Alice trigger overview

A. Jusko, on behalf of trigger group
The University of Birmingham

Triggering Discoveries in High Energy Physics II, Puebla

Contents

- Time line
  - Trigger highlights, protocol
- Overview of the Central Trigger Processor (CTP)
- The Local Trigger Unit
- Run2 upgrade
- Infrastructure
- Software
  - Development
  - Run control
  - Monitoring

Time line

- **2003** hardware design of LTU board
- **2004** hardware design of CTP boards
- **2005** LTU distribution to ALICE groups
- **2006** installation of CTP in P2
- **2008** run1 start
- **2013** LS1 - Trigger upgraded for run2
- **2015** run2 start
- **2019** LS2
- **2021** run3

Trigger higlights

- ***Three levels of hierarchical hardware triggers*:**
  - ***L0*** (1.2µs after beam interaction at FE) → ***L1*** (7µs) → ***L2*** (105 µs)
- **At any time, 24 ALICE sub-detectors *dynamically partitioned* in up to 6 *clusters.***
  - Cluster configuration arbitrary and fully programmable – could be exclusive,
  - more likely to overlap.
- ***Past-future Protection* logic selects events with either no pile-up, or**
  - a number of pile-up interactions up to a programmable limit.
  - Independent protection for each cluster; operates on all three trigger levels.
  - **(Not used much in Run 1)**
- **Clock and data traffic over the TTC system**
  - *Channel A*: L0, L1 signal (L0: also LVDS can be used)
  - *Channel B*: Orbit, Pre-pulse
  - But also, for each trigger:
    - L1 Data Message - 9 16-bit words (5 in run1)
    - L2a Message - 13 words (8 in run1)
- **LTU concept allowing standalone operation**
  - Developments in labs
  - Parallel development/maintenance in Point 2

CTP context diagram

CTP inputs

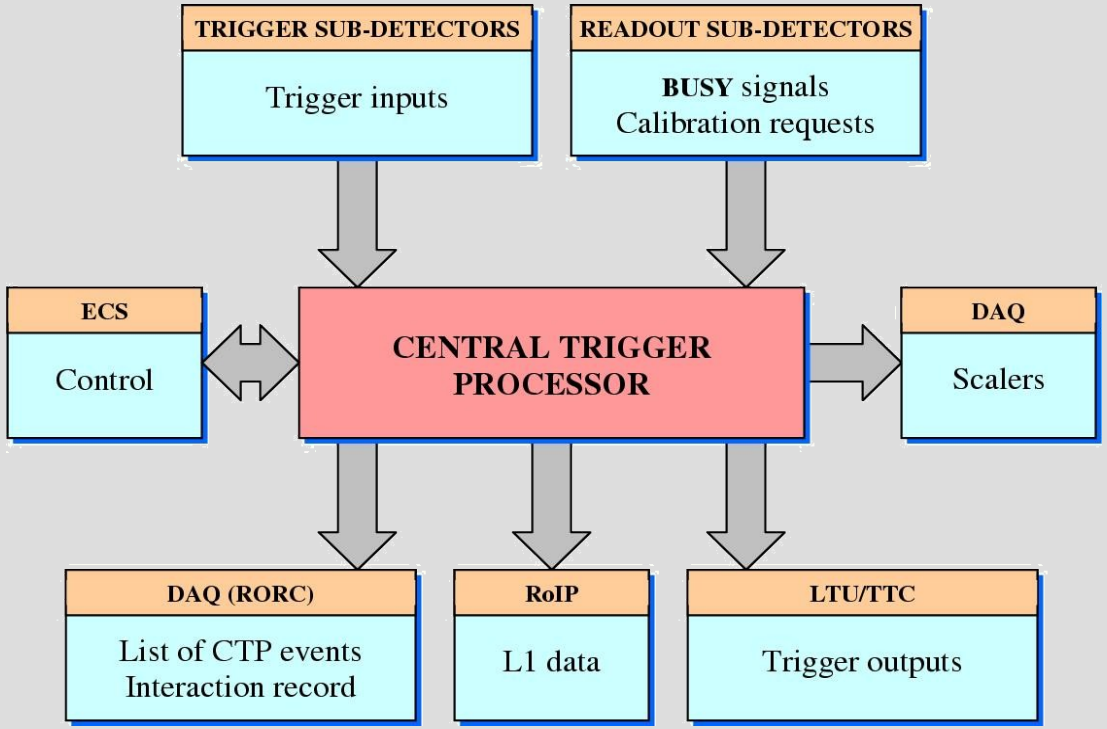
- LHC timing – BC, Orbit
- 60 trigger inputs
  - 24 L0
  - 24 L1
  - 12 L2
- 24 BUSY inputs

ECS control, scalers

- Over VME bus

CTP readout

- Copy of all L2am sent to sub-detectors
- Interaction records
- Status of L0 inputs in L0 time

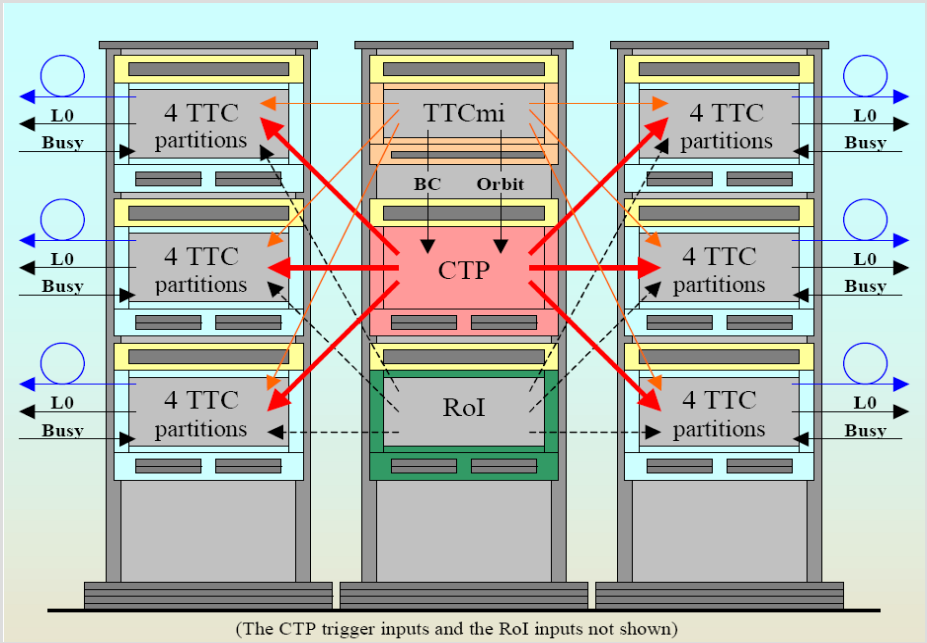


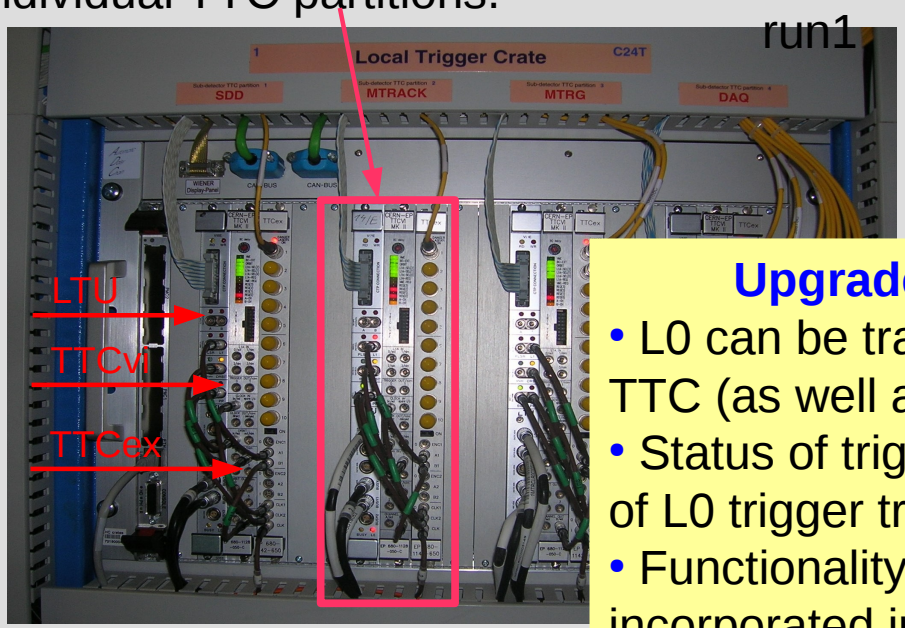
CTP outputs

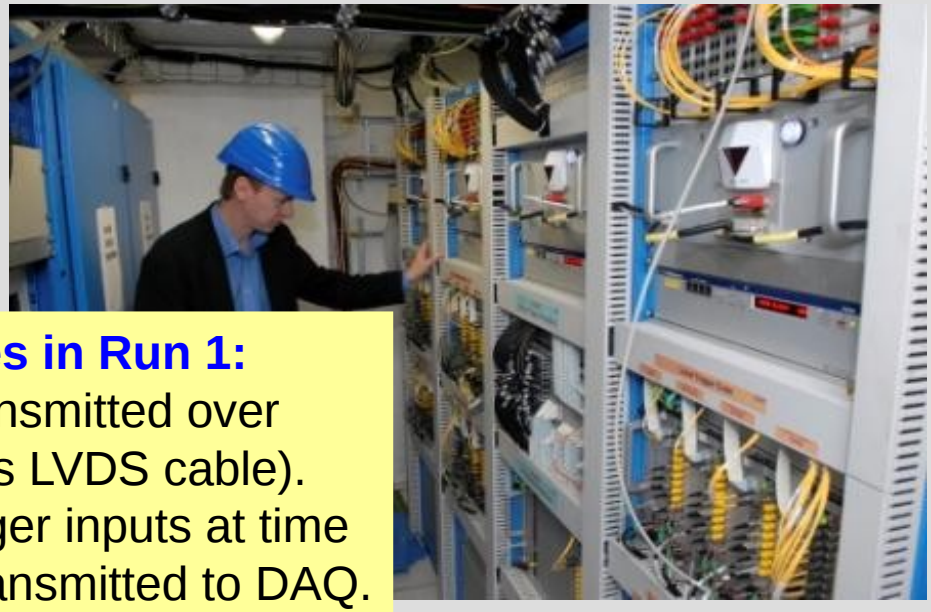
- 24 independent sets, 7 outputs per sub-detector
- Sent out through 6 FO boards

Trigger implementation

- Racks are located below the Di-Muon magnet in cavern.
- short latency for L0 – 1.2μs, but…
  - stray magnetic field (~10mT)
  - low-level radiation
  - no access when beam is on
- TTC crate supplies LHC clock direct to CTP and to Time-of-Flight detectors.
- CTP sends L0, L1, L2 signals and messages to sub-detectors via individual TTC partitions.



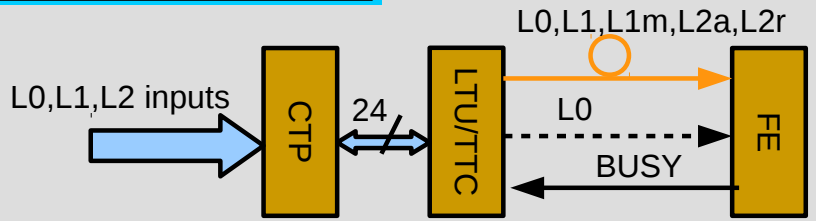




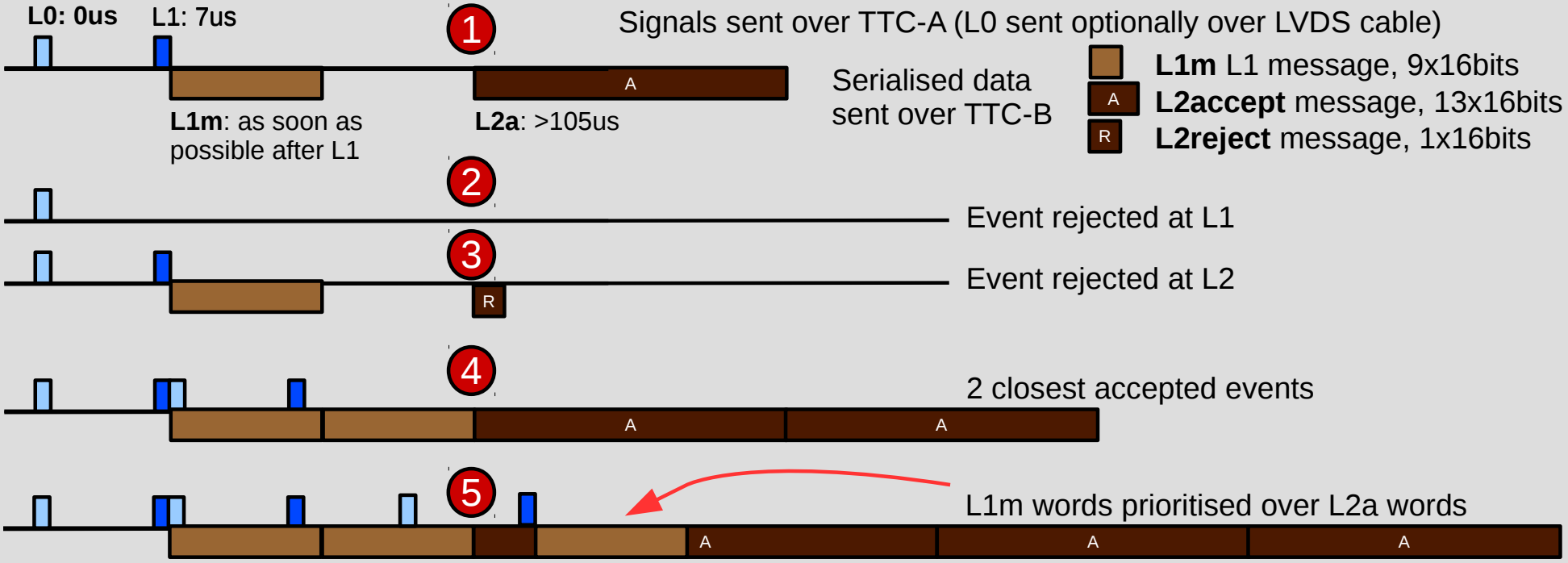
Upgrades in Run 1:

- L0 can be transmitted over TTC (as well as LVDS cable).
- Status of trigger inputs at time of L0 trigger transmitted to DAQ.
- Functionality of TTCvi board incorporated in LTU

Trigger protocol



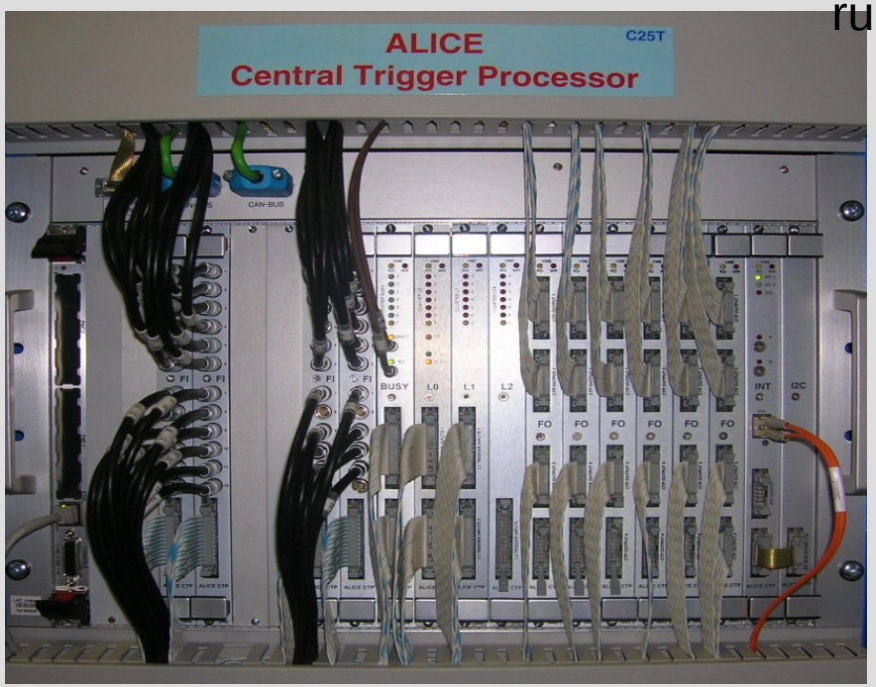
Trigger sequence of an accepted event:

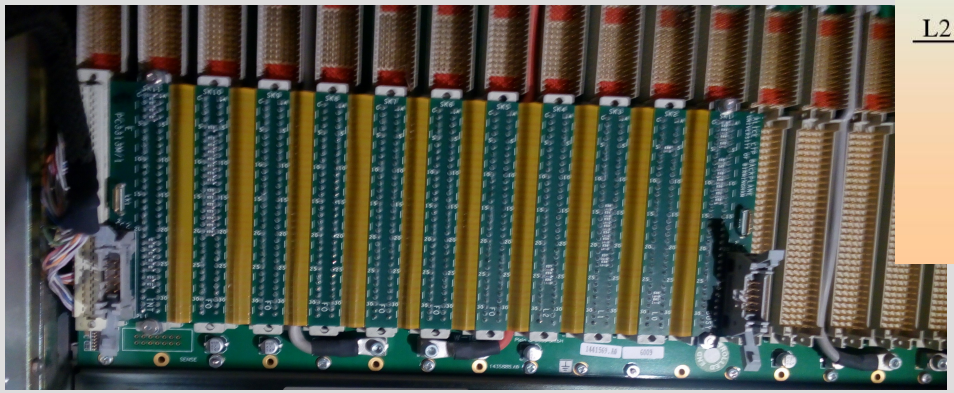


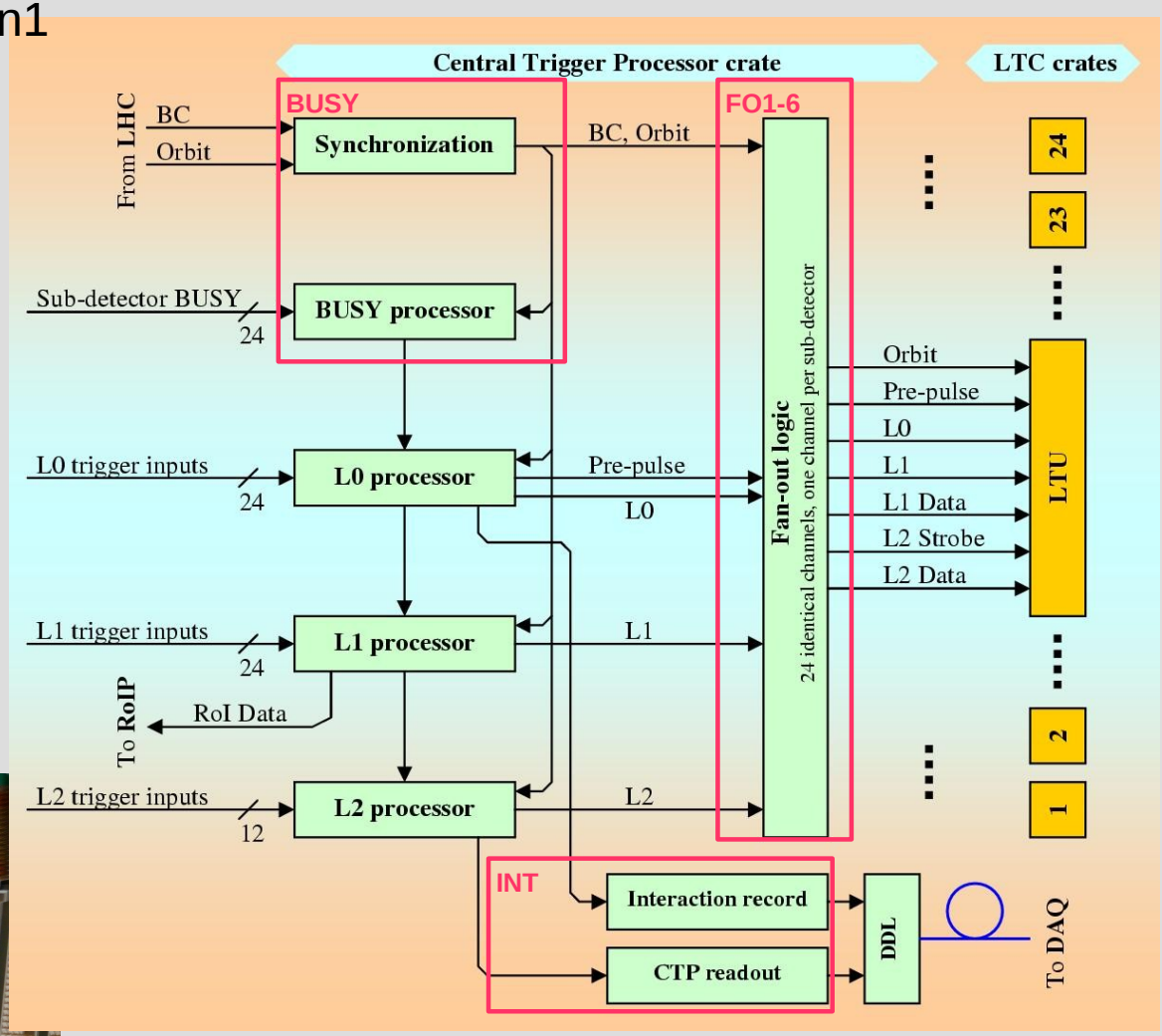
- L0, L1, L1m, L2a come ordered as they happened in time. L1m prioritized w.r.t. L2a.
- L2a message contains BCID of corresponding L0.
- TTC-B data: individually addressed command (16bits word in one 42bits TTC-frame).
- L0-L1: FE protected against another L0 -right time for raising BUSY.
- Each trigger sequence is finished (i.e. BUSY = do not send L0).

CTP

run1





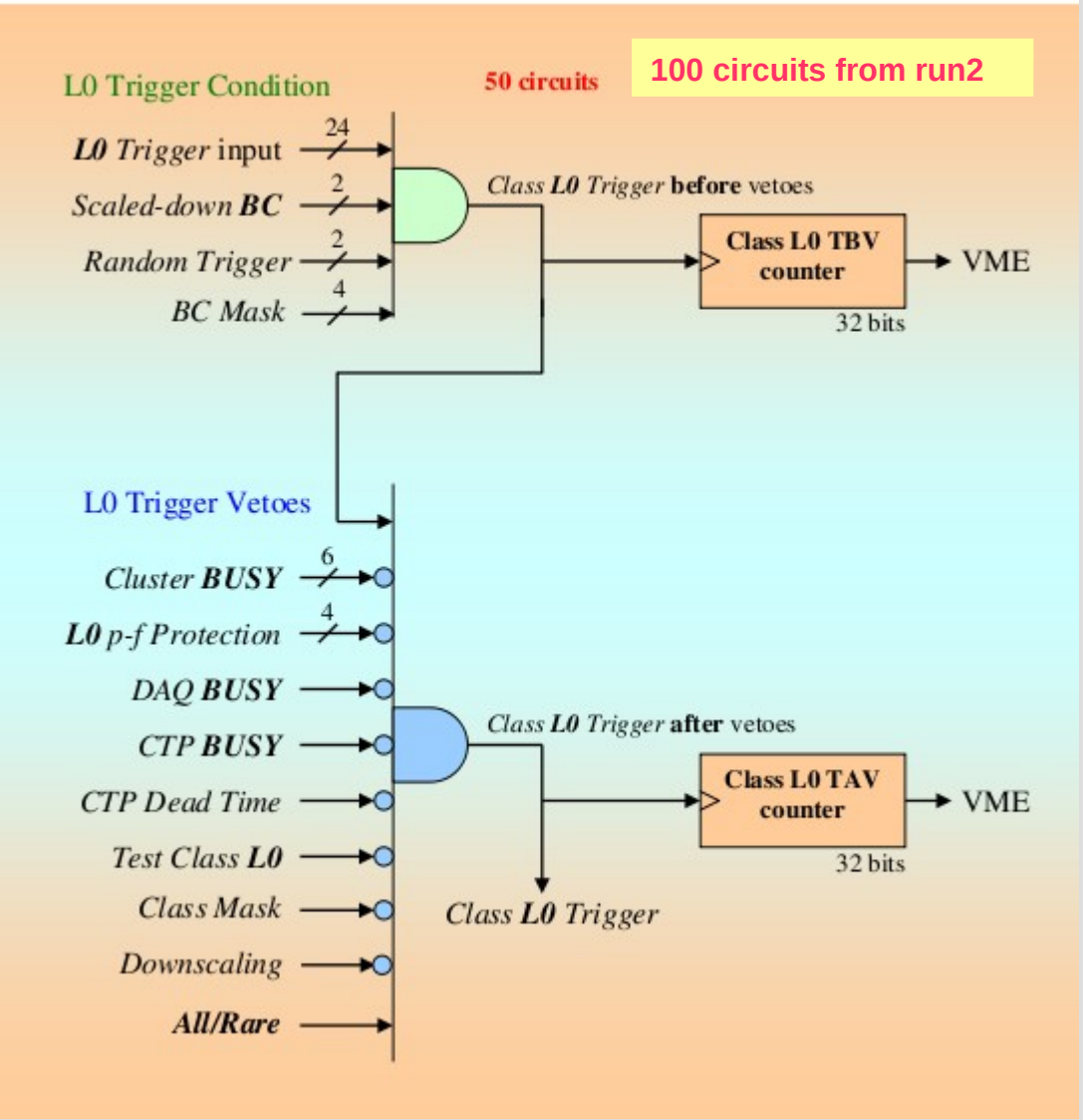


- 11 6U VME boards, 8 PCB layers
- CTP boards connected by VME custom backplane

CTP class, cluster concept

Trigger class
• Logical AND of any subset of CTP inputs (on each level) AND
• special L0-fuction: any logical combination of first 8 L0-inputs
• Each class affects a single cluster
Shared among all classes:
• 4 internal generators
• 12 BC Masks (3564 bits each)
• 4 L0 p/f circuits

Trigger cluster (6 in CTP)
• Group of detectors triggered together (busy is shared)
• A cluster can be associated with an arbitrary number of trigger classes
• Clusters run independently (different rates for different clusters)
• Overlapping clusters: at least one common detector



DAQ partition menu

```
RND1 50khz
BCM1 bcmB
Clusters:
MUON
DTRUE(rnd1) D0MSL(BCM1)
MUON_TRG MUON_TRK
ALL
DEMCEG1(L0pr=20%) D0TVX(BCM1)
SPD SDD SSD TPC TOF HMPID PHOS CPV FMD T0 V0 ZDC ACORDE EMCAL AD TRD
```

2 classes contributing to MUON cluster

2 classes contributing to ALL cluster

```
D0TVX 0TVX
D0MSL 0MSL
DEMCEG1 0EMC 1EG1
```

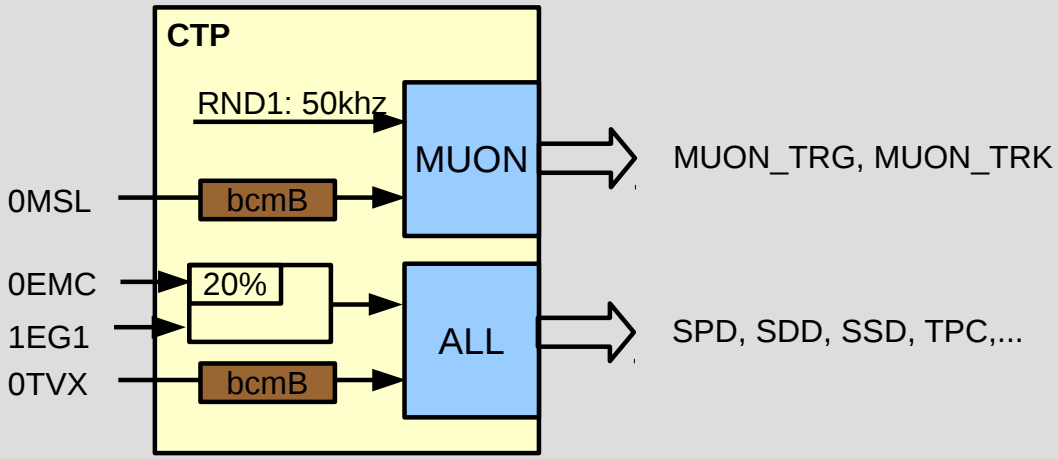
```
#nam = det       L Si  In dn Sw E  D DeltaMin DeltaMax
0TVX = T0        0  72  3  3  3 1 18 1000 1000 NA T0_3
0MSL = MUON_TRG  0 115 20 4 19 0 0 1000 1000 NA L0_Trig
0EMC = EMCAL     0  10 22  0 14 1  0 1000 1000 NA EMC_L
1EG1 = EMCAL     1  12  2  0  0 1  2 1000 1000 NA 40284
```

Level

Input (1..24)

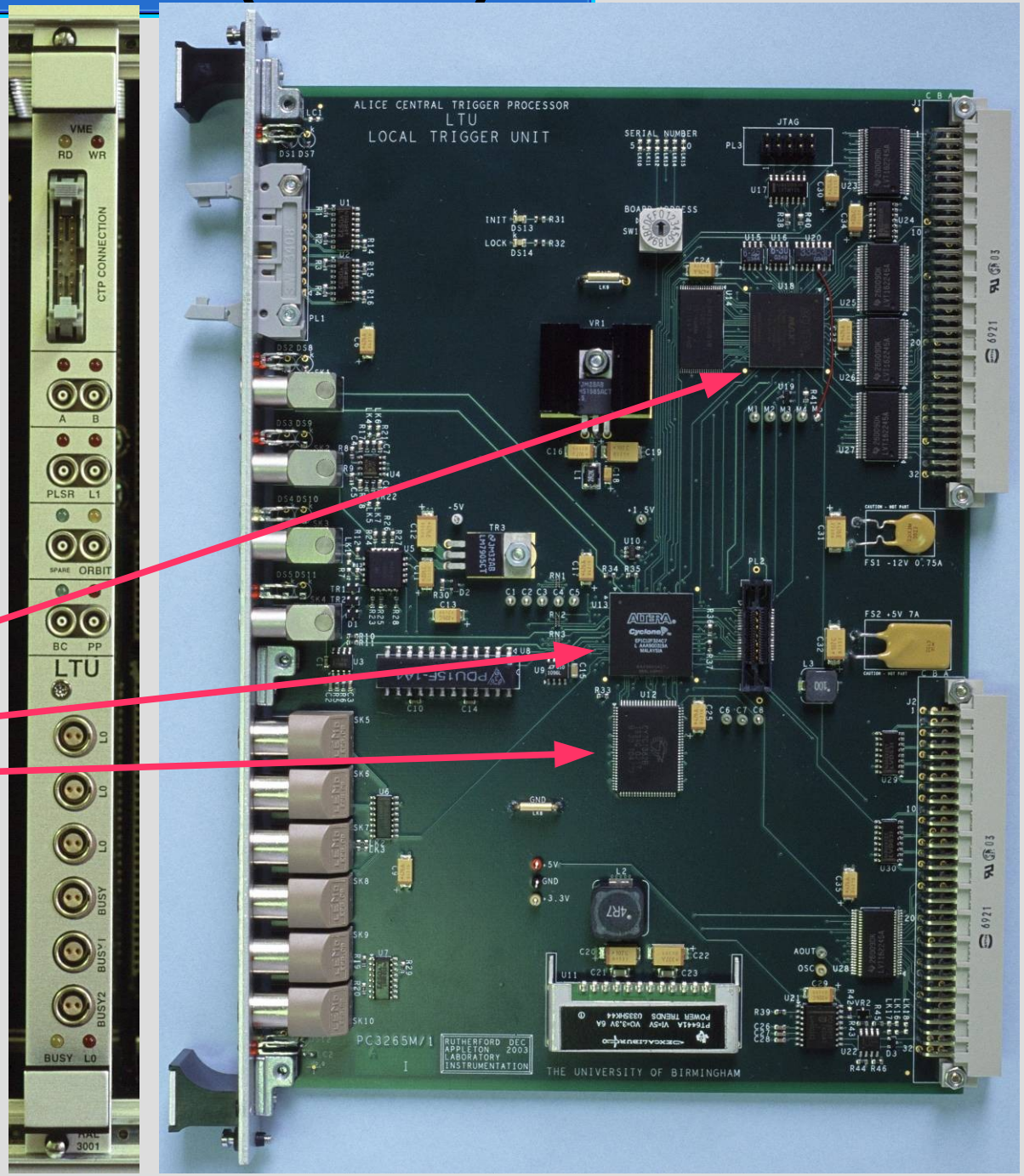
```
# 25ns_1740b_1728_1570_1636_96bpi_21inj
bcmB 23L40H1L38L8H1L8H1L38L306H1L44L34H1L44L11H1L44L34H1L44L11H1L44L34H1L44L11H1
L44L34H1L44L11H1L44L309H1L44L34H1L44L11H1L44L34H1L44L11H1L44L34H1L44L11H1L44L34H
1L44L11H1L44L309H1L44L34H1L44L11H1L44L34H1L44L11H1L44L34H1L44L11H1L44L34H1L44L11
H1L44L306H1L38L40H1L38L8H1L8H1L38L40H1L38L8H1L8H1L38L40H1L38L8H1L8H1L15L
```

Configuration files saved in ACT db (MySQL).

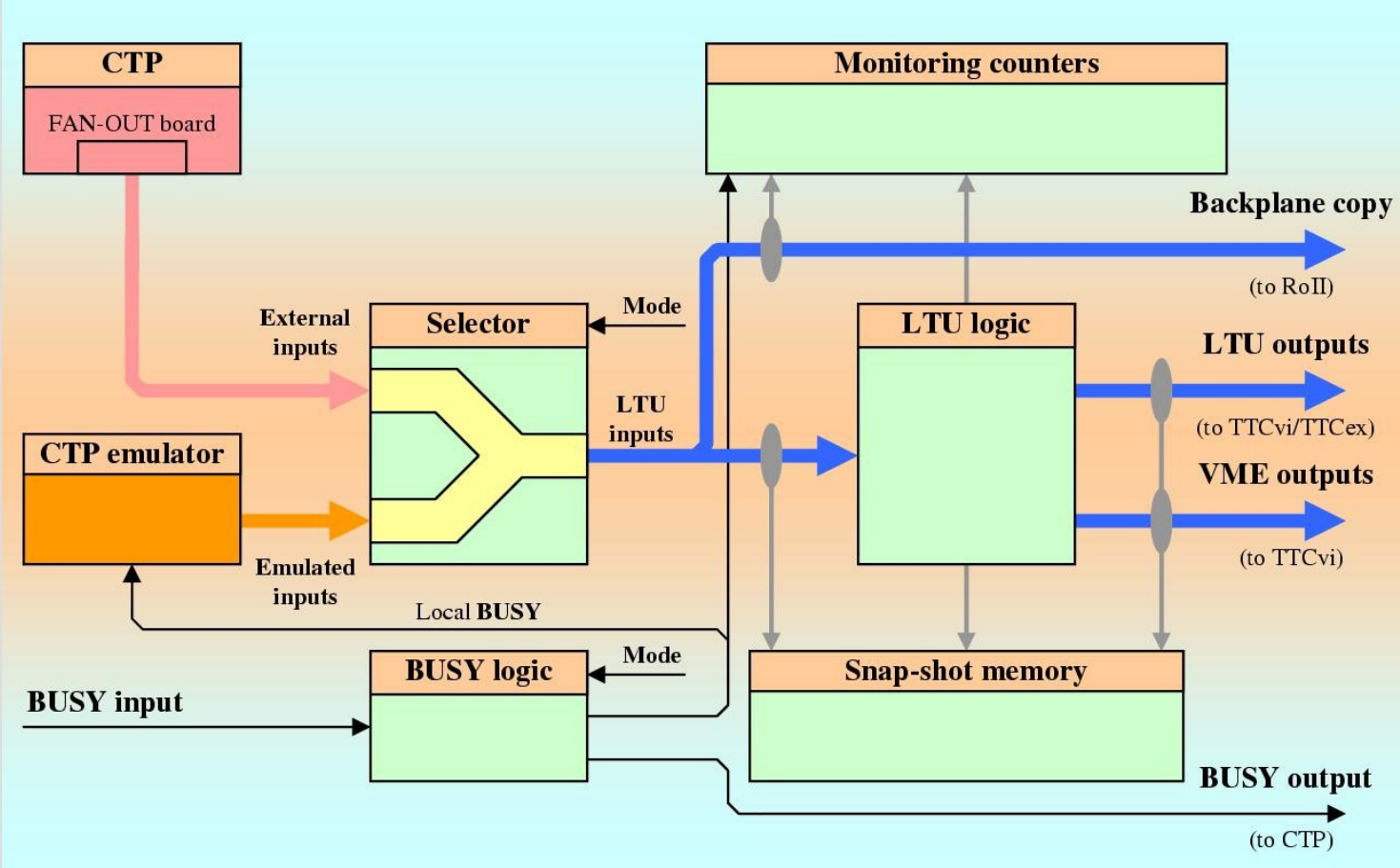


Local Trigger Unit (LTU)

- Uniform interface between the CTP and sub-detectors:
  - easier control
  - easier mods/upgrades
- Unique features:
  - Full CTP emulation (*stand-alone* mode)
  - Error emulation (front-end tests)
- VME, 6U form-factor
- 2x ALTERA FPGA, SSM
- Similar to other CTP boards
- Flexible, i.e. with different firmware used as a simple fanin/fanout or signal converter



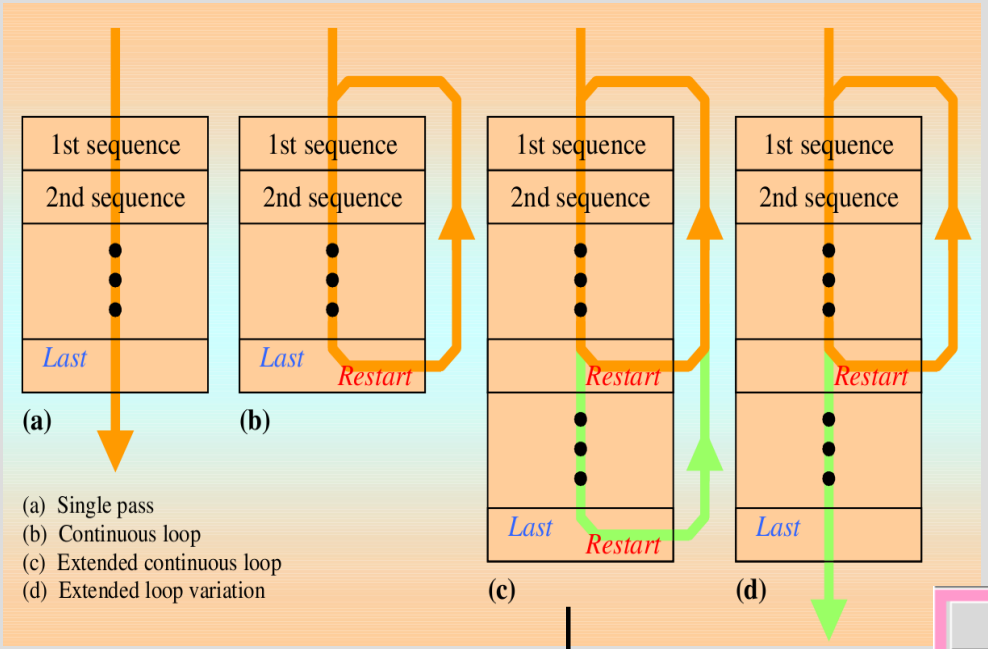
LTU block diagram



Global: signals are propagated from CTP to the output
Standalone: CTP inputs not connected, output signals generated by CTP emulator

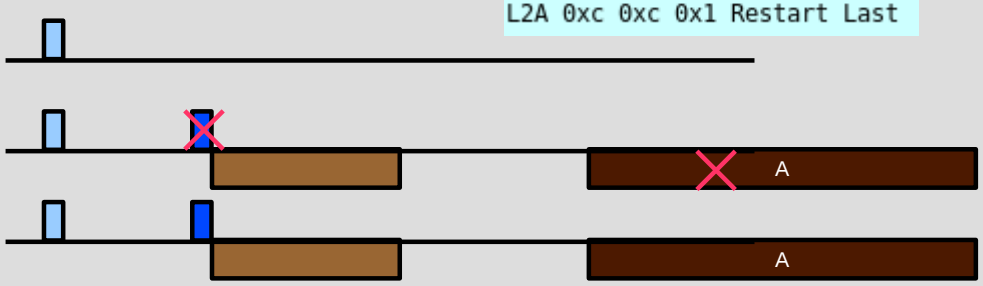
Monitoring (counters, SSM) available in both modes

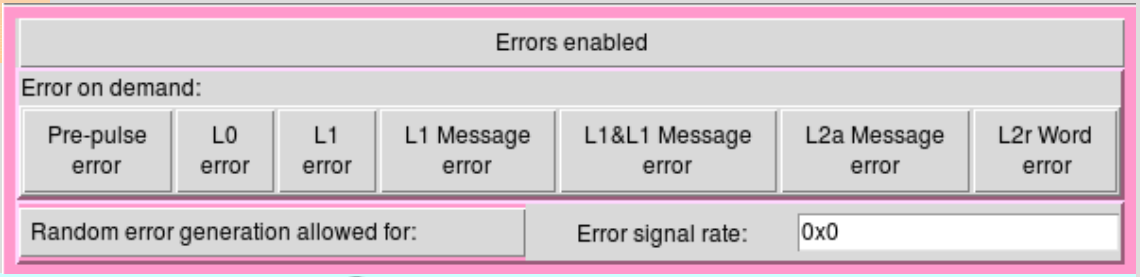
LTU ctp emulation, errors



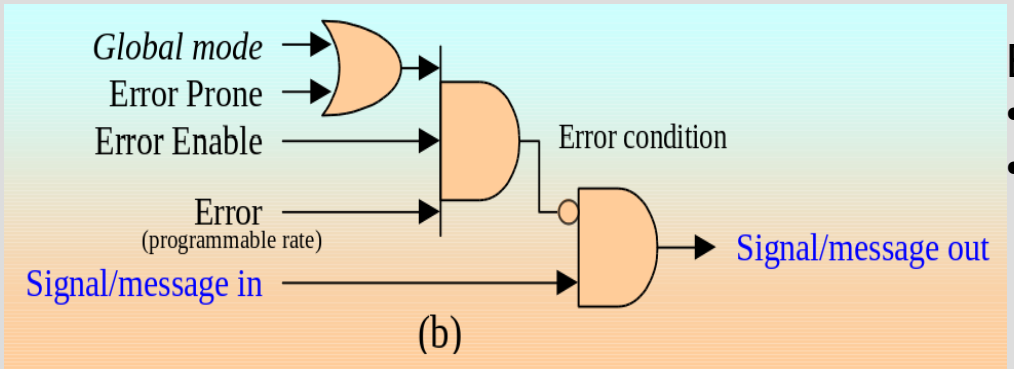
Sequences defined in a text file. An example requesting a faulty trigger between two correct (L1 rejected and L2 accepted) triggers:

```
#Errors: L1 L2aM
L0
L2A 0xc 0xc 0x1 ErrProne
L2A 0xc 0xc 0x1 Restart Last
```






```
L2A 0xcf000000000f000000000001 0xc000000000f000000000001 0x1 Restart
L2R 0xc000000000f000000000001 0xc000000000f000000000001 0x2 Restart Last
```



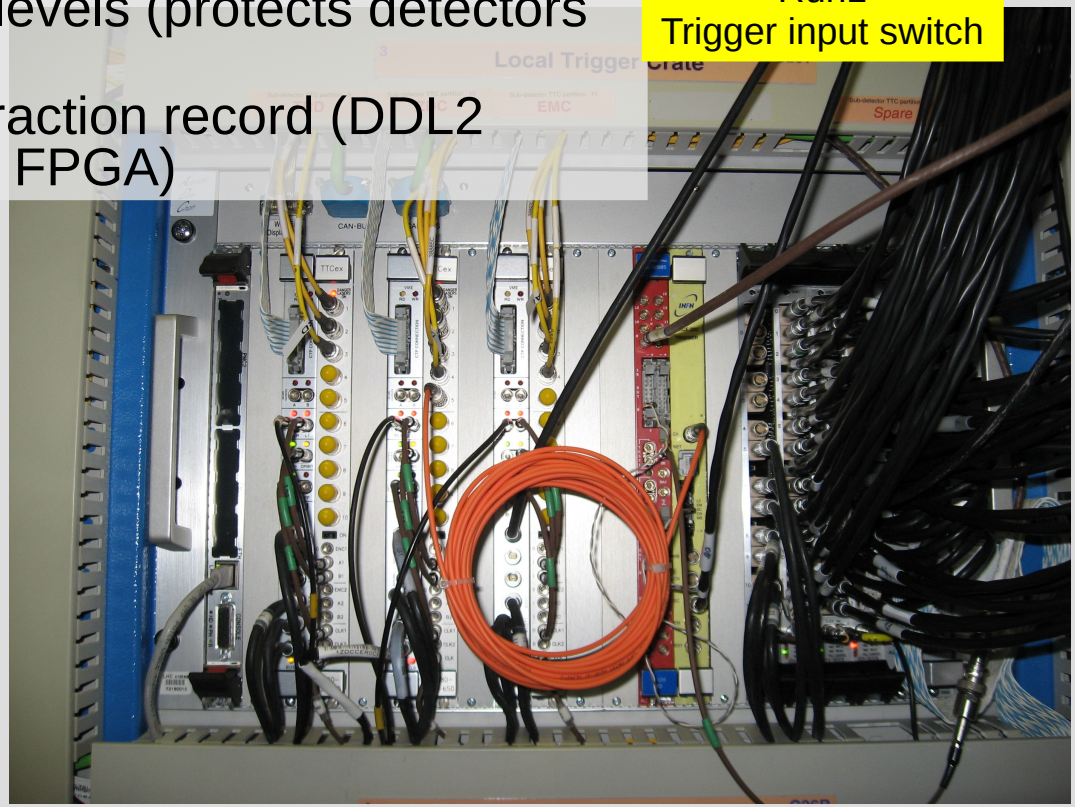
Errors:

- Always of type „missing signal/data“
- Programmable:
  - Occurance: on demand (just one)/random
  - 'on demand' errors available also in global
  - Error prone flag for choosen sequence

Run2 upgrade requirements

- Add LM trigger level (before L0 trigger level) for TRD pre-trigger
- Increase number of classes from 50 to 100
- Trigger input switch (48->24) integrated into CTP
- New snapshot/generator memory (using 1+1 GB of DDR3 memory)
- All functions (special type of trigger inputs) from 8 inputs (only 4 inputs before)
- LM and L0 interactions (definition of basic trigger) from first 8 inputs
- Past-Future protection for LM and L0 levels (protects detectors from Pile-Up)
- Second link to DAQ for extended Interaction record (DDL2 firmware implemented in CTP LM0 FPGA)

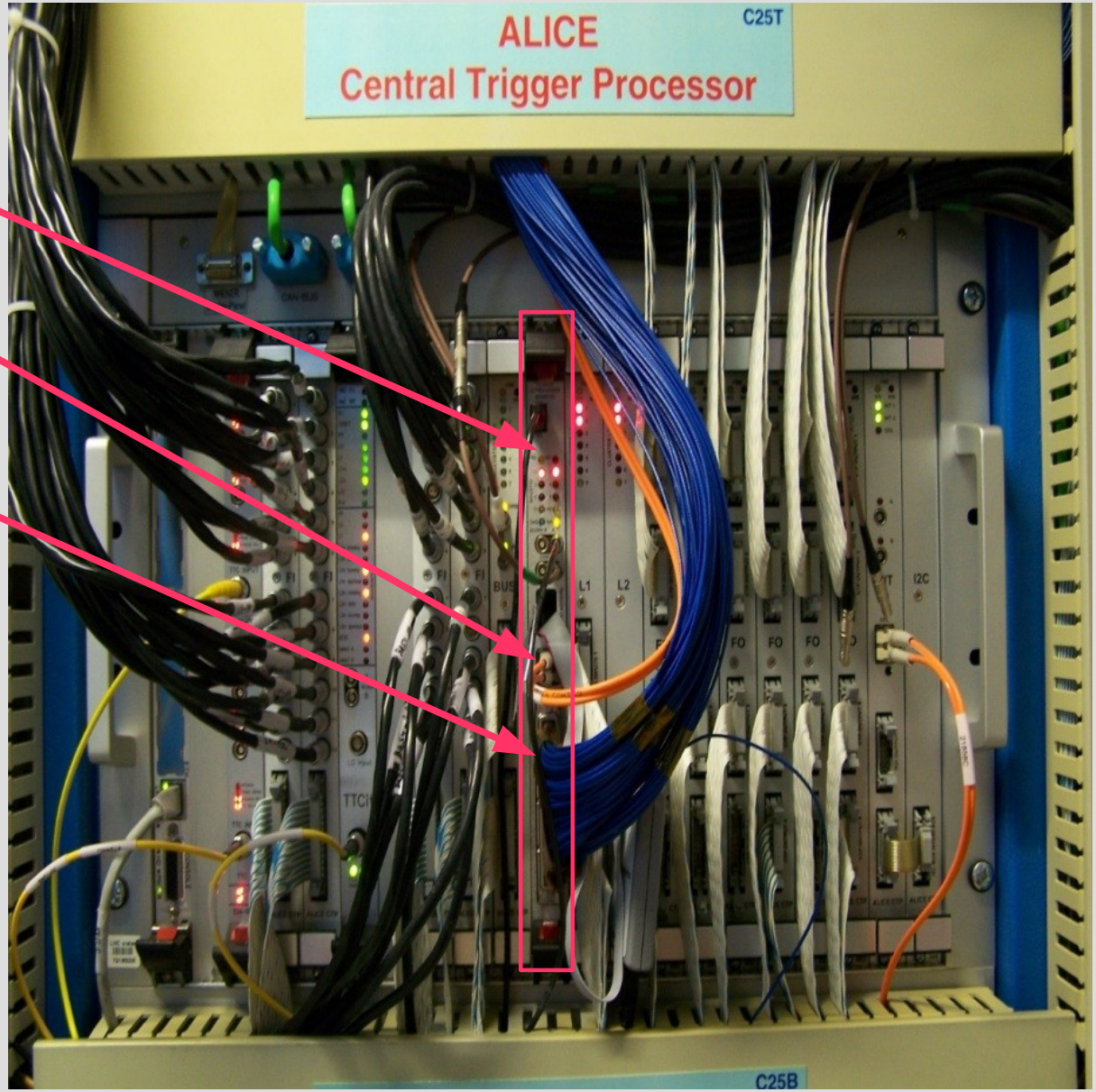
Run1
Trigger input switch



Run2 upgrade

Changes with respect to LHC Run 1

- New LM0 board with Kintex-7 FPGA
- New Detector Data Link (DDL2) to DAQ
- 96 differential IOs at front panel
- New octopus cable (blue) for CTP inputs
- New FPGA for LM0, all other CTP and LTU boards, updated
- Installed new trigger cables from T0, V0, CPV and DCAL
- Repaired many old cables
- Timing on CTP backplane, CTP-LTU and LTU-TTCex connections re-checked after upgrade of all FPGA designs



Run2 upgrade:LM0 board

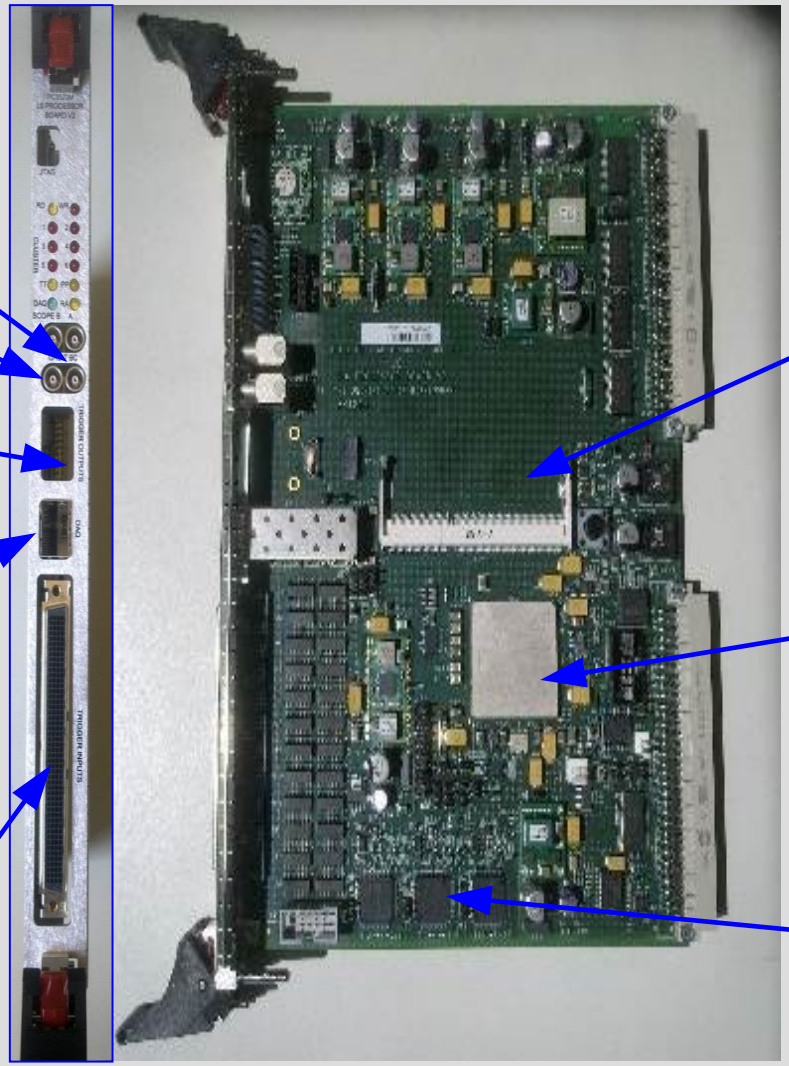
BC input (ECL)

ORBIT input (ECL)

SAMTEC FireFly cable
- 12 differential links

SFP+
- link to DAQ

96 LVDS I/O
- Trigger inputs
- BUSY inputs
- LM output



2 GB DDR3 memory

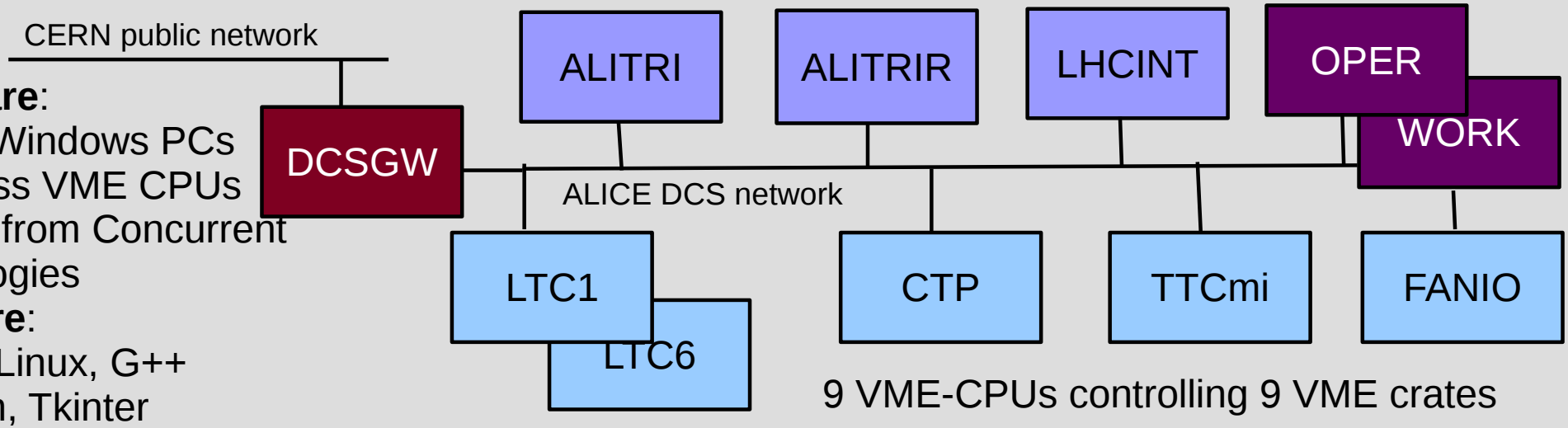
Kintex-7 FPGA

Power controllers

New LM0 board designed as a CTP prototype for Run 3

Infrastructure

Servers:
- BOOTP/NFS providing for diskless VME CPUs, trigger software
- Application server: running root apps, monitoring
- LHC interface: bunch by bunch and instantaneous luminosity published to LHC
- Operator/working WinCC nodes: VME crates switching on/off, clock monitoring

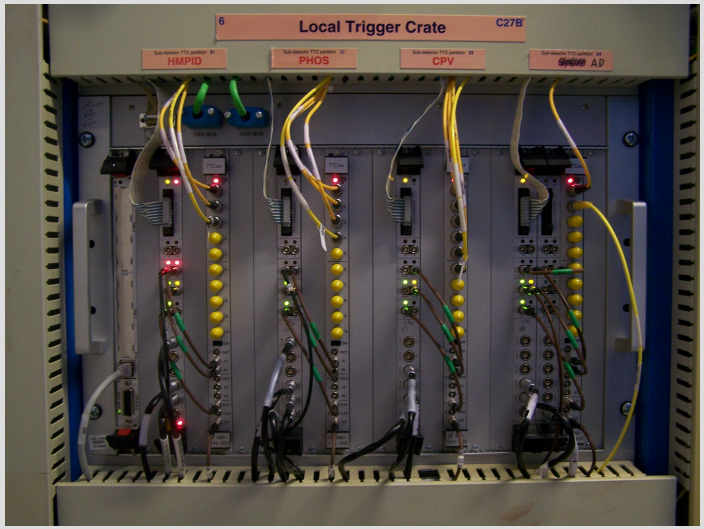


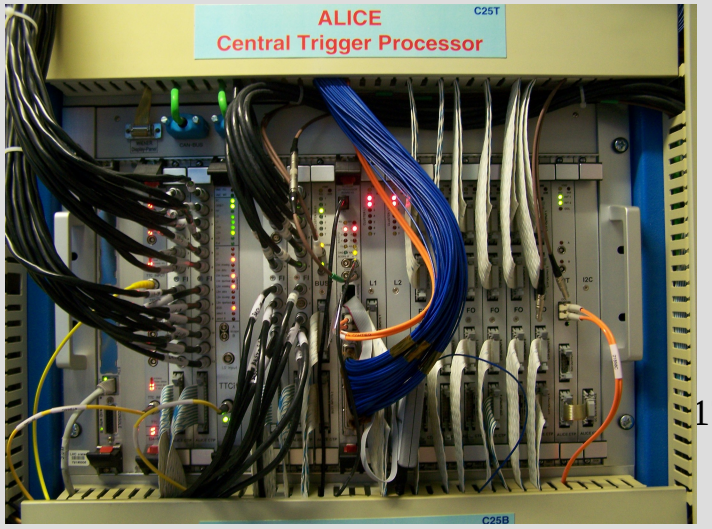
Hardware:
- Linux/Windows PCs
- Diskless VME CPUs

VP-315 from Concurrent technologies

Software:
- SLC6 Linux, G++
- Python, Tkinter
- VMERCC driver
- SMI/DIM
- OneSiS
- RRD
- DAQlogbook, infoLogger,
- root
- REDIS (run2)

9 VME-CPUs controlling 9 VME crates





Development

- 7 different VME boards
- Quick prototyping
- Distribution of the software together with LTUs
- Sw deployment at P2

„**vme b**oards“ framework

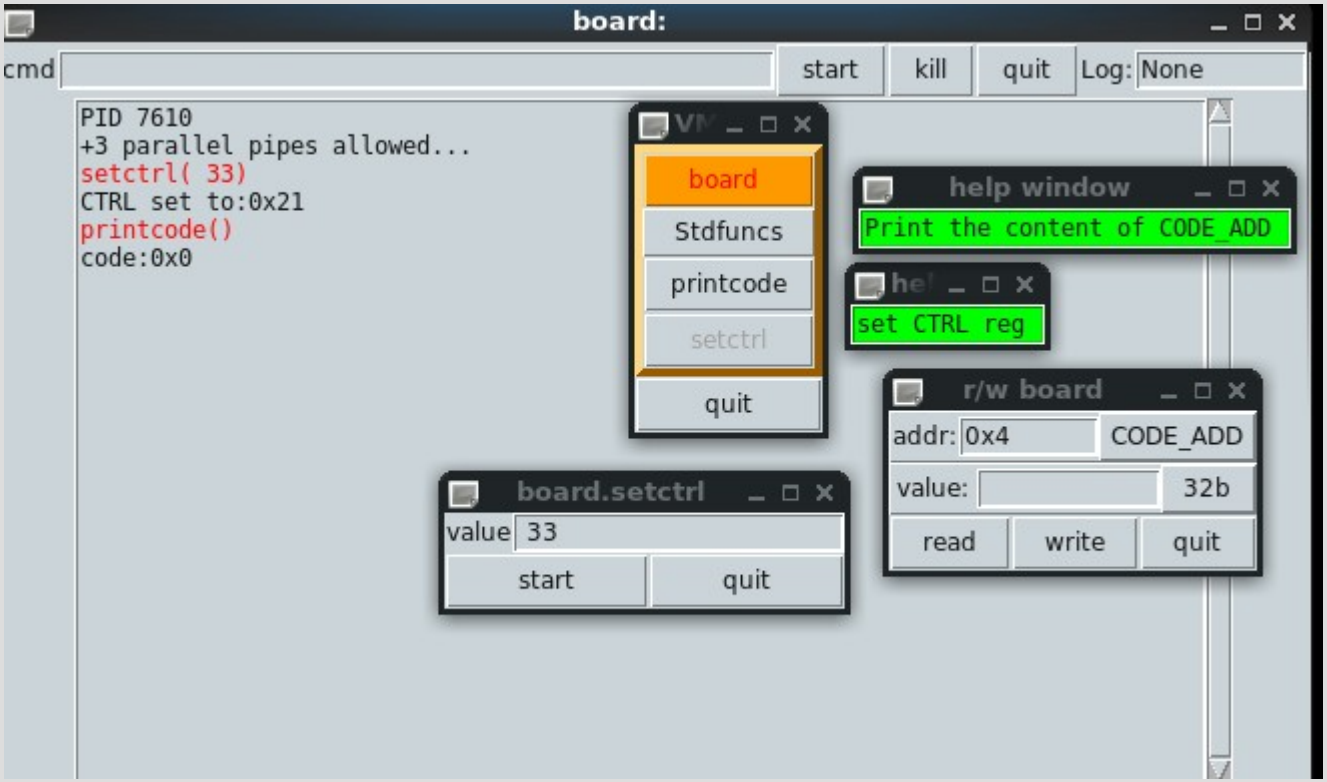
Py/Tkinter

vmecomp.py board → board.exe → crate.py board

VMERCC

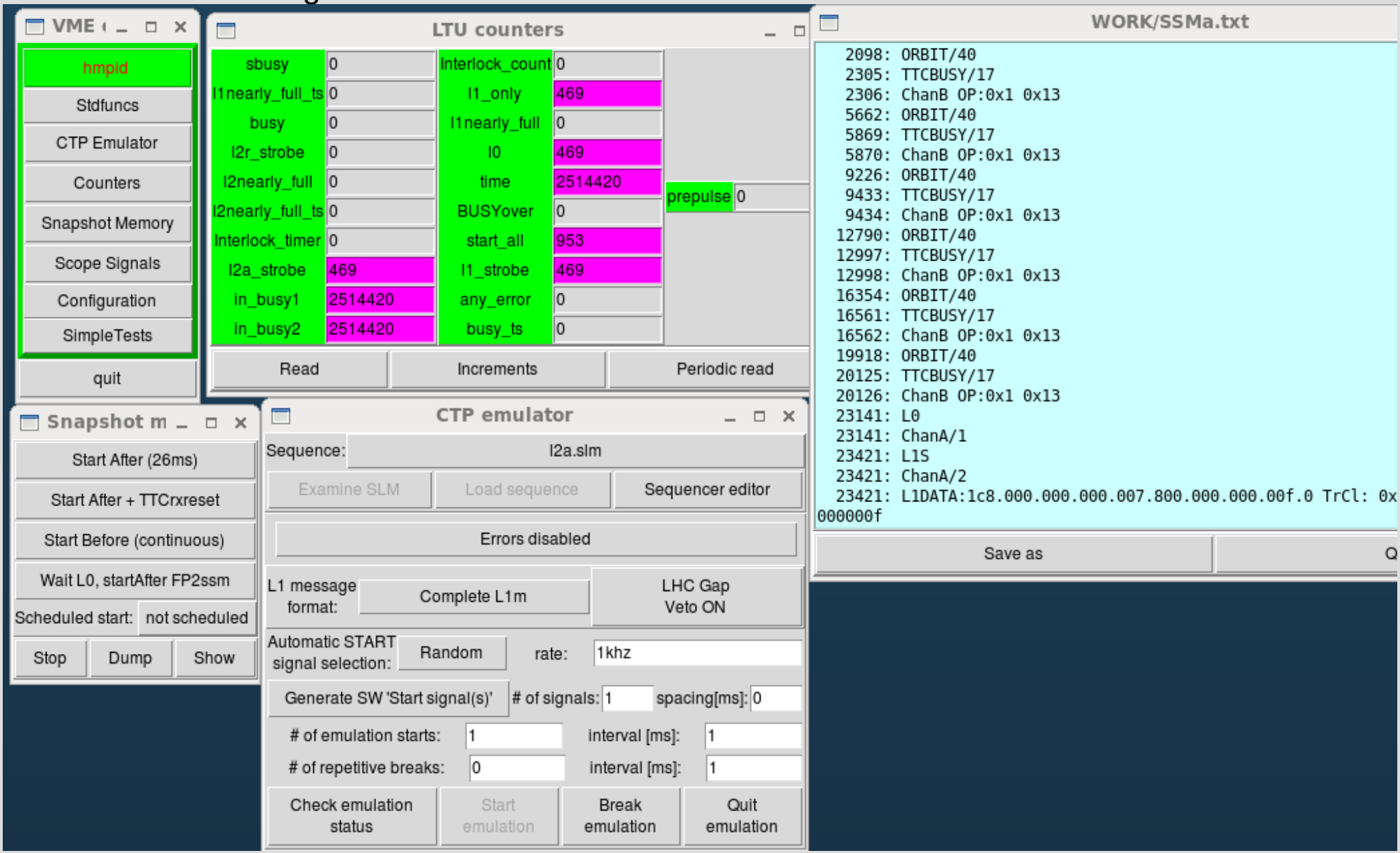
```
/*BOARD board 0x829000 0xd000 A24
*/
/*REGSTART32 */
#define CODE_ADD   0x4   /* board type */
#define CTRL       0x8   /* control reg */
/*REGEND */

/*FGROUP
Print the content of CODE_ADD */
void printcode() {
w32 code;
code= vmer32(CODE_ADD);
printf("code:0x%x\n", code);
}
/*FGROUP
set CTRL reg */
void setctrl(w32 value) {
vmew32(CTRL, value);
printf("CTRL set to:0x%x\n", value);
}
```



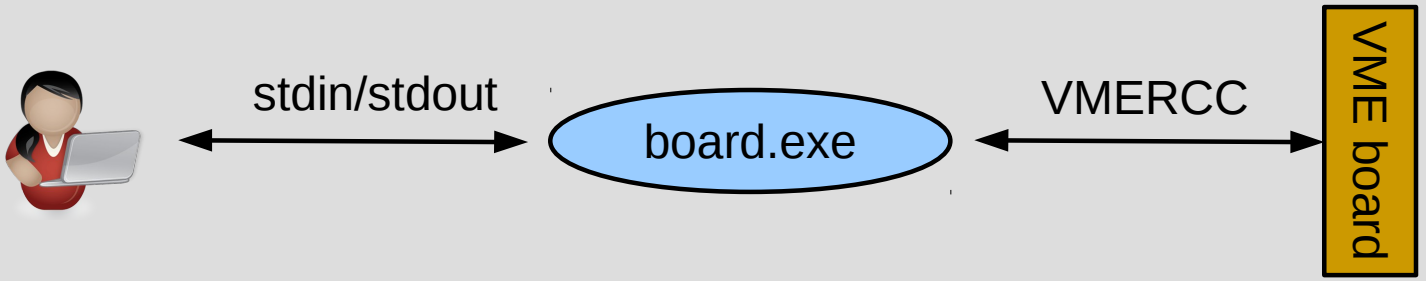
Development -ltu

VMEB framework was successfully used during development phase (from 2004).
It was delivered together with LTU boards.

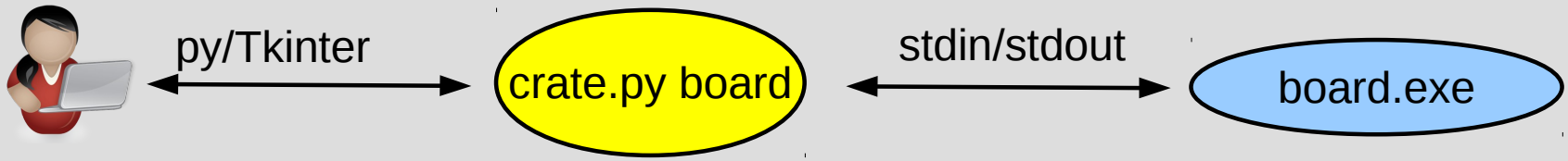


LTU software

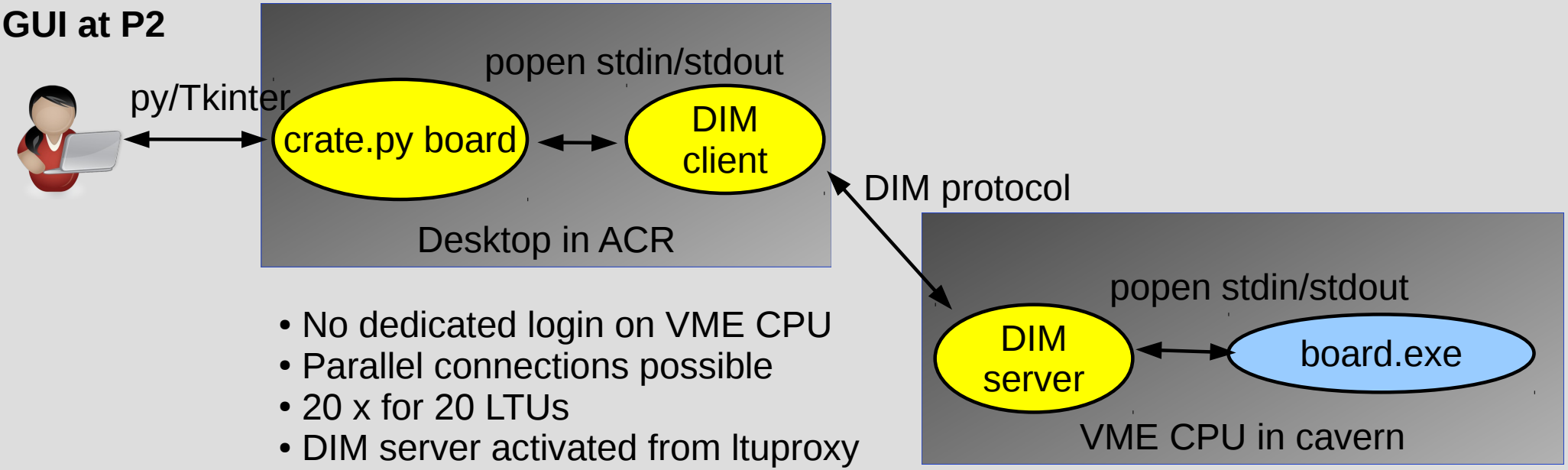
Command line



GUI



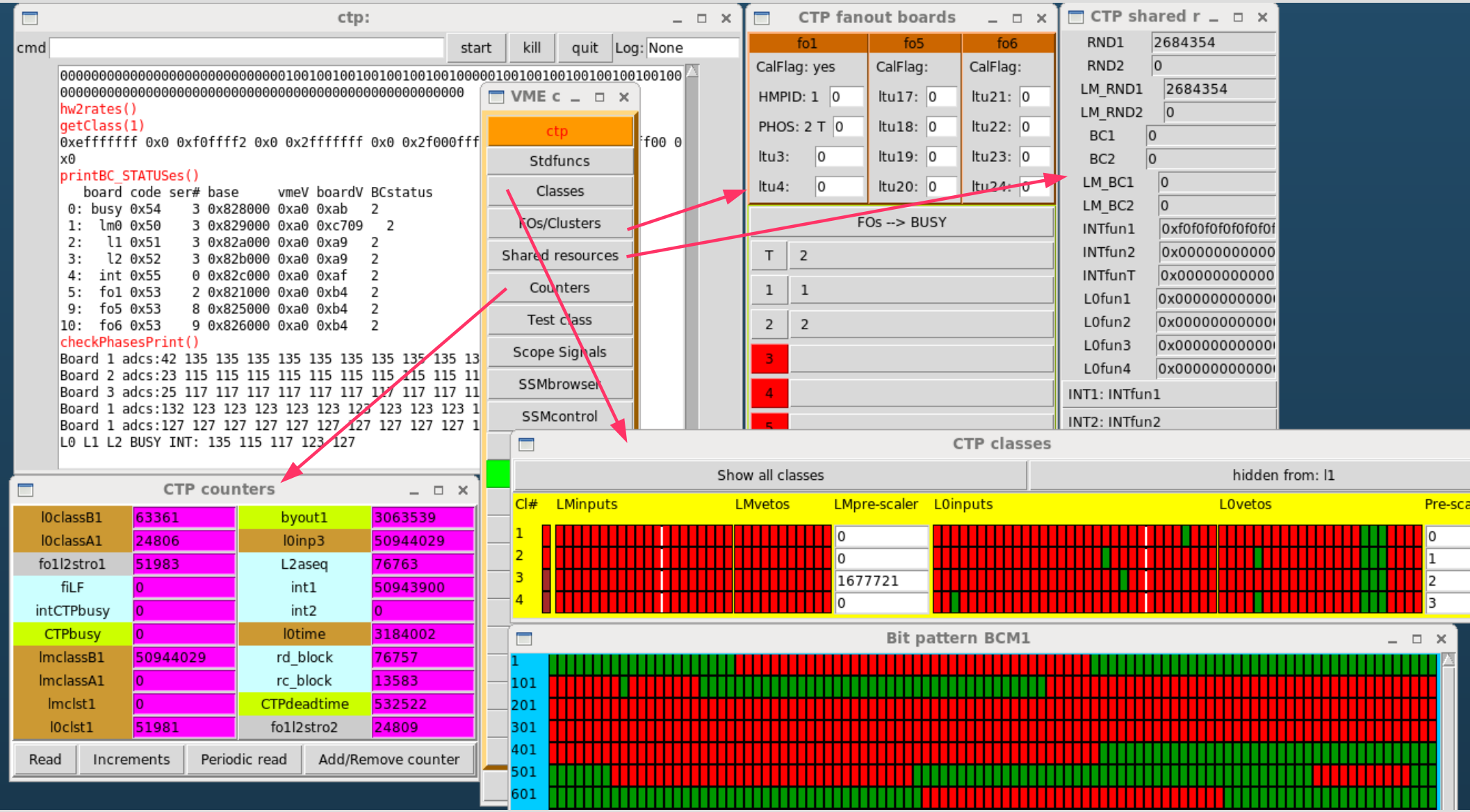
GUI at P2



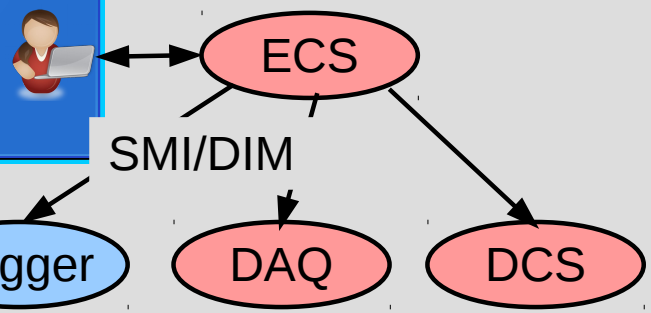
- No dedicated login on VME CPU
- Parallel connections possible
- 20 x for 20 LTUs
- DIM server activated from ltuproxy

Development -ctp

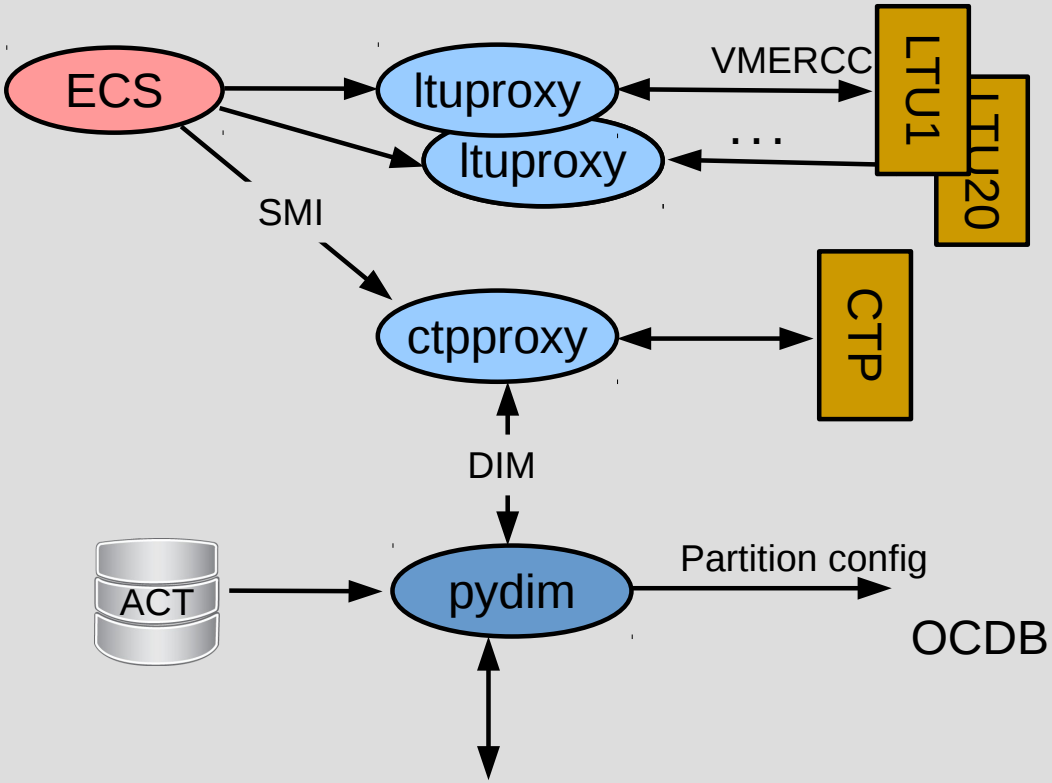
VMEB framework was also used for ctp development, is used as an expert software



Run control



Trigger run control:
set of proxies (unix daemons) controlling trigger boards.

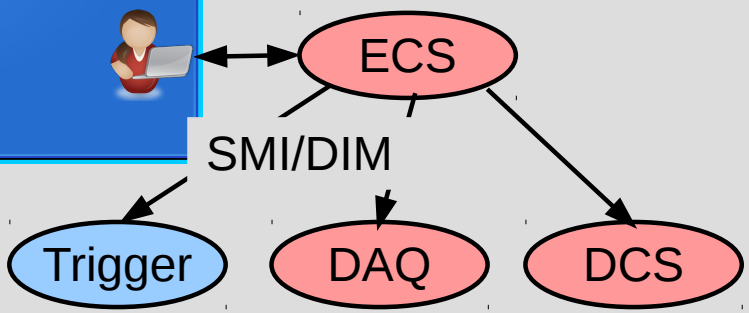


ltuproxy
- GO_STDALONE/GO_GLOBAL
- START/STOP -CTP emulation for STDALONE runs

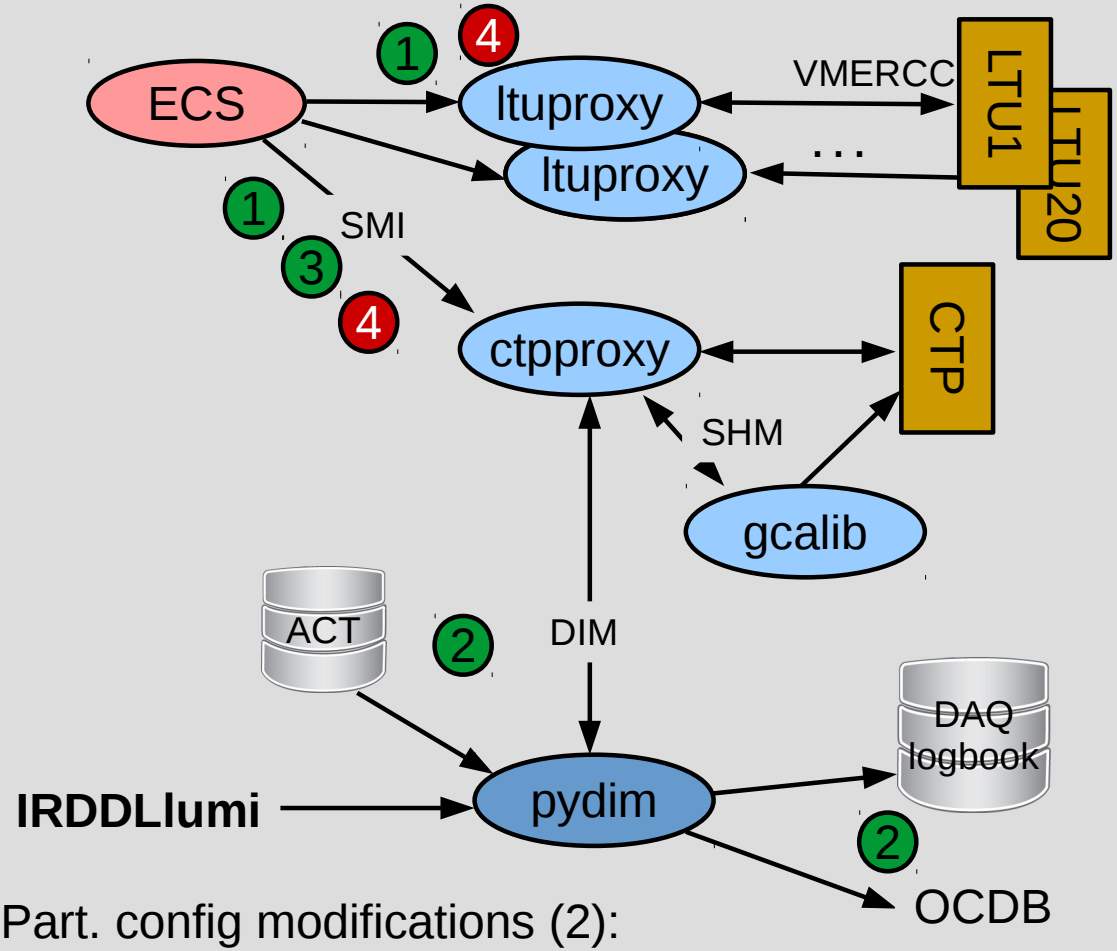
ctpproxy
- INIT_PARTITION: check/allocate resources
- START_PARTITION: set CTP boards, release triggers
- STOP_PARTITION: halt triggers, release resources

pydim
- Publishing online DIM services
- Making ctpproxy 'lighter'
- Big part written in python

Run control



Global run:



Part. config modifications (2):
- list of readout detectors
- removal of classes (ctp inputs filter)
- class downscaling recalculation w.r.t to current lumi

1) GO_GLOBAL: for participating LTUs. INIT_PARTITION name
2) Partition definition downloaded from ACT and **modified**, CTP resources checked, DAQlogbook updated, corresponding config file for OCDB created
3) START_PARTITION name: CTP boards programmed, triggers released
4) STOP_PARTITION name: stop triggers, reprogram CTP (release resources), GO_STDALONE for corresponding LTUs

gcalib: calibration triggers, several Hz for any detector

Monitoring

Counters

- Monitoring signals and slow control (temperature/power)
- 1760 counters in CTP, 24 in each LTU board
- Read 1/min
- CTP counters archived in sequential files and RRD, subset in OCDB
- WEB interface to RRD available

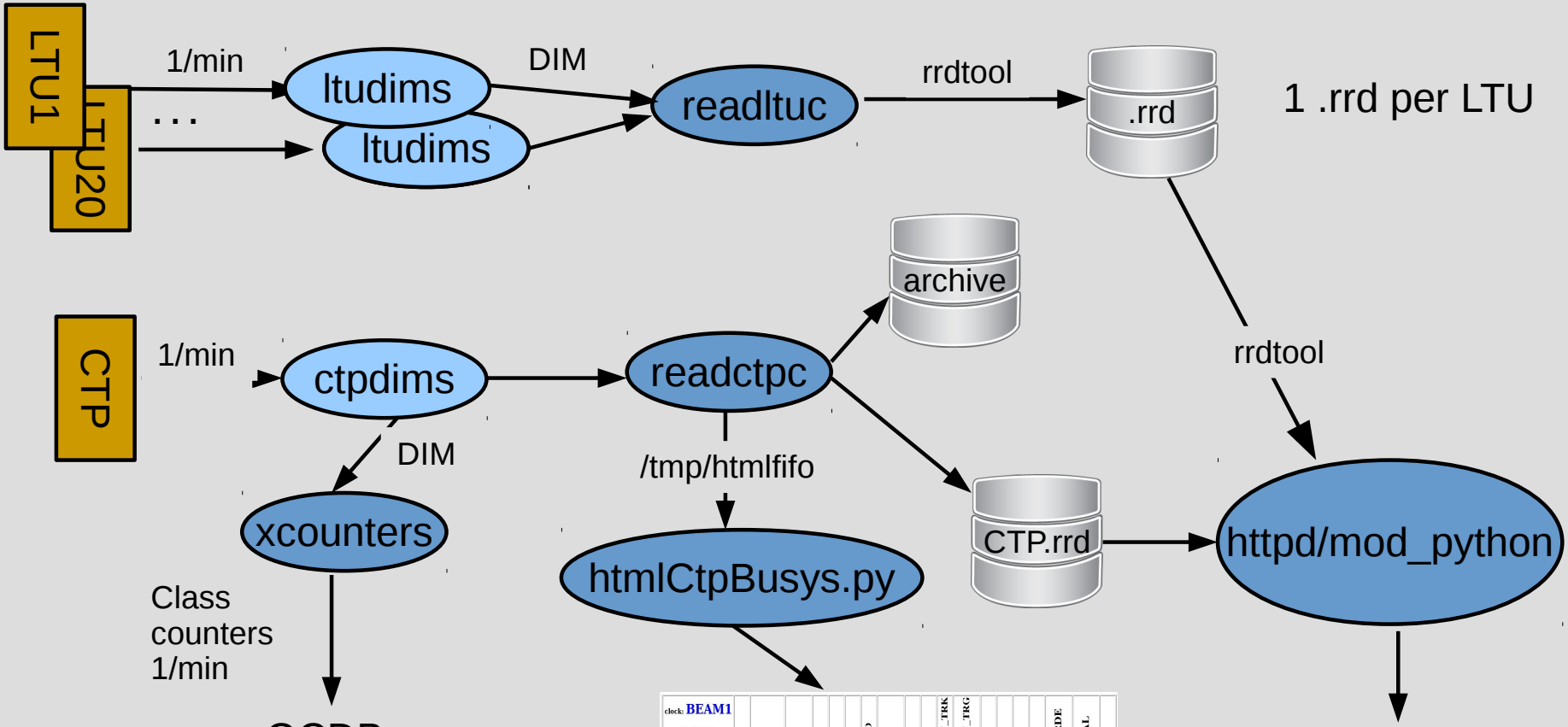
SSM

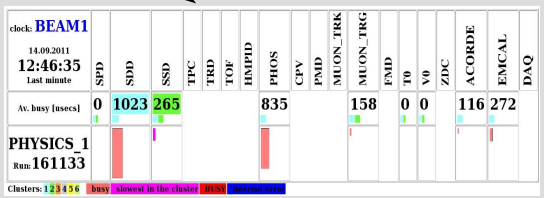
- Each VME board equipped with 1M words/32bits memory = ~27ms of 32 signals/40MHz
- 2 basic modes:
  - Recorder (before/after modes)
  - Generator
- Used a lot during development, main usage now:
  - Checking the signature of CTP inputs
  - CTP inputs alignment (SMAQ)
  - Orbit counter synchronisation on LM0 board (run2)

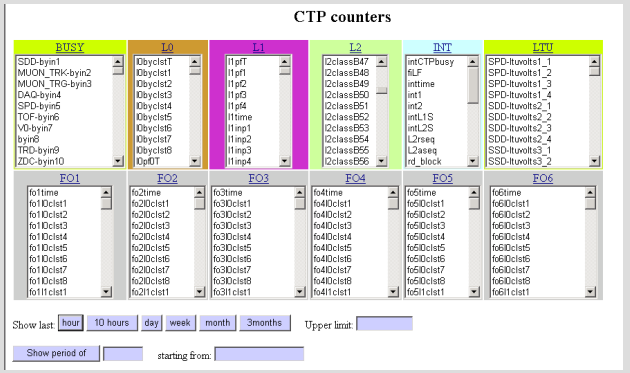
DQM

- Root application supplying many histograms from data acquired by:
  - Reading ctp counters (1/min, DIM)
  - Subset of events from DAQ stream
  - Resulting histograms, available only during data taking, are stored in DAQlogbook

Counters







ltudims -started together with ltuproxy
ctpdims -an independent daemon (parallel to ctpproxy)

readltuc,readctpc -DIM clients running on alitri server

SSM, expert interface

INMON

L0 inputs

Snapshot Memory (SSM)
1048576 BCs, 32 bits/signals =
26.2ms = 294 orbits

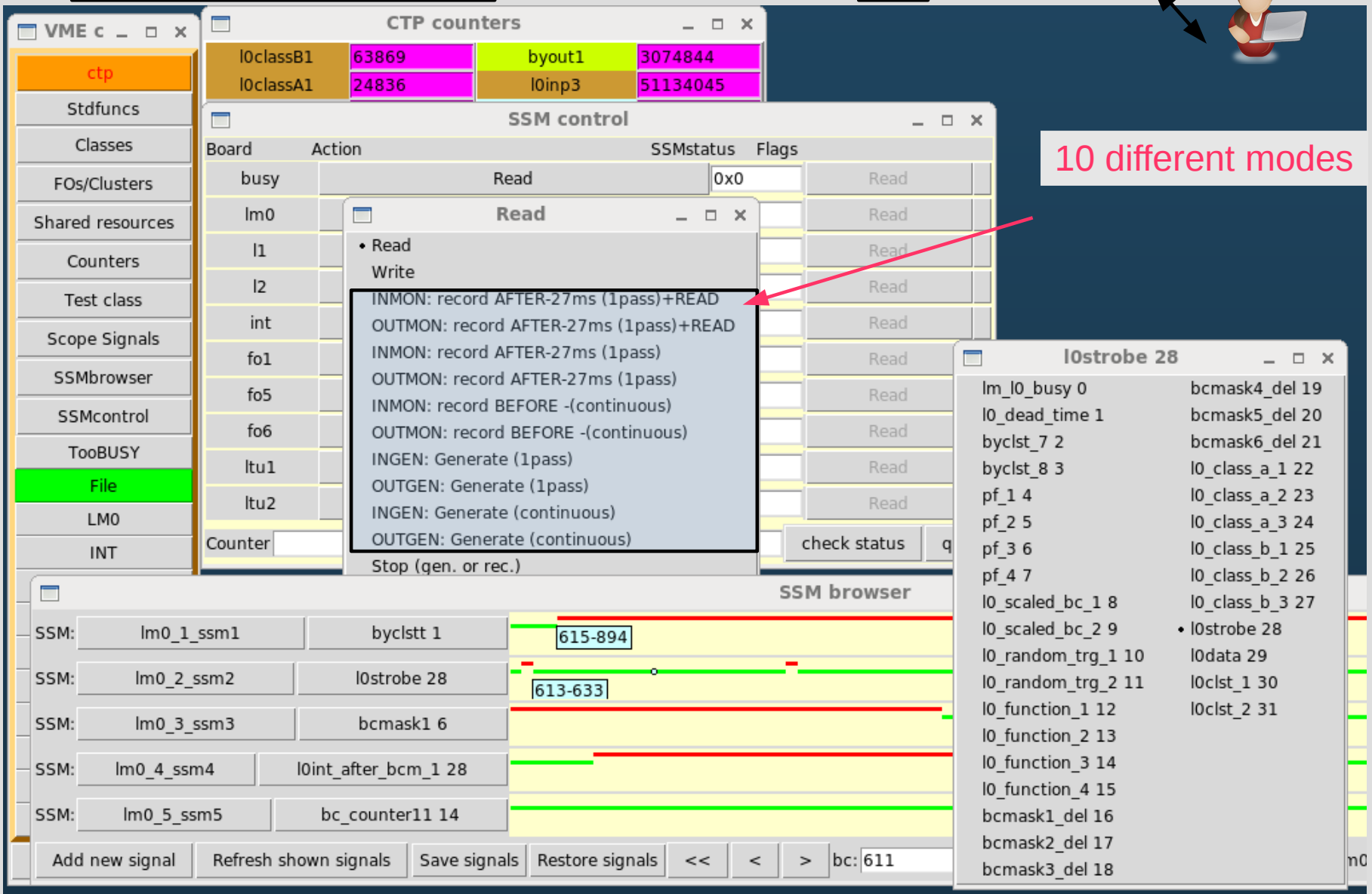
~1 (20 LM0) sec

CPU

CTP

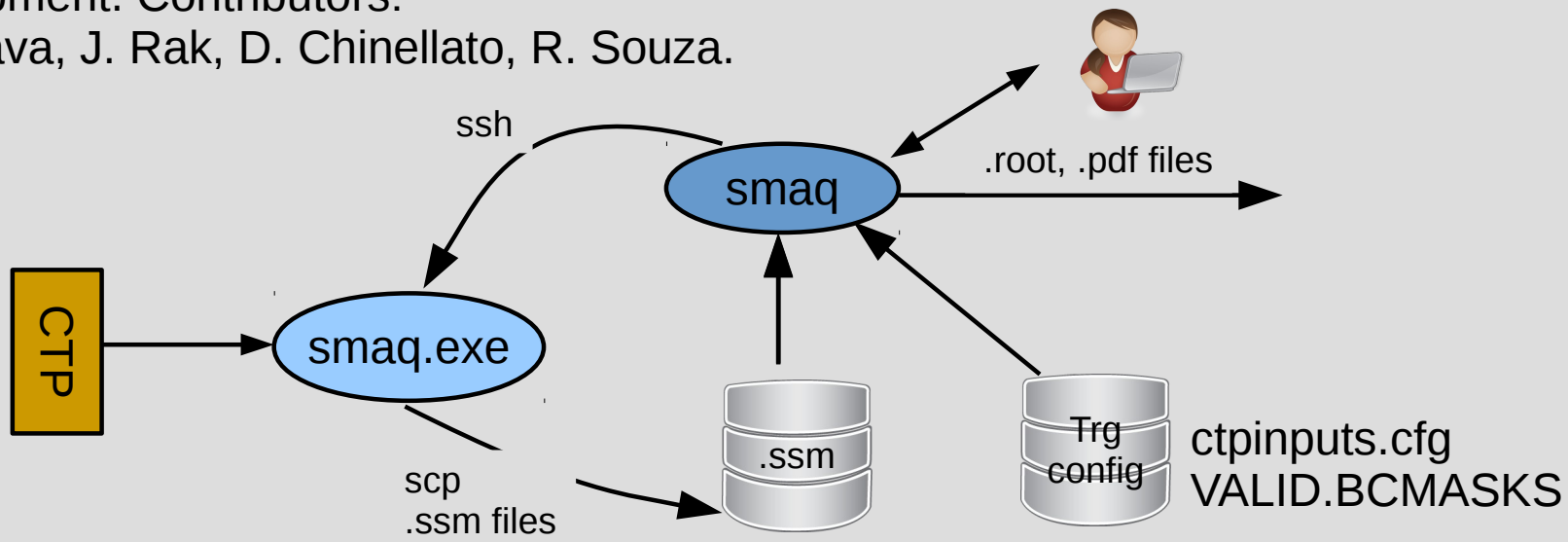
smaq.exe

.ssm



SSM, SMAQ

SMAQ (Snapshot Memory AcQuisition) underwent probably through wildest development. Contributors:
R. Lietava, J. Rak, D. Chinellato, R. Souza.

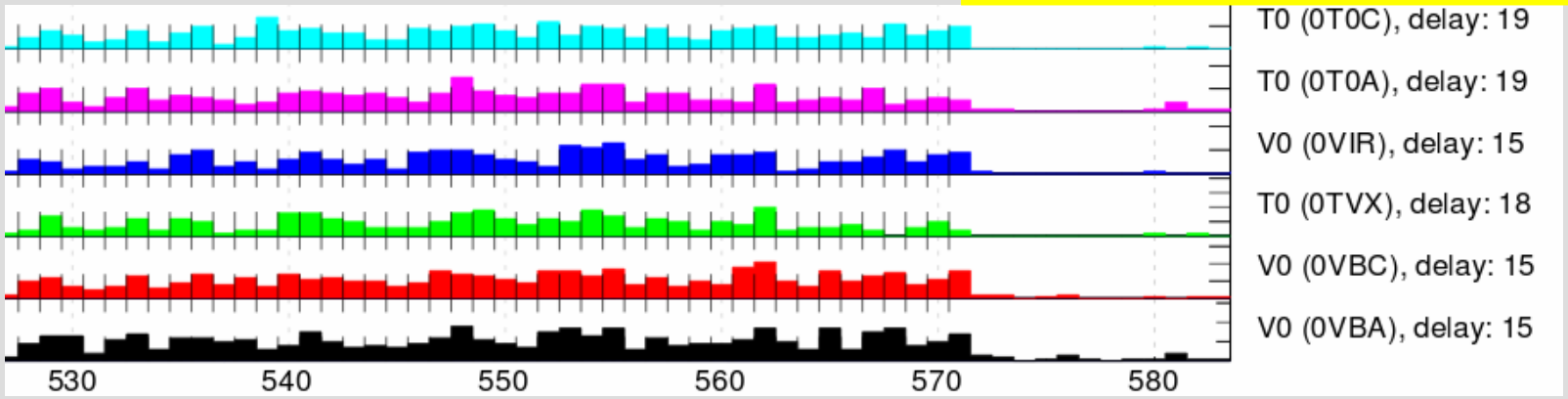


2 types of plots:

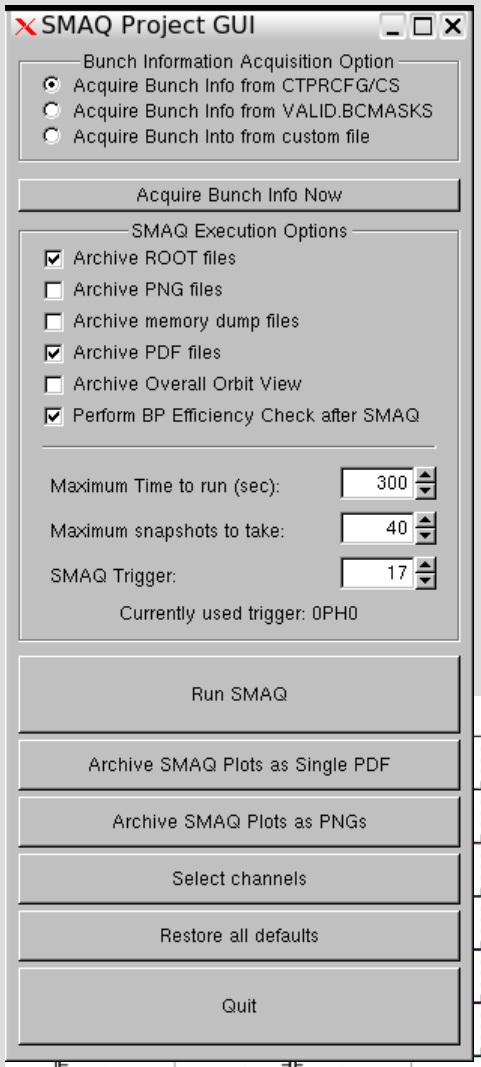
a

- Orbit profile -overlaying orbits
- Correlation analysis showing relation of 2 signals in time - convenient for dense filling schemes, where orbit profile becomes useless

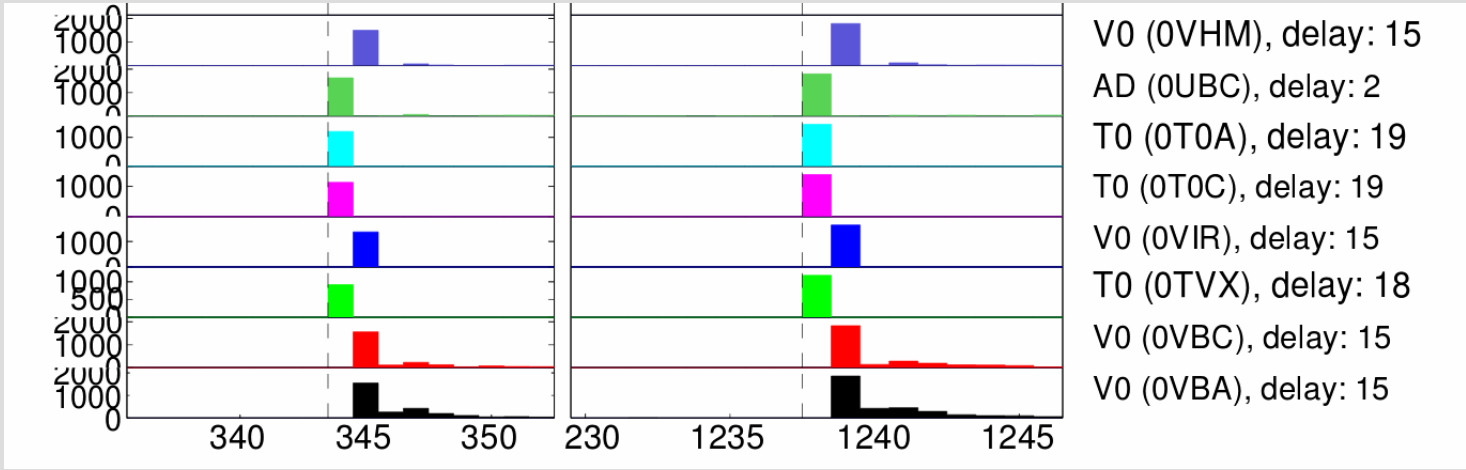
25ns_2220b_2208_1940_2036_96bpi_24inj



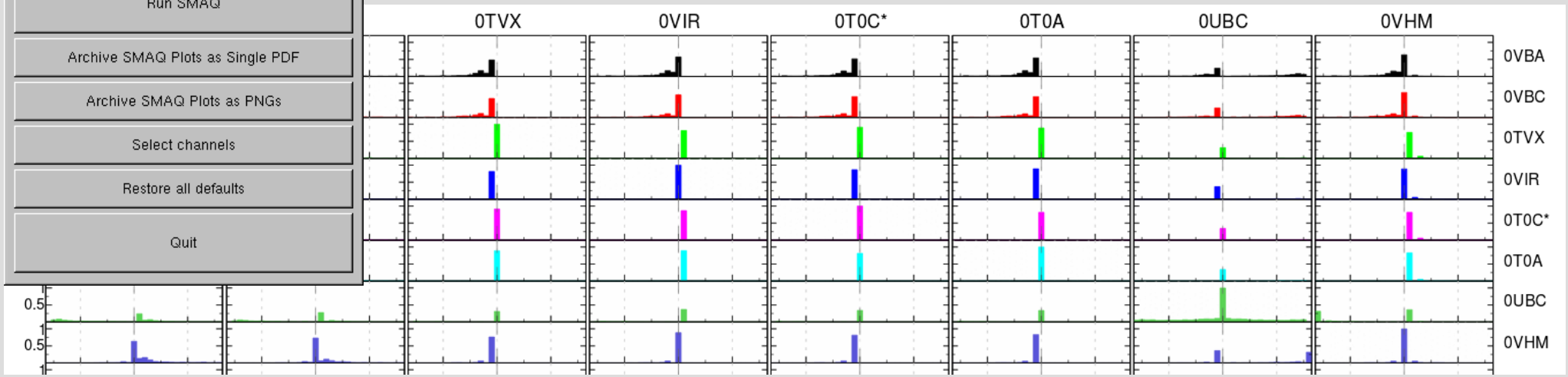
SSM, SMAQ



Orbit profile

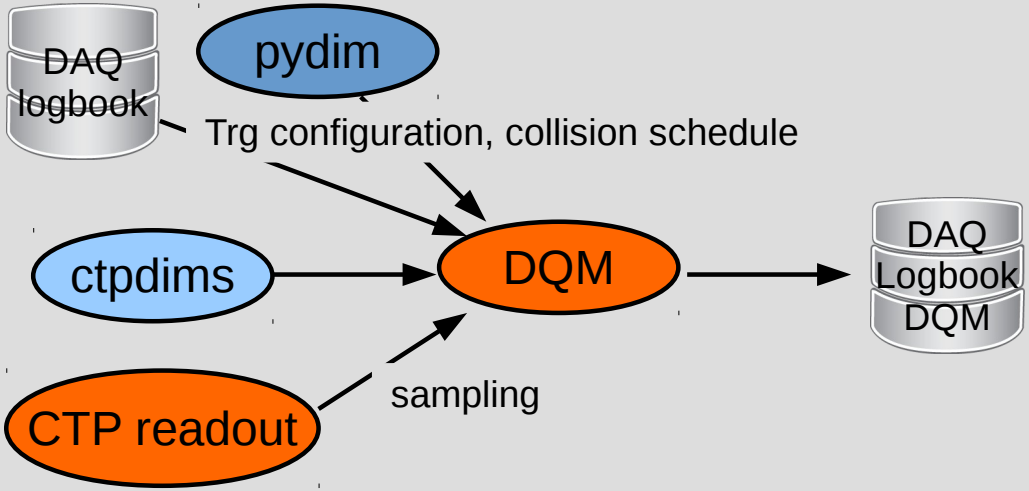


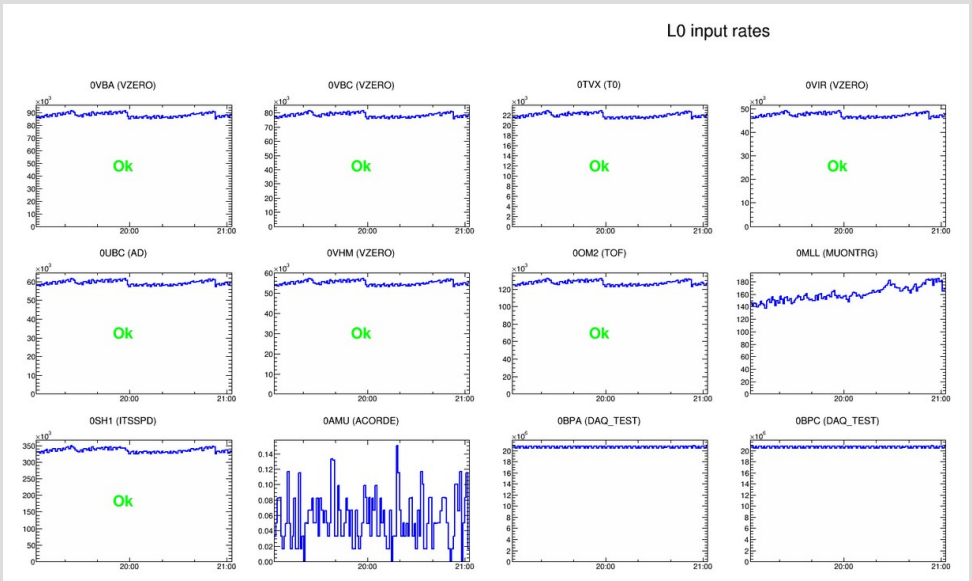
Correlation plots, reference: 0T0C

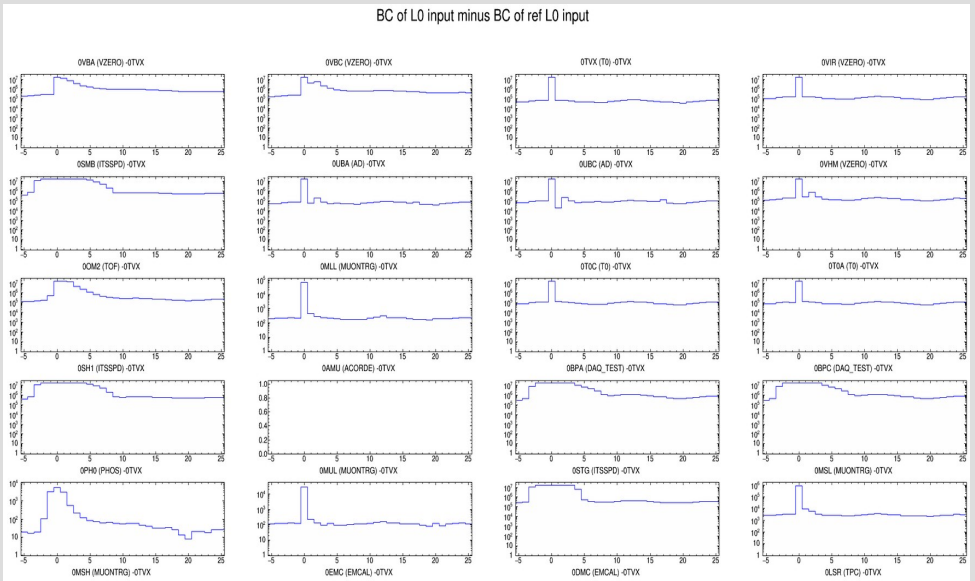


DQM

Trigger DQM follows Alice DQM standard developments. Contributions came from R. Lietava, A. Palaha, E. Kryshen who wrote last stable version known today and M. Bombara -current maintainer.

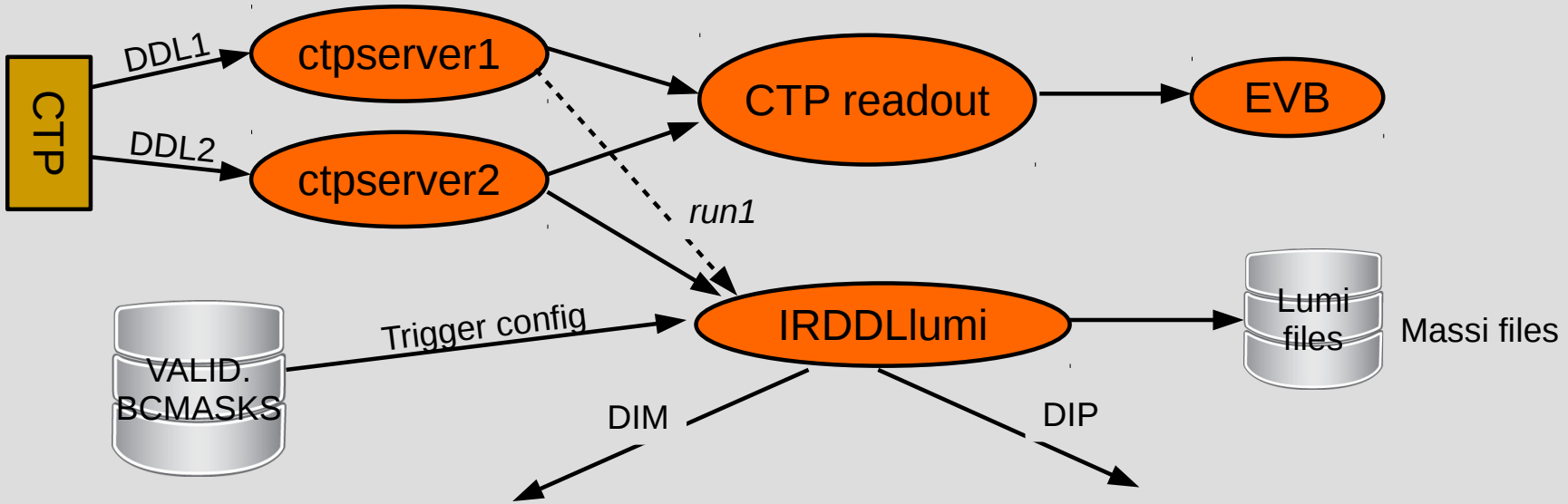






IR/DDL luminosity

CTP sends over DDL2 **orbitid+bcid** of bunch crossings when its inputs were active.
Instantaneous and bunch-by-bunch luminosity is calculated (1/min, using usually 0TVX signal) and sent back to LHC as a feedback.
IRDDLlumi project is maintained by I. Kralik.



IR_MONITOR/CTP/Luminosity
Used by:
- DCS big screen in ARC
- pydim in case of automatic downscaling

LHC feedback:
dip/ALICE/CTP_IR/Luminosity_IR
dip/ALICE/LHC/LumiPerBunch

Summary

- Trigger system operation has been reliable during run1/2
- Most difficult time was 2015, when we were finishing run2 upgrade
- An experience with LM0 board good also for run3
- 2017 probably smoothest operation (no firmware update!)

Spare slides

Spare slides

References

http://alicetrigger.web.cern.ch/alicetrigger

http://ttc.web.cern.ch/TTC/

https://ific.uv.es/tical/old/rod/docs_vme_daq/VME_API_Linux.pdf

http://smi.web.cern.ch/smi
http://dim.web.cern.ch/dim

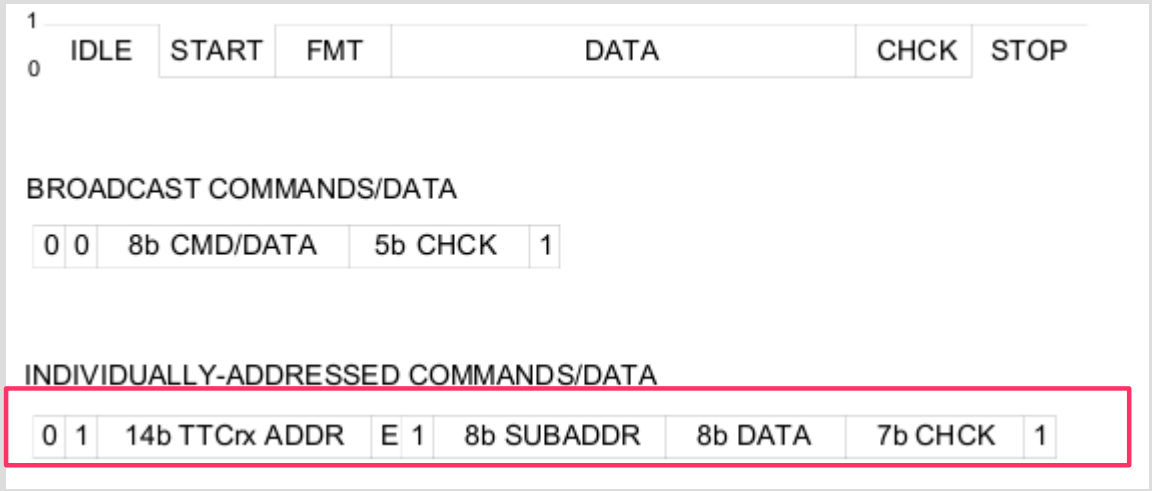
http://onesis.org

https://oss.oetiker.ch/rrdtool

https://redis.io

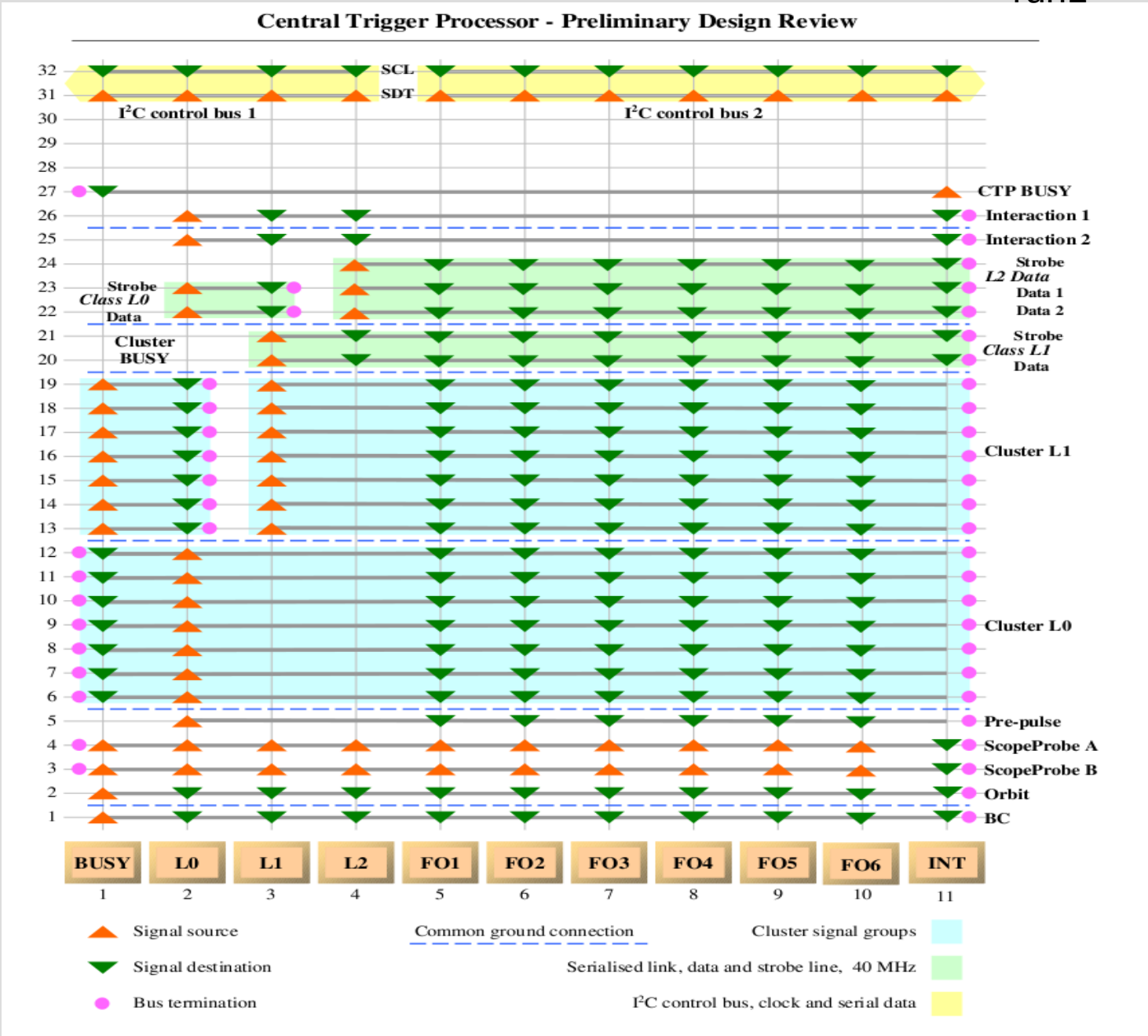
TTC protocol

- downstream connection only
- two 40MHz streams (A/B) multiplexed over a passive optical fibre
  - A: 1 bit, first level accept, synchronous
  - B: 16 data bits in 42 bits frame, asynchronous, Hamming code capable of double error detection and single bit error correction.



CTP backplane

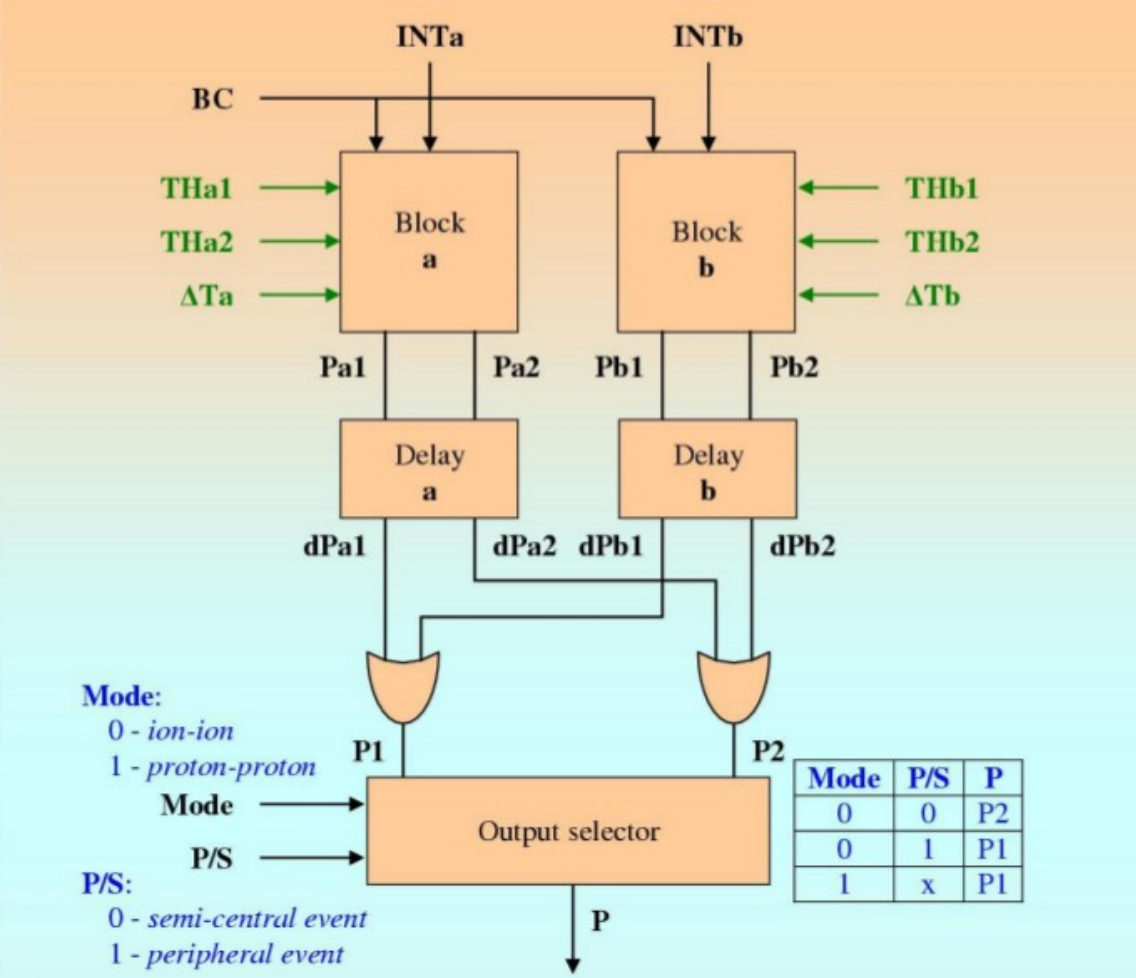
run1

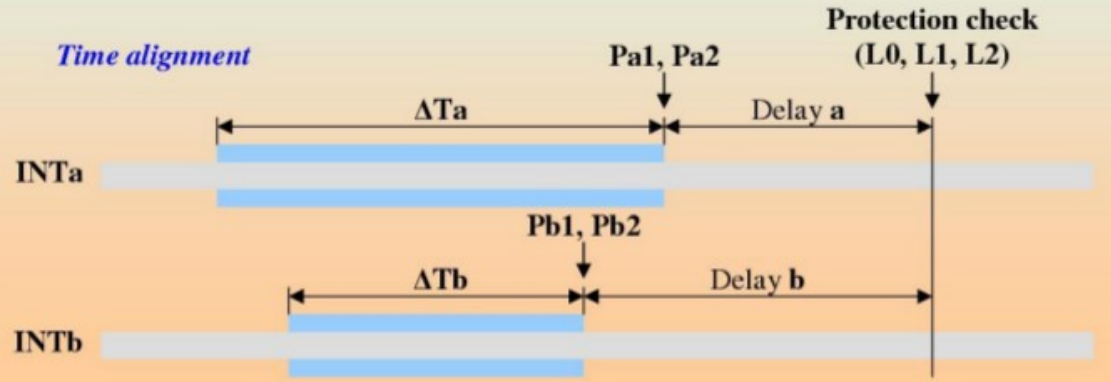


CTP Past-Future protection

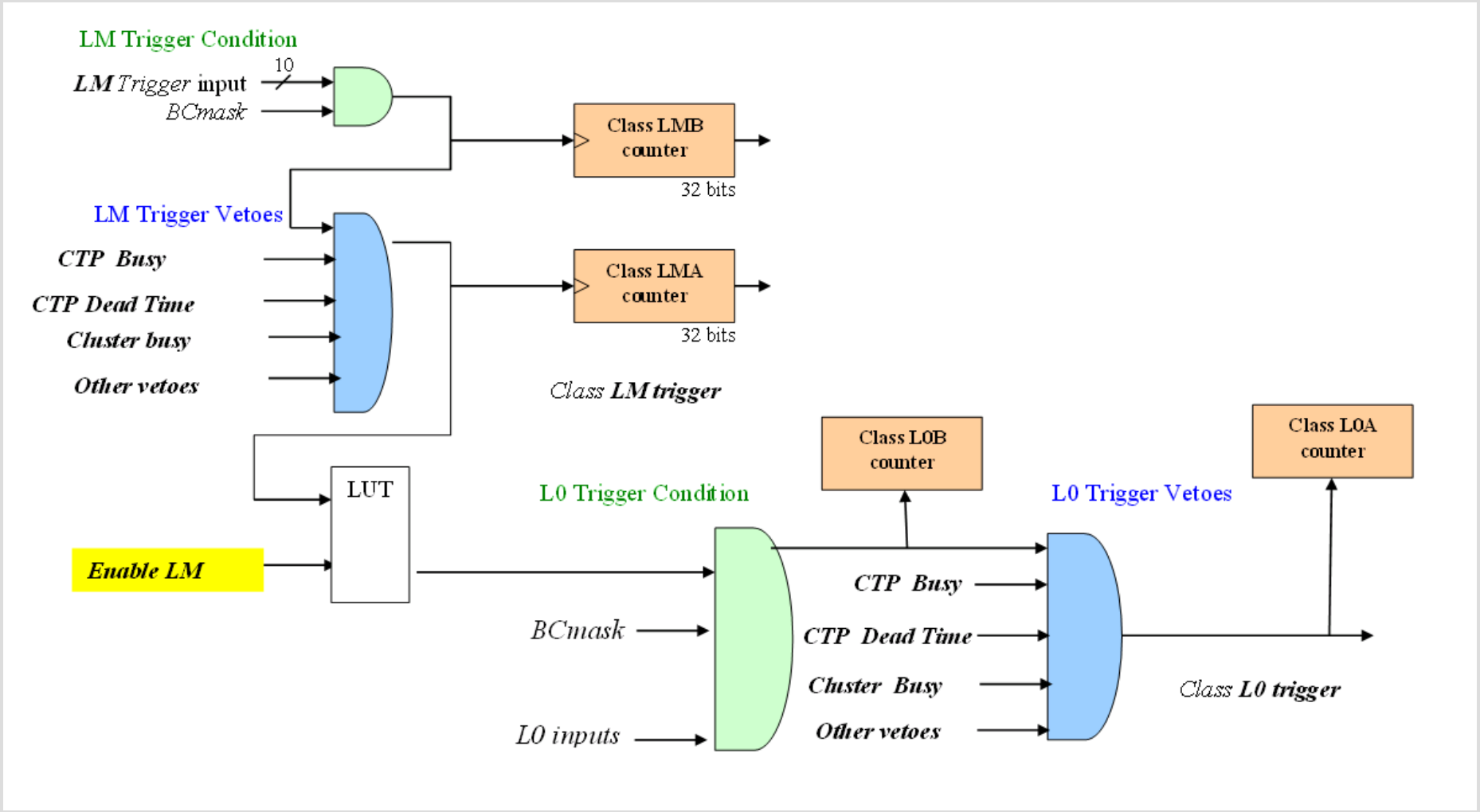
- 4 independently programmable circuits at each level
- 2 identical boards, based on dual-port memory
- Sliding time-window during which the interaction signal (INTa/b is counted
- Programmable parameters:
  - Protection interval (Δta/b)
  - 2 Threasholds (Tha1/2, Thb1/2)
  - Output delay (a/b)
  - Output logic function
- Delay and alignment of output signals

Revisited for LM0 in Run2:
- Incorporating new LM inputs





LM0 logic LM part



Run2 Data format DDL2

Word	[31]	[30]	[29]	[28..12]	[11..0]
1	**1 (Orbit)**	**1 (first word)**	**Err**	Don`t care	**Orbit** number [23..12]
2	**1**	**0 (second w.)**	**Err**	Don't care	**Orbit** number [11..0]
3	0 (BC)	1 (first word)	**L0ti[18..1]**		**BC** number [11..0]
4	0	**0 (second w.)**	**L0ti[48..19]**		
⋮			⋮		⋮
n	0	1	**L0ti[18..1]**		**BC** number [11..0]
n+1	0	**0**	**L0ti[48..19]**		
⋮			⋮		⋮
251	0	1	**L0ti[18..1]**		**BC** number [11..0]
252	0	**0**	**L0ti[48..19]**		
253	0	0	0	*Don`t care*	*Incomplete record* (hFFF)

Run2 Data format DDL1

L1 Data serial format						
Serial bit	*Data*	*Sequence List*		*L1 Message*		*N of bits*
		Word	*Bit*	*Word*	*Bit*	
1	Spare	0	15	1	11	1
2	**CIT**	0	14	1	10	1
3..6	**RoC[4..1]**	0	13..10	1	9..6	4
7	**ESR**	0	9	1	5	1
8	**L1SwC**	0	8	1	4	1
9..10	**L1Class[100..99]**	0	7..6	1	3..2	2
11..12	**L1Class[98..97]**	1	15..14	1	1..0	2
13..24	**L1Class[96..85]**	1	13..2	2	11..0	12
25..26	**L1Class[84..83]**	1	1..0	3	11..10	2
27..36	**L1Class[82..73]**	2	15..6	3	9..0	10
37..42	**L1Class[72..67]**	2	5..0	4	11..6	6
43..48	**L1Class[66..61]**	3	15..10	4	5..0	6
49..58	**L1Class[60..51]**	3	9..0	5	11..2	10
59..60	**L1Class[50..49]**	4	15..14	5	1..0	2
61..72	**L1Class[48..37]**	4	13..2	6	11..0	12
73..74	**L1Class[36..35]**	4	1..0	7	11..10	2
75..84	**L1Class[34..25]**	5	15..6	7	9..0	10
85..90	**L1Class[24..19]**	5	5..0	8	11..6	6
91..96	**L1Class[18..13]**	6	15..10	8	5..0	6
97..106	**L1Class[12..3]**	6	9..0	9	11..2	10
107..108	**L1Class[2..1]**	7	15..14	9	1..0	2
	Spare	7	13..0			

L2 Data serial format						
Serial bit	*Data*	*Sequence List*		*L2a Message*		*N of bits*
		Word	*Bit*	*Word*	*Bit*	
1	**L2arF**					1
2..13	**BCID[12..1]**			1	11..0	12
14..25	**OrbitID[24..13]**			2	11..0	12
26..37	**OrbitID[12..1]**			3	11..0	12
38	Spare	0	30	4	11	1
39	**ESR**	0	29	4	10	1
40	**CIT**	0	28	4	9	1
41	**L2SwC**	0	27	4	8	1
42..49	**L2Cluster[8..1]**	0	26..19	4	7..0	8
50..52	**L2Class[100..98]**	0	18..16	5	11..9	3
53..61	**L2Class[97..89]**	1	31..23	5	8..0	9
62..68	**L2Class[88..82]**	1	22..16	6	11..5	7
69..73	**L2Class[81..77]**	2	31..27	6	4..0	5
74..84	**L2Class[76..66]**	2	26..16	7	11..1	11
85	**L2Class[65]**	3	31	7	0	1
86..97	**L2Class[64..53]**	3	30..19	8	11..0	12
98..100	**L2Class[52..50]**	3	18..16	9	11..9	3
101	**L2Class[49]**	4	31	9	8	1
102..109	**L2Class[48..25]** or **Detector[24..1]**	4	30..23	9	7..0	8
110..116		4	22..16	10	11..5	7
117..121		5	31..27	10	4..0	5
122..125		5	26..23	11	11..8	4
126..132	**L2Class[24..18]**	5	22..16	11	7..1	7
133	**L2Class[17]**	6	31	11	0	1
134..145	**L2Class[16..5]**	6	30..19	12	11..0	12
146..148	**L2Class[4..2]**	6	18..16	13	11..9	3
149	**L2Class[1]**	7	31	13	8	1
	Spare	7	14..0	13	7..0	