Speckle Instrument GUI – Programmers Guide

Dave Mills (rfactory@theriver.com) – February 2021

Table of Contents

1. Development Environment

The recommended development environment is Ubuntu Linux x86_64 (currently 20.04 LTS) kernel and the normal development toolchain installed. Initial development took place on a CentOS 7 system , but work was also done on Ubuntu 20. Any modern Linux variant should suffice with the caveat that the *install* script may require some changes (apt vs yum etc).

1.1 Dependencies

The following package should be installed and configured before working on the code.

- autoconf
- autogen
- bwidget
- build-essential
- default-mysql-client
- default-mysql-server
- g++ automake

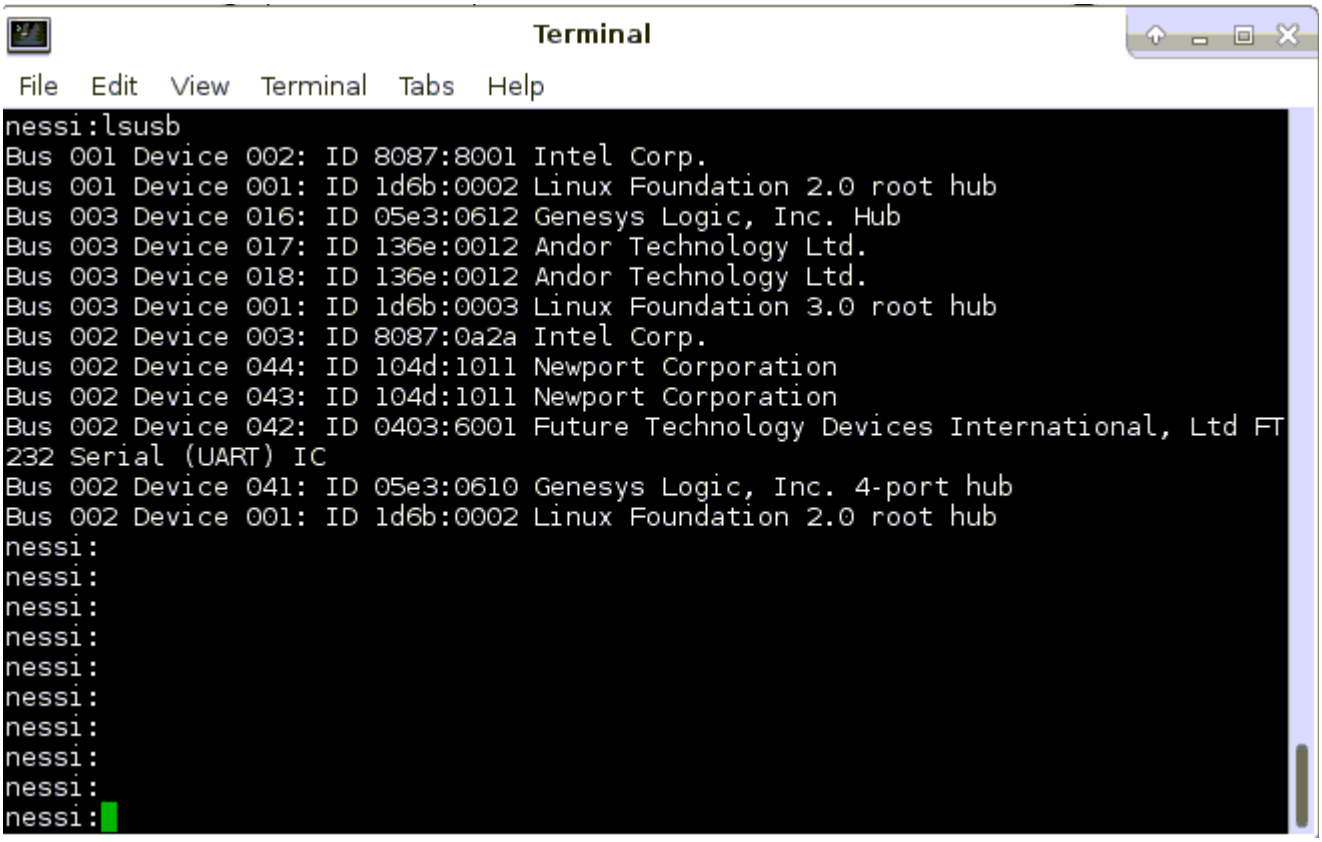
- glib-2.0-dev
- gnuplot
- imagemagick
- libcfitsio-bin
- libcfitsio-dev
- libexif-dev
- libexpat1-dev
- libfftw3-bin
- libfftw3-dev
- libgif-dev
- libglib2.0-dev
- libgsf-1-dev
- libjpeg62-turbo-dev
- libjpeg-dev
- libjpeg-dev
- libmagick++-dev
- libpng-dev
- libtiff5-dev
- libusb-1.0-0
- libusb-1.0-0-dev
- libwcs6
- libwebp-dev
- make
- mysqltcl
- pkg-config
- qfits-tools
- qfitsview
- saods9
- tcl
- tcl-dev
- tcl-fitstcl
- tk
- tk-dev
- topcat
- wcslib-dev
- xpa-tools
- zlib1g
- zlib1g-dev

The VIPS source tree is included and will need to be compiled and installed.

1.2 Hardware

A minimum of 2 USB 3.0 capable ports are required to interface the cameras.
Another 3 USB 2.0 or better ports are needed to interface the Filter Wheels and the Zaber motion control stages. All the Zaber devices are daisy-chained off a single port.

Once the system is configured , use *lsusb* to examine the device complement



The Filter Wheels show up as Newport Corporation, and the Zabers as Future Technology....
In order to control these devices as a non-root user run the script

./setDevicePermissions

located in the base folder of the Speckle software installation.

1.3 Serial Numbers

If it becomes necessary to change out either Filter Wheel or Camera components, the appropriate configuration files will need adjustment. The configuration files are in the $HOME/speckle-control directory

andorsConfiguration.[telescope]

filtersConfiguration.[telescope]

In each case the serial number information will need to be updated.

The Filter Wheel serial numbers can be found using the lsusb command

```
nessi:lsusb
Bus 001 Device 002: ID 8087:8001 Intel Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 016: ID 05e3:0612 Genesys Logic, Inc. Hub
Bus 003 Device 017: ID 136e:0012 Andor Technology Ltd.
Bus 003 Device 018: ID 136e:0012 Andor Technology Ltd.
Bus 003 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 002 Device 003: ID 8087:0a2a Intel Corp.
Bus 002 Device 044: ID 104d:1011 Newport Corporation
Bus 002 Device 043: ID 104d:1011 Newport Corporation
Bus 002 Device 042: ID 0403:6001 Future Technology Devices International, Ltd FT
232 Serial (UART) IC
Bus 002 Device 041: ID 05e3:0610 Genesys Logic, Inc. 4-port hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
nessi:
nessi:
nessi:lsusb -v -s 002:043 | grep iSerial
  iSerial                 128 061D088E010F5400
nessi:lsusb -v -s 002:044 | grep iSerial
  iSerial                 128 1B18177A01135400
nessi:
```

2. Software Architecture

The Speckle GUI and control package includes the following components

Tcl/Tk scripts for GUI and controls
Shared Libraries for Andor cameras and Filter Wheel control
Shared libraries for FITS I/O and Image processing
ds9 and XPA for Image display

When the software is running there are normally 6 processes involved

ds9red - displays the red camera images
ds9blue – displays the blue camera images
andorCameraServer 0 – controls Andor camera with id = 0
andorCameraServer 1 – controls Andor camera with id =1
gui2 – GUI and control windows
xpans – XPA protocol server

The following communication methods are used

The GUI interacts with the camera servers via sockets (2001 and 2002)
It is also possible to telnet to these sockets and send camera server commands manually.

The GUI interacts with the ds9 displays via XPA (using xpaget, xpaset programs)

The Camera servers interact with the ds9 displays via shared memory buffers and XPA

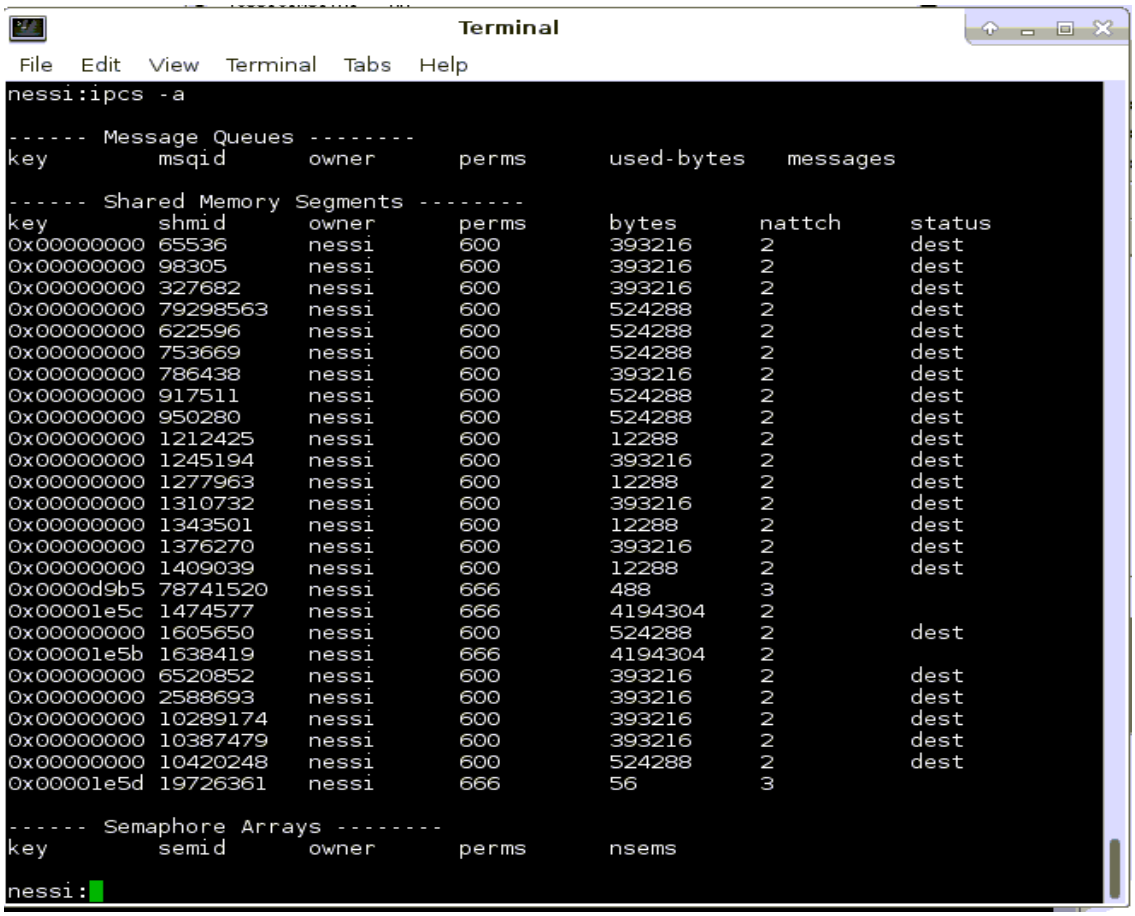
The GUI interacts with the Camera servers via a shared memory area (a small number of items are used to communicate during the high speed acquisition loops)

```
typedef struct shmControlRegisters {
  int iPeak[2];
  int iMin[2];
  int iFrame[2];
  int displayFFT;
  int displayLucky;  // not used in Speckle
  int saveLucky;
  int iabort;
  int iLuckyThresh[2];  // not used in Speckle
  int iLuckyCount[2];  // not used in Speckle
} shmControl;
```

Some prototype code is included in the Filter Wheel and Zaber scripts to facilitate their use in socket based server mode (Not yet implemented).

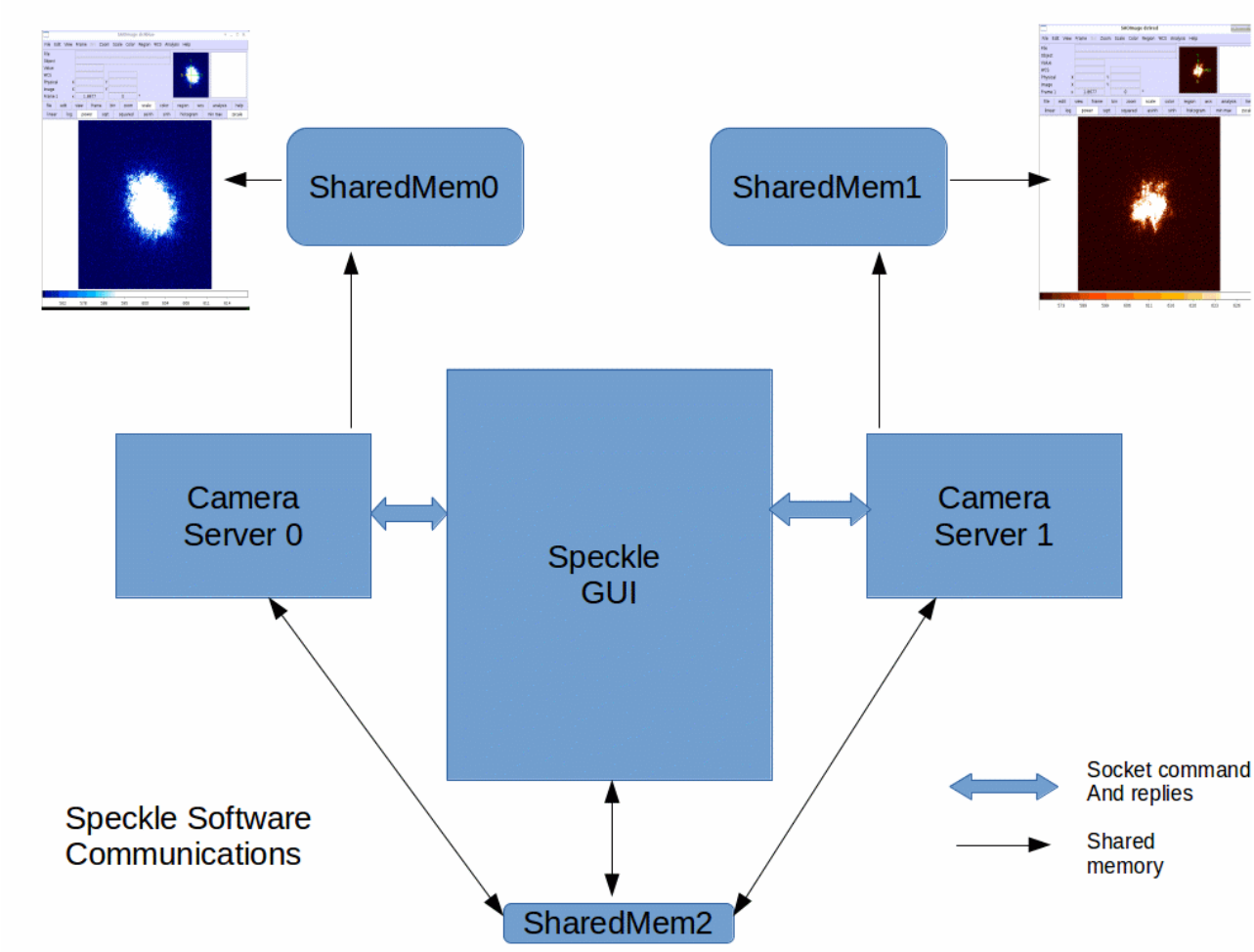
Separating the Camera low level control into servers makes it possible to reset a misbehaving camera without affecting the rest of the system.

The shared memory areas can be listed using the ipcs tool



For Speckle, the ones involved have keys 1e5b, 1e5c (the image buffers will have an nattch count of 2 during normal operations, and acquisition control area 1e5d a count of 3).

The 2 large areas contain the image buffers, each has sufficient space for a full-frame image at 4 bytes per pixel (1024 x 1024 x4).



3. Directory structure

The base directory is

speckle-control

It holds the configuration files and main control scripts.

andorsConfiguration.[telescope] – Andor camera configurations
filtersConfiguration.[telescope] – Filter wheels confgurations
picomotorConfiguration.[telescope] - Configures Pico stage defaults
powerSwitchConfiguration – Which device is where on the power strip
wcsPars.[camera].[mode].[telescope] - Configures image world coord system
zabersConfiguration.[telescope] - Configures Zaber stage defaults

andorUpgrade – Download and reinstall the Andor drivers
checkUpdates – Check for and optionally install new Speckle software (github)
cleanRestart.[telescope] – Power cycle hardware and restart software
compareVersions – compare versions of speckle-control directories
install – Build and install all software components
setDevicePermissions – Set the access mode for USB devices
setup[telescope] – Setup the environment (testing)
startspeckle2.[telescope] – Start the Speckle GUI and camera controllers

The subdirectories are

andor – the Andor camera scripting and C code
bin – executables
ccd – Utility ccd data processing library and wrappers
doc – Device and code documentation
gui-scripts – GUI and control scripts
guider – package with centroiding code
include – Include files for re-compilation of libraries
lib – shared libraries and tcl packages
oriel – Filter wheel control scripts and library
picomotor – Picomotor control scripts
share – installed by VIPS package
vips-8.5.9 – VIPS sources
zaber – Zaber stage control scripts

3.1 andor

The *andor* subdirectory contains the following

andorCameraServer.tcl – The main camera server script (2 instances of this are run)
andorCodeGen.tcl – Script to auto-generate wrappers for most Andor library calls (Set/Get)
Produces andorCreateTclCmds.c
andorGenTclInterfaces.c
andorGenTclInterfaces.h
andor.tcl – Scripts to facilitate communication with GUI
andor_tcl.c – Wrappers for data acquisition and processing, andor functions
andor_tcl.h - Wrappers for data acquisition and processing, andor functions
atcmdLXd.h – Andor library function definitions
buildlib – script to rebuild andorTclInit.so, move it to ../lib afterwards
ccd_astro.c – Utility astronomy functions
dofft.cpp – Uses VIPS to perform FFT on image data
ds9refresher.tcl – Loaded into ds9's to perform fast image refresh
examples – Andor provided example code

3.2 ccd

ccd_astro.c – misc astronomical time routines
ccd_tcl.c - ccd data frame buffer utilities
ccd.h – header for buffer utilities
ccdPackage.c – tcl package boilerplate
ccdVersion.c – tcl package boilerplate
makefile - use the make command and then cp libccd_linux.so to ../lib/libccd.so

3.3 doc

ASCII_Protocol_Manual_6.22.pdf – Zaber ASCII protocol
code – subdirectory containing doxygen code documentation
ds9-hints
iXon Ultra 888 Hardware Guide 1.0.pdf
MUSFW-100-Rev-A.pdf – Oriel filter wheel manual
NESSI operation notes.pdf
Picomotor-8742-Manual.pdf
Software Development Kit.pdf – Andor driver software developers guide
Software_Zaber Console - ZaberWiki_files
Software_Zaber Console - ZaberWiki.html
Speckle_Programmers_Guide.pdf
Speckle_with_NESSI_files
Speckle_with_NESSI.html
user-guide – Online guide for Help menu entry
user-guide.odt – User guide source
user-guide.pdf
vips-compiling-hints

3.4 gui-scripts

andorTelemetry.tcl – routines to subscribe to and use camera metadata
astrometryv2.tcl – routines to create ds9 and FITS world coordinate system information
autogain.py – routine to calculate the EMCCD gain needed
autolog – a metadata auto-logger for WIYN
camera_init.tcl – camera data init and shutdown
checkusb.tcl – check the access permissions of usb devices (deprecated)
cleanup.tcl – test REDIS server (WIYN only)
colorprint.tcl – colored text control (deprecated)
convert.tcl – coordinate system conversion
display.tcl – ds9 interaction
execengine_plugins.tcl – observing scripting engine (not used)
filechecks.tcl – sanity checks on file/directory naming and access
fits-common – FITS keyword guidelines
fits-iraf – NOAO Fits guidelines
fits-standard – FITS standard (old)
gemini_telemetry.tcl – Subscribe to and use Gemini Telescope telemetry
general.tcl – Generic routines for time, logging etc
gui2.tcl – Main GUI features
headerBuilder.tcl – Construct main FITS headers
headers.conf.gemini – Configuration files for header contents
headers.conf.wiyn
headerSpecials.tcl – Special processing for selected data types
inventory.tcl – Itemize installed hardware
mimic.gif – Mimic diagram base images
mimic-picoin.gif
mimic-picoout.gif
mimic.tcl – Create and update the Mimic diagram window
noise.tcl – Make a beep sound
observe.tcl – Data acquisition main loop control and configuration
plot3d.tcl – Graph plotting routines (deprecated)
plotaxis.tcl
plotchart.tcl
plotcontour.tcl
plotpriv.tcl
postproc.tcl – Image post processing (WIYN)
powerControl.tcl – Control the ethernet connect power strip
redisquery.tcl – Query REDIS for WIYN Telemetry
sample-header – Example FITS header
scaling.tcl – Scale graph plots (deprecated)
simwiyntlm.tcl – Simulate WIYN Telemetry
speckle_gui.tcl – More main GUI controls
speckle-icon.gif – Desktop icon
speckle-sample-header
sqltable – Create Speckle_Observations Mysql table
table.dat – Data table for checkgain function
telem-gemini.conf – Define sources of FITS header data
telem-wiyn.conf

temperature.tcl – camera temperature control and monitoring
testgui.tcl – early testing environment (deprecated)
test_telemetry – test Gemini Telemetry availability
xpak_header.tcl – Old WIYN Telemetry support (deprecated)

3.5 guider

chisqLib.c – Calculate chi squared
chisqLib.h
derotate.c – calculate field rotation
detrot.h
gauss.c – calculate gaussian centroid
guiderAppInit.c – tcl package boilerplate
guider_calc.c – calculate a variety of image centroids
guider.h
guiderPackage.c – tcl package boilerplate
guider_tcl.c – wrap the centroiding for tcl
guiderVersion.c – tcl package boilerplate
makefile - use the make command and then cp libguider_linux.so to ../lib/libguider.so
newstar.c – profile fitting routines

3.6 lib

andorTclInit.so – Camera control
girepository-1.0 – VIPS installed
libandor.so.2 – Low level camera driver
libatsifio.so.x86_64
libccd.so – image buffer management
libfitstcl.so – Tcl interface for FITS
libguider.so – image centroiding
liboriel.so – Filter wheel control
libshamrockcif.so.2 0 - Andor installed
libUSBI2C.so.2 - Andor installed
libvips.a – VIPS installed
libvipsCC.so
libvips-cpp.a
libvips-cpp.so
libvips.la
libvips.so
pkgconfig – pkg-config support for VIPS

3.7 oriel

buildlib – script to rebuid liboriel.so, move liboreil.so to ../lib afterwards
filterWheelServer.tcl – prototype server implementation
filterWheel.tcl – Filter wheel control
GenOneLinuxUSB.cpp – Low level USB driver
GenOneLinuxUSB.h
orielPackage.cpp – tcl package boilerplate

oriel_tcl.cpp – tcl wrapper for low level access
test1 – test filter move
testFilterExists.tcl – Find the filter wheels
testrepeat.cpp – test programs
testwheel1.cpp
testwheel2.cpp
testwheel.cpp
usb_constants.tcl – USB protocol details
usbpermit.tcl – Set access permissions

3.8 picomotor

picomotor-commands.txt – ascii commands for stage
picomotor.tcl – move the pico stage

3.9 vips

The VIPS source code.

3.10 zaber

usbpermit.tcl - Set access permissions
zaber-commands.abw – Zaber stage commands guide
zaber.tcl – Control and monitor the Zaber stages

4. Tcl wrappers

All the low-level C and C++ software is accessed from the tcl scripting layer by means of wrappers. The wrappers are C code which is compiled with the tcl API and the resulting library is loaded at runtime into the *wish* executable (Tcl/Tk shell program).

In order to add a new command , the following steps need to be performed

e.g. To add a command to the andor interface (compiled into andorTclInit.so)

1. Edit the file andor/andor_tcl.c to add a prototype

```
/**
 * \brief tcl_andorMyNewCommand A very useful addition
 * \param ClientData Tcl handle
 * \param Tcl_Interp interpreter pointer
 * \param argc Argument count
 * \param argv Arguments
 *
 */
int tcl_andorMyNewCommand(ClientData clientData, Tcl_Interp *interp, int argc, char **argv);
```

2. Inside the Andortclinit_Init function body, add a line to define the new tcl command

```
Tcl_CreateCommand(interp, "andorMyNewCommand", (Tcl_CmdProc *) tcl_andorMyNewCommand, NULL, NULL);
```

3. Write the code for tcl_andorMyNewCommand and add it to the file

```
/**
* Parameters are passed by Tcl and must be decoded from strings
* \param width Width of area
* \param height Height of area
 */
int tcl_andorMyNewCommand(ClientData clientData, Tcl_Interp *interp, int argc, char **argv)
{
  int width,height;

    if (argc < 3) {
     Tcl_AppendResult(interp, "wrong # args: should be \"",argv[0],"  width height\"", (char *)NULL);
     return TCL_ERROR;
    }

    sscanf(argv[1],"%d",&width);
    sscanf(argv[2],"%d",&height);
   printf "Got %d %d",width,height");
  return TCL_OK;
}
```

4. Compile and rebuild the library following the steps in *buildlib*

```
gcc -g -c -fPIC andor_tcl.c -fpic -DLINUX -DWall -g -I./include -I./include/tcl $(pkg-config --cflags vips)
g++ -g -shared -o andorTclInit.so ccd_astro.o andor_tcl.o andorGenTclInterfaces.o andorCreateTclCmds.o \
        dofft.o -L../lib -lcfitsio ../lib/libandor.so.2 ../lib/libUSBI2C.so.2 $(pkg-config --libs vips)
mv andorTclInit.so ../lib/.
```

5. Test

```
wish
load $env(SPECKLE_DIR)/lib/andorTclInit.so
andorMyNewCommand 123 456
```

6. Rebuild the code documentation

```
cd $SPECKLE_DIR/doc/code
./rebuild-code-docs
```

5 Code documentation

Detailed code documentation showing all the routines and relationships between them can be found in

$SPECKLE_DIR/doc/code/html/index.html