CARLOSrx v4 rel 1.1 for October 2004 ITS test beam reference manual

Samuele Antinori, Filippo Costa, Davide Falchieri,
Alessandro Gabrielli, Enzo Gandolfi,
Massimo Masetti, Samuele Zannoli

Department of Physics and INFN Bologna

November 2004

Outline

N.B. This document only contains documentation about the changes between CARLOSrx release 1.1 and release 1. For any issue not mentioned in this document, please refer to CARLOSrx release 1 datasheet.

What's new in CARLOSrx v4 rel 1.1

CARLOSrx v4 rel 1.1 main upgrade with respect to CARLOSrx rel 1 concerns the trigger interface. In fact release 1.1 is able to interface a 3-level trigger system as the one provided by the LTU and TTC system.

Hardware upgrades:

- Same board as CARLOSrx rel 1.
- The TTCrx board has been integrated in the CARLOS – CARLOSrx box.

Firmware upgrades:

- The new firmware expects to receive a 3-level trigger: L0 – L1a – L2a.
    - L0 is received 1.2 μs after the interaction;
    - L1a (accept) is received 5.3 μs after L0. If it not received in the due time, the previously received L0 signal is rejected and no event is acquired.
    - L2a (accept) is received within 100 μs after L0. In this case an event is sent to the DAQ. On the contrary if a L2r (reject) is received, the trigger is rejected and no event is acquired.
- L1a and L2a bring with them global and local identification numbers, above all orbit ID and bunch number ID, used to identify events. These information have to be attached to the standard header of each event: so far the header has a new structure.
- A new word has been added at the end of each event, after the 3 footer words, containing the information of the latency, in clock cycles, of the trigger signal from the moment it is received on CARLOSrx to the moment it is actually sent in output over the serial link towards CARLOS.
- The JTAG words, containing the JTAG answers from the front-end chip after JTAG programming, are no longer stored in CARLOSrx internal FIFO: they are discarded.
- No dummy event is sent in output even after a synchronization error between AMBRA and CARLOS.

Software upgrades:

- An upgraded version of the C++ decoding software *carlosrx* has been developed. It features an easy to use GUI interface and it also allows to view reconstructed data by using ROOT functions.

Main features

- XC2V1000 Xilinx Virtex2 FPGA;
- 40 MHz working frequency;
- 1.8 V core power supply; 2.5 V I/O pads power supply;
- standard IEEE 1149.1 JTAG implemented;
- interface towards CARLOS v4 implemented;
- interface towards the SIU implemented (with flow control);
- interface towards the trigger system (LTU + TTC) implemented;
- it can be directly interfaced either to CARLOS or to the optical link.

Known limitations

- Even if the board allows to acquire data coming from two CARLOS chips (= two detectors) this version of the firmware of CARLOSrx allows to acquire data coming from 1 CARLOS (= 1 detector only).
- The CARLOSrx board can be connected to CARLOS in either of 2 ways:
    - direct connection;
    - with the optical link (GOL + optical fiber + TLK1501)

  If used in the second configuration, the clock and the serial back-link running from CARLOSrx to CARLOS have to be carried with wires (not with optical fibers as expected in ALICE). In fact no optical transceiver for these two signals has been foreseen on the CARLOSrx board.

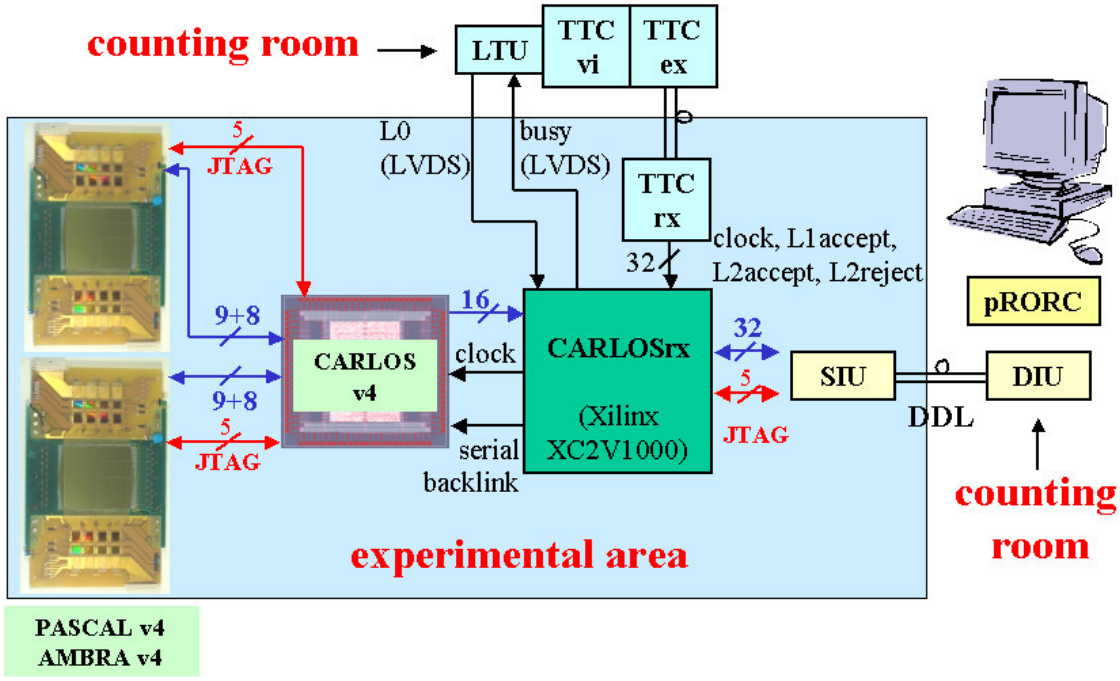

Fig. 1: SDD readout chain at the ITS October 2004 test beam

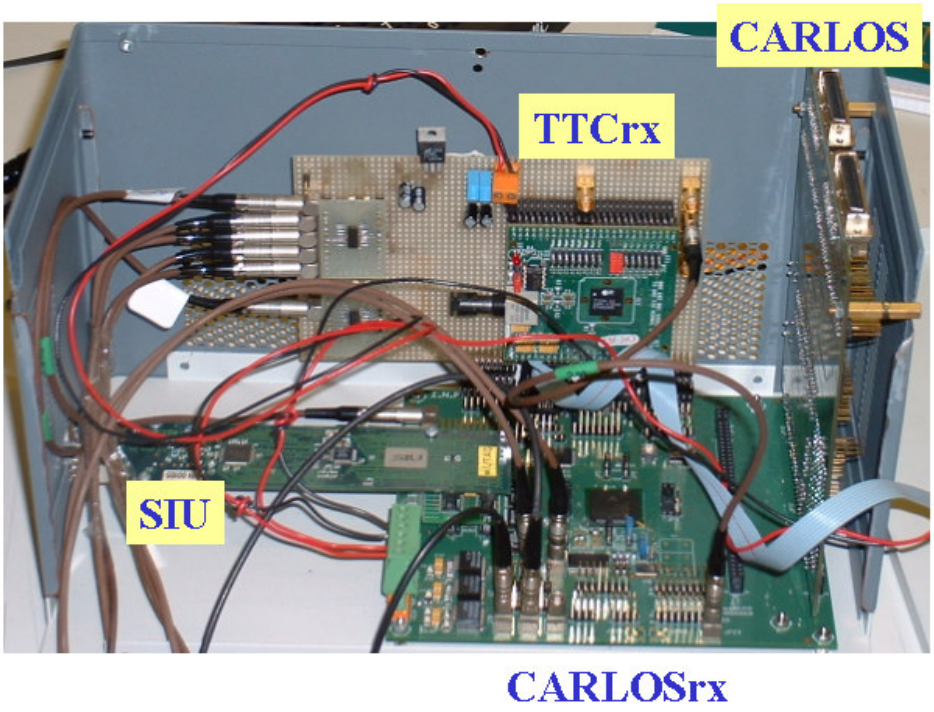

Fig. 2: CARLOS – CARLOSrx: internal view of the box

What's new in October 2004 ITS test beam

The most important changes in the chain setup between the August 2004 SDD test beam and the October 2004 ITS test beam are the following:

- 6 different detectors are used at the same time:
    - 2 SPDs
    - 2 SDDs
    - 2 SSDs
- the current prototype of the final ALICE trigger system has been used;
- the ECS (Experiment Control System) software has been used to control:
    - DAQ
    - DCS
    - Trigger

From the point of view of the SDD readout chain (see Fig. 1), the main change was the integration with the new trigger system. The trigger system consists of the following parts:

- LTU (Local Trigger Unit): used in standalone mode it is able to generate trigger sequences (L0-L1-L2a, L0-L1-L2r, L0) after receiving the trigger input from the scintillators coincidence.
- TTC (Timing, Trigger and Control): this is the device used to distribute clock and trigger information to the ALICE sub-detectors. It is composed by:
    - TTCvi
    - TTCex (sends clock and trigger over an optical fiber)
    - TTCrx (decodes clock and trigger received over an optical fiber)

So far the TTCrx board has been integrated in the CARLOS – CARLOSrx box (see Fig. 2) that now contains:

- CARLOS v4 board;
- CARLOSrx board;
- SIU;
- TTCrx

General description

CARLOSrx v4 rel 1.1 is a Xilinx Virtex2 FPGA-based device with the main purpose of concentrating data coming from one SDD detector on one LDC.
CARLOSrx packs data coming from the front-end electronics and CARLOS into 32-bit words, stores them in a large data FIFO and then sends them towards the DDL system, after a transaction has been opened by the SIU. The use of a FIFO allows to avoid losing data even when the DDL asserts the flow control: in fact CARLOSrx asserts the back-pressure towards CARLOS and CARLOS will stop AMBRA, thus freezing the data acquisition process until the DDL is ready to accept data again.
CARLOSrx also drives the serial backlink port towards CARLOS for sending JTAG information to PASCAL, AMBRA, CARLOS and GOL and for sending reset commands, trigger signals and control commands to CARLOS .
It also interfaces the trigger system by receiving the *L0* signal, the L1a signal and L2a message and asserting the *busy* signal. When CARLOSrx does not receive the L1a signal at the proper time or when it receives the L2reject information, CARLOSrx sends the related command to CARLOS for aborting the trigger and no event is sent to the DAQ.
CARLOSrx also puts at the end of each event an information regarding the latency between the moment L0 is received and the moment in which it is actually sent to AMBRA (jitter information).
The firmware implemented on the FPGA contains 5 major logic blocks:

1. *data packing*: CARLOSrx receives the 16-bit data words coming from CARLOS, groups them depending on their type, packs them into 32-bit words and stores them into a FIFO, before they are sent towards the SIU.
2. *SIU interface*: this block manages the protocol interface towards the SIU. It is able to recognize the commands sent from the SIU and then to send packed data towards the SIU.
3. *trigger interface*: this block directly interfaces the trigger system by receiving the trigger inputs (L0, L1a, L2a, L2r and related numbers) and asserting the busy signal.
4. *JTAG interface to SIU*: this block receives the JTAG signals from the SIU and encapsulates them on the serial back-link towards CARLOS. It also implements 4 JTAG instructions: "Put CARLOS in JTAG mode", "Put CARLOS in RUN mode", "Load anode length" and "Load trigger delay". When one of these instructions is detected, a signal is sent to the serial backlink block in order to send to CARLOS the corresponding command.
5. *serial backlink*: this block drives the *serial backlink* signal from CARLOSrx to CARLOS. It is used to send JTAG commands, reset commands, trigger signals, prepulse and testpulse signals, the commands "Enter JTAG mode" and "Enter RUN mode".

Interface to the trigger system

CARLOSrx rel 1.1 receives a 3-level trigger L0 – L1 – L2 from the LTU and the TTC system and sends back to the LTU a busy signal. In particular:

- *L0*: LVDS signal, from the LTU to CARLOSrx on a 60m long CAT 6 UTP, AWG 24 cable.
- *busy*: LVDS signal, from CARLOSrx to the LTU on a 60m long CAT 6 UTP, AWG 24 cable.
- *L1-L2*: information sent using the TTC system (TTCvi, TTCex, TTCrx). The TTCex and TTCrx boards are connected through a 60m optical fiber. CARLOSrx receives the following information from the TTCrx (see Fig. 4):
    - *clock40:* 40 MHz clock;
    - *L1accept*
    - *BCnt* (12 bits): bunch counter value locally counted on the TTCrx chip;
    - *BCntStr*: bunch counter strobe;
    - *Dout* (8 bits): data bus for sending L1a and L2a messages;
    - *SubAddr* (8 bits): address bus for sending L1a and L2a messages;
    - *DoutStr*: data bus strobe

The information received from the TTCrx are packed together with the corresponding event in the DDL header (see "Event header format" paragraph).
For what concerns the busy signal, two CARLOSrx firmware versions have been developed and tested:

- *single buffer version*: the busy signal is asserted after receiving L0 and put to 0 again when:
    - either when the whole event has been transmitted to the DDL;
    - or when no L1a is received in the due time or an L2r is received.

  In this configuration only one of the 4 AMBRA buffers is used at a time. When using anode length = 256, the busy on mean time is 2 ms. So far during the ITS test beam we were able to acquire 1500 events per burst.
- *multi buffer version*: the busy signal is asserted after receiving L0 and put to 0 again when the corresponding busy signal from CARLOS turns to 0 again. In this way CARLOSrx busy is a copy of the busy signal received from CARLOS in the error flag words.

  In this configuration all the 4 AMBRA buffers can be used. Using standard configuration for PASCAL (analog memory full frequency and ADC half frequency) and anode length = 256, the busy on time is 500 μs when some buffer is free and 1.6 ms when all AMBRA buffers are full. So far during the ITS test beam we were able to acquire 2500 events per burst with a busy on mean time of 1.2 ms. A 5M events run was acquired in this configuration. No double events were observed coming from AMBRA using the following parameters:
  AMBRA SOP delay = 160
  CARLOSrx trigger delay = 209

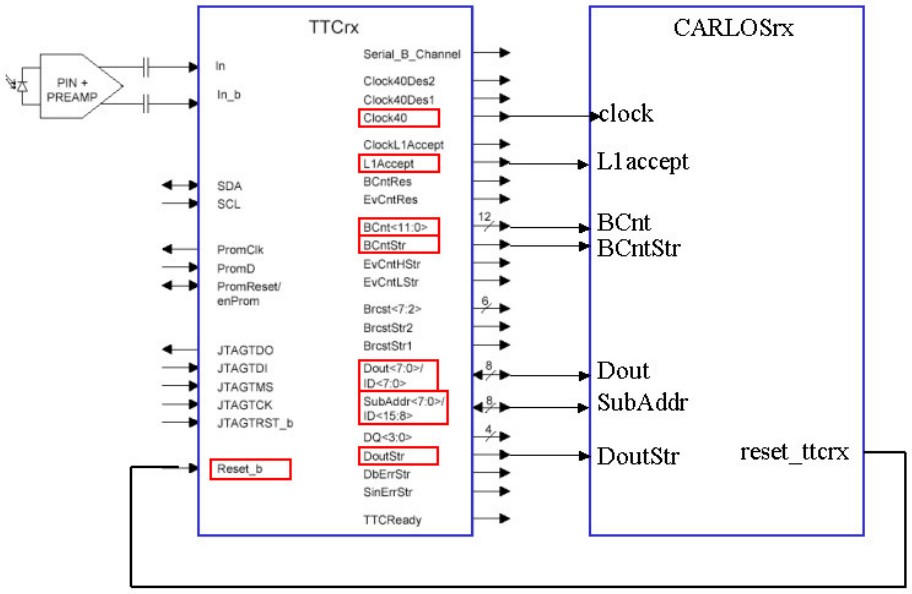

Fig. 4: TTCrx – CARLOSrx connections

Event header format

Every event begins with the DDL header (8 32-bit words) and ends with the FESTW (Front End Status Word) (see Fig. 5). The DDL header is required by the DATE SW.
The DDL header is filled with information received from the trigger system with the aim of identifying events from each other. This information is also useful for the event builder, so to put together sub-events coming from the same event.
The most interesting information are highlighted in blue in Fig. 5.

- ddl_header1:
    - L1a header message;
    - L2a message: bunch counter ID;
- ddl_header2:
    - L2a message: orbit ID;
- ddl_header3:
    - L2a message: participating sub-detectors;
- ddl_header4:
    - from TTCrx: mini event ID locally counted bunch counter ID on TTCrx;
- ddl_header5:
    - L2a message: trigger classes low;
- ddl_header6:
    - L2a message: trigger classes high.

When decoding data some criteria can be taken into account, just to check data consistency:

1. the orbit ID has always to increase between an event and the following one;
2. the difference between the bunch counter ID (from L2a message) and the mini event ID has always to be constant. Even if they are 12 bit numbers, they are modulo 3564 (0xDEC): this has to be taken into account when performing the difference.

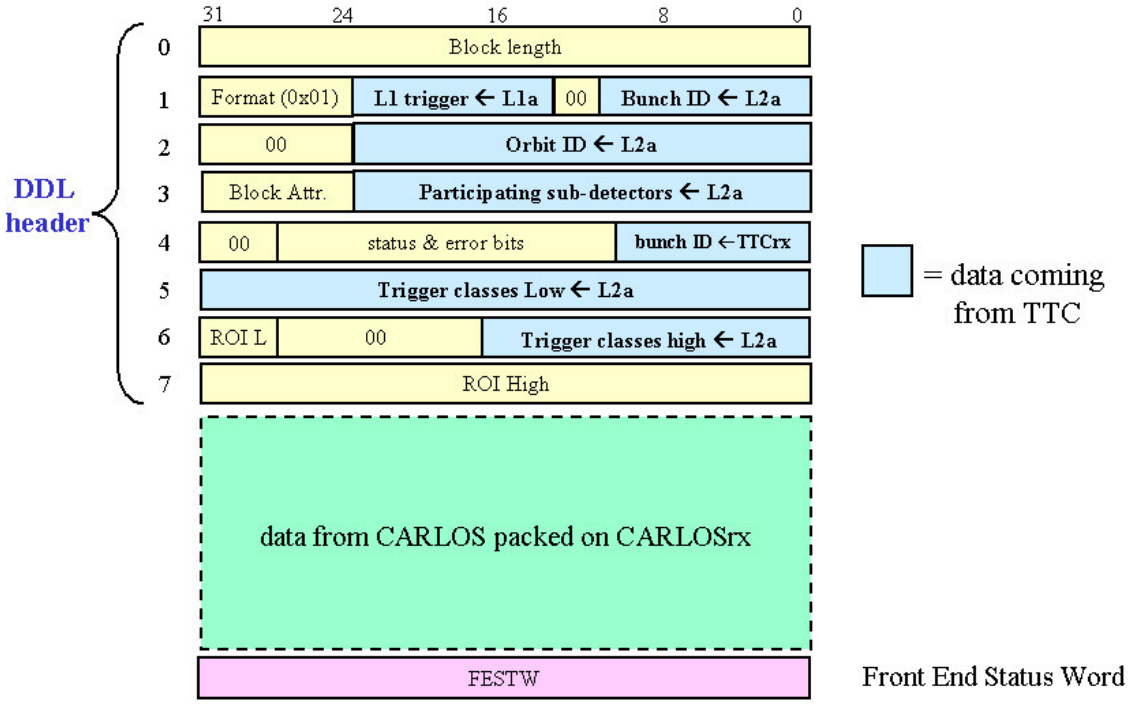

Fig. 5: CARLOSrx event data format

JTAG instruction set

CARLOSrx receives the JTAG bus from the SIU and encapsulates the JTAG information on the serial back-link towards CARLOS as it is. Beside that, CARLOSrx internal JTAG unit monitors the input JTAG port looking for the JTAG instructions reported in Table 1.

JTAG instruction	JTAG IR value	Length of scan register involved
Put CARLOS in JTAG mode	10001	5
Put CARLOS in RUN mode	10010	5
Load anode length	10011	5
Load trigger delay	10000	5

Table 1: List of CARLOSrx JTAG instructions

- After decoding the instruction "Put CARLOS in JTAG mode", the command "Enter JTAG mode" is sent to CARLOS through the serial backlink.
- After decoding the instruction "Put CARLOS in RUN mode", the command "Enter RUN mode" is sent to CARLOS through the serial backlink.
- After decoding the instruction "Load anode length", the value of a 8-bit register is read from the JTAG port and put in output as ninth word of each event sent towards the LDC as LSBs. All the other bits of the ninth word are set to 1. For example, if the anode length value received is 0xC7, the value of the ninth word sent in output within each event is: FFFFFFC7. This information may prove useful when decoding data of unknown anode length (decoding information mixed together with data).
- After decoding the instruction "Load trigger delay", the value of a 8-bit register is read from the JTAG port and stored internally. The value of this register is the number of clock cycles of latency with which the backpressure is applied after receiving a trigger, with the purpose of avoiding faulty double events from AMBRA. This value is directly related to the SOP delay register value on AMBRA with the following relationship:

$$trigger\ delay\ value = SOP\ delay + 49$$

Interface to CARLOS

CARLOSrx receives data coming from CARLOS v4. These are the interface signals:

- *output_data* (16 bits): this is the 16-bit bus containing the data coming from CARLOS v4;

- *tx_en*: it is a strobe signal, active high. When active the *output_data* bus contains a valid value, whichever its type (header, footer, data from ch1, data from ch0, error flag word, JTAG word).
- *cav*: it is a strobe signal, active high. When active, the *output_data* bus contains a valid control word, that is either a JTAG word or an error flag word.
- *serial backlink*: it is a serial link from CARLOSrx to CARLOS. It is used to send 8-bit commands to CARLOS, such as reset signals, trigger signals, JTAG information and commands for putting CARLOS in JTAG mode or in RUN mode.

The *serial backlink* block on CARLOSrx is used to drive the *serial backlink* signal. After the reset is activated, the block sends a number of IDLE codes for link synchronization, then it sends the commands for resetting all the front-end chips (PASCAL, AMBRA, CARLOS and GOL). Then it continues sending IDLE codes until it has to send one of the following commands (from highest priority to lowest: if two commands have to be sent at the same time, the highest priority one is sent first)):

- enter JTAG mode;
- enter RUN mode;
- trigger signal;
- JTAG information;
- L1reject;
- L2reject;
- prepulse25;
- testpulse;
- prepulse50, 75, 100, 125, 150, 175, 200
- stop acquisition;
- restart acquisition.

Since these commands are to be sent over a 8-bit serial link, there is a variable jitter of a few clock cycles from the moment the trigger signal *L0* is received till the moment it is actually sent to CARLOS over the serial back-link. CARLOSrx computes the jitter value after receiving each trigger signal, then this value is put in output as the last word of the corresponding event.

The jitter value is computed in the following way (see Fig. 3):

- on the rising edge of the clock *L0* is sampled and its value stored in *trigger_sync1*;
- on the next rising edge of the clock *trigger_sync1* is sampled and its value passed to *trigger_sync*;
- on the next rising edge of the clock *trigger_sync* is sampled and its value passed to *trigger_to_serial* if busy = 0;
- on the first rising edge of the clock in which *trigger_to_serial* is sampled high, a counter *jitter_trigger* is started (*jitter_trigger* += 1);
- on the first rising edge of the clock in which *tdc* is sampled high, the counter *jitter_trigger* is incremented for the last time (the *tdc* internal signal is activated for

one clock cycle when a trigger command is being sent over the serial back-link and during the third clock cycle out of 8 due cycles).

- then this 8-bit value is copied as the LSBs of the last word of the corresponding event (the 8 MSBs are set to 1, while the others to 0). If, for example, the value of this number is 0xE the last word of the related event is 0xFF00000E.

The jitter word has quite a flat distribution in the range 0x7 – 0xE. In a very few cases also larger jitter values may occur, such as 0x12. This happens when a trigger L0 is received just when an other command is being or about to be sent over the serial back-link, such as a stop or restart acquisition command. In this case the time for the trigger input before being sent in output is larger than usual. This behavior has some chances to occur when the back-pressure is activated very often. For example when running in single-buffer and compressing data (6 KB per event as at the test beam) the back-pressure is never activated and the jitter word has a flat distribution in the range 0x7 – 0xE. When running in multi-buffer in the same conditions the back-pressure may be activated and some jitter value bigger than 0xE can be found. The same may also happen when running in single-buffer with an event size larger than CARLOS internal data FIFO (80 KB).

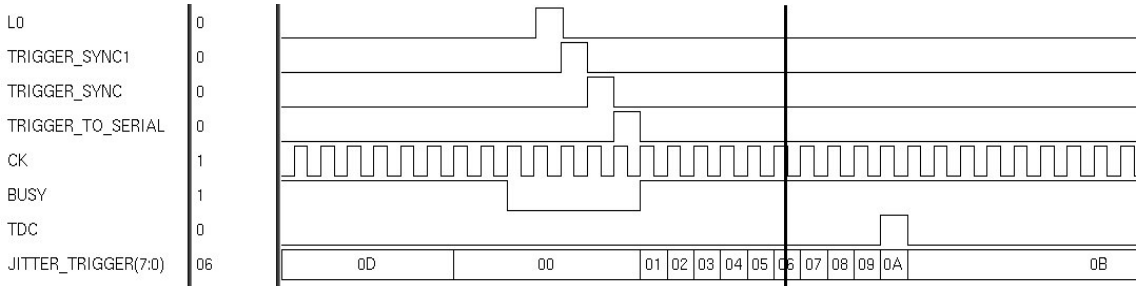

Fig. 3: Timing of the jitter information: in this case its value is 0x0B

CARLOSrx FIFOs

CARLOSrx hosts on-chip memories for storing data, trigger and jitter information . In particular:

- data FIFO: 20 K 32-bit words divided in
    - large fifo: 16 K 32 bit words
    - small fifo: 4 K 32 bit words
- trigger FIFO: 511 32-bit words
- jitter FIFO: 127 32-bit words.

The FIFOs for trigger and jitter data are necessary only for the multi-buffer version: in fact the input trigger rate is much higher than event transmission in output, at least locally in time. So these information have to be stored in a FIFO and then popped when the corresponding event is being sent in output. When using CARLOSrx in single-buffer version no trigger and jitter FIFO is necessary since before accepting a new trigger sequence the whole event has to be sent in output first.

CARLOSrx v4 rel 1.1 pin function

Terminal name	Type	Description
carlos4_output(15-0)	I	Input data bus coming from CARLOS v4
tx_en	I	Input signal coming from CARLOS v4: when 1 it means that CARLOS is sending a valid word
cav	I	Input signal coming from CARLOS v4: when 1 it means that CARLOS is sending a valid error flag word or JTAG word
dav	I	Input signal coming from CARLOS v4: when 1 it means that CARLOS is sending a valid data
serial backlink	O	Output signal towards CARLOS v4: it carries JTAG information and commands (reset, trigger, backpressure, …)
reset_carlos	O	Output signal resetting CARLOS v4 (active low reset)
gol_ready	O	Output signal towards CARLOS v4 (when the GOL is not used): its value is fixed to 1
ck	I	Input clock from TTCrx
reset_n	I	Active low reset
fidir	I	From the SIU: it decides the bidirectional port direction
fiben_n	I	From the SIU: it decides if the bus is enabled or not (active low)
filf_n	I	From the SIU: it decides if the link is full or not (active low)
foclk	O	To the SIU: 20 MHz free running clock
fbten_n	INOUT	To and from the SIU: when active (low) the data *fbd* is valid
fbctrl_n	INOUT	To and from the SIU: when active (low) *fbd* contains the FESTW
fobsy_n	O	To the SIU: when active (low) CARLOSrx is not ready to accept data from the SIU
fbd	INOUT	32-bit data bus between CARLOSrx and SIU
tdi_from_siu	I	JTAG *tdi* from SIU
tck_from_siu	I	JTAG *tck* from SIU
tms_from_siu	I	JTAG *tms* from SIU
trst_from_siu	I	JTAG *trst* from SIU
tdo_to_siu	O	JTAG *tdo* to SIU
L0	I	Input trigger, active high, 25 ns wide
busy	O	Output busy
reset_ttcrx	O	Reset towards the TTCrx chip
L1accept	I	L1accept signal
BCnt(11-0)	I	Bunch Counter from TTCrx
BCntStr	I	Bunch Counter Strobe

Dout(7-0)	I	Data Out bus from TTCrx
SubAddr(7-0)	I	Address bus from TTCrx
DoutStr	I	Data Out strobe from TTCrx