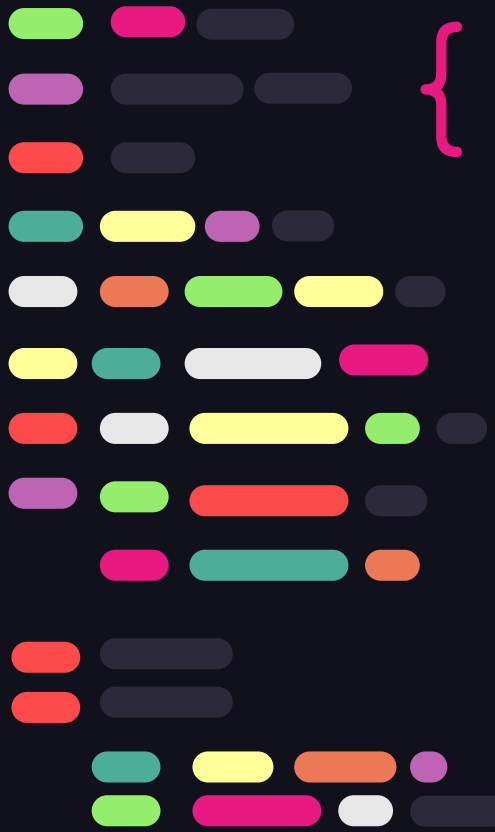
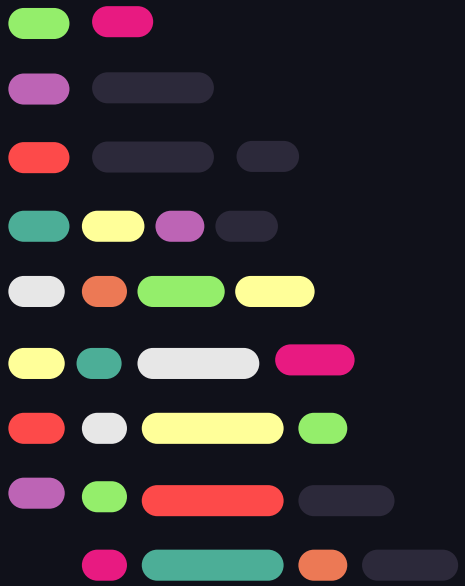


Introduction to LMD File format

Presenter: 204d617274696e2042616a7a656b
Date: 000001ce 03e1ac08 04e13d2f 05e16672 06e1179c

Table of contents



What does this do with physics?

Physics result is the final product of a long process which springs from detector response to charged particle hits. Then signal digitalisation, encoding and readout. Data in this format is then unpacked and processed and yields physics.

After readout, data can already be stored. It's still entangled but can be 'decrypted' at any point.

LMD, amongst other things, is how the rawest physics data, in bits and bytes, can be assessed by us physicists.

Events and subevents

Event

Building blocks of our binary data. Comprised of event header and zero or more subevents

Subevent

Blocks of data from one DAQ node (usually). Comprised of subevent header and subevent data

01 LMD structures (I)

LMD (List Mode Data) format is used both for data transport and storage, based on notion of continuous data stream. Each singular item is called an event

LMD File	Control data structure: sLmdcontrol (fLmd.h)
↓	
LMD File Header	s_filhe.h
LMD Buffer Header	s_bufhe.h
LMD Event Header	s_ve10_1.h
LMD Subevent Header	s_ves10_1.h
((s_tpccomm))	TBD

Info:
http://web-docs.gsi.de/~bloeher/eventapi/html/eventapi.html

01 LMD structures (II)

For majority of users, we mainly deal with LMD Events and Subevents

GSI VME Event Header	s_ve10_1.h	
l_dlen	Data length in 16-bit words + 4 bytes	(32 bits)
i_type	Event type	(16 bits)
i_subtype	Event subtype	(16 bits)
i_dummy	Used for padding	(16 bits)
i_trigger	Trigger number	(16 bits)
l_count	Current event number	(32 bits)

Info:
http://web-docs.gsi.de/~bloeher/eventapi/html/eventapi.html

01 LMD structures (III)

For majority of users, we mainly deal with LMD Events and Subevents

GSI VME Subevent Header	s_ves10_1.h	
l_dlen	Data length in 16-bit words + 4 bytes	(32 bits)
i_type	Event type	(16 bits)
i_subtype	Event subtype	(16 bits)
h_procid	Processor ID	(16 bits)
h_subcrate	Subcrate number	(8 bits)
h_control	Processor type code	(8 bits)

Info:
http://web-docs.gsi.de/~bloeher/eventapi/html/eventapi.html

Short reminder on binary and hex concepts

Binary

Each digit is 0 or a 1. This is where digital machines live. Each

Hexadecimal

Digits range from 0-9,a,b,c,d,e,f
Each hex digit is a unique combination of 4 binary digits.

- 2 hex digits comprise a byte (8 bit)
- 8 hex digits comprise a word (32 bit)
- Word ⇔ int type, in most languages

Decimal

World we live in

01 Glossary

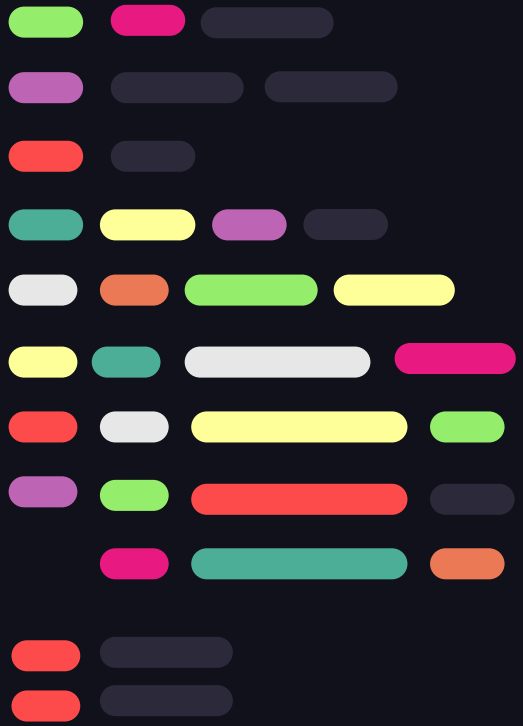
1) DAQ Data acquisition.
2) LMD List Mode Data.
3) Header describes attributes and organisation of the data.
4) Event is ideally a snapshot of one ‘physics event’ bundled together in the data. Made of header + subevents.
5) Subevent is a snapshot of a DAQ node. Usually encapsulates data from one detector system. Comprised of header + underlying data.

02 { ..

Practical examples

} ..

Instructions



Clone the following repository anywhere:

```
$> git clone https://github.com/kLayz3/lmd-training
```

Enter the directory

```
$> cd lmd-training
```

Execute the run.bash

```
$> ./run.bash
```

Complete the initial task!

02 Word slicing

A word (32-bit block) will always have data `hidden` in certain bits, while the other bits are used for tagging and indicators. To extract the slice [hi:lo] starting from low-bit lo, ending with high bit hi (where 0 ≤ lo≤hi < 31), we can do the following trick:

We are little endian

- The least significant bit (bit 0) is on the right, while the most significant bit is on the left (bit 31)

Algorithm on the next slide

02 Word slicing

A word (32-bit block) will always have data `hidden` in certain bits, while the other bits are used for tagging and indicators. To extract the slice [hi:lo] starting from low-bit lo, ending with high bit hi (where 0 ≤ lo≤hi < 31), we can do the following trick:

We are little endian

- The least significant bit (bit 0) is on the right, while the most significant bit is on the left (bit 31)

```
# Algorithm (in pseudocode)
# 1. Create a word ‘mask’ with which we want to
# capture only the important bits

Mask = (1 << (hi-lo+1)) - 1

# 2. Shift the word to the right, so that ‘lo’ becomes
# the first bit (bit 0)

Word = (Word >> lo)

# Bitwise AND the `Mask` and the shifted `Word`

Value = Word & Mask
```

02 Tools of the trade (I)

Ucesb is a useful tool primarily used to unpack event-wise data to ROOT structures.

However, we can use its empty unpacker to print raw events and look into the LMD

Credits:
Håkan T. Johansson, Chalmers SE

Clone the following repository in the lmd-training dir

```
$> git clone https://git.chalmers.se/expsubphys/ucesb.git
```

Enter the directory

```
$> cd ucesb
```

Make the empty target

```
$> make empty -j4
```

Add it to path/bashrc

```
$> export PATH=$PATH:$PWD/ucesb/empty/
```

02 Tools of the trade (II)

Ucesb is a useful tool primarily used to unpack event-wise data to ROOT structures.

However, we can use its empty unpacker to print raw events and look into the LMD

Credits:
Håkan T. Johansson, Chalmers SE

In moments of doubt, can always pass a --help flag

```
$> empty --help
```

Try and execute

```
$> empty example.lmd --data --print  --max-events=1
```

Can you identify different events & subevents?

In blue should be event header data
In purple should be subevent header data

Below that lies the hexadecimal subevent data

02 GSI LMD module data

Continue with the run.bash exercises and help Alice follow the White Rabbit!

“Oh dear, oh dear, I shall be too late”
– Someone

Current WR status in Nov 2023:

```
00000100 03e1ac08 04e13d2f 05e16672 06e1179c
   ↑        ↑        ↑        ↑        ↑
 WR ID    LoLo     LoHi     HiLo     HiHi
```

White Rabbit is a ethernet-based timing distribution network for data transfer and sub-nanosecond timing accuracy

Once decoded the white rabbit value is UNIX time elapsed since Jan 1. 1970 00:00:00 UTC measured in nanoseconds

At GSI (and other facilities, like CERN) it’s used to synchronise different data acquisition systems

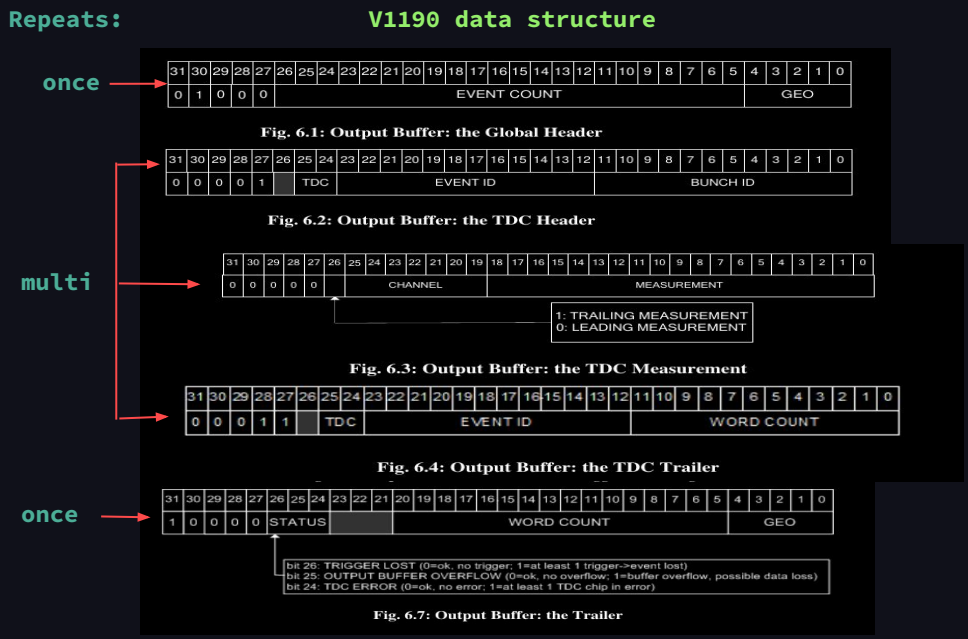
In WR block, we tag the WR ID (in orange), and in the Low 16 bits of next 4 words we encode each of the Consecutive four 16-bit slices of a full 64-bit WR With LoLo being least significant, HiHi most significant

02 GSI LMD module data

Continue with the run.bash exercises and help Alice beat the big bad boss V1190!

“Curiouser and curiouser!”
– Someone

Tip: usually at GSI, we more clearly separate data from different modules in same subevent by inserting ‘barrier’ words. These are usually bababababa or (new) f520****



Congratulations

You have officially advanced to the intermediate level of LMD hacker

1) Next step: advanced features of `empty`
2) MBS examples

03 Advanced (I)

empty can also `peek` at the running data acquisition systems and fetch fresh data

This data can be then printed or recorded as LMD, or piped further

Credits:
Håkan T. Johansson, Chalmers SE

```
In moments of doubt, can always pass a --help flag

$> empty --help


Examples for future (won't work in this directory)
Event server is usually on port 6003, we can safely listen there
$> empty event://localhost:6003 --data --print --max-events=1

Stream server is usually on port 6001, not advised to tap
Usually this port is reserved for a Remote Event Server (REV).
So better use it spawn a REV server

Need to record a quick LMD file of a couple dozen events?
$> empty event://localhost:6003 --output=test.lmd --max-events=100
```

03 Advanced (II)

Output can be piped to shell commands for quick and dirty inspection

We could try and fetch just a single subevent from our file/DAQ. Grep will print just the line with the match. However adding -A10 will print 10 lines after the match, -B1 can print one before the match

```
$> empty example.lmd --data --print | grep -P 'ProcID\s+20' -B1 -A10
```

Want to somehow save the output of the empty hexdump into a dumped_lmd.txt text file?

```
$> empty example.lmd --data --print >dumped_lmd.txt
```

Make life easier with get_bits.bash

Always remember: PHYSICS IS FUN

Resources

Further reading:

- Bastii's web doc: https://web-docs.gsi.de/~bloeher/
- GSI MBS: https://www.gsi.de/en/work/research/experiment_electronics/data_processing/data_acquisition/mbs/documents
- Go4 analysis framework: https://github.com/gsi-ee/go4
- A lot of coffee looking at source codes

The real LMD experts

Dr. Nikolaus Kurz

Senior researcher at GSI. Data acquisition, data analysis, system management real-time OS (Linux, LynxOS), digital electronics design

Dr. Bastian Löher

Former senior researcher at GSI. Responsible for DAQ and experiment control for the R3B experiment

… and many others from different groups !

Cheers!

<> Do you have any questions? <>

M.Bajzek@gsi.de


slidesgo