Slow-Control preparation for ARICH and TRG

Yun-Tsung Lai

KEK IPNS

ytlai@post.kek.jp

Belle II Trigger/DAQ workshop 2022 @ Nara Women's University

30th Nov., 2022

Outline

- Only the detector-specific upgrades are discussed today.
  - Not for conf db, nsm2cad, HLT, ttdb, data comparison, etc.

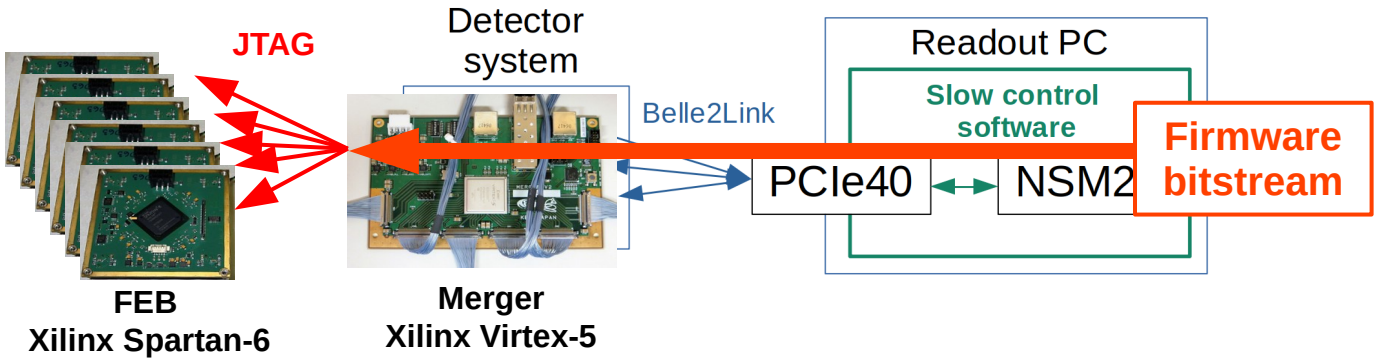
- Upgrade for ARICH
  - BOOT function
  - RCState related issue
  - Threshold scan software

- Upgrade for TRG
  - Firmware update
  - CSS GUI
  - Misc.

- Summary

BOOT function for ARICH

- ARICH system: 5~6 FEB → 1 Merger → Belle2Link → PCIe40
  - JTAG of FEB is controlled by Merger.
- ARICH uses Belle2Link to transfer an entire bitstream file to Merger.
  - Merger downloads the firmware to FEBs via JTAG and configure FEBs.
- Original Copper readout: 4 Mergers → 1 Copper.
  - Each Merger processed one-by-one.
  - Consumed time: ~1.5 min.
- PCIe40 readout: 36 Mergers → 1 PCIe40.
  - Parallel slow control processes.
  - The same consumed time: ~1.5 min.



RCState related issue

- In ARICH slow control software, both BOOT and LOAD take long time (> 10 s).
  - In the beginning of the software preparation, we always saw RC FATAL while BOOTING or LOADING.

- Reason:
  - While BOOTING/LOADING takes too long time, it returns timeout while getting RCState.
    - "cannot get rcstate".
  - Timeout repeats → FATAL.

- Solution:
  - Repeat to get rcstate when timeout (only for ARICH).

RCState related issue: the first fix

- **checkLinkNodes() of pcie40controld**:
  loop over all pcie40linkd to get the rcstate, and determine its own state.
  - This parameter is only used in ARICH.

```
void Pcie40controlCallback::checkLinkNodes(int checklinks_num_timeout)
{
  bool hasunknown = false;
  bool haserror = false;
  bool hasfatal = false;
  for(std::vector<pcie40link>::iterator link = m_links.begin(); link != m_links.end(); ++link) {
    if (link->enable) {
      RCNode node(link->nodename);
      std::string rcstate = "UNKNOWN";

      int timeout_count = 0;
      while (timeout_count <= checklinks_num_timeout){
        try {
          get(node, "rcstate", rcstate);
        } catch (const TimeoutException& e) {
          timeout_count++;
          LogFile::debug("Cannot get rcstate of %s : %s within %d sec", node.getName().c_str(), e.what(), timeout_count*5);
          if (timeout_count > checklinks_num_timeout) {
            LogFile::error("Cannot get rcstate of %s : %s within %d sec", node.getName().c_str(), e.what(), timeout_count*5);
          }
          continue;
        }
        break;

      }

      // LogFile::debug("%s  %s", node.getName().c_str(), rcstate.c_str());
      if (rcstate == "UNKNOWN") {
        hasunknown = true;
      } else if (rcstate == "FATAL") {
        hasfatal = true;
      } else if (rcstate == "ERROR") {
        haserror = true;
      }
    }
```

Just repeat to get it

RCState related issue: Other problem

- With the first fix, the timeout→FATAL problem should be gone.
  - But there was new problem.

- If some of the channels are masked, during BOOTING:
  - Used ones: BOOTING takes time, so keeps waiting.
  - Masked ones: NOTREADY.

- In the loop of checkLinkNodes(), when it goes to the masked ones:
  - Get NOTREADY.
  - Then pcie40controld just goes to NOTREADY,
    while other used ones are still in BOOTING.

- Conclusion:
  - Bug in checkLinkNodes(): It doesn't make summary of all channels.

RCState related issue: the second fix

- The function has been updated to summarize all the links' rcstate:
  https://stash.desy.de/projects/B2DAQ/repos/daq_slc/pull-requests/472/commits/be76db3c21d9935e602eb61a33a7369fd85cbaca
  - rcstate is determined only when all the links are looped.
  - Timeout on 1 of them: repeat to get its rcstate.

- FATAL/UNKNOWN/ERROR: First priority (when anyone is).
- (NOT)READY: When all are (NOT)READY.
- BOOTING/LOADING: When any one is.
- Else: Stay with original state.

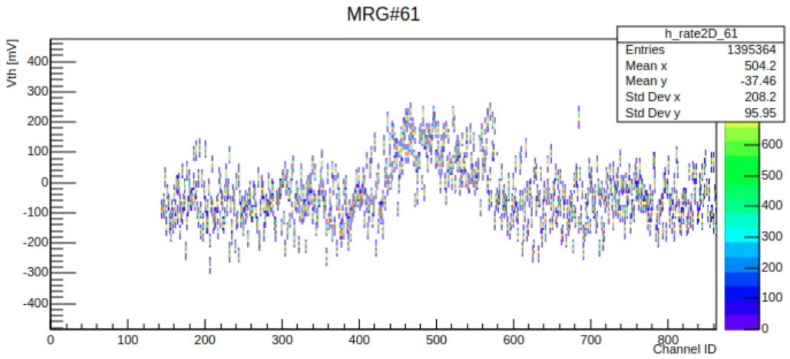
```
  if (hasunknown) {
    setState(RCState::FATAL_ES);
  } else if (hasfatal) {
    setState(RCState::FATAL_ES);
  } else if (haserror) {
    setState(RCState::ERROR_ES);
  } else if (N_validch == N_notready) {
    setState(RCState::NOTREADY_S);
  } else if (N_validch == N_ready) {
    setState(RCState::READY_S);
  } else if (N_booting > 0) {
    setState(RCState::BOOTING_RS);
  } else if (N_loading > 0) {
    setState(RCState::LOADING_TS);
  }
}
```

- For BOOTING/LOAD, we actually "cannot get rcstate".
  - So we have to set it as BOOTING/LOADING at first.
  - Then, during BOOTING/LOADING:
    Stuck at repeating to get rcstate.
    User will still see the BOOTING/LOADING flag.

```
void Pcie40controlCallback::boot(const std::string& opt, const DBObject& obj)
{
  maskUnusedChannels();
  setState(RCState::BOOTING_RS);
  distribute(NSMMessage(RCCommand::BOOT), false);
  LogFile::debug("Boot done");
  checkLinkNodes(m_checklinks_num_timeout);
}
```

Update on threshold scan software

- ARICH threshold and offset scan: Parameter writing to each board, then local run.
  - Heavy-loaded request to B2L.
    As daemon process (ARICHPcie40FEE::monitor()) is always running to synchronize FPGA and NSM →confilct/busy.



- Sometimes failure (shift) in the scan:
  - Failure in B2L register writing to FEB?
  - In ARICH, B2L→Merger→FEB has its own special protocol.

- 1st fix:
  - Writing→readback→repeat if bad.
  - Even readback could fail.

```
[2022-06-15 21:08:44] [DEBUG] Wrong value of 81 at trial 0: 40, 1. Try again.
[2022-06-15 21:08:44] [DEBUG] Wrong value of 82 at trial 0: 8, 4. Try again.
[2022-06-15 21:08:44] [DEBUG] Wrong value of 84 at trial 0: 1, 3. Try again.
[2022-06-15 21:08:44] [DEBUG] Wrong value of 84 at trial 1: 1, 3. Try again.
[2022-06-15 21:08:44] [DEBUG] Wrong value of 84 at trial 2: 4f, 3. Try again.
```

- 2nd fix:
  - Use a new nsm variable to skip "ARICHPcie40FEE::monitor()" function.
    Daemon B2L activity will be stopped.
  - If we need to do many requests of B2L R/W.
  - Result: The readback failure never happens again.
    ARICH scan results all look good.
  - Thanks to Harsh for this idea!

```
void ARICHPcie40FEE::monitor(RCCallback& callback, B2LINK& b2link)
{
  std::string vname = StringUtil::form("arich[%d].", b2link.get_link()+baseid);

  // if stop_monitor is 1, do nothing in this monitor() function.
  int stop_monitor = 0;
  callback.get(vname + StringUtil::form("stop_monitor"), stop_monitor);
  if (stop_monitor) {return;}
```

Upgrade for TRG

- Slow control for TRG system is not using Belle2Link, so no update is required.

- TRG has 2 UT boards, 4 types of transceivers, and various firmwares for trigger logic.
  - Belle2Link connection and data validation with PCIe40 has to be checked for each one.
  - The auto-reset function was not implemented for Virtex-6 GTH: 3D and NN.

- Difficulty: Re-compilation of 3D and NN firmwares.
  - Especially, it took several months to re-compile 3D, where small changes in 3D firmware codes are needed to improve timing condition.

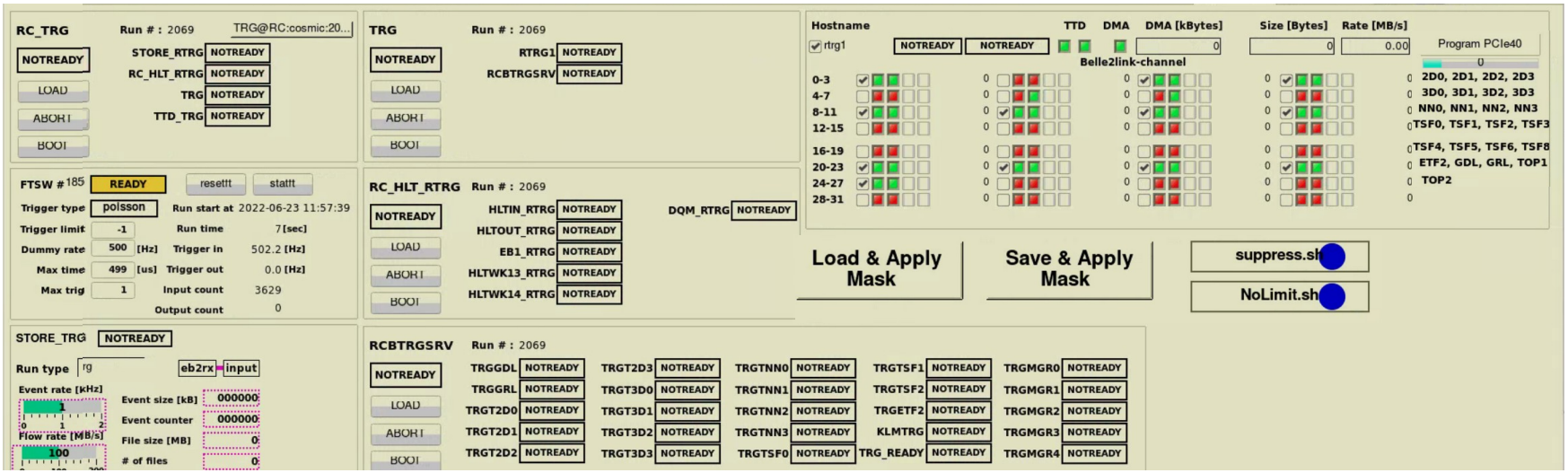
TRG module	Board	Transceiver
2D tracker (x4)	UT4	UltraScale GTY
3D tracker (x4)	UT3	Virtex-6 GTH
Neural 3D tracker (x4)	UT3	Virtex-6 GTH
Event Timing Finder	UT4	UltraScale GTY
Track Segment Finder (x9)	UT4	UltraScale GTY, GTH
Global Reconstruction Logic	UT3	Virtex-6 GTX
Global Decision Logic	UT3	Virtex-6 GTX
TOP Trigger (x2)	UT3	Virtex-6 GTX

UT3
Xilinx Virtex-6
GTX, GTH

UT4
Xilinx UltraScale
GTH, GTY

CSS GUI or TRG

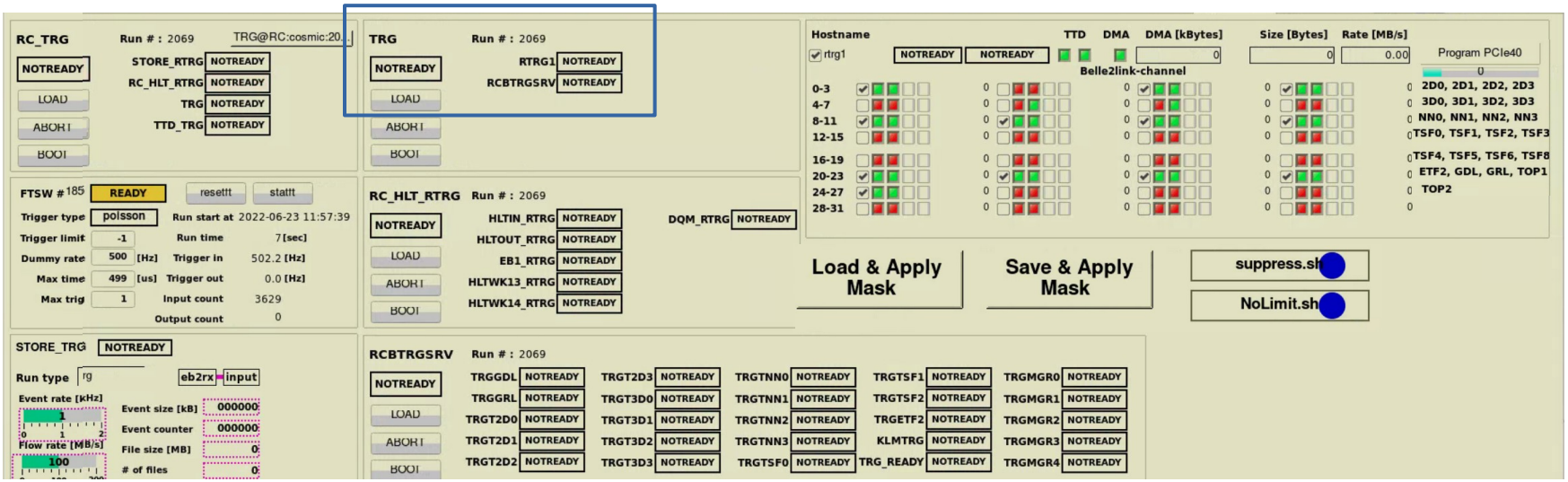
- CSS GUI for TRG:



Minor problem of stucking in CONFIGURING

- For TRG, if we use the masking button, it will be stuck in CONFIGURING state in the end.
  - Need to do ABORT manually.
- Reason: RCBTRGSRV node
  - We can just exclude this node while using the masking buttons.
  - It is in TRG's local repository.
- Qi-Dong already prepared a fix for SVD.
  - Need to ask TRG people to include it in their local repository.

Id	Node	State
RC	TRG	CONFIGURE
01	RTRG1	NOTREADY
02	RCBTRGSRV	NOTREADY



A request from TRG people

- Additional labels for maskdb:
  - Every time we do "Save & Apply Mask", a new maskdb will be created.
    maskdbtrg:pcie40:XX
  - TRG people want to have a "label" for maskdb:
    maskdbtrg:**physics**:pcie40:XX
    maskdbtrg:**highrate**:pcie40:XX
    maskdbtrg:**test**:pcie40:XX
    such that they can select different configuration to save or load.

```
[b2trg@rtrg1 ~]$ daqdblist maskdb maskdbtrg:pcie40
-----------------------------------------------------------
 id   | table       | name                  | date
-----------------------------------------------------------
 1423 | maskdb_2022 | maskdbtrg:pcie40:58 | 23/06 12:22:58
 1422 | maskdb_2022 | maskdbtrg:pcie40:57 | 23/06 12:06:49
 1421 | maskdb_2022 | maskdbtrg:pcie40:56 | 23/06 11:51:49
```

Summary

- ARICH PCIe40 has been ready from the beginning of 2022 and been stable so far.
  - No problem and no further plan for now.
  - For slow control and device's conf db, it might a bit difficult to maintain.
    - Documents have been prepared.
    - If ARICH people or DAQ group's liaison encounter problem in this part, please contact me.

- TRG PCIe40 has been ready from 2022 June.
  - Necessary firmware re-compilation has been done.
  - No update for slow control software.
  - Commissioning using cosmic run and high rate test have been done.
  - Just few minor updates might be needed.