Research School of Astronomy and Astrophysics

Mount Stromlo and Siding Spring Observatories

RSAA
- Home
- About Us
- Research
- Technology
- Study@RSAA
- Observing
  - Telescope Schedules
  - Applying for Time
  - Observing Capabilities
  - Telescopes
    - 2.3m
    - 40inch
    - 24inch
  - Detectors
  - Computer Accounts
  - CICADA
  - Travel & Tourist info
  - SSO Accommodation Booking
  - Support & Reports
- Gemini
- Public Information
- News
- Jobs

Quick Links
- RSAA People
- Contacts
- Picture Gallery
- What's On
- RSAA Alumni

Intranet (Staff only)
- Intranet Connect
- Webmail

Search RSAA
Go

40inch Telescope

Overview

In the early 1960s, the Australian National University established a new astronomical site at Siding Spring in central New South Wales, partly in the expectation that it would provide somewhat clearer skies than those at Mount Stromlo, and partly to escape the increasing light pollution of the growing Canberra. The selection of Siding Spring followed an extensive site-testing programme organized by the then Director, Bart Bok. The new site was instrumented with three reflectors of modest size, all made by Boller & Chivens. The 40-inch is the largest of these, the other two being a 24-inch and a 16-inch. The 40-inch is heavily scheduled, and it has proved a useful instrument especially for photometric and imaging work.

Specifications

- 40inch (1.0m), f/5 primary
- Ritchey-Chritien optics
- Secondaries for f/8 and f/18 Cassegrain imaging (~8100mm and ~18300mm focal length)
- German Equatorial mount

Telescope Control

- ETS Manual (24,30 & 40in)
- Instrument Computer link to ETS

Instruments

- Direct Imaging
- "Wide Field Imager (WFI)"
  - Andrew Jacob User Notes

Page last updated: 16 February 2005
Please direct all enquiries to: Webmaster
Page authorised by: Director, RSAA

Research School of Astronomy and Astrophysics

Mount Stromlo and Siding Spring Observatories

RSAA
- Home
- About Us
- Research
- Technology
- Study@RSAA
- Observing
- Gemini
- Public Information
- News
- Jobs

Quick Links
- RSAA People
- Contacts
- Picture Gallery
- What's On
- RSAA Alumni

Intranet (Staff only)
- Intranet Connect
- Webmail

Search RSAA
Go

Page last updated: 07 January 2005
Please direct all enquiries to: Webmaster
Page authorised by: Director, RSAA

J. van Harmelen, 17-NOV-1988, V3.1
J. van Harmelen, 1993, V4.0 View Command added
J. van Harmelen, 25 October 1996, V4.1 Configure Command added, general review.
J. van Harmelen, 7 July 1999 V4.2 Other Telescope Commands added
Autoguide Command added, Status Command added
J. van Harmelen, 12 July 2000 V4.3 Added "Halt" command for small telescope control systems

SPECIFICATION OF A CONNECTION BETWEEN TELESCOPE AND INSTRUMENT COMPUTERS (ETS_LINK)

1. SCOPE

This document relates to the connection of stand alone instrument control computers to telescope control computers for the purpose of conveying telescope coordinate and time information. Other information and control commands have been added to the list of commands.

It does not concern instrument control by VAX computers, either by the telescope computer itself, or by other networked VAX's, as user programs have access to telescope data and can control the telescope by the methods described in the 2.3m Telescope Observer's Manual (Chapter 12).

The use of a dedicated serial line for this purpose should be regarded as a simple and convenient solution for the desired data exchange. At a later time network connection will considered.

2. HARDWARE

The Telescope and Instrument computers are to be connected by R5232 ports, operating at 9600 bps, no parity, one stop bit.

On a VAX this will be a general purpose terminal port, on a PDP11/24 port SLU2:, on other PDPll's a comparable port with the same address as SLU2:, on other computers a convenient serial port.

3. DATA EXCHANGE PROTOCOL

All data exchange will be in ASCII format.

The Instrument computer is the master of the datalink: it requests information when needed, by sending a command to the Telescope computer, which will then reply with the requested data, or with an error message.

The Telescope computer is available for exchange of information at any time while the telescope system is running. When the telescope system is not running, there will be no reply from the telescope computer to any Instrument computer commands. The instrument computer software shall implement a time-out to cope with this condition.

4. COMMAND AND DATA FORMATS

The command and reply repertoire described below is thought to cover the requirements of most existing instruments which are controlled by their own computer systems, but may need future extension. The CONFIGURE and VIEW commands are two such extensions.

The VAX Telescope computers use DCL parse routines and will generate the error message "UNRECOGNISED COMMAND" if the commands can not be recognised. Other types of telescope control computers shall mimic the behaviour of a VAX/VMS machine as much as possible. All systems implement as a minimum the TELESCOPE, COORDINATES, TIME and VIEW commands. Commands which control the telescope are only implemented on a needs basis.

A command or qualifier can be abbreviated to the minimum number of characters necessary to uniquely identify it, but not to less than two characters, and has to be terminated with a carriage return <CR>. No more than four characters shall be required to uniquely identify a command or qualifier. The command strings are not case sensitive.

All replies are terminated with a <CR><LF> combination. All alphabetic information in the reply messages will be in capitals.

The commands TELESCOPE, COORDINATES, TIME, VIEW and CONFIGURE are available together with qualifiers to indicate which coordinates or time are required, and whether the reply values will be in 'string' or 'real' format.

4.1 CONFIGURE

The command: CONFIGURE configuration_variable value

shall be accepted by the telescope system as if it were issued on the command line at the telescope control terminal, or the equivalent information entered in the configuration dialog in menu driven systems. Refer to the Telescope Manuals for further details.

A <CR><LF> reply indicates successful completion of the command, any error is signalled with the reply: UNRECOGNISED COMMAND.

A special case is the control of the Dome_Control configuration variable, which on successful completion reports the previous value:

DOME_CONTROL WAS MANUAL or DOME_CONTROL WAS AUTOMATIC

This addition to the command repertoire introduced in version 4.1 has only been implemented on the 74" and 50" telescope systems. Other systems will be revised as need for this command arises.

4.2 COORDINATES

If the telescope is tracking, this command causes the telescope computer to reply with the message:

"object name" ra dec equinoxstring

The object name is only included if available, and is always in capitals and enclosed in double quotation marks.

The format of RA and DEC variables depends on the format qualifiers and is described in detail under the /REAL and /STRING headers.

The equinox string contains seven or eight characters: the FK4/FK5 identifier "B" or "J", followed by the epoch as "yyyy.y" (seven characters), or the string "APPARENT" (eight characters) when apparent place is selected for the telescope display.

If the telescope is not tracking, an error message is generated:
TELESCOPE NOT TRACKING

If the coordinate data are inaccessible or suspect, the reply is:
DATA ACCESS ERROR

/BASE

The requested coordinates are Base Coordinates (refer to 2.3m Telescope Observer's Manual for definition). (Not available on telescopes which do not support the TRACK command)

/FILE

The requested coordinates are File Coordinates (refer to 2.3m Telescope Observer's Manual for definition). (Not available on telescopes which do not support the TRACK command)

/TRACK (default)

The requested coordinates are the current telescope Tracking Coordinates (refer to 2.3m Telescope Observer's Manual for definition).

/REAL

RA and Dec are supplied in ASCII representation of their floating point values:
0 < RA < 2*PI, format: r.rrrrrr
-PI/2 < Dec < PI/2, format: sd.dddddd (s=sign)

The sign is only included if it is negative.

/STRING (default)
RA and Dec are supplied in string format:

RA: hh mm ss.d Dec: sdd mm ss (first s=sign)

If the information is inaccessible or suspect, the reply is:
DATA ACCESS ERROR

4.3 TIME
This command causes the telescope computer to reply with the message:

mjd sidereal_time selected_time selected_date

"mjd" is the Modified Julian Date (defined as JD - 2400000.5) in the format: ddddd.dddddd
"sidereal_time" is Local Apparent Sidereal Time
"selected_time" is either Universal Time or Civil Time as specified by the qualifiers /UT and /CT.

The format of "sidereal_time" and "selected_time" variables depends on the format qualifiers and is described in detail under the /REAL and /STRING headers.

"selected_date" is either UT Date or Civil Date as specified by the qualifiers /UT and /CT.
The "selected_date" is returned in string format:

dd-mmm-yyyy

If the time information is inaccessible or suspect, the reply is:
DATA ACCESS ERROR

/CT
The "selected time" variable is returned as Civil Time: Australian Eastern Standard Time or Australian Eastern Daylightsaving Time as applicable (UT + 10, resp UT + 11 hours).

/UT (default)
The "selected time" variable is returned as Universal Time.

/REAL
selected_time and sidereal time are supplied in ASCII representation of their floating point values:

0 < time < 2*PI format: t.tttttt

/STRING (default)
selected time, which is UT or Civil Time depending on the /UT and /CT qualifiers, and sidereal time are returned in string format:

hh:mm:ss.s in the range 0 to 24 hours.

4.4 TELESCOPE
This command causes the telescope computer to reply with an identification message:

tel_id_string astr_latitude astr_east_longitude height_AMSL

"tel_id_string" is the site and telescope identifier, a fixed length string of fifteen characters left justified, in the format:
SSO 2.3METRE or MSO 74INCH etc.

"astr_latitude" and " astr_east_longitude " are the astronomical latitude and east longitude in decimal degrees in the format:

sdd.ddddd and ddd.ddddd (s=sign)

"height_AMSL" is the integer value of the height above sea level of the telescope in metres: hhhhh

4.5 VIEW
The VIEW command permits an observer to see the value of any telescope display variable
(even if it is not currently featured on one of the display screens).

The command: VIEW variablename [,...]

causes the names, values (and units where appropriate) of the specified display variable(s) to be returned. A list of display variable names for your telescope can be found in the chapter 'Telescope Display' in your Telescope Observer 's Manual. The details of the format of the returned data vary with the type of data requested. Rather than trying to detail all possible cases here, it is recommended to obtain the data by typing in the VIEW command at the control terminal of the 2.3m, 74" or 50" telescopes.

Examples:

command:	reply:
TELESCOPE	SSO 2.3METRE -31.27336 149.06119 1149
TEL	MSO 74INCH -35.32065 149.02433 768
COORDINATES	"B CENT" 14 03 00.3 -60 19 05 51988.5
COORD/REAL	"BCENT" 3.678281 -1.052749 J1988.5
COORD	12 34 56.7 +06 54 32 B1950.0
COORD	TELESCOPE NOT TRACKING
TIME	47465.711806 05:41:57.1 17:05:00.0 31-OCT-1988
TIME/CT	47465.711806 05:41:57.1 04:05:00.0 1-NOV-1988
TIME/REAL/CT	47465.711806 1.492038 1.060288 1-NOV-1988

4.6 OTHER TELESCOPE COMMANDS (2.3m, 74" and 50")

Any telescope command can be given and will be executed. The reply for successful commands is just a <CR>, for unsuccessful commands the error message UNRECOGNISED COMMAND plus the body of the VMS error message generated by the control system. On the 2.3m problems with the delivery of error messages have not yet been solved. It is not possible to discern a command failure.

4.7 TRACK/COORDINATE, OFFSET[/BASE] and HALT COMMANDS (40", 30" and 24")

For those small telescopes which operate under computer control (40"and 24" to be implemented in winter 2000), the following telescope commands are supported: "TRACK/COORDINATE coordinates" (in standard MSSSO format), "OFFSET ra dec", "OFFSET/BASE ra dec", and HALT. The reply for successful commands is just a <CR>, for unsuccessful commands the error message UNRECOGNISED COMMAND plus the body of the VMS style error message generated by the control system.

4.8 AUTOGUIDE COMMAND (2.3m only)

The "AUTOGUIDE ew ns" command puts the sky offsets supplied into the autoguider offset fields of the control database. The telescope will react to this on the next control loop.

4.9 STATUS COMMAND

The telescope control system will reply to the "STATUS" command with one of the following strings indicating the current telescope status: "OFF", "FAULT", "HALTED", "WAITING", "SLEWING", "TRACKING". External systems controlling the telescope should use this command to determine when to continue with the next phase of their control program.

5. IMPLEMENTATION ON THE TELESCOPE COMPUTERS

The implementation of the instrument connection on the VAX Telescope computers takes the form of a process as part of the telescope system. The designated serial port is allocated to this process. The process reads from the port and waits till a command arrives. DCL parse routines are used to decode the command, the reply string is assembled from information taken from the telescope database, where necessary using the telescope library routines to do data conversions, and then written out to the port.

On other telescope computer types the actions of the VAX computers shall be imitated as closely as possible. The communication port used shall be polled frequently enough not to produce any noticeable delay in reply or be interrupt driven.

6. IMPLEMENTATION ON INSTRUMENT COMPUTERS

The implementation of the instrument connection on PDP11 Instrument computers uses an RT11 device handler to communicate with the serial port. The instrument control program will contain a subroutine which writes the required commands to the port and decodes from the received reply string the
information it requires.

On other types of instrument computers there is no prescribed method of implementation. The use of reply parsing routines which are character position dependent is to be discouraged. Small differences in replies between types of telescope computers are possible and should be coped with. Examples of these differences are the number of significant digits in scientific notation of variable values returned by the VIEW command and the number of spaces separating fields.

40inch Direct Imager

This manual describes the operation and specifications of the direct imager on the 40inch telescope at SSO.

The imager is suitable for optical imaging at pixel scales of 0.4-0.6"/pixel, depending on the CCD. The MSO Detector Lab has a list of currently available CCDs. The automated 6-position filter wheel is normally used with standard UBVRI filters, but can be loaded with observer-provided 50??mm filters if required.

A f/18 secondary is also available, however the focal scale (0.2-0.3"/pixel) and typical SSO seeing mean that it seldom used.

Intrument Capabilities

Overview

The imager is suitable for optical imaging at pixel scales of 0.4-0.6"/pixel, depending on the CCD. The MSO Detector Lab has a list of currently available CCDs. A f/18 secondary is also available, however the focal scale (0.2-0.3"/pixel) and typical SSO seeing mean that it seldom used.

Filters

The filter wheel can hold six 50??mm filters and is normally used with standard UBVRI(+clear) filters. It can be loaded with observer-provided filters if required. The filter wheel is fully controllable from the observing software, Cicada.
A f/18 secondary is also available, however the focal scale (0.2-0.3"/pixel) and typical SSO seeing mean that it seldom used.

Sensitivity

Sensitivity data is sparse. Measurements made in late 1993 indicate about 20 and 15 electrons/sec for a 20th magnitude star, in V and I respectively. Figures for newer CCDs are unknown.

Research School of Astronomy and Astrophysics

Mount Stromlo and Siding Spring Observatories

RSAA
- Home
- About Us
- Research
- Technology
- Study@RSAA
- Observing
  - Telescope Schedules
  - Applying for Time
  - Observing Capabilities
  - Telescopes
  - Detectors
    - Current Info
    - 2.3m Imager
    - DBS Blue
    - DBS Red
    - WiFeS
    - WFI
    - SkyMapper FPA
    - Useful Links
    - Misc. Systems
  - Computer Accounts
  - CICADA
  - Travel & Tourist info
  - SSO Accommodation Booking
  - Support & Reports
- Gemini
- Public Information
- News
- Jobs

Quick Links
- RSAA People
- Contacts
- Picture Gallery
- What's On
- RSAA Alumni

Intranet (Staff only)
- Intranet Connect
- Webmail

Search RSAA
Go

Current Detector Information

Dewar numbers Dn indicate re-furbished heads.

Detector	Dewar	Type	Frame Format	Pixel Size	Read Noise(e)	Details
SiTE 2k	D10	Tek	2048x2048	24um	12	thinned, on 40" f/8
E2V CCD4210	D11	E2V	Imaging: 2048x512 Frame: 2148x562	13.5um (Full Well ~120ke)	3.5(G=2) 4.65(G=3)	thinned, R/O time 7s, on 2.3m telescope New DBS-B (Old Tek 1kx1k
E2V CCD4240	D6	E2V	Imaging: 2048x2048 Frame: 2148x2148	13.5um (Full Well ~120ke)	4.4(G=2) 3.6(G=1)	thinned, R/O time 15s, on 2.3m telescope New Imager (Old EEV 2kx1k)
E2V CCD4210	D7	E2V	Imaging: 2048x512 Frame: 2148x562	13.5um (Full Well ~120ke)	3.5(G=2) 4.65(G=3)	thinned, R/O time 7s, on 2.3m telescope Science Detector will be installed on May 9th Commissioning Delayed till early June New DBS-R (Old Tek 1752x532)
DBS-R	D9	SiTE	1752x532	15um	5	grade 1, DBS only. This head to be retired

E2V Imager Timing Files and Gain Settings

For the Imager, fast = medm = slow, the detector is read-out at the same speed i.e. about 15 seconds. There is now NO slow or medium speed with the new Imager.

So the DEFAULT, 2.4e/adu lod file is - fast.lod

The GAIN setting of the Imager may now be changed using the GAIN option on the Cicada Menu, rather than re-loading a timimg file

The gain settings and noise are as follows

GAIN	CDS Gain	System Gain (e/adu)	Noise(e)	Bias	Comment
0	9.5	0.53	4	--	Very Hi Gain
1	4.75	1.1	3.6	1343	Hi Gain
2	2	2.4	4.4	1190	Default
3	1	5.6	6.8	--	Low Gain

DBSB Timing Files and Gain Settings

For the DBSB, fast = medm = slow,
the detector is read-out at the same speed i.e. about 7 seconds.
There is now NO slow and medium speed with the new DBSB (or DBSR)

So the DEFAULT, 1.04e/adu lod file is - fast.lod

However, the GAIN setting of the system may be changed according to the table below -

The gain settings and noise are as follows

GAIN	CDS Gain	System Gain (e/adu)	Noise(e)	Bias	Comment
1	4.75	0.45	9	--	Hi Gain
2	2	1.04	3.5	1230	Default
3	1	2.35	4.65	1241	Low Gain

Enquiries about Current Detector Information?
detman@mso.anu.edu.au

Page last updated: 06 May 2005
Please direct all enquiries to: Webmaster
Page authorised by: Director, RSAA

A short user's guide to the 40-inch and its CCD-camera system

April 1999

This document describes the basics of the use of the 40-inch for CCD imaging. It is recommended that it be read in its entirety before the observing run commences.

1 Before you start

Some things which should be done early in the afternoon before you start your run:

1. Check that the CCD/Dewar you want is on the telescope and is cold.
2. Check that the LN_2 dewar is in the dome and full enough to do a couple of fills. If it is Friday it will need to last over the weekend. Hold times for the large dewars is 18-22 hours, the smaller ones hold for up to 12 hours.
3. Check that the filters that you need have been installed in the filter wheel in their correct positions.
4. Check that you can run CICADA (see Section 3) and can write files to the data disk.
5. Make sure you have tapes and can write to them with the tape drive.
6. Check that the CCD sees light. Verify that the gain and readout speed is what you want and that the bias levels and RON are reasonable (bias is typically 600-1200 ADU).
7. Check that the telescope computer and TCS is running and reading the coordinates.
8. Check that you can move the telescope. (see Section 2). Check that when the telescope is tracking and you do an exposure that the FILTER and RA and DEC are written to the fits headers of your files.

In all cases, if you have problems - contact the technicians as early as possible.

2 Starting up and using the telescope

3 Starting up and using Cicada

4 Shutting things down

Once your flats and biases are taken:

1. replace the telescope lid, and close the dome slit.
2. **TURN OFF THE TRACKING** so the telescope doesn't run into the floor.
3. fill the CCD dewar so you don't have to get up just to fill it before the vacuum is degraded.
4. write in the weather log book, and dim the lights.
5. submit any faults to the so the technicians can work on it during the morning.
6. watch the sunrise, go to the lodge, and go to sleep.

5 General Tips about observing with the 40''

6 Troubleshooting

Loud alarm goes off while slewing

You've slewed past the RA axis limit.

1. Hit alarm off button (the sound yet won't stop yet)
2. Hold down the top of the horizontal limit button **while**
3. slewing in the direction opposite to what got you in trouble in the first place. Once the telescope is back within the limits, the alarm will stop.

Dome won't rotate

Dome being domes, get stuck sometimes.

1. Try rotating the dome back in the opposite direction until it gets unstuck, then try again.
2. If the dome is completely stuck, the circuit-breakers might have tripped. To check, try the flourescent switch under the dome shutter - if the lights don't go on then you have to reset the circuit breakers. Go to the ground floor and check all the breakers in the loading dock area.
3. If that didnt work, fuses might be blown or worse, and the technicians will have to fix it.

You can't find any of your objects, well-known bright stars, globular clusters nor even the Moon.

The telescope pointing calibration is bad.

1. If you used the 1996 Astronomical Almanac Bright Stars section to calibrate your pointing, go back and do it again using a *different* year's edition of the Astronomical Almanac. The 1996 Almanac Bright Star coordinates are incorrect.
2. see section on Pointing

7 Acknowledgements

This document is an (updated) compendium of various user guides written over the years. Thanks go to the following people for their earlier guides, upon which much of this document is based.

Mike Bessell, Jayanne English, Albert Bosma, Garry Kitley and Helmut Jerjen.

Version Information
V1.1: 990404, Kim Sebo: Updated and rearranged for inclusion on the WWW, including notes from Helmut Jerjen, and autoguider info from Mike Bessell.
V1.0: 970922, Mike Bessell: original version based on the combined notes of Jayanne English, Albert Bosma, Garry Kitley and others.

File translated from TEX by TTH, version 2.25.
On 25 May 1999, 21:54.

Research School of Astronomy and Astrophysics

Mount Stromlo and Siding Spring Observatories

RSAA
- Home
- About Us
- Research
- Technology
- Study@RSAA
- Observing
- Gemini
- Public Information
- News
- Jobs

Quick Links
- RSAA People
- Contacts
- Picture Gallery
- What's On
- RSAA Alumni

Intranet (Staff only)
- Intranet Connect
- Webmail

Search RSAA
Go

CICADA PROJECT DOCUMENTATION

- Requirements Specification
- Functional Specification
- CICADA Design
- CICADA V3.1 User's Manual (postcript)
- CICADA V3.1 Writing Plug-Ins (postscript)
- CICADA V3.1 Configuration Manual (postscript)
- Graphing and Imaging Tool V3.0 User's Manual (postscript)
- CICADA ADASS '96 Paper
- CICADA ADASS '98 Paper (postscipt)
- CICADA System Documentation (in prep)
- Cicada binaries
- GSFind Manual V2.0 (postscript)
- AAO 6dF V1.0 (postscript)
- Pipeline Manual V1.0 (postscript)
- Pipeline ADASS 1998 Paper (postscript)
- Developer Notes
- Configuration Manager
- Testing
- Versions

Page last updated: 07 January 2005
Please direct all enquiries to: Webmaster
Page authorised by: Director, RSAA

MOUNT STROMLO & SIDING SPRING OBSERVATORIES CCD/2.2
Computer Section
2.2

17 June, 1994

CCD User Interface Software -- Requirements Specification

Purpose

The CCD User Interface System (CUIS) will provide access to, control of, and retrieval of data from, the CCD detectors attached to instruments mounted on observatory telescopes. User access will be in the context of a scheme whereby users will access software at the telescope through a workstation similar to those used for data reduction at MSO. [This scheme will be specified in a separate paper.]

Rationale

The existing situation with the CCD systems is that they are controlled through i386 or i486 systems running Interactive Unix. The observer enters commands into these systems via keyboard and tablet, views output on two monitors (image and data). This system has a number of shortcomings relating to the specific hardware (tablet, and monitors) that are required. There is also a proliferation of keyboards and monitors in the observing rooms. The user interface is quite specific to the Astromed controller, and would need to be replaced when a controller from another source was introduced.

CUIS will allow the observer to operate the CCD from a standard workstation through a window-based interface, using a keyboard and mouse for input. The same workstation will be used for other tasks, such as data reduction, telescope control, and more mundane tasks such as email.

The observer will be dealing with a computer system much like that in his office, and will have fewer (ideally, one) workstations to access.

Capabilities

The CUIS will:

1. Be the point from which the observer will enter commands to be transmitted to the CCD controller.
2. Provide the observer with a means of monitoring the CCD.
3. Collect data from the CCD, storing it on both disk and removable media.
4. Provide technical staff with a means of testing the CCD.
5. Provide the above capabilities for multiple CCD systems on a given telescope.
6. Be designed to adapt to future CCD controllers, while presenting the same interfact to users.

[The CUIS will not provide data analysis capabilities. These will be provided by standard analysis packages , such as IRAF or Figaro.]

Technical Constraints

Throughput

Data throughput will be comparable to or better than that of the Astromed system in use in October 1993, with regard to

1. read out
2. display
3. file archiving

External Standards

1. Data files written to tape, or to disk for access by other software systems, will be in FITS format.
2. Windowing will be based on X. OpenWindows may be used initially; Motif at a later stage.

Configuration

Hardware

1. The present CCD controller is an EISA card in a i386 or i486 system, which is connected to the workstation is via ethernet. Future controllers may operate through an S-bus card on the ethernet, or other similar configuration.
2. The CUIS will run on a Sun (or similar Unix-based) workstation. Currently Sun IPC's are used.

Operating System

1. The workstation will run Unix. Currently SunOS 4.1.3 is used. The system will operate under Solaris 2.3 as well. The design will anticipate the possibility of Unix from other vendors.
2. The ix86 systems run Unix.
3. The windowing software will be based on X11.

Third-Party Software

1. The present CCD controller on the ix86 uses software provided by Astromed. Future controllers will probably be supplied by the manufacturer with low-level software.
2. IRAF - compatible image display software (e.g. SAOimage or Ximtool.)

Users

The CUIS will be used by observers -- both MSO staff and visitors. All users can be assumed to be frequent computer users, and have familiarity with some windowing systems, but they will vary widely in the depth of their knowledge of Unix and X11. Furthermore there will be a wide range in knowledge of the functioning of the CCDs themselves.

The CUIS will be used by people who have been up all night and may not be too wide awake. Appropriate fail-safe features will be included.

Tasks

The CUIS will provide the tools for the user to perform a number of tasks, including initialising the CCD, setting exposure parameters, initiating exposures, monitoring the system, and archiving data. The complete list of tasks is given in the Functional Specification (CCD/1).

Documentation

Documentation will include

1. On-line manual pages in the standard Unix format.
2. Tutorial documentation available in hard copy at the telescopes, aimed especially at casual users.
3. On-line help available from window system menus. These will incorporate hypertext cross-referencing where appropriate.

Technical Risks

[This section itemises those areas where problems may arise which would make it difficult to meet the requirements.]

Technical difficulties

The Unix implementation on the ix86 (Interactive Unix v 2.x) is not robust.

External Dependencies

The system depends on the Astromed software and controller.

Certainty of requirements

The CUIS will be used by observers with a range of requirements. Some of these requirements may not be identified or articulated until the system is put into use.

About this document ...

This document was generated using the **LaTeX**2HTML translator Version 99.2beta8 (1.43)

The command line arguments were:
latex2html `-init_file /opt/local/html/latex2html/cicada.init ccdrs.tex`

The translation was initiated by Super-User on 2001-10-12

Next Group Up Previous

MOUNT STROMLO & SIDING SPRING OBSERVATORIES CCD 1.2
Computer Section

2 August 1994

CCD User Interface System - Functional Specification

Contents

Introduction

This document describes the required functionality of the CCD User Interface System (CUIS) which is to be developed by the MSSSO Computing Section. The operation of each component of the system is described in general terms -- the details of how each operation is to be implemented are beyond the scope of this document. However, it is envisaged that the software will be written in C++ and/or C with code written for CCD controllers as required. The eventual aim is that the system will run under Solaris 2 on a Sun workstation with the graphical part of the interface built using the Motif widget set.

Required features of the MSSSO CCD User Interface System

This section lists, in no particular order, those features of any CCD user interface which would be required before such a system could be considered useful to observers using MSSSO CCDs.

General Requirements

The following requirements are taken from the CUIS Requirements Specification.

- provide a common interface to all CCD systems regardless of CCD-vendor supplied hardware or software
- deliver the images to the image tool, disk and/or tape in similar amount of time to the current Astromed 3200 system.
- a Graphical User Interface (GUI) which adheres to de-facto standards for the ‘‘look and feel’’ of such an interface. The idea being that observers are presented with an interface which operates in a way familiar to users of Openwindows.
- provide on-line ‘‘help’’
- provide LARGE exposure counters, audible end-of-exposure alarms
- use colour coded windows and widgets adhering to Australian Standards for layout and colour of control panels. This will ensure that the CCD interface and the telescope control system will be using the same colour codings for warnings, errors, OK status, etc.

Set up and Data Acquisition

- provide facilities for technical staff to test CCDs and fine tune the hardware setup of CCD controllers
- provide a focussing ‘‘mode’’. (e.g. the system takes a series of exposures at different focus settings allowing the observer to see the results of each exposure. During the focus exposures the telescope focus is altered manually or automatically depending on the observers preference and/or the sophistication of the CCD system/Telescope Control system interface.) Software must be provided which will calculate Point Spread Functions and assist the observer to obtain the best focus.
- ability to define and read out regions of the CCD. Interaction (using the mouse) with the Image Display Tool will allow for rectangular regions to be specified on a previously taken image. The number of regions which can be specified will be limited by the capabilities of the CCD controller.
- ability to alter the row and column binning of CCD pixels. This will permit different binning in different regions where the CCD controller allows this
- initialise the CCD hardware.
- provide a facility to perform drift-scan exposures. This will mean being able to read out one row of the CCD upon receipt of a signal from the telescope control system.
- provide facilities for performing time-resolved imaging and spectroscopy
- allow for control of multiple CCD systems (e.g. as in operation at the DBS). This will require synchronising the start of exposures on multiple CCD controllers. Status information from the multiple controllers will be required. Multiple fits files will be written. (For the DBS, allowance will be made for use of only one side of the DBS and for different exposure times to be specified for each CCD controller.)
- allow for exposures to be interrupted and restarted or aborted and allow for exposure times to be changed when an exposure has been ‘‘paused’’
- allow for exposures to be automatically repeated. The number of such exposures will be set by the observer.
- provide the observer with the facility to vary the number of flushes performed before each exposure
- Allow for the use of table-driven sequences of instructions (e.g. a series of exposures, changing filters between exposures). The system will provide a standard set of such tables and allow the observer to specify new sequences.
- provide filter wheel control where this is available on the CCD controller.
- provide basic graphical image analysis tools to help with checking data quality, calculation of required exposure times, etc. Such tools will include summing a number of rows or columns and producing a plot of the data.
- provide facility to save and reload an individual observer’s set-up files. Data such as default image directories, optional fits header keywords, the observer’s own sets of instruction sequences, observer defined CCD regions will be held in such a file

Data Output

- write fits files with header information obtained from the telescope and the observer. This will include object name (from the telescope system) where possible

- allow for all hardware set-up information to be included in fits headers.
- allow observer to select from all available possible fits header keywords. There will be a minimum set of ‘‘mandatory’’ keywords (e.g. RA, Dec, dates) and many specialised keywords (e.g. CCD hardware info) from which the observer can choose. The set of required keywords will be stored in the users set-up file.
- place fits files in a user specified directory. This will allow the observer to place images in the /home area or on one of MSSSO’s data or book disks
- allow the observer to specify a fits filename to which <run number>.fits will be appended.
- display details of available disk space

System Status Information

- provide feedback on system status. This includes such things as voltages and temperatures as well as the ‘‘dead or alive’’ status of any software sub-processes
- provide feedback on progress of exposures, readouts, initialisation (even when window is ‘‘iconified’’)
- provide a restart facility should any part of the system fail. This will include advice to the observer that an error has occurred and that part or all of the system should be restarted. Of course, wherever possible, any error recovery will be carried out without requiring any action on the part of the observer.

About this document ...

This document was generated using the **LaTeX**2HTML translator Version 99.2beta8 (1.43)

The command line arguments were:
latex2html `-init_file /opt/local/html/latex2html/cicada.init ccdfs.tex`

The translation was initiated by Super-User on 2001-10-12

Next Group Up Previous

MOUNT STROMLO & SIDING SPRING OBSERVATORIES
Computer Section

October 12, 2001

Computerised Instrument Control and Data Acquistion System (CICADA) Programmers Guide

Contents

Introduction

This guide describes the MSSSO CICADA software system design from a programmer's point of view. Other documentation describes the rationale for developing the system, this instead will concentrate on describing the techniques adopted to develop the software.

Description

The aim of CICADA is to build on the techniques developed during the CCD system prototype [#!ref1!#] That is, to extend the CCD prototype to be a general MSSSO telescope instrument interface using the X Window system from an observers Unix workstation.

As shown in Fig 1, CICADA consists of a process group running on the observers workstation and potentially more than one process group running on perhaps more than one instrument computer (which may also be the observers workstation).

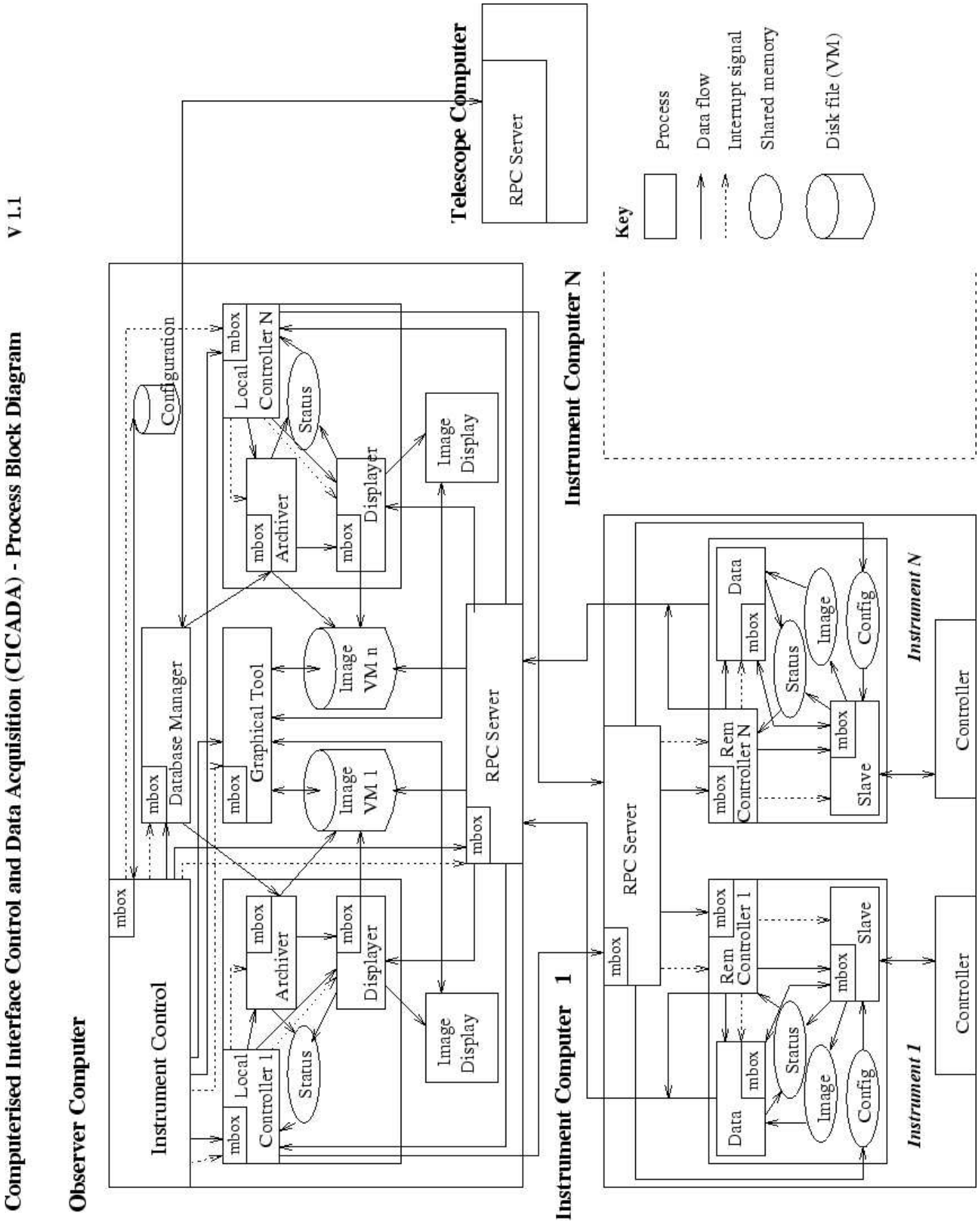


Figure 1: CICADA Block Diagram

Note that the block diagram is drawn in a way that illustrates the hierarchy in the process chain. A process box which encloses other process boxes is a parent to those enclosed and has responsibility for organizing

their execution (starting, stopping and ensuring they are still alive when required).

Inter process communication (IPC) between processes on the same machine is achieved by using standard Unix SysV IPC routines (see [#!ref2!#]), while communication between the two machines is by Remote Procedure Call (RPC) routines (see [#!ref3!#]) across an ethernet link.

The basic functioning of the system requires that an observer sitting at the Sun workstation be able to issue a command to the instrument and receive feedback (image data or status information) at the workstation without having to know anything about the instrument computer at all. The image data needs to be displayed and then archived as FITS files in real time (limited by the speed of the ethernet link). Full functionality is described in the Functional Specification (see [#!ref4!#]).

This basic functionality is extended to allow the observer to monitor more than one instrument at, possibly, more than one intrument computer simultaneously. Obviously this model is constrained be hardware at the CPU, display and communication levels, but should not be constrained by software.

Component description

The observer computer runs the process *Instrument Control* which spawns processes *Database Manager*, *Graphical Tool and RPC Server*. When the observer selects an instrument *Instrument Control* will spawn a *Controller* process which in turn makes contact with the instrument computer and starts up a remote *RPC Server* and remote *Controller* process. This group of processes form the link between observer and instrument and are the essential components of the system. The following pseudocode algorithms describe the specific functionality of each of these components:

- *Instrument Control*

```
Start ‘‘Database Manager’’, ‘‘Graphical Tool’’ and ‘‘RPC Server’’.
Establish mailbox.
Loop
 Wait on observer requests and mailbox messages.
 If new instrument requested start instrument ‘‘Controller’’.
 otherwise post requests to instrument ‘‘Controller’’ mailbox.
Endloop
```

- *Database Manager*

```
Establish mailbox.
Loop
 Wait on requests for data from mailbox.
 Collect data from telescope or cache, update cache if necessary.
 Post data to requesting process’s mailbox.
Endloop
```

- *Graphical Tool*

```
Establish mailbox.
Loop
 Wait on requests from observer for graphical analysis of image
  data.
 Connect to image in virtual memory, do computation and display
  results.
Endloop
```

- *RPC Server*

```
Establish mailbox.
Loop
 Listen for RPC image data requests on network, and watch for
  mailbox messages fron ‘‘Instrument Control’’.
 Interpret request and post to appropriate ‘‘Controller’’
  process which will be a ‘‘Displayer’’ sub-process.
Endloop
```

- *Remote RPC Server*

```
Loop
 Listen for RPC instrument requests on network.
 Interpret request and post to appropriate ‘‘Remote
  Controller’’ process depending on instrument. That is,
  if Instrument Computer hosts more than one instrument, the
  ‘‘Remote RPC Server’’ needs to pass requests to the
  correct ‘‘Remote Controller’’ process.
 Start and restart ‘‘Remote Controller’’ processes as
  appropriate, ie, monitor that they are still alive.
Endloop
```

- *Controller*

```
Establish mailbox.
```

```
Start sub-processes ''Archiver'', ''Displayer'' and ''Remote''.
Start remote ''RPC Server'' and ''Remote Controller''.
Establish a shared memory segment for status.
Loop
 Check for user requests from mailbox.
 If any requests current, monitor progress and sub-process status.
 Restart any sub-processes or remote processes if failed.
 If new request pass on to sub-processes. ''Remote'' handles
  requests to instrument, ''Displayer'' handles requests to
  get data from the image display and ''Archiver'' takes
  requests to create new FITS files (memory mapped).
 Display progress and status information.
Endloop
```

- *Remote Controller*

```
Establish mailbox.
Start sub-processes ''Slave'', and ''Data''.
Establish shared memory segments for status and image data.
Loop
 Check for user requests from mailbox.
 If any requests current, monitor progress and sub-process status.
 Restart any sub-processes if failed.
 If new request pass on to process ''Slave''.
 Post status and progress requests back to ''RPC Server'' on the
  Observer Computer.
Endloop
```

- *Archiver*

```
Establish mailbox.
Establish interrupt handler (can be interrupted by ''Controller'').
Connect to the status shared memory segment.
Loop
 Check for requests in mailbox from ''Controller''.
 Requests will be either to create a new FITS file, gather
  telescope information (post request to ''Database Manager''
  to construct the FITS header and close the FITS file.
 Post a ''ready'' message to the ''Displayer'' process when
  FITS file ready for data. (Or use semaphore to control access)
 Place regular heatbeat signal onto status memory.
Endloop
```

- *Displayer*

```
Establish mailbox.
Establish interrupt handler (can be interrupted by ''Controller'').
Connect to the status shared memory segment.
Loop
 Check for requests in mailbox from ''RPC Server'' to display
  and store image data from the instrument.
 Requests will be either to start a new image, build on an already
  started image or to close it off.
 If the request is to start a new image then wait for a ''ready''
  message from the ''Archiver'' process (or wait on semaphore)
  before copying data to virtual memory FITS file.
 Place regular heatbeat signal onto status memory.
Endloop
```

- *Remote*

```
Establish mailbox.
Establish interrupt handler (can be interrupted by ''Controller'').
Connect to the status shared memory segment.
Loop
 Check for requests in mailbox from ''Controller'' to formulate
  RPC requests to the instrument computer (or, if the instrument
  is local, post mailbox messages to the ''Remote Controller'').
 Post results of status requests onto the status memory.
 Place regular heatbeat signal onto status memory.
Endloop
```

- *Slave*

```
Establish mailbox.
Establish interrupt handler (can be interrupted by ''Remote
 Controller'').
Connect to the status shared memory segment.
Connect to the image shared memory segment.
Loop
 Check for requests in mailbox from ''Remote Controller'' for
  instrument action requests.
 Instrument requests will involve loading the controller with
  appropriate commands, starting, pausing, resuming, aborting
  the controller and retrieving data from the controller.
 Post regular progress information onto status memory during
  an instrument request, otherwise post heartbeat information.
 Image data retrieved from the controller is placed into the
  image shared memory when ready (Either wait for ''ready''
  message in mailbox from ''Data'' or wait on semaphore).
  Signal ''Data'' that new data is ready.
Endloop
```

- *Data*

```
Establish mailbox.
Establish interrupt handler (can be interrupted by ''Remote
 Controller'').
Connect to the status shared memory segment.
Connect to the image shared memory segment.
Loop
 Wait for requests in mailbox from ''Slave'' for new image
  data (or wait on semaphore).
 Formulate RPC request to Observer Computer with the image data.
  (or if on same computer post mailbox message to
  ''Displayer'').
 Post regular heartbeat information to status memory.
Endloop
```

Error Handling

In a distributed processing environment like CICADA it is easy for the error locality and the notification interface to become detached thus making reporting difficult. In CICADA it is proposed that the *Local Controller* process for each instrument be responsible for reporting all error conditions. Therefore any errors generated by child processes (local and remote) should be propagated back, through the IPC interface, to the *Local Controller* for reporting.

Within a process, the C++ exception handling facilities [#!ref5!#] will be used to concentrate error processing at one location in the code thus eliminating the need for explicit error handling right through. Size and complexity of the code is, hence, also reduced.

For example the pseudocode version for the *slave* process, rewritten with exception handling, becomes:

```
try {
 Establish mailbox.
 Establish interrupt handler (can be interrupted by ''Remote
  Controller'').
 Connect to the status shared memory segment.
 Connect to the image shared memory segment.
 Loop
  try {
   Check for requests in mailbox from ''Remote Controller'' for
    instrument action requests.
   Instrument requests will involve loading the controller with
    appropriate commands, starting, pausing, resuming, aborting
    the controller and retrieving data from the controller.
   Post regular progress information onto status memory during
    an instrument request, otherwise post heartbeat information.
   Image data retrieved from the controller is placed into the
    image shared memory when ready (Either wait for ''ready''
    message in mailbox from ''Data'' or wait on semaphore).
    Signal ''Data'' that new data is ready.
  }
  catch(CICADA_error E) {
   Post error condition to status memory, or rethrow to outer
   error handlers.
  }
 Endloop
}
catch(CICADA_error E) {
  Post error condition to status memory
}
```

About this document ...

This document was generated using the **LaTeX**2HTML translator Version 99.2beta8 (1.43)

The command line arguments were:
latex2html `-init_file /opt/local/html/latex2html/cicada.init cicada_design2.tex`

The translation was initiated by Super-User on 2001-10-12

Next Group Up Previous

CICADA V3.1
User's Manual

Computer Section
Mount Stromlo and Siding Spring Observatories
Research School of Astronomy and Astrophysics
Australian National University

December 13, 2000

Contents

List of Figures

Introduction

This manual describes the operation of the CICADA system (Configurable Instrument Control and Data Acquisition). This version of the manual accompanies release 3.1 of CICADA. To find out what changes have been made since earlier releases, select *What's New* from the CICADA main window *Help* menu.

Version 3 of CICADA supports the Astromed 3200 controller and the SDSU Generation 1 and Generation 2 controllers.

CICADA features

The aim of the CICADA system is to provide a common interface to many different CCDs and several different types of CCD controllers. It has been designed and built to be as flexible as possible so that new CCDs can easily be incorporated into the system and the support of new controller types can be provided in a relatively straightforward manner. Version 3 furthers gains made with version 2 - the code has, again, been extensively re-worked to provide a more general software solution for multi-part instruments and CCD mosaics. It also supports the simple control of instrumentation associated with a CCD camera.

The system is comprised of several unix processes each of which handles one aspect of the data acquisition and control process. These processes are divided into two groups - observer control and instrument control, and may run on one or more computers. For example, in the observer control group there is a process which displays images, another which writes FITS files, and a graphical user interface (GUI) with which the observer monitors and controls the system. In the instrument group, there is a slave process which communicates with the CCD controller, a data process which moves image data and a director process for managing operations. More than one instrument group is possible allowing for the control of multiple-part instruments. The use of several processes allows a certain amount of ''parallelism'' in the process of taking an exposure (e.g. a section of an image is displayed while the next section is being read out) resulting in some system performance gains. Further parallelism is achieved through the use of multi-threading in the internals of these sub-processes where appropriate.

For a complete description of the CICADA design, refer to CICADA technical documentation at http://www.mso.anu.edu.au/computing/cicada/

Getting Started

Quick Start Guide

CICADA is predominantly GUI-based observing software with a familiar ''look-and-feel'' and an intuitive interface. If you want to start up CICADA and get going quickly without first reading the rest of this manual, do the following (presuming CICADA hardware configuration has already been performed):

- type *cicada* and press <Return>
- once the CICADA menu-bar appears, select *Start Ximtool* from the *Tools* pull-down menu
- select the Telescope/Instrument you are observing at from the *Start Observing* pull-down menu
- optionally select *Start GIT* from the *Tools* pull-down menu
- set your desired exposure options and start observing.

You will now be taking data and FITS files will be written to a default location (probably your home area). You may wish to alter your output directory via *Preferences* from the *Options* pull-down menu.

Use the *Start Teldisk* option of the *Tools* pull-down menu to clear the previous observer's data from any ''observer's'' disk and create a directory for your data. This will depend on the disk configuration at your site.

Requirements

To run CICADA, you will need an account and sufficient disk space to store captured images. Start CICADA by using the command *cicada*, you will then be presented with a menu bar from which an instrument and configuration can be selected. CICADA runs in conjunction with the Graphing and Imaging Tool (GIT) and a modified version of Ximtool - these must be started from within CICADA so that correct environments and interconnections are set up. Use the *Start Ximtool* option of the *Tools* pull-down menu (if you forget to do this, CICADA will automatically start Ximtool the first time you start an exposure).

CICADA can also run in conjunction with its own (IRAF based) data reduction pipeline. Use the *Start Pipeline* option to get this going - refer to the Pipeline User's Manual for more information.

Take Care

NOTE: CICADA and GIT can be very memory hungry! How much memory the system requires depends on the size of images you are taking. Both programs indicate when memory is getting low, so please take note of these warnings as *performance* can be affected. In particular,

- *Limit running of non-CICADA processes.* Other utilities and packages such as `Netscape` and `IRAF` can compete with CICADA for system resources - try to run these on other hosts on the network.
- *Stop and restart CICADA if in trouble.* If system resources have been drained - it is fast and simple to stop and restart the observing window. This will recover system resources to the starting point, provided no other memory hungry processes are running.

Configuring fonts and colours

Setting up fonts and colours for X windows applications can be a complicated process. X applications use ''resources'' to determine the fonts and colours they should use when displayed on the screen. If you wish to change the default colours and fonts for CICADA, you will need to add the following lines to your `.Xdefaults` file:

```
cicada*fontList: -*-lucida sans-medium-r-*-*-*-100-*-*-*-*-*-*=TAG1,\
                 -*-lucida sans-bold-r-*-*-*-110-*-*-*-*-*-*=TAG2
cicada*background: wheat
cicada*foreground: black
cicada*selectColor:   yellow
```

To select a font you like, use the program `xfontsel` which allows you to preview various fonts and then cut and paste your chosen font descriptor into your .Xdefaults file. Note that there are two fonts listed in the `fontlist` resource. CICADA uses the first font for most of the labels in all windows. The second font is used to highlight various buttons and text fields such as the status indicators. You do not need to specify a second font as CICADA will use the first font you specify for all labels and text if no second font is in the fontlist. Note the syntax of the fontlist resource as the `=TAG1` and `=TAG2` are very important.

The `cicada*background` resource sets the colour of all window, button, etc. backgrounds. Text on buttons and labels will be in the colour specified by the `cicada*foreground` resource. `cicada*selectColor` sets the colour used by toggle buttons when they are turned ''on''.

Similar selections may be made for other CICADA tools, that is, GIT, Pipeline. GSFind and Teldisk.

CICADA Main Menu Bar

Figure 3.1 shows the CICADA Main Menu Bar. This is the first window the observer will see upon starting CICADA.

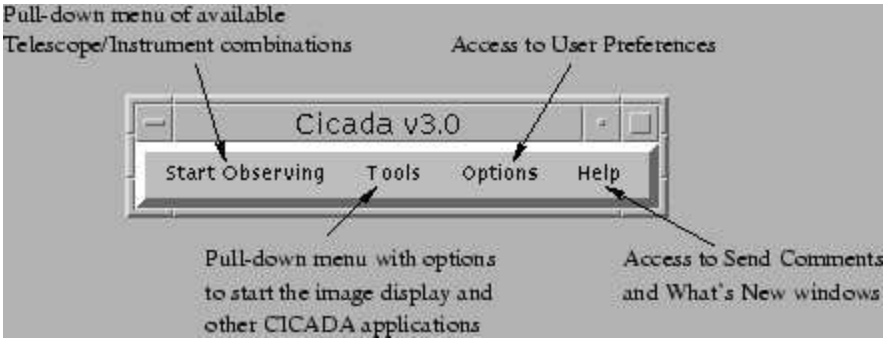


Figure 3.1: CICADA startup window

Start Observing

This pull-down menu contains a list of possible telescope/instrument configurations. The observer selects the particular telescope/instrument for which they have been scheduled. CICADA has an internal table which is configured by technical staff that indicates to the system such things as the CCD in use on the selected instrument, the controlling computer, the controller type, etc. After selecting the appropriate item, an observing window will appear.

Data Archiving

As part of supporting the archiving of RSAA's Wide Field Imager data, version 3 of CICADA can optionally request the observer to provide name and proposal ID details for inclusion in FITS headers. These details can be used for retrieving data from an archive at a later date. The proposal ID will be matched with a password and used as part of a system to restrict access to data in the archive. Figure 3.2 shows the Proposal ID confirmation window. This window will appear if the instrument has been configured to support archiving of data. If for some reason the details are incorrect, this window provides the observer with an opportunity to correct the error. It is also possible for the observer to select *not* to archive any FITS data.

Figure 3.2: CICADA startup window

Tools

This menu has items for starting the Graphing and Imaging Tool (GIT), Ximtool, Pipeline, Teldisk and GSFind.

GIT provides ''quick look'' plotting and data reduction tools - see the GIT User's Manual for full details.

Ximtool is used for displaying the image as it is read out. You may either start Ximtool manually via this menu option or it will be started automatically if display mode is set when taking an exposure. *CICADA uses a modified version of Ximtool and it MUST be started from within CICADA*

Pipeline is a data reduction pipeline that can run stand-alone or closely integrated with CICADA. It uses the IRAF ccdred and mscred packages for performing the actual work. The idea behind Pipeline is that it provides an automated way of performing simple recipe reduction on a collection of images.

GSFind is a Guide Star Finder that uses the Guide Star Catalogue and a description of the instrument/telescope to assist with locating guide stars. It is loosely coupled with CICADA and can also run in standalone mode.

The other option available under *Tools* is *Start Teldisk*. `Teldisk` allows the current observer to clear the previous observer's data from the ''Observer Data disk'' and create a new directory for storing FITS files. The configuration of disks is site dependent, so talk to the local Systems Managers about disk space for observers.

Options

Preferences

Preferences are user configurable parameters which affect the operation of CICADA. CICADA Preferences are stored in the observer's home area in a file called `.cicadarc-hostname`. The file is read once at CICADA startup time. This file is host name differentiated because of the possibility that a user might run more that one session of CICADA at a site with multiple instruments and CICADA observer hosts. It is important that these files are not mixed.

General Preferences

Figure 3.3 shows the Preferences window with the General Preferences displayed.

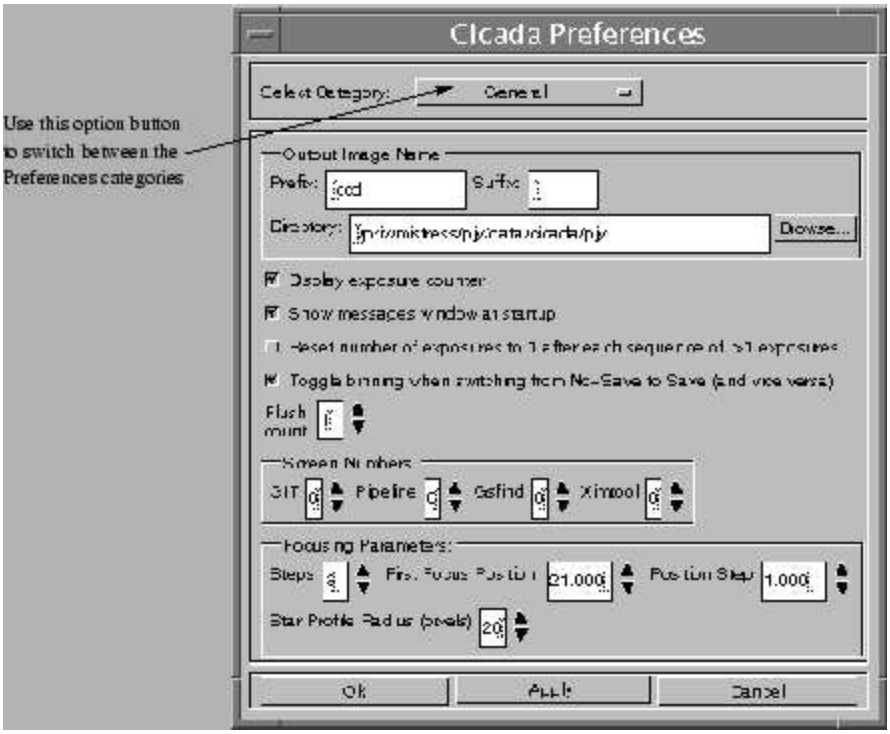


Figure 3.3: General Preferences

Use these preferences to specify the default filename prefix, suffix and location (output directory) for FITS files. There are also options here to:

- use the large counter in a separate window as well as using an exposure counter in the progress display of the observing window. Turn the toggle ''on'' to use the large counter in a separate window.
- show messages window at startup will cause the CICADA messages window to be displayed and positioned below the observing window when the first status messages are received from a CICADA sub-process (i.e. there may be a delay after the observing window has appeared before the messages window is seen).
- reset the *Number of exposures* field to 1 after running a sequence of exposures. e.g. (with this toggle ''on'') the observer enters ''5'' in *Number of exposures* and selects *Expose*. The five exposures are run and at the end of the last one, *Number of exposures* is automatically reset to 1.
- toggle binning when switching between Save and No-Save modes. e.g. the observer bins 4x4 while positioning the science object on the CCD and using *Display Only* mode. However, the observer uses 1x1 binning during data acquisition with *Save and Display* mode. With this toggle ''on'', CICADA will remember the two different binning factors and toggle the binning whenever the observer switches between *Save and Display* and *Display Only* modes.
- The number of CCD flushes to perform when doing an exposure.
- X window screen numbers to display each CICADA tool - only applicable for multi-screen workstations. By setting the screen number here, each of the tools will be started on the preferred screen.
- Automated focusing parameters. See section 4.9 on automated focusing for a description of how this works. Set the following parameters for running the sequence.
  - *Focusing steps* - the number of focus steps to perform in the sequence.
  - Value of the *first focus position* in the sequence.
  - *The focus position step size*. This can be used for systems that have a computer controlled focusing mechanism. The focus position is adjusted by this amount each successive step in the sequence.
  - The *radius* (in pixels) from the star centroid to use for fitting a gaussian.

Image Display & Scaling Preferences

Figure 3.4 shows the Preferences window with the Image Display and Scaling Preferences displayed.

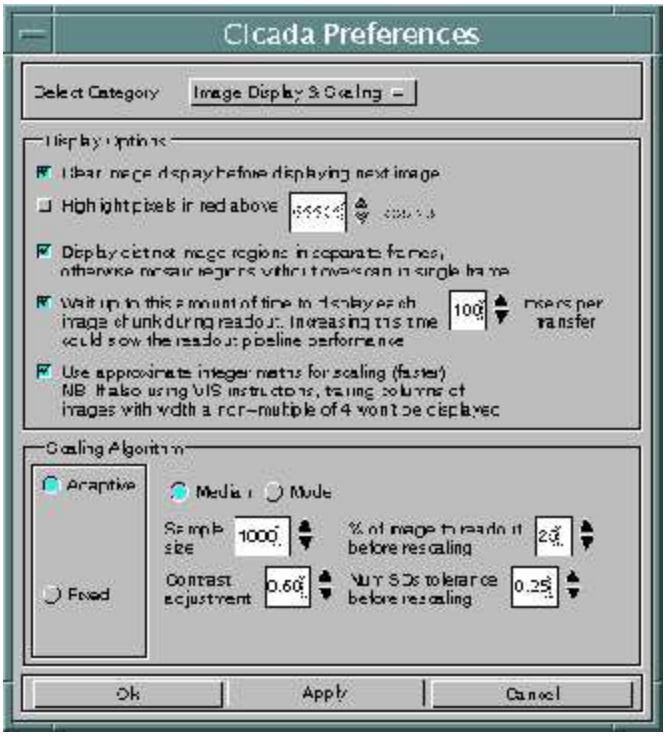


Figure 3.4: Image Display and Scaling Preferences

These property items affect the way in which images are displayed as they are read out:

- *Clear display before next image* will cause the image display to be cleared before the next image is displayed. When not selected, a following image will gradually overwrite the currently displayed image. This may be useful in focussing or positioning the science object on the CCD.
- *Highlight saturated pixels* will cause any pixels with a value greater than that specified in the ''saturation value'' field to be highlighted in red on the image display.
- *Display distinct image regions in separate frames,...* This option affects the way images are displayed from CCD mosaics, multi-amplifier CCDs and multi-region readouts. If ''on'', each separate region is displayed in a separate frame of the image tool. If ''off'' all regions are displayed in one image tool frame in ''detector'' coordinates.
- *Use approximate integer maths for scaling...* Uses shift division for placing each pixel into its display bin (for Ximtool this is a maximum of 200 levels). This, if used in conjunction with the Ultra Sparc VIS instruction set (see *General options*), can speed up image display by nearly 4x for most images.
- *Wait for image display during readout...* sets the maximum amount of time CICADA will wait for the image display to show a chunk of data before sending the next one. Small values will result in ''gaps'' in the image display, but does NOT mean data has been lost. The default value will result in good performance and zero or minimal gaps in the display.
- *Adaptive scaling* will calculate appropriate minimum and maximum image display values as each ''chunk'' of image is read out. Depending on the settings of *''% of image to readout before rescaling''* and *''Num SDs tolerance before rescaling''* the range is re-calculated after each chunk and the entire image redisplayed. Minimum and maximum values are calculated as a weighted deviation from the median or mode after sampling a proportion of the pixels. Better performance, but perhaps poorer scaling will result from decreasing the *number of pixels to sample*. The min/max range can be modified by altering the *contrast adjustment* value between 0 and 1.
- *Fixed scaling* allows the observer to specify the range of image display values to be used by the image display. This may be necessary in certain situations where adaptive scaling is not giving the desired results.

Sounds Preferences

These preferences control the number of beeps issued by Cicada at certain key points during an exposure. The observer can set the number of beeps to be heard when the shutter opens, the shutter closes and when readout finishes.

Expert

These preferences control the following:

- Whether or not the observer is warned at startup if CICADA cannot allocate a colour. CICADA uses a small number of colours on the GUI for highlighting buttons and warning of error conditions. If the observer is running a colour-hungry application such as Netscape, it is possible that CICADA may not be able to use one or more colours and so the software will try to find an alternative. If an alternative is needed or no colour can be found, a warning will be displayed at startup if this preference is ''on''.
- *Write an observations log*. When ''on'' CICADA will write a log of each image taken showing RA, Dec, run number, exposure time, sidereal time, object.
- *Show timestamps in messages window*. Whether or not to show a timestamp with every line in the CICADA Messages Window.
- *Show Balloon Help*. Turns on and off the ''balloon help'' on toolbar items
- *Show toolbar items as text, not pictures*
- *Use the Ultra Sparc VIS instruction set*. For improved performance in some operations. (eg byte swapping or image display).
- *Write a temperature log*. Selecting this option will allow a CCD temperature log be kept for the run.

Help

Send Comments

This will pop-up a window allowing the observer to enter a message which will be sent to the local CICADA administrators. Use this to report problems, ask questions or request changes/new features. CICADA is configured to have an email address of the local CICADA administrator. At RSAA this is staff of the Computer Section who will endeavour to respond promptly to your submission with a resolution or an estimation of when a resolution can be made.

What's New

This will popup a window showing a revision history of CICADA which outlines the major changes between versions.

About Cicada

Well, everyone else has one of these...

The CICADA Observing Window

Figure 4.1 shows the CICADA observing window. It is from this window that the observer initiates and controls data acquisition. The appearance of this window is largely the same irrespective of the particular controller used to drive the CCD (e.g. AM3200, SDSU Generation 1). Of course, each controller has certain unique features and these are accessed via a pull-down menu on the observing window. e.g. when using the AM3200 controller, the pull-down menu will be called `AM3200` and will have items such as *''Edit Gain Values''* and *''Continuous Flush & Read Temps''*.

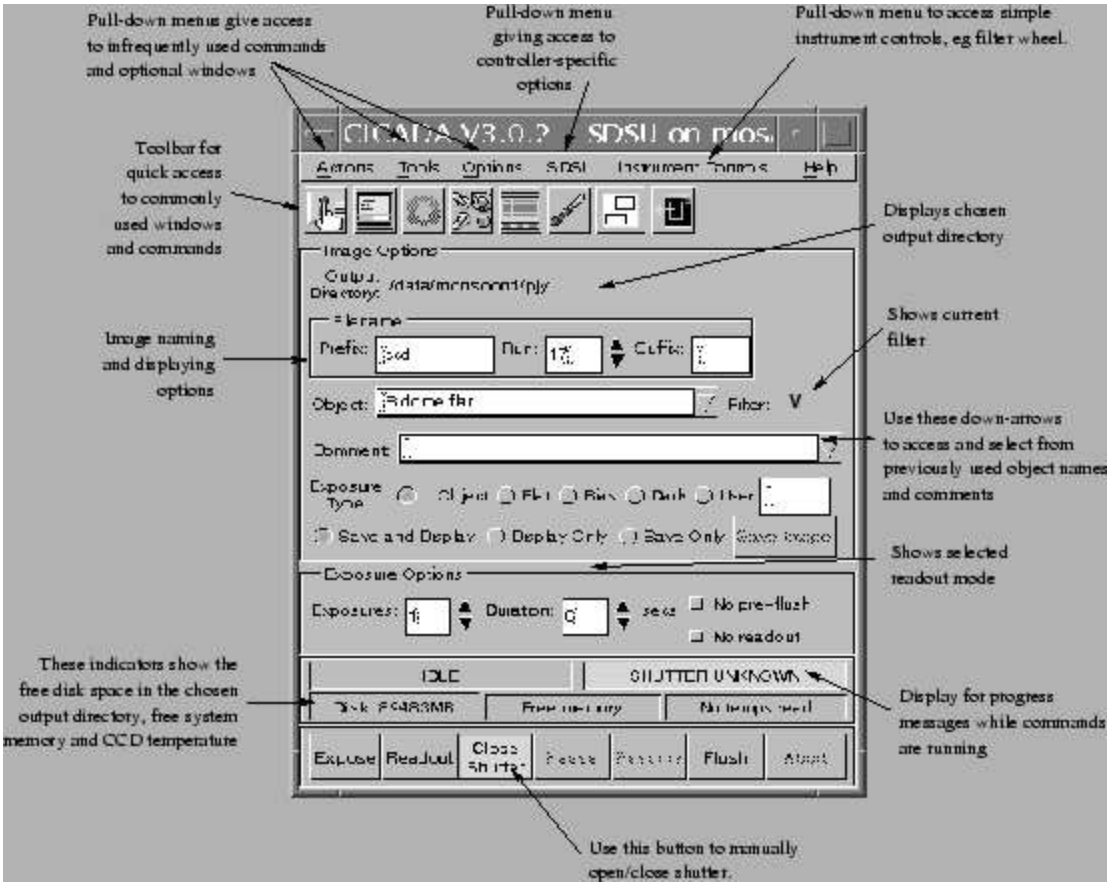


Figure 4.1: CICADA observing window

Menu Options

Actions menu

Initialise CCD Hardware

Selecting this action will cause hardware initialisation of the CCD controller to be performed. During initialisation status messages will be displayed in the `Cicada Messages Window` and progress will be shown in the status indicators of the Observing Window (see below).

Run CICADA Script...

This action will pop-up a file selection box allowing the observer to select a file containing a TCL script to be executed. The script must be a valid TCL script with CICADA commands. See Chapter 8 for further details.

Repeat Last Script

This item will become active once a script has been run and when selected will re-run the most recently executed script.

Do Focus Sequence...

A focus sequence consists of a series of exposures run with different telescope focus settings followed by analysis of the resultant image to determine the optimal setting. Focus settings are either set manually by the observer when prompted or, if the capability exists, automatically through the telescope control interface. A

plot of the calculated optimal setting (a quadratic fit through the set of points) is produced at the end of the sequence. The user is required to select a small region around a single star before performing the focus sequence.

Quit Observing Window

Close the Observing Window and shutdown all sub-processes. The observer should use this action when finished observing or if a restart of the CICADA sub-processes is required.

Tools

This pull-down menu is a copy of the *Tools* menu of the CICADA Main window and has items for starting GIT, Ximtool, Teldisk, Pipeline and GSFind.

Options menu

Preferences

Select this to pop-up the User Preferences window

Exposure Time Unit

Use this pull-right menu to change the time unit used when specifying exposures.

Show Regions Window

Selecting this will pop-up the Regions window allowing the observer to specify regions of interest on the CCD

Show Messages Window

This option will pop-up a messages window. Certain parts of the CICADA software send messages to this window and it may be useful to the observer to see these messages - for example, during a hardware initialisation.

Show Counter

The observer can use this option to re-display the counter window if it has been dismissed during an exposure countdown.

Show FITS Keyword Window

This will display a window where the user can enter any extra keyword/value pairs they would like to add to their FITS files. See section 4.8 at the end of this Chapter for more information on the FITS keyword window.

Hide Toolbar

Selecting this option will remove the toolbar from the observing window. The Toolbar can be redisplayed by selecting *Show Toolbar* from the *Options* menu (the Hide button changes to Show and vice versa as appropriate).

Controller menu

This menu will appear on the menu bar with a name such as *AM3200* or *SDSU* and holds all hardware specific actions and options for a particular controller type. See the relevant chapter in this manual for information about this menu item for specific controller types.

Instrument Controls menu

Note: This menu item will only appear if the CICADA instrument description includes site instrument control or filter control specifications.

Simple instrument controls are optionally configured for an instrument. This facility is provided as part of CICADA's ''plug-in'' capability. Each of the items under this menu is built as a simple instrument GUI plug-in - see the CICADA plug-in reference manual for further information.

An example simple control plug-in is one for a filter wheel (See Figure 4.2).

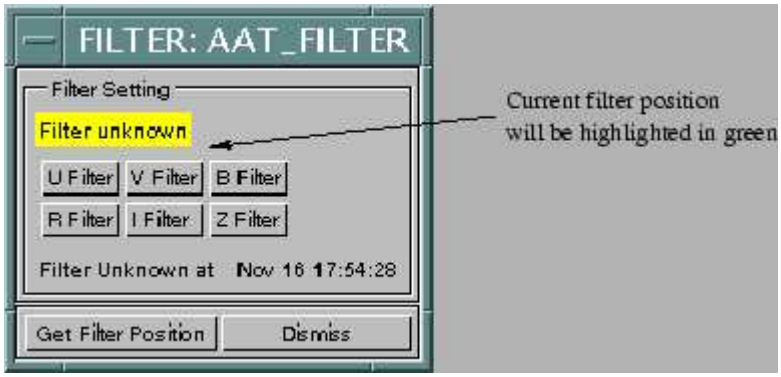


Figure 4.2: A Filter Control GUI

This popup window displays the current filter wheel position by highlighting the current position in green. The filter wheel is moved to another position by selecting the item corresponding to the desired filter. The selected position will flash orange while the filter wheel moves and will be highlighted in green once the wheel is locked into position.

Another example is one for telescope control (See Figure 4.3).

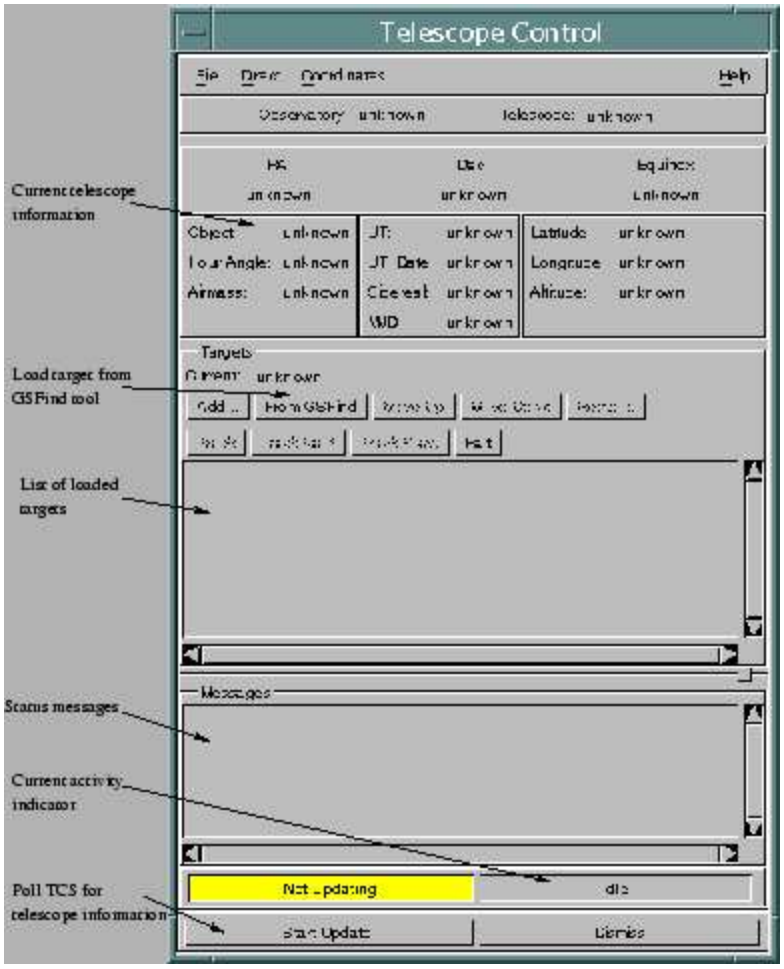


Figure 4.3: A Telescope Control GUI

Toolbar

The *Toolbar* is intended to give one-button-click access to some of the more commonly used items from the pull-down menus. ''Balloon Help'' will pop-up under the Toolbar item if the cursor is positioned in the Toolbar. (To turn off the balloon help see the *Expert* section of the *Preferences* window)

There are buttons in the Toolbar for *Hardware Initialisation, Run CICADA Script, Repeat last script, show Preferences, show FITS keyword window, show Temperatures window, show Regions window and Read Regions from Image Display* (See Figure 4.4).

Figure 4.4: GUI Toolbar Buttons

Image naming and displaying options

This section of the window displays the destination directory the observer has selected along with a filename prefix, suffix and run number which will be used to create a filename for the FITS file. There are also fields for entering a comment and object name to be added to the FITS header. CICADA stores up to 20 object names and comments. Use the down arrow next to the object name or comment field to select from previously used names and comments . New items can be added to these lists by simply entering a new value in the text field and hitting `<Enter>` or by entering a new value and taking an exposure. To delete an item, blank it out and hit `<Enter>`. These lists of names will be remembered by CICADA between runs (they are kept in the `.cicadarc-hostname` file in your home area *Note that the list of objects and comments may grow longer than 20 during one invocation of CICADA - but only 20 will be stored when the observer quits*.

If a filter is in use, the filter name will be displayed next to the object name.

Before the exposure is carried out, the system checks to see if a file already exists with the same name as the one about to be generated. If such a file is found, the observer is prompted with a pop-up window asking if this existing image should be overwritten. If the observer responds that the image should be overwritten, **THE EXISTING FILE IS IMMEDIATELY REMOVED**. Note also that NO EXPOSURE is happening while the prompt is displayed.

Exposure Type

The setting of these toggles affects the value of the IMAGETYP FITS keyword. In addition, when BIAS is selected, CICADA sets the exposure time to zero. The previous exposure time is restored when an option other than BIAS is subsequently selected.

The Expose button will change to the orange warning colour and the *Exposure Type* label will flash orange when anything other than *Object* is selected.

Save and Display options

Images read out from the CCD can optionally be displayed and optionally written to disk. Appropriate warnings are displayed on the GUI if the observer chooses NOT to save images. There is also a button in this section of the window which will allow the observer to save the last image taken when taking images in ‘‘no

save'' mode. This might be useful if the observer was taking a series of frames of the science object while focusing the telescope. Images are discarded until a good frame is taken, when pressing the *Save Image* button will write the last image to disk.

Exposure options

This section of the window is where the observer specifies exposure time and the number of exposures to be taken once the *Expose* button is selected.

No pre-flush and No Readout toggles

Normally, an exposure consists of a flush, followed by integration and readout. The observer can alter this sequence with these two toggles by choosing to omit the pre-flush and/or the readout. The Expose button will change to the orange warning colour when this toggle is selected and the selected toggle(s) will flash orange. If *No readout* is selected, *No pre-flush* is also automatically selected.

Status indicators

These indicators give the observer some feedback as to the state the system is in at any particular moment. The upper left-hand indicator will read one of *Idle, Running or Paused* and indicates the state of the process responsible for communicating with the CCD controller. The upper right-hand indicator gives more detail of the particular function currently being carried out. This indicator will display such messages as *Initialising, Shutter Open, Shutter Closed, Reading Out*. In addition to these messages, this indicator will show a green background when the shutter is closed and an orange background when the shutter is open (NOTE: this indicator is not controlled by any hardware and reflects only a status of the software. Therefore, if the shutter mechanism were to jam open or closed, this indicator WILL BE INCORRECT).
This indicator is also where an exposure counter will be shown if the observer has de-selected the CICADA preference *Display exposure counter*. During CCD readout, this indicator will show progress as a number of rows read out against the total to be read.

Also in this section of the window are three indicators displaying free disk space on the observer's chosen output disk, free memory available in the workstation and the current CCD temperature.

- The free disk space indicator will flash orange when there is not enough space for two full readouts of the CCD. When there is a warning about low disk space, either free some space on the disk by removing files or select another output directory via *Preferences* from the *CICADA Options* pull-down menu.

- The free memory display will flash orange when there is less than 16 Mb of free memory in the workstation. When there is a warning about low memory, quit some applications running on the same host as CICADA. In this situation, it might also be a good idea to quit and restart CICADA and GIT if you are using it.
- The CCD temperature indicator will display *HWinit needed* if the temperature has not been read or if the controller hardware needs re-initialising. The behaviour of this indicator is controller dependent. Additionally, the temperature indicator will change to an orange background if any focalplane temperature sensors measure a temperature which is out of range (the name of the sensor will appear in this indicator should this happen). Choose *Show Temperatures* from the controller-specific menu in the observing window (e.g. the menu labelled AM3200 or SDSU) to see all temperature sensors for the current instrument (or click on the thermometer button in the toolbar).

Buttons

Here is a brief description of the function of each button:

- *Expose:* Start an exposure using the various fields in the GUI for exposure parameters, file name, destination directory. This button will be orange if the observer has selected an option in the GUI which may result in no data being written to disk or if some other non-standard exposure option has been chosen (e.g. dark frame, no readout, no pre-flush). CICADA will check to see if there is already an image with the same name as the one about to be created and prompt the observer to ask if the existing file should be overwritten. CICADA will also check to make sure there is sufficient disk space for the new image and sufficient free memory in the workstation for the readout to be completed.
- *Readout:* Read out the CCD, display the image and save the FITS file (if save to disk has been selected).
- *Pause:* Pause the exposure by closing the shutter and stopping the counter. The observer may then continue the exposure by selecting the *Resume* button. It is also possible to change the exposure time and number of exposures while paused and these new values will be used when the exposure is resumed. Note that even though the shutter is closed, dark current and cosmic rays are still being accumulated on the CCD.
- *Resume:* Continue the paused exposure
- *Flush:* Flush the CCD
- *Abort:* Stop whatever is happening (eg. exposure, readout). **Note that aborting an action may take from a few seconds to several 10's of seconds. This is because, depending on the controller, some actions cannot be immediately interrupted and must finish before the abort request can be carried out and the GUI reset to allow further action.**

The Regions Window

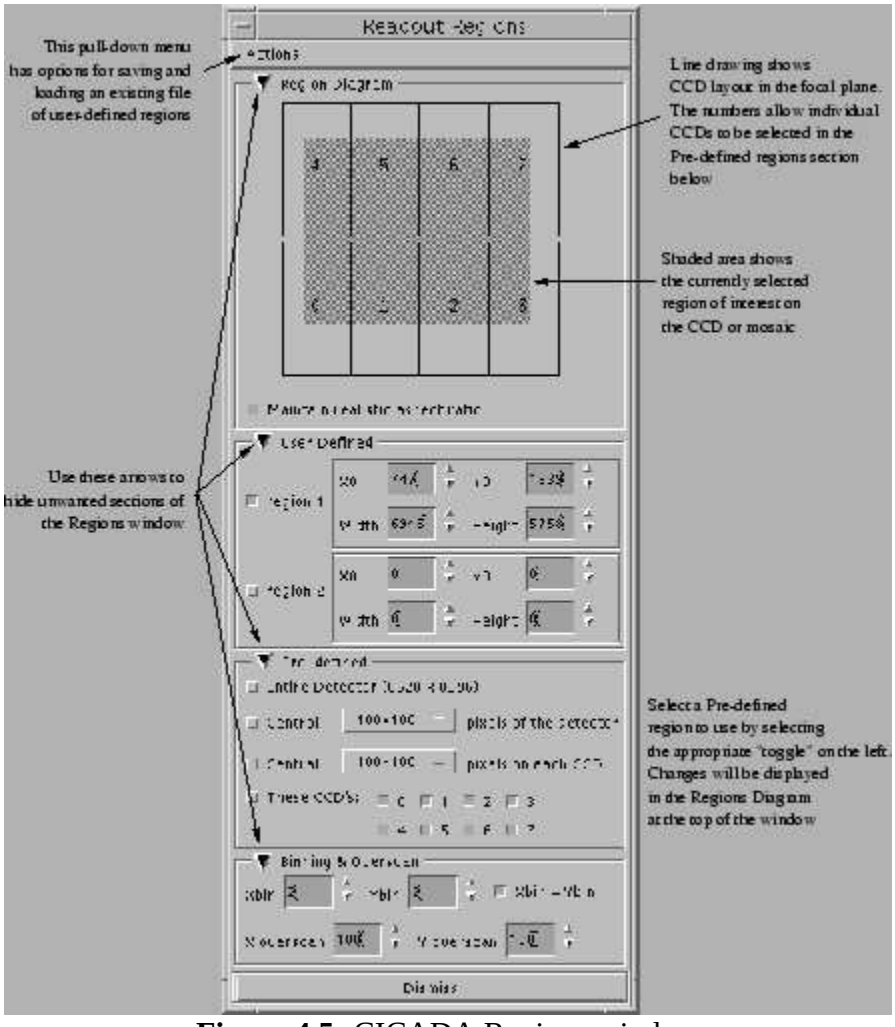


Figure 4.5: CICADA Regions window

The Regions Window is where the observer can specify the region(s) of the CCD or CCD mosaic to be read out and saved to disk. There are also fields in this window for changing the row and column binning factors and for specifying row and column overscan.

The window has five main sections:

- **The Region Diagram** This diagram shows the layout of the CCD(s) on the focal plane and will display a shaded area indicating the currently selected region(s) of interest. The diagram will be updated as changes to the region of interest are made in other section of the window.

- **User Defined regions** This section can be used to manually key in a region of interest. However, a simpler way to set the values in these fields is to have CICADA do it via the *Read Regions from Image Tool* feature. To use this, first do a full readout of the CCD or mosaic, then mark the region of interest by drawing a box on the image display (click mouse-button 1 and drag in the Ximtool window), then select *Read regions from imtool* from the Actions menu in the Regions window. There is also a button on the observing window toolbar to perform this function. It is possible to mark either one or two regions in this way.

  Also in this section of the window are settings which allow the observer to specify which of the two regions is to be read out (both regions can be selected simultaneously). Attempting to turn ''off'' the only ''on'' toggle will automatically turn ''on'' the other toggle and display that region.

- **Pre-defined regions** This section allows the observer to select from the following regions of interest:
  - Entire detector
  - A region in the centre of the detector. The size is selected using the option button. The sizes available are determined dynamically by CICADA based on the size of the entire detector
  - A region in the centre of each CCD. The size is selected using the option button. The sizes available are determined dynamically by CICADA based on the size of each CCD.
  - selected CCDs. The numbers in this section correspond to the numbers shown in the Regions Diagram. This ''masking'' can be used in conjunction with selecting a region in the centre of each CCD

- **Binning and Overscan** This section contains input fields where the observer can specify row and column binning and also the amount of row and column overscan. Note that when overscan is specified, it will NOT be shown in the image display for CCD mosaics if the display is showing the entire mosaic (overscan will be seen if each CCD is shown in a separate frame of the image display). The overscan IS written to the FITS file, regardless of what the display is showing.

- **Actions pull-down menu** The Actions menu at the top of the window has buttons which allow the observer to:
  - Read the region(s) marked on the image display
  - reset the region values to their initial state (full CCD for region 1 and nothing for region 2)
  - Save the currently defined regions to a file
  - Load regions from an existing regions file (e.g. to use the regions from a previous observing run)

The FITS Keyword Window

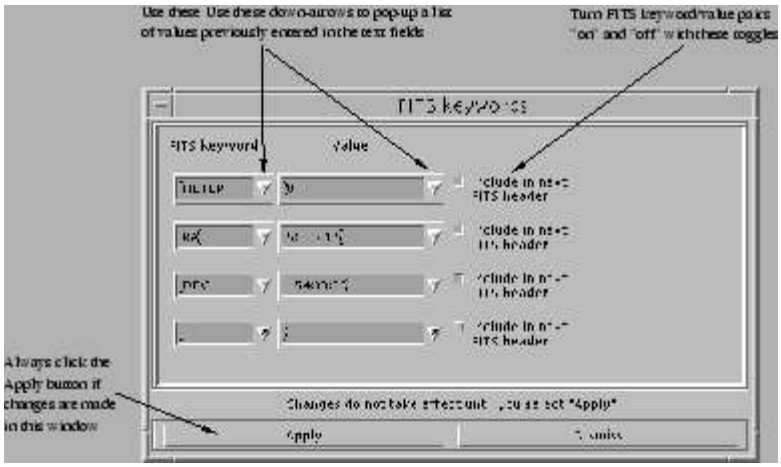


Figure 4.6: CICADA FITS keyword window

The FITS Keyword Window (Figure 4.6) allows the observer to enter keyword/value pairs to be added to the header of each FITS file written by CICADA. These keywords are in addition to any standard keywords provided by CICADA (e.g. EXPTIME) or by any telescope system (e.g. AIRMASS). The intention of this window is to allow CICADA users to enter keywords that the system does not supply (e.g. where there is no telescope computer or only a manual filter wheel).

Keywords and values can simply be typed into the text fields. CICADA will remember up to six items for each keyword and value field. Use the down-arrow to the right of the field to see a list of previously entered items and click on an item to select it. The list may grow longer than six items during one invocation of CICADA, but upon exit, CICADA will only store the six most recently used items.

NOTE: Changes made in these text fields will not propagate into FITS files unless the *Apply* button is pressed. Visual confirmation that your changes have been communicated to the CICADA sub-processes will be in the form of a command string in the CICADA Messages Window. Provided the *Apply* button is pressed before a readout starts, it is possible to modify these values during an exposure and have these modified values appear in the FITS file about to be produced.

Each keyword/value pair has a toggle next to them labelled ''Include in next FITS header''. This toggle can be used to stop a keyword/value pair from being added to a FITS header. Turning all these toggles off will disable all user-specified FITS keywords. Remember to press the *Apply* button if you change these toggles.

CICADA stores any keywords and values that the observer enters in the `.cicadarc-hostname` file in the observer's home area so that they remain available after a restart of the CICADA software. When an observing window starts up, CICADA will check to see if there are any ''active'' user-specified FITS keyword/value pairs left-over from a previous CICADA run. If any are found, CICADA will prompt the observer to either view these keywords (and then press *Apply* if they are to be used) or decide not to use them at all.

FITS keywords are edited by CICADA to make sure they have no leading or embedded blanks and that they are uppercase. Embedded blanks will be replaced by a hyphen (''-'').

If the keyword ''FILTER'' is specified, the value associated with this keyword is displayed next to the object name in the observing window.

A full description of CICADA FITS file construction is contained in chapter 7.

Automated Focusing

An automated focus sequence consists of a series of exposures taken of a star with incremental focus shifts

made between each exposure. These exposures are stacked together to form a composite image to which a focus fitting routine is applied. The FWHM of each star is measured and then a quadratic is fitted to determine the optimal focus position. This fitted curve is then plotted and an indication of the minimum is displayed.

When *Do Focus Sequence...* is chosen from the *Actions* menu the window shown in figure 4.7 is shown.

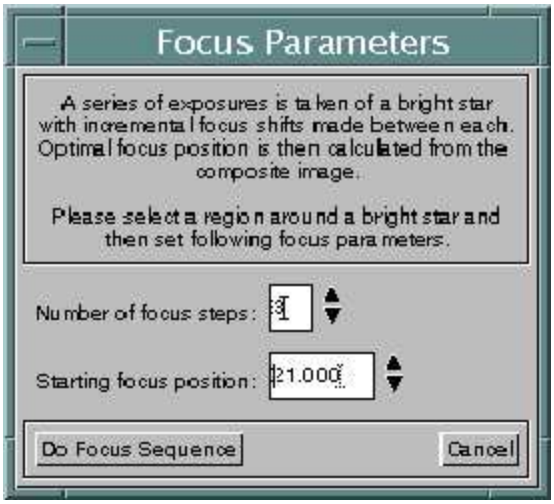


Figure 4.7: CICADA Automated Focusing Parameters

Select the number of focus steps to make and the initial focus position. CICADA will then expose the selected region (this should be a small region around a star) for the required exposure time. After the exposure, depending on whether the telescope has computer controlled focus setting or not, CICADA will prompt for the next focus position. The observer should set the focus position and enter the value in the dialog box before proceeding with the next exposure in the sequence. The composite image will then look like figure 4.8.

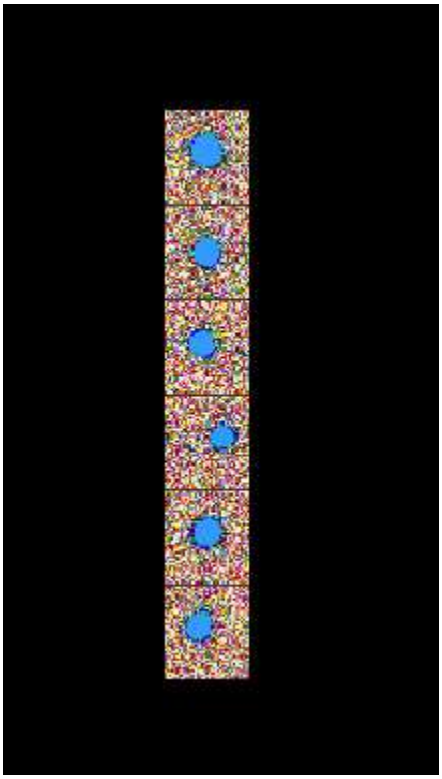


Figure 4.8: CICADA Focus Image

CICADA will then calculate the star centroid and fit a gaussian using the user preferences. The result will be a quadratic fit of FWHM and focus position which is then displayed. See figure 4.9.

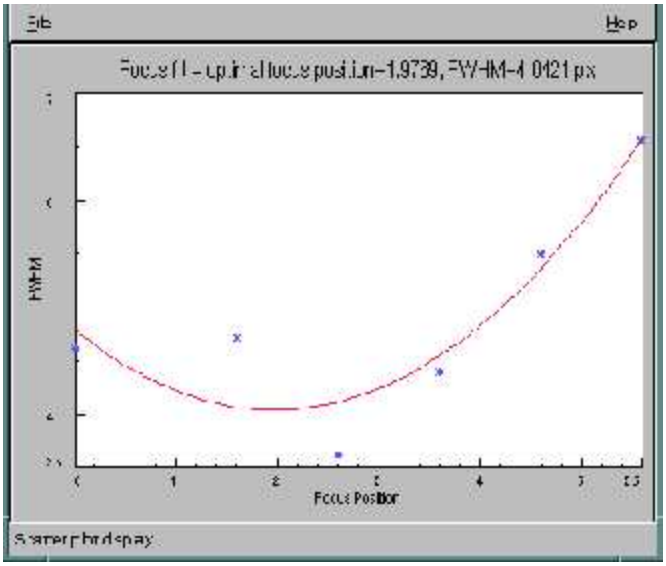


Figure 4.9: CICADA Focus Fit

The SDSU controller

Initialisation of the SDSU

CICADA will **not** initialise the SDSU controller when the observing window starts up. This is because re-initialising will mean resetting the controller electronics and possibly destabilising the CCDs. However, the observer can re-initialise the controller at any time using the *Initialise CCD Hardware* option from the *Actions* menu of the Observing Window. This should only be performed as a last resort when all other remedies for restoring normal operation have failed. Progress of an initialisation will be shown in the *Cicada Messages Window*.

The SDSU menu

These are the SDSU specific menu items appearing under the *SDSU* menu button in the Observing Window:

Readout modes

The SDSU controller has been set up to allow the selection of different readout modes. These are selected from the SDSU menu on the observing window. There is a menu item *Readout mode* which is a ''pull-right'' menu from which it is possible to select *Fast, Medium or Slow*. Once a selection is made, the mode will be shown in the exposure options section of the observing window. See

Figure 5.1

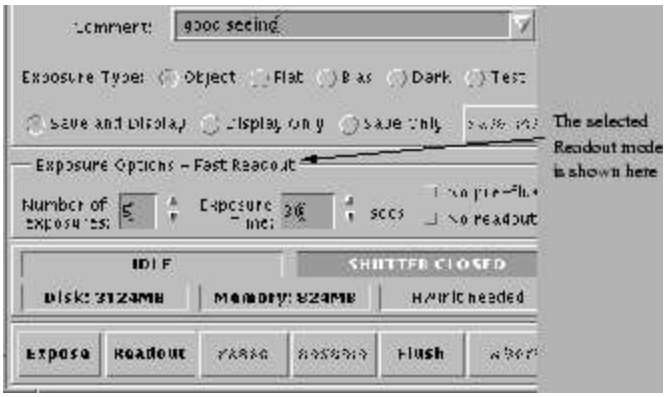


Figure 5.1: SDSU readout mode display

Show Temperatures

Selecting this item will popup a window showing the last read values for all active temperature sensors. Press the *Read Temperatures* button to begin reading temperatures at an interval specified in the *Read Temperatures every* ... field. When idle, CICADA will read temperatures at this interval automatically. While CICADA is busy taking data it may not always be possible for the temperatures to be read. In this case, the software waits until the system is idle before updating the temperatures window and the temperature display on the observing window. Figure 5.2 shows available options.

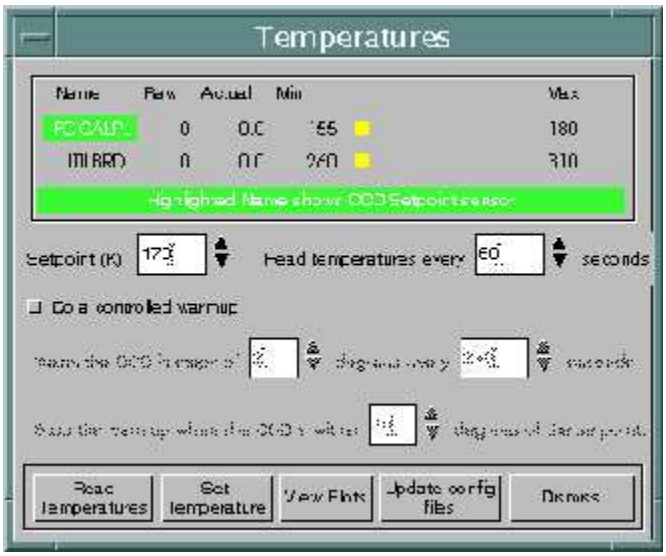


Figure 5.2: Temperature Display and Control Window

CICADA can perform a controlled rate warmup of a CCD - parameters for controlling this operation are set dynamically from this window and are loaded, by default, from the Cicada focalplane configuration. This is an *engineering mode* only operation.

Progressive plots showing temperatures for those sensors configured can also be viewed. These plots will only show values read back since the plot window has been popped up. A log of temperatures for the whole CICADA run is stored in /opt/cicada/logs in a date stamped file. More extensive analysis of temperature performance can be made using these data.

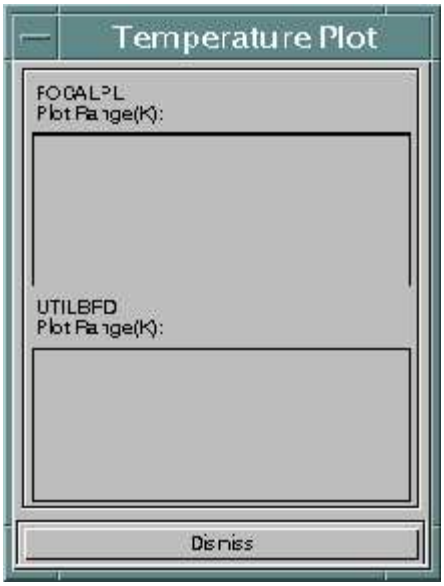


Figure 5.3: Temperature Plots

Note that changing the temperature read interval to a small value will impact on the performance of CICADA.

Shutdown Controller

This option should be used before disconnecting the power to the controller. A shutdown sends a POF command to the SDSU controller making it safe for the SDSU controllers to be powered down.

Get Hardware Status

This option simply sends a `GET_STATE` command to the SDSU camera slave process. Normally these are run automatically when the system is `IDLE` - at a regular interval (determined by the temperature readback interval in the *Temperatures* window). This option allows for the retrieval of system status whenever required.

SDSU Configuration

Engineering Mode Operation

Controlling the setup of the SDSU controllers is via the loading of *SDSU setup* parameters. This is normally done from the preconfigured file `/opt/cicada/config/sdsu/controller_name/sdsu.setup`. A user in the `cicada` group will be able to setup this file by using the *Configure Controller...* option under the *SDSU menu*. Choosing this option will popup the window displayed in Figure 5.4.

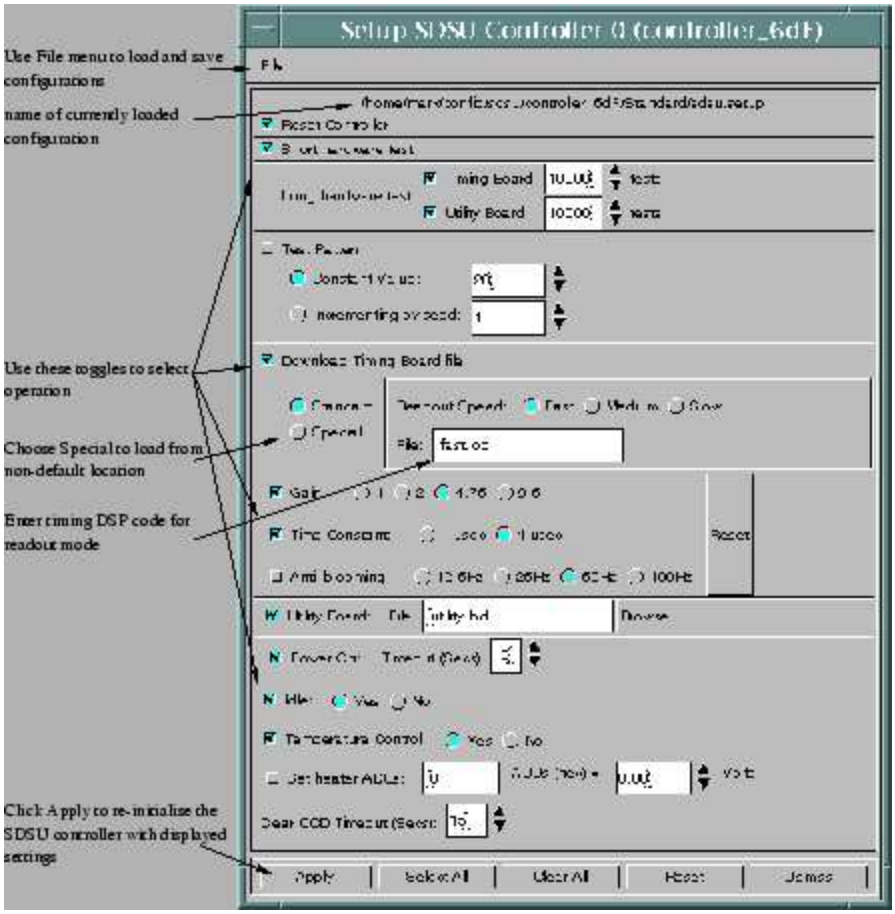


Figure 5.4: SDSU Configuration Engineering Interface

Here, the ability to selectively enable different initialisation options is available. Once a stable configuration is arrived at, it can be saved in the default configuration file for default loading by the observer.

SDSU Controller Commands

CICADA also provided the engineer with the ability to send individual commands to the SDSU controller via the *DSP Commands* window found under the *SDSU* menu. This window allows the engineer to direct commands to each of the `SBUS`, `Timing` or `Utility` circuit boards of the SDSU controller.

To increase the diagnostic output from these commands, the CICADA *debug* level needs to be modified. This debug level operates as a mask and the values in Table5.2.1 are pertinent for SDSU operations.

Table 5.1: Debug levels for SDSU operation

Level	Output Description
0	Show nothing
256	Show only errors from submitted commands
512	Show commands sent to the SDSU controller
1024	Show replies from the SDSU controller
2048	Run in dummy mode - neither sending or receiving data

Figure 5.5 shows the commands available for sending to the SBUS board. These allow the readback of the Command Status Register (*CSR*), the Address register (*ADR*) and the Byte Count Register (*BCR*) of the

SDSU astro device driver. The ability to *reset* the device driver is also available.

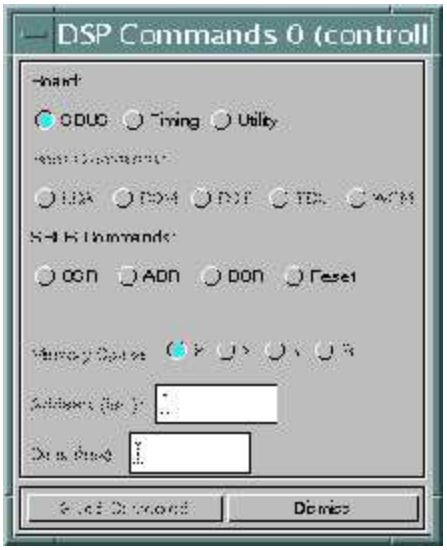


Figure 5.5: SDSU Sbus DSP Commands

Figure 5.6 shows the commands available for sending to the Timing board. Here basic (boot) commands to read a memory location (*RDM*), write a memory location (*WRM*), reset the timing board (*RST*) and test the data link (*TDL*) (fibre-optic link between SBUS interface card and timing card) are available. In addition, basic controller operations can be performed including: clear (flush) the CCD (*CLR*), go into idle mode (*IDL*), set gain to high (*HGN*), and low (*LGN*), start a CCD read (*RDC*) (though CICADA will not be ready to capture read back bytes - they will just spill) and stop any current operation (*STP*).

Memory operations can be directed to each of *P, X, Y or R* memory locations and need an address setting. Some operations also require a data value to use, for example the *TDL* command will echo the supplied data value - if the controller communications are functioning correctly.

Figure 5.6: SDSU Timing DSP Commands

Figure 5.7 shows the commands available for sending to the Utility board. Here basic (boot) commands to read a memory location (*RDM*), write a memory location (*WRM*), reset the timing board (*RST*) and test the data link (*TDL*) are available. In addition, basic utility operations can be performed including: abort an exposure countdown (*AEX*), open the shutter (*OSH*), turn on the power (*PON*), and turn off the power (*POF*), start an exposure countdown (*SEX*), close the shutter (*CSH*), pause an exposure countdown (*PEX*), resume an exposure countdown (*REX*) and perform a full system reset (*SYR*).

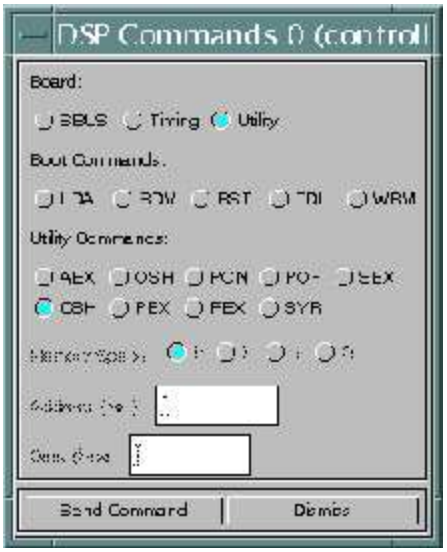


Figure 5.7: SDSU Utility DSP Commands

Refer to the SDSU documentation for a full description of each of the DSP commands available.

The Astromed 3200 controller

Initialisation of the AM3200

The observer may start a hardware initialisation of the AM3200 controller at any time by selecting the *Initialise CCD Hardware* option from the *Actions* menu of the Observing Window. Progress will be shown in the `Cicada Messages Window` and voltages and temperatures will be shown as the hardware initialisation proceeds. When the initialisation is complete a window showing the final settings for all voltages will be displayed. (This window can be redisplayed at any time by selecting *Hardware Status* from the *Options* menu. Be aware that the figures shown are as read at the end of the last hardware initialisation). Occasionally during Astromed 3200 hardware initialisation, the black level is not set correctly (very high values). This bug also exists with the current Imager software - we are tracking it down. Just repeat the hardware initialisation to clear.

NOTE: The CCD temperature display on the Observing Window will not display a value until after a Hardware Initialisation has been run. This is a limitation of the interaction between CICADA and the AM3200 hardware - basically, the software is unaware of the hardware state until after a hardware initialisation has been run.

The AM3200 menu

These are the AM3200 specific menu items appearing under the *AM3200* menu button in the Observing Window:

Continuous Flush & Read Temps

Selecting this button will place the system in a loop where it flushes the CCD and then reads the temperatures and updates the temperature meters display. The temperature of the CCD is *CCDTMP1* and the figure is under the *Actual* heading in Kelvin.

Edit Gain Values

This item will pop-up a window allowing the observer to change the default values for *PGain*, *CCDI* and *TC*.

- *PGain* is a scaling factor which affects the number of electrons per ADU. It's value and what this value means by way of a final e/ADU is CCD dependent and the observer should consult with Electronics staff to find out appropriate values for their particular CCD. A sensible default for each chip is always set and shown in the pop-up window.
- *CCDI* is the integration time per pixel and adjusting this affects CCD readout time and noise levels. Once again, consult with Electronics staff before modifying this.
- *TC* is the integrating Time Constant and is used in conjunction with CCDI to set gain. If CCDI is changed, it is likely that there must be a corresponding change to TC. Electronics staff have the details about this relationship.

A Hardware Initialisation can be started using the *Initialise* button in this pop-up window. Use the *Defaults* button to reset PGain, CCDI and TC to the defaults for the current CCD.

If the observer wishes to alter the gain of the CCD and modifies these values, the new values are stored in the `.cicadarc-hostname` file in the observer's home area. These values will be restored the next time CICADA is run and a warning will be given the next time a hardware initialisation is requested so that the observer has the opportunity to revert to the default values.

Show Temperatures

This is as for the *SDSU* controller - see section 5.2.

Show Voltage Meters

This option will display a window where the observer can choose to monitor Bias voltages, Clock voltages or Temperatures. Note that it is not possible to do this concurrently with taking exposures as the reading out of voltages requires separate software to be loaded into the Astromed controller.

Show Hardware Status

This will display a window showing the voltages set in the Astromed controller at the most recent hardware initialisation. Some of these voltages change with time and so the values shown in this window may be incorrect depending on how much time has elapsed since the last hardware initialisation.

Shutdown Controller

This option should be used before disconnecting the power to the controller. A shutdown sets all controller voltages to zero.

CICADA FITS files

CICADA writes the new multi-extension FITS format. This format is understood in IRAF V2.11 but requires special processing. For a full description see NOAO's mosaic image definition.

For single region readouts from single CCDs, normal single image format is written (i.e. no special data reduction software is required in this case).

FITS files are sized according to the current instrument configuration. Because data and headers are written by separate Unix processes, enough space has to be set aside in the FITS file to accommodate all headers at the beginning of an exposure. This space is predetermined by the local site CICADA system administrator using the camera configuration table. See the CICADA Configuraionanual http://www.mso.anu.edu.au/computing/cicada/cicada_config/cicada_config.html for detailed information on

CICADA hardware configuration.

FITS file header composition is made up of a standard set of keywords and additional site-specific keywords.

CICADA FITS Dictionary

CICADA will write a standard set of FITS keywords from the CICADA FITS dictionary - table 7.1:

Table 7.1: CICADA FITS Keyword Dictionary Version 1.1

Keyword	Type	Note	Example Value	Description
SIMPLE	Bool	P	T	Fits standard
BITPIX	Int	P E	16	Bits per pixel
NAXIS	Int	P E	0	Number of axes
BSCALE	Float	P E	1.000000E0	REAL = TAPE*BSCALE + BZERO
BZERO	Float	P E	3.276800E4	
EXTEND	Bool	P	T	FITS extensions are part of this file
NEXTEND	Int	P	8	Number of extensions
OBSERVER	String	P	'Gary Da Costa'	Observer name
PROPID	String	P	'P12345'	Site specific observing proposal identifier
FILENAME	String	P	'/datagdc/ccd0037.fits'	Original host filename
OBSID	String	P	'T_40.20000904.184013'	Observation ID
RUN	Int	P	37	Run number
OBJECT	String	P	'47 Tuc 120s R'	Observation title
IMAGETYP	String	P	'object'	Observation type
EXPTIME	Float	P	1.200000e+02	Exposure time (sec)
EXPREQ	Float	P	1.200000e+02	Requested exposure time (sec)
DARKTIME	Float	P	1.206810e+02	Dark time (sec)
INSTRUME	String	P	'WFI'	Instrument name
CAMERA	String	P	'WFI1'	Camera name
FILTYP	String	P	''	Type of filter
FILTER	String	P	'R'	Filter name
FILPOS	Int	P	4	Filter position
DEWAR	String	P	'DWFI'	Dewar name
READMODE	String	P	'FAST'	CCD readout mode
WINDOW	String	P	'USER_DEFINED'	Readout window name
DETECTOR	String	P	'WFI1'	Detector Name
DETSIZE	String	P	'[1:8422,1:8282]'	Detector size
NCCDS	Int	P	8	Number of CCDs

NAMPS	Int	P	8	Number of amplifiers
CCDTEMP	Float	P	1.582721e+02	CCD temperature (K)
FITSKW1	String	P	''	Optional user keyword 1
FITSKW2	String	P	''	Optional user keyword 2
FITSKW3	String	P	''	Optional user keyword 3
FITSKW4	String	P	''	Optional user keyword 4
RA	String	P T	'00:24:05.4'	Initial RA 1.051125e-01
DEC	String	P T	'-72:09:08'	Initial Dec -1.259294e+00
EQUINOX	String	P T	'J2000.0'	
LST-OBS	String	P T	' 3:32:-968092763.0'	LST at start (hh:mm:ss.s) -7.0401e+04 rad
LSTEND	String	P	' 3:32:-968092763.0'	LST at end (hh:mm:ss.s) : -7.0401e+04 rad
MJD-OBS	Float	P T	4.058700000000e+04	MJD of observation (days)
OBSERVAT	String	P T	'SSO'	Observatory
TELESCOP	String	P T	'40INCH'	Telescope
OBJNAME	String	P T	''	Telescope object name
LAT-OBS	Float	P T	3.127336e+01	Degrees
LONG-OBS	Float	P T	1.490610e+02	Degrees east of Greenwich
ALT-OBS	Float	P T	1.149000e+03	Metres above mean sea level
AIRMASS	Float	P T	1.483511e+00	Airmass
HA	String	P T	'11:30:23.3'	Hour angle at MJD (hh:mm:ss.s): rr.r (radians)
HAEND	String	P	'11:40:11.6'	Hour angle at end (hh:mm:ss.s): rr.r rad
ZD	Float	P T		Zenith distance at start of exposure
ZDEND	Float	P		Zenith distance at end of exposure
TIMESYS	String	P	'UTC'	Default time system
UTC-OBS	String	P T	'18:40:13'	UTC of observation start
UTCEND	String	P	'18:50:23'	UTC of observation end
DATE-OBS	String	P T	'2000-09-04T18:40:13'	Y2K date of observation start
TELPAN	Float	P T	dd.d	Telescope position angle (degrees)
PIERSIDE	String	P T	'East'	Side of pier
TELFOCUS	Float	P T	0.000000e+00	Focus position
FOCNEXPO	Int	P F	5	Number of focus steps
FOCSHIFT	Int	P F	40	Number of rows read each focus step
FOCPOSn	Float	P F	32.3	Focus position n
FOCSOLN	Float	P F	32.8	Optimal focus position
IMAGESWV	String	P	'CICADA Release 3.1'	Image creation software version
KWDICT	String	P	'CICADA FITS V 1.1'	Keyword dictionary version

XTENSION	String	E	'IMAGE '	Image extension
EXTNAME	String	E	'im1 '	Extension name
EXTVER	Int	E	1	Extension version
INHERIT	Bool	E	T	Inherit global keywords
IMAGEID	Int	E	1	Image identification
CONTROLR	String	E	'WFI_Bottom'	Controller Name
CONSWV	String	E	'V0.1493'	Controller software version
CONHWV	String	E	''	Controller hardware version
READTIME	Int	E	52416	Controller readout time (ms)
CCDNAME	String	E	'WFI0_CCD20'	CCD identification
CCDSIZE	String	E	'[1:2098,1:4136]'	CCD size
CCDNAMPS	Int	E	1	Number of amplifiers used to readout CCD
AMPNAME	String	E	'A'	Amplifier identification
GAIN	Float	E	0.000000e+00	Amplifier Gain (e-/ADU)
RDNOISE	Float	E	0.000000e+00	Read noise for amp (e-)
SATURATE	Float	E	0	Maximum good data value - ADU
LINCOEF	Float	E	0.000000e+00	Linearity coefficient No=Nt(1+Nt**lincoef)
SUMSAT	Int	E	65000	Saturation level of summing wells
AMPSIZE	String	E	'[1:2098,1:4136]'	Amplifier size
CCDSEC	String	E	'[1:2098,1:4136]'	Region of CCD read
CCDSUM	String	E	'1 1'	CCD pixel summing
BIAS0001	String	E	'[1:10,1:4136]'	Bias section - prescan cols
BIAS0002	String	E	'[2059:2098,1:4136]'	Bias section - postscan cols
AMPSEC	String	E	'[1:2098,1:4136]'	Amplifier section
DATASEC	String	E	'[1:2098,1:4136]'	Data section
DETSEC	String	E	'[6325:8422,1:4136]'	Detector section
P. Primary header unit				
E. Extension header unit				
T. Delivered from site-specific telescope control system plugin				
F. Only in CICADA generated focus sequence images				

Site Specific FITS headers

In addition to the CICADA standard keywords (table 7.1) other keywords can be added to cater for site-specific requirements. There are a number of ways CICADA can be customised so that these extra keywords can be included.

1. *Telescope Control System (TCS) interface*. Local plug-in interface to the TCS can incorporate *extra* keywords.

2. *Optional Interface to the Exposure Controller*. Local plug-in to the exposure controller to account for local exposure timing information.
3. *Optional Site Instrument FITS additions*. Site instrument plug-ins can, optionally, have an associated set of FITS keywords.
4. *Optional FITS Table*. A table of keywords copied into the FITS file without interpretation.
5. *Optional FITS Plug-In*. A plug-in called after the FITS file is closed to allow it to be rewritten with updated information.
6. *Keyword name translation*. In addition to adding extra keywords, keywords from the standard set may be renamed by using an IRAF like keyword translation table.

These techniques are described fully in the following sections.

Interface to the Telescope Control System

CICADA expects that an interface to the site telescope control system (TCS) be provided through a site-specific plug-in. By default, CICADA will use a *dummy* TCS that will provide fake telescope information for the FITS headers. This *dummy* TCS will use the system time and date to generate UTC date and time headers.

For each exposure a single call is made to the TCS plug-in for supply of the FITS keywords flagged with a T in the 7.1 table. If some of these keywords are not available then they should be set to be invalid by the plug-in - CICADA will then indicate so by placing a COMMENT keyword in its place in the FITS file. The call to the TCS is made near the start of an exposure but because CICADA is a multi-process system, this is not necessarily at the same time that the shutter opens. To account for this corrections are applied to the returned TCS time information so that they are correctly aligned with when the shutter actually opened.

In addition to the *standard* telescope keywords, the site TCS plug-in can deliver an arbitrary number of *extra* keywords that are of local interest. These are placed in the primary header unit of the FITS file without interpretation.

Optional Interface to the Exposure Controller

Some sites will have exposure controllers (shutter, filter wheel controller), that can provide more accurate local information about each exposure. For example, shutter opening/closing behaviour that cannot be accounted for by the simple shutter delay setting in CICADA's configuration. CICADA supports these instruments by providing a call to the exposure controller plug-in at the end of each exposure. The extra exposure information is added to the FITS headers along with standard exposure information.

Note that CICADA supports accurate timing of exposures that have been paused/resumed. Using exposure controller timing information provided by a plug-in might not be able to cope in these situations - check with the local site to see if this is the case.

Optional Site Instrument FITS additions

For each configured *site instrument* CICADA allows for the addition of an arbitrary number of keywords which have been setup for the site instrument. These are configured as a table of keywords in a file set up (and probably rewritten for each exposure) by the plug-in. This file is re-opened and read at the end of every exposure and the data is copied without interpretation - it is imperative that the formatting comply with to standard FITS keyword formatting rules.

This feature allows for the support of site instrument specific keywords that CICADA needs to know nothing of.

Optional FITS Table

A simple copy of any data in the file /opt/cicada/config/fits_table is made (to the FITS file) if the file exists.

This data is copied without interpretation, so it is important that it be formatted with standard eighty character FITS keyword entries. CICADA provides no way of linking the contents of this file with the current exposure other that through other optional plug-ins. The file is re-opened and read after each exposure finishes and also after the optional exposure-controller plug-in has been called. This gives the site the opportunity, for example, to rewrite the file during the exposure controller plug-in - if necessary.

Optional FITS Plug-In

At the end of CICADA's FITS file construction one last opportunity is given to the local site to modify the final FITS file. CICADA closes the FITS file and then, optionally, calls a FITS plug-in routine to update the file with any extra information that cannot be provided by using any of the other methods described above. For example, the AAO 6dF instrument has FITS BINTABLE extensions that describe the configuration of the instrument for each exposure. This information can only be added through this FITS plug-in mechanism.

The disadvantage of using the FITS plug-in is the extra time required to read and rewrite a new FITS file. This cannot be avoided when other methods do not provide the required functionality.

Keyword Translation Table

The keyword translation table enables CICADA standard keywords from the dictionary in Table 7.1 to be rewritten so that they comply with site requirements. This table lives in the file `/opt/cicada/config/keyword_translation_table`. An example file is as follows and consists of simple keyword name pairs, with the CICADA (IRAF) standard keyword name on the left.

Table 7.2: An example keyword translation table for CICADA.

```
# FITS Keyword translations required by IRAF.
# IRAF expects keywords on left which are mapped by CICADA
# to keywords on right.
amp                     imageid
subset                  filter
imagetyp                runcmd
exptime                 exposed
readmode                speed
gain                    ro_gain
rdnoise                 ro_noise
```

The same table can also be used during IRAF data reduction to translate site keywords back to the standard names expected by IRAF. See the IRAF help system for more information about keyword translation.

CICADA Scripting

It is possible to write scripts in the Tcl language for use with CICADA. These scripts must be run from within the CICADA observing window using the *Run Cicada Script* item from the *Actions* pulldown menu. CICADA starts a Tcl interpreter and passes the script to the interpreter for execution. CICADA commands embedded in the script are read by the Tcl interpreter and passed back to CICADA for processing.

A CICADA command consists of the keyword `cicada` followed by a verb, optionally an instrument part name, and then optional parameter/value pairs. For example: *cicada readout xo1=100 width1=100*. The

following table lists available verbs with associated parameters.

A CICADA command has the following structure:

```
[cicada] command [instrument_part_name] [parameter=value [parameter=value ...]]
```

For example:

```
expose dbs_red duration=100 do_save=1 do_display=1 run_number=100
```

will run an `expose` request on the `dbs_red` camera for 100 seconds saving and displaying the image.

The full CICADA command dictionary is detailed in the following table. The table is formatted into groups of commands belonging to a particular context, each command is then given in bold followed by a set of parameters.

The parameters are described with data type and default value enclosed in square brackets. For example, the parameter ''comment [string , undefined]'' is a string parameter with an undefined initial value. String values can be enclosed in double quotes. Boolean parameters take either 0 for false or 1 for true. List parameters have values separated by commas, for example ''r_x=0,100,200''. All parameter/value pairs are separated by white space.

CICADA also sets values of a set of global TCL variables that can be used inside a script. The table lists these variables, if avaiable, next to each parameter's default value, for example ''object [string , undefined, CICADA_OBJECT]''. They are always upper-case. As an example, one could make use of the variable CICADA_RUN_NUMBER as follows:

```
expose dbs_red duration=100 do_save=1 do_display=1 run_number=$CICADA_RUN_NUMBER
```

These global TCL variables are set at the start of a script and not updated during a script execution.

- General Commands
  - **status**
    Fetch current status of the instrument.
    - state [boolean , false]
      If true, get only status structures without status messages.
  - **abort**
    Abort the current operation. Usually tries to do this in a clean way so sometimes could see small delay. Takes no parameters.
  - **set**
    Set the value of a parameter for any other command. The set value remains current until modified by another set or specific command. The following list of parameters can only be changed with `set`.
    - comment [string , undefined]
      Set an optional COMMENT FITS keyword.
    - object [string , undefined, CICADA_OBJECT]
      Set an optional OBJECT FITS keyword.
    - observer [string , undefined]
      Current observer name.
    - prop_id [string , undefined]
      Current proposal id.
    - read_mode [fast|medium|slow , undefined]
      Readout mode.
    - auto_archive [boolean , undefined]
      True if FITS files to be archived to configured archive disk.
    - imagetyp [string , undefined]
      Type of current observation.
    - fitskw1 [string , undefined]
      Set first user-defined FITS keyword.
    - fitsv1 [string , undefined]

Set value of first user-defined FITS keyword.
- fitskw2 [string , undefined]
  Set second user-defined FITS keyword.
- fitsv2 [string , undefined]
  Set value of first user-defined FITS keyword.
- fitskw3 [string , undefined]
  Set third user-defined FITS keyword.
- fitsv3 [string , undefined]
  Set value of first user-defined FITS keyword.
- fitskw4 [string , undefined]
  Set fourth user-defined FITS keyword.
- fitsv4 [string , undefined]
  Set value of first user-defined FITS keyword.

◦ **script**
Execute a TCL script. Cicada runs a TCL interpretor to execute the script that has embedded Cicada commands.
- cmds [filename , undefined]
  Identifies the name of a TCL script file - use a full pathname of a file on the observer's computer.

• Camera Commands.
These commands are specific to Cicada cameras.

◦ **get_state**
Get state of active components of an instrument - this command forces a reread of instrument meters rather than collecting state from current shared memory, as the `status` command does.
- get_temps [boolean , true]
  Make a reading of the camera's temperature sensors.
- get_volts [boolean , false]
  Make a reading of the camera controller's voltages.
- get_filter [boolean , false]
  Get camera's current filter position.

◦ **pause**
Pause the exposure. The shutter is closed and the system is ready for a change of exposure time or repeat count - see command `resume`.

◦ **resume**
Resume the exposure with optionally updated parameters.
- duration [seconds , undefined]
  Resume with new exposure duration - this is expressed as a total exposure time, so any exposure before the pause will be subtracted from this value.
- repeat [count , undefined]
  Readjust the exposure sequence counter.
- do_readout [boolean , false]
  Do a readout after resume.

◦ **init**
Run a camera initialisation sequence
- config [boolean , true]
  If true then use configuration data from database, else use data from parameters in memory and those set on this command line.
- do_warmup [boolean , true]
  Do a controlled warmup of the camera using specified warmup rates.
- warmup_step [degrees K , 5]
  Warmup at this step for each warmup interval.
- warmup_tol [degrees K , 3]
  Continue warmup until temperature is within this proximity to target temperature.
- warmup_interval [seconds , 600]
  The interval at which to apply the warmup step.
- setpoint [degrees , undefined]
  Temperature setpoint degrees K.

- **flush**
  Flush the CCDs of charge.
  - count [integer , 1]
    The number of times to perform the flush operation.

- **shutter**
  Open the shutter for the specified duration.
  - unit [mseconds , 1000]
    The number of millisecond units.
  - duration [integer , 1]
    The shutter unit counter. For example, if set to 1 with the unit set to 1000, then the shutter will open for 1000*1 milliseconds.
  - dark [boolean , false]
    If true, then the shutter is not opened - ie a dark frame.
  - open [boolean , false]
    Direct open shutter flag. Interpreted when shutter command used alone.

- **readout**
  Readout the CCD array. Use the parameters set to configure the readout.
  - do_save [boolean , true]
    save the exposure
  - do_display [boolean , true]
    display the image
  - clear_display [boolean , true]
    clear display
  - frame_mode [boolean , false]
    separate frame display
  - approx_scaling [boolean , true]
    true for faster integer scaling
  - vis_instructions [boolean , true]
    faster ultra VIS instructions wanted
  - camera_coords [boolean , true]
    camera coordinates, otherwise CCD
  - wait_display [integer , 100]
    display timeout msecs
  - display_levels [integer , 200]
    number display levels
  - ccdmask [integer , 65535]
    which ccds on a mosaic to read out
  - window [string , undefined]
    readout window name
  - nregions [integer , 1]
    number of regions
  - xo [integer list , 0]
    x offset of region
  - yo [integer list , 0]
    y offset of region
  - width [integer list , 0]
    width of regions
  - height [integer list , 0]
    height of regions
  - overscan_cols [integer , 0]
    number overscan columns
  - overscan_rows [integer , 0]
    number overscan rows
  - rcf [integer , 1]
    row compression factor
  - ccf [integer , 1]
    column compression factor

        - adaptive_scaling [boolean , true]
          auto adapt scaling
        - contrast [double , 0.5]
          contrast adjustment (0-1)
        - rescale_tol [double , 0.1]
          sigmas diff before rescaling
        - rescale_rate [double , 0.1]
          rescale rate - fraction of image
        - sample_pix [integer , 1000]
          number of pixels to sample
        - display_min [double , 0.0]
          fixed min scaling
        - display_max [double , 65535.0]
          fixed max scaling
        - display_xmag [integer , 1]
          display row zoom
        - display_ymag [integer , 1]
          display column zoom
        - hotpix_val [double , 65535.0]
          hot pixel value
        - do_hotpix [boolean , false]
          mark hot pixels in red
        - scaling_type[mode|median,mod]
          adaptive scaling type
        - run_number [integer , undefined, CICADA_RUN_NUMBER]
          run number for FITS header
        - bitpix [integer , 16]
          number of bits per pixel
        - bscale [double , 1.0]
          FITS scaling factor
        - bzero [double , 0.]
          FITS Offset factor
        - out_dir [directory , undefined, CICADA_OUT_DIR]
          output directory
        - out_prefix [string , undefined, CICADA_OUT_PREFIX]
          output filename prefix
        - out_suffix [string , undefined, CICADA_OUT_SUFFIX]
          output filename suffix
    - **expose**
      Run an exposure sequence. This command would normally run a `flush`, `shutter` and `readout` in sequence to complete, so all parameters for these commands can be set here as well. The command also has a few specific parameters.
        - do_readout [boolean ,true]
          optionally do the readout
        - do_flush [boolean , true]
          optionally do the flush
        - repeat [count , 1]
          exposure repeat count
    - **focus**
      Run an focus sequence. This command would normally run a `flush`, `shutter` and `readout` in sequence to complete, so all parameters for these commands can be set here as well. The command also has a few specific parameters.
        - focus_steps [count , 5]
          number of focus steps
        - do_flush [boolean , true]
          optionally do the flush
    - **drawrect**
      This command draws a set or rectangles of a specified colour on the image display.
        - n_rects [integer , 0]

number rectangles to draw.
- r_colour [colour number , BLUE]
  rectangle colour
- r_x [integer list , 0]
  x position of rectangles
- r_y [integer list , 0]
  y position of rectangles
- r_width [integer list , 0]
  width of rectangles
- r_height [integer list , 0]
  height of rectangles

- **set_filter**
  Set a new filter position.
  - filter_pos [integer , 0]
    The numerical position of the required filter.

- Sdsu Commands. These commands are for the SDSU I and II CCD controllers.
  - **init**
    Extra initialisation parameters can be set for the SDSU in addition to the standard camera parameters.
    - do_long_test [boolean , false]
    - do_short_test [boolean , false]
    - do_reset [boolean , false]
    - do_power_on [boolean , false]
    - pon_timeout [integer , false]
      Timeout for performing a power-on - secs.
    - clr_timeout [integer , false]
      Timeout for performing a flush - secs.
    - do_tim_board [boolean , false]
    - do_tim_download [boolean , false]
    - do_util_board [boolean , false]
    - do_util_download [boolean , false]
    - do_target_temp [boolean , false]
    - do_heater_adu [boolean , false]
    - do_gain [boolean , false]
    - do_tc [boolean , false]
    - do_ab [boolean , false]
    - do_idle [boolean , false]
    - do_test_pattern [boolean , false]
    - test_pattern_constant [boolean , false]
    - test_pattern_constant_val [integer , 0]

        - test_pattern_seed_val [integer , 0]
        - gain_setting [integer , 0]
        - tc_setting [integer , 1]
        - ab_setting [integer , 0]
        - idle_set [boolean , false]
        - auto_rdc [boolean , false]
        - tim_set [boolean , false]
        - util_set [boolean , false]
        - tim_filename [string , undefined]
        - util_filename [string , undefined]
    - **dsp_cmd**
      This command sends a specific DSP command directly to the hardware. To be used for low level testing of specific actions.
        - board [integer , undefined]
        - cmd [sdsu_cmd , undefined]
        - memory_space [P|X|Y , undefined]
        - address [integer , undefined]
        - data [integer , undefined]
    - **set_clock** clk,hl,bd [string , undefined]
      Change a clock parameter, where `clk` is name of clock, `hl` is either hi or lo and `bd` is board number.
    - **set_bias** bias,bd [string , undefined]
      Change bias parameter.
    - **set_board_param** par,bd [string , undefined]
      Change board parameter.
    - **set_param par** [string , undefined]
      Change parameter.
    - **set_board_define** def,bd [string , undefined]
      Change board define.
    - **set_define** def [string , undefined]
      Change define.
    - **build_dsp**
      Rebuild DSP code with new parameters

- Astromed 3200 Commands
  These commands are specific to the Astromed 3200 controller.
    - **init** Specific Astromed 3200 initialisation parameters in addition to standard set.
        - tc [integer , undefined]
          time constant
        - pgain [integer , undefined]
          programmable gain
        - ccdi [integer , undefined]
          integration time
        - ccds [integer , undefined]
          serial transfers
        - ccdp [integer , undefined]
          parallel transfers
    - **volts**
      This command runs the Astromed volts sequence, where voltages are read back and reported in a continuous loop.

        - mode [integer , undefined]
          Mode of volts monitoring, either 1=BIAS, 2=CLOCK or 3=TEMPS.
        - iterations [integer , undefined]
          Number of iterations to run the cycle.
    - **set_volts**
      Set a particular voltage value.
        - vod [volt , undefined]
          controller voltages.
        - vog [volt , undefined]
        - vrbg [volt , undefined]
        - vrd [volt , undefined]
        - vabg [volt , undefined]
        - vabd [volt , undefined]
        - vssl [volt , undefined]
        - vssh [volt , undefined]
        - ids [volt , undefined]
        - vrol [volt , undefined]
        - vroh [volt , undefined]
        - vrspl [volt , undefined]
        - vrsph [volt , undefined]
        - viml [volt , undefined]
        - vimh [volt , undefined]
        - vstoh [volt , undefined]
        - vtherm [volt , undefined]
        - userp [volt , undefined]

- TipTilt Commands
  Commands specific to the Tiptilt camera are:
    - **change_state**

- Dummy Camera Commands. These commands are for the simulation CCD controllers. This allows for testing of the CICADA data flows when no hardware is attached. A standard FITS image is used for input data. Alternatively by specifying `GENERATE_DATA` as the input image, test data is generated.
    - **readout**
      Extra readout parameters need to be set for the dummy controller.
        - in_image [string , undefined]
          name of input FITS file.
        - pixel_time [double , 3.4]
          time delay to clock a pixel (usecs).

- Telescope Plugin Commands
  Cicada supports a simple interface to the local telescope control system, through the use of a plugin.
    - **telescope_get_data** Get all standard CICADA FITS header keywords from the TCS.
        - timeout [integer , undefined]
          Command timeout
    - **telescope_get_param** Get the value of a particular parameter from the TCS.
        - timeout [integer , undefined]
          Command timeout.
        - param [string , undefined]
          Name of parameter to get.
    - **telescope_set_param** Set the value of a particular TCS parameter.
        - timeout [integer , undefined]
          Command timeout.
        - param [string , undefined]
          Name of parameter to set.
        - value [string , undefined]
          Value to set parameter.
    - **telescope_command** Run an arbitrary telescope command.
        - timeout [integer , undefined]

Command timeout.
- command [string , undefined]
  Name (and parameter list) of command to run.

- Filter (Exposure Controller) Plugin Commands
  Cicada supports a simple interface to the local filter wheel controller (sometimes named exposure controller), through the use of a plugin. The following filter controller commands are supported.
  - **get_filter** Gets the current filter position.
    - timeout [integer , undefined]
      Command timeout.
  - **get_filter_exposure_info** Get exposure information from the filter controller if available. Returns exposure start time, exposure duration, filter name and filter position.
    - timeout [integer , undefined]
      Command timeout.
    - param [string , undefined]
      Name of parameter to get.
  - **set_filter** Sets the filter position.
    - timeout [integer , undefined]
      Command timeout.
    - filter_pos [integer , undefined]
      Filter position to set.

- Site Instrument Control Plugin Commands
  Cicada does not interpret commands for site instruments, so support for a simple interface is provided with an arbitray string command. The site instrument plugin can interpret the string as required.:
  - **site_command** Only command supported for site instruments - takes two parameters:
    - command [string , undefined]
      Site instrument command string.
    - timeout [integer , undefined]
      Command timeout.

Example scripts

Following are some simple script examples showing, for instance, the use of global CICADA Tcl variables, Tcl looping, Tcl local variables, Tcl error handling, quoted strings etc.

1. Exposures with different filters.

```
# Sample Cicada TCL script for reading out a CCD with different filter
# settings. In this instance, the filter wheel is controlled through the
# camera interface.
#
set filter 1
# Set exposure parameters
cicada set width=416 height=578 XO1=0 YO1=0
cicada set DO_SAVE=1 DO_DISPLAY=1 DARK=0 DO_READOUT=1
cicada set UNIT=1000 DURATION=5 REPEAT=1
cicada set out_dir=${CICADA_OUT_DIR} out_prefix=${CICADA_OUT_PREFIX}
cicada SET object=\"${CICADA_OBJECT}\"
cicada set comment=\"This is a comment\"

# Loop to read 3 images, automatically incrementing run_number
for {set i 0} {${i} < 3} {incr i 1} {
# remove the temporary image if already there, catch any failures
    catch {exec /bin/rm -f /tmp/IM${i}.fits}
# execute an expose request - abort script if an error occurs
    set filter [expr ${filter} + ${i}]
    cicada set_filter filter_pos=${filter}
    set run [expr ${CICADA_RUN_NUMBER} + ${i}]
    cicada EXPOSE RUN_NUMBER=${run} comment=\"Exposure ${i} of script\"
}
```

2. Following script shows how to command a telescope instrument control module and take some data.

```
# Sample Cicada TCL script for slewing the telescope and taking some
# exposures
#

cicada set width1=100 height1=100 XO1=0 YO1=0
cicada set DO_SAVE=1 DO_DISPLAY=0 DARK=0 DO_READOUT=1
cicada set UNIT=1000 DURATION=5 REPEAT=1
cicada set out_dir=${CICADA_OUT_DIR} out_prefix=${CICADA_OUT_PREFIX}
cicada SET object=\"${CICADA_OBJECT}\"
for {set i 0} {$i < 1} {incr i 1} {
# remove the temporary image if already there, catch any failures
    catch {exec /bin/rm -f /tmp/IM${i}.fits}
# execute an expose request - abort script if an error occurs
    set run [expr ${CICADA_RUN_NUMBER} + ${i}]
    cicada telescope_command T_APT command=\"coord 12 13 14.5 -23 12 13 slew\"
    cicada telescope_get_data T_APT
    cicada EXPOSE RUN_NUMBER=${run}
}
cicada telescope_command T_APT command=\"coord 20 13 14.5 -15 12 13 slew\"
cicada telescope_get_data T_APT
```

CICADA Debug Levels

To increase the diagnostic output from CICADA, the *debug* level needs to be modified. This debug level operates as a mask and the values in Table9 are used for assistance during troubleshooting diagnosis. The level can be set as a command line switch when starting CICADA or from the *Cicada Configuration* dialog when running in *Engineering Mode*.

Table 9.1: Debug levels for CICADA

Level	Name	Output Description
0		Show nothing
1	DEBUG_ESSENTIAL	Essential debugging info
2	DEBUG_ENTRYPOINTS	Entry point messages
4	DEBUG_INDICATORS	Tracewrites to indicate location
8	DEBUG_CALCS	Progressive calculations
16	DEBUG_STATUS	All status handling
32	DEBUG_SPECIAL	Special case temporary debugging

Known Bugs

- The free disk space shown in the `Mb free` field does not give the correct amount of free space available to the observer on quota controlled disks (e.g. /home, /book disks). The figure displayed will be the total free space on the disk and not the space remaining in the user's quota.
- When using the SDSU controller for the WFI, during readout an occasional buffer overflow occurs. This results in missing data in the top four CCDs. At present the observed rate of these failures is less than 1/100. The top half of the mosaic image from WFI should be discarded.

Troubleshooting

- `Things don't work`: Most problems with CICADA are resolved by quitting and restarting the software. So, the first thing to try if there are problems is quitting CICADA and starting up again. When CICADA starts up it looks for and kills any old CICADA processes as well as initialising operating system resources that are needed by the software. Note that for the Astromed 3200 controller this does

not mean you need to do another hardware initialisation. Unlike the Astromed Imager software, CICADA does not require you to go through the HWINIT procedure upon restart. If problems still persist, and you suspect the software, contact the Computer Section for help.

- `Things still don't work:` make sure your home area is not full. Run the command `quota -v` and make sure you have not hit your hard disk limit. CICADA writes preferences and log files in your home area and insufficient free space can cause problems.
- `Noise levels rising:` Check the temperature of the CCD and/or try re-initialising the controller.
- `Low memory - performance affected:` CICADA and GIT can be very memory hungry, depending on the size of images being taken. Keep a close watch on the memory warning indicators both programs have. In the case of GIT, a list of images is maintained - only one of which is loaded into memory at any one time. These could be *floating point images*, thus requiring four bytes or memory per pixel. Sometimes the system fails to free space occupied by previously loaded images - the space has become fragmented - it is probably a good idea to stop and restart GIT. In general stopping and restarting both CICADA and GIT is fast and easy, these problems are then managed.

About this document ...

CICADA V3.1
User's Manual

This document was generated using the **LaTeX**2HTML translator Version 99.2beta8 (1.43)

The command line arguments were:
latex2html `-init_file /opt/local/html/latex2html/cicada.init cicada.tex`

The translation was initiated by Hoa Nguyen on 2000-12-13

Next Group Up Previous

webmaster@mso.anu.edu.au

CICADA V3.1
Writing Plug-Ins

Computer Section
Mount Stromlo and Siding Spring Observatories
Research School of Astronomy and Astrophysics
Australian National University

December 13, 2000

User Plug-In Extensions to CICADA

V3.0 of CICADA supports the concept of a user plug-in. This feature extends the capabilities of CICADA so that local site conditions are handled. Specifically, plugins can be written for:

1. An interface to a local telescope control system (TCS)
2. An interface to the local filter (exposure) controller.
3. Interfaces to other site instrument controllers.
4. A FITS plug-in for modifying the contents of each FITS file at the end of each exposure.
5. Site-specific GUI plug-ins to interface to the low level control plug-ins.

Figure 1.1 shows how these plug-ins fit into the CICADA design. The shaded boxes represent the plug-in code written to support local requirements.

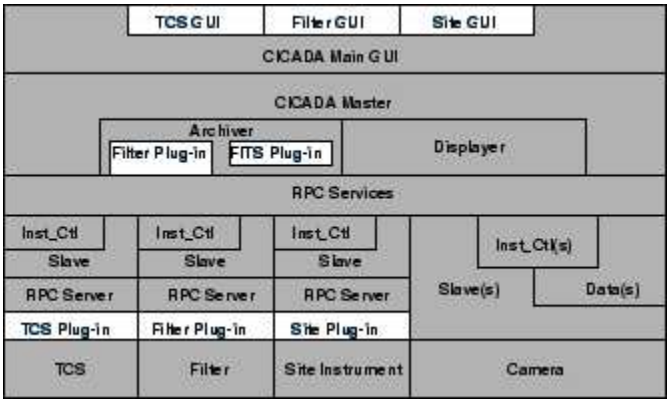


Figure 1.1: CICADA Plug-In Architecture

Implementation

Plug-ins are implemented though the use of shared libraries. The user writes a plug-in that conforms with the CICADA C++ API and then builds the shared library using the supplied makefile. The built shared library replaces the default supplied shared library for the particular facility.

The API is defined with the remote procedure call language (RPCL) and is compiled using the "rpcgen" command to generate a "C" header file, which is then referenced in the user plug-in. RPC is used so that plug-in code can run on a machine remote from the user's workstation. This functionality is transparent to the user.

Template and include files are supplied in the directory structure under `/opt/cicada/user_libs` where shared libraries can be rebuilt after modification/addition of site specific plug-ins. Once built the shared library should be moved to `/opt/cicada`, replacing the shared library that was already installed. Care should be taken to make a copy of the original shared library in case problems occur.

The plug-in facility requires the use of a recent version of the Sun C++ compiler. GNU compilers (or any other C++ compiler) will not work since C++ object files from different compilers cannot be linked.

For the GUI plug-ins the user is required to write a C++ Motif class that inherits basic attributes from a Sun Workshop Visual generated C++ GUI definition. Though not necessary to use Visual to generate the plug-in, it is recommended for simplicity - and in that way the CICADA `simple_control` GUI can be included as a base definition.

Telescope Control System Interface

CICADA's interface to the telescope control system is through the shared library `libtelescope.so`. The interface requires that the following functions be written by the user, where the User_Telescope class is derived from the base Cicada_Telescope class:

- Cicada_Telescope* create_telescope_object(Cicada_Telescope_Desc tel, int do_connect, FILE* report, int debug)

  Takes as input a `Cicada_Telescope_desc` which describes the characteristic of a particular telescope including its name. Based on this name a new User_Telescope instance is created.

- int User_Telescope::get_data(int timeout)

  Retrieves all available telescope data as described in the `Cicada_Telescope_Data` structure which is defined using the RPCL definition. This might consist of a series of individual `get_param` calls to make up the full data structure.

- int User_Telescope::get_param(char* parameter, char* value, int timeout)

  Gets the `value` of an arbitrary (locally defined) telescope `parameter`. Sets `Cicada_Tele_Config` with retrieved value.

- user_telescope::set_param(char* parameter, char* value, int timeout)

  Sets the `value` of an arbitrary telescope `parameter`.

- int User_Telescope::command(char* command, char* value, int timeout)

  Runs an arbitrary `command` which might have an associated parameter `value`. Sets `Cicada_Tele_Config` with retrieved result of command.

Supplied with CICADA is an example implementation of a plug-in: `mssso_telescope.cc`. This example provides support for the current MSSSO telescopes and is, perhaps, a little more complex than needed for most cases. However, the user should easily be able to follow this example to construct a new plug-in and then modify the supplied `Makefile` to build it.

The definition `Cicada_Telescope_Desc` for a user telescope is maintained through the standard CICADA configuration GUI. This GUI allows the user to create a new named telescope with specified interface information such as controlling host, available data etc. Currently, both serial line and unix socket support are available for communicating with a telescope computer.

Filter Control System Interface

CICADA's interface to a filter controller is through the shared library `libfilter.so`. As for the telescope interface, a user filter class, derived from `Cicada_Filter` needs to be written. The following functions need to be supplied:

- Cicada_Filter* create_filter_object(Cicada_Filter_Desc filter_desc, int do_connect, FILE* report, int debug)

  Takes as input a `Cicada_Filter_Desc` which describes the characteristics of a particular filter including its name. Based on this name a new filter instance is created.

- int User_Filter::get(int timeout)

  Returns as much of the information a filter can deliver for the CICADA filter data structure. This is set in the filter objects filter_status member variable.

- int User_Filter::set(char* value, int timeout)

  Sets the filter to `value` requested.

- int User_Filter::get_exposure_info(int timeout)

  Returns as much of the information a filter can deliver for the `Cicada_Exposure_Ctl_Info` data structure. This is set in the filter objects filter_status member variable.

Again, the example `mssso_filter.cc` is provided to show how a user filter plug-in can be built.

Site Instrument Control System Interface

CICADA's interface to an arbitrary site instrument controller is through the shared library `libsiteinst.so`. As for the telescope and filter interfaces, a site instrument class, derived from `Cicada_Site_Instrument`, needs to be written. The following functions need to be supplied:

- Cicada_Site_Instrument* create_siteinst_object(Cicada_Site_Instrument_Desc siteinst_desc, int do_connect, FILE* report, int debug)

  Takes as input a `Cicada_Site_Instrument_Desc` which describes a particular site instrument including its name. Based on this name a new site instrument instance is created.

- int User_Siteinst::command(char* command, int timeout)

  Runs an arbitrary `command` which might have embedded parameter value information. CICADA does not interpret command in any way.

Again, the example `mssso_siteinst.cc` is provided to show how a user site instrument plug-in can be built.

FITS Plug-in

The shared library `libfits_plugin.so` contains the users FITS plug-in code. This is called to update a FITS file at the end of an exposure. The plug-in must supply a function to create an instance of a `Cicada_Fits_Plugin`, which must be the base class of a user defined object.

- Cicada_Fits_Plugin* create_fits_plugin_object(Nametype fits_plugin_name, FILE* report, int debug_on)

  Simply takes the name of the the FITS plugin object to create.

- int User_Fits_Plugin::update(char* fits_name)

  Takes the name of a fits file to update.

The example `mssso_fits_plugin.cc` is provided to show how a user fits plug-in can be built.

GUI Plug-ins

GUI plug-in derived class must include as part of its ‘‘public’’ definition the declarations provided in `simple_control_public.h`. `simple_control_public.h` has the following public methods which must be implemented by the plug-in:

- void Simple_Control_c::initialise_control()

  Called immediately after the creation of the telescope GUI by CICADA, allows the plug-in to perform arbitrary initialisation steps.

- void Simple_Control_c::start_status_check()

  CICADA calls this routine when it wants the GUI to start a telescope interface status check. This is usually in response to a CICADA initiated query to the telescope interface which would then require the plug-in to perform a display update of status.

- void Simple_Control_c::handle_status(void *this_status, int idle)

  This function must be provided so that CICADA can pass a pointer to a Cicada_Telescope_Status structure. CICADA calls this routine after the the telescope interface returns status in response to the plug-in calling the send_command callback. The GUI can then update its window based on the status information.

Telescope GUI Plug-in

CICADA’s interface to the telescope GUI is through the shared library `libsimplegui.so`. The GUI requires that public methods be written by the user, where the User_Telescope_GUI class is derived from the base Simple_Control_c class. The plug-in must provide a GUI creation function `create_telescope_control` that returns a pointer to the Simple_Control_c object. This definition is provided in the header `cicada_simplegui.h`.

- Simple_Control_c* create_telescope_control(Widget parent, Cicada_Telescope_Desc *tel)

  This routine must be provided by the plug-in. CICADA calls it in response to a trigger to popup the telescope GUI. CICADA passes as input a `Cicada_Telescope_Desc` which describes the characteristics of a particular telescope including its name. Also provided as input is the name of a parent X11 widget for the telescope GUI. Two callbacks are provided, one for handling telescope commands which the

plug-in uses and the other for requesting command status. These callbacks must be used by the plug-in for accessing the telescope interface from within CICADA. Based on this information a new User_Telescope_GUI instance is created and returned by the plug-in.

Filter GUI Plug-in

CICADA's interface to the filter GUI is also through the shared library `libsimplegui.so`. Again, the User_Filter_GUI class is derived from the base Simple_Control_c class and the plug-in must provide a GUI creation function `create_filter_control` that returns a pointer to the Simple_Control_c object. This definition is provided in the header `cicada_simplegui.h`.

- Simple_Control_c* create_filter_control(Widget parent, Cicada_Filter_Desc *filter)

  This routine must be provided by the plug-in. CICADA calls it in response to a trigger to popup the filter GUI. CICADA passes as input a `Cicada_Filter_Desc` which describes the characteristics of a particular filter including its name. Also provided as input is the name of a parent X11 widget for the filter GUI. Two callbacks are provided, one for handling filter commands which the plug-in uses and the other for requesting command status. These callbacks must be used by the plug-in for accessing the filter interface from within CICADA. Based on this information a new User_Filter_GUI instance is created and returned by the plug-in.

Site Instrument GUI Plug-in

As for the telescope and filter GUI plug-ins a number of site instrument interfaces can be supported through the GUI plug-in mechanism. Any interfaces built must also be included in the shared library `libsimplegui.so`. And a GUI creation `create_siteinst_control` must be provided.

- Simple_Control_c* create_siteinst_control(Widget parent, Cicada_Site_Instrument_Desc *siteinst)

  Its definition is similar to those for telescope and filter plug-ins.

Error Handling

All plug-in functions are designed to handle a returned integer which indicates error status. If an error occurs, the plug-in should set this return status to a non-zero value and fill the telescope or filter integer `valid` and string `error` variables with appropriate values.

About this document ...

CICADA V3.1
Writing Plug-Ins

This document was generated using the **LaTeX2HTML** translator Version 99.2beta8 (1.43)

The command line arguments were:
latex2html `-init_file /opt/local/html/latex2html/cicada.init cicada_plugins.tex`

The translation was initiated by Hoa Nguyen on 2000-12-13

Next Group Up Previous

webmaster@mso.anu.edu.au

CICADA V3.1 Configuration Manual

Computer Section
Mount Stromlo and Siding Spring Observatories
Research School of Astronomy and Astrophysics
Australian National University

December 13, 2000

Contents

Introduction

This document describes the use of the CICADA configuration tables.

Design of the Tables

Early releases of CICADA had many built-in site-specific sections of code which meant that it was very difficult to ‘‘export’’ to other observatories. One of the aims of CICADA v2 was to replace these sections of the code with ‘‘self-configuring’’ code based around simple ASCII tables.

Another aim of releases 2 and later of CICADA was to provide support for multi-controller, multi-CCD instruments. This requirement also impacted on the design of the Configuration Tables.

The Table Hierarchy

Introduction

An instrument is specified using relational type text tables arranged in a hierarchy. The following diagram 2.1 depicts an instrument consisting of a camera with a dual CCD focalplane and a single ‘‘simple instrument’’ controller.

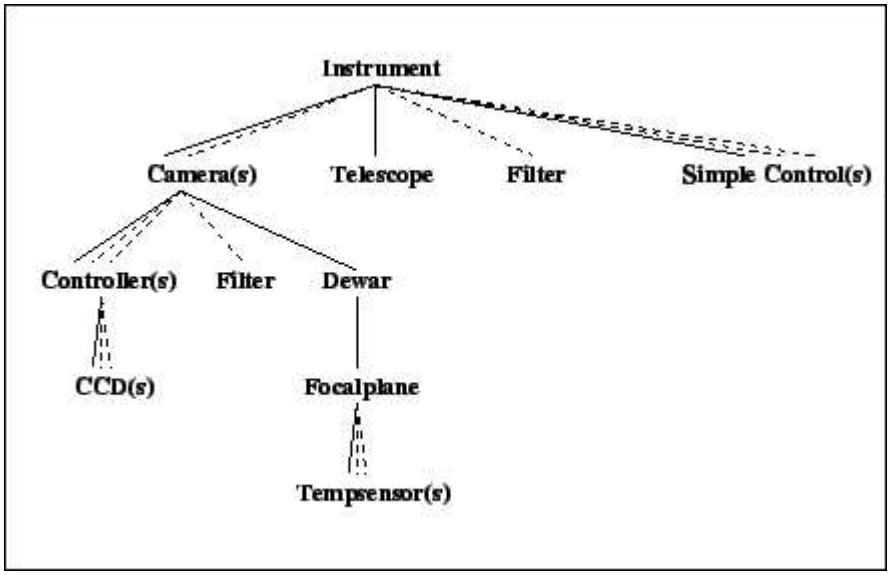


Figure 2.1: Example instrument component hierarchy

The dashed lines in the diagram indicate that multiple components can be configured. Each node in the diagram is described by an entry in one of the component tables and is referenced by a component name. Each entry occupies a single named item in the text table with fields indicated by namevalue pairs. A typical entry from the camera table looks like this:

```
[DUMMY_WFI]
  comment=Dummy camera with 2 controllers;  config1=;  config2=;
  cont_master=1,0;  cont_shutter=1,0; cont_temp=1,0;  cont_xgap=0,0;
  cont_xo=0,0;  cont_ygap=0,0;  cont_yo=0,4096; controller1=DUMMY1;
  controller2=DUMMY2;  dewar=DUMMY;  host=mistress;  ihu=2;
  n_controllers=2;  phu1=3;  phu2=3;  shutter_closed_when_idle=1;
  shutter_delay=0; synchronise_readout=1;
```

This entry describes the camera called `DUMMY_WFI` which is hosted on `mistress`. It has two CCD controllers named `DUMMY1` and `DUMMY2` and is mounted onto dewar named `DUMMY` and does not have a filter wheel. Controller positioning information is given by vectors `cont_xo` and `cont_yo`. Flags are set indicating which controller controls the shutter and temperature sensors and which is the master. The size of generated FITS header units is given by `phu` and `ihu`. Parameters for `shutter_delay` and `synchronisation` during readout are also available.

Once an instrument is described by the configuration tables, it can be selected from the user interface and the necessary processes are started on the configured computers. Configuration tables are maintained either by a text editor or through the CICADA table GUI interface (accessed via the *Hardware Configuration* option of the *Options menu*).

Viewing the tables

When viewing/editing the configuration tables using the CICADA GUI, the window in figure 2.2 is displayed. Simply edit the fields as indicated and click the *Apply* button to rewrite the table.

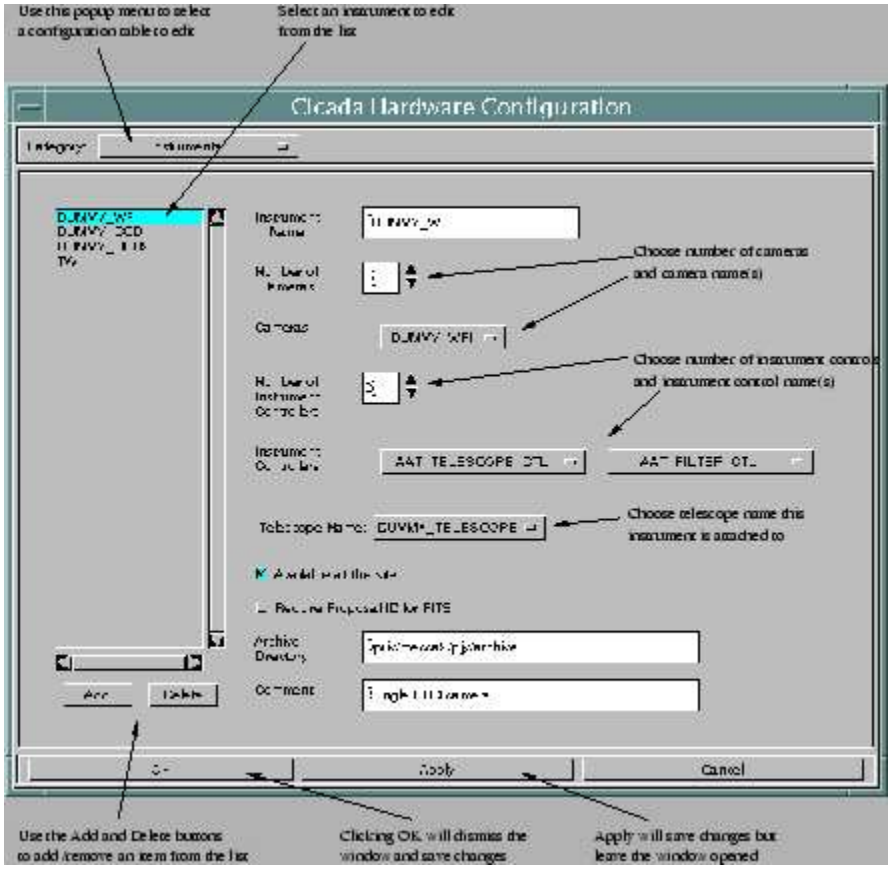


Figure 2.2: CICADA Hardware Configuration GUI

As indicated in figure 2.2 use the *Category* popup menu to select a table for editing. The GUI will then display the available fields for that table - be careful to select the desired item for editing from the scroll-able list. Use the *Add* and *Delete* buttons to change the number of items in the list for each table. More information on modifying the CICADA configuration tables is contained in chapter 3.

The Tables

The Instrument Table - the top level

CICADA is used to control data acquisition `instruments` and associated simple components controllers. These instruments are made up of one or more `cameras`, zero or more `instrument controllers` and (optionally) a connection to a telescope system.

To make the instrument available from CICADA's *Start Observing* menu click on the *Available at this site* toggle. To enable CICADA to write an observing proposal id into the FITS file headers, select *Require Proposal ID for FITS*. This is a site-dependent option, it requires that a relational database be available with observing proposal information. Currently CICADA uses the free MySQL package for implementing a relational database.

Finally, site archiving of data is supported by CICADA. If an archiving directory is entered in the *Archive Directory* field a second copy of the FITS file is made and placed in the specified directory with an observation identification style filename.

Cameras

Cameras are made up of several components - the dewar, focalplane, temperature sensors, CCD controllers, possibly filter wheels and CCDs. As explained above, these are configured using the instrument tables in a relational fashion. The format of each of these tables is described below. Figure 2.3 shows the GUI for editing the camera table.

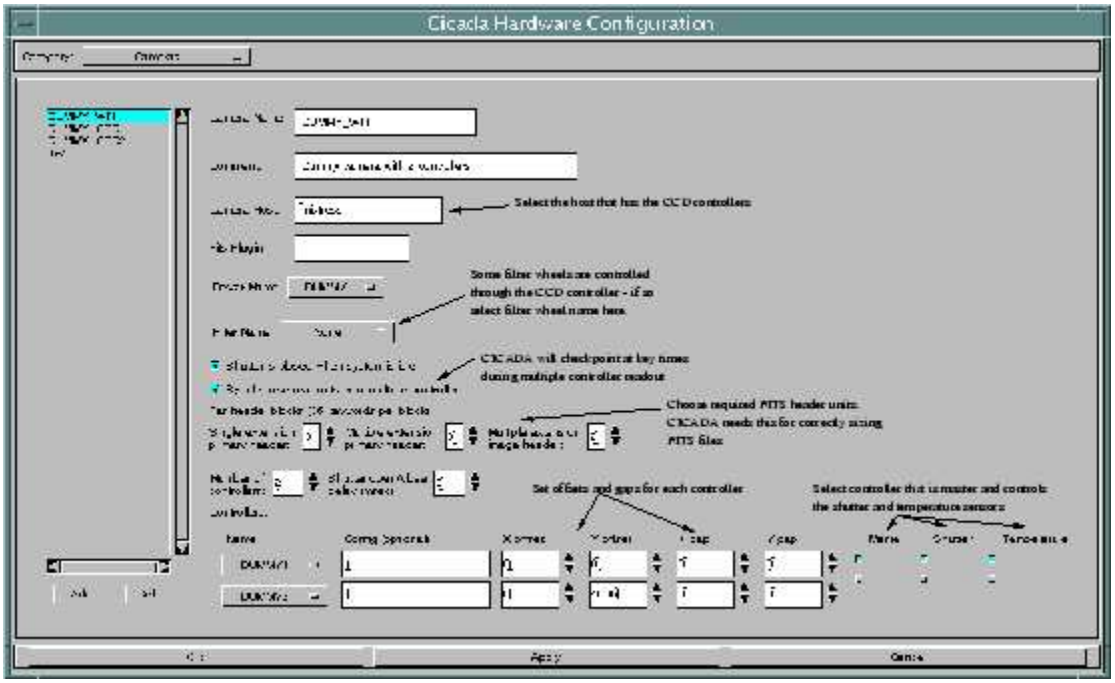


Figure 2.3: CICADA Camera Table

CCD Controllers

The entries in this table tell the CICADA software the type of controller in use, the names of the CCDs connected to that controller and their geometry specified in ''controller coordinates'' (i.e. if using a mosaic of 8k x 8k pixels, this table holds the information that allows the software to determine where an individual CCD sits inside the 8k x 8k coordinate space). Figure 2.4 shows the GUI for editing the controller table.

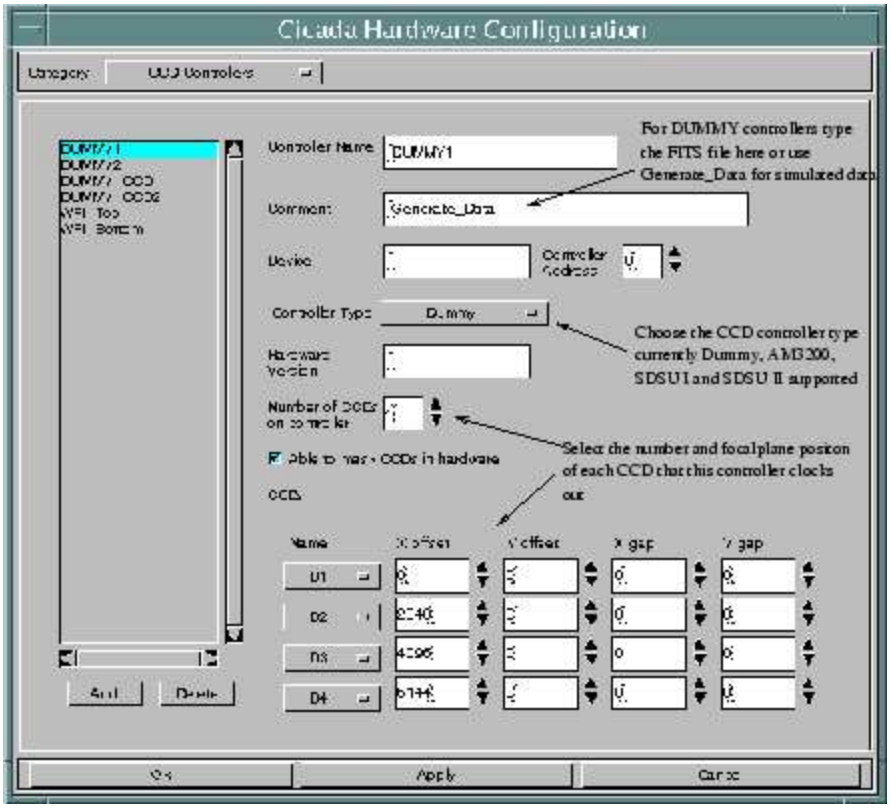


Figure 2.4: CICADA Controller Table

CCDs

Each entry in this table describes the size and amplifier geometry of a CCD. The values entered in this table are used in conjunction with the controller geometry to assemble the FITS file and image display generated by CICADA. Figure 2.5 shows the GUI for editing the CCD table.

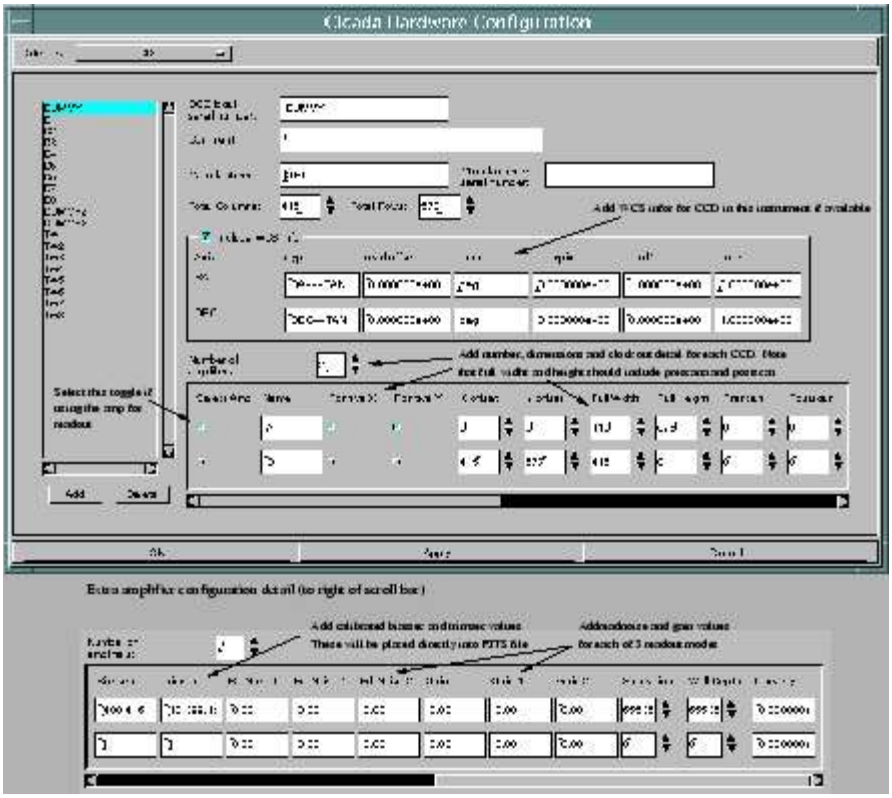


Figure 2.5: CICADA CCD Table

Dewars

At present, for the purposes of CICADA, dewars contain nothing but a `focalplane`.

focalplanes

Each entry in the focalplane table describes the parameters to do with CCD temperature control. There are settings here for controlling the rate at which a CCD will be warmed up as well as temperature setpoint and parameters for heater control and temperature sensor min/max values. Figure 2.6 shows the GUI for editing the focalplane table.

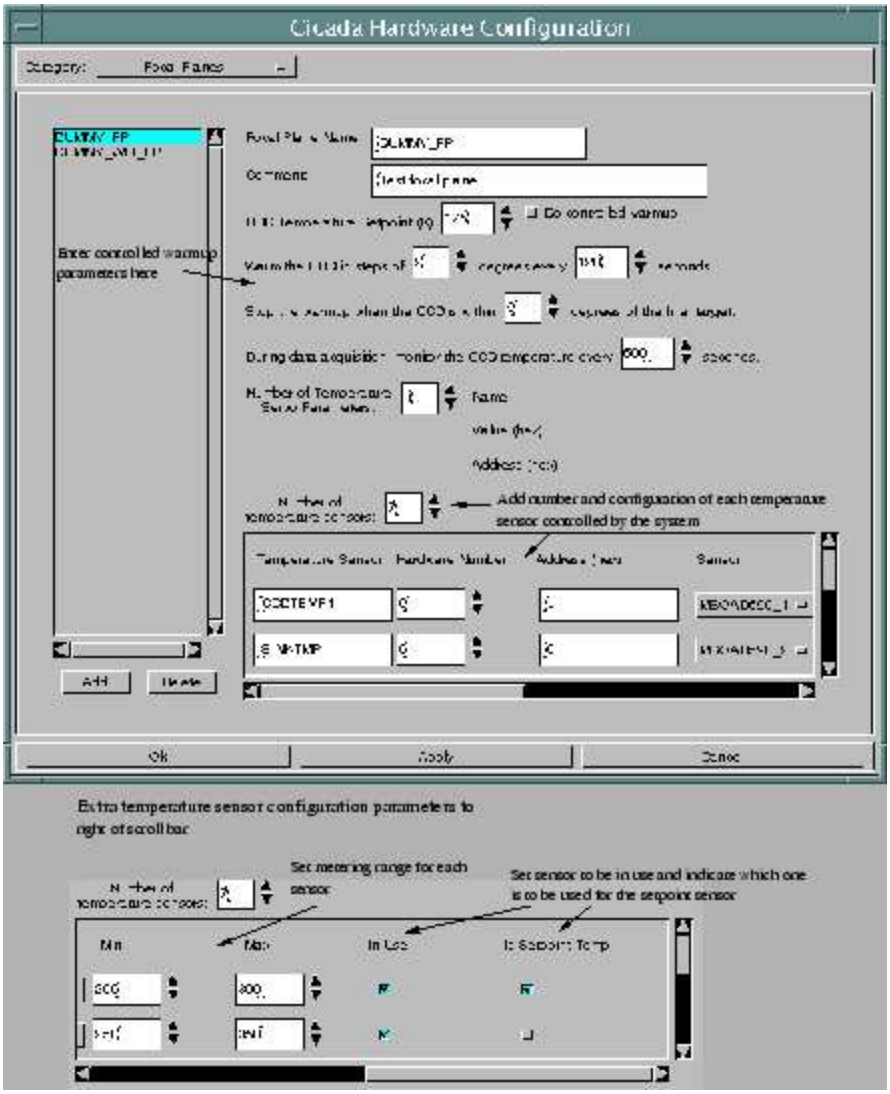


Figure 2.6: CICADA focalplane Table

The function of some of these fields are controller dependent. At the time of writing, CICADA supports the Astromed 3200 controller (AM3200) and the Leach SDSU Generation 1 and 2 controllers. The heater parameters are not used when the controller is an AM3200. Additionally, because many features which are configurable with the Leach controllers are ‘‘hard-wired’’ in the AM3200, the specification of temperature sensors for the AM3200 **MUST** be as follows:

- CCDTEMP1, hardware number 0
- CCDTEMP2, hardware number 1
- LN2, hardware number 2
- PCBTEMP, hardware number 3

In addition, CCDTEMP1 must have the `Is Setpoint Temp` turned on.

Filter Wheels

This table provides the mapping between filter wheel position and filter name (the name added to FITS file headers). Note that filter wheels may be controlled directly from a Unix host or from a CCD controller. If via Unix then either through a serial line or across the network via a socket. Enter the relevant host and port detail required. Figure 2.7 shows the GUI for editing the filter table.

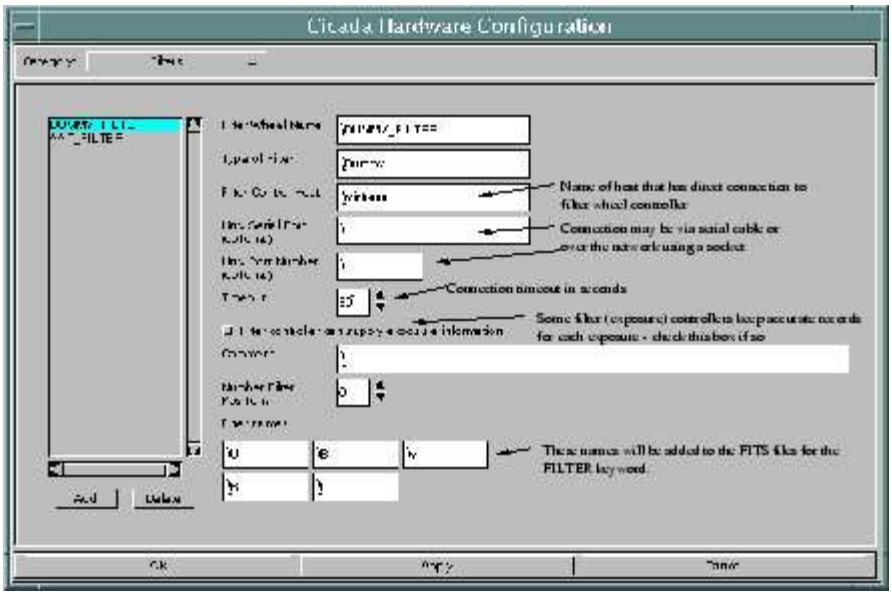


Figure 2.7: CICADA Filter Table

Telescopes

This table holds configuration items relating to CICADA's communication with Telescope Control Systems (TCS). As for filter wheels access to a TCS is via a Unix host serial line or socket interface. Figure 2.9 shows the GUI for editing the telescope table.

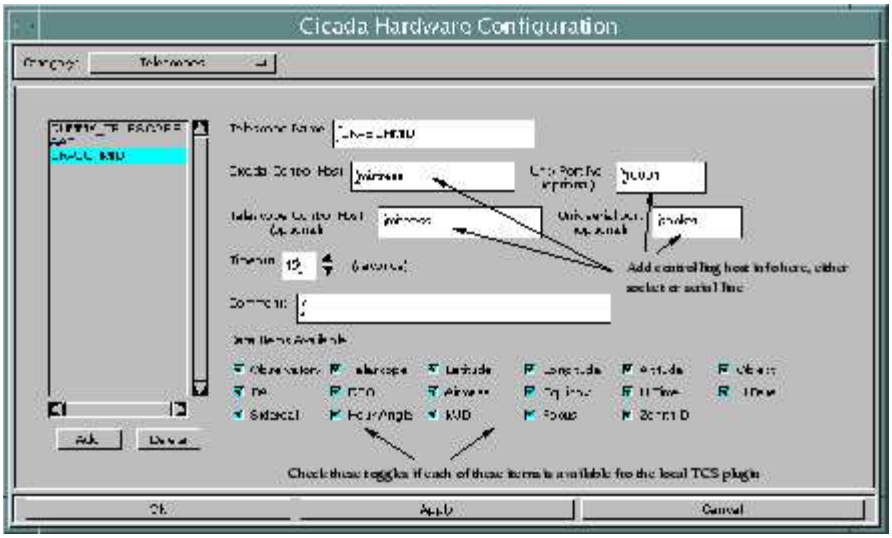


Figure 2.8: CICADA Telescope Table

The CICADA TCS interface is through a site-specific plug-in. Standard data items are delivered by the plug-in for insertion in each FITS file - items that are available for each site telescope are indicated by checking the toggle buttons on this window. Take particular note of the *Focus* toggle - if this is checked, CICADA will try to use the FOCUS command to set the telescope focus for automated focus sequences.

Instrument Controllers

CICADA supports control of instrument hardware in addition to CCD controller hardware. The *Instrument Controller* table is used to store instrument controller configuration items. Instrument control consists of passing instrument specific commands and setting/getting parameters.

Specific support is available for interfacing to the local telescope control system and the instruments filter

(exposure) controller.

More generic support is available for site-specific instruments. Here CICADA will allow the specification of simple instrument controls and provide no interpretation of any I/O to and from the controller.

Figure 2.9 shows the GUI for editing the telescope table.

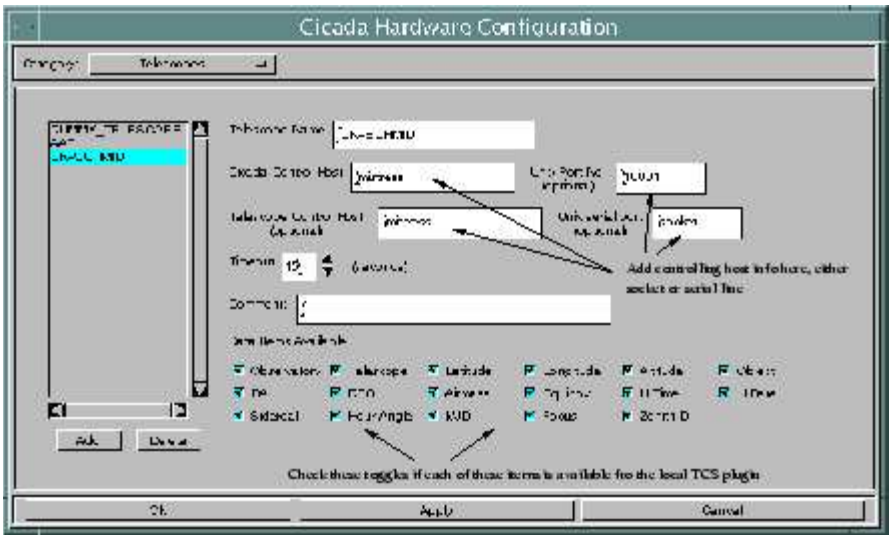


Figure 2.9: CICADA Telescope Table

Temperature Sensor Types

This table allows for the specification of temperature sensor types to be controlled by and connected to a Leach SDSU controller. The entries in this table contain coefficients for converting ADUs output by the sensor to temperature. As temperature sensors are usually acquired in batches and the individual sensors of each type will have nearly identical characteristics, each entry in this table can be re-used many times as the temperature sensor type in the focalplane table. If individual sensors require minor changes to a standard type, they should be specified in the `Temperature Sensor Table`.

Temperature Sensors

The entries in this table inherit characteristics from one of the ‘‘standard’’ temperature sensor types defined in the `Temperature Sensor Types Table`. Changes can be made to the polynomial coefficients where they vary from the standard values.

CICADA Table

This table holds a few software settings such as CICADA administrators e-mail address and group name as well as a location for CICADA logs.

Modifying the tables

Introduction

The GUI which allows modification of the tables is accessed by selecting `Hardware Configuration` from the

`Options` menu, either from the CICADA main window or the Observing Window. This menu item is not available to users who are not members of the CICADA administrators group.

Once the window is visible, you will see the Instrument Table with the entry names of the table displayed in the scrolling list on the left and the details of the first Instrument in the table displayed on the right.

At the top of the window is a popup menu which gives access to the other tables. When another table is selected, the scrolling list will change to show the entry names of that table and the details section of the window will show the first entry in the table (or the entry most recently selected from the scrolling list the last time that table was selected for viewing).

Depending on the speed of the workstation you are viewing the tables from, you may see the GUI constructing itself as you move from table to table or entry to entry. This is because the GUI is built dynamically based on the contents of the table entries.

Adding Entries

In General

The easiest way to add a new entry to a table is to select an entry from the scrolling list which is similar to the one you wish to add and then click the `Add` button below the scrolling window. This will make a copy of the selected entry and blank out the name ready for you to key in a new name and modify any fields as necessary.

If you are building the tables from scratch, just click the Add button and then enter the values in the fields as required.

Note that not all fields have bounds checking or checks for correctness, so make sure you key details in carefully.

Note also that the new entry is not really in the table until you click on the `Apply` button or the `Ok` button (the Ok button performs an Apply and dismisses the window).

If you change your mind about adding the new entry, you can click on the `Delete` button to remove it.

Guidelines

If you are constructing a new instrument, the easiest way to set up the tables is to work from the bottom up. i.e. define the entries in the tables lower in the table hierarchy before the new entries in upper tables of the hierarchy. e.g. in the example shown in 2.1, you would define the tempsensors, then the focalplane, then the dewar, followed by the CCDS, the controller, the filter, then the simple instrument control and telescope and finally the instrument itself.

Modifying Entries

Modifying an entry is simply a matter of selecting it from the scrolling list, making the required changes and selecting the `Apply` button or the `Ok` button.

Deleting Entries

To delete a table entry, select it from the scrolling list and click on the `Delete` button beneath the scrolling list. CICADA will check to see if the table entry you are trying to delete is in use by any other tables higher up in the hierarchy and will display an error if this is the case and not allow the delete to proceed.

Once you have succeeded in deleting the entry from the table click on the `Apply` button or the `Ok` button to write the changes to disk.

When is it necessary to restart CICADA after changing Tables?

Most changes do not require a full CICADA restart. However, if you modify the configuration while an observing window is open, you will need to quit and restart the observing window for your changes to take effect. (This is because the tables are read once when an observing window is opened and used to configure the sub-processes which are started. Changes made to tables are not communicated to already running sub-processes.)

If you modify the Cicada Table, you must quit and restart CICADA for changes to take effect.

Instrument Changeovers

Let us suppose that your site has a telescope with cassegrain and coudé focii and that there are several CCDs in common use at either of these focus positions. There are two approaches to setting up the CICADA tables to cope with changeovers in this case. One is to define all the CCDs and controllers and just have two top-level Instruments (one for cass and one for coudé). Then, each time there is a changeover someone will need to go through the hierarchy of tables and make sure that all settings are correct.

Depending on the complexity of the situation at your site, a better approach might be to have multiple instruments set up for all common configurations, so that you might have instruments named ‘‘Cass CCD1 SDSU1’’, ‘‘Cass CCD2 SDSU1’’, ‘‘Coudé CCD1 AM3200’’, etc. This way, the tables need only be set up once and rarely modified. All that the changeover staff need to do is make one instrument ‘‘available’’ and the other unused setups ‘‘unavailable’’ using the `Available at this site` toggle in the Instrument Table entry.

About this document ...

CICADA V3.1
Configuration Manual

This document was generated using the **LaTeX**2HTML translator Version 99.2beta8 (1.43)

The command line arguments were:
latex2html `-init_file /opt/local/html/latex2html/cicada.init cicada_config.tex`

The translation was initiated by Hoa Nguyen on 2000-12-13

Next Group Up Previous

webmaster@mso.anu.edu.au

Graphing and Imaging Tool V3.0 User's Manual

Computer Section
Mount Stromlo and Siding Spring Observatories
Research School of Astronomy and Astrophysics
Australian National University

May 29, 2001

Contents

List of Figures

Introduction

As a part of the CICADA system MSSSO astronomers identified the need for a quick look window-based tool that would allow some basic interactive manipulation and display of raw CCD data. To this end the Graphing and Image Tool (GIT) was devised..

The GIT is not designed to be an image reduction facility but simply to provide basic functionality (e.g. display, statistics, basic arithmetic, and I/O). It operates on FITS [1] files, and as such can be used in tandem with versions of IRAF (*Image Reduction Analysis Facility*, KPNO).

This manual describes version 3.0 of the GIT.

Starting GIT

From CICADA

When running the CICADA software the GIT and image display can be started up via the *Tools* menu. For more details see Chapter and the *''CICADA User Manual''* [2].

Standalone

The GIT can also be operated independently of CICADA in standalone mode. To start up the GIT simply type `/opt/cicada/git [optional parameters]` (see Section for optional parameter details). This will fire up the GIT graphical user interface (GUI) and the image display.

Optional Parameters

The GIT accepts the standard X11 optional parameters (e.g. geometry, colour, etc) as well as:

-imagedir *directory*
: Default directory to read and write images and FITS files from and to. If not specified the directory from which the GIT was started is used.

-cicada *n*
: The GIT will be started in CICADA mode. See Chapter for details.

-debug
: Switches on debug mode.

The User Interface

The GIT user interface (Figure) was designed and constructed using a GUI builder. It is an X-window based GUI built from Motif widgets, with a few extra specialist widgets.

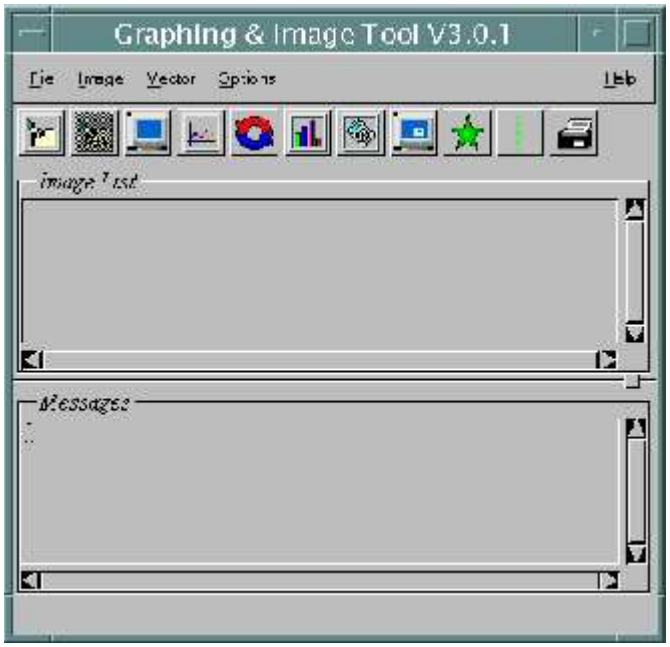


Figure: GUI upon initial startup

The GUI is comprised of six distinct and separate areas :

- Menu Bar (Section)
- Toolbar Button area (Section)
- (scrollable) Image List (Section)
- (scrollable) Messages area (Section)
- Help text (Section)
- Graphics Help text (Section)

Menu Bar

The menu bar contains pull-down menus (*File*, *Image*, *Vector*, *Options* and *Help*). These menus can be activated with either left or right mouse button. Single clicking will pop the menu up. Holding the mouse button down will only keep menu items visible while the button is depressed over the menu title.

Some menu items will appear greyed out from time to time. When an item is greyed out the specific option is not currently meaningful. For example, when the GIT is started up the menu item for displaying an image will be inactive, and will stay that way until the user loads an image.

Some menu items have *keyboard accelerators* (Section) associated with them.

The File Menu

The file menu has three main uses :

- Reading and writing FITS files into and out of the GIT (Chapter).
- Displaying FITS header keywords (Section)
- Exiting the GIT

The Image Menu

The image menu is designed mainly to operate on 2-dimensional images and allows the user to :

- Display an image (Section)
- Perform arithmetic on images (Chapter)
- Examine stellar profiles (Chapter)
- Calculate full width half-maximum (Section)
- Perform a focus fit of a focus image (Section)
- Calculate statistics (Chapter)
- Extract a sub-image (Section)
- Get a hardcopy of an image (Section and Chapter)
- Store an image in the GIT internal memory (Chapter)

The entries for *Statistics*, *Subimage* and *Print* are described in detail in Chapter , and Sections , and .

The Vector Menu

The vector menu is designed to operate on 1-dimensional images and allows the user to :

- Plot a vector (Section)
- Show/hide the vector plot window
- Store a vector in the GIT internal memory (Chapter)

The Options Menu

The Options menu has items that permit the user to start up a private image display when in CICADA mode, or to reconnect to the CICADA display. The user can also access the GIT *preferences* (Chapter) under this menu.

The Help Menu

The help menu currently has options *What's New* and *About GIT*. For more information see Chapter .

Keyboard Accelerators

A number of menu items have keyboard accelerators associated with them. All are of the form `control+<key>`. This means the appropriate control sequence at any time will select that command without use of the menu. Table lists all accelerators.

Table: Keyboard Accelerators

`Ctrl+<Key>`	Operation
A	Image Arithmetic
D	Display Image
Q	Quit
H	FITS Header
I	Sub-Image (via display)
M	Interactive Vector Modification
P	Star Profile
F	Full Width Half Maximum
L	Load Image
S	Image Statistics (whole Image)
U	Unload Image
V	Vector Plot mode
W	Write Image

Toolbar Button Area

This area contains a number of icon buttons for commonly used operations, some of which do not have keyboard accelerators associated with them. The buttons are one-off operations as opposed to putting the GIT into a mode. The buttons and the commands they carry out are described in Figure . When the cursor is moved over one of these icons, a line of help text is displayed in the line help area of the GUI.

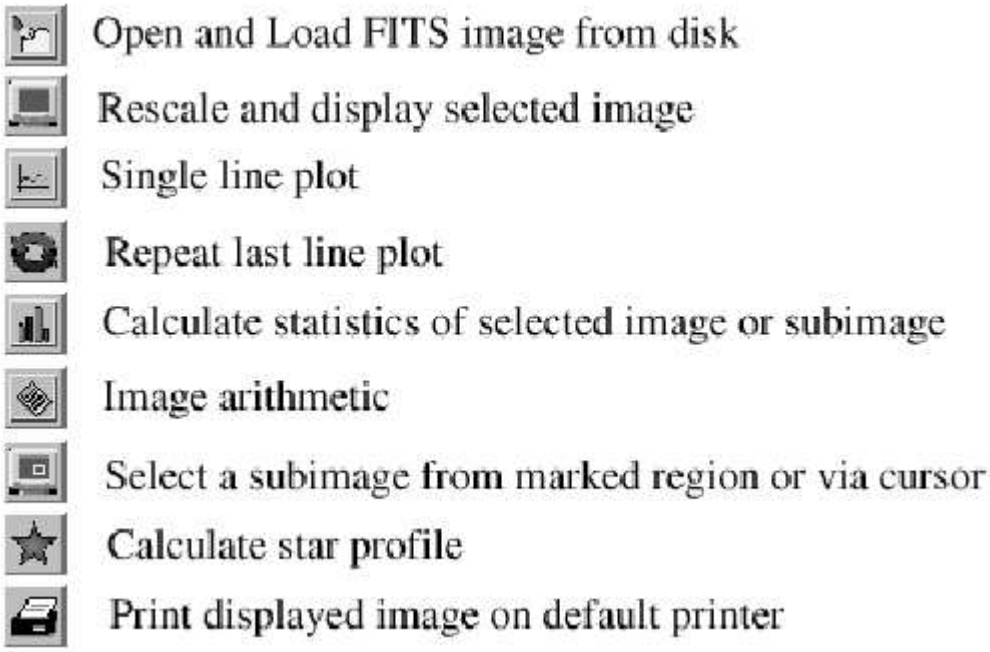


Figure: GUI toolbar buttons

Image List

The image list is a scrollable window containing the names of all images currently loaded into the GIT. These names are full pathnames of FITS files on disk. If the user tries to load or create an image with the same name as one already in memory they will be asked whether they wish to overwrite. Only one image can be currently in use, and the name of that image is shown in inverse video.

Single clicking with the mouse will select an image as the *current* one, and an informational message indicating that it has been loaded is displayed. The selected image is also displayed on the appropriate display depending on whether it is 1- or 2-dimensional.

Informational Text

Informational text is located in a scrollable window in the right portion of the GUI. Messages appear when images are loaded, unloaded, displayed; when statistics are calculated; sub-images taken, etc. As well, warnings and error messages are written to this area. There

There are also progress and informational messages displayed in the text area of the GUI window while GIT is processing or when the cursor is moved over the toolbar.

Graphics Help Text

When the user selects a mode that requires graphics input, either from the `image display` or the `vector display`, legal keystrokes and their operation are displayed in this help text area. Legal keystrokes for each mode are listed in the appropriate sections of this manual.

Preferences

There are a number of preferences available for the user to customise the behaviour of GIT. Choose*preferences* under the *Options* menu to see a pop up a window with available parameters (see Figure for an example). Separate preference screens for `Image Display` (Section), `Vector Display` (Section), `Statistics` (Chapter), `Star Profile` (Section) and `General` (Sections and) are available.

Appropriate sections of this manual describe each of the preferences, with valid ranges and values. To select a screen click either the LH and RH mouse button over the *Select Category* menu button, and select the required preference set to display.

After setting one or more preferences the user may click *Apply* which will ensure that the new values are picked up by the GIT but leave the window open. Selecting *Ok* will save the preferences and close the popup window. *Cancel* will remove the window without updating preferences (although any prior *Apply*'s will have already been carried out).

Preferences File

Preferences are written to disk in the file `$HOME/.gitrc`, where `$HOME` is a pre-defined system environment variable pointing to a user's home area (see Appendix for an example `$HOME/.gitrc` file). When the GIT is started up the `$HOME/.gitrc` file is read, if it exists, and is re-read whenever the preferences window is popped up.

Users can edit the `.gitrc` file by hand but must ensure that they do not corrupt the format. If the file is edited while the GIT is running, changes will take effect when the preferences option is selected.

Lines beginning with a `#` are comment lines, they are followed by the parameter name and value separated by a colon. For example,

```
# Image display scaling type (HOPT/MINMAX)
image_display_mode : HOPT
```

FITS Files

The GIT uses its own internal data format which has many similarities to the FITS format.

Images are read into, and written out from, the GIT's internal memory via disk FITS format files. These may be 1- or 2-dimensional FITS images. Irrespective of the FITS data type (integer, real or double precision) the GIT stores the data internally in 4 byte floating format (i.e. single precision). This means a double precision FITS file may cause over or under-flows. At this stage there is no plan to enable the use of double precision numbers due to memory and speed restrictions. If a double precision FITS file is read into the GIT the user will receive a warning message.

Preferences

There is no separate preferences page for FITS files, but one user-definable preference is found under the *General* category - *FITS output*.

There are two standard FITS header keywords : BSCALE and BZERO. Their default values are 1 and 0, respectively. Data values in a FITS file are scaled such that

$$true\ value = file\ value \times \mathrm{BSCALE} + \mathrm{BZERO}$$

This permits data with a large dynamic range, or a large scale value, to be represented by a more compact data type than would be otherwise possible. There is a drawback in a possible loss of precision.

A user has the ability to set the preference such that either precision is preserved or the output data type is the same as the input data type. If precision is preserved then the output data type is set to real (`BITPIX = -32`) so that BSCALE is left as 1. Otherwise, BSCALE and BZERO are calculated, written to the FITS header record, and data values scaled accordingly with a possible loss of accuracy.

Reading an Image

To read an image into the GIT select the *Load Image* item in the *File* menu (accelerator: `Ctrl+L`). A standard file-selection popup window is then displayed allowing directory browsing for the required file.

As a FITS file is being read a message of the form:
`FITS READ:` *`directory/filename`*
will appear in the information window. Upon successful loading a further message will be issued:
`IMAGE LOAD:` *`filename`* `(x × y)` where (x ×y) are the dimensions of the image.

Images that are successfully loaded are visible in the scrollable image list portion of the main GIT window. Their name is the full pathname of the original FITS filename on disk.

When an image is loaded it becomes the *current* image and is displayed. The current image is indicated in the image list by being shown in reverse video.

Unloading an Image

Any image read into the GIT takes up an amount of computer memory approximately equal to $4 \times x \times y$ bytes (i.e. 4Mb for a 1024×1024image). - when it is the current image. All other images in the list are not loaded into memory until they are selected. The GIT provides an unload mechanism to remove unwanted images. To use this, select the image to be unloaded, and then *Unload Current Image* (`Ctrl+U`) from the *File* menu. When in CICADA mode images are unloaded automatically as soon as more than the maximum number of CICADA images is exceeded (see Chapter).

The GIT provides a mechanism to ensure that a user does not accidentally run out of memory when reading images in, or operating on them. Before each memory intensive operation a check is made of the available memory to determine if it will still leave a GIT-defined amount free after the operation. If not, the user will be notified and will not be able to load the image until some memory is freed up.

There are several possible sources of memory usage. Examples to look for are:

- Too many memory intensive applications running (for example, Web browsers, News readers). If this is a likely cause kill off some to free memory.
- `/tmp` may have been filled up with junk files. Check and remove if necessary.

Writing an Image

Any image in the GIT memory may be written out to disk in FITS format. To do this, select the image to be written, and then *Write Image* from the *File* menu (accelerator: `Ctrl+W`). A file selection box will popup whose areas and operation is the same as when reading a FITS file into the GIT (see Section). The default selection is made up of the current directory and a filename comprising the image name stub with `.fits` appended. The user can then change the output directory, filename and/or the file extension and write the image out by hitting the *OK* button, or `<return>`.

It is not possible to overwrite a FITS file that already exists. If this is attempted a message will be printed to the informational window :
`File (`*`filename`*`) already exists`

Any images created within the GIT will be written out as real format FITS files (i.e. `BITPIX = -32`). If an input image was a FITS file the output type will be the same as the input type (e.g. short integer input will be written out as short integers). However, in this case it is necessary to determine whether the desired output type is compatible with the image range and scaling. If not, if the user has asked for precision to be maintained (Section) the output bitpix is set to real, otherwise the output type is kept the same and `BSCALE` and `BZERO` are calculated and used.

FITS Headers

If an image is read in from a FITS file the user is able to look at all the FITS header keywords/values present in the original file. This is done by selecting *FITS Header* from the *File* menu (accelerator: `Ctrl+H`). If the operation is done on an image that is not a FITS file a warning is printed to the informational window: `FITS HDR: Not a FITS file.`

WARNING : FITS Header Information

An image stored in the GIT internal memory that was read from a FITS file retains only a few of the original FITS header keywords that are directly required in creation of the image :

BITPIX
: The number of bits/pixel (i.e. the datatype)

NAXIS
: The number of axes (i.e. 1- or 2-dimensional)

NAXIS1, NAXIS2
: The dimensions of the image axes

BSCALE, BZERO
: (if they exist) Specifying scaling of the data values from the FITS file to actual values

All other keywords are ignored. Hence, if the user reads a FITS file in, and then writes a copy of it back out (whether or not it has been modified) any other original FITS keywords will be lost.

Image Operations

2-Dimensional Images

Image Display

The GIT uses the MSSSO modified version of the IRAF `ximtool` for displaying 2-dimensional images. When the GIT is started it automatically fires up a custom version of `ximtool` using private input and output pipes. The full functionality of `ximtool` is available to a user using the GIT. To make the best use of `ximtool` users should be familiar with its capabilities. When the GIT is exited the image display will be closed down.

To display an image, select it from the image list. To redisplay the currently selected image, select *Display* under the *Image* menu (accelerator: `Ctrl+D`), or use the *Display* button.

Preferences

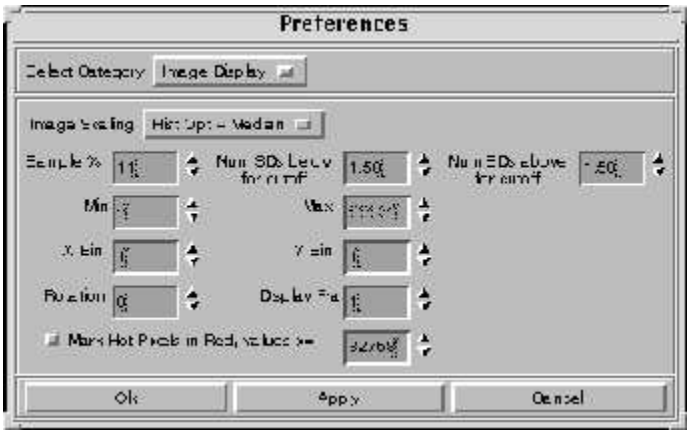


Figure: Popup window for image display Preferences

Image Scaling
: The image scaling preference defines how the minimum and maximum displayed pixel intensities are determined. Pixels below the minimum value (lower cutoff) are shown in ''black'' while those greater than the maximum (upper cutoff) appear ''white''. The three possible values are: *HistOpt - Median*, *HistOpt - Mode* and *Min/Max*. The *Min/Max* option allows the user to specify the cutoff values, while with *HistOpt* values are calculated automatically. [Valid values : `HOPT_MEDIAN`, `HOPT_MODE`, `MINMAX`]

Sample %
: Used with *Hist Opt* scaling. It represents the percentage of total number of pixels in an image to sample when determining the lower and upper intensity cutoffs. The sampling is done linearly. The larger the value the slower the calculation of cutoff values. For most images a value of 5 - 15% is adequate. [Valid range 1...100]

Num SDs below for cutoff
: Used with *Hist Opt* scaling to calculate lower intensity cutoffs to be a number of standard deviations below the median or mode. Values of 1 - 1.5% are usually adequate in obtaining images displayed with a good dynamic range. [Valid range 0.01...10]

Num SDs above for cutoff
: Used with *Hist Opt* scaling to calculate upper intensity cutoffs to be a number of standard deviations above the median or mode. Values of 1 - 1.5% are usually adequate in obtaining images displayed with a good dynamic range. [Valid range 0.01...10]

X Bin
Binning factor for display compression in the x-direction. [Valid range 1...20]
Y Bin
Binning factor for display compression in the y-direction. [Valid range 1...20]
Rotation
Angle to rotate the image by before display. [Valid range 0...359]
Display Frame
`Ximtool` frame in which to display the image. `Ximtool` has the ability to dynamically create frames when requested. [Valid range 1...4]
Hot Pixels
If checked, will enable pixel intensities greater than the selected value to be displayed in red.
Hot Pixel Value
A numerical value which is the limit, above which, hot pixels will be shown if that preference is selected. [Valid values: any real number]

Saturated Pixels

The ability for a user to be able to easily see saturated pixels on a CCD image has driven this functionality. The user can specify whether pixels greater than a given threshold limit are shown in red by checking the *Hot Pixels* toggle button to enable it.

Note that if a user modifies the image display colour look-up table to one that has reddish tones these ''saturated'' pixels may not be visible.

Sub-Images

Sub-images (either 1 or 2-dimensional) can be created from a displayed image (Section) using the image display, or the limits can be specified manually (Section). In both cases the sub-image is sent to the appropriate image display, and the user can then elect to store it in the GIT's internal memory if required (Chapter).

When a 1-dimensional image is extracted, sky subtraction will be performed if the user has selected that preference (Section) and the extraction was done via the image display. See Section for details of 1-dimensional image extraction.

Sub-image via Image Display

A sub-image can be selected from a displayed image via the image display using the *Subimage via Display* option under the *Image* menu (accelerator: `Ctrl+I`). This puts the GIT into **subimage** mode and warps the cursor into the image display. Valid keystrokes (case independent) which can be used in the image display window are listed in the *Graphics Help Text* section of the main GIT window (Table). **This mode must be exited before control is returned to the main GIT window.**

Table: Valid Sub-image mode keystrokes (case independent) for the image display.

`Keystroke`	Operation	Dimension
C	Column extraction	1
Q	Quit subimage mode	
R	Row extraction	1
X	X-Y slice extraction	1
<space>	Use twice specifying diagonally	
	opposite corners	2

Sub-image via Popup Window

A sub-image of a displayed image can be selected via a popup window by the *Subimage via Popup* option under the *Image* menu. Lower and upper limits in X and Y are specified. If a value is entered outside the image limits it is automatically reset to the lowest or highest legitimate value. Selecting *Apply* will extract the sub-image, leaving the popup visible. Selecting *Ok* will extract the sub-image, and close the popup.

To extract a 1-dimensional sub-image give either the X or Y low and high values the same, as required. It is not possible to extract a 1-dimensional X-Y slice via this method, use the image display method instead. (Section).

Hardcopy

A hardcopy of the currently displayed image can be made by selecting *Print* under the *Image* menu. For more details on printing see Chapter .

1-Dimensional Images

Image Display

The GIT uses MIT plotter widget for displaying 1-dimensional images (vectors) (see Figure for an example).

To display a vector choose it from the image list and then select `Display` under the *Image* menu (accelerator: `Ctrl+D`), or use the *Display* button. Double clicking on a vector in the image list will also load and then display it.

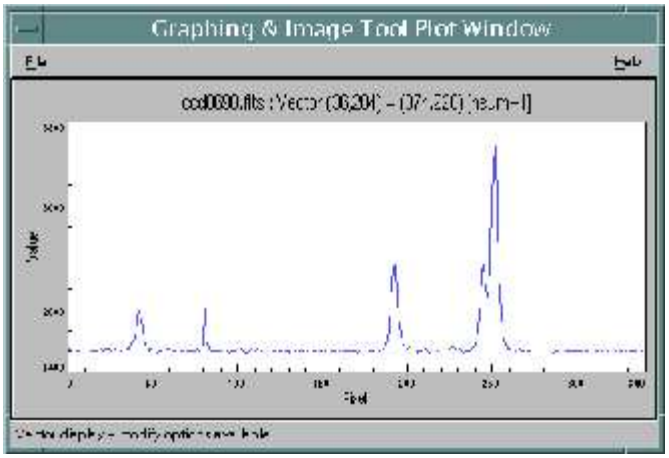


Figure: X-Y slice from pixel (36,204) to (374,220) of the image `ccd0698.fits` not summing any additional rows, with no sky subtraction, and not zoomed.

The plot window may be resized at any time and the current plot will be resized and redrawn.

Preferences

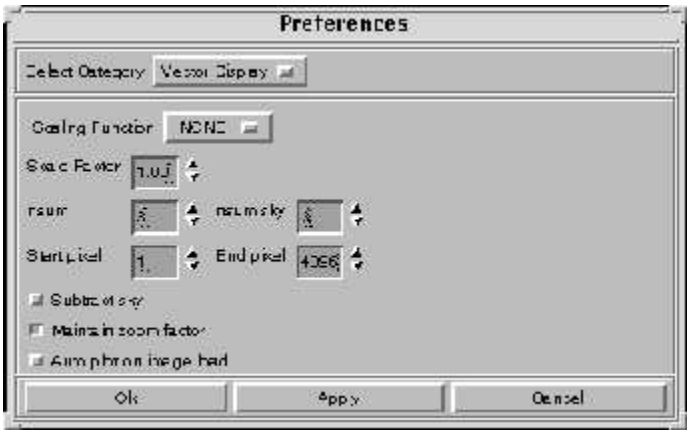


Figure: Popup window for the Vector display Preferences

Scaling Function
Define a function (*none*, *natural logarithm*, *log base 10*, *mode*, *square*, *square root*, or *inverse*) to be applied to the data before plotting the vector. With the *mode* calculation the lower and upper cutoff values are taken from those given as the statistics preferences (Section). [Valid values: `NONE`, `LN`, `LOG10`, `MODE`, `SQR`, `SQRT`, or `INVERT`]

Scale Factor
A multiplicative factor applied to the data before plotting if a scaling function of either *none* or *mode* is used. [Valid values: any real number]

nsum
Number of image ''lines'' to add together before plotting. ''Lines'' can be either rows, columns, or X-Y slices depending on the type of extraction used (see Section). [Valid range: 1...100]

nsum sky
Number of image sky ''lines'' to add together. These rows are normalised to match the image `nsum`. [Valid range: 1...100]

Start pixel
Lowest pixel to use in a vector plot. [Valid range: 1...4196]

End pixel

Highest pixel to use in a vector plot. [Valid range: 1...4196]

Subtract sky
If checked, then sky subtraction is performed before plotting.

Maintain zoom factor
If checked, causes the vector plot zoom factor to be remembered and used with subsequent plots.

Auto plot on image load
If checked, then the last plotted ''line'' is re-plotted when a new image is loaded in the GIT memory when running in CICADA mode.

Sub-image Extraction

To extract a 1-dimensional image from a displayed 2-dimensional image select *Line Plot* from the *Vector* menu (accelerator: `Ctrl+V`). This will put the GIT into **line plot** mode and warp the cursor to the image display. Valid keystrokes are those listed in Table . **Note that the mode needs to be exited before control is returned to the main GIT window.**

Table: Valid line plot mode keystrokes (case independent) for the image display

`Keystroke`	Operation
C	Column extraction
Q	Quit subimage mode
R	Row extraction
X	X-Y slice extraction

If the user wishes to do a single line plot, then the `Line Plot` icon button may be used instead of entering line plot mode. The same keystrokes are available.

The *Repeat Line Plot* icon button will repeat the last line plot again (same type of plot, same coordinates, same zoom factor). This is of use when displaying a number of images and wanting to do a vector plot of the same section.

When a 1-dimensional sub-image (vector) is extracted from a 2-dimensional image there are a number of parameters that affect how the vector is formed. These are described above (Section).

Firstly, the parameter *nsum* is used to determine how many ''lines'' to sum together. The ''line'' that the user specifies is the central ''line''. If sky subtraction is required the user will have specified a sky ''line'' as well. The parameter *nsum sky* is then used to create a summed sky ''line''. The sky is normalised to the image before being subtracted off the vector. Finally, if a *scaling function* and/or *scale factor* was specified they are applied to the vector, and the result plotted.

If the user has selected sky subtraction then after specifying either the row or column to plot they will be prompted for a sky row or column. **Note: in the current version there is no sky subtraction for X-Y slices, nor for vector extraction through the sub-image popup window.**

If the *maintain zoom factor* option is set then new vector plots will be done with the current zoom factor.

Vector plots are titled with: image name, row/column/x-y values, *nsum*, an indication if sky subtracted, and zoom factor (if zoomed).

Hardcopy

To get a hardcopy of the currently plotted vector select *Print* under the *Vector* menu. For more details on printing see Chapter .

Manipulation

It is possible to manipulate a 1-dimensional vector plot somewhat via **modify plot** mode. This is entered by selecting the *Modify* item under the *Vector* menu (accelerator: `Ctrl+M`). Table shows the valid keystrokes available from within the plot window while in this mode. Note that this mode must be exited before control is returned to the main GIT window, and that the GIT window will **not** be refreshed while you are in the mode. It is anticipated that this limitation will be addressed in a future release.

Table: Modify Vector Plot Mode Keystrokes

Key	Operation
A	Expand and Auto-scale
C	Redraw Original
F	Full Width Half-Maximum
M	Statistics
Q	Quit
R	Reverse/Flip
S	Smooth
U	Zoom out
Z	Zoom in
> *or* .	Display line plot section to the Right
< *or* ,	Display line plot section to the Left
<space>	Print (x,y) coordinates

Expand and Auto-scale
Hit this key twice, specifying the x-position of both ends of the expanded plot. The y-scale of the new plot will be determined based on the data in the expanded section.

Redraw Original
The original plot is redrawn, with no smoothing or zooming.

Full Width Half-Maximum
This key must be hit twice to determine the FWHM of a profile, once on either side of the line. The y-position is important because it is used to calculate a base-line. The result is output to the vector plot window. See Chapter for general details on FWHM calculations. This does not work for absorption lines.

Statistics
This key must be hit twice, to specify the x-position of both ends of the vector segment for which statistics are calculated. The minimum, maximum, and mean values, together with standard deviation,

are output to the vector plot window.

Quit
Use this to exit modify vector plot mode.

Smooth
The user will be prompted to enter a smoothing value (odd number) in the vector plot window. A running box of the requested size is used for smoothing.

Zoom In/Unzoom
The plot is zoomed in (or out) by a factor of 2 in the x-direction around the x-position of the cursor. The y-scale is auto-scaled.

Left/Right Sections
Using the less-than and greater-than keys (or comma and period) allows a user to move along an expanded plot in the x-direction.

(x,y) Coordinates
Hitting the <space> bar causes the x-value of the cursor position to be output to the vector plot window, together with a linearly interpolated y-value.

Storing an Image in Memory

Any image (either 1- or 2-dimensional) that is displayed can be stored in the GIT's internal memory. Select *Store in Memory* under the *Vector* or *Image* menu as appropriate. A file selection popup window will appear allowing the user to specify the new FITS file name.

Hardcopy

The hardcopy device is that which the user has defined in their Unix environment at the time the GIT is started up. This is controlled by the PRINTER Unix environment variable. If not set, hardcopies will be sent to the default system-defined printer.

The engine driving the output for 2-dimensional images is `ilaser` -- the same as that available from FIGARO. There are no user-configurable options with the GIT version.

Vector (1-dimensional) plots are carried out using the MIT plotter widget (see, for example, Figure).

Image Statistics

When statistics are calculated the results are written to the informational text section of the main GIT window. Information comprises the image name, and dimensions; the minimum and maximum values; the mean, mode and standard deviation. For example,

```
IMAGE STATS: /tmp/130.fits (416x578)
   min=0.00, max=65535.00, mean=1622.30,
   std dev=754.61, mode=1607.00
```

2-Dimensional Images

2-dimensional image statistics can be calculated for the whole image, or specified in either a popup window or via cursor input on the image display.

For the whole image select *All Image* from the *Statistics* menu under the *Image* menu (accelerator: `Ctrl+S`). Alternatively, use the *Statistics* icon button on the main GIT window.

To specify part, or all, of an image via a popup window select *via Popup* from the *Statistics* menu. A window will pop up with default values in X and Y of the image dimensions. The operation of this window is identical to when specifying sub-image extraction (see Section).

To specify part of an image via the image display select *via Display* from the *Statistics* menu. The GIT will enter **statistics** mode which is identical in operation to sub-image extraction via the image display (Section). Either a 1- or 2-dimensional section will be used for statistics depending on the keystroke(s) (Table).

1-Dimensional Images

Statistics can be calculated for part, or all, of a 1-dimensional image while in modify vector mode (Section).

If a 1-dimensional image is the current image in memory, either via selection from the image list or by being the last image displayed, the *Statistics* button (accelerator: `Ctrl+S`) will operate on that image.

Preferences

There are a number of configurable statistics preferences which are accessed via the *Statistics* category from the *Preferences* option under the *Options* menu (Figure).

Show Statistics Plot
If checked, a line plot window will show a histogram of values in the statistics.
Ignore Values < =
Values less than this will be ignored when calculating statistics. [Valid values: any real number]
Ignore Values > =
Values greater than this will be ignored when calculating statistics. [Valid values: any real number]

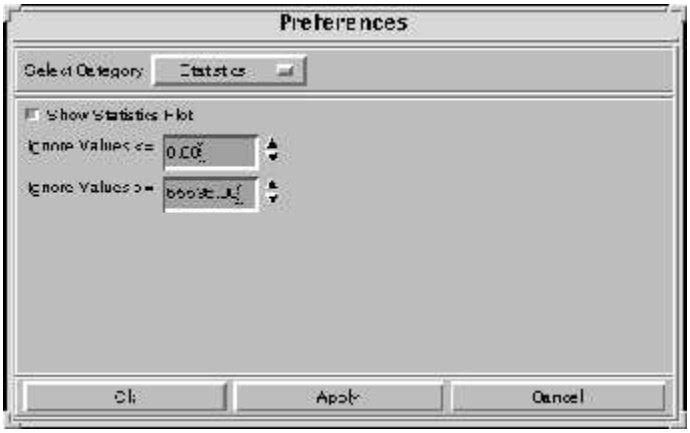


Figure: Popup window for the image Statistics Preferences

Image Arithmetic

The GIT is able to carry out a range of basic arithmetic functions on images. These are operations:

- with images as both operands
- with an image and a scalar
- to apply a function to an image

To enable calculations to be carried out select *Arithmetic* from the *Image* menu (accelerator: `Ctrl+A`). This will pop up the arithmetic window (Figure).

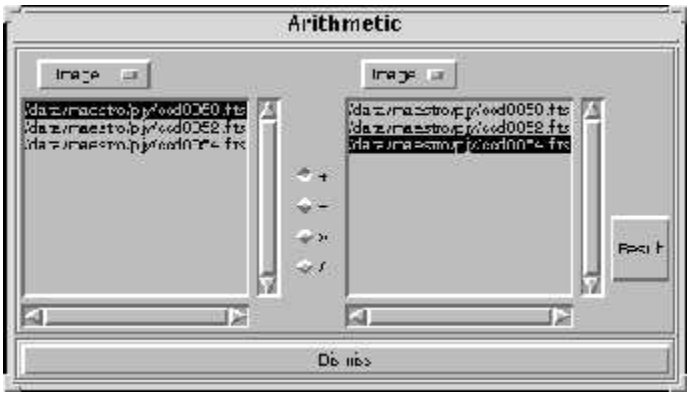


Figure: Arithmetic popup window

To select the type (that is, Image, Scalar, or Function) for the left and right operands use the appropriate pull-down menus in the Arithmetic popup window. When an image is used the list of currently loaded images is displayed. When a scalar operand is required a numeric box will show up. For functions a list of available options will appear as check boxes.

Function types available are:

- Natural Logarithm (`ln`)
- Square (`sqr`)
- Inverse
- Square Root (`Sqrt`)
- Base-10 Logarithm (`Log10`)

Results

Clicking the *Result* button will carry out the calculation, and if an image is involved pop up a window allowing the user to save and the resulting image. The result is then displayed and set to be the current image. Memory checking is done, as with loading any image, to ensure that sufficient resources exist on the computer.

Irrespective of whether an image is involved information on the calculation will be written to the informational section of the main GIT window.

The input type of images, and the types of any function or scalar used are checked so that the FITS file is created with appropriate precision. Some examples are:

- two *short integer* input images would create a *short integer* output FITS file
- two *integer* input images would create an *integer* output FITS file
- any division creates a float output FITS file
- any function creates a float output FITS file
- application of an *integer* scalar to an *integer* image will create an *integer* FITS file
- application of a *float* scalar to an *integer* image will create a *float* FITS file

Error Checking

When arithmetic is done on two images they are first checked to ensure they have the same X and Y

dimensions. If not, the user is informed and no calculation done.

It is possible to get divide by zero and other non-meaningful numbers in calculations, and these are handled gracefully. For example:

```
ARITH: 1.000000 / 0.000000 = Inf
ARITH: Sqrt(-1.000000) = NaN
```

These results are part of the IEEE floating point standards, and values such as infinity and not a number (NaN) are representable in the IEEE floating point format.

Image Object Profiles

The image display can be used to plot radial profiles of objects. The option *Star Profile* under the *Image* menu (accelerator: Ctrl+P) will put the user into **profile** mode. The keystrokes listed in Table are the only ones available.

Table: Profile Mode Keystrokes

Key	Operation
Q	Quit
<space>	Select object

Preferences

The *Star Profile* page (Figure) of *Preferences* under the *Options* menu has the following:

Display profile
: If checked, the profile is plotted to the line plot window.

Aperture Size
: The radius of the aperture (pixels) to be used when determining the centroid, and for setting plotting limits. [Valid range: 1...50]

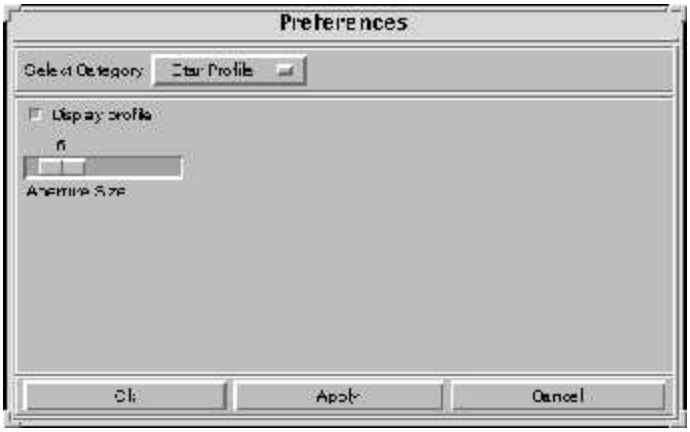


Figure: Popup window for the Star Profile Preferences

Use

The user selects an object using the image display cursor, and the GIT determines a centroid based on an aperture size. If the centroiding fails a message is printed out to the informational text area on the main GIT window. If the *display profile* preference is selected the radial profile is plotted.

Irrespective of whether the profile is plotted the (x,y) values of the centroid and full width half-maximum (Chapter) are written to the informational text area. For example,

```
Centroid: (203.78,341.86)  Aperture Size=12
FWHM = 4.52
```

Full Width Half-Maximum

The GIT may be used to determine the full width half-maximum (FWHM) from a 2-dimensional image via the display, or from a 1-dimensional image via the line plot window.

The FWHM is calculated by fitting a Gaussian to the data values using the Levenburg-Marquardt method.

2-Dimensional FWHM

This is invoked by the *FWHM* button on the *Image* menu and will put the user into **FWHM** mode with the keystrokes listed in Table available. The GIT stays in the mode until explicitly exited, at which stage control is returned to the main window.

Table: FWHM Mode Keystrokes

Key	Operation
Q	Quit
X	at each end to select object

The user must specify each end of the X-Y slice that is used to determine a FWHM. This 1-dimensional vector is plotted to the line plot window. The value of the FWHM is written to the GIT informational text window.

1-Dimensional FWHM

A 1-dimensional image which is plotted can have a FWHM calculated from within modify line plot mode -- see Section for details of use.

Focus Fitting

This is invoked by the *Focus Fit* button on the *Image* menu and will start an automated focus fitting procedure of the selected image - if it is a focus image. GIT will check the image FITS headers to see if the `FOCNEXPO` and `FOCSHIFT` keywords are present, if not then GIT prompts the user for values to use (see figure).

Figure: Popup window for the Focus Fitting parameters

Using these parameters GIT attempts to locate a sequence of stars in the image to which FWHM are calculated and then a quadratic is fitted to the FWHM and focus position data. This is then plotted with the optimal focus position indicated.

Miscellaneous

CICADA Mode

When the GIT is started up with the optional parameter `-cicada` *n* it will automatically load images into its memory directly as they are read out from a CCD with CICADA. Informational messages will appear normally, and the images will show up in the image list together with any images the user reads in or creates.

Display of the CCD images is done automatically by the CICADA software [2].

The parameter `n` indicates the maximum number of CICADA images to be retained. This value can be modified from within the GIT as a preference. NB. Normally the GIT is started from CICADA itself where this value defaults to 10.

Image Display

By default, when in CICADA mode the GIT uses the CICADA ximtool image display. If GIT attempts to display an image while CICADA is displaying, a message indicating that the ''display is already in use'' is written to the message window.

Alternatively, the user can start up a ''private'' image display when in CICADA mode, thus avoiding any contention. This is just the normal standalone display that is started when not running CICADA. There is a preference that can be set to allow this as default (Section). When this is used the CICADA images continue to be displayed automatically to one display, while the user is free to use the other for display and/or manipulation.

A user can start this separate image display using *Start Private Ximtool* under the *Options* menu. It can be cancelled and the GIT reconnected to the CICADA display at any time via *cancel Private Ximtool* under the *Options* menu.

Preferences

CICADA preferences are found under the *General* preferences screen (see Figure).

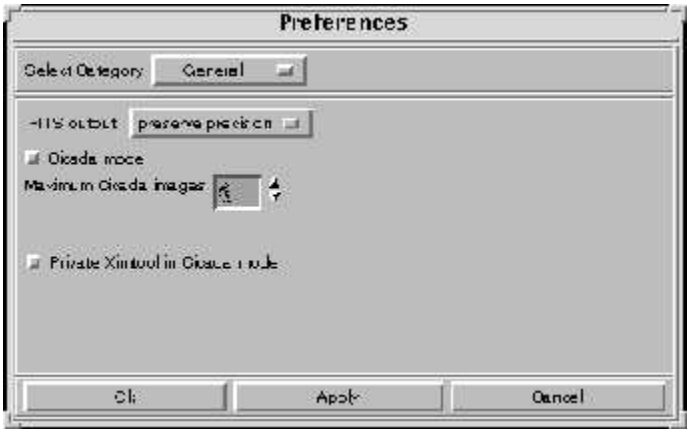


Figure: Popup window for the General Preferences

Cicada mode
If checked, indicates to use CICADA mode upon startup. If the GIT is started by CICADA then this box will not need to be checked by the user for correct operation.

Maximum Cicada images
The user can modify the maximum number of images to be held in the GIT's image list while it is running. [Valid range: 1...10]

Private Ximtool in Cicada mode
Checking this box allows the GIT, upon startup, to use its own ''private'' image display when in CICADA mode (see Section for details).

Quitting

To quit the GIT select *Quit* from the *File* menu (accelerator: `Ctrl+Q`). Any images which have been stored in memory but not written to disk can be saved at this time. If there are any such images a window will pop up asking the user if they wish to save the image(s). The user has the following options:

- Saving none and exiting immediately (`No to all`).
- Saving all automatically (`Yes to all`).
- Prompting whether to save one by one (`Yes` or `No`).

If `Yes to All` is selected the popup window will appear for each unsaved image (see Section).

Getting Help

This user manual is the first step in getting help. If something is not described clearly or fully in here, or something simply does not work as described please report it.

If you have a problem that is reproducible, or one that is intermittent but not reproducible at will, please report it in as much detail as possible.

If you have a ''glitch'' -- something that occurs only once and you don't see again it may not be useful to report that.

The GIT maintains a time-stamped log file to which much information is written. This is not stored in the user's area. The contents of this file can be used with any bug report to help track down problems.

All reports should be sent via email to `cicada (don't forget to add @mso.anu.edu.au if outside RSAA)` to the RSAA Computer Section. Or, if you prefer, in person.

Limitations

These limitations are known and will be fixed in future releases:

- There is no sky subtraction for X-Y slices.
- There is no sky subtraction for vector extraction through the sub-image popup window.
- Sub-image and statistics via display modes, modify vector mode and profile mode must be exited before control is returned to the main GIT window.
- Colour Table Problems - the current version of GIT uses the IRAF application `ximtool` modified by MSSSO as the image display. If you already have a number of applications running which use a fair number of colours then `ximtool` will load its own private colourmap and colour flashing will occur when moving in and out of that window. This will definitely happen if you are running a colour intensive application such as Netscape Navigator irrespective of whether you have told it to start up with a private colour map or not. We have not found it possible to have Netscape running at all while `ximtool` is in use and not encounter colour problems.

There will be other things that some users will consider limitations. It should be remembered that the GIT was designed for limited quick-look purposes and in no way should it be thought of as being a general image analysis or reduction tool. However, the authors are always open to suggestions on how its functionality, ease of use, etc could be improved.

Sample `$HOME/.gitrc` Preferences File

```
#=========================================================================
# Configuration Table: .gitrc
# This table has been automatically generated and consists of a series of
# named items , eg [name], followed by a set of parameter and value
# pairs, which could be multiple per line.
# Default valued items might not appear. Take care if editing by hand.
```

```
#=========================================================================

cicada.max_loaded_images:8
cicada.mode:0
image_display.angle:0
image_display.binx:1
image_display.biny:1
image_display.display_frame:1
image_display.display_max:1761.220318
image_display.display_min:1456.779682
image_display.hopt_contrast:0.500000
image_display.hopt_sampling:1000
image_display.hot_pix:0
image_display.hot_pix_value:65535.000000
image_display.mode:HOPT_MEDIAN
image_display.vis_scaling:0
image_display.ximtool_height:564
image_display.ximtool_screen:1
image_display.ximtool_width:512
image_display.ximtool_x:564
image_display.ximtool_y:310
misc.auto_line_plot:0
misc.image_load_dir:/priv/mistress/pjy/data/cicada/pjy/
misc.image_save_dir:/priv/mistress/pjy/cicada/
misc.mef_extension_list:1-8
misc.mosaic_mef:0
misc.private_ximtool:0
plot.plot_height:417
plot.plot_width:614
star.aperture:18
statistics.display_profile:1
statistics.plot:1
statistics.sample_full:0
statistics.sample_size:1000
statistics.scale_function:LINEAR
vector.end:4196
vector.maintain_zoom:0
vector.nsum_sky:1
vector.scale_factor:1.000000
vector.scale_function:LINEAR
vector.smooth_box:3
vector.start:1
vector.subtract_sky:0
vector.use_bscale:0
vector.vector_nsum:1
```

Bibliography

1

NASA/Science Office of Standards and Technology. *Definition of the Flexible Image Transport System (FITS)*
(September 29, 1995).

2

MSSSO Computing Section. *CICADA User Manual*
(1998).

Index

About this document ...

Graphing and Imaging Tool V3.0
User's Manual

This document was generated using the **LaTeX2HTML** translator Version 99.2beta8 (1.43)

The command line arguments were:
latex2html `-init_file /opt/local/html/latex2html/cicada.init git.tex`

The translation was initiated by Bill Roberts on 2001-05-29

Next Group Up Previous

webmaster@mso.anu.edu.au

Next Up Previous
Next: Introduction

CICADA

CCD and Instrument Control Software

CICADA - Computerised Instrument Control and Data Acquisition is a software system for control of telescope instruments in a distributed computing environment. It is designed using object-oriented techniques and built with standard computing tools such as RPC, SysV IPC, Posix threads, Tcl and GUI builders. The system is readily extensible to new instruments and currently supports the Astromed 3200 CCD controller and MSSSO's new tip-tilt system. Work is currently underway to provide support for the SDSU CCD controller and MSSSO's Double Beam Spectrograph. A core set of processes handle common communication and control tasks, while specific instruments are ''bolted'' on using C++ inheritance techniques. CICADA offers consistent user interface, image display, data archiving and quick-look analysis for a variety of instruments.

P.J. Young, M. Brooks, S.J. Meatheringham, W.H. Roberts
Mount Stromlo and Siding Spring Observatories
pjy@mso.anu.edu.au
17 September, 1996

Peter Young
Tue Oct 29 11:48:29 EST 1996

CICADA - Configurable Instrument Control and Data Acquisition

Peter J. Young, William H. Roberts, Kim M. Sebo

Mount Stromlo and Siding Spring Observatories, Private Bag, Weston Creek, ACT, 2611, AUSTRALIA

Abstract.

CICADA (Young, et al, 1997) is a multi-process, distributed application for the control of astronomical data acquisition systems. It comprises elements that control the operation of, and data flow from CCD camera systems; and the operation of telescope instrument control systems. CICADA can be used to dynamically configure support for astronomical instruments that can be made up of multiple cameras and multiple instrument controllers. Each camera is described by a hierarchy of parts that are each individually configured and linked together. Most of CICADA is written in C++ and much of the configurability of CICADA comes from the use of inheritance and polymorphism. An example of a multiple part instrument configuration - a wide field imager (WFI) - is described here. WFI, presently under construction, is made up of eight 2kx4k CCDs with dual SDSU II controllers and will be used at Siding Spring's ANU 40in and AAO 3.9m telescopes.

1. Introduction

Early releases of CICADA[1] had many built-in site-specific sections of code which meant that it was very difficult to "export" to other observatories. One of the aims of CICADA v2 was to replace these sections of the code with "self-configuring" code based around simple ASCII tables. Another aim of release 2 of CICADA was to provide support for multi-controller, multi-CCD instruments. This requirement also impacted on the design of the Configuration Tables. The table design now used by CICADA is a linked list of table items and each item is a hash table of fields which describe the item. There are tables describing Instruments, Cameras, Controllers, CCDs, Temperature Sensors, Focalplanes, Dewars, etc.

2. Building a Configuration - the Configuration GUI

Figure 1 shows the CICADA Hardware Configuration windows for the Controller Table. It is by using these and other configuration windows that the tables are

[1] `http://msowww.anu.edu.au/computing/cicada`

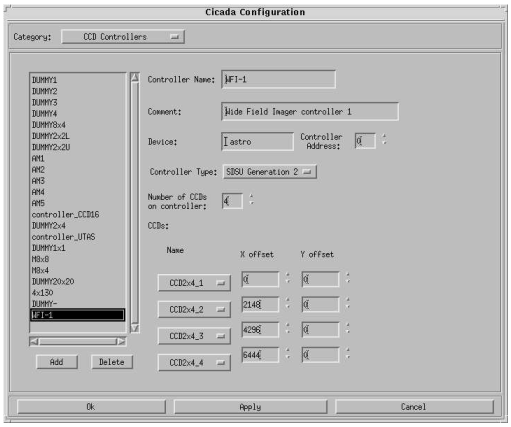


Figure 1. The Controller Configuration GUI.

built which are eventually used to dynamically configure the CICADA software.

3. Dynamic Implementation of a Configuration

The Cicada_Master process is the communication and operations control centre for CICADA, functioning as the request interpreter, the activity status monitor and the traffic director. It is started at the beginning of a CICADA run and is responsible for interpreting the instrument configuration.

This is done by instantiating a Cicada Instrument object that represents the hierarchical instrument description by using a set of individual Instrument_Part objects. Each of these Instrument_Part objects starts a set of instrument control processes on configured instrument hosts and handles activity and status requests passed to it by the Cicada_Master process. Each activity request runs as a separate POSIX thread, so simultaneous control of each part is maintained. In addition separate threads are started to fetch status information from the instrument parts. Only one activity can be running on an individual part at one time, attempts to start new activities are blocked.

The use of threads allows for full control of activities running on an instrument part, including monitoring, pausing/resuming and aborting.

The Cicada_Master process also starts a TCL interpreter and handles the execution of embedded CICADA commands in TCL scripts. These commands are passed to the Instrument_Part objects in the same way as the interactive command stream which could be from a GUI or from a command line interface.

4. Handling Arbitrary Multi-Amp Detectors

An important component of the configurability of CICADA is the ability to handle the geometry of CCD mosaic layouts in a generalised way.

At the GUI level (see Figure 2, left), the astronomer selects one (or several) regions of interest, spanning some part, or all of the mosaic and possibly in-

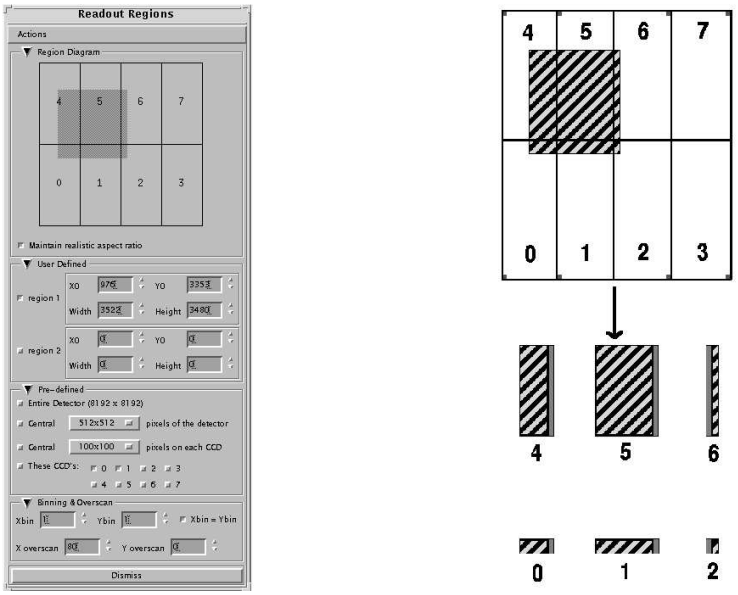


Figure 2. The regions GUI and output from the CICADA Regions Class.

cluding overscan and/or prescan pixels. However, at the readout level, a simple rectangular region in mosaic coordinates may actually span multiple CCDs and amplifiers of the mosaic.

The CICADA Regions class is a general solution to the problem of converting the specified rectangular readout subregion (in mosaic coordinates) into the actual pixels clocked out by the CCD controller(s) (in CCD/amplifier coordinates).

The geometry of the mosaic camera is specified in the CICADA hardware configuration window, where the sizes, positions and readout directions of the CCDs in the mosaic are enumerated.

The CICADA Regions class is capable of dealing with almost any practical mosaic geometry. It handles any (reasonable) number of CCDs of any size, with arbitrary multiple amplifier positions and directions, including support for overscan rows and columns for each amplifier, and independent pixel binning in either axis. Up to two possibly overlapping (mosaic-space) rectangular regions are supported, however this limit is arbitrary and could be increased without difficulty. The only assumption made is that the size of the full readout of each amplifier must be the same for all amplifiers. This constraint is likely to be true for all conventional CCD mosaic systems.

In general, a single mosaic sub-region spanning multiple amplifiers is converted to multiple smaller readout regions, one region per readout amplifier. Each of these smaller regions are ultimately written into separate FITS extensions of a multi-extension FITS file. The FITS extension headers contain sufficient geometry information so that the original mosaic-level geometry can be reconstructed.

The returned information from the Regions class is used by CICADA to construct the output FITS file from the interleaved steam of pixels coming from the controller(s), and also to display the resultant image reassembled into the same rectangular mosaic-space region selected by the astronomer.

Figure 2 at right shows at the top an example readout region selected by the astronomer in mosaic coordinates, spanning multiple CCDs. Below are shown the multiple sub-regions for each amplfier, returned to CICADA by the Regions class including overscan from each amplfier.

5. Site-specific Plug-Ins

Currently CICADA can be extended to support telescopes and filter controllers for sites other than MSSSO by user-written "plug-in" code. This is done by writing code that conforms to a CICADA defined API, building a shared library using a supplied Makefile and then replacing the default shared library. A base level CICADA C++ class implements the API as pure virtual methods that need overriding by the user derived class. Template telescope and filter control definitions are provided for a user to easily follow the required steps.

The GUI for telescope and filter control is then dynamically built based on descriptions provided in the CICADA configuration tables. This method has been used successfully at the UNSW APT telescope, the University of Tasmania and work is now progressing toward code to control the Anglo Australian Telescope.

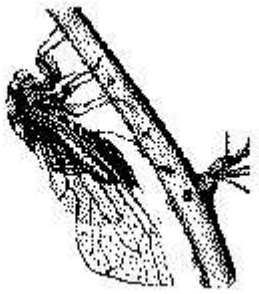
6. Current Applications

CICADA is in use at MSSSO, controlling instruments at several telescopes. The same software is used at each telescope, only the configuration files differ to cater for different controller types and CCD configurations. Currently, CICADA supports the SDSU Generation I and II controllers as well as the Astromed 3200 controller. MSSSO is currently building a Wide Field Imager (WFI)[2] in collaboration with the Anglo-Australian Observatory and the University of Melbourne. This new Instrument is an 8 CCD mosaic using dual SDSU Generation II controllers and will make full use of the configurability of CICADA as well as its multi-process, distributed design. The next major project for CICADA is control of MSSSOs existing Double Beam Spectrograph which is a dual-camera configuration(i.e. two shutters, two controllers, two CCDs, two image displays).

References

Young, P. J., Brooks, M., Meatheringham, S. J. & Roberts, W. H. 1997, in ASP Conf. Ser., Vol. 125, Astronomical Data Analysis Software and Systems VI, ed. Gareth Hunt & H. E. Payne (San Francisco: ASP), 385

[2] `http://msowww.anu.edu.au/observing/wfi/`



GSFIND Manual V2.0

Computer Section
Mount Stromlo and Siding Spring Observatories
Research School of Astronomy and Astrophysics
Australian National University

December 13, 2000

Contents

List of Figures

List of Tables

1 Introduction and Context

This manual describes the operation of a guide star location tool. This tool uses the Guide Star Catalogue published by NASA as the source of guide star positions and magnitudes. The aim of this tool is to provide a fast and convenient method of locating a guide star for astronomical observations, and assistance for guide star acquisition while observing. This guide star tool[1] was originally developed for the Wide Field Imager (WFI)[2] being developed jointly by MSSSO, AAO and the University of Melbourne, however it provides a mechanism for describing the layout of science and guide detectors in the instrument focal plane, and is therefore configurable for use with almost any instrument. While it has been developed to run in conjunction with CICADA (Configurable Instrument Control and Data Acquisition), software developed at MSSSO, it may also be used off–line in a stand alone mode.

2 Installation

2.1 GSFind

GSFind is part of the CICADA package. See the CICADA manual.

2.2 Guide Star Catalogue

GSFind requires the Guide Star Catalogue to be installed on a local disk. This catalogue has two CD volumes, and each volume should be copied to disk in separate directories. If set, the environment variables GSC_NORTH and GSC_SOUTH should contain the full path to the northern and southern catalogues respectively. If these variables are not defined, GSFind assumes the northern catalogue was installed in the directory /gsc/vol1, and the southern catalogue in /gsc/vol2.

2.3 Configuration Table

The instrument and focal plane configuration data is stored in a text formated table. If set, the environment variable GSFIND_TABLE should contain the fully qualified filename of the configuration file, otherwise GSFind will look for the file /opt/cicada/config/gsfind_table. If this file is not found, an example configuration file will be written to /tmp/gsfind_table, and GSFind will exit. The content of the configuration file will be described in section 3.

3 Configuring GSFind

3.1 Supported Systems

A single GSFind configuration table can support up to 10 focal planes. Each focal plane may contain up to 16 science detectors, and 16 guide detectors. All science detectors (CCDs) are assumed to be the same size, both in terms of the number of pixels, and the pixel dimensions. The physical dimensions of all guide detectors are also assumed to be the same size. The guider systems supported by GSFind are fixed, orbital, and rotatable. Both fixed and rotatable guider system means the location of the guide detectors are fixed relative to the science detectors. In the case of the rotatable system, the entire focal plane may be rotated about the center of the science detectors field of view. For an orbital guider system, only one guide detector is permitted. The center of this detector is constrained to move along a circle, centered on the science detectors field of view. The orientation of the guide detector does not rotate as the detector moves. This system is designed for guide detectors that are carried on an X-Y platform, but always positioned equidistant from the center of the science field.

[1] http://msowww.anu.edu.au/computing/cicada
[2] http://msowww.anu.edu.au/observing/wfi/

3.2 Table Content

The table entries for each instrument consist of a square bracketed encapsulated instrument label, and a series of semicolon separated variable=value pairs. Where the variable is an array, the value field becomes a comma separated list of values. The data in the configuration table describes the physical layout of the instrument focal plane in an X-Y coordinate system, with all dimensions measured in mm. The location of the origin is arbitrary. The orientation of the coordinate axes is such that the +ve X-direction corresponds to East on the sky, and the +ve Y-direction corresponds to North on the sky.

The nature and type of the variables in a GSFind configuration table is described below. If variable, value pairs are omitted from the configuration table, default values are assumed.

Variable	Type	Description.
n_ccds	Integer	The number of science detectors.
ccd_height	Real	Height of each science CCD in mm.
ccd_width	Real	Width of each science CCD in mm.
ccd_pixel_size	Real	Size of each science CCD pixel in mm.
ccd_xo	Real (array)	ordinate of the origin of the science detectors.
ccd_yo	Real (array)	abscissa of the origin of the science detectors.
ccd_pixel_xo	Integer (array)	X-offset of science pixel ordinate in a mosaic reference frame.
ccd_pixel_yo	Integer (array)	Y-offset of science pixel abscissa in a mosaic reference frame.
dead_ccd	Integer (array)	Set to 0 if science detector is functioning, 1 otherwise.
east_at_left	Integer	Display East at right if set to 0, or East at Left if set to 1.
is_equatorial	Integer	Set to 1 if telescope is equatorially mounted, 0 otherwise.
focal_plane_pa	Real	Indicates the direction on the sky of the +ve Y axis of the X-Y reference frame of the focal plane. This allows for rotation of the instrument on the telescope without the need to reconfigure the focal plane layout. Measured in degrees from North, increasing though East.
orientation_of_north	Real	Indicates the direction to display North in the GSFind sky chart window. Measured anti-clockwise in degrees from the right horizontal. This allows the GFSind sky–chart to be oriented the same way on the display as the image read from the detector.
n_guide_ccds	Integer	Number of guide detectors.
guide_ccd_height	Real	Height of each guide detector in mm.
guide_ccd_width	Real	Width of each guide detector in mm.
guide_ccd_xo	Real (array)	ordinate of the origin of the guide detectors.
guide_ccd_yo	Real (array)	abscissa of the origin of the guide detectors.
dead_guide_ccd	Integer (array)	Set to 0 if guide detector is functioning, 1 otherwise.
guider_type	Integer	0=fixed : 1=orbiter : 2=rotatable.
xy_rotation	Integer	0=0 : 1=90 : 2=180 : 3=270 degree rotation of the reference frame of the X-Y stage of an orbiter guider, with respect to the focal plane reference frame.
n_vig	Integer	Number of vignetted regions in the field of view.
vignette_width	Real (array)	Width of vignetted region in mm.
vignette_height	Real (array)	Height of vignetted region in mm.
vignette_xo	Real (array)	ordinate offset of vignetted region.
vignette_yo	Real (array)	abscissa offset of vignetted region.

Table 1: GSFind configuration table description

As an example, the table for the WFI instrument is shown below. This instrument has 8 science and 8 guide CCDS in a fixed guider system, and no vignetted regions.

```
[WFI at 40inch]
    n_ccds=8;
    ccd_height=61.440000;
    ccd_width=30.720000;
    ccd_pixel_size=0.015;
    ccd_xo=100.740000,69.200000,37.660000,6.120000,6.120000,37.660000,69.200000,
100.740000;
    ccd_yo=0.000000,0.000000,0.000000,0.000000,62.060000,62.060000,62.060000,62.
060000;
    ccd_pixel_xo=6144,4096,2048,0,0,2048,4096,6144;
    ccd_pixel_yo=0,0,0,0,4096,4096,4096,4096;
    dead_ccd=0,0,0,0,0,0,0,0;
    east_at_left=0;
    is_equatorial=1;
    focal_plane_pa=270.00000;
    orientation_of_north=90.0000;
    focal_plane_scale=25.037000;
    n_guide_ccds=8;
    dead_guide_ccd=0,0,1,0,0,0,0,0;
    guide_ccd_height=3.360000;
    guide_ccd_width=2.460000;
    guide_ccd_xo=135.120000,135.120000,135.120000,135.120000,0.000000,0.000000,0
.000000,0.000000;
    guide_ccd_yo=75.880000,65.520000,55.160000,44.800000,75.880000,65.520000,55.
160000,44.800000;
    guider_type=0;
    xy_rotation=0;
    n_vig=0;
```

4 GSFind Display

The GUI to GSFind is shown in figure 1.

4.1 Sky Display

The square region at the middle left of the GSFind GUI is the sky display. This shows the stellar objects as solid white circles, and (optionally) non-stellar objects as open circles. A red cross marks the center of the field. Overlaid on the sky plot is the instrument focal plane. This shows the outline of the science detectors and guide detectors in green. Vignetted regions are displayed as red rectangles, with lines across the diagonals.

4.2 Legend

The sky plot legend is shown at the bottom left of the GUI. This indicates the direction of North and East in the sky display, and the key to determine the magnitude of the objects in the sky display.

4.3 Instrument Selector

The desired instrument may be selected from the drop–down menu of the Instrument Selector. This load the configuration data from the table, and displays the currently selected instrument.

4.4 Position display and configurables

The current RA and Dec coordinates of the center of the sky display are shown in the Field : RA, Dec text boxes. These values may be modified by the user, and initiating a Read GSC will

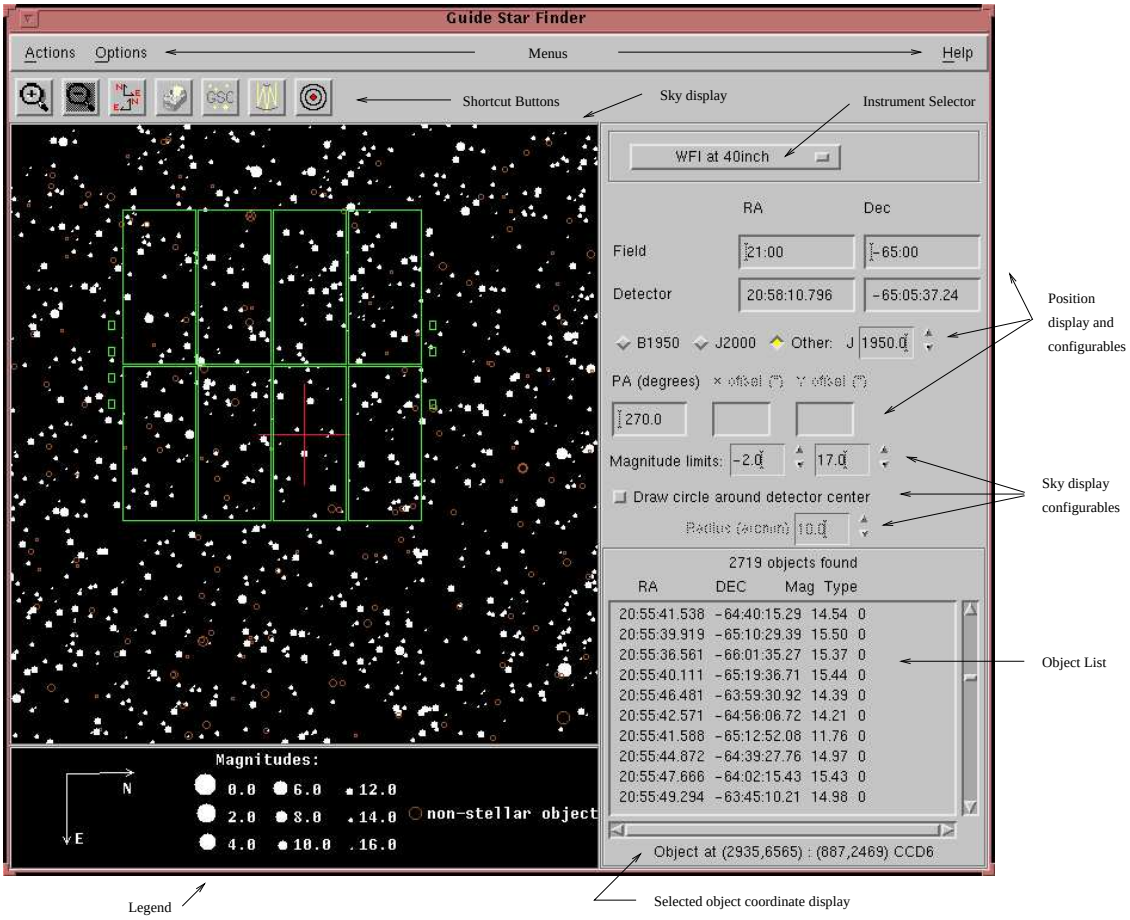


Figure 1: GSFind GUI display

redisplay at the coordinates entered. The current coordinates of the center of the focal plane are displayed in the text boxes labelled Detector : RA and Dec. These values cannot be modified by the user. The coordinate system used for both the focal plane and field center can be set to B1950, J2000, or or a Julian epoch of the users choice. The default system is J2000. The position angle of the focal plane is displayed in the textbox labelled "PA (degrees)". For rotatable guider systems, this value may be also modified to set the position angle as the user desires. The adjacent textboxes labelled X offset and Y offset display the X and Y coordinates of of an orbital guider system. These values cannot be modified by the user, and are only displayed for instruments with orbital guider systems.

4.5 Sky display configurables

The magnitude range of the objects displayed on the sky plot can be configured by modifying the magnitude limit fields. The left textbox shows the bright limit, and the right textbox the faint limit. The user may also plot a circle around the center of the focal plane. This is designed to assist the user where the instrument field of view is circular. The radius of this circle may entered in the textbox labelled "Radius (arcmin)" once the "Draw circle around detector center" checkbox is checked.

4.6 Object list

The object list contains the positions, magnitude and object type flag for all of the objects near the field center, with magnitudes between the configured limits. The object list also contains objects that are not displayed. This allows for field rotation without the need to read the GSC again.

4.7 Selected object coordinate display

An object may be selected by the user (see section 5.6). If this object falls on a science detector, the pixel coordinates of this object are shown below the object list. This is designed to assist the user in acquiring a guide star, as they may compare the coordinates of this object to those from a real exposure. The display also indicates if the objet is not on the sky ploy display area, or not on a science detector.

4.8 Menu

The primary controls, run–time configurables and user hints for GSFind can be instigated, modified or displayed using the drop–down menus at the top of the GUI. The functionality of these menu items will be discussed in turn.

4.8.1 Actions

- Zoom In : Reduce the size of the sky display field of view by a factor of 2, retaining the current field center.
- Zoom Out : Increase the size of the sky display field of view by a factor of 2, retaining the current field center.
- Print : Popup a box to specify the printer, and print a hardcopy of the sky display and legend.
- Read GSC : Read the Guide Star Catalogue to update the object list and display, with the field center specified by the Field RA and Dec values.
- Load from TCS : Grab the current coordinates published by the Telescope Control System GUI, and Read GSC with this position as the field center.
- Recenter at Location : Set the field center to the current location of the focal plane, and Read GSC.
- Quit : Exit GSFind.

4.8.2 Options

- Swap East/West : Flip the display about the N-S axis. This overrides the east_at_left configuration table variable.
- Swap Side of Pier : Rotate the focal plane by 180 degrees with respect to the sky. This option is only available if the is_equatorial variable is set to 1.
- Show Non–stellar Objects : Toggles the display of non–stellar objects in the sky display.
- Show Lines of RA and Dec : Toggles the display of an RA and Dec grid overlay on the sky display.

4.8.3 Help

- About GSFind : Popup display of copyright information and acknowledgements.
- What's New : Popup display of the new features in this release of GSFind.
- Help : Popup display of hints for users.

4.9 Shortcut Buttons

There are 7 buttons below the menubar that provide a single click implementation of a menu item action or option. From left to right these buttons implement Zoom In, Zoom Out, Swap East/West, Print, Read GSC, Load from TCS and Recenter at Location.

5 Using GSFind

5.1 Command Line Options

The command line options presently supported in GSFind are shown in Table 2.

Option	Value	Description
RA DEC Epoch	hh:mm:ss sdd:mm:ss YYYY	Initial position and epoch (Julian Refernce)
–cicada	—	Run in cicada mode
X11 options	—	Standard X11 options – see man X11

Table 2: Summary of command line options for GSFind

5.1.1 CICADA Mode

GSFind will run in cicada mode when the –cicada flag is given on the command line. This is how GSFind is started when it is launched from CICADA. The main reason for doing this is so that users' cicada preferences can be used in GSFind.

5.2 Moving the Focal Plane

In order to align a guide star with a guide detector, the focal plane may need to be moved. This may be done either by using the mouse or the keyboard. To use the mouse, position the cursor over the science detector and depress the left mouse button. Now move the mouse to reposition the focal plane. Release the mouse button when you're done. Alternatively, use the arrow keys to move focal plane up/down and left/right. For finer steps, hold down the "shift" key while using the array keys.

5.3 Moving the Orbital Guider

For orbital guider systems, the guider can be moved around the focal plane. This operation can only be done using the mouse. To reposition the guider with the mouse, move the cursor over the guide detector and depress the right mouse button. As the mouse is moved, the guider will track the position angle of the cursor. Release the mouse button when you're done. The "X offset" and "Y offset" coordinates are updated as the guider is repositioned.

5.4 Rotating the Focal Plane

For rotatable guider systems, the position angle of the focal plane may be changed to reposition the guider. This can be done by entering a new value for the position angle into the "PA" textbox, using the mouse, or by keystrokes. To use the mouse, position the cursor over the guide detector and depress the right mouse button. As the mouse moves, the focal plane will rotate to track the position angle of the cursor. Release the mouse button when your done. The focal plane and sky will then be rotated together to maintain the original orientation of the focal plane. To use the keyboard, press the "<" and ">" keys to rotate the focal plane in 1 degree steps. Hold down the "shift" key to move in 0.1 degree steps. Again, the orientation of the focal plane will be maintained, so the sky plot will be rotated in the display. The "PA" display field will be updated as the focal plane is rotated to reflect the current instrument position angle.

5.5 Zooming

To allow the focal plane or guider to be positioned with greater accuracy, the display may be zoomed. This can be done about the field center using the menu items or shortcut buttons. To zoom in about a different point, position the cursor at that point and click the middle mouse button. This will reduce the field size by a factor of 2, and recenter the display about the location originally pointed to by the cursor. Using the middle mouse button results in cyclic zooming. Once the field size is reduced to one eighth of its original size, the next click of the middle mouse button will return the display to the original field size and center.

5.6 Selecting an Object

An object may be selected to display it's detector coordinates, or to find out more information about the object. This can be done by positioning the cursor over an object in the sky display and pressing the spacebar. The object will then be highlighted in the object list. Alternatively, an object may be selected from the object list by clicking the left mouse button on its list entry. If the selected object is on the sky plot, it will be marked by a flashing red box.

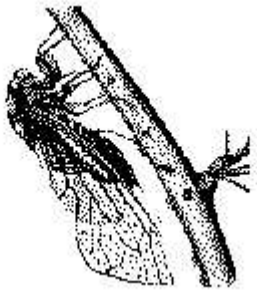
5.7 Finding a Guide Star

To locate a suitable guide star near to your target object, enter your target coordinates into the Field : RA and Dec textboxes, and then Read GSC. Look for a suitable guide star near to one of the guider detectors. Move the focal plane and guider to position the chosen guide star on a guide detector. Now zoom in and make fine adjustments to the position of the guider to center the guide star on the detector. Note the current detector coordinates (These coordinates are also published, and may be read in by the telescope control system).

5.8 Acquiring a Guide Star

The selected objects pixel coordinates are displayed at the bottom of the object list to aid the observer in acquiring a guide star. Should the guide star not fall on the guide detector when they slew to their target coordinates, the observer can take a short exposure of the target to locate a suitable star on the science detector. They can then set the field coordinates in GSFind to the target coordinates, Read GSC, and select the star they identified on from the exposure. The displayed pixel coordinates of the selected star can then be compared to the pixel coordinates from

the data. With knowledge of the pixel scale, the telescope offset required to place the guide star on the guide detector can be trivially calculated.



CICADA and the AAO 6dF V1.0

Computer Section
Mount Stromlo and Siding Spring Observatories
Research School of Astronomy and Astrophysics
Australian National University

December 13, 2000

Contents

Chapter 1

Introduction

This document describes the use of CICADA with the AAO 6dF instrument at the AAO UK Schmidt telescope. It should be read in conjunction with the general CICADA User Manual, the CICADA Configuration Manual and the CICADA Plug-in Users Guide. Material here concentrates on the specific operation of the 6dF instrument from within CICADA.

Chapter 2

CICADA Customisations for 6dF

CICADA has been customised for 6dF in the following ways:

- *A 6dF instrument configuration plug-in added*
- *An interface to the UK Schmidt TEL_CONTROL built as a TCS plug-in*
- *A FITS plug-in written to insert 6dF BINTABLE extensions describing the 6dF instrument configuration*

These are described more fully in following sections. A description of how CICADA is configured for 6dF is also included.

2.1 6dF Instrument Configuration

Table 2.1 shows the FITS headers saved in each 6dF FITS file that represent the instrument configuration when the image was taken. The values for these parameters need to be entered by the observer before an exposure is taken. Care has to be taken that these values are correct because there is **NO** automatic checking that can be done.

Table 2.1: 6dF Configuration FITS Keywords

Keyword	Type	Example Value	Description
GRATID	String	'250B'	Name of grating
GRATSET	Float		Micrometer setting (micrometers)
GRATSLOT	String	'A'	Grating slot
GRATBLAZ	String	'COLLIMATOR'	Blaze direction
LAMPNAME	String	'RUBIDIUM'	Lamp name
SOURCE	String	'Plate_1'	Source plate
FOCUS	Float	-2.65	6dF Spectrogrpah focus encoder value

The observer enters the parameters by first opening the *Site: AAO 6dF* panel found under the *Instrument Controls* menu of the CICADA observing

window (see Figure2.1), and then selecting values as appropriate. Once the configuration is correct click the *Record Setup* button to ensure the next FITS file will include desired values. See Figure 2.2.

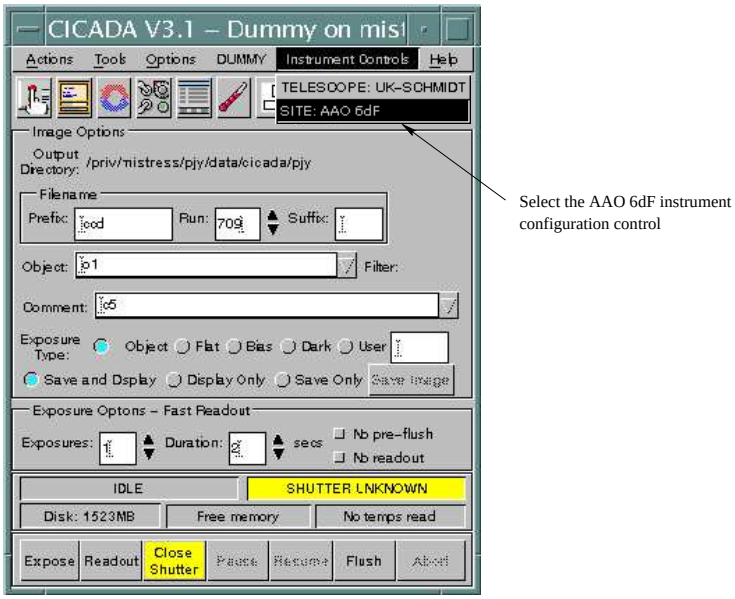


Figure 2.1: Selecting the AAO 6dF configuration window

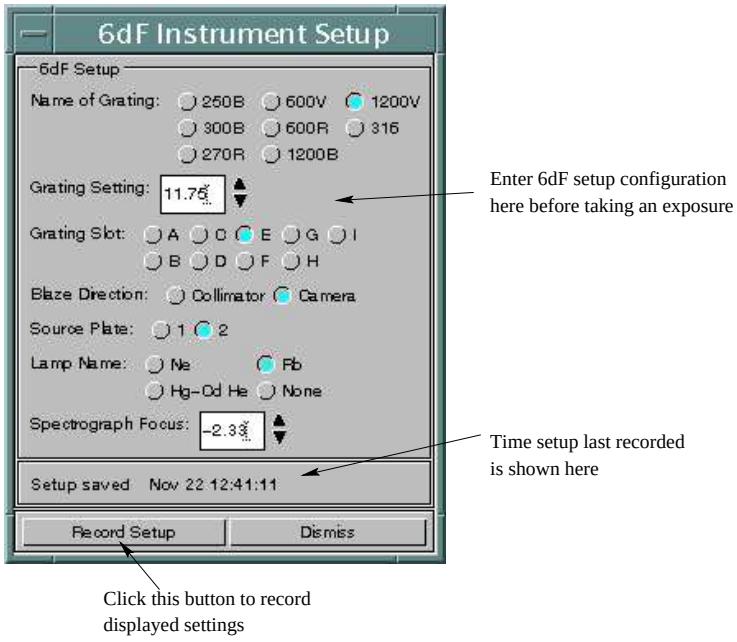


Figure 2.2: AAO 6dF configuration window

Settings are saved in a configuration file on disk where they can be picked

up by the next run of CICADA. Table 2.2 shows example contents of this file.

```
#=======================================================================
# Configuration Table:
# This table has been automatically generated and consists of a series of
# named items , eg [name], followed by a set of parameter and value
# pairs, which could be multiple per line.
# Default valued items might not appear. Take care if editing by hand.
#=======================================================================

6df.blaze:CAMERA
6df.focus:-2.320000
6df.grating:1200V
6df.lamp:RUBIDIUM
6df.plate:Plate_2
6df.setting:11.750000
6df.slot:E
```

Table 2.2: AAO 6dF Example Configuration Data

2.2 The UK Schmidt TCS Interface

CICADA interfaces to the local TCS using a site-specific plug-in module. Refer to the CICADA Plug-in Users Guide for full detail on how these plug-ins are written and implemented. The UK Schmidt plug-in accesses TCS information using the AAO *TEL_CONTROL* specification with a socket connection to the *TEL_CONTROL* server.

For each exposure the plug-in sends the *WHERE* command (for telescope information), awaits a response from the *TEL_CONTROL* server, then sends a *FOCUS_GET* command to get the telescope focus and finally sends the *ROTATOR* command to get the current rotator angle. The *WHERE* command should return the mean RA, DEC, equinox, apparent hour angle, airmass of the telescope, UT and ST from the telescope master clock and the zenith distance.

These data are added to the standard set of CICADA telescope keywords for insertion in the resultant FITS file. In addition the plug-in also writes the following *extra keywords* - see Table 2.2. Some of these keywords are just duplicates of the standard CICADA keywords but included with preferred AAO formatting and name.

Table 2.3: 6dF Extra Telescope FITS Keywords

Keyword	Type	Example Value	Description
ORIGIN	String	'AAO'	UK-Schmidt 1.2m AAO
UTDATE	String	'04-12-00'	AAO UTDATE keyword
UTSTART	String	'23:17:28.0'	AAO UTSTART keyword
ZDSTART	Float	7.647585e+01	AAO ZDSTART keyword (degrees)
STSTART	String	'01:23:51.0'	AAO STSTART keyword
HASTART	String	'08:13:38.21'	AAO HASTART keyword

Table 2.3: 6dF Extra Telescope FITS Keywords

Keyword	Type	Example Value	Description
ALT_OBS	Float	1130	AAO ALT_OBS keyword (meters)
LAT_OBS	Float	-31.27333	AAO LAT_OBS keyword (meters)
LONG_OBS	Float	149.0700	AAO LONG_OBS keyword (meters)
MEANRA	Float	1.75400e+01	AAO MEANRA keyword (degrees)
MEANDEC	Float	-5.64564e+01	AAO MEANDEC keyword (degrees)
ROTST	Float	1.23456e+01	Rotator value at start of exposure
ROTEND	Float	2.34567e+01	Rotator value at end of exposure

The TCS plug-in is configured to use a socket interface from a configured *TEL_CONTROL* server. See Section 3.6 for details on how to configure the plug-in.

2.3 The FITS Plug-in for Adding 6dF BINTABLE Extensions

In addition to the FITS information added using the configuration plug-in and the TCS interface, CICADA also adds information about the 6dF instrument contained in FITS binary tables. This data includes the fibre positioning information used for the exposure and also a file containing the telescope temperature.

CICADA calls the FITS plug-in after closing the FITS image file at the end of an exposure. The plug-in rewrites the file with the extra data added. The extra binary table is read in from disk files described in the FITS plug-in configuration file `/opt/cicada/config/AAO_6dF_fits.config`. This configuration file must be world-readable and contains parameter value pairs as shown if table 2.4.

```
# AAO 6dF FITS plug-in configuration table
plate1_file:/opt/cicada/config/AAO_6dF_plate1.fits
plate2_file:/opt/cicada/config/AAO_6dF_plate2.fits
teltemp_file:/opt/cicada/config/AAO_6dF_teltemp.dat
teltemp_timeout:5
```

Table 2.4: AAO 6dF FITS Plug-in Configuration Data

The fibre positioning data files are expected to be in FITS binary table format, CICADA will just ignore the primary header unit and copy the BINTABLE extension into the new multi-extension FITS (MEF) image file.

Chapter 3

Configuring CICADA for 6dF

Refer to the CICADA Configuration Manual for full detail on CICADA hardware configuration. This section concentrates on the configuration settings for the AAO 6dF instrument. Use the CICADA configuration GUI for building the 6dF setup.

3.1 6dF Instrument

At the top of the configuration hierarchy is the instrument table, and for 6dF this looks like Figure 3.1.

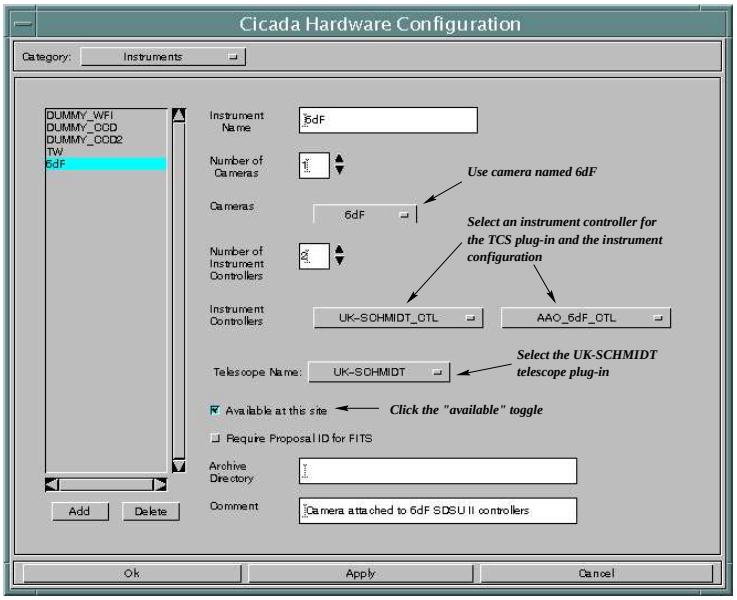


Figure 3.1: CICADA 6dF Instrument Configuration GUI

3.2 6dF Camera

The camera is then configured as shown in Figure 3.2. Note that is it here where the FITS plug-in is specified. Also note, that CICADA requires that FITS file header size be pre-configured - it is done here by specifying the number of 36 keyword blocks to reserve. Initially 6dF FITS files will be single extension FITS images but will grow to multi-extension files after the FITS plug-in rewrites them with the extra data.

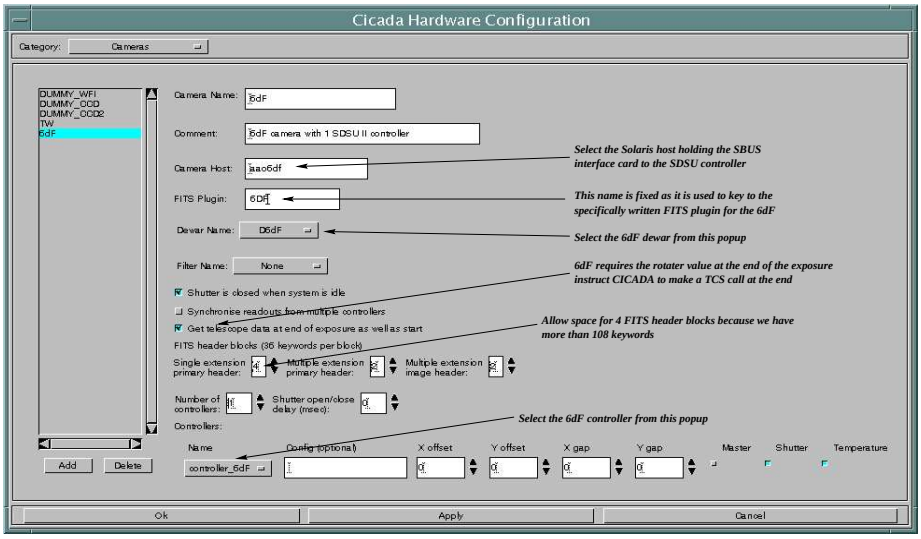


Figure 3.2: CICADA 6dF Camera Configuration

3.3 6dF CCD Controller

Nothing specific to 6dF to note here - make sure the SBUS device address is correctly specified. See Figure 3.3

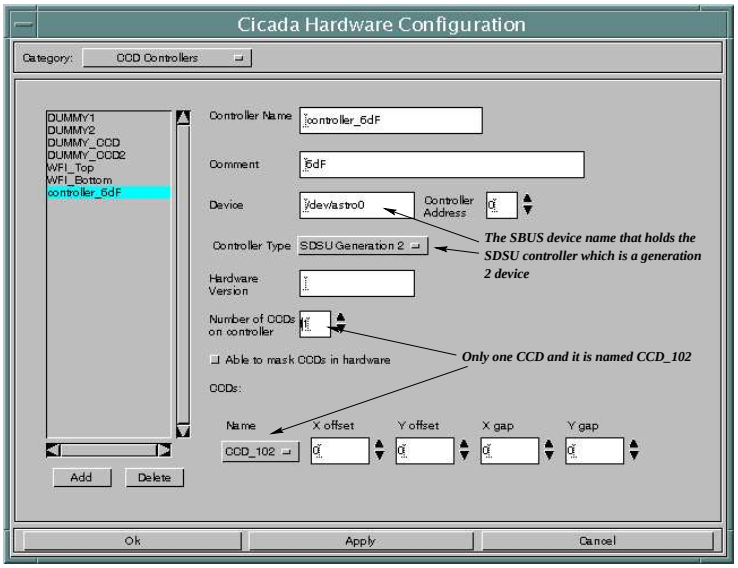


Figure 3.3: CICADA 6dF CCD Controller Configuration

3.4 6dF CCD

Figure 3.4 shows the 6dF CCD configuration for CICADA. It has one readout amplifier configured and is at the bottom left corner of the CCD.

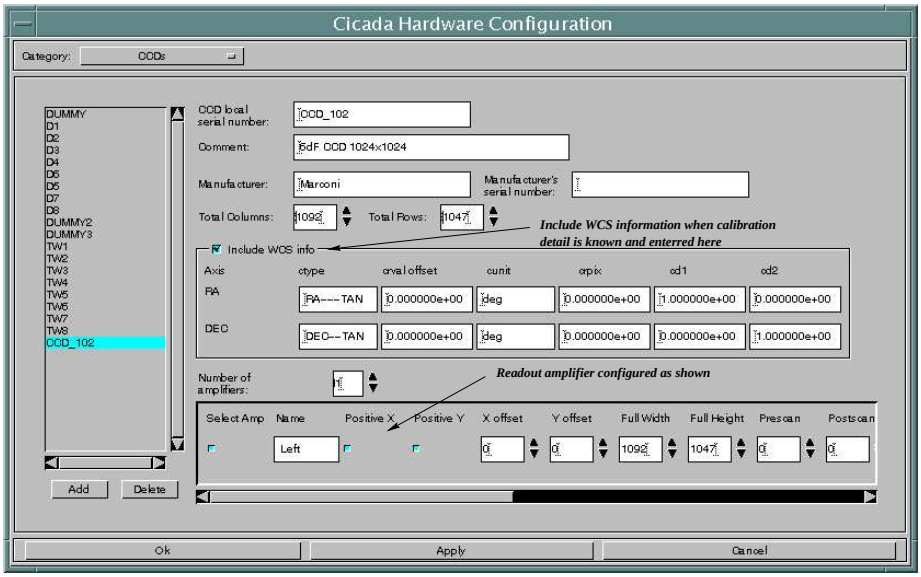


Figure 3.4: CICADA 6dF CCD Configuration

3.5 6dF Focalplane

The focalplane configuration window is used to enter details of camera temperature control. A full description of how CICADA/SDSU performs temperature control can be found at the RSAA's Detector Lab Detector Lab[1] web page. Figure 3.5 shows the settings for the 6dF dewar.

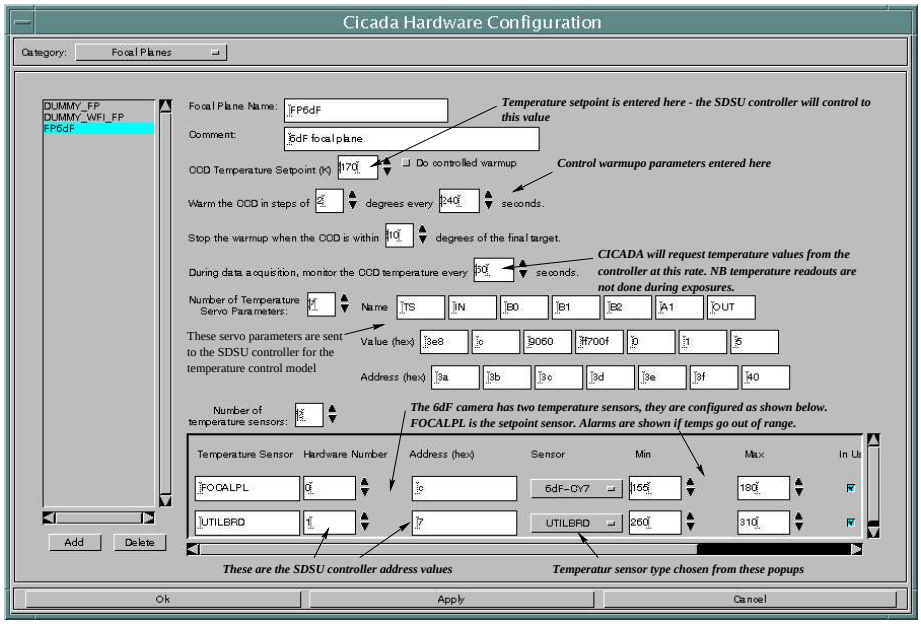


Figure 3.5: CICADA 6dF Focalplane Configuration

[1] http://www.mso.anu.edu.au/observing/detlab/ccdlab/ccdcontroller/sdsu1/temperature/temperature.htm

3.6 UK Schmidt Telescope Configuration

Figure 3.6 shows the GUI for setting the details of the interface to the UK Schmidt TCS. The interface is via a network socket to the 6dF Wincenter NT machine. Enter the socket number and server hostname in this window. The CICADA control host should be the same host from which the CICADA observing interface is run.

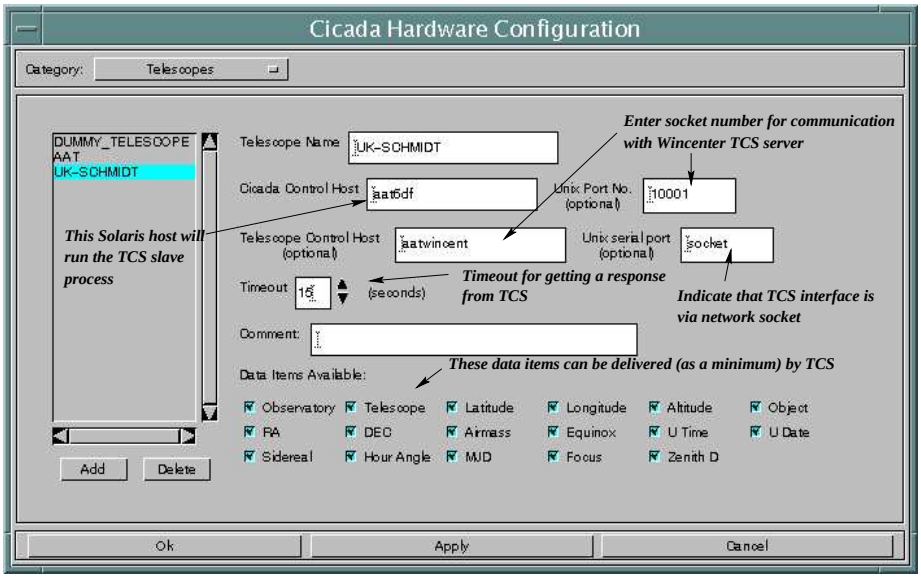


Figure 3.6: CICADA UK Schmidt Telescope Configuration

3.7 Instrument Controllers and the 6dF Site Instrument

The 6dF instrument requires at least one instrument controller configured. As described in Section 2.1, 6dF configuration is entered by the observer in the 6dF instrument control window. This is added to CICADA using a combination of the *Instrument Control*, *Site Instrument* and *Instrument* configuration tables. By adding an instrument control CICADA will know to add a controlling menu item to the main observing GUI, allowing the user to interface to the 6dF configuration plug-in. Figures 3.7 and 3.8 show the required settings for 6dF.

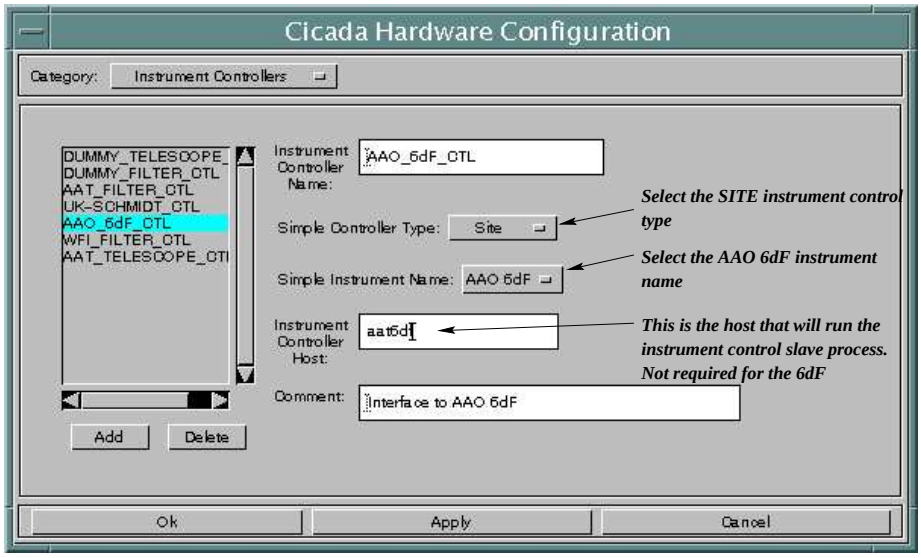


Figure 3.7: CICADA 6dF Settings Instrument Control Configuration

3.8 Temperature Sensor Types

The 6dF dewar has two temperature sensors. One is for measuring the temperature of the focalplane and the other for the control electronics. This configuration table allows CICADA to understand how the temperature sensors operate. See Figure 3.9.

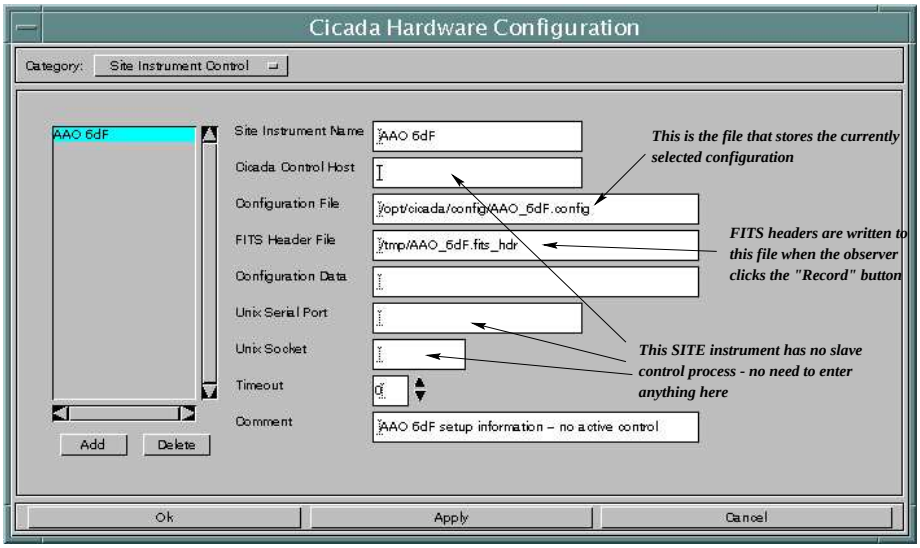


Figure 3.8: CICADA 6dF Site Instrument Configuration

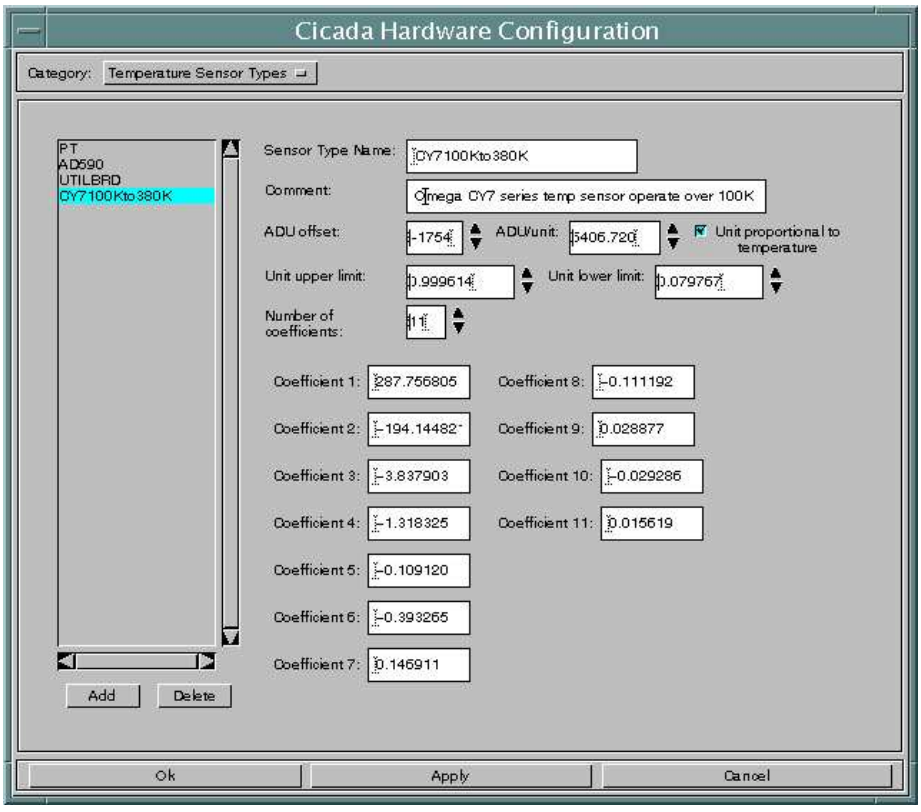
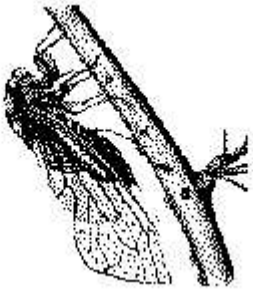


Figure 3.9: CICADA 6dF CCD Temperature Sensor



PIPELINE Manual V1.0

Computer Section
Mount Stromlo and Siding Spring Observatories
Research School of Astronomy and Astrophysics
Australian National University

December 13, 2000

Contents

List of Figures

List of Tables

1 Introduction and Context

This manual describes the operation of a pipeline tool for automated calibration of image data and accompanies the release 1.0 of Pipeline. The aim of this pipeline tool is to provide a more streamlined means of completing the calibration processing steps required before CCD images can be used for other measurement purposes.

This pipeline tool[1] was originally developed for the Wide Field Imager (WFI)[2] being developed jointly by MSSSO, AAO and the University of Melbourne, but it can be used to process other CCD image data. It has been developed in a general manner so that other processing tasks contained in IRAF[3] can be added. One might envisage adding facilities for calibration processing of 1D spectra for example. While it has been developed to run in conjunction with CICADA (Configurable Instrument Control and Data Acquisition), software developed at MSSSO, it may also be used offline in a stand alone mode. Figure 1 shows how this tool fits into the overall process.

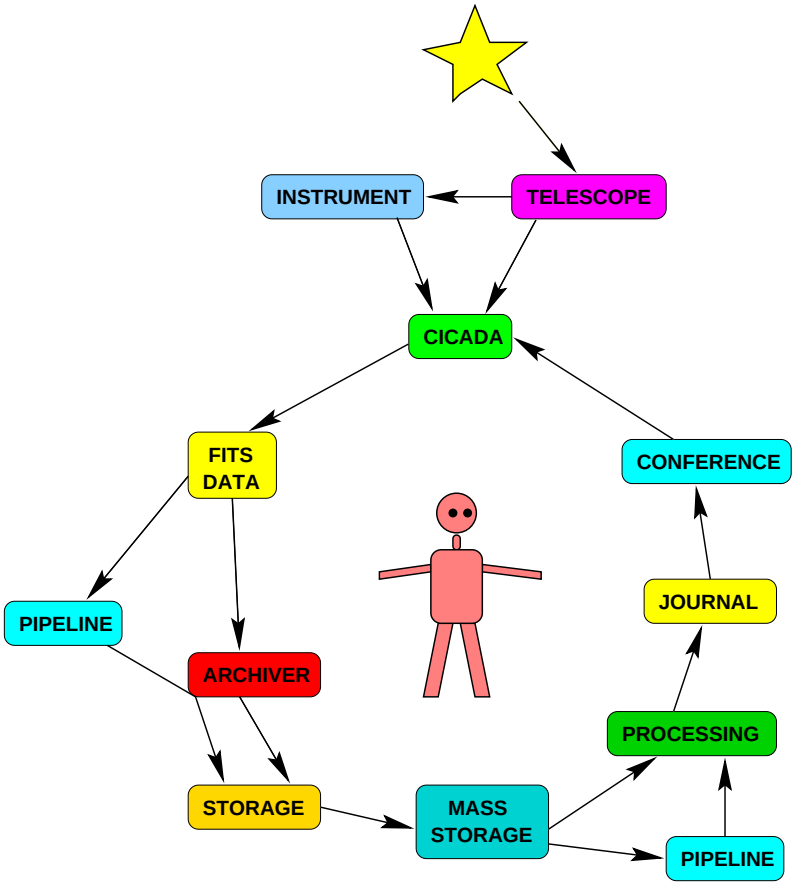


Figure 1: How Pipeline fits into the general scheme

[1] http://msowww.anu.edu.au/computing/cicada
[2] http://msowww.anu.edu.au/observing/wfi/
[3] http://iraf.noao.edu/

2 Using Pipeline

2.1 Command Line Options

The command line options presently supported in Pipeline are shown in Table 1.

Option	Value	Description
–cicada	—	Run in cicada mode
–imagedir	a directory	Directory in which Pipeline will look for raw data
–fileprefix	ccd	Wildcard for selecting files to process (eg. ccd*.fits)
X11 options	—	Standard X11 options – see man X11
–debug	n	Level of debugging output - not implemented

Table 1: Summary of command line options for Pipeline

2.1.1 CICADA Mode

Pipeline will run in cicada mode when the –cicada flag is given on the command line. This is how Pipeline is started when it launched from CICADA. The main reason for doing this is so that users' cicada preferences can be used in Pipeline. The preferred data directory is an example of one such preference.

2.2 Quick Start Guide

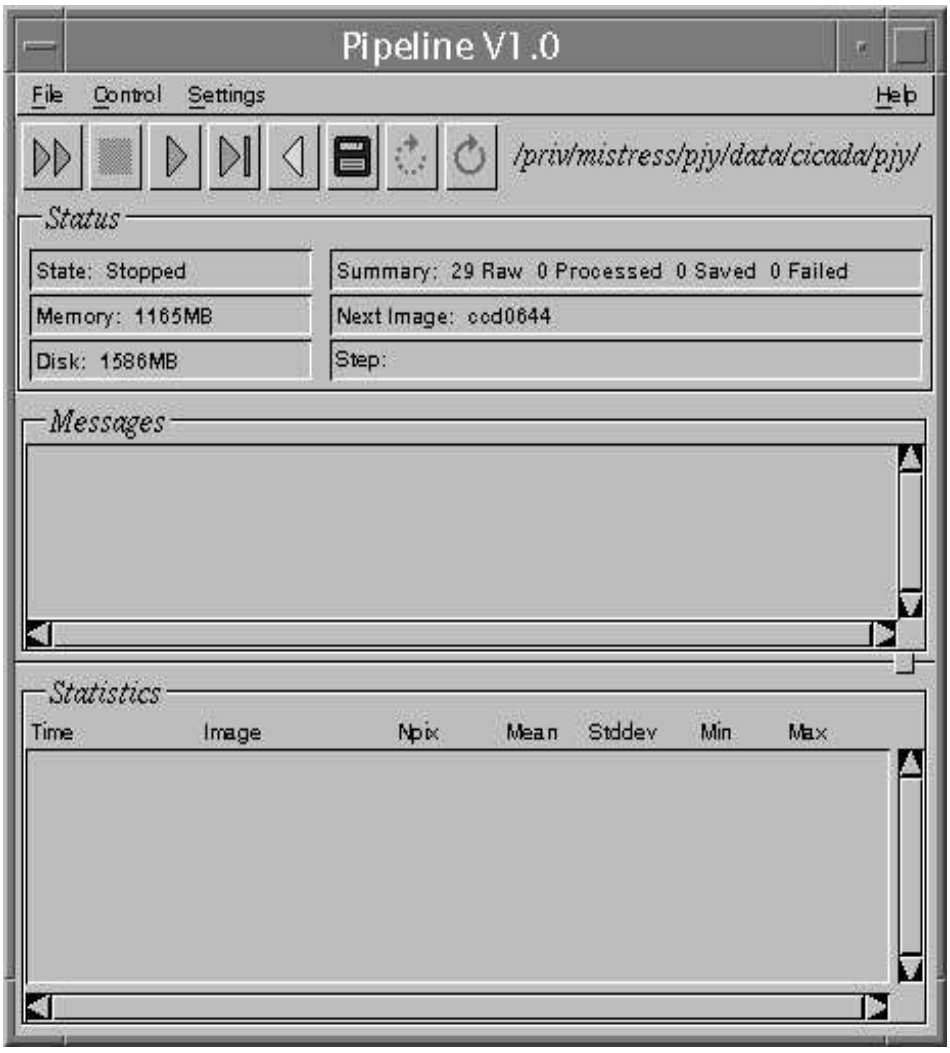


Figure 2: Pipeline GUI Window

1. Start Pipeline by typing `pipeline`. You should see the main GUI window shown in Figure 2
2. Choose *Preferences...* from the *Settings* Menu. You should see the preferences window appear - Figure 3
3. In the bottom panel, select whether you wish to have monitoring statistics and/or images displayed
4. Select other preferences as desired
5. Choose *Input Data...* from the *Settings* Menu
6. Set up your *Data Directory, Files to Process* in the bottom frame
7. Choose *Processing Steps...* from the *Settings* Menu
8. Select your calibration steps and images
9. Press the *Start Pipeline Running* button on the toolbar

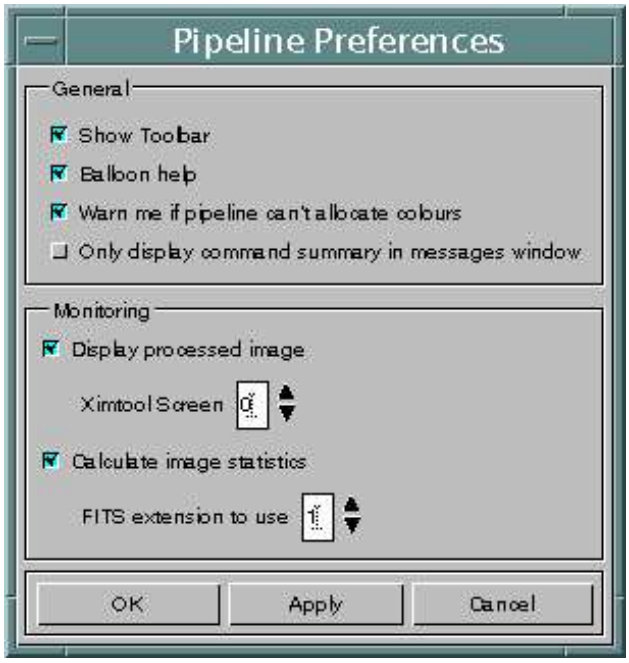


Figure 3: Pipeline Preferences Window

2.3 Files and Their Locations

All the files used or created by Pipeline when it is running are shown in Figure 4 and described in Table 2. The raw data are assumed to be in the user's preferred data directory and the processed data are deposited in a subdirectory called pipeline.

The Pipeline logs files (pipeline.981225.log and similar names) are most likely to be in /tmp or /opt/cicada/logs, depending on the configuration specified in /opt/cicada/pipeline_table.

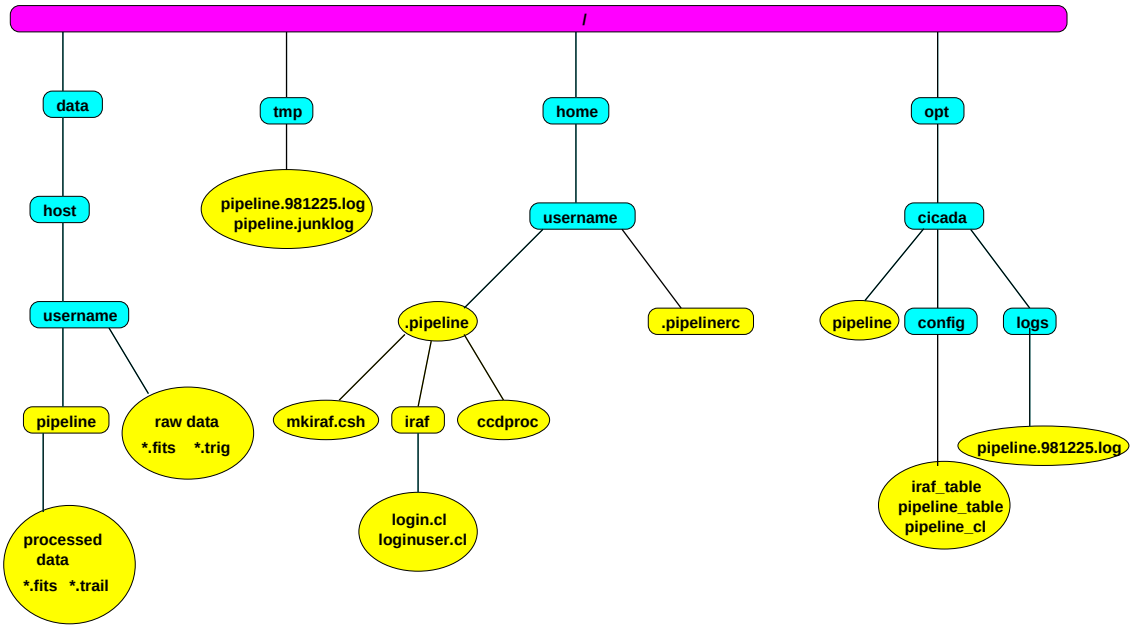


Figure 4: Locations of files used or created by Pipeline. Boxes indicate directories and ovals indicate files.

File Locations	Description
/image_dir/*.fits	raw data
/image_dir/.*.trig	trigger files to initiate processing
/image_dir/pipeline/*.fits	processed data
/image_dir/pipeline/.*.trail	log of IRAF message from processing
/tmp/pipeline.junklog	log of IRAF message from processing
/tmp/pipeline.981225.log	log of pipeline events
/home/username/.pipelinerc	user specific pipeline settings
/home/username/.pipeline/mkiraf.csh	script to initialise IRAF setup
/home/username/.pipeline/iraf/login.cl	IRAF command language startup script
/home/username/.pipeline/iraf/loginuser.cl	user specific IRAF start up items
/opt/cicada/pipeline	pipeline executable
/opt/cicada/config/pipeline_table	site specific pipeline configuration
/opt/cicada/config/iraf_table	site specific pipeline IRAF configuration
/opt/cicada/logs/pipeline.981225.log	log of pipeline events

Table 2: Description of files used and created by Pipeline.

2.4 Calibration Images

2.4.1 Preparing Calibration Images

At present Pipeline assumes that all calibration images have had all the prior processing that they may require completed. For example the flat fields should have been overscan corrected, bias subtracted, dark subtracted, trimmed and combined before being used in the pipeline. If you are using library flat fields (as may be the case for WFI) this will have already been done. Otherwise, you have to do the preprocessing yourself. In future you will be able to use Pipeline to do this, once some of the issues described in section 7.2 are resolved.

2.5 Processing Steps

The processing sequence that are presently implemented are shown in Figure 5, starting at the top and flowing down. The right most box describes the steps involved. Following the successful processing of each image, the image may be displayed, along with image statistics, so the user can quickly verify that the processing is proceeding as intended. At present, due to the complicated nature of preparing flat field images for WFI and other wide field CCD mosaics, it is assumed that the calibration images have been pre-prepared (or obtained from a library) and the path on the right is followed for images containing science data. When the formation of flat fields becomes a more reliable process, it may be possible to automate the pipelining of all the steps shown above as IRAF already provide some support for this.

Step 1	All images have their overscan subtracted
Step 2	Overscan regions are trimmed off all images
Step 3	Zero images are combine (median filtered)
Step 4	Combined zero image is subtracted from Object, Darks and Flats
Step 5	Dark images are combined (median filtered)
Step 6	Combined dark image is subtracted from Object, Flats
Step 7	Flat images are combined (median filtered)
Step 8	Combined flat image is subtracted from Object
Step 9	Interpolation across bad pixels
Step 10	Any other corrections can now be applied

Table 3: Summary of processing steps (shown in Figure 5) involved in calibrating ccd images

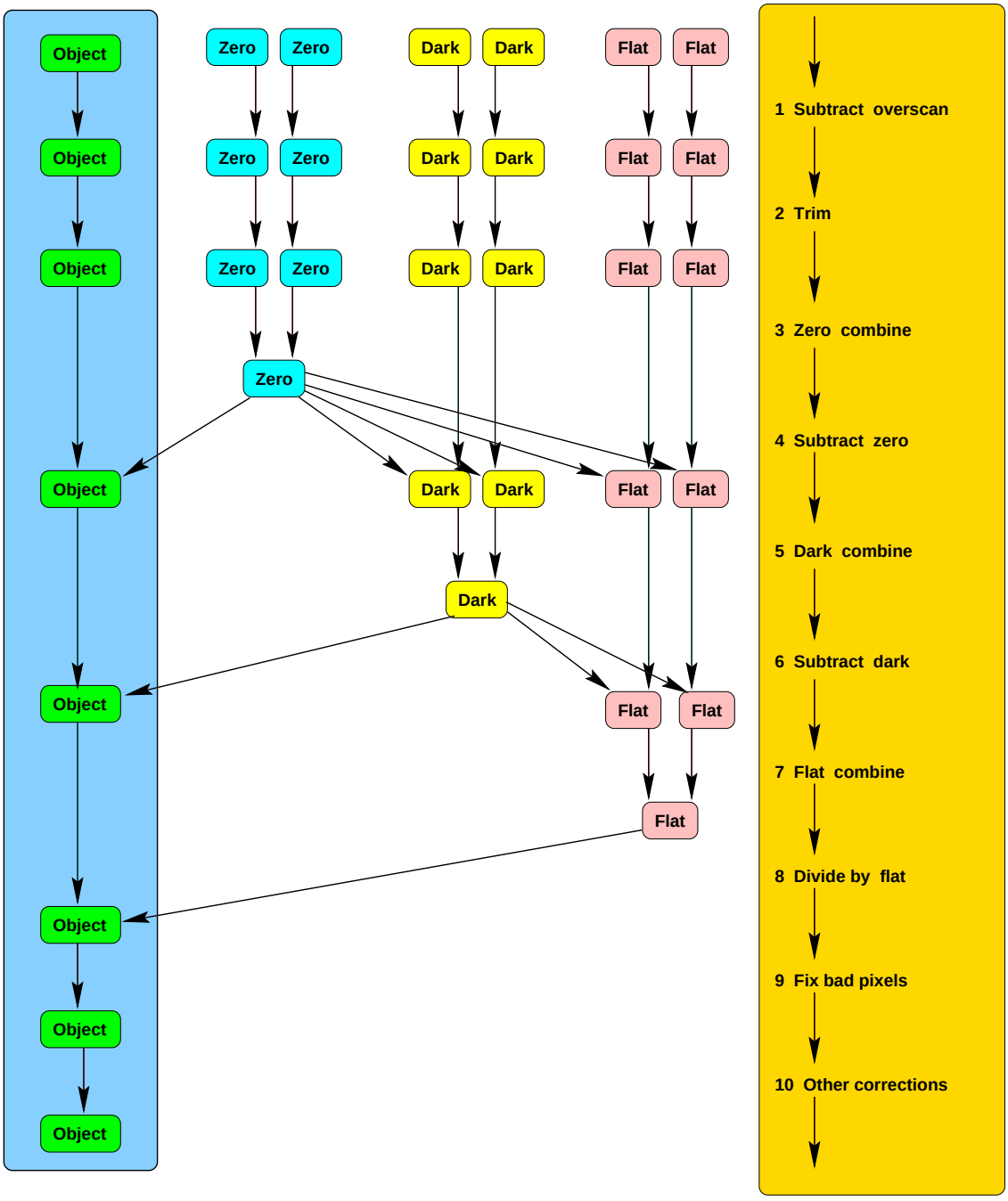


Figure 5: Outline of the processing steps (summarised in Table 3) involved in calibrating ccd images.

2.6 Input Data

Input Data may be set up using the popup shown in Figure 6.

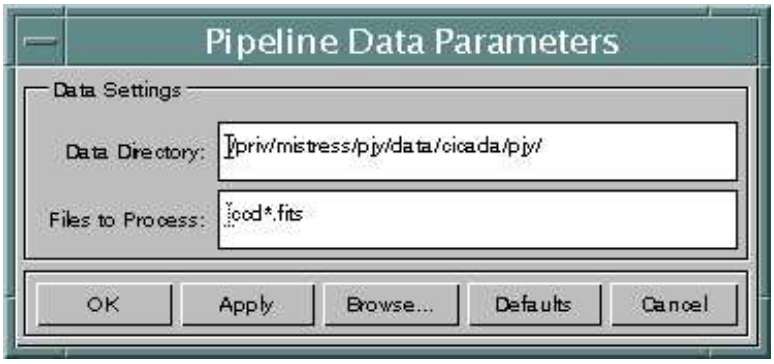


Figure 6: Popup window for setting data paths and parameters

2.7 Processing Parameters

Processing parameter may be edited using the popup shown in Figure 7.

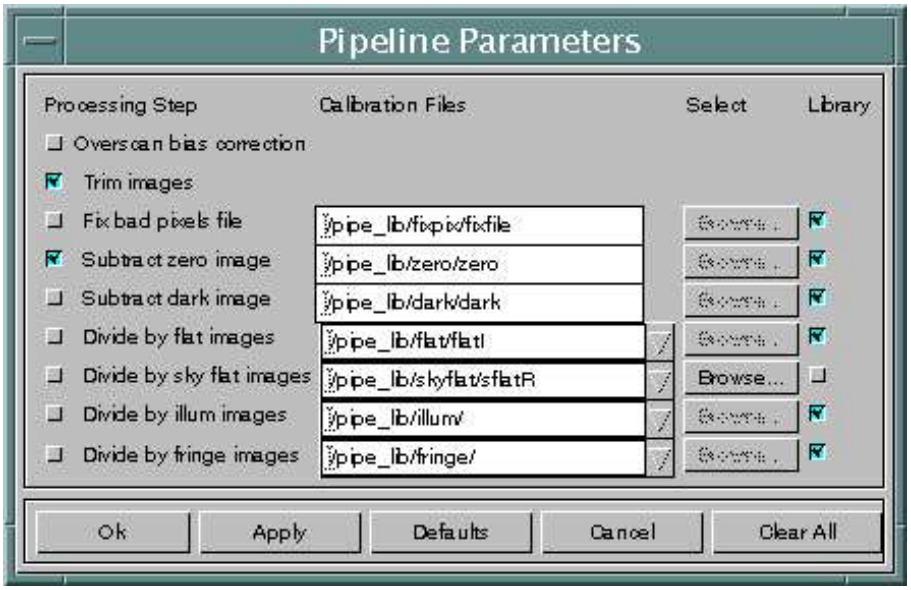


Figure 7: Popup window for editing processing parameters

2.8 Using the Image Chooser

Directories, files and images may be selected using the image chooser (file picker) shown in Figure 8.

2.9 Controlling Processing

Processing is controlled using the *Control* menu or by using the toolbar icons. Pipeline can either be run in normal continuous mode, that is, it will wait on the arrival of FITS files and automatically process all required steps to completion, or can operate in single-step mode where each of the processing steps will be performed individually. Single-step mode allows close inspection of each processing step and might be used when first setting up Pipeline for a batch of processing.

The *Status* panel of the main GUI indicates Pipeline's current activity. It shows Pipeline state, resource availability, summary of processing and the next image and/or next step to process.

Figure 8: Popup window for selecting images or files

2.9.1 Start Pipeline Running

The *Start Pipeline Running* button or *Run* menu item will cause processing to be launched according to any settings applied in the Pipeline GUI. If you apply any new setting you will need to stop and restart Pipeline for them to take effect.

2.9.2 Stop

Pressing this button will terminate processing of data as soon as the image that is currently being processed is completed. Pressing the *Run* button will resume processing at the next step unless a reset operation has been performed.

2.9.3 Run Single File

Pipeline can be set to process just the next image displayed in the *Status* panel. This mode can be useful when closer monitoring of processing is required.

2.9.4 Run Single Step

Similarly, just the next processing step as indicated in the *Status* panel can be performed.

2.9.5 Undo Processing for Last File

This option allows the most recent processing to be backed up so that it can be repeated. This applies to just the last file. Continual activation of this button will undo processing on each prior file.

2.9.6 Save Current Progress

As processing proceeds Pipeline stores flag files in the data directory so that it can keep track of its progress. Pressing this button will protect currently processed images from reset operations. This is useful for when processing has completed satisfactorily for a set of files and a checkpoint is desired.

2.9.7 Resetting Unsaved for a Restart

Pressing this button will clear out any unsaved processed files in the data directory, allowing them to be reprocessed.

2.9.8 Resetting All for a Restart

Pressing this button will clear out all processed files in the data directory, allowing all data to be reprocessed.

2.10 Monitoring Processing

There are several ways in which the processing status can be monitored.

1. In the *Status* panel of the main Pipeline GUI, the number of raw, processed, saved and failed images are summarised.
2. The message window on the main Pipeline GUI receives details of processing steps that have occured. These messages can have full detail of the parameters passed to the IRAF ccdproc task or just simple summary information. Control of this is via the *Settings/Preferences* dialog.
3. Statistics of the processed images are available by turning on the toggle button on the *Preferences* popup which is started from the *Settings* menu.
4. Similarly, an Ximtool display of the processed images is available by turning on the toggle button on the *Preferences* popup. The Ximtool can be restarted from the *File* menu.

2.11 Configuration and Settings

2.11.1 Configuration

Pipeline has three configuration files (/opt/cicada/config/pipeline_table, /opt/cicada/config/pipeline_cl and /opt/cicada/config/iraf_table) in which site specific settings are stored.

As shown in Table 4 the pipeline table contains settings for things such as the directories in which the library calibration files are stored, the path to IRAF executables and the default directory in which Pipeline log files are written. This configuration file would normally be set up by the person who installs Pipeline and would be readable, but not writable by Pipeline users.

The IRAF startup file `pipeline_cl` is used for inserting commands required to startup IRAF at the local site. These should be Bourne shell commands. An example file for IRAF V2.11.3 follows:

The IRAF configuration table (Table 6) contains IRAF package (eg CCDRED, MSCRED) processing parameters. These should be setup by the administrator of Pipeline at each site. It is not intended that they be edited by Pipeline users, as this would complicate the original aim of simplicity in the design of Pipeline.

2.11.2 Using Non-Library Configuration Files

The default configuration settings can be overridden by specifying alternate configuration files using the *Pipeline Configuration...* option from the *Settings* menu. Simply select directories for locating configuration files using the dialog pictured in Figure 9.

```
#=========================================================================
# Configuration Table: pipeline_table
# This table has been automatically generated and consists of a series of
# named items , eg [name], followed by a set of parameter and value
# pairs, which could be multiple per line.
# Default valued items might not appear. Take care if editing by hand.
#=========================================================================
library.darkimage.dir:/pipe_lib/dark/
library.fixfile.dir:/pipe_lib/fixpix/
library.flatimage.dir:/pipe_lib/flat/
library.fringeimage.dir:/pipe_lib/fringe/
library.illumimage.dir:/pipe_lib/illum/
library.instrument:/opt/cicada/config/keyword_translation_table
library.skyflatimage.dir:/pipe_lib/skyflat/
library.zeroimage.dir:/pipe_lib/zero/
pipeline.fifodir:/opt/cicada/pipeline_dir/
pipeline.irafroot:/iraf/iraf/
pipeline.logfiledir:/tmp/
pipeline.waiting_time:30
```

Table 4: An example configuration file for Pipeline.

```
# IRAF cl startup commands
# Add bourne shell commands that are needed to run IRAF
# at site using pipeline
#

# first set some env vars
IRAFARCH=ssun
export IRAFARCH
arch=.$IRAFARCH
export arch
IRAFBIN=${iraf}bin$arch
export IRAFBIN

# Now run IRAF
${IRAFBIN}/cl.e
```

Table 5: An example Pipeline configuration file for starting IRAF.

```
#=======================================================================
# Configuration Table: IRAF Table
# This table has been automatically generated and consists of a series of
# named items , eg [name], followed by a set of parameter and value
# pairs, which could be multiple per line.
# Default valued items might not appear. Take care if editing by hand.
#=======================================================================
[mscred]
  pixeltype = "real real"; verbose = yes; logfile  = logfile; plotfile = plotfile;
  backup = none; bkuproot = ""; ampfile = amps; ssfile = subsets; im_bufsize = 2.0;
  graphics = stdgraph; cursor = ""; mode = ql;
[mscproc]
  ccdtype = object; noproc = no; minreplace = 1.0; interactive = no;
  function = legendre;  order = 1; sample = *; naverage = 1; niterate = 1;
  low_reject = 3.0; high_reject = 3.0; grow = 0.0; fd = ""; merge = no; fd2 = "";
  mode = ql;
[imred]
  keeplog = no; logfile = imred.log; mode = ql;
[ccdred]
  pixeltype = short; verbose = yes; logfile = logfile; plotfile = plotfile;
  backup = none; ssfile = subset; graphics = stdgraph; cursor = ""; lpar = no;
  mode = ql;
[ccdproc]
  ccdtype = object; max_cache=2.0; noproc = no; minreplace = 1.0;
  interactive = no; function = spline1; order = 200; sample = *; naverage = 1;
  niterate = 1; low_reject = 3.0; high_reject = 3.0; grow = 1.0; mode = ql;
```

Table 6: An example Pipeline configuration file with IRAF task parameters.

2.11.3 User Settings

User specific settings are stored in a file - usually /home/username/.pipelinerc. An example is shown in Table 7. This file may be edited by Pipeline users, so long as care is taken to preserve the correct format of the information contained in it.

2.11.4 Fonts and Colours

Setting up fonts and colours for Xwindows applications can be a complicated process. X applications use "resources" to determine the fonts and colours they should use when displayed on the screen. If you wish to change the default colours and fonts for Pipeline, you will need to add the following lines to your `.Xdefaults` file:

```
pipeline*fontList: -*-lucida sans-medium-r-*-*-*-100-*-*-*-*-*-*=TAG1,
                   -*-lucida sans-bold-r-*-*-*-110-*-*-*-*-*-*=TAG2
pipeline*background: gray
pipeline*foreground: black
pipeline*selectColor: cyan
```

To select a font you like, use the program `xfontsel` which allows you to preview various fonts and then cut and paste your chosen font descriptor into your .Xdefaults file. Note that there are two fonts listed in the `fontlist` resource. Pipeline uses the first font for most of the labels in all windows. The second font is used to highlight various buttons and text fields such as the status indicators. You do not need to specify a second font as Pipeline will use the first font you specify for all labels and text if no second font is in the fontlist. Note the syntax of the fontlist resource as the `=TAG1` and `=TAG2` are very important.

The `pipeline*background` resource sets the colour of all window, button, etc. backgrounds. Text on buttons and labels will be in the colour specified by the `pipeline*foreground` resource. `pipeline*selectColor` sets the colour used by toggle buttons when they are turned "on".

```
#========================================================================
# Configuration Table: .pipelinerc
# This table has been automatically generated and consists of a series of
# named items , eg [name], followed by a set of parameter and value
# pairs, which could be multiple per line.
# Default valued items might not appear. Take care if editing by hand.
#========================================================================
ccdproc.darkcor:0
ccdproc.fixpix:0
ccdproc.flatcor:0
ccdproc.fringecor:0
ccdproc.illumcor:0
ccdproc.instrument:/opt/cicada/config/keyword_translation_table
ccdproc.overscan:0
ccdproc.readcor:0
ccdproc.scancor:0
ccdproc.trim:1
ccdproc.zerocor:1
combine.combine_algorithm:median
combine.input_images:
combine.output_image:zero
combine.reject_algorithm:minmax
image_display.ximtool_height:564
image_display.ximtool_screen:0
image_display.ximtool_width:512
image_display.ximtool_x:564
image_display.ximtool_y:310
library.darkimage:1
library.fixfile:1
library.flatimages:1
library.fringeimages:1
library.illumimages:1
library.sflatimages:0
library.zeroimage:1
monitoring.fits_ext:1
monitoring.monitor_disp:0
monitoring.monitor_stats:1
mscproc.darkcor:0
mscproc.fixpix:0
mscproc.flatcor:0
mscproc.fringecor:0
mscproc.illumcor:0
mscproc.instrument:/opt/cicada/config/keyword_translation_table
mscproc.overscan:0
mscproc.readcor:0
mscproc.scancor:0
mscproc.sflatcor:0
mscproc.trim:1
mscproc.zerocor:1
pipeline.cllog:/tmp/pipeline_102.junklog
pipeline.fifodir:/opt/cicada/pipeline_dir/
pipeline.filesto_proc:ccd*.fits
pipeline.irafroot:/iraf/iraf/
pipeline.logfiledir:/opt/cicada/logs/
pipeline.workdir:/data/monsoon1/cicada/user
preferences.showballoonhelp:1
preferences.showcolourmsgs:1
preferences.showsummarycmd:0
preferences.showtoolbar:1
user_calib.darkimage:/pipe_lib/dark/dark
user_calib.fixfile:/pipe_lib/fixpix/fixfile
user_calib.flatimages:/pipe_lib/flat/flatB,/pipe_lib/flat/flatV,/pipe_lib/flat/flatI
user_calib.fringeimages:/pipe_lib/fringe/
user_calib.illumimages:/pipe_lib/illum/
user_calib.sflatimages:/pipe_lib/skyflat/sflatB,/pipe_lib/skyflat/sflatV,/pipe_lib/skyflat/sflatI
user_calib.zeroimage:/pipe_lib/zero/zero
```

Table 7: An example .pipelinerc file

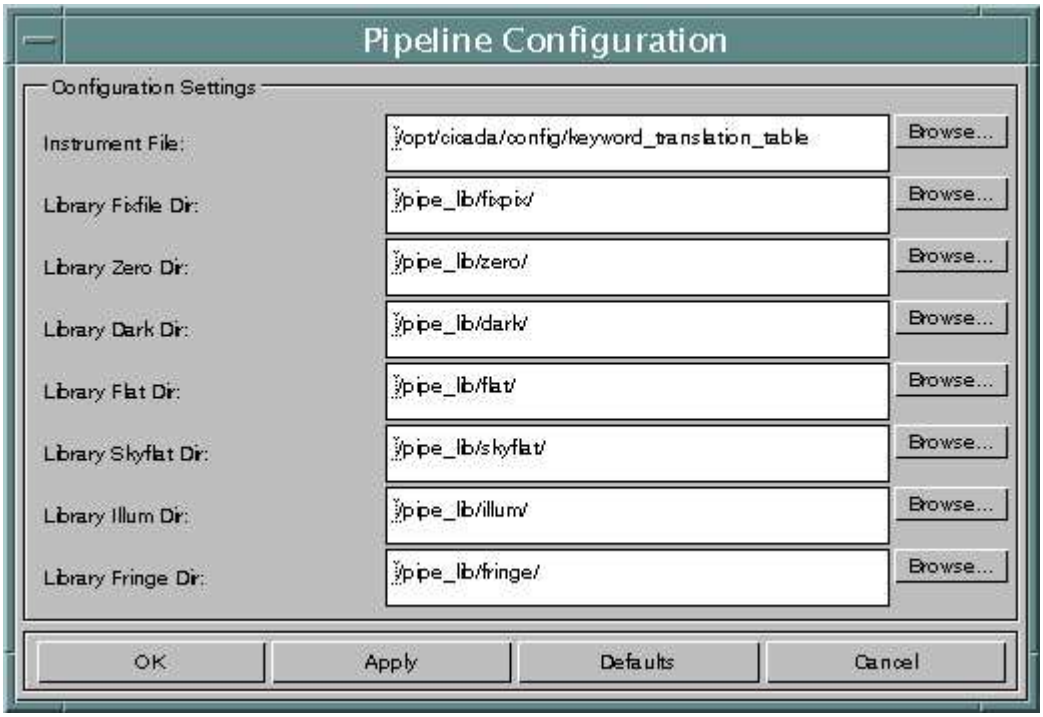


Figure 9: Popup window for selecting configuration directories

3 Troubleshooting

When I run pipeline it doesn't process any of my data!

There are numerous things that could cause this. First things to check are:

1. Have you told Pipeline the correct path to your data files?
2. Does the "files to process" wildcard mask match your file names?

If these are right and it still won't go, most likely there are things wrong with your FITS headers.

1. The IMAGETYP keyword must be "object". Pipeline will only attempt to process files that have IMAGETYP=object in the header.
2. The various CCD Section keywords must be sane, consistent and sensible. Check the values of keywords such as CCDSEC, DATASEC, AMPSEC, BIASSEC, TRIMSEC, CCDSIZE and AMPSIZE. For WFI data these should already be correct, but other data may have these keywords missing or wrong.

The processed image displayed by Pipeline is "chunky"

The image display task (mscdisplay) displays a binned image to increase speed and reduce resource needs. It is designed for a 'quick-look'. If you want to inspect the images more carefully, use `git` or IRAF.

4 Requirements

4.1 CPU resources

The data used by the processes that form the pipeline will determine how fast a CPU and how much memory and disk are required. In general, your system should be capable of running the IRAF processes outside of Pipeline to a satisfactory performance level; this will ensure satisfactory performance when using the pipeline tool. There is a 10-20 second startup dead time, when the

processing is launched to allow Ximtool to startup. The processing time for large images thereafter is dominated either by the CPU and IO resources required by IRAF. A very rough guide to a physical memory requirement is at least three times the size of the images.

To give an example, in testing we used 138 MB images containing 8 image extensions on a SUN Ultra Enterprise 3000 server with two 336 MHz processors, 512MB memory, 15 MB/sec striped disks. To complete the following processing steps took under 10 minutes per image: subtract overscan, trim, subtract zero, subtract dark, divide by flat, fix bad pixels.

4.2 Data Format

Pipeline only works with FITS data, either standard FITS files containing images, or MEF FITS files. Example headers are shown in sections 4.2.1 and 4.2.2. For more information on these formats see:
FITS web page[4]
NOAO Mosaic Data Structure[5]

4.2.1 Example Primary FITS Header

```
SIMPLE  =                    T / Fits standard
BITPIX  =                   16 / Bits per pixel
NAXIS   =                    0 / Number of axes
BSCALE  =           1.000000E0 / REAL = TAPE*BSCALE + BZERO
BZERO   =           3.276800E4 /
ORIGIN  = 'NOAO-IRAF FITS Image Kernel July 1999' / FITS file originator
DATE    = '2000-09-10T10:16:24' / Date FITS file was generated
IRAF-TLM= '21:08:45 (10/09/2000)' / Time of last modification
EXTEND  =                    T / FITS extensions
NEXTEND =                    8 / Number of image extensions
OBSERVER= 'Gary Da Costa'      / Observer name
COMMENT = 'PROPID'             / Not required for this instrument
FILENAME= '/data/monsoon1/cicada/gdc/ccd0037.fits'/ Original host filename
OBSID   = 'T_40.20000904.184013'/ Observation ID
RUN     =                   37 / Run number
OBJECT  = '47 Tuc 120s R'      / Observation title
IMAGETYP= 'object'             / Observation type
COMMENT = 'focus 32.04'        /
EXPOSED =         1.200000e+02 / Exposure time (sec)
EXPREQ  =         1.200000e+02 / Requested exposure time (sec)
DARKTIME=         1.206810e+02 / Dark time (sec)
INSTRUME= 'WFI'                / Instrument name
CAMERA  = 'WFI1'               / Camera name
COMMENT = 'FILTYP'             / Not available
FILTER  = 'R'                  / No Filter or filter position invalid
COMMENT = '4'                  / No Filter or filter position invalid
DEWAR   = 'DWFI'               / Dewar name
SPEED   = 'FAST'               / CCD readout mode
WINDOW  = 'USER_DEFINED'       / Readout window name
DETECTOR= 'WFI1'               / Detector Name
DETSIZE = '[1:8422,1:8282]'    / Detector size
NCCDS   =                    8 / Number of CCDs
NAMPS   =                    8 / Number of amplifiers
CCDTEMP =         1.582721e+02 / CCD temperature (K)
COMMENT = 'FITSKW1'            / Not available
COMMENT = 'FITSKW2'            / Not available
COMMENT = 'FITSKW3'            / Not available
```

[4] http://fits.gsfc.nasa.gov/
[5] http://iraf.noao.edu/projects/ccdmosaic/Imagedef/imagedef.html

```
COMMENT = 'FITSKW4'            / Not available
RA      = '00:24:05.4'         / Initial RA 1.051125e-01
DEC     = '-72:09:08'          / Initial Dec -1.259294e+00
EQUINOX = 'J2000.0'            /
LST-OBS = ' 3:32:-968092763.0' / LST of observation -7.040077e+04
MJD-OBS =  4.058700000000e+04 / MJD of observation (days)
OBSERVAT= 'SSO'                / Observatory
TELESCOP= '40INCH'             / Telescope
COMMENT = 'Telescope object name'/ Not available
LAT-OBS =       -3.127336e+01 / Degrees
LONG-OBS=        1.490610e+02 / Degrees east of Greenwich
ALT-OBS =        1.149000e+03 / Metres above mean sea level
AIRMASS =        1.483511e+00 / Airmass
COMMENT = 'HA'                 / Not available
COMMENT = 'ZD'                 / Not available
TIMESYS = 'UTC'                / Default time system
UTC-OBS = '18:40:13'           / UTC of observation start
DATE-OBS= '2000-09-04T18:40:13'/ Y2K date of observation start
COMMENT = 'TELPAN'             / Not available
COMMENT = 'Side of pier'       / Not available
TELFOCUS=        0.000000e+00 / Focus position
IMAGESWV= 'CICADA Release 3.0.1'/ Image creation software version
KWDICT  = 'CICADA FITS V 1.0'  / Keyword dictionary version
END
```

4.2.2 Example FITS Image Extension Header

```
XTENSION= 'IMAGE   '           / Image extension
BITPIX  =                   16 / Bits per pixel
NAXIS   =                    2 / Number of axes
NAXIS1  =                 2098 / Axis length
NAXIS2  =                 4136 / Axis length
PCOUNT  =                    0 / No 'random' parameters
GCOUNT  =                    1 / Only one group
BSCALE  =           1.000000E0 / REAL = TAPE*BSCALE + BZERO
BZERO   =           3.276800E4 /
ORIGIN  = 'NOAO-IRAF FITS Image Kernel July 1999' / FITS file originator
EXTNAME = 'im1     '           / Extension name
EXTVER  =                    1 / Extension version
IRAF-TLM= '21:08:45 (10/09/2000)' / Time of last modification
DATE    = '2000-09-10T10:15:54' / Date FITS file was generated
INHERIT =                    T / Inherit global keywords
IMAGEID =                    1 / Image identification
CONTROLR= 'WFI_Bottom'         / Controller Name
CONSWV  = 'V0.1493'            / Controller software version
CONHWV  = ''                   / Controller hardware version
READTIME=                52416 / (ms) Controller readout time
CCDNAME = 'WFI0_CCD20'         / CCD identification
CCDSIZE = '[1:2098,1:4136]'    / CCD size
CCDNAMPS=                    1 / Number of amplifiers used to readout CCD
AMPNAME = 'A'                  / Amplifier identification
RO_GAIN =        0.000000e+00 / Amplifier Gain (e-/ADU)
RO_NOISE=        0.000000e+00 / Read noise for amp (e-)
SATURATE=                    0 / Maximum good data value - ADU
LINCOEF =        0.000000e+00 / Linearity coefficient No=Nt(1+Nt**lincoef)
SUMSAT  =                    0 / Saturation level of summing wells
AMPSIZE = '[1:2098,1:4136]'    / Amplifier size
CCDSEC  = '[1:2098,1:4136]'    / Region of CCD read
CCDSUM  = '1 1'                / CCD pixel summing
```

```
BIAS0001= '[1:10,1:4136]'      / Bias section - prescan cols
BIAS0002= '[2059:2098,1:4136]' / Bias section - postscan cols
AMPSEC  = '[1:2098,1:4136]'    / Amplifier section
DATASEC = '[1:2098,1:4136]'    / Data section
DETSEC  = '[6325:8422,1:4136]' / Detector section
TRIMSEC = '[11:2048,1:4136]'   / Trim section
END
```

5 How Pipeline works

To build a pipeline we needed three main components:

A processing engine: A key requirement was to process multi-extension FITS files. Our investigations revealed that the best means of doing this was to use the recently developed MSCRED package for IRAF. MSCRED is an IRAF package developed by F. Valdes and others at NOAO for processing multi-extension FITS files (MEF files). It provides multi-extension analogs of the most of the tasks found in the ccdred packages of IRAF, as well a number of new tasks specific to MEF data. For more information see the NOAO Mosaic web pages[6]. A C++ class developed for calling selected IRAF packages and tasks is described in Section 5.2.

A data management and scheduling tool: Critical components of any pipeline processing are data management and process scheduling.

A graphical user interface: An easy to use interface for Pipeline was required as the users will be many and varied in terms of their computing skills. CICADA[7] has been implemented using the Sun WorkShop Visual interface builder and Motif widget libraries, so we developed Pipeline with a similar look and feel using the same tools.

5.1 Architecture

The above three components had to mesh neatly with the existing CICADA environment, shown at the top of Figure 10. They also had to be readily and freely available to all groups involved with the development of WFI. We chose IRAF (see panel below) for the central engine and built a Motif GUI (examples below) with a similar look and feel to CICADA. For a data scheduling and managing tool we experimented with both OPUS and our own code and for simplification our own code is used. While OPUS performed well, we only made use of a small subset of its capabilities.

5.2 IRAF Class

A key requirement of the processing engine in Pipeline was the ability to process MEF FITS files. Our investigations revealed that the best means of doing this was to use the recently developed IRAF package MSCRED. It provides multi-extension analogs of most of the tasks found in the ccdred package of IRAF, as well as a number of mosaic specific tasks. While IRAF tasks can be used in an automated fashion by running the package executables directly, we found it much simpler and more robust to use scripts to run the IRAF cl directly.

The structure of IRAF lends itself very nicely to a multiply inherited OO class structure, in the sense that a task or sub-package inherits all the properties of its parent. In order to implement the use of IRAF in our pipeline, we developed a multiply-inherited C++ class structure as shown in Figure 11. This has been done in a very general way, so other tasks and packages can easily be added. As shown in Figure 11, there is a base class for the FITS kernel, which is inherited by the IRAF class. Packages such as MSCRED then inherit the IRAF base class, thereby also inheriting the FITS kernel base class. In the present implementation the functions in these classes receive all the necessary information from the GUI or configuration files and write out commands to the

[6] http://iraf.noao.edu/projects/ccdmosaic/Reductions/
[7] http://msowww.anu.edu.au/computing/cicada/cicada_user_manual/cicada/cicada.html

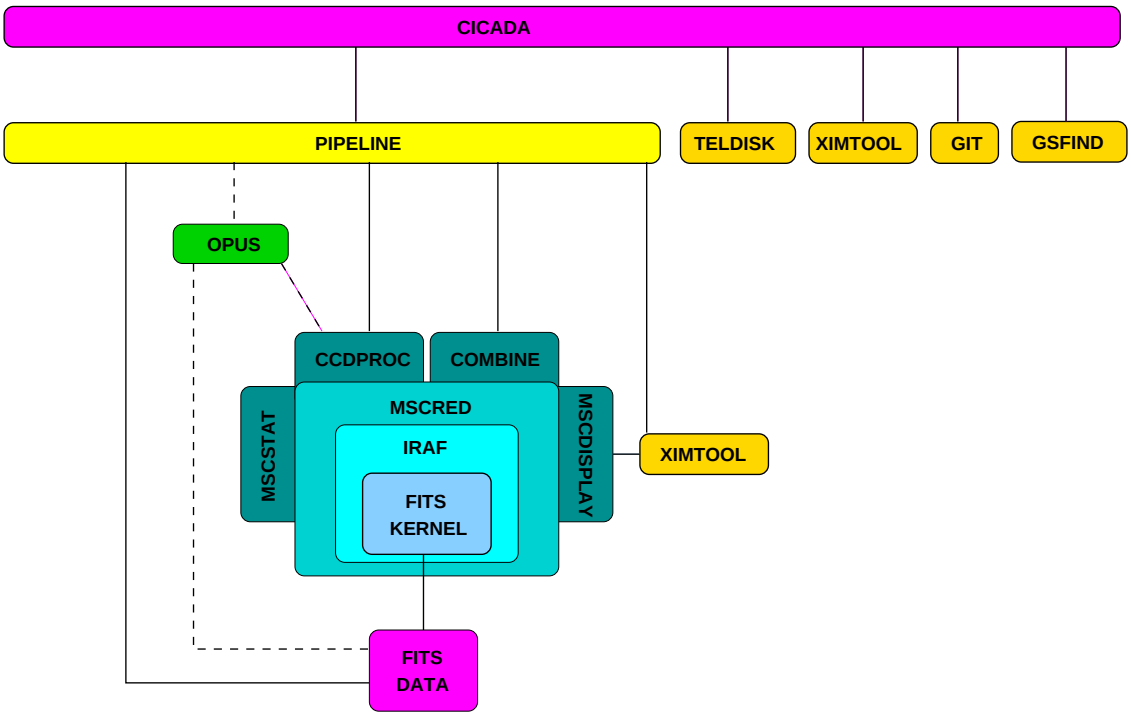


Figure 10: Pipeline architecture

IRAF engine via a UNIX pipe. These commands contain all the necessary details to run each task.

Getting information back from IRAF can be another tricky issue, when it is run in an automated way. We have approached this issue, by turning the IRAF logging facilities on and using event handlers and timeouts to return information from the log files. When Open IRAF is available it would be possible to use that to directly call IRAF tasks, rather than the method noted above.

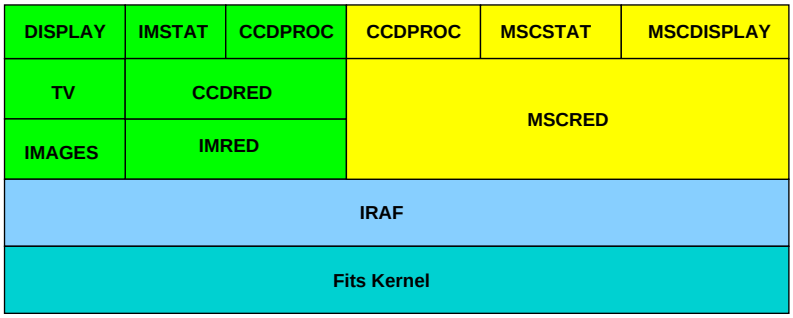


Figure 11: IRAF class inheritance structure

5.3 Retrieving IRAF Messages

In order to keep the user informed of how the processing within IRAF is going, the IRAF logging facilities are turned on, and the log files are parsed, with some of the more relevant information being returned to the message window.

5.4 Tools and Libraries Used

The GUI was developed using the SUN Workshop Visual Interface Builder V2.0[8], generating C++ code. Extensive use was made of the Tools.h++[9] class library as well as other locally developed libraries for process communication and exception handling. All modules were built and tested using the Sparc C++ compiler on Solaris 2.6 and Solaris 7. Versions of third party software used were: IRAF: V2.11.1-2.11.3 and MSCRED: V1.1-4.1. The system resources required by this pipeline are dominated by the data.

[8] http://www.sun.com/workshop/visual
[9] http://www.sun.com/workshop

6 Programming Information

6.1 Location of code in CVS tree

All the code and documentation for Pipeline is stored under CVS at MSSSO. The locations under the cicada tree are summarised in Figure 12.

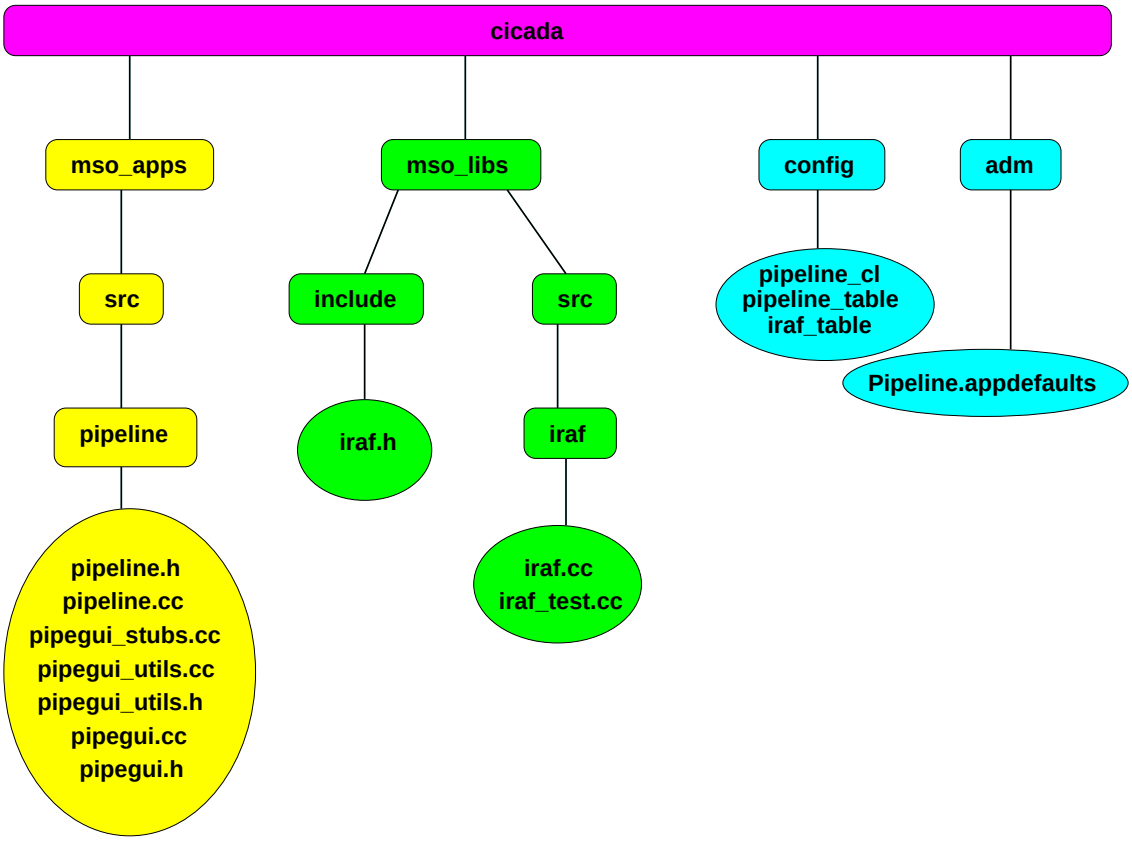


Figure 12: Location of Pipeline code in the MSSSO CVS tree

7 Features to be implemented, Issues to deal with and known bugs

7.1 Preprocessing of calibration images

Presently this is not possible and attempting to do so may damage data. Pipeline attempts to deal solely with FITS files. Unfortunately at present, when ccdproc preprocesses calibration images (eg applying a zero correction to a dark) the output image is the same as the input image and the image is changed from FITS to imh. ie The raw data is processed and the type and file name is changed. The changing of the file type is a ccdproc bug that will hopefully be fixed in a later version of IRAF. The issue of calibration image be processed in place is one that needs to be dealt with by the pipeline, possibly by copying all the calibration to a subdirectory before processing

7.2 Preparation of calibration images

Related to the previous issue, is the simpler issue of using Pipeline to preprocess calibration images. For example, one could envisage using Pipeline to apply and overscan correction to a bunch of biases, dark and flats. However, in order to prevent this happening at random, Pipeline (and IRAF) check to see if the IMAGETYP keyword is set to "object" before unleashing ccdproc on the image. One approach to this would be to add a toggle button menu to the processing parameters popup

allowing the user to specify whether the images to be processed had IMAGETYP "object", "flat" or "dark" etc.

7.3 Implement combine for preparing calibration images

The software working group advised that it would be useful to have a means of combining images built into the pipeline. A place holder GUI popup exists, the code to launch the combine process needs to be written.

7.4 Handling crosstalk

The software working group advised that some means of correcting for cross talk may be needed. It may be appropriate to wait until real WFI images are available to see what may be required.

7.5 Flat field library

There are a number of questions still to be resolved regarding the flat field library, including how they are created, maintained and stored.

8 Acknowledgements

We are very grateful to Frank Valdes from NOAO for his help, advice, and additions to IRAF and MSCRED in getting our pipeline going. We appreciated the advice and support from Jim Rose from the OPUS group at STSci. The WFI software working group: Chris Tinney, Gary Da Costa, Brian Schmidt, Tim Axelrod, Rachel Webster.

A Pipeline Tool for CCD Image Processing

Jon F. Bell, Peter J. Young, William H. Roberts, Kim M. Sebo

Mount Stromlo and Siding Spring Observatories, Private Bag, Weston Creek, ACT, 2611, AUSTRALIA

Abstract. MSSSO is part of a collaboration developing a wide field imaging CCD mosaic (WFI). As part of this project, we have developed a GUI based pipeline tool that is an integrated part of MSSSO's CICADA data acquisition environment and processes CCD FITS images as they are acquired. The tool is also designed to run as a stand alone program to process previously acquired data. IRAF tasks are used as the central engine, including the new NOAO mscred package for processing multi-extension FITS files. The STScI OPUS pipeline environment may be used to manage data and process scheduling. The Motif GUI was developed using SUN Visual Workshop. C++ classes were written to facilitate launching of IRAF and OPUS tasks. While this first version implements calibration processing up to and including flat field corrections, there is scope to extend it to other processing.

1. Context

This pipeline tool[1] was originally developed for the Wide Field Imager (WFI)[2] being developed jointly by MSSSO, AAO and the University of Melbourne, but it can be used to process other CCD image data. It has been developed in a general manner so that other processing tasks contained in IRAF[3] can be easily added. One might envisage adding facilities for calibration processing of 1 D spectra for example. While it has been developed to run in conjunction with CICADA[1] (Configurable Instrument Control and Data Acquisition), software developed at MSSSO, it may also be used offline in a stand alone mode. Figure 1 shows how this tool fits into the overall process.

2. Architecture

To build a pipeline we needed 3 main components: A processing engine, a data managing and scheduling tool and a graphical user interface. All of these components had to mesh neatly with the existing CICADA environment, shown

[1] `http://msowww.anu.edu.au/computing/cicada`

[2] `http://msowww.anu.edu.au/observing/wfi/`

[3] `http://iraf.noao.edu/`

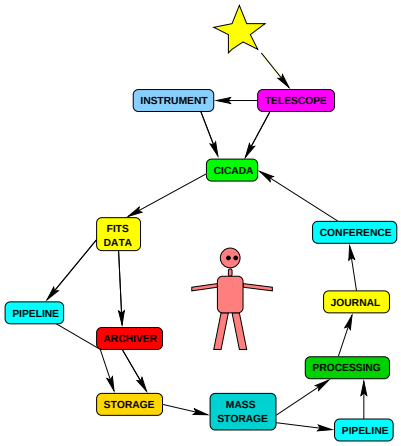


Figure 1. This pipeline tool in context.

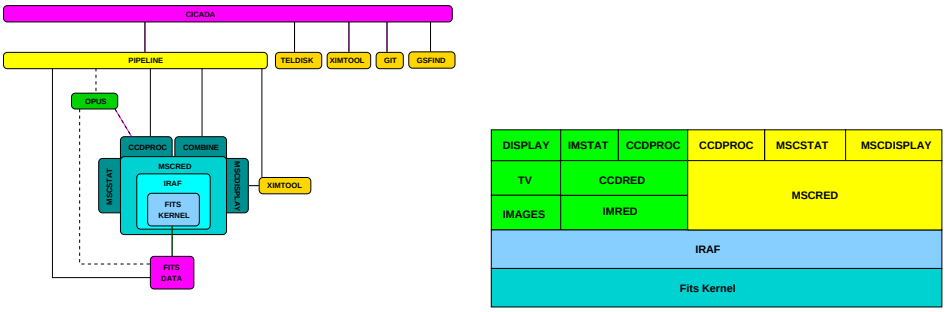


Figure 2. Left: Architecture of Pipeline and Cicada environment. Right: Inheritance structure of IRAF classes.

at the top of Figure 2. They also had to be readily and freely available to all groups involved with the development of WFI. We chose IRAF (see panel below) for the central engine and built a Motif GUI (examples below) with a similar look and feel to CICADA. For a data scheduling and managing tool we experimented with both OPUS and our own code and at present either may be used (as indicated by the dashed lines in Figure 2) . While OPUS performed well, we have only made use of a small subset of its capabilities. Therefore future versions will be based on our own simple scheduler, reducing the complexity of this pipeline tool.

3. Central Engine – IRAF

A key requirement of the processing engine in this pipeline was the ability to process multi-extension FITS (MEF) files. Our investigations revealed that the best means of doing this was to use the recently developed IRAF package mscred. It provides multi-extension analogs of most of the tasks found in the ccdred package of IRAF, as well as a number of mosaic specific tasks. While IRAF tasks can be used in an automated fashion by running the package executables

directly, we found it much simpler and more robust to use scripts to run the IRAF cl directly. The structure of IRAF lends itself very nicely to a multiply inherited OO class structure, in the sense that a task or sub package inherits all the properties of its parent. In order to implement the use of IRAF in our pipeline, we developed a multiply inherited C++ class structure as shown in Figure 2. This has been done in a very general way, so other tasks and packages can easily be added. As shown in Figure 2, there is a base class for the FITS kernel, which is inherited by the IRAF class. Packages such as mscred then inherit the IRAF base class, thereby also inheriting the FITS kernel base class. In the present implementation the functions in these classes receive all the necessary information from the GUI or config files and write out scripts containing all the necessary details to run each task. Getting information back from IRAF can be another tricky issue, when it is run in an automated way. We have approached this issue, by turning the IRAF logging facilities on and using event handlers and timeouts to return information from the log files. When Open IRAF is available it would be possible to use that to directly call IRAF tasks, rather than the method noted above.

4. Typical Processing Steps

The processing sequence that is presently implemented is shown in Figure 3, starting at the top and flowing down. The right most box describes the steps involved. Following the successful processing of each image, the image is displayed, along with image statistics, so the user can quickly verify that the processing is proceeding as intended. At present, due to the complicated nature of preparing flat field images for WFI and other wide field CCD mosaics, it is assumed that the calibration images have been pre-prepared (or obtained from a library) and the path on the right is followed for images containing science data. When the formation of flat fields becomes a more reliable process, it will be possible to automate the pipelining of all the steps shown above as IRAF already supports this.

5. Development Tools and Platform

The GUI was developed using the SUN Workshop Visual Interface Builder V2.0[4], generating C++ code. Extensive use was made of the Tools.h++ 4.0.1[5] class library as well as other locally developed libraries for process communication, exception handling and automatic setting up of OPUS[6]. All modules were built and tested using the Sparc C++ compiler on Solaris 2.6. Versions of third party software used were: IRAF: V2.11.1, MSCRED: V1.1 and OPUS: V1.0. The system resources required by this pipeline are dominated by the data.

---

[4] `http://www.sun.com/workshop/visual/`

[5] `http://www.inmark.com/products/`

[6] `http://www.stsci.edu/software/OPUS/`

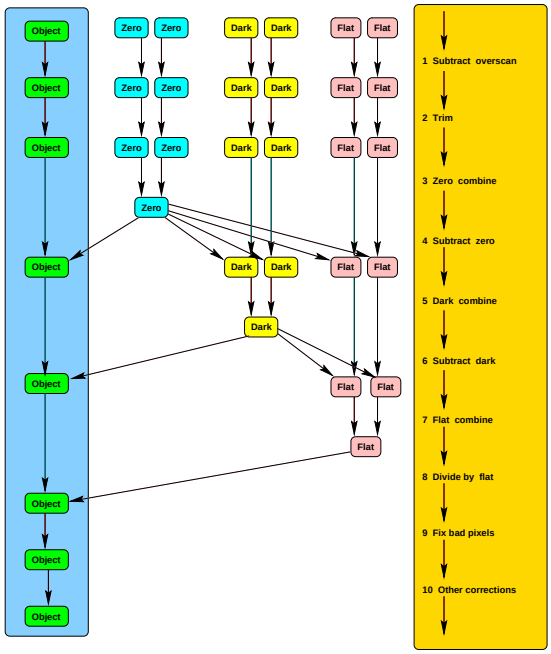


Figure 3. Processing steps presently implemented.

Acknowledgments. We gratefully acknowledge very helpful advice and support from Frank Valdes and other IRAF developers at NOAO and Jim Rose and other OPUS developers at STSCI. The WFI software working group also contributed many useful ideas and suggestions.

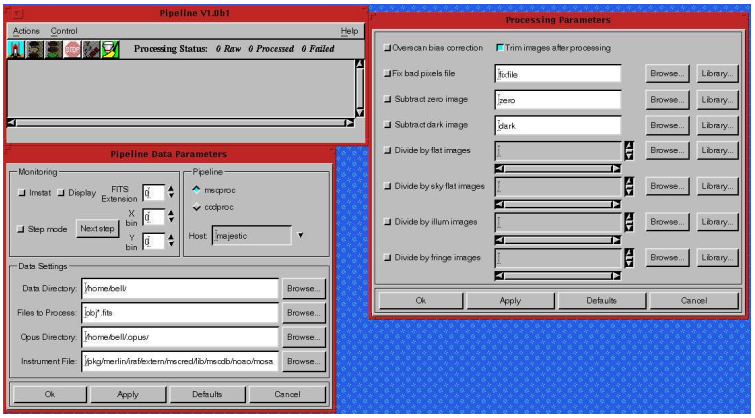


Figure 4. Snapshots of the GUIs

Developer Notes

Developing and maintaining a system such as cicada is not easy. A stable maintainable version is required for production use. Developers need the freedom to work on new features. These two requirements are not compatible. To help make this problem easier to deal with please read the following guidelines and try to follow them. Additional background information can be found in the rationale.

Tagging

- Every release of cicada that is installed at a telescope will be tagged.
- Tags look like this: Rel-4-2-1, where the first digit indicates the major release number of cicada, the second digit indicates the minor release number, and the third digit indicates the bugfix release number.
- Minor release numbers that are even indicate that this release is considered stable. Minor release numbers that are odd indicate that this release is considered unstable.
- Unstable versions of cicada should never be installed at a telescope except for testing purposes.
- Unstable versions of cicada will periodically be tagged when they reach a point that is considered meaningful in some way. Examples of this include the addition of major new functionality, or the rearchitecture of some part of the system.

Bug fixing

- Before working on a bugfix you should consult the versions web page to determine which tagged stable release your fix will apply to. The web page will show which version of cicada is installed for which instruments on which telescopes.
- Check out the tagged version appropriate to the bugfix you need to make.
- Anything that is not a bugfix does not belong on a stable branch. You will most likely know whether what you have done counts as a bug fix or whether it is something more complicated. If in doubt, check with the configuration manager.
- Before any fixes you have made can be committed you must thoroughly test cicada, using the testing checklist.
- Bugfixes to a stable version of cicada can only be checked in on a branch. Branch tags look like this: Rel-4-2-1-branch. If the stable version you are working on does not already have a branch then see the configuration manager.
- You must include a meaningful log message when you commit your changes.
- The configuration manager is responsible for building a new cicada package, installing it at the appropriate telescope, and merging your bugfixes back into the main trunk (if appropriate). See their web page if you want to know what they will do, or if they're not around and you need to do it yourself.

Feature addition

- Feature additions do not belong on a stable branch.
- Feature additions only belong on the main trunk.
- I think you get the idea.

Code portability

- The code you write should be as portable as possible. Keep in mind that not only will the code be run on Solaris machines, but potentially also on VxWorks, QNX, and Linux machines in the future. #ifdefs will be (and are) used where appropriate. Before changing any code that is inside an #ifdef block, be sure you know what you are doing and how it will affect each of the platforms involved.
- If there are good reasons to include OS specific code in a module then by all means use it (but delimit it by #ifdefs). For example we gain performance improvements on the UltraSPARC architecture by using the VIS instruction set, and we use the mmap system call on Solaris because it's incredibly handy.
- Your code may run on both big and little endian machines.

Configuration Manager

The configuration manager is responsible for looking after the cicada source tree, and making sure that it is always possible to build a stable working version of cicada for each of the instruments and telescopes it supports. Their tasks are as follows.

Tagging, Branching, and Merging

- Every stable release of cicada must be tagged with a unique tag of the form: Rel-4.2.1.
- If a bugfix must be applied to a stable release then the source tree branches at that point with a branch tag of the form: Rel-4.2.1-branch.
- Subsequent stable releases do not need to be branches.
- Bugfixes **must** be merged back into the main trunk whenever this is possible.
- Unstable versions of cicada should be tagged when a major new feature is added, when a rearchitecture of any part of the system takes place, before a new port is undertaken, and probably at some other points when it feels right to do so.
- The source tree should never branch at an unstable release.

Installing cicada at a telescope

Before cicada can be installed at a telescope there are a number of things to be done:

- The version of cicada to be installed must be stable. The only exception to this is if it will only be used for testing purposes, in which case the previous stable version must be reinstalled before the next observer is due to use the telescope (probably that night).
- A version of cicada is only considered stable if it passes all the automated tests **and** all the tests on the testing checklist.
- The version number in cicada_version.h must match the tag, and these must be unique so that there can be no doubt about which bugs and bugfixes are in the currently installed version.
- The cicada_knownbugs and cicada_whatsnew files must be updated to indicate what has changed since the last release.
- The OPTFLAGS variable in all the appropriate makefiles must include optimisation flags and not debugging flags.
- The package built must include a working cicada_ximtool for that platform (only sparc and ultra at this stage).
- The versions page must be kept up to date.

Testing Checklist

The following is a checklist of those things that must be tested before a release of cicada can be given a stable version number. After any changes to a stable version it must pass these tests again before being installed at a telescope. Most of these tests have been incorporated into a Tcl script. This belongs in the cicada tree but it hasn't been put there yet.

- start cicada without errors
- initialise the controller
- close the shutter
- open the shutter
- perform a flush
- perform a readout
- read temperatures
- do 20 consecutive exposures of the whole detector
- for each controller:
    - do 20 consecutive exposures of a single CCD
    - do 20 consecutive exposures of a region within a single CCD
    - do 20 consecutive exposures of a region that spans multiple CCDs
    - do 20 exposures that alternative between regions on different CCDs
- do 20 exposures that alternative between regions on different controllers
- do 20 exposures of a region that spans CCDs on multiple controllers
- with a region that has dimensions that are odd in both x and y:
    - do 20 exposures using binning that is not 1x1
    - do 20 exposures using x overscan that is not 0
    - do 20 exposures using y overscan that is not 0
    - do 20 exposures using x and y overscan that are not both 0
    - do 20 exposures using x and y overscan that are not both 0, with binning that is not 1x1
- pause and resume an exposure
- pause and resume multiple times during a single exposure
- pause and resume multiple times during a single exposure, changing the exposure time
- pause then readout an exposure
- pause then abort an exposure
- abort an exposure
- abort an exposure then successfully take a new exposure
- verify that FITS headers are correct
- verify that FITS files can be read into IRAF
- get the filter position using the appropriate filter plugin
- set the filter position using the appropriate filter plugin
- get telescope information using the appropriate telescope plugin
- send telescope commands using the appropriate telescope plugin
- test any site control plugins
- check that no cicada processes have memory leaks

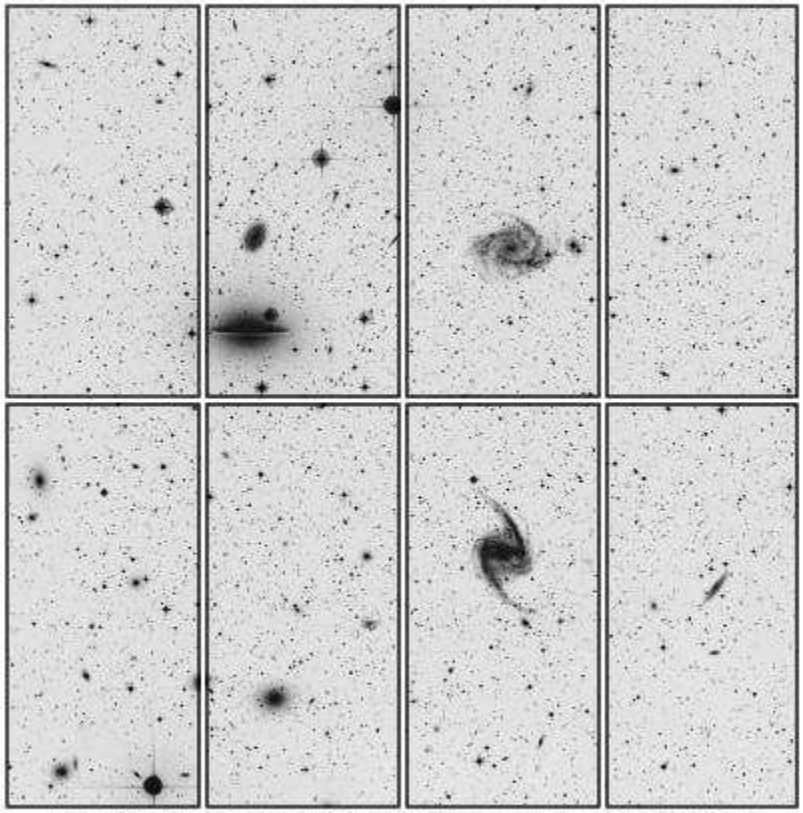
The tests should be performed with a configuration as close to the actual production configuration as possible. In particular consider:

- Testing with a dummy controller is not the same as testing with real hardware
- Remember that cicada usually runs in a multi-machine configuration at the telescopes. Testing on a single machine does not exercise the RPC services fully.

Versions

Telescope	Instrument	Version	Date installed	Bugs reported	Comments
40"	WFI	4.0.10	April 2004	WFI bugs	None
2.3m	DBS Blue	5.1.10	May 2005	2.3m bugs	None
	DBS Red	2.0.10	Unknown	2.3m bugs	None
	Imager	5.1.10	May 2005	2.3m bugs	None
UK-Schmidt	6dF	4.0.12	June 2004	6dF bugs	None
AAT	WFI	4.0.12	June 2004	WFI bugs	None
UTas	UTas camera	4.0.13	July 2004	None	A new filter plugin was developed for the UTas filter controller. No reports have yet been received as to how well it works.

WIDE FIELD IMAGER

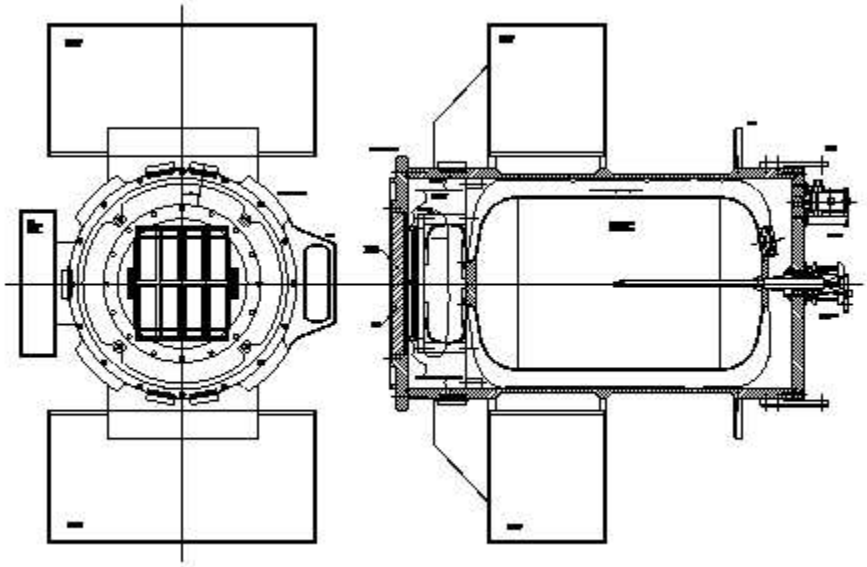


AUSTRALIAN NATIONAL UNIVERSITY

Introduction | Specs and status | Development (password protected)

WFI - Wide Field Imager

The Wide Field Imager will provide a focal plane CCD detector system with a size of 123 by 123 mm, 8192 by 8192 pixels, made up of a 4 x 2 array of 2048 x 4096 pixel thinned, back-illuminated CCDs, to be used on both the Anglo Australian and ANU 40inch Telescopes. The output of the system is pixel data in the form of multi-extension FITS files written on an archiving medium which is part of the system, and/or transported to a device of the user's choice over a Local Area Network, or fed into an automated archiving and reduction pipeline where available.



Top and side-on views of the WFI dewar showing the eight CCDs in place

At the centre of the system is a 123 x 123 mm focal plane mosaic of eight three side buttable 2048 x 4096 15 micron pixel CCDs which AAO and MSSSO are procuring as a result of a being partners in a consortium of astronomical institutions funding development of such devices at Lincoln Labs in the USA.

The focal plane will be housed in an evacuated vessel (the "dewar") and will be cooled to about 170 K by the use of liquid nitrogen refrigeration.

The dewar fits onto exposure controllers located on the AAT Prime Focus and the 40" Cassegrain Focus, each comprising a filterwheel with at least four positions, shutter and their control electronics frames. The AAT exposure controller will also accommodate their existing acquisition and guide cameras.

Autoguiding at the 40" will be accomplished by the use of one of eight small guide CCDs located on the focal plane next to the scientific detectors. They are connected to (preferably) a commercial CCD controller (SBIG ST4 or similar/better) and a PC which interfaces to the Telescope Control System. Alternatively, the guiding facility which will have to be incorporated in the exposure controller for the tiptilt system can be used. This can probe inside the camera field, shadowing it. If such a system is compatible with the existing AAT autoguider, it may be used there as well.

Two San Diego State University second generation CCD controllers will be mounted on the dewar to read out the CCDs and transfer the data over fibre optic links to the

data acquisition computer system.

The control and data acquisition computer system will consist of two Sun workstations with enough capacity to efficiently handle the data stream from the CCD controller for display, storage and archiving, plus data reduction. The use of two computers will allow the time-critical control of CCD readout to be separated from the user interface and data reduction computer. It will also allow user interface and data reduction computers to be set up permanently at both telescopes, optimally configured for each organisation's data pipelining procedures. The control and data handling software will be the MSSSO CICADA package, suitably adapted to deal with reading out CCD mosaics through the SDSU controller.

Partners

The following organisations are involved in the WFI project:
MSSSO: MSSSO will provide the overall project management, CCD expertise, four 2048 x 4096 CCD chips and the MSSSO's workshop infrastructure.
AAO: the Anglo Australian Observatory will provide 4 of the 8 CCDs. University of Melbourne: The astrophysics group will develop the scientific requirements for the Project.
Auspace Ltd: Auspace will provide project management, quality assurance and mechanical design to the project.

Contractors

The following are the major contractors that will help in the WFI project:
GL Scientific: Building the focal plane, assembly of instrument and testing.

WFI, the Wide Field Imager for the ANU 1m and the AAT.

The Wide Field Imager (WFI) is CCD mosiac camera being fabricated by the Research School of Astronomy & Astrophysics, ANU, for use on both the ANU 1m telescope and on the AAT prime focus at Siding Spring Observatory.

Construction

- WFI will use 8 4k x 2k 3-side buttable CCDs arranged in a 2 x 4 mosaic to give a total format of 8k x 8k pixels. The CCDs in WFI will come from the MIT-LL consortium lot runs in which the RSAA and the AAO are joint participants. Each institution will contribute 4 CCDs to WFI.
- The AAO is engaged in upgrading the prime focus to accommodate WFI while the RSAA has designed (in collaboration with Auspace and the University of Melbourne) and manufactured the WFI dewar.
- The RSAA is also providing the CCD Controllers (dual 4 channel SDSU V2 controllers), controller software, user interface and data pipeline processing software, as well as commissioning the instrument.
- The focal plane mosaic is being assembled by Dr. Gerry Luppino in Hawaii.
- The goal is to be able to read out the full WFI array in under 60 sec; recent tests at the RSAA demonstrated that this goal is likely to be met - a full 4k x 2k readout through a single channel took under 50 sec with 5 electron read noise. WFI will have one channel per CCD.

WFI on the ANU 1m telescope at Siding Spring Observatory

- The CCDs have 15 micron pixels which will give a scale of 0.38"/pix at the 1m telescope Cass focus with a field-of-view of 52 arcmin on a side (diagonal ~1.2 deg). Available filters will be standard UBVRI plus a far-red z filter. Any 5 can be held in the wheel at the same time.
- Gary Da Costa (gdc@mso.anu.edu.au), is the RSAA astronomer who acts as WFI Project Scientist.

WFI at the AAT

- WFI will be a stand-alone instrument at the AAT.
- The scale at the AAT prime focus will be 0.23"/pix with a field-of-view of 31 arcmin on a side (diagonal ~0.75 deg). Filters available will be standard UBVRI.
- Chris Tinney (cgt@aaoepp.aao.gov.au), is the AAO astronomer responsible for WFI on the AAT and the AAT prime focus upgrade.
- For more information see the WFI/PFU WWW Page.

Current Status

The WFI dewar and the controllers are in Hawaii awaiting sufficient CCDs to complete the focal plane mosaic, after which the first part of the integrate and test phase will be carried out. The complete instrument will then be shipped back to Australia and commissioning on the ANU 1m telescope will follow. Commissioning on the AAT will occur once the prime focus upgrade is complete and WFI is successfully operating on the ANU 1m.

Gary Da Costa, WFI Project Scientist, RSAA (gdc@mso.anu.edu.au)
Last Update 9 July 1999

WFI on MSSSO 40" Notes

June 2001

As a first-time observer at the 40 inch I found there was virtually no useful documentation, ie troubleshooting notes. However, with just a few simple fixes, and some minor changes to the way I used CICADA, I could have saved about three lost nights. I hope this saves you similar trouble. Although I did have first-night training and excellent help from technical support that doesn't cover a problem that crops up after midnight later in the run!

These should be properly html'ised but I don't have time.
At present they are at
http://www.physics.usyd.edu.au/~ande/astronomy/WFI_40inch.html

These notes have grown out of notes written by Terry Bridges (his file was /data/murky3/tjb/transit/WFI_notes but I can't find it now as murky3 has gone) and the collective experience of myself, Bob & John Shobbrook, John Goodyear, Winston Campbell, Gary Da Costa, Malcolm Harris and others.

Things change: new problems appear, old ones get fixed, new fixes are discovered. So feel free to copy, edit, improve, correct, add to, embelish, simplify or whatever these notes.

Clear Skies!
Andrew Jacob

****

CONTENTS

****

1/ BEFORE YOUR RUN...

Before your run I recommend you read:-
  a) The "40-Inch & WFI DIY Manual".
       [hardcopy only: email sbrook@mso.anu.edu.au]
  b) http://www.mso.anu.edu.au/observing/40in/DirectIm/40in/
  c) http://msowww.anu.edu.au/computing/cicada/cicada/
  d) These notes

****

2/ SOURCES OF INFORMATION
There are many sources of information, some more useful than others (from an observers point-of-view). Not all of it is on the web.

A) The "40-Inch & WFI DIY Manual"
Contents self-explanatory. Essential reading for first-time observers using WFI at the 40inch.
Get your hands on a copy of this. Written by John & Bob Shobbrook.
This manual is not on the web. Would someone please put it there?
Email John at sbrook@mso.anu.edu.au for a copy.

This manual has sections:
- Running CICADA
- Telescope Setup
- Filling the Dewars
- Bias Frames
- Flat Fields
- Dark Frames
- Pointing The Telescope - Calibrating the Pointing
- Focusing the Telescope
- Autoguiding
- Data taking
- Data Archiving
- Shutdown
- Accessingthe MSSSO Online Comment File
- Autoguider Settings
- Initialising CCD Hardware
- Shutter Priming
- WFI - General

B) "40 Inch Telescope Observer's Technical Guide"
A new (June 2001) compendium of useful information and fixes.
More essential reading. You are encouraged to add to this guide to asssist future observers.
Located on the control room bookshelf.

C) http://www.mso.anu.edu.au/observing/40in/DirectIm/40in/
This contains some basic 40" observing info. In particular, a list of what to check when you arrive at the telescope.

D) http://msowww.anu.edu.au/computing/cicada/cicada/
The online Cicada User's Manual

E) And the WFI web pages:
MSSSO: http://msowww.anu.edu.au/observing/wfi/index.shtml
AAO : http://www.aao.gov.au/local/www/cgt/wfi/wfi_pfu.html
The AAO page contains QE curves or the CCDs.

F) Other Hardcopy Manuals in the Control Room
"CICADA User's Manual" - white folder on 40" control-room bookshelf
Probably the same as the Cicada web pages, but best to use D above.
"40 inch Telescope ETS - A Short Users Manual" - located as above.
A useful guide to operating the ETS.
Only exists in hardcopy.
"A Short User's Guide to the 40-Inch and its CCD Camera System"
An old manual. Not entirely relevant to WFI but still useful in parts.
Now contained in the "40 Inch Telescope Observer's Technical Guide" folder

G) http://msowww.anu.edu.au/observing/tc_manual/
An online "Telescope Command Reference Manual". More technical than the average user needs. Use the "40 inch Telescope ETS - A Short Users Manual" instead (see F).

H) Additional Notes for Shobbrooks manual
Some additional or alternative notes to add to the Shobbrook manual.
Running CICADA:
-See additioinal notes below
Filling the dewars:
- Press the big red STOP button on the dome console to prevent errant telescope movement.
- Remove (or replace) the lens-cap BEFORE filling the dewar to avoid (minor) LN2 spillage from the camera dewar.
- Need a step by step procedure for pressurising the small LN2 tank??
- Need a step by step procedure for refilling small LN2 tank from large LN2 tank??

Flat fields:
Dome flats

```
  -Do these BEFORE filling the dewar to prevent major LN2 spillage.
  -Dome at azimuth=~70deg. Telescope at ZD~=-70deg gets flat screen in place.
  -Use dome lights (tungstens only!) or preferably sunlight by slightly
   opening the upper shutter.
  -Dome flats are not good. Use twilight and dark sky flats. [Terry Bridges]
Focusing the Telescope
  -See additional notes below
Autoguiding
  -If the autoguider PC has to be restarted:
    -When asked for a password, just click "cancel".
    -After windows starts open "vncviewer for WFI guider" icon.
     The "VNC server" is m40slave:0 . Password is in the back of the
     40" CCD Temp Log Book. You will need to scroll down the "Maxim Dl"
     window to access the "Guide CCDs - WFI" chooser.
Data archiving
  -the DDS3 DAT drive is now on monsoon (since Nov 2000).
  -mt rewoffl  rewinds and ejects the tape
  -How to skip over a tape file if you want to read or extract the
   second, etc file??
Shutdown
  -Press the big red STOP button when not using the telescope to prevent
   errant telescope movement.
Initialising CCD Hardware
  -this should be done as a last resort, though there is some disagreement
   on this!
Shutter priming
  -See additional notes below

*****************************************************************************

3/ GENERAL INFORMATION

Some useful information, most of which doesn't seem to appear anywhere else.

BASIC DATA
==========
WFI chips:
Eight 2k x 4k (2048x4096) CCDs in a 2x4 mosaic.
Each sees 13'x26' (arcmin) making a 52'x52' field.
Pixels are 15micron giving 0.375"/pix (on the 40in at f/8)
Gaps between CCDs are ~20-30arcsec.
Gains are ~1.5-2.1 electron/ADU
RONs  are ~3.5-5.5 electrons
 (To get Gain & RON follow procedure in  "40 Inch Telescope Observer's
  Technical Guide")
The awesome SDSU controllers read out the entire array in 55 seconds!!

A full array image is 140Mb in size.
One CCD is 17.5Mb
The /data/monsoon1 disc on Monsoon holds almost 100Gb - About 4-8 days data.
A DDS3 125m tape holds fractionally over 18Gb compressed
 (ie using allocate mt1)
Writing to tape takes ~11.5min/Gb (~3.5hrs per full tape). So don't get
caught out on your last night, save early!

See also the AAO and MSSSO WFI pages.


CICADA, GIT & Ximtool
=====================
General
-------
Strategies to deal with CICADA.

Cicada has a tendency to hang, usually during readout, and sometimes
for no obvious reason (even if you are doing none of the bad things listed
below).

The only remedy is to restart it. To do this:
 - In the Observers Window menu select actions|quit_observing_window
 - In the CICADA menu select Start_Observing|Exit_CICADA
 - From the command line do:
   - psef |grep cicada
   - Look for cicada processes. Ignore *_svc. Kill others with
     cicada_cleanup -p 0
     cicada_cleanup -p 1
     cicada_cleanup -p 2
     cicada_cleanup -p 3
   - Restart using "cicada &"
```

In general restarting solves Cicada problems, but it's a pain so...

Follow these next practices for relatively trouble-free operation:-

Pause - DON'T use 'pause'. This crashes cicada so badly you need to logout to restart! (apparently, but I never dared try it)
NumLock - Keep NumLock untoggled. It interferes with ximtool (and GIT?) operation. In particular, it prevents you using right-mouse scrolling to change brightness/contrast, and inactivates the ximtool menus.
Initialise Hardware - advice is to use this as a last resort. (Doesn't appear to cause any trouble tho, despite comments to the contrary)

Avoid:
- Using ABORT. You'll often then need to exit cicada and restart.
- doing too much (processor or memory intensive stuff) on monsoon during readout. In particular avoid:
  - Beginning or ending a tar command during readout (during exposing is OK)
  - Doing much in Netscape (on-line banking is bad!)
  - Copying images (and other operations) in iraf

Could do these things from murky but its way slower.
(For 6 CCDs (104Mb image) the limit might be ~400Mb (as shown on the Observers window at "Memory:???MB"))
- leaving the little "graphics keystrokes" window open during readout
- changing regions (in regions window) during readout [This may not be true?]

During readout you CAN do these:
- change filter, change exposures, change duration

GIT (Graphic & Imaging Tool)

GIT (presently) has a problem with displaying mosaiced images. It displays them OK on readout but it cannot set sensible scaling parameters to allow you to calculate star profiles, get single line plots, etc. The ximtool window just goes all black or white and displays a single value.

To overcome this limitation do the following:
- untoggle NumLock
- select options|start_private_ximtool
- select options|preferences :
  - select category "General"
  - untoggle "Mosaic MEF files in Cicada mode"
  - enter the "MEF extension list", ie a list of which CCDs you want available for loading into the private ximtool.
    Note that if you are only using CCD numbers 1,2,7,8 (say) their MEF image extension numbers will be 1,2,3,4. ie CCD number and extension number are different things.
  - set "Maximum Cicada images" if necessary
- The list of extensions you have requested will appear in the "image list" window. Click one and it loads into the private ximtool.

Now you can get star profiles, cross sections, stats, etc. But when the next image is read out it will automatically be loaded into the private ximtool.

But avoid leaving the little "graphics keystrokes" window open during readout or GIT will crash out.

Ximtool

North is right, East down.
1 pixel = 0.375arcsec

To move a star N dec offset on ETS is -ve
To " S dec " +ve
To " W RA " +ve
To " E RA " -ve
(This is better seen with a diagram)

Astigmatism leads to elliptical star images when defocused (see Focusing below)

Screen Preferences for GIT and Ximtool

Select the monitor for these in options|preferences|general
Common practice puts GIT and Ximtool on screen 1.

Other

Binning
If you change the binning, it may not change back again when you reset

it to 1 x 1, even though the regions window shows it as 1 x 1.
Watch the file sizes: should be 271440 blocks (full 8xCCDs?)

Bleeding
If readout shows bleeding on brighter stars on some CCDs and bias changes, restart cicada.

Wrong Frame Size
If there is an error message about ximtool having the wrong frame size, check that there is not an old cicada ximtool process running:
ps -ef | grep cicada will show it

Many Red Error Windows
If lots of red error windows appear from cicada, and restarting impossible or doesn't work, try logging out (and back in). That can fix it. If the error messages involve SDSU, then probably need hardware init (from action menu): if such error message reported when cicada is restarted, may need two hwinits. Do the hwinit right after restarting cicada.

COMPUTERS

```
  mosaic  (in rack)   runs controllers
  monsoon (on desk)   does rest - cicada
  murky   (in office) is a bit slow.
```

Data is stored on monsoon in /data/monsoon1/cicada/username

FOCUSING

This is difficult to maintain. There is backlash in the mechanism. And it's pretty temperamental. But then just wait till you're 38 years old!

I found the best method was:

1/ At the beginning of a run (or if focus is lost completely)
    Use Cicada's semi-automatic routine under actions|do_focus_sequence...
    Follow the prompts and/or read the Cicada manual.
    Move the focus by 0.03-0.05 units between each image, starting at about 31.70. When do_focus_sequence is done set the focus position with the bottons on the round control paddle

2/ From now on use the telescope's astigmatism. Star images are elongated:
     NW-SE => focus too high: Press out [display decreases]
     NE-SW =>   "   too low :   "   in ["   increases]

3/ Check star profiles too as you go. Out of focus => lots of scatter in the profile, of course.

...but then I was only taking short exposures (1-3 mins). Maybe you could use the autoguider image to maintain focus with long exposures? See also the Shobbrook manual.

It takes a couple of nights to get the hang of focusing.
Approach a focus position from the same direction to remove backlash. Otherwise get double and/or smeared images.

Typical focus values range from 31.65 - 31.88

OTHER

Readout speed
 Use fast readout for broadband imaging when there is a choice.

CCD temperature
 This appears on the Cicada observers window
 (from 158-165K for TB). Was ~183.0 for my run.

Filter control
 Select instrument_control|Filter:F_40 to open the filter control window. To initialise, hit "get filter position"

Dewar vacuum
 Watch the vacuum gauge - rises about 1E-07 per day.
 Probably OK up to about 1E-04. LN2 hold time is about 12 hours.
 If much less, dewar needs pumping (can be done cold in daytime).[TB]

CCD GUIDER SYSTEM

I had no trouble with autoguiding, just used all the settings that were there.
The following are Terry's notes. [AJ]

To disable completely, turn off the switch near the mains plug boards
on the north side of guider box, low (on telescope)

GSFIND

This shows outline of the main and guiding CCDs relative to GSC
stars. Can start from cicada|tools menu or directly from command line.
Then drag to enlarge the window.

Choose monitor from cicada|options|preferences|general. Display on
monitor 0 to see a green outline against a black background. On
monitor 1 the outline and the sky are black [OK for me on monitor 1, AJ].

Put in the field coordinates and magnitude limits for stars (< 12 for 5 sec
exposures).

Set position angle = 90.

Move CCD outline with the LMB - gives (RA, dec) coordinates of the detector
center.

Put cursor on star and hit spacebar to see the star highlighted in table.
Also gives (x,y) position of selected star in the focal plane.

Procedure

[needs editing? go ahead...]
-Select a guide star and the guide CCD.
-Set telescope, *open shutter*.
-Now go to the autoguider console.
 VNCviewer is the guider program to choose from desktop. Maxim is the
 autoguider program itself, similar to the one on the DBS.
-Choose "Guide CCDs - WFI" from the menu bar at the bottom of the
 window. Choose which guide CCD you are using: same orientation as in GSFIND.
-In the "Maxim CCD" indow click the guide tab, enter the exposure time and
 declination.
-Take an exposure with guide CCD: Click "expose" then "start".
-Wait for exposure to read out in big window.
-Move the star where you want it in the guide CCD field:
 Use telescope offsets on round "box" to offset telescope;
 North pulls the star *down* on guide window,
 East pulls the star *right*.
-When star is centered (or where you want it), left-click on it, then
 click on Track, then click Start.
-After it reads out it will create a little box in the upper left-hand
 corner where you should see the star centered (roughly).
-Wait a bit to make sure telescope is tracking OK (no huge jumps in
 star in guide window).
-When telescope tracking ok, close shutter and start the real
 exposure.

If you don't see any star in guide window, make sure shutter open!!

Restarting Guider PC

The Autoguider PC will ask for a password, just click "cancel".
 -After windows starts open "vncviewer for WFI guider" icon.
 The "VNC server" is m40slave:0 . Password is in the back of the
 40" CCD Temp Log Book. You will need to scroll down the "m40slave"
 window to access the "Guide CCDs - WFI" chooser.

Hit VNCviewer - this downloads display from m40slave (= m40spare).
Should see icons for WFI GS selector and Maxim. Run Maxim. Get the main
gui up by hitting the CCD control toggle button on the upper
menu bar. Go to setup/restart (don't worry about the "controller
off" message). Need to scroll window up and down to access the
bars at the top and bottom. Don't forget to put in the dec in the
main gui.

Settings for guider: make sure x,y axes enabled. Aggressiveness 4
is OK. X speed 10, Y speed 4.83, x axis cal time 1, y axis cal time 3.

On the guider display: all RA, dec buttons push the image.

TELESCOPE & DOME
================
Red Emergency STOP buttons
 One is on dome console
 One is on electronics rack in control room

Typically, you will run the telescope from the control room while observing
and from the dome when starting-up/closing-down, setting up for dome-flats,
filling dewar, showing-off to visitors, etc.

To transfer control TO dome:
 1/ On ETS PC select Configuration|configure. Toggle Dome Control to manual.
 2/ Go to dome.
 3/ On dome console: switch to slew/set
 4/ Move telescope, dome and floor as necessary
 5/ Hit big red STOP button if filling dewar or stopping observing.

To transfer control TO control-room:
 1/ Hit "emergency-stop reset" button (illuminated beside STOP button)
 2/ Ensure floor is fully down
 3/ Switch to auto
 4/ Go to control room
 5/ On ETS PC select Configuration|configure. Toggle Dome Control to automatic.

Telescope

Telescope RA tracking sometimes drops out. Just redo the tracking command
to restart.

SDSU Cooling

The SDSU controllers are cooled by the NESLAB cooler (downstairs in corridor
through library on the left)

If controllers get hot, probably means NESLAB has failed.
Check temperature of controller heat sinks occasionally - touch them.

CAMERA SHUTTER
==============
The shutter is badly designed. At large easterly hour-angles (>~4hrs) it
fails to reset. Recognise this by the star trails on an
image (trails on CCDs 1-4 trail up, on CCDs 5-8 trail down). There will be
a ramp in the signal across the CCDs too. [need an example image here]

To fix, you will need to prime the shutter manually.

You can listen to the shutter operation (loud CL-LICK on open and close)
with control room door open. A failed shutter sounds different, a woosy,
fainter click. You'll just have to listen! [need an example sound file here]

Priming the shutter

Transfer control to the dome.
Move telescope to ~-45deg dec, HA=2.5hrs East. Move floor up to the level
of step 5 on the concrete stairs. This gives access to the big electronics
rack on the side of the telescope tube.

Follow the procedure in the "Observers Technical Guide" and/or Shobbrooks
procedure.

But then test the shutter is working again:
 -return prime switch to NOR
 -toggle mode switch to manual
 -toggle display switch on
 -toggle shutter switch open and closed to test.
  (Don't forget to wait 8 seconds for the shutter to reset after closing)
 -Return all four switches UP before resuming observing.

Shutter Priming Delay

The shutter takes a few seconds to prime, so need to **wait** 8-10 seconds
after closing the shutter (eg after acquiring guide star) before exposing.

IRAF
====
WFI produces mosaiced images. In iraf need to specify the extension number in

```
the filename, as below.

Hopefully you've used IRAF before!
If not see http://msowww.anu.edu.au/computing/lun/lun35/lun35.html
 Set the terminal type as xgterm (at the prompt after mkiraf)
 and edit login.cl (or loginuser.cl) to set
   set stdimage = imt4096
   set imtype   = "fits"

Then:
 ximtool &   :brings up an ximtool to display images in.
 cl          :starts iraf

And:
 display filename[#CCD] 1

 imcopy filename[#CCD][xmin:xmax,ymin:ymax] outfile.fits

 imexamine filename[#CCD][xmin:xmax,ymin:ymax]
  image operations: v does plot from two cursor input positions
                    r gives radial profile of star, fwhm etc
                    a gives mag and fwhm etc for star, no plot

Even for a single CCD readout or a small region readout, need
the extension filename[1]

Use the mscred package in iraf to reduce mosaiced images.

ds9, saotng are nice alternative display tools. ds9 can handle
mosaics.


3/ TROUBLESHOOTING
==================
If stuff goes wrong...
Here are some possible causes. Solutions are in these notes or somewhere
in the sources listed above. Sorry, I didn't have time to finish this quite
the way I would have liked. But I'm sure you'll manage :)

First, follow the suggestions above to keep Cicada as stable as possible.

In control Room
---------------
My object doesn't appear on Ximtool after readout?
 -Cloud
 -Dome is blocking view
 -Dome shutter blocking view
 -Camera shutter failure

I see streaks on my image?
 -Camera shutter failure
 -Initialise hardware (a last resort)

I can't analyse images (star profiles, stats, etc) in ximtool.
The image just goes to a single value!
 -start a private ximtool in GIT.

Cicada has hung?
 -Exit Cicada, cleanup, restart Cicada.
 -Stop processor/memory intensive processes
 -Avoid using abort
 -Don't use pause

Lots of Red Error Windows, Can't Stop Them!
 -Kill Cicada from the command line, cleanup and restart

I can't point the telescope more accurately than 20arcsec?
 -Tough! The telescope only points to within about 20arcsec
 -But, use buttons on round focus/guide/set paddle. Use set or guide speeds.
  With practice this becomes fairly accurate (to within a few pixels).

Ximtool menus are inactive
 -untoggle NumLock

I can't change brightness/contrast using left mouse scrolling?
 -untoggle NumLock

New images keep loading into my private Ximtool?
 -turn off automatic loading of images by toggling the "cicada mode"
  button under the GIT options|preferences|general menu.
```

(This is suposed to work, but doesn't)

My star images are elliptical
-focus telescope

Focussing is difficult
-too bad!
-try focusing from same direction to take up backlash

There is bleeding on stars & bias has changed
-restart cicada.

I can't see my guide star (while setting up guiding)
-Open the shutter!

I can't find the "Guide CCDs - WFI" chooser window
-scroll down in the m40slave window

Autoguiding doesn't seem to be working accurately
-set the declination correctly on the "guide" tab in the Maxim CCD window

In Dome

Dome doesn't follow telescope?
-Check dome position encoder (high on the western wall) is flashing its green LEDs at you. If not restart ETS (select file|shutdown, then startup at the prompt).
-Put dome in auto mode on ETS (select configuration|configure)

Telescope pointing is bad
-Calibrate the pointing
-Reset the time (Phone 1-1194)

Telescope doesn't move
-ensure floor is fully down
-Fault in ETS (Call technical support?)

Telescope moves unexpectedly by itself
-Press big red STOP button on dome console, or on electronics rack in control room

Shutters don't move
-press shutter|stop button and try again (Listen for the relays clicking downstairs)
-If upper shutter doesn't close it needs to be driven fully open (onto its springs) first.

LN2 leaking from filling tube?
-Tighten the various nuts on the tube

LN2 leaking from camera dewar at high ZD angles?
-Remove lens-cap before filling camera dewar
-do dome-flats before filling camera dewar

LN2 small tank pressure low?
-small tank needs filling from large tank (call technical support?)
-pressurise small tank with N2(or air?) line (How?)

SDSU controllers feel hot
-Water cooler has failed. Check it and restart it, if necessary.

Dewar vacuum rising or focal plane temperature not holding
-pump dewar (call technical support?)

John Norris
3 November 2004

RULES FOR OBSERVING AT SSO

1. Observers are reminded of the dangers of working at night. Torches must be carried when moving within and about buildings at night. Torches and batteries are available at the ANU Office and the Lodge. Rechargeable torches are wall mounted in domes.

2. Observers are requested to establish contact with technical staff (telephone: **042 8639241** or **6224**) on the first day of their run to let them know of any outstanding requirements and to learn of any recent instrument and/or telescope developments.

3. All accidents to personnel should be reported immediately to the Site Officer, SSO:

   - Telephone : Office - **6234/6244**; Mobile - **(1) 042 7685288**; Home - **(1) 6842 1861**

   Portable safety phones are provided in the domes of the 24 inch, 40 inch and 2.3m telescopes. These have the numbers of safety officers programmed in memory, and should be carried when leaving the console area. Make sure you know how to operate the touch-pad of the portable phone.

4. When using a given piece of equipment or telescope for the first time, an observer must receive training from a qualified observer. The amount of training will depend upon the equipment to be used and the experience of the trainee, but will normally be at least two nights. The training requirements will be specified by the Director, MSSSO. To accommodate such training requirements, you should ensure that sufficient time is available before your observing run commences, *and you should liaise with observers using the same equipment as yourself prior to your run.*

5. Only observers listed on the schedule are permitted in the domes at night or at Siding Spring Lodge. Exceptions to this are the trainees defined in (3) above, technical staff, and visitors authorized by the Director.

6. All mechanical, electrical, and software difficulties must be reported via the Telescope Fault Reporting System[1]. Some indication of weather conditions and objects observed should be given in the log book.

   Observers are requested to complete an electronic report at the end of their run using the appropriate link under the Telescope Fault Reporting System.

7. Observers should note that a member of the electronics technical staff works a shift ending at 6 pm Monday through Friday. The function of the staff member is to attend to equipment malfunctions which become evident during afternoon setup and testing and to progress project work for MSSSO as a whole.

   Astronomers are urged to test their equipment as thoroughly as possible during the afternoons to minimize the need for night time callouts.

   A technical staff member is on call until 10pm, Monday - Friday and from 11 am to 11 pm on Saturday and Sunday and may be reached by mobile phone **(1) 042 8639241**, or failing this, by consulting the home phone number of the technician on duty displayed in the telescope. Advice will be offerred by phone, and the staff member will come to the telescope as necessary. It is not intended that the staff member be used as a telescope operator/tutor. Outside of the specified hours, if a telescope or equipment condition occurs that endangers either life or equipment, the Site Officer should be phoned at home - **(1) 6842 1861** or on mobile - **(1) 042 7685288**.

   There is no technical support during Easter and the ANU Christmas Break.

[1] http://www.mso.anu.edu.au/observing/support/faults.php

8. It is the observer's responsibility to ensure that the CCD dewars are kept topped up so they do not run out of liquid nitrogen. Details of the filling should be entered into the appropriate dewar book.

9. 2.3m observers are required to fully power down the telescope at the console at the end of each night's observing in order to reduce heat input into the dome during daytime. Please also turn off the dome fluorescents from the console.

10. Any alterations or revisions to the schedule must be arranged through the Observing Schedule Officer (Mr. Vince Ford at Mt Stromlo Observatory) and are subject to the approval of the Director.

11. Unscheduled time must be applied for to the Director in writing at least seven days prior to the desired commencement date.

12. Unscheduled equipment changeovers must be applied for 48 hours (exclusive of weekends) in advance of the desired changeover date, and are subject to the approval of the Director. Observers should communicate their equipment requirements to the Operations Officer and the Site Engineer at SSO, at least 48 hours before commencing their run, to ensure that the equipment is available.

13. It is the observer's responsibility to ensure that he/she has suitable filters, exabyte or DAT tapes, etc.

14. Observers are requested to check their equipment in the very early afternoon of the first day of use of a telescope. Any necessary alterations or additions can then be carried out during normal working hours (preferably) or before the end of the shift ending 6pm Monday through Friday. On weekends callouts should be used if necessary.

15. The domes, telescopes and auxiliary equipment are under the control of the assigned observer. All other staff, including technical staff, may not use the facilities or equipment without the permission of the observer. At Siding Spring there is one exception to this rule. From time to time it will be necessary for technical staff to have access to the telescope and equipment in the course of their testing and maintenance duties. In this case they will return the equipment to its original state, and inform the observer of their actions when relevant.

16. It is important to remember that you are only a temporary user of the telescopes and equipment and, in fairness to others, you should attempt only the most minor repairs - **all major repairs must be undertaken by official maintenance personnel**.

17. Telescopes should be closed down (and the dome firmly shut) when :

    - (a) the wind exceeds :
      - 45 knots = 51 miles/hr = 23 m/sec = 83 km/hr for the 2.3m telescope.
      - 30 knots = 35 miles/hr = 15 m/sec = 56 km/hr for other ANU telescopes. Observing can be carried out at the 40-inch at wind speeds up to 45 knots provided the lower shutter is down and/or the dome aperture is facing away from the prevailing wind.
    - (b) the humidity exceeds :
      - 95% in the case of the Uppsala and 2.3m telescopes The observer should record in the log whenever the telescope is operated over periods during which the humidity exceeds 90%. *If the internal handrails in any dome feels wet, particularly those at the Nasmyth level of the 2.3m , observing should cease.* Observers are reminded that observing time is lost when damaged mirrors have to be realuminized.
      - 99% for all other ANU telescopes.

    (In the absence or malfunction of anemometer or hygrometer, the observer should use his/her best judgement to protect the equipment.)

18. Extreme care must be taken to avoid having the dome open during rain. In the event of observations having been terminated for some reason, it is advisable to close the mirror cover and dome shutter.

    The observer must inform the Site Engineer if rain enters the dome.

19. Please do not consume food and drink in the vicinity of electronic equipment, particularly computer peripherals such as key-boards, disk units, etc.

20. All safety codes generally adopted within the Australian National University must be adhered to, together with specific codes for Siding Spring Observatory as may be adopted from time to time by the appropriate safety committee. Safety devices such as alarms on moving floors, etc, must not be overridden by the observer.

Professor Penny D Sackett
Director
Research School of Astronomy & Astrophysics
November 2004


AUSTRALIA
information
zoom out
Darwin
Kakadu National Park
Gulf of Carpentaria
Great Barrier Reef
INDIAN OCEAN
PACIFIC OCEAN
Cooktown
Cairns
The Kimberley
Broome
Northern Territory
Townsville
Queensland
Alice Springs
Uluru (Ayers Rock)
Western Australia
South Australia
Flinders Ranges
New South Wales
Noosa
Brisbane
Perth
Barossa Valley
Great Australian Bight
Adelaide
Canberra
Sydney
Victoria
Melbourne
Snowy Mountains
Great Ocean Road
Bass Strait
Freycinet Peninsula
Hobart
0 400 km
0 250 mi
SOUTHERN OCEAN
© Lonely Planet

street-directory.com.au

Virtual Map

LOCATION BASED SOLUTIONS FOR ONLINE MAPPING

NEW SOUTH WALES MAP

1 2 3 4 5 6 7

Buy this map & Integrate into your website

Back To Normal Map

THE LOWEST PRICE IN AUSTRALIA - GUARANTEED ! (All prices below inclusive of GST)

Research School of Astronomy and Astrophysics

Mount Stromlo and Siding Spring Observatories

How to get to Siding Spring Observatory

From Sydney

Coonabarabran is about 500 kilometres north west of Sydney and Siding Spring Observatory is about 40km from Coonabarabran. Most people travel to Coonabarabran either by air or car, though other modes of travel (train, bus) are possible for the adventurous or those with time to spare.

Travel options

For detailed information on travel between SSO and Sydney, please refer to the AAO's Travel and Accommodation page.

Travelling by road - some advice

It is worth reminding people who travel to or from Coonabarabran by road of the main risks, which are:

- The standard of even the main highways in NSW varies from excellent to potentially dangerous and the journey from Sydney to Coonabarabran involves mainly secondary roads. There are several places where road conditions change rather suddenly and can catch drivers unaware, especially if they are travelling fast.
- Some stretches of road pass through countryside frequented by kangaroos which constitute a serious traffic hazard, especially in twilight and at night. Try to ensure that you start your journey early enough to avoid them when approaching Coonabarabran.
- The road journey between Sydney and Coonabarabran is quite long (around 7 hours). Drivers should be well rested before they embark on it, and should also stop for short rest breaks along the way.

From Mount Stromlo Observatory

By Air

There are no air services between Canberra and Coonabarabran but check the AAO's Travel and Accommodation page for air (and other) services between Coona and Sydney.

By Car

Stromlo staff and students travel by car from MSO to SSO. It should be possible for you to share a car with someone else going to SSO for an observing run. The trip takes about seven hours, so be prepared to share the driving and read the above warnings about driving on N.S.W. country roads.

If you have your own hire car, ask anyone at Stromlo for advice on directions to SSO.

Page last updated: 07 January 2005
Please direct all enquiries to: Webmaster
Page authorised by: Director, RSAA

The Australian National University —CRICOS Provider Number 00120C

AAO Travel and Accommodation

Search the AAO:

Search »

General Links
Press releases
Newsletter
Annual Reports
Employment
Images
Students
UK Mirror

Professional Links
AAT Schedule
UKST Schedule
Users' Committee

Want to book your travel for your next observing run? Use our Travel Booking Form.

Travel between Sydney and Coonabarabran

Coonabarabranis about 500 kilometres north west of Sydney. Most people travel to Coonabarabran either by air or car, though other modes of travel (train,bus) are possible for the adventurous or those with time to spare. For some helpful advice on places along the route to Coonabarabran and Siding Spring, please go to our Librarian's information on the Road to Coona

By air: Either via Dubbo Airport, which is the closest main airport in the area, or Coonabarabran Airport*
*NOTE: BIG SKY EXPRESS has suspended it flights to Coonabarabran until further notice. A review is planned for April 2005, and any developments will be posted here.

If you require further assistance please contact the Travel Officer.

Dubbo:
The next closest airport is Dubbo, which is currently only serviced by Qantas (code sharing with Eastern Airlines.) Details of services and charges can be found on their web site (you can also book on-line): www.qantas.com.au

Getting to Siding Spring from Dubbo Airport:
If you want a taxi booked, please let us know on the travel form and we will arrange it. The taxi fare is now $AUD 350 each way and takes about two hours.

You can also arrange a hire car. It would be wise to do so in advance. (See the warnings below relating to car travel.) Thrifty Car Rental operates out of Dubbo airport. Further details on their web page:

- By car:
- Road travel notes are available here.
- It is worth reminding people who travel to or from Coonabarabran by road of the main risks, which are:
  - The standard of even the main highways in NSW varies from excellent to potentially dangerous and the journey from Sydney to Coonabarabran involves mainly secondary roads. There are several places where road conditions change rather suddenly and can catch drivers unaware, especially if they are travelling fast.
  - Some stretches of road pass through countryside frequented by kangaroos which constitute a serious traffic hazard, especially in twilight and at night. Try to ensure that you start your journey early enough to avoid them when approaching Coonabarabran.
  - The road journey between Sydney and Coonabarabran is quite long (around 7 hours). Drivers should be well rested before they embark on it, and should also stop for a short rest breaks along the way.

It is possible to hire a car in Sydney to do your round trip to site. The prices for the main hire car companies vary slightly but for under 5 days $60.00 to $70.00 is about average per day, plus, in some cases, an additional kilometre charge.
The AAO usually books hire cars through Red Spot Car Rentals, or Avis Car Rentals. Both companies deliver to where observers are staying in Sydeny, and pick up the returned vehicle, for no additional cost (within a 10Km radius of the nearest rental office). Their rates are also fairly competative, with Red Spot averaging at $42 per day.
Alternatively, Bayswater Car Rentals is even cheaper, however clients must pick up the vehicle in person at the Kings Cross office, and they are not open on Saturday afternoons or Sunday all day. The Travel Officer can book the car if instructed to do so on the Travel Form

By train:

The CountryLink division of State Rail offer a linked train/bus service to Coonabarabran, this has the longest journey time but the train leg through the Blue Mountains is very picturesque. The service departs from Sydney Central Railway Station, arriving in Lithgow 2hrs and 20 mins later. In Lithgow meet the coach to Coonabarabran, which takes between 5hrs to 5hrs and 25mins. The total cost is $AUD119.00 return. A combination train/coach ticket can be purchased as one ticket from the CountryLink office.
The website for CountryLink is www.countrylink.nsw.gov.au
(Note: Currently, the website does not include a complete schedule for the train/coach between Sydney and Coonabarabran.)
For international visitors, you will find information on international agents for CountryLink on their website. These are listed in the 'Bookings' section under 'International'. You are able to book the ticket through an agent closest to you.
For Australian observers, the bookings phone number within Australia is 13 22 32
The current schedule is as follows:
TO COONABARABRAN:
MONDAYS:
Sydney to Lithgow; 7.10 - 9.30
Lithgow to Coonabarabran; 10.00am - 3.25pm
TUESDAYS, WEDNESDAYS, THURSDAYS:
Sydney to Lithgow; 7.10am - 9.30am
Lithgow to Coonabarabran; 10.00am -2.50pm
FRIDAYS:
Sydney to Lithgow; 7.10am - 9.30am
Lithgow to Coonabarabran; 10.00am -2.50pm
SATURDAYS:
No Service
SUNDAYS:
Sydney to Lithgow; 4.00pm - 6.45pm

Lithgow to Coonabarabran; 7.05pm - 12.15am
TO SYDNEY:
MONDAYS
Coonabarabran to Lithgow; 12.35pm - 6.05pm
Lithgow to Sydney; 6.25pm -8.50pm
TUESDAYS, WEDNESDAYS, THURSDAYS
Coonabarabran to Lithgow; 1.00pm - 6.05pm
Lithgow to Sydney; 6.25pm - 8.50pm
FRIDAY
Coonabarabran to Lithgow; 9.50-2.50pm
Lithgow to Sydney; 3.05pm-5.45pm
SATURDAY
No service.
SUNDAY
Coonabarabran to Lithgow; 1.00pm - 6.05pm
Lithgow to Sydney; 6.25pm - 8.50pm

Top of AAO Travel

Home | Contact Us | Sitemap | Feedback | Privacy Statement

travel@aaoepp.aao.gov.au

26 October 2004

Research School of Astronomy and Astrophysics

Mount Stromlo and Siding Spring Observatories

RSAA
- Home
- About Us
- Research
- Technology
- Study@RSAA
- Observing
  - Telescope Schedules
  - Applying for Time
  - Observing Capabilities
  - Telescopes
  - Detectors
  - Computer Accounts
  - CICADA
  - Travel & Tourist info
  - SSO Accommodation Booking
  - Support & Reports
- Gemini
- Public Information
- News
- Jobs

Quick Links
- RSAA People
- Contacts
- Picture Gallery
- What's On
- RSAA Alumni

Intranet (Staff only)
- Intranet Connect
- Webmail

Search RSAA
Go

Accommodation Facilities at Siding Spring

Accommodation on Siding Spring Mountain is at The Lodge, which is run by the Australian National University. Bookings must be made well in advance of your observing run.

The current Lodge tariff is $113.30 (including GST) per day, including meals, and can be paid with a Credit Card (VISA/MASTERCARD/BANKCARD), in Australian dollar traveller's cheques or Australian currency. Personal cheques drawn on an Australian Bank are accepted.

Partners can stay at The Lodge (1 or 2 larger rooms are available) for an additional charge (this covers the extra meals and linen required). There is also a self-contained cottage on the mountain for families; enquiries about its availability should be made directly to The Lodge staff (Tel +61 2 6842 6246, Fax +61 2 6842 6252).

Lodge staff are usually able to cater for special dietary needs or preferences, but prior arrangement should be made. Alcoholic beverages may be consumed at The Lodge, but strictly on a BYO (Bring Your Own, there is a bottle shop in Coona) basis. Two meals are prepared each day - a breakfast/lunch (you choose depending on your body clock - just let the kitchen staff know the night before) and dinner served before twilight each day. There is also a small kitche available 24 hours a day where Lodge guests can make their own tea, coffee o breakfast if they choose. A "midnight lunch" is normally provided for observers to have with them at the telescopes.

The Lodge reading room has a range of recreational reading matter, including the latest newspapers. There is a small library of videos for those cloudy nights as well as a pool table and dart board in the recreation room. A washing machine, dryer and ironing board are also available.

SSO Accommodation Booking Form

This form uses JavaScript and cookies, so will not work if you have disabled either of these in your preferences. You will need to turn these features back or and reload this form.

NB: This service is for observers external to RSAA only.
RSAA staff should follow the internal admin procedure.

Please enter your name (required):

Please enter your full e-mail address (required and where the system will send auto-replies):

Is your host institution in Australia or a foreign country? (required) Australia foreign country

Please specify the name of your host institution (required):

Please enter your arrival details (required) —Time: :00 Date: - - (dd-mmm-yyyy)

Please enter your departure details (required) —Time: :00 Date: -

[ ]-[ ] (dd-mmm-yyyy)

If you are flying into Coonabarabran please give your flight details:

Do you need a taxi to pick you up from the airport? ○ Yes ◉ No

If you are flying out of Coonabarabran please give your flight details:

Do you need a taxi to take you to the airport? ○ Yes ◉ No

Do you need to book accommodation for more than yourself (e.g. partner, children)? ○ Yes ◉ No

Please enter the names of any other guests accompanying you (comma-separated):

Please specify any special requirements (i.e. dietary needs etc.):

Submit Booking Request | Reset Form

Page last updated: 08 March 2005
Please direct all enquiries to: Webmaster
Page authorised by: Director, RSAA

Research School of Astronomy and Astrophysics

Mount Stromlo and Siding Spring Observatories

RSAA
- Home
- About Us
- Research
- Technology
- Study@RSAA
- Observing
  - Telescope Schedules
  - Applying for Time
  - Observing Capabilities
  - Telescopes
  - Detectors
  - Computer Accounts
  - CICADA
  - Travel & Tourist info
  - SSO Accommodation Booking
  - Support & Reports
    - After hours help
    - Observing Guides
    - Time & Weather info
    - Observing Reports
- Gemini
- Public Information
- News
- Jobs

Quick Links
- RSAA People
- Contacts
- Picture Gallery
- What's On
- RSAA Alumni

Intranet (Staff only)
- Intranet Connect
- Webmail

Search RSAA
Go

After hours support - Siding Spring Observatory

The telescopes operated by the ANU (RSAA) at Siding Spring (2.3m, 40", 24", 16") are covered after hours by the following arrangements:

- A member of ANU SSO Technical Staff is present at the Observatory unti 18:00. This person can be contacted in the 16" dome or on the mobile phone number posted in each dome.
- A member of ANU SSO Technical Staff is on-call from 18:00-22:00 each working day. This person's name and contact details are posted in each dome.
- A member of ANU SSO Technical Staff is on-call from 11:00-23:00 at weekends and public holdays EXCEPT that there is no support during the Christmas and Easter holidays

Computer problems at the SSO telescopes are covered by an informal arrangement as follows:

- If possible, first consult with the on-site SSO Technical Staff member
- The following staff have agreed to offer advice by phone, but no later thar 10:30 pm. On occasions and at their discretion, they may offer further assistance. If problems arise before 10:30 pm, call the staff listed in the order on the following list.

  Jon Nielsen
  Greg Wilson
  Bill Roberts
  Peter Young

Page last updated: 07 January 2005
Please direct all enquiries to: Webmaster
Page authorised by: Director, RSAA