FMC-TDC developer’s manual

# Introduction

In order to operate the FMC-TDC board it is necessary to give values to certain configuration registers. Some of these registers target the ACAM chip, some are used for the operation by the FPGA on the SPEC board, and two of them are required to setup the GNUM chip.

All the registers are accessed through the GNUM’s Base Address Register spaces (BAR).

- The GNUM chip registers are accessed through BAR 4.

- All other registers are presently mapped to BAR 0:

* Registers for the GNUM core inside the FPGA: addresses 0:00000 to 0:00020
* Registers for the ACAM chip: addresses 0:20000 to 0:2007F
* Registers for the TDC core inside the FPGA: addresses 0:20080 to 0:200FF
* Register for the carrier 1 Wire: address 0:40000
* Register for the mezzanine I2C: address 0:60000
* Register for the mezzanine 1 Wire: address 0:80000
* Register for the interrupts: address 0:A0000

Amongst the registers for the operation of the TDC core, one in particular is utterly important: the Control Register allows commanding the main Finite State Machine.

The mode chosen for the operation of the ACAM TDC-GPX chip is the I-mode.

Hereon follows a description of each register and a suggested operation procedure.

# Registers description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **R/W** | **Description** | **ADDRESS** | **typical value (if any)** |
| Acam config reg. 0 | R/W | rising/falling edges config | 0:20000 | x01F0FC81 |
| Acam config reg. 1 | R/W | Channel adjustments (other modes) | 0:20004 | x00000000 |
| Acam config reg. 2 | R/W | mode I and disable unused channels according to the application | 0:20008 | x00000E02 |
| Acam config reg. 3 | R/W | resolutions and tests (other modes) | 0:2000C | x00000000 |
| Acam config reg. 4 | R/W | start timer set to 16 and resets | 0:20010 | x0200000F |
| Acam config reg. 5 | R/W | start retrigger OFF and offset set to 2.000 | 0:20014 | x000007D0 |
| Acam config reg. 6 | R/W | LF flags levels to be defined according to the application | 0:20018 | x00000003 |
| Acam config reg. 7 | R/W | PLL values: RefClkDiv=7, HSDiv=234, PhaseNeg | 0:2001C | x00001FEA |
| Acam config reg. 11 | R/W | ERR flag config on the 8 Hit FIFOs | 0:2002C | x00FF0000 |
| Acam config reg. 12 | R/W | INT flag config on Start nb overflow + HFIFO & IFIFO status flags | 0:20030 | x04000000 |
| Acam config reg. 14 | R/W | 16-bit mode control | 0:20038 | x00000000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acam readback reg. 0 | R | rising/falling edges config | 0:20040 | xc1F0FC81 |
| Acam readback reg. 1 | R | Channel adjustments (other modes) | 0:20044 | xc0000000 |
| Acam readback reg. 2 | R | mode I and disable unused channels | 0:20048 | xc0000E02 |
| Acam readback reg. 3 | R | resolutions and tests (other modes) | 0:2004C | xc0000000 |
| Acam readback reg. 4 | R | start timer set to 100 and resets | 0:20050 | xc200000F |
| Acam readback reg. 5 | R | start retrigger OFF and offset set to 2.000 | 0:20054 | xc00007D0 |
| Acam readback reg. 6 | R | LF flags levels to max | 0:20058 | xc00000FC |
| Acam readback reg. 7 | R | PLL values: RefClkDiv=7, HSDiv=234, PhaseNeg | 0:2005C | xc0001FEA |
| Acam readback reg. 8 | R | IFIFO 1 | 0:20060 |  |
| Acam readback reg. 9 | R | IFIFO 2 | 0:20064 |  |
| Acam readback reg. 10 | R | Start01 | 0:20068 |  |
| Acam readback reg. 11 | R | ERR flag config on the 8 Hit FIFOs | 0:2006C | xc0FF0000 |
| Acam readback reg. 12 | R | INT flag config on Start nb overflow + HFIFO & IFIFO status flags | 0:20070 | xc4000800 |
| Acam readback reg. 14 | R | 16-bit mode control | 0:20078 | xc0000000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| starting UTC time | R/W | is updated on demand by a PCI-e command or reset | 0:20080 |  |
| input enable control | R/W | controls the termination enabling for each input (bits 4 downto 0) and general enable input (bit 7) | 0:20084 | x0000009F |
| delay for start pulse phase | R/W | number of cycles to delay the StartFromFPGA pulse with respect to the reference clock rising edge (only for debug) | 0:20088 | x00000000 |
| delay for one Hz pulse phase | R/W | number of cycles to delay the one second pulse with respect to the reference clock rising edge (only for debug) | 0:2008C | x00000000 |
| IRQ tstamp thresh | R/W | an interrupt is issued if the number of accumulated timestamps since the last irq exceeds this threshold (only 7 LSbits considered) | 0:20090 | x000000FF |
| IRQ time thresh | R/W | an interrupt is issued if this amount of seconds has passed after the last irq and at least a timestamp has been registered | 0:20094 | x00000100 = 256 sec |
| DAC word | R/W | word to be sent to the TDC mezzanine DAC (only 24 LSbits considered) | 0:20098 | x0000A8F5 |
|  |  | RESERVED | 0:2009C |  |
| current UTC time | R | calculated by the core according to the local clk | 0:200A0 |  |
| interrupt code | R | provided to PCI-e for action | 0:200A4 |  |
| circular buffer wr pointer | R | provided to PCI-e for DMA configuration (includes the DaCapo flag) | 0:200A8 |  |
| core status | R | provided to PCI-e for diagnostic | 0:200AC |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Control Register | W |  | 0:200FC |  |
| Bit 0 |  | Activate acquisition |  | x00000001 |
| Bit 1 |  | De-activate acquisition |  | x00000002 |
| Bit 2 |  | Load Acam config |  | x00000004 |
| Bit 3 |  | Read Acam configuration |  | x00000008 |
| Bit 4 |  | Read Acam status |  | x00000010 |
| Bit 5 |  | Read Acam IFIFO 1 |  | x00000020 |
| Bit 6 |  | Read Acam IFIFO 2 |  | x00000040 |
| Bit 7 |  | Read Acam Start01 register |  | x00000080 |
| Bit 8 |  | Reset Acam chip |  | x00000100 |
| Bit 9 |  | Load UTC time |  | x00000200 |
| Bit 10 |  | Clear Da Capo flag |  | x00000400 |
| Bit 11 |  | Configure DAC by sending the DAC word |  | x00000400 |

### ACAM REGISTERS:

**Acam config reg. 0 (cf. ACAM TDC-GPX doc):**

Is set to enable the internal oscillator and the rising and falling edges for the TTL inputs 1 to 5.

**Acam config reg. 1 (cf. ACAM TDC-GPX doc):**

Not used in the ACAM operational mode chosen for this application (I-mode).

**Acam config reg. 2 (cf. ACAM TDC-GPX doc):**

Sets the operational mode of the ACAM chip to the I-mode. Disables channels 6 to 8.

**Acam config reg. 3 (cf. ACAM TDC-GPX doc):**

Not used in the ACAM operational mode chosen for this application (I-mode).

**Acam config reg. 4 (cf. ACAM TDC-GPX doc):**

Sets the StartTimer to 16. Sets the EF pin to drive all the time.

**Acam config reg. 5 (cf. ACAM TDC-GPX doc):**

Sets start retrigger to off. Sets the programmable internal start offset to 2000.

**Acam config reg. 6 (cf. ACAM TDC-GPX doc):**

Sets the threshold level for the LF flags arbitrary to 3. Can be changed if required for further developments of the application.

**Acam config reg. 7 (cf. ACAM TDC-GPX doc):**

Sets the ACAM internal PLL values. RefClkDiv=7, HSDiv=234 and inverts the phase output.

**Acam config reg. 11 (cf. ACAM TDC-GPX doc):**

Sets the ErrFlag pin to report for any full flags on the HitFIFOs.

**Acam config reg. 12 (cf. ACAM TDC-GPX doc):**

Sets the IntFlag to the highest bit of the Start# (Start number) counter.

Since all of these ACAM registers are Read/Write, the readback of their value is stored in the **ACAM ReadBack Registers (Reg. 0 to Reg. 14).** This set of registers includes all the configuration registers detailed above, plus the Read-only registers to access the Interface FIFOs registers as well as the Start01 register.

### TDC CORE REGISTERS:

#### Read/Write

**Starting UTC time:**

Sets the initial value for the TDC core internal time base to which all timestamps will be referenced.

**Input enable controls:**

Controls the terminations on each input as well as the general enable of the inputs.

**Start pulse delay:**

Controls the phase between the Ref clock edge and the Tstart pulse in multiples of the 125 MHz clock period. Not currently used, only if required for debug or further developments.

**One Hz pulse delay:**

Controls the phase between the Ref clock edge and the ‘One Hz pulse’ in multiples of the 125 MHz clock period. Not currently used, only if required for debug or further developments.

**DMA word:**

Word to be sent to the TDC mezzanine DAC that controls the OSC1 oscillator. The 11th bit of the control register has to be activated for the reconfiguration of the DAC to take place. Note that after the reconfiguration of the DAC is always followed by the reconfiguration of the local PLL.

**IRQ timestamps threshold:**

Sets the threshold according to which interrupts on IRQ register bit 2 are issued. If the accumulated timestamps after the last IRQ (or the beginning of time) exceed this threshold then an interrupt is raised.

**IRQ time threshold:**

Sets the threshold according to which interrupts on IRQ register bit 3 are issued. If the amount of seconds that have passed since the last IRQ (or the beginning of time) exceed this threshold and at least one timestamp has been registered, then an interrupt is raised.

#### Read only

**Current UTC time:**

As the TDC core keeps track of UTC time according to its local oscillator, this registers provides the current local value used for the timestamps, in order for the software application to perform the correspondent correction with respect to the official UTC.

**Interrupt code:**

This register is reserved for use when interrupt will be enable. At present time no interrupts are implemented.

**WR pointer:**

Keeps track of the next position to be written in the circular buffer memory for the timestamps (12 lowest bits). It includes the ‘Da Capo counter’ that keep track of the number of overruns of the memory block (20 highest bits).

**Core status:**

This register is reserved for future use. At present time no status codes are implemented.

#### Write only

**Control register:**

Only one bit at a time can be activated since each bit carries a command. The value is cleared upon writing.

### GNUM core Registers

The following registers belong to the GNUM core and their use is explained in detail in the corresponding documentation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **R/W** | **description** | **ADDRESS** | **typical value (if any)** |
| DMACTRLR | R/W | DMA engine control | 0:00000 | x00000000 |
| DMASTATR | R | DMA engine status | 0:00004 | x00000001 |
| DMACSTARTR | R/W | DMA start address in the carrier | 0:00008 | Last read address |
| DMAHSTARTLR | R/W | DMA start address (low) in the PCIe host | 0:0000C | From Page List |
| DMAHSTARTHR | R/W | DMA start address (high) in the PCIe host | 0:00010 | x00000000 |
| DMALENR | R/W | DMA read length in bytes | 0:00014 | Last timestamp address - Last address read |
| DMANEXTLR | R/W | Pointer (low) to next item in list | 0:00018 | x00000000 |
| DMANEXTHR | R/W | Pointer (high) to next item in list | 0:0001C | x00000000 |
| DMAATTRIBR | R/W | DMA chain control | 0:00020 | x00000000 |

# Configuration sequence

At power-up:

* It is necessary to load the bit file into the FPGA.
* The clock frequency for the GNUM chip local bus needs to be defined (reg. 4:808).
* A reset is forced on the RST\_N output pin of the GNUM (reg. 4:800). This launches the initialization sequence of the FPGA that sets the parameters for the local PLL on the TDC mezzanine.
* After a reset, the FSM of the core is in the “Acquisition inactive” state, which means that the configuration registers can be accessed. All the configuration registers for the ACAM are then written into the TDC core.
* The command to load the configuration into the ACAM is issued through the Control Register.
* (Optionally the configuration of the ACAM can be read back for verification by issuing the corresponding command through the Control Register and reading the corresponding Read-back Registers.)
* Before starting the acquisition, it is necessary to reset the ACAM chip through the Control Register command.
* (Optionally the Status Register of the ACAM can be read back for verification by issuing the corresponding command through the Control Register and reading the corresponding Read-back Register.)
* The input are enable and its termination resistors set though the dedicated register.
* The reference starting time for the local UTC is set through the corresponding register and loaded for operation with a command on the Control Register.
* Finally the acquisition is launched through the corresponding command of the Control Registers. This generates the Tstart signal for the ACAM chip, and from that moment on, every pulse arriving to the ACAM inputs will generate a timestamp that will be immediately fetch by the TDC core and stored in the Circular Buffer memory.

After a reset of the TDC core:

* After a reset, the FSM of the core is in the “Acquisition inactive” state, which means that the configuration registers can be accessed. All the configuration registers for the ACAM are then written into the TDC core.
* The command to load the configuration into the ACAM is issued through the Control Register.
* (Optionally the configuration of the ACAM can be read back for verification by issuing the corresponding command through the Control Register and reading the corresponding Read-back Registers.)
* Before starting the acquisition, it is necessary to reset the ACAM chip through the Control Register command.
* (Optionally the Status Register of the ACAM can be read back for verification by issuing the corresponding command through the Control Register and reading the corresponding Read-back Register.)
* The input are enable and its termination resistors set though the dedicated register.
* The reference starting time for the local UTC is set through the corresponding register and loaded for operation with a command on the Control Register.
* Finally the acquisition is launched through the corresponding command of the Control Registers. This generates the Tstart signal for the ACAM chip, and from that moment on, every pulse arriving to the ACAM inputs will generate a timestamp that will be immediately fetch by the TDC core and stored in the Circular Buffer memory.

# Operation

By reading the WR Pointer Register, the software will know how many new timestamps are available in the Circular Buffer and will perform a DMA accordingly.

In order to configure the DMA, at least the DMACSTARTR, DMAHSTARTLR and DMALENR registers in the GNUM core need to be set. Then the DMA is launched through the DMACTRLR and its success can be verified in the DMASTATR.

DATA FORMAT:

Each timestamp is 128 bits. It therefore appears in the host memory in 4 consecutive 32-bit words:

* 127 downto 96: Metadata including Input Channel, edge type…
* 95 downto 64: Local UTC with a 1s resolution.
* 63 downto 32: Coarse time within the second with a 8 ns resolution.
* 31 downto 0: Fine time to be added to the coarse time. Each bit representing 81 ps.

Whenever the drift between the local UTC and the official UTC needs to be corrected, the new value for the local UTC is set and updated through the corresponding command of the Control Register. It is not necessary to stop the acquisition for this.

## notes:

In case GNUM core addresses decoding is changed, the TDC core address decoding for the TDC registers may need adjusting. The decoding is performed at the level of the TOP module.

In case polling is abandoned for interrupt driven acquisition, the generation of the interruptions and the management of the interrupt code should be implemented at the level of the BUFFER MANAGEMENT processes inside the DATA FORMATTING module.

## Appendix

Comments on the code review of 8/11/2011 and corrective actions:

Summary of all the comments to the code by authors.

General:

========

TOM

- Wait statements in synthesizable code look suspicious...

They indeed synthesize correctly, but to be honest, I've never seen processes

coded in such way.

- (googled a bit) Technically, it should be "wait until rising\_edge(clk)" or

" wait until (clk'event and clk = '1')".

- Are I/O always assigned to internal signals for some particular reason?

- I'd suggest declaring commonly used components (e.g. counters) in a shared

package to avoid repetitive declarations.

- Clock signal assignments are dangerous.

On a simulation, there is a big risk of getting timing errors when co-simulated

with Verilog code due to incompatibilities between the way events are

scheduled in VHDL and Verilog simulators

(in VHDL, a continuous assignment is scheduled as an event, while in verilog

it's purely continous).

*Gonzalo: No problem for ‘wait until’ statements. Clock signal assignments(“clk <= clk\_i;”) have been removed from all blocks.*

top\_tdc.vhd

===========

JAVIER

- Line 1057 (and others). The wait until spec\_clk = '1'; at the end of

the process looks awkward to me. I guess it means this is a

synchronous process working on the rising edge of spec\_clk. Is there

any advantage to using this notation?

*Gonzalo: No problem for ‘wait until’ statements, improves readability by removing one indent.*

TOM

- what are g\_span and g\_width generics (a comment would be helpful)

- gnum\_reset signal is asynchronous, but used throughout the design as

synchronous. Add a sync chain.

- put together all the components which form the TDC core into a single VHDL

entity, with the ACAM I/F on one side and Wishbone on other side

(i.e. without the gennum or other platform-specific stuff inside)

- lines 1024, 1039: when decoding addresses, define base addrs as constants

instead of using hardcoded values

*Gonzalo: comments for ‘g\_span’ and ‘g\_width’ added. Base addresses for decoding defined as constants. Block for TDC core* ***still to be done.***

acam\_databus\_interface.vhd

==========================

JAVIER

- Line 73. "read" and "write" are not VHDL reserved words but they are

names of functions people use to do I/O. Probably wise to chose

other names for states.

- Line 244. address\_o going to the ACAM is not registered. This signal

is driven by adr\_i, the wishbone input address. These lines are

being used in another entity (data\_engine.vhd) as inputs to other

processes so chances are they will not use IOB FFs, which might be

important to respect setup and hold constraints of the ACAM.

*Gonzalo: state names changed. ‘address\_o’ not registered because ACAM timing constraints are referred to the WR or RD signals, which arrive several clock cycles later. In any case the constraint is specified in the .sdc constraint file for Synplify.*

TOM

- line 192: these signals (efX, lfX) are asynchronous. I'd suggest using a

sync chain of 2-3 flip-flops.

*Gonzalo: ‘ef’ and ‘lf’ signals registered.*

EVA

- lines 227-241: signals ef1/2, lf1/2 shouldn t be synchronized as well

(use one more DFF)?

- The ack signals are pulses (\_p)

acam\_databus\_interface, ack, line 63

*Gonzalo: ‘ef’ and ‘lf’ signals registered.” \_p” added to signal names.*

acam\_timecontrol\_interface.vhd

==============================

JAVIER

- Line 246 (and others). This can be written in one line:

"start\_trig\_r <= start\_trig & start\_trig\_r(1 downto 0);"

- Line 255 (and others). This edge detector uses signal ref\_clk\_r(3),

which is potentially metastable. It also relies very heavily on the

fact that ref\_clk should be at a given frequency.

*Gonzalo: ‘start\_trig\_r’ assignment rewritten to “start\_trig\_r <= start\_trig & start\_trig\_r(2 downto 1);”. ‘acam\_refclk’ is now properly synchronized within the ‘clk\_rst\_managr’ module.*

TOM

- line 89: the name doesn't explain the purpose of this constant. What delay

does it describe?

- line 125: are you sure this will work correctly? err\_flag\_r(2) is written

twice.

How about using slice & join instead?

err\_flag\_r <= err\_flag\_r(err\_flag\_r'left-1 downto 0) & err\_flag\_i;

- line 249: acam\_refclk and clk are normally phase aligned by the AD9516 PLL,

so there may be a setup time violation here. Is the AD9516 shifter programmed

to ensure the FPGA will correctly sample acam\_refclk signal?

(otherwise, the 1st stage could sample on the falling edge of clk).

- line 255: refclk\_r(3) can be metastable, causing refclk\_edge signal to be

unreliable.

Consider adding 1 more sync stage (or using only (1) and (0) indices)

- line 260: the same for start\_trig\_edge

*Gonzalo: comment add for ‘constant\_delay’. Assignments for ‘err\_flag\_r’ and ‘int\_flag\_r’ rewritten.*

EVA

- synchronization and edge detection of refclk takes place in both units

one\_hz\_gen lines 153-170; acam\_timecontrol\_interface lines 240-258

refclk\_edge is a pulse (refclk\_edge\_p)

use of the first DFF s\_acam\_refclk(3) should be avoided

- lines 260: start\_trig\_edge is a pulses (\_p)

use of the first DFF (start\_trig\_r(2)) should be avoided

- lines 120-146 : good!-)

*Gonzalo: ‘acam\_refclk’ is now properly synchronized within the ‘clk\_rst\_managr’ module.*

clk\_rst\_managr.vhd

==================

JAVIER

- Line 284. This process can create metastability in signal gral\_incr,

which is then going to many destinations inside the incr\_counter

block, possibly leading to non-deterministic behavior of the counter.

- Line 309. Signal cs seems to be negative logic. This should be visible

in its name.

- Line 571. cs will not use an IOB FF because it is read in line

366. Please check all other cases when this can happen.

*Gonzalo: issues not addressed yet.* ***Still to be done.***

TOM

- line 166: IBUFDS + BUFG can be merged into single IBUFGDS

- line 225: chain of two global buffers on spec\_clk\_i

(IBUFG drives a global clock net, so there's no need to follow it with another

BUFG)

- lines 409+: I'd suggest defining these regs as an array of records?

- lines 289+: I couldn't understand the way the power-on-reset is generated.

A comment would be greatly appreciated.

*Gonzalo: issues not addressed yet.* ***Still to be done.***

one\_hz\_gen.vhd

==============

JAVIER

- Line 153. Similar comments on metastability as before. Also, the

assumption on the clock frequency of s\_acam\_refclk is very strong and

might be in trouble if sampling happens close to the edges and the

signals are a bit jittery. Why is this "frequency test" needed?

*Gonzalo: ‘acam\_refclk’ is now properly synchronized within the ‘clk\_rst\_managr’ module. No “frequency test” anymore.*

TOM

- lines 161+: you're syncing the same signal (tdc refclock) twice in the design

(here and in acam\_timecontrol\_interface). Due to possible metastability,

you can get inconsistent pulses in these two modules.

*Gonzalo: ‘acam\_refclk’ is now properly synchronized within the ‘clk\_rst\_managr’ module.*

EVA

synchronization and edge detection of refclk takes place in both units

one\_hz\_gen lines 153-170; acam\_timecontrol\_interface lines 240-258

refclk\_edge is a pulse (refclk\_edge\_p)

use of the first DFF s\_acam\_refclk(3) should be avoided

*Gonzalo: ‘acam\_refclk’ is now properly synchronized within the ‘clk\_rst\_managr’ module.*

data\_engine

===========

TOM

- state names look like signal names, consider using uppercase or prefixes to

avoid confusion.

- line 294: is the others block ever reached?

- define addresses of commonly used ACAM regs as constants

(e.g. c\_ACAM\_IFIFO1 for x"08", etc...) to improve readability.

*Gonzalo: state names written in UpperCase. ‘when others’ condition not necessary but used for good engineering practice. ACAM register addresses defined as constants in the pkg file.*

EVA

- I would suggest a bit cleaner names for the WBs:

stb/ack/.. in the data\_engine could be renamed to

acam\_stb/ acam\_ack..

*Gonzalo: prefexis added to internal wishbone names.*

circular\_buffer

===============

TOM

- pipelined WB is not that complex, there's no need for an FSM.

In case of a non-stalling peripheral (stall == 0 always), the ack signal can

be generated like this:

process(clk)

ack <= stb and cyc;

(adr and dat go straight to the block ram).

- Consider replacing Coregen cores with generic ones. The circular buffer can

be done as a simple array.

*Gonzalo: FSM maintained for readability. Replacement of coregent core by a generic one* ***still to be done.***

EVA

- I would suggest a bit cleaner names for the WBs:

classic\_stb/ classic\_ack/.. in the circular\_buffer renamed to

gnum\_classic\_stb/ gnum\_classic\_ack..

pipe\_stb/ pipe\_ack/.. in the circular\_buffer renamed to

gnum\_pipe\_stb/ gnum\_pipe\_ack..

- lines 37, 50: class\_clk\_i and pipe\_clk\_i are actually the same clock;

maybe they could just be named gnum\_clk\_i

- The ack signals are pulses (\_p)

circular\_buffer, classic\_ack, line 123

*Gonzalo: signal names have been modified. Just one clk is now used. “\_p” prefix is not added.*

countdown\_counter:

free\_counter:

incr\_counter:

=============

TOM

- line 49, 52: (un)signeds can be compared with integers directly

(numeric\_std supports this).

if (value = 0) ...

- coding style (\_i suffix for inputs, etc.).

*Gonzalo: these counters are re-used old designs before the VHDL guidelines.*

reg\_ctrl:

=========

- use constants for defining register adresses

- line 135+: avoid repetitive assignments. Use loop construct instead.

- line 126+: reg\_ack <= reg\_stb and reg\_cyc and not reg\_ack;

*Gonzalo: register addresses defined as constants in the pkg module. Loop construct avoided for clearer readability. ‘reg\_ack’ assignment not modified in order to stay compatible with Wishbone classic. To be modified only if the CSR port of the GNUM core is changed.*

tdc\_core\_pkg:

============

- lines 73+: consider defining these constants in decimal format

(these are timeouts, and in decimal they are easier to understand).

*Gonzalo: not modified, comments added instead.*

sim/

----------------

- Try to avoid uploading binary files if they are not absolutely necessary

(i.e. compiled Xilinx libraries).

- A system-level testbench should be provided (tb\_tdc.vhd doesn't include

any actual testbench code, just the models connected together).

*Gonzalo: the removal of the simulation binary files from the repository is* ***still to be done.*** *The test-bench vectors that complete the system level test-bench are actually in the ‘DATA’ folder.*