Speckle Instrument GUI – Programmers Guide

Dave Mills (rfactory@theriver.com) – July 2018

Table of Contents

1. Development Environment

The recommended development environment is Linux x86_64 with a 3.x series kernel and the normal development toolchain installed. Initial development took place on a CentOS 7 system , but work was also done on Ubuntu 18. Any modern Linux variant should suffice.

1.1 Dependencies

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

automake
autogen
cfitsio*
doxygen
ds9
fftw3*
gcc
g++
glib-2.0*
gmodule-2.0*
gnuplot
gobject-2.0*
libpng*
make*
mysql-client
mysql-server
mysqltcl
pangoft2*
pkg-config

python
tcl*
tk*
xpa-tools
zlib*

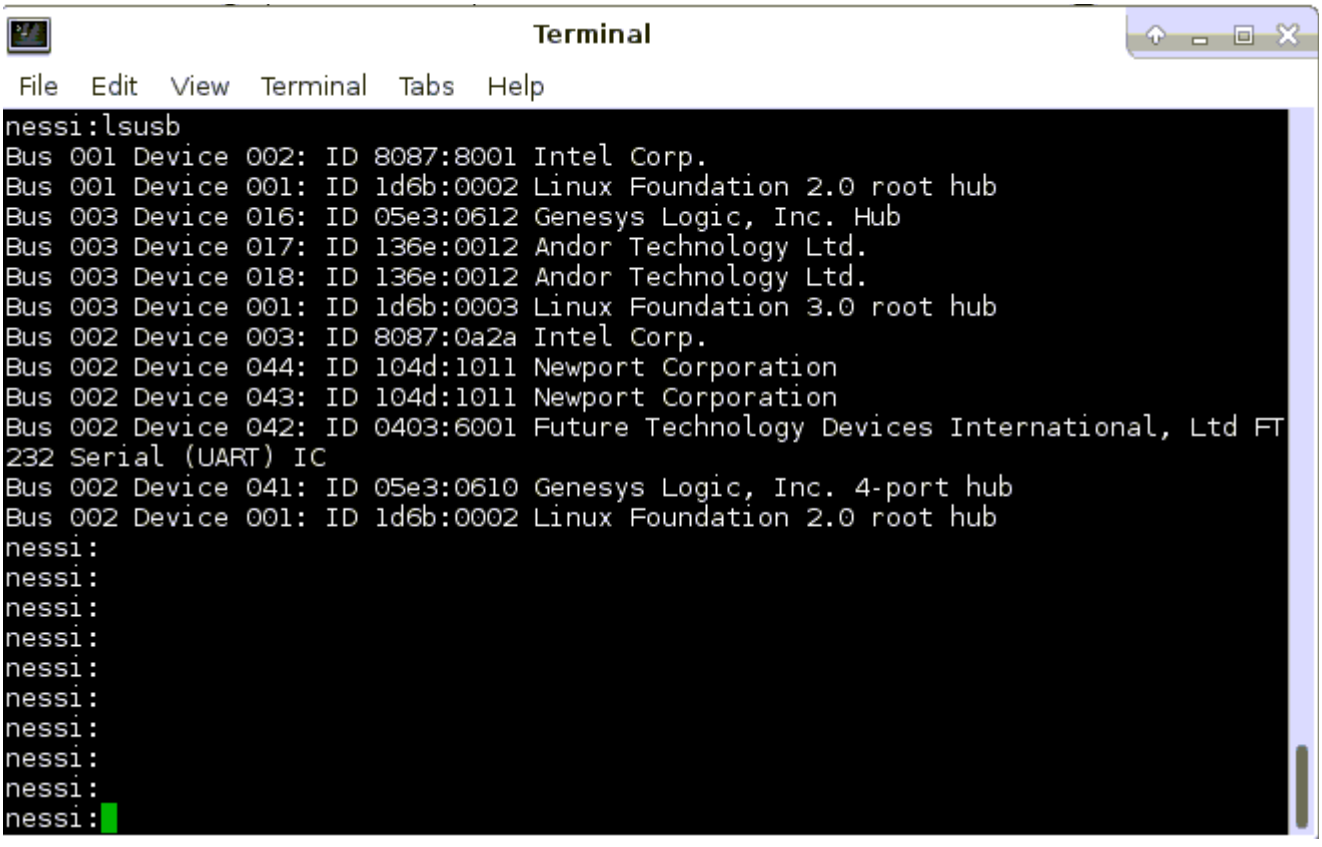
and their associated -dev or -devel packages for the ones marked with *.
The binaries and libraries for ds9, xpa-tools, tcl, tk are included in the Speckle code distribution, so those can be used as-is with a fully compatible system.

The VIPS source tree is also included and will almost certainly need recompiling in most cases.

1.2 Hardware

A minimum of 2 USB 3.0 capable ports are required to interface the caneras.
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

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 be 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:
```

The Andor Serial numbers can be found by examining the “dmesg” log at system boot time.

2. Software Architecture

The Speckle GUI and control package includes the following components

- Tcl/Tk scripts for GUI and controls
- Shared Libraries for Andor and Filter Wheel controlled
- 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)

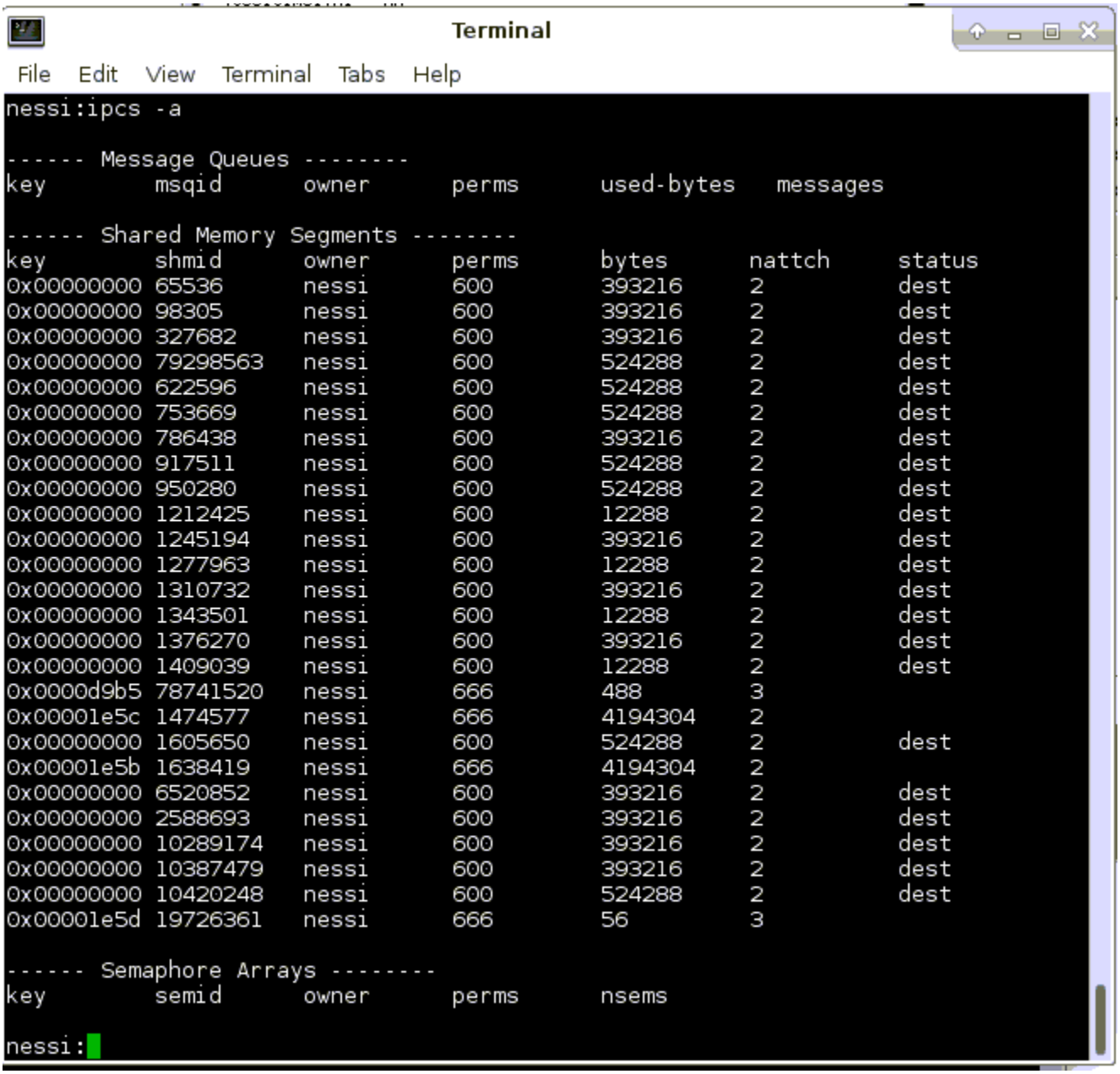
The Camera servers interact with the ds9 displays via shared memory buffers

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

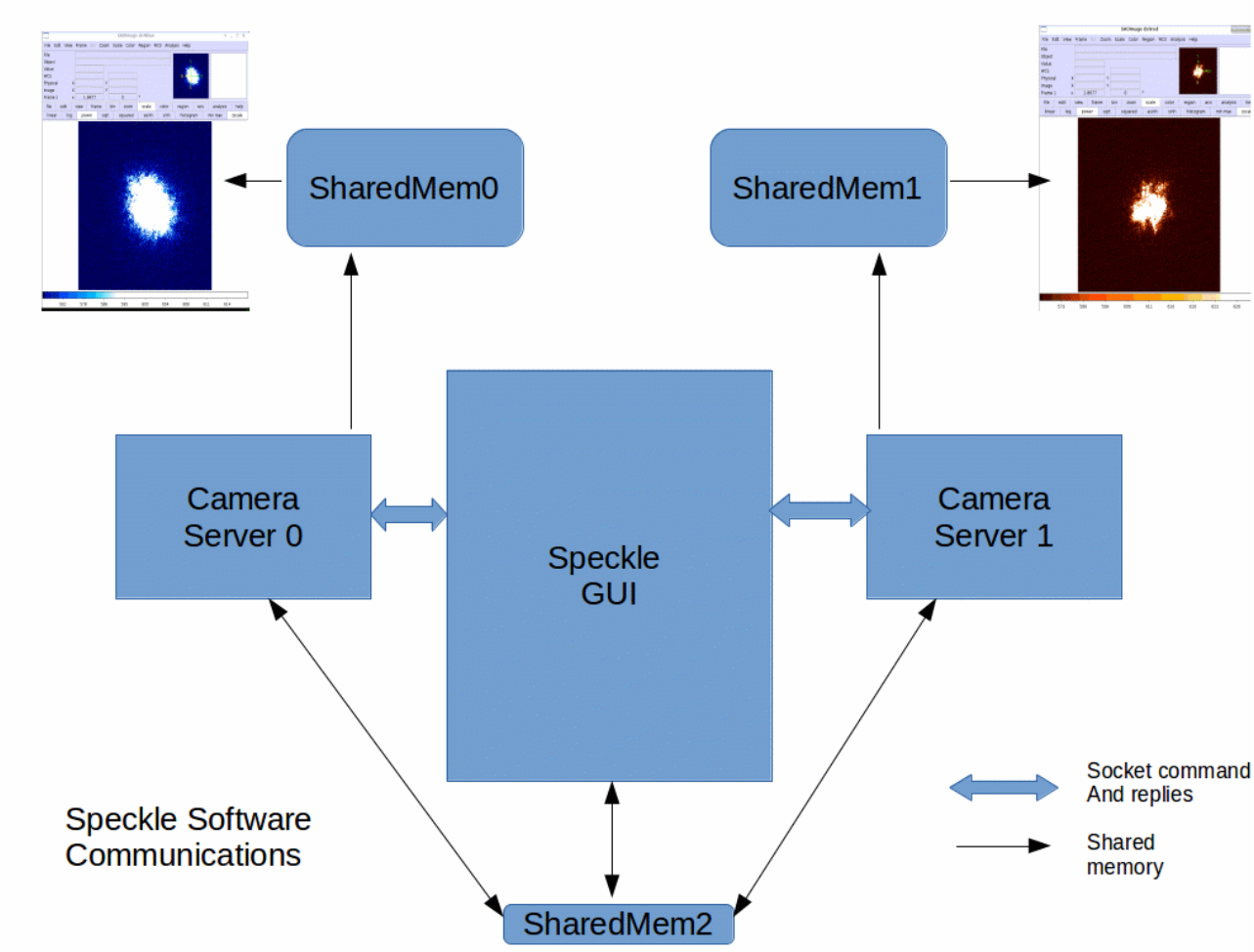
Some prototype code is included in the Filter Wheel and Zaber scripts to facilitate their use in socket based server mode (Not implemented).

Separating the Camera low level control into servers make 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, and 1e5d, and each will have an nattch count of 2 during normal operations.
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).



The 3rd small area contains a number of control and reporting variables.

```
typedef struct shmControlRegisters {
  int iPeak[2];
  int iMin[2];
  int iFrame[2];
  int displayFFT;
  int displayLucky;
  int saveLucky;
  int iabort;
  int iLuckyThresh[2];
  int iLuckyCount[2];
} shmControl;
```

3. Directory structure

The base directory is

speckle-control

It holds the configuration files, startup scripts, and gemini library archive.

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
include – Include files for re-compilation of libraries
lib – shared libraries and tcl packages
oriel – Filter wheel control scripts and library
picomotor – Pico motor control scripts
share – installed by VIPS packaged
vips-8.5.9 – VIPS sources

The contents of most of the subdirectories is self explanatory.

The andor subdirectory contains the following

andorCameraServe.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
buildandorWrap – script to rebuild andorTclInit.so
ccd_astro.c – Utility astronomy functions
dofft.cpp – Uses VIPS to perform FFT on image data-directory
ds9refersher.tcl – Loaded into ds9's to perform fast image refresh
examples – Andor provided example code

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 buildandorWrap

```
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
doyxgen -g specklecode
```

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