

## **HPUCore specificantions**

Author: Francesco Diotalevi

Company: Istituto Italiano di Tecnologia - EDL

| History: |                                 |                     |   |
|----------|---------------------------------|---------------------|---|
| Revision | Date                            | Author              | Description   |
| 1.0      | September 19 <sup>th</sup> 2016 | Francesco Diotalevi | First Release (DRAFT)   |
| 1.1      | June 14 <sup>th</sup> 2017      | Francesco Diotalevi | Modified the AXIstream module and added some debug ports. Added Reset_DMA_stream bit into CTRL_REG.                               |
| 2.0      | November 15 <sup>th</sup> 2017  | Francesco Diotalevi | Bug fixed for the HSSAER Channel Enable. Added AUX Threshold Error register and AUX Rx Counter registers. Some Fixes in HDL code. |
| 2.1      | June 15 <sup>th</sup> 2018      | Francesco Diotalevi | Enlarged to 24 the SSAER data transfer. Different header coding.  |
| 2.2      | July 31 <sup>st</sup> 2018      | Francesco Diotalevi | Bug fixed for issue <a href="#">#10</a>   |

## Contents

|   |           |
|---|-----------|
| <b>HPUCORE SPECIFICANTIONS</b>  | <b>1</b>  |
| <b>GLOSSARY</b>   | <b>3</b>  |
| <b>HPUCORE IMPLEMENTATION PARAMETERS</b>                              | <b>3</b>  |
| <b>INTRODUCTION</b>   | <b>4</b>  |
| <b>1 HPUCORE BLOCK DIAGRAM</b>  | <b>5</b>  |
| 1.1 UNDERSTANDING THE HPUCORE   | 7         |
| <b>2 HPUCORE INTERNAL REGISTERS</b>                                   | <b>8</b>  |
| 2.1 CONTROL REGISTER (CTRL_REG)                                       | 10        |
| 2.2 LOOPBACK LR CONFIGURATION REGISTER (LPBK_LR_CNFG_REG)             | 11        |
| 2.3 RX DATA BUFFER REGISTER (RXDATA_REG)                              | 12        |
| 2.4 RX TIME BUFFER REGISTER (RXTIME_REG)                              | 13        |
| 2.5 TX DATA BUFFER REGISTER (TXDATA_REG)                              | 15        |
| 2.6 DMA REGISTER (DMA_REG)  | 16        |
| 2.7 RAW STATUS REGISTER (STAT_RAW_REG)                                | 17        |
| 2.8 IRQ REGISTER (IRQ_REG)  | 19        |
| 2.9 MASK REGISTER (MSK_REG)   | 20        |
| 2.10 WRAPPING TIMEStamp REGISTER (WRAPTimeStamp_REG)                  | 21        |
| 2.11 HSSAER STATUS REGISTER (HSSAER_STAT_REG)                         | 22        |
| 2.12 HSSAER RX ERROR REGISTER (HSSAER_RX_ERR_REG)                     | 23        |
| 2.13 HSSAER RX MSK REGISTER (HSSAER_RX_MSK_REG)                       | 24        |
| 2.14 RX CONTROL REGISTER (RX_CTRL_REG)                                | 25        |
| 2.15 TX CONTROL REGISTER (TX_CTRL_REG)                                | 26        |
| 2.16 RX PAER CONFIGURATION REGISTER (RX_PAER_CFNG_REG)                | 27        |
| 2.17 TX PAER CONFIGURATION REGISTER (TX_PAER_CFNG_REG)                | 28        |
| 2.18 IP CONFIGURATION REGISTER (IP_CFNG_REG)                          | 29        |
| 2.19 FIFO THRESHOLD REGISTER (FIFO_THRSH_REG)                         | 30        |
| 2.20 LOOPBACK AUX CONFIGURATION (LPBK_AUX_CNFG_REG)                   | 31        |
| 2.21 IDENTIFICATION REGISTER (ID_REG)                                 | 32        |
| 2.22 AUXILIARY RX CONTROL REGISTER (AUX_RX_CTRL_REG)                  | 33        |
| 2.23 HSSAER AUX RX ERROR REGISTER (HSSAER_AUX_RX_ERR_REG)             | 34        |
| 2.24 HSSAER AUX RX MSK REGISTER (HSSAER_AUX_RX_MSK_REG)               | 35        |
| 2.25 AUX ERROR COUNTER THRESHOLD REGISTER (HSSAER_AUX_RX_ERR_THR_REG) | 36        |
| 2.26 AUX ERROR COUNTER CH0 REGISTER (HSSAER_AUX_RX_ERR_CH0_REG)       | 37        |
| 2.27 AUX ERROR COUNTER CH1 REGISTER (HSSAER_AUX_RX_ERR_CH1_REG)       | 38        |
| 2.28 AUX ERROR COUNTER CH2 REGISTER (HSSAER_AUX_RX_ERR_CH2_REG)       | 39        |
| 2.29 AUX ERROR COUNTER CH3 REGISTER (HSSAER_AUX_RX_ERR_CH3_REG)       | 40        |
| <b>3 REFERENCES</b>   | <b>41</b> |
| <b>4 APPENDIXES</b>   | <b>42</b> |

## Glossary

| Acronym | Meaning       |
|---------|---------------|
| TBD     | To Be Defined |
| TBT     | To Be Tested  |
|         |               |
|         |               |

## HPUCore implementation parameters

The generic parameters used in SynthTactAER design are shown in Table 1

**Table 1 HPUCore implementation parameters in SynthTactAER design**

| Parameter name            | Description                          | Value |
|---------------------------|--------------------------------------|-------|
| <b>C_PAER_DSIZE</b>       | Size of PAER address                 | 24    |
| <b>C_RX_HAS_PAER</b>      | If true the RX PAER I/f is exposed   | False |
| <b>C_RX_HAS_HSSAER</b>    | If true the RX HSSAER I/f is exposed | True  |
| <b>C_RX_HAS_GTP</b>       | If true the RX GTP I/f is exposed    | False |
| <b>C_RX_HSSAER_N_CHAN</b> | The number of RX HSSAER channels     | 3     |
| <b>C_TX_HAS_PAER</b>      | If true the TX PAER I/f is exposed   | False |
| <b>C_TX_HAS_HSSAER</b>    | If true the TX HSSAER I/f is exposed | False |
| <b>C_TX_HAS_GTP</b>       | If true the TX GTP I/f is exposed    | False |
| <b>C_TX_HSSAER_N_CHAN</b> | The number of TX HSSAER channels     | 2     |

## Introduction

The Head Processor Unit Core (HPUCore) is an AXI peripheral used to managed different AER streaming and transfer the acquired data into memory through DMA interface or by reading registers with Host CPU.

It has an Axi4 lite bus I/f for writing/reading internal registers and delivers two AXI stream bus @ 32bit. Can be configured with up to 4 input AER channels and each channel can be PAER, SAER or GTP (TBT).

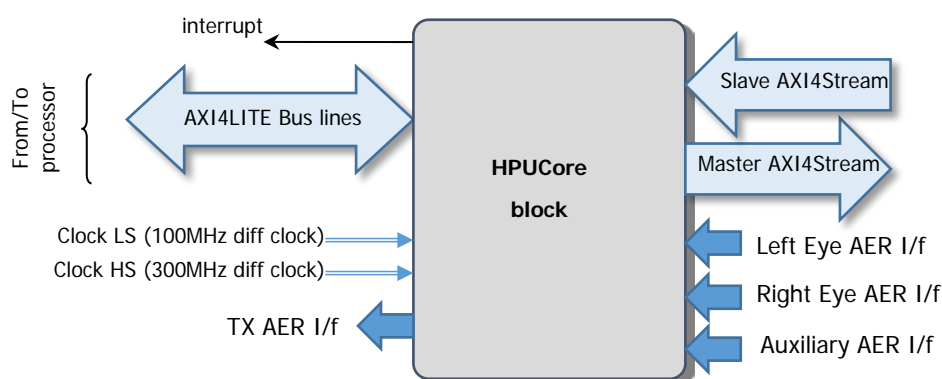


Figure 1 HPUCore block

The list of the ports and their description is shown in Table 2.

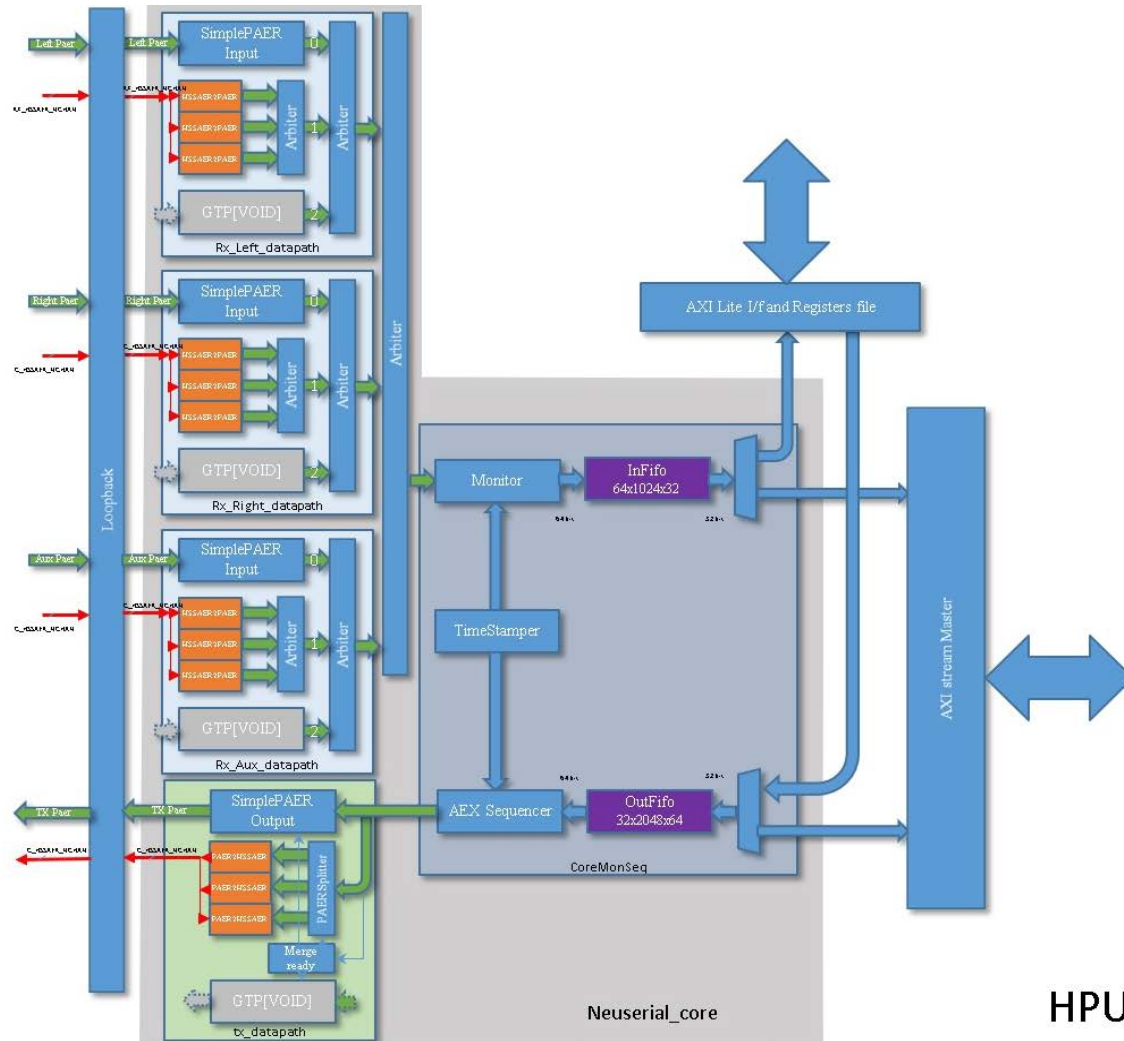
Table 2 HPUCore interface signals description

| Comment             | Port name     | Width | Dir | Description  |
|---------------------|---------------|-------|-----|--|
| Interrupt           | Interrupt     | 1     | O   | Level interrupt active high signal   |
| AXI4 Lite Bus lines | S_AXI_ACLK    | 1     | I   | AXI Clock, System clock line<br><i>IT MUST BE THE SAME AS HSSAER_CIKLS_p</i> |
|                     | S_AXI_ARESETN | 1     | I   | AXI Reset active low line  |
|                     | S_AXI_AWADDR  | 32    | I   | AXI Write address  |
|                     | S_AXI_AWVALID | 1     | I   | Write address valid  |
|                     | S_AXI_WDATA   | 32    | I   | Write data   |
|                     | S_AXI_WSTRB   | 4     | I   | Write strobes  |
|                     | S_AXI_WVALID  | 1     | I   | Write valid  |
|                     | S_AXI_BREADY  | 1     | I   | Response ready   |
|                     | S_AXI_ARADDR  | 32    | I   | Read address   |
|                     | S_AXI_ARVALID | 1     | I   | Read address valid   |
|                     | S_AXI_RREADY  | 1     | I   | Read ready   |
|                     | S_AXI_ARREADY | 1     | O   | Read address ready   |
|                     | S_AXI_RDATA   | 32    | O   | Read data  |
|                     | S_AXI_RRESP   | 2     | O   | Read response  |
|                     | S_AXI_RVALID  | 1     | O   | Read valid   |
|                     | S_AXI_WREADY  | 1     | O   | Write ready  |
|                     | S_AXI_BRESP   | 2     | O   | Write response   |

| Comment               | Port name                      | Width | Dir | Description  |
|-----------------------|--------------------------------|-------|-----|--|
|                       | S_AXI_BVALID                   | 1     | O   | Write response valid   |
|                       | S_AXI_AWREADY                  | 1     | O   | Write address ready  |
| RX Left Eye I/F       | LRx_PAER_Addr                  | 18    | I   | Parallel AER address   |
|                       | LRx_PAER_Req                   | 1     | I   | Parallel AER request   |
|                       | LRx_PAER_Ack                   | 1     | O   | Parallel AER acknowledge   |
|                       | LRx_HSSAER                     | 4     | I   | 4 channels High Speed Serial AER signal                              |
| RX Right Eye I/F      | RRx_PAER_Addr                  | 18    | I   | Parallel AER address   |
|                       | RRx_PAER_Req                   | 1     | I   | Parallel AER request   |
|                       | RRx_PAER_Ack                   | 1     | O   | Parallel AER acknowledge   |
|                       | RRx_HSSAER                     | 4     | I   | 4 channels High Speed Serial AER signal                              |
| RX Auxiliary I/F      | AuxRx_PAER_Addr                | 18    | I   | Parallel AER address   |
|                       | AuxRx_PAER_Req                 | 1     | I   | Parallel AER request   |
|                       | AuxRx_PAER_Ack                 | 1     | O   | Parallel AER acknowledge   |
|                       | AuxRx_HSSAER                   | 4     | I   | 4 channels High Speed Serial AER signal                              |
| Slave Axi stream I/f  | S_AXIS_TREADY                  | 1     | O   | Tready   |
|                       | S_AXIS_TDATA                   | 32    | I   | Data bus   |
|                       | S_AXIS_TLAST                   | 1     | I   | Last signal  |
|                       | S_AXIS_TVALID                  | 1     | I   | Valid signal   |
| Master Axi stream I/f | M_AXIS_TREADY                  | 1     | I   | Tready   |
|                       | M_AXIS_TDATA                   | 32    | O   | Data bus   |
|                       | M_AXIS_TLAST                   | 1     | O   | Last signal  |
|                       | M_AXIS_TVALID                  | 1     | O   | Valid signal   |
| TX I/F                | Tx_PAER_Addr                   | 18    | O   | Parallel AER address   |
|                       | AuxRx_PAER_Req                 | 1     | O   | Parallel AER request   |
|                       | AuxRx_PAER_Ack                 | 1     | I   | Parallel AER acknowledge   |
|                       | AuxRx_HSSAER                   | 3     | O   | 3 channels High Speed Serial AER signal                              |
| System signals        | HSSAER_ClkLS_p, HSSAER_ClkLS_n | 2     | I   | Differential 100MHz clock<br><i>IT MUST BE THE SAME AS S_AXI_CLK</i> |
|                       | HSSAER_ClkHS_p, HSSAER_ClkHS_n | 2     | I   | Differential 300MHz clock  |
|                       | nSyncReset                     | 1     | I   | Active low Synchronous reset   |
|                       | DefLocFarLpbk                  | 1     | I   | Default Far loopback value   |
|                       | DefLocNearLpbk                 | 1     | I   | Default Near loopback value  |

## 1 HPUCore Block Diagram

The HPUCore Block diagram is shown in Figure 2.



HPUCore Block Diagram

Figure 2 HPUCore architecture

## 1.1 Understanding the HPUCore

The HPUCore can have until 3 different AER generator connected to him. In the HPUCore implementation for SynthTactAER design the AER sources are configured as Serial AER lines. The Serial AER lines are LVDS signals and each source has 3 different channels. The 3 interfaces are:

- Left ATIS camera,
- Right ATIS camera and
- AUXiliary interface.

The HPUCore deserializes the data coming from its channels and fill a FIFO with data and associated time stamp.

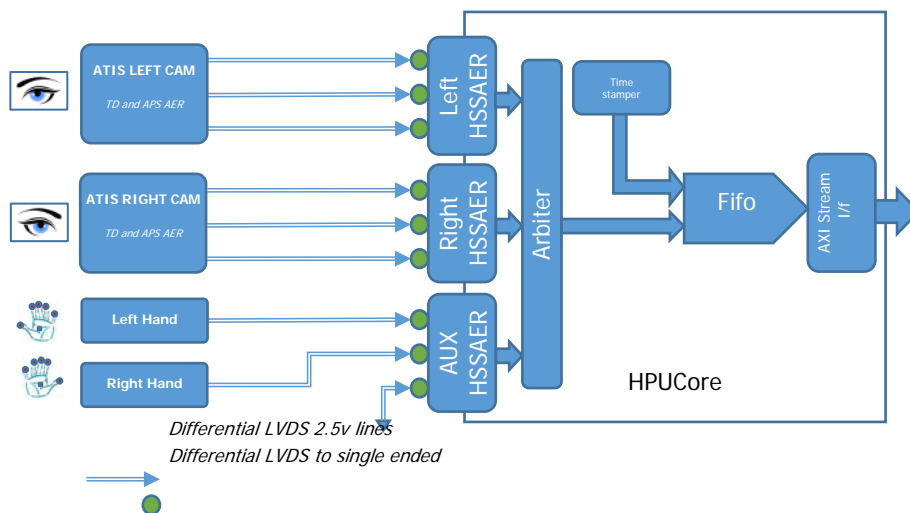


Figure 3 Simplified block diagram of the HPUCore.

Table 3 Meaning of Timestamp and Data values.

**TIMESTAMP (31 downto 0) when CTRLReg.15 is '0'**

| TIME ID (Reserved)<br>31 downto 24 | Payload<br>23 downto 0 |
|------------------------------------|------------------------|
| 10000000                           | Time value             |

**TIMESTAMP (31 downto 0) when CTRLReg.15 is '1'**

| Payload<br>31 downto 0 |
|------------------------|
| Time value             |

Note: One-unit difference into Time value means 80ns.

According to the AERSensorsMap.xlsx (svn version r12867) the data are packed as in the tables that follow:

**DATA (31 downto 0) coming from Left/Right eyes channels**

| 31 | 30       | 29 | 28 | 27 | 26 | 25 | 24 | Payload<br>23 downto 0 |
|----|----------|----|----|----|----|----|----|------------------------|
| 0  | Reserved |    |    |    | 0  | 0  | 0  | Data                   |

## DATA (31 downto 0) coming from Aux channel

|    |          |    |    |    |                        |    |    |                        |      |  |  |
|----|----------|----|----|----|------------------------|----|----|------------------------|------|--|--|
| 31 | 30       | 29 | 28 | 27 | 26                     | 25 | 24 | Payload<br>23 downto 0 |      |  |  |
| 0  | Reserved |    |    |    | Copy of<br>data[21:19] |    |    |                        | Data |  |  |

The AXI stream I/f is connected to a DMA used to perform slave to memory transfers.

Reading from `/dev/iit-hpu` we obtain couples of 32bit wide data. The first data is a timestamp while the second one is the data associated to the timestamp.

The Table 3 shows the meaning of both timestamp and data values.

Then a typical acquired sequence is as in the table below:

| Example of acquired sequence      | Events Notes:<br><i>All events in the table come from Left Eye through SAER interface</i>             |
|-----------------------------------|---|
| T: 0x80FFFD1D --> TD: 0x040132E5  | The payload for TD event is 0x132E5   |
| T: 0x80FFFD9E --> TD: 0x040132E6  | The payload for TD event is 0x132E6   |
| T: 0x80FFFE1F --> TD: 0x040132E7  | The payload for TD event is 0x132E7   |
| T: 0x80FFFEA0 --> TD: 0x040132E8  | The payload for TD event is 0x132E8   |
| T: 0x80FFFE8B --> APS: 0x0405C600 | This is a APS event because of the 18 <sup>th</sup> bit is high. The payload for APS event is 0x1C600 |
| T: 0x80FFFF21 --> TD: 0x040132E9  | The payload for TD event is 0x132E9   |
| T: 0x80FFFFA2 --> TD: 0x040132EA  | The payload for TD event is 0x132EA   |
| T: 0x80000023 --> TD: 0x040132EB  | Here the time stamp has wrapped incrementing the wrap value. The payload for TD event is 0x132EB      |

## 2 HPUCore internal registers

In this Section a detailed view of the registers internal to the HPUCore module is given.

The HPUCore block has an Axi Light Slave interface [1] to interface the registers with the hosting processor.

AXI is part of ARM AMBA, a family of micro controller buses first introduced in 1996. The first version of AXI was first included in AMBA 3.0, released in 2003. AMBA 4.0, released in 2010, includes the second version of AXI, AXI4. There are three types of AXI4 interfaces:

- AXI4—for high-performance memory-mapped requirements.
- AXI4-Lite—for simple, low-throughput memory-mapped communication (for example, to and from control and status registers).
- AXI4-Stream—for high-speed streaming data.

Xilinx introduced these interfaces in the ISE® Design Suite, release 12.3.

In the following the complete list of accessible HPUCore registers.

| # | Offset | Mnemonic                         | Description                        | Type | Reset Value |
|---|--------|----------------------------------|------------------------------------|------|-------------|
| 0 | 0x00   | <a href="#">CTRL_REG</a>         | Control register                   | R/W  | 0x00000000  |
| 1 | 0x04   | <a href="#">LPBK_LR_CNFG_REG</a> | Loopback LR Configuration register | R/W  | 0x00000000  |
| 2 | 0x08   | <a href="#">RXData_REG</a>       | RX Data Buffer                     | R/O  | 0x00000000  |



| #  | Offset | Mnemonic                                  | Description  | Type | Reset Value |
|----|--------|---|--|------|-------------|
| 3  | 0x0C   | <a href="#">RXTime_REG</a>                | RX Time Buffer                                     | R/O  | 0x00000000  |
| 4  | 0x10   | <a href="#">TXData_REG</a>                | TX Data Buffer                                     | R/W  | 0x00000000  |
| 5  | 0x14   | <a href="#">DMA_BREG</a>                  | DMA Burst Register                                 | R/W  | 0x00000000  |
| 6  | 0x18   | <a href="#">STAT_RAW_REG</a>              | Status RAW register                                | R/O  | 0x00000000  |
| 7  | 0x1C   | <a href="#">IRQ_REG</a>                   | IRQ register                                       | R/C  | 0x00000000  |
| 8  | 0x20   | <a href="#">MSK_REG</a>                   | Mask register for the IRQ_REG register             | R/W  | 0x00000000  |
| 10 | 0x28   | <a href="#">WRAPTimeStamp_REG</a>         | Wrapping TimeStamp Register                        | R/C  | 0x00000000  |
| 13 | 0x34   | <a href="#">HSSAER_STAT</a>               | HSSAER status register                             | R/O  | 0x00000000  |
| 14 | 0x38   | <a href="#">HSSAER_RX_ERR</a>             | HSSAER RX Error register                           | R/O  | 0x00000000  |
| 15 | 0x3C   | <a href="#">HSSAER_RX_MSK</a>             | HSSAER RX Mask register                            | R/W  | 0x00000000  |
| 16 | 0x40   | <a href="#">RX_CTRL_REG</a>               | RX Control register                                | R/W  | 0x00000000  |
| 17 | 0x44   | <a href="#">TX_CTRL_REG</a>               | TX Control register                                | R/W  | 0x00000000  |
| 18 | 0x48   | <a href="#">RX_PAER_CNFG_REG</a>          | RX PAER configuration register                     | R/W  | 0x00000000  |
| 19 | 0x4C   | <a href="#">TX_PAER_CNFG_REG</a>          | TX PAER Configuration register                     | R/W  | 0x00000000  |
| 20 | 0x50   | <a href="#">IP_CNFG_REG</a>               | IP implemented configuration register              | R/O  | 0x0000????  |
| 21 | 0x54   | <a href="#">FIFO_THRS_REG</a>             | FIFO threshold value register                      | R/W  | 0x00000000  |
| 22 | 0x58   | <a href="#">LPBK_AUX_CNFG_REG</a>         | Loopback AUX Configuration register                | R/W  | 0x00000000  |
| 23 | 0x5C   | <a href="#">ID_REG</a>                    | ID Register  | R/O  | 0x48505522  |
| 24 | 0x60   | <a href="#">AUX_CTRL_REG</a>              | Auxiliary interface Control register               | R/W  | 0x00000000  |
| 25 | 0x64   | <a href="#">HSSAER_AUX_RX_ERR</a>         | HSSAER AUX RX Error register                       | R/O  | 0x00000000  |
| 26 | 0x68   | <a href="#">HSSAER_AUX_RX_MSK</a>         | HSSAER AUX RX Mask register                        | R/W  | 0x00000000  |
| 27 | 0x6C   | <a href="#">HSSAER_AUX_RX_ERR_THR_REG</a> | HSSAER AUX RX error threshold register             | R/W  | 0x10101010  |
| 28 | 0x70   | <a href="#">HSSAER_AUX_RX_ERR_CH0_REG</a> | HSSAER AUX RX error counter register for Channel 0 | R/C  | 0x00000000  |
| 29 | 0x74   | <a href="#">HSSAER_AUX_RX_ERR_CH1_REG</a> | HSSAER AUX RX error counter register for Channel 1 | R/C  | 0x00000000  |
| 30 | 0x78   | <a href="#">HSSAER_AUX_RX_ERR_CH2_REG</a> | HSSAER AUX RX error counter register for Channel 2 | R/C  | 0x00000000  |
| 31 | 0x7C   | <a href="#">HSSAER_AUX_RX_ERR_CH3_REG</a> | HSSAER AUX RX error counter register for Channel 3 | R/C  | 0x00000000  |

## 2.1 Control register (CTRL\_REG)

This register is used to control the behaviour of the HPUCore block.

| CTRL_REG (HPUCore Base + 0x00) |                        |                         |                         |                           |                           |                     |                 |                             |                          |                          |                    |              | Reset Value: 0x00000000 |           |                |
|--------------------------------|------------------------|-------------------------|-------------------------|---------------------------|---------------------------|---------------------|-----------------|-----------------------------|--------------------------|--------------------------|--------------------|--------------|-------------------------|-----------|----------------|
| 31                             | 30                     | 29                      | 28                      | 27                        | 26                        | 25                  | 24              | 23                          | 22                       | 21                       | 20                 | 19           | 18                      | 17        | 16             |
| LocFar<br>RPAER<br>Lpbk        | LocFar<br>PAERL<br>pbk | LocFar<br>RSAER<br>Lpbk | LocFar<br>LSAER<br>Lpbk | LocFar<br>AuxPA<br>ERLpbk | LocFar<br>AuxSA<br>ERLpbk | Loc<br>Near<br>Lpbk | Remot<br>e Lpbk | Rese<br>rved                |                          |                          |                    |              |                         |           |                |
| r/w                            | r/w                    | r/w                     | r/w                     | r/w                       | r/w                       | r/w                 | r/w             |                             |                          |                          |                    |              |                         |           |                |
| 15                             | 14                     | 13                      | 12                      | 11                        | 10                        | 9                   | 8               | 7                           | 6                        | 5                        | 4                  | 3            | 2                       | 1         | 0              |
| Full<br>Time<br>stam<br>p      | Reserved               |                         | Reset<br>DMA<br>stream  | Reserved                  |                           |                     |                 | AuxRxPa<br>er FIFO<br>FLush | RRxPaer<br>FIFO<br>FLush | LRxPaer<br>FIFO<br>FLush | Flush<br>FIFO<br>S | Rese<br>rved | EN<br>INT               | EN<br>DMA | DMA<br>running |
| r/w                            |                        |                         |                         |                           |                           |                     |                 | s/c                         | s/c                      | s/c                      | s/c                |              | r/w                     | r/w       | ro             |

- DMA running
  - When '1' it shows that the DMA transfer is on going
  - When '0' it shows that no DMA transfer is active
- EN DMA is the DMA interface Enable
  - When '1' the DMA I/f is enabled
  - When '0' the DMA I/f is disabled
- Enable Interrupt
  - When '1' the Interrupt is enabled
  - When '0' the Interrupt never rises up
- Flush FIFOS
  - When set to '1' the FIFOS of the HPUCore are flushed. This bit is automatically cleared.
- LRxPAER Flush FIFOS
  - When set to '1' the FIFOS of the Left PAER interface are flushed. This bit is automatically cleared.
- RRxPAER Flush FIFOS
  - When set to '1' the FIFOS of the Right PAER interface are flushed. This bit is automatically cleared.
- AuxRxPAER Flush FIFOS
  - When set to '1' the FIFOS of the AUX PAER interface are flushed. This bit is automatically cleared.
- Reset DMA stream
  - By writing this bit to 1, the AXI stream master module is put in reset.
- Fulltimestamp
  - When set to '1' the Timestamp is 32 bit wide, when set to '0' the time stamp is 24 bit wide and the higher part is equal to 0x80.

The bits from 31 to 24 are used in debug to test the connection. For further details, look at the RTL code.

## 2.2 Loopback LR Configuration register (LPBK\_LR\_CNFG\_REG)

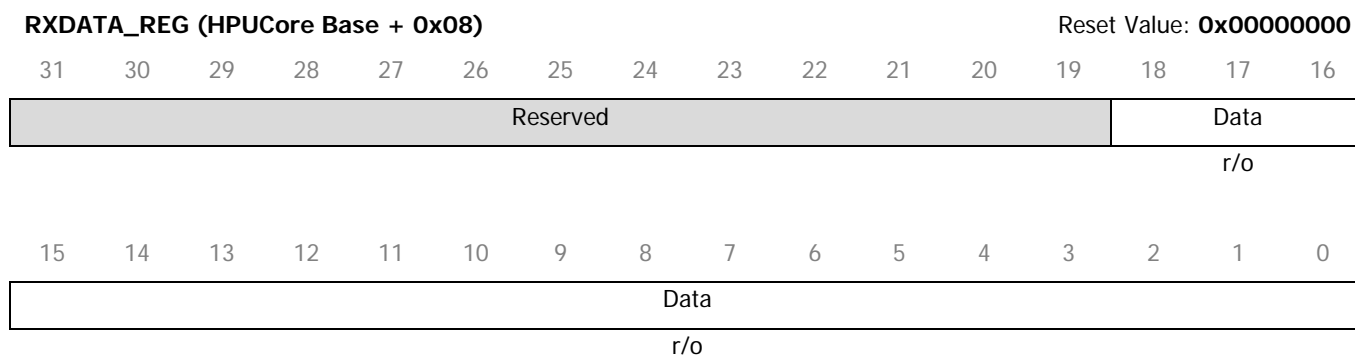
This register contains the configuration for Left and Right loopback.

| LPBK_LR_CNFG_REG (HPUCore Base + 0x04) |    |    |    |                          |    |    |    |                          |    |    |    |                          |    |    |    | Reset Value: 0x00000000 |  |  |  |
|--|----|----|----|--------------------------|----|----|----|--------------------------|----|----|----|--------------------------|----|----|----|-------------------------|--|--|--|
| 31                                     | 30 | 29 | 28 | 27                       | 26 | 25 | 24 | 23                       | 22 | 21 | 20 | 19                       | 18 | 17 | 16 |                         |  |  |  |
| Right RX chan 3 LPB cnfg               |    |    |    | Right RX chan 2 LPB cnfg |    |    |    | Right RX chan 1 LPB cnfg |    |    |    | Right RX chan 0 LPB cnfg |    |    |    |                         |  |  |  |
| r/w                                    |    |    |    | r/w                      |    |    |    | r/w                      |    |    |    | r/w                      |    |    |    |                         |  |  |  |
| 15                                     | 14 | 13 | 12 | 11                       | 10 | 9  | 8  | 7                        | 6  | 5  | 4  | 3                        | 2  | 1  | 0  |                         |  |  |  |
| Left RX chan 3 LPB cnfg                |    |    |    | Left RX chan 2 LPB cnfg  |    |    |    | Left RX chan 1 LPB cnfg  |    |    |    | Left RX chan 0 LPB cnfg  |    |    |    |                         |  |  |  |
| r/w                                    |    |    |    | r/w                      |    |    |    | r/w                      |    |    |    | r/w                      |    |    |    |                         |  |  |  |

The register is used in debug to test the connection. For further details, look at the RTL code.

### 2.3 RX Data Buffer register (RXDATA\_REG)

This register contains the data (read from the INFIFO) coming from the selected by N\_MuxAddr NMC. The format of the register is depicted into the figure below.



The meaning of this register is as explained in Table 3.

**NOTE: The reading of this register must follow the reading of the RX Time Buffer register (RXTIME\_REG).**

## 2.4 RX Time Buffer register (RXTIME\_REG)

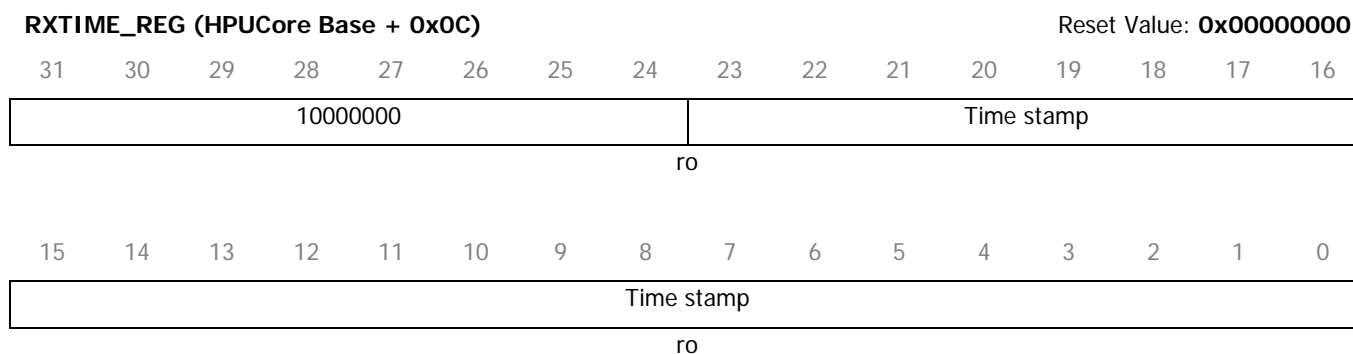
This register contains the time stamp associated to the received data (see Loopback LR Configuration register (LPBK\_LR\_CNFG\_REG))

This register contains the configuration for Left and Right loopback.

| LPBK_LR_CNFG_REG (HPUCore Base + 0x04) |    |    |    |                          |    |    |    |                          |    |    |    | Reset Value: 0x00000000  |    |    |    |
|--|----|----|----|--------------------------|----|----|----|--------------------------|----|----|----|--------------------------|----|----|----|
| 31                                     | 30 | 29 | 28 | 27                       | 26 | 25 | 24 | 23                       | 22 | 21 | 20 | 19                       | 18 | 17 | 16 |
| Right RX chan 3 LPB cnfg               |    |    |    | Right RX chan 2 LPB cnfg |    |    |    | Right RX chan 1 LPB cnfg |    |    |    | Right RX chan 0 LPB cnfg |    |    |    |
| r/w                                    |    |    |    | r/w                      |    |    |    | r/w                      |    |    |    | r/w                      |    |    |    |
| 15                                     | 14 | 13 | 12 | 11                       | 10 | 9  | 8  | 7                        | 6  | 5  | 4  | 3                        | 2  | 1  | 0  |
| Left RX chan 3 LPB cnfg                |    |    |    | Left RX chan 2 LPB cnfg  |    |    |    | Left RX chan 1 LPB cnfg  |    |    |    | Left RX chan 0 LPB cnfg  |    |    |    |
| r/w                                    |    |    |    | r/w                      |    |    |    | r/w                      |    |    |    | r/w                      |    |    |    |

The register is used in debug to test the connection. For further details, look at the RTL code.

RX Data Buffer register (RXDATA\_REG)) from the INFIFO.



The Time Stamp value read from this register is the Time Stamp that the HPUCore sticks to the Received data available into the RX Data Buffer register (RXDATA\_REG).

## 2.5 TX Data Buffer register (TXDATA\_REG)

This register is used to fill the OUTFIFO.

**TXDATA\_REG (HPUCore Base + 0x10)**

Reset Value: **0x00000000**



When writing to this register, keep in mind that it is used by the internal hw as follows:

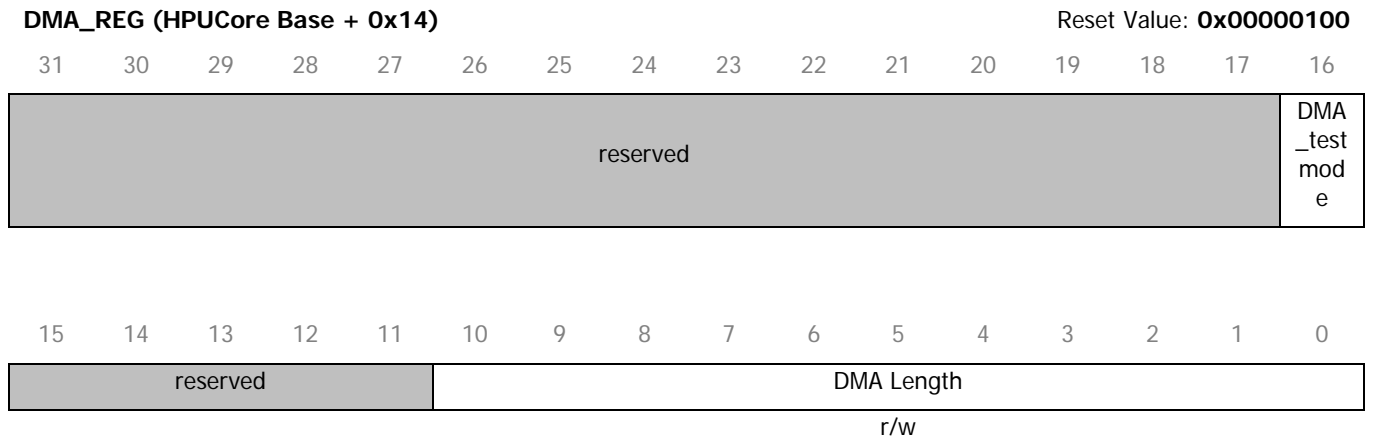
The register needs to be written twice to enable the correct behaviour.

The first data written into the register represents the time, elapsed which, the second data written into the register is delivered to the *loopback* module

## 2.6 DMA register (DMA\_REG)

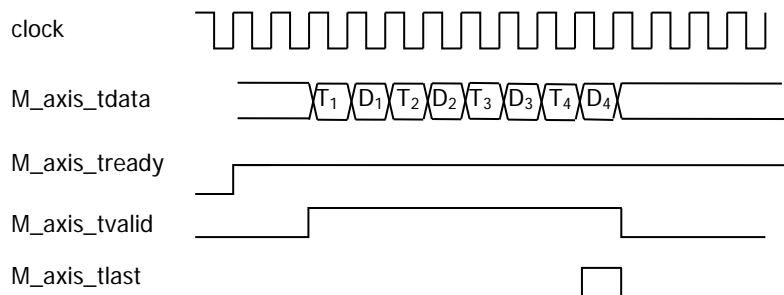
This register is used set the behaviour of the Axistream interface. It represents the number of data (32 bit size length) sent to the DMA interface.

NOTE: This register can be written only if CTRL\_REG.ENDMA='0'.



The DM\_test\_mode set to '1' enables the DMA to write consecutive incremental values at high rates.

For example, if it is set to 8, the burst from/to the DMA I/f will be in terms of 8 data length.





## 2.7 RAW Status Register (STAT\_RAW\_REG)

When read, this register gives a snapshot of the status of warning or errors signals. It is a Read Only register.

| STAT_RAW_REG (HPUCore Base + 0x18) |                       |                    |                    |          |    |                   |                 |                      |           |              |                      | Reset Value: 0x00000000 |                |                        |                |
|------------------------------------|-----------------------|--------------------|--------------------|----------|----|-------------------|-----------------|----------------------|-----------|--------------|----------------------|-------------------------|----------------|------------------------|----------------|
| 31                                 | 30                    | 29                 | 28                 | 27       | 26 | 25                | 24              | 23                   | 22        | 21           | 20                   | 19                      | 18             | 17                     | 16             |
| Reserved                           |                       |                    |                    |          |    |                   |                 |                      |           |              |                      | Glbl RX err_of          | Glbl RX err_to | Glbl RX err_tx         | Glbl RX err_ko |
|                                    |                       |                    |                    |          |    |                   |                 |                      |           |              |                      | ro                      | ro             | ro                     | ro             |
| 15                                 | 14                    | 13                 | 12                 | 11       | 10 | 9                 | 8               | 7                    | 6         | 5            | 4                    | 3                       | 2              | 1                      | 0              |
| RXFIFO > THRS                      | AUX RX PAER FIFO FULL | RRX PAER FIFO FULL | LRX PAER FIFO FULL | Reserved |    | RX FIFO Not Empty | RxBu fferR eady | Time Stam p Wra pped | Rese rved | TX Data Full | TX Data Almo st Full | TX Data Empt y          | RX Data Full   | RX Data Almo st Empt y | RX Data Empt y |
| ro                                 | ro                    | ro                 | ro                 |          |    | ro                | ro              | ro                   |           | ro           | ro                   | ro                      | ro             | ro                     | ro             |

- RxDataEmpty
  - When '0', the INFIFO is not empty
  - When '1' the INFIFO is empty
- RxDataAlmostEmpty
  - When '1' the INFIFO has 1 or 0 data to be read.
  - When '0' the INFIFO has more or equal two data to be read.
- RxDataFull
  - When '1' the INFIFO is full.
  - When '0' the INFIFO is not full.
- TxDataEmpty
  - When '0', the OUTFIFO is not empty
  - When '1' the OUTFIFO is empty
- TxDataAlmostFull
  - When '1' the OUTFIFO has 2047 or 2048 data within himself.
  - When '0' the OUTFIFO has less than 2047 data within himself.
- TxDataFull
  - When '1' the OUTFIFO is full.
  - When '0' the OUTFIFO is not full.
- Bias Finished
  - When '1' the Bias signals have been latched
  - When '0' no Bias signals have been latched
- Time stamp wrapped (this bit is high for one clock period only, when the counter wraps its value)
  - When '1' the counter inside the TimeStamp module has wrapped its value.
  - When '0' the counter inside the TimeStamp module has not yet wrapped its value
- RXBufferReady
  - When '1' the Rx Fifo has at least DMA\_REG value of data available
  - When '0' the Rx Fifo has less than DMA\_REG value of data available
- RXFifoNotEmpty
  - When '1' the RX Fifo is not empty.
  - When '0' the RX Fifo is empty
- LRXPaerFifoFull
  - When '1' the Left RX Fifo is not empty.
  - When '0' the Left RX Fifo is empty
- RRRXPaerFifoFull

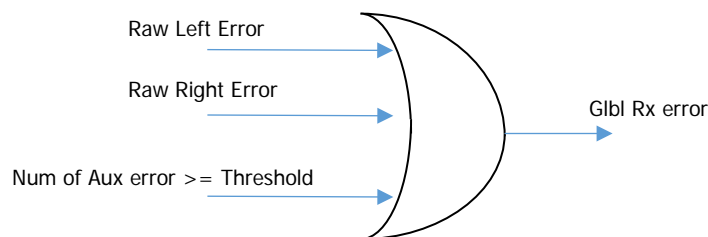
- When '1' the Right RX Fifo is not empty.
  - When '0' the Right RX Fifo is empty
- AuxRXPaerFifoFull
  - When '1' the Aux RX Fifo is not empty.
  - When '0' the Aux RX Fifo is empty
- RXfifo>Threshold
  - When '1' the Infifo has more elements with respect the value written into Fifo Threshold register (FIFO\_THRSH\_REG).
  - When '0' the Infifo has lesse elements with respect the value written into Fifo Threshold register (FIFO\_THRSH\_REG).
- Glbl Rx err ko
  - Global Rx err ko. (see [3] for further details)

It's an logic *or* between any unmasked errors detected on Left eye, Right eye and the number of errors in Aux interface overcoming the Aux Error counter threshold register (HSSAER\_AUX\_RX\_ERR\_THR\_REG).
- Glbl Rx err rx
  - Global Rx err rx. (see [3] for further details)

It's an logic *or* between any unmasked errors detected on Left eye, Right eye and the number of errors in Aux interface overcoming the Aux Error counter threshold register (HSSAER\_AUX\_RX\_ERR\_THR\_REG).
- Glbl Rx err to
  - Global Rx err to. (see [3] for further details)

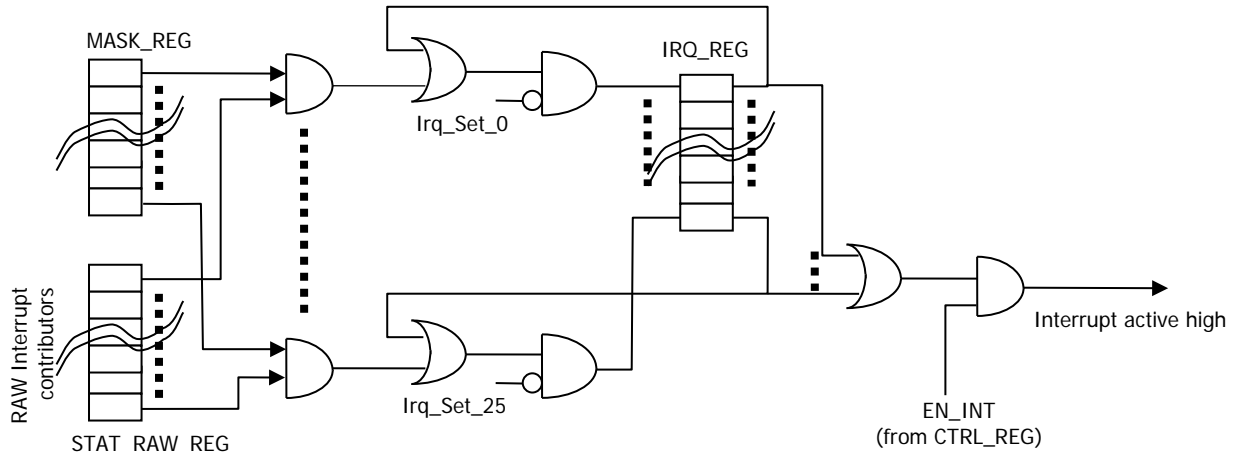
It's an logic *or* between any unmasked errors detected on Left eye, Right eye and the number of errors in Aux interface overcoming the Aux Error counter threshold register (HSSAER\_AUX\_RX\_ERR\_THR\_REG).
- Glbl Rx err of
  - Global Rx err of. (see [3] for further details)

It's an logic *or* between any unmasked errors detected on Left eye, Right eye and the number of errors in Aux interface overcoming the Aux Error counter threshold register (HSSAER\_AUX\_RX\_ERR\_THR\_REG).



## 2.8 IRQ Register (IRQ\_REG)

When read, this register gives the status of the collected warning or errors signals. It is a Read/Set register, i.e., to clear the warning/error bit the user has to write '1' on the corresponding bit position.



### IRQ\_REG (HPUCore Base + 0x1C)

Reset Value: **0x00000000**

| 31            | 30                    | 29                 | 28                 | 27       | 26 | 25 | 24                | 23              | 22                   | 21        | 20           | 19                   | 18             | 17             | 16                     |
|---------------|-----------------------|--------------------|--------------------|----------|----|----|-------------------|-----------------|----------------------|-----------|--------------|----------------------|----------------|----------------|------------------------|
| Reserved      |                       |                    |                    |          |    |    |                   |                 |                      |           |              | Glbl RX err_of       | Glbl RX err_to | Glbl RX err_rx | Glbl RX err_ko         |
|               |                       |                    |                    |          |    |    |                   |                 |                      |           |              | r/c                  | r/c            | r/c            | r/c                    |
| 15            | 14                    | 13                 | 12                 | 11       | 10 | 9  | 8                 | 7               | 6                    | 5         | 4            | 3                    | 2              | 1              | 0                      |
| RXFIFO > THRS | AUX RX PAER FIFO FULL | RRX PAER FIFO FULL | LRX PAER FIFO FULL | Reserved |    |    | RX FIFO Not Empty | RxBu fferR eady | Time Stam p Wra pped | Rese rved | TX Data Full | TX Data Almo st Full | TX Data Empt y | RX Data Full   | RX Data Almo st Empt y |
| r/c           | r/c                   | r/c                | r/c                |          |    |    | r/c               | r/c             | r/c                  |           | r/c          | r/c                  | r/c            | r/c            | r/c                    |

The meaning of the masked contributors of this register is the same of the RAW Status Register (STAT\_RAW\_REG).

## 2.9 Mask Register (MSK\_REG)

This is the Mask register used to mask the contributors for the interrupt signal.

**MSK\_REG (HPUCore Base + 0x20)**

Reset Value: **0x00000000**

| 31               | 30                                | 29                          | 28                          | 27       | 26 | 25                         | 24                    | 23                               | 22           | 21                 | 20                               | 19                       | 18                           | 17                                    | 16                           |
|------------------|-----------------------------------|-----------------------------|-----------------------------|----------|----|----------------------------|-----------------------|----------------------------------|--------------|--------------------|----------------------------------|--------------------------|------------------------------|---------------------------------------|------------------------------|
| Reserved         |                                   |                             |                             |          |    |                            |                       |                                  |              |                    |                                  | Glbl<br>RX<br>err_<br>of | Glbl<br>RX<br>err_<br>t<br>o | Glbl<br>RX<br>err_<br>t<br>x          | Glbl<br>RX<br>err_<br>k<br>o |
|                  |                                   |                             |                             |          |    |                            |                       |                                  |              |                    |                                  | r/w                      | r/w                          | r/w                                   | r/w                          |
| 15               | 14                                | 13                          | 12                          | 11       | 10 | 9                          | 8                     | 7                                | 6            | 5                  | 4                                | 3                        | 2                            | 1                                     | 0                            |
| RXFIFO<br>> THRS | AUX<br>RX<br>PAER<br>FIFO<br>FULL | RRX<br>PAER<br>FIFO<br>FULL | LRX<br>PAER<br>FIFO<br>FULL | Reserved |    | RX<br>FIFO<br>Not<br>Empty | RxBu<br>fferR<br>eady | Time<br>Stam<br>p<br>Wra<br>pped | Rese<br>rved | TX<br>Data<br>Full | TX<br>Data<br>Almo<br>st<br>Full | TX<br>Data<br>Empt<br>y  | RX<br>Data<br>Full           | RX<br>Data<br>Almo<br>st<br>Empt<br>y | RX<br>Data<br>Empt<br>y      |
| r/w              | r/w                               | r/w                         | r/w                         |          |    | r/w                        | r/w                   | r/w                              |              | r/w                | r/w                              | r/w                      | r/w                          | r/w                                   | r/w                          |

The meaning of the masked contributors of this register is the same of the RAW Status Register (STAT\_RAW\_REG).

## 2.10 Wrapping TimeStamp Register (WRAPTimeStamp\_REG)

This register is used to read how many times the internal 32bit counter of the TimeStamp module has wrapped its value.

In case the user writes any value in this register, it will be cleared and also the internal 32bit counter of the TimeStamp module will be cleared.

**WRAPTIMESTAMP\_REG (NEUELAB Base + 0x28)**

Reset Value: **0x00000000**

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

Wrapping times

r/c

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Wrapping times

r/c

## 2.11 HSSAER STATus register (HSSAER\_STAT\_REG)

This is the HSSAER Status register.

| HSSAER_STAT_REG (HPUCore Base + 0x34) |    |    |    |                      |                      |                      |                      |          |    |    |    | Reset Value: 0x00000000 |                      |                      |                      |
|---------------------------------------|----|----|----|----------------------|----------------------|----------------------|----------------------|----------|----|----|----|-------------------------|----------------------|----------------------|----------------------|
| 31                                    | 30 | 29 | 28 | 27                   | 26                   | 25                   | 24                   | 23       | 22 | 21 | 20 | 19                      | 18                   | 17                   | 16                   |
| Reserved                              |    |    |    | Chan3<br>Aux<br>run  | Chan2<br>Aux<br>run  | Chan1<br>Aux<br>run  | Chan0<br>Aux<br>run  | Reserved |    |    |    | Chan3<br>TX run         | Chan2<br>TX run      | Chan1<br>TX run      | Chan0<br>TX run      |
|                                       |    |    |    | ro                   | ro                   | ro                   | ro                   |          |    |    |    | ro                      | ro                   | ro                   | Ro                   |
| 15                                    | 14 | 13 | 12 | 11                   | 10                   | 9                    | 8                    | 7        | 6  | 5  | 4  | 3                       | 2                    | 1                    | 0                    |
| Reserved                              |    |    |    | Chan3<br>R RX<br>run | Chan2<br>R RX<br>run | Chan1<br>R RX<br>run | Chan0<br>R RX<br>run | Reserved |    |    |    | Chan3<br>L RX<br>run    | Chan2<br>L RX<br>run | Chan1<br>L RX<br>run | Chan0<br>L RX<br>run |
|                                       |    |    |    | ro                   | ro                   | ro                   | ro                   |          |    |    |    | ro                      | ro                   | ro                   | ro                   |

The user can read the status of the 4 channels of Left Rx Eye, Right Rx Eye, Aux Rx or Tx channel.

## 2.12 HSSAER RX Error register (HSSAER\_RX\_ERR\_REG)

This is the HSSAER Rx error register.

| HSSAER_RX_ERR_REG (HPUCore Base + 0x38) |                         |                         |                         |                         |                         |                         |                         |                         |                         |                         |                         |                         |                         | Reset Value: 0x00000000 |                         |
|---|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| 31                                      | 30                      | 29                      | 28                      | 27                      | 26                      | 25                      | 24                      | 23                      | 22                      | 21                      | 20                      | 19                      | 18                      | 17                      | 16                      |
| Chan3<br>R RX<br>err_of                 | Chan3<br>R RX<br>err_to | Chan3<br>R RX<br>err_rx | Chan3<br>R RX<br>err_ko | Chan2<br>R RX<br>err_of | Chan2<br>R RX<br>err_to | Chan2<br>R RX<br>err_rx | Chan2<br>R RX<br>err_ko | Chan1<br>R RX<br>err_of | Chan1<br>R RX<br>err_to | Chan1<br>R RX<br>err_rx | Chan1<br>R RX<br>err_ko | Chan0<br>R RX<br>err_of | Chan0<br>R RX<br>err_to | Chan0<br>R RX<br>err_rx | Chan0<br>R RX<br>err_ko |
| ro                                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | Ro                      |
| 15                                      | 14                      | 13                      | 12                      | 11                      | 10                      | 9                       | 8                       | 7                       | 6                       | 5                       | 4                       | 3                       | 2                       | 1                       | 0                       |
| Chan3<br>L RX<br>err_of                 | Chan3<br>L RX<br>err_to | Chan3<br>L RX<br>err_rx | Chan3<br>L RX<br>err_ko | Chan2<br>L RX<br>err_of | Chan2<br>L RX<br>err_to | Chan2<br>L RX<br>err_rx | Chan2<br>L RX<br>err_ko | Chan1<br>L RX<br>err_of | Chan1<br>L RX<br>err_to | Chan1<br>L RX<br>err_rx | Chan1<br>L RX<br>err_ko | Chan0<br>L RX<br>err_of | Chan0<br>L RX<br>err_to | Chan0<br>L RX<br>err_rx | Chan0<br>L RX<br>err_ko |
| ro                                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      | ro                      |

The user can read the error contributors for Left and Right 4 channels. See [3].

## 2.13 HSSAER RX MSK register (HSSAER\_RX\_MSK\_REG)

This is the HSSAER Rx mask register.

| HSSAER_RX_MSK_REG (HPUCore Base + 0x3C) |                         |                         |                         |                         |                         |                         |                         |                         |                         |                         |                         |                         |                         | Reset Value: 0x00000000 |                         |
|---|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| 31                                      | 30                      | 29                      | 28                      | 27                      | 26                      | 25                      | 24                      | 23                      | 22                      | 21                      | 20                      | 19                      | 18                      | 17                      | 16                      |
| Chan3<br>R RX<br>err_of                 | Chan3<br>R RX<br>err_to | Chan3<br>R RX<br>err_rx | Chan3<br>R RX<br>err_ko | Chan2<br>R RX<br>err_of | Chan2<br>R RX<br>err_to | Chan2<br>R RX<br>err_rx | Chan2<br>R RX<br>err_ko | Chan1<br>R RX<br>err_of | Chan1<br>R RX<br>err_to | Chan1<br>R RX<br>err_rx | Chan1<br>R RX<br>err_ko | Chan0<br>R RX<br>err_of | Chan0<br>R RX<br>err_to | Chan0<br>R RX<br>err_rx | Chan0<br>R RX<br>err_ko |
| rw                                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      |
| 15                                      | 14                      | 13                      | 12                      | 11                      | 10                      | 9                       | 8                       | 7                       | 6                       | 5                       | 4                       | 3                       | 2                       | 1                       | 0                       |
| Chan3<br>L RX<br>err_of                 | Chan3<br>L RX<br>err_to | Chan3<br>L RX<br>err_rx | Chan3<br>L RX<br>err_ko | Chan2<br>L RX<br>err_of | Chan2<br>L RX<br>err_to | Chan2<br>L RX<br>err_rx | Chan2<br>L RX<br>err_ko | Chan1<br>L RX<br>err_of | Chan1<br>L RX<br>err_to | Chan1<br>L RX<br>err_rx | Chan1<br>L RX<br>err_ko | Chan0<br>L RX<br>err_of | Chan0<br>L RX<br>err_to | Chan0<br>L RX<br>err_rx | Chan0<br>L RX<br>err_ko |
| rw                                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      | rw                      |

The user can mask (writing 0) or not (writing 1) the corresponding contributors of error register. See [3].



## 2.14 RX Control register (RX\_CTRL\_REG)

This is the HSSAER Left and Right Rx control register.

**RX\_CTRL\_REG (HPUCore Base + 0x40)**

Reset Value: **0x00000000**

|          |    |    |    |                       |           |           |           |          |    |    |    |            |    |             |               |
|----------|----|----|----|-----------------------|-----------|-----------|-----------|----------|----|----|----|------------|----|-------------|---------------|
| 31       | 30 | 29 | 28 | 27                    | 26        | 25        | 24        | 23       | 22 | 21 | 20 | 19         | 18 | 17          | 16            |
| Reserved |    |    |    | RRX HSSAER Channel En |           |           |           | Reserved |    |    |    | RRX GTP En |    | RRX PAER En | RRX HSSAER En |
|          |    |    |    | Channel 3             | Channel 2 | Channel 1 | Channel 0 |          |    |    |    |            |    |             |               |
|          |    |    |    | rw                    |           |           |           |          |    |    |    | rw         |    | rw          | rw            |
| 15       | 14 | 13 | 12 | 11                    | 10        | 9         | 8         | 7        | 6  | 5  | 4  | 3          | 2  | 1           | 0             |
| Reserved |    |    |    | LRX HSSAER Channel En |           |           |           | Reserved |    |    |    | LRX GTP En |    | LRX PAER En | LRX HSSAER En |
|          |    |    |    | Channel 3             | Channel 2 | Channel 1 | Channel 0 |          |    |    |    |            |    |             |               |
|          |    |    |    | rw                    |           |           |           |          |    |    |    | rw         |    | rw          | rw            |

Where:

- LRX HSSAER Enable
  - When '0', the Left HSSAER interface is not enabled
  - When '1', the Left HSSAER interface is enabled
- LRX PAER Enable
  - When '0', the Left PAER interface is not enabled
  - When '1', the Left PAER interface is enabled
- LRX GTP Enable
  - When '0', the Left GTP interface is not enabled
  - When '1', the Left GTP interface is enabled
- LRX HSSAER Channel Enable
  - Write 1 in the corresponding channel to enable it
- RRX HSSAER Enable
  - When '0', the Right HSSAER interface is not enabled
  - When '1', the Right HSSAER interface is enabled
- RRX PAER Enable
  - When '0', the Right PAER interface is not enabled
  - When '1', the Right PAER interface is enabled
- RRX GTP Enable
  - When '0', the Right GTP interface is not enabled
  - When '1', the Right GTP interface is enabled
- RRX HSSAER Channel Enable
  - Write 1 in the corresponding channel to enable it

## 2.15 TX Control register (TX\_CTRL\_REG)

This is the HSSAER Tx control register.

| TX_CTRL_REG (HPUCore Base + 0x44) |    |    |    |                      |           |           |           |          |    |    |    | Reset Value: 0x00000000 |    |            |    |
|-----------------------------------|----|----|----|----------------------|-----------|-----------|-----------|----------|----|----|----|-------------------------|----|------------|----|
| 31                                | 30 | 29 | 28 | 27                   | 26        | 25        | 24        | 23       | 22 | 21 | 20 | 19                      | 18 | 17         | 16 |
| Reserved                          |    |    |    |                      |           |           |           |          |    |    |    |                         |    |            |    |
| rw                                |    |    |    |                      |           |           |           |          |    |    |    | rw                      |    |            |    |
| 15                                | 14 | 13 | 12 | 11                   | 10        | 9         | 8         | 7        | 6  | 5  | 4  | 3                       | 2  | 1          | 0  |
| Reserved                          |    |    |    | TX HSSAER Channel En |           |           |           | Reserved |    |    |    | TX GTP En               |    | TX PAER En |    |
|                                   |    |    |    | Channel 3            | Channel 2 | Channel 1 | Channel 0 |          |    |    |    |                         |    |            |    |
|                                   |    |    |    | rw                   |           |           |           |          |    |    |    | rw                      |    | rw         |    |

Where:

- TX HSSAER Enable
  - When '0', the Left HSSAER interface is not enabled
  - When '1', the Left HSSAER interface is enabled
- TX PAER Enable
  - When '0', the Left PAER interface is not enabled
  - When '1', the Left PAER interface is enabled
- TX GTP Enable
  - When '0', the Left GTP interface is not enabled
  - When '1', the Left GTP interface is enabled
- TX HSSAER Channel Enable
  - Write 1 in the corresponding channel to enable it

## 2.16 RX PAER Configuration register (RX\_PAER\_CFNG\_REG)

This is the RX PAER configuration register.

| RX_PAER_CFNG_REG (HPUCore Base + 0x48) |    |    |    |    |    |    |    |                      |    |                         |          |    |                       | Reset Value: 0x00000000 |    |
|--|----|----|----|----|----|----|----|----------------------|----|-------------------------|----------|----|-----------------------|-------------------------|----|
| 31                                     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23                   | 22 | 21                      | 20       | 19 | 18                    | 17                      | 16 |
| RXPAER Ack release Delay               |    |    |    |    |    |    |    | RXPAER Ack Set Delay |    |                         |          |    |                       |                         |    |
| rw                                     |    |    |    |    |    |    |    | rw                   |    |                         |          |    |                       |                         |    |
| 15                                     | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7                    | 6  | 5                       | 4        | 3  | 2                     | 1                       | 0  |
| RXPAER Data Sample Delay               |    |    |    |    |    |    |    | Reserved             |    | RXPAER ignore Fifo Full | Reserved |    | RXPAER Ack Active Lvl | RXPAER Req Active Lvl   |    |
| rw                                     |    |    |    |    |    |    |    |                      |    | rw                      |          |    | rw                    | rw                      |    |

Where:

- RXPAER Req Active level
  - When '0', the Request signal is active low
  - When '1', the Request signal is active high
- RXPAER Ack Active level
  - When '0', the Acknowledge signal is active low
  - When '1', the Acknowledge signal is active high
- RX PAER ignore Fifo Full
  - When '0', the Fifo Full stops the acknowledge signal
  - When '1', the Fifo Full doesn't stop the acknowledge signal by acknowledging the request
- RXPAER Data Sample Delay
  - This is the number of system clock used to sample the PAER address.
- RXPAER Ack Set Delay
  - This is the number of system clock used to set the ACK signal after that the request becomes active
- RXPAER Ack Release Delay
  - This is the number of system clock used to release the ACK signal after that the request becomes active

## 2.17 TX PAER Configuration register (TX\_PAER\_CFNG\_REG)

This is the TX PAER configuration register.

| TX_PAER_CFNG_REG (HPUCore Base + 0x4C) |    |    |    |    |    |    |    |    |    |    |    | Reset Value: 0x00000000 |                                    |                                    |    |
|--|----|----|----|----|----|----|----|----|----|----|----|-------------------------|------------------------------------|------------------------------------|----|
| 31                                     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19                      | 18                                 | 17                                 | 16 |
| Reserved                               |    |    |    |    |    |    |    |    |    |    |    |                         |                                    |                                    |    |
| 15                                     | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3                       | 2                                  | 1                                  | 0  |
| Reserved                               |    |    |    |    |    |    |    |    |    |    |    |                         | TXPAE<br>R<br>Ack<br>Active<br>Lvl | TXPAE<br>R<br>Req<br>Active<br>Lvl |    |
|  |    |    |    |    |    |    |    |    |    |    |    |                         | rW                                 | rW                                 |    |

Where:

- TXPAER Req Active level
  - When '0', the Request signal is active low
  - When '1', the Request signal is active high
- TXPAER Ack Active level
  - When '0', the Acknowledge signal is active low
  - When '1', the Acknowledge signal is active high

## 2.18 IP Configuration register (IP\_CFNG\_REG)

This is the HPUCore configuration register. It shows how the HPUCore has been implemented in terms of features.

**IP\_CFNG\_REG (HPUCore Base + 0x50)**

Reset Value: **0x0000????**

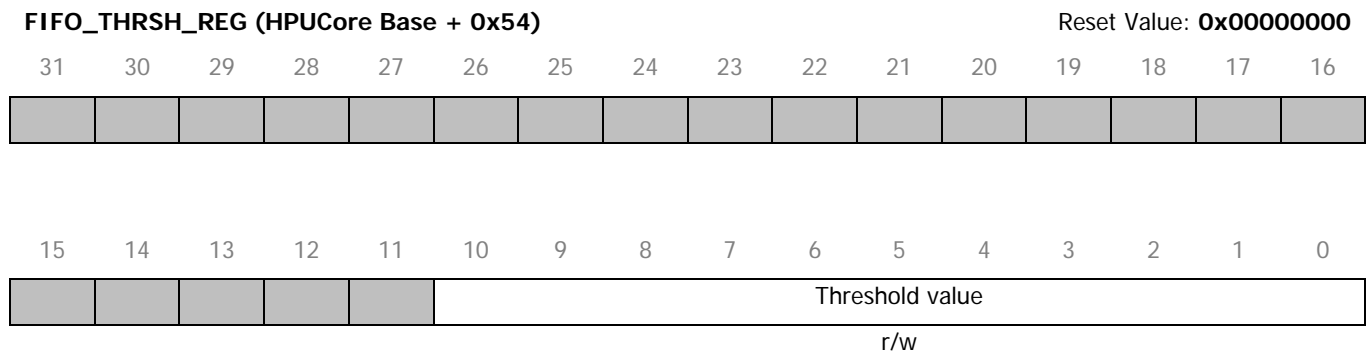
|          |    |                           |    |              |                  |                   |                         |          |    |                           |    |              |                  |                   |                         |
|----------|----|---------------------------|----|--------------|------------------|-------------------|-------------------------|----------|----|---------------------------|----|--------------|------------------|-------------------|-------------------------|
| 31       | 30 | 29                        | 28 | 27           | 26               | 25                | 24                      | 23       | 22 | 21                        | 20 | 19           | 18               | 17                | 16                      |
| Reserved |    |                           |    |              |                  |                   |                         |          |    |                           |    |              |                  |                   |                         |
| 15       | 14 | 13                        | 12 | 11           | 10               | 9                 | 8                       | 7        | 6  | 5                         | 4  | 3            | 2                | 1                 | 0                       |
| Reserved |    | TX HSSAER<br>#channels -1 |    | Reserv<br>ed | HAS<br>TX<br>GTP | Has<br>TX<br>PAER | Has<br>TX<br>HSSAE<br>R | Reserved |    | RX HSSAER<br>#channels -1 |    | Reserv<br>ed | HAS<br>RX<br>GTP | Has<br>RX<br>PAER | Has<br>RX<br>HSSAE<br>R |
|          |    |                           |    |              |                  |                   |                         |          |    |                           |    |              | ro               | ro                | ro                      |

Where:

- Has RX HSSAER
  - When '0', the IP doesn't have the RX HSSAER I/f
  - When '0', the IP has the RX HSSAER I/f
- Has RX PAER
  - When '0', the IP doesn't have the RX PAER I/f
  - When '0', the IP has the RX PAER I/f
- Has RX GTP
  - When '0', the IP doesn't have the RX GTP I/f
  - When '0', the IP has the RX GTP I/f
- RX HSSAER #channels -1
  - This shows the number of channels of RX HSSAER I/f. For instance, if it is 2'b10, it means that the RX HSSAER interface has 3 channels
- Has TX HSSAER
  - When '0', the IP doesn't have the TX HSSAER I/f
  - When '0', the IP has the TX HSSAER I/f
- Has TX PAER
  - When '0', the IP doesn't have the TX PAER I/f
  - When '0', the IP has the TX PAER I/f
- Has TX GTP
  - When '0', the IP doesn't have the TX GTP I/f
  - When '0', the IP has the TX GTP I/f
- RX HSSAER #channels -1
  - This shows the number of channels of RX HSSAER I/f. For instance, if it is 2'b10, it means that the RX HSSAER interface has 3 channels

## 2.19 Fifo Threshold register (FIFO\_THRSH\_REG)

This register contains the number of elements of the INFIFO after which the “RXFIFO > THRS” of the IRQ Register (IRQ\_REG) bit goes high.



## 2.20 LoopBack AUX Configuration (LPBK\_AUX\_CNFG\_REG)

This register contains the configuration for the AUX interface loopback.

| LPBK_AUX_CNFG_REG (HPUCore Base + 0x58) |    |    |    |                        |    |    |    |                        |    |    |    |                        |    |    |    | Reset Value: 0x00000000 |  |
|---|----|----|----|------------------------|----|----|----|------------------------|----|----|----|------------------------|----|----|----|-------------------------|--|
| 31                                      | 30 | 29 | 28 | 27                     | 26 | 25 | 24 | 23                     | 22 | 21 | 20 | 19                     | 18 | 17 | 16 |                         |  |
| Reserved                                |    |    |    |                        |    |    |    |                        |    |    |    |                        |    |    |    |                         |  |
| 15                                      | 14 | 13 | 12 | 11                     | 10 | 9  | 8  | 7                      | 6  | 5  | 4  | 3                      | 2  | 1  | 0  |                         |  |
| AUX RX chan 3 LPB cnfg                  |    |    |    | AUX RX chan 2 LPB cnfg |    |    |    | AUX RX chan 1 LPB cnfg |    |    |    | AUX RX chan 0 LPB cnfg |    |    |    |                         |  |
| r/w                                     |    |    |    | r/w                    |    |    |    | r/w                    |    |    |    | r/w                    |    |    |    |                         |  |

The register is used in debug to test the connection. For further details, look at the RTL code.

## 2.21 Identification register (ID\_REG)

This register contains the ID of the NeuElab.

**ID\_REG (HPUCore Base + 0x5C)**

Reset Value: **48505522**

|     |    |    |    |    |    |    |    |       |    |    |    |       |    |    |    |
|-----|----|----|----|----|----|----|----|-------|----|----|----|-------|----|----|----|
| 31  | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23    | 22 | 21 | 20 | 19    | 18 | 17 | 16 |
| H   |    |    |    |    |    |    |    | P     |    |    |    |       |    |    |    |
| r/o |    |    |    |    |    |    |    | r/o   |    |    |    |       |    |    |    |
| 15  | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7     | 6  | 5  | 4  | 3     | 2  | 1  | 0  |
| U   |    |    |    |    |    |    |    | Major |    |    |    | Minor |    |    |    |
| r/o |    |    |    |    |    |    |    | r/o   |    |    |    | r/o   |    |    |    |

Minor =2;

Major = 2;



## 2.22 AUXiliary RX Control register (AUX\_RX\_CTRL\_REG)

This is the Auxiliary Rx control register.

| AUX_RX_CTRL_REG (HPUCore Base + 0x60) |    |    |    |                       |              |              |              |          |    |    |    | Reset Value: 0x00000000 |                   |                         |    |
|---------------------------------------|----|----|----|-----------------------|--------------|--------------|--------------|----------|----|----|----|-------------------------|-------------------|-------------------------|----|
| 31                                    | 30 | 29 | 28 | 27                    | 26           | 25           | 24           | 23       | 22 | 21 | 20 | 19                      | 18                | 17                      | 16 |
| Reserved                              |    |    |    |                       |              |              |              |          |    |    |    |                         |                   |                         |    |
| rw                                    |    |    |    |                       |              |              |              |          |    |    |    |                         |                   |                         |    |
| 15                                    | 14 | 13 | 12 | 11                    | 10           | 9            | 8            | 7        | 6  | 5  | 4  | 3                       | 2                 | 1                       | 0  |
| Reserved                              |    |    |    | AUX HSSAER Channel En |              |              |              | Reserved |    |    |    | AUX<br>GTP<br>En        | AUX<br>PAER<br>En | AUX<br>HSSAE<br>R<br>En |    |
|                                       |    |    |    | Channel<br>3          | Channel<br>2 | Channel<br>1 | Channel<br>0 |          |    |    |    |                         |                   |                         |    |
| rw                                    |    |    |    | rw                    |              |              |              | rw       |    |    |    | rw                      | rw                | rw                      |    |

Where:

- AUX HSSAER Enable
  - When '0', the AUX HSSAER interface is not enabled
  - When '1', the AUX HSSAER interface is enabled
- AUX PAER Enable
  - When '0', the AUX PAER interface is not enabled
  - When '1', the AUX PAER interface is enabled
- AUX GTP Enable
  - When '0', the AUX GTP interface is not enabled
  - When '1', the AUX GTP interface is enabled
- AUX HSSAER Channel Enable
  - Write 1 in the corresponding channel to enable it

## 2.23 HSSAER AUX RX Error register (HSSAER\_AUX\_RX\_ERR\_REG)

This is the HSSAER Rx error register.

| HSSAER_AUX_RX_ERR_REG (HPUCore Base + 0x64) |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              | Reset Value: 0x00000000      |                              |
|---|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|
| 31  | 30                           | 29                           | 28                           | 27                           | 26                           | 25                           | 24                           | 23                           | 22                           | 21                           | 20                           | 19                           | 18                           | 17                           | 16                           |
|   |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |
| 15  | 14                           | 13                           | 12                           | 11                           | 10                           | 9                            | 8                            | 7                            | 6                            | 5                            | 4                            | 3                            | 2                            | 1                            | 0                            |
| Chan3<br>AUX<br>RX<br>err_of                | Chan3<br>AUX<br>RX<br>err_to | Chan3<br>AUX<br>RX<br>err_rx | Chan3<br>AUX<br>RX<br>err_ko | Chan2<br>AUX<br>RX<br>err_of | Chan2<br>AUX<br>RX<br>err_to | Chan2<br>AUX<br>RX<br>err_rx | Chan2<br>AUX<br>RX<br>err_ko | Chan1<br>AUX<br>RX<br>err_of | Chan1<br>AUX<br>RX<br>err_to | Chan1<br>AUX<br>RX<br>err_rx | Chan1<br>AUX<br>RX<br>err_ko | Chan0<br>AUX<br>RX<br>err_of | Chan0<br>AUX<br>RX<br>err_to | Chan0<br>AUX<br>RX<br>err_rx | Chan0<br>AUX<br>RX<br>err_ko |
| ro  | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           | ro                           |

The user can read the error contributors for the 4 channels of the AUX interface. See [3].

## 2.24 HSSAER AUX RX MSK register (HSSAER\_AUX\_RX\_MSK\_REG)

This is the HSSAER AUX Rx mask register.

| HSSAER_AUX_RX_MSK_REG (HPUCore Base + 0x68) |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              | Reset Value: 0x00000000      |                              |                              |                              |
|---|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|------------------------------|
| 31  | 30                           | 29                           | 28                           | 27                           | 26                           | 25                           | 24                           | 23                           | 22                           | 21                           | 20                           | 19                           | 18                           | 17                           | 16                           |
|   |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |                              |
| 15  | 14                           | 13                           | 12                           | 11                           | 10                           | 9                            | 8                            | 7                            | 6                            | 5                            | 4                            | 3                            | 2                            | 1                            | 0                            |
| Chan3<br>AUX<br>RX<br>err_of                | Chan3<br>AUX<br>RX<br>err_to | Chan3<br>AUX<br>RX<br>err_rx | Chan3<br>AUX<br>RX<br>err_ko | Chan2<br>AUX<br>RX<br>err_of | Chan2<br>AUX<br>RX<br>err_to | Chan2<br>AUX<br>RX<br>err_rx | Chan2<br>AUX<br>RX<br>err_ko | Chan1<br>AUX<br>RX<br>err_of | Chan1<br>AUX<br>RX<br>err_to | Chan1<br>AUX<br>RX<br>err_rx | Chan1<br>AUX<br>RX<br>err_ko | Chan0<br>AUX<br>RX<br>err_of | Chan0<br>AUX<br>RX<br>err_to | Chan0<br>AUX<br>RX<br>err_rx | Chan0<br>AUX<br>RX<br>err_ko |
| rw  | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           | rw                           |

The user can mask (writing 0) or not (writing 1) the corresponding contributors of error register. See [3].

## 2.25 Aux Error counter threshold register (HSSAER\_AUX\_RX\_ERR\_THR\_REG)

This register is used for setting the threshold of the AUX Rx counter error.

As soon as the number of the corresponding error overcome the threshold here set, the interrupt related will be raised if opportunely masked.

**HSSAER\_AUX\_RX\_ERR\_THR\_REG (HPUCore Base + 0x6C)**

Reset Value: **0x10101010**

|                        |    |    |    |    |    |    |    |                         |    |    |    |    |    |    |    |
|------------------------|----|----|----|----|----|----|----|-------------------------|----|----|----|----|----|----|----|
| 31                     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23                      | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Number of AUX of error |    |    |    |    |    |    |    | Number of AUX to error  |    |    |    |    |    |    |    |
| r/w                    |    |    |    |    |    |    |    | r/w                     |    |    |    |    |    |    |    |
| 15                     | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7                       | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Number of AUX rx error |    |    |    |    |    |    |    | Number of AUX ko errors |    |    |    |    |    |    |    |
| r/w                    |    |    |    |    |    |    |    | r/w                     |    |    |    |    |    |    |    |

- Number of AUX ko errors
  - The number of ko errors which, if overcome, can raise an interrupt
- Number of AUX rx errors
  - The number of rx errors which, if overcome, can raise an interrupt
- Number of AUX to errors
  - The number of to errors which, if overcome, can raise an interrupt
- Number of AUX of errors
  - The number of of errors which, if overcome, can raise an interrupt

## 2.26 Aux Error counter CH0 register (HSSAER\_AUX\_RX\_ERR\_CH0\_REG)

This register is used to read the number of errors occurred in HSSAER AUX channel 0 lines.

Reading this register we also clear it.

**HSSAER\_AUX\_RX\_ERR\_CH0\_REG (HPUCore Base + 0x70)**

Reset Value: **0x00000000**

|                        |    |    |    |    |    |    |    |                         |    |    |    |    |    |    |    |
|------------------------|----|----|----|----|----|----|----|-------------------------|----|----|----|----|----|----|----|
| 31                     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23                      | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Number of CH0 of error |    |    |    |    |    |    |    | Number of CH0 to error  |    |    |    |    |    |    |    |
| r/c                    |    |    |    |    |    |    |    | r/c                     |    |    |    |    |    |    |    |
| 15                     | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7                       | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Number of CH0 rx error |    |    |    |    |    |    |    | Number of CH0 ko errors |    |    |    |    |    |    |    |
| r/c                    |    |    |    |    |    |    |    | r/c                     |    |    |    |    |    |    |    |

- Number of CH0 ko errors
  - The number of ko errors
- Number of CH0 rx errors
  - The number of rx errors
- Number of CH0 to errors
  - The number of to errors
- Number of CH0 of errors
  - The number of of errors

**2.27 Aux Error counter CH1 register (HSSAER\_AUX\_RX\_ERR\_CH1\_REG)**

This register is used to read the number of errors occurred in HSSAER AUX channel 1 lines.

Reading this register we also clear it.

**HSSAER\_AUX\_RX\_ERR\_CH1\_REG (HPUCore Base + 0x74)**

Reset Value: **0x00000000**

|                        |    |    |    |    |    |    |    |                         |    |    |    |    |    |    |    |
|------------------------|----|----|----|----|----|----|----|-------------------------|----|----|----|----|----|----|----|
| 31                     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23                      | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Number of CH1 of error |    |    |    |    |    |    |    | Number of CH1 to error  |    |    |    |    |    |    |    |
| r/c                    |    |    |    |    |    |    |    | r/c                     |    |    |    |    |    |    |    |
| 15                     | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7                       | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Number of CH1 rx error |    |    |    |    |    |    |    | Number of CH1 ko errors |    |    |    |    |    |    |    |
| r/c                    |    |    |    |    |    |    |    | r/c                     |    |    |    |    |    |    |    |

- Number of CH1 ko errors
  - The number of ko errors
- Number of CH1 rx errors
  - The number of rx errors
- Number of CH1 to errors
  - The number of to errors
- Number of CH1 of errors
  - The number of of errors

## 2.28 Aux Error counter CH2 register (HSSAER\_AUX\_RX\_ERR\_CH2\_REG)

This register is used to read the number of errors occurred in HSSAER AUX channel 2 lines.

Reading this register we also clear it.

**HSSAER\_AUX\_RX\_ERR\_CH2\_REG (HPUCore Base + 0x78)**

Reset Value: **0x00000000**

|                        |    |    |    |    |    |    |    |                         |    |    |    |    |    |    |    |
|------------------------|----|----|----|----|----|----|----|-------------------------|----|----|----|----|----|----|----|
| 31                     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23                      | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Number of CH2 of error |    |    |    |    |    |    |    | Number of CH2 to error  |    |    |    |    |    |    |    |
| r/c                    |    |    |    |    |    |    |    | r/c                     |    |    |    |    |    |    |    |
| 15                     | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7                       | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Number of CH2 rx error |    |    |    |    |    |    |    | Number of CH2 ko errors |    |    |    |    |    |    |    |
| r/c                    |    |    |    |    |    |    |    | r/c                     |    |    |    |    |    |    |    |

- Number of CH2 ko errors
  - The number of ko errors
- Number of CH2 rx errors
  - The number of rx errors
- Number of CH2 to errors
  - The number of to errors
- Number of CH2 of errors
  - The number of of errors

## 2.29 Aux Error counter CH3 register (HSSAER\_AUX\_RX\_ERR\_CH3\_REG)

This register is used to read the number of errors occurred in HSSAER AUX channel 3 lines.

Reading this register we also clear it.

**HSSAER\_AUX\_RX\_ERR\_CH3\_REG (HPUCore Base + 0x7C)**

Reset Value: **0x00000000**

|                        |    |    |    |    |    |    |    |                         |    |    |    |    |    |    |    |
|------------------------|----|----|----|----|----|----|----|-------------------------|----|----|----|----|----|----|----|
| 31                     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23                      | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Number of CH3 of error |    |    |    |    |    |    |    | Number of CH3 to error  |    |    |    |    |    |    |    |
| r/c                    |    |    |    |    |    |    |    | r/c                     |    |    |    |    |    |    |    |
| 15                     | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7                       | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Number of CH3 rx error |    |    |    |    |    |    |    | Number of CH3 ko errors |    |    |    |    |    |    |    |
| r/c                    |    |    |    |    |    |    |    | r/c                     |    |    |    |    |    |    |    |

- Number of CH3 ko errors
  - The number of ko errors
- Number of CH3 rx errors
  - The number of rx errors
- Number of CH3 to errors
  - The number of to errors
- Number of CH3 of errors
  - The number of of errors



### **3 References**

- [1] ARM AMBA AXI protocol v2.0
- [2] "Combining the ADS1202 with an FPGA Digital Filter for Current Measurement in Motor Control Applications", Texas Instruments, Application Report SBAA094 – June 2003.
- [3] "Asynchronous DC-Free Serial Protocol for Event-Based AER Systems", P. Motto Ros, M. Crepaldi, C. Bartolozzi and D. Demarchi, 2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)

## **4 Appendixes**