

L1EmtOnline and L1EmtOdf

EMT Software Review

Paul Dauncey
Imperial College

L1EmtOnline

- Dual-use package
- Package coordinator: Phil Clark
- Latest tag V00-03-06 (yesterday!)

Contains the classes which contain and manipulate the configuration and event data

- **Not** the classes to move them around

Consider configuration and event data separately

L1EmtOnline: but first...

Two classes span both configuration and event data

- **L1EmtTypeIds** contains unique number for every EMT TC stored in an enum in the class
 - Required to identify TC objects
 - Stores both configuration and event data TC identifiers
 - Actual association done using macros in **L1EmtOnlineTypeInit.cc**
- **L1EmtDlinkHeader** is class to handle 16-bit header which is sent before any data on the D-link
 - Similar for configuration and event data
 - Copied from **L1EmtOdf/L1EmtHeader** as needed offline
 - **L1EmtOdf/L1EmtHeader** now redundant; should be removed

L1EmtOnline: configuration data

Four types of configuration data are required for EMT

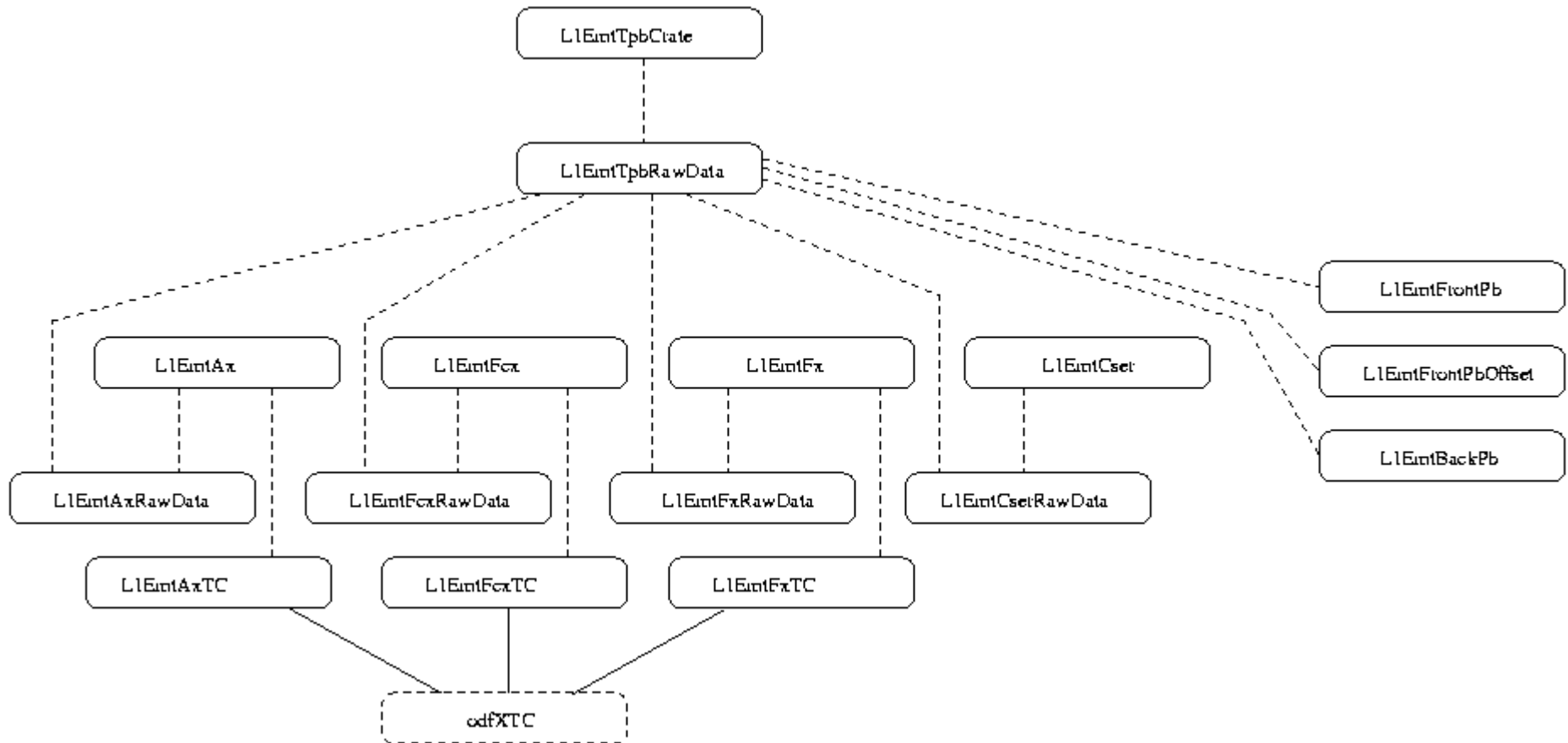
- Control, status and error register (Cser) data
- Algorithm Xilinx (Ax) data
- Fast Control Xilinx (Fcx) data
- Formatter Xilinx (Fx) data

Two types are optional and used in tests

- Front-end playback (Fepb) data (and offsets)
- Back-end playback (Bepb) data

Write these all generically here as “Xxx”

L1EmtOnline: configuration data classes



L1EmtOnline: configuration data (cont)

Take Ax data as typical of the required data

- **L1EmtAxRawData** is basis; this is an exact bit-level copy of the data which get sent on the C and D-links
- **L1EmtAx** is a class to manipulate these data; it is a friend of **L1EmtAxRawData**
- **L1EmtAxTC** is the tagged container class for moving the Ax data to and from the ROM; it inherits from **odfXTC**. (N.B. there is no TC for Cser)

(More on the front and back-end playback in **L1EmtOdf**)

All configuration data for a TPB stored in **L1EmtTpbRawData**

All configuration data for system stored in **L1EmtTpbCrate**

L1EmtOnline: event data

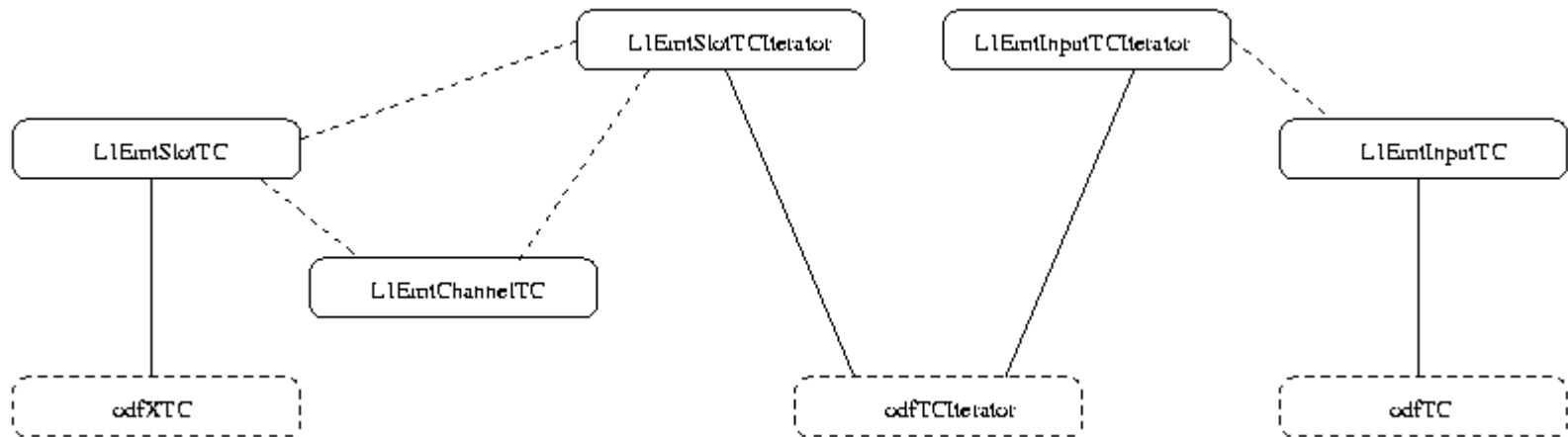
The event data have two separate structures

- Input TC's which describe the raw data format coming up the D-link on a ReadEvent; it is the input to the feature extraction (FEX)
- Slot and Channel TC's which describe the data after formatting and zero-suppression; it is the output of the FEX

The **L1EmtInputTC** class does not appear to actually be used by the FEX; it is (apparently) redundant

The **L1EmtSlotTC** and **L1EmtChannelTC** classes are the ones used to analyse data in XTC files; there is also an **L1EmtSlotIterator** to help find the **L1EmtChannelTC** objects inside the **L1EmtSlotTC**

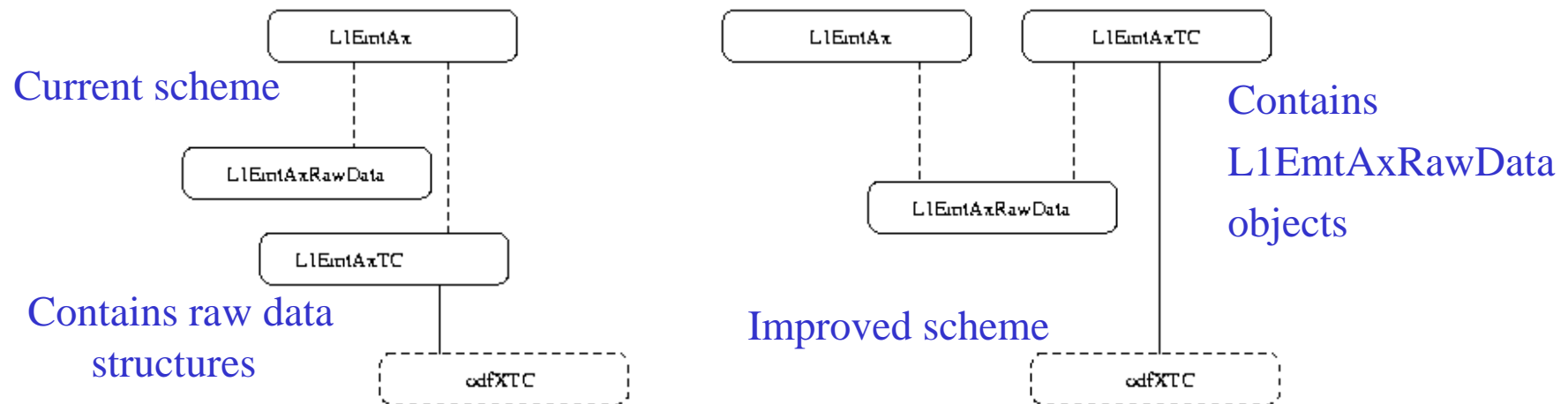
L1EmtOnline: event data (cont)



- L1EmtSlotTC inherits from odfXTC; i.e. it is a genuine tagged container
- L1EmtChannelTC does not; i.e. it is not a real tagged container but is called TC to indicate it plays the same role as one
- Note old versions, **L1EmtSlotTCV1**, etc., exist: not used since early cosmic runs in 1999, before physics

L1EmtOnline: comments

L1EmtXxxRawData, L1EmtXxx and L1EmtXxxTC structure not optimal. Take Ax as typical:



L1EmtAxTC should **not** know about internals of L1EmtAxRawData

- Should simply carry around a block of data
- Same holds for L1EmtAxRecord and L1EmtAxRawDataP
- Should L1EmtXxxRawData and L1EmtXxx merge (or inherit)?

L1EmtOnline: comments (cont)

Most of the configuration (and event) data come in two-byte words, with a few exceptions

- char variables and bit structures

Writes and reads assume two-byte structure; all data are two-byte swapped in

- L1EmtWrite:: setData()
- L1EmtAbsRead constructor

Exceptions are done by hand in raw data classes

- char and bit variables are ordered the “wrong way round”

Endian issues much easier if **everything** is two-byte

- Must clean up the exceptions using explicit bit masks and shifts in all cases

L1EmtOdf

- Online-only package
- Package coordinator: Phil Clark
- Latest tag V00-01-18

Contains the classes which operate the C and D-links

- Send commands to the EMT
- Writes and reads of configuration and event data

Also classes which do VME access

- Read only; no chance of overwriting configuration
- Intended for use for spy monitoring

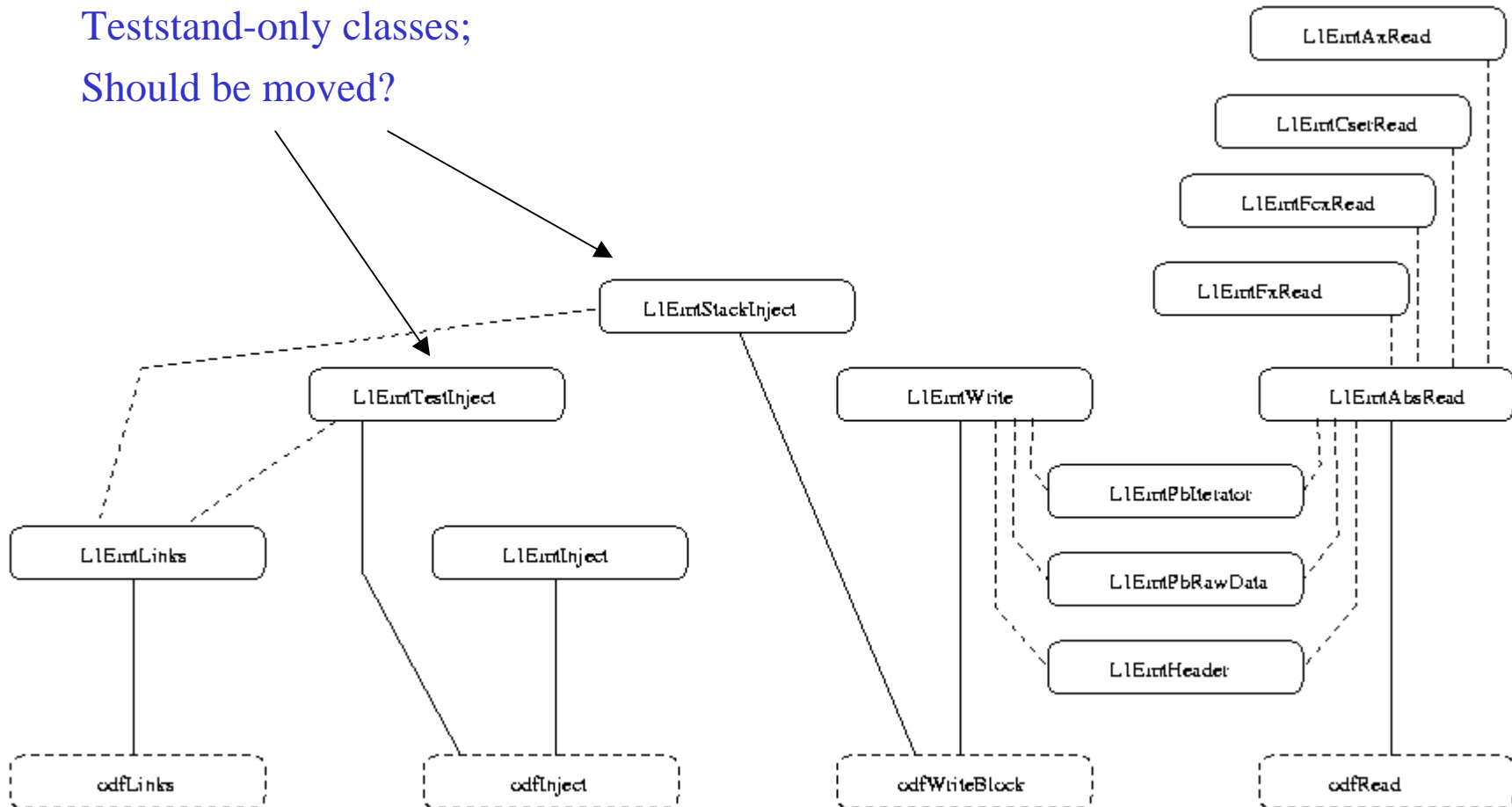
L1EmtOdf: C and D-link commands

Four functions

- Links; tells central ODF which TPB's are installed. The `L1EmtLinks` class hardwires this.
- Injects; commands with no data and no reply. All EMT-specific injects done in `L1EmtInject`.
- Writes; sends configuration data, but no reply. All EMT configuration data written using `L1EmtWrite`.
- Reads; sends command and waits for reply of data back. Event data handled by ODF, configuration data read using `L1EmtXxxRead`, which inherit from `L1EmtAbsRead<L1EmtXxxRawData>`

L1EmtOdf: C and D-link (cont)

Teststand-only classes;
Should be moved?



L1EmtOdf: playback

Front and back-end playback are more complicated

- Data size too big to write or read in one command
- Need to chop into packets and send one at a time
- **L1EmtPbRawData** is bit-level copy of one packet
 - Used for both front and back-end, write and read
- Size of **L1EmtPbRawData** fixed by enum
 - **L1EmtFcxRawData::nPbRawDataWords**
 - Written down to TPB as part of Fcx configuration
 - Don't change this value unless you **really** know what you are doing!
- Packet handling done using **L1EmtFrontPbIterator** (messy!) and **L1EmtBackPbIterator**
 - Also need offsets for LSB of front-end; **L1EmtFrontPbOffset**

L1EmtOdf: spy and VME

Read-only VME access to TPB's

- “Unofficial”, i.e. not part of ODF; used for monitoring
- Must never effect running, hence read-only
- Access to Cser and spy memory data

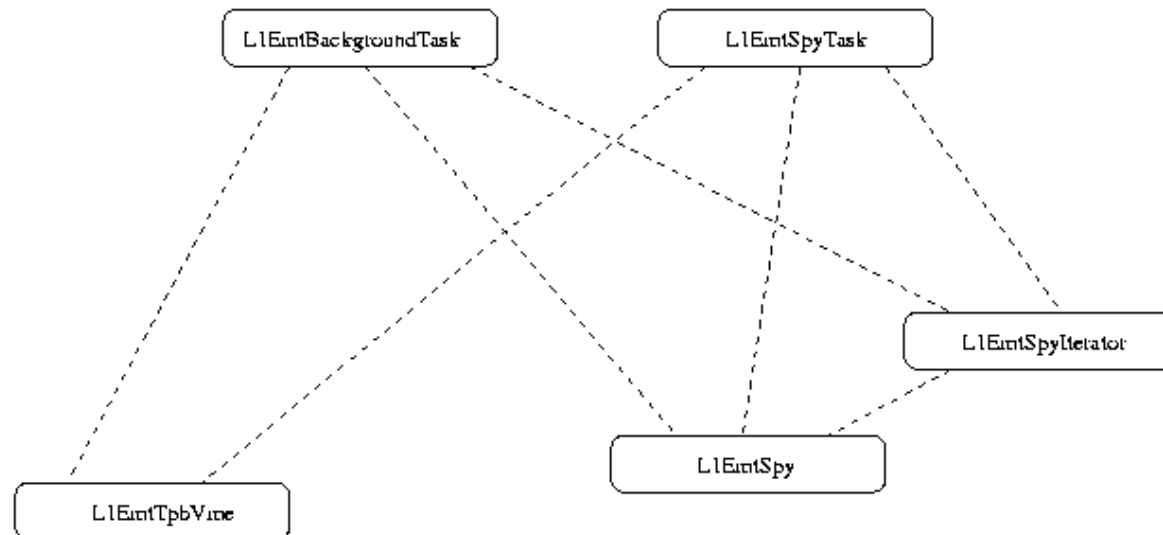
TPB VME memory “footprint” described by `L1EmtTpbVme`

- Never instantiate a `L1EmtTpbVme` object
- Cast a const pointer from the correct memory location, given by `L1EmtTpbVme::base()`
- All class data and methods are const, reflecting read-only access

L1EmtOdf: background task

Background task is supposed to periodically read spy

- Data read four bytes at a time, stored in L1EmtSpy
- Use L1EmtSpyIterator to handle the bit manipulations (also messy!)
- Compare data with calculation, L1EmtSpy/L1EmtSpyModel
- Report any errors to FEX, send out in L1EmtSlotTC



L1EmtOdf: comments

- There is a background spy task in both `L1EmtSpyTask` and `L1EmtBackgroundTask`; these should be merged
- `L1EmtTrgConfig` seems to be in the wrong package; move to `L1EmtCalOdf`?
- `L1EmtSoakTestAction` seems to be for stand-alone teststand; should be moved (to a new package?)
- Remove all use of `L1EmtHeader` and replace with `L1EmtOnline/L1EmtDlinkHeader`

General comments

Comments applicable to both packages

- Software rot occurring; original class structure being eroded in places
- Many uses of assert, cout and cerr; what is best way to handle error reporting, particularly in dual-use packages?
- Lots of comments saying code is temporary (from years ago), needs improving or is obsolete. These should be fixed or the comments removed.
- No consistent use of defined types d_UShort, etc. and most classes do not include BaBar.hh

Conclusions

Both L1EmtOnline and L1EmtOdf in reasonable shape.

Work needed, in order of priority:

1. Need to break dependency of L1EmtXxxTC, L1EmtXxxRecord and L1EmtXxxRawDataP classes on L1EmtXxxRawData structure by using L1EmtXxx methods
2. Need to make all TC classes two-byte structures throughout
3. General clean-up of messy code

Nothing critical, but some care needed to get it right