DØ/TRIGGER/3681
August 25, 1999, Version 1.0 October 29,1999, Version 1.1

Data Structure for the Level 2 Silicon Tracking Trigger Detector Readout

Silvia Tentindo Repond
Florida State University

Abstract

This note details the contents of the Level 2 Silicon Tracking Trigger readout and the corresponding Unpacked Data Chunk. It is designed for users of the Trigger Simulator as well as for people working on the readout and online Examine for STT monitoring. It is intended to be a working document for the members of the STT group.

1 Introduction

This note describes the structure of the readout for the Level2 STT preprocessor. Section 2 provides some details about the D0 RunII Trigger Simulator, to whom the STT Trigger Simulator belongs, and where the STT readout has been implemented. Section 3 describes some of the constraints and inputs used to design the STT readout. Section 4, 5 and 6 describe the readout structure. Section 7 describes the current status of the project.

2 Some remarks about the D0 RunII Detector and Trigger online Data Readout

2.1 Detector and Trigger online readout

In the Run II software environment [1], as well as in L3, real data is stored in the form of a Raw Data Chunk (RawDataChunk or RDC), defined in a package called l3base. An RDC is the sum total of the data read out from the Front End (FE), Level 1 (L1) and Level 2 (L2) components of the trigger and is used as input to Level 3 (L3). Offline access to these data is simplified through the use of the Unpacker [2] which converts the data from the compressed RDC format into readily accessible pieces called Unpacked data Chunks (UnpData Chunks or UDC's). There is one UDC for each detector or trigger component (e.g. the L2STT preprocessor readout is contained in the STT_L2 UDC). The complement to the Unpacker is the Packer which takes any number of UDC's and can pack them into a single RDC.

2.2 Trigger Simulator and readout issues

These two complementary utilities, the Packer and the Unpacker, greatly simplify the trigger simulation framework [3] and allow the design to be modular. This has the added benefit of making creation and comparison of trigger results to simulation straightforward.

In the RunII trigger simulation, for L1 and L2, the simulation of a single component of the trigger (e.g. the L2STT preprocessor) is implemented as an individual framework package (tsim_L2STT) which reads in data in the form of UDC's (e.g. L1_CTT and SMT_FE) and outputs its results to a single UDC (e.g. L2STT).

3 Requirements for the L2STT readout

The L2STT readout must fulfill the requirements described in the STT TDR [4] and contains the parameters calculated by the STT preprocessor itself. The L2STT chunk must also fulfill the requirement to contain and be able to forward all the information it receives from the L1CTT processor, i.e. the CFT and PS information.

The readout for STT contains three types of data that go to L3 [1] : data information, Header and auxiliary information. The Header is described in the following section. The

[1]Note that for Simulation the L2STT chunk declares FT_L1 and SMT_FE 'parent' chunks.

data information contains the parameters for all the tracks in the event and is described in sections 5 and 6. The structure for the data information has been defined in such a way that the information from CFT and PS is read out and saved into the L2STT readout, together with the information specific to the STT processor,*on a track by track basis.* The auxiliary information, that will appear into the STT readout only for minimum bias and mark-and-pass events, is described in section 7.

4 Class HEADER of L2STT readout

In the UnpDataChunk scheme (CMS vsn 55 and 56), the Header is going to be a 'Channel' as well as the channel describing each track, with special value for module id and channel id .

```
 sysid : STT_L2

header[0]= processing_bits
header[1]= status_bits
header[2]= cluster_algo_version
header[3]= track_fit_version
header[4]= parity

modid = 99
chanid = 0

status_bits  content :
bit 0 = STT info    ( 0=OK; 1=error or missing )
bit 1 = STT cluster     ``       ``
bit 2 = STT road         ``      ``
bit 3 = STT track        ``       ``
bit 4 = STT vertex       ``       ``
bit 5 = L1CFT tracks       ``       ``
bit 6 = PS                ``       ``
```

5 Data Readout

5.1 Information from CFT and PS of L2STT readout

The FT_L1 is read out and the following information is stored for the L2STT readout:

```
data[0] = track phi
data[1] = track bin
data[2] = track pt
data[3] = track PS isolation
data[4] = track PS Match
data[5] = track error
data[6] = track isolation
data[7] = STT Track duplicate
```

```
data[8] = PS location info

modid = 0
chid = track number  (CFT numbering)
```

5.2 Information from the L2STT processor to L2STT readout

Here the results from the L2STT calculations (clustering, hit filtering, tracking and vertexing) will be stored. We had mentioned that the clustering and hit filtering are intermediate steps in the calculations, so they are not requested into the 'official' L2STT readout. Nevertheless there will be special events [monitoring sample] that will contain that more complete information. Also, for internal debugging and STT studies, we may want to create a chunk with more complete data. For the moment, this part of the L2STT chunk is filled with arbitrary values[2]

```
modid     bits 0-4 = slot#  (tracker=0)
chid    tracker = track ID # (0-31 from CFT L1) ---32 or 46?
        broadcaster # (0-5)
        # track broadcaster  ----> 7 bits

data[0] = phi0
data[1] = impact parameter
data[2] = pt
data[3] = chi2
data[4] = de/dx
data[5] = STT info not in header
```

The modid and chid are the same as for CFT and PS.

The information is accessed (from FT_L1), calculated (for L2STT) and stored on a track by track basis. The track numbering comes from CFT numbering. Cases of tracks seen in STT only need special treatment.

6 Special L2STT readout for Monitoring events

In case when all the components of L2STT processor will contribute to the output chunk[3] , this is the content:

```
Information for Clustering:
```

[2]Note for Simulation Programmers - The way the L2STT simulator works is similar for all its components. For example, for the clusters, there is a method called ClusterFactory, that contains the cluster algorithms, and provides the values that are stored into the class 'Clusters' as attributes. The L2STT Chunk will then get loaded with these attributes. Similarly for the 'RoadFactory' and the Class 'HitsInRoad' , or the 'TrackFactory' and Class 'Tracks'

[3]This document discusses the full STT readout. If we want debugging quantities for the simulator into the UnpDataChunk, these can only be put in the Chunk if they will be part of the *full STT readout*

```
   modid    bits 0-4 = slot# (cluster=3-31)
            bits 5-7 = crate # ( 0 -5 by sector )
   chid     bits 0-7 = channel # on the chip
            bits 9-11 = chip #
            bits 12-14 = HDI #
--> from class Cluster:
   data[0] = offset (1/4 strip position)
   data[1] =  size
   data[2] = pulse_height
   data[3] = channel
   data[4] = chip
   data[5] = HDI
   data[6] = Sequencer
   data[7] = crate
   data[8] = Address
   data[9] = Barrel
   data[10] = Layer
   data[11] = Ladder
   data[12] = View
   data[13] = x
   data[14] = y
   data[15] = z
```

*** Rem : data[3] to data[7] give the hardware address of the cluster centroid, same as provided already in modid and chid.

```
Information for Roads:
        modid    bits 0-4 = slot# (road=2) - 1 or 2 if CPS axial or
                                             stereo
        chid     road = road#
        evid     event#

 --> from class HitsInRoad:


        data[0] = track flag (0 or 1)
        data[1] = track number
        data[2] = H layer hit ot Pt bin
        data[3] = A layer hit
        data[4] = r coord. of H layer hit
        data[5] = r coord. of A layer hit
        data[6] = phi coord. of H layer hit
        data[7] = phi coord. of A layer hit
        data[8] = number of clusters associated to road
        then loop over clusters:
```

```
data[10] = 1/4  strip position (offset)
data[11] = clusters centroid strip number
data[12] = ...
data[..] =
end loop over clusters


Information for Vertex:
       modid   bits 0-4 = slot# (vertex=1)
       chid     vertex  = vertex #

       data[0] = vertex_z
       data[1] = number of tracks on vertex
```

7 Current status of the project

Currently we are testing the design of the STT readout by filling it using the STT Simulator (L2STT chunk). At present the STT Simulator is still under development. Specifically, the clustering and the road finder components are operative but do not contribute information to the so far agreed L2STT chunk. The track fitter of STT is not in yet. Its present release to CVS [v55 or v56] is meant to enable other components of the D0 Trigger Simulator (specifically CTT_L2) to develop the code for read out from L2STT. Its release to v57 is in progress.

This document reflects the agreements made at several STT and Trigger Meetings and with the CTT experts. It should be considered as a proposal to the hardware experts of the STT Group, subject to improvement and modifications for the part containing STT specific attributes. The part containing information from the CTT system has been discussed and agreed with the CTT experts, and new proposals will more improbably lead to modifications.

8 Acknowledgments

This note is an upgrade from the draft written on June 23, 1999 and comes from discussions made during the Seattle workshop specifically with Dave Toback, and more recently with Manuel Martin.

References

[1] http://www-d0.fnal.gov/run2_offline_software/framework/framework.html

[2] http://niuhep.physics.niu.edu/~fortner/d0/algo/unpack

[3] Overview of the D0 RunII Trigger Simulator - D.Toback *etal* - D0Note 3555

[4]
A Silicon Track Trigger fot the D0 Experiment in RunII - Proposal to Fermilab - D0Note 3516