**FMC TDC 1 ns 5 channels | firmware guide**

on a SPEC carrier board

E.Gousiou | November 2013



FMC TDC 1 ns 5 cha | ohwr.org wiki pages

**Contents**

[1. Introduction 5](#_Toc372537480)

[2. GN4124 core 7](#_Toc372537481)

[3. TDC core 8](#_Toc372537482)

[3.1 Clock Domains 10](#_Toc372537483)

[3.2 Timestamping concept 11](#_Toc372537484)

[3.3 Clocks and Resets Manager 15](#_Toc372537485)

[3.4 Registers controller 16](#_Toc372537486)

[3.5 Data Engine 19](#_Toc372537487)

[3.6 ACAM databus interface 22](#_Toc372537488)

[3.7 Circular buffer 23](#_Toc372537489)

[3.8 One Hz Generator 23](#_Toc372537490)

[3.9 Start Retrigger Controller 24](#_Toc372537491)

[3.10 Data Formatting 26](#_Toc372537492)

[3.11 Interrupts generator 30](#_Toc372537493)

[3.12 ACAM timecontrol interface 30](#_Toc372537494)

[3.13 LEDs 30](#_Toc372537495)

[4. Interrupt Controller 31](#_Toc372537496)

[5. Mezzanine 1-Wire 31](#_Toc372537497)

[6. Mezzanine I2C 31](#_Toc372537498)

[7. Carrier 1-Wire 32](#_Toc372537499)

[8. Carrier CSR 32](#_Toc372537500)

[9. HDL repository 32](#_Toc372537501)

[10. Xilinx XC6SLX45T resources 32](#_Toc372537502)

[11. TDC mezzanine board calibration 33](#_Toc372537503)

[12. TDC mezzanine performance 33](#_Toc372537504)

# Introduction

The Time to Digital Converter mezzanine board FmcTdc1ns5cha has 5 input channels. Its purpose is to calculate time differences between pulses arriving to the channels with a precision of ±700 ps. It follows the [FMC architecture](http://www.ohwr.org/projects/fmc-projects/wiki) and can be carried by any of the carrier boards: [SPEC](http://www.ohwr.org/projects/spec/wiki) or [SVEC](http://www.ohwr.org/projects/svec/wiki). It is using a dedicated time-to-digital converter chip [TDC-GPX](http://www.acam.de/fileadmin/Download/pdf/English/DB_GPX_e.pdf), developed by the European company ACAM.

This document describes the hdl developed to support the FmcTdc1ns5cha mezzanine board, later referred to as fmc-tdc, on a SPEC carrier. The firmware structure is described in detail as well as the configuration and operation. For the software support of the fmc-tdc a different dedicated documentation is available [here](http://www.ohwr.org/projects/fmc-tdc-sw/wiki).

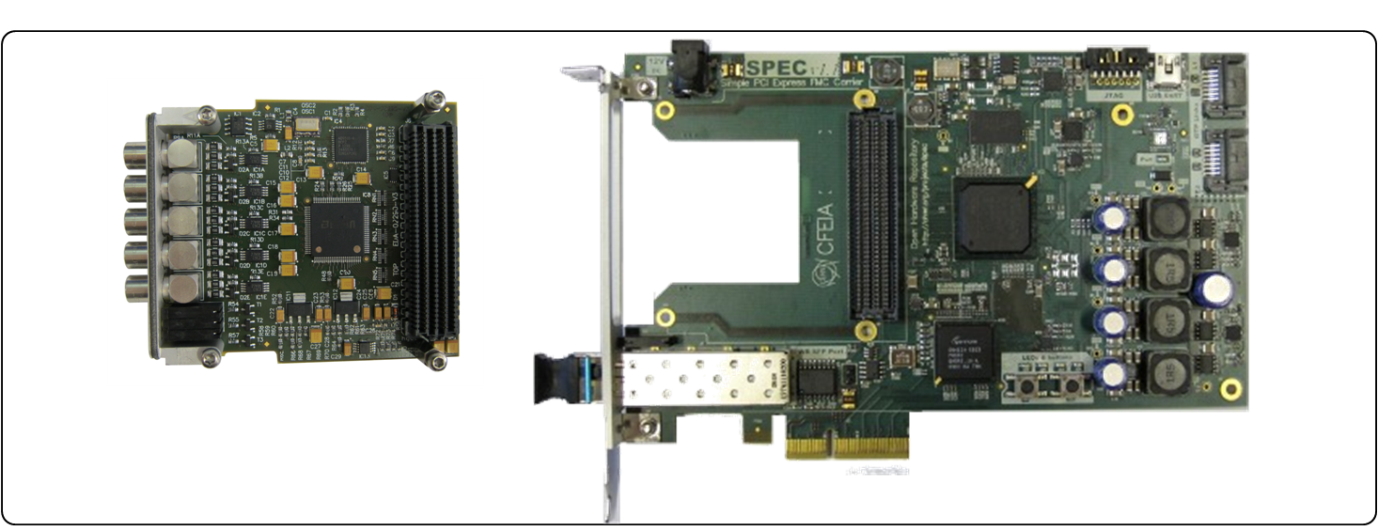


Figure 1: TDC mezzanine and SPEC carrier boards

The SPEC carrier board, mounted inside a PCIe crate, provides FPGA logic, power supplies, clocking resources as well as the interface to the PCIe bus. The TDC mezzanine board houses mainly the five input channels and the ACAM time-to-digital converter chip.

The TDC core is housed in the FPGA of the carrier board.

The TDC core is first configuring the ACAM chip. The configuration instructions are provided through the PCIe interface. Once the ACAM chip is configured, rising edges arriving to any of its channels are time-stamped. The TDC core is responsible for retrieving the timestamps; it is then putting them in a convenient format and finally it is making them available to the PCIe interface.

Figure 2 shows the firmware architecture, followed by a short description of the main components.

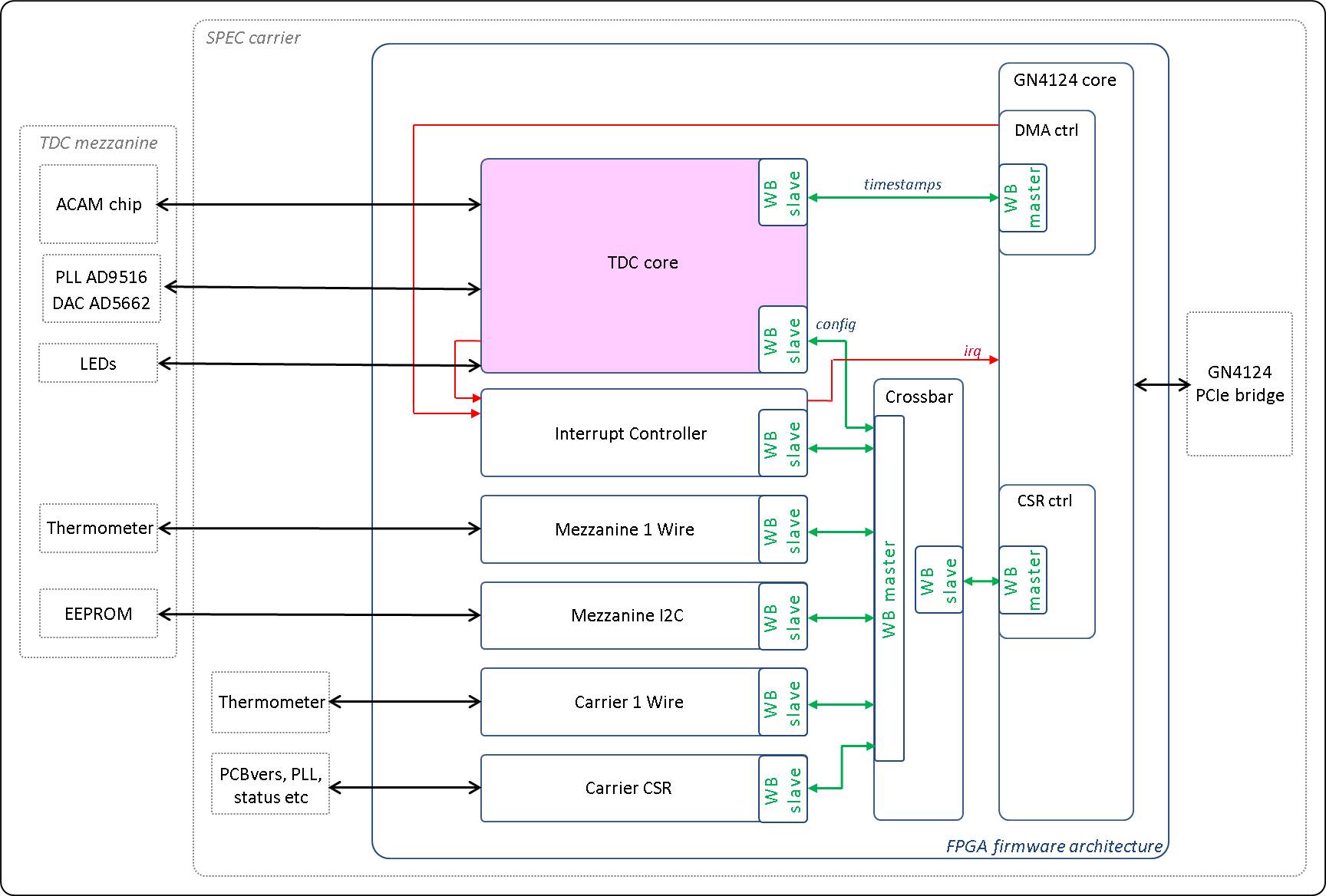


Figure 2: SPEC TDC firmware architecture

The [**GN4124 core**](http://www.ohwr.org/projects/gn4124-core/wiki) is the interface to the Gennum 4124 PCIe bridge chip. Internally it provides two separate WISHBONE buses: the Control and Status Register (CSR) standard WISHBONE interface and the Direct Memory Access (DMA) pipelined WISHBONE interface.

For the configuration of the **TDC core** as well as the ACAM chip, dedicated registers are allocated; they are accessible through the GN4124 core CSR interface. The timestamps that are received by the ACAM and formatted in the core are then stored in a circular buffer, that is accessible through the DMA interface. Note finally that the TDC core has also embedded an SPI master interface for the configuration of the PLL AD9516 and DAC AD5662 located on the mezzanine.

The **interrupts controller** gathers all the interrupts into one single interrupt request line.

The **mezzanine 1-wire** controls the DS18B20 thermometer and unique ID chip on the mezzanine.

The **mezzanine I2C** controls the 24AA64 EEPROM memory chip located on the mezzanine board.

The **carrier 1-wire** controls the DS18B20 thermometer and unique ID chip on the SPEC carrier board.

The **carrier csr** contains control and status registers related to the carrier board (ex. pcb version).

The **crossbar** is used to map the WISHBONE slaves in the PCIe BAR 0 address space.

# GN4124 core

As Figure 2 shows, this block is the interface between the GN4124 local bus and the rest of the blocks in the FPGA.

The [GN4124 chip](http://www.alcom.nl/binarydata.aspx?type=doc/Gennum_GN4124x4.pdf) is a four lane PCI Express Generation 1.1 bridge. In addition to the PHY, it also contains the data link and transaction layers. The GN4124 chip is used to access the FPGA registers, but also to generate MSI interrupts and re-program the FPGA. The BAR 4 (Base Address Register in the PCIe memory space) allows access to the GN4124 internal registers. The BAR 0 is connected to the local bus allowing access to the FPGA.

The [GN4124 core](http://www.ohwr.org/projects/gn4124-core/wiki) in the FPGA is made of a local bus interface with the GN4124 chip on one side and a CSR controller and a DMA controller on the other side.

# TDC core

The TDC core is retrieving timestamps generated by the ACAM chip, adapting them to a comprehensive format and making them available in a RAM to the PCIe interface.

Each final timestamp is a 128-bit word with the following structure:

|  |  |
| --- | --- |
| Bits | Description |
| [127:96] | Metadata: rising/falling tstamp, channel number |
| [95:64] | Local UTC time: each bit represents 1 s |
| [63:32] | Coarse time within the current UTC time: each bit represents 8 ns |
| [31:0] | Fine time: each bit represents 81.03 ps |

Table 1: Timestamp format

As the structure indicates, each timestamp is referred to a UTC second. The coarse and fine times indicate with 81.03 ps resolution the amount of time passed after the last UTC second.

Figure 3 shows the main components of the core followed by a short description of each component.

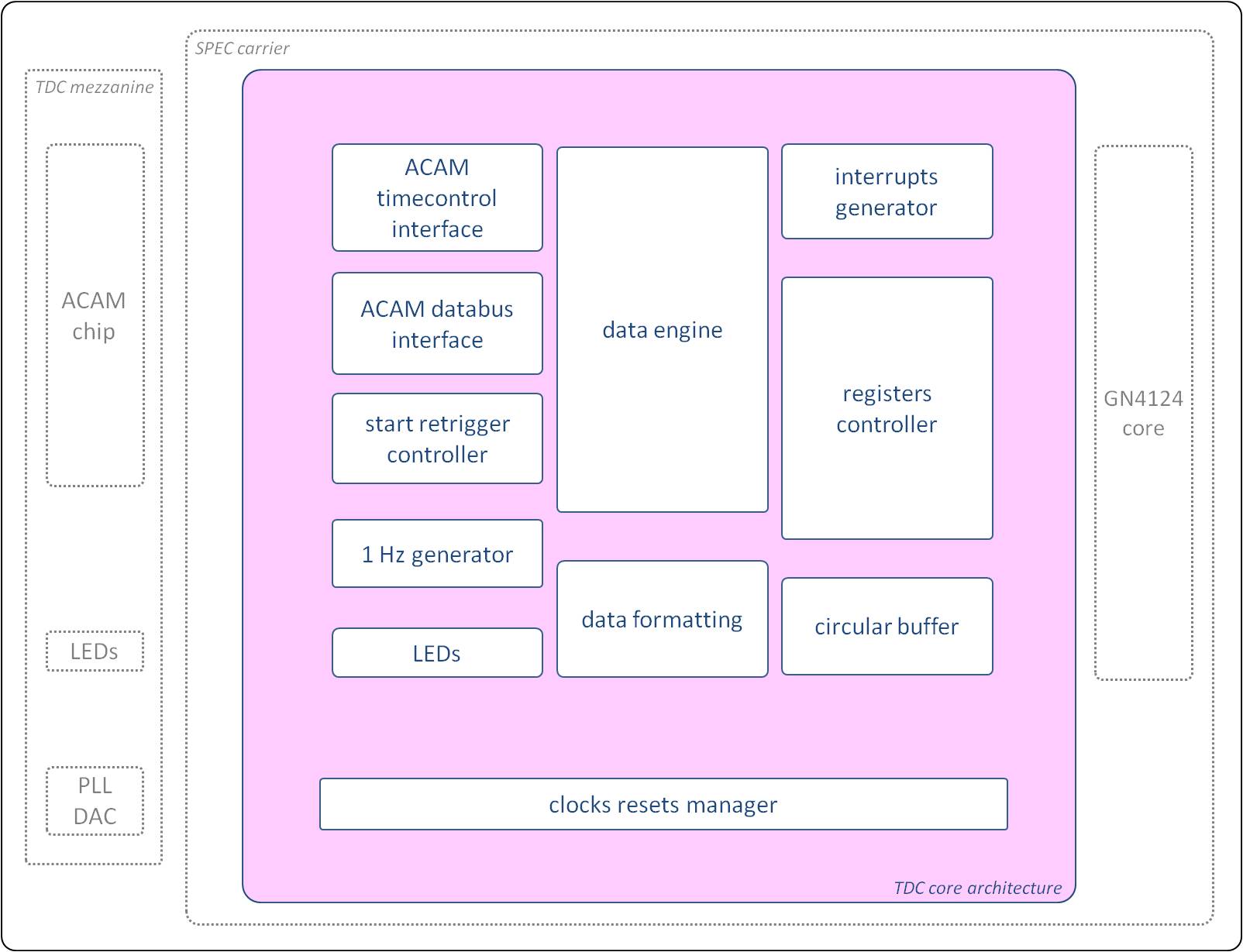


Figure 3: Main components of the TDC core

The “**one Hz generator**” unit is responsible for keeping the UTC time. Currently this timekeeping depends on the AD9516 PLL on the mezzanine board, which provides a 125 MHz clock. Future upgrades of the core will provide White Rabbit accuracy.

The timestamps are formatted to the structure of Table 1 in the “**data formatting**” unit.

Then they are stored in a RAM block implemented in the “**circular buffer**” unit, where the GN4124 core has direct access.

In this application, the ACAM is used in I-Mode. This provides unlimited measuring range with internal start retriggers. ACAM's counter of retriggers however is not large enough and there is the need to follow ACAM’s retriggers inside the core; the “**start retrigger control**” unit is responsible for that.

The “**registers controller**” implements the communication with the GN4124 core for the configuration of the TDC core and of the ACAM chip. Amongst the registers, one in particular is utterly important: the “control register” defines the action to be taken in the core (ex. activation of timestamps acquisition, loading of configuration to the ACAM chip, etc)

The “**data engine**” unit is managing the state of the core according to the “control register”.

The “**acam databus interface**” implements the communication with the ACAM chip for its configuration as well as for the timestamps retrieval.

The “**acam timecontrol interface**” is responsible for delivering to the ACAM chip the start pulse, to which all timestamps are related.

The core is providing an interrupt in any of the following three cases:

* accumulation of timestamps larger than the settable threshold
* more time passed than the settable time threshold and >=1 timestamps accumulated
* error occurred in the ACAM chip

The “**interrupts generator**” unit is responsible for generating the corresponding interrupt pulses.

The “**clocks and resets manager**” unit is configuring the PLL AD9516 and DAC AD5662 on the TDC mezzanine for the production of two clocks: a 125 MHz clock for the TDC core and 31.25 MHz clock for the ACAM chip. It is also managing the TDC core resets.

Finally, the “**leds**” unit manages the six TDC mezzanine front panel LEDs.

## Clock Domains

There are two clock domains in the TDC core, as Figure 4 shows. Upon power-up or after a PCIe reset, the “clocks and resets manager” is using the 20 MHZ VCXO of the SPEC carrier to configure the DAC and the PLL on the TDC mezzanine. The PLL generates a 125 MHz clock arriving to the TDC core to be used by the rest of the logic. The 31.25 MHz clock is a division of the 125 MHz clock provided by the same PLL and is the reference clock for the ACAM chip.

The “clocks and resets manager” is keeping the rest of the TDC core logic under reset until the PLL gets locked. The only signal crossing domains in the core is the reset.

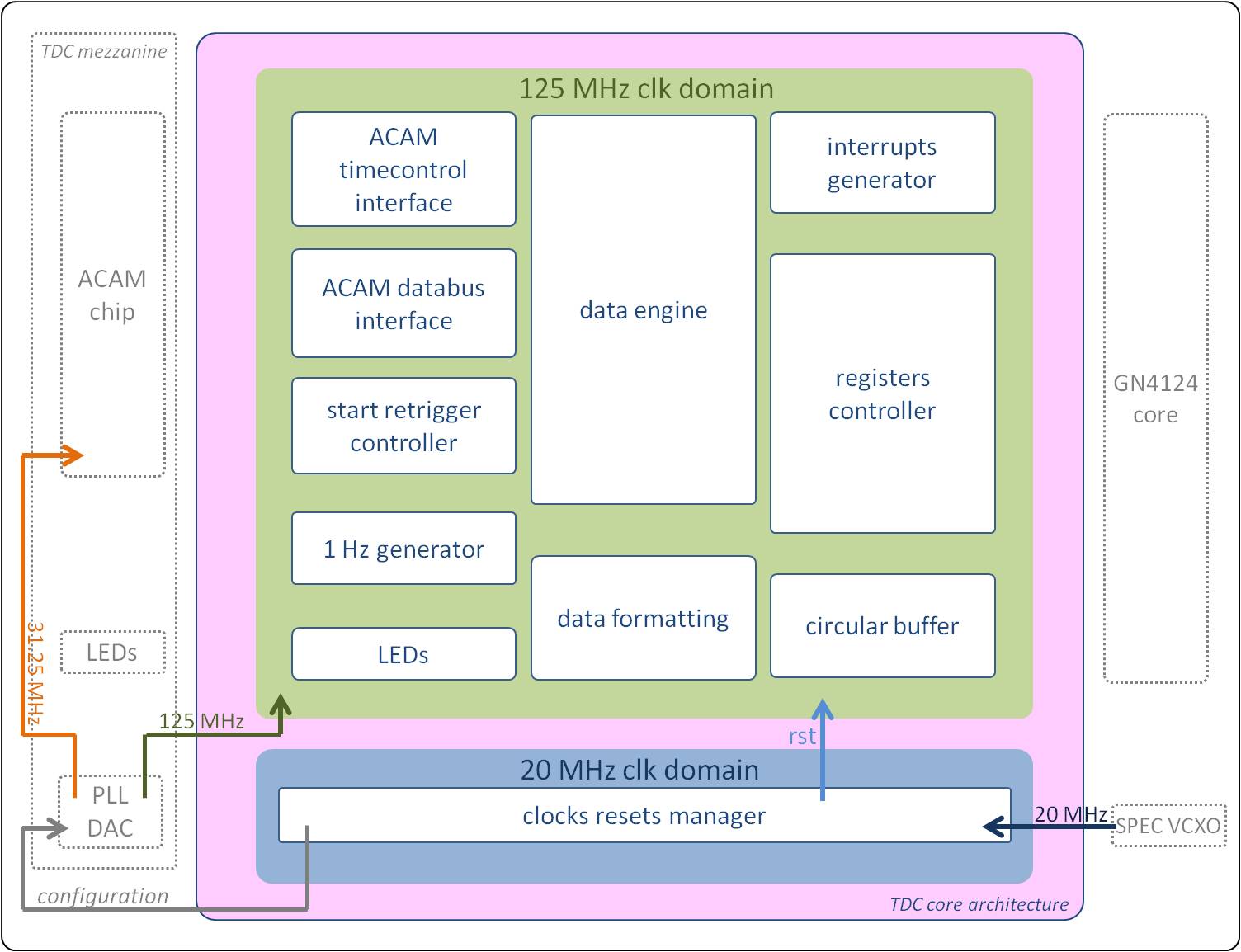


Figure 4: TDC core clock domains

## Timestamping concept

The ACAM chip is receiving a 31.25 MHz clock from the AD 9516 PLL. The TDC core is receiving a 125 MHz clock from the same PLL.

The ACAM chip notifies the TDC core of a new timestamp by deactivating the signal “Empty flag”. When the core sees a falling edge on the Empty flag, it starts reading from the ACAM FIFO. When the Empty flag goes back to high, no reading should take place, as that could unbalance the FIFO pointers.

The ACAM is configured in I-mode. One start pulse is sent by the FPGA and all the ACAM timestamps are referred to that start pulse. Figure 5 shows the FPGA start pulse and a stop pulse arriving later through one of the TDC mezzanine input channels.

The ACAM is programmed to retrigger every (16\*acam\_clk\_period) = 512 ns. On each timestamp the ACAM is providing three measurements:

* Start01: the difference between the start pulse and the first ACAM retrigger
* Start#: the number of retriggers
* Stop: the difference between the last retrigger and the stop pulse.

The timestamp = (Start01) + (Start# \* retrigger period) + (Stop)

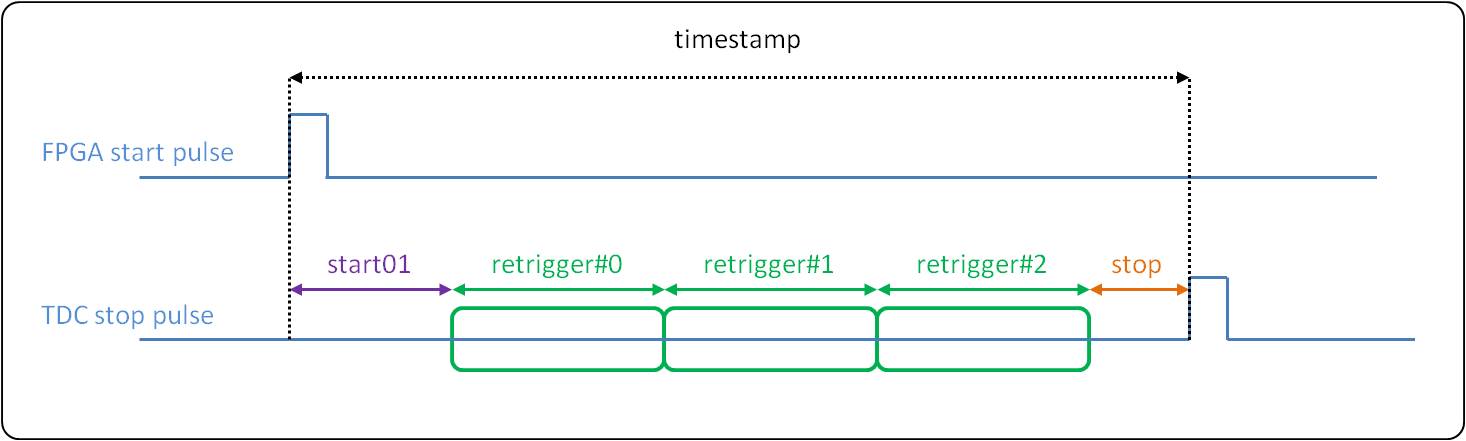


Figure 5: ACAM timestamping

According to the ACAM documentation, there is indeterminacy to whether the stop time refers to the previous retrigger or the current one. This means that any of the figures 4 or 5 could be possible.

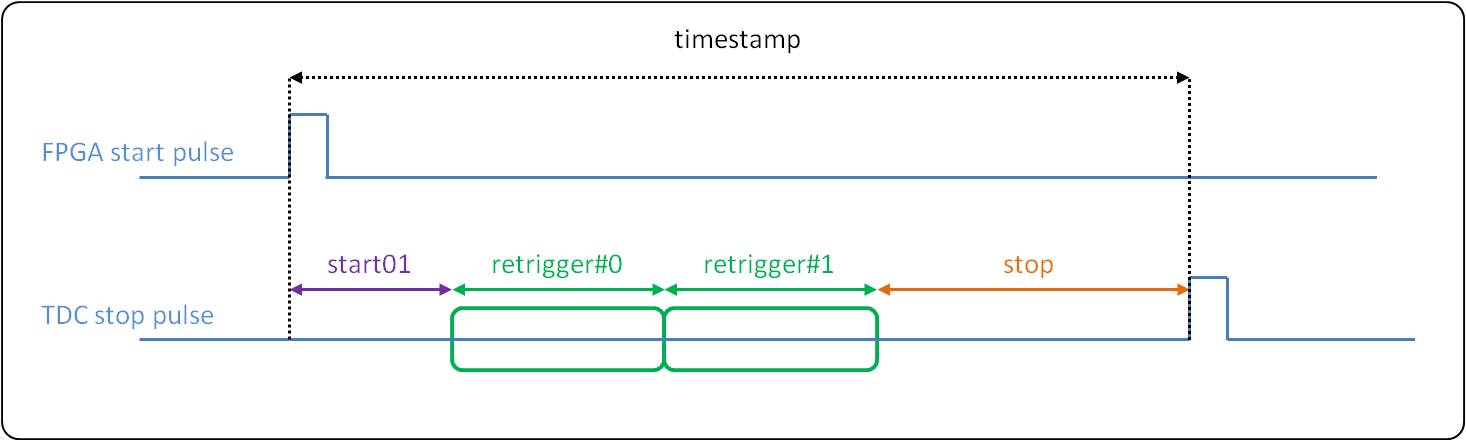


Figure 6: ACAM timestamping. Stop time refers to the previous retrigger

In this TDC application, we are only interested in differences between stop pulses, therefore the constant value “Start01” is irrelevant in the calculations. The following figure shows a simple scenario. timestamp B – timestamp A =

[(Start01) + (Start#A \* retrigger period) + (StopA)] - [(Start01) + (Start#B \* retrigger period) + (StopB)]

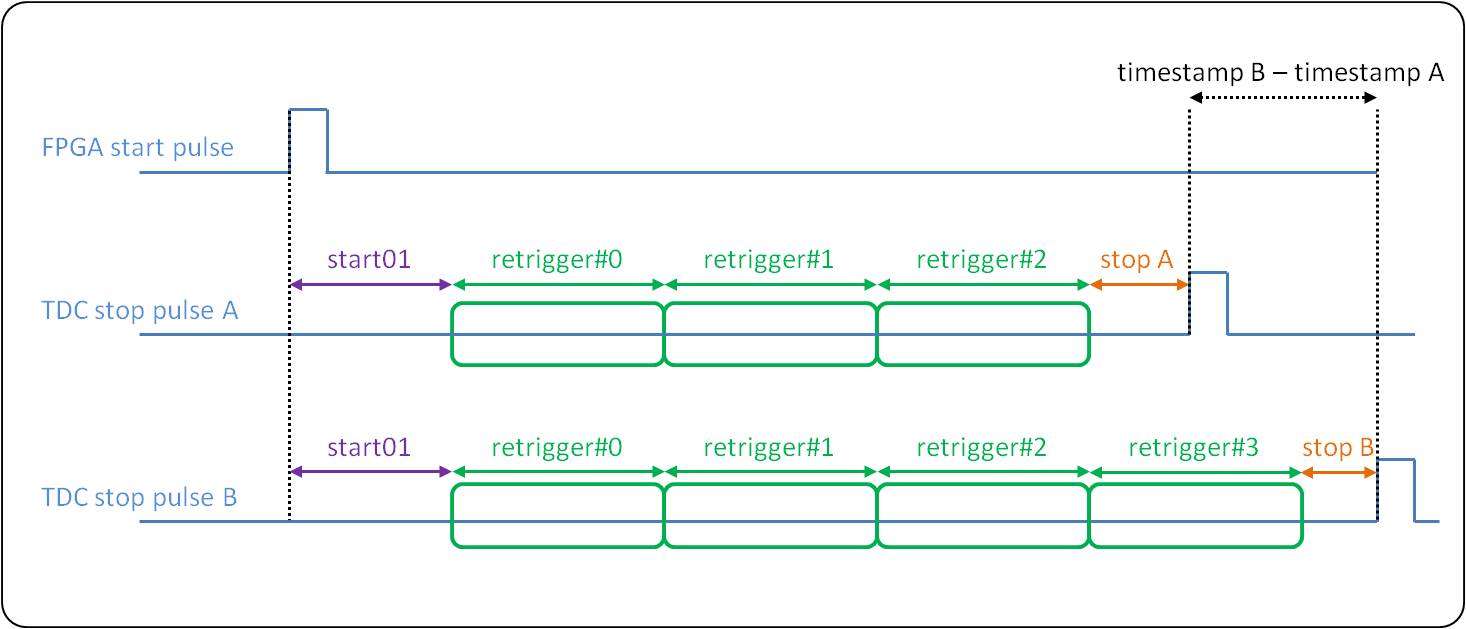


Figure 7: Time difference between pulses

The counter of the ACAM chip that counts the retriggers, Start#, is restricted to 8 bits, and can only count up to 256 retriggers = 131’072 ns. Since our application needs to measure pulses separated by any amount of time, the TDC core needs to be tracking how many times the ACAM retriggers counter overflows. To do so, the ACAM Interrupt flag has been configured to follow the highest bit of the Start#. In the TDC core, a counter called “rollover counter” is following the Interrupt flag and is counting the overflows.

Note also that in one second, there are 7629 rollovers.

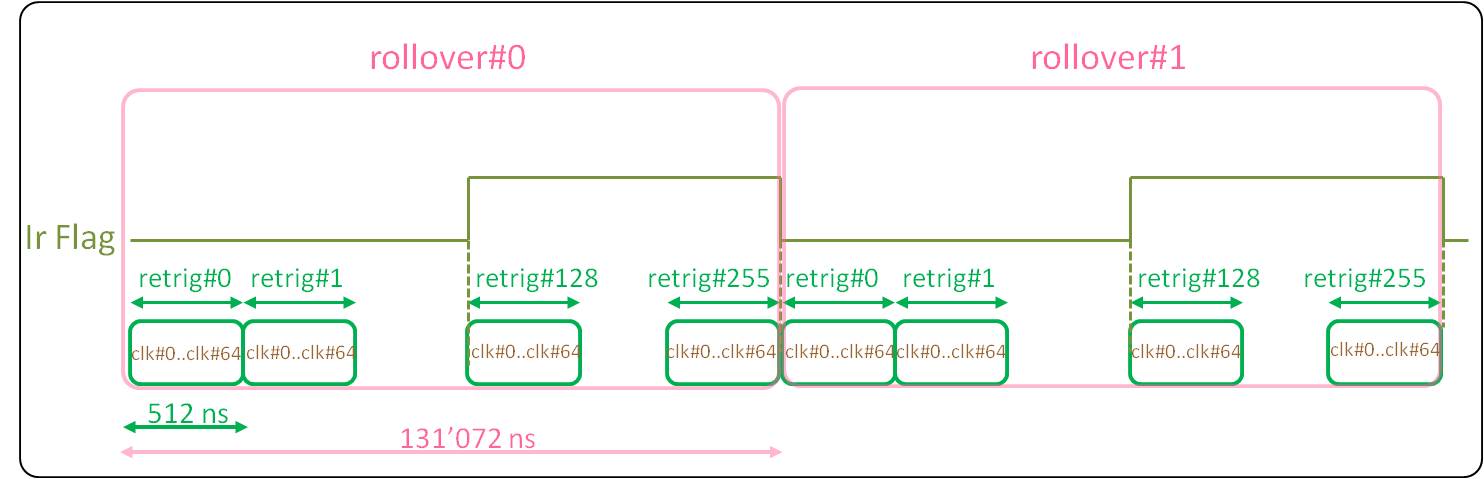


Figure 8: The TDC core is following the ACAM retriggers

In addition, the core is keeping track of the retrigger number and the number of clock cycles after the last retrigger, as Figure 8 shows.

As described in Table 1, each timestamp provided to the PCIe interface, has a UTC time component, a coarse time component and the fine time. Therefore, the ACAM retriggers have to be associated to the UTC time. The ACAM however has no knowledge of the UTC time and the arrival of a new second happens completely independently.

The “one Hz generator” unit is responsible for keeping the UTC time and it is generating a pulse once a second has been counted. When this “one Hz pulse” arrives:

* The rollover counter is reset.
* The current retrigger number is registered (retrigger#\_offset).
* The current number of clock cycles after the last retrigger is registered (clk#\_offset).

Through the retrigger#\_offset and clk#\_offset the amount of time [1] can be calculated. Figure 9 shows in detail these actions.

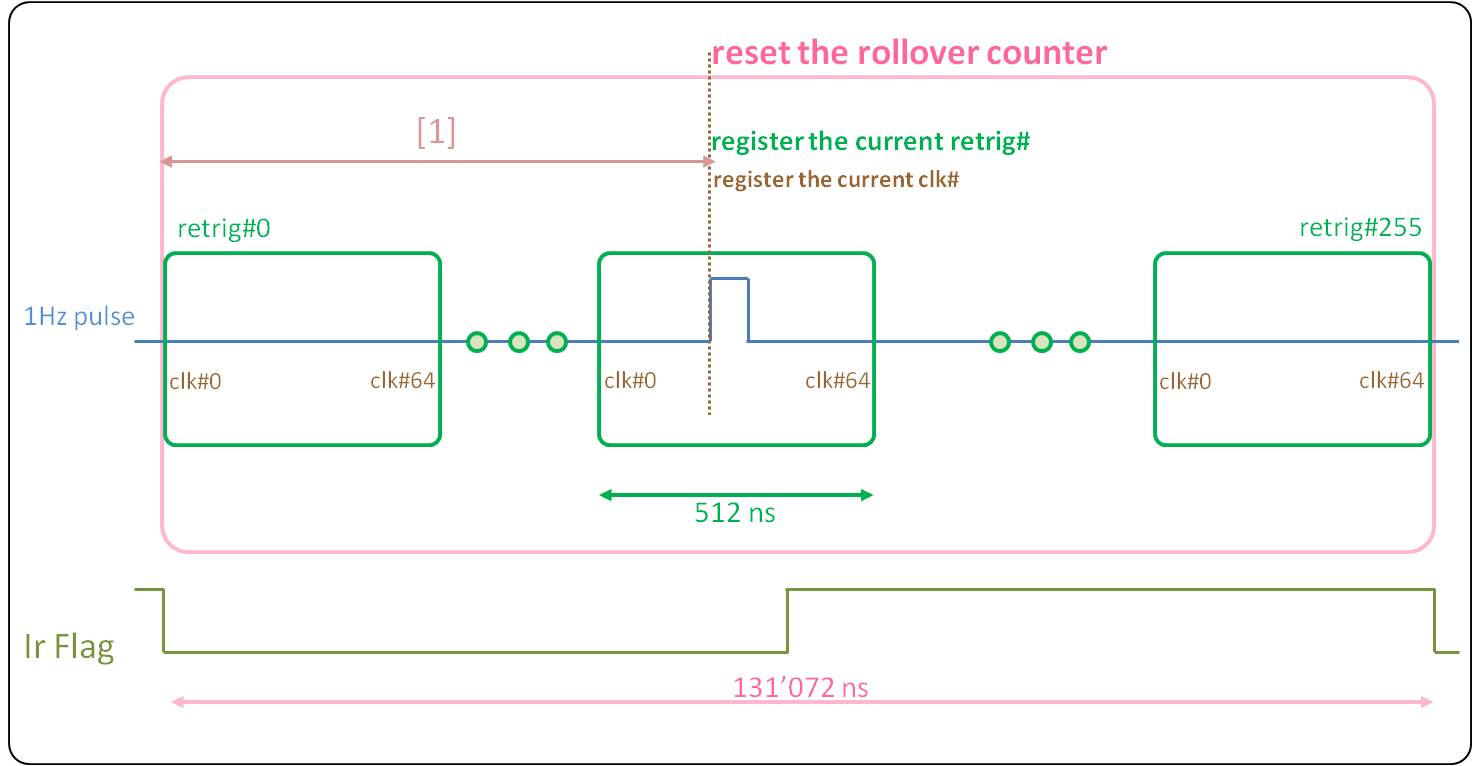


Figure 9: Actions upon the arrival of a new "1 Hz pulse"

Putting everything into the picture now, when a stop pulse arrives to any of the TDC channels, the ACAM chip registers the timestamp to its FIFO and puts the “Empty flag” to low. The TDC core receives the timestamp and in the “data formatting” unit it brings it into the form of Table 1. For that, as Figure 10 shows, the amounts of time [1’], [2] and [3] need to be calculated.

The amount [3] is exclusively provided by the ACAM chip in the form of Start# and Stop.

The amount [2] is the number of rollovers of the IrFlag between the last “one Hz pulse” and the Stop pulse. The rollover counter that has been reset upon the “one Hz pulse” arrival provides this amount.

The amount [1’] is the complementary of [1] and is calculated using the retrigger#\_offset and the clk#\_offset registered values.

This is the way the TDC core is functioning in terms of timespamps retrieval and formatting. The sections that follow describe in detail each unit of the TDC core.

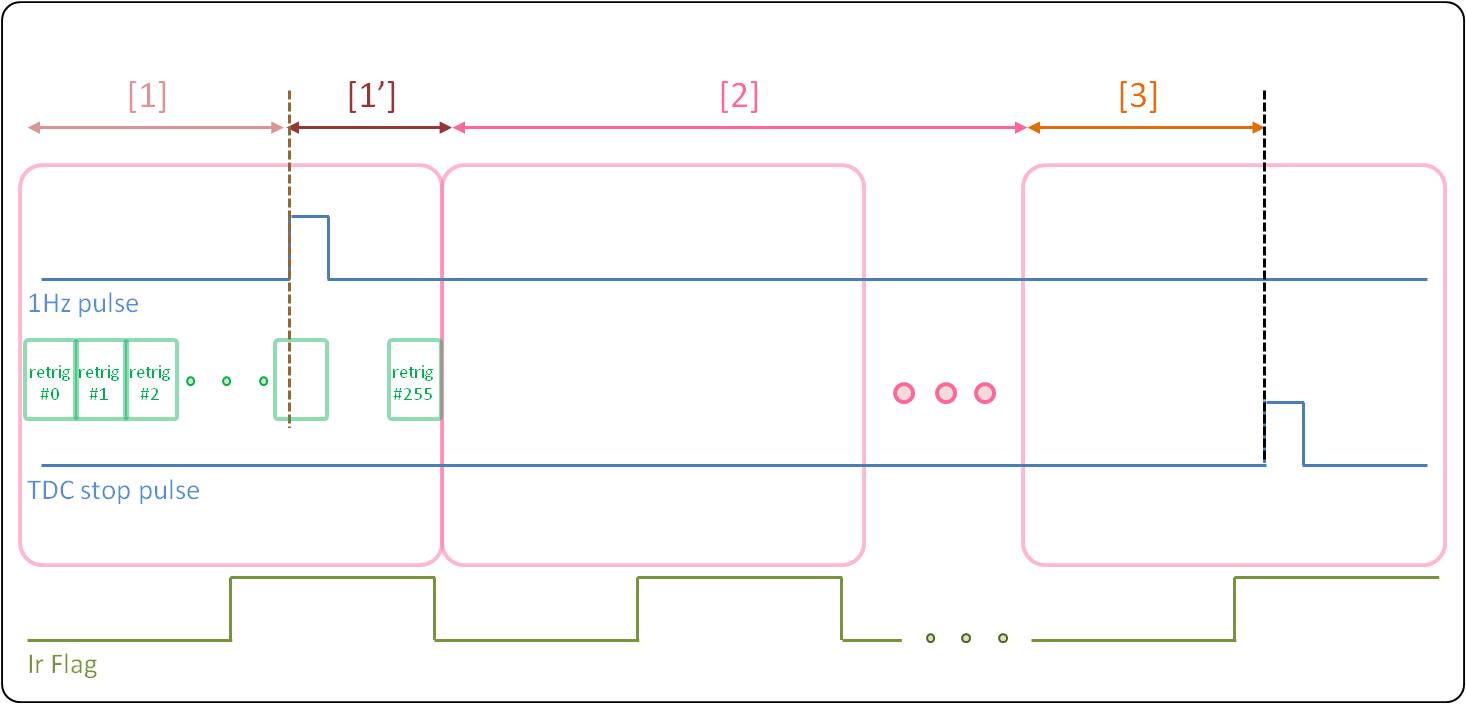


Figure 10: Essential calculations to associate each timestamp to the current UTC time

Notice that the Start pulse is not in the picture and the Start01 amount of time is not used.

## Clocks and Resets Manager

The “clocks and resets manager” unit is responsible for providing the clocking resources to the rest of the modules. It configures the PLL AD9516 and the DAC AD5662 on the TDC mezzanine board. Upon the power up or after a PCIe reset, the unit is using the 20 MHz clock on the SPEC carrier board to configure the DAC and PLL through their SPI interfaces.

The PLL is configured to provide a 125 MHz clock for the rest of the TDC core and a 31.25 MHz clock to the ACAM chip (i.e. the ACAM chip is running 4 times slower than the TDC core). The registers for the PLL configuration are hard-coded in the unit.

The DAC output is connected to a voltage-controlled-oscillator which in turn controls the PLL. Alterations on the DAC voltage change slightly the frequency of the pulses coming out of the PLL. Since the “one Hz generator” unit is counting seconds based on the 125 MHz of the PLL, the timestamps are affected by the DAC voltage. For example, the difference between two pulses separated by 9 ms can be measured as 8.999967000 ms with the DAC at 1.25 V and as 9.000000300 ms with the DAC at 1.65 V. During the calibration of each TDC mezzanine board, the optimal value for the DAC is estimated (see Section 11 on TDC calibration). The register for the DAC configuration is received through the PCIe interface. Note that extension of the TDC core with White Rabbit, will relieve the design from this dependence on the local oscillator.

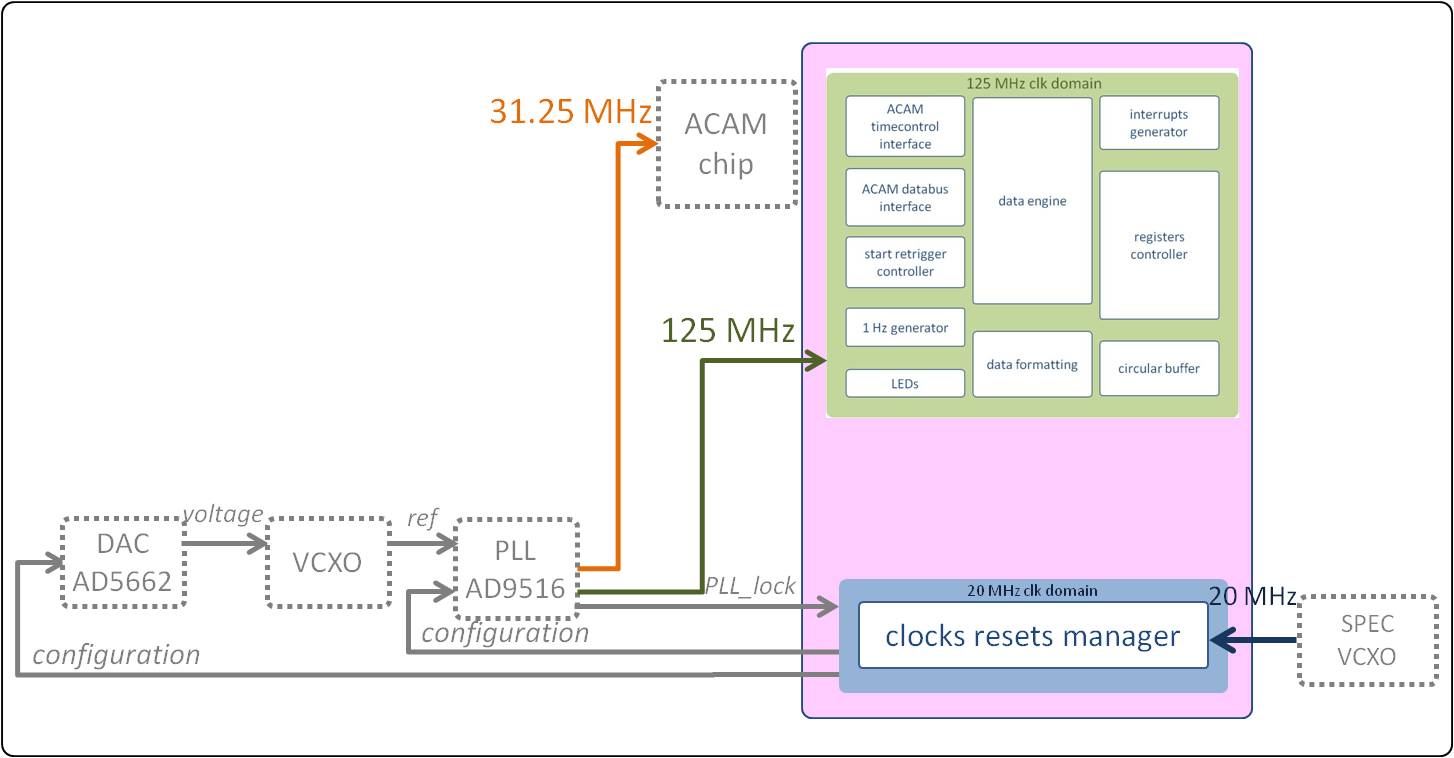


Figure 11: TDC clocks

The “clocks resets manager” is resetting the rest of the logic until the “PLL\_lock” signal is activated. The reset pulse is synchronized to the 125 MHz domain.

## Registers controller

The unit interfaces with the GNUM core for the configuration of the ACAM chip and of the TDC core. The unit implements a WISHBONE slave. All the registers are of size 32 bits, as the WISHBONE data bus.

Through **WISHBONE write cycles**, the unit receives:

* The ACAM configuration registers. Table 2 describes the registers.
* The TDC core configuration registers. Table 3 describes the registers.
* The “control register” that defines the action to be taken in the core; the register is decoded and the corresponding signals are used by the different units in the design.

Through **WISHBONE read cycles**, the unit transmits:

* The ACAM configuration registers read back from the ACAM chip. Table 5 describes these registers.
* The TDC core status registers. These are the read-only registers in Table 3.

Table 2 lists the ACAM configuration registers. The [ACAM documentation](http://www.acam.de/fileadmin/Download/pdf/English/DB_GPX_e.pdf), section 1.7 “Register Settings” describes in detail each register. Note that the registers are first transferred in the TDC core by the PCIe interface and then they have to be loaded to the ACAM chip. For that, the “control register” has to be properly activated (Table 4, bit 2 = ‘1’).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACAM Config. | R/W | Description | Address | Set Value |
| reg. 0 | R/W | Timestamping of both rising and falling edges | 0:5000 | x01F0FC81 |
| reg. 1 | R/W | Not used (channel adjustments for other modes) | 0:5004 | x00000000 |
| reg. 2 | R/W | I-mode selection | Disabling of unused channels | 0:5008 | x00000E02 |
| reg. 3 | R/W | Not used (resolutions and tests for other modes) | 0:500C | x00000000 |
| reg. 4 | R/W | Start retriggers set to 16 | Resets | 0:5010 | x0200000F |
| reg. 5 | R/W | External start retrigger OFF | Offset set to 2.000 | 0:5014 | x000007D0 |
| reg. 6 | R/W | Not used for the moment (load flags) | 0:5018 | x00000003 |
| reg. 7 | R/W | PLL values: RefClkDiv=7 | HSDiv=234 | PhaseNeg | 0:501C | x00001FEA |
| reg. 11 | R/W | Error flag configured on the 8 Hit FIFOs | 0:502C | x00FF0000 |
| reg. 12 | R/W | Interrupt flag configured on the Start# overflow | 0:5030 | x04000000 |
| reg. 14 | R/W | 16-bit mode control | 0:5038 | x00000000 |

Table 2: ACAM configuration registers

Table 3 lists the local configuration registers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | R/W | Description | Address | Typical Value |
| Starting UTC time | R/W | updated by the PCI-e; initialized after a reset | 0:5080 |  |
| Inputs enable | R/W | [bits 4 downto 0]: termination enable for each input  [bit 7]: general enable input | 0:5084 | x0000009F |
| IRQ tstamp thresh | R/W | [bits 7 downto 0]: an interrupt is issued if the number of accumulated timestamps since the last irq exceeds this threshold | 0:5090 | x000000FF  = full mem |
| IRQ time thresh | R/W | an interrupt is issued if this amount of time has passed after the last irq and at least a timestamp has been registered. The threshold value is: register value\* 8 ns | 0:5094 | x0001E848  = 1 ms |
| DAC word | R/W | [bits 24 downto 0]: word to be sent to the mezz DAC | 0:5098 | x0000A8F5  = 1.65 V |
| Current UTC time | R | calculated by the core according to the local 125 MHz clk | 0:50A0 |  |
| Circular buffer wr pointer and  Da Capo counter | R | [11 downto 0]: number of 8-bit-words to be read from the circular buffer (= number of 128-bit-timestamps \* 16)  [32 downto 12]: number of times the circular buffer has been overwritten | 0:50A8 |  |
| Control Register | W | Commands the main core state machine | 0:50FC |  |

Table 3: TDC core local registers

Table 4 describes the control register, which defines the state of the “data engine” state machine. Only one bit at a time can be activated since each bit is carrying a command. Upon a WISHBONE write of the control register by the PCIe, a 1-tick-long control pulse is generated defining the action to be taken (ex: activate\_acquisition\_p, reset\_acam\_p, read\_acam\_fifo1\_p..etc); then the register is cleared.

|  |  |  |  |
| --- | --- | --- | --- |
| Control Register Bit | Action Description | Control Register Value | |
| Bit 0 | Activate acquisition | | x00000001 |
| Bit 1 | De-activate acquisition | | x00000002 |
| Bit 2 | Load ACAM configuration registers | | x00000004 |
| Bit 3 | Read back ACAM configuration | | x00000008 |
| Bit 4 | Read ACAM status register | | x00000010 |
| Bit 5 | Read ACAM IFIFO 1 register | | x00000020 |
| Bit 6 | Read ACAM IFIFO 2 register | | x00000040 |
| Bit 7 | Read ACAM Start01 register | | x00000080 |
| Bit 8 | Reset ACAM chip | | x00000100 |
| Bit 9 | Load UTC time | | x00000200 |
| Bit 10 | Clear Write pointer and Da Capo counter | | x00000400 |
| Bit 11 | Configure DAC and reset the core | | x00000800 |

Table 4: The bits of the control Register

describes the registers read back from the ACAM chip. To initiate a reading of this set of registers, the “control register” bit 3 needs to be activated. Note that this set of registers includes the configuration registers listed in plus the ACAM Read-only registers Start01 and Interface FIFOs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACAM Config.  Read-back | R/W | Description | Address | Typical Value |
| reg. 0 | R | Timestamping of both rising and falling edges | 0:5040 | xC1F0FC81 |
| reg. 1 | R | Not used (channel adjustments for other modes) | 0:5044 | xC0000000 |
| reg. 2 | R | I-mode selection | Disabling of unused channels | 0:5048 | xC0000E02 |
| reg. 3 | R | Not used (resolutions and tests for other modes) | 0:504C | xC0000000 |
| reg. 4 | R | Start retriggers set to 16 | Resets | 0:5050 | xC200000F |
| reg. 5 | R | External start retrigger OFF | Offset set to 2.000 | 0:5054 | xC00007D0 |
| reg. 6 | R | Load flag levels at maximum | 0:5058 | xC00000FC |
| reg. 7 | R | PLL values: RefClkDiv=7 | HSDiv=234 | PhaseNeg | 0:505C | xC0001FEA |
| reg. 8 | R | Interface FIFO 1 | 0:5060 | Start# and Stop |
| reg. 9 | R | Interface FIFO 2 | 0:5064 | Start# and Stop |
| reg. 10 | R | Start01 | 0:5068 | Start01 |
| reg. 11 | R | Error flag configured on the 8 Hit FIFOs | 0:506C | xC0FF0000 |
| reg. 12 | R | Interrupt flag configured on the Start# overflow | 0:5070 | xC4000000 |
| reg. 14 | R | 16-bit mode control | 0:5078 | xC0000000 |

Table 5: ACAM read-back registers

## Data Engine

According to the value of the “control register” and the generated control pulses, the “data engine” unit is managing:

* The writing of the ACAM configuration.
* The reading back of the ACAM configuration/status registers.
* The timestamps' acquisition from the ACAM.

Figure 12 shows the states structure of the state machine. The default state is “inactive”.

From the “inactive” state the FSM jumps to “load ACAM configuration” upon the activation of the “control register” bit 3. In this state, the set of registers described in Table 2 are transferred one by one to the ACAM chip. The “data engine” is providing the values of the registers (as received by the PCIe interface in the “registers controller” unit) as well as the ACAM addresses to the “acam databus interface” unit that is actually implementing the interface with the ACAM chip. Figure 13 shows the flow of actions for one register. When all the eleven registers have been transferred, the FSM jumps back to the “inactive” state. Note that the configuration registers need to be loaded before setting the “control register”.

Similarly, the “read back ACAM configuration” state initiates readings of the set of registers described in Table 5. When all the fourteen registers have been read, the FMS goes back to the “inactive” state

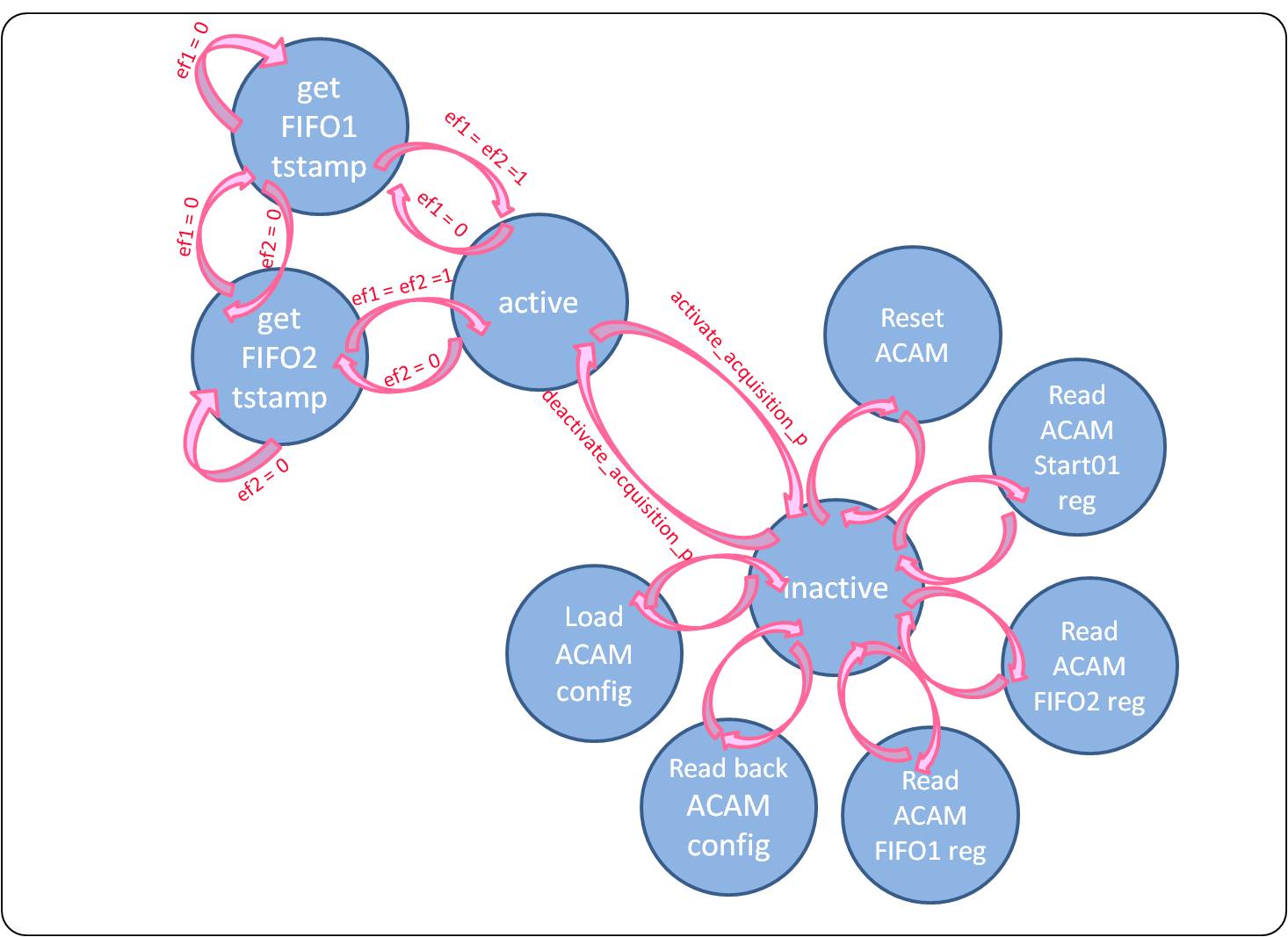


Figure 12: Data engine main FSM

Each one of the states “read ACAM FIFO1 register”, “read ACAM FIFO2 register” and “read ACAM Start01 register” initiates a single reading of the corresponding register (read-back reg 8, 9 and 10 respectively. See Table 5).

Finally the state “reset ACAM” initiates the writing of a particular word to the ACAM configuration register 4 (see Table 2) that resets the chip.

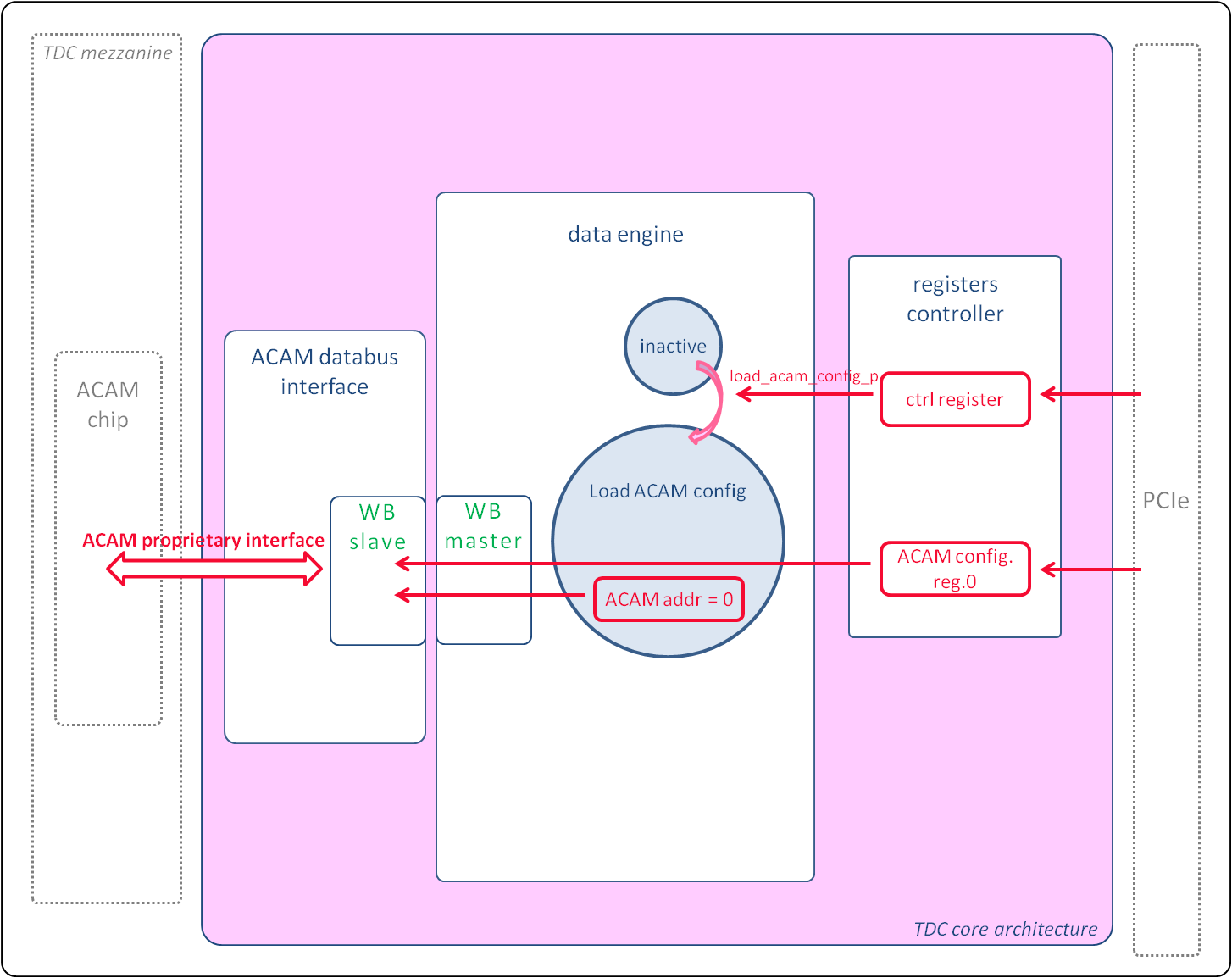


Figure 13: Snapshot of the transfer of one configuration register to the ACAM chip

In acquisition mode (“control register” bit 0 = ‘1’) the unit monitors permanently the ACAM Empty flags (ef1, ef2) and retrieves the timestamps by reading the ACAM read-only registers 8 or 9 (see Table 5). As described in Section 2, the register Start01 is not read. The timestamps are received by the “data formatting” unit for the formatting and are finally stored in the “circular buffer”. Figure 14 shows the sequence of events upon the Empty flag deactivation.

The maximum speed with which the ACAM can be retrieving timestamps is 31.25 MHz. If the ACAM is receiving more timestamps than that, then the ACAM Error flag is raised and the TDC core is issuing an interrupt. For the TDC core to be able to keep up with the maximum speed of 31.25 MHz (so that no timespamp gets lost by the core) it has to retrieve, format and store a timestamp within 4 cycles of the 125 MHz clock (4 \* 31.25 = 125!).

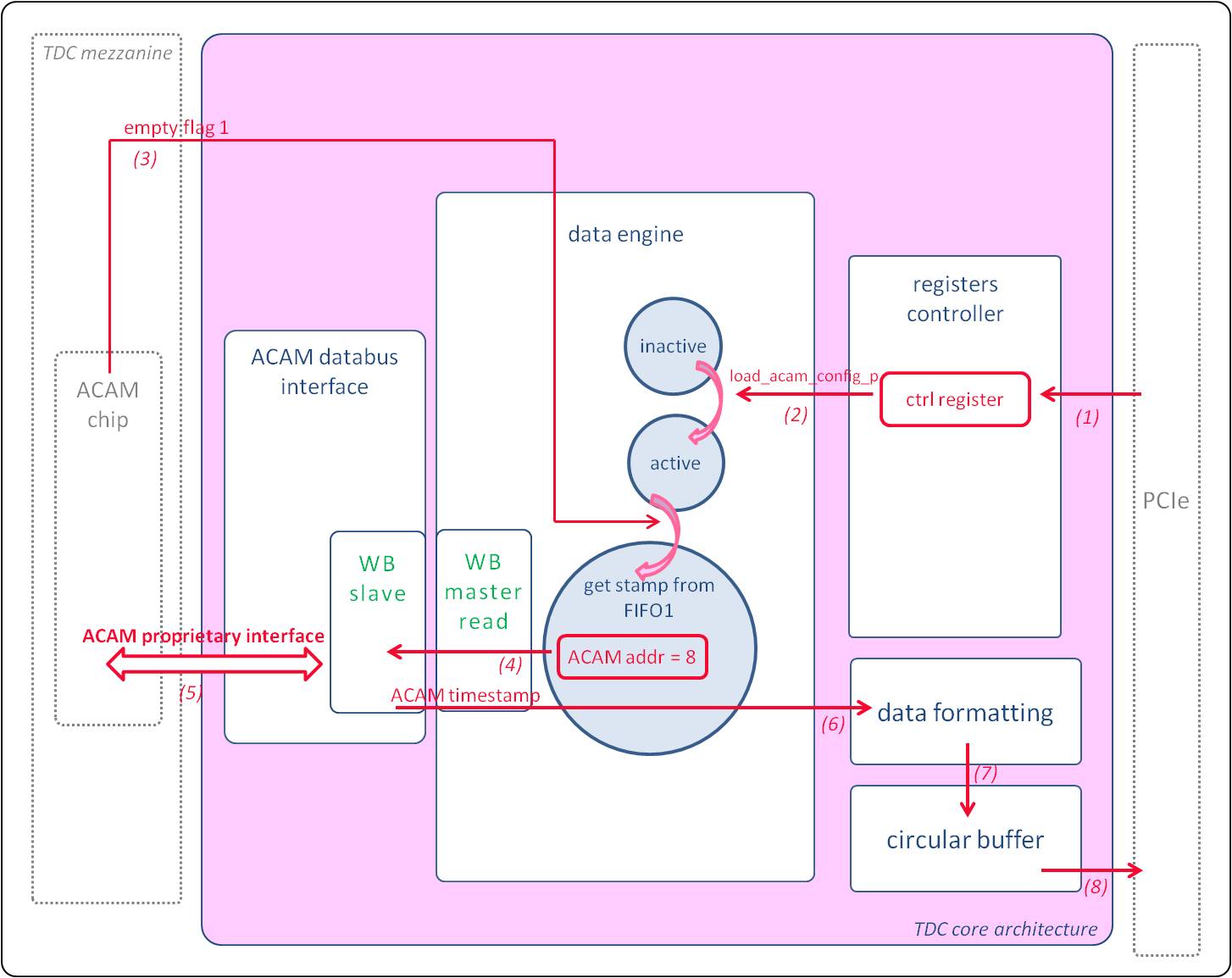


Figure 14: Snapshot of the transfer of a timestamp from the ACAM to the circular buffer. The numbers below the arrows indicate the sequence of events.

## ACAM databus interface

The unit interfaces with the ACAM chip pins for the configuration of the registers and the acquisition of the timestamps. The ACAM proprietary interface is converted to a WISHBONE classic interface, with which the unit communicates with the “data\_engine” unit. The WISHBONE master is implemented in the “data\_engine” and the slave in the “acam\_databus\_interface” unit.

Note that the TDC mezzanine board has 5 input channels. The ACAM chip is treating channels 1 to 4 in the FIFO1 and channel 5 in the FIFO2.

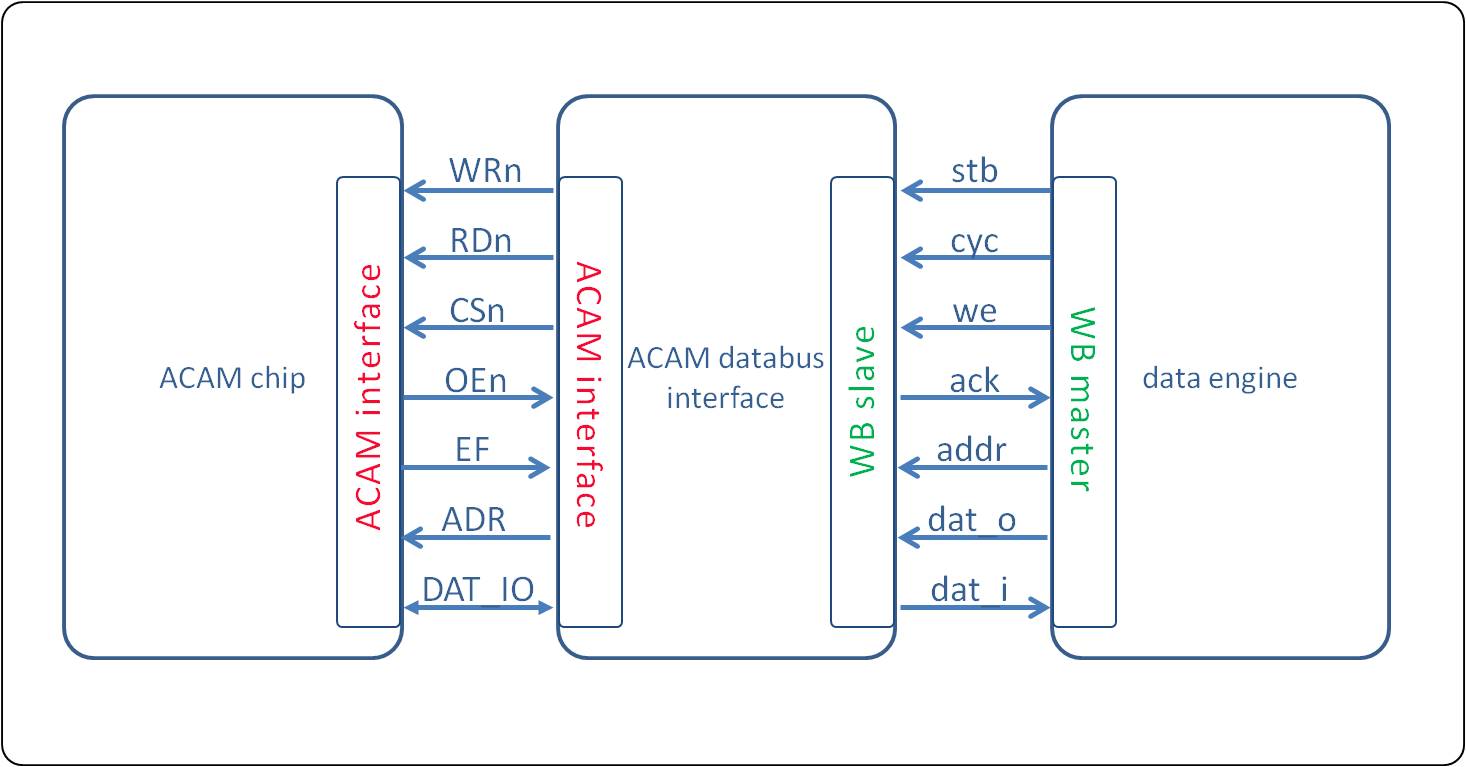


Figure 15: Interfaces between the "data engine", "ACM databus interface" units and the ACAM chip

Here is the description of the ACAM proprietary interface for a read operation:

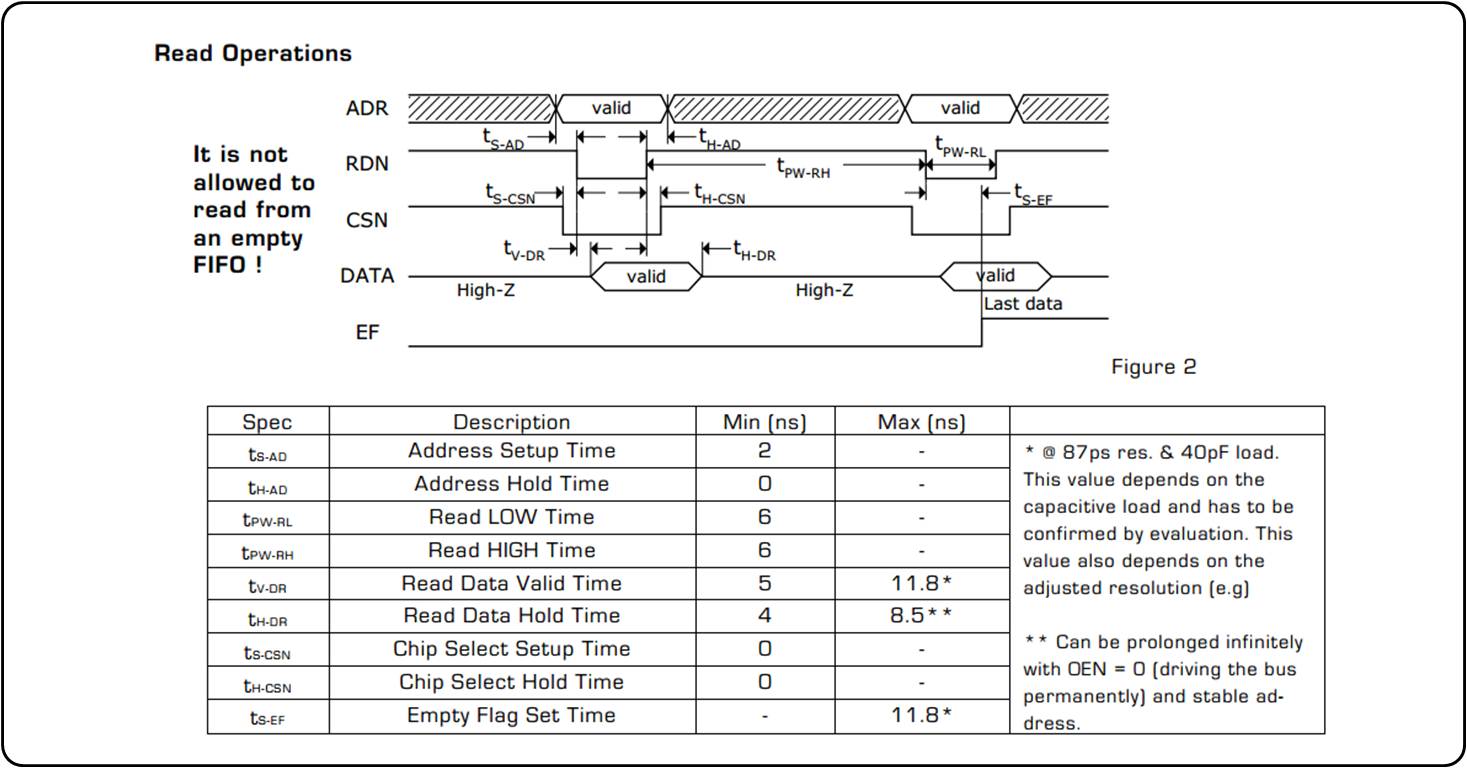


Figure 16: The ACAM proprietary interface

Special attention needs to be put on the ACAM Empty Flag maximum set time of Figure 16. A violation of this time, by initiating a new reading before the Empty Flag has been updated, could translate to reading from an empty FIFO and would unbalance the ACAM FIFO pointers.

In order for the “ACAM databus interface” unit to be able to keep retrieving timestamps from the ACAM at the ACAM's maximum speed (31.25 M timestamps/ sec), it needs to complete one retrieval per 4 \* clk cycles = 4 \* 8 ns = 32 ns. To achieve that it is allowing 16 ns from the moment it activates the RDN until it checks the ACAM Empty flag signal and decides whether to continue reading or not. ACAM's specification defines that the maximum Empty flag set time is 11.8 ns; this allows for >4 ns for the signals routing. To make sure this constraint is met, the **Xilinx design map option "Pack IO Registers/Lathes into IOBs" should be enabled**.

## Circular buffer

The unit implements a Dual port RAM. Port A is attached to a classic WISHBONE interface, where the “data formatting” unit is writing timestamps. Port B is connected to the pipelined WISHBONE interface of the GN4124 core DMA. The “data formatting” unit is only writing in the RAM and the GN4124 core is only reading from it.

The “data formatting” unit is writing timestamps of size 128 bits each (see Table 1). From port A the memory is of size: 256 \* 128 bits. The GN4124 core is reading 32-bit words. Readings take place using pipelined WISHBONE interface. From this side the memory is of size: 1024 \* 32.

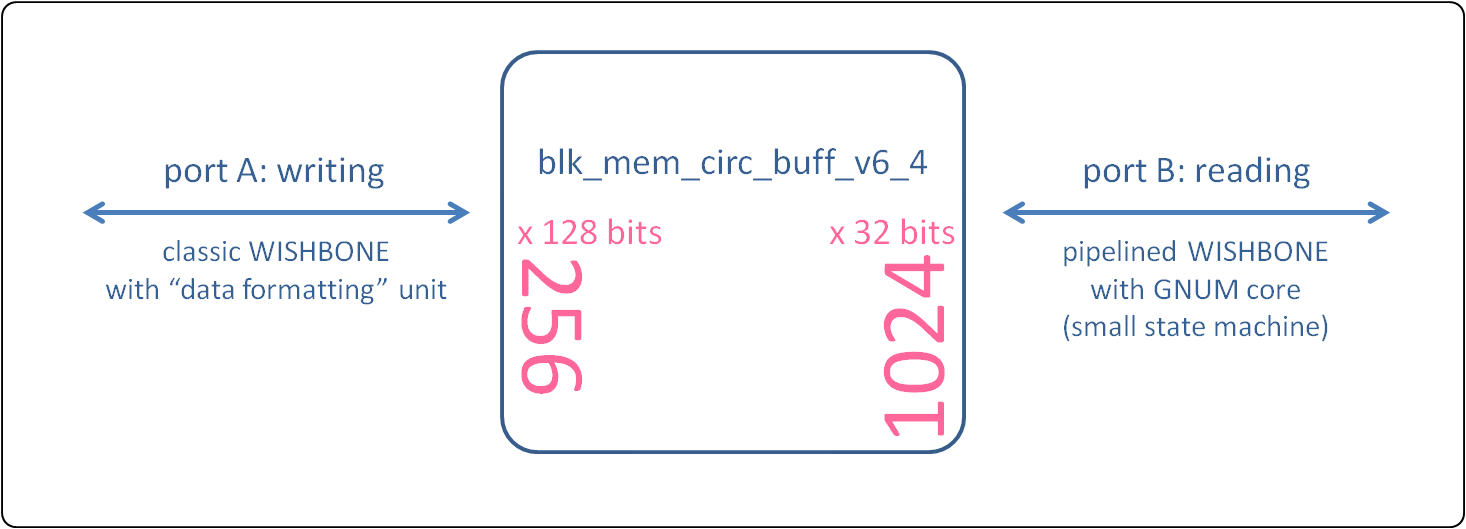


Figure 17: The circular buffer

The GN4124 core is reading words of 32-bits, but the addressing is per byte. This is why the “write pointer” register (Table 3) is providing the number of bytes to be read (=number of timestamps\*16).

## One Hz Generator

The unit implements two counters. The first one counts seconds using the 125 MHz clock. When a second has passed a 1-tick-long pulse is generated. The second counter is used to count the amount of those 1-tick-long pulses. The initial value of this second counter is zero, after a reset or after the power-up. A different initial value can be given through the PCIe register “starting utc time”. To this initial value a +1 is added after each second.

Finally, the current value of the counter can be read back through the PCIe register “local utc time”.

## Start Retrigger Controller

The “start retrigger controller” unit provides the main components for the calculation of the "coarse time" of the final timestamps (see Table 1). These components are sent to the “data\_formatting” unit where the actual coarse time calculation takes place. Figure 8 and Figure 9 in section 3.2 describe the main actions taken in this unit. Note again that the Interrupt flag of the ACAM chip has been configured to follow the highest bit of the Start# counter.

In more detail, three counters are implemented. The “retrigger period counter” and the “retrigger number counter” keep continuously track of the current internal start retrigger of the ACAM, in parallel to the ACAM itself. The “retrigger period counter” is counting clock periods, up to 512 ns; this is one ACAM retrigger. The counting is initiated by an ACAM Interrupt flag falling edge. The “retrigger number counter” is counting the number of ACAM retriggers, from 0 to 255, in the same way the Start# counter in the CAM chip does.

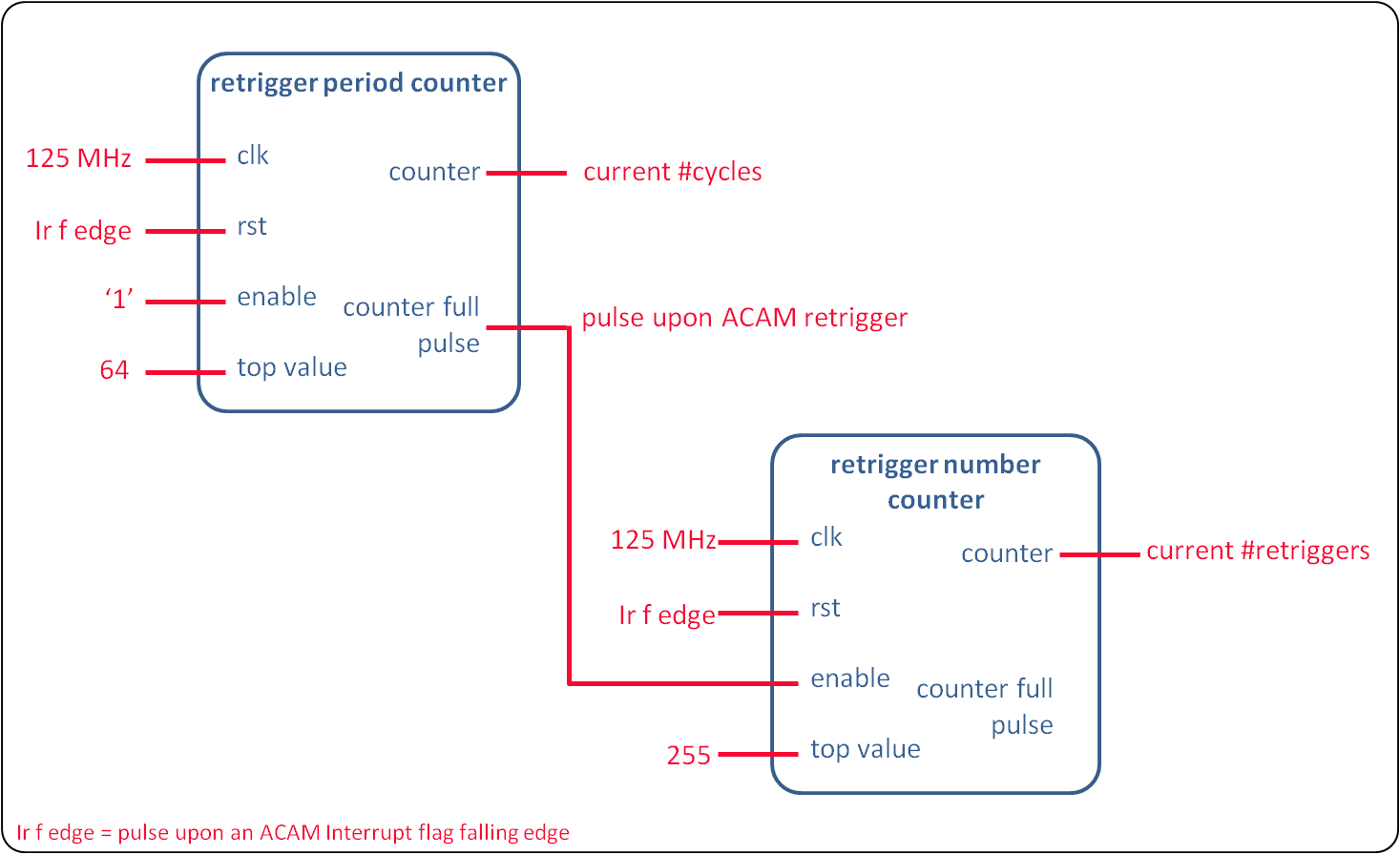


Figure 18: The “retrigger period counter” and the “retrigger number counter”

The arrival of a “one HZ pulse” means that a new second has just started and the timestamps that will come after that point should be referenced to that new second. Upon the “one Hz pulse”, the values of the “retrigger period counter” and of the “retrigger number counter” are registered. The registered values are referred to as “clk#\_offset” and “retrigger#\_offset”. As Figure 10 shows, these values represent the amount of time [1] and need to be subtracted upon the arrival of a stop pulse.

One more counter is essential. The “rollover counter” is counting the number of overflows (rollovers) of the ACAM Start# counter since the last “one HZ pulse”. In Figure 10, this represents the amount of time [2].

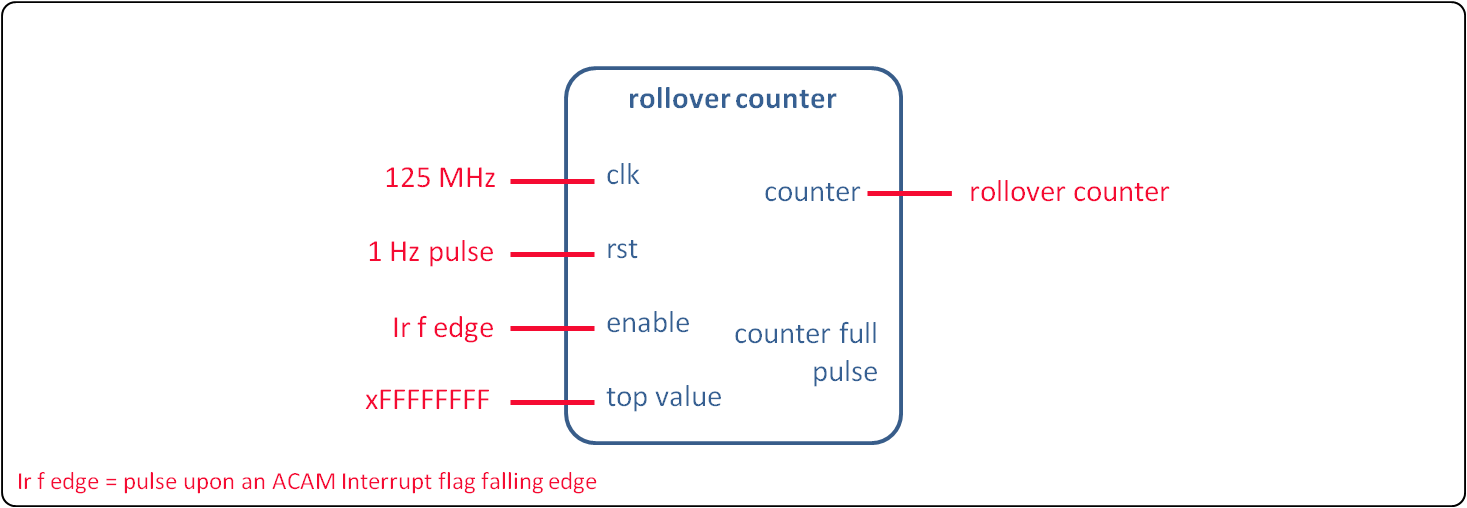


Figure 19: The “rollover counter”

Upon the arrival of a timestamp, the “data formatting” unit is using the “clk#\_offset”, “retrigger#\_offset” and the current value of the “rollover counter” for the calculations of the values to be used to arrive to a timestamp related to the current second.

## Data Formatting

The “data formatting” unit brings the ACAM timestamps to the format of Table 1. It also writes them to the “circular buffer” unit is responsible for keeping track of how many timestamps have been written.

When a timestamp arrives the UTC time, coarse time and fine time need to be calculated.

The UTC time comes directly from the “one Hz generator” unit. The timestamp will be referenced to the last UTC second.

The fine time comes directly from the Stop value of the ACAM timestamp (Figure 5).

Now, for the coarse time, i.e. amount of 8 ns cycles between the last “one Hz pulse” and the timestamp, the following calculations need to take place.

* The “roll over counter” has been counting the multiples of 256 retriggers since the last “one Hz pulse” (i.e. number of pink boxes in Figure 20).
* The “retrigger#\_offset” has kept the amount of retriggers that had preceded the arrival of the “one Hz pulse” (ie. number of green boxes inside the first pink box in Figure 20).
* The ACAM Start# indicates the amount of retriggers within the last pink box of Figure 20.

Therefore, the number of retriggers:

“#retriggers” = “current value of the rollover counter”\*256 + “Start#” – “retrigger#\_offset” [I]

Now we need to translate this number of retriggers to the amount of 8 ns cycles. The ACAM is configured to retrigger each 512 ns.

Therefore “#cycles from retriggers” = “#retriggers” \* 64.

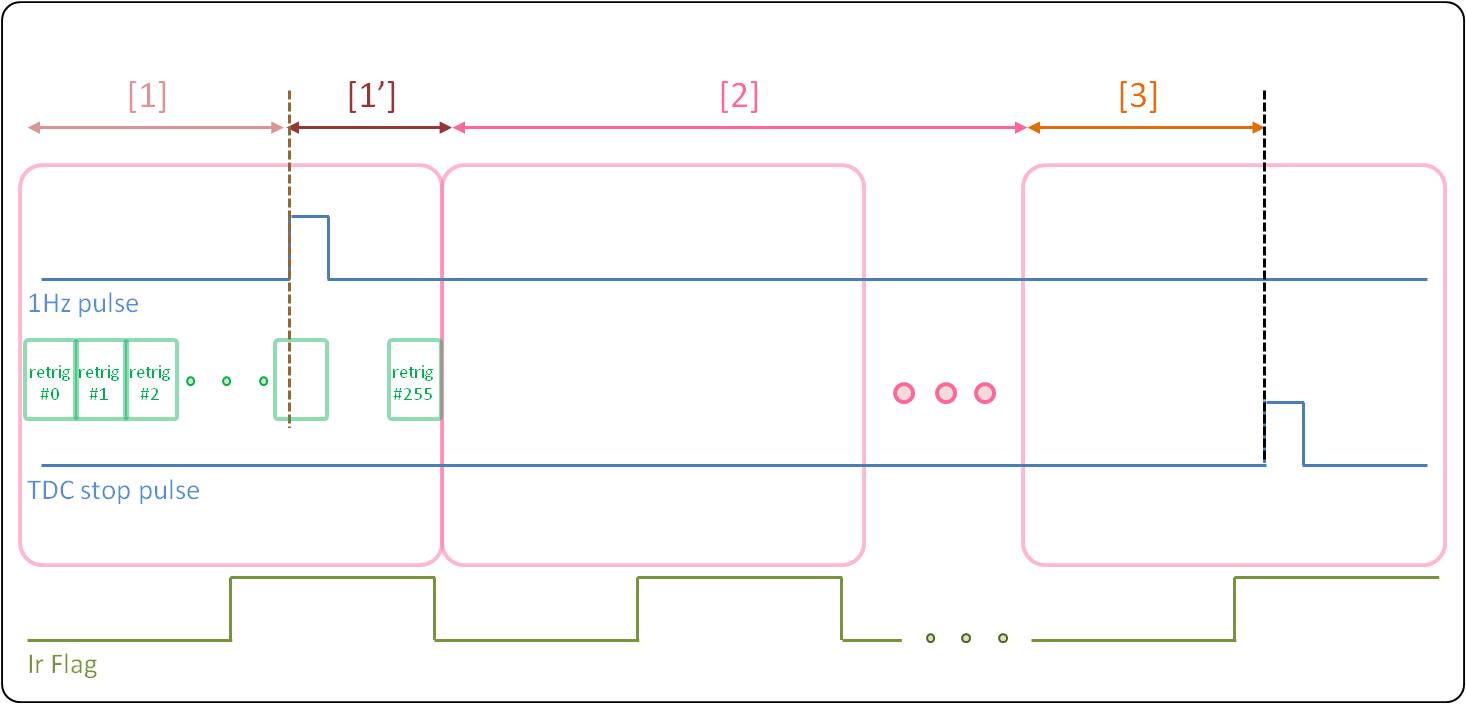


Figure 20: Data formatting calculations. Same as Figure 10

Zooming into the first pink box of Figure 20, brings us to Figure 21, which shows that to the number of 8 ns cycles between the last “one Hz pulse” and the “stop” pulse, we need to add the ones described by the “clk#\_offset”.

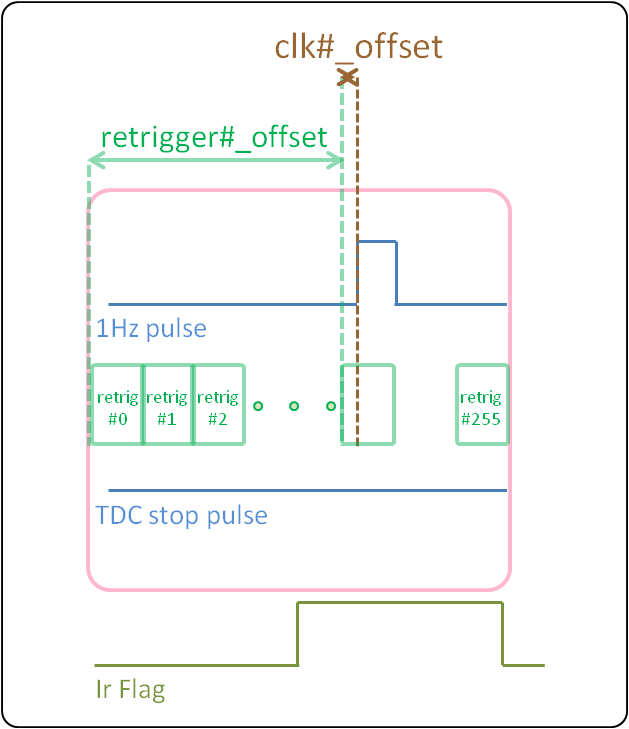


Figure 21: Zoom into Figure 20 first pink box

Finally coarse time = “#cycles from retriggers” + “clk#\_offset”.

Unfortunately, there are few exceptions in the calculation above.

The first exception comes from the marginal case where the “one Hz pulse” and the “stop” pulse arrive within the same retrigger. In this case equation [I] equals to zero! The “current value of the rollover counter” is zero and both the ACAM and the TDC core are at the same retrigger number.

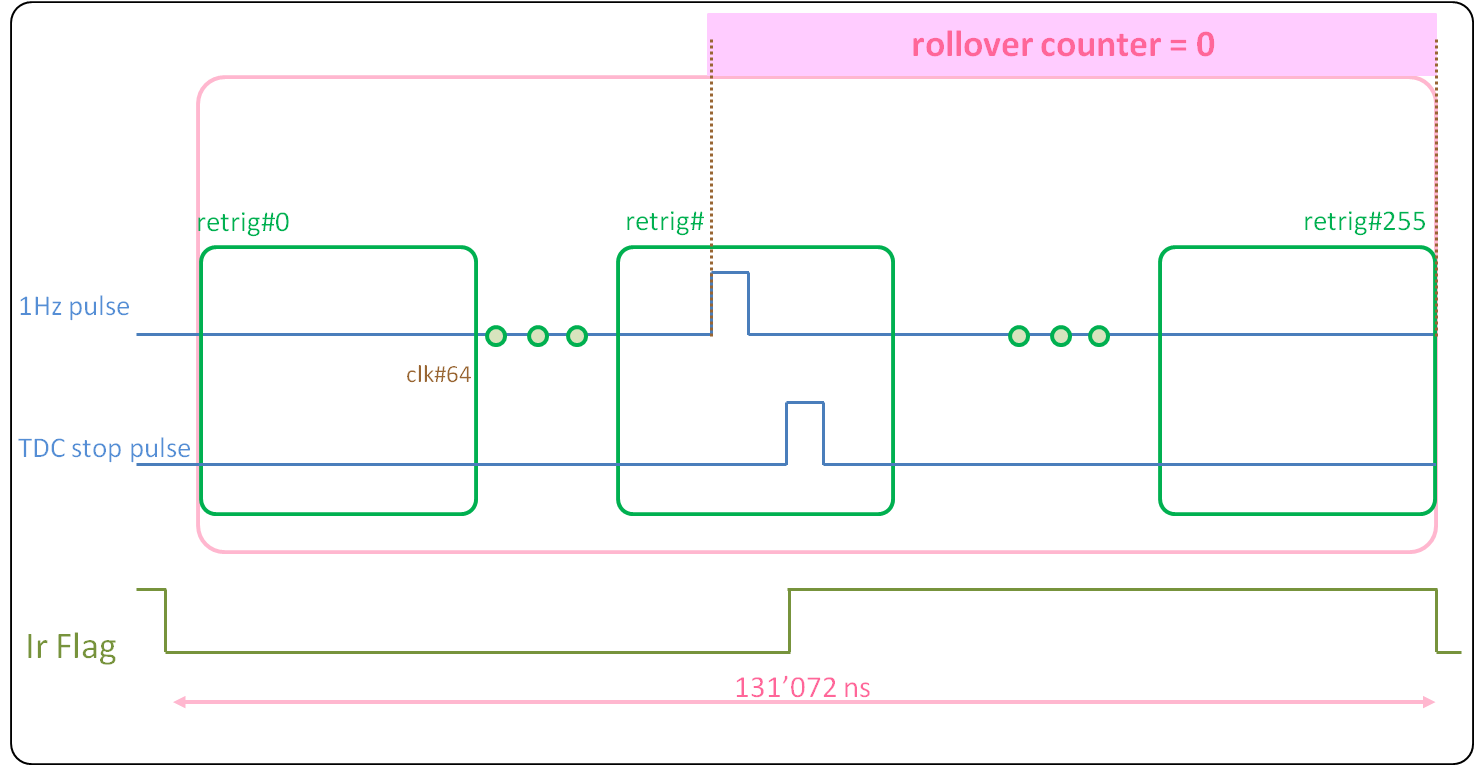


Figure 22: Marginal case where the “one Hz pulse” and the “stop” pulse arrive within the same retrigger

For this case, the timestamp needs to be referenced to the previous second. As shows, the pulse is referenced to the previous “one Hz pulse” and the amount of time [3] is provided exclusively by the ACAM.

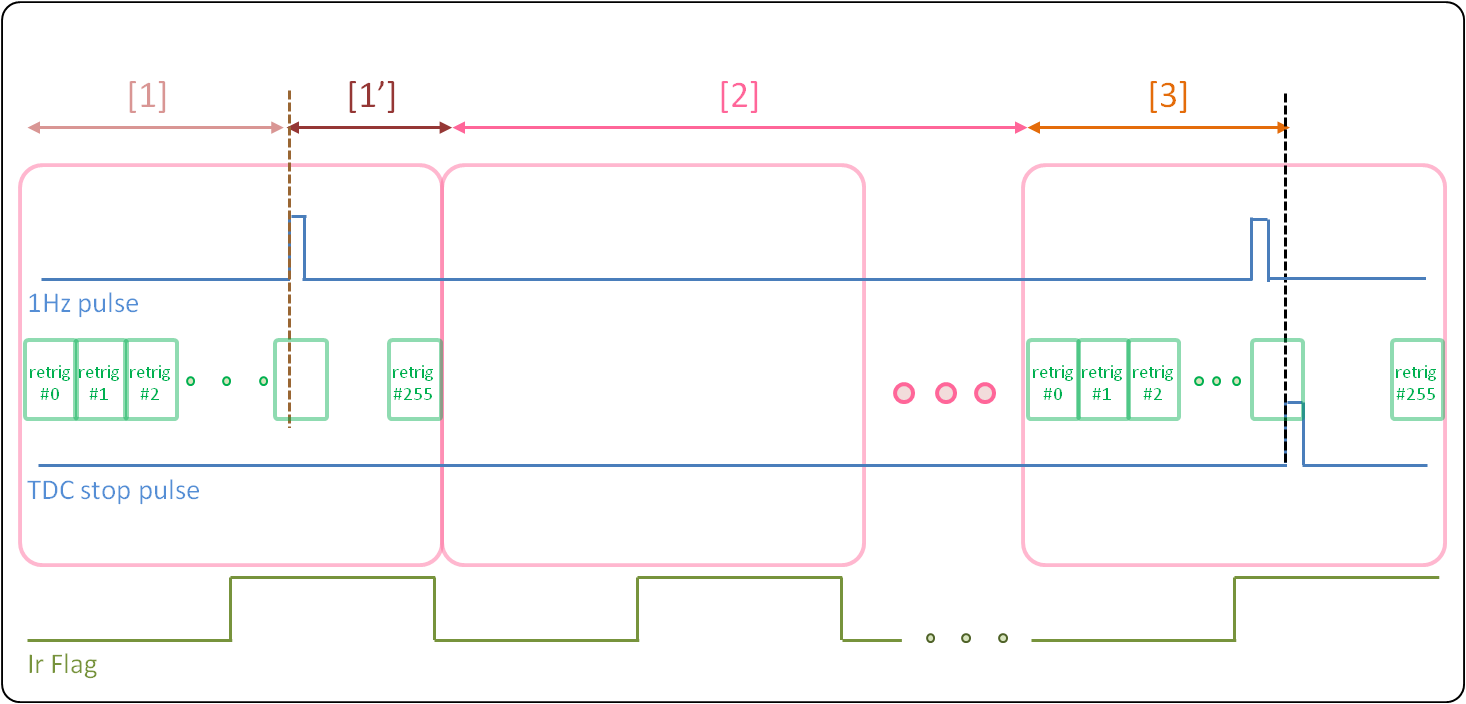


Figure 23: Calculations on the marginal case where the “one Hz pulse” and the “stop” pulse arrive within the same retrigger

Attention though needs to be put in the case described in Figure 6 where the fine time refers to the previous retrigger rather than the current one. In this case:

* “current value of the rollover counter” = 0
* “Start#” = “retrigger#\_offset” - 1
* ACAM “Stop” is > 6318 (6318 \* 81 ps = 512 ns)

If all the conditions above are valid, it means, exactly as described before, that that the last “one Hz pulse” and the “stop” pulse have arrived within the same retrigger. In this case again, as Figure 23 shows the timestamp is referenced to the to the previous “one Hz pulse”.

The last exception comes from the marginal case where a timestamp is received from the ACAM when the “rollover counter” has just been increased. In this case, it is possible that the timestamp belongs to the previous roll-over value. This is because the moment the ACAM IrFlag is taken into account in the TDC core is different from the moment the “stop” pulse has arrived to the ACAM (several 8 ns cycles are needed to empty ACAM FIFOs). In this case if the ACAM “Start#” of the received timestamp is close to the upper end (close to 255) and on the moment of processing in the “data formatting” unit the IrFlag has recently increased, that means that in equation [I], the “current value of the rollover counter” - 1 needs to be considered.

## Interrupts generator

The “interrupts generator” is responsible for the generation of 1-tick-long pulses in any of the following 3 cases:

* when the amount of timestamps written in the “circular\_buffer”, since the last interrupt or since the startup of the acquisition, exceeds the PCIe settable threshold irq\_tstamp\_threshold
* when some timestamps have been written in the circular\_buffer (>=1 timestamp) and the amount of time passed since the last interrupt or since the acquisition startup, exceeds the PCIe settable threshold irq\_time\_threshold
* when the ACAM raises the Error flag; this means that the ACAM Hit FIFOs have been receiving pulses with a frequency > 31.25 MHz.

The pulses are retrieved by the “irq\_controller” core (see Figure 2) which interfaces with the GN4124 core interrupt input.

## ACAM timecontrol interface

Upon the activation of the acquisition through the “control register” bit 0 (see Table 4), the unit generates a 16 ns long pulse that represents the ACAM start pulse (see Figure 5).

## LEDs

The LEDs manager is responsible for the generation of the signals that drive the front panel LEDs on the TDC mezzanine. There are 6 orange LEDs on the front panel of the TDC mezzanine board.

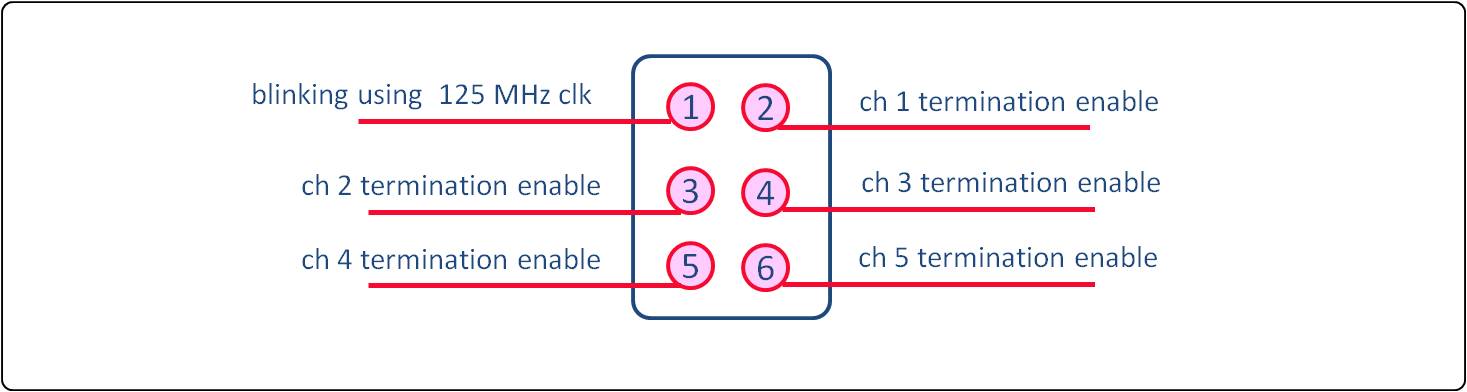


Figure 24: TDC mezzanine board front panel LEDs

# Interrupt Controller

The purpose of the “interrupt controller” unit is to concentrate several interrupt sources into one interrupt request line. It has three interrupt inputs, as described in Section 3.11 and one interrupt request output. It also has one interrupt enable mask register and one interrupt source register.

Each time a valid rising edge is detected on one of the inputs, a pulse is generated on the interrupt request output. The interrupt request output is connected to the GPIO 8 of the GN4124 chip. Note that the GN4124 chip must be configured to generate a MSI when a rising edge is detected on GPIO 8.

A rising edge is valid if the corresponding bit in the interrupt enable mask register is set. When a valid rising edge is detected, the corresponding bit in the interrupt source register is set as well. This indicates to the host which source caused the interrupt. To clear a bit in the interrupt source register, a ’1’ must be written to it.

# Mezzanine 1-Wire

The 1-Wire master controls the “DS18B20 thermometer and unique ID” chip located on the mezzanine board. The unit is based on an [OpenCores](http://opencores.org/project,sockit_owm) design.

The unit is clocked by the 125 MHz clock.

The dividers configuration are CDR\_N=624 and CDR\_O=124.

CDR\_N = f\_sys \* 5E-6 - 1

CDR\_O = f\_sys \* 1E-6 - 1

# Mezzanine I2C

The I2C master accesses the [24AA64 64Kb EEPROM](http://ww1.microchip.com/downloads/en/devicedoc/21189f.pdf) memory chip located on the mezzanine board. This memory is mandatory as specified in the FMC standard (VITA 57.1). It is connected to the system management I2C bus, also specified in the FMC standard. The unit is based on an [OpenCores](http://opencores.org/project,i2c) design.

The unit is clocked by the 125 MHz clock. For a SCL clock of 100 kHz, the prescaler configuration is PRESCALER=249.

PRESCALER = f\_sys / (5 \* f\_scl) – 1

The I2C slave address is 0x50.

# Carrier 1-Wire

Similarly to the “mezzanine 1-Wire master” of section 5, the “carrier 1-wire master” controls the “DS18B20 thermometer and unique ID” chip located on the SPEC carrier board. The unit is based on an [OpenCores](http://opencores.org/project,sockit_owm) design.

The unit is clocked by the 125 MHz clock.

The dividers configuration are CDR\_N=624 and CDR\_O=124.

CDR\_N = f\_sys \* 5E-6 - 1

CDR\_O = f\_sys \* 1E-6 – 1

# Carrier CSR

The unit contains control and status registers related to the carrier board. Table 6 describes the registers. Note that the “carrier type” field is used only for test purposes, as the carrier board identification is done through the PCI Express vendor and device ID.

|  |  |  |  |
| --- | --- | --- | --- |
| Register | Address | Bits | Description |
| Reg. 0 | 0x4C00 | [3..0] | PCB version |
| Reg. 0 | 0x4C00 | [4] | FMC PLL status, active high |
| Reg. 0 | 0x4C00 | [31..16] | Carrier type |
| Reg. 1 | 0x4C04 | [0] | FMC presence, active low |
| Reg. 1 | 0x4C04 | [1] | GNUM4124 status, active high |

Table 6: Carrier CSR registers map

# HDL repository

The TDC design is available under svn in the ohwr.org repository: <http://www.ohwr.org/projects/fmc-tdc/repository/show/hdl>

# Xilinx XC6SLX45T resources

Table 7 is describing the resources.

|  |  |
| --- | --- |
| XC6SLX45T | |
| DFF | 10% |
| LUT | 16% |
| RAM | RAMB16:10% | RAMB8: 1% |

Table 7: TDC core resources in the Xilinx XC6SLX45T

# TDC mezzanine board calibration

The TDC mezzanine board has 5 input channels. The track length from an input to the ACAM chip differs from channel to channel. In addition the input logic (clock buffers) of each channel is introducing different delays that can vary up to 2 ns. That is why it is essential to calibrate the TDC board channels. For more information, refer to the [channels calibration](http://www.ohwr.org/projects/fmc-tdc/repository/changes/documentation/TDC_calibration/Channels_calibration.pdf) documentation.

Moreover the DAC IC2 on the board controls the OSC2 voltage-controlled-oscillator which in turn controls the PLL IC6. Alterations on the DAC voltage change slightly the frequency of the pulses coming out of the PLL and therefore the timestamp measurements. For example, the difference between two pulses separated by 9 ms can be measured as 8.999967000 ms with the DAC at 1.25 V and as 9.000000300 ms with the DAC at 1.65 V. Note also that the VCXO OSC2 is strongly influenced by temperature changes; the documentation specifies 0.5ppm/oC. For more information, refer to the [DAC calibration](http://www.ohwr.org/projects/fmc-tdc/repository/changes/documentation/TDC_calibration/DACcalibration.xlsx) document.

The [TDC-PTS-Calibration suite](http://www.ohwr.org/projects/pts/repository/revisions/fmctdc1ns5cha_calib/changes/test/fmctdc1ns5cha/calibration/documentation/TDC_PTScalibration_UserGuide.pdf) provides the automated environment for the calibration of each board. It is using a very precise pulse generator, the Pendulum CNT-91 connected to a Cesium clock. The calibration data along with IPMI information is written in the TDC EEPROM.

# TDC application performance

Long term tests on the precision and general performance of the TDC mezzanine application are available [here](http://www.ohwr.org/projects/fmc-tdc/repository/show/board_testing).