MCC-I2.1 Specifications

MCC Design Group

Receiver

Revision:	1.0
Reference:	ATLAS ID-xx
Created:	12 December 2003
Last modified:	1 September 2004 10:58
Edited By:	R. Beccherle and G. Darbo
	ATLAS/Pixel Collaboration

Abstract

This document describes the Receiver of the *MCC-I2.1*.

There are 16 identical copies of the Receiver inside the *MCC*.

Table Of Contents

List of Figures

3 Receiver

This chapter will describe the Receiver block of the *MCC-I2.1*.

The architecture that we have chosen for the read-out of the pixel detector is a two level hierarchical structure that starts from the *FE* chips, and goes to the *MCC*. The interconnections between the FE chips and the MCC are unidirectional links. Being 16 *FE* chips hosted on a module there are 16 identical Receiver blocks inside the *MCC*.

The transmission protocol is "data push": the *FE* chips send the hit coordinates as soon as they are ready in the end-of-column buffers. Since the chosen architecture is data push, the *MCC* cannot stop the *FE* chips from transmitting data.

In RunMode there are two ways for the *MCC* to interact with the data flow generated by the *FE* chips:

1. Buffer input data into ReceiverFifo's:

   The FIFO inside each Receiver is necessary to de randomize the hit data sent by the *FE* and to reduce the data lost due to random fluctuations of the data rate. Since it could always happen that such *FIFO*'s overflow, a Warning mechanism is introduced in the *MCC* EventBuilder to flag truncated events or to re-synchronize the system when needed. The FIFO size as been chosen in such a way that up to 112 Hits can be stored for each Receiver block before an overflow will occur.

2. Block the LV1 trigger to the *FE*'s:

   This happens when more than 16 events are still inside the module: either in the *FE* chips or in the *MCC* EventBuilder. The *MCC* keeps track on how many LV1 were transmitted to the *FE* chips. In case a Trigger command arrives to the *MCC* (having already sent 16 Triggers to the *FE* chips) before it has finished to process at least one complete event, the *MCC* blocks the LV1 signal to all the *FE*'s it controls. As soon as one event has been fully processed and sent to the *ROD,* Trigger issuing is resumed. This mechanism will cause some events to be lost in some detector modules, but will avoid the risk of mixing hits belonging to different events due to lack of synchronization. Each time there is a dropped event, the information is recorded by the *MCC* (up to 16 consecutive dropped Trigger commands) and is propagated to the *ROD* in order to keep up with event synchronization.

   STOP This is done by generating a SkippedLV1 number that is added to the Output Data Stream (in the 4 most significant bits of the LV1 field) of the last event before the skipping of events takes place. For more information on this please refer to Chapter 5.4, "MCC to ROD Event Format"

MCC signals that are active during data acquisition are graphically represented in Figure 3-1:

CK: Input Clock from the *ROD*

CK is a 40 Mhz clock, meant to be synchronous with the *LHC* bunch crossing. It is transmitted using the LVDS electrical standard. This signal will be buffered inside the *MCC* and the buffered signal, called XCK will be sent to all *FE* chips and to the *MCC* itself in order to minimize possible clock skew between different components of the module. The DCI line is first synchronized using this reference clock.

DCI: Data and Command Input.

DCI is used to send either data or commands to the MCC. During RunMode DCI is used only for Trigger and Fast commands, otherwise the *MCC* will exit from RunMode and Trigger and Fast commands will be automatically disabled. The most frequent command received is Trigger which is encoded using 5 bits, allowing the possibility of generating a trigger every fifth bunch crossing

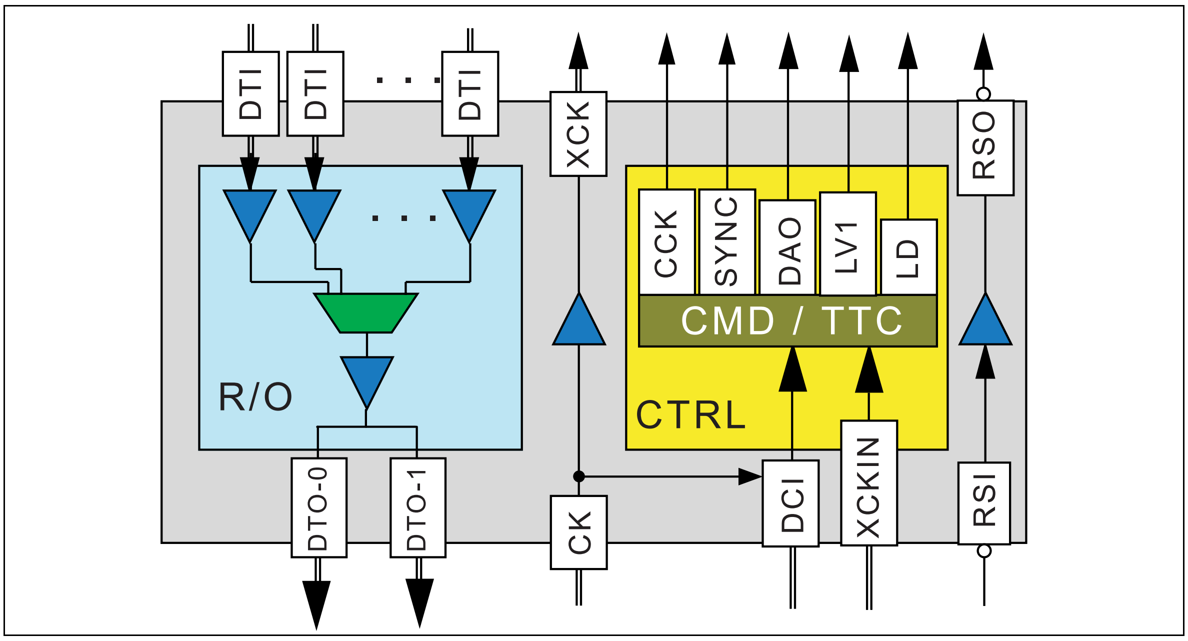

Figure 3-1 *MCC* signal used by the Event Read Out protocol

as required by ATLAS. Other commands are the BCR and ECR resets of the *MCC*, or the CAL or SYNC commands of the *FE* chips.

This signal is synchronized using two stages of latches. The first latch uses CK as its reference clock while the second one uses XCKIN as reference clock. This scheme ensures that the DCI line will be correctly in phase with the signals inside the *MCC* that are all synchronous with XCKIN.

DTI: Data Input from the *FE*.
Each *FE* chip uses a separate input line (MCC-DTI). The transmission from each *FE* is serial, it uses low voltage differential signaling (LVDS), and data are in phase with the clock XCK. The maximum throughput is 40 Mbit/sec using this data/clock protocol since the clock XCK is 40 MHz.

All 16 DTI lines are synchronized using a latch that uses XCKIN as the reference clock.

DTO: Data Output lines to the *ROD*.

The *MCC* puts together different slices of the same event received from the *FE* chips and retransmits with appropriate encoding into the readout chain using the DTO-0 and DTO-1 serial LVDS line. The maximum throughput is 160 Mbit/sec using this data/clock protocol since the XCKIN clock is 40 MHz and the data can be shared on two lines and encoded on both edges of the clock.

Both DTO lines are synchronized using a standard flip-flop using XCKIN as the reference clock.

LV1: Level 1 trigger.

Trigger signal is generated in response to an ‘encoded’ trigger signal received by the *MCC* from the DCI input and decoded by the CommandDecoder. A single Trigger signal generated by the *MCC* lasts one clock cycle and is synchronous with the XCK clock. Triggers can be suppressed in the case that in the whole module there are more than 16 events to send out. Depending on the setting of the LV1 register, up to 16 contiguous Trigger commands can be generated by the *MCC* for each *FE* enabled chip in response to a single Trigger command.

SYNC: *FE* Synchronization and Reset.

The *MCC* has to provide correct reset signals to the *FE*. There are two synchronization signals that the *MCC* can send to the *FE*’s. Both signals are sent to the *FE* chips via the MCC-SYNC pin of the

MCC. The first one is in response to a Sync command that produces a MCC-SYNC signal that is active for five clock cycles. This command is a Fast command and can be issued without taking the *MCC* out of RunMode.

The second reset command that can be sent to the *FE* is a Slow command and is issued in response to a GlobalResetFE command. It generates a MCC-SYNC signal that does a global reset of all the *FE* chips according to the width of the generated MCC-SYNC signal.

For more information on how to issue this commands and on their correct syntax please refer to Chapter 1.2.2.4, "SYNC: SyncFE signal" and Chapter 1.2.3.10, "GlobalResetFE: Reset FE chips".

XCK: Output clock to the *FE* chips and to *MCC* itself.

XCK is generated from the CK clock by the *MCC*. XCK has the same frequency as the CK clock. The XCK signal is routed to all 16 *FE* chips and also to the *MCC* itself that uses this clock as it's main clock. This allows for minimal skew of the clock between all chips on the module.

XCKIN: System clock for the *MCC*.

The XCK output must be externally connected to the XCKIN input. The XCKIN clock is used as master clock for the *MCC*, while the CK clock is only used to synchronize the MCC-DTI signals.

3.1 Building blocks

Event building is performed by two concurrent processes running in the *MCC*. The first one (performed inside the Receiver) deals with the filling of the 16 input FIFO's with data received from the corresponding *FE* chips, while the second one (performed inside the EventBuilder) extracts data from the FIFO's and builds up the events. Each *FE* sends data as soon as they are available with two constraints: event hits must be ordered by event number and for each event an End of Event (EoE) word is always generated. EoE is also sent for the case of an empty event to keep event synchronization.

The block diagram of Figure 3-2 is an expanded view of one of the sixteen Receivers present in the *MCC*. As we can see the Receiver is composed by three main building blocks, the receiving ShiftRegister, the FIFO and the ControlStateMachine that deals with Warning conditions and FIFO pointer management.

3.1.1 Input Shift Register

FE's send data serially to the *MCC*. The input shift register is 26 bit wide and continuously reads data from the MCC-DTI line. As soon as the 26-th bit is '1', which means that the *FE* has transmitted a complete Hit or EoE word, the shift register is blocked for one clock cycle and a signal is sent to the control state machine which eventually copies the data in the FIFO.

⚠ Data coming from the *FE* while the shift register is blocked is copied to a temporary location in order to allow a continuous data stream.

The input shift register has to provide to the control state machine another couple of signals: a signal to tell is if the stored data is a Hit or an EoE word and a signal that tells if in the EoE word there are some warning bits coming from the *FE* chip connected to that particular Receiver. This allows the control state machine to keep track of how many Hits and EoE words have been written to the FIFO and to generate correct Warning bits to be written to the FIFO.

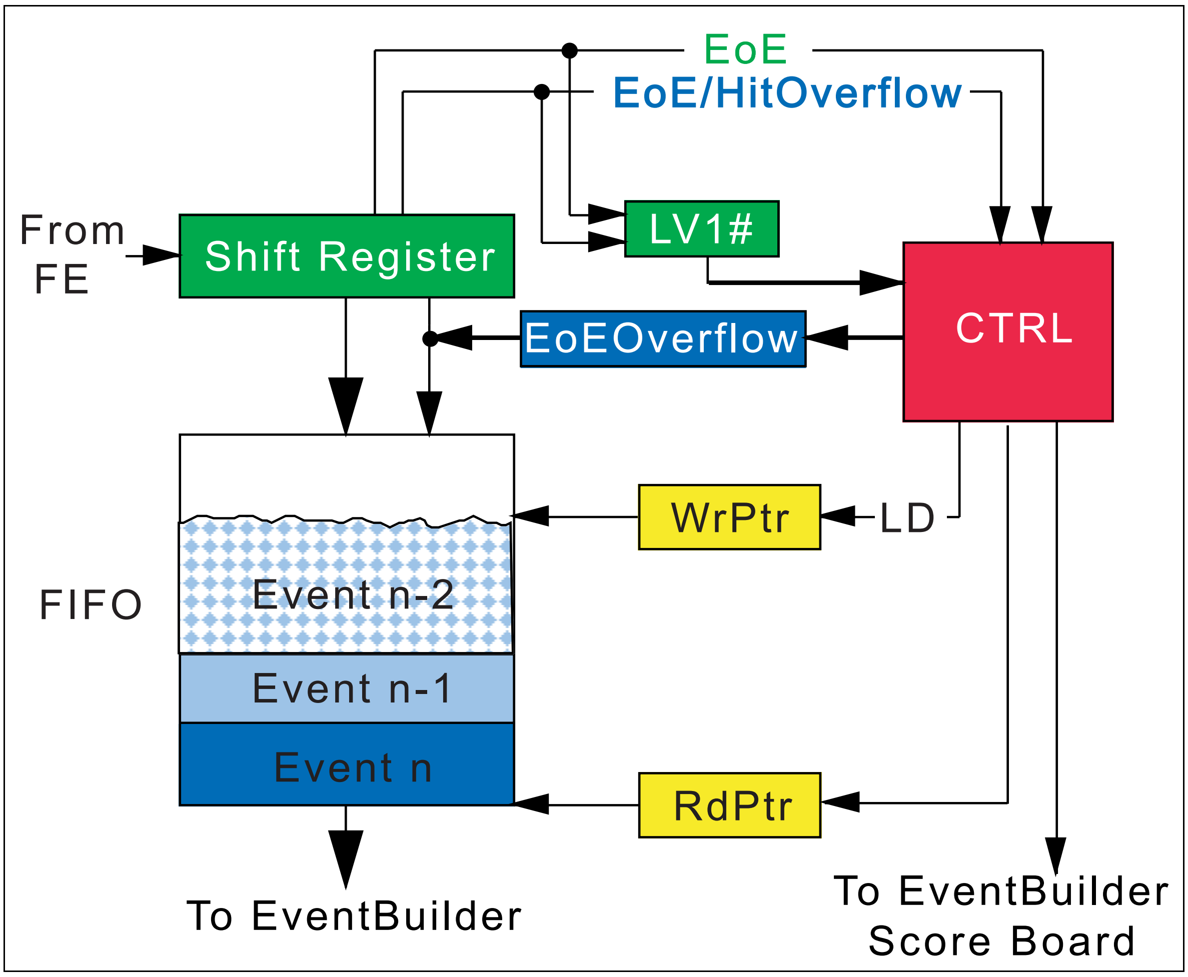

Figure 3-2 Event R/O by *MCC*: Front-End Receiver and input FIFO.

3.1.2 FIFO

All data sent to the chip is eventually copied to the FIFO which is used to store data until all *FE* chips have finished transmitting Event data and event building can begin.

⚠ The ReceiverFifo used in the *MCC* in reality is a 128 x 27 bit wide dual port SRAM full custom block that is used as a FIFO. In order to do this the control state machine provides all the control logic needed for correct pointer management[3-1].

As can be seen in Figure 3-2 the FIFO has two different pointers, one write pointer (WrPtr) and a read pointer (RdPtr). Both pointers are dealt with inside the ControlStateMachine.

Both Hit and EoE words are stored inside the FIFO and the EoE words are used by the EventBuilder to keep event synchronization.

Hits are stored in the FIFO using a “0000” bit pattern in the MSB location of each word, while an EoE word is characterized by a “0111” bit pattern. This format ensures a good rejection against eventual SEU events that might corrupt data inside the FIFO as described in Section 3.3, "SEU protection"

As described in Chapter 5.3, "FE to MCC Event Format", the Hit word contains Row#, Col# and ToT information while the EoE word contains a 3 bit MCCFlag word (warnings generated by the Receiver), an 8 bit FEFlag word (warnings generated by the *FE* chip), a 4 bit LV1Id word, and an 8 bit BCId word.

If the warning message generated by the *FE* chip is not disabled (see Section 2.1, "Register Set") the information of a warning in the data is written to the corresponding bit of the WFE register inside the RegisterBank (see Chapter 2.1.4, "WFE: Warning from FE").

The output of the FIFO is the word pointed by the RdPtr together with some control signals (generated by the Receiver) that inform the EventBuilder if the word read is an Hit or an EoE word, and in such a case if in the current event there are Warnings generated either by the *FE* or by the Receiver itself.

3.1.3 Control State Machine

This block deals with all signals used to correctly write and read information from the ReceiverFifo.

3.1.3.1 Pointer management

Two 7 bit counters keep track of RdPtr and WrPtr. These pointers are incremented each time there is a read or write access to the FIFO. The read access is requested by the OutputPort block of the EventBuilder (see Chapter 4.1.6, "Output Port"). The write access instead is generated inside the Receiver block looking at data coming from *FE* chips.

A check is always performed in order to avoid pointer overrun or underflow.

As both EoE and Hit words are to be written to the FIFO there are two additional up-down counters that keep track of how many Hits and EoE words have been written. As the *MCC* can accept, like the *FE* chip, up to 16 Trigger commands the EoE counter has an upper limit of 16. The Hit counter is limited to 128 -16 = 112 possible values in order to always ensure that we have enough room in the FIFO for writing EoE words. This allows to make sure that there will never be skipped EoE words which would lead to event synchronization problems.

Figure 3-3 describes a normal readout phase. Each time an EoE word is found in the data stream coming from the *FE* the ScoreBoard inside the EventBuilder is updated. As soon as the EventBuilder starts reading data from a FIFO data is read until an EoE word is found in the FIFO. When this happens the EventBuilder stops reading and starts processing the next receiver FIFO that contains some data.
In case of warning messages inside the EoE word the information is processed by the EventBuilder (see Chapter 4.1.5.1, "FeFlag Generator").

3.1.3.2 Hit overflow

This block deals with the case of a Hit overflow inside the ReceiverFifo.

Each time a Hit is written to the FIFO the HitCounter is incremented by one. The counter is decremented each time a Hit word is removed from the FIFO. As soon as the counter reaches the value of 128 - 16 = 112 and a new Hit is detected this Hit is simply dropped and in the next EoE word a warning flag is added (HitOverflow) in order to inform the EventBuilder that, in turn, informs the *ROD*.

In case of Hit overflow this information is also written to the corresponding bit of the WMCC register (see Chapter 2.1.5, "WMCC: Warning from MCC Receivers"), if this check is not disabled (see Section 2.1, "Register Set").

3.1.3.3 EoE overflow

This block deals with the case of a EoE overflow inside the ReceiverFifo.

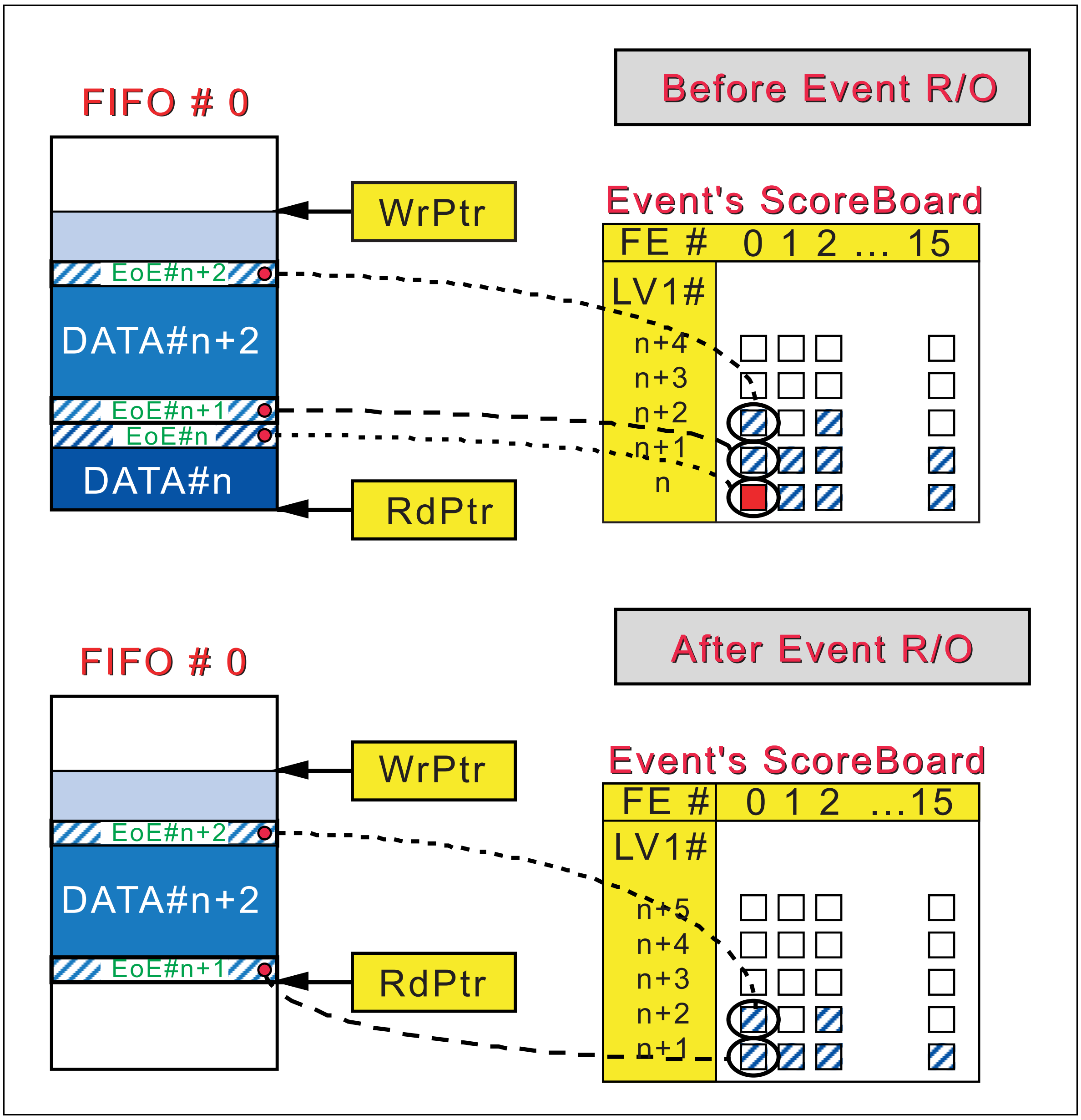

Figure 3-3 Event ReadOut: EoE information is written to the Scoreboard. When event building starts data is read out from the FIFO till the first EoE word is found. Then the EventBuilder proceeds with the next *FE* that contains data until the whole event has been read out. At the same time data can be written to the FIFO.

The MCC also implements an overflow check for the maximum number of EoE words that can be written to the ReceiverFifo. There is a counter that is incremented each time a new EoE word is written to the FIFO and decremented as soon as an EoE word is read-out. If there are already 16 EoE words written in the FIFO and a new EoE word is detected in the *FE* dataflow the EoE word is simply dropped and in this case a different warning flag is added to the EoE word (HitOverflow).

Also in this case the overflow information is written to the corresponding bit of the WMCC register (see Chapter 2.1.5, "WMCC: Warning from MCC Receivers"), if this check is not disabled (see Section 2.1, "Register Set").

In principle this condition should never happen because the *MCC* always ensures that no more than 16 Triggers are sent to the *FE* chips. Being EoE information crucial to correct event synchronization in the whole module we decided to add this feature to be able to check if some data corruption coming from the *FE* chips (for example due to a SEU) has occurred.

3.1.3.4 Data consistency check

Another task that the control state machine has to perform is to check data consistency. Each Hit word coming from the *FE* chip has a 4 bit wide BCId field that contains the FE-BCId of the current event.

The data consistency check is done making sure that all data belonging to the same *FE* event has the same BCId number. To perform this task the first data word of an event is checked and it's 4 bit BCId information is copied into a 4 bit register. In all subsequent data words received from the *FE* belonging to the same event (all words until an EoE is detected) the BCId information is compared with the stored one and in case of a mismatch a warning signal (BCIdChkFail) is issued. The check is also performed between the last Hit word and the corresponding EoE word, as also the EoE word contains the BCId information.

The information is written to the corresponding bit of the WMCC register, if this check is not disabled (see Section 2.1, "Register Set").

3.1.3.5 IncrementScoreBoard signal

There is an additional control signal for the EventBuilder: the IncrementScoreBoard.

This task is performed checking the FIFO word pointed to by the RdPtr, i.e. the next word to be read. If this word is an EoE word the IncremenScoreBoard signal is generated.

3.1.3.6 EoEWngFE signal

This signal is active if an EoE word has been detected on the next word of the FIFO to be read and at least one of the 8 *FE* warning bits is turned on.

Its purpose is to inform the EventBuilder that it has to generate a FE-Flag word in the data stream, if this type of check is not disabled (see Section 2.1, "Register Set").

3.1.3.7 EoEWngMCC signal

This signal is active if an EoE word has been detected on the next word of the FIFO to be read and at least one of the 3 *MCC* warning bits, that can be generated by the Receiver, have been detected.

Its purpose is to inform the EventBuilder that it has to generate a FE-Flag word in the data stream, if this type of check is not disabled (see Section 2.1, "Register Set").

3.2 Reset Actions

This section will describe the response of the Receiver on all available reset commands that can be used in the *MCC*.

The only block that is not affected by any reset signal is the full-custom FIFO. For all the others block the reset is synchronous, as throughout the whole chip, and therefore one needs to have the clock active in order to perform a reset of this block.

3.2.1 Response to a Pin Reset

The pin reset (MCC-RSIb) has the same functionality as the GlobalResetMCC command. It is a synchronous reset and therefore one has to ensure that the clock signal is applied in order to be able to reset the chip in this way.

3.2.2 Response to a GlobalResetMCC command

In case of a GlobalResetMCC command the whole Receiver gets cleared.

Also in this case all the counters are reset to ‘0’, the pointers are re initialized and point both to location ‘0’ of the ReceiverFifo and the command state machine is put to it’s idle state.

⚠ The contents of the 128 FIFO locations are not affected by a GlobalResetMCC command. If one reads the contents of a ReceiverFifo with a series of 128 RdFifo commands immediately after a GlobalResetMCC command the result will be to read the whole contents of the FIFO before the GlobalResetMCC command starting from location ‘0’.

3.2.3 Response to a BCR command

The BCR command has no effect on the Receiver.

3.2.4 Response to an ECR command

An ECR command clears all structures in the Receiver with the exception of the FIFO. It is therefore completely equivalent to a GlobalResetMCC command, with the only exception of being a Fast command, i.e. it does not take the *MCC* out of RunMode. This means that all the counters are reset to ‘0’, the pointers are re initialized and point both to location ‘0’ of the ReceiverFifo and the command state machine is put to it’s idle state.

This allows for a fast reset of the DataPath of the *MCC* as prescribed by the ECR command.

3.3 SEU protection

In order to protect critical configuration information stored inside the *MCC* a certain number of internal structures of each Receiver have been protected against SEU. Not all structures were protected in order to save die area and to minimize power consumption.

3.3.1 Internal structures

The protection against SEU has been implemented using triplicated flip-flops and a majority voting circuit at their output to select the correct value to be used inside the MCC.

Hit and EoE counters as well as all pointer address generators have been triplicated.
Also the IncrementScoreBoard and the EoEWng signals have been protected in this way.

In the Receiver block here is no XOR tree of all triplicated registers to calculate if there has been a SEU due to the fact that this would have required an additional 16 bit wide register, to be implemented inside the RegisterBank, in order to store this information. In addition to this consideration one has to recall that all control lines going from the Receiver block to the *MCC*-core are long, and therefore critical, metal lines that have to be treated very carefully during synthesis and layout due to their high capacitance.

3.3.2 FIFO

The FIFO of each Receiver is one of the most critical parts of the design with respect to SEU.

The FIFO itself is not SEU tolerant and data can be stored inside for long time before being read out and is therefore very sensible to SEU. Nevertheless one has to recall that if there is a single bit-flip inside one data word this would lead to one corrupted data word which can not compromise the correct *MCC* behaviour. The only critical point is therefore being able to correctly separate EoE words from Hit words even in presence of single bit-flips.

In order to address this problem we implemented a redundant EoE/Hit marker.
A Hit word is identified in the FIFO by a “0000” pattern whereas an EoE word is written using a “0111, 1011, 1101, 1110” pattern depending on the fact that there are warnings and/or parity errors in the EoE word.

In the EventBuilder an EoE word is checked against any 4-bit pattern with two or more bits to ‘1’, whereas a Hit word is checked against any 4-bit pattern with up to one bit at ‘1’. This redundant scheme insures that in case of a single bit-flip there can be no mixing of EoE/Hit words, thus ensuring that there cannot be event synchronization problems due to single bit-flips in the data stored inside the FIFO.

3.4 References

3-1 K. Kloukinas, “A Configurable Radiation Tolerant Dual-Ported Static RAM macro, designed in a 0.25 µm CMOS technology for applications in the LHC environment.”, Proceedings of the “8th Workshop on Electronics for LHC Experiments”, 9-13 Sept. 2002, Colmar France