How to compile the Calice Software?

The CALICE Software is used for mainly the reconstruction of data and simulation (digitisation).

> The software is migrated to git now. **Please use the new repository at https://stash.desy.de/projects/CALICE.**
>
> Request access with your DESY account at it-atlassian@desy.de for pull requests.

Packages

- **calice_userlib:** contains most important codes of the software
- **calice_reco:** contains code for reconstruction of the data and simulation *(dependencies: calice_userlib, labview_converter)*
- **calice_analysis:** contains code for analysis and event display *(dependencies: calice_userlib, calice_reco)*
- **calice_sim:** contains code for digitisation of the simulation and Mokka detector models *(dependencies: calice_userlib, calice_reco)*
- **calice_calib:** contains code for calibration *(dependencies: calice_userlib, calice_reco)*
- **calice_cddata:** contain code for writing calibration/mapping to the reconstruction database *(dependencies: calice_userlib, calice_reco)*
- **labview_converter:** contain code to handle "raw" data *(dependencies: none)*
- **RootTreeWriter:** contain code to write a rootfile *(dependencies: calice_userlib, calice_reco)*
- **calice_dd_testbeams:** contain code for the DD4hep detector models *(dependencies: none)*
- **calice_cmake:** contain cmake file specific to the CALICE Software

---

Pre-compiled librairies

> Pre-compiled libraries of the CALICE Software are available on the afs.

Current HEAD compiled with modded DD4hep, gcc 4.9 and iLCSoft v02-00-01

You can find pre-compiled libraries and binaries on the cvmfs: **/cvmfs/calice.desy.de/*version*/build/myInstall**

Compiling against the cvmfs release (avoid to have to build all packages)

You can compile any processor or any sub-package from the calice software against the cvmfs release. This cuts downs the compilation time to the really necessary as there is no need to re-compile all the calice soft libraries, only your library is compiled (custom processor or sub-package of calice soft. Example: calice_sim.

> If you need to recompile calice_userlib, one may need to be careful as this is the core package!

First clone the sub-package you need or make your own directory. The directory needs to contain a CMakeLists.txt file for cmake.

> A bit of cmake knowledge is beneficial. For more details, see the cmake documentation here: https://cmake.org/documentation/

You ll need to create a .cmake file that specify few variables:

- **CMAKE_PREFIX_PATH**: contains the prefix path for all the necessary software pointing to the cmake configuration of each package
- **CMAKE_MODULE_PATH**: needs to point to cmake modules that are used in the CMakeLists.txt file
- **CMAKE_INSTALL_PREFIX**: where you want to install the binaries and libraries compiled (installation step)
- **USE_CXX11:** True if compiled with flag -std=c++11 or False
- **CMAKE_CXX_STANDARD:** If USE_CXX11 is set to False, it needs to be specified, cmake will automatically add the flag. Choice: 11 or 17

Here is for gcc4.9 and gcc8.2

You ll need to create a CMakeLists.txt file for compiling your processor. The structure of the folder should be similar to this:

```
.
 build
 CMakeLists.txt
 include
    Ahc2NoiseProcessor.hh
    ElectronSelectionProcessor.hh
    EventQuality.hh
    MIPPreSelectionProcessor.hh
    MultipleEventRejection.hh
    PionSelectionProcessor.hh
    SplitMIPCollection.hh
    T0CollectionProcessor.hh
 lib
    libTimingAnalysisProcessors.so -> libTimingAnalysisProcessors.so.1.1
    libTimingAnalysisProcessors.so.1.1 -> libTimingAnalysisProcessors.so.1.1.0
    libTimingAnalysisProcessors.so.1.1.0
 src
    Ahc2NoiseProcessor.cpp
    ElectronSelectionProcessor.cpp
    EventQuality.cpp
    MIPPreSelectionProcessor.cpp
    MultipleEventRejection.cpp
    PionSelectionProcessor.cpp
    SplitMIPCollection.cpp
    T0CollectionProcessor.cpp
 user-pro-test_x86_64_gcc48_sl6_v17-10.cmake
```

- build/ is where you want to build the processor(s) (avoid mixing build files with source code)
- include/ will have the .h/.hh files for your processor(s) (only the one you create no need to copy all includes from somewhere because you need it)
- lib/ where the processor(s) library will be installed (this is the lib that Marlin will need by adding it to the variable $MARLIN_DLL)
- src/ where the source code of your processor(s) is (***You can have several processor there that will be compiled into a single library***)
- CMakeLists.txt is used by cmake to know what to do
- user-****.cmake will contains some variable that cmake needs to know (for example where calice_userlib is (lib/include/cmake) to be able to compile against it)

Here is a simple example of a CMakeLists.txt

```
#######################################################
# cmake file for building Marlin example Package
# @author Jan Engels, Desy IT
CMAKE_MINIMUM_REQUIRED(VERSION 2.6 FATAL_ERROR)
#######################################################

# project name
PROJECT( MyProcessor )

# project version
SET( ${PROJECT_NAME}_VERSION_MAJOR 1 )
SET( ${PROJECT_NAME}_VERSION_MINOR 0 )
SET( ${PROJECT_NAME}_VERSION_PATCH 0 )

### DEPENDENCIES ############################################################

FIND_PACKAGE( ILCUTIL REQUIRED COMPONENTS ILCSOFT_CMAKE_MODULES )

# load default settings from ILCSOFT_CMAKE_MODULES
INCLUDE( ilcsoft_default_settings )
# if compiling against calice soft
# INCLUDE( calice_default_settings )

FIND_PACKAGE( Marlin REQUIRED ) # minimum required Marlin version
ADD_DEFINITIONS( ${Marlin_DEFINITIONS} )

# TO CHANGE !
# optional package
# Use FIND_PACKAGE() .i.e FIND_PACKAGE( CALICE_USERLIB ) if you need calice userlib librairies

# example ROOT
# FIND_PACKAGE( ROOT REQUIRED Core EG Minuit2 TMVA TMVAGui )
# ADD_DEFINITIONS( ${ROOT_DEFINITIONS} )

# FLAGS
# Optional flags to add for compilation modify to your needs
# SET(CMAKE_CXX_FLAGS  "-Werror -std=c++17 -pedantic -Wno-long-long -Wno-sign-compare -Wshadow -fno-strict-
aliasing ${CMAKE_CXX_FLAGS}")

### LIBRARY #################################################################

# include directories (also from the optional package) TO CHANGE!
INCLUDE_DIRECTORIES( BEFORE include ${Marlin_INCLUDE_DIRS} )
# INCLUDE_DIRECTORIES( BEFORE include ${GSL_INCLUDE_DIRS} ${ROOT_INCLUDE_DIRS} ${CLHEP_INCLUDE_DIRS}
${Marlin_INCLUDE_DIRS} ${CALICE_USERLIB_INCLUDE_DIRS} )

# add sources to the variable $SRC_FILES
AUX_SOURCE_DIRECTORY( src SRC_FILES )

ADD_LIBRARY( ${PROJECT_NAME} SHARED ${SRC_FILES} )
# If you have optional libraries TO CHANGE!
TARGET_LINK_LIBRARIES( ${PROJECT_NAME} ${Marlin_LIBRARIES} )
# TARGET_LINK_LIBRARIES( ${PROJECT_NAME} ${Marlin_LIBRARIES} ${ROOT_LIBRARIES} )

INSTALL( TARGETS ${PROJECT_NAME} LIBRARY DESTINATION lib )

# display some variables and write them to cache
DISPLAY_STD_VARIABLES()
```

To compile your package just do:

```
mkdir build
cd build
cmake -C ../<config.cmake> ..
make
make install
```

Compiling individually the packages

This is mostly useful for developers. If you only do analysis, see here.

You need to compile the packages in this order:

1. calice_userlib
2. labview_converter
3. calice_reco
4. calice_analysis
5. calice_calib
6. calice_sim
7. calice_cddata
8. RootTreeWriter

This is fairly simple. All you need is

- cmake
- ILCSoft (v01-17-10 minimum)
- git
- gcc 4.8 minimum

First prepare your working space and download the needed calice packages

```
cd <workspace>
git clone https://stash.desy.de/scm/calice/calice_cmake.git calice_cmake
git clone https://stash.desy.de/scm/calice/calice_userlib.git calice_userlib
git clone https://stash.desy.de/scm/calice/calice_reco.git calice_reco
git clone https://stash.desy.de/scm/calice/labview_converter.git labview_converter
git clone https://stash.desy.de/scm/calice/calice_analysis.git calice_analysis
git clone https://stash.desy.de/scm/calice/calice_calib.git calice_calib
git clone https://stash.desy.de/scm/calice/calice_sim.git calice_sim
git clone https://stash.desy.de/scm/calice/calice_cddata.git calice_cddata
git clone https://stash.desy.de/scm/calice/RootTreeWriter.git RootTreeWriter
```

Once this is done, create a build folder for each package

```
mkdir build_<package_name>
```

Before building, you need to create a cmake file that links all dependencies, the ilcsoft and where to install the compiled libraries. Template file: user-pro-test_x86_64_gcc48_sl6.cmake (for ILCSoft v01-17-11). You need to replace the first part of the file from `SET( ILC_HOME ... to option (Boost_NO_..... )` by the line supplied in the `ILCSoft.cmake` located on the root directory of your desired ILCSoft version (`/cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-XX`) and the variable `CALICE_HOME` by the path to your workspace.

You can start to build each packages by doing

```
cd build_<package_name>
cmake -C ../user-pro-test_x86_64_gcc48_sl6.cmake ../<package_name>
```

You should get an output similar to this one

```
loading initial cache file ../user-pro-test_x86_64_gcc48_sl6.cmake
-- Check for ILCUTIL (1.3.0)
-- FLAGS -Wall;-Wextra;-Wshadow;-Weffc++;-pedantic;-Wno-long-long;-Wuninitialized;-Wl,-no-undefined
-- Adding -Wall to CXX_FLAGS
-- Adding -Wextra to CXX_FLAGS
-- Adding -Wshadow to CXX_FLAGS
-- Adding -Weffc++ to CXX_FLAGS
-- Adding -pedantic to CXX_FLAGS
-- Adding -Wno-long-long to CXX_FLAGS
-- Adding -Wuninitialized to CXX_FLAGS
-- Adding -Wl,-no-undefined to CXX_FLAGS
-- Performing Test CXX_FLAG_WORKS_-std=c++11
-- Performing Test CXX_FLAG_WORKS_-std=c++11 - Success
-- 64 bit architecture detected
-- Check for Marlin (1.9.0)
-- Check for Marlin_LIBRARIES: Marlin
-- Check for Marlin_MARLIN_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/Marlin/v01-09/lib
/libMarlin.so -- ok
-- Found Marlin: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/Marlin/v01-09
-- Check for LCIO_LIBRARIES: lcio;sio
```

```
-- Check for LCIO_LCIO_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/lcio/v02-07-03/lib/liblcio.
so -- ok
-- Check for LCIO_SIO_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/lcio/v02-07-03/lib/libsio.
so -- ok
-- Found LCIO: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/lcio/v02-07-03/include
-- Check for LCCD (1.3.1)
-- Check for LCCD_LIBRARIES: lccd
-- Check for LCCD_LCCD_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/lccd/v01-03-01/lib/liblccd.
so -- ok
-- Found LCCD: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/lccd/v01-03-01
-- Check for CondDBMySQL (0.9.6)
-- Check for CondDBMySQL_LIBRARIES: conddb
-- Check for CondDBMySQL_CONDDB_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/CondDBMySQL/CondDBMySQL_ILC-
0-9-6/lib/libconddb.so -- ok
-- Found CondDBMySQL: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/CondDBMySQL/CondDBMySQL_ILC-0-9-6
-- Check for ROOT_CONFIG_EXECUTABLE: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/bin/root-
config
-- Check for ROOT (6.08.00)
-- Check for ROOT_EXECUTABLE: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/bin/root
-- Check for ROOT_CINT_EXECUTABLE: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/bin
/rootcint
-- Check for ROOT_LIBRARIES: Core;RIO;Net;Hist;Graf;Graf3d;Gpad;Tree;Rint;Postscript;Matrix;Physics;
MathCore;Thread;MultiProc
-- Check for ROOT_CORE_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib/libCore.
so -- ok
-- Check for ROOT_RIO_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib/libRIO.so
-- ok
-- Check for ROOT_NET_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib/libNet.so
-- ok
-- Check for ROOT_HIST_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib/libHist.
so -- ok
-- Check for ROOT_GRAF_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib/libGraf.
so -- ok
-- Check for ROOT_GRAF3D_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib
/libGraf3d.so -- ok
-- Check for ROOT_GPAD_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib/libGpad.
so -- ok
-- Check for ROOT_TREE_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib/libTree.
so -- ok
-- Check for ROOT_RINT_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib/libRint.
so -- ok
-- Check for ROOT_POSTSCRIPT_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib
/libPostscript.so -- ok
-- Check for ROOT_MATRIX_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib
/libMatrix.so -- ok
-- Check for ROOT_PHYSICS_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib
/libPhysics.so -- ok
-- Check for ROOT_MATHCORE_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib
/libMathCore.so -- ok
-- Check for ROOT_THREAD_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib
/libThread.so -- ok
-- Check for ROOT_MULTIPROC_LIBRARY: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/lib
/libMultiProc.so -- ok
-- Check for libdl.so: /usr/lib64/libdl.so
-- Found ROOT: /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/root/6.08.00/include
CMake Warning (dev) at src/CMakeLists.txt:84 (GET_TARGET_PROPERTY):
  Policy CMP0026 is not set: Disallow use of the LOCATION target property.
  Run "cmake --help-policy CMP0026" for policy details.  Use the cmake_policy
  command to set the policy and suppress this warning.
  The LOCATION property should not be read from target "userlib".  Use the
  target name directly with add_custom_command, or use the generator
  expression $<TARGET_FILE>, as appropriate.
This warning is for project developers.  Use -Wno-dev to suppress it.
-- /afs/desy.de/group/flc/pool/ebrianne/Projects/AHCAL/CaliceSoft/myInstall/lib
-- /afs/desy.de/group/flc/pool/ebrianne/Projects/AHCAL/CaliceSoft/myInstall/include
-- Found Doxygen: /usr/bin/doxygen (found version "1.6.1")
CMake Warning (dev) at /afs/desy.de/group/flc/pool/ebrianne/Projects/AHCAL/CaliceSoft/calice_cmake
/DOCUMENTATION.cmake:81 (SET):
  Cannot set "global_doc_exists": current scope has no parent.
Call Stack (most recent call first):
  CMakeLists.txt:225 (INCLUDE)
This warning is for project developers.  Use -Wno-dev to suppress it.
--
-- ----------------------------------------
--  Documentation for CALICE_USERLIB
--  CALICE_USERLIB_DOC_AUTOMAKE      = ON
```

```
--  CALICE_USERLIB_DOC_INSTALL       = ON
--  CALICE_USERLIB_DOC_WORK_DIR      = /afs/desy.de/group/flc/pool/ebrianne/Projects/AHCAL/CaliceSoft
/build_calice_userlib/doc/CALICE_USERLIB
--  CALICE_USERLIB_DOC_INSTALL_DIR   = doc/CALICE_USERLIB
--  CALICE_USERLIB_DOC_VERBOSE       = OFF
-- ------------------------------------
--
--
-- ------------------------------------------------------------------------------
-- BUILD_SHARED_LIBS                       = ON
-- CMAKE_INSTALL_PREFIX                    = /afs/desy.de/group/flc/pool/ebrianne/Projects/AHCAL/CaliceSoft
/myInstall
-- CMAKE_BUILD_TYPE                        = RelWithDebInfo
-- BUILD_WITH_BOUNDARY_CHECK               = OFF
-- BUILD_WITH_VALUE_CHECK                  = OFF
-- BUILD_WITH_CONV_DEBUG                   = OFF
-- BUILD_WITH_RECO_DEBUG                   = OFF
-- BUILD_WITH_DEBUG_CONDDB_WRITER          = OFF
-- BUILD_WITH_USE_LCCD                     = ON
-- BUILD_WITH_HAVE_ROOT                    = ON
-- LCCD_HOME                            =
-- Marlin_HOME                          =
-- BUILD_EXAMPLES                          = OFF
-- BUILD_WITH_DEPRECATED                   = OFF
-- BUILD_WITH_TRIGGER_HANDLER_IS_SINGLETON = ON
-- LCIO_HOME                               =
-- LCCD_DIR                                = /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-11/lccd/v01-03-
01
-- Marlin_HOME                             =
-- ROOT_HOME                               =
-- DB_INIT_STRING                          = flccaldb01.desy.de:calice:caliceon:Delice.1:3306
-- Change a value with: cmake -D<Variable>=<Value>
-- ------------------------------------------------------------------------------
-- Configuring done
-- Generating done
-- Build files have been written to: /afs/desy.de/group/flc/pool/ebrianne/Projects/AHCAL/CaliceSoft
/build_calice_userlib
```

Then you can build the package with `make -j4 install`.

CMake super-package to compile CALICESoft

A cmake package is available here: https://github.com/CALICETB/CaliceInstall. This super-package handles everything (downloading the package, update and building).

> The update step has been removed to avoid overwriting of files. This should be safe now.

First, download the package and modify the `user-pro-test_x86_64_gcc48_sl6.cmake` file with the correct ILCSoft version.

(In addition you might need to initialize a file that contains the path to the ILCSoft Version and compiler version (such as this: init_ilcsoft.sh - "source init_ilcsoft.sh").)

```
git clone https://github.com/CALICETB/CaliceInstall.git CaliceInstall
cd CaliceInstall
mkdir build
cd build
cmake -C ../user-pro-test_x86_64_gcc48_sl6.cmake [-DOPTIONS=...] ..
make -j4
```

Options are:

- ENABLE_C++11 [ON/OFF] to build with c++11
- BUILD_CALICE_SIM [ON/OFF] to install calice_sim package
- BUILD_CALICE_CALIB [ON/OFF] to install calice_calib package

Once it is built all the includes, binaries and librairies are available in build/myInstall.

If modifications are made in any package, one can rebuild by doing

```
cd build
make clean
make -j4
```
