SLS Detectors software installation

Anna Bergamaschi, Dhanya Thattil

February 23, 2018

Contents

1 The Software Package

The SLS detectors software is intended to control the detectors developed by the SLS Detectors group. The detectors currently supported are:
MYTHEN, GOTTHARD, EIGER and JUNGFRAU.

The package provides software for the distributed system comprises of detectors, data receivers (to process detector data), and the client (to control or monitor the system). The client and data receivers can be embedded in the user's acquisitions system. Furthermore, the package also provides some tools for detector calibration.

1.1 Binaries

The complete software package is composed of several programs which can be installed (or locally compiled) depending on one's requirements:

- libSlsDetector.so, libSlsReceiver.so:
  The *slsDetector shared and static libraries*, which are necessary for all user interfaces. The *C++ API* via the class *slsDetectorUsers* (installed with the default package) or the *Python API* via the class *sls_detector* (installed with the package including Python API), which can be used from the user's acquisition software to control the detectors and the data receivers.

- sls_detector_put, sls_detector_get, sls_detector_acquire, sls_detector_help:
  The *command line interfaces*, which are provided to communicate with the detectors and data receivers using the command line.

- slsReceiver:
  The *data receiver*, which can be run on a different machine than the client, receives the data from the detector and processes it. The receiver can be configured, controlled and monitored by the client.

- slsDetectorGUI:
  The *graphical user interface*, which provides a user friendly way of operating the detectors and data receivers with online data preview.

- energyCalibrationWizard,angularCalibrationWizard:
  The *calibration wizards* to analyze the data and produce the energy or angular calibration files.

- The *virtual Detector servers* to simulate the detectors behavior. However, only control commands work, not the data acquisition itself.

2 Install Binaries via Conda

This section is useful only if one wants to download only the binaries for specific distribution and use the package via command line. Please refer later sections to download source code and compile them.

The conda package uses Travis CI for continuous integration with automatic deployment to Anaconda Cloud. One can download only the package or the package including the python interface.

- Only the package

```
#Add conda channels
conda config --add channels conda-forge
conda config --add channels slsdetectorgroup

#Install latest version
conda install sls_detector_software

#Install specific release (GLIBC2.14)
conda install sls_detector_software=3.0.1

#Scientific Linux 6 version (GLIBC2.12)
conda install sls_detector_software=SL6_3.0.1
```

- The package including Python interface

```
#Add conda channels
conda config --add channels conda-forge
conda config --add channels sls_detector

#Install latest version
conda install sls_detector

#Install specific release (GLIBC2.14)
conda install sls_detector=3.0.1

#Scientific Linux 6 version (GLIBC2.12)
conda install sls_detector=SL6_3.0.1
```

3 Install via Source Code

This section is useful if one wants to use the API and embed it in their acquisition system, or if one wants to download the source code and compile.

3.1 Download Source Code

- Only the package

```
#Clone source code with specific release
git clone https://github.com/slsdetectorgroup/slsDetectorPackage.git --branch
3.0.1
```

- The package including Python interface

```
#Clone source code with specific release
git clone https://github.com/slsdetectorgroup/sls_detector.git --branch
3.0.1
```

3.2 Requirements

These are the basic requirements to install and use the software. Fine Tuning the system will be discussed in other documentation provided.

- *C/C++*:
  The software is written in C/C++. If Python API is used, it is a wrap around to the C++ software. Any Linux installation with working libgcc should be sufficient.

- *Shared Memory*:
  Access to the shared memory of the control PC is required for the client.

- *Network*:
  The control PC communicates to the detectors and data receivers over TCP/IP. Therefore, the detector should receive a proper IP address (either DHCP or static) and no firewall should be present between the control PC and the detector.

- *Compilation*:
  cmake is required to compile. make is also possible, but is harder to find dependencies.

- *GUI*:
  To use the GUI, one requires atleast Qt4.8.2 and Qwt6.0. Installation of these are discussed in the next sections.

- *Calibration Wizards*:
  They are based on the CERN Root data analysis framework. Installation of it is discussed in the next sections.

3.2.1 Qt4 Installation for GUI

It must be installed before Qwt. A Qt version equal or higher than 4.6 is required. One can install it:

- via YUM:

  ```
  yum install qt-devel
  ```

- via download from:
  `https://download.qt.io/archive/qt/4.8/4.8.2/qt-everywhere-opensource-src-4.8.2.tar.gz`

  To install:

  ```
  > gunzip qt-everywhere-opensource-src-4.8.2.tar.gz
  > tar xvf qt-everywhere-opensource-src-4.8.2.tar
  > ./configure
  > make
  > make install
  ```

  By default Qt4 will be installed in /usr/local/Trolltech/Qt-4.8.2/.

Setup Environment
One has to ensure that `PATH` and `LD_LIBRARY_PATH` have been updated to include Qt4 install path, binaries and libraries. Confirm by executing `qmake -v` and ensuring the result points to Qt4 (not Qt3 or Qt5).

If the environment is not set up, one can add the libraries and executables to the .bashrc by adding `LD_LIBRARY_PATH` and `PATH`:

```
export QTDIR=/usr/local/Trolltech/Qt-4.8.2
export PATH=$QTDIR/bin:$PATH
export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
```

3.2.2 Qwt Installation for GUI

Before installing Qwt, one must install Qt and ensure that `QTDIR`, `LD_LIBRARY_PATH` and `PATH` point to the correct Qt4 version.

A Qwt version equal or higher than 6 is required. One can install it:

- via YUM:

  ```
  yum install qwt-devel
  ```

- via download from:
  `https://sourceforge.net/projects/qwt/files/qwt/6.0.0/qwt-6.0.0.zip/download`

  To install:

  ```
  > cd qwt-6.0.0
  > qmake
  > make
  > make install
  ```

  By default Qwt will be installed int /usr/local/qwt-6.0.0

Setup Environment
One has to ensure that `QWTDIR` and `LD_LIBRARY_PATH` have been updated to include Qwt install path and libraries.

If the environment is not set up, one can add the libraries to the .bashrc by adding `LD_LIBRARY_PATH`:

```
export QWTDIR=/usr/local/qwt-6.0.0/
export LD_LIBRARY_PATH=$QWTDIR/lib:$LD_LIBRARY_PATH
```

3.2.3 Root Installation for Calibration Wizards

The software has been developed and tested with root 5.20, but any version should work. One can download it from:

```
> svn co https://root.cern.ch/svn/root/trunk root
```

To install:

```
> cd root
> ./configure --enable-qt
> make
> make install
```

Edit your .bashrc to define the ROOTSYS enviroment variable and annd the libraries and executables to the `LD_LIBRARY_PATH` and `PATH`:

```
export ROOTSYS=/usr/local/root-5.34
export PATH=$ROOTSYS/bin:$PATH
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
```

You can also download the binaries, assuming that your linux and gcc versions match as in:

`http://root.cern.ch/drupal/content/production-version-534`

3.3 Compilation

One requires `cmake` to compile and can be done in two ways:

3.3.1 Using script cmk.sh

The script uses `cmake`. After compiling, the libraries and executables will be found in 'slsDetectorPackage/build/bin' directory. Usage: [-c] [-b] [-h] [-d HDF5 directory] [-j]

- -[no option]: only make
- -c: Clean
- -b: Builds/Rebuilds CMake files normal mode
- -h: Builds/Rebuilds Cmake files with HDF5 package
- -d: HDF5 Custom Directory
- -t: Build/Rebuilds only text client
- -r: Build/Rebuilds only receiver
- -g: Build/Rebuilds only gui
- -j: Number of threads to compile through

Some example options for compilation:
For only make: `./cmk.sh`
For make clean;make: `./cmk.sh -c`
For using hdf5 without custom dir /blabla: `./cmk.sh -h -d /blabla`
For rebuilding cmake without hdf5: `./cmk.sh -b`
For using multiple cores to compile faster: `./cmk.sh -j9`
For rebuilding only certain parts: `./cmk.sh -tg` (only text client and gui)

3.3.2 Directly using cmake

Use cmake to create out-of-source builds, by creating a build folder parallel to source directory.

```
$ cd ..
$ mkdir slsDetectorPackage-build
$ cd slsDetectorPackage-build
$ cmake ../slsDetectorPackage  -DCMAKE_BUILD_TYPE=Debug -DUSE_HDF5=OFF
$ make
```

Use the following as an example to compile statically and using specific hdf5 folder

```
$ HDF5_ROOT=/opt/hdf5v1.10.0 cmake ../slsDetectorPackage
-DCMAKE_BUILD_TYPE=Debug -DUSE_HDF5=ON
```

After compiling, the libraries and executables will be found at ‘bin‘ directory

```
 $ ls bin/
    gui_client  libSlsDetector.a  libSlsDetector.so  libSlsReceiver.a
libSlsReceiver.so    sls_detector_acquire  sls_detector_get  slsDetectorGui
sls_detector_help sls_detector_put  slsReceiver
```

4 Software Upgrade

The upgrade of the package could require an upgrade of the on-board detector server and/or firmware running on the detector as well.

4.1 MYTHEN

In such cases, the users are not expected to compile the software themselves (which would require dedicated softwares) but only to download on the detector board the programming files and/or software package provided by the SLS Detectors group.

4.1.1 Firmware

To upgrade the firmware you need either a working version of the Altera Quartus software or of the Quartus programmer, which can easily be downloaded from:
`https://www.altera.com/download/programming/quartus2/pq2-index.jsp`

Normally, installation of the software and of the driver for the USB-Blaster (provided together with the MYTHEN detector) are simpler under Windows.

Under Windows, the first time that you connect the USB-Blaster to one of your USB ports, you will be asked to install new hardware. Set the path to search for the driver to: `C:\altera\80sp1\qprogrammer\drivers\usb-blasterp` (where `C:\altera\80sp1\qprogrammer\` is assumed to be ther path where your Quartus version is installed).

1. After starting the Quartus programmer, click on Hardware Setup and in the ”Currently selected hardware” window select USB-Blaster.
2. In the Mode combo box select ”Active Serial Programming”.
3. Plug the end of your USB-Blaster WITH THE ADAPTER PROVIDED in the connector ASMI on the MCS board taking care that pin1 corresponds to the one indexed and with the rectangualr pad.
4. Click on add file and from select the programming file provided when the upgrade has been reccomended.

5. Check "Program/Configure" and "Verify".

6. Push the start button and wait until the programming process is finished (progress bar top left).

7. In case the programmer gives you error messages, check the polarity of your cable (pin1 corresponds) and that you have selected the correct programming connector.

4.1.2 On-board Software

1. Connect to the board using telnet:

```
telnet  mymcs.mydomain.com
username: root
password: pass
```

2. Kill currently running servers and ensure `/mnt/flash/root` exists.

```
killall mythenDetectorServer
ls /mnt/flash/root
#if the directory does not exist mkdir /mnt/flash/root
```

3. Transfer the provided software by ftp to the MCS.

```
ftp  mymcs.mydomain.com
username: root
password: pass
cd /mnt/flash/root
put mythenDetectorServer
quit
```

4. After pressing reset on the board, the board should reboot.

5. If the program does not correctly start

   (a) Check by using the http interface that it is started by the inittab (check that the file `/mnt/etc/inittab` ends with the line `myid2:3:once:/mnt/flash/root/mythenDetectorServer`).

   (b) If program has not started, make the program executable by telnetting to the MCS and executing:
   `chmod a+xrw  /mnt/flash/root/mythenDetectorServer`

   (c) After pressing reset on the board, the board should reboot and the acqusition program correctly start.