An UML description of the ATLAS ROB viewed from the Level-2 Trigger.

Authors : M Huet
Keywords : ROB, ROBin, ROB Complex, UML.

Abstract

This document presents a description of the ATLAS Read Out Buffer(ROB). It is focused on the interaction with the Level-2 Trigger. However it tackles interactions with some other components: detectors, Trigger/DAQ run control and monitoring. This document includes the analysis of requirements and a high level design.

Table of Contents

1 Introduction

This document presents the results from analysis of the ATLAS Read Out Buffer (ROB) done in the context of the Level-2 Trigger activity. Starting from the interactions with the actors of the Atlas Trigger/DAQ, the main components of the ROB are described. This analysis is focused on the interaction with the Level-2 Trigger. The connection with the electronics of detectors is shown because it brings severe constraints on the design of the ROB. We also tackle interaction with the run control of the DAQ/Trigger for ROB configuration and monitoring for the statistics management.

2 Organisation and Definitions

2.1 Methodology

We have followed the UML formalism[1] and corresponding methodology.

First, the interactions with external actors, mainly based on the ROB URD[2], is presented in “use cases”. Corresponding URs are introduced to the convenient place. Then, “sequence and/or collaboration diagrams” introduce the main objects to satisfy the corresponding scenarios.

It must be mentioned that between the creation of the ROB URD and now, the L3 has been changed into Event Filter (EF). The Event Builder (EB) collects data for the EF.

2.2 ROB and ROB Complex definitions

The ROB is an entity which stores temporarily a fragment of each event. The event fragment is provided by the front-end electronics of detectors. Part of or full event fragment must be made accessible to the Level-2 Trigger and to the Event Filter. When data is no more useful it is deleted to free place for further events.

At that level, it seems not to be an implementation issue to speak of the access to the ROB data. The foreseen ROB number is about 1500 and the number of processors which have to access to Event data in ROBs is probably of the order of several hundreds. The direct access from processors to ROBs is not possible therefore the connection between ROBs and processors is foreseen through switching networks.

The ROB functionality and the connections to the external actors can be ensured by several diverse entities. The whole set of components is called the **ROB Complex.**

During the rest of this document, the ROB designation will apply to the ROB Complex.

2.3 Components of the Level-2 Trigger

The Level-2 Trigger system is built with four main components:

- A RoI builder [3] that receives the RoI pattern of the Level-1 Trigger and drives it to several supervisors of the Level-2 Trigger system.
- Supervisors [3] dispatch the events to different processing units. Due to the Level-1 Trigger rate, several supervisors are necessary.

- Processing Units to process the algorithms of the Level-2 Trigger.
- A switching network to interconnect supervisors and processing units. ROBs are connected to the processing units through a common network.

From here, the “Level-2” designation will be abbreviated to “LVL2” in component names and to “L2” in the signal names and URs.

3 Interaction of the ROB with external Actors.

In this chapter we describe how external actors interact with the ROB. Each kind of interaction is presented in a “Use Case” diagram. An additional sequence diagram shows the type of messages that are exchanged and their sequence.

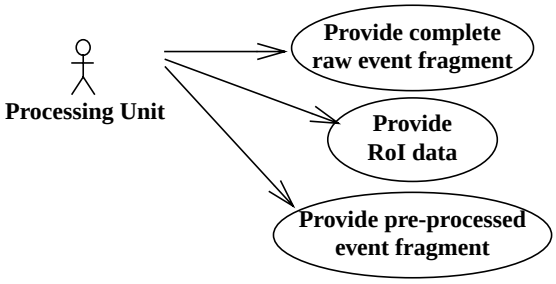
We start with the interactions specific to LVL2 actors, the processing unit and the supervisor. In a second step we present interaction with the front-end. It is not specific to the LVL2 activity but it can affect the ROB performance because the event buffer is shared between the storage of incoming data and the furniture of data to downstream components. At the end, the interaction with the Run Control and the Monitoring system is presented with particular aspects of the LVL2 activity. All these interactions try to follow the ROB and LVL2 URs which are included at the appropriate place in the document.

3.1 LVL2 Processing Unit.

To collect data necessary to process the LVL2 algorithms, the LVL2 processing Unit sends data requests to ROBs.

3.1.1 Data requests

UR L2-ROI RoI_request Handling [2]
The ROB must receive an RoI_request and use it to select either RoI_data or the full fragment from memory. This data is then sent to the Level-2 Trigger. The RoI_request may also contain parameters which define further processing which may be required on the fragment. These are described in the following URs.



UR L2-REF RoI_data Reformatting [2]
The ROB may have to reformat the RoI_data before transmission to the Level-2 Trigger.

UR L2-PRE RoI_data Pre-processing [2]
The ROB may have to pre-process the data before transmission to the Level-2 Trigger.

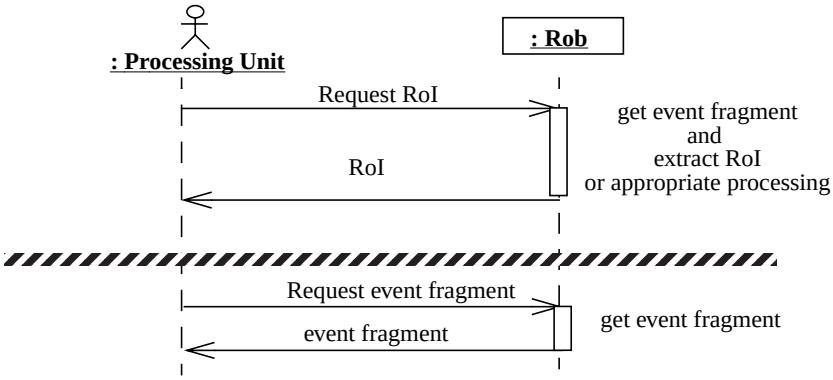
UR L2-COMP RoI_data Compression [2]
The ROB may have to apply compression to data before sending it to the Level-2 Trigger.

UR DI-EXP Data Expansion [2]
Data sent from the front-ends may be compressed before transmission. Since the ROB may be required to reformat and pre-process the data, it may have to apply expansion to data coming in

from the ROD. This would only be performed on fragments which were selected by an RoI_request.

Up to now, no request types have been defined to include a specific preprocessing on data except for RoI extraction in an event fragment.

The sequence diagram corresponding to this scenario is the following:



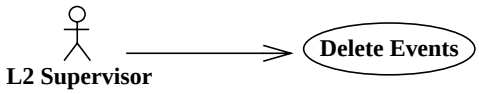
3.2 LVL2 Supervisor

The LVL2 supervisor is the central actor of the LVL2 trigger architecture. It receives the results of the Level-1 trigger, dispatches the events in different processing units and collect all results. It delivers these results to all ROBs.

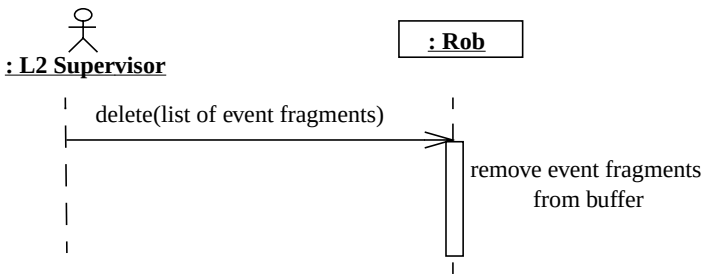
The two possible results are an accept or a reject decision. Several supervisors will be used to cope with the required rate of the global LVL2 system but for the ROB it makes no difference.

3.2.1 L2 reject / delete event

UR L2-REJ L2_reject Handling [2]
The Level-2 Trigger generates an L2_reject signal for unwanted fragments which should cause the ROB to release the buffer space relating to that fragment.



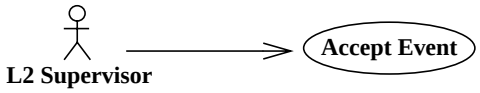
For efficiency, the delete message includes a list of identifiers of events to be deleted.



3.2.2 L2 Accept

UR L2-ACC L2_accept Handling [2]

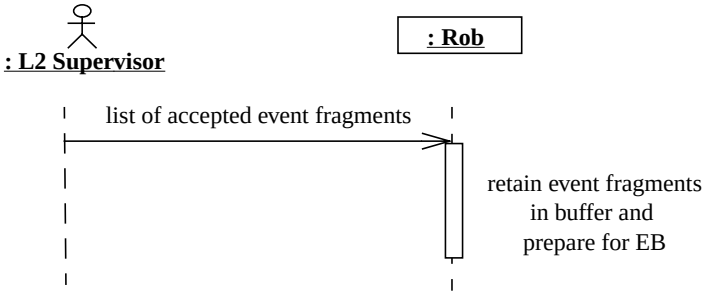
For selected fragments, the ROB receives an L2_accept which causes it to retain the fragment in memory until the data has been transferred to the Level-3 Trigger.



In the scenario proposed by the DAQ, the L2 Accept is used to prepare data to be delivered to the EB/EF or according to UR L2-ACC only kept in memory. On the other side, the DataFlow Manager of the EB sends to all ROBs the address of the processor where the data have to be sent. If required, data can be re-formatted, expanded and local event building can be started between the L2 Accept and the processor address delivery by EB.

The L2 Accept message can contain a list of event identifiers to be retained in memory.

The sequence diagram corresponding to this scenario is the following:



An other scenario based on an architecture similar to the LVL2 architecture has been proposed for event building. The L2 Accept is not directly sent to ROB but to EB. The later distributes to all ROBs the L2 Accept with the address of the farm which will process the event.

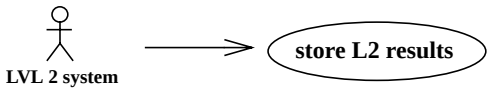
3.2.3 result storage

Two L2 URD exist about that point.

UR-OUT 2 LVL2 output storage [4]
The data generated by the LVL2 processing (RoI building, pre-processing, feature extraction, global processing and internal error conditions encountered during processing) shall be transferred to the Event Filter system via standard ROBs, depending on the operational mode and on the final LVL2 decision(accept or reject). The original RoI data received from LVL1 shall be included. The data shall be sufficient to permit checks on LVL2 algorithm correctness and trigger efficiency to be performed at a subsequent stage or off-line. It shall be possible to include data to guide Event Filter processing.

UR-ROB 5 LVL2 decision record [4]
Level 2 shall send decision records to ROBs in a format which obeys the ROB-ROD protocol.

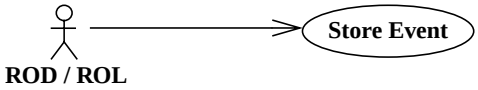
According to UR-OUT 2 , several LVL2 components can provide information to be stored. To group all informations for an event in the same ROB data block, the best strategy consists to use



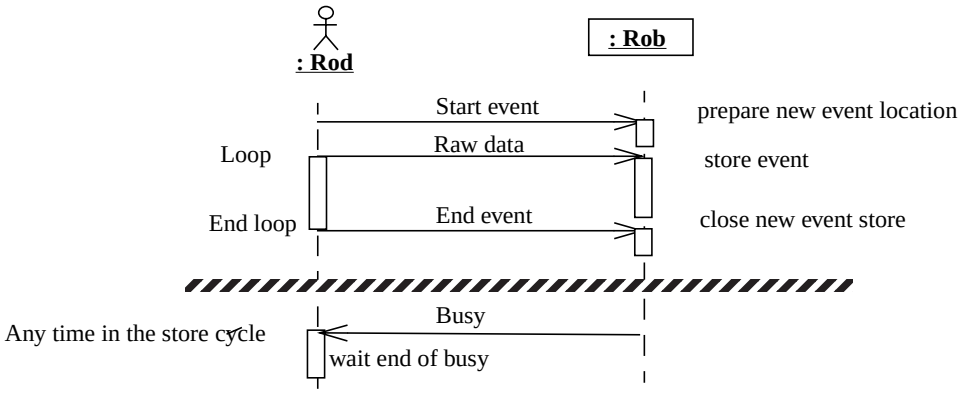
the event fragment format. The supervisor collects all these informations after the reply of each processing unit and seems the best entity to build the complete LVL2 result.

3.3 Detectors via RODs (Read Out Driver).

The input data flow drives the data storage. No external signal is used to trigger it. Each event is flanked by two control words, start and end of event.

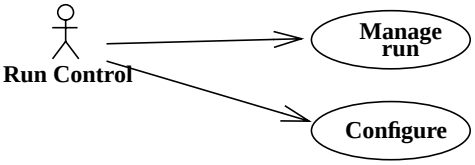


The sequence diagram corresponding to this scenario is the following:



3.4 Run Control

UR CTL-SC State Changes
The ROB must respond to commands from Run Control. These will consist of commands to change to one of a series of defined states.



UR CTL-CONF Configuration
The ROB will be configured on power-up by an external controller. This controller could load programs and parameters.

3.5 Level-2 Monitoring.

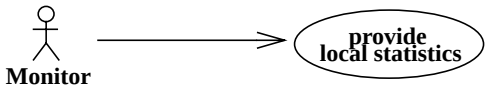
UR ERR-MON Error Monitoring
The ROB must be able to monitor all kinds of error conditions and keep statistics.

UR GBL-HIST History Maintenance
The ROB must keep a history record of all signals and messages received.

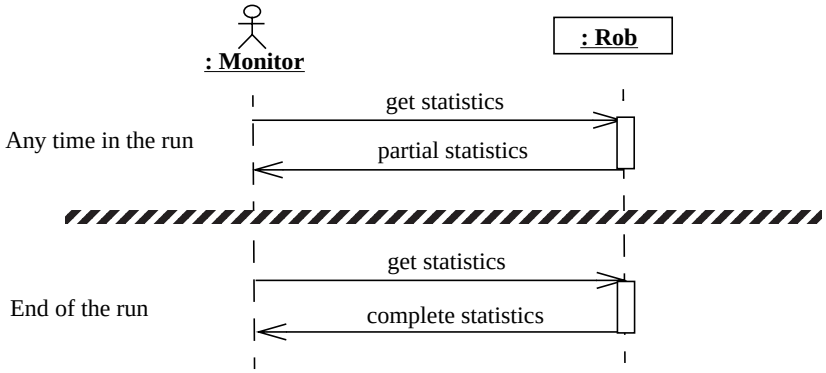
UR GBL-PERF Performance Recording
The ROB must maintain statistics on performance parameters such as buffer occupancy.

UR GBL-MON Monitor Data Handling
The ROB must respond to user request to send selected data to a parallel (to the data flow) server for monitoring of data quality. User requests will take the form of random or occasional tagging of events which must then be packaged and sent along the monitoring data path.

No monitoring on event fragment data is foreseen by the Level-2 community. Only the behaviour of the ROB in its Level-2 activity can be transmitted as statistics to a Monitor



This is an example of possible exchange between monitor and ROB.



3.6 TTC

Contrary to the ROB URD no L1_inhibit is foreseen from the ROB to a central Trigger. It is agreed that, only back pressure will be applied on corresponding ROD via the ROL data transmission protocol [5]

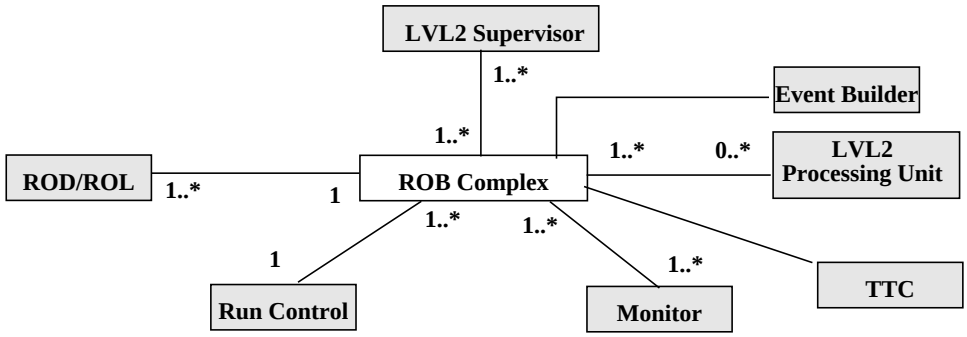
UR TTC-ID ID reception
Each time a trigger is issued, the ROB must receive, from the TTC or its interface module, the event_ID, the bunch_crossing_ID and the trigger type.

This is a DAQ requirement, in particular to drive it in partitioned mode during calibration runs.

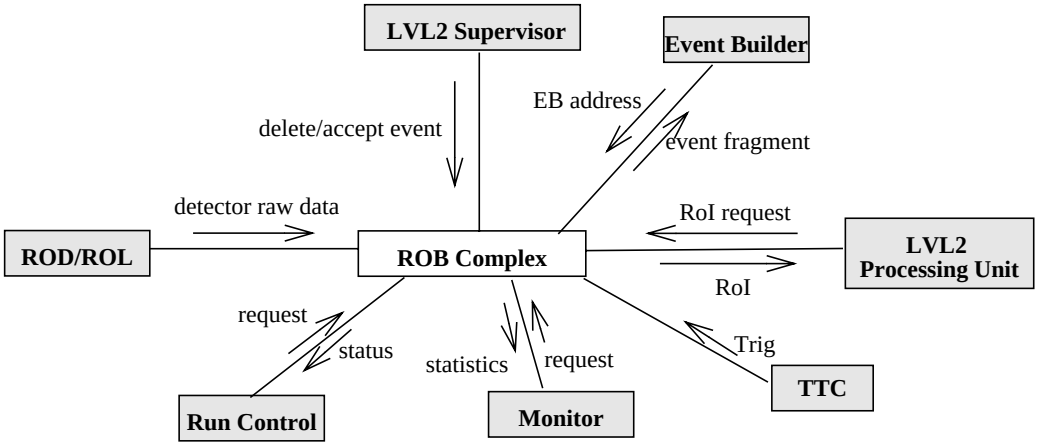
3.7 To summarize.

ROB is connected to:

- several supervisors
- several processing units
- 1 (or several) monitor
- 1 run control
- several links from detector (ROL)



The exchanged messages are:



4 High level ROB components

In this chapter we try to show what are the main components necessary to the ROB. For each "use case" presented in previous chapter we define what are internal components able to answer to the required functionality.

In a first step we can intuitively propose some components to deal with the ROB functionality. After, their own functionality is refined for each previous Use Case.

4.1 Proposed objects in a ROB.

4.1.1 Event Manager

The main functionality of the ROB is the storage of event fragments during a certain laps of time. We must define an object able to give access and manage the event fragments. This object knows the location of event fragments according their identifier (event id [6]). It must register and provide the location of the event fragment and remove the event fragment from the ROB.

4.1.2 Event Buffer

We must define a repository for event fragments to be stored in the ROB. We propose to call it Event Buffer. It must be able to provide information on free (and/or used) space, to store event fragment.

4.1.3 Input port

Data from the detector are transmitted from the Read Out Driver (ROD) to the ROB Complex via a Read Out Link (ROL)[7].

4.1.4 Message manager

The large number of ROBs and LVL-2 processors does not allow the direct access of the processors to the data of the ROBs. Only the ROBs know the location of the event data in their event buffer. That leads to an interaction between the processors and the ROBs by a message delivery.

The management of messages is common to all interactions between LVL-2 actors and ROB. It deals with reception of messages, dispatching in several local mail boxes and, if required, with sending of the reply.

4.1.5 Ports to/from LVL2 and DAQ

Messages are transmitted to/from other nodes of the Trigger/DAQ through networks. The ROB is connected to these networks via specific network ports.

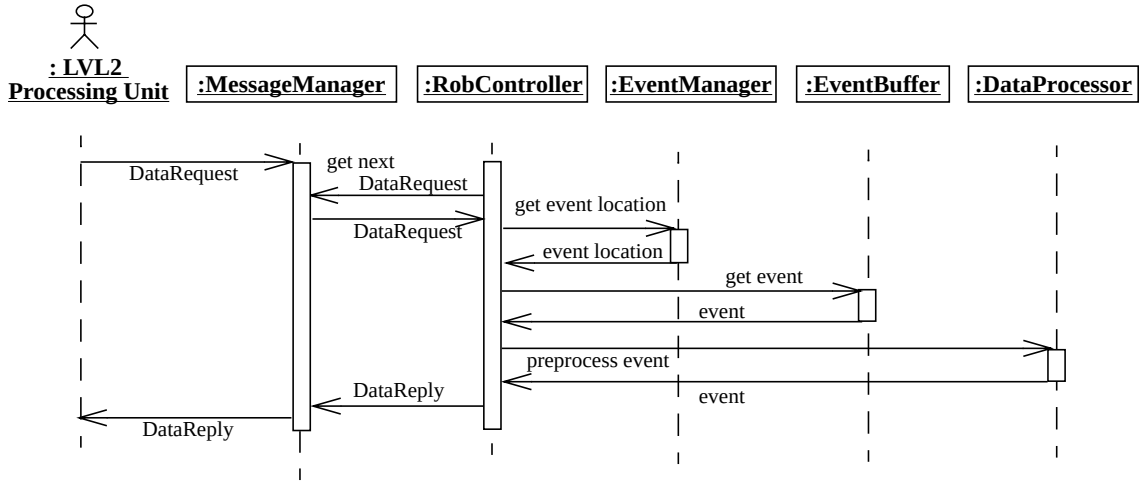
4.1.6 ROB Controller

A central object can be used to schedule the activity of the ROB. Its role consists in sharing time and activity between the objects described above.

4.2 Data Request

For the Level-2 trigger, data are mainly RoIs. RoI can cover a part or the whole ROB. In last case it is an event fragment. The RoI request includes the event identifier, the particle type and its location in the detector or the RoI limits for the particle.

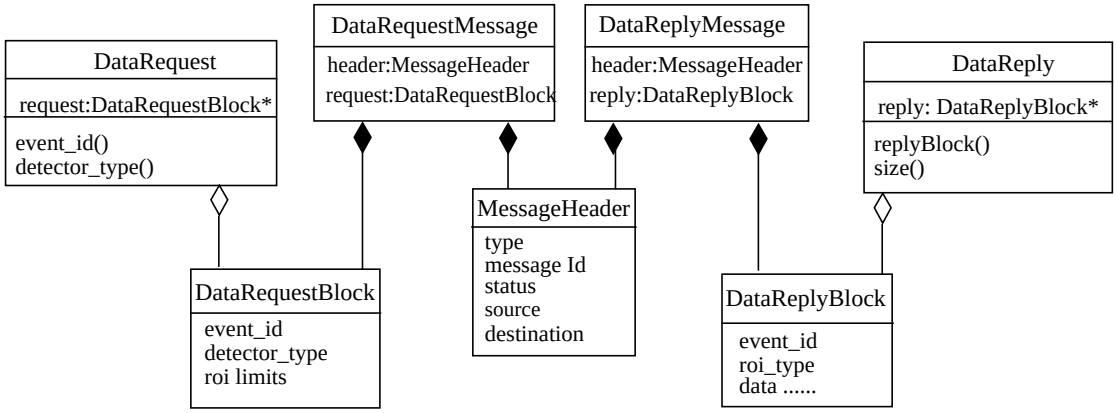
The RoI request is received by the message manager. The Rob controller gets the request when ready to process it. The event manager provides the corresponding event location. Then, event data are processed according to the request of RoI. A reply is built to be returned. The message manager returns RoI data to the requesting processing unit.



The presented scheme is not mandatory. Here, it is developed but it is possible to compress the path followed by data. In particular, in some cases, data location can be directly provided to the message manager instead of a copy of the event. In the Rob Controller, only the frame of the reply is built. In the same way, the data processor can access to the event buffer to get data to

be pre processed and location of the result transmitted to rob controller. All these variants are implementation issues.

The structure of the message can be the following

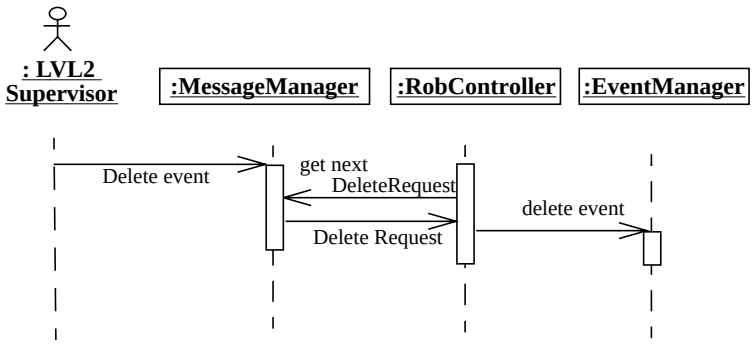


The actual request is the DataRequestBlock. To travel, it is dressed with the DataRequestMessage. A MessageHeader is added at top. When sited in a node of the system (Rob or Processing Unit) a DataRequest class gives access to the inside of the request.

The same mechanism is used for the DataReply.

4.3 Event delete request

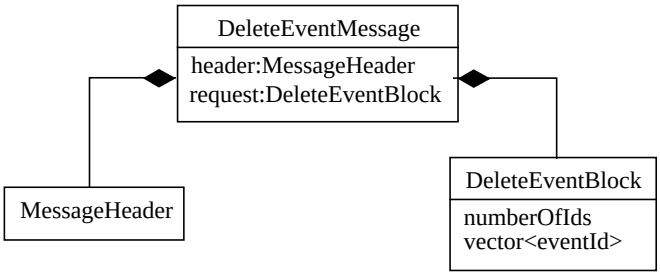
Up to the Rob Controller, the path is identical to previous request. When the rob controller has got in the corresponding mail box the delete of event to be deleted it requests the event manager to remove the events from its event table.



The event delete request includes several event identifiers to reduce the overhead due to multi-message headers and multi message transmissions.

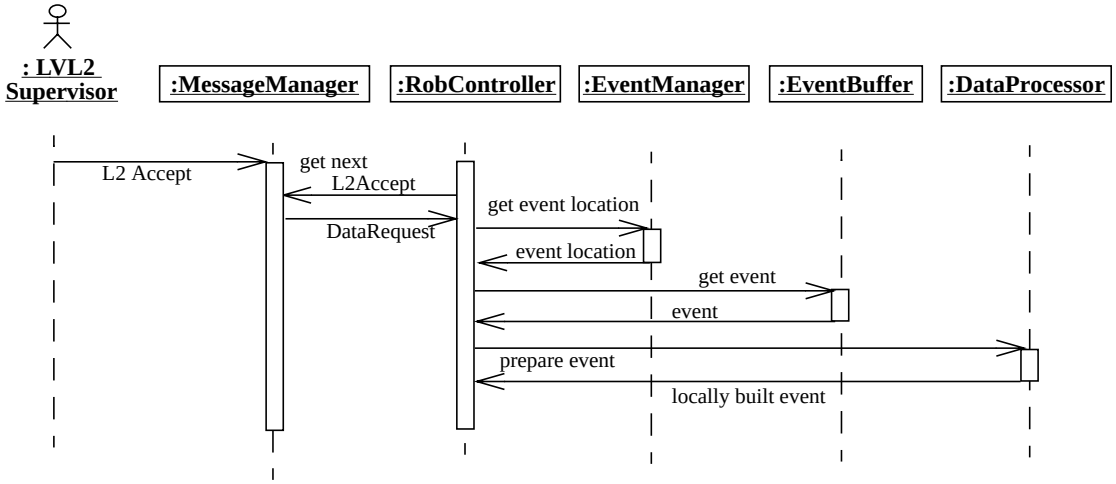
The event delete message is also sent by the EF when the event building has finished successfully.

The delete event message can have the following structure.

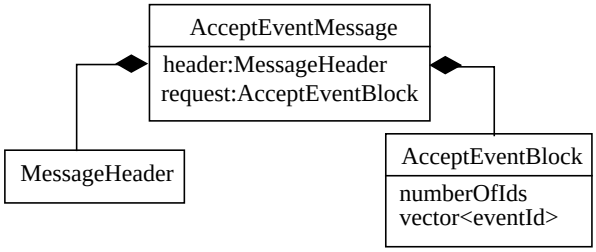


4.4 L2 Accept

According to the ATLAS TP, this scenario can be compared to a Data request but with no return of the prepared data. They are kept, ready to be transmitted to the Event Filter on request.



Like for the delete event message, the accept event message can group several event identifiers.



4.5 store event fragment

The input port receives data from the Read Out Link. When the ROB is unable to receive data it has to set a busy signal for the upstream partner (UR DI-FC). The input port is characterised by its rate (UR DI-IDR and UR DI-Rate) for the data flow and by control words creating a frame around the data.

UR DI-RATE Fragment Arrival Rate

The ROB shall be able to handle a fragment arrival rate of up to 100 KHz without dead time losses. Incoming fragments will not be de-randomised in time and so, at peak times, fragments can arrive immediately following each other. However, there are some limitations and, in particular, there can be no more than 16 fragments in any 16 μs period.

UR DI-IDR Input Data Rate
The Read-out Link will transfer data to the ROB at a rate of 100 MByte/sec. The ROB shall be able to handle such an input data rate.

Note that this number has been increased to 160 MByte/sec following the Calorimeter requirements.

UR DI-FC Flow Control
The ROB must be able to control the flow of input data on the Read-out Link in the event that its buffer becomes full.

Raw data is stored in the event buffer. It must be able to store fragments with variable lengths and to be large enough to keep events up to the EB requests.(UR L2-BS). Several types of event buffer can be considered. It will be described in the next chapter.

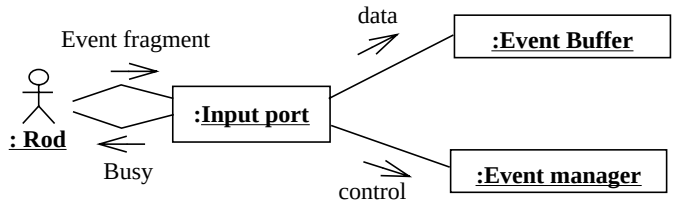
UR DI-ALLO Buffer Space Allocation
The ROB must allocate a buffer space for each incoming fragment. The ROB can read the event_ID in the header of the event and also can cross-check this with TTC_IDs (event_ID plus bunch_crossing_ID) generated by the TTC receiver board which is accessible by the ROB.

Note that the cross-check with the TTC information is not assumed in any design.

UR DI-EFS Event Fragment Size
The ROB shall be able to receive event fragments of various sizes

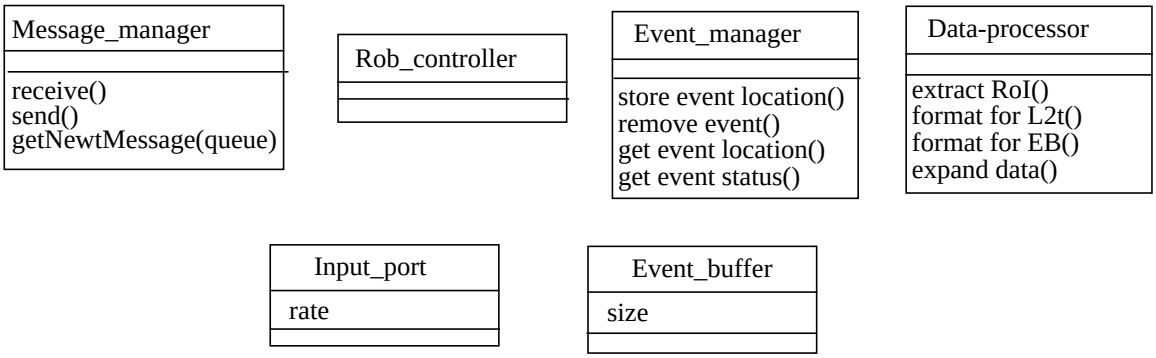
UR L2-BS Buffer Size
The ROB must be able to store incoming fragments until the Level-2 Trigger has issued an L2_accept or L2_reject. It must, thereafter, continue to store accepted fragments until they have been sent to Level-3. A formula is given to calculate the buffer size.

Control words are used by the event manager. It memorises the location of events in the event buffer.

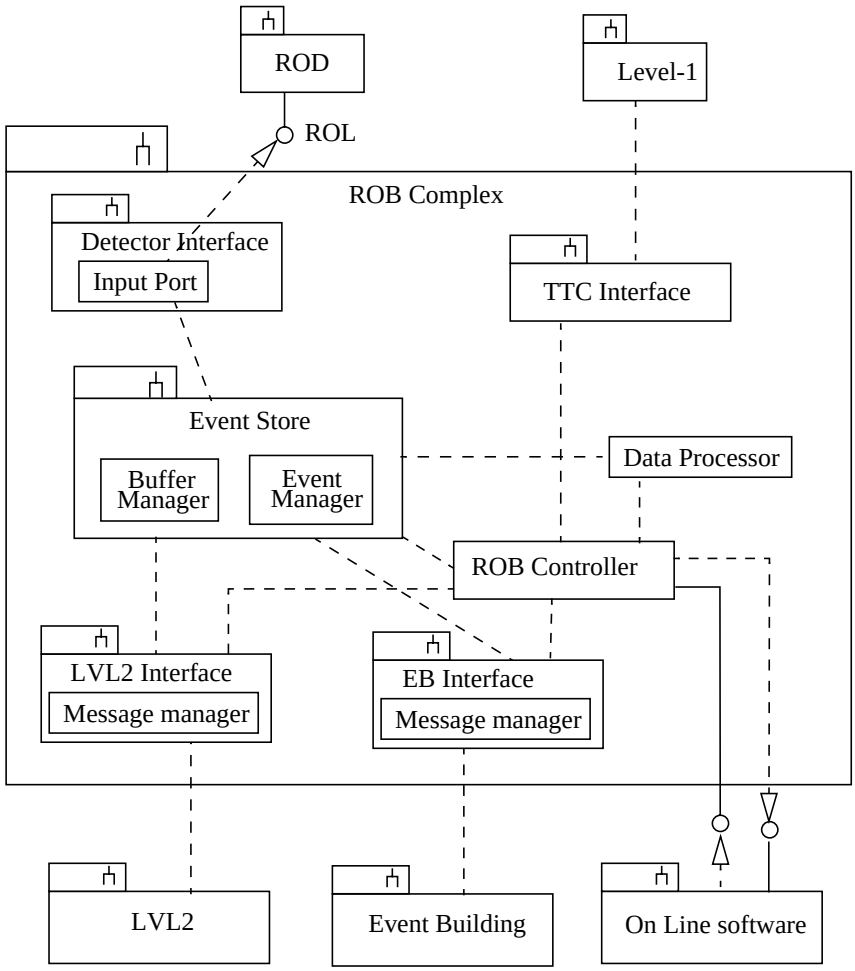


4.6 Global view

We have defined the following classes of objects to build a ROB.



It is possible to describe the ROB Complex as an assembly of sub-systems which include the classes described above.



5 Conclusions

The architecture of the ROB Complex as studied by the Level-2 Trigger community was presented. This high level design has showed the main components used to build this system. The "reference software[8]" from the "Level-2 Pilot Project[9]" has implemented a part of this design. The Event Store is emulated and the LVL2 Interface is completely implemented to receive and return messages form/to the Level-2 system. The Input Port has been implemented in the DAQ prototype[10].

References:

[1] UML Documentation; http://www.omg.org/uml/

[2] ATLAS ROB User Requirement Document (ROB-URD-V1.3.0); http://www.cern.ch/HSI/rob/URD/

[3] Specification of the LVL1 / LVL2 trigger interface, Version 1.0, M.Abolins et al., ATL-DAQ-99-015, 7 Oct. 1999.

[4] ATLAS Level-2 Trigger User Requirement Document (LVL2-URD-1.00); http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/LEVEL2/DOCUMENTS/urdv1.ps

[5] Timing, Trigger and Control, and Dead-time handling. Ph.Farthouat. ROD meeting. Geneva 1998

[6] C.Bee et al. The event format in the ATLAS DAQ/EF prototype -1. ATL-DAQ-98-129.

[7] http://www.cern.ch/HSI/s-link/spec/

[8] Level-2 Reference Software; http://www.cern.ch/Atlas/project/LVL2testbed/www/

[9] Level-2 Pilot Project; http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/L2PILOT/

[10] S.Veneziano. The Read-Out Crate in ATLAS DAQ/EF prototype -1; CHEP 2000