Debug of B2L

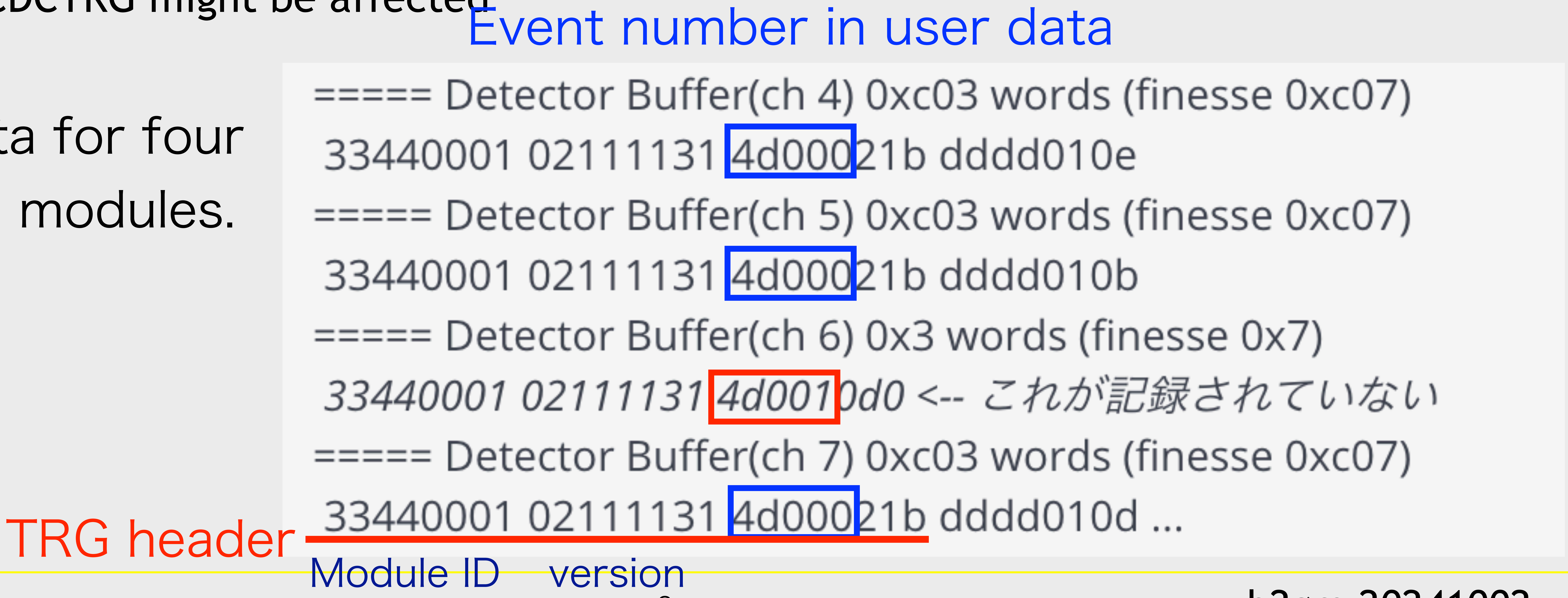
H. Nakazawa (NTU)

20241002@B2GM

Symptom; event shift

- We record CDCTRG data content with suppress factor of 256, namely only when the event number is multiple of 0x100.
  - Only TRG header info is recorded for remaining 255 events
- Event shift happened during data taking, thus the content-recorded event is shifted
- All UT4 CDCTRG might be affected

Data for four 3D modules.



```
===== Detector Buffer(ch 4) 0xc03 words (finesse 0xc07)
 33440001 02111131 4d00021b dddd010e
===== Detector Buffer(ch 5) 0xc03 words (finesse 0xc07)
 33440001 02111131 4d00021b dddd010b
===== Detector Buffer(ch 6) 0x3 words (finesse 0x7)
 33440001 02111131 4d0010d0 <-- これが記録されていない
===== Detector Buffer(ch 7) 0xc03 words (finesse 0xc07)
 33440001 02111131 4d00021b dddd010d ...
```

Symptom; Corrupted Data

```
[2024-03-19 09:37:32] [FATAL] RTRG1 : rtrg1 ch=3 : ERROR_EVENT : CORRUPTED DATA: Different event number over channels. Event_number(12272583) is different from other channel(eve 0). Exiting...: exprun 0x0788ea00
[2024-03-19 09:37:32] [DEBUG] state transit : RUNNING >> ERROR
```

- Happened when the global run stopped and started without ABORT
- Looks event number was not cleared for the problematic board when the next run started

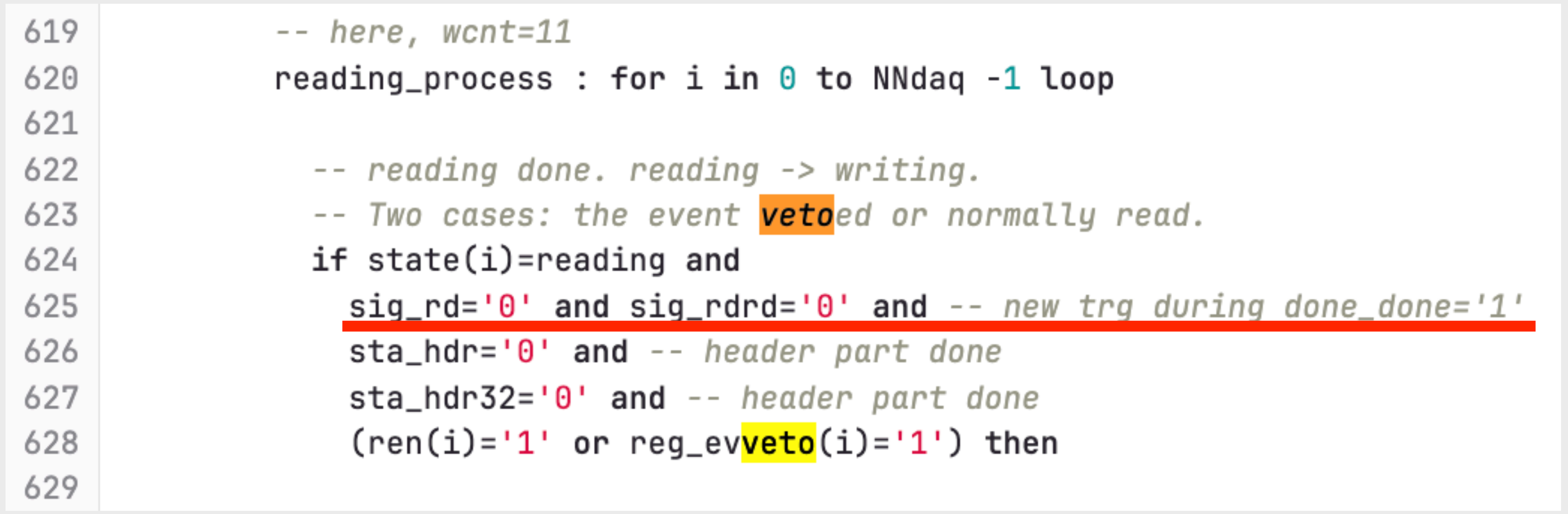
Readout protocol signals

Writing for waiting suppressed events
L1 signal
Writing for unsuppressed events
sig_trgin
b2l_we
sig_hdrwe
sta_hdr
cnt_hdr[5:0]
sig_b2lout[7:0]
tflag
rennum[4:0]
nreadingvn
nreadingvp
reading_done
reading_done_done
itarget[4:0]
0
0
0
0
00
00
0
00
0
0
0
1
01
00
01
02
Writing Done
Busy by B2L
Writing done during busy

- When a writing process is done, the next writing can start only after busy signal (150nsec) is negated.
- But when the next L1 comes during busy signal for the last event of the waiting queue, because of the missing “start next writing if not busy” condition, the event buffer for the L1 is released.

Debug

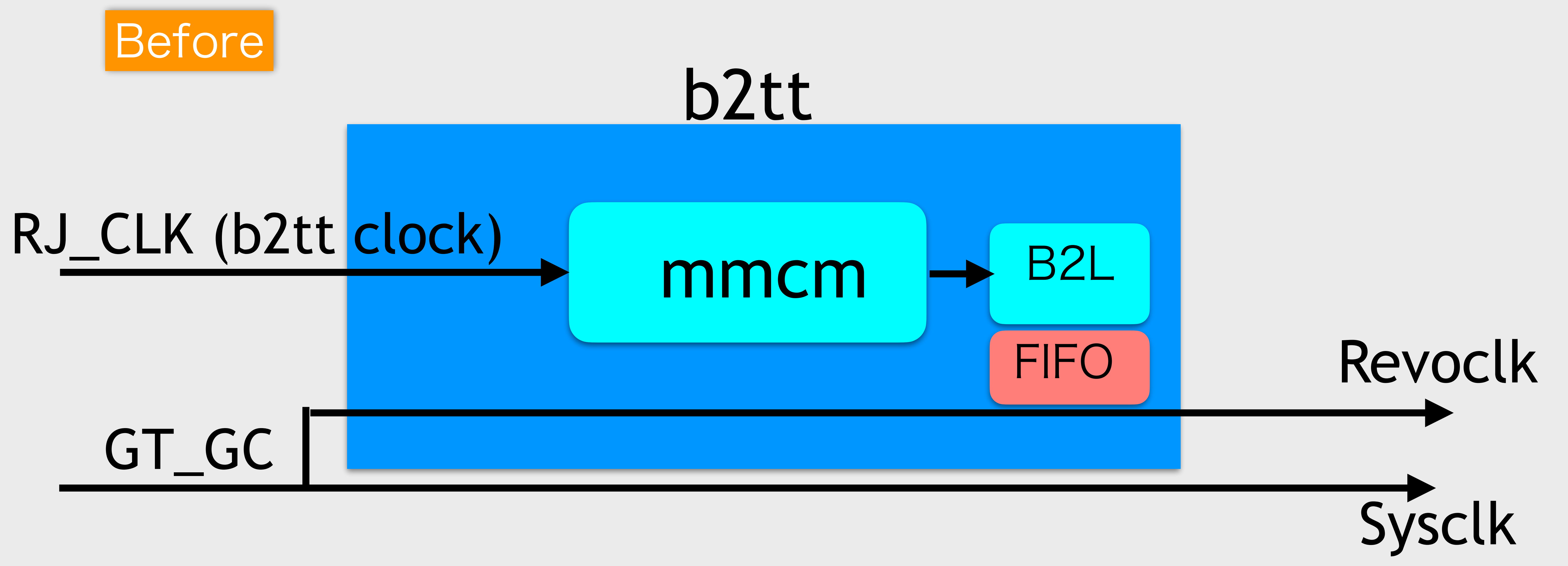
- Added on line to Library/libut4_02/B2L/src/CB/cb_top_32MHz.vhd
- Corresponding file for gdl, cb_top_32MHz_gdl.vhd, ok (the line has existed)
- The same file for UT3 ok (the line has existed)
- Maybe I used very old cb_top_32MHz.vhd when I started making UT4 version of B2L.

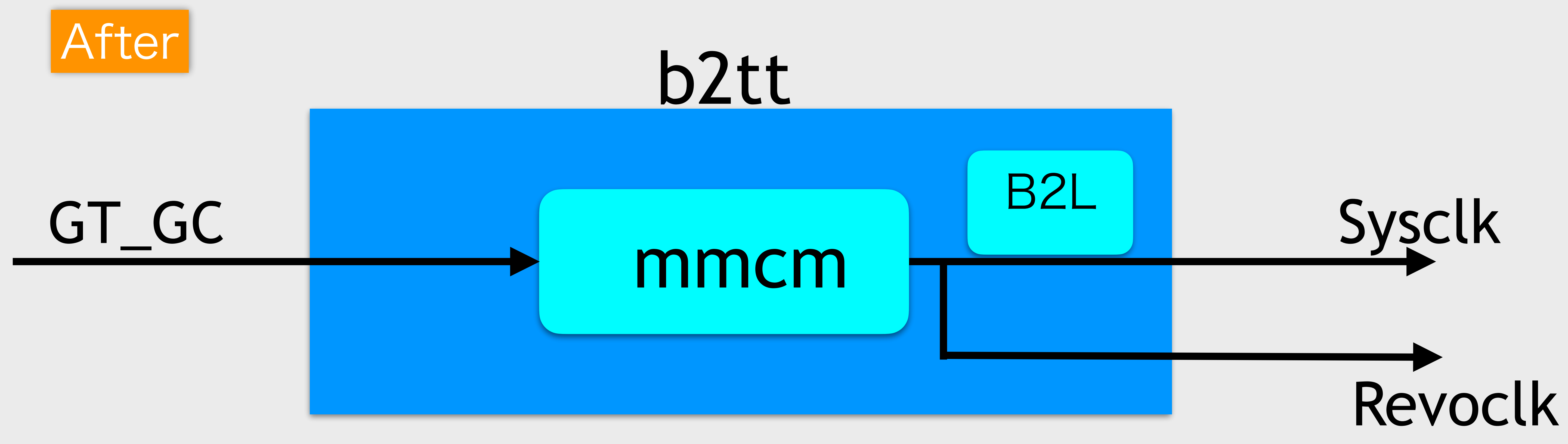


```
        -- here, wcnt=11
        reading_process : for i in 0 to NNdaq -1 loop

          -- reading done. reading -> writing.
          -- Two cases: the event vetoed or normally read.
          if state(i)=reading and
            sig_rd='0' and sig_rdrd='0' and -- new trg during done_done='1'
            sta_hdr='0' and -- header part done
            sta_hdr32='0' and -- header part done
            (ren(i)='1' or reg_evveto(i)='1') then

```

One more improvement (clock unification)

- For CDCTRG modules, different clock sources are used for sysclk and B2L
- Unified clock source (GDL,ECLTRG) look stable

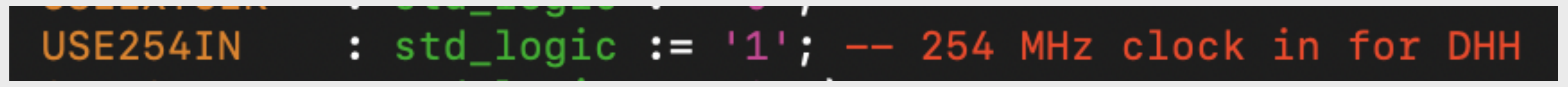




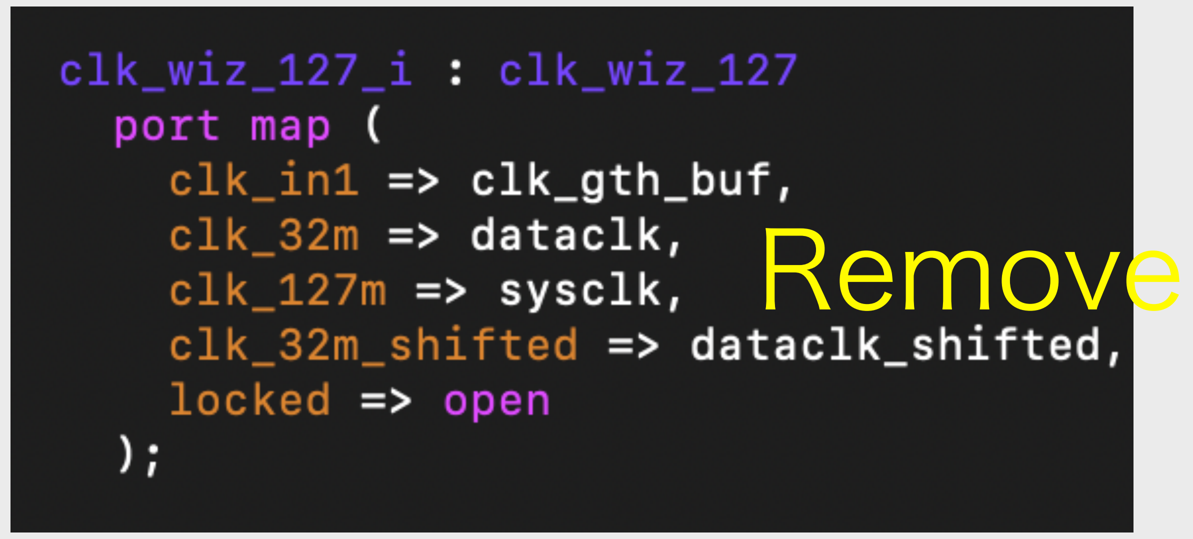
One more improvement (clock unification)

- TOP.vhd
  - Sysclk from b2l.clk127m
  - CLKGT?_GC_P/N to b2l.rj?clkp/n

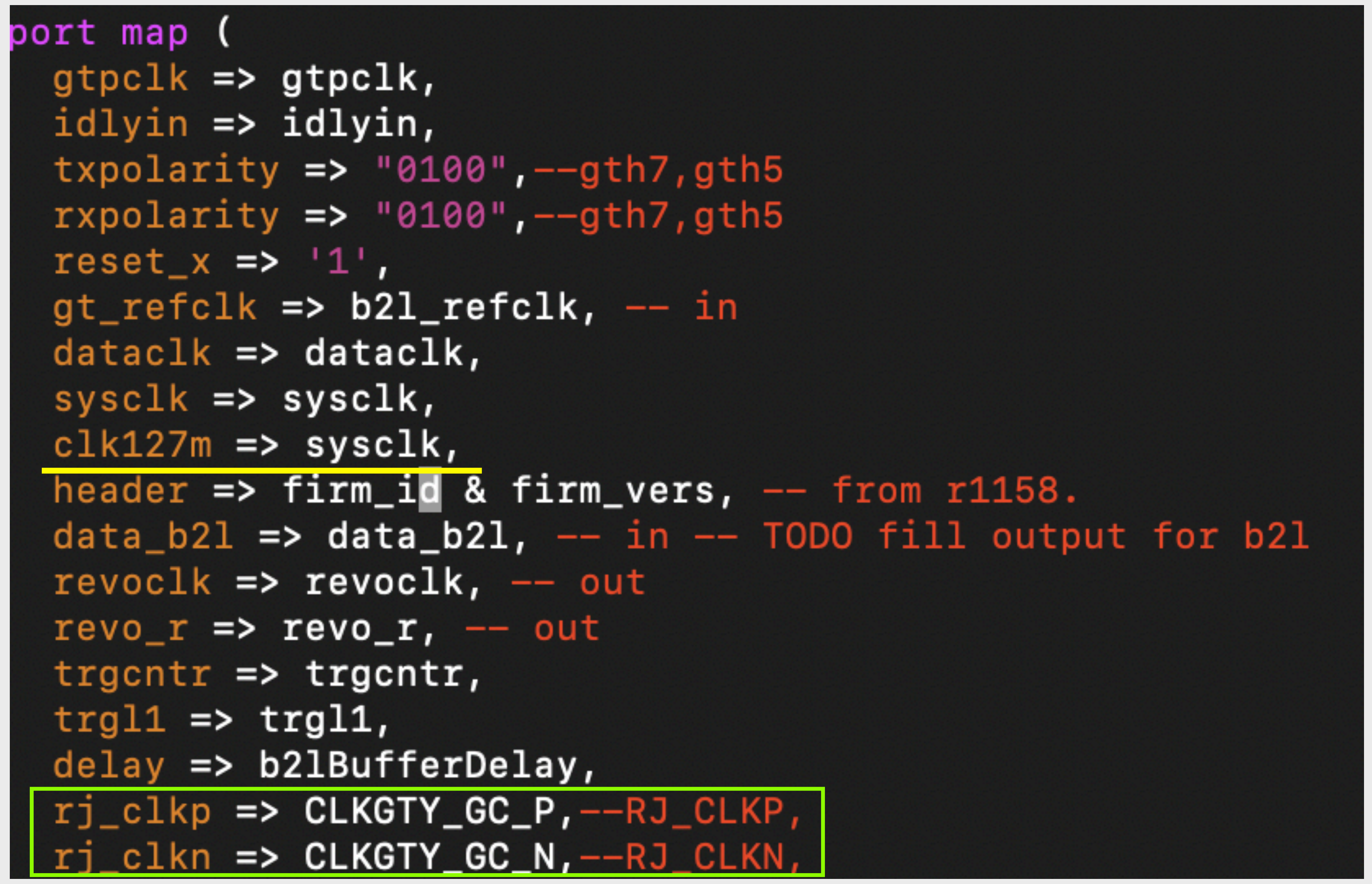
- If you are using 254MHz reference clock for CLKGT_GC, USE254IN=1 in Library/libut4_02/B2L/src/b2tt/b2tt.vhd



```
USE254IN    : std_logic := '1'; -- 254 MHz clock in for DHH
```



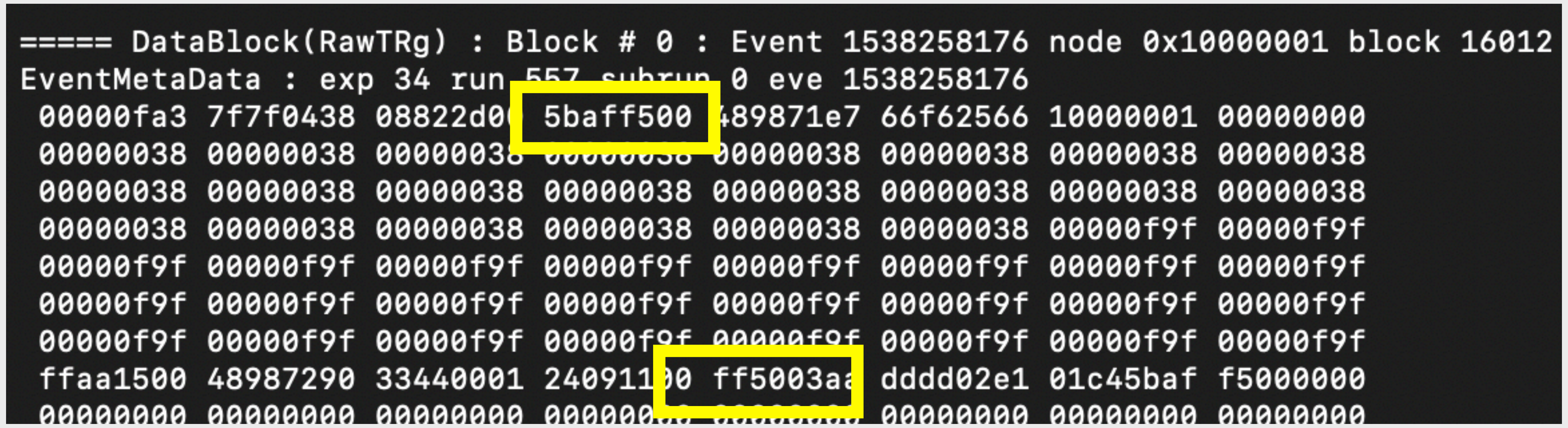
```
clk_wiz_127_i : clk_wiz_127
  port map (
    clk_in1 => clk_gth_buf,
    clk_32m => dataclk,
    clk_127m => sysclk,
    clk_32m_shifted => dataclk_shifted,
    locked => open
  );
```



```
port map (
  gtpclk => gtpclk,
  idlyin => idlyin,
  txpolarity => "0100",--gth7,gth5
  rxpolarity => "0100",--gth7,gth5
  reset_x => '1',
  gt_refclk => b2l_refclk, -- in
  dataclk => dataclk,
  sysclk => sysclk,
  clk127m => sysclk,
  header => firm_id & firm_vers, -- from r1158.
  data_b2l => data_b2l, -- in -- TODO fill output for b2l
  revoclk => revoclk, -- out
  revo_r => revo_r, -- out
  trgcntr => trgcntr,
  trgl1 => trgl1,
  delay => b2lBufferDelay,
  rj_clkp => CLKGTY_GC_P,--RJ_CLKP,
  rj_clkn => CLKGTY_GC_N,--RJ_CLKN,
```

Test after debug

- 40 hours with 10 kHz, OK.



```
===== DataBlock(RawTRg) : Block # 0 : Event 1538258176 node 0x10000001 block 16012
EventMetaData : exp 34 run 557 subrun 0 eve 1538258176
 00000fa3 7f7f0438 08822d00 5baff500 489871e7 66f62566 10000001 00000000
 00000038 00000038 00000038 00000038 00000038 00000038 00000038 00000038
 00000038 00000038 00000038 00000038 00000038 00000038 00000038 00000038
 00000038 00000038 00000038 00000038 00000038 00000038 00000f9f 00000f9f
 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f
 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f
 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f 00000f9f
 ffaa1500 48987290 33440001 24091100 ff5003aa dddd02e1 01c45baf f5000000
 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

- No problem with stop-start without ABORT
  - 10 times in global run with four 3D modules

Comments and conclusion

- I do not understand how this bug caused observed problems
- Why only one module out of four which are same firmware?
  - 4 modules are not so precisely synchronized at ~200 nsec level.
- The bug did not cause event shift when no suppression (cosmic ray data taking).
  - Full data (~16kB) always read, thus always events in waiting queue and no "busy signal for the last event in the queue".
- Seems to be solved. Sorry for bothering you but please update your modules. Let's see global runs.