Requirements to the L0 front-end electronics

LHCb Technical Note

Issue: Second version
Revision: 1.0

Reference: LHCb 2001-014
Created: July 1999
Last modified: July 3, 2001

Prepared By: Jorgen Christiansen

Abstract

The major parameters and features of the front-end electronics performing data capture and data storage during the latency of the level 0 trigger (L0 front-end electronics) is defined and analysed. Based on this analysis a set of requirements to the L0 front-end is specified.

This document supersedes LHCb note FE 99 - 29

Document Status Sheet

Table 1 Document Status Sheet

1. Document Title: Requirements to the L0 front-end electronics			
2. Document Reference Number: LHCb 2001-014			
3. Issue	**4. Revision**	**5. Date**	**6. Reason for change**
Draft	0.1	30 July 1999	First preliminary version
Draft	0.2	22 November 1999	Updated version after front-end electronics workshop Released as LHCb FE 99-29 first version
Final	1.0	July 2001	DCS changed to ECS Updated bunch structure Definition of time adjustment range. Definition of calibration pulse injection limitations. Availability of interaction trigger added Require L0 latency programmable: 2.0us – 4.0us Use of L0 reset for B-ID at input to L0 pipeline Inclusion of timing diagrams of B-ID and L0 reset Error status data tag emphasised Comment on number of words per event in L1 buffer L0 reset redefined Use of TTCrx strobe2 for short broadcast not allowed. Short broadcast definition updated ECS interface must not have SEU induced lock-up. Small chapter on analogue front-end added Released as LHCb 2001-014

Table of Contents

1. Introduction

The L0 front-end electronics [1] must correctly capture and store detector signals, from particles generated by the bunch collisions of the LHC machine on a large number (~1 million) of electronic channels. The detector signals must be captured with sufficient time resolution, to determine the exact bunch crossing from which the particles originate. Captured data (analogue or digital) must be stored in the level 0 trigger pipeline buffer, until the level 0 trigger decision accepts or rejects data from a given bunch crossing. Accepted data must be extracted from the pipeline buffer and temporarily stored in the L0 derandomizer buffer, waiting to be transferred to the following L1 front-end electronics.

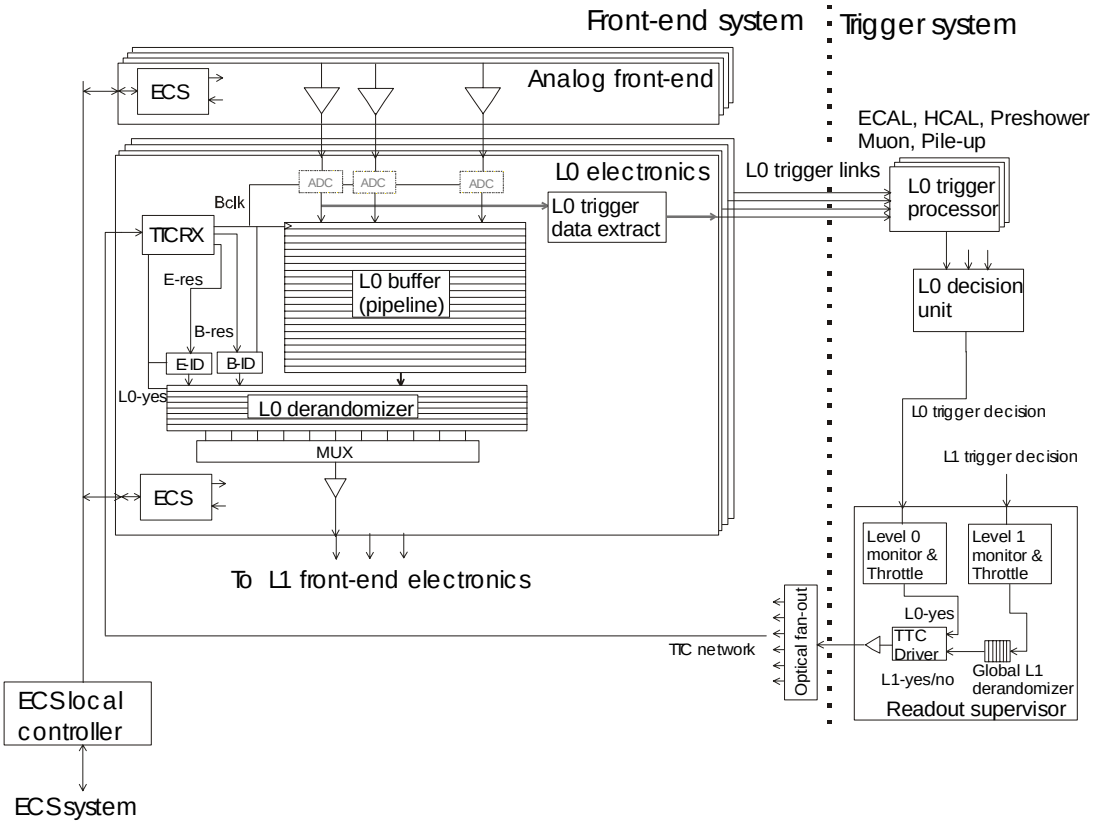

Figure 1. General architecture of the L0 front-end electronics.

A predictable behaviour of the L0 front-end electronics system is of special importance in the LHCb experiment, because of the high bunch crossing rate (40 MHz) and the relatively high trigger rates. All front-ends must be perfectly synchronised to the beam crossings, to insure correct capture of detector signals. The arrival of the L0 trigger-accept to the front-end must also be in perfect synchronisation, to insure the extraction of correct event data from the level 0 pipeline. Caused by the relative high level 0 trigger accept rate, event data from the Level 0 pipeline must be temporarily stored in a derandomizer buffer before being transferred to the L1 front-end electronics. The size of this derandomizer buffer, and the speed of which data can be transferred to the L1 buffer, is a major bottleneck in the LHCb front-end that has a significant impact on the physics performance of the experiment. The L0 trigger-accept rate must be limited centrally, based on an emulation of the L0 derandomizer buffers. For such a scheme to work reliable for a

large system, it is required that the L0 pipelines and the L0 derandomizers work in perfect synchronisation across the whole experiment. Otherwise one would have to reduce the L0 trigger rate significantly, to insure that no buffers overflow in any part of the front-end, or build a very large and fast signalling network which could throttle the level 0 trigger, when any front-end buffer risk to overflow.

The TTC (Timing and Trigger Control) system [3] is used for the distribution of all time critical signals to the front-end. The use of a common system for this ensures that the control of the different front-end implementations have a unified interface with a predictable behaviour. The Readout Supervisor [7], that receives trigger decisions from the trigger systems, drives the TTC system. The Readout Supervisor is responsible for preventing buffer overflows in the front-end and DAQ systems. During testing, debugging and special calibrations each sub-detector is driven from a separate Readout Supervisor via the partitioning system of the Timing and Fast Control system (TFC) [6]. During normal physics running all sub-detectors are driven from one common central Readout Supervisor.

In the large and complex front-end system, consisting of several completely different implementations for different sub-detectors, it is also important to define a minimum set of error checking and monitoring features. These kinds of functions are vital to insure that data collected from the experiment can be considered correct, with a sufficient confidence level, when working in a hostile environment (radiation).

The key parameters and requirements to the L1 part of the front-end electronics will be dealt with in a separate note, as these are, to a large extent, independent of the L0 front-end.

All front-end systems must comply with the definitions, rules and parameters given in this document. Any possible exception needs to be discussed with the front-end electronics coordinator and can only be officially accepted after an endorsement by the technical board.

A continuously updated short-form list of key front-end parameters is available on the LHCb web [8].

2. Analogue front-end

The analogue front-end must amplify the weak detector signals with minimum noise, and interface it to the sampling devices (e.g. analogue to digital converters) that allow the signal to be stored in the L0 pipeline (analogue or digital). In a high rate experiment, like LHCb, it is extremely important that the shaping time of the analogue signals is such that spillover from preceding bunch crossings is minimized and that minimum baseline shifts are introduced. In the very sensitive analogue front-end special attention must be paid to the immunity to external noise sources (power supplies, ground loops, electromagnetic coupling, etc.).

3. Bunch crossing rate

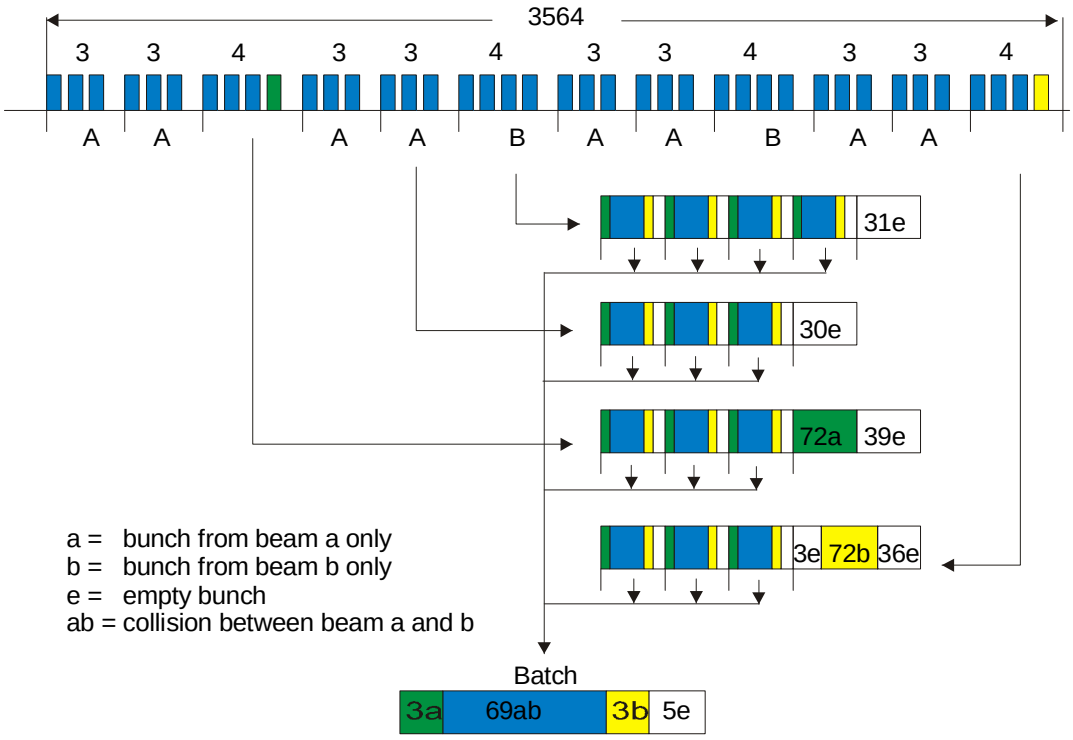

Figure 2 Bunch structure at the LHCb interaction point.

The bunch-crossing rate at the LHCb interaction point is given by the LHC machine to be 40,08 MHz with 2622 out of 3564 bunches having real collisions (different from ATLAS and CMS). At the LHCb interaction point (IP8) the two beams are 1/4 machine cycle out of phase (1/8 for each beam) plus three additional bunch periods caused by the 11.2 m displacement of the LHCb interaction point in direction of IP7 (to one side of the LHCb cavern). The bunch structure will possibly be changed during the design phase, commissioning and optimization of the LHC machine. Variations between fills may also be expected. For most recent information on the bunch structure it is recommended to refer to the LHCb electronics web pages [8].

4. Bunch crossing synchronisation

Each sub-detector must synchronize the sampling of detector signals to the bunch crossings of the LHC machine, to correctly identify the origin of the detected particles. The bunch crossing synchronization of the front-end is divided into a precise clock phase alignment and a coarse clock cycle alignment.

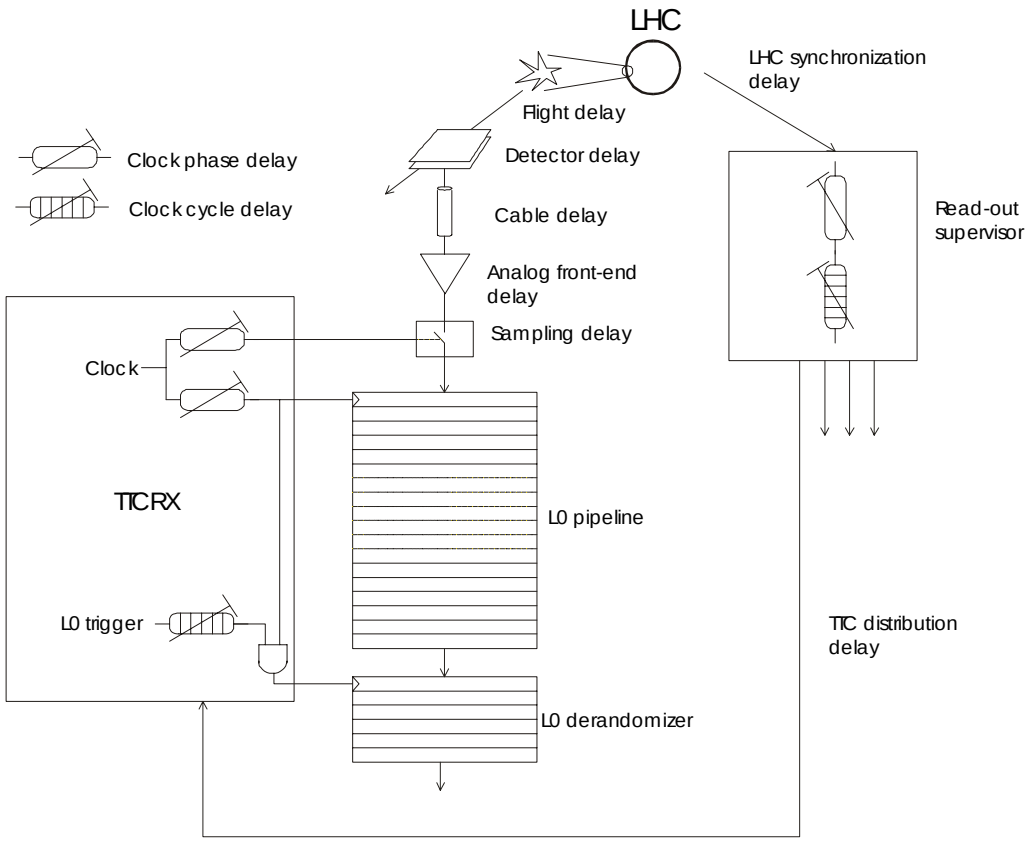

Figure 3. Bunch crossing synchronization of front-end.

The synchronization of the experiment is based on a set of synchronization signals from the LHC machine. The bunch-crossing rate is given by the LHC bunch-crossing clock, delivered with a constant phase to the real bunch crossings. The longitudinal dimension of the LHC bunches is expected to give a 175ps RMS (4σ = ~700ps) time variation in the real interactions in relation to this reference [3]. The synchronization to the LHC machine cycle is obtained from a LHC machine cycle synchronization signal. The LHC synchronization signals are received by the readout supervisor, which generates the required synchronization signals to the TTC system. The readout supervisor will be capable of delaying the machine cycle synchronization of up to one complete LHC machine cycle period.

Each sub-detector must implement sufficient hardware and software functions to perform the required phase alignment during special calibration runs, where sub-detectors run independently. In this mode each sub-detector is driven from independent readout supervisors that can generate specified trigger and control sequences (resets, L0 and L1 triggers, injection of calibration signals, etc). Real physics triggers cannot be generated at this point, as the trigger system and their corresponding detectors cannot be assumed to be correctly synchronized. During such runs the DAQ system will be partitioned into separate processing clusters for each sub-detector, to run the required software to perform the synchronization and calibration. In some cases, a detector can only be time aligned based on information from other detectors (co-incidence and track correlations).

A dedicated interaction trigger function will be available, at an early stage, to generate trigger sequences (single trigger or sequence of consecutive triggers) based on the occurrence of real interactions (to be described in more details in the trigger TDR). This interaction trigger can be conditioned on the fact that no interactions have been detected in preceding and/or following bunch crossings. In addition the readout supervisor can enforce that it only generates this trigger to the front-end in specific bunch crossings in the

LHC machine cycle (e.g. first or last bunch crossing in a bunch group). This trigger can in many cases reduce the required number of triggers, to perform a complete time alignment, especially when running at low luminosity. The special conditioning of the interaction trigger can in many cases resolve time alignment problems in detectors with serious background and/or spillover problems. Interaction trigger information, in the two bunch crossings preceding and following a L0 trigger, is recorded for every event. At nominal luminosity 44% (20%) of normal physics triggers will have no hard pp interactions in the two (four) neighboring crossings.

When individual sub-detectors have been properly time aligned to the bunch crossings, an LHCb wide time alignment between sub-detectors will be performed. In this case the whole experiment is driven from one common Readout Supervisor and the timing of complete sub-detectors are shifted by adding common timing offsets to their local timing adjustment parameters.

During normal running, the correct synchronization of the front-end system must be verified in the DAQ system by checking the consistency of accepted events and make histograms of event data. Special triggers (random, bunch gap, with pulse injection, etc.) can be generated at regular intervals to monitor the correct synchronization of the front-end.

Correct bunch crossing synchronization is in general obtained by analyzing collected event data, over a large number of triggers, and make adjustments to the front-end in an iterative fashion. It is required that a sufficient number of signal pulses can be collected on each detector channel (or group of channels) within a reasonable acquisition time (minutes). For high occupancy channels the collection of signal pulses and its analysis can conveniently be performed by the DAQ system with a maximum event rate of ~40KHz. For low occupancy channels it may be problematic to collect sufficient hits for all channels with an accept rate of ~40 kHz. It may in some cases be required that data is collected and analyzed at higher rates (1MHz), implying dedicated hardware in the front-end electronics. Each sub-detector must estimate the time required to collect sufficient event data to perform a complete bunch crossing alignment, taking into account a potential low luminosity of the LHC machine in the start up phase.

The local bunch crossing synchronization in the front-end electronics is in general performed using a set of programmable delay features in the TTCrx receiver chip as indicated in Figure 3.

4.1. Phase alignment

The sampling of the analog detector signals must be phase aligned to the bunch crossings with a precision and stability of a few ns (high-resolution detectors may need even better phase alignment). The bunch-crossing clock, delivered to the front-end by the TTC system, has a constant but unknown phase to the LHC machine clock (drivers, cable delays, receivers).

In some cases the clock phase alignment must be performed on a channel by channel basis. In most cases the clock phase alignment can be performed on groups of channels, when delay differences between channels in a group are kept under tight control (same time of flight, small detector differences, common high voltage supply, matched cable lengths, front-end chips from same production batch, small temperature gradients, etc.)

The clock phase alignment to the beam crossings will typically consist of a clock phase sweep, until sampling at the maximum signal amplitude has been obtained (peak find). Because of the relative low occupancy of detector channels in many detectors, a large set of triggers must be generated to collect sufficient information to insure a correct clock phase alignment of all channels. If a group of channels by construction has equivalent timing characteristics they can be time calibrated as one thereby decreasing the required number of triggers significantly. For detectors with significant drift time and/or sampling of

discriminated detector signals, more elaborate schemes using track reconstruction between detector stations may be needed.

After correct sampling of detector channels, the acquired data must be inserted into the clock synchronous L0 pipeline buffer. The clock used to drive the L0 pipeline will often be common to all channels on a module, to have the data at the output of the L0 pipeline synchronized across the whole module. It will therefore be required to align the sampled data to the L0 pipeline clock. Normally the phase spread between channels can be kept significantly below 25ns and in this case the L0 clock phase alignment can be performed using one of the programmable clock generators in the TTCrx.

4.2. Bunch cycle alignment

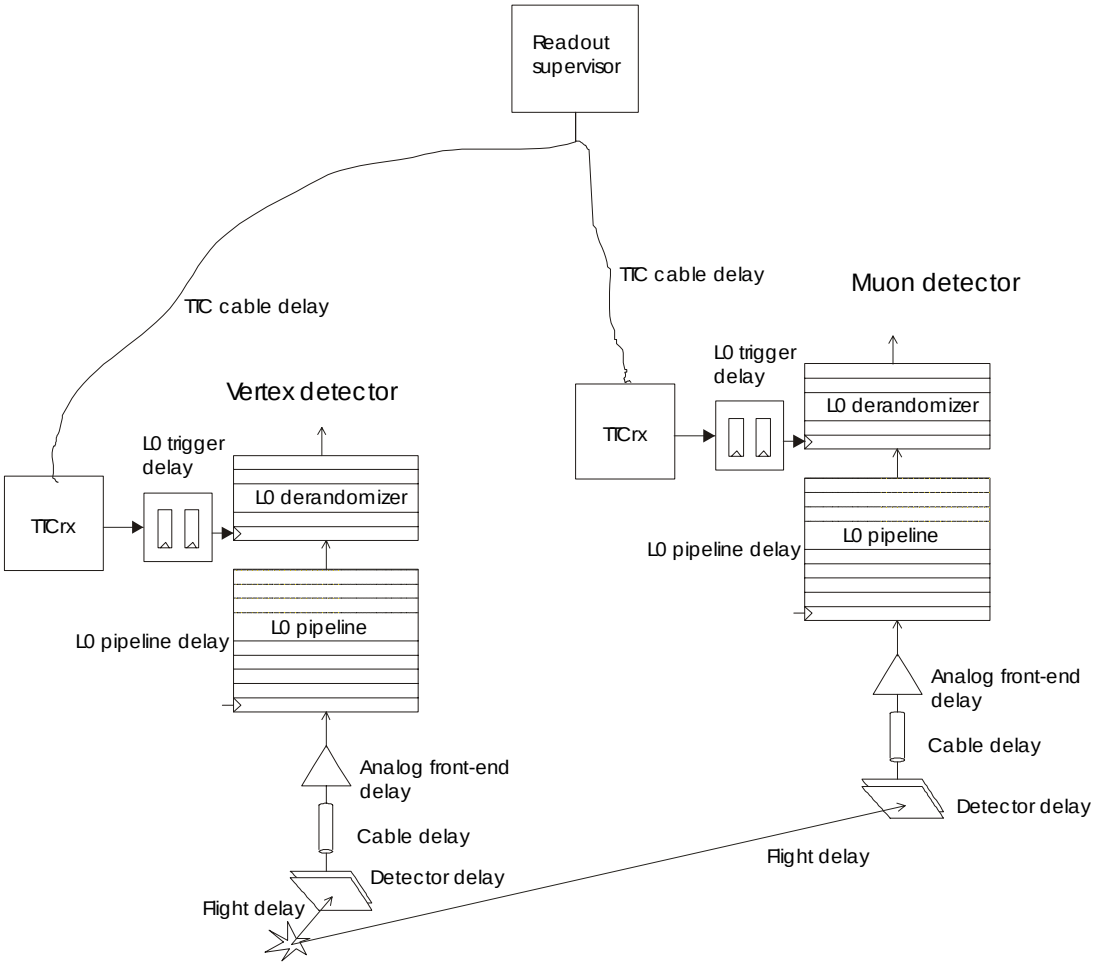

Figure 4. Time alignment of sub-detectors

A coarse alignment, in number of clock cycles, is required to extend the dynamic range of the fine phase alignment. This alignment is performed by the TTCrx, introducing an additional delay of up to 15 clock

cycles on clock synchronous TTC signals (L0 trigger, resets, short broadcast, etc.). The limited range of this alignment requires the complete front-end system to have a maximum skew of 400ns between any channel including: particle flight time, detector response, analog front-end, input to L0 pipeline, TTC distribution and the final extraction of data from L0 buffer (assuming that all L0 pipeline buffers have equivalent depth). The time alignment scenario of LHCb is sketched in Figure 4 with two detectors at extreme positions of the experiment (Vertex and Muon). The condition for a global time alignment is that the two timing paths, starting at the interaction point and ending at the central readout supervisor, are equalized.

Each timing branch is divided into the following delays:

T_{flight}: Flight time from interaction point
$T_{detector}$: Detector delay
$T_{detector\_cable}$ Cable delay from detector
T_{analog}: Delay in analog front-end
$T_{l0\_pipeline}$: L0 pipeline latency
$T_{l0\_pipe\_delay}$: Delay from L0 trigger from TTCrx to data latched in L0 derandomizer
T_{ttcrx}: Delay in TTCrx (programmable)
T_{ttc_cable}: Delay in optical TTC distribution

The condition that the whole experiment can be aligned is that timing branches can be equalized with a T_{ttcrx} regulation range of 400ns. To arrive at a set of well-defined timing restrictions for the front-ends, an extreme configuration between the Vertex and the Muon detector is considered. The worst-case scenario is a Vertex detector with a minimum delay into the L0 pipeline and a maximum delay in the trigger path, together with the Muon detector with a maximum delay into the pipeline and a minimum delay in the trigger path. The L0 latency pipelines are considered to be of equal length in the two detectors (this is actually not a strict requirement as the length of the pipelines in the front-end is assumed to be programmable). The cable length difference of the TTC system is set to be limited to 10m ($T_{ttc\_cable,difference}$ <= 50ns). The difference in delays of the L0 trigger from the TTCrx to the L0 derandomizer is set to be limited to two clock cycles ($T_{l0\_pipe\_delay,\ difference}$ <= 50ns). The flight time difference is limited by the size of the LHCb detector ($T_{flight,\ difference}$ <= 100ns). Under this worst case scenario, the muon detector (or others) will be restricted to have a maximum delay from the detector to the input of the L0 pipeline of:
400ns – 50ns – 50ns – 100ns = 200ns ($T_{detector}$ + $T_{detector\_cable}$+T_{analog}) which seems realistic.
Otherwise the L0 pipeline depth must be reduced locally in the muon detector, or an additional delay must be added to the L0 trigger signal (longer TTC cables, delay on front-end boards, etc.). If the L0 pipeline depth is modified locally, to obtain time alignment, it must though be considered that this will have side effects on the alignment of a calibration pulse injection and its related trigger.

The basic principle of the clock cycle alignment is to assign a bunch crossing ID to all data before entering the L0 pipeline. This in principle allows all data in following processing stages to be aligned based on the bunch crossing ID. The fact that the L0 pipeline buffer is specified to be of constant latency makes this large overhead in the L0 buffer unnecessary and enables the bunch ID to be attached to event data after the L0 pipeline buffer. **All clock synchronous signals from the TTC system are therefore generated under the assumption that the bunch crossing identification is generated at the output of the L0 pipeline buffer**.

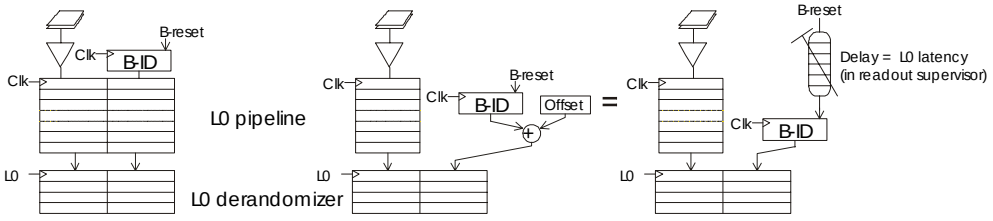

Figure 5. Bunch ID assignment before and after L0 pipeline.

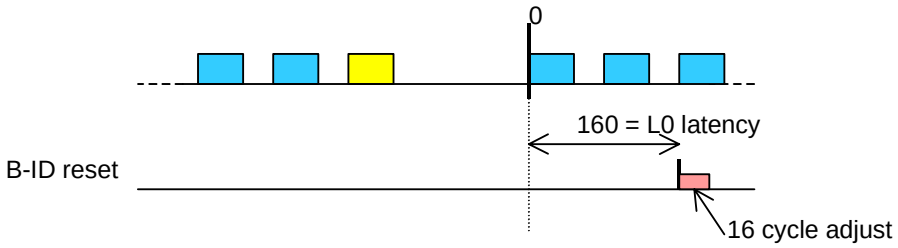

Figure 6. Timing of bunch ID reset

For detectors supplying data to the L0 trigger system, the bunch cycle assignment may need to be performed before data is transferred to the trigger system. Trigger data from different sources can only be correctly correlated to each other, to detect events of interest, if it is known that the data being analyzed originates from the same bunch crossing. To perform correct bunch crossing identification before the L0 pipeline the assumptions used in the TTC system must be compensated for. This can be done by loading a predefined offset into the bunch ID counter, when the B-ID reset signal is asserted, as shown in Figure 7. In addition it must be insured that the bunch-ID counter in this case overflows to zero at the end of a LHC machine cycle (3563 -> 0). Alternatively the L0 trigger systems can use a Bunch ID with a known offset. To insure that the Bunch ID counters are correct from the beginning of the LHC machine cycle, when making a “cold” start of the system, the L0 reset signal (see later) can be used as a reset.

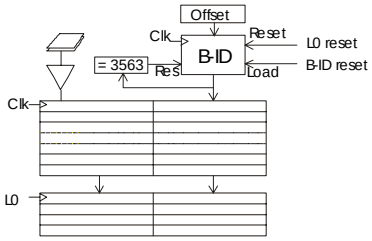

Figure 7. Bunch ID assignment at input of L0 pipeline.

The correct bunch cycle alignment must be obtained during special timing calibration runs. The bunch cycle alignment can in principle be obtained for any detector channel, with no significant drift time, by generating a large set of triggers and histogram the number of hits in each bunch cycle (3564 entries). By correlating this histogram to the known bunch structure of the LHC machine (with bunch gaps), a bunch count offset can be calculated. Such an alignment scheme will be rather slow especially for detector channels with low occupancies. The correct bunch cycle alignment parameters should under normal conditions be known within a few clock cycles (delays between different detector parts are limited). This can increase the speed of the alignment procedure, by using the fact that the LHC machine cycle consists of identical batches of bunch collisions.

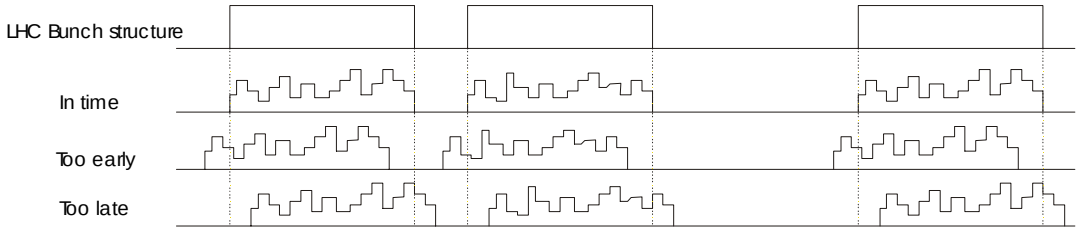

Figure 8. Bunch cycle alignment using hit histograms.

4.3. Calibration pulse injection

The final adjustment of the bunch crossing synchronization can only be performed when the LHC machine has real colliding beams. It is a significant advantage if all sub-detectors can perform a majority of the synchronization procedures without having colliding beams. This will significantly speed up the alignment procedure, when the LHC machine finally has beam collisions, and will also enable extensive system tests to be made without beam. This kind of no-beam alignment requires a scheme to inject calibration signals into the analog front-ends or into the detectors themselves (laser or alike). The injection of such a calibration signal for time alignment must be distributed precisely to all front-end systems, with known and stable delays. This could be obtained using a dedicated distribution system with well-characterized delays. A better alternative is to use the existing TTC system. A dedicated TTC broadcast code is reserved for the distribution of a calibration signal. The timing of such a calibration signal must be variable locally in the front-ends, over a sufficient dynamic range, to insure that the front-end can be aligned with an “emulation” of realistic delay variations in the real experiment.

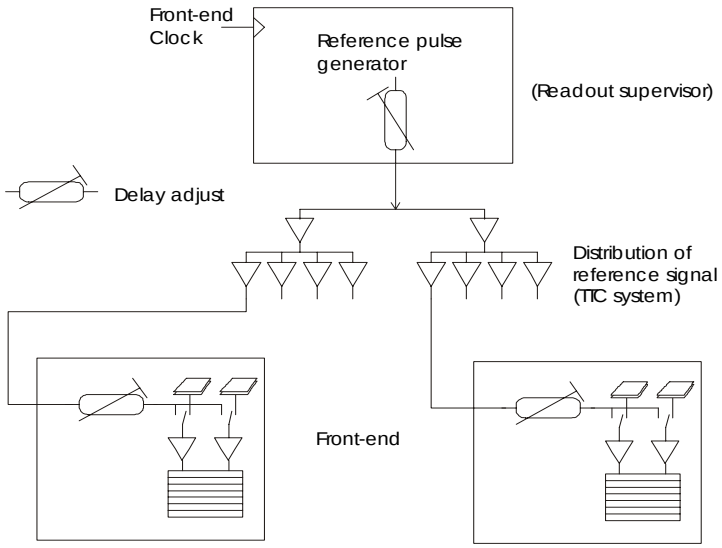

Figure 9. No-beam synchronization using reference timing signal.

The injection of such a calibration signal can also be useful during normal running. A reference signal, and a corresponding set of L0 and L1 triggers, can be generated at regular intervals to monitor the correct function of the complete front-end system (and possibly also trigger systems). When used at the system level, running with all sub-detectors (or a subset), the timing relationship between the calibration signal broadcast and the related L0 trigger accept, both generated by the readout supervisor, can not be adapted to the individual needs of each sub-detector (only one central readout supervisor). In this case the readout supervisor must assume a fixed delay between a calibration pulse broadcast and the L0 trigger accept. Up to four different calibration pulse types are supported by the calibration pulse injection broadcast command. This in principle allows up to four different calibration broadcasts with different timings, before the related L0 trigger accept. The minimum spacing between two calibration broadcasts is 16 clock cycles determined by the TTC serialization of commands. The front-ends must be capable of adapting to this by having programmable delays on the injection of the calibration signal (minimum range: 0 – 16x25ns). Because the short broadcast command is also used for distributing resets to the front-end, the calibration pulse injection commands cannot be generated in a time window of 16 clock cycles after a Bunch ID reset as indicated in the figure below. The default calibration signal type (type = 0) is made with a fixed timing of 16 + 160 clock cycles before the related trigger, which must then be time aligned by programmable delays in the

front-ends. The additional 16 clock cycle spacing between the calibration signal and its trigger has been added to allow additional delays of the calibration signal in the front-end. The use of other calibration types (1,2,3: which can have different timings) can only be allowed after a specific request to the front-end electronics and TFC coordinators. Normal triggers (not related to the calibration) will also be disabled in a programmable time window, to prevent collecting normal physics events with possible contributions from the calibration pulse injection.

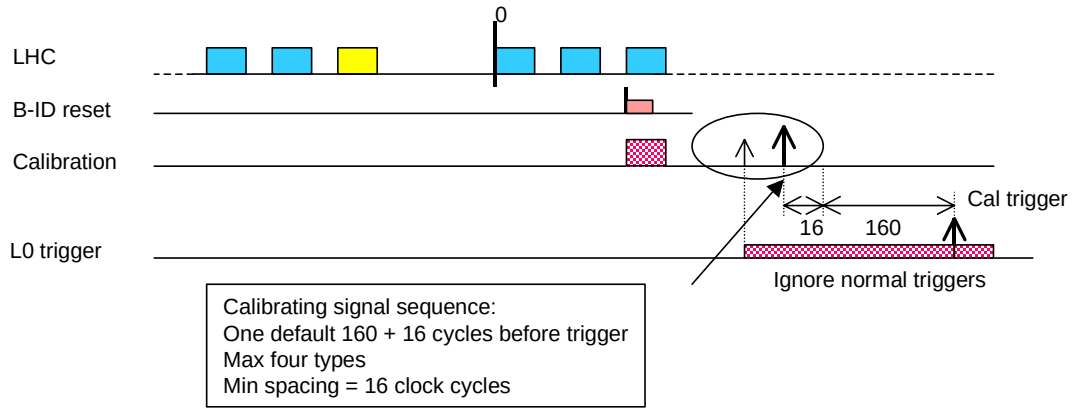

Figure 10. Calibration pulse injection command timing

5. L0 trigger rate

An average trigger level 0 accept rate of 1MHz has been chosen as a compromise between the physics performance of the experiment and the cost and complexity of the front-end electronics. The actual L0 trigger rate will be monitored by the readout supervisor, restricting the L0 trigger accepts to prevent overflows in data buffers in the front-end, and prevent overloading the L1 trigger system. The absolute maximum L0 trigger rate acceptable by the front-end system is determined by the readout time of the L0 derandomizer buffer (1.11 MHz).

6. L0 latency and L0 buffer

The level 0 trigger latency has been set to 4.0 us as a compromise between the cost and complexity of the front-end pipeline buffers and the processing time (plus signal collection and distribution) available for the L0 trigger system [2]. The L0 latency is defined as the maximum delay from the actual bunch crossings to the L0 trigger decision arrives to the output of the L0 pipeline buffer in any sub-detector. This corresponds to a L0 pipeline buffer depth of 160, not including the L0 derandomizer buffer. The actual latency of the L0 trigger decision, when arriving to a front-end module via the TTC optical network, will have some variations. This will be corrected for by a programmable delay in the TTCrx of up to 15 clock cycles and a common programmable delay in the readout supervisor (see also Bunch crossing identification and Use of TTC system). The Latency of the L0 pipelines in the front-end must be programmable over a reasonable range (2.0us – 4.0us) to allow global and local corrections if needed.

It is in the general front-end architecture assumed that the L0 latency buffer is made as a clocked pipeline. It may in certain cases be necessary to deviate from this architecture and only store zero-suppressed data. In

this case, the buffering requirements depend on the occupancy of the individual detector channels. Channel occupancies will have large statistical variations over time and is in most cases not well determined before the experiment is started, because of significant uncertainties in background rates. It must be guaranteed that the L0 buffer can work correctly with significant safety factors on the channel occupancies. The loss of event data on a few channels for isolated events can in most cases be tolerated. It must though under no condition happen that the front-end looses clock or event synchronization, caused by an unexpected high occupancy.

7. Consecutive L0 triggers

The introduction of "dead" bunch crossing periods, after each L0 accept, could potentially lead to simplifications of the front-end electronics of some sub-detectors. The use of an enforced gap between L0 triggers will though result in a potential physics loss of 25ns/1000ns x 3564/2622= ~3.4 %. An investigation covering all sub-detectors has shown that they are capable of working without any L0 gap. Consecutive triggers are also very useful for calibration, monitoring and testing purposes, where a sequence of L0 triggers accepts can be generated to get an extended "picture" of detector signals.

The absolute maximum number of consecutive L0 triggers that can be accepted is determined by the size of the L0 derandomizer buffer.

8. L0 trigger types

It could be of interest to define different kinds of L0 triggers (Normal, Random, Bunch gap, Calibration, etc.) to let the front-end respond differently to different kinds of triggers. The problem doing this is that the TTC system does not directly support the distribution of a trigger type for each L0 trigger accept. The cause of triggers can be multiple, but this information cannot be signaled to the L0 part of the front-end electronics. It has been determined that there is no specific need for the L0 front-end to know the cause of a trigger, as it is not expected to react differently to different types of triggers during normal running. It is sufficient to distribute the trigger type information to the front-end, together with the L1 trigger decision (not L0) to enable the zero-suppression processing in the front-end to adapt its function to the acquired event data (normal zero-suppression, pedestal and threshold monitoring, no zero-suppression, etc.).

9. Samples per L0 trigger

In the baseline configuration of the front-end system it is assumed that only one sample per channel needs to be extracted from the L0 pipeline buffer for each L0 trigger accept. If multiple samples need to be extracted from the L0 buffer, it must not introduce any dead time (related to consecutive triggers) and the effective read-out bandwidth of the L0 derandomizer must be increased proportionally.

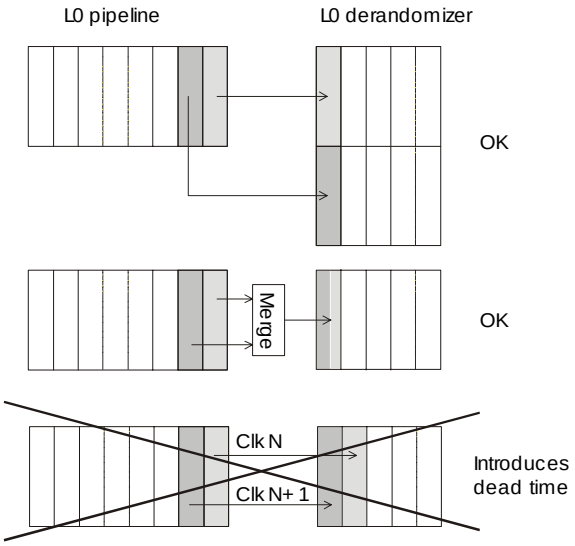

Figure 11. Allowed and not allowed ways to have multiple samples per trigger

10. L0 derandomizer

At the reception of a L0 trigger accept, the data stored in the corresponding L0 pipeline buffer location must be transferred into the L0 derandomizer buffer. A sufficient amount of data per detector channel must be transferred into the derandomizer, to enable later stages of the front-end electronics or the DAQ to correctly identify the value and origin of the signal. The derandomizer must have sufficient depth and be read out sufficiently fast, to keep up with the L0 trigger accept rate. The L0 derandomizer buffer will in some implementations of the front-end be a part of the same physical memory as the L0 pipeline buffer. In this case the effective depth of this memory must equal the L0 pipeline length plus the depth of the L0 derandomizer.

For a derandomizer buffer to be efficient, it must have sufficient depth and be read out at a higher speed than it is written. A high readout speed though have high system costs related to the large amount of data to be transported on a large number of links from the derandomizer buffer, typically located in the detector, to the L1 electronics, located in crates in the cavern or the counting room. A reasonable readout speed has been found to be 10-20% faster than the average input rate [5]. This gives a maximum readout speed of 800 - 900ns per event for a L0 accept rate of 1MHz. An 800ns readout speed is convenient since it allows data from 32 channels to be multiplexed at 40 MHz. Two clock cycles are added to make space for two words of header data in the data stream. An additional overhead of two clock periods per event has been requested to simplify the implementation of certain front-ends. This makes up a **maximum allowable readout speed of 900ns per event**. With this readout speed **the depth of the derandomizer must be bigger or equal to 16 events** to get an effective dead time below 1% at a 1MHz trigger rate.

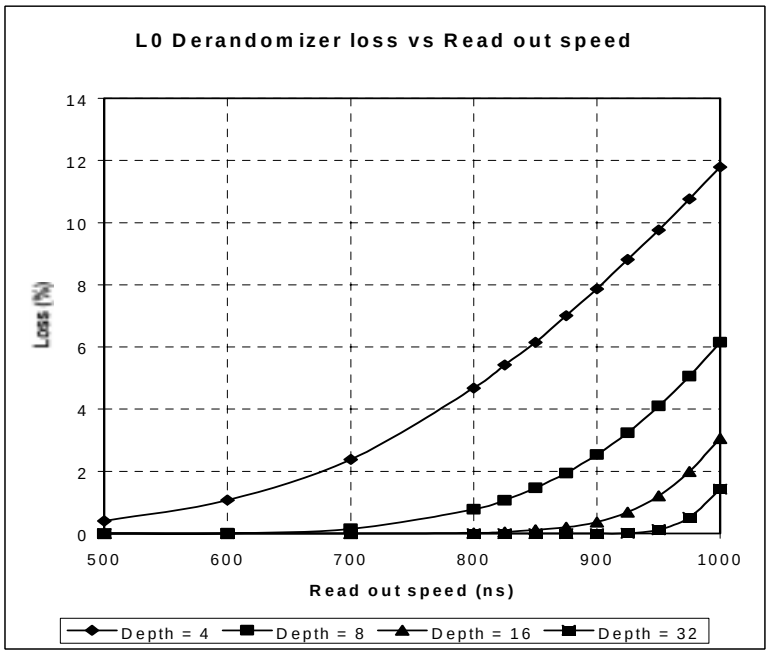

Figure 12. Event loss as function of derandomizer depth and readout time.

The input to the L0 derandomizer buffers needs to be perfectly synchronized across the whole experiment to correctly extract data related to a L0 trigger. The output of the derandomizers does in principle not need to be perfectly synchronized across the experiment. When a new event has been loaded into an empty derandomizer buffer, some implementations (digital buffers) will be capable of starting the readout of the derandomizer in the following clock cycle. Other implementations (analog buffers) need several clock cycles before the new event can be read out. In some cases the readout of the derandomizer can only occur at predetermined time slots (fixed readout cycles) to stay synchronized with the following stages of the data processing. When several events are stored in the L0 derandomizer it is required that all implementations can read out a new event every 900ns. The only allowed difference between implementations is the time needed to prepare the readout of a new event when the derandomizer has been empty (this time must though not exceed 900ns). The readout supervisor, monitoring the occupancies of the buffers in the front-end, will take this into account and make its decisions based on the worst-case implementation. The readout supervisor can safely handle this by assuming an effective L0 derandomizer depth of 15 instead of 16.

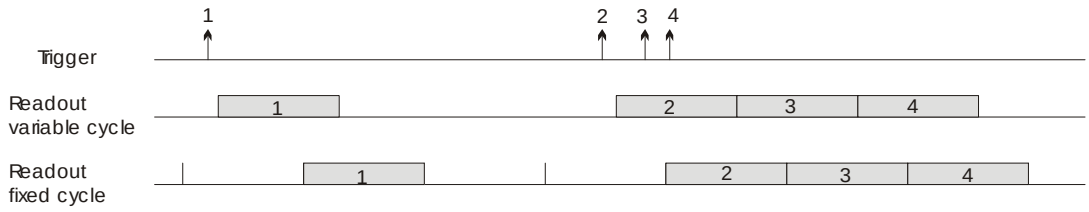

Figure 13. Fixed and variable cycle derandomizer readout schemes.

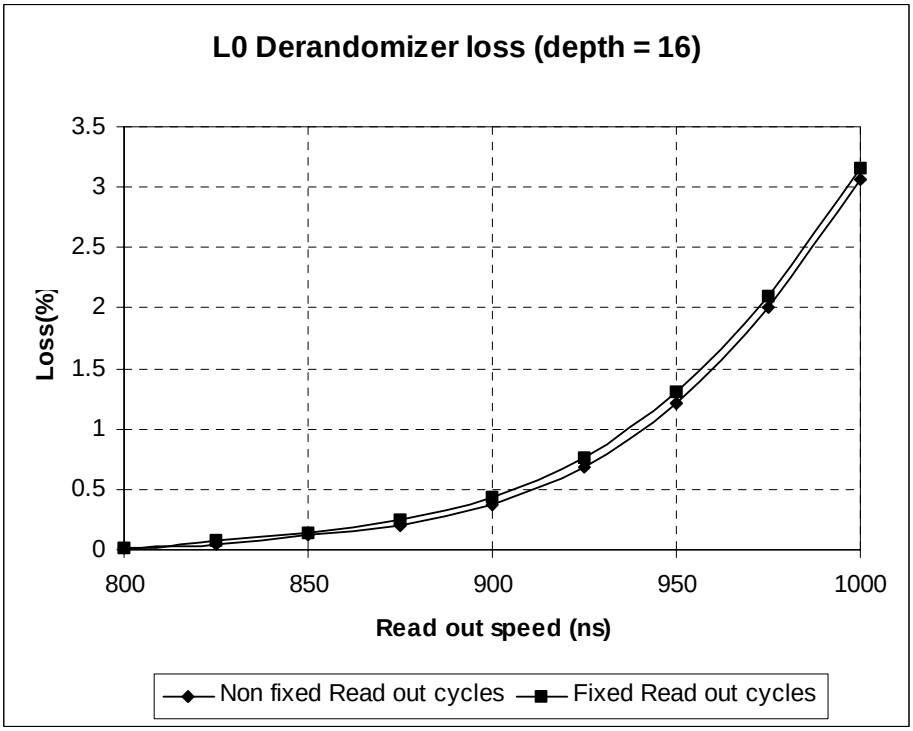

Figure 14. Derandomizer dead time with fixed and non-fixed readout

The optimization of the derandomizer size and its readout time assumes that no data are zero-suppressed. This insures that its function is completely independent of the actual channel occupancy, and simplifies the general system architecture significantly. In case the derandomizer must handle zero-suppressed data, it becomes impossible to prevent derandomizer overflows in a centralized fashion. When handling zero-suppressed data, it must be guaranteed that the event synchronization at the output of the L0 derandomizer is never lost, independently of the actual occupancy. It can though be accepted that event data itself is lost on rare occasions, if properly marked for all event fragments having lost data. To accommodate this in a system with zero-suppression, it will be required to have an extended derandomizer depth and truncate events (only event data, not event separators) when the buffer starts to fill up. Handling zero-suppressed data must be proven to work with extensive simulations of the front-end, assuming very high channel occupancies.

11. Data tagging

During data taking, with the complete LHCb experiment, large amounts of data must be collected from many sources (~1 million) at high speed (40 MHz). Correct processing of all data in the front-end, trigger and DAQ systems relies on the correct synchronization and correct function of a large number of front-end chips (~ 50 K), modules (~ 1500), links (thousands) and processors (thousands). Malfunctions (hard and soft) of components in such a large system must be expected during data taking making it vital to include extensive error checking functions in the system.

A large part of the electronics in the front-end system has to work in a hostile environment with radiation. This radiation will not only limit the lifetime of the components used (total dose effect) but will also

provoke intermittent malfunctions caused by Single Event Upsets (SEU). Single event upsets will occasionally corrupt data being collected for an event. This will in most cases not be a serious problem, as corrupted data can be identified and corrected during detailed event reconstruction. Single event upsets in the control part of the front-end, will on the other hand generate serious problems. If a part of the front-end losses correct clock synchronization or losses an entire event fragment, without being identified, large amounts of event data from the experiment will be corrupted.

To enable the correct function of the L0 front-end electronics to be continuously monitored, all data must be tagged as early as possible with information that allows an effective verification of event data to be performed. Tagging of data at the input to the L0 pipeline will carry a large overhead and will in most cases not obtain an increased error detection capability, as the L0 buffer is assumed to be a simple pipeline buffer with constant latency. Tagging of data must be performed when extracted from the L0 pipeline buffer, by a L0 trigger accept, or at the latest when being read out of the L0 derandomizer buffer. The readout speed of the L0 derandomizer has been specified such that data from 32 detector channels can have up to four data words for tagging with a minimum hardware overhead. The data tagging used must carry information that enables most failures in the front-end to be detected with simple means. At later stages of the data processing, the data tags of each event fragment must be compared against known reference values and/or data tags from the same event, but from a different source. Appropriate information to use as data tags has been found to be the Bunch crossing identification, error status bits and when possible the L0 event number. The use of the bunch ID enables the correct synchronization of the front-end to be verified to a large extent. Error flags can be used to signal detected error conditions to the next level of data processing. The event ID enables to detect the loss of triggers or event fragments. The data tags can also be considered as event headers and/or trailers.

All data tags read from the L0 derandomizer buffer does not necessarily need to be written into the level 1 buffer, as local error checking is assumed to be performed before writing data into the L1 buffer. The maximum number of words to write into the L1 buffer has been limited to 32 data samples and two tags (34 words) to prevent possible saturation effects in the L1 buffer (to be described in more detail in a separate specification of the L1 front-end). The additional two tags from the L0 derandomizer can therefore only be used for “local” verifications of the correct function of the L0 front-end and the data links to the L1 front-end electronics.

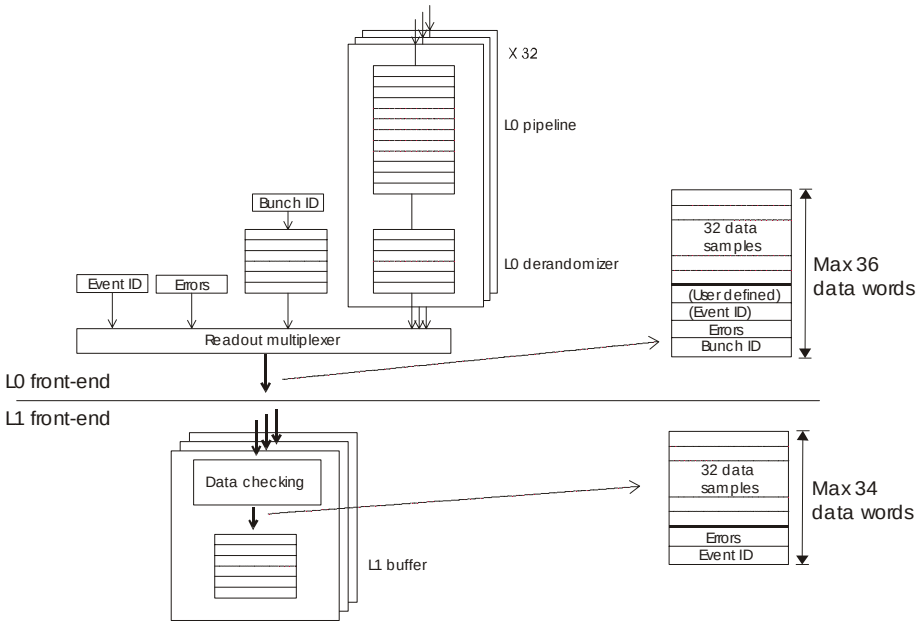

Figure 15. Adding data tags to accepted events.

11.1. Bunch crossing ID tag

The LHC machine cycle consists of 3564 bunch crossing intervals of 25ns. These must be counted with a 12-bit counter, reset by a dedicated bunch count reset signal from the TTC system, to cover a complete LHC machine cycle. Sub-detectors using a data path of less than 12 bits from the L0 derandomizer can choose to use only a limited number of LSB bits of the bunch ID in the data tags, or split it into multiple words.

In some cases it is advantageous to use alternative “representations” of the bunch ID to simplify the front-end implementation and/or improve the error detection capability. The use of the address, where the event was stored in a combined L0 pipeline/derandomizer buffer, gives an additional effective verification of the correct function of the buffer control logic.

11.2. Error status tag

Error status bits (buffer overflow, synchronization error, corruption of setup registers, etc.) are useful for following stages of the data handling to determine immediately if received data can be considered valid. Any detected error in the front-end should also be visible to the ECS system, via local status registers to be capable of tracing error sources.

11.3. L0 Event ID tag

The L0 event ID is a count of L0 accepts since the issue of the L0 Event count reset signal from the TTC system. The event count reset is not guaranteed to occur at predefined intervals and the front-end (and DAQ) system must allow the event count to rollover by itself and “restart” counting from 0. The number of bits from the event counter used for data tagging can, as the bunch ID, be scaled to the width of the data path by using the least significant bits (or split into multiple words).

In some cases, it is difficult to have more than two data tags on event data from the L0 derandomizer. In this case it can be accepted that the L0 event ID is not used in limited parts of the L0 front-end. The loss of a trigger or an event fragment can in most cases be detected directly from the Bunch ID tag, as it is unlikely that two consecutive events have identical bunch ID tags. Just a few LSB bits of the L0 event ID are in most cases sufficient to detect a problem in the event sequence. It can therefore in many cases be merged with the error status bits into a single word.

12. Resetting front-end

When the L0 front-end electronics systems have been correctly set up, it must be forced into a correct synchronization state by a set of reset signals, before real data acquisition across the whole experiment can start. It is assumed that all required clock phase alignments have been performed via local clock phase parameters downloaded from ECS. Such real time resets must be distributed to all front-end systems in a highly accurate clock synchronous fashion, to insure a correct system wide synchronization. A short broadcast message has been defined in the TTC system for the distribution of this kind of time critical clock synchronous control signals (see use of TTC system).

12.1. Bunch crossing ID reset

During normal running, it is required to keep the front-end system synchronized to the LHC machine. The synchronization of many parts of the front-end system, and the verification of correct synchronization, is based on the correct assignment of bunch crossing ID's to acquired data (see data tagging). To insure a continuously correct bunch crossing identification in all front-end systems, and also support any given LHC machine cycle period, a specific bunch crossing ID reset signal is distributed to all front-ends, to signal the beginning of each LHC machine cycle (see use of TTC system). The B-ID reset generated by the TTCrx will have a timing as indicated in Figure 16, as the B-ID counters in the front-end are related to event at the output of the L0 pipeline.

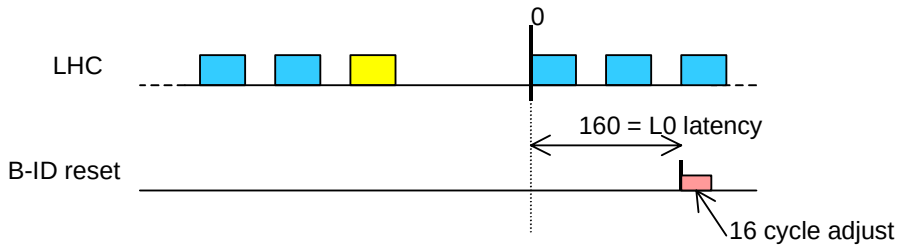

Figure 16. Timing of B-ID reset as generated by TTCrx in front-ends.

12.2. L0 event count reset

The L0 event count identification, also used for data tagging, must be reset before a new run can start. There is though no specific requirement that it is reset at specified intervals, as it can be allowed to overflow on its own and restart counting from zero. The L0 event counter is only incremented for each L0 trigger accept and does not have stringent requirements to its synchronous distribution to the whole front-end system. It is also distributed by the TTC system (see use of TTC system). For simplicity the event count reset will only be asserted together with the B-ID reset (but not for each B-ID reset).

12.3. Global L0 reset

State machines, buffers, error flags, etc. must be reset by a global clock synchronous reset signal before data taking can start. Such a global reset, in the ideal case, only needs to be asserted after a reconfiguration of the system (after power on, after change of front-end parameters, etc.). In practice this kind of global reset will most likely be asserted at regular intervals, to insure that correct synchronization is maintained. In case of detected synchronization errors in the system, the only way to recover synchronization is to assert the global reset sequence (global reset + new sequence of bunch resets).

In a large and complicated system, such as the front-end system of an LHC experiment, working in a hostile environment with radiation, it can be feared that system failures will occur frequently. Above a certain error rate it becomes inefficient to reset the front-end when an error has been detected, because the response time of the ECS system is expected to be relatively slow. In case of high error rates, the best approach is to periodically reset the front-end. In this case it becomes vital that such a global reset can be performed quickly and data taking can restart immediately. When a global reset of the front-end system does not resolve an error condition, a complete re-initialization of a part of the front-end system must be performed

by the ECS system, and special hardware test routines must potentially be executed to determine if the system needs repair.

It could in principle be of interest to define a set of reset signals to reset independent parts of the front-end. The idea behind this would be to only reset a limited part of the front-end, when a specific kind of error has been detected. This could be applied in a fashion where only one sub-detector (or even a part of a sub-detector) is reset and the others are left running. It could also be envisaged to only reset specific functional blocks of the front-end across detectors (e.g. The L0 pipeline but not the L0 derandomizer). The potential advantage of this would be that the number of events lost by a reset could be decreased. In practice both schemes will be very difficult (impossible) to get to work. If only one sub-detector is reset, it will be hard to get event data from the different sub-detectors correctly aligned in the DAQ system. Only resetting the L0 pipeline, and not the L0 derandomizer, will in most hardware implementations be impossible, as the L0 pipeline and the L0 derandomizer are sharing the same physical memory. In general it will also be unsafe to only reset a limited part of the front-end, when a specific error has been seen. In many cases, an error condition in a part of the front-end will result in erroneous data having been passed to the following stages, potentially provoking additional malfunctions. A detected error in a part of the front-end could also have been caused by an undetected malfunction in previous stages of the system. The reset of individual functional blocks in the L0 front-end will not be considered at the global system level. Specific reset commands needed during special calibration or testing runs in individual sub-detectors can be mapped into special user defined commands in the TTC system (see use of TTC). For the total L0 front-end system only one global reset is defined.

It has been decided to have separate resets for the L0 and the L1 front-end electronics. In case of a “cold” start of the experiment or serious system problems, both of these resets will be issued simultaneously in one TTC broadcast command. In case the L0 reset is issued alone, the readout supervisor will insure that all L0 derandomizers are empty when issued. This prevents the L1 electronics from receiving event fragments truncated by a L0 reset during their transmission.

It is advantageous if the front-end system can start to accept new event data immediately after a global L0 reset. It is though also known that such a global reset will enforce the loss of data stored in the L0 pipeline and the L0 derandomizer (if not empty). In addition, when the L0 reset is asserted, the L1 will most likely also be used to reset data buffers in the L1 front-end to insure a completely clean front-end system after a reset, implying an additional loss of events. A certain dead time after a L0 reset can therefore be considered acceptable. The most convenient time to start data acquisition after a reset will be the start of the first coming LHC machine cycle. The readout supervisor will enforce a set of rules to the L0 reset in relation to the LHC machine cycle:

1. No L0 trigger accepts will be generated in a time window of 16x900ns = 14.4us preceding the L0 reset. This will insure that the L0 derandomizer is empty when the L0 reset is generated and will prevent truncating events being sent to the L1 electronics. This preconditioning will only be performed when a L0 reset is issued without the L1 reset. If both the L0 reset and the L1 reset are asserted (as it will be at startup or after a serious front-end malfunction) no preconditioning of L0 triggers will be needed.

2. The L0 reset will be generated from the TTCrx at the beginning of the LHC machine cycle. This reduces the loss of data in the L0 pipeline, because of the large bunch gap in the end of the LHC machine cycle. In addition it also makes it useful for resetting B-ID counters related to the input of the L0 pipeline (as previously described).

3. After the L0 reset is generated, it is acceptable with a reasonable dead time (e.g. less that 8 clock cycles) before the L0 pipeline starts to store correct detector data.

4. The first L0 trigger accept, after a L0 reset, will only be generated after the L0 latency (plus some dead time). The readout supervisor will enforce L0 triggers decisions to be rejects before this

point. (This allows additionally ~160 clock cycles to initialized the L0 derandomizer and its related readout logic).

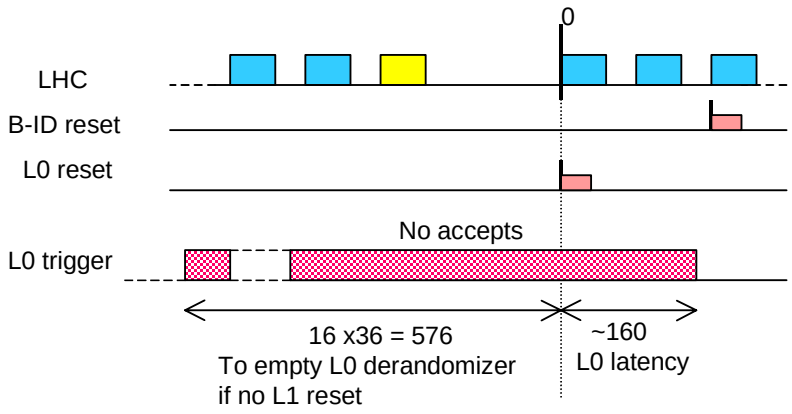

Figure 17. Timing of L0 reset from TTCrx

13. Error checking and testing

A significant design effort must be invested in error checking/verification and testing features in the front-end system. Large parts of the front-end system must work in a hostile environment with radiation and magnetic fields. In addition, access to all electronics inside the detector or in the cavern will be strongly limited. While the front-end system is actively running, it must be capable of detecting malfunctions on-line. Extensive testing facilities must also be built into the hardware (and related software), to enable remote fault diagnosis while modules are located in the system.

13.1. On-line checking and reliability

As previously mentioned, all data accepted by the L0 trigger must carry a set of data tags to enable following stages of the front-end/DAQ system to verify if event fragments have been properly processed in previous stages. Any detected error condition must be signaled to the ECS system. Any event fragment, which could be erroneous because of this, must be flagged as being error prone to enable the DAQ system to take this into account in its event processing. If a small part of a sub-detector has detected an error, it may not be considered serious enough to reinitialize (reset) the whole front-end system and the remaining system must be capable to continue working correctly.

The readout supervisor is in principle responsible for preventing any buffers in the front-end to overflow. Buffer overflows may though still occur because of malfunctions. The front-end logic must therefore continuously verify the occupancies of local buffers and in case of an overflow detect this as an error condition.

The behavior of the different front-end systems is to a large extent depending on a large set of front-end parameters, which must be downloaded before a run. If any of these parameters have been corrupted during their downloading, or have been modified by a single event upset in the register/memory storing its value, the behavior of the front-end module will change. Significant front-end failures will normally be detected

by missing or wrongly tagged event fragments. Less significant changes may though remain unnoticed for extended time periods. The use of a parity bit per parameter register or a common parity for a whole set of parameters can detect this kind of failure immediately (parity check must be performed continuously). In certain critical cases it will be required to use error-correcting codes (e.g. Hamming coding) to be immune to single bit errors

Flip-flops used to implement state machines, controlling the different functions of the front-end electronics, are also sensitive to single event upsets. When state machines are implemented with normal binary state encoding, it is very hard (impossible) to determine if the state machine has gone wild. If one-hot encoded state machines are used (state encoding where one and only one state flip-flop is set at any time) it is possible to detect any single bit upset of the state register. One-hot encoded state machines need more flip-flops to implement the state register (e.g. 8 instead of 3) but the overhead is in general quite small as the state encoding/decoding logic is reduced when using one-hot encoding. The state check consists simply of verifying that one and only one bit is set in the state register in any clock cycle. One-hot encoded state machines are normally supported by most logic synthesis tools, but the state checking must be "hand" coded in the input (VHDL or Verilog) to the synthesis tool. Self-correcting state machines using Hamming state encoding can also be envisaged.

Counters are a special type of state machines where one-hot encoding cannot be used in practice. In this case special precautions must be taken to be capable of detecting errors. The value of important counters (bunch ID, Event ID) can be added as tags to data being read out, enabling their value to be verified in following stages of the data processing.

Address pointers are extensively used in the control of the L0 pipeline and the L0 derandomizer buffers. In many implementations address pointers are also stored in internal memories (FIFO’s) waiting for data to be read out. Upsets in the address pointers or the memories storing them will in most cases imply a serious malfunction of the circuit. Faults in internal memories can be detected straightforward using a parity check. Faults in the address pointer control can in many cases be detected by adding the physical address pointer as a tag to the data read out.

The use of SRAM based FPGA’s pose a specific problem in a radiation environment. The configuration data stored in volatile memory in the FPGA can change on-line by a SEU. In most cases this will probably cause system failures detectable immediately. It can though in some cases happen that minor failures remains undetected for extended periods. It is as a minimum required that the content of the configuration data can be read back to determine if it has been corrupted.

The rate of SEU induced malfunctions must be estimated and it must be shown to be compatible with a fully working experiment over reasonable time periods (hours, days). In certain cases, it will be required to use special SEU mitigation techniques, like triple voting or Hamming error correction/detection, to obtain a reasonable reliability of the system.

13.2. Testing

If the front-end system has been found to malfunction repeatedly it must be possibly to perform extensive tests of the front-end hardware to determine the failing part of the system. If the failing part can be precisely identified remotely, it can be exchanged quickly during the first access to the detector. It must be possible to continue running with the system, by isolating the failing part of the system and only loose data from a limited part of a sub-detector.

To perform extensive hardware tests, it must be possible to access key parts of a module via a data path independent from the normal readout data path. This separate access path will normally be the ECS interface, also used to download setup parameters. To perform any efficient module testing it is required

that all registers have a read-back feature to be capable of verifying that data has in fact been correctly loaded (also for look-up tables and FPGA configuration data). If the readout path from the L0 electronics is suspected to be malfunctioning, it is advantageous to be capable of reading event data via the ECS data path. This only needs to work in a single-shot mode, where new triggers only are generated when the previous event has been read out completely.

Extensive hardware tests of modules can be performed if they have been designed to support JTAG boundary scan testing (IEEE 1149 standard). Having implemented this kind of test features on a module allows its functions to be tested in-situ down to a very detailed level. Interconnections between components on the PCB can be verified, and in many cases the functions of individual components can also be verified. The use of boundary scan testing is in many cases required anyway, to be capable of performing sufficient tests of modules after their production. The high integration level of modern electronic modules, using high pin count devices in small packages (surface mount, BGA packages, etc.), makes it impossible in most cases to perform production testing using external probes (in-circuit testing). Most integrated circuits today (processors, FPGA's, etc.) supports JTAG boundary scan testing. When implemented, it must be insured that the JTAG test facilities can be accessed while the module is located in the final system (interface to ECS required).

All front-end systems are expected to include extensive testing and debugging facilities to verify its correct function being a part of a complete front-end system. The generation of specific event data must be supported, by loading predefined patterns into the L0 pipeline buffer, and/or injecting special test pulses into selected parts of the analog front-end.

Sufficient testing will require a hierarchy (component, module, crate, sub-system, system, etc.) of test procedures that can be performed in-situ (while modules located in the system). At the module level it is advantageous if extensive self-tests can be performed when a simple "self test" command is issued from ECS (this is the case for most VXI data acquisition modules). This will though require local intelligence (processor) to run trough test procedures. If a module relies on external intelligence (ECS and/or DAQ) for hardware tests, the required performance from this must be defined. The required software to run the hardware test of a module must be considered an integral part of the development of a given module. At the sub-system level, test procedures for interconnects (busses, links) must also be defined and implemented.

13.3. Trigger system verification

To monitor the correct function of the L0 trigger system, the front-end electronics must read out enough data to allow a complete reconstruction of the data used by the trigger. This will enable software routines in the DAQ system (and off-line) to emulate the function of the trigger system and determine if correct trigger decisions have been taken. The verification of trigger decisions will be performed on a small sub-set of events to keep track of the performance of the trigger system. To emulate the function of the trigger systems with high precision, it will be required that all trigger sub-systems are 100% digital. The detailed function of analog circuits depends on noise levels and is not 100% predictable. Random triggers will be used during normal running to get a representative sample of events to verify the correct function of the trigger system.

To insure that all data used by the L0 trigger systems are available to the trigger monitoring routines, it may in some cases be required that the L0 trigger data is not zero-suppressed in the L1 electronics. This can be the case for all events or alternatively for a small set of events only, pre-selected by the readout supervisor to be used for the L0 trigger system monitoring. (This will be described in more details in the specifications of the L1 front-end electronics)

14. Use of TTC system

The TTC system [3] is used as the general distribution system for the bunch crossing clock and associated time critical clock synchronous control signals to the front-end. It is assumed that all front-end systems are driven by a clock and a set of control signals, as available from the TTCrx receiver chip (either directly or indirectly). Detailed use of the TTC system in LHCb is described in a set of separate documents [6][7][9].

14.1. Clock

The bunch-crossing clock from the TTC system is distributed with low jitter (< 50ps RMS, < 350ps Peak-peak) and a "constant" but unknown phase to the LHC machine. The TTCRX has a programmable clock phase generator with 100 ps resolution, which allows the output clock to be phase aligned with the detector signals based on a timing calibration. All clock synchronous signals from the TTCRX will be generated with this clock as its reference. One additional clock phase generator is available in the TTCRX for user-defined purposes.

The phase of the clock generated by the TTCRX, to drive the front-end electronics, has in principle a constant phase to the LHC machine. In practice the clock seen in the front-end systems will have some variations with time caused by temperature/supply-voltage drifts in the system (cables, cable drivers/receivers, TTC laser driver, TTCRX, local clock drivers, etc.). The characteristics of such phase drifts in a large LHC experiment are currently not well determined. It will be required that all front-end systems have means of monitoring that the clock phase has not drifted excessively during normal data taking.

The clock used in the front-end systems may under certain circumstances have glitches caused by radiation effects or electrical noise. The TTCRX may also loose lock in one of its internal phase locked loops or delay locked loops and generate a drifting clock. The use of a bunch ID tag on all accepted event data insures that this kind of malfunction in the front-end can be detected in most cases.

14.2. L0 accept/reject

The L0 trigger decision (called L1 in all other LHC experiments) is transmitted on a separate data channel in the TTC system as a clock synchronous accept/reject signal. Nearly any sequence of L0 triggers can be distributed by the TTC system (Max 24 consecutive accepts). The received L0 trigger is phase aligned with the local clock from the TTCRX fine delay generator and can in addition be delayed by up to 15 clock cycles (same as used for short broadcast commands) to perform the extraction of event data from the correct bunch crossing. For each L0 accept, an internal L0 event ID counter in the TTCRX is incremented.

No error checking or error correction is performed in the TTC system on the distribution of L0 triggers, because of its relatively high bandwidth. It may therefore potentially happen that fake triggers are generated in parts of the front-end system caused by single event upsets. The use of Bunch ID and Event ID data tags in the L0 front-end will enable this kind of problem to be detected.

A multiplexed output bus is available to signal the bunch ID and the event ID of accepted events. To observe both the Bunch ID and the event ID two clock cycles are needed on the multiplexed bus following a L0 accept. As LHCb requires the acceptance of consecutive triggers the multiplexed bunch ID/ event ID scheme cannot be used. The TTCRX has an alternative working mode, where the output bus is used for

bunch ID only or event ID only. It is recommended to use the Bunch ID mode and generate the event ID by external counters (in ASICs, FPGA's, etc.).

14.3. Short broadcast command

The short 8-bit broadcast command is used to distribute several important signals to the front-end. This broadcast has been implemented especially for this kind of purpose and is protected against single bit errors by the use of hamming code error detection/correction. When a broadcast command has been initiated the TTC system cannot accept new broadcasts during the following 16 clock cycles.

The Bunch count reset and the Event count reset are distributed by the TTC system as the two least significant bits of the broadcast command. They are generated as separate reset strobes with programmable delays (strobe delay 1). The timing of the remaining broadcast bits can be defined with two programmable strobes (strobe1 and/or strobe2). To insure correct decoding of received LHCb broadcast commands (L1 triggers, front-end resets, etc.) on bit [7:2] it is required that all broadcast bits use strobe1 (use of strobe2 not allowed).

The remaining 6 bits of the broadcast command is encoded for LHCb specific signals. The L0 reset and the equivalent L1 reset are encoded such that simultaneous L0 and L1 resets can be distributed with one broadcast command. Four different calibration signal types can be transmitted.

Bit mapping:

<table>
<tr><th>Command</th><th>Bit[7]</th><th>Bit[6]</th><th>Bit[5]</th><th>Bit[4]</th><th>Bit[3]</th><th>Bit[2]</th><th>Bit[1]</th><th>Bit[0]</th></tr>
<tr><td>NOOP</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="6">L0 Event ID reset</td><td rowspan="6">Bunch-ID reset</td></tr>
<tr><td>L1 trigger</td><td>1</td><td colspan="3">TYPE:
0: Reject
1: Physics
2 – 7: Reserved for future use</td><td>ID[1]</td><td>ID[0]</td></tr>
<tr><td>Reset</td><td>0</td><td>1</td><td>Reserve</td><td>L1 ID reset</td><td>L1 reset</td><td>L0 reset</td></tr>
<tr><td>CMD1 (calibration)</td><td>0</td><td>0</td><td>0</td><td>1</td><td colspan="2">TYPE:
0: Default
1-3: Reserved for future use</td></tr>
<tr><td>CMD2 (Reserve)</td><td>0</td><td>0</td><td>1</td><td>0</td><td>x</td><td>x</td></tr>
<tr><td>CMD3 (Reserve)</td><td>0</td><td>0</td><td>1</td><td>1</td><td>x</td><td>x</td></tr>
</table>

Complete command decoding of the broadcast commands (bit[7:4]) is required to prevent any interference between different commands in the system.

Reserved commands and bit fields can only be assigned to specific functions after a request to the front-end electronics coordinator has been officially accepted. A continuously update version of TTC commands is available on the web [9]

14.3.1. Bunch count reset

The generation of local bunch identifiers in the front-end is supported by the TTC system having a dedicated bunch count reset bit in the short broadcast command. The TTCRX generates an external bunch count reset signal that can be used by local bunch id counters in the front-end. In addition the TTCRX has its own internal Bunch id counter which value can be available on an output bus for each L0 trigger accept (can alternatively be used for event ID).

The bunch count ID generated in the front-end controlled by the TTC system is related to the bunch ID of events at the output of the L0 pipeline. The bunch ID related to the input of the L0 pipeline is of little interest in most cases, as all L0 buffers in the front-end are of constant latency. The Bunch ID at the input of the L0 buffer can, as previously described, be calculated from this.

14.3.2. L0 event count reset

In addition to the bunch crossing identification of each accepted event the TTCrx can generate a 12/24-bit L0 event ID. The L0 event ID counter is incremented for each L0 trigger accept and is initialized to zero by the event count reset. The event counter is reset by a dedicated L0 event count reset bit in the short broadcast command.

The LHCb requirement of accepting consecutive triggers prevents the use of the internal event-ID counter. The event ID count must be generated externally, using the event count reset and the L0 trigger accept signals from the TTCRX.

14.3.3. Reset of L0 front-end

The L0 reset signal in the reset command must reset the complete L0 front-end system as defined previously.

14.4. Use of individually addressed commands

The TTC system can in principle be used to download parameters to the front-end system via individually addressed commands. This function of the TTC system is though far from ideal as there is no associated read command to verify that data have actually arrived correctly to its destination. Parameters and setup data needed in the front-end must be down loaded via the ECS system and a read-back facility (not using the readout data path!) should always be available. CONCLUSION: DO NOT USE THIS FEATURE OF THE TTC SYSTEM.

The individually addressed commands can also be used as an additional broadcast facility of 16 bit data (8 bit data + 8 bit sub-address) by setting the TTCRX address in the command to zero. The use of this

second type of broadcast can only be allowed after proper permission from the front-end electronics coordinator for reasons already mentioned.

15. Required L0 functions in readout supervisor

The readout supervisor is the main real-time controller of the complete front-end system. It performs a long list of important tasks for the L0 front-end:

- Receive bunch crossing and bunch structure synchronisation signals from the LHC accelerator.
- Distribute bunch-crossing clock to all front-end modules via TTC system.
- Generate real time resets to L0 front-end (Bunch count reset, L0 Event count reset, L0 reset).
- Receive L0 trigger decisions from L0 trigger decision unit.
- Insure correct synchronisation of received L0 triggers.
- Apply restrictions to L0 trigger accepts to prevent L0 derandomizer overflows.
- Apply restrictions to L0 trigger accepts to prevent the L1 trigger system to overflow.
- Distribute L0 trigger decisions to L0 front-end electronics via TTC system.
- Generate resets, calibration signals and trigger sequences for debugging/testing and calibration runs.

Detailed specifications of the readout supervisor exist in a separate document [7].

16. ECS interface

The control, monitoring, and verification of the front-end electronics is performed by the Experiment Control System (ECS) [10]. The global ECS system will communicate with the front-end electronics via the ECS interfaces. The ECS interface to the front-end electronics has been standardized, to insure a consistent and well functioning control and monitoring system.

In the counting room, with no radiation, a credit card PC with an Ethernet interface has been chosen as an appropriate solution. The intelligence of such an interface allows monitoring and verification routines to be run locally, in a standard software environment.

For front-end electronics located in the cavern special solutions, with built-in immunity to radiation induced single events, are needed. For the cavern itself, with limited radiation levels, an ECS interface called SPECS (Serial Protocol for Experiment Control System) has been chosen as an appropriate solution. SPECS is based on a serial protocol running on standard cat5 cables with a maximum distance of 100 meters. A SPECS slave interface, with built-in immunity to single event upsets, will be available as a standard component within LHCb. An alternative solution based on a special SEU immune CAN slave, called the ELMB, is also considered.

For front-end electronics, located inside the detector in a high radiation level environment, special solutions are needed. Limited distance (~10m) simple serial protocols like I2C and JTAG are used to reach SPECS and CAN slave modules in the cavern.

17. Qualification of L0 front-end electronics

The L0 electronics is in general located very close to the detector, either inside the sub-detector itself or on its periphery. Electronics located inside a sub-detector can only be accessed for service or repair during the long shutdown periods of the LHC machine once per year (may in exceptional cases be accessed to perform limited repairs). Electronics located outside the detector can be accessed on a regular basis for simple repairs and service.

If standard commercial components are used in the L0 electronics, it must be located in places with low radiation levels (less than 10 Krad total dose during 10 years) and the system must be verified to work correctly after realistic radiation doses. In locations with higher radiation levels special radiation hard/tolerant components must be used which have been qualified with appropriate qualification procedures. Integrated circuits must be verified to be immune to single event latch-up (can be destructive) in the given application.

The effects of single event upsets on the reliability of the electronics must be evaluated and the detection and recovery from such failures must be considered. The specific problems related to SEU effects on configuration data in FPGA's must be shown to be at an acceptable level. Especially the ECS interface of the front-end electronics must be shown to be immune to lockup states caused by SEU, as this interface is the only path available to recover normal functionality.

Major components of the front-end systems must have passed a series of design reviews before it can be considered ready for production. The review process will consist of up to three specific reviews: A specification review to verify a sufficiently detailed specification, to insure a correct function within the experiment; A prototype review before the submission of final prototypes of major components (e.g. ASIC's and boards); Finally a production readiness review must evaluate if a design can be considered ready for final production.

18. References

[1] LHCb Front-end electronics. http://lhcb-elec.web.cern.ch/lhcb-elec

[2] "The latency of the level 0 trigger", LHCb TRIG 99-15

[3] TTC system. WWW: http://www.cern.ch/TTC/intro.html

[4] TTCrx reference manual, http://www.cern.ch/TTC/TTCrx_manual3.0.pdf

[5] "Simulations of LHCb front-end electronics", LHCb FE 99-047.

[6] "Timing and fast control", LHCb 2001-016 and
http://lhcb-comp.web.cern.ch/lhcb-comp/daq/TFC/html/TFC_intro.html

[7] "The Readout Supervisor design specifications", LHCb DAQ 2001-012 and http://lhcb-comp.web.cern.ch/lhcb-comp/daq/TFC/html/rs.html

[8] Key front-end parameters, http://lhcb-elec.web.cern.ch/lhcb-elec/html/key_parameters.htm

[9] Use of TTC in LHCb, http://lhcb-elec.web.cern.ch/lhcb-elec/html/ttc_use.htm

[10] Experiment Control System: http://lhcb-comp.web.cern.ch/lhcb-comp/ECS/default.htm