMER2 = 865710408 / [us]
PTIME = 2 / pixel-time-base index
PREAD = 10000 / [ns] pixel read selection
PSKIP = 150 / [ns] pixel skip selection
LSKIP = 150 / [ns] line skip selection
READMODE = 'sample.ramp.read' / read cycle-type
IDLEMODE = 'break ' / idle to read transition
IDLETYPEx = 'ReadWoConv' / idle cycle-type
SAVEMODE = 'o2.single.corr.read' / save cycle-type
CPAR1 = 50 / cycle type parameter
ITIME = 1000. / [s] (on chip) integration time
CTIME = 1001.370302 / [s] read-mode cycle time
CRATE = 0.000999 / [Hz] read-mode cycle rate
EMSAMP = 1 / [ct] electronic multi-sampling
FRAMENUM = 47 / of 50 saved
NCOADDS = 1 / [ct] # of software coadds
DETSEC = '[1:2048,2049:4096]' / [px] xrange and yrange of window
DATASEC = '[5:2044,5:2044]' / [px] xrange and yrange of science data
FRAMENUM = 1 / of 1 saved
SKYFRAME = 'unknown '
DETSIZE = '[1:4096,1:4096]' / [px] x-range, y-range of full frame
CHIPSIZX = 2048 / [px] single chip pixels in x
CHIPSIZY = 2048 / [px] single chip pixels in y
B_EXT1 = 2.299805 / [V] external bias 2355
B_EXT2 = 2.685547 / [V] external bias 2750
B_EXT3 = 2.685547 / [V] external bias 2750
B_EXT4 = 2.685547 / [V] external bias 2750
B_DSUB1 = 1.199755 / [V] det. bias voltage DSUB 2614
B_DSUB2 = 1.744141 / [V] det. bias voltage DSUB 3800
B_DSUB3 = 1.744141 / [V] det. bias voltage DSUB 3800
B_DSUB4 = 1.744141 / [V] det. bias voltage DSUB 3800
B_VREST1 = 0.699951 / [V] det. bias voltage VRESET 1525
B_VREST2 = 1.193359 / [V] det. bias voltage VRESET 2600
B_VREST3 = 1.193359 / [V] det. bias voltage VRESET 2600
B_VREST4 = 1.193359 / [V] det. bias voltage VRESET 2600
B_VBIAG1 = 2.199707 / [V] det. bias voltage VBIASGATE 3604
B_VBIAG2 = 2.199707 / [V] det. bias voltage VBIASGATE 3604
B_VBIAG3 = 2.199707 / [V] det. bias voltage VBIASGATE 3604
B_VBIAG4 = 2.199707 / [V] det. bias voltage VBIASGATE 3604
B_VNBIA1 = 0. / [V] det. bias voltage VNBIA 0
B_VNBIA2 = 0. / [V] det. bias voltage VNBIA 0
B_VNBIA3 = 0. / [V] det. bias voltage VNBIA 0
B_VNBIA4 = 0. / [V] det. bias voltage VNBIA 0
B_VPBIAS1 = 0. / [V] det. bias voltage VPBIAS 0
B_VPBIAS2 = 0. / [V] det. bias voltage VPBIAS 0
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_VPBIA3=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VPBIAS 0</td>
</tr>
<tr>
<td>B_VPBIA4=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VPBIAS 0</td>
</tr>
<tr>
<td>B_VNCAS1=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VNCASC 0</td>
</tr>
<tr>
<td>B_VNCAS2=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VNCASC 0</td>
</tr>
<tr>
<td>B_VNCAS3=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VNCASC 0</td>
</tr>
<tr>
<td>B_VNCAS4=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VNCASC 0</td>
</tr>
<tr>
<td>B_VPCAS1=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VPCASC 0</td>
</tr>
<tr>
<td>B_VPCAS2=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VPCASC 0</td>
</tr>
<tr>
<td>B_VPCAS3=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VPCASC 0</td>
</tr>
<tr>
<td>B_VPCAS4=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VPCASC 0</td>
</tr>
<tr>
<td>B_VBOUB1=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VBIASOUTBUF 0</td>
</tr>
<tr>
<td>B_VBOUB2=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VBIASOUTBUF 0</td>
</tr>
<tr>
<td>B_VBOUB3=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VBIASOUTBUF 0</td>
</tr>
<tr>
<td>B_VBOUB4=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage VBIASOUTBUF 0</td>
</tr>
<tr>
<td>B_REFSA1=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage REFSAMPLE 0</td>
</tr>
<tr>
<td>B_REFSA2=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage REFSAMPLE 0</td>
</tr>
<tr>
<td>B_REFSA3=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage REFSAMPLE 0</td>
</tr>
<tr>
<td>B_REFSA4=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage REFSAMPLE 0</td>
</tr>
<tr>
<td>B_REFCB1=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage REFCOLBUF 0</td>
</tr>
<tr>
<td>B_REFCB2=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage REFCOLBUF 0</td>
</tr>
<tr>
<td>B_REFCB3=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage REFCOLBUF 0</td>
</tr>
<tr>
<td>B_REFCB4=</td>
<td>0.</td>
<td>[V]</td>
<td>det. bias voltage REFCOLBUF 0</td>
</tr>
<tr>
<td>TEMP_A</td>
<td>-9999.</td>
<td>[K]</td>
<td>sensor A (-10272.15 C)</td>
</tr>
<tr>
<td>TEMP_B</td>
<td>-9999.</td>
<td>[K]</td>
<td>sensor B (-10272 C)</td>
</tr>
<tr>
<td>PRESS1</td>
<td>0.000372</td>
<td>[Pa]</td>
<td>(3.720e-09 bar)</td>
</tr>
<tr>
<td>TEMPMON</td>
<td>8</td>
<td>[ct]</td>
<td># of temp. 2013-11-14 09:00 monitor loc. t</td>
</tr>
<tr>
<td>TEMPMON1</td>
<td>73.740997</td>
<td>[K]</td>
<td>(-199.41 C) 2013-11-14 10:32 Sensor 1</td>
</tr>
<tr>
<td>TEMPMON2</td>
<td>74.575996</td>
<td>[K]</td>
<td>(-198.57 C) 2013-11-14 10:32 Sensor 2</td>
</tr>
<tr>
<td>TEMPMON3</td>
<td>74.069</td>
<td>[K]</td>
<td>(-199.08 C) 2013-11-14 10:32 Sensor 3</td>
</tr>
<tr>
<td>TEMPMON4</td>
<td>73.061996</td>
<td>[K]</td>
<td>(-200.09 C) 2013-11-14 10:32 Sensor 4</td>
</tr>
<tr>
<td>TEMPMON6</td>
<td>76.603996</td>
<td>[K]</td>
<td>(-196.55 C) 2013-11-14 10:32 Sensor 6</td>
</tr>
<tr>
<td>TEMPMON7</td>
<td>86.221001</td>
<td>[K]</td>
<td>(-186.93 C) 2013-11-14 10:32 Sensor 7</td>
</tr>
<tr>
<td>TEMPMON8</td>
<td>123.300003</td>
<td>[K]</td>
<td>(-149.85 C) 2013-11-14 10:32 Sensor 8</td>
</tr>
</tbody>
</table>

This is generated by running PANIC, because the number of keywords is roughly a maximum for this instrument. The outcome is different for other instruments.

GEIRS generates FITS images with 2 bytes per pixel when storing single frame data (created either through some single-frame read cycle type or by using the -S switch of the save command or from the single frame dumps of the guide mode), and images with 4 bytes per pixel for all the others (created by correlated cycle types). So the simplest filter for fishing for FITS files with correlated images in the local directories of CARMENES—assuming no data cubes were stored—is to select FITS files larger than 30 MB, for example:

```bash
find . -name "*.fits" -size +30M
```

because the single full frame files are slightly larger than 16 MB and the correlated full frame files are slightly larger than 33 MB.
To check compliance of FITS files on the computer with the standards call `ftverify` of the `heatools` (Section A.6.4). This is equivalent to the `fitsverify` online tool of HEASARC.

### 6.2 Online Keyword Modification

*Section 6.2 is irrelevant for CARMENES because there are no auxiliary FITS data on the NIR computer.*

#### 6.2.1 PANIC Temperatures and Pressures

*This section is only relevant for PANIC.* The values of the keywords `TEMPPMONi` (where `i` is a small number with offset 1), and of `PRESSi` are acquired by GEIRS by parsing the last line of the current contents of the text file `GEIRS/log/instrtemp.log`. This must contain twelve blank-separated values as follows: (i) A date in the YYYY-MM-DD format, (ii) a time in the HH:MM format, (iii) two pressures in scientific/ floating point format units of mbar—which is the same as hPa—, (iv) eight temperatures in floating point format in units of K. The file `caha/INSTALL.CAHA` in the source code addresses how this may be automated. By convention (and implementation), the time stamps are in the local time zone of the computer on which GEIRS is run. The pressure unit is changed to Pascal (applying a factor of 100) within GEIRS while transferring the pressures to the FITS header. For the test cryostats at the MPIA, this can be achieved as described in `caha/INSTALL.MPIA` in the source code.

#### 6.2.2 File-based Subscriptions

Supervisor software can funnel primary header keyword lines into the new FITS files by writing them into the `$CAMTMP/geirsPhduAdd.instrument` or `$CAMTMP/geirsPhduAdd.instrument_i` file before the FITS file is generated with the `save` command.

(For the two LUCI branches the `instrument` is already ending on either 1 or 2. This number is not the index `i` used here.)

Here `i` is a small integer from 1 to 6. The effective line set is the concatenation of the lines in these files in the natural order, as if first `geirsPhduAdd.instrument`, then `geirsPhduAdd.instrument_1`, etc and finally `geirsPhduAdd.instrument_6` was acting on the raw default FITS headers. Having a range of six files at the disposal allows multiple subsystems to update or erase these files with different frequencies.

#### 6.2.2.1 PANIC convention

The current convention is that

1. `$CAMTMP/geirsPhduAdd.instrument` is manipulated by online tools, [18]
2. `$CAMTMP/geirsPhduAdd.instrument_1` is reserved for the `lamp.sh` output (Section ??),
3. `$CAMTMP/geirsPhduAdd.panic_2` for any further generic cleanup within GEIRS;
4. `$CAMTMP/geirsPhduAdd.panic_3` is reserved for infrequent long-term logging and event data [19].
$\text{CAMTMP/geirsPhduAdd.panic}_5$ is reserved for logging of meteo data with \texttt{geirs\_ambiPhdu.sh}.

### 6.2.2.2 Linc-Nirvana convention

1. $\text{CAMTMP/geirsPhduAdd.nirvana}_1$ is used to adapt the GEIRS conventions to some quasi-conventions of the LBTO;

2. $\text{CAMTMP/geirsPhduAdd.nirvana}_2$ are keywords collected by the Python BASDA script on lircs;

3. $\text{CAMTMP/geirsPhduAdd.nirvana}_3$ is constructed by the initialization window filled in at startup time, Figure 3.

4. $\text{CAMTMP/geirsPhduAdd.nirvana}_4$ contains IIF keywords. It contains snapshots of a subset of the values of the IIF dictionary and some entries of the \texttt{config/ltcs/lts.iif-dev.cfg} configuration file with telescope operator names and the like. The file is created by the \texttt{iif} instance—which usually runs on \texttt{lsys.linc}—, which polls the keywords controlled by the configuration \texttt{lnsw/config/lbcs/lbcs.iif-fits.xml}.

5. $\text{CAMTMP/geirsPhduAdd.nirvana}_5$ duplicates RA and DEC as equivalent WCS keywords.

### 6.2.2.3 LUCI convention

1. $\text{CAMTMP/geirsPhduAdd.luci2}_1$ and $\text{CAMTMP/geirsPhduAdd.luci1}_1$ clean up superfluous GEIRS keywords and boost them to hierarchical keywords, and

2. $\text{CAMTMP/geirsPhduAdd.luci2}_2$ and $\text{CAMTMP/geirsPhduAdd.luci1}_2$ are manipulated by online tools to add the telescope, motor, temperature and optics configuration. These two files are linked to \texttt{fitsheader\_lucifer.txt} for backward compatibility of older configurations at GEIRS startup time.

### 6.2.2.4 CARMENES convention

$\text{CAMTMP/geirsPhduAdd.carmenes}_5$ is used by GEIRS to push its keywords to the hierarchical format and to clean up keywords related to instrument control software.

### 6.2.2.5 NTE convention

$\text{CAMTMP/geirsPhduAdd.instrument}_1$ is used by GEIRS to push its keywords to the hierarchical format and to clean up keywords related to instrument control software.

### 6.2.2.6 Timing

In general GEIRS cleans up these files each time it is started up, because some online tools forget to erase their associated files when they are shut down; this would leave obsolete contents in these files if GEIRS is afterwards started as a standalone program which then erroneously pile up in FITS headers.

For PANIC, however, the files $\text{CAMTMP/geirsPhduAdd.panic}_3$, $\text{CAMTMP/geirsPhduAdd.panic}_4$, and $\text{CAMTMP/geirsPhduAdd.panic}_5$ are preserved during startup.
This mechanism is not synchronized; GEIRS reads the contents of the geirsPhduAdd configurations and edits the FITS header according to their instructions just before composing the FITS file. Obviously there is some risk of losing information if the frame rate exceeds 1 Hz and the supervisor software updates that geirsPhduAdd file at a similar frequency.

The functionality with the fedithead syntax (see Section 5.5) is available: The geirsPhduAdd files can remove, replace and add keywords of the forthcoming FITS header all in one step. A set of proposals for such configuration files on a per-instrument basis is in $CAMHOME/branch/admin/geirsPhduAdd.* in the source code. A use case example for LUCI is the keyword PIXSCALE that is in the standard list of GEIRS header keywords (Section 6.4). Because GEIRS never knows the position of the camera wheel, it cannot fill in that value reliably; consequently the geirsPhduAdd.luci1_1 and geirsPhduAdd.luci2_1 contain a line that deletes the PIXSCALE keyword—and leave it to any of the other geirsPhduAdd files to refill the keyword and value.

To add flexibility, GEIRS calls the script scripts/QueueAFiles before assembling the FITS files (if it exists). This script is “triggered” at the start of each save command. This is currently used for LN to create $CAMTMP/geirsPhduAdd.nirvana_2. (The mechanism assumes that at GEIRS startup a basdard has been activated which writes (unsupervised, asynchronously) temperature and pressure data from the various cabinets into a file set in $CAMTMP.)

6.2.3 fits Command

The keywords added with the fits command of Section 5.3 are piped through the same file-based mechanism by using the file $CAMTMP/geirsPhduAdd.instrument_6. So that file with index 6 is reserved for that purpose and not to be used by any other software. It is effectively rewritten just before each save. Because it is the last in the list, these keywords are not rewritten with the methods in the files with the smaller indices. (This does not harm because the individual instruments are supposed to know which local conventions are in place when using the fits command.)

6.3 Optional Cleanup

The most important aspect of this list of keywords is that, although GEIRS has no information on the telescope pointing and status in the LBT, CARMENES, AIP or NTE environment, it has inserted information on the primary star coordinates (RA, DEC) and a set of derived information, including ALT, AZ, PARANG, HA, AIRMASS, and OBJECT. This behavior is actually triggered by starting the software setting the telescope control system to the offline mode (Figure 3). All of this information is a consistent but randomly simulated and invalid set of data and needs to be removed/replaced by a software that has this information. For LN, this layer of the software would use the IIF of the TCS.

A second set of telescope and optics related variables which is not useful in the LBT context consists of WPOS, FILTER, FILTERS, T_FOCUS, TEMP_A, TEMP_B, PRESS1, PRESS2, all of which stem from CAHA methods and all of which should also be deleted unless the instrument is PANIC.

Another set of potentially useless data are the detector voltages set to 9999 V which are templates created for cameras with Hawaii2-RG or Hawaii-4RG detectors, and also to be removed/ignored for the LN case.

Another task is to translate the remaining keywords to match any particular FITS dictionary
applicable to the instrument or observatory.

This cleanup and translation is typically done by putting the keywords to be deleted and to be translated into a configuration file and calling a translator like `fedithead` in some sort of pipeline stage. This may be customized by calling the translator from within the shell script `$CAMHOME/scripts/QueueFiles` that is called by GEIRS for each new FITS file that is created. If the keyword-value pairs are already known at the end of the exposure, the method of Section 6.2 is also applicable (and more efficient) to modify the primary header keywords.

### 6.4 GEIRS Core Keywords

Some keywords remain after the purging mentioned above; there are FITS mandatory keywords concerning the image dimensions and bits-per-pixel format [15], plus the following:

- **MJD-OBS = 56433.495665 / [d]** Modified julian date (TT) of DATE-OBS
  
  This time refers to the same time as the DATE-OBS. For CAHA instruments it is converted from the UTC to terrestrial time (TT); for LBT instruments remains a UTC time because another keyword is used which flags that times are in UTC [20]. Accuracy of this value depends on running a reasonably recent GEIRS such that the leap seconds are known in the (external) SOFA library.

  Note that the CARMENES ICS overwrites this keyword such that it gets a different meaning unrelated to GEIRS.

- **DATE-OBS = ‘2013-05-21T11:53:45.4834’ / [d]** UTC date of end of first frame read
  
  This rephrases `STRT_INT` and is a close approximation to the start of integration.

- **DATE = ‘2013-05-21T11:54:17.5317’ / [d]** UT-date of file creation
  
  DATE is just mentioned for completion. Following the FITS standards, this time stamp will be updated and overwritten each time some other layer of the software modifies the images or keywords, so it has essentially no significance to astrometric data reduction.

- **UT = 42825.483405 / [s]** 11:53:45.4834 UTC end of first frame read

- **LST = 73883.640000 / [s]** local sidereal time: 20:31:23.640 (EOread)

  The value of the local sidereal time is to be considered an estimate based on the observatory coordinates at the end of the readout. Effects of nutation and so on are completely neglected [21, 22].

  For LN, the keywords UTC, LST, HA and so on are just copies from the TCS polled by the LTCS subsystem at times that are not correlated with the GEIRS exposures; this explains jitters between their time scales and internal sky-related data emitted by GEIRS.

- **ORIGIN = ‘Mount Graham, MGIO, Arizona’**

  - **TELESCOP= ‘LBT’**
  - **FRATIO = ‘F/15’ / [1]**
  - **OBSGEO-L= -109.889000 / [deg] telescope geograph. longit.**
  - **OBSGEO-B= 32.701300 / [deg] telescope geograph. latit.**
  - **OBSGEO-H= 3221.000000 / [m] above sea level**
OBSCOD = 'G39' / Minor Planet Center Observatory code

These keywords related to the name and location of the observatory are hardcoded in the software. The OBSGEO keywords comply with the proposal on WCS coordinates [23]. Three additional keywords OBSGEO-X, OBSGEO-Y, and OBSGEO-Z will be created if the preprocessor variable GEIRS_FITS_OBSGEOKW is defined at compile time. This is switched off by default.

- **OBSERVER= 'mathar'**
  This is equivalent to the most recent observer command received by GEIRS (Section 5.3) or submitted with the start-up GUI, Figure 3.

- **INSTRUME= 'Nirvana'**
  **CAMERA = 'HgCdTe (2048x2048) IR-Camera'**
  **OPTIC = 'very high res.'**
  **PIXSCALE= 0.005110 / [arcsec/px]**

  These keywords are constants hardcoded in the software.

- **EGAIN= 2.010000 / [ct] electrons/DN**
  **ENOISE = 14.000000 / [ct] electrons/read**

  Electronic gain and noise are hardcoded constants. This noise generally refers to the lir read mode [24]. For PANIC’s rrr-mpia mode however, a separate set of these 2 parameters for each of the 4 chips has been measured, so these 8 parameters are copied into the header cards when PANIC is in fact using that readout mode. The noise in the actual FITS images is a function of (amongst others) the readout modes, electronic sampling etc as surveyed in [25]. For instruments with more than one detector chip, both keywords are adorned with 1-based integers: EGAIN1, EGAIN2 and so on.

  For LN these keywords are those of the original Hawaii-2 detector, not the ones of the previous LUCI detector that was installed during COM-6. (That detector has never been calibrated with the suite of detector voltages that are used...)

- **ROVER = 'MPIA IR-ROelectronic Vers. 3' / Version det. electronics**
  A (rough) characterization of the MPIA readout electronics. The FPGA program versions are not reported in the header.

- **STRT_INT= 42822.774880 / [s] '11:53:42.7749' start integration (UT)**
  **STOP_INT= 42825.483222 / [s] '11:53:45.4832' stop integration (UT)**

  These two UTC time stamps are the most accurate timing information available for astrometry in any follow-up pipeline. STRT_INT measures time when the first frame has arrived on the workstation, and is very close to when reading the first frame was completed on the ROE, see Section 7.7. The STOP_INT is slightly earlier than the end-of-read time stamp in UT.

- **EQUINOX = 2000. / [a]**
  Julian year of the RA and DEC information and of the data acquisition.

  Note that the precision of $1 \times 10^{-6}$ years in the numerical value of a year is only equivalent to $\approx 30$ seconds.

- **POINT_NO= 0 / [ct] pointing counter**
  **DITH_NO = 0 / [ct] dither step**
  **EXPO_NO = 1 / [ct] exposure/read counter**
The three numbers are modified by the `counter` command (Section 5.3). The intent of the `POINT_N0` and `DITH_N0` variables is to keep track of dithered (nodding) imaging with imaging optics. It is entirely up to the software/operator that drives GEIRS whether these two may differ from zero.

The regular update of `EXPO_N0` if not intervened by such commands is to start at one as GEIRS is started, then to increase by one for each `read`—where it does not matter if the FITS file name is changed in between. If the cycle repetition factor is chosen larger than one (Repeat in Figure 5 or command `crep` in Section 5.3), the `EXPO_N0` is the same in all the individual files that are created.

- **FILENAME= `normal0003.fits`**
  The filename of the FITS file in the local file system of the detector workstation as requested by the observer.

If the source file `geirs_save.cxx` is compiled with the preprocessor option `GEIRS_CREA_SAVE_LINK` defined, a link from the file given by `FILENAME` to a file with canonical name derived from `STRT_INT` is created at run time. This may facilitate robotic archival software and even be a trivial form of overwrite protection, but has been disabled by default because—in the eyes of the principal GEIRS developer—links may confuse operators with little knowledge of UNIX-type operating systems.

- **TPLNAME = ' ' / macro/template name**
  Name of the macro file (Section 5.4) if applicable. Empty if the observation was driven on a command-by-command basis.

- **TIMER0 = 2667 / [ms]**
  **TIMER1 = 2667 / [ms]**
  **TIMER2 = 0 / [us]**
  Three time intervals that help debugging the GEIRS timing.

- **PTIME = 1 / pixel-time-base index**
  **PREAD = 10000 / [ns] pixel read selection**
  **PSKIP = 150 / [ns] pixel skip selection**
  **LSKIP = 150 / [ns] line skip selection**
  Four parameters that detail in which way the fundamental clock of the ROE was subdivided to drive some basic actions on the detector chip.

- **READMODE= 'line.interlaced.read' / read cycle-type**
  **IDLEMODE= 'wait' / idle to read transition**
  **IDLETYPE= 'ReadWoConv' / idle to read transition**
  **SAVEMODE= 'line.interlaced.read' / save cycle-type**
  These four parameters define the reset-read pattern of gathering the frames, how the read-out electronics clocks the detector while no data are taken, and in which way the frames send from the ROE are packed into FITS images (by averaging, subtracting, fitting...) by GEIRS. See [9, 25].

The **READMODE** defines the scheme of patterns and timings in use while the frames were generated by the detector and ROE and arrived on the workstation. The value of **SAVEMODE** may be different if the mode was changed (either via the button labeled **Read Mode** in Figure 5...
or with the \texttt{ctype} command or by using the \texttt{-S} option of the \texttt{save} command before executing \texttt{save}. In this case the packaging of frames into files of FITS images (by subtraction, averaging...) is modified by the \texttt{save} procedure and departs from the “standard” associated with the read mode. [The software allows to save the same set of frames more than once and switching the mode without any intermediate \texttt{read}. This is helpful if one wants to store correlated images but also the bare frames for debugging purposes.]

- \textbf{CPAR1} = \texttt{1 / cycle type parameter}
  This is the integer parameter given to the \texttt{ctype} command (Section 5.3), basically the number of frames that are correlated in the multi-correlated modes (Fowler, sample up the ramp...) \[26, 27\]. The value is actually a filtered version of the command in case that the associated \texttt{save}-mode does not support a variable parameter.

If the integration along the ramp was disrupted with the \texttt{abort} command, the value is still the one that was scheduled when the \texttt{read} started, not the (smaller) number of frames that were actually read.

- \textbf{ITIME} = \texttt{2.667059 / [s] (on chip) integration time}
  The scheduled integration time. The actual integration time may have been shorter if the exposure was aborted (see \texttt{EXPTIME}). If the \texttt{read} obtained more than one image (as set by the \texttt{crep} command), the integration time is still the integration time of the individual readout, not the accumulated sum over all exposures triggered by that \texttt{read}.

For multiple-endpoint readout modes, the integration time is the time between each correlated pair. The actual time between the first and the last frame is longer by a time proportional to the number of pairs (see \texttt{CPAR1})—but this is obviously not relevant to the photometry.

For sample-up-the-ramp modes the integration time is the time difference between the readout of the first and the readout of the last frame.

- \textbf{CTIME} = \texttt{5.345815 / [s] read-mode cycle time}
  The cycle time is the shortest time between starting repeated exposures. This is longer than the integration time because all relevant readout modes read the detector line-by-line, and that time appears as an overhead to be added to the integration time. So the cycle time is not relevant for photometric interpretation of the images, but an indicator of how much time is “lost” due to incomplete overlap between line resets and reads. The value is a function of readout mode and integration time, and therefore not an input in some operator’s menu or command.

- \textbf{CRATE} = \texttt{0.187062 / [Hz] read-mode cycle rate}
  The value is basically superfluous because it just shows the inverse of the cycle time.

- \textbf{EMSAMP} = \texttt{0 / [ct] electronic multi-sampling}
  The electronic multi-sampling correlated with the \texttt{roe} command (Section 5.3). Values of 0 or 1 mean sampling once with the ADCs, otherwise the value may be 2 or 4 with the benefit of noise reduction \[4, \S 2.5.4\]

- \textbf{NCOADDS} = \texttt{1 / [ct] effective coadds (total)}
  Software coadding is selected by the option \texttt{-i} of the \texttt{save} command (Section 5.3) and indicates how many frames have been added to generate one image.
• **EXPTIME** = 2.667059 / [s] total integ. time

The exposure time spent creating an image. The total time that was spent integrating the flux that defined the value of an individual pixel of the FITS file. Usually this equals the integration time. If the data have been created using a repetition factor larger than one (command `crep` and keyword `NEXP`), **EXPTIME** still is the time for the single image, in case of saving the images in a FITS cube the time for each individual slice in the cube.

If the data have been saved with the `-i` option of the `save` command, **EXPTIME** is the product of `NEXP` and `ITIME`, because each pixel in the image represents the arithmetic sum of the pixels in the individual exposures. To calculate the mean contribution of each exposure to the image then, one must divide **EXPTIME** and each pixel value through `NCOADDS`.

If the exposure was aborted, **ITIME** is the scheduled integration time, but **EXPTIME** the (shorter) exposure time derived from the arrival time of the frames on the GEIRS computer.

For multi-correlated modes **EXPTIME** is still the exposure time that went into the pixel, not any sort of difference between the non-destructive reads.

If GEIRS has dropped one or more initial frames to improve the image quality in multi-correlated modes (Section 5.6.2.3), the **EXPTIME** is still the time that went effectively into the pixel values.

Note that GEIRS may use non-integral `BSCALE` values in FITS image headers.

• **FRAMENUM=** 1 / OF 1 as save range

1-based enumeration of the images or of the frames (if single frames are stored). For images this is only relevant if the `Repeat` option was used to generate a series exposures with a constant set of parameters (`Repeat` entry in Figure 5 and `crep` in Section 5.3).

• **FRAME=**

The 1-based enumeration of the frame in FITS files that were created with the single-frame-dump method of Section 6.7.

• **SKYFRAME= ’(tmp-img)’**

Generally an empty string, but a file name if some other FITS image has been subtracted to obtain the current FITS image, and a string in parentheses if this image was taken from another frame in the online image buffer.

• **DETSEC = ’[1:2048,1:2048]’ / [px] xrange and yrange of window**

Coordinates of the detector window in the FITS image. The value is the same as **DETSIZE** if the full window has been read out.

• **DATASEC = ’[5:2044,5:2044]’ / [px] xrange and yrange of science data**

Coordinates of the detector window in the FITS image. This is basically the same as **DETSEC** but smaller for the case of Hawaii-2 RG detectors if some pixels fall into the 4-pixels frame along the edges.

• **DETSIZE = ’[1:2048,1:2048]’ / [px] x-range, yrange of full frame**

• **CHIPSIZX=** 2048 / [px] single chip pixels in x

• **CHIPSIZY=** 2048 / [px] single chip pixels in y

---

34 which means: do not use software which is partially FITS unaware...
Three values that describe the geometry of the detector and which are always the same because all instruments use Hawaii-2 or Hawaii-2 RG detectors.

- \( B_{\text{EXT}1} = 2.530273 \) [V] external bias
- \( B_{\text{DSUB}1} = 0.000000 \) [V] det. bias voltage DSUB
- \( B_{\text{VREST}1} = 0.500000 \) [V] det. bias voltage VRESET
- \( B_{\text{VBIAG}1} = 3.222656 \) [V] det. bias voltage VBIASGATE

Four values per chip (Hawaii-2) or 10 values per chip (Hawaii-2 RG) that show the voltages applied to the detector chip, which are set by DAC’s and are defined by keywords in the GEIRS patterns (and potentially modified by the bias command). The comments show the DAC inputs in the range 0–4095 for the most recent GEIRS version.

- \( \text{CREATOR} = '\text{GEIRS : trunk-r700M-13 (May 16 2013, 15:51:59)}' \)

GEIRS SVN branch, version, and timestamp in parentheses. The timestamp is the time when GEIRS was compiled on the local computer, and does not reflect the issue date of the GEIRS version—which may be much earlier.

- \( \text{EOF00000} = \ldots \)
- \( \text{EOF00001} = \ldots \)
- \( \text{EOF00002} = \ldots \)

These keywords denote end-of-frame time of arrival of the last byte of the frames in the GEIRS DMA buffers. The units are the same as the STRT_INT and STOP_INT units, i.e., UT seconds in the range from 0 to \( 24 \times 3600 = 86400 \) (the number of seconds per day). Details:

- More precisely: the keyword EOF00000 is not a time that marks the end of a frame but a start of triggering the read; therefore the time difference between EOF00000 and EOF00001 depends on the idle modes. The number of values with positive index is the product of CPAR1 and NEXP, covering the entire set of frames. If the exposure was aborted, the number of values is smaller.

- For the correlated double-sampling modes, the arrival of the reset-frame is not measured and the even indices (with the exception of 000) are absent.

- Where CAMDPORTS equals 2 (Section 3.2), each time is the mean of the two arrival times of the parallel streaming through both fibers.

- The first differences are added in the comments and ought to be basically the same on the milliseconds level. The jitter in these first differences indicates the standard deviation of the time accuracies, as sampled on the Linux workstation. The actual jitter of the timing on the ROE is much smaller.

- In simulation mode the jitter is larger than collecting OPTPCI data, because simulated images are calculated in a non-privileged user process on the workstation. In simulation mode the EOF timing differences are basically always larger than one second because the simulation always computes full-frame images and is unaware of any of the speed-up methods (Section 7.8). So these keywords may not match EXPTIME or ITIME but may just indicate a maximum speed at which the software generated some diffused star images.

- \( \text{PERCT025} = \ldots \)
- \( \text{PERCT050} = \ldots \)

\ldots
PERCT500 = ...

PERCT975 = ...

provide the ADU levels of 2.5%, 5%,...97.5% percentiles. The value of PERCT500, for example, is the median ADU in the corresponding image or frame. The data allow a quick look at the saturation level inside the image. If the keywords are generated, a quick extraction of the median for example of a sequence of FITS files can be generated with a script like

```bash
#!/bin/bash

cd .../2015-03-02 # move to the data directory
for j in Linr*.fits ; do # loop over the FITS files of interest
    # extract PERCT500 (the 50.0 percentile) from extension 1
    dfits -x 1 $j.fits | fgrep PERCT500 | awk '{print $6}' ;
done

or for named extensions

#!/bin/bash

cd .../2015-03-02 # move to the data directory
for j in Linr*.fits ; do # loop over the FITS files of interest
    # copy the extension of interest to the primary header of tmp.fits
    rm tmp.fits
    ftcopy "$j[SCA1]" tmp.fits copyall=no ;
    # extract PERCT500 (the 50.0 percentile) from primary header of tmp.fits
    dfits tmp.fits | fgrep PERCT500 | awk '{print $6}' ;
done

• RESWN001 = ...
RESWN002 = ...

indicate the location and dimension of reset windows in the srre mode. The indices in the keywords are generally not the same as in the operator’s initial files. The format is the same as with the DETSEC and DATASEC keywords above: the 1-based x and y FITS coordinate system spanning all detector chips, the x range before the comma and the y-range after the comma. For LUCI1 and LUCI2 the maximum of the both ranges is 2048. For CARMENES, the y-ranges are up to 2048, the x ranges on SCA2 are subranges of 1–2048, and the x ranges on SCA1 are subranges of 2049–4096. (This means that the indices continue across the chips as if there was no gap.)

• NRESWN = ...

The number of srre reset windows on this chip. This counts the number of RESW{i} header cards in this image extension. The value may be zero.35

35There is an unfortunate and irritating practise inside the CARMENES ICS client to use the srre mode with zero reset windows configured. So one cannot rely on the READMODE card to differentiate srr and srre mode but must scan the NRESWN in both image extensions to figure this out. Newer GEIRS versions try to patch the READMODE keyword in the FITS files to correct for that error. These cases ought of course be replaced by using the plain srr mode.
• **ABRT = ...**

  The time when GEIRS last received an `abort` command. This is only relevant if that time is later than `DATE-OBS`, because otherwise this happened before the exposure of this FITS file. It mainly serves to track and debug the behavior of client software which has unpredictable or undocumented itches of sending `abort`.

• **UUID =**

  A version 1 Universally Unique Identifier. May be decomposed into time stamp and MAC for example here, here, or here.

**Warning:** The first-stage-pipeline or ICS may overwrite some or all of these keywords or values, which may change their meaning.

The keywords **CHECKSUM** and **DATASUM** appear if the associated `save` option is used.

A warning to ds9 users for PANIC: the all-mosaic composite image created by GEIRS (for example if `–M` is not used) does not contain any filler pixels to represent the gap between the chips. The ds9 display of these images shows nevertheless a grid of astronomical coordinates which cannot be aware of this—presumably derived from the pixel scale and assuming that the α/δ pointing refers to the center of the image. Obviously, that grid is typically wrong by roughly half of the gap, ≈ 80 pixels or the order of 40 arcseconds.

To simplify looking at the images with ds9, GEIRS places a WCS coordinate system on the two CARMENES FITS extensions. This has its origin at the middle of the detector plane in the gap between the two chips, and measures millimeters along the right (X) and up (Y) direction in the optical plane (i.e., ignoring the rotations and flips of the image).

### 6.5 Image Location

For Hawaii-2 RG detectors (PANIC, CARMENES, Luci1, Luci2), GEIRS copies the four reference pixels along each of the four edges into the FITS images (if they are inside any of the subwindows). Postprocessing programs ought be aware of the fact that these pieces of the images do not contain regular data, and that the usable region is only a maximum of 2040 × 2040 pixels per chip.

Using (or not using) the `save` options `–1` (requesting FITS cubes) and/or `–M` (requesting the multiple extension FITS format) leads to four different layouts of the FITS files:

• Without the two options, each window of each image is stored in the first (primary) HDU of a single file. This leads to the largest number of files and the smallest individual sizes of the files. In the extended syntax of the form `filename[.extname...]`, where the piece in brackets is the name of the extension as shown in the EXTNAME keyword of the HDU, this is:

  ```
  fname_0001_win1.fits # 1st window, first image/frame
  fname_0001_win2.fits # 2nd window, first image/frame
  ... 
  fname_0002_win1.fits # 1st window, second image/frame 
  fname_0002_win2.fits # 2nd window, second image/frame 
  ```

  The first part of the file name is under user control with the standard mechanisms (Section 5.3), but not the trailing part of the underscore, `width` and suffix.
• With -1, each window is stored in a separate file. Each image is a slice in a FITS cube of the primary HDU.

    fname_0001_win1.fits # first window, all frames as a cube in primary HDU
    fname_0001_win2.fits # second window, all frames as a cube in primary HDU

The first part of the file name is under user control with the standard mechanisms (Section 5.3), but not the trailing part of the underscore, wini and suffix.

• With -M, each image is stored in a single file; the second, third HDU and so on contain the various windows of the image.

    fname_0001[win1_1].fits # 1st image/frame, first window on first chip
    fname_0001[win1_2].fits # 1st image/frame, second window on first chip
    fname_0001[win2_1].fits # 1st image/frame, first window on second chip
    ...
    fname_0002[win1_1].fits # 2nd image/frame, first window on first chip
    fname_0002[win1_2].fits # 2nd image/frame, second window on first chip
    fname_0002[win2_1].fits # 2nd image/frame, first window on second chip

In general, the extension name starts with win, attaches a number (starting at 1) for the infrared chip, an underscore, and another number (starting again at 1) as the index of the window in the set of all windows on that chip. For detectors with a single chip (LUCI1, LUCI2, LN, NTE), the first number is always 1.

• With -1 and -M, all images of an exposure are stored in a single file. Individual windows are stored as a FITS cube in the first, second HDU and so on, where the layers in the cube are formed by the consecutive images. (If there is only one exposure, the format is automatically reduced to the standard 2D image format, which means the NAXIS keyword becomes 2.) This is the best organized display for multi-exposures with more than one window, but yields the largest files.

    fname_0001[win1_1].fits # first window on first chip, all frames as cubes
    fname_0001[win1_2].fits # second window on first chip, all frames as cubes
    fname_0001[win2_1].fits # first window on second chip, all frames as cubes
    ...

In summary, without -M all images are in the primary HDU, with -M no images are in the primary HDU. For CARMENES the -M is permanently switched on, even if the option is not added to the save command.

The extension SCA1 contains the data of the upper of the two detector chips (which is addressed with index 1 and det1 in the commands), the extension SCA2 contains the data of the lower of the two detector chips (which is addressed with index 2 and det2 in the commands): Figure 22.

After a change request of M. Zechmeister in late 03/2015, the FITS coordinate system in the detector plane was changed by modification of the parameters (Section 3.2) such that SCA1 is shown to the right in the online display and placed at the horizontal FITS coordinates $2049 \leq x \leq 4096$, and that SCA2 is shown to the left in the online display and is placed at the horizontal FITS
coordinates $1 \leq x \leq 2048$ (Figure 12). In addition there is flip around the long axis of the mosaic; FITS images are equivalent to looking at the detector plane from the *rear* side. This flip also causes a different orientation of the FITS images compared to Figures 29 onwards of the Optics FDR [28].

GEIRS adds a WCS coordinate system named `Det. Plane` to the FITS image extensions. Its two axes are measured in millimeters. The origin is in the middle of the gap between the two infrared chips. The first coordinate is along the direction horizontal-right in the laboratory if looking onto the surface of the chips, equivalent to up in the FITS images. The second coordinate is along the direction up in the laboratory, which means the direction right in the FITS images. The determinant of the WCS matrix is negative. Altogether this means sticking a left-handed coordinate pair to the FITS images which represents a right-handed coordinate pair in the laboratory frame coordinates.

Any postprocessing software knows from the `DATASEC` value which region of the full detector is covered by the window of any particular HDU, and retrieves the number of frames or images from the `NAXIS` and `NAXIS3` values.

Single-frame output from GEIRS uses 16-bit data types in the images; correlated output uses 32-bit data types. Converting all images to 32-bit data can be implemented by calling `chimgtyp` from within `QueueFiles`. The current name convention for the extensions (`EXTNAME`) is `Qd_w` for PANIC, `SCA1|SCA2_w` for CARMENES, and `wind_w` for the other instruments, where $d$ is the chip.

---

[^46]: There is a bug of incorrect BSCALE/BZERO values in some `chimgtyp` output files which will probably be fixed in HEAsoft 6.23. See [FTOOLS #2427].
number from 1 to 4 and \( w \geq 1 \) is a window number. If the operator did not use subwindows, \( w \) is always 1. The index \( w \) is not necessarily the same as used in the \texttt{subwin} command; exceptions occur if

1. the operator skipped numbers,
2. defined but disabled some of the intermediate subwindows,
3. or let some windows stretch over multiple chips.

The physical order of the MEF extensions is by window number \( w \), which just reflects the operator’s liking for the order of enumeration in the \texttt{subwin} command. If a window has been split because it covers more than one detector, the split windows stay close together huddled in a group, so there is an “inner” or “fast” loop over the chips then.

### 6.6 Image Construction With \texttt{srre(e)}

If GEIRS has obtained a sequence of frames in the “sample-up-the-ramp” modes, it generates by default an image with the following procedure, pixel by pixel:

1. The ADU values are (virtually) plotted along the time axis.

2. The first datum — the one of the reset frame — is discarded to eliminate the reset frame anomaly. (This elimination happens only if there are at least three reads along the ramp, as a protection against having only a single point left in the plot.) Basically all instruments have a reset value that is a few ADU’s higher than what would be obtained by interpolating the later values backward in time; this measure considers the first datum to be worse than the others and better be ignored if possible.

   The number of frames that are discarded can be changed by the operator with the \texttt{use rrr skip} command, see Section 5.3. The current value is obtained with the \texttt{status use} command.

3. Other points in that plot exceeding a threshold ADU value are also discarded. Because the MPIA electronics uses 16-bit ADC’s, the range for these thresholds is somewhere smaller 65535 (which equals saturation). So this is a single number parameter with the intent to ignore values that are near saturation or not appropriate for a standard linear fit because they are too high up in the nonlinear regime.

   That value is the \texttt{ADC SATUR} parameter in the shared memory data base, so it can be changed and read by the operator with the \texttt{put} and \texttt{get} commands of Section 5.3.\footnote{Changing the default that applies after starting GEIRS needs a change in the source code that initializes these data.}

4. A simple linear least-squares fit through the remaining points of the plot follows. The slope of that straight line is multiplied with the exposure time and that product becomes the ADU value for that pixel in the image. “Image” refers to the display in the GUI, Figure 12, and to the FITS image stored on disk with \texttt{save}\footnote{Of course the reduction does not apply to the the single-frame formats described elsewhere in the manual.}.
For CARMENES a dedicated postprocessing procedure has been added that mainly i) applies a non-linearity-correction based on quadratic fitting coefficients and ii) narrows the number of frames that contribute to the fit to a small number of frames at the start and at the end of the procedure [5]. That sort of pipeline is not integrated into the other instruments. All these efforts are considered part of the data reduction pipeline and not part of GEIRS, the detector control software.

In summary, GEIRS does not have a build-in nonlinearity correction nor a cosmics suppression scheme that is applied when it reduces the raw frame data of successive non-destructive reads to an image. All instruments which need these improvements must save the individual frames to disk with one of the methods offered by GEIRS, apply their corrections to each frame, and re-correlate the frames to obtain the images.

6.7 Single Frame Dumps

This operative software mode refers to saving uncorrelated single frame snapshots to FITS or to a raw binary files in a scratch directory—while the packages of the 16-bit data of the (nondestructive) readouts arrive in the kernel buffers. If activated, this software on the GEIRS workstation considers each frame as soon as it has been read out by the detector, cuts out rectangular regions of interest, and dumps these pixels to an interface where the information is available to other (online) pipeline procedures.

The information extracted this way from an incremental read-while-integrate exposure may be used to steer other optical elements of the telescope looking at jitters and shifts/drifts in these images. The aim is that one does not need to terminate the readout cycle with abort or wait for the end of the integration time to get hands on the images. The profit is that any online tool may analyze the frames. In principle another profit may be that one can skip a save -S command at the end of the exposure which saves some time if there are hundreds of frames in long exposures—supposed the dumped frames are moved to their final destination during the exposure by some other mechanisms to avoid that they are overwritten by the next exposure.

The principle of operation is that these image data are stored with the frame arrival frequency to individual files without effecting otherwise the mixes of resets, readout patterns and windows without waiting for the end of the exposure. This almost always implies that the operation is bestowed with its local definition of data sections (windows) so the GEIRS data interface may cut out only those data essential to monitoring the data quality such that

1. the computational load due to the additional disk transfer (including the load by the reading application) is kept low.

2. the risk of stalling the main data processing task enforced by additional locking mechanisms with these buffers remains small. (The data interface works by drawing local copies of the standard shared memory data buffers parallel to the read process; if it is too slow, the standard procedure may fall behind its schedules working through the “read” and the “save” pairs of buffers.)

To stabilize the operation/mechanism against overloading by too frequent or too large window files, the implementation skips frames that are scheduled to be created while a previous frame is still being worked on. So depending on disk write speed, any disk activities of other processes running on the same computer, CPU speed, number of pixels in the dumped images, and of course frame
frequency (depending for example on the delay used with the data generator), some of the files might not be created. Even nowadays, computer speed is not infinite. If you entertain the system with stupid tasks (like asking GEIRS ten times a second about the current status), the probability of not observing the intended number of FITS files on disk grows. Another example of the kind is the data transfer of up to 800 FITS files on the CARMENES computer to another computer after the save command has finished; this will transport 12 GB across the CAHA network and therefore will still continue after the next exposure has started.

Note that FITS header keyword NFRAME relates to the sequential enumeration of frames in the shared memory buffer. If the FITS files have NFRAME=30, NFRAME=31, NFRAME=32, NFRAME=34, NFRAME=35, for example, frame number 33 has not been dumped because the operating system was too busy at that time. That scenario can be uncovered with a command like

```
dfits *.fits | fgrep ' FRAME ' 
```

in the current save directory and looking for gaps.

The operator may in addition slow down the dumping frequency below once per read with two keywords in the configuration file: The relation between the number of created FITS files $N_F$, the integer subsampling factor $s$ and the number of frames $N$ (effective, optionally after abortion) in the RAM is

$$N_F = 1 + \left\lfloor \frac{N - 1}{s} \right\rfloor. \quad (9)$$

Also note that this final save is not flagged as done at the end of the exposure (because obviously that computes a correlated image from all the previous frames and is of a very different kind of quality, depending on the save mode).

There requirements to install/activate this concurrent eaves-dropping mechanism are:

1. The sfdump (single frame dump) command (Section 5.3) is called to tell GEIRS which sections of the windows (or full frame) are to be written where. The creation of these pixel data files happens up to the time it is switched off with sfdump off or until GEIRS is shut down. The sfdump command actually points to a configuration file that contains the bounding boxes of the windows’ geometries, and auxiliary parameters.

2. The readout mode is the LIR mode or one of the multi-correlated modes (Fowler, sample-up-the ramp, ...). The single-frame dumps are not created for other types, because the reset frame is supposedly useless and the next frame anyway to be saved in these cases. (One does not need to call sfdump off if a sequence of different readout modes is started that mixes double and multi-correlated modes. The creation of the intermediate files will simply pause if the current mode is not a multi-correlated one.)

The ADC data within the windows specified in the configuration file named in the sfdump are written either in

- a MEF format with BITPIX=16 and one window per extension if the RAWF flag in the configuration file is F or not given.

- or a binary stream with two bytes per pixel in the endianess of the GEIRS computer window-after-window if the RAWF flag in the configuration file is T.\textsuperscript{39}

\textsuperscript{39}This file format can for example be read with od -d ....
The intended scenario is that the monitoring programs are using the commands like `sfdump sfdump.cfg` once, and edit the file `sfdump.cfg` after a `save` and prior to the next `read` if the window number or geometry needs to be adjusted. GEIRS re-reads the configuration file (that was `sfdump.cfg` in the example above) for each frame arriving from the detector, so editing the file while a `read` is ongoing may lead to unpredictable results.

The regions/windows specified in the configuration file do not need to aligned in any particular way with the hardware and software windows specified by the `subwin` command. The windows in specified in the `sfdump.cfg` may overlap. Any pixels of the regions that fall outside the subwindows which actually are covered by the detector data are filled with zeros.\(^{40}\)

The implementation is by default dumping data into a directory without any overwrite protection (!)\(^{41}\) and iterating over the same base file names during every new read. We assume that these windows contain scratch data for online processing and do not have any lasting value, and in this way avoid that an extra monitor on available disk space in this part of the file system is needed. We assume that the lasting files are written explicitly with the `save` command to a different (!) directory.

The configuration file contains parameters, one per line, following a FITS-style template syntax as described in the cfitsio manual:

- **COMMENT** [anything...] lines to be ignored, only for documentation purposes
  - **WINidx = ['xstrt:xend,ystrt:yend']** A portion of the detector image in the standard 1-based FITS syntax. On the right hand side, the two brackets, two colons and comma must be present as single-letters and the entire string on the right hand side must be encapsulated by quotes. The `idx` are distinct positive integers enumerating the windows.

There must be at least one `WINidx` keyword in the configuration file—otherwise no files are produced.

This window set defined by the `WIN` keywords usually differs from any of the sets that are specified with the `subwin`. The regions of the detector that are copied with the `sfdump.cfg` mechanism are fixed by a 2-step process: (i) The detector is read out in the regions configured with the `subwin` command. In most instruments that command is not used, which means actually all detetor pixels are read (full-frame hardware windows). (ii) This is followed by another cut-out process by the GEIRS software that virtually lays out these hardware windows and extracts sub-regions with the geometries defined by the `WINidx` parameters. Think of this as stacking two sets of masks (harware and software windows) on top of each other.

The portions of the areas defined by the `WIN` keywords that lie outside the regions that are read out will be filled with zeros. The windows may overlap; this leads to replicated shared pixel values inside intersections in the output.

If there are two `WIN` keywords with the same index `idx`, only the latter one (further down in the file) will be used.

The indices do not need to be in consecutive integer order; there may be holes. (Actually all keywords that start with `WIN` and have a value string with the syntax of the four corner

\(^{40}\)The current implementation also copies reference pixels of Hawaii2-RG detectors into the regions, which may change in the future.

\(^{41}\)i.e., the definition of the `clobber` command are ignored
coordinates will be included in the window list.) If these indices are integers, they are copied into the EXTNAME of the FITS extensions for cross-identification.

In the case of an instrument with a single Hawaii-2 or Hawaii-2RG detector, one may for example copy all pixels to the file with the specification \( WIN1 = '\{1:2048,1:2048\}' \).

- **RAWF = T or F (boolean)** Use a bare unsigned 16-bit binary format in the endianess of the GEIRS host, if true, otherwise a FITS format. The default is F (i.e., output file format is FITS, not raw, if this keyword is missing. The bare format has as many bytes as the number of pixels in all windows (defined above) multiplied by 2, where 2 is the number of bytes per pixel. The order of the pixels is first a block for the first window, then a block for the next window, in the order implied by the WIN keywords. In each window, pixels of the bottom line (smaller y-coordinates) come first, pixels of the top line last. Within each line of pixels the order is left-to-right (smaller x-coordinates first).

- **VERB = T or F (boolean)** If true, pack a standard (more complete) list of keywords into the FITS headers. This means that the GEIRS standard FITS keyword list is produced, and that keywords are also modified according to the rules of the geirsPhduAdd files. If false, include only a minimum set of keywords. Writing the minimum set is faster, and usually sufficient if the files are anyway only scratch image files. The default (if the VERB specification is missing) is F.

- **PERCT = float. If \( > 0 \) and \( < 0.9 \), calculate a histogram of values and add these as PERCT keywords in the associated headers. This is the difference of percentiles; a value of 0.05 means for example that 19 values are effectively calculated at 0.05, 0.1, 0.15 and so forth. The default (if the PERCT specification is missing) is -1, so this is disabled for performance reasons.

- **FDIR = 'string'** The name of a directory to which the files are written. If the keyword is missing, the default directory is \$CAMTMP/fits. If the string is empty, the directory is the same directory (dynamically) as where the other FITS files go.\(^{42}\) Of course this should be a directory which is cleaned up with a cron tab entry on a regular basis. The directory will be created with standard permission mask 022 if it does not exist. Of course this will fail if the GEIRS operator has insufficient write permission on any of the parent directories.

- **FNAM = 'string'** The base name of the files to be written. If missing, the default is an empty string. The full name of the files will be \(<\text{FDIR}>/<\text{FNAME}><4\text{digitFrameNo}\.fits\) if they are FITS files, otherwise \(<\text{FDIR}>/<\text{FNAME}><4\text{digitFrameNo}>\). These files are overwritten if existing, independent of what has been specified with the clobber command.

- **TSTMP = 'string'** The name of a file in the FDIR which is touched after each dump. This is another passive form of signalling to monitoring processes, which might poll that file’s content. If missing, no such time stamp files are created. The file contains the most recently created FITS or binary file name, a time stamp, and the number of subwindows (extensions) in that file.

- **SUBSAMP = integer** Subsampling of the frames such that not all frames collected by the computer are dumped but only a regular subset. The number of frames skipped in between (not dumped) is one less than the integer. If not specified a number of 1 (effectively no sub-sampling) is used. If the integer is 2, for example, the first, third, fifth, . . . frame is copied to the file.

\(^{42}\) Again: this is definitely not recommended because the files there are considered permanent data and the sfdump subroutine may erase files there…
\begin{itemize}
\item \textbf{MAXSAMP = integer} The maximum number of files to be created for the exposure. This is another way of defining the subsampling factor through a more dynamic interface than with the SUBSAMP keyword. If the number of frames predicted by the integer parameter of \texttt{ctype} is larger than the product of MAXSAMP by SUBSAMP, SUBSAMP will implicitly be increased such at most MAXSAMP files will be created by the single frame dumps. If not specified a number of 99999 (effectively no limit) is used.

\item \textbf{CALLB = 'string'} The name of an executable (callback) to be called after the file is created. If missing or empty, no action is induced. There are two optional placeholders \%s and \%d in the string. The first is replaced by the name of the new file, the second by the increasing number of the frame. This string should be ending on a \& to put the callback in the background. Otherwise, if the callback needs more computation time, it might block the next round of the callback to be executed. The implementation is based on system(2) calls, so redirection of its stderr and stdout need some embedding into sh calls.
\end{itemize}

Each of these configuration lines may be followed by a slash and a comment. This trailing part does not matter to GEIRS.

Header cards with other keywords than those listed above are ignored.

The line lengths in the configuration file do not matter much, but the keyword and value part must not surpass the standard 80 bytes of FITS header lines. (This effectively puts a limit on the length of the FDIR.)

A rough check that the configuration file is readable is made at the time \texttt{sfdump} is used. GEIRS attempts to open and read the configuration file are done later with the next \texttt{read}.

Example of a well-formed configuration file:

\begin{verbatim}
COMMENT xample file like sfdump.cfg
WIN2 = '[40:100,700:900]' / first window, EXTNAME WIN2 size 61 x 201
FDIR = '/tmp/mathar/fits' / directory of FITS SFR files
FNAM = 'sf' / the FITS files will be sf0001.fits, sf0002.fits.. 
WIN5 = '[80:110,700:900]' / second window, EXTNAME WIN5; overlaps with WIN2
COMMENT PIDSGL = -1
TSTMP = '/home/mathar/tmp/last' / updated with each new frame
RAWF = F / create FITS files
VERB = T / include full FITS information
SUBSAMP = 3 / dump not all but each 3rd frame (skip 2)
CALLB = 'touch /tmp/mathar/cb%d &' / shallow log trace of callbacks
COMMENT end of xample file
\end{verbatim}

If the keyword above were changed to \texttt{RAWF = T}, files of $2 \times (61 \times 201 + 31 \times 201) = 36984$ bytes would be created.

The frame dumping mechanism is permanently switched on for CARMENES by default with a line of the type

\begin{verbatim}
ln -s -t ${CAMTMP} ${cambuild}/admin/sfdump.carmenes
\end{verbatim}

in the startup script. One can disable that all-frame dumping by commenting this line with a hash
In this case the first-stage pipeline will realize that the configuration file is not at its standard place and dump the (few) frames it is configured to use when the `save` command arrives [5].
7  EXPOSURE TIME

7.1  Nomenclature

The expected time that expires between the start command and the receipt of the last pixel values of the last frame is of interest to exposure time calculators. It is a function of readout mode parameters and is estimated by the formulas summarized below.

The overhead of (i) additional computations if the frames are to be averaged/integrated with special options of the save command and the overhead of (ii) actually writing the FITS frames to disk is not included here. These are functions of number and types of CPUs and disk speeds of the computer on which GEIRS is run, and depend also on any post-processing tasks added to the QueueFiles like the first state of the pipeline [5].

The number of frames still to be read may be monitored by sending the status frame read to the server, which responds by counting upwards as a function of time. This is equivalent to looking at the numbers that appear at the Read label in Figure 5 which turns yellow after the start is received. The two dominant parameters are the repeat factor (which is available by sending status crep) and the cycle time (which is available by sending status ctime). For any supervisor script it is much easier to deduce the real time of exposures by taking the cycle time as the base unit than taking the integration time, because the influence of parameters like EMSAMP, PREAD, PSKIP, the pair count of the multi-correlated (Fowler-type) samplings, and any form of hardware windowing (first type in Section 5.6.1) has already been incorporated then. The composition of the cycle time by interlaced execution of resets, reads, and idle waits is described elsewhere (see Section 1.2).

Note that the precision of this prediction is generally not better than the cycle time for all modes that use (or are coupled with) the ROE idle mode named wait. The reason is basic and simple: the start command is generally not synchronized with the idle cycles of the detector readout. The first pixel read waits (as the name says) for the end of the present idle cycle. (The need to read the detector even if no data are emitted by the electronics is a fundamental aspect of infrared detector exposure management and not discussed in this software manual.) The mean value of the time is the value expected for the break idle mode plus half of the cycle time. (One can mitigate this effect by adding a sort of dummy sfr exposure with minimum short integration time at the end of all long exposures—which will be adjusted upwards by GEIRS to the shortest manageable value—. The next exposure will then find the detector in a short cycle mode and react with predictable latency. The associated waste of disk space and overhead time can be kept low by saving these with the -d option.)

The formulas below contain small fudge factors that have been obtained by fitting a small number of exposures. They realize some overhead caused by the data transfer chain from the ROE via DMA control to the GEIRS buffers on the server.

7.2  Lir with idle break

If the readout mode is line.interlaced.read with idle mode break the time is

\[ t[\text{sec}] \approx 0.3 \times N_f + t_{cyc}[\text{sec}] \times N_f \]  

(10)

where the number of frames \( N_f \) has been set by the application with the crep command and where \( t_{cyc} \) is the cycle time.
7.3 frr with idle break

If the readout mode is `fast-reset-read.read` with idle mode `break` the time is

\[ t[\text{sec}] \approx N_f \times t_{\text{cyc}}[\text{sec}] + 0.03 \times N_f. \]  
\hspace{1cm} (11)

7.4 mer with idle break

If the readout mode is `multiple.endpoints` with idle mode `break` the time is

\[ t[\text{sec}] \approx N_f \times t_{\text{cyc}}[\text{sec}] + 0.003 \times t_{\text{cyc}}[\text{sec}] + 0.005 \times N_f. \]  
\hspace{1cm} (12)

There is no explicit dependence on the `CPAR1` parameter (number of Fowler pairs) which is already incorporated in the cycle time.

7.5 sfr with idle break

If the readout mode is `single.frame.read` with idle mode `break` the time is

\[ t[\text{sec}] \approx N_f \times t_{\text{cyc}}[\text{sec}] + 0.06 \times N_f. \]  
\hspace{1cm} (13)

7.6 Hardware Windowing

The action of hardware windowing (Section 5.6.1) skips line set blocks along the “slow” readout direction of each of the detector chips. The slow direction is parallel to the stripes of the 32 or 64 readout channels. For Hawaii2 RG or Hawaii4 RG chips run with an odd `CAM_DETROT90` parameter (LUCI, CARMENES, NTE), the slow direction is left-right in the images. For Hawaii2 or Hawaii4 RG RG chips run with an even `CAM_DETROT90` parameter (PANIC), the slow direction is up-down. For Hawaii2 chips (LN) the slow direction depends in which of the four quadrants the subwindow is placed.\(^4\)

Neglecting details, the time is shortened proportional to the number of pixels that are not fed into the 32, 64 or 128 ADC’s, because the conversion takes the lion’s share of the readout time. An estimate of the maximum speedup (and associated shortest integration time) relative to the full-frame readout is obtained by projecting all hardware windows (on a per-chip basis for the Hawaii2/4 RG and per-quadrant basis for the Hawaii2) as “shadows” onto their slow directions, which defines a set of one-dimensional pixel intervals (overlaps merged where occurring). Due to the back-to-back mounts of Hawaii2 RG’s for PANIC and CARMENES, the orientation of interval must be chosen different for half of the chips, from a corner of the mosaic into the direction of the midpoint of an edge of the mosaic.

The total number of pixels in that set of intervals relative to 2048 is the relative speedup and reduction in integration time that can be achieved. This is not proportional to the ratio of the pixel-sum in the windows over the pixel-sum in any of the detectors, but proportional to some kind of edge-length sum along the slow readout direction. An example with three subwindows placed

\(^4\)The `subwin auto on` command dissects windows that cross chip or quadrant boundaries so the observer does not need to be fully aware of details.
Figure 23: Illustration of the relevant dimensions that govern timing with three hardware subwindows \( w_1, w_2 \) and \( w_3 \) in the LUCI or NTE case.

over a LUCI or NTE detector with a full frame width of 2048 pixels is given in Figure 23: the ratio \( (x_1 + x_{23})/2048 \) of the projected pixel widths is the expected reduction in cycle time (inverse of the speedup factor), where \( x_1 \) and \( x_{23} \) are the number of projected pixels of the windows measured along the slow (horizontal) readout direction.

The GUI in Figure 5 can be used as a pocket calculator for these times. Once the subwindow is defined and enabled, so the associated Subwins button is green, one can enter an integration time of zero into the IT; GEIRS sums up the pixel clocks in its patterns according to the selected readout mode, and inserts this minimum time back into the GUI. (This works also in simulation mode.)

A numerical example for the Hawaii2 4-quadrant case of LN: If the width of an isolated window is increased by one pixel along the slow direction, the total number of pixels read out increases by \( 4 \times 1 \times 1024 \). The number of pixels channeled through a single ADC increases by \( 4 \times 1 \times 1024/32 = 128 \). At a pace of the (standard) pixel read time of 10,000 ns (\texttt{prd} time in Figure 5), the increase in time is \( 128 \times 10 \) ms = 0.00013 s. This number is for a single read; for an \texttt{ir} double read this becomes 0.0025 s (which will usually be announced in the controls GUI of Figure 5 as twice as that as long as the repetition factor is kept at 1 because the group of the first read-reset-read and the second read-reset-read is added all up).

A more detailed timing analysis of the most recently enabled pattern is kept in \$\text{CAMTMP/timing-cmds.log} \), and \texttt{status subwin} shows some of the window geometries that are involved [8]. A coarser measured timing of frame arrival times on the workstation is found in the \texttt{EOF} keywords in the FITS headers.

As a practical result of this analysis, one does not “lose” time if windows are stretched along their maximum extension along the fast direction. So for LUCI an assignment of the format

\texttt{subwin SW i x y w h}
can always be replaced by

\texttt{subwin SW \textit{i x 1 w 2048}}

expanding the window up-down. For PANIC, the assignment can be replaced by

\texttt{subwin SW \textit{i 1 y 4096 h}}

expanding the window right-left over both detectors. This will keep the integration times almost constant, but lead to larger detector regions in the FITS files.

### 7.7 Higher resolutions

#### 7.7.1 Readout times across the detector surface

The fact that the MPIA electronics reads 32 channels of 4 quadrants of the Hawaii-2 detector chip in parallel leads to a characteristic pattern of 32 time ramps of pixel reads across the detector. Figure 24 illustrates for a single full-frame reset-read at which time the individual pixels are reset and read. The first 32 pixels are read at time 0; the last pixels are read at time \( \frac{2048^2}{32} = 131,072 \), which is scales to \( \approx 1.4 \) seconds—half of (1)—for the standard PSKIP, LSKIP etc. parameters.

![Figure 24: Pattern/distribution of effective pixel time as a function of Hawaii-2 pixel position. The transformation of the two axes directions to the FITS and image coordinates depends on the currently active \texttt{CAM\_DETR0T90} and \texttt{CAM\_DETXYFLIP} parameters (Section 3.2).](image)

For Hawaii-2 RG detectors (not relevant to LN) there are not 32 ramps in quadrants but 32 ramps with tops and valleys stretching over each chip. Otherwise the time scales are the same as above, because the number of channels per chip, the number of pixels per chip, the pixel read base times and ADC conversion times are the same as for the Hawaii-2 types. (The “fast” direction along
the 64 pixels of each channel of Hawaii-2 RG detectors is alternating by default after the detector is powered on; 16 of them are oriented left-right and 16 of them right-left. The GEIRS patterns, however, use the chip’s registers to toggle the direction of each second channel such that the actual readout direction is the same for all 32 channels. This aims to simplify a lot the arithmetics of mapping the serial 16-bit datastreams after arrival on the computer to two-dimensional images.)

The 1.4 seconds difference in the readout time correspond to the CARMENES radial velocities as follows: we assume that the diurnal change in the radial velocity is of the shape

\[ v = \hat{v} \cos\left(2\pi \frac{t}{24 \times 3600\text{s}}\right) \]  

where the amplitude of the velocity at the CAHA geographic latitude is \( \hat{v} \approx 360 \text{ m/s} \). A shift of \( \Delta t \) in the readout corresponds to

\[ \Delta v = \frac{dv}{dt} \Delta t = -\hat{v} \frac{2\pi}{86400\text{s}} \sin\left(2\pi \frac{t}{24 \times 3600\text{s}}\right) \Delta t \]

such that \( \Delta t < 1.4\text{s} \) leads to \( \Delta v < 3 \text{ cm/s} \), which is generally negligible.

For all relevant readout modes, the times of the pixel reset and the times of their readout are coordinated such that both have the same type of “offset” on absolute time scales [9]. In consequence,

- the differences (the exposure time) between reset and readout are constant across all pixels and all detector chips (with the exception of the reset windows in the \texttt{srre} mode);
- the mean (center) time of the photon flux has the same, predictable offset as a function of pixel location in the detector.

Note that if hardware subwindowing is used, these time axes can be squeezed considerably and become a more complicated function of placement and size of the windows on the chips. (If instead the windows are only established by slicing the images by software on the GEIRS computer, the pixel timing is the same as for the full-frame readout. This way of obtaining the information in windows by pure software postprocessing is not much relevant in practise.)

To visualise the timing across the detector chips one may actually take an exposure in the single frame read mode (sfr) under rather strong illumination with the default (=shortest) exposure time. Because this readout mode resets all chips of the detector at (almost) the same time and then starts reading the pixels in their “channeled” order, the actual exposure time is zero for the pixels read out early and longest for the pixels read out last. Just looking at the FITS image at sufficient contrast then displays “bars” of brightness variations along each readout channel.

### 7.7.2 Chopped illumination

As explained above, the start time of the exposure is a function of the position on the detector. With CARMENES for example, the first rows of the two detectors are actually the outermost vertical columns in the FITS system. Let \( T_0 \) be some exposure start time of the readout and

\[ T_s(x) \equiv T_0 + \frac{1}{2048} T_c \times \begin{cases} 
(x - 1), & \text{SCA2} \\
(4096 - x), & \text{SCA1} 
\end{cases} \]  

(16)
the start time as a function of FITS x coordinate. For the \texttt{srr(e)} mode the ramp time is $T_c \approx 1.4$ seconds and depends in detail on parameters like the electronic multisampling. The exposure ends at $T_s(x) + E$, where $E$ is the scheduled exposure time.

A model of a chopped illumination with an interception of the light path before the detector has three parameters, the time $T_{s0}$ when the shutter opens, the duration $D_1$ during which it is open, and the duration $D_0$ when it is closed. We assume the shutter opens and closes with a period of $D_1 + D_0$. The fraction $D_1/(D_0 + D_1)$ is the average attenuation due to the shutter. The effective exposure time of a pixel is the sum of all times in the interval $[T_s(x), T_s(x) + E]$ where the shutter is open; in the diagram which shows the shutter periods in the upper part and the detector exposure time in the lower part these are the blue intervals:

The effective time includes a number $c$ (possibly none, $c = 0$) of full open times $D_1$ and potentially two fractions of $D_1$ that depend on whether $T_s(x)$ and/or $T_s(x) + E$ fall into the periods where the shutter is open. Because the start of the integration time is not synchronized with the shutter open time $T_{0s}$—see Section 8.1—, the effective time is a basically random function of the difference $T_0 - T_{0s}$ (modulo $D_1 + D_0$). The integer number of shutter periods is the floor function

$$c \equiv \lfloor \frac{E}{D_0 + D_1} \rfloor.$$  \hspace{1cm} (17)

The effective integration time is

$$E_s(x) = (c + \epsilon(x))D_1$$  \hspace{1cm} (18)

where $0 \leq \epsilon(x) \leq 1$ is the sum of two potential fractional pieces of $D_1$ covered at the start and/or end of the exposure. $E_s$ is a periodic trapezoidal function of $x$ with a randomized offset depending on $T_0 - T_{0s}$. An example with $D_1 = 0.12s$, $D_0 = 4D_1$, $E = 4$ s and $T_c = 1.4$ s looks as follows:

Here $c = \lfloor 4/(5 \times 0.12) \rfloor = 6$ and $E_s(x)$ switches between $6D_1$ and $7D_1$. For long exposures,
\[ E \gg D_0 + D_1, \; c \gg 1 \] and the wiggles introduced by a lack of interlock between shutter open and exposure start times become unimportant.

The influence of the shutter can be corrected (as a correction factor for photometry) in a pipeline if the shutter phases are logged in a fashion similar to the GEIRS readout time stamps.

### 7.8 Bright Sources, High Speed

If the illumination on the detector is faint, the fundamental means to adjust to the basially fixed detector gain is prolongation of the integration time. If on the contrary the illumination on the detector is too strong, there is only a limited set of tools to avoid detector saturation and the associated memory/persistence effects—because the minimum integration time is rigidly limited by the fixed number of channels that are read in parallel and by the maximum 800 kHz speed of ADC conversions. From the point of view of the GEIRS control model, these are the prospective tuning parameters:

1. Roughly a factor of 5 in speed is available by clocking faster, which means decreasing the default pixel read time (typically 10,000 ns) by roughly a factor of 5, see the \texttt{prd} button in Figure 5 and the \texttt{ptime} command in section 5.3. This is merely restating that the chip’s reference design is at 100 kHz pixel frequencies whereas MPIA’s ADC’s are capable of 800 kHz sampling. This implies that electronic multi-sampling is not used (see the \texttt{roe} command).

2. Skipping pixel lines in the slow direction by hardware windowing (Section 7.6) offers speedup factors of the order of 10 or 30 depending on how much coverage of the detector is needed.

3. Roughly a factor of 2 is gained if not the \texttt{lir} mode with two reads per scan but only a mode with one read per scan is used, for example the \texttt{srr} with only two reads in total. If relative photometry across the detector is not important but only identification of positions on the detector, one might consider the \texttt{sfr} mode which has the advantage of a full-frame reset (avoiding saturation in all areas of the detector) but reads all pixels only once.

4. The voltage of the external bias may be increased (Section 9.1).

5. Taking an idle mode with the most frequent resets is also advantageous to avoid persistence effects (button in Figure 5 and the \texttt{idemode} command). Note that for a \texttt{srr} mode with two reads the \texttt{ReadWoConv} may be faster than the default \texttt{Lir} idle mode, because the associated cycle time may be slower if the integration times are short anyway. The \texttt{Reset} idle mode is the fastest one offered.

6. If the saturating regions on the detector are a few, and the problem at hand is rather a problem of large contrast through the areal regions, some detector types and instruments offer to mask these (i.e., reset them frequently) with the \texttt{srre} mode (Section 5.6.2).

In summary, going high-speed means primarily using subwindowing with small windows, but perhaps also increasing the pixel rate (at the cost of higher noise), disabling all on-line FITS activities, using sample-up-the-ramp modes\textsuperscript{45} or even reducing logging. That sort of package options of commands looks like:

\textsuperscript{44}For the AIP setup where the 64 channels of two Hawaii-2RG are fed into each fiber the limit is actually 590 kHz set by the clocking of the serialization [8].

\textsuperscript{45}with the disadvantage that one needs a pipeline that subtracts consecutive pairs of frames
sfdump off
autosave off
satcheck off
subwin clear
subwin SW 1 777 999 64 64
subwin auto on
log all 1
roe ems 1
roe pread 2000
# disable intermediate image calculation: show single frames
# (actuate the lower left button in the image GUI...)
put DISP_FRAMEFLAG 1
# ensure fastest frame rate is used...
itime 0
read
sync
# save the individual frames as a fits cube...
save -1 -S

Note that the display (Section 4.3.3) is artificially slowed down to roughly one update each second, skipping intermediate frames if they arrive faster. Saving the frames as an image cube and reviewing these slices with other tools may be useful, or clicking through the single frames with the − and + buttons of the GUI after the exposure ended.

Table 1 shows image cycle times for the \texttt{lir} mode and frame cycle times for the \texttt{srr 10} mode measured with a spare LUCI-ROE with GEIRS 756M-48.

- If pixel read cycle times less than 2000 ns are chosen, the ROE chain may start to drop pixels (because the ADC’s start to drop end-of-conversion signals), and this type of instability is indicated with the ∼ signs in the column of the time differences.

- If the user’s window size falls below 64 pixels (the channel widths) also in the horizontal direction (horizontal in the Hawaii chip’s standard coordinate system), the frame rate starts to become proportional to the window’s area, not just the projected edge length—because the patterns are designed to start skipping pixels also in the horizontal direction if they can.

- In all cases the predicted/calculated cycle times are well within a percent of the jitter in the frame arrival times measured on the workstation.

- The image rates of the correlated double read (\texttt{lir}) are half the frame rates of the non-destructive reads (\texttt{srr},...).

All these measurements are using the 32-channel standard readout mode, which means that the number of pixel data forwarded from the ROE to the workstation is actually larger than product of the pixel counts of the window’s edges [8]. As a guideline we can say that for a 128 × 128 area image rates of 30 Hz are achievable by tuning the pixel clock to 2000 ns, but frame rates of 800 Hz are achievable in small 16 × 16 subwindows.
<table>
<thead>
<tr>
<th>ctype</th>
<th>ems</th>
<th>subwin corner</th>
<th>subwin size</th>
<th>pread (ns)</th>
<th>$\Delta T$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lir</td>
<td>4</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>10000</td>
<td>0.157</td>
</tr>
<tr>
<td>lir</td>
<td>1</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>10000</td>
<td>0.171</td>
</tr>
<tr>
<td>lir</td>
<td>1</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>9000</td>
<td>0.155</td>
</tr>
<tr>
<td>lir</td>
<td>1</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>8000</td>
<td>0.139</td>
</tr>
<tr>
<td>lir</td>
<td>1</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>6000</td>
<td>0.103</td>
</tr>
<tr>
<td>lir</td>
<td>1</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>4000</td>
<td>0.071</td>
</tr>
<tr>
<td>lir</td>
<td>1</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>2000</td>
<td>0.03535</td>
</tr>
<tr>
<td>lir</td>
<td>1</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>1500</td>
<td>$\sim$</td>
</tr>
<tr>
<td>lir</td>
<td>1</td>
<td>700 $\times$ 700</td>
<td>16 $\times$ 16</td>
<td>6000</td>
<td>0.00434</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>10000</td>
<td>0.0858</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>8000</td>
<td>0.0696</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>4000</td>
<td>0.03555</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>2000</td>
<td>0.0177</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>128 $\times$ 128</td>
<td>1500</td>
<td>$\sim$</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>64 $\times$ 64</td>
<td>10000</td>
<td>0.0430</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>64 $\times$ 64</td>
<td>8000</td>
<td>0.0348</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>64 $\times$ 64</td>
<td>7000</td>
<td>0.0300</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>64 $\times$ 64</td>
<td>6000</td>
<td>0.0259</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>64 $\times$ 64</td>
<td>4000</td>
<td>0.0178</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>64 $\times$ 64</td>
<td>2000</td>
<td>$\sim$ 0.0089</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>32 $\times$ 32</td>
<td>4000</td>
<td>0.00515</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>32 $\times$ 32</td>
<td>2000</td>
<td>0.00271</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>16 $\times$ 16</td>
<td>6000</td>
<td>0.00222</td>
</tr>
<tr>
<td>srr</td>
<td>10</td>
<td>700 $\times$ 700</td>
<td>16 $\times$ 16</td>
<td>3000</td>
<td>0.00123</td>
</tr>
</tbody>
</table>

Table 1: Image and frame rates as a function of window size and pixel read parameter measured with a Hawaii2 RG LUCI setup.
7.9 Time between Reset and First Read

The internal analysis of the timing is dumped by GEIRS into the file \texttt{$\$\text{CAMTMP/timing_cmds.log}$}. This file is overwritten each time the clocking patterns are changed, which means, the current contents reflects whatever was the most recent readout/idle mode used by the operator. This is the primary source of what GEIRS accumulates while adding its patterns.

GEIRS basic pattern modes use a line reset followed by clocking through the 64 pixels of each channel (along the “fast” direction, “fast” aligns with the vertical in the LUCI and CARMENES images, with the horizontal in PANIC images). So the distance between reset and readout is a linear function of pixel distance to the line of its channel border. Whether that border is up, down, left or right from the pixel depends on the image flip/rotation parameters (Section 3.2), and for multi-chip cameras also on the detector number in the mosaic.

As an estimate, take the cycle time of the LIR mode (5.01 secs for LUCI) as shown in the cycT of the \texttt{controls} GUI; it consists of 2 “ramps” (each a read-reset-read), which gives 2.5 secs for each ramp. Divide through 2048 lines along the slow direction to get 1.22 ms for each detector line, and divide through the 64 pixels of the channel to get 19 $\mu$s per pixel. (We don’t divide through $2048 = 32 \times 64$ pixels along the fast direction because the 32 channels are handled concurrently by the 32 ADC’s.) Because the 19 $\mu$s correspond to 2 reads and one reset, the distance between the reset and the read is only half of this, 9.5 $\mu$s.

The standard timing currently uses 800 kHz ADC’s, with a 4-fold sampling (FITS keyword \texttt{EMSAMP}, Section 6.4), which is 200 kHz effectively for each value submitted; that is packed into the second phase of a base clock, which gives an observed 100 kHz pixel speed, 10 $\mu$s. That is the 10000 ns reported in the \texttt{prd: (pixel read)} field of the \texttt{controls} GUI, Figure 5. Note that there is a discrepancy of 0.5 $\mu$s per pixel between this timing and the estimate of the previous paragraph.

This ought to be the same for the MER, which also uses line resets. Of course these delays between reset and read cancel in the base modes, as the two frame reads are subtracted pixel-by-pixel to define the image. So the exposure time does not depend on the distance to the channel boundary.

Note that there are also full-frame reset modes, SRR modes... [8].
8 COORDINATE SYSTEMS

8.1 Exposure Start

We summarize the causes of delays between sending the `start` command and receiving data with GEIRS:

1. The standard idle mode loops through the detector lines, resets them, but does not trigger ADC’s. The start command does not interrupt this idle mode but uses a well-defined break point at the “last” detector lines to leave these loops. [The “break” idle modes do not wait until the pattern program reaches the break point, but they lead to well known biases (steps) at the lines where the loops are exited.] Because the start command is basically uncorrelated in time with the phases in the idle loop, a delay of typically up to the full frame readout will occur.

2. If some srre reset windows are modified, an entirely new pattern will be downloaded to the ROE, which (as a function of internet latencies, number of reset windows etc.) will typically lead to a delay of 10 to 20 seconds.

3. The `geirs_dropcaches(1)` automatism will be invoked if the start of the readout realizes that the free memory has dropped below half of the full RAM. Experience with a 32 GB computer shows that this will lead to a delay of a few seconds.

4. Before each readout, GEIRS allocates “kernel” memory with the aid of the PLX library, 8 MB for each $2k \times 2k$ detector chip, in chunks which are some fraction of the maximum of 4 MB. If this does not succeed right away (usually caused by fragmentation of the slub tables, which unfortunately is correlated with the file caching mechanism), GEIRS attempts this multiple times with intermediate waits of the order of half a second. This adds an essentially unpredictable delay to each start. This behaviour can to some degree be manipulated by changing default value of the `geirs_dropcaches(1)` and with the `put` command the subdivision factor `kmalloc_split` in the shared memory database.

Note that a static single allocation of that memory is not implemented because the daisy chain of the DMA depends on the size and number of subwindows used by the observer, and this set of parameters can change prior to each `read`, and is basically unpredictable.

---

46 a slightly higher mark is chosen for CARMENES
47 which actually does not exist under Linuxes...
9 TROUBLE-SHOOTING

9.1 ROE Interface

1. Problem: No data appear and the main screen of Figure 12 remains gray after a read has been initiated and the associated exposure time is over. GEIRS emits errors of the sort that init returns error codes equivalent to timeouts while trying to connect to the camera. Check list: First check that the rack of the readout electronics and all intermediate switches, hubs & c are powered on. Check that the yellowish LED of the ROE board flickers at least once or twice when the ROE is powered on. Then ensure that the shell variable CAMPORT (Section 3.2) in the scripts GENERIC is correct, including the TCP marker and the port number, that the readout electronics has actually been set up to listen to that address [4], and that a ping command with that (numerical) address from the GEIRS computer gets an answer from the readout electronics.

   ping 192.168.90.20

   If messages of the sort

   INFO MPIA-ROE3 reset - '33 8 0 1'
   INFO Seen ROE3 rocon 'DETFPGA' version '3 1 7 5'
   INFO Seen ROE3 rocon 'ADCFPGA' version '3 0 2 2'

   appear when GEIRS is started up, the network interface between the host computer and the ROE is working.

   Unfortunately starting GEIRS with the Java GUI Figure 4 never generates output on the Linux standard output, so that test is not available if that method of starting is used.

   If you powered the ROE on after starting GEIRS, GEIRS will be unaware of the presence of the ROE, and the ROE will not host any patterns. The options are

   (a) load a standard pattern to the ROE with the File→Init/Reboot ROE menue of the controls menue (Figure 5).

   (b) quit GEIRS and restart it while the ROE is on.

   If it is impossible to build up an Ethernet connection to the ROE, and if no spare ROE is available, an alternative connection is available if the RS232 connection still works, as sketched up in Figure 25. The command exchange via the RS232 serial interface is estimated to be a factor one hundred or one thousand slower than via the Ethernet, and therefore impractical for standard operation.

2. Problem: No data appear and the main screen of Figure 12 remains gray. Solution: If messages of the

   (E_ptimeout=21) timeout on OPTPCI interface

   kind appear in the GEIRS logs because GEIRS waits longer than expected for the video data, the fiber connections are disrupted, or the more fundamental problem of communication failure of the command channel to the ROE of the previous item exists.

48 Unfortunately starting GEIRS with the Java GUI Figure 4 never generates output on the Linux standard output, so that test is not available if that method of starting is used.
49 GEIRS does not poll thr ROE status
Internet

192.x.y.z:400u

MOXA Nport 5410

4001 4002 4003 4004

RS232

Figure 25: An engineering configuration where the ROE is not reached by its Ethernet port as in Figure 1 but by its serial port mediated through some server. In this case the CAMPORT variable in the software needs to be set to the Ethernet address of the intermediate server—in the MOXA case the port numbers are from 4001 upwards, \( u \geq 1 \).

3. Problem: The main screen of Figure 12 turns black, i.e., the ADU values received via the fibers are zero. For instruments with single chips (LN, LUCI, NTE) check that the two fiber heads have not been swapped on the OPTPCI side where they enter the workstation (or to the same effect, on the ROE side where they enter the ROE rack). For CARMENES, swapping fibers leads to a right-left swap of the two detector pairs (i.e., exchange of pair images in the display and in the FITS files). The OPTPCI board offers plugs for two fiber pairs on the rear side of the workstation that receives the detector data. The basic industrial application of this type of hardware/connector is bi-directional network data transfer, but the MPIA ROE uses them only for one-way detector image data transport of the 16-bit data from the ROE into the workstation, so two of the plugs are never used and usually covered by some dust cover (Figure 26) [29]. Effectively a single fiber pair connects an ROE and an OPTPCI at the workstation. Both fiber cores are used for data transfer for PANIC and CARMENES, but only one core for data transfer for LN, LUCI and NTE. Because the equivalent selection of plugs is to be made on the ROE side, this gives a probably of 3/4 for LN, LUCI and NTE to get no data and a probably of 1/2 for PANIC and CARMENES to get swapped images if fibers are plugged in at random.

A red LED on the OPTPCIe board indicates that the fiber on that port is disrupted. Instruments like LN, LUCI and NTE can live with one working fiber—if that is the one configured—but the other two instruments need both.

Ensure that the OPTPCI driver is compiled and installed (\texttt{lsmod} as in Section 2.1.9).

Run any of the tests in the appendix of the pattern manual [8] to ensure that data from that board’s data generator generate stripes in the GEIRS display.
If more than one OPTPCI is plugged into the computer, check the correct DATAINPORT1/DATAINPORT2 setup in scripts/GENERIC, otherwise make sure this is the /dev/plx-00//dev/plx-01 pair.50

4. Problem: GEIRS says

ERROR (91) opening line: ’(E_camline=91)’

Solution: GEIRS cannot open a socket via the Internet to the readout electronics. This indicates errors as already discussed above. Either the ROE is not powered on, or the GEIRS configuration of the CAMPORT (in the GENERIC startup script) does not match the ROE’s actual IP. For debugging note that GEIRS displays the current value at startup with a line of the format

Setting ROE port to tcp://192.168.3.xxx:4000

on the Linux shell and also in the RO-Electronic field of the GUI in Figure 3. For a quick temporary check whether the IP address is the culprit, one can either use the engineering GUI in Figure 4, or set the environment variable CAMPORT before starting GEIRS (because, as mentioned in Section 3.2, the startup script does not override an existing value).

5. Problem: The cycle time stays at zero seconds in the GUI. Potential causes:

(a) the value of the environment variable CAMROE_REV (Sec. 3.2) defined in the scripts/GENERIC file points to a wrong or non-existing directory.

(b) GEIRS never got the init camera command (Section 5.3). This command is actually submitted by clicking all or OK in the startup GUI, Figure 3. However, if the two main processes (shmmanager and cmdServer) and/or the other processes (control, disp) are called directly from the UNIX/Linux command line without using this interface, the command may not have been issued. This can be submitted for example with the Re-init ROElec submenu of Figure 5.

(c) The internet connection to the ROE does not work (see above). Occasionally this is caused by temporary congestion (and the error log monitor will display timeouts) and sending the patterns again to the ROE—with the Re-init ROElec button of Figure 5 or the init camera command of Section 5.3—will remove the problem.

(d) GEIRS was not started in simulation mode but the ROE does not respond—for any of the reasons described in Section 9.1.

(e) The rotype has been set to dgen (the OPTPCI data generator). Execute

status rotype

in the GEIRS shell to see whether this is the case and set it back with

rotype plx

(f) The environment variable CAMERA was set to a string before starting the instrument and the start.* command used a different name. In this case delete the environment variable before using the start.* command:

export CAMERA=

50This is currently the case on one of the two LUCI computers at the LBT and on elablx01 and irws2 at the MPIA.
6. Problem: The detector images appear to be basically flat zeros, because the raw single frames (prior to the subtraction/correlation) are highly saturated close to the maximum of 65,535 counts. (Switch to single frame display with the current button in the display in Figure 12 to look at these counts.) Solution: This has been observed if the CARMENES detector is operated at rather warm or ambient temperatures. This can be improved by rising the external bias voltage applied to the chip(s) from the default value ($\approx 2.2$ V) to values near 2.5 or 2.6 V. The current value is revealed by the command

\texttt{bias}

The value would be altered with the \texttt{bias} command (Section 5.3) in the style of

\texttt{bias det1 extbias -V 2.55}

and if there is more than one Hawaii-chip in the instrument for the others by increasing the index up to 2 for CARMENES and up to 4 for PANIC:

\texttt{bias det2 extbias -V 2.55}
\texttt{bias det3 extbias -V 2.55}
\texttt{bias det4 extbias -V 2.55}

The same effect with the opposite sign has been observed with the LN detector after cooling down the entire optics and just switching on the ROE: the single frames may have pixel values close to zero ADU’s. The effect vanished slowly within hours afterwards when the ROE was switched on\textsuperscript{51}. During Com-3 of LN for example, this needed of the order of 4\textsuperscript{1\frac{1}{2}} hours before reasonable images were received from the ROE. In this case one could lower temporarily the external bias in steps of 0.05 to a value near 0 Volt for a first visual check of the LN detector image,

\texttt{bias det1 extbias -V 1.0}

each time followed by a \texttt{read} to see whether some noisy image appears.

The voltages remain until they are either overwritten with another \texttt{bias} command or until GEIRS resets the electronics (Button \texttt{Re-init ROElec} in Figure 5) or is shut down and restarted.

For instruments which use only one of the fibers (LUCI, LN or both NTE), zeros may also mean that the fibers are crossed and GEIRS scans data from the wrong fiber of the pair. As a first debugging aid in this case, shut down and re-start GEIRS with \texttt{geirs start} and change the last, rightmost digit in the \texttt{plx} number in the \texttt{DATAINPORT} in the GUI (Figure 4). If this allows to get images, swap the fibers at the source (ROE at the dewar) or the destination (OPTPCI board of the workstation) and return to the normal operation where the last digit is zero.

For instruments which use both fibers (PANIC and CARMENES), swapped fibers mean that the associated sub-images are swapped, which is checked by verifying that the characteristic bad-pixel patterns are at their expected places.

7. Error messages of the form

\textsuperscript{51}which warms up the pre-amplifiers that have some minimum operating temperature
libplxmpia.c:233: [plx_find_device] ERROR) Error in Plx device found (u=2/ch=0): ffff ffff

or

ERROR Error: plx_find_device: 'PLX ApiError 516 - ApiNoActiveDriver'

mean that the driver for the board that interfaces with the RoCon fiber optics has died or not been installed. This should be fixed by loading the driver at boot time—see Section 2.1.9. One can temporarily fix this by by executing

cd $CAMHOME/scripts
sudo plxstartup

but this means the same problem reappears each time the workstation is rebooted.

8. Problem: Error messages of the form

Unable to allocate Memory for Buffer...

appear and no frames are read. Workaround: This indicates that the driver is not capable of allocating the kernel memory for the next exposure. This typically arises if the last lines in the INSTALL script (Section 2.2.1) were not executed, for example because the user installing GEIRS had insufficient rights. The only known solution for unprivileged users is to shut down GEIRS and to reboot the computer. The advice is to use only the standard tools for shutting down GEIRS as documented in this manual, never to kill the geirs_rdbase process from the operating system while is uses kernel buffers for reading (i.e., while a read command is active).

9. Problem: Communication with the ROE times out with messages like ERROR 23 Command 'ctype srr 4' returned errorcode = 23: (E_timeout=23) timeout from camera (control line). This is occasionally caused by very high traffic in the network. The associated timeout is set to 5 seconds generally and to 10 seconds at the MPIA network in camsend.h and can be increased (followed by recompilation with make install) if this is a permanent problem.

10. Problem: The ROE lamps die after a while. Solution: When GEIRS is started, a first action in the patterns downloaded to the ROE is to switch off as many of the ROE’s lamps as possible. The reason is that the standard operation of the ROE is in telescope domes where permanent light polution near the telescopes is undesirable. If you need this blinking for debugging purposes, put the include ledoff in the file registers.* in the pttrens directory into a comment, which means, insert a sharp (#) at the front of the line. Some of the ROE lamps, the Ethernet RJ45 connector and the power unit, are not under that type of remote software control; these need to be taped to mute them.

9.2 Software

1. Problem: The commands

plxshutdown
plxstartup
don’t load the PLX driver relevant to the chip that is on the OPTPCI board.

Solution: apparently the driver was not compiled. Each time the operating system has been patched with

zypper up

and a new kernel appears in /usr/src, recompile the driver in the following order as root:

reboot now # reboot the computer so the new kernel so the new kernel version is recognized
cd /home/.../GEIRS/trunk-r.... # move into the GEIRS directory with the installation script
./INSTALL.plx # recompile and install the driver
cd ../scripts ; ./plxstartup # load the new driver

Then recompile all GEIRS versions to link with the new driver under the usual login account:

reboot now # reboot the computer so the new kernel so the new kernel version is recognized
cd ${HOME}/GEIRS/trunk-r.... # move into the GEIRS directory with the installation script
make -f Makefile clean
make -f Makefile install

2. Problem: the startup command does not produce the GUI of Figure 3.

Solution: you may have modified your window manager such that new windows are not popping up in the front layer of the window stack. Search through the stack of windows to detect it if hidden/covered by other windows.

3. Problem: An attempt to start GEIRS does not open the GUI of Figure 3, but instead it just shows some process list of the operating system with processes like geirs_shmmanager, geirs_cmdServer and says that some shmsocket exists. There is some output that says cannot attach info page.

Solution: This means that GEIRS is already/still running, which means you or someone else with access to the user account has started it and did not shut it down. Ring up all people in that user class and ask them whether they are still operating the readout electronics, and figure out with

journalctl

when the last action of this session took place. If you are absolutely sure that there is no harm done by forcing that application to quit, you may call

ps -elf | fgrep geirs
geirs_cleanup
ps -elf | fgrep geirs

on the Linux shell to kill that GEIRS session and then try again to start a new one.

4. From time to time it can happen that a process hangs. Mostly you can simply kill the hanging process. Some commands are prepared for this, as documented in the command list (Section 5.3):
• kill read terminates a read command
• kill save terminates a save command
• kill telescope terminates any command to the telescope
• kill wheel terminates any command for the filter wheels

Type these commands in the interpreter window where you have started the GUI, not into the UNIX/Linux shell (where it refers to processes of the operating system).

5. GEIRS does not start, and some logs with the operator’s name and some process names appear. Solution: the previous GEIRS session was not closed and remains active under the same Unix account. Run `geirs_clean up -a`, then `ps -u $USER` | `fgrep geirs` to ensure all GEIRS processes have died, and restart again.

It seems that this situation may arise if some process send a command to the GEIRS shell and terminated or was killed before it received the answer.

6. Problem: The GUI does not open, and there is a message like `can’t allocate info page`. Solution: Type `geirs_clean up -a` before you start the GUI. This program deletes shared memory pages left over by the same Linux/Unix user from a previous session and shared memory sockets `tmp/shmsocket`. The underlying problem is often that GEIRS was not properly shutdown, for example because the computer rebooted due to power failures. On some computers running openSUSE 13.2 this rebooting happens when sleep (suspend to RAM) does not wake up as intended.

7. Problem: Anything seems to work well but there are no stars. Solution: Check the third button in the display window Figure 12 for the image selection back to current so the images are updated.

8. Problem: The GUI in Figure 5 and the associated commands `crep` and `ctype` accept only small numbers; the GUI sets values back to smaller ones, and the status shown by the commands (without parameters) also shows smaller counts than requested. Pseudo-Solution: Increase the `CAMSHMSZ` parameter in `scripts/GENERIC` (section 3.2) and/or the limit set by the operating system (section 2.5.5) before starting GEIRS. This will usually not work because the standard parameters are already set limits measured with respect to the available RAM. The general solution is to split the exposures into smaller packets so each of them fits into the margins.

9. Problem: When saveing, a FITS filename and a message of the form `save: (E fopen=48) could not open file` appear. Solution: Either

   • the disk is full (tested with `df -h`) or
   • the GEIRS user does not have write permission on the current data directory. This is revealed for example if one attempts to create an empty dummy test file in the style of `touch junk.txt` in that directory. A workaround then is to create a new directory with the `SavePath` button of Figure 5 for future use, which will by default be created with the corresponding write+executable permissions, or to use `mkdir` of the Linux shell in conjunction with a `set savepath` of the GEIRS shell, or to obtain modifying privileges of the intended data directory and execute `chmod g+wx` on this if owned by another user.

\footnote{The exception is the two LUCI’s where the assumption is made that binocular mode requires two GEIRS sessions; so there is room by a factor of two then.}
Keep in mind that GEIRS does not overwrite existing FITS files (with the exception of those created via the `sfdump` command or if explicitly permitted via the `clobber` command or with the `-c` of the `save` command). This is important if operators set explicit file names with each `save` command instead of relying on the automated file selection.

10. Problem: the `ctype sre` responds with an error of the format `ERROR Too large tblindex 256 of max. 256 in dettable=2`. Solution: reduce the number of reset windows defined in the configuration file. The current limit is near 80 windows on each individual detector chip.

11. Problem: `geirs_clean up` responds with a message of the from *If ‘clean up‘ is not a typo...*. Solution: expand the `PATH` variable as described in Section 2.5.2.

12. Problem: After the read process finished the `save` button in the controls GUI in Figure 5 stays yellow. Solution: This happens for example if automated save processes fail due to a disk full state. This is in particular a thread on the CARMENES computer with only 180 GB of disk where single frames saving with the `sfdump` interface is on by default. (This is equivalent to less than 4 hours observing time at a maximum speed of 1 frame each 1.3 seconds.)

13. Problem: After calling `read`, GEIRS and other processes seem to hang for up to 30 seconds. Solution: Ensure that the installation is complete, including the last lines of the `INSTALL` file concerning file owners and permissions.

14. Problem: The `read` of Linc-Nirvana never produces any frames or images, not even with the data generator of the OPTPCI nor if GEIRS is started in simulation. Solution: Linc-Nirvana may be configured to start a rewind of the derotator stage of the detector prior to each readout. If the associated motor server does not finish this rewinding, GEIRS may wait forever in that phase without actually forwarding the read command to the ROE in the next phase. The simplest workaround is to insert a `exit 0` very early in the file `~/GEIRS/scripts/QueueEFiles` such that the bash script that tries to initiate the motor is effectively not doing anything. An alternative is to uncheck the `-Q` flag in the controls GUI before starting the `read`, which also skips calling the rewinder script. A third option is to stop the server that is running the motor such that requests for the rewinding are quickly rejected:

```
lirks@lirks:> rcbasdard.sh stop lirks.moe.derot-svr
```

Perhaps (not tested) switching the power of CRY-MOT-1 off in the `lsys.cab.ps-svr.GUI.sh` has the equivalent effect.

15. Problem: The single frame dumps of CARMENES seem to miss some frames in LIR mode. Solution: Operate GEIRS in accordance with standard parameter ranges. In detail:

- Avoid disk full states.
- Do not abort the reads in correlated double sampling modes before the second frame is read. The first stage pipeline will reject processing output of that kind with error messages.

---

53 The obvious disadvantage is that the casual observer forgets to undo that change later on.
• Do not impose heavy disk I/O loads besides GEIRS’s own automated guide mode dumps unless you are sure that your disk writing speed exceeds the throughput of the 16 MB per frame by at least a factor or two. GEIRS drops single frame dumps if it cannot keep up with the frame rate.\textsuperscript{54}

• Avoid \texttt{crep} parameters larger than one in conjunction with the \texttt{ctype lir}. This will generate the raw frames but the first stage pipeline (and further processing) will discard any images but the last one.

• Because the FITS name convention for CARMENES uses time stamps rounded to full seconds, GEIRS starts to drop frames if the frame frequency becomes larger than one frame per second. This happens for example if subwindowing is used or the pixel read time is reduced. To store all frames anyway, use an explicit \texttt{save} with the single frame option (although these will not be recognized by the first stage pipeline).

16. Problem: Macros with \texttt{crep 30} and \texttt{ctype srr 45} miss frames with CARMENES. GEIRS stores only 33 but not the expected 45 frames. Solution: The RAM requirement for the frames would be $30 \times 45 \times 16$ MB, which is larger than then 16 GB of (half of the total RAM) on the NIR computer, see Section 2.5.5 and the \texttt{CAMSHMSZ} parameter in Section 3.2. Make sure that the arithmetic product of the repetition value by the number of frames along the ramp is less than 800; if needed split exposures into multiple \texttt{reads} to stay below that limit for each single \texttt{read}.

17. It has been reported for LUCI that one can press the Endless button of the control GUI (Figure 5) and that the other LUCI control GUI reacts synchronously, although such a cross-talk is obviously not desired or expected. The most likely cause is that different people are using the instrument at the same time (under the same Linux account) and are just watching each other’s actions on the local displays. This is supported because the number of control GUI’s is not limited and everybody can join a GEIRS session for example by opening another GUI with \texttt{snd\_luci\{12\}\_new control}, can open further shells or can send one-shot commands (Section 3.1). It is recommended to scan the command logs (drop-down menu in Figure 5) and figure out whether all the commands appearing there are actually yours, and if not check who else might be operating the DCS. We realize that stealing sessions is a quite common operator pattern and that adding limits on session counts would lead to restrictions which are not desired.

18. Problem: pressing the Modules→debug log monitor does not open anything or just shows a GUI for a split second that closes immediately. Solution: check that

\begin{verbatim}
journalctl
\end{verbatim}

shows at least a few log lines. If this responds with \ldots due to insufficient permissions, check that the directory \\
/\texttt{var/log/journal} exists. If not, switch to persistent (and per-user) journaling by changing

\begin{verbatim}
#Storage=auto
\end{verbatim}

to

\begin{verbatim}
Storage=persistent
\end{verbatim}

\textsuperscript{54}This is a deliberate design choice to support smooth processing with the first stage pipeline.
in `/etc/systemd/journald.conf` and reboot or restart the journal with

```
systemctl restart systemd-journald
```

as superuser.

19. Problem: after pressing `read` one can save the exposure to files and there are no complaints in the log monitor, but the real-time display is not updated. Solution: If the range of the actual pixel (frame or image) data is narrow, the cut levels may be too narrow to let any of the new pixels pass, and the algorithm in the real-time display does not take them as a trigger to update the display. This may for example happen if a warm detector is read out, where the image is “flat.” In this case switch to the 100 min-max selection in the menu of the display, Figure 12.

### 9.3 Operating System

1. Problem: After `start* -gui` time GEIRS complains that `DISPLAY` is not set.\(^{55}\) Solution: For all steps of establishing tunnels and using `ssh` to login to the GEIRS workstation, use the `-X` option as documented `ssh(1)`.

   In addition, if commands are run through a `sudo`

   - the `env.keep` list of variables in `/etc/sudoers` ought include the `DISPLAY` variable to forward the variable from the user who runs the `sudo` to the effective user after the `sudo`.
   - the effective new user needs to be authenticated with the information of (basically) `.Xdefaults` of the user who runs `sudo`, see [11].

2. Problem: the startup scripts prints some dots and then says `Cannot connect to shmmanager`. Solution: The shared memory allowances set in Section 2.4.1 are too small, so the shared memory manager does not start.

3. Problem: the command `geirs_cleanup` is not found. Solution: Add `SCAMHOME/scripts` to the `PATH` as described in Section 2.5.2.

4. Problem: GEIRS fails to open its GUIs claiming that it cannot allocate its color maps. Solution: close some of the other graphics intense programs that are currently running on the same display and/or invest into contemporate hardware.

5. Problem: the compilation of the GEIRS source enters an infinite loop with `recheck` messages. Solution: this may happen if the time stamps of the source code bundle (which has been created on another computer) are severely out-of-sync with the clock on the computer where GEIRS is compiled. Use `date` to check that the system clock is reasonable on the GEIRS computer and connect the computer with a NTP server if it is not.

### 9.4 Motor Interface

This has no relevance to instruments besides PANIC.

\(^{55}\)Of course this has nothing to do with GEIRS.
1. Problem: A lot of ERROR with MOT2-answer... timeout... before getting complete answer appear. Solution: check that cables are connected to the auxiliary rack and that the rack is powered on and connected to the Ethernet. If this does not seem the cause of the problem, restart GEIRS to start a freshly synchronized command exchange with the MoCon.

9.5 External Software

(Of course, these things have nothing to do with GEIRS.)

1. If `fv` displays in `pow` a transparent image, the `kde4` allows to change this behavior by either `<Shift><Alt><F12>` momentarily, or by disabling these effects in the Application Launcher Menu in Personal Settings (Configure Desktop) → Workspace Appearance and Behavior → Desktop Effects and unchecking Enable desktop effects at startup.
Figure 26: Fiber connectors of the OPTPCI board on the rear side of the workstation. Note that depending on which riser board is used on the computer—typically on racks of 2U height—, the entire configuration is rotated. At the upper left we sketch the vertical LUCI, MPIA and AIP installation (host: Transtec Opteron with 2 Opteron-8C 6220, Opteron-4C 2382, Tower PC respectively), in the lower left the horizontal LN installation (host: Dell R515), at the right the Figure 30 of the ROE manual [29] or PANIC (host: PowerEdge R720).
9.6 Recent Changes

A summary of the GEIRS changes between version 780M-21
(as described in my e-mail of Feb 15, 2019) and the current
version 789M-7 (available on ftp://ftp.mpia-hd.mpg.de/pub/mathar/ )
from the application/operator point of view looks as follows:

- Hawaii-4 RG support (for the PANIC upgrade) has been debugged
to the extend that at least the Multiplexer is read out
as expected with the standard 64-channels readout.
The selection of PANIC at startup now requires a single Hawaii-4 RG
chip (or multiplexer or simulation); the old 2x2 Hawaii-2RG variant is no
longer supported and moved to a camera Aip. The distinction between
DETSIZE and DATASIZE in the FITS headers has been dropped for Hawaii-4RG
(assuming that the 4 ref pixels will be used like standard pixels).

- At startup, the default FITS output file name is set to
a device name and time stamp (not aa0001). The default data
(sub)directory is derived by subtracting 7 hrs from the local time
at the observatory and using the associated date as the name.

- The PANIC telescope menu (probably never used ?) has also
been ported from X11 to Java. Replacement of X11 GUIs
by Java GUIs is complete.
Availability of a Java compiler is now mandatory.
Font sizes can be selected in the startup menu.
The left-mouse click and drag is used to compute a histogram
of the data in a rectangle of the real-time display (no longer
a method to adapt cut levels and saturation levels).

- All logging (with the exception of PANIC temperature logs
and its file saved logs) now go into syslog, no longer
into log files. The log rotator script and the log filters
in the menu are adapted. Jitter log files are no longer created.

- Due to restricting changes of the Oracle JDK licensing
as introduced for their JDK 12, , heading towards commercializing Java,
development is now based on the default Linux open Java library which allows
use in our telescopes.
Support for Ubuntu 16.xx has been added, mostly related to the autotools variables; further updates of the NTEimg and NTEspec cameras (which are currently just Luci1/2 equivalents). At MPIA we will not have a OPTCI board plugged into an Ubuntu-computer, so tests of that operating system and its 5.xx kernels with the PLX driver are postponed...

- Support for PLX device drivers <= 7.10 has been dropped. The platform is now PLX SDK 8.00 (which supports 4.xx and 5.xx Linux kernels).

- All locally compiled external packages (cfitsio, xercesc, iausofa etc) have been continuously upgraded.

- On request a default voltage for Linc-Nirvana has been changed after the detector replace campaign. Voltages for PANIC are now the same as for the CARMENES/LUCI detectors (and remain to be defined).

- Some bug in observing the TMOUT parameter (automatic GEIRS termination after idle period) has been removed.

- A program fits2tiff that converts FITS images to TIFF 6.0 files has been added.

- The parameter NQCHAN (the number of detector channels) is now pushed into the FITS header data (anticipating that readouts with less than the maximum number of channels might be useful in "long" spectroscopic modes...).

- The meaning of the 'fits' command has been changed: it does no longer report a virtual set of keywords that would appear in potential FITS files of the future, but adds FITS lines into the files that will be saved afterwards. [This is another keyword merger channel beyond the previous file-based FITS-keyword gatherer...]

- The nominal beam center in the Linc-Nirvana images has been updated to a value calibrated in Com-9.

- The xserver command has been removed.

- The status scanner and status collect servers have been removed.

- The development platform is openSUSE 15.1.

Developer’s aspects:

- There is some support for engineering type setups where
the number of ROE ADC36-boards is not the full number
required for the chip-type and number. (That was triggered
by the need to simulate the Hawaii-4 RG readout on a test
ROE with only one ADC36 board.)

- Untested variants of 4-channel readout-mode for Hawaii-2
  and 32/16-channel readout-modes for Hawaii-4 RG have been added.

- Once more cleanup of old patterns that are no longer supported
  (other than Hawaii-2, Hawaii-2RG and Hawaii-4RG). (Software coadds,
some readout mode residuals for Omega2000, the CAHA 1.2m,...)

- Further (incomplete) upgrade of the ICE server for Linc-Nirvana.
  This is actually not yet used.

- The interface from the shared
  memory data to the real-time display has been re-factorized
  and is now based on an intermediate data server.

- Almost all patterns which are compatible between instruments
  and detector types have been massively reduced in number by
  using a large amount of symbolic links between the pattern
  subdirectories. (Benefit of an earlier decision not to put
  the patterns in distributed SVN repositories anymore.)

- Compilation of the test subdirectory is also supported on CentOs 7.

The detailed changes can be reviewed with the source code browser in https://svn.mpia.de/
trac/gulli/geirs/browser/src/trunk.

A BEYOND GEIRS

This section adds information on processes, other programs or aspects of the operating system that
are not under GEIRS control nor part of the source distributed by the MPIA.

A.1 Installment of a new ROE IP address

How to change the IP address of the MPIA ReadOutElectronics

A.1.1 Using RS232

Uninstall the ROCon board and set the configuration DIP switches 5 and 7 to ON. Start a terminal
program like PuTTY. Reinstall ROCon board and connect it to your computer using a null modem

56Contribution by U. Mall, 29 Feb 2015
cable. The serial settings are: 9600N81. Power on ReadOutElectronics. You should see a message like this:

33 6 0 4 COS_XC161 V2.16, Jul 11 2007
33 6 0 4 ReadOut Controller V3.00beta, Jan 10 2013
33 6 0 4 System ready...

Now set the IP address (192.168.3.160 for example):

33 30 0 192 168 3 160

Note that there are blanks instead of dots separating the four numbers of the IP address. The new address can be read back after a soft reset (33 8 0), a pushbutton reset or a power on reset:

33 31 0

The ROCon boards responds:

33 31 0 2 192.168.3.160
33 31 0 1

If necessary the subnet mask can be set with:

33 34 0 255 255 255 0

The Subnet mask can be read back after a reset (see above):

33 35 0

Don’t forget to set switch 5 to OFF for regular operation with new IP address.

A.1.2 Using ethernet

In case of configuring via ethernet your computers network adapter has to have an IP address in the same subnet as the ReadOutElectronics. Then you can telnet the ReadOutElectronics on port 4000:

>telnet 192.168.3.167 4000
Trying 192.168.3.167...
Connected to 192.168.3.167.
Escape character is '^[]'.

The ROCon board responds with a message like this:

33 6 0 4 COS_XC161 V2.16, Jul 11 2007
33 6 0 4 ReadOut Controller V3.00beta, Jan 10 2013
33 6 0 4 System ready...
The next step is to login and reserve a module number:

33 21 0 user
33 22 0 mpia
33 23 0

For every command the ROCon board sends acknowledge:

33 21 0 1
33 22 0 1
33 23 0 1

Now setup new IP address (192.168.3.160 for example):

33 30 0 192 168 3 160

Note that there are blanks instead of dots seperating the four numbers of the IP address. The new IP address is activated after a soft reset(33 8 0), a pushbutton reset or a power on reset. After reset your telnet connection is lost. Ensure that your computers network adapter is in the same subnet as the new IP address and reconnect:

>telnet 192.168.3.160 4000
Trying 192.168.3.160...
Connected to 192.168.3.160.
Escape character is '^]'.

If you have done everything right you will see this message:

33 6 0 4 COS_XC161 V2.16, Jul 11 2007
33 6 0 4 ReadOut Controller V3.00beta, Jan 10 2013
33 6 0 4 System ready...

If necessary the subnet mask can be set with:

33 34 0 255 255 255 0

The Subnet mask can be read back after a reset(see above):

33 35 0

A.2 Image Rotation

The two configuration parameters CAM_DETROT90= r and CAM_DETXYFLP= f specify an image transformation \((r, f)\) defined by a rotation by a multiple of 90° \((r = 0, 1, 2, 3)\) followed by an optional image flip of \(f = 0\) (none), \(f = 1\) (right-left) or \(f = 2\) (up-down).
The four choices for CAM\_DETROT90 combined with the three choices for CAM\_DETXYFLIP supply $4 \times 3 = 12$ combinations. This is only half of the $4! = 24$ possible permutations of all 4 corners, because only one of the orders of the two operations is implemented/supported. A closer look shows that each of the rotations followed by a right-left flip can be replaced by a rotation through another $180^\circ$ and a up-down flip: $(3, 2) = (1, 1), (2, 2) = (0, 1), (1, 2) = (3, 1),$ and $(0, 2) = (2, 1)$. So there are not 12 but only 8 image operations available. Those of the 24 that appear to be missing are group operations which would try to generate images where North and South remain not opposite to each other but end up at right angles. The transformation $(r, f)$ is an element of a non-abelian group of order 8, isomorphic to $D_8$, the dihedral group with 8 elements. The group multiplication table is shown in Table 2.

<table>
<thead>
<tr>
<th></th>
<th>(0,0)</th>
<th>(1,0)</th>
<th>(2,0)</th>
<th>(3,0)</th>
<th>(0,1)</th>
<th>(1,1)</th>
<th>(0,2)</th>
<th>(1,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0)</td>
<td>(0,0)</td>
<td>(1,0)</td>
<td>(2,0)</td>
<td>(3,0)</td>
<td>(0,1)</td>
<td>(1,1)</td>
<td>(0,2)</td>
<td>(1,2)</td>
</tr>
<tr>
<td>(1,0)</td>
<td>(0,0)</td>
<td>(1,0)</td>
<td>(2,0)</td>
<td>(3,0)</td>
<td>(0,0)</td>
<td>(1,1)</td>
<td>(0,2)</td>
<td>(1,2)</td>
</tr>
<tr>
<td>(2,0)</td>
<td>(0,0)</td>
<td>(1,0)</td>
<td>(2,0)</td>
<td>(3,0)</td>
<td>(0,0)</td>
<td>(1,1)</td>
<td>(0,2)</td>
<td>(1,2)</td>
</tr>
<tr>
<td>(3,0)</td>
<td>(0,0)</td>
<td>(1,0)</td>
<td>(2,0)</td>
<td>(3,0)</td>
<td>(0,0)</td>
<td>(1,1)</td>
<td>(0,2)</td>
<td>(1,2)</td>
</tr>
<tr>
<td>(0,1)</td>
<td>(0,1)</td>
<td>(1,2)</td>
<td>(0,2)</td>
<td>(1,1)</td>
<td>(0,0)</td>
<td>(3,0)</td>
<td>(2,0)</td>
<td>(1,0)</td>
</tr>
<tr>
<td>(1,1)</td>
<td>(0,1)</td>
<td>(1,2)</td>
<td>(0,2)</td>
<td>(1,1)</td>
<td>(0,0)</td>
<td>(3,0)</td>
<td>(2,0)</td>
<td>(1,0)</td>
</tr>
<tr>
<td>(0,2)</td>
<td>(0,2)</td>
<td>(1,1)</td>
<td>(0,1)</td>
<td>(1,2)</td>
<td>(0,0)</td>
<td>(3,0)</td>
<td>(2,0)</td>
<td>(1,0)</td>
</tr>
<tr>
<td>(1,2)</td>
<td>(0,2)</td>
<td>(1,1)</td>
<td>(0,1)</td>
<td>(1,2)</td>
<td>(0,0)</td>
<td>(3,0)</td>
<td>(2,0)</td>
<td>(1,0)</td>
</tr>
</tbody>
</table>

Table 2: Cayley multiplication table of the group of order 8 constructed with the CAM\_DETROT90 and CAM\_FLIPXY keywords. The operation on the left is executed before the operation on the top.

The 8 group elements are

- the unit element (no change of the image),
- the three pure rotations $(r, 0)$ with $r = 1, 2, 3$—generated by $(1, 0)$ of order 4—,
- the two pure flips $(0, 1)$ and $(0, 2)$—each of order 2—,
- and the two flips along the two diagonals, $(1, 1)$ and $(1, 2)$—each of order 2.

### A.3 Remote Sound

This is a user’s note that has nothing to do with GEIRS; any other means of the local computer network may be implemented as well. It is only of interest if operators need to hear GEIRS sound effects.

The computer that runs GEIRS may or may not have a sound card—see the output of any of the commands

```
cat /proc/asound/cards
amidi -l
/usr/sbin/alsa-info.sh
```

Usually GEIRS will be run on a remote server in the catacombs of the observatory, whereas the sound is supposed to be trumpeted on some controller’s desktop. In that case the GEIRS computer does not need a sound card.
There is at least one technique to forward the sound to the operator under openSUSE, which feeds the digitized pulse modulation into a PulseAudio channel on the GEIRS (=remote) computer, and forwards this as an RTP package to the pulseaudio channel on the operator’s (=local) machine, Figure 27. This is configured basically as follows:

1. Install the `paprefs` (pulseaudio preferences) openSUSE module on the remote and also on the local computer.

   If `paprefs` does not show anything, this is essentially done by calling `sudo /sbin/yast2`, selecting the Software management submenu, searching for `paprefs` and downloading and installing it.

   There are two variants to configure the forwarding.

   - `paprefs` is then called on the local computer, setting the Network Access to Make... PulseAudio network... available locally, setting the Network Server to Enable network access to local sound devices, setting the Multicast/RTP to Enable Multicast/RTP receiver. Again `paprefs` is called on the remote workstation, but setting Multicast/RTP to Enable Multicast/RTP sender and Create separate audio device for....

   - `paprefs` can alternatively be called from the Desktop menu via System → Configuration → PulseAudio Preferences.

   The disadvantage of this setup is that the remote computer broadcasts continuously the local audio stream to every other computer on the network, which eats bandwidth and is a waste of resources.

   - An equivalent setup can be reached by enabling the TCP related modules in `/etc/pulse/default.pa` on the two machines by removing the hash marks before the two `tcp` lines and the `zeroconf` line. `paprefs` is then called on the local computer, setting the Network Access to...
Make...PulseAudio network...available locally, setting the Network Server to Enable network access to local sound devices and Don't require authentication, and not checking any of the Multicast/RTP buttons. Again paprefs is called on the remote workstation, but not enabling any of the options in the submenus.

paprefs can alternatively be called from the Desktop menu via System -> Configuration -> PulseAudio Preferences.

These calls modify the $HOME/.gconf/system/pulseaudio files on the two computers and “called” from there with the aid of the module-gconf in /etc/sound/default.pa.

2. Enable pulseaudio either with

   setup-pulseaudio --enable

   or with sbin/yast2 under System -> /etc/sysconfig Editor -> Hardware -> Soundcard -> PulseAudio such that the PULSEAUDIO_ENABLE="yes" appears in /etc/sysconfig/sound.

3. On the remote computer the pulseaudio server needs to run. This can be checked with

   ps -C pulseaudio

   and is generally implemented by a non-comment line of the format

   autospawn = yes

   in /etc/pulse/client.conf. If this does not work, start the pulseaudio server on the remote computer manually:

   pulseaudio --start

   and if this is refused with

   pulseaudio -D

   (This might be included in the scripts/Generic of the GEIRS startup because the call is harmless if the server is already running.) On the local computer it probably is running already, because this would have detected the sound card:

   pactl info

   If one of the pulseaudio is not running, aplay or paplay will show (misleading) error messages of the form “connection refused.”

4. An intermediate test of the functionality is that pulseaudio works on the local machine, to be tested by copying a sound file to that machine and playing it with

   paplay *.au

5. Tell the server on the local workstation to accept the stream from the remote workstation. The least fuzzy way is to forward that information by accessing the remote computer with the -X switch of the ssh, such that the cookie appears on the remote computer, which can be checked with
xprop -root | fgrep PULSE

on the remote computer. If this information does not show up on the remote machine, either

start-pulseaudio-X11

or (more painfully) uncommenting the load-module module-x11-publish in /etc/pulse/default.pa

on the local machine—before calling the ssh—may be needed.

The files $HOME/.pulse-cookie in the home directories of the two computers seem to be no longer in use.

6. If alsa is used on the remote workstation, tell it to feed the output into its pulseaudio. The appropriate configuration is probably already in /etc/asound-pulse.conf on the remote workstation.

```bash
# PulseAudio plugin configuration

pcm.!default {
    type pulse
    hint {
        show on
        description "Default ALSA Output (currently PulseAudio Sound Server)"
    }
    fallback "sysdefault"
}

ctl.!default {
    type pulse
    fallback "sysdefault"
}
```

Since the (reverse) feeding of the pulseaudio channel to the alsa channel is likely also needed on the local workstation, an equivalent file is likely also needed on the local file system.

7. On the remote workstation, tell the pulseaudio server which machine ought to receive its output by setting the PULSE_SERVER variable to the local host:

```bash
RMHOST=`who -m | awk '{print $6}' | sed 's/\[[()]/\/'`
# RMHOST=`echo $SSH_CLIENT | awk '{print $1}'`  # alternative
export PULSE_SERVER=$RMHOST
```

This might be inserted (after translation to csh syntax) in the $CAMHOME/scripts GENERIC file on the remote workstation. If this forwarding service is also needed for other programs, it is a good idea to add these few lines also to the user’s .bash_login. Whether the numerical IP-address is needed depends on the availability of a DNS server from the remote computer.

8. Set the environment variable CAMAUDIOPLAY (in the scripts GENERIC) on the remote machine to paplay, such that aplay on the GEIRS workstation feeds its output of the audio file to its local pulseaudio daemon.
The installation is working once the command

```bash
cd $CAMHOME/SOUNDS
aplay -Dpulse rooster.au
paplay rooster.au
```

on the remote (GEIRS) workstation plays sound on the local workstation. If the call

```bash
cd $CAMHOME/SOUNDS
paplay rooster.au
```

on the remote workstation still says “connection refused,” this may be caused by a firewall on the local workstation—as for example enabled by default on fresh openSUSE 13.1 installations. The firewall must then be weakened (or just shut down) via `/sbin/yast2`, allowing the TCP packages from the remote computer with port 4713:

```
A.4 Network Time

Under openSUSE, configuration of the NTP is to be done in `/etc/ntp.conf`, or easier with the network configuration within `yast`. The daemon appears as `/usr/sbin/ntpd` with `ps -ef | grep ntp`. A running daemon does not guarantee that the clock on the system is updated, for example if hosted behind a firewall\(^\text{57}\), so it is advised to monitor `/var/log/ntp` or the equivalent log-file set in `/etc/ntp/conf` for the (irregular) corrections and to check that for example `ntpd` is responding.

Under CentOS 7, we edit `/etc/chrony.conf` (for example adding

```bash
server time.mpia-hd.mp.de iburst
```

at the MPIA), or

```bash
server gps0.mountain.lbto.org
server gps1.mountain.lbto.org
server ntp1.arizona.edu
server ntp2.arizona.edu
```

at the LBT, then

```
systemctl enable chronyd.service
systemctl start chronyd.service
```

A.5 X11

A.5.1 Forwarding

Under newer versions of openSUSE X11 forwarding with `ssh -X` may fail because the DISPLAY variable is not forwarded, although the forwarding is enabled in `/etc/ssh/sshd_config`. The

\(^{57}\) This is the MPIA case. `nslookup time` will reveal the IP address of the local time server.
solution of the problem is to enable IPv6 in the network configuration of the remote workstation, or to set the AddressFamily explicitly to inet (thus replacing the default, which is any).

Remote login from another place to a workstation may fail if the ssh daemon is not enabled on the remote site. To enable it, use /sbin/yast2 on openSUSE, the submenu Security and Hardening, then the submenu Enable extra services in runlevel 5 and switch the entry for the sshd to Yes. On Ubuntu use apt install openssh-server.

If the GEIRS workstation is hidden in a remote local network, the usual mechanism with port matching and X11 forwarding may be used. The example is

verdi9> ssh -X yoursshname@ssh.lbto.org

and then in that new shell on the intermediate machine

ssh> ssh -X geirsusername@Luci.luci.lbto.org

to log into a remote machine on the LBT network. We showed the prompts to illustrate on which computer's shell these commands are entered. Note that incomplete names like luci.luci do no longer work since changes in the DNS in the network in 2014.

If one needs to work on the remote machine with sudo(8) mechanisms, permissions to use the X11 interface need also to be added before trying to open GEIRS or other windows xauth(1).

xauth list
sudo -u effnewuser /bin/bash -i
# touch ~/.Xauthority # usually only needed for new users here
echo $DISPLAY
# Below add the full line after the 'add' that was the output of the
# previous xauth command. The correct line is the one which (almost)
# matches the current setting of DISPLAY. If DISPLAY is for example
# 'localhost:13', take the line from the 'list' that has 'somehost/unix:13'.
xauth add ... MIT-MAGIC-COOKIE-1 ...

A.5.2 Tunneling

Supposed one whishes to exchange files with a remote computer on the LBT network, this can basically be done by copying them first to ssh.lbto.org and from there to the destination. There are two possible directions of such a transfer. The example to copy a file tst.txt is

1. From the local computer named verdi9 to the remote computer named luci.luci.lbto.org:

   verdi9> scp -p tst.txt yoursshname@ssh.lbto.org:.
   verdi9> ssh -X yoursshname@ssh.lbto.org
   ssh> scp -p tst.txt geirsname@luci.luci.lbto.org:.
   ssh> rm tst.txt
   ssh> ~.

2. From the remote computer to the local computer:
CARMENES-AIV04B-NIR-DCS-MAN01

verdi9> ssh -X yoursshname@ssh.lbto.org # log into the ssh
ssh> scp -p geirsname@luci.luci.lbto.org:tst.txt . # copy from remote computer to ssh
ssh> ~. # log out from ssh
verdi9> scp -p yoursshname@ssh.lbto.org:. . # transfer file from ssh to local
verdi9> ssh -X yoursshname@ssh.lbto.org # log again into the ssh
ssh> rm tst.txt # clean up ssh intermediate copy

This chain of copying is complicated, and needs local disk space on the ssh intermediate computer that ought to be cleaned up. The more elegant alternative is to set up a tunnel that passes the data from the local computer to the remote computer, such that no intermediate files are created. There are again two directions. The most common task is to copy the FITS files from a remote disk to your local disk as follows. First set up a tunnel through the intermediate computer calling

verdi9> ssh -X -N -L 2022:xxx.yyy.www.zzz:22 yoursshname@ssh.lbto.org

on your local computer. (This command will respond nothing, so the output seems to hang after the password was typed in. Close the tunnel with CTRL-C after the connection is no longer needed, to return to the Linux shell prompt.) The xxx.yyy.www.zzz should be the IP address of the remote computer, for example 192.168.60.12 for luci.luci. Then transfer the files with

verdi9> scp -p -r -P 2022 geirsname@localhost:/dir/full/path/on/luci /full/path/on/verdi9

using the same number after the -P as the first port number in the previous tunneling setup. It is useful to move first into the target directory on the local computer, so the dot (.) can be used as the destination address. To use wild cards in the remote file names, surround the URI with simple quotation marks:

verdi9> cd /full/path/on/verdi9
verdi9> scp -p -r -P 2022 'geirsname@localhost:/dir/full/path/on/luci/*.fits' .

If one is logged into a computer outside the MPIA network, one can log into a computer inside the MPIA network if one has an account on rigel.mpia-hd.mpg.de. The principle is again to open a tunnel through rigel with some unused port (here: 2022) mapped on port 22 of the MPIA computer:

extr> ssh -X -N -L 2022:yourmpiacomp:22 rigelusername@rigel.mpia-hd.mpg.de
extr> rigeluserpassw

and then open another terminal on the external computer to reach your MPIA computer:

extr> ssh -X -p 2022 mpiacompusername@localhost
extr> mpiacompuserpassw

A.5.3 vnc client

Since X11 over ssh is an extremely slow setup for working with overseas computers, the standard Linux tool to open remote displays is the vncviewer(1). Installation under openSUSE:
zypper install libXvnc1 libvncclient0 libvncserver0 tigervnc vncmanager xorg-x11-Xvnc # openSUSE
yum install tigervnc-server # CentOS
apt-get install tigervnc-common tigervnc-standalone-server # Ubuntu

Log into the remote computer via ssh and start the vnc server there:

verdi9> ssh -X yoursshname@external
extr> vncserver -autokill

Remember the password just entered and the display number extr:N, and start the client on the local machine:

verdi9> vncviewer extr:N

In practise this is combined with tunneling to the remote X11 session, using the fact that the display number N reported by the server is port 5900 + N on the remote machine. To run a X11 session on the local computer verdi9 connected to the remote computer ln-lircs, for example:

verdi9> ssh -X lneng@ln-lircs
ln-lircs> vncserver -autokill # example response: ln-lircs:2
verdi9> ssh -N -L 5922:localhost:5902 ln-lircs # take 5922 because smallle probably already in use
verdi9> # this will appear to hang, keep the window open and open another one
verdi9> vncviewer localhost:22

Logout from the remote session as usual by clicking with the right-most button in a free part of the window manager of the remote screen (but do not shutdown the computer).

Alternative tools for X11 speed-up are NX (Section A.5.4) and x2go (Section A.5.5).

A.5.4 NX client

To connect via tunneling through the LBTO port machine to a remote computer on the LBT network with newer versions of NX, first set up a tunnel through ssh.lbto.org

verdi9> ssh -X -N -L 2022:xxx.yyy.www.zzz:22 yoursshname@ssh.lbto.org

in one terminal. (This will not show anything after you typed in your password and seems to hang.) Here xxx.yyy.www.zzz is the IP address of the remote computer; using a symbolic name like luci.luci may no longer work. Then start the NX client with

nxplayer  (under Linux)
nxplayer.exe  (under Windows)

in another window. If the command nxplayer is not found under Linux, use the full path name of the installation to start (/usr/NX/bin/nxplayer) or add /usr/NX/bin to the PATH. If /usr/NX is absent, install the software by downloading the RPM package from the company and install it first (as root) with
yum install nomachine_4.6.4_13*.rpm #under CentOS
zypper install nomachine_4.6.4_13*.rpm #under openSUSE

In the NX configuration use

- `ssh` as the protocol,
- use the same port as with the tunnel (2022 in the example),
- use the `localhost` as the machine to connect to,
- use the login account (for example `readout1` and password on the remote machine)

Just after installation, the NX support is running under an openSUSE system (`ps -elf | fgrep nx`), because `/etc/systemd/system/multi-user.target.wants` contains a `nxserver.service` entry. To disable this automated start each time the computer boots, use `/sbin/yast2`, the System submenue with the Service Manager, and disable the `nxserver`. In this case one needs to activate the service explicitly (as root) either from the same menu or by calling `/etc/NX(nxserver --startup`.

### A.5.5 x2go

If the operating system is openSUSE, `x2go` is installed on the remote workstation with

```bash
zypper ar obs://X11:RemoteDesktop:x2go/openSUSE_13.2 x2go
zypper ar obs://X11:RemoteDesktop:x2go/openSUSE_Leap_42.1 x2go # for openSUSE Leap 42.1
zypper refresh
zypper in x2goclient
```

If the operating system is CentOS 7, `x2go` is installed on the remote workstation with

```bash
yum install epel-release
yum --enablerepo=epel install xgoserver-xsession
yum --enablerepo=epel install x2goclient
```

The session is started with

`x2goclient`

Note that GNOME sessions seem not to work, only KDE sessions.

### A.5.6 Fonts

If the font system of the current X11 system does not offer the `courier-medium` and `courier-bold` fonts for the GUI's (revealed with `xfontsel` and `xlsfonts`) a modest adaptation is available by switching to another font in the style of
setenv CAMFONT courier

before starting GEIRS, for example fixed. There is an equivalent editable entry in Figure 4 for experimentation.

A.6 FITS

A.6.1 Chopping MEF

If images have been stored in the extensions and we wish to create versions with images in the primary header, the ftcopy command of the heatools is one way to create copies of that simpler format.\(^{58}\) Example: the four images extensions win1_1–win2_2 of the FITS file dcrsave0007.fits are restored in four new FITS files tmp_win_i.fits with the four Linux commands

heainit # not necessary if already in ~/.bash_login
ftcopy ‘dcrsave0007.fits[win1_1]’ tmp_win1.fits copyall=no
ftcopy ‘dcrsave0007.fits[win1_2]’ tmp_win2.fits copyall=no
ftcopy ‘dcrsave0007.fits[win2_1]’ tmp_win3.fits copyall=no
ftcopy ‘dcrsave0007.fits[win2_2]’ tmp_win4.fits copyall=no

A note to CARMENES observers: The usual way to open both detector images at the same time with ds9 is

ds9 -multiframe -cmap bb file.fits

Since March 2015 a 2D WCS coordinate system in units of millimeters has been added to the FITS headers, so one can also use for example

ds9 -mosaicimage -cmap bb -zoom 0.5 file.fits

to render the image with an approximately correct gap between the two chips.

A.6.2 ds9loop

A command ds9loop with the syntax

ds9loop [ds9options...] dir1 [dir2 ...]

is in the GEIRS scripts which calls ds9 in a loop over all fits files in the named directories. The only required interaction by the user is to close ds9 for moving on to the next. Examples:

ds9loop .
ds9loop -mosaicimage /data1/Panic

A.6.3 fits2csv

The program fits2csv opens the GUI of Figure 28 and scans recursively a list of directories for all files with suffix \textit{.fits}. The FITS header keywords that match a finite list of strings defined by

\(^{58}\)This is a user’s note that has nothing to do with GEIRS.
the user are searched in a HDU of each of the files and written as a comma-separated list of values (CSV), into a text file specified by the user.

The keywords should be provided as regular expressions of the form HIER.*keyword if there are some general hierarchical prefixes in front of them.

The GUI is not opened if fits2csv is called with any command-line options (in batch mode), see the man-page in Section 5.5.

The standard way of using this new text file is to open it with a spread-sheet editor like open office, specifying the comma as the delimiter.
Figure 28: The GUI called in by fits2csv
A.6.4 FTOOLS

The heatools mentioned at many places in this manual are compiled as follows:

1. Ensure that you have a recent version of compilers of your operating system, including gfortran. On openSUSE for example, use /sbin/yast2, the software management, and look into the RPM group under Development - Languages - Fortran. You may also need to install the libXt-devel package such that X11/Intrinsics.h is known.

2. Download the source code from the download page. Select the Source code (CentOS or openSUSE or both), not any precompiled binaries, and select the General-Use FTOOLS, deselect the Attitude, Caltools, HEASim, HEASPtools, HEAGen, Time and Xspec packages which are not of much interest to optical Astronomy, and click Submit. Download everything (roughly 100 MB) to $HOME/heasoft-6.23.src.tar.gz.

3. Unbundle with

```
cd $HOME
rm -rf heasoft-6.25
tar xzf heasoft-6.25src.tar.gz
```

4. umask a+rx

```
cd heasoft-6.25/BUILD_DIR # depending on tar version also ..-6.22.1...
./configure --x-libraries=/usr/lib --x-includes=/usr/include # openSUSE 13.2
./configure # CentOS 7 or openSUSE Leap 42.2
nice make |& tee build.log
nice make install |& tee install.log
chmod +x headas-init.*
```

5. add to $HOME/.bash_login or $HOME/.bashrc (details of the libc will probably differ, current CentOS 7 systems end in libc2.17):

```
export HEADAS=${HOME}/heasoft-6.25/x86_64-pc-linux-gnu-libc2.26
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${HEADAS}/lib
. $HEADAS/headas-init.sh >& /dev/null
```

and make sure that your terminals are login terminals.

As an example of how shell scripting with these tools work, consider the task of subtracting the first slice from the second, the second from the third, and the third from the fourth slice of an image cube. The script extracts slices with ftcopy, converts them to a floating point (BITPIX -32) version to avoid underruns while subtracting them if they are unsigned short integers, and subtracts them pairwise with ftpixcalc:

```
#!/usr/bin/env bash

# Generate arithmetic differences of consecutive slices of a FITS cube.
# Note: this makes heavy use of the HEATOOLS.

# Usage:
```
# pipFits_cdiff.sh fitsinputfile.fits outfilestub

# The first command line argument must be an existing and readable FITS file,
# the second command line argument is a file name stub for the differential
# images to be created. The second command line argument should not collide
# with names of existing files.

# $Header$

# since 2018-03-14

if [ $# -ne 2 ] ; then
    echo Usage: $0 inputfitsfile.fits outfilenamestub
    exit 1
fi

# save command line argument for input file name
ifi=$1
ofi=$2

if [ ! -r ${ifi} ] ; then
    echo Cannot read $ifi
    exit 1
fi

naxes3=$(dfits ${ifi} |fgrep NAXIS3 | awk '{print $3}')

if [ $naxes3 -lt 2 ] ; then
    echo Less than 2 slices in ${ifi} PHDU image
    exit 1
fi

# loop over slices 2 up to the number of slices
for (( j=2 ; $j <= $naxes3 ; j = $j + 1 )) ; do
    # get the previous slice number (one less)
    i=$(( $j -1 ))

    echo $i $j

    # extract slices i and j into files named with suffix tmp1 and tmp2
    ftcopy "${ifi}[*,*,$i]:$i]" ${ofi}.tmp1 clobber=yes
    ftcopy "${ifi}[*,*,$j]:$j]" ${ofi}.tmp2 clobber=yes

    # convert slices to floating point representation (avoid underflow if integer)
    chimgtyp ${ofi}.tmp1 ${ofi}.tmp1F FLOAT Fnull=0.0 clobber=yes
    chimgtyp ${ofi}.tmp2 ${ofi}.tmp2F FLOAT Fnull=0.0 clobber=yes

    # subtract the two slices and call the result sub.....fits
    ftpixcalc ${ofi}$_$j.fits 'A-B' a=${ofi}.tmp2F b=${ofi}.tmp1F clobber=yes
done


exit 0

Note that this program is imperfect. It does not erase the invalid CHECKSUM and DATASUM keywords
of the old header, for example.

A.6.5 ds9

ds9 is obtained from http://ds9.si.edu/site/Download.html. If xslt-config is not in the path (apparently the case for CentOS 7) install the package with

```
yum install libxslt libxslt-devel # CentOS
zypper install libxslt-tools libxslt1 libxslt-devel # openSUSE
```

Ensure for openSUSE that the xml2 and openssl libraries are available:

```
zypper install libxml2-devel libopenssl-devel
```

Assuming the sources are from the Beta Version of the web site, compile ds9 with

```
tar xzf ds9.7.6.tar.gz
cd SAOImageDS9
# patch for hard-coded library placement under openSUSE
mkdir -p lib
ln -s lib lib64
unix/configure
make
# cp bin/ds9 ${HOME}/bin # optional
cp bin/ds9 ${INSROOT}/bin
cd ..
rm -rf SAOImageDS9
```

If the openssl version of the operating system is 1.0 or higher, indicated for example in /usr/include/openssl/opensslv.h, DS9 versions up to and including 8.0rc2 cannot compiled that way, because the API of the library changed. A patch for that scenario exists. For recent CentOS and openSUSE systems RPM packages have been put on openSUSE build server.

Ubuntu users can obtain ds9 via

```
apt install saods9
```

A.6.6 siril

The fastest way to fit a star blob in a FITS image to a Gaussian is apparently by opening it in siril and dragging a box around the feature. This is in particular useful for LN where the diameters under standard seeing conditions are 100 to 200 pixels. The source code is obtained from https://free-astro.org/index.php?title=Siril:0.9.8. Compilation (under openSUSE):

```
bunzip2 siril*tar.bz2
tar xf siril-0.9.8.tar
```
zypper install libconfig-devel opencv-devel
(cd siril-0.9.8
./autogen.sh
make
make install)

Under Centos one needs to install a higher version than 1.5 of libpng first from https://sourceforge.net/projects/libpng/files/:

unxz -c libpng-1.6.32.tar.xz | tar x
(cd libpng-1.6.32
./configure --prefix=/usr
make install)

bunzip2 siril*tar.bz2
tar xf siril-0.9.8.tar
(yum install gtk3-devel libconfig-devel opencv-devel # Centos
(cd siril-0.9.8
./autogen.sh
# edit Makefile and src/Makefile to replace png15 -> png16 throughout
make
make install)

A.6.7 SkyMaker

To simulate images with the TwoMassCnvert(1) program one needs sky(1). The source code is obtained from http://www.astromatic.net/software/skymaker. The FFTW library is needed, for example under CentOS implemented with yum install fftw-devel. SkyMaker is then compiled with

tar xzf skymaker-3.10.5.tar.gz
cd skymaker-3.10.5
./configure --prefix=$HOME --disable-threads
make
make install

We assume that $HOME/bin is in the $PATH.

A.7 SVN installation

An SVN installation happens by downloading the gzipped tar ball from https://www.apache.org/dist/subversion/, then

tar xzf subversion-1.9.3.tar.gz
cd subversion-1.9.3
./configure --prefix=$HOME
make
make install