Table of Contents

L0Muon Software

L0Muon event model

see L0Muon event model

(top)

L0Muon emulator

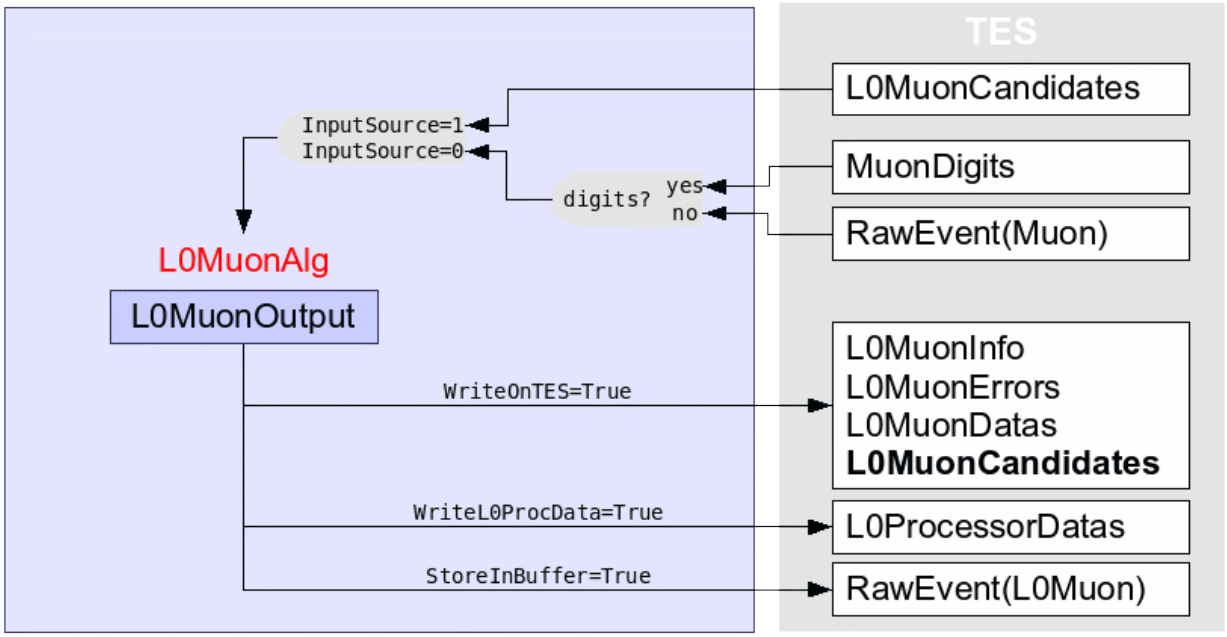
The L0Muon emulator emulates the L0Muon trigger processing.

It is based on 2 main generic classes : Unit and Register which are defined in L0/ProcessorKernel package (LHCb project).

These classes are specialized in L0/L0MuonKernel where is implemented the L0Muon processing.

The emulator runs in L0MuonAlg. It is configured using the L0MuonKernel.xml file in the PARAM/ParamFilesParam.

- L0Muon emulator event model diagram:

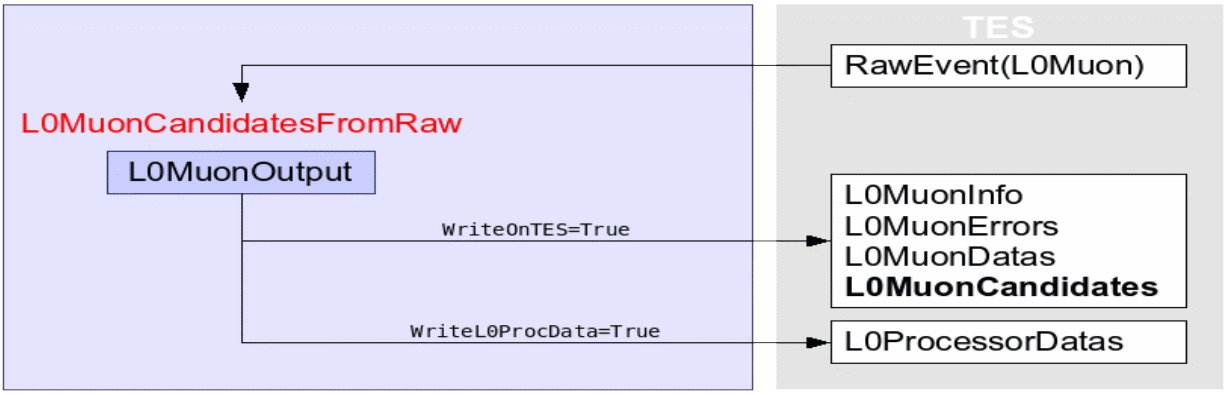


(top)

L0Muon banks decoding

The decoding runs in L0MuonCandidatesFromRaw. It is configured using the L0MuonKernel.xml file in the PARAM/ParamFiles package. The decoding also depends on L0/L0MuonKernel.

- L0Muon decoding event model diagram:



- The L0Muon bank content is described in EDMS818559.

(top)

L0Muon monitoring

to be completed

(top)

L0Muon data bases

2 separated data bases are used to describe the L0Muon (see EDMS818447) :

- the L0Muon configuration data base
- the L0Muon cabling data base

The L0Muon condition data base contains information relative to the running condition.

L0Muon configuration DB

describes the L0Muon hardware architecture and components as well as some implementation details (boards, FPGAs, optical & internal links connections & content, ...).

There is one such a DB per quarter.

They are created the appCreateDB.py application in L0/L0mPy.

The L0Muon configuration DB is used to generate :

- back plane inter-connection list used in the CA0 of the backplane
- vhdl codes used in the PU FPGA
- xml configuration files for emulator and decoding software

L0Muon cabling DB

describes the cabling between the muon ODEs and the L0Muon

Condition data base

Some parameters defined at run time are stored in the condition data base :

- for each controller board (i.e. each quarter) :
  - ♦ Processor version
  - ♦ CU mask
  - ♦ SU mask
- for each processing board :
  - ♦ Fields of interest (X and Y)
  - ♦ Optical link masks and values
  - ♦ Backplane masks
  - ♦ PU mask

Note : the processor version defines the version of :

- the various FPGA firmwares
- the LUTs with the PT information
- the raw bank (integer value of the processor version)

(top)

L0Muon packages

LHCb project

L0/ProcessorKernel (CVS)

This package contains generic classes used by the L0Muon emulator and decoding software :

- Register : contains a boost::dynamic_bitset.
- TileRegister : special Register which contains in addition a list of MuonTileID associated to the bitset.
- RegisterFactory : factory owning the registers. 3 different instances of the RegisterFactory may be created.
- Unit : a Unit contains :
  - ♦ a list of pointers to registers used as inputs
  - ♦ a list of pointers to registers used as outputs
  - ♦ a list of pointer to 'child'-units (thus, the units forms a hierarchical strucuture)
  - ♦ a list of properties than can be defined at run time
  - ♦ a set of virtual methods where specialized units (see L0/L0MuonKernel) will perform specific tasks using the input registers and fill their output registers
    - ◊ initialize
    - ◊ preexecute
    - ◊ execute
    - ◊ postexecute
    - ◊ finalize
- Property : generic class to set the Unit's properties

L0/L0MuonKernel (CVS)

Content :

- a list of useful functions :
  - in BankUtilities(.h/.cpp) : for bank decoding
  - in ProcUtilities(.h/.cpp) : for processing

- usefull class to manipulate candidates :
  - MuonCandidate
  - CandRegisterHandler

- a set of specialized units inheriting from L0MUnit. One can distinguish :
  - Pure containers necessary to build the hierarchy of units (MuonTriggerUnit, CrateUnit, BoardUnit, ProcUnit)
  - Working units where a task is actually performed; the input register are read, a task is perform, result is set in the output registers. 3 type of tasks can be considered :
    - data formatting (FormattingUnit, FormattingOutUnit)
    - candidate search (CoreUnit)
    - candidate selection (BCSUnit, CtrlUnit both inheriting from SelectionUnit)

- a set of 'converters' used to extract information from a set of Registers and TileRegisters corresponding to data written in raw banks or L0Buffer

- a set of 'error classes' used to handle the errors field in the converters

Unit tasks flow :

- At the beginning of each job :
  - initialize
- For each event :
  - preexecute : *the data exchange between the processing unit is emulated by the FormattingUnit*
  - execute : *the candidate are search by the CoreUnit*
  - postexecute :
    - *the candidates are selected by the BCSUnit and then by the CtrlUnit*
    - *each FormattingOutUnit performs the data formatting in view of the output in banks or L0Buffer by the converters*
    - *the registers are cleared*
- At the end of each job :
  - finalize

Lbcom project

L0/L0Muon (CVS)

- L0MuonAlg : algorithm to run the L0Muon emulator.

- L0MuonCandidatesFromRaw : algorithm to decode the L0Muon banks (except the L0MuonRaw banks).

- L0MuonFromRawTrans : algorithm to decode the L0MuonRaw banks (transparent mode).

- L0MuonOutputs : tool to write on TES the output of the various L0Muon algorithms.

L0/L0MuonMonitoring

(top)

Additional packages

L0/L0mSQLite (CVS)

L0/L0mConf (CVS)

L0/L0MuonPyTools (CVS)

L0/ReportLab (CVS)

L0/L0mPy (CVS)

This package is used to fill the L0Muon configuration data base (see above).

L0/L0MuonKernelBuild (CVS)

L0/L0mCAO (CVS)

L0/L0mTestBench (CVS)

L0/L0Analysis (CVS)

L0/L0MuonPresenter (CVS)

top

-- JulienCogan - 02 Oct 2008

This topic: LHCb > L0MuonSoftware
Topic revision: r10 - 2008-10-09 - JulienCogan