# Table of Contents

# 1 Build and install dependencies

## 1.1 Install dependencies from repositories

Install `boost-devel libuuid libuuid-devel motif-devel motif-static pam-devel bzip2-devel openldap-devel java java-devel` with `dnf` on the host PC.

## 1.2 Install a newer gcc version.

In our case we are building and installing gcc 8.2.

## 1.3 Build and install Boost

We are using Boost version 1.62
On the ZynqMP:

```
cd {build_dir}
./bootstrap.sh --prefix=/usr/local
./b2 install --with=all
```

## 1.4 Build and install Xerces

We are using Xerces version 3.1.4-4
On the ZynqMP:

```
cd {build_dir}
./configure --prefix=/usr/local
make
make install
```

## 1.5 Build and install

We are using CppUnit version 1.14
On the ZynqMP:

```
cd {build_dir}
./configure --build arm
make
make check
make install
```

## 1.6 Build and install i2c-tools

```
cd i2c-tools-4.1/
make
make install
```

## 1.7 Build and install ROOT

Instructions here.
*Note: If you are compiling with gcc version> 7.1 TDAQ uses c++17 standard by default, but in ROOT you have to enable compiling with c++17 standard using the* `cxx17` *cmake flag*

# 1.8 Build and install

We are building I2C and its dependencies using yocto.

```
cd <your yocto rpm dir>
cp libzynq-log-dev-1.0-r0.aarch64.rpm libzynq-menu-dev-1.0-r0.aarch64.rpm libzynq-util-dev-1.0-r0
```

From the ZynqMP side:

```
chmod +666 /dev/i2c-*
cd ${dest_dir}
rpm -U --nodeps *.rpm
```

# 2 Build and install TDAQ

## 2.1 Checkout tdaq and tdaq common versions

On the host side

```
export PREFIX=<your_TDAQ_top_dir>
cd $PREFIX
git clone https://:@gitlab.cern.ch:8443/atlas-l1ct/zynq-tdaq-cmake.git -b zcu102
git clone https://:@gitlab.cern.ch:8443/atlas-tdaq-software/cmake_tdaq.git
export PATH=${PREFIX}/zynq-tdaq-cmake/checkout_release:${PATH}
checkout_release tdaq-common tdaq-common-03-03-00 ${PREFIX}/zynq-tdaq-cmake/checkout_release/tdaq
checkout_release tdaq tdaq-08-02-01 ${PREFIX}/zynq-tdaq-cmake/checkout_release/tdaq-packages.txt

cd ${PREFIX}/tdaq/tdaq-08-02-01
git clone ssh://git@gitlab.cern.ch:7999/atlas-l1ct/TestRCApp.git -b zcu102
cd -
```

We are using tdaq and tdaq-common version 03-03-00, adjust your paths accordingly.

## 2.2 Copy files and apply patch

```
cp ${PREFIX}/zynq-tdaq-cmake/tdaq-common/CMakeLists.txt ${PREFIX}/tdaq-common/tdaq-common-03-03-0
cp ${PREFIX}/zynq-tdaq-cmake/tdaq/CMakeLists.txt ${PREFIX}/tdaq/tdaq-08-02-01/
cd ${PREFIX}/tdaq/tdaq-08-02-01/rcc_time_stamp
patch -b -p1 <${PREFIX}/zynq-tdaq-cmake/tdaq/patches/rcc_time_stamp.patch
```

ZynqMP runs out of RAM while compiling RunControl targets. To compile them successfully add **`set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} --param ggc-min-expand=1 --param ggc-min-heapsize=262144")`** to the file **`${PREFIX}/tdaq/tdaq-08-02-01/RunControl/CMakeLists.txt`**

## 2.3 Set environment

From the zynq side:

```
export PREFIX=/root/TDAQ
export CC=gcc
export CXX=g++
export CMAKE_PREFIX_PATH=/root/TDAQ/cmake_tdaq:/root/TDAQ:${CMAKE_PREFIX_PATH}
export PATH=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.aarch64/bin:${PATH}
export JAVA_ROOT=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.aarch64
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.aarch64
export TDAQ_JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.aarch64
```

Adjust your java paths to point to your java version.

## 2.4 Build tdaq-common

```
cd ${PREFIX}/tdaq-common/tdaq-common-03-03-00
mkdir build&&cd build
cmake3 -DBINARY_TAG=aarch64-xlnx-gcc8-opt ../
make
make install
```

From the host side

```
sudo cp -r /cvmfs/atlas.cern.ch/repo/sw/tdaq/tdaq/tdaq-08-02-01/installed/share/lib/ ${PREFIX}/td
```

## 2.5 Build tdaq

```
cd ${PREFIX}/tdaq/tdaq-08-02-01/
source ${PREFIX}/ROOT/build/bin/thisroot.sh
cmake3  -DTBB_ROOT_DIR=${PREFIX}/ROOT/build \
        -DBINARY_TAG=aarch64-xlnx-gcc8-opt \
        -DCPPUNIT_LIBRARY=/usr/lib/libcppunit.a \
        -DROOT_INCLUDE_DIR=${PREFIX}/ROOT/build/include \
        -DROOT_LIBRARY_DIRS=${PREFIX}/ROOT/build/lib \
        -DROOT_BINARY_PATH=${PREFIX}/ROOT/build/bin \
        -DCMAKE_INSTALL_PREFIX=${PREFIX}/tdaq/tdaq-08-02-01/installed \
        ..
make
make install
```

For compiling you can choose the number of concurrent processes with the `-j` option. Some targets need a lot of RAM. So, the compilation will eventually crash. You can then restart it again with less processes.

# 3 Run the Run Control Application

## 3.1 Setup

From the ZynqMP side:
If you haven't yet, create a user and group same as the ones used by your host PC

```
useradd <your_host_user>
groupadd <your_host_group>
```

In `/etc/hosts` add the ip and names of host and client. In our case:

```
192.168.1.10 zcu102
192.168.1.1 pcphl1ct10.cern.ch
```

Then run:

```
sudo ip route add default via 192.168.1.1 dev eth0
```

Create a directory where the setup script is expected to be on the host PC and copy the setup script to it.

```
/cvmfs/atlas.cern.ch/repo/sw/tdaq/tdaq/tdaq-08-02-01/installed/
```

Create a symbolic link for the ipc_root.ref file pointing to the location of the ipc_root.ref in respect to the host pc:

```
mkdir -p ${NFS_PREFIX}/${PREFIX}
ln -s ${PREFIX}/ipc_root.ref ${NFS_PREFIX}/${PREFIX}/ipc_root.ref
```

To run the binaries from the zcu:

```
export CMTCONFIG=aarch64-xlnx-gcc8-opt
export TDAQ_INST_PATH=/root/TDAQ/tdaq/tdaq-08-02-01/installed/
chmod +x /root/TDAQ/tdaq/tdaq-08-02-01/installed/setup.sh
chmod +x /root/TDAQ/tdaq-common/tdaq-common-03-03-00/installed/setup.sh
source /root/TDAQ/tdaq/tdaq-08-02-01/installed/setup.sh
source /root/TDAQ/tdaq-common/tdaq-common-03-03-00/installed/setup.sh
export TDAQ_IPC_INIT_REF=file:/root/TDAQ/ipc_root.ref
chmod +777 /root/TDAQ/ipc_root.ref
```

From the Host side:
Disable firewalld and iptables

Add rsa authentication to the ZynqMP.

```
ssh-copy-id -i ~/.ssh/id_rsa.pub ppapageo@zcu102
```

Depending on the number of services that run in your PC (tdaq partitions, pmgserver, ipc etc) there is a chance that you need to increase the resource limits. We used the following values for the host PC on which both the initial and the training partitions run:

```
vim /etc/security/limits.conf

ppapageo   soft     nproc          32768
ppapageo   hard     nproc          32768
ppapageo   soft     nofile         65535
ppapageo   hard     nofile         65535
```

From the host side:

```
git clone ssh://git@gitlab.cern.ch:7999/atlas-l1ct/muctpi-test-partition.git /home/ppapageo/repos
export TDAQ_INST_PATH=/cvmfs/atlas.cern.ch/repo/sw/tdaq/tdaq/tdaq-08-02-01/installed/
export CMTCONFIG=x86_64-centos7-gcc8-opt
source ${TDAQ_INST_PATH}/setup.sh
cd muctpi-test-partition/installed
source ./setup.sh
export TDAQ_IPC_INIT_REF=file:${NFS_PREFIX}/root/TDAQ/ipc_root.ref
chmod +777 ${NFS_PREFIX}/root/TDAQ/ipc_root.ref
```

Create the directories where the setup script is to be expected on the host machine, and copy there the setup script.
From the ZynqMP side:

```
mkdir -p /cvmfs/atlas.cern.ch/repo/sw/tdaq/tdaq/tdaq-08-02-01/installed
```

From the host PC side:

```
cd muctpi-test-partition
cp zynq_setup/setup.sh ${NFS_PREFIX}/cvmfs/atlas.cern.ch/repo/sw/tdaq/tdaq/tdaq-08-02-01/installe
```

# 3.2 Edit the database

```
dbe -f Training/partitions/part_training.data.xml
```

- Add pcphl1ct10.dyndns.cern.ch and zcu102 on Computers
- Edit Partition initial and part_training "DefaultHost" to read `pcphl1ct10.dyndns.cern.ch`.
- Edit Segment MyDetector "Hosts" to `pcphl1ct10.dyndns.cern.ch`.
- Edit TrainingRCApplication rc_app_2 "RunsOn" to `zcu102`.
- Edit Partition part_training "LogRoot" to a path visible to both computers (e.g. nfs mounted)
- Create the following at SW_Repository and edit their installation paths:
  - BOOST_EXTERNAL_Zynq e.g. `/usr/local`
  - USR_EXTERNAL_Zynq e.g. `/usr/local/lib64`
  - CXX_EXTERNAL_Zynq e.g. `/usr/local/lib64`
  - I2C_EXTERNAL e.g. `/usr//lib`
  - TBB_EXTERNAL_Zynq e.g. `${PREFIX}/ROOT/build/lib`
- Add the aarch64-xlnx-gcc8-opt to Tag if it does not exist. Then add the tag to the above Software Repositories, as well as to the ones below:
  - Training_SW_zynq
  - TDAQ Common Zynq
  - ROOTZynq

For the installation paths to work a directory named after the tag is expected. In our case we are either creating symbolic links which point to their parent directory or a direcory named after the tag where a lib symbolic link points to the orgininal direcory. For our installation paths this is achieved as follows:

```
ln -s /usr/local /usr/aarch64-xlnx-gcc8-opt

cd /usr/local
ln -s . aarch64-xlnx-gcc8-opt

cd /usr/local/lib64
mkdir aarch64-xlnx-gcc8-opt
cd aarch64-xlnx-gcc8-opt
ln -s /usr/local/lib64 lib

cd /usr/local/lib
```

```
mkdir aarch64-xlnx-gcc8-opt
cd aarch64-xlnx-gcc8-opt
ln -s /usr/local/lib lib

cd /usr/lib64
mkdir aarch64-xlnx-gcc8-opt
cd aarch64-xlnx-gcc8-opt
ln -s /usr/lib64 lib

cd ${PREFIX}/ROOT/build/lib
mkdir aarch64-xlnx-gcc8-opt
cd aarch64-xlnx-gcc8-opt
ln -s ${PREFIX}/ROOT/build/lib lib
```

## 3.3 Run the application

```
setup_daq daq/segments/setup-initial.data.xml initial -ng
setup_daq -p part_training -d Training/partitions/part_training.data.xml
```

at this point do an `ipc_ls`, and if there is no pmg server running on the ZynqMP, start it manually (you have to source the binaries).

## 3.4 Cleanup

```
yes|pmg_killall_everywhere ; pkill ipc_server ; pkill pmgserver; pkill pmglancher;ssh zcu102 "pki
```

Then verify that everything is dead:

```
ps aux |grep pmg; ps aux |grep ipc; ps aux |grep -v root|grep afs; ssh zcu102 "ps aux |grep pmg";
```

## 3.5 Miscellaneous

If at any point you cannot access the state buttons on the GUI:

- Make sure Access Control is on Control and not on Display
- If you cannot switch to Control, make sure to delete the /tmp/tdaq-08-02-01 directory, where the lock files are stored

You have to run

```
chmod +666 /dev/i2c-*
ip route add default via 192.168.1.1 dev eth0
```

each time you reboot the ZynqMP

-- PanagiotisPapageorgiou - 2019-03-27

---

This topic: SystemOnChip > BuildTDAQForZynqMP
Topic revision: r13 - 2019-09-20 - unknown