GSI MBS – Multi Branch System

Bastian Löher

18.05.2016

Institut für Kernphysik – Universität zu Köln

GSI Helmholtzzentrum für Schwerionenforschung GmbH

Today

- Introduction – What is a DAQ and what is MBS
- SBS – A simple MBS
  - hardware (RIO, TRIVA, VULOM, TRIXOR, ...)
  - m_read_meb (f_user.c)
- MBS – Multiple crates
- Use cases:
  - MBS at Duke University for Gamma^3
  - MBS at the R^{3}B setup at GSI
- TRLOII – A flexible trigger logic
- nurdlib – The nustar readout library
- ucesb – Unpack and check every single bit (the sorting code)
- Outlook

What is a data acquisition system (DAQ)?

- Handle trigger signals from detectors
- Make a trigger decision
- Read data from hardware to memory
- Check data integrity
- Transport data through the network
- Store data to disk

What is MBS?

- Handle trigger signals from detectors
- Make a trigger decision
- Read data from hardware to memory
- Check data integrity
- Transport data through the network
- Store data to disk

What is MBS?

- Handle trigger signals from detectors
- Make a trigger decision
- Read data from hardware to memory
- Check data integrity
- Transport data through the network
- Store data to disk

Responsibility of the user

What is MBS?

- Some facts:
  - Started in 1993
  - Over 90 systems installed world wide (2011)
  - Based on real-time LynxOS or Linux
  - Support for VME, VXI, CAMAC, FASTBUS, PCI & PCIe
  - Data transport via address mapped buses or TCP/IP

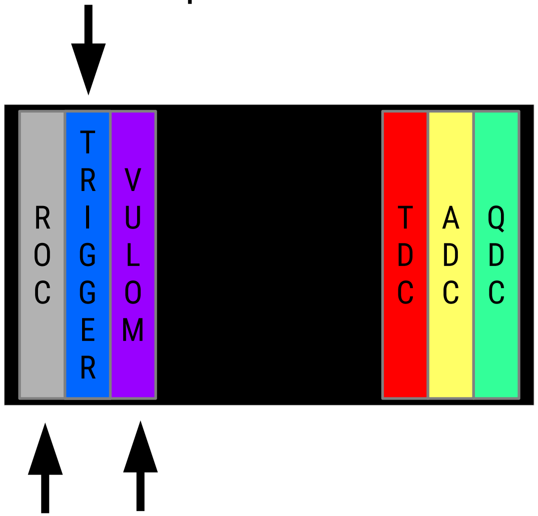
SBS – A simple (single) branch system

- Only a single VME crate or PC
- Any MBS consists of two parts:
  - Hardware:
    - Trigger module (TRIVA, TRIXOR, VULOM)
    - Readout processor (RIO2, RIO3, RIO4, x86 PC)
  - Software:
    - m_read_meb – Data readout to internal data pipe
    - m_collector – Collect data from pipe to event buffer
    - m_transport – Transport data over network
    - m_stream_serv – Serve additional data stream (e.g. for online)

SBS – A simple (single) branch system

- VME crate example
  - Readout speed
    - Single cycle: ~7 MB/s
    - 64 bit block transfer: ~40 MB/s
    - 2eSST: ~150 MB/s
    - VME access time: ~500 ns
    - Trigger to readout latency: 5-10 us

TRIVA7 trigger module
- 15 trigger inputs
- Deadtime output

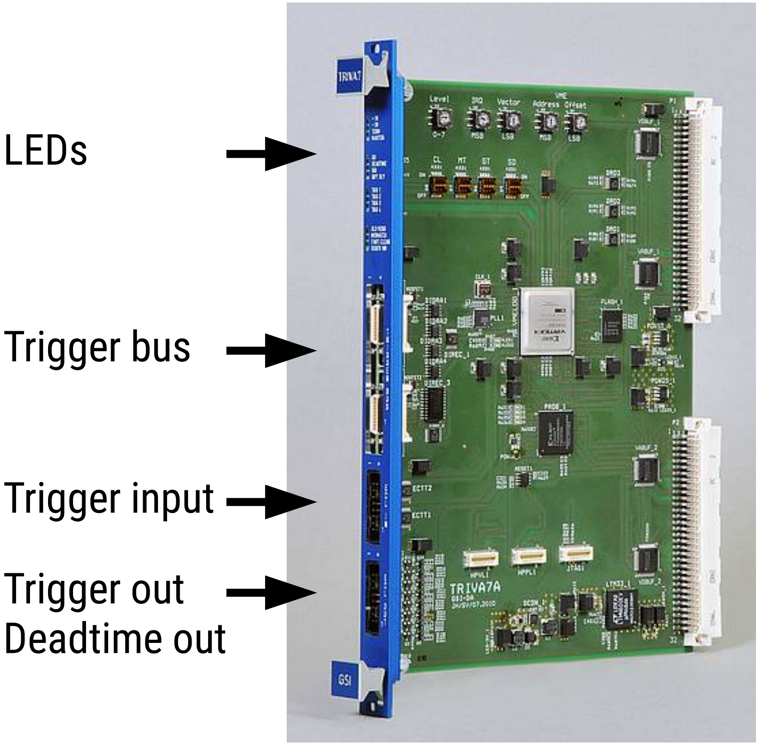


ReadOut Controller:
- Gigabit ethernet
- Network booted (no disk)
- Either Linux or LynxOS

VME User LOgic Module:
- Provides deadtime locking with TRIVA7
- 16 ECL inputs and outputs
- 2 LEMO inputs and outputs

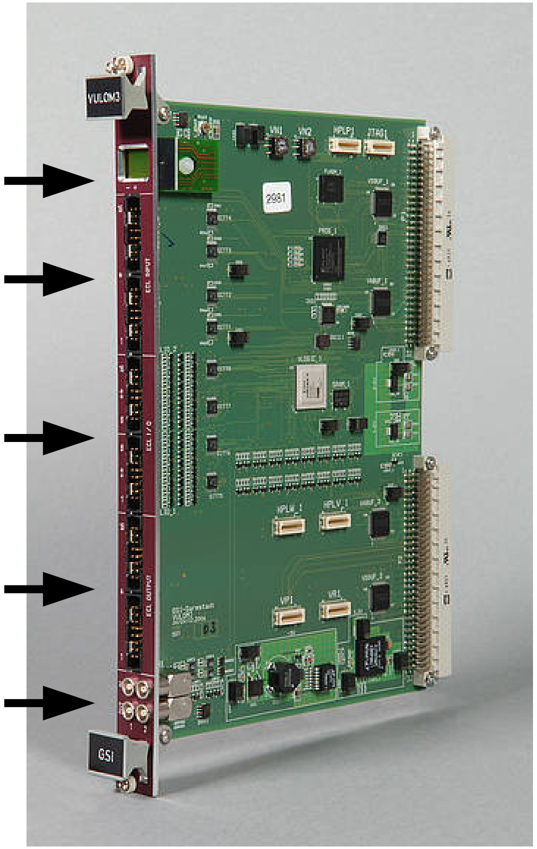
TRIVA7 – Trigger module

- Shows DAQ status via LEDs
- Receives 4bit encoded trigger number
  - 13 experiment triggers
  - 14, 15 reserved for start and stop
- Deadtime / busy output
- Master trigger output
- Connects several MBS subsystems via trigger bus
- Generates interrupt to start data readout



VULOM4 – User Logic module

- Standard firmware:
  - 13 trigger inputs
  - Deadtime locked trigger outputs
  - Delay (input-output): ~30 ns
  - Jitter: ~2.5 ns
  - Connects to TRIVA7 for signal exchange



LCD display

16 Inputs

Triva comm

16 outputs

2 inputs
2 outputs

SBS – A simple (single) branch system

- Software:
  - Today developed and maintained by Nik Kurz (GSI)
  - Production version: 6.2
  - Most parts are generic, i.e. experiment independent
  - User needs to focus on:
    - `m_read_meb` – readout code
    - `setup.usf` – user setup file for each subsystem

SBS – A simple (single) branch system

- `m_read_meb`:
  - Main readout loop, contains three entry points for user code
    - `f_user_get_virt_ptr()`
      - Create virtual pointers to the hardware (memory mapping or DMA setup)
    - `f_user_init()`
      - Setup hardware (configure settings, set thresholds, etc …)
    - `f_user_readout()`
      - Read data from hardware
  - These must be implemented in the `f_user.c` file and compiled into the complete `m_read_meb`

SBS – A simple (single) branch system

- `setup.usf`:
  - Main setup file for a single system with many options, e.g.
    - `LOC_MEM_BASE:` vme address start
    - `LOC_MEM_LEN:` vme memory size
    - `LOC_PIPE_BASE:` data pipe address
    - `PIPE_SEG_LEN:` data pipe size
    - `PIPE_LEN:` max. number of sub-events in pipe
    - `RD_FLAG:` switch readout on/off
    - `COL_MODE:` switch local event building on/off
    - `TRIG_CVT:` trigger conversion delay

SBS – A simple (single) branch system

- `setup.usf`:
  - Main setup file for a single system with many options, e.g.
    - `LOC_MEM_BASE:` vme address start
    - `LOC_MEM_LEN:` vme memory size
    - `LOC_PIPE_BASE:` data pipe address
    - `PIPE_SEG_LEN:` data pipe size
    - `PIPE_LEN:` max. number of sub-events in pipe
    - `RD_FLAG:` switch readout on/off
    - `COL_MODE:` switch local event building on/off
    - `TRIG_CVT:` trigger conversion delay

Usually does
Not need to
Be touched

SBS – A simple (single) branch system

- Directory structure:
  - `rio4-1:`
    - `Makefile –` Compile m_read_meb
    - `f_user.c –` User functions
    - `setup.usf –` User setup file
    - `start.scom –` Startup script
    - `stop.scom –` Shutdown script
    - `m_read_meb –` compiled readout

SBS – A simple (single) branch system

- Start and shutdown scripts (VME, single crate):
  - Any file with .scom can be used from MBS command line

```
start.scom:
start task m_util
load setup setup.usf
set trig_mod
enable irq
start task ./m_read_meb
start task m_collector
start task m_transport
start task m_stream_serv
start task m_daq_rate
set stream 1
start acq
```

```
stop.scom:
stop task m_daq_rate    -kill
stop task m_stream_serv -kill
stop task m_transport   -kill
stop task m_collector   -kill
stop task m_read_meb    -kill
stop task m_util        -kill
```

SBS – A simple (single) branch system

- Starting MBS:

```
rio4-1> resl    # reset local MBS
rio4-1> mbs     # start MBS command line

mbs> @start     # execute start.scom script
. . .
-rio4-1 :collector  :acquisition running

mbs> sho(w) acq(uisition)
-rio4-1 :util :Collected: 0.0164 MB,   1 Buffers, 17 Events.
-rio4-1 :util :Rate    :    0 KB/s, 0 Buffers/s, 1 Events/s

mbs> @stop      # execute stop.scom script
```

SBS – A simple (single) branch system

- Writing data:
  - Requires a running RFIO server on the fileserver PC
  - MBS supplies `rawDispRFIO64`
  - Storage location is specified in `filenum.set`

```
filenum.set:
rfiocopy:lxgs08:/data/lmd/run001_
1000
```

```
mbs> connect rfio lxgs08 -diskserver   # Connect to server

mbs> open file size=1000 -auto -rfio   # Open new file
mbs> close file                        # Close file
```

SBS – A simple (single) branch system

- Data format:
  - LMD (list mode data) format encapsulates data from modules in **subevents**, which are combined into one **event** per trigger
  - Each event has a unique **event number** and can contain any number of subevents
  - Each subevent within an event has a unique combination of **type-subtype-control-subcrate** numbers used for sorting
  - The maximum size of subevents is specified in the setup.usf file
  - The `event_api` library can be used to unpack / sort LMD files
  - In reality we make use of the **`ucesb`** unpacker

SBS – A simple (single) branch system

- Monitoring – The `rate` program

```
rio4-1> rate
# Event building                | Server | File output
#  MB     Events Kb/sec  Ev/sec | Kb/sec |  Kb/sec Index
  1714    615378   16.4      10 |    0.0 |     0.0  0001  cl
  1714    615388    0.0      10 |    0.0 |     0.0  0001  cl
  1714    615398    0.0      10 |    0.0 |     0.0  0001  cl
  1714    615408   16.4      10 |    0.0 |     0.0  0001  cl
  1714    615418   16.4      10 |    0.0 |     0.0  0001  cl
  1714    615428    0.0      10 |    0.0 |     0.0  0001  cl
...
```

Today

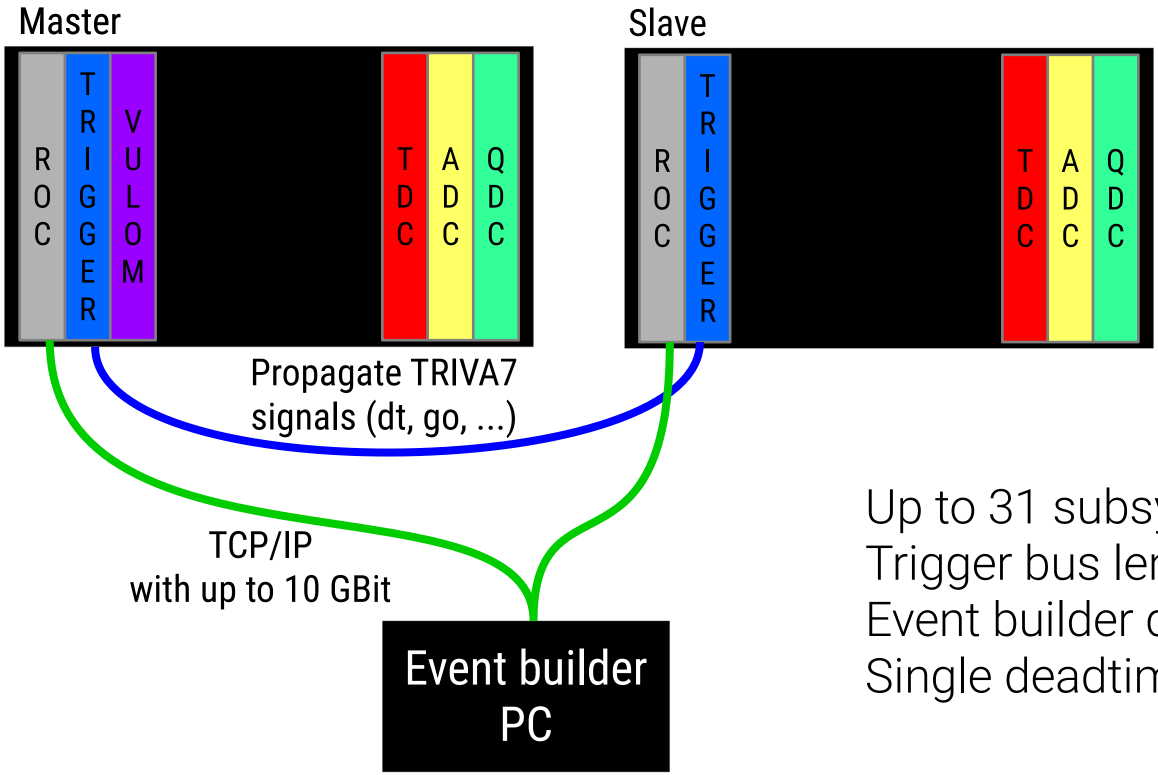
- Introduction – What is a DAQ and what is MBS
- SBS – A simple MBS
  - hardware (RIO, TRIVA, VULOM, TRIXOR, ...)
  - m_read_meb (f_user.c)
- MBS – Multiple crates ⇦
- Use cases:
  - MBS at Duke University for Gamma^3
  - MBS at the R^{3}B setup at GSI
- TRLOII – A flexible trigger logic
- nurdlib – The nustar readout library
- ucesb – Unpack and check every single bit (the sorting code)
- Outlook

SBS to MBS (multi branch system)

- Multiple subsystems require synchronisation based on
  - Trigger (single deadtime domain)
  - Timestamp (multiple deadtime domains)
- Needs separate event builder PC to combine subsystem subevent data
- Timestamp synchronisation needs time sorter PC
- Possible to combine both methods

SBS to MBS (multi branch system)

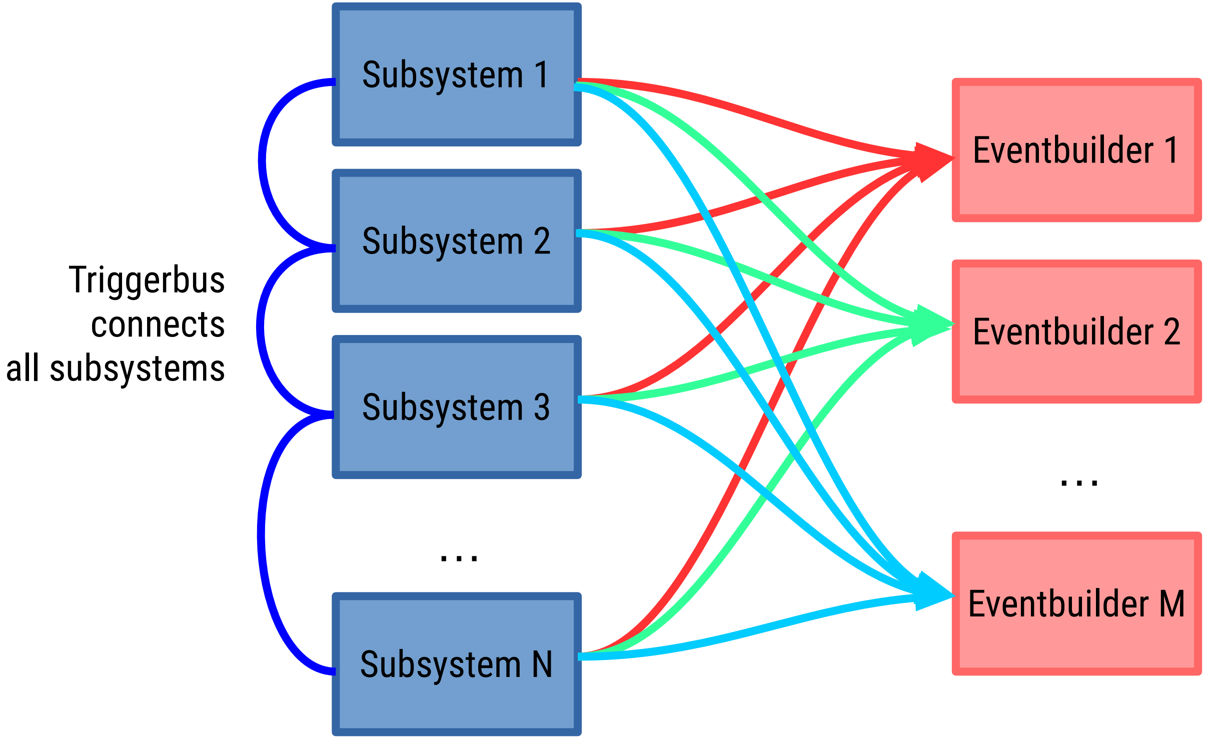
- Trigger synchronisation uses trigger bus



Up to 31 subsystems
Trigger bus length >250 m
Event builder data rate >500 MB/s
Single deadtime domain

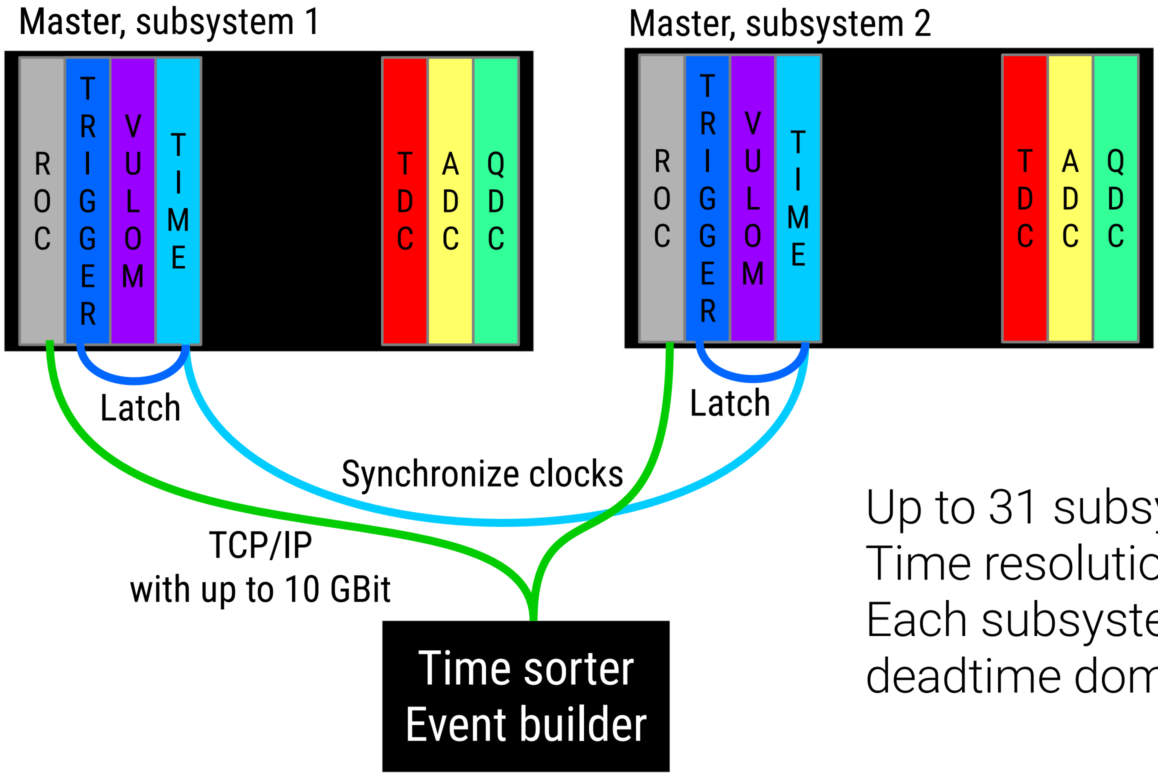
MBS – NxM configuration

- N subsystems with M event builders for high data rate applications



SBS to MBS (multi branch system)

- Timestamp synchronisation with timestamp modules



Up to 31 subsystems
Time resolution 8 ns (PCIe 1 ns)
Each subsystem has its own deadtime domain

SBS to MBS (multi branch system)

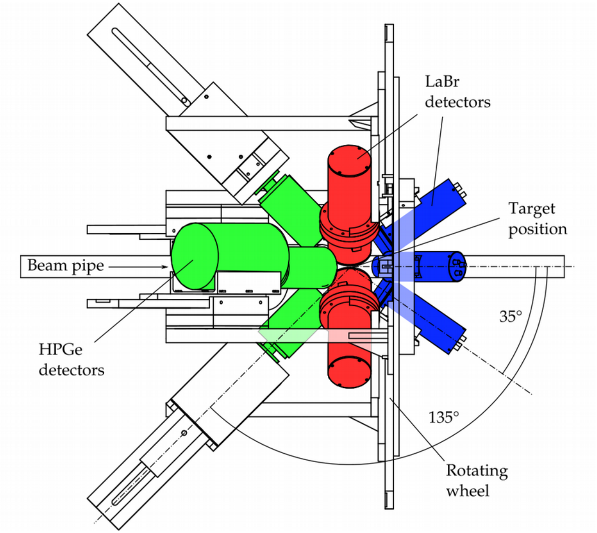
- Setup file `setup.mo` needed to specify layout
- Data senders (subsystems):
  - `DS_HOSTNAME_0 = „rio4-1“`
  - `DS_HOSTNAME_1 = „rio4-2“`
- Data readers (event builders)
  - `DR_HOSTNAME_0 = „x86g-1“`
- start.scom and stop.scom files look slightly different

Today

- Introduction – What is a DAQ and what is MBS
- SBS – A simple MBS
  - hardware (RIO, TRIVA, VULOM, TRIXOR, ...)
  - m_read_meb (f_user.c)
- MBS – Multiple crates
- Use cases:
  - MBS at Duke University for Gamma^3
  - MBS at the R^{3}B setup at GSI
- TRLOII – A flexible trigger logic
- nurdlib – The nustar readout library
- ucesb – Unpack and check every single bit (the sorting code)
- Outlook

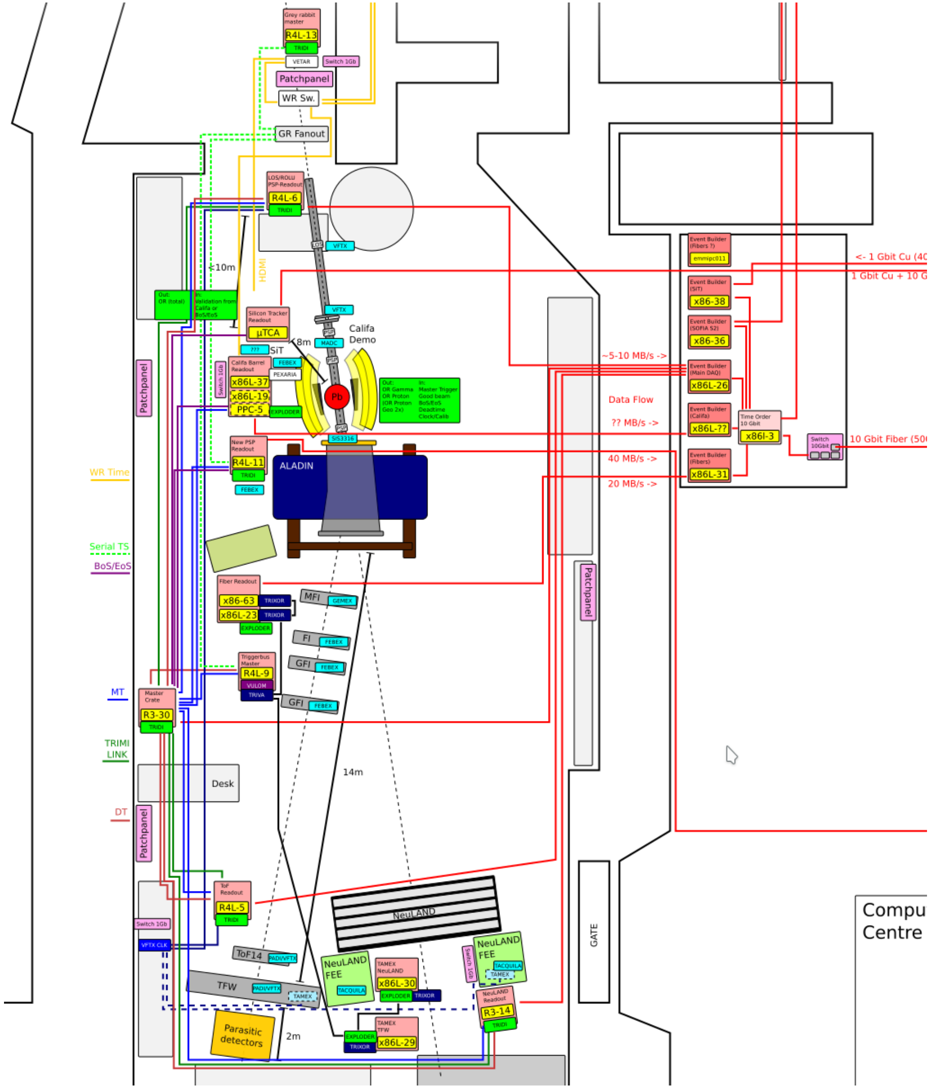
MBS @ Duke

- Readout of 4 HPGe and 4 LaBr detectors
- Single VME crate setup
  - Used for LaBr singles and coincidences
  - Data rate: 2-4 MB/s
  - Event rate: 6-10k Events/s at 20-30% deadtime
  - Deadtime / event: ~70 us
  - Uses TRLOII for trigger conditions and downscaling, scalers
  - Uses nurdlib for module readout
  - Uses ucesb for unpacking / sorting



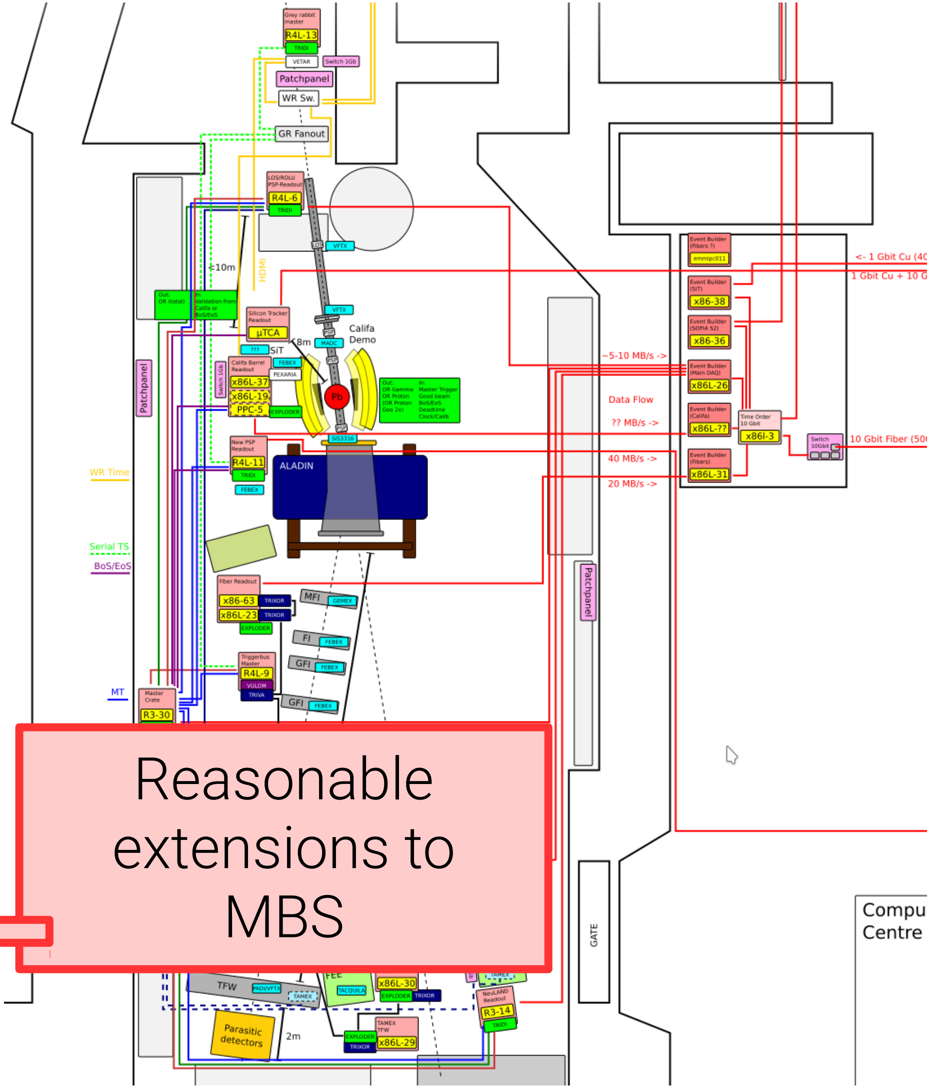
MBS @ R3B / LAND (oct 2014)

- Readout of 15 different detector types spread across 3 experimental sites
- 12 VME crates + 9 PCs, 6 event builder PCs, 10 Gbit Timeorder PC
- 5 deadtime domains, 2 trigger bus chains, 1 trimi link master
- Combined serial timestamp distribution and White rabbit timing
- 13 different detector triggers used in main DAQ
- Data rate at time order PC: 20-200 MB/s
- Event rate: 200 - 10000 Events/s depending on deadtime domain
- Uses TRLOII / nurdlib / ucesb



MBS @ R3B / LAND (oct 2014)

- Readout of 15 different detector types spread across 3 experimental sites
- 12 VME crates + 9 PCs, 6 event builder PCs, 10 Gbit Timeorder PC
- 5 deadtime domains, 2 trigger bus chains, 1 trimi link master
- Combined serial timestamp distribution and White rabbit timing
- 13 different detector triggers used in main DAQ
- Data rate at time order PC: 20-200 MB/s
- Event rate: 200 - 10000 Events/s depending on deadtime domain
- Uses TRLOII / nurdlib / ucesb



Reasonable extensions to MBS

Today

- Introduction – What is a DAQ and what is MBS
- SBS – A simple MBS
  - hardware (RIO, TRIVA, VULOM, TRIXOR, ...)
  - m_read_meb (f_user.c)
- MBS – Multiple crates
- Use cases:
  - MBS at Duke University for Gamma^3
  - MBS at the R^{3}B setup at GSI
- TRLOII – A flexible trigger logic
- nurdlib – The nustar readout library
- ucesb – Unpack and check every single bit (the sorting code)
- Outlook

Warning!

You are now leaving the realm of
MBS-supported software!
Use at your own risk!

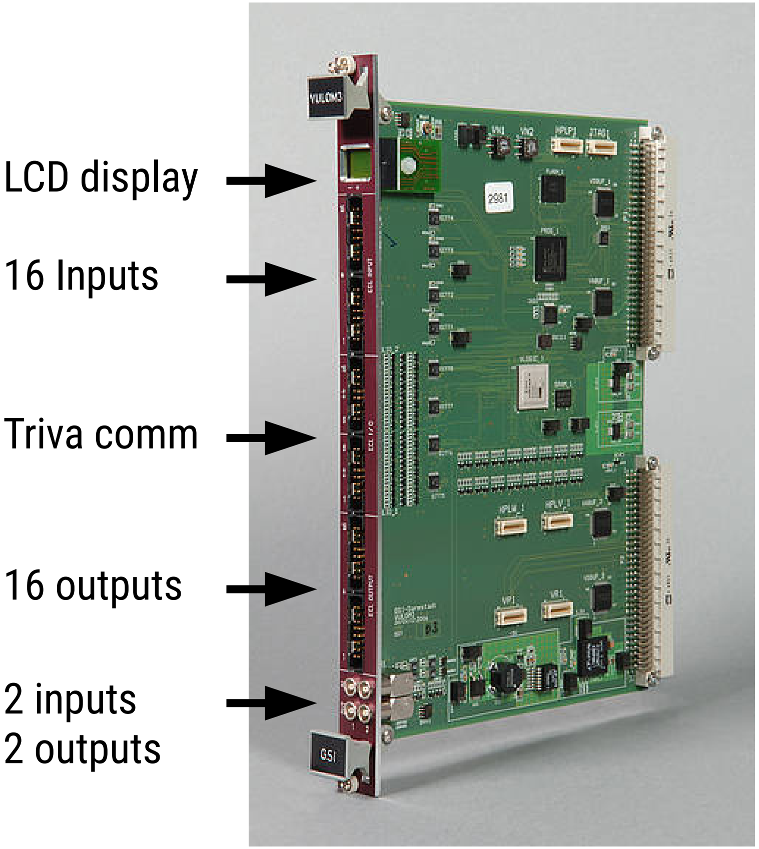
Warning!

You are now leaving the realm of MBS-supported software!
Use at your own risk!

This does not mean you won‘t find any help, just don‘t rely on it at 3 am in the morning.

VULOM4 – User Logic module

- Standard firmware:
  - 13 trigger inputs
  - Deadtime locked trigger outputs
  - Delay (input-output): ~30 ns
  - Jitter: ~2.5 ns



LCD display

16 Inputs

Triva comm

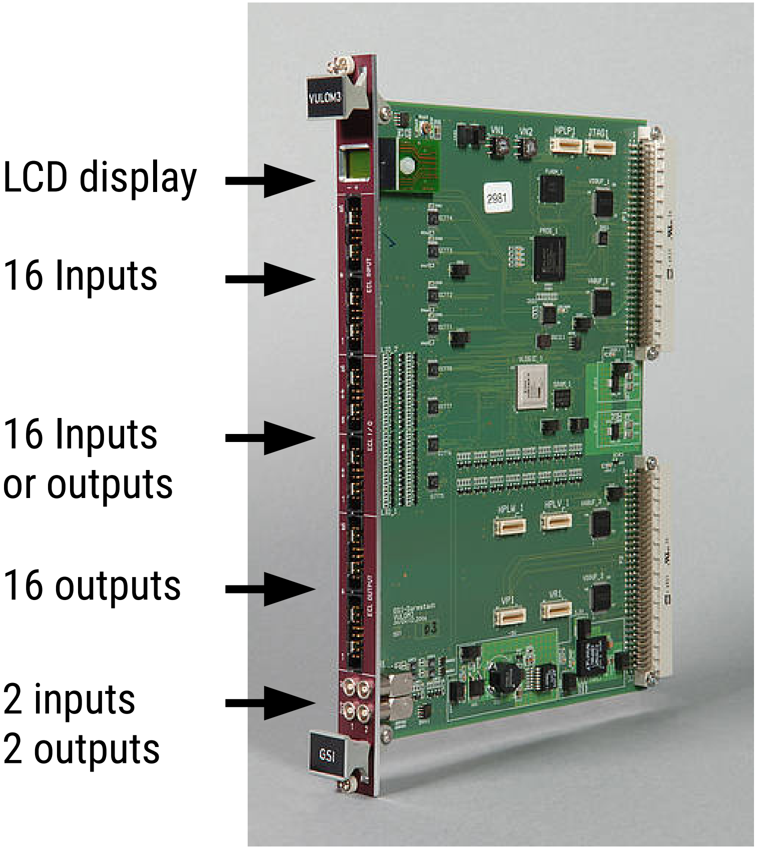
16 outputs

2 inputs
2 outputs

GSI Helmholtzzentrum für Schwerionenforschung GmbH

VULOM4 – With TRLOII

- Trigger Logic 2 firmware:
  - 16 trigger inputs with variable delay and stretcher
  - Trigger matrix for coincidences
  - Deadtime locked master trigger
  - Trigger reduction (downscaler)
  - Scalers everywhere
  - Multi-event trigger buffer



LCD display

16 Inputs

16 Inputs
or outputs

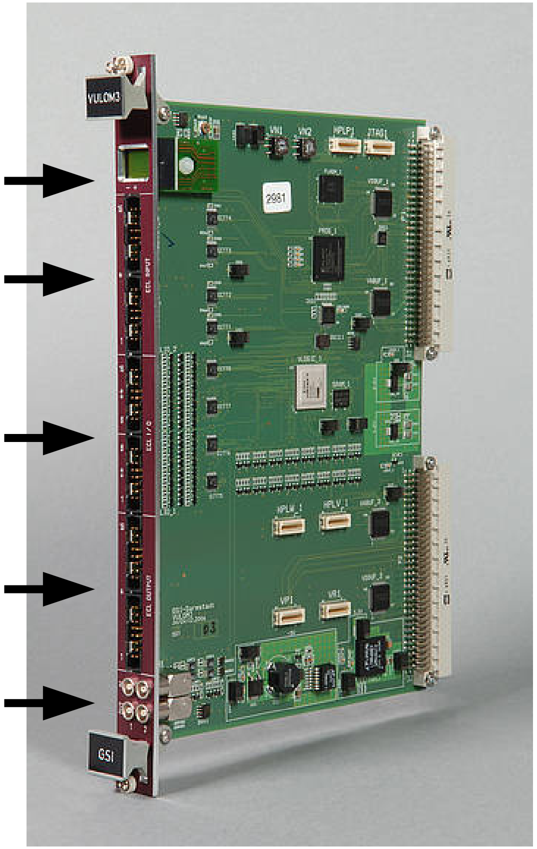
16 outputs

2 inputs
2 outputs

http://fy.chalmers.se/~f96hajo/trloii/

VULOM4 – With TRLOII

- Trigger Logic 2 firmware:
  - Generic pulsers
  - Generic logic matrix unit
  - Gate and delay generators
  - Edge to gate converters
  - Fan-In (OR) function
  - Generic coincidence units



LCD display

16 Inputs

16 Inputs
or outputs

16 outputs

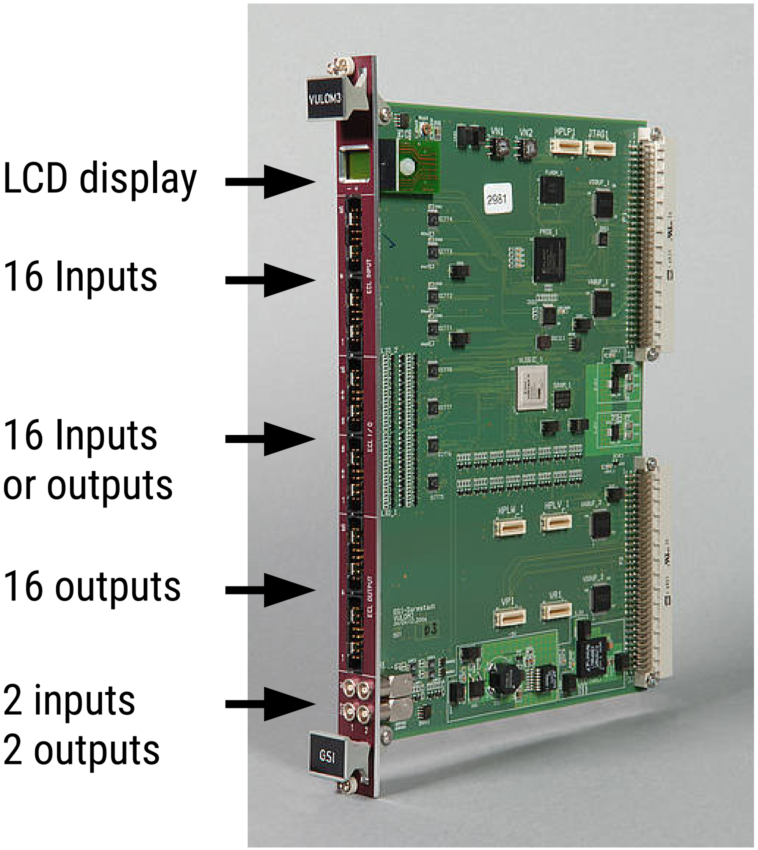
2 inputs
2 outputs

http://fy.chalmers.se/~f96hajo/trloii/

VULOM4 – With TRLOII

- Trigger Logic 2 firmware:
  - Additional scalers
  - Timer latches
  - Self-triggering soft scope (for input time alignment)
  - Front-panel LEDs and LCD

Capable of replacing a crate full of NIM delays, LMUs, trigger boxes, pulsers, scalers and FIFOs.



LCD display

16 Inputs

16 Inputs
or outputs

16 outputs

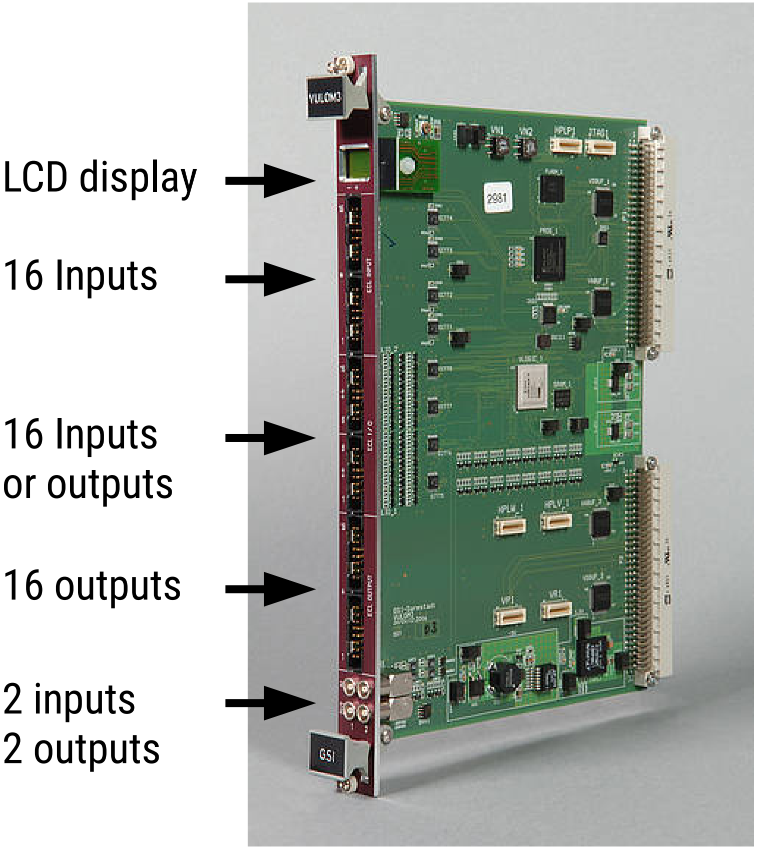
2 inputs
2 outputs

http://fy.chalmers.se/~f96hajo/trloii/

VULOM4 – With TRLOII

- Trigger Logic 2 firmware:
  - Serial timestamp input/output (ratatime) with 10 ns resolution
  - TRIVA7 mimic (TRIMI)
  - TRIMI link to act as triggerbus replacement (ratatrig)

Even replaces the TRIVA7 module and additional Timestamp modules.



LCD display

16 Inputs

16 Inputs or outputs

16 outputs

2 inputs
2 outputs

http://fy.chalmers.se/~f96hajo/trloii/

VULOM4 – With TRLOII

- TRLOII is very complex with 500 setup registers and 200 multiplexable signals
- `trloctrl` program:
  - Can control and monitor a VULOM4 with installed TRLOII firmware
  - Configuration of TRLOII via setup files (`vulom.trlo`)
- Try:
  - `trloctrl --addr=2 --print-config`
  - `trloctrl --addr=2 --mux-src-scalers`

http://fy.chalmers.se/~f96hajo/trloii/

Nustar Readout Library - nurdlib

http://webdocs.gsi.de/~land/nurdlib/

Nustar Readout Library - nurdlib

- The missing piece in MBS: **readout code** -> Nurdlib fills the gap!
- Main Features:
  - Text-based configuration of crate layout and readout modules
  - Sane default configurations included
  - Independent of platform or DAQ environment
  - Online data integrity checking
  - Multi-event support
  - Single cycle and block transfer (DMA) modes where supported
  - Strict ansi C compliance and harsh GCC flags

http://webdocs.gsi.de/~land/nurdlib/

Nustar Readout Library - nurdlib

- Supported hardware:
  - CAEN V775/785/792/830/895/965/1190/1290
  - Mesytec MADC32/MTDC32/MQDC32
  - GSI SAM4&5/TACQUILA/VULOM/VETAR/VFTX2/VUPROM
  - Struck SIS3316
- ~700 lines of code per module

http://webdocs.gsi.de/~land/nurdlib/

Nustar Readout Library - nurdlib

- Example config file:

```
CRATE("XBL") {
    acvt = true
    GSI_VULOM(0x02000000) {
        trlo2_master = true
        trlo2_timerlatcher = true
        trlo2_timestamper = true
    }
    MESYTEC_MADC32(0x00700000) {}
    MESYTEC_MADC32(0x00710000) {}
    MESYTEC_MADC32(0x00720000) {}
    MESYTEC_MADC32(0x00730000) {}
    GSI_VUPROM(0x05000000) {}
}
```

```
CRATE("TOF") {
    GSI_VULOM(0x02000000) {}
    BARRIER
    GSI_VFTX2(32, 0x09000000) {
            channel_invert = 0xaaaa
    }
    GSI_VFTX2(32, 0x0a000000) {
            channel_invert = 0xaaaa
    }
    GSI_VFTX2(32, 0x0b000000) {
            channel_invert = 0xaaaa
    }
}
```

http://webdocs.gsi.de/~land/nurdlib/

Nustar Readout Library - nurdlib

- Nurdlib and MBS -> `r3bfuser`
  - Needs ‚glue code‘ to attach nurdlib to the MBS functions in the `f_user.c` file
  - `r3bfuser` aims to be generic glue code for MBS and nurdlib
  - Simplified:
    - `f_user_get_virt_ptr()` does nothing
    - `f_user_init()` calls `nurdlib_setup(„main.cfg“)`
    - `f_user_readout()` calls `crate_readout()`

http://webdocs.gsi.de/~land/nurdlib/

Nustar Readout Library - nurdlib

- Directory structure:
  - `rio4-1:`
    - `setup.usf` – User setup file
    - `start.scom` – Startup script
    - `stop.scom` – Shutdown script
    - `vulom.trlo` – TRLOII setup file
    - `main.cfg` – nurdlib setup file
  - `nurdlib`
  - `trloii`
  - `r3bfuser`

Unpack and check every single bit - ucesb

http://fy.chalmers.se/~f96hajo/ucesb/

Unpack and check every single bit - ucesb

- ucesb is a generic unpacker generator
  - Based on a specification file an experiment specific data unpacker is generated
  - Transforms LMD (and other) event-wise packed data into ROOT files (or PAW ntuples)
  - Physical (hardware) channels are mapped to logical (detector) channels, support for multi-hit and multi-event data
  - Calibration can be applied in the same process

```
# Read stream output from rio4-1 and write to test.root file
# ROOT file contains a tree ‚h101‘ with mapped detector
# branches
> ./ucesb stream://rio4-1 --ntuple=RAW,test.root
```

http://fy.chalmers.se/~f96hajo/ucesb/

Unpack and check every single bit - ucesb

- ucesb is a data stream multiplexer
  - Reads from MBS stream or transport or event server output, from an LMD file, from the output of another ucesb instance
  - Filters based on event and subevent type
  - Writes MBS-like stream output, writes to file or sends data in a fixed structure over network

```
# Read stream output from rio4-1 and serve only events with
# type 88 it on the network on port 8000
> ./ucesb stream://rio4-1 --server=stream:8000,incl=type=88
```

http://fy.chalmers.se/~f96hajo/ucesb/

Unpack and check every single bit - ucesb

- ucesb is a time sorter and event stitcher
  - Sorts events from several input streams into a single output stream based on a timestamp (white rabbit or titris style)
  - Stitches events from different subsystems together with matching timestamps (closer than N timestamp units)

```
# Read two streams and combine, then do time-stitching
> ucesb --stream=rio4-1 --stream=rio4-2 --merge=wr,2 \
  | ucesb --file=- --time-stitch=40 --ntuple=RAW,test.root
```

http://fy.chalmers.se/~f96hajo/ucesb/

Unpack and check every single bit - ucesb

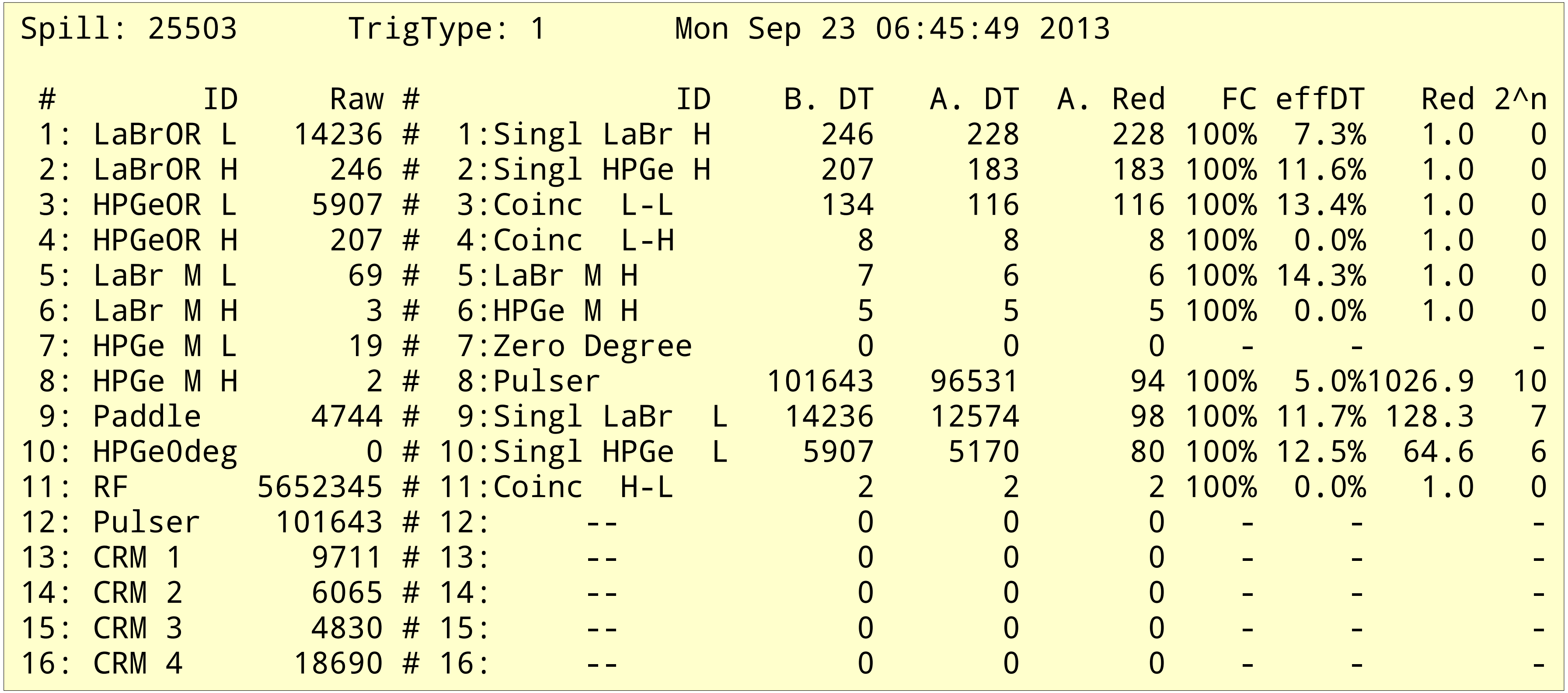
- ucesb is a DAQ debugging tool
  - Gives instant access to LMD event and subevent data structure
  - Shows where in the data stream the unpacking failed
  - Shows ascii histograms of detector channels
- ucesb can be extended by user functions

ucesb is your swiss army knife for event sorting and data handling

http://fy.chalmers.se/~f96hajo/ucesb/

ucesb and TRLOII

- TRLOII experiment specific scaler display (via ucesb)



```
Spill: 25503      TrigType: 1       Mon Sep 23 06:45:49 2013

 #        ID     Raw #              ID    B. DT   A. DT  A. Red   FC effDT   Red 2^n
 1: LaBrOR L   14236 #  1:Singl LaBr H      246     228     228 100%  7.3%   1.0   0
 2: LaBrOR H     246 #  2:Singl HPGe H      207     183     183 100% 11.6%   1.0   0
 3: HPGeOR L    5907 #  3:Coinc  L-L        134     116     116 100% 13.4%   1.0   0
 4: HPGeOR H     207 #  4:Coinc  L-H          8       8       8 100%  0.0%   1.0   0
 5: LaBr M L      69 #  5:LaBr M H            7       6       6 100% 14.3%   1.0   0
 6: LaBr M H       3 #  6:HPGe M H            5       5       5 100%  0.0%   1.0   0
 7: HPGe M L      19 #  7:Zero Degree         0       0       0    -     -         -
 8: HPGe M H       2 #  8:Pulser         101643   96531      94 100%  5.0%1026.9  10
 9: Paddle      4744 #  9:Singl LaBr  L   14236   12574      98 100% 11.7% 128.3   7
10: HPGe0deg       0 # 10:Singl HPGe  L    5907    5170      80 100% 12.5%  64.6   6
11: RF       5652345 # 11:Coinc  H-L          2       2       2 100%  0.0%   1.0   0
12: Pulser    101643 # 12:     --             0       0       0    -     -         -
13: CRM 1       9711 # 13:     --             0       0       0    -     -         -
14: CRM 2       6065 # 14:     --             0       0       0    -     -         -
15: CRM 3       4830 # 15:     --             0       0       0    -     -         -
16: CRM 4      18690 # 16:     --             0       0       0    -     -         -
```

http://fy.chalmers.se/~f96hajo/trloii/

Today

- Introduction – What is a DAQ and what is MBS
- SBS – A simple MBS
  - hardware (RIO, TRIVA, VULOM, TRIXOR, ...)
  - m_read_meb (f_user.c)
- MBS – Multiple crates
- Use cases:
  - MBS at Duke University for Gamma^3
  - MBS at the R^{3}B setup at GSI
- TRLOII – A flexible trigger logic
- nurdlib – The nustar readout library
- ucesb – Unpack and check every single bit (the sorting code)
- Outlook ⇦

Outlook - What‘s cooking...

- Currently gearing up for 2016 beam time at GSI
- Investigating a successor of MBS with
  - same f_user.c interface, same MBS data format
  - faster startup time
  - higher flexibility and better fault handling in multi-crate setups
  - auto-connect of temporarily missing or new crates
  - tightly coupled to ucesb for data transport
- Triggerbus handling from TRIMI, to fully replace TRIVA7 in all instances
- Improved version of VULOM for Nustar signal exchange points. To act as generic signal relay station