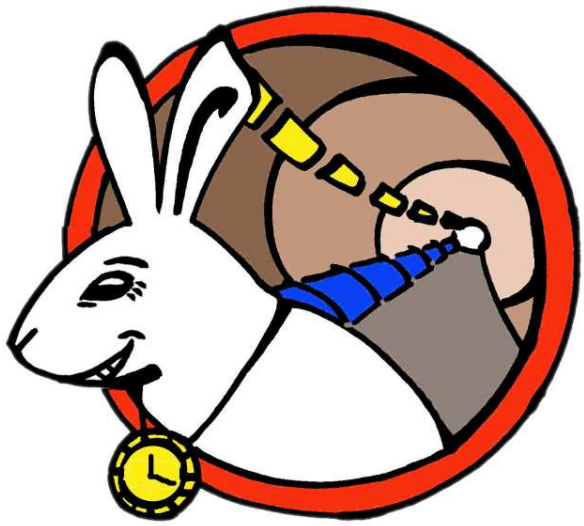
White Rabbit Switch HDL software interface

version 1.1
(07-05-2013)

Grzegorz Daniluk
CERN BE-CO-HT

May 2013



Contents

1 Introduction

This document describes Wishbone configuration registers of the modules inside the Gateware of the White Rabbit Switch. Each section gives a short description of the module's role in the Switch design and is followed by a detailed description of each configuration register available through Wishbone bus.

2 General overview

Figure 1 shows the internals of the WR Switch HDL design. It contains numerous modules connected with the Wishbone Crossbar. Each of them has a Wishbone Slave interface and a number of configuration registers that are read/written from main CPU through the CPU EBI/WB bridge (WB Master interface). Table 1 contains Wishbone base address of each module. Blue arrows in figure 1 represent WR Frabric interface connections responsible for carrying Ethernet frames between the Endpoints, Switching Core and Network Interface Controller.

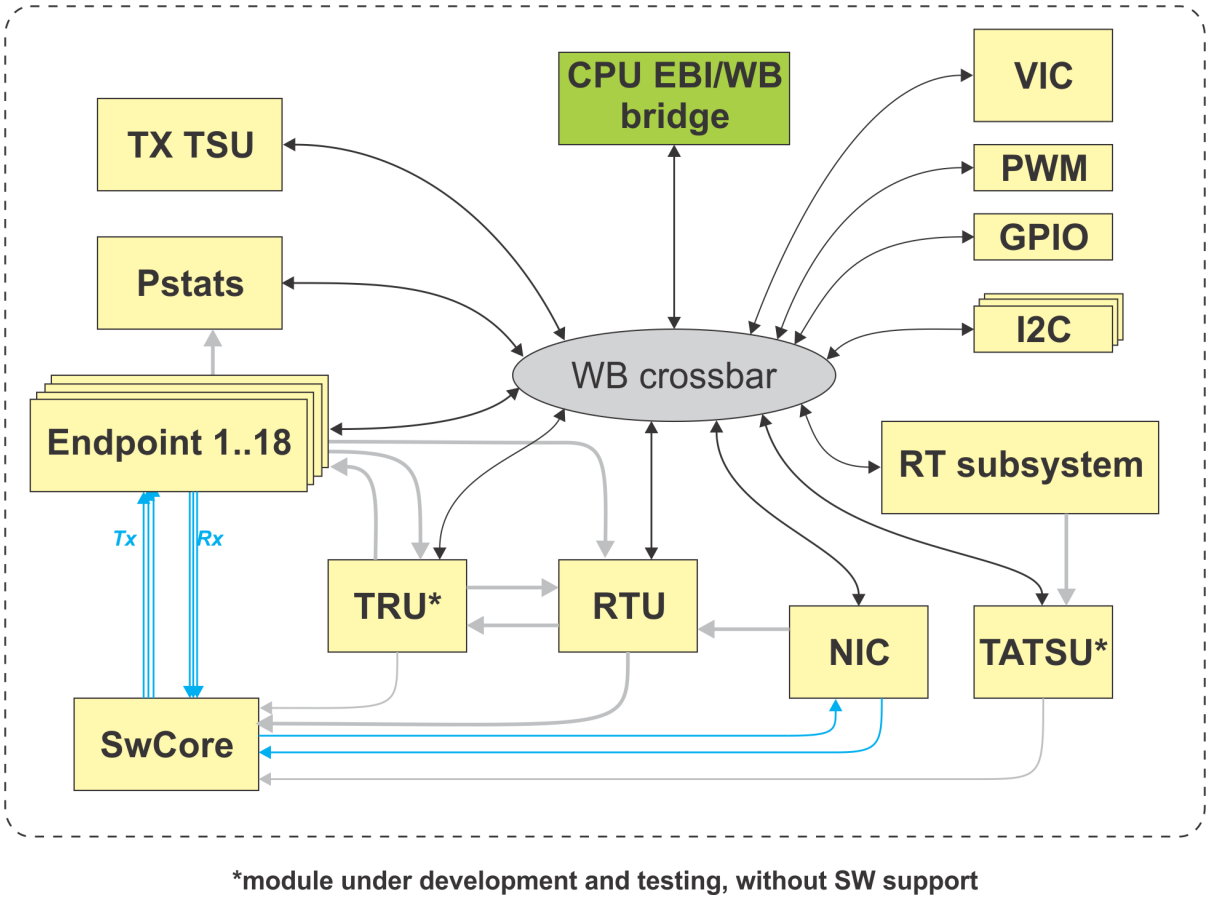


Figure 1: Top HDL design of the WR Switch

module name	base address
Real-Time subsystem	0x00000
Network Interface Controller (NIC)	0x20000
Endpoint fanout	0x30000
Vectored Interrupt Controller (VIC)	0x50000
Tx Timestamping Unit (TX TSU)	0x51000
GPIO port (GPIO)	0x53000
I^2C master (I2C0)	0x54000
I^2C master (I2C1)	0x55000
I^2C master (Sensors_I2C)	0x56000
PWM Controller	0x57000
Topology Resolution Unit (TRU)	0x58000
Time-Aware Traffic Shapper Unit (TATSU)	0x59000
Routing Table Unit (RTU)	0x60000
Per-port Statistics (Pstats)	0x70000

Table 1: Wishbone base addresses of modules inside WR Switch HDL

3 Modules description

3.1 Real-Time Subsystem (RT subsystem)

Contains modules responsible for the timekeeping. The components are internally connected through WB crossbar (fig. 2) and controlled from Lattice Mico 32 and main CPU (through primary WB crossbar in the top design). Table 2 contains Wishbone base address of each module inside the Real-Time Subsystem.

module name	base address
Dual-port RAM	0x00000
debug UART	0x10000
Soft-PLL	0x10100
SPI	0x10200
GPIO	0x10300
Timer block (TICS)	0x10400
1-PPS generator (PPSGEN)	0x10500

Table 2: Wishbone base addresses of modules inside the Real-Time Subsystem

3.1.1 LatticeMico32 (LM32)

Description:
Soft-core processor executing software implementation of PLLs (with *Soft-PLL* module).

Wishbone interface:
LM32 is a Wishbone Master, does not have any WB configuration registers.

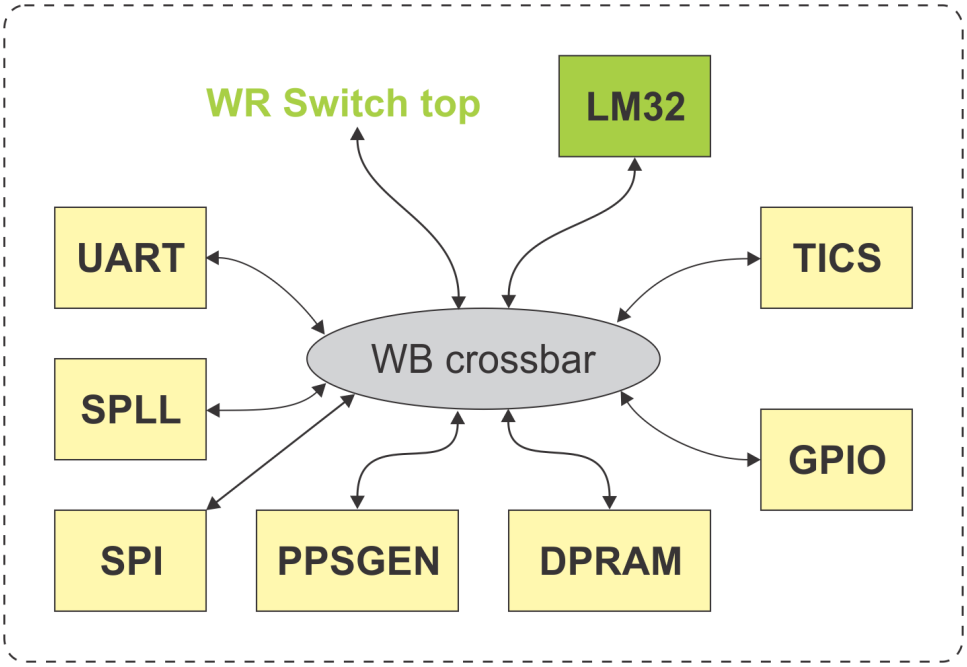


Figure 2: Internal layout of Real-Time Subsystem block

3.1.2 Dual-port RAM (DPRAM)

Description:
Instruction and data memory for *LM32* soft-core processor controlling *Soft-PLL*. It has to be programmed with *LM32* binary from the main CPU when the WR Switch boots.

3.1.3 Debug UART

Description:
UART driven by *LM32*, outputs debug information from Software PLLs.

Wishbone interface:
Only *LM32* talks to this interface, WR Switch software does not access it.

3.1.4 Soft-PLL (SPLL)

Description:
HDL part of Soft-PLL implementation. Controlled over Wishbone from *LM32* software.

Wishbone interface:
Only *LM32* talks to this interface, WR Switch software does not access it.

3.1.5 SPI

Description:

Used by software running on *LM32* to adjust the tunable oscillators (part of SoftPLL implementation).

Wishbone interface:
Only *LM32* talks to this interface, WR Switch software does not access it.

3.1.6 GPIO

Description:
Controls GPIO lines through Wishbone interface. All of the signals currently used are output-only. They are mostly (besides GPIO2) driven from *LM32* software as part of SoftPLL algorithm implementation.

Signals controlled by the module:

GPIO No.	direction	used by	description
0	output	*LM32*	selects system clock for FPGA modules to be a startup clock (0) or the clock coming from from PLL (1) i.e. aligned to WR time
1	output	*LM32*	resets AD9516 clock generator
2	output	*WR Switch SW*	active high reset of *LM32* softcore, used by the *LM32* firmware loader while it is being stored to DPRAM
3	output	*LM32*	active low reset of WR Switch peripherals (Endpoint, Switching Core, Topology Resolution Unit, GTX Ser/Des, Tx Timestamping Unit, GPIO port of top design, I^2C master interfaces, Per-port Statistics)
4 - 31			not used

Wishbone interface: section 4.7.

3.1.7 Timer block

Description:
Used by *LM32* for counting time intervals independent from the local timebase (being adjusted).

Wishbone interface:
Only *LM32* talks to this interface, WR Switch software does not access it.

3.1.8 1-PPS generator

Description:
The module is responsible for generating and inputting 1-PPS signal. It contains two timekeeping counters: *cntr_utc* and *cntr_nsec*. The former keeps full seconds of the White Rabbit time, while the latter represents fractional part of each second. *cntr_nsec* is clocked

with 62.5 MHz reference clock which means that its granularity is 16 ns. The 1-PPS pulse is generated at the beginning of each second i.e. *cntr_nsec* equals to 0.

Wishbone registers of 1-PPS generator allow modifications of *cntr_utc* and *cntr_nsec* counters. It can be performed in two ways: setting or adjusting the time. The former simply stores new values to the counters when requested. The adjustment is done by adding new values to the current state of the counters (*cntr_utc*, *cntr_nsec*) at the beginning of a new second (when *cntr_nsec* is 0).

Wishbone interface: section 4.1.

3.2 Network Interface Controller

Description:

Module responsible for passing Ethernet frames between the Linux running on the main ARM processor and 18 ports of WR Switch. It contains a *frame buffer* and two RAM blocks (*TX descriptors memory*, *RX descriptors memory*) storing descriptors for frames received and frames to be sent. Frame buffer stores all frames received from physical ports of WR Switch (addressed to the main processor) and frames that main processor wants to send to the ports of WR Switch. Each frame has an associated descriptor which contains various information about its structure (figure 3 and figure 4).

When software running on main CPU wants to send an Ethernet frame, it has to write it to the *frame buffer* and then store the Tx descriptor describing this frame into *TX descriptors memory*. The other way round, first software has to write an empty Rx descriptor (*empty* bit set to *1*) into the *RX descriptors memory*. It has to describe the offset and length of the area in the *frame buffer* where *NIC* can store received frame. When a new frame is received *NIC* will fill the Rx descriptor and set *empty* bit to *0*.

When configured, *Network Interface Controller* can also trigger three different interrupts when:

- a new Ethernet frame was received
- transmission of the frame is completed
- an error has occurred during the frame's transmission

They are all combined into a single IRQ line connected to *Vectored Interrupt Controller* (sec.3.4) so wishbone control register has to be read to get the event which caused the interrupt.

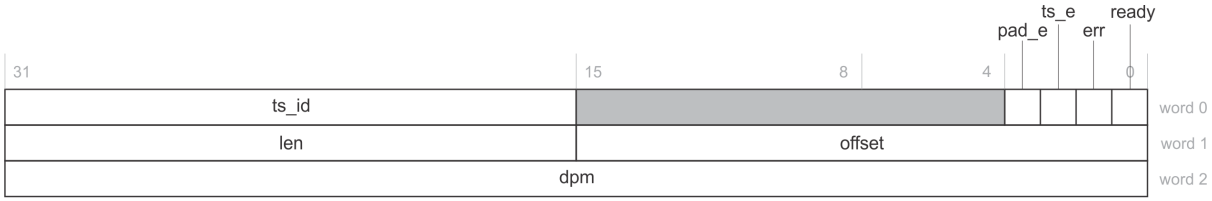


Figure 3: Tx descriptor

ready	if *1*, whole descriptor is stored in memory, frame can be transmitted
err	if *1*, an error has occurred during frame's transmission
ts_e	if *1*, request frame timestamping
pad_e	if *1*, frame is runt and requires padding
ts_id	frame id, needed to associate Tx timestamp (from Tx Timestamping Unit) with appropriate frame
offset	offset of the frame inside *frame buffer*
len	length of the frame
dpm	destination port mask, each bit set to *1* will result in Ethernet frame sent from that physical port

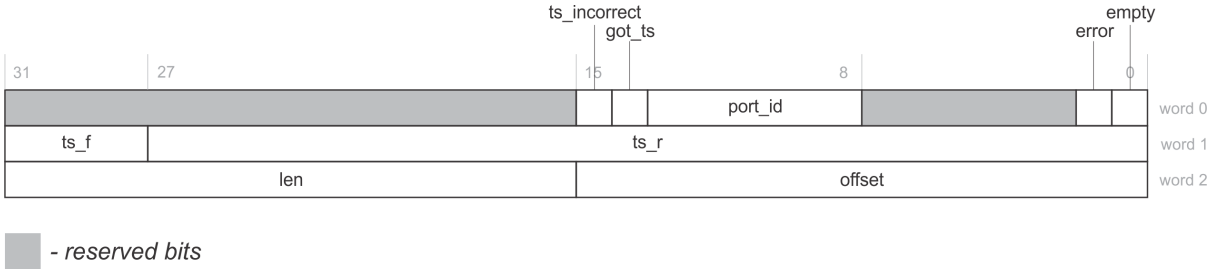


Figure 4: Rx descriptor

empty	if *1*, descriptor is empty, and can be written by reception FSM
error	if *1*, an error has occurred during the reception of the frame
port_id	the ID of the physical port which has received the frame
got_ts	if *1*, received frame contains OOB data with Rx timestamp
ts_incorrect	if *1*, Rx timestamp may be incorrect (generated during the adjustment of time base)
ts_r	Rx timestamp generated on the rising edge of the reference clock
ts_f	least significant bits of the Rx timestamp generated on the falling edge of the reference clock
offset	offset of the frame inside *frame buffer*
len	length of the allocated buffer for Rx frame (when *empty* is *1*) or length of the received frame (when *empty* is *0*) received frame when

Wishbone interface:

Wishbone interface of WR NIC contains two areas with different base addresses:

configuration registers	0x20000
frame buffer	0x28000

The description of configuration registers can be found in section 4.2.

3.3 Endpoint fanout

The Endpoint fanout contains configurable amount of Endpoint modules connected together with the Wishbone Crossbar.

Base Wishbone addresses of Endpoints:

module name	base address
Endpoint 0	0x30000
Endpoint 1	0x30400
Endpoint 2	0x30800
Endpoint 3	0x30c00
Endpoint 4	0x31000
...	...
Endpoint n	0x30000 + n*0x400

Description:

Endpoint module implements Gigabit Ethernet MAC functionality and PCS for Gigabit optical link. It sends and receives Ethernet frames from a physical link and is able to generate precise Tx and Rx timestamps. It also has VLAN and Flow Control (receiving Pause frames) support.

Additionally it is able to inject frames required by the Topology Resolution Unit (sec. 3.9) for hardware supported RSTP and generates events counted by Per-port Statistics module (sec. 3.11).

Endpoint contains also a programmable packet filter, which can classify incoming Ethernet frames into 8 different classes.

Wishbone interface: section 4.3.

3.3.1 Programmable packet filter

The description of the packet filter inside the Endpoint module is taken from *dev/ep_pfilter.c* file stored in White Rabbit PTP Core Software repository.

The classifier processes the incoming frame, and assigns it to one of 8 classes (an 8-bit word, where each bit corresponds to a particular class) or eventually drops it. Hardware implementation of the unit is a simple VLIW processor with 32 single-bit registers (0 - 31). The registers are organized as follows:

- 0: don't touch (always 0)

- 1 - 22: general purpose registers
- 23: drop frame flag: if 1 at the end of the frame processing, the frame will be dropped.
- 24..31: packet class (class 0 = reg 24, class 7 = reg 31).

Program memory has 64 36-bit words. Packet filtering program is restarted every time a new frame comes. There are 5 possible instructions:

1. *CMP offset, value, mask, oper, Rd*:
   Rd = Rd oper ((((uint16_t *)frame) [offset] & mask) == value)

   Examples:

   - *CMP 3, 0xcafe, 0xffff, MOV, Rd*
     will compare the 3rd word of the frame (bytes 6, 7) against 0xcafe and if the words are equal, 1 will be written to Rd register.
   - *CMP 4, 0xbabe, 0xffff, AND, Rd*
     will do the same with the 4th word and write to Rd its previous value ANDed with the result of the comparison. Effectively, Rd now will be 1 only if bytes [6..9] of the payload contain word 0xcafebabe.

     Note that the mask value is nibble-granular. That means you can choose a particular set of nibbles within a word to be compared, but not an arbitrary set of bits (e.g. 0xf00f, 0xff00 and 0xf0f0 masks are ok, but 0x8001 is wrong.

2. *BTST offset, bit_number, oper, Rd*:
   Rd = Rd oper (((uint16_t *)frame) [offset] & (1≪bit_number) ? 1 : 0)

   Examples:

   - *BTST 3, 10, MOV, 11*
     will write 1 to reg 11 if the 10th bit in the 3rd word of the frame is set (and 0 if it's clear)

3. Logic opearations:

   - *LOGIC2 Rd, Ra, OPER Rb* - 2 argument logic (Rd = Ra OPER Rb). If the operation is MOV or NOT, Ra is taken as the source register.
   - *LOGIC3 Rd, Ra, OPER Rb, OPER2, Rc* - 3 argument logic Rd = (Ra OPER Rb) OPER2 Rc.

4. Miscellaneous:

   - *FIN* instruction terminates the program.
   - *NOP* executes a dummy instruction (LOGIC2 0, 0, AND, 0)

IMPORTANT:

- the program counter is advanced each time a 16-bit words of the frame arrives.

- the CPU doesn't have any interlocks to simplify the HW, so you can't compare the 10th word when PC = 2. Max comparison offset is always equal to the address of the instruction.
- Code may contain up to 64 operations, but it must classify shorter frames faster than in 32 instructions (there's no flow throttling)

3.3.2 Injection engine

The frame injection engine has to be programmed first with frames' templates and after that sending (injecting) each template can be triggered from the TRU module.

Storing frame templates is done through the *VCR1* Wishbone register. Each word written to this register is a concatenation of the offset value and the data word (check section 4.3). Since the same block RAM is shared by the Injection engine and VLAN unit, templates have to be stored starting with the offset **512**. This buffer can store up to 8 templates each not longer than 128 bytes.

Storing a frame's template requires splitting such frame into 2-byte words and performing a set of writes into the *VCR1* register (each time the offset value has to be incremented). The format of data word is presented in figure 5:

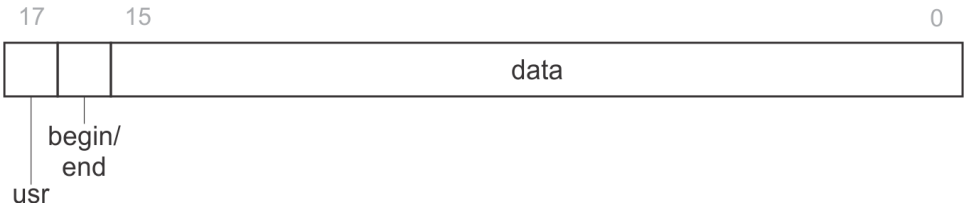


Figure 5: Format of data word for programming the injection engine

data	2-byte chunk of the template
begin/end	if set to 1, the written word is first or last word of template's data
usr	if set to 1, this data word will be replaced by the value provided by the TRU module

3.4 Vectored Interrupt Controller (VIC)

Description:

The module combines multiple interrupt lines from different modules inside the WR Switch Gateware and outputs a single IRQ fed to the main CPU. It inputs up to 32 prioritized internal interrupts. The priorities of the interrupts are fixed: IRQ0 has the highest priority while IRQ31 has the lowest priority. Currently 4 internal interrupts are connected to *VIC*:

IRQ No.	source module
0	Network Interface Controller (*NIC*)
1	Tx Timestamping Unit (*TS TSU*)
2	Routing Table Unit (*RTU*)
3	Per-port Statistics (*Pstats*)
4 - 31	currently not used

Wishbone interface: section 4.5.

3.5 Tx Timestamping Unit (Tx TSU)

Description:

The module collects Tx timestamps for Ethernet frames sent through all 18 Endpoints available in WR Switch. Those timestamps are stored inside the FIFO queue and can be fetched by the software running on the main CPU together with the Frame ID (FID) and Port ID (PID). This additional information is used by the CPU to associate the timestamp with appropriate frame that was sent through *NIC* and one (or multiple) of the Endpoints.

The module can also generate an interrupt (fed to VIC described in section 3.4). It indicates that TxTSU stores at least one timestamp in its FIFO.

Wishbone interface: section 4.6.

3.6 GPIO port

Description:

The module controls GPIO lines through Wishbone interface. The direction (input/output) is set separately for each line.

Currently only one GPIO line driven from this module is used in WR Switch. It selects whether the debug UART (from Real-Time Subsystem) or UART from the main CPU will be available on a physical UART connector of the WR Switch.

GPIO No.	description
0 - 30	not used
31	uart select

Wishbone interface: section 4.7.

3.7 I^2C master

There are three identical I^2C masters:

- *I2C0*: drives I^2C GPIO expansion chip for ports 16-17 [1]
- *I2C1*: drives I^2C GPIO expansion chip for ports 0-15
- *Sensors I2C*: connected to the set of *TMP100* digital thermometers

Description:

I^2C master as its name says, implements the master of I^2C bus. Before any byte transfer is performed it must have the prescale value, used to generate SCL clock, set.

The module is capable of generating interrupt, but it is not used in the WR Switch design.

Wishbone interface: section 4.8.

[1] Check the schematics of WR Switch miniBackplane for details what is driven by those GPIOs.

3.8 PWM Controller

Description:

Pulse Width Modulation controller is used by the WR Switch software to adjust the speed of two fans installed at the back of a WR Switch box. The module can drive up to 8 PWM channels, but only two are currently used:

Channel No.	description
0	controlling the speed of the left fan
1	controlling the speed of the right fan

Wishbone interface: section 4.9.

3.9 Topology Resolution Unit (TRU)

Description:

Module provides hardware support for topology resolution protocols implemented in software (e.g. RSTP, LACP). It is still under testing and development so it shouldn't be supported by the next official release of the WR Switch software.

Wishbone interface: section 4.10.

3.10 Time-Aware Traffic Shaper Unit (TATSU)

Description:

The module implements simple Time-Aware Traffic Shaper. It creates a time window at a given (configured) moment (TAI + cycles) in a data stream transmitted through WR Switch. In this window only selected output queues are allowed (the others are blocked).

The module is still under testing and development so it shouldn't be supported by the next official release of WR Switch Software.

Wishbone interface: section 4.11.

3.11 Per-port Statistics (Pstats)

Description:

The module provides per-port statistics of the events presented in the table:

Event no.	name	source	description
0	tx_underrun	*Endpoint*	TX FIFO in PCS underrun
1	rx_overrun	*Endpoint*	RX FIFO in PCS overrun
2	rx_invalid_code	*Endpoint*	received invalid 8B10B code
3	rx_sync_lost	*Endpoint*	link synchronization lost
4	rx_pause	*Endpoint*	received pause frame
5	rx_pfilter_drop	*Endpoint*	received frame dropped by Packet Filter
6	rx_pcs_err	*Endpoint*	some error has occurred during frame reception (e.g. invalid code, fifo overrun, wrong frame termination, etc.)
7	rx_giant	*Endpoint*	received *giant* frame (bigger than Maximum Receive Unit
8	rx_runt	*Endpoint*	received *runt* frame (smaller than 64 bytes)
9	rx_crc_err	*Endpoint*	CRC of the received frame does not match
10	rx_pclass(0)	*Endpoint*	RX frame got assigned to class 0 by Packet Filter
...	...	...	...
17	rx_pclass(7)	*Endpoint*	RX frame got assigned to class 7 by Packet Filter
18	tx_frame	*Endpoint*	frame was transmitted
19	rx_frame	*Endpoint*	frame was received
20	req_valid	*RTU*	got valid RTU request
21	rsp_valid	*RTU*	outputting valid RTU response
22	rsp_drop	*RTU*	frame dropped
23	fm_hp_frame	*RTU*	got high priority frame
24	fm_fast_fwd	*RTU*	fast forward frame (depending on RTU configuration, this can be broadcast frames, PTP frames, etc.)
25	fm_non_fwd	*RTU*	don't forward frame (recognized link-limited frame, e.g. BPDU)
26	*reserved*	*RTU*	
27	*reserved*	*RTU*	

Each event on each port has a separate 16-bit counter associated. It stores the number of occurrence of the event. To be able to handle the bust of events from a Gigabit Ethernet link, each counter is divided into two 8-bit halves creating two layers of counting (L1, L2). Counters are aligned in those layers such a way that each 32-bit word in memory stores the values of 4 counters:

	31 24	23 16	15 8	7 0
L1	CNTR3[7:0]	CNTR2[7:0]	CNTR1[7:0]	CNTR0[7:0]
L2	CNTR3[15:8]	CNTR2[15:8]	CNTR1[15:8]	CNTR0[15:8]

Therefore, one read operation from *Pstats* module returns two 32-bit words (L1, L2) from which user can extract the state of four subsequent counters.

The module generates an interrupt when any of the 16-bit counters on any of the ports has overflown. After that *EIC_ISR* register indicates per-port IRQ, and based on its content user can read per-port IRQ state using *CR* Wishbone register.

Wishbone interface: section 4.12.

3.12 Routing Table Unit (RTU)

Description:

RTU is responsible for making the decisions where (to which port or ports of the Switch) each frame received from any of the ports has to be forwarded. The decision is made based on the information from:

- VLAN table: storing information about the VLAN configuration. Each entry contains a mask describing which physical ports belong to the VLAN (each bit set to *1* means the port belongs to VLAN), index of VLAN group (FID) to which the VLAN belongs to, VLAN-assigned priority and also two additional bits, one saying that all frames belonging to the VLAN should be dropped and the other saying that every frame belonging to the VLAN should have its priority replaced with VLAN priority
- Hash table (HTAB): stores dynamic (the result of learning process) and static entries describing the presence of MAC addresses on physical ports of the Switch (i.e. MAC addresses of the devices connected to each physical port of the Switch)
- Topology Resolution Unit: responsible for dynamic network topology reconfiguration, more detailed description in section 3.9.

The Hash Table contains a set of MAC entries of the structure presented in the table:

word no.	offset	length	name	description
0	0	1	valid	if 1, the entry contains valid data
	1	1	*reserved*	
	2	1	is_bpdu	if 1, accept frame even if the port is disabled in RTU configuration
	3	1	go_to_cam	obsolete, will be removed or replaced
	4	8	fid	Filtering Database ID which was used to generate the address (hash) of the entry
	12	4	*reserved*	
	16	16	mac[47:32]	MAC address (high part)
1	0	32	mac[31:0]	MAC address (low part)
2	0	16	cam_addr	obsolete, will be removed or replaced
	16	1	has_prio_src	validates prio_src and prio_override_src fields
	17	3	prio_src	priority of the frame having source MAC matching with MAC of this entry
	20	1	prio_ override_src	override frame's priority with *prio_src* when its SMAC matches MAC of this entry
	21	1	drop_when_src	drop frame when its SMAC matches MAC of this entry
	22	1	drop_when_ unmatched_ src_ports	drop the frame when it comes from source port which does not belong to port_mask_src
	23	1	has_prio_dst	validates prio_dst and prio_override_dst
	24	3	prio_dst	priority of the frame having DMAC matching with MAC of this entry
	27	1	prio_ override_dst	override frame's priority with *prio_dst* when its DMAC matches MAC of this entry
	28	1	drop_when_dst	drop frame if its DMAC matches MAC of this entry
3	0	16	port_mask_ src[15:0]	low part of the port mask for SMAC, bit set to 1 indicates that frame with SMAC matching MAC of this entry can be forwarded from that port/ports. Ports having their bits set to 0 shall drop the frame
	16	16	port_mask_ dst[15:0]	low part of the port mask for DMAC, bit set to 1 indicates to which physical ports the frame with DMAC matching MAC of this entry shall be forwarded
4	0	16	port_mask_ src[31:16]	high part of the port mask for SMAC
	16	16	port_mask_ dst[31:16]	high part of the port mask for DMAC

HTAB is organized in buckets. Each bucket stores five words of MAC entry described in the table above. The bucket is addressed with the hash value calculated from the FID and MAC address of the MAC entry it stores. This hash is a CRC generated using one of the supported

polynomials (*POLY_VAL* of GCR Wishbone register). Since different MACs and FIDs can result in the same hash value, each of the hash is associated with four buckets. If it turns out that first bucket already stores a valid MAC entry, next one is checked. A new MAC entry is then written to first free (not already used) bucket.

HTAB is written by the switch software and can contain both static entries required by the configuration of the Switch and dynamic entries resulting from learning process implemented in software. Write-only access to HTAB is provided through MFIFO Wishbone registers (*MFIFO_R0*, *MFIFO_R1* and *MFIFO_CSR*). The address of this memory has the structure presented in figure 6.

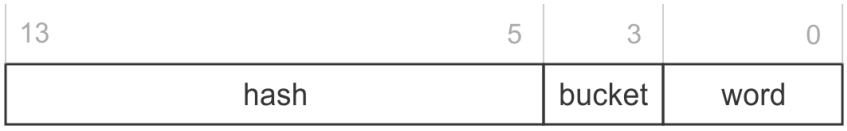


Figure 6: Structure of HTAB address

To write a new MAC entry to HTAB first you have to write an address where the first word of MAC entry will be written (least significant three bits - *word* - of the HTAB address equal 0) and then continue writing word by word a complete entry:

register	**value**
`MFIFO_R0`	*0x01*
`MFIFO_R1`	<HTAB address>
`MFIFO_R0`	*0x00*
`MFIFO_R1`	<word 0>
`MFIFO_R1`	<word 1>
`MFIFO_R1`	<word 2>
`MFIFO_R1`	<word 3>
`MFIFO_R1`	<word 4>

The dynamic entries in HTAB should be created by learning mechanism based on the information got from HDL through UFIFO (when learning on a port is enabled). RTU module is capable of generating an interrupt when UFIFO is not empty i.e. it stores at least one learning request. The UFIFO provides basic information about the unrecognized frame: source MAC, destination MAC, optionally VLAN Id and priority.

The RTU contains also the memory called ARAM which stores an aging bitmap. Each bit corresponds to one MAC entry in HTAB and, when set to *1*, indicates that the MAC entry has been matched with the source MAC of the forwarded frame. The memory is addressed in a similar way as HTAB, but since here each bit corresponds to one MAC entry and each word is 32-bits long, the word in ARAM is addressed with address from figure 6 shifted 8 bits to the right (i.e. hash[8..3]). Three least significant bits of hash value concatenated with the bucket number are used to find an appropriate bit in the word read from the ARAM.

Besides the basic functionality described above, RTU has two main extensions:

- **port mirroring**: allows the ingress or egress traffic from selected port to be retransmitted on one or multiple other ports configured with appropriate mask

- **Fast Match mechanism**: provides quick (23 clock cycles in worst case) routing response for selected types of frames

Fast Match can generate routing responses for (if appropriate bits are set in *RX_CTR* control register):

- broadcast frames: DMAC is FF:FF:FF:FF:FF:FF
- PTP frames: DMAC is 01:1B:19:00:00:00
- Link-Limited frames: DMAC in range from 01:80:C2:00:00:00 to 01:80:C2:00:00:0F
- High Priority frames: which have priority included in the *PRIO_MASK* of *RX_CTR* register
- Any frame that has its DMAC in configured range of MAC addresses or DMAC equal to single configured MAC

Wishbone interface: section 4.13

4 Wishbone configuration interfaces

4.1 WR Switch PPS generator and RTC

Unit generating PPS signals and acting as a UTC real-time clock

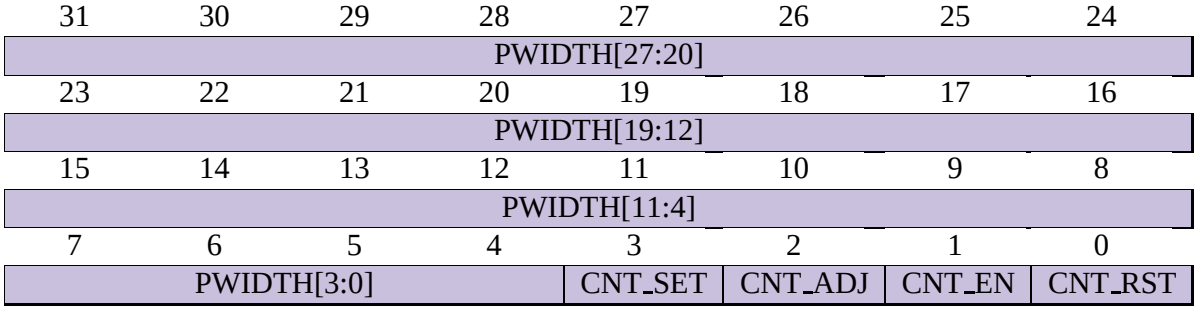
4.1.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	Control Register	ppsg_cr	CR
0x1	REG	Nanosecond counter register	ppsg_cntr_nsec	CNTR_NSEC
0x2	REG	UTC Counter register (least-significant part)	ppsg_cntr_utclo	CNTR_UTCLO
0x3	REG	UTC Counter register (most-significant part)	ppsg_cntr_utchi	CNTR_UTCHI
0x4	REG	Nanosecond adjustment register	ppsg_adj_nsec	ADJ_NSEC
0x5	REG	UTC Adjustment register (least-significant part)	ppsg_adj_utclo	ADJ_UTCLO
0x6	REG	UTC Adjustment register (most-significant part)	ppsg_adj_utchi	ADJ_UTCHI
0x7	REG	External sync control register	ppsg_escr	ESCR

4.1.2 Register description

Control Register

HW prefix: ppsg_cr
HW address: 0x0
SW prefix: CR
SW offset: 0x0



- **CNT_RST** [*write-only*]: Reset counter
  write 1: resets the counter
  write 0: no effect

- **CNT_EN** [*read/write*]: Enable counter
  1: PPS counter is enabled

- **CNT_ADJ** [*read/write*]: Adjust offset
  write 1: Starts adjusting PPS/UTC offsets by adding the values taken from ADJ_NSEC, ADJ_UTCLO, ADJ_UTCHI registers to the current PPS counter value. These registers need to be programmed prior to update.
  write 0: no effect
  read 0: adjustment operation is done
  read 1: adjustment operation is in progress

- **CNT_SET** [*write-only*]: Set time
  write 1: Sets the UTC/PPS counter to values taken from ADJ_NSEC, ADJ_UTCLO, ADJ_UTCHI registers

- **PWIDTH** [*read/write*]: PPS Pulse width
  Width of generated PPS pulses in 62.5 MHz refernce clock cycles

Nanosecond counter register

HW prefix: ppsg_cntr_nsec
HW address: 0x1
SW prefix: CNTR_NSEC
SW offset: 0x4

Nanosecond part of current time, expressed as number of 62.5 MHz reference clock cycles

31	30	29	28	27	26	25	24
-	-	-	-	CNTR_NSEC[27:24]			
23	22	21	20	19	18	17	16
CNTR_NSEC[23:16]							
15	14	13	12	11	10	9	8
CNTR_NSEC[15:8]							
7	6	5	4	3	2	1	0
CNTR_NSEC[7:0]							

- **CNTR_NSEC** [*read-only*]: Nanosecond counter

UTC Counter register (least-significant part)

HW prefix: ppsg_cntr_utclo
HW address: 0x2
SW prefix: CNTR_UTCLO
SW offset: 0x8

Lower 32 bits of current UTC time

31	30	29	28	27	26	25	24
CNTR_UTCLO[31:24]							
23	22	21	20	19	18	17	16
CNTR_UTCLO[23:16]							
15	14	13	12	11	10	9	8
CNTR_UTCLO[15:8]							
7	6	5	4	3	2	1	0
CNTR_UTCLO[7:0]							

- **CNTR_UTCLO** [*read-only*]: UTC Counter

UTC Counter register (most-significant part)

HW prefix: ppsg_cntr_utchi
HW address: 0x3
SW prefix: CNTR_UTCHI
SW offset: 0xc

Highest 8 bits of current UTC time

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
CNTR_UTCHI[7:0]							

- **CNTR_UTCHI** [*read-only*]: UTC Counter

Nanosecond adjustment register

HW prefix: ppsg_adj_nsec
HW address: 0x4
SW prefix: ADJ_NSEC
SW offset: 0x10

Adjustment value for nanosecond counter

31	30	29	28	27	26	25	24
-	-	-	-	ADJ_NSEC[27:24]			
23	22	21	20	19	18	17	16
ADJ_NSEC[23:16]							
15	14	13	12	11	10	9	8
ADJ_NSEC[15:8]							
7	6	5	4	3	2	1	0
ADJ_NSEC[7:0]							

- **ADJ_NSEC** [*write-only*]: Nanosecond adjustment

UTC Adjustment register (least-significant part)

HW prefix: ppsg_adj_utclo
HW address: 0x5
SW prefix: ADJ_UTCLO
SW offset: 0x14

Lower 32 bits of adjustment value for UTC

31	30	29	28	27	26	25	24
ADJ_UTCLO[31:24]							
23	22	21	20	19	18	17	16
ADJ_UTCLO[23:16]							
15	14	13	12	11	10	9	8
ADJ_UTCLO[15:8]							
7	6	5	4	3	2	1	0
ADJ_UTCLO[7:0]							

- **ADJ_UTCLO** [*write-only*]: UTC Counter adjustment

UTC Adjustment register (most-significant part)

HW prefix: ppsg_adj_utchi
HW address: 0x6
SW prefix: ADJ_UTCHI
SW offset: 0x18

Highest 8 bits of adjustment value for UTC

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
ADJ_UTCHI[7:0]							

- **ADJ_UTCHI** [*write-only*]: UTC Counter adjustment

External sync control register

HW prefix: ppsg_escr
HW address: 0x7
SW prefix: ESCR
SW offset: 0x1c

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
-	-	-	-	-	TM_VALID	PPS_VALID	SYNC

- **SYNC** [*read/write*]: Sync to external PPS input
  write 1: Waits until a pulse on external PPS input arrives and re-synchronizes the PPS counter to it
  write 0: no effect
  read 1: external synchronization done
  read 0: external synchronization in progress

- **PPS_VALID** [*read/write*]: PPS output valid
  write 1: PPS output provides reliable 1-PPS signal
  write 0: PPS output is invalid

- **TM_VALID** [*read/write*]: Timecode output(UTC+cycles) valid
  write 1: Timecode output provides valid time
  write 0: Timecode output does not provide valid time

4.2 White Rabbit Switch NIC's spec

This NIC is in between the endpoints and the on-board Linux CPU of the White Rabbit Switch.

4.2.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	NIC Control Register	nic_cr	CR
0x1	REG	NIC Status Register	nic_sr	SR
0x8	REG	Interrupt disable register	nic_eic_idr	EIC_IDR
0x9	REG	Interrupt enable register	nic_eic_ier	EIC_IER
0xa	REG	Interrupt mask register	nic_eic_imr	EIC_IMR
0xb	REG	Interrupt status register	nic_eic_isr	EIC_ISR
0x20 - 0x3f	MEM	TX descriptors mem	nic_dtx	DTX
0x40 - 0x5f	MEM	RX descriptors mem	nic_drx	DRX

4.2.2 Register description

NIC Control Register

HW prefix: nic_cr
HW address: 0x0
SW prefix: CR
SW offset: 0x0

31	30	29	28	27	26	25	24
SW_RST	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	TX_EN	RX_EN

- **RX_EN** [*read/write*]: Receive enable
  write 1: enables receiving path
  write 0: disables receiving path
  read 1: receiving path enabled
  read 0: receiving path disabled

- **TX_EN** [*read/write*]: Transmit enable
  Enables the NIC to transmit data. When reset, the internal transmit pointer points to the first entry in the TX descriptor pool
  write 1: enables transmitting path
  write 0: disables transmitting path
  read 1: transmitting path enabled
  read 0: transmitting path disabled

- **SW_RST** [*write-only*]: Software Reset
  write 1: reset the NIC, zero all registers and reset the state of the module
  write 0: no effect

NIC Status Register

HW prefix: nic_sr
HW address: 0x1
SW prefix: SR
SW offset: 0x4

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	CUR_RX_DESC[2:0]		
15	14	13	12	11	10	9	8
-	-	-	-	-	CUR_TX_DESC[2:0]		
7	6	5	4	3	2	1	0
-	-	-	-	TX_ERROR	TX_DONE	REC	BNA

- **BNA** [*read-only*]: Buffer Not Available
  read 1: no buffers were available when receiving a packet.

- **REC** [*read/write*]: Frame Received
  read 1: one or more frames have been received.
  read 0: there is no new frame received
  write 1: clear the flag
  write 0: no effect

- **TX_DONE** [*read/write*]: Transmission done
  read 1: All non-empty TX descriptors have been transmitted
  read 0: Transmission in progress
  write 1: Clears the flag
  write 0: No effect

- **TX_ERROR** [*read/write*]: Transmission error
  read 1: A TX error occured and the transmission was stopped. CUR_TX_DESC is pointing the TX descriptor for which the error occured
  read 0: No TX error
  write 1: Clears the flag
  write 0: No effect

- **CUR_TX_DESC** [*read-only*]: Current TX descriptor
  Index of the currently handled TX descriptor

- **CUR_RX_DESC** [*read-only*]: Current RX descriptor
  Index of the currently handled RX descriptor

Interrupt disable register

HW prefix: nic_eic_idr
HW address: 0x8
SW prefix: EIC_IDR
SW offset: 0x20

Writing 1 disables handling of the interrupt associated with corresponding bit. Writin 0 has no effect.

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
-	-	-	-	-	TXERR	TCOMP	RCOMP

- **RCOMP** [*write-only*]: Receive Complete
  write 1: disable interrupt 'Receive Complete'
  write 0: no effect
- **TCOMP** [*write-only*]: Transmit Complete
  write 1: disable interrupt 'Transmit Complete'
  write 0: no effect
- **TXERR** [*write-only*]: Transmit Error
  write 1: disable interrupt 'Transmit Error'
  write 0: no effect

Interrupt enable register

HW prefix: nic_eic_ier
HW address: 0x9
SW prefix: EIC_IER
SW offset: 0x24

Writing 1 enables handling of the interrupt associated with corresponding bit. Writin 0 has no effect.

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
-	-	-	-	-	TXERR	TCOMP	RCOMP

- **RCOMP** [*write-only*]: Receive Complete
  write 1: enable interrupt 'Receive Complete'
  write 0: no effect

- **TCOMP** [*write-only*]: Transmit Complete
  write 1: enable interrupt 'Transmit Complete'
  write 0: no effect

- **TXERR** [*write-only*]: Transmit Error
  write 1: enable interrupt 'Transmit Error'
  write 0: no effect

Interrupt mask register

HW prefix: nic_eic_imr
HW address: 0xa
SW prefix: EIC_IMR
SW offset: 0x28

Shows which interrupts are enabled. 1 means that the interrupt associated with the bitfield is enabled

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
-	-	-	-	-	TXERR	TCOMP	RCOMP

- **RCOMP** [*read-only*]: Receive Complete
  read 1: interrupt 'Receive Complete' is enabled
  read 0: interrupt 'Receive Complete' is disabled

- **TCOMP** [*read-only*]: Transmit Complete
  read 1: interrupt 'Transmit Complete' is enabled
  read 0: interrupt 'Transmit Complete' is disabled

- **TXERR** [*read-only*]: Transmit Error
  read 1: interrupt 'Transmit Error' is enabled
  read 0: interrupt 'Transmit Error' is disabled

Interrupt status register

HW prefix: nic_eic_isr
HW address: 0xb
SW prefix: EIC_ISR
SW offset: 0x2c

Each bit represents the state of corresponding interrupt. 1 means the interrupt is pending. Writing 1 to a bit clears the corresponding interrupt. Writing 0 has no effect.

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
-	-	-	-	-	TXERR	TCOMP	RCOMP

- **RCOMP** [*read/write*]: Receive Complete
  read 1: interrupt 'Receive Complete' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Receive Complete'
  write 0: no effect

- **TCOMP** [*read/write*]: Transmit Complete
  read 1: interrupt 'Transmit Complete' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Transmit Complete'
  write 0: no effect

- **TXERR** [*read/write*]: Transmit Error
  read 1: interrupt 'Transmit Error' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Transmit Error'
  write 0: no effect

TX descriptors mem

HW prefix:	nic_dtx
HW address:	0x20
C prefix:	DTX
C offset:	0x80
Size:	32 32-bit words
Data width:	32
Access (bus):	read/write
Access (device):	read/write
Mirrored:	no
Byte-addressable:	no
Peripheral port:	bus-synchronous

RX descriptors mem

HW prefix: nic_drx
HW address: 0x40
C prefix: DRX
C offset: 0x100
Size: 32 32-bit words
Data width: 32
Access (bus): read/write
Access (device): read/write
Mirrored: no
Byte-addressable: no
Peripheral port: bus-synchronous

4.2.3 Interrupts

Receive Complete

HW prefix: nic_rcomp
C prefix: RCOMP
Trigger: high level

A frame has been stored in memory.

Transmit Complete

HW prefix: nic_tcomp
C prefix: TCOMP
Trigger: high level

Frame successfully transmitted

Transmit Error

HW prefix: nic_txerr
C prefix: TXERR
Trigger: high level

4.3 WR Switch Endpoint

Implementation of MAC and PCS layer capable of generating precise Tx and Rx timestamps

4.3.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	Endpoint Control Register	ep_ecr	ECR
0x1	REG	Timestamping Control Register	ep_tscr	TSCR
0x2	REG	RX Deframer Control Register	ep_rfcr	RFCR
0x3	REG	VLAN control register 0	ep_vcr0	VCR0
0x4	REG	VLAN Control Register 1	ep_vcr1	VCR1
0x5	REG	Packet Filter Control Register 0	ep_pfcr0	PFCR0
0x6	REG	Packet Filter Control Register 1	ep_pfcr1	PFCR1
0x7	REG	Traffic Class Assignment Register	ep_tcar	TCAR
0x8	REG	Flow Control Register	ep_fcr	FCR
0x9	REG	Endpoint MAC address high part register	ep_mach	MACH
0xa	REG	Endpoint MAC address low part register	ep_macl	MACL
0xb	REG	MDIO Control Register	ep_mdio_cr	MDIO_CR
0xc	REG	MDIO Address/Status Register	ep_mdio_asr	MDIO_ASR
0xd	REG	Identification register	ep_idcode	IDCODE
0xe	REG	Debug/Status register	ep_dsr	DSR
0xf	REG	DMTD Control Register	ep_dmcr	DMCR
0x10	REG	DMTD Status register	ep_dmsr	DMSR

4.3.2 Register description

Endpoint Control Register

HW prefix: ep_ecr
HW address: 0x0
SW prefix: ECR
SW offset: 0x0

General endpoint control register

31	30	29	28	27	26	25	24
-	-	-	-	FEAT_DPI	FEAT_PTP	FEAT_DMTD	FEAT_VLAN
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
RX_EN	TX_EN	RST_CNT	PORTID[4:0]				

- **PORTID** [*read/write*]: Port identifier
  Unique port identifier which will be embedded into OOB with the timestamp value

- **RST_CNT** [*write-only*]: Reset event counters
  write 1: resets all event counters
  write 0: no effect

- **TX_EN** [*read/write*]: Transmit path enable
  read/write 1: TX path is enabled
  read/write 0: TX path is disabled

- **RX_EN** [*read/write*]: Receive path enable
  read/write 1: RX path is enabled
  read/write 0: RX path is disabled

- **FEAT_VLAN** [*read-only*]: Feature present: VLAN tagging
  read 1: this implementation of WR Endpoint supports VLAN processing (tagging/untagging). VCR register can be used to control the VLAN functionality
  read 0: no VLAN support

- **FEAT_DMTD** [*read-only*]: Feature present: DDMTD phase measurement
  read 1: this implementation of WR Endpoint can do fine phase measurements using a DDMTD phase detector
  read 0: no phase measurement support

- **FEAT_PTP** [*read-only*]: Feature present: IEEE1588 timestamper
  read 1: this implementation of WR Endpoint can timestamp packets
  read 0: no timestamping support

- **FEAT_DPI** [*read-only*]: Feature present: DPI packet classifier
  read 1: this implementation of WR Endpoint includes Deep Packet Inspection packet classifier/filter
  read 0: no DPI support

Timestamping Control Register

HW prefix: ep_tscr
HW address: 0x1
SW prefix: TSCR
SW offset: 0x4

Register controlling timestamping features of the endpoint

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
-	-	-	-	CS_DONE	CS_START	EN_RXTS	EN_TXTS

- **EN_TXTS** [*read/write*]: Enable timestamping of transmitted frames
  write 1: enable TX timestamping. Endpoints passes timestamps to shared TX timestamping unit
  write 0: disable TX timestamping
  read 1: TX timestamping enabled
  read 0: TX timestamping disabled

- **EN_RXTS** [*read/write*]: Enable timestamping of received frames
  write 1: enable RX timestamping. RX timestamps are embedded into OOB field on the fabric interface. Must be enabled if used in a multi-port configuration (e.g. in a switch)
  write 0: disable RX timestamping
  read 1: RX timestamping enabled
  read 0: RX timestamping disabled

- **CS_START** [*write-only*]: Timestamping counter synchronization start
  write 1: start synchronizing the local PPS counter used for timestamping TX/RX packets with an external pulse provided on pps_i input. After synchronization, the CS_DONE flag will be set to 1. The counter value equals to 0 when PPS line is high.
  write 0: no effect

- **CS_DONE** [*read-only*]: Timestamping counter synchronization done
  read 1: the counter synchronization procedure is done.
  read 0: the counter synchronization procedure is pending

RX Deframer Control Register

HW prefix: ep_rfcr
HW address: 0x2
SW prefix: RFCR
SW offset: 0x8

31	30	29	28	27	26	25	24
-	-	-	-	-	-	MRU[13:12]	
23	22	21	20	19	18	17	16
MRU[11:4]							
15	14	13	12	11	10	9	8
MRU[3:0]				HPAP[7:4]			
7	6	5	4	3	2	1	0
HPAP[3:0]				KEEP_CRC	A_HP	A_GIANT	A_RUNT

- **A_RUNT** [*read/write*]: Accept RX runt frames
  read/write 1: endpoint accepts 'runt' frames (shorter than 64 bytes)
  read/write 0: 'runt' frames are dropped

- **A_GIANT** [*read/write*]: Accept RX giant frames
  read/write 1: endpoint accepts 'giant' frames (longer than 1516/1522 bytes)
  read/write 0: 'giant' frames are dropped

- **A_HP** [*read/write*]: Accept RX High Priority frames
  read/write 1: endpoint accepts HP frames
  read/write 0: HP frames are dropped

- **KEEP_CRC** [*read/write*]: Keep CRC of received frames
  read/write 1: endpoint keeps FCS fields (CRC) on the fabric side
  read/write 0: FCS fields (CRC) are stripped

- **HPAP** [*read/write*]: RX Fiter HP Priorities
  Map of 802.1q PCP values which qualify the incoming frame as HP. Each bit corresponds to one PCP value (bit 7: PCP == 7, bit 0: PCP == 0).

- **MRU** [*read/write*]: Maximum receive unit (MRU)
  Maximum size of a frame which is considered valid (in bytes)

VLAN control register 0

HW prefix: ep_vcr0
HW address: 0x3
SW prefix: VCR0
SW offset: 0xc

31	30	29	28	27	26	25	24
-	-	-	-	PVID[11:8]			
23	22	21	20	19	18	17	16
PVID[7:0]							
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	PRIO_VAL[2:0]			-	FIX_PRIO	QMODE[1:0]	

- **QMODE** [*read/write*]: RX 802.1q port mode
  00: ACCESS port - tags untagged received packets with VID from RX_VID field. Drops all tagged packets not belonging to RX_VID VLAN
  01: TRUNK port - passes only tagged VLAN packets. Drops all untagged packets.
  10: VLAN disabled on port - passes the packets as is.
  11: unqualified port - passes all traffic regardless of VLAN configuration

- **FIX_PRIO** [*read/write*]: Force 802.1q priority
  1: ignores the 802.1x priority (if 802.1q header is present) and sets it to fixed value
  0: uses priority from 802.1q header

- **PRIO_VAL** [*read/write*]: Port-assigned 802.1q priority
  Packet priority value for retagging. When FIX_PRIO is 1, the endpoint uses this value as the packet priority. Otherwise, priority value is taken from 802.1q header if it's present. If there is no 802.1q header, the priority is assumed to be PRIO_VAL.

- **PVID** [*read/write*]: Port-assigned VID
  VLAN id value for tagging incoming packets if the port is in ACCESS mode. For TRUNK/unqualified the value of VID is ignored.

VLAN Control Register 1

HW prefix: ep_vcr1
HW address: 0x4
SW prefix: VCR1
SW offset: 0x10

Provides access to the egress VLAN untagged set and packet injection template buffer. In order to write to the buffer, set the DATA and OFFSET fields to the desired buffer location/value. The buffer layout goes as follows:
- the lower part (offsets 0 to 255) contains the VLAN untagged set bitmap. Each bit represents a single VLAN, where VID = OFFSET * 16 + bit position. For bits set to 1, VLAN headers containing corrensponding VID value are untagged.
- the higher part (offsets 512 to 1024) contains the packet injection template buffer. The buffer can store up to 8 packet templates of up to 128 bytes of size. Bits [15:0] of each entry contain the data value to be sent, bit 16 indicates the last word to transfer and bit 17 indicates that the current word shall be replaced by the user value (inject_user_value_i).

31	30	29	28	27	26	25	24
-	-	-	-	DATA[17:14]			
23	22	21	20	19	18	17	16
DATA[13:6]							
15	14	13	12	11	10	9	8
DATA[5:0]						OFFSET[9:8]	
7	6	5	4	3	2	1	0
OFFSET[7:0]							

- **OFFSET** [*write-only*]: VLAN Untagged Set/Injection Buffer offset
  Buffer address to be written

- **DATA** [*write-only*]: VLAN Untagged Set/Injection Buffer value
  Buffer value to be written

Packet Filter Control Register 0

HW prefix: ep_pfcr0
HW address: 0x5
SW prefix: PFCR0
SW offset: 0x14

Controls the microcode memory access of the Packet Filter Unit.
See the Endpoint documentation for more details

31	30	29	28	27	26	25	24
MM_DATA_MSB[23:16]							
23	22	21	20	19	18	17	16
MM_DATA_MSB[15:8]							
15	14	13	12	11	10	9	8
MM_DATA_MSB[7:0]							
7	6	5	4	3	2	1	0
ENABLE	MM_WRITE	MM_ADDR[5:0]					

- **MM_ADDR** [*write-only*]: Microcode Memory Address
- **MM_WRITE** [*write-only*]: Microcode Memory Write Enable
- **ENABLE** [*read/write*]: Packet Filter Enable
- **MM_DATA_MSB** [*write-only*]: Microcode Memory Data (24 MSBs)

Packet Filter Control Register 1

HW prefix: ep_pfcr1
HW address: 0x6
SW prefix: PFCR1
SW offset: 0x18

Controls the microcode memory access of the Packet Filter Unit.
See the Endpoint documentation for more details

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	MM_DATA_LSB[11:8]			
7	6	5	4	3	2	1	0
MM_DATA_LSB[7:0]							

- **MM_DATA_LSB** [*write-only*]: Microcode Memory Data (12 LSBs)

Traffic Class Assignment Register

HW prefix: ep_tcar
HW address: 0x7
SW prefix: TCAR
SW offset: 0x1c

Controls the mapping of VLAN priority fields into Swcore's traffic classes.. See Endpoint's documentation for more details.

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
PCP_MAP[23:16]							
15	14	13	12	11	10	9	8
PCP_MAP[15:8]							
7	6	5	4	3	2	1	0
PCP_MAP[7:0]							

- **PCP_MAP** [*read/write*]: 802.1Q priority tag to Traffic Class map
  Controls the mapping of PCP into Traffic Classes. The mapping algorithm is: TC = PCP_MAP[PCP * 3 + 2 : PCP * 3];

Flow Control Register

HW prefix: ep_fcr
HW address: 0x8
SW prefix: FCR
SW offset: 0x20

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
-	-	-	-	-	RXPAUSE_802_1Q	-	RXPAUSE

- **RXPAUSE** [*read/write*]: RX Pause 802.3 enable
  1: enable reception of pause frames defined in 802.3 (all priorities) and TX path throttling
  0: disable reception of pause frames defined in 802.3

- **RXPAUSE_802_1Q** [*read/write*]: Rx Pause 802.1Q enable
  1: enable reception of priority-based pause frames (IEEE 802.1Q-2012 Flow Control)
  0: disable reception of priority-based pause frames

Endpoint MAC address high part register

HW prefix: ep_mach
HW address: 0x9
SW prefix: MACH
SW offset: 0x24

Register containing bits [47:32] of the endpoint's MAC address

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
MACH[15:8]							
7	6	5	4	3	2	1	0
MACH[7:0]							

- **MACH** [*read/write*]: MAC Address
  MAC Address bits [47:32]

Endpoint MAC address low part register

HW prefix: ep_macl
HW address: 0xa
SW prefix: MACL
SW offset: 0x28

Register containing bits [31:0] of the endpoint's MAC address

31	30	29	28	27	26	25	24
MACL[31:24]							
23	22	21	20	19	18	17	16
MACL[23:16]							
15	14	13	12	11	10	9	8
MACL[15:8]							
7	6	5	4	3	2	1	0
MACL[7:0]							

- **MACL** [*read/write*]: MAC Address
  MAC Address bits [31:0]

MDIO Control Register

HW prefix: ep_mdio_cr
HW address: 0xb
SW prefix: MDIO_CR
SW offset: 0x2c

Register controlling the read/write operations on the MDIO PHY/PCS interface. Writing to this register clears the READY bit in the MDIO Status Register

31	30	29	28	27	26	25	24
RW	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
ADDR[7:0]							
15	14	13	12	11	10	9	8
DATA[15:8]							
7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA** [*write-only*]: MDIO Register Value
  Data word to be written to the MDIO
- **ADDR** [*read/write*]: MDIO Register Address
  Address of the MDIO register to be read/written
- **RW** [*read/write*]: MDIO Read/Write select
  1 = Performs a write to MDIO register at address ADDR with value DATA
  0 = Reads the value of MDIO register at address ADDR

MDIO Address/Status Register

HW prefix: ep_mdio_asr
HW address: 0xc
SW prefix: MDIO_ASR
SW offset: 0x30

Register with the current status of the MDIO interface

31	30	29	28	27	26	25	24
READY	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
PHYAD[7:0]							
15	14	13	12	11	10	9	8
RDATA[15:8]							
7	6	5	4	3	2	1	0
RDATA[7:0]							

- **RDATA** [*read-only*]: MDIO Read Value
  The value of the recently read MDIO register.
- **PHYAD** [*read/write*]: MDIO PHY Address
  Address of the PHY on the MDIO bus
- **READY** [*read-only*]: MDIO Ready
  1 = MDIO read/write operation is complete (for read operations, that means that RDATA contains a valid value)
  0 = MDIO operation in progress

Identification register

HW prefix: ep_idcode
HW address: 0xd
SW prefix: IDCODE
SW offset: 0x34

Constant value 0xcafebabe, can be used for identification of the Endpoint module

31	30	29	28	27	26	25	24
IDCODE[31:24]							
23	22	21	20	19	18	17	16
IDCODE[23:16]							
15	14	13	12	11	10	9	8
IDCODE[15:8]							
7	6	5	4	3	2	1	0
IDCODE[7:0]							

- **IDCODE** [*read-only*]: IDCode

Debug/Status register

HW prefix: ep_dsr
HW address: 0xe
SW prefix: DSR
SW offset: 0x38

Provides data useful for debugging

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
-	-	-	-	-	-	LACT	LSTATUS

- **LSTATUS** [*read-only*]: Link status
- **LACT** [*read/write*]: Link activity

DMTD Control Register

HW prefix: ep_dmcr
HW address: 0xf
SW prefix: DMCR
SW offset: 0x3c

31	30	29	28	27	26	25	24
-	-	-	-	N_AVG[11:8]			
23	22	21	20	19	18	17	16
N_AVG[7:0]							
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	EN

- **EN** [*read/write*]: DMTD Phase measurement enable
  write 1: enable DMTD phase measurement
  write 0: disable DMTD phase measurement
  read 1: DMTD phase measurement enabled
  read 0: DMTD phase measurement disabled

- **N_AVG** [*read/write*]: DMTD averaging samples
  Number of raw DMTD phase samples averaged in every measurement cycle

DMTD Status register

HW prefix: ep_dmsr
HW address: 0x10
SW prefix: DMSR
SW offset: 0x40

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	PS_RDY
23	22	21	20	19	18	17	16
PS_VAL[23:16]							
15	14	13	12	11	10	9	8
PS_VAL[15:8]							
7	6	5	4	3	2	1	0
PS_VAL[7:0]							

- **PS_VAL** [*read-only*]: DMTD Phase shift value

- **PS_RDY** [*read/write*]: DMTD Phase shift value ready

4.4 WR Endpoint 1000base-X TBI PCS register block

4.4.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	MDIO Control Register	mdio_mcr	MCR
0x1	REG	MDIO Status Register	mdio_msr	MSR
0x2	REG	MDIO PHY Identification Register 1	mdio_physid1	PHYSID1
0x3	REG	MDIO PHY Identification Register 2	mdio_physid2	PHYSID2
0x4	REG	MDIO Auto-Neg Advert Register	mdio_advertise	ADVERTISE
0x5	REG	MDIO Auto-Neg Link Partner Ability Register	mdio_lpa	LPA
0x6	REG	MDIO Auto-Negotiation Expansion Register	mdio_expansion	EXPANSION
0xf	REG	MDIO Extended Status Register	mdio_estatus	ESTATUS
0x10	REG	WhiteRabbit-specific Configuration Register	mdio_wr_spec	WR_SPEC

4.4.2 Register description

MDIO Control Register

HW prefix: mdio_mcr
HW address: 0x0
SW prefix: MCR
SW offset: 0x0

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
RESET	LOOPBACK	SPEED100	ANENABLE	PDOWN	ISOLATE	ANRESTART	FULLDPLX
7	6	5	4	3	2	1	0
CTST	SPEED1000	UNI_EN	RESV[4:0]				

- **RESV** [*read-only*]: Reserved
  read: always return 0s
  write: no effect

- **UNI_EN** [*read/write*]: Unidirectional Enable
  1 = Enable transmit regardless of whether a valid link has been established
  0 = Normal operation

- **SPEED1000** [*read-only*]: Speed Selection (MSB)
  Always 1, indicating (together with bit 13) a fixed speed of 1000 Mbps

- **CTST** [*read-only*]: Collision Test
  Always equal to 0, since collision detection is not supported

- **FULLDPLX** [*read-only*]: Duplex Mode
  Always equal to 1 to indicate Full-Duplex operation

- **ANRESTART** [*write-only*]: Restart Auto-Negotiation
  write 1: restart Auto-Negotiation process
  write 0: no effect

- **ISOLATE** [*read-only*]: Isolate
  GMII Electrical isolation enable. Ignored since the PCS doesn't use GMII

- **PDOWN** [*read/write*]: Power Down
  1: Power down
  0: Normal operation
  This bit controls directly the PHY Enable pin

- **ANENABLE** [*read/write*]: Auto-Negotiation Enable
  1: Enable Auto-Negotiation process
  0: Disable Auto-Negotiation process

- **SPEED100** [*read-only*]: Speed Selection (LSB)
  Always 0, indicating (together with bit 6) a fixed speed of 1000 Mbps

- **LOOPBACK** [*read/write*]: Loopback
  1: enable loopback mode
  0: disable loopback mode
  With the TBI version, loopback bit is connected to PHY loopback enable pin. When set to 1, indicates to the external PHY to enter loopback mode

- **RESET** [*write-only*]: Reset
  write 1: triggers reset of the PCS core
  write 0: no effect

MDIO Status Register

HW prefix: mdio_msr
HW address: 0x1
SW prefix: MSR
SW offset: 0x4

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
100BASE4	100FULL	100HALF	10FULL	10HALF	100FULL2	100HALF2	ESTATEN
7	6	5	4	3	2	1	0
UNIDIRABLE	MFSUPPRESS	ANEGCOMPLETE	RFAULT	ANEGCAPABLE	LSTATUS	JCD	ERCAP

- **ERCAP** [*read-only*]: Extended Capability
  Always 0, since extended register set is not supported

- **JCD** [*read-only*]: Jabber Detect
  Always 0, since Jabber Detect is not supported

- **LSTATUS** [*read-only*]: Link Status
  read 1: Link is up
  read 0: Link is down (or has been down)
  Latches '0' if Link Status goes down. Clears to current Link Status on read.

- **ANEGCAPABLE** [*read-only*]: Auto-Negotiation Ability
  Always 1, to indicate the support for Auto-Negotiation.

- **RFAULT** [*read-only*]: Remote Fault
  read 1: Remote fault condition detected
  read 0: No remote fault condition detected
  The bit clears itself after being read by the host.

- **ANEGCOMPLETE** [*read-only*]: Auto-Negotiation Complete
  read 1: Auto-Negotiation process completed
  read 0: Auto-Negotiation process not completed

- **MFSUPPRESS** [*read-only*]: MF Preamble Suppression
  Always 0, feature not supported.

- **UNIDIRABLE** [*read-only*]: Unidirectional Ability
  Always 1, as the Unidirectional mode is supported.

- **ESTATEN** [*read-only*]: Extended Status Capable
  Always 1, indicating the presence of the Extended Status Register

- **100HALF2** [*read-only*]: 100BASE-T2 Half Duplex
  Always 0 (unsupported medium)

- **100FULL2** [*read-only*]: 100BASE-T2 Full Duplex
  Always 0 (unsupported medium)

- **10HALF** [*read-only*]: 10 Mbps Half Duplex
  Always 0 (unsupported medium)

- **10FULL** [*read-only*]: 10 Mbps Full Duplex
  Always 0 (unsupported medium)

- **100HALF** [*read-only*]: 100BASE-X Half Duplex
  Always 0 (unsupported medium)

- **100FULL** [*read-only*]: 100BASE-X Full Duplex
  Always 0 (unsupported medium)

- **100BASE4** [*read-only*]: 100BASE-T4
  Always 0 (unsupported medium)

MDIO PHY Identification Register 1

HW prefix: mdio_physid1
HW address: 0x2
SW prefix: PHYSID1
SW offset: 0x8

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
OUI[15:8]							
7	6	5	4	3	2	1	0
OUI[7:0]							

- **OUI** [*read-only*]: Organizationally Unique Identifier (bits 7-21)
  Always 0.

MDIO PHY Identification Register 2

HW prefix: mdio_physid2
HW address: 0x3
SW prefix: PHYSID2
SW offset: 0xc

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
OUI[5:0]						MMNUM[5:4]	
7	6	5	4	3	2	1	0
MMNUM[3:0]				REV_NUM[3:0]			

- **REV_NUM** [*read-only*]: Revision Number
  Always 0.

- **MMNUM** [*read-only*]: Manufacturer Model Number
  Always 0.

- **OUI** [*read-only*]: Organizationally Unique Identifier (bits 0-5)
  Always 0.

MDIO Auto-Neg Advert Register

HW prefix: mdio_advertise
HW address: 0x4
SW prefix: ADVERTISE
SW offset: 0x10

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
NPAGE	RSVD1	RFAULT[1:0]		RSVD2[2:0]			PAUSE[1:1]
7	6	5	4	3	2	1	0
PAUSE[0:0]	HALF	FULL	RSVD3[4:0]				

- **RSVD3** [*read-only*]: Reserved
  Always 0.

- **FULL** [*read-only*]: Full Duplex
  Always 1, since Full Duplex Mode is the only supported mode.

- **HALF** [*read-only*]: Half Duplex
  Always 0, since Half Duplex Mode is not supported.

- **PAUSE** [*read/write*]: Pause
  00: No PAUSE
  01: Symmetric PAUSE
  10: Asymmetric PAUSE towards link partner
  11: Both Symmetric PAUSE and Asymmetric PAUSE towards link partner

- **RSVD2** [*read-only*]: Reserved
  Always 0.

- **RFAULT** [*read/write*]: Remote Fault
  00: No Error
  01: Offline
  10: Link Failure
  11: Auto-Negotiation Error

- **RSVD1** [*read-only*]: Reserved
  Always 0.

- **NPAGE** [*read-only*]: Next Page
  Always 0, since Next Page feature is not supported

MDIO Auto-Neg Link Partner Ability Register

HW prefix: mdio_lpa
HW address: 0x5
SW prefix: LPA
SW offset: 0x14

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
NPAGE	LPACK	RFAULT[1:0]		RSVD2[2:0]			PAUSE[1:1]
7	6	5	4	3	2	1	0
PAUSE[0:0]	HALF	FULL	RSVD3[4:0]				

- **RSVD3** [*read-only*]: Reserved
  Always 0.

- **FULL** [*read-only*]: Full Duplex
  read 1: Remote partner supports Full Duplex operation
  read 0: It doesn't

- **HALF** [*read-only*]: Half Duplex
  read 1: Remote partner supports Half Duplex operation
  read 0: Remote partner doesn't support Half Duplex

- **PAUSE** [*read-only*]: Pause
  read 00: No PAUSE
  read 01: Symmetric PAUSE
  read 10: Asymmetric PAUSE towards link partner
  read 11: Both Symmetric PAUSE and Asymmetric PAUSE towards link partner

- **RSVD2** [*read-only*]: Reserved
  Always 0.

- **RFAULT** [*read-only*]: Remote Fault
  read 00: No Error
  read 01: Offline
  read 10: Link Failure
  read 11: Auto-Negotiation Error

- **LPACK** [*read-only*]: Acknowledge
  Used by Auto-Negotiation function to indicate reception of a link partner's base or next page.

- **NPAGE** [*read-only*]: Next Page
  read 1: Next Page functionality is supported
  read 0: It isn't

MDIO Auto-Negotiation Expansion Register

HW prefix: mdio_expansion
HW address: 0x6
SW prefix: EXPANSION
SW offset: 0x18

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
RSVD2[12:5]							
7	6	5	4	3	2	1	0
RSVD2[4:0]					ENABLENPAGE	LWCP	RSVD1

- **RSVD1** [*read-only*]: Reserved
  Always 0.

- **LWCP** [*read-only*]: Page Received
  Always 0, since we don't support the Next Page function

- **ENABLENPAGE** [*read-only*]: Next Page Able
  Always 0, since we don't support the Next Page function

- **RSVD2** [*read-only*]: Reserved
  Always 0.

MDIO Extended Status Register

HW prefix: mdio_estatus
HW address: 0xf
SW prefix: ESTATUS
SW offset: 0x3c

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
1000_XFULL	1000_XHALF	1000_TFULL	1000_THALF	RSVD1[11:8]			
7	6	5	4	3	2	1	0
RSVD1[7:0]							

- **RSVD1** [*read-only*]: Reserved
  Always 0.

- **1000_THALF** [*read-only*]: 1000Base-T Half Duplex
  Always 0, since this mode is not supported.

- **1000_TFULL** [*read-only*]: 1000Base-T Full Duplex
  Always 0, since this mode is not supported.

- **1000_XHALF** [*read-only*]: 1000Base-X Half Duplex
  Always 0, since this mode is not supported.

- **1000_XFULL** [*read-only*]: 1000Base-X Full Duplex
  Always 1, indicating the support for 1000Base-X Full Duplex mode.

WhiteRabbit-specific Configuration Register

HW prefix: mdio_wr_spec
HW address: 0x10
SW prefix: WR_SPEC
SW offset: 0x40

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	BSLIDE[4:4]
7	6	5	4	3	2	1	0
BSLIDE[3:0]				-	CAL_CRST	RX_CAL_STAT	TX_CAL

- **TX_CAL** [*read/write*]: TX Calibration Pattern
  Controls the transmission of WR PHY calibration pattern.
  1: PCS is sending calibration pattern
  0: Normal PCS operation

- **RX_CAL_STAT** [*read-only*]: Calibration Pattern RX Status
  1: Valid calibration pattern is being received
  0: no calibration pattern detected by the receiver

- **CAL_CRST** [*write-only*]: Reset calibration counter
  write 1: resets the calibration pattern valid counter.
  write 0: no effect

- **BSLIDE** [*read-only*]: GTP RX Bitslide
  Current receive path bit slide (valid only for Xilinx GTP/GTX versions)

4.5 Vectored Interrupt Controller (VIC)

Module implementing a 2 to 32-input prioritized interrupt controller with internal interrupt vector storage support.

4.5.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	VIC Control Register	vic_ctl	CTL
0x1	REG	Raw Interrupt Status Register	vic_risr	RISR
0x2	REG	Interrupt Enable Register	vic_ier	IER
0x3	REG	Interrupt Disable Register	vic_idr	IDR
0x4	REG	Interrupt Mask Register	vic_imr	IMR
0x5	REG	Vector Address Register	vic_var	VAR
0x6	REG	Software Interrupt Register	vic_swir	SWIR
0x7	REG	End Of Interrupt Acknowledge Register	vic_eoir	EOIR
0x20 - 0x3f	MEM	Interrupt Vector Table	vic_ivt_ram	IVT_RAM

4.5.2 Register description

VIC Control Register

HW prefix: vic_ctl
HW address: 0x0
SW prefix: CTL
SW offset: 0x0

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	EMU_LEN[15:13]		
15	14	13	12	11	10	9	8
EMU_LEN[12:5]							
7	6	5	4	3	2	1	0
EMU_LEN[4:0]					EMU_EDGE	POL	ENABLE

- **ENABLE** [*read/write*]: VIC Enable
  write 1: enable VIC operation
  write 0: disable VIC operation
  read 1: VIC enabled
  read 0: VIC disabled

- **POL** [*read/write*]: VIC output polarity
  1: IRQ output is active high
  0: IRQ output is active low

- **EMU_EDGE** [*read/write*]: Emulate Edge sensitive output
  1: Forces a low pulse of `EMU_LEN` clock cycles at each write to `EOIR`. Useful for edge-only IRQ

controllers such as Gennum.
0: Normal IRQ master line behavior

- **EMU_LEN** [*read/write*]: Emulated Edge pulse timer
  Length of the delay (in `clk_sys_i` cycles) between write to `EOIR` and re-assertion of `irq_master_o`.

Raw Interrupt Status Register

HW prefix: vic_risr
HW address: 0x1
SW prefix: RISR
SW offset: 0x4

31	30	29	28	27	26	25	24
RISR[31:24]							
23	22	21	20	19	18	17	16
RISR[23:16]							
15	14	13	12	11	10	9	8
RISR[15:8]							
7	6	5	4	3	2	1	0
RISR[7:0]							

- **RISR** [*read-only*]: Raw interrupt status
  Each bit reflects the current state of corresponding IRQ input line.
  read 1: interrupt line is currently active
  read 0: interrupt line is inactive

Interrupt Enable Register

HW prefix: vic_ier
HW address: 0x2
SW prefix: IER
SW offset: 0x8

31	30	29	28	27	26	25	24
IER[31:24]							
23	22	21	20	19	18	17	16
IER[23:16]							
15	14	13	12	11	10	9	8
IER[15:8]							
7	6	5	4	3	2	1	0
IER[7:0]							

- **IER** [*write-only*]: Enable IRQ
  write 1: enables interrupt associated with written bit
  write 0: no effect

Interrupt Disable Register

HW prefix: vic_idr
HW address: 0x3
SW prefix: IDR
SW offset: 0xc

31	30	29	28	27	26	25	24
IDR[31:24]							
23	22	21	20	19	18	17	16
IDR[23:16]							
15	14	13	12	11	10	9	8
IDR[15:8]							
7	6	5	4	3	2	1	0
IDR[7:0]							

- **IDR** [*write-only*]: Disable IRQ
  write 1: enables interrupt associated with written bit
  write 0: no effect

Interrupt Mask Register

HW prefix: vic_imr
HW address: 0x4
SW prefix: IMR
SW offset: 0x10

31	30	29	28	27	26	25	24
IMR[31:24]							
23	22	21	20	19	18	17	16
IMR[23:16]							
15	14	13	12	11	10	9	8
IMR[15:8]							
7	6	5	4	3	2	1	0
IMR[7:0]							

- **IMR** [*read-only*]: IRQ disabled/enabled
  read 1: interrupt associated with read bit is enabled
  read 0: interrupt is disabled

Vector Address Register

HW prefix: vic_var
HW address: 0x5
SW prefix: VAR
SW offset: 0x14

31	30	29	28	27	26	25	24
VAR[31:24]							
23	22	21	20	19	18	17	16
VAR[23:16]							
15	14	13	12	11	10	9	8
VAR[15:8]							
7	6	5	4	3	2	1	0
VAR[7:0]							

- **VAR** [*read-only*]: Vector Address
  Address of pending interrupt vector, read from Interrupt Vector Table

Software Interrupt Register

HW prefix: vic_swir
HW address: 0x6
SW prefix: SWIR
SW offset: 0x18

Writing 1 to one of bits of this register causes a software emulation of the respective interrupt.

31	30	29	28	27	26	25	24
SWIR[31:24]							
23	22	21	20	19	18	17	16
SWIR[23:16]							
15	14	13	12	11	10	9	8
SWIR[15:8]							
7	6	5	4	3	2	1	0
SWIR[7:0]							

- **SWIR** [*write-only*]: SWI interrupt mask

End Of Interrupt Acknowledge Register

HW prefix: vic_eoir
HW address: 0x7
SW prefix: EOIR
SW offset: 0x1c

31	30	29	28	27	26	25	24
EOIR[31:24]							
23	22	21	20	19	18	17	16
EOIR[23:16]							
15	14	13	12	11	10	9	8
EOIR[15:8]							
7	6	5	4	3	2	1	0
EOIR[7:0]							

- **EOIR** [*write-only*]: End of Interrupt
  Any write operation acknowledges the pending interrupt. Then, VIC advances to another pending interrupt(s) or releases the master interrupt output.

Interrupt Vector Table

HW prefix:	vic_ivt_ram
HW address:	0x20
C prefix:	IVT_RAM
C offset:	0x80
Size:	32 32-bit words
Data width:	32
Access (bus):	read/write
Access (device):	read-only
Mirrored:	no
Byte-addressable:	no
Peripheral port:	bus-synchronous

Vector Address Table. Word at offset N stores the vector address of IRQ N. When interrupt is requested, VIC reads it's vector address from this memory and stores it in VAR register.

4.6 Shared TX Timestamping Unit (TXTSU)

4.6.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	Interrupt disable register	txtsu_eic_idr	EIC_IDR
0x1	REG	Interrupt enable register	txtsu_eic_ier	EIC_IER
0x2	REG	Interrupt mask register	txtsu_eic_imr	EIC_IMR
0x3	REG	Interrupt status register	txtsu_eic_isr	EIC_ISR
0x4	FIFOREG	FIFO 'Timestamp FIFO' data output register 0	txtsu_tsf_r0	TSF_R0
0x5	FIFOREG	FIFO 'Timestamp FIFO' data output register 1	txtsu_tsf_r1	TSF_R1
0x6	FIFOREG	FIFO 'Timestamp FIFO' data output register 2	txtsu_tsf_r2	TSF_R2
0x7	REG	FIFO 'Timestamp FIFO' control/status register	txtsu_tsf_csr	TSF_CSR

4.6.2 Register description

Interrupt disable register

HW prefix: txtsu_eic_idr
HW address: 0x0
SW prefix: EIC_IDR
SW offset: 0x0

Writing 1 disables handling of the interrupt associated with corresponding bit. Writin 0 has no effect.

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
-	-	-	-	-	-	-	NEMPTY

- **NEMPTY** [*write-only*]: TXTSU fifo not-empty
  write 1: disable interrupt 'TXTSU fifo not-empty'
  write 0: no effect

Interrupt enable register

HW prefix: txtsu_eic_ier
HW address: 0x1
SW prefix: EIC_IER
SW offset: 0x4

Writing 1 enables handling of the interrupt associated with corresponding bit. Writin 0 has no effect.

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
-	-	-	-	-	-	-	NEMPTY

- **NEMPTY** [*write-only*]: TXTSU fifo not-empty
  write 1: enable interrupt 'TXTSU fifo not-empty'
  write 0: no effect

Interrupt mask register

HW prefix: txtsu_eic_imr
HW address: 0x2
SW prefix: EIC_IMR
SW offset: 0x8

Shows which interrupts are enabled. 1 means that the interrupt associated with the bitfield is enabled

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
-	-	-	-	-	-	-	NEMPTY

- **NEMPTY** [*read-only*]: TXTSU fifo not-empty
  read 1: interrupt 'TXTSU fifo not-empty' is enabled
  read 0: interrupt 'TXTSU fifo not-empty' is disabled

Interrupt status register

HW prefix: txtsu_eic_isr
HW address: 0x3
SW prefix: EIC_ISR
SW offset: 0xc

Each bit represents the state of corresponding interrupt. 1 means the interrupt is pending. Writing 1 to a bit clears the corresponding interrupt. Writing 0 has no effect.

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
-	-	-	-	-	-	-	NEMPTY

- **NEMPTY** [*read/write*]: TXTSU fifo not-empty
  read 1: interrupt 'TXTSU fifo not-empty' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'TXTSU fifo not-empty'
  write 0: no effect

FIFO 'Timestamp FIFO' data output register 0

HW prefix: txtsu_tsf_r0
HW address: 0x4
SW prefix: TSF_R0
SW offset: 0x10

31	30	29	28	27	26	25	24
VAL_F[3:0]				VAL_R[27:24]			
23	22	21	20	19	18	17	16
VAL_R[23:16]							
15	14	13	12	11	10	9	8
VAL_R[15:8]							
7	6	5	4	3	2	1	0
VAL_R[7:0]							

- **VAL_R** [*read-only*]: Rising edge timestamp
- **VAL_F** [*read-only*]: Falling edge timestamp
  Timestamp value taken on falling clock edge (few LSBs)

FIFO 'Timestamp FIFO' data output register 1

HW prefix: txtsu_tsf_r1
HW address: 0x5
SW prefix: TSF_R1
SW offset: 0x14

31	30	29	28	27	26	25	24
FID[15:8]							
23	22	21	20	19	18	17	16
FID[7:0]							
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	PID[4:0]				

- **PID** [*read-only*]: Physical port ID
  Identifier of the TXTSU port to which came the timestamp. There may be multiple timestamps sharing the same FID value for broadcast/multicast packets.

- **FID** [*read-only*]: Frame ID
  OOB Frame Identifier. Used to associate the timestamp value with transmitted packet.

FIFO 'Timestamp FIFO' data output register 2

HW prefix: txtsu_tsf_r2
HW address: 0x6
SW prefix: TSF_R2
SW offset: 0x18

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
-	-	-	-	-	-	-	INCORRECT

- **INCORRECT** [*read-only*]: Timestamp (possibly) incorrect
  1: This timestamp may be incorrect (generated during PPS adjustment)
  0: Timestamp is correct.

FIFO 'Timestamp FIFO' control/status register

HW prefix: txtsu_tsf_csr
HW address: 0x7
SW prefix: TSF_CSR
SW offset: 0x1c

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	EMPTY	FULL
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
USEDW[7:0]							

- **FULL** [*read-only*]: FIFO full flag
  1: FIFO 'Timestamp FIFO' is full
  0: FIFO is not full

- **EMPTY** [*read-only*]: FIFO empty flag
  1: FIFO 'Timestamp FIFO' is empty
  0: FIFO is not empty

- **USEDW** [*read-only*]: FIFO counter
  Number of data records currently being stored in FIFO 'Timestamp FIFO'

4.6.3 Interrupts

TXTSU fifo not-empty

HW prefix: txtsu_nempty
C prefix: NEMPTY
Trigger: high level

Interrupt active when TXTSU shared FIFO contains any timestamps.

4.7 Wishbone GPIO

4.7.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	Clear Output Register	gpio_codr	CODR
0x1	REG	Set Output Register	gpio_sodr	SODR
0x2	REG	Data Direction Register	gpio_ddr	DDR
0x3	REG	Pin State Register	gpio_psr	PSR

4.7.2 Register description

Clear Output Register

HW prefix: gpio_codr
HW address: 0x0
SW prefix: CODR
SW offset: 0x0

31	30	29	28	27	26	25	24
CODR[31:24]							
23	22	21	20	19	18	17	16
CODR[23:16]							
15	14	13	12	11	10	9	8
CODR[15:8]							
7	6	5	4	3	2	1	0
CODR[7:0]							

- **CODR** [*write-only*]: Clear GPIO outputs
  Each bit corresponds to one GPIO line
  write 1 to bit n: clear n-th line
  write 0: no effect

Set Output Register

HW prefix: gpio_sodr
HW address: 0x1
SW prefix: SODR
SW offset: 0x4

31	30	29	28	27	26	25	24
SODR[31:24]							
23	22	21	20	19	18	17	16
SODR[23:16]							
15	14	13	12	11	10	9	8
SODR[15:8]							
7	6	5	4	3	2	1	0
SODR[7:0]							

- **SODR** [*write-only*]: Set GPIO outputs
  Each bit corresponds to one GPIO line
  write 1 to bit n: set n-th line to 1
  write 0: no effect

Data Direction Register

HW prefix: gpio_ddr
HW address: 0x2
SW prefix: DDR
SW offset: 0x8

31	30	29	28	27	26	25	24
DDR[31:24]							
23	22	21	20	19	18	17	16
DDR[23:16]							
15	14	13	12	11	10	9	8
DDR[15:8]							
7	6	5	4	3	2	1	0
DDR[7:0]							

- **DDR** [*read/write*]: GPIO direction
  Each bit corresponds to one GPIO line
  1: n-th line is output
  0: n-th line is input

Pin State Register

HW prefix: gpio_psr
HW address: 0x3
SW prefix: PSR
SW offset: 0xc

31	30	29	28	27	26	25	24
PSR[31:24]							
23	22	21	20	19	18	17	16
PSR[23:16]							
15	14	13	12	11	10	9	8
PSR[15:8]							
7	6	5	4	3	2	1	0
PSR[7:0]							

- **PSR** [*read-only*]: Read GPIO inputs
  Each bit corresponds to one GPIO line
  read: current status of n-th input

4.8 Wishbone I2C Master

4.8.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	Clock prescale register LSB	i2c_prer_lsb	PRER_LSB
0x1	REG	Clock prescale register MSB	i2c_prer_msb	PRER_MSB
0x2	REG	Control register	i2c_ctr	CTR
0x3	REG	Transmit/Receive register	i2c_txrx	TXRX
0x4	REG	Command/Status register	i2c_crsr	CRSR

4.8.2 Register description

Clock prescale register LSB

HW prefix: i2c_prer_lsb
HW address: 0x0
SW prefix: PRER_LSB
SW offset: 0x0

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
PRER_LSB[7:0]							

- **PRER_LSB** [*read/write*]: Clock prescale LSB
  Bits 7:0 of 16-bit PRER register

Clock prescale register MSB

HW prefix: i2c_prer_msb
HW address: 0x1
SW prefix: PRER_MSB
SW offset: 0x4

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
PRER_MSB[7:0]							

- **PRER_MSB** [*read/write*]: Clock prescale MSB
  Bits 15:8 of 16-bit PRER register. Register stores the prescale value for SCL clock
  scl_clk = clk_sys / (5* PRER)
  a new value can be stored only when the module is disabled (CTR register)

Control register

HW prefix: i2c_ctr
HW address: 0x2
SW prefix: CTR
SW offset: 0x8

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
EN	IEN	RSV[5:0]					

- **RSV** [*read/write*]: reserved

- **IEN** [*read/write*]: Interrupt enable
  Enable interrupt generation
  1: interrupt enabled
  0: interrupt disabled

- **EN** [*read/write*]: Enable module
  1: module enabled
  0: module disabled

Transmit/Receive register

HW prefix: i2c_txrx
HW address: 0x3
SW prefix: TXRX
SW offset: 0xc

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
TXRX[7:0]							

- **TXRX** [*read/write*]: value
  write: byte to be transmitted to i2c bus
  read: byte received from i2c bus

Command/Status register

HW prefix: i2c_crsr
HW address: 0x4
SW prefix: CRSR
SW offset: 0x10

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
CRSR[7:0]							

- **CRSR** [*read/write*]: value
  write:
  bit 7: generate (repeat) start condition
  bit 6: generate stop consitoin
  bit 5: read from slave
  bit 4: write to slave
  bit 3: if 0, send ACK; if 1, send NACK
  bit 0: acknowledge interrupt
  read:
  bit 7: if 0, received ACK; if 1, no ACK received
  bit 6: i2c bus busy
  bit 5: i2c bus arbitration lost
  bit 1: transfer in progress
  bit 0: interrupt pending

4.9 Simple Pulse Width Modulation Controller

A very simple multichannel PWM controller.

4.9.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	Control Register	spwm_cr	CR
0x1	REG	Status Register	spwm_sr	SR
0x2	REG	Channel 0 Drive Register	spwm_dr0	DR0
0x3	REG	Channel 1 Drive Register	spwm_dr1	DR1
0x4	REG	Channel 2 Drive Register	spwm_dr2	DR2
0x5	REG	Channel 3 Drive Register	spwm_dr3	DR3
0x6	REG	Channel 4 Drive Register	spwm_dr4	DR4
0x7	REG	Channel 5 Drive Register	spwm_dr5	DR5
0x8	REG	Channel 6 Drive Register	spwm_dr6	DR6
0x9	REG	Channel 7 Drive Register	spwm_dr7	DR7

4.9.2 Register description

Control Register

HW prefix: spwm_cr
HW address: 0x0
SW prefix: CR
SW offset: 0x0

31	30	29	28	27	26	25	24
PERIOD[15:8]							
23	22	21	20	19	18	17	16
PERIOD[7:0]							
15	14	13	12	11	10	9	8
PRESC[15:8]							
7	6	5	4	3	2	1	0
PRESC[7:0]							

- **PRESC** [*read/write*]: Prescaler Ratio
  PWM Base clock prescaler. Divides the system clock to obtain the PWM counter clock. The division ratio is (PRESC + 1).

- **PERIOD** [*read/write*]: Period
  PWM Cycle Period. The real-time period value `PERIOD * (PRESC + 1) * t_clk_sys`. Acceptable values: 0..65534.

Status Register

HW prefix: spwm_sr
HW address: 0x1
SW prefix: SR
SW offset: 0x4

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
-	-	-	-	N_CHANNELS[3:0]			

- **N_CHANNELS** [*read-only*]: Channel count
  Number of channels supported by this particular implementation, from 1 to 8.

Channel 0 Drive Register

HW prefix: spwm_dr0
HW address: 0x2
SW prefix: DR0
SW offset: 0x8

Current PWM duty cycle for channel

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DR0[15:8]							
7	6	5	4	3	2	1	0
DR0[7:0]							

- **DR0** [*read/write*]: Value

Channel 1 Drive Register

HW prefix: spwm_dr1
HW address: 0x3
SW prefix: DR1
SW offset: 0xc

Current PWM duty cycle for channel

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DR1[15:8]							
7	6	5	4	3	2	1	0
DR1[7:0]							

- **DR1** [*read/write*]: Value

Channel 2 Drive Register

HW prefix: spwm_dr2
HW address: 0x4
SW prefix: DR2
SW offset: 0x10

Current PWM duty cycle for channel

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DR2[15:8]							
7	6	5	4	3	2	1	0
DR2[7:0]							

- **DR2** [*read/write*]: Value

Channel 3 Drive Register

HW prefix: spwm_dr3
HW address: 0x5
SW prefix: DR3
SW offset: 0x14

Current PWM duty cycle for channel

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DR3[15:8]							
7	6	5	4	3	2	1	0
DR3[7:0]							

- **DR3** [*read/write*]: Value

Channel 4 Drive Register

HW prefix: spwm_dr4
HW address: 0x6
SW prefix: DR4
SW offset: 0x18

Current PWM duty cycle for channel

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DR4[15:8]							
7	6	5	4	3	2	1	0
DR4[7:0]							

- **DR4** [*read/write*]: Value

Channel 5 Drive Register

HW prefix: spwm_dr5
HW address: 0x7
SW prefix: DR5
SW offset: 0x1c

Current PWM duty cycle for channel

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DR5[15:8]							
7	6	5	4	3	2	1	0
DR5[7:0]							

- **DR5** [*read/write*]: Value

Channel 6 Drive Register

HW prefix: spwm_dr6
HW address: 0x8
SW prefix: DR6
SW offset: 0x20

Current PWM duty cycle for channel

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DR6[15:8]							
7	6	5	4	3	2	1	0
DR6[7:0]							

- **DR6** [*read/write*]: Value

Channel 7 Drive Register

HW prefix: spwm_dr7
HW address: 0x9
SW prefix: DR7
SW offset: 0x24

Current PWM duty cycle for channel

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DR7[15:8]							
7	6	5	4	3	2	1	0
DR7[7:0]							

- **DR7** [*read/write*]: Value

4.10 Topology Resolution Unit (TRU)

4.10.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	TRU Global Control Register	tru_gcr	GCR

4.10.2 Register description

TRU Global Control Register

HW prefix: tru_gcr
HW address: 0x0
SW prefix: GCR
SW offset: 0x0

Control register containing global (port-independent) settings of the TRU.

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
-	-	-	-	-	-	-	G_ENA

- **G_ENA** [*read/write*]: TRU Global Enable
  Global TRU enable bit. Overrides all port settings.
  0: TRU is disabled, it does not affect the forwarding response
  1: TRU is enabled.

4.11 Time Aware Traffic Shaper

4.11.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	TATSU Control Register/Status	tatsu_tcr	TCR

4.11.2 Register description

TATSU Control Register/Status

HW prefix: tatsu_tcr
HW address: 0x0
SW prefix: TCR
SW offset: 0x0

General TATSU control and status register

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
-	-	-	-	-	-	DISABLE	RESERVED

- **RESERVED** [*write-only*]:
- **DISABLE** [*write-only*]: Stop TATSU
  write 1: disable Time Aware Traffic Shapper
  write 0: no effect

4.12 WR Switch Per-Port Statistic Counters

The set of counters for counting traffic statistics on each Ethernet port of WR Switch

4.12.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	Control Register	pstats_cr	CR
0x1	REG	L1 Counter Value/1st word of IRQ state	pstats_l1_cnt_val	L1_CNT_VAL
0x2	REG	L2 Counter Value/2nd word of IRQ state	pstats_l2_cnt_val	L2_CNT_VAL
0x3	REG	Debug register	pstats_dbg	DBG
0x8	REG	Interrupt disable register	pstats_eic_idr	EIC_IDR
0x9	REG	Interrupt enable register	pstats_eic_ier	EIC_IER
0xa	REG	Interrupt mask register	pstats_eic_imr	EIC_IMR
0xb	REG	Interrupt status register	pstats_eic_isr	EIC_ISR

4.12.2 Register description

Control Register

HW prefix: pstats_cr
HW address: 0x0
SW prefix: CR
SW offset: 0x0

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	ADDR[4:0]				
15	14	13	12	11	10	9	8
-	-	-	PORT[4:0]				
7	6	5	4	3	2	1	0
-	-	-	-	-	-	RD_IRQ	RD_EN

- **RD_EN** [*read/write*]: Enable transfer of the selected counter's content
  write 1: start reading content
  write 0: no effect
  read 1: reading in progress
  read 0: reading done, counter value available

- **RD_IRQ** [*read/write*]: Enable transfer of per-counter IRQ state for selected counters
  write 1: start reading content
  write 0: no effect
  read 1: reading in progress
  read 0: reading done, counter value available

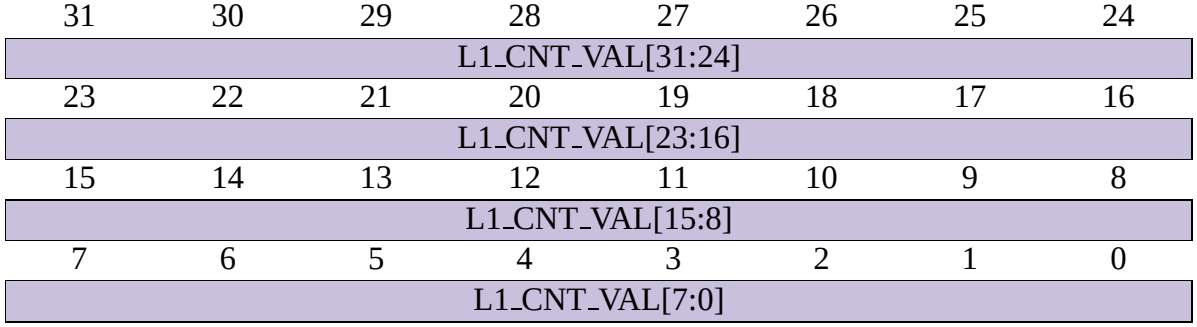
- **PORT** [*read/write*]: Port number
  Number of port (0-17) from which couter's value (or per-counter IRQ state) is read

- **ADDR** [*read/write*]: Memory address
  Address of the 32-bit word in selected port's memory that contains the counter to be read

L1 Counter Value/1st word of IRQ state

HW prefix: pstats_l1_cnt_val
HW address: 0x1
SW prefix: L1_CNT_VAL
SW offset: 0x4

32-bit word read from given memory address of selected Ethernet port containing the value of 4 counters
or lower half of per-counter IRQ state for port given in CR register

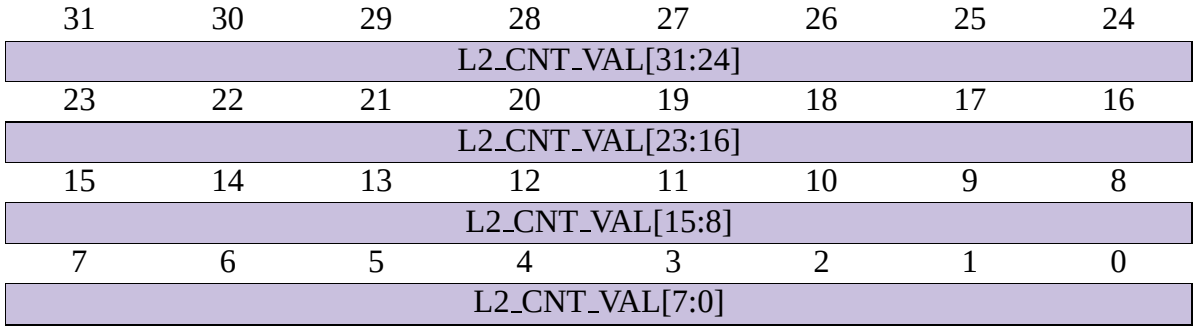


- **L1_CNT_VAL** [*read-only*]: 4 counters' values

L2 Counter Value/2nd word of IRQ state

HW prefix: pstats_l2_cnt_val
HW address: 0x2
SW prefix: L2_CNT_VAL
SW offset: 0x8

32-bit word read from given memory address of selected Ethernet port containing the value of 4 counters
or higher half of per-counter IRQ state for port given in CR register (if more than 32 counters per-port instanciated)



- **L2_CNT_VAL** [*read-only*]: 4 counters' values

Debug register

HW prefix: pstats_dbg
HW address: 0x3
SW prefix: DBG
SW offset: 0xc

31	30	29	28	27	26	25	24
CLR	L2_CLR	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	L2_EVT_OV	EVT_OV[17:16]	
15	14	13	12	11	10	9	8
EVT_OV[15:8]							
7	6	5	4	3	2	1	0
EVT_OV[7:0]							

- **EVT_OV** [*read-only*]: Layer-1 events overflow
  Some Layer-1 event on the port was missed because there were too many of them
  each bit corresponds to one port (bit set to 1 means event overflow)

- **L2_EVT_OV** [*read-only*]: Layer-2 events overflow
  read 1: some Layer-2 event (i.e. L-1 overflow) was missed, because there were too many of them
  read 0: no events overflow on Layer-2

- **L2_CLR** [*write-only*]: Layer-2 clear flag
  write 1: clear Layer-2 events overflow flag
  write 0: no effect

- **CLR** [*write-only*]: Layer-1 clear flags
  write 1: clear Layer-1 events overflow flags
  write 0: no effect

Interrupt disable register

HW prefix: pstats_eic_idr
HW address: 0x8
SW prefix: EIC_IDR
SW offset: 0x20

Writing 1 disables handling of the interrupt associated with corresponding bit. Writin 0 has no effect.

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PORT17	PORT16
15	14	13	12	11	10	9	8
PORT15	PORT14	PORT13	PORT12	PORT11	PORT10	PORT9	PORT8
7	6	5	4	3	2	1	0
PORT7	PORT6	PORT5	PORT4	PORT3	PORT2	PORT1	PORT0

- **PORT0** [*write-only*]: Port0 IRQ
  write 1: disable interrupt 'Port0 IRQ'
  write 0: no effect

- **PORT1** [*write-only*]: Port1 IRQ
  write 1: disable interrupt 'Port1 IRQ'
  write 0: no effect

- **PORT2** [*write-only*]: Port2 IRQ
  write 1: disable interrupt 'Port2 IRQ'
  write 0: no effect

- **PORT3** [*write-only*]: Port3 IRQ
  write 1: disable interrupt 'Port3 IRQ'
  write 0: no effect

- **PORT4** [*write-only*]: Port4 IRQ
  write 1: disable interrupt 'Port4 IRQ'
  write 0: no effect

- **PORT5** [*write-only*]: Port5 IRQ
  write 1: disable interrupt 'Port5 IRQ'
  write 0: no effect

- **PORT6** [*write-only*]: Port6 IRQ
  write 1: disable interrupt 'Port6 IRQ'
  write 0: no effect

- **PORT7** [*write-only*]: Port7 IRQ
  write 1: disable interrupt 'Port7 IRQ'
  write 0: no effect

- **PORT8** [*write-only*]: Port8 IRQ
  write 1: disable interrupt 'Port8 IRQ'
  write 0: no effect

- **PORT9** [*write-only*]: Port9 IRQ
  write 1: disable interrupt 'Port9 IRQ'
  write 0: no effect

- **PORT10** [*write-only*]: Port10 IRQ
  write 1: disable interrupt 'Port10 IRQ'
  write 0: no effect
- **PORT11** [*write-only*]: Port11 IRQ
  write 1: disable interrupt 'Port11 IRQ'
  write 0: no effect
- **PORT12** [*write-only*]: Port12 IRQ
  write 1: disable interrupt 'Port12 IRQ'
  write 0: no effect
- **PORT13** [*write-only*]: Port13 IRQ
  write 1: disable interrupt 'Port13 IRQ'
  write 0: no effect
- **PORT14** [*write-only*]: Port14 IRQ
  write 1: disable interrupt 'Port14 IRQ'
  write 0: no effect
- **PORT15** [*write-only*]: Port15 IRQ
  write 1: disable interrupt 'Port15 IRQ'
  write 0: no effect
- **PORT16** [*write-only*]: Port16 IRQ
  write 1: disable interrupt 'Port16 IRQ'
  write 0: no effect
- **PORT17** [*write-only*]: Port17 IRQ
  write 1: disable interrupt 'Port17 IRQ'
  write 0: no effect

Interrupt enable register

HW prefix: pstats_eic_ier
HW address: 0x9
SW prefix: EIC_IER
SW offset: 0x24

Writing 1 enables handling of the interrupt associated with corresponding bit. Writin 0 has no effect.

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PORT17	PORT16
15	14	13	12	11	10	9	8
PORT15	PORT14	PORT13	PORT12	PORT11	PORT10	PORT9	PORT8
7	6	5	4	3	2	1	0
PORT7	PORT6	PORT5	PORT4	PORT3	PORT2	PORT1	PORT0

- **PORT0** [*write-only*]: Port0 IRQ
  write 1: enable interrupt 'Port0 IRQ'
  write 0: no effect
- **PORT1** [*write-only*]: Port1 IRQ
  write 1: enable interrupt 'Port1 IRQ'
  write 0: no effect
- **PORT2** [*write-only*]: Port2 IRQ
  write 1: enable interrupt 'Port2 IRQ'
  write 0: no effect
- **PORT3** [*write-only*]: Port3 IRQ
  write 1: enable interrupt 'Port3 IRQ'
  write 0: no effect
- **PORT4** [*write-only*]: Port4 IRQ
  write 1: enable interrupt 'Port4 IRQ'
  write 0: no effect
- **PORT5** [*write-only*]: Port5 IRQ
  write 1: enable interrupt 'Port5 IRQ'
  write 0: no effect
- **PORT6** [*write-only*]: Port6 IRQ
  write 1: enable interrupt 'Port6 IRQ'
  write 0: no effect
- **PORT7** [*write-only*]: Port7 IRQ
  write 1: enable interrupt 'Port7 IRQ'
  write 0: no effect
- **PORT8** [*write-only*]: Port8 IRQ
  write 1: enable interrupt 'Port8 IRQ'
  write 0: no effect
- **PORT9** [*write-only*]: Port9 IRQ
  write 1: enable interrupt 'Port9 IRQ'
  write 0: no effect
- **PORT10** [*write-only*]: Port10 IRQ
  write 1: enable interrupt 'Port10 IRQ'
  write 0: no effect
- **PORT11** [*write-only*]: Port11 IRQ
  write 1: enable interrupt 'Port11 IRQ'
  write 0: no effect
- **PORT12** [*write-only*]: Port12 IRQ
  write 1: enable interrupt 'Port12 IRQ'
  write 0: no effect

- **PORT13** [*write-only*]: Port13 IRQ
  write 1: enable interrupt 'Port13 IRQ'
  write 0: no effect

- **PORT14** [*write-only*]: Port14 IRQ
  write 1: enable interrupt 'Port14 IRQ'
  write 0: no effect

- **PORT15** [*write-only*]: Port15 IRQ
  write 1: enable interrupt 'Port15 IRQ'
  write 0: no effect

- **PORT16** [*write-only*]: Port16 IRQ
  write 1: enable interrupt 'Port16 IRQ'
  write 0: no effect

- **PORT17** [*write-only*]: Port17 IRQ
  write 1: enable interrupt 'Port17 IRQ'
  write 0: no effect

Interrupt mask register

HW prefix: pstats_eic_imr
HW address: 0xa
SW prefix: EIC_IMR
SW offset: 0x28

Shows which interrupts are enabled. 1 means that the interrupt associated with the bitfield is enabled

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PORT17	PORT16
15	14	13	12	11	10	9	8
PORT15	PORT14	PORT13	PORT12	PORT11	PORT10	PORT9	PORT8
7	6	5	4	3	2	1	0
PORT7	PORT6	PORT5	PORT4	PORT3	PORT2	PORT1	PORT0

- **PORT0** [*read-only*]: Port0 IRQ
  read 1: interrupt 'Port0 IRQ' is enabled
  read 0: interrupt 'Port0 IRQ' is disabled

- **PORT1** [*read-only*]: Port1 IRQ
  read 1: interrupt 'Port1 IRQ' is enabled
  read 0: interrupt 'Port1 IRQ' is disabled

- **PORT2** [*read-only*]: Port2 IRQ
  read 1: interrupt 'Port2 IRQ' is enabled
  read 0: interrupt 'Port2 IRQ' is disabled

- **PORT3** [*read-only*]: Port3 IRQ
  read 1: interrupt 'Port3 IRQ' is enabled
  read 0: interrupt 'Port3 IRQ' is disabled

- **PORT4** [*read-only*]: Port4 IRQ
  read 1: interrupt 'Port4 IRQ' is enabled
  read 0: interrupt 'Port4 IRQ' is disabled

- **PORT5** [*read-only*]: Port5 IRQ
  read 1: interrupt 'Port5 IRQ' is enabled
  read 0: interrupt 'Port5 IRQ' is disabled

- **PORT6** [*read-only*]: Port6 IRQ
  read 1: interrupt 'Port6 IRQ' is enabled
  read 0: interrupt 'Port6 IRQ' is disabled

- **PORT7** [*read-only*]: Port7 IRQ
  read 1: interrupt 'Port7 IRQ' is enabled
  read 0: interrupt 'Port7 IRQ' is disabled

- **PORT8** [*read-only*]: Port8 IRQ
  read 1: interrupt 'Port8 IRQ' is enabled
  read 0: interrupt 'Port8 IRQ' is disabled

- **PORT9** [*read-only*]: Port9 IRQ
  read 1: interrupt 'Port9 IRQ' is enabled
  read 0: interrupt 'Port9 IRQ' is disabled

- **PORT10** [*read-only*]: Port10 IRQ
  read 1: interrupt 'Port10 IRQ' is enabled
  read 0: interrupt 'Port10 IRQ' is disabled

- **PORT11** [*read-only*]: Port11 IRQ
  read 1: interrupt 'Port11 IRQ' is enabled
  read 0: interrupt 'Port11 IRQ' is disabled

- **PORT12** [*read-only*]: Port12 IRQ
  read 1: interrupt 'Port12 IRQ' is enabled
  read 0: interrupt 'Port12 IRQ' is disabled

- **PORT13** [*read-only*]: Port13 IRQ
  read 1: interrupt 'Port13 IRQ' is enabled
  read 0: interrupt 'Port13 IRQ' is disabled

- **PORT14** [*read-only*]: Port14 IRQ
  read 1: interrupt 'Port14 IRQ' is enabled
  read 0: interrupt 'Port14 IRQ' is disabled

- **PORT15** [*read-only*]: Port15 IRQ
  read 1: interrupt 'Port15 IRQ' is enabled
  read 0: interrupt 'Port15 IRQ' is disabled

- **PORT16** [*read-only*]: Port16 IRQ
  read 1: interrupt 'Port16 IRQ' is enabled
  read 0: interrupt 'Port16 IRQ' is disabled

- **PORT17** [*read-only*]: Port17 IRQ
  read 1: interrupt 'Port17 IRQ' is enabled
  read 0: interrupt 'Port17 IRQ' is disabled

Interrupt status register

HW prefix: pstats_eic_isr
HW address: 0xb
SW prefix: EIC_ISR
SW offset: 0x2c

Each bit represents the state of corresponding interrupt. 1 means the interrupt is pending. Writing 1 to a bit clears the corresponding interrupt. Writing 0 has no effect.

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	PORT17	PORT16
15	14	13	12	11	10	9	8
PORT15	PORT14	PORT13	PORT12	PORT11	PORT10	PORT9	PORT8
7	6	5	4	3	2	1	0
PORT7	PORT6	PORT5	PORT4	PORT3	PORT2	PORT1	PORT0

- **PORT0** [*read/write*]: Port0 IRQ
  read 1: interrupt 'Port0 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port0 IRQ'
  write 0: no effect

- **PORT1** [*read/write*]: Port1 IRQ
  read 1: interrupt 'Port1 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port1 IRQ'
  write 0: no effect

- **PORT2** [*read/write*]: Port2 IRQ
  read 1: interrupt 'Port2 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port2 IRQ'
  write 0: no effect

- **PORT3** [*read/write*]: Port3 IRQ
  read 1: interrupt 'Port3 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port3 IRQ'
  write 0: no effect

- **PORT4** [*read/write*]: Port4 IRQ
  read 1: interrupt 'Port4 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port4 IRQ'
  write 0: no effect

- **PORT5** [*read/write*]: Port5 IRQ
  read 1: interrupt 'Port5 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port5 IRQ'
  write 0: no effect

- **PORT6** [*read/write*]: Port6 IRQ
  read 1: interrupt 'Port6 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port6 IRQ'
  write 0: no effect

- **PORT7** [*read/write*]: Port7 IRQ
  read 1: interrupt 'Port7 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port7 IRQ'
  write 0: no effect

- **PORT8** [*read/write*]: Port8 IRQ
  read 1: interrupt 'Port8 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port8 IRQ'
  write 0: no effect

- **PORT9** [*read/write*]: Port9 IRQ
  read 1: interrupt 'Port9 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port9 IRQ'
  write 0: no effect

- **PORT10** [*read/write*]: Port10 IRQ
  read 1: interrupt 'Port10 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port10 IRQ'
  write 0: no effect

- **PORT11** [*read/write*]: Port11 IRQ
  read 1: interrupt 'Port11 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port11 IRQ'
  write 0: no effect

- **PORT12** [*read/write*]: Port12 IRQ
  read 1: interrupt 'Port12 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port12 IRQ'
  write 0: no effect

- **PORT13** [*read/write*]: Port13 IRQ
  read 1: interrupt 'Port13 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port13 IRQ'
  write 0: no effect

- **PORT14** [*read/write*]: Port14 IRQ
  read 1: interrupt 'Port14 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port14 IRQ'
  write 0: no effect

- **PORT15** [*read/write*]: Port15 IRQ
  read 1: interrupt 'Port15 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port15 IRQ'
  write 0: no effect

- **PORT16** [*read/write*]: Port16 IRQ
  read 1: interrupt 'Port16 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port16 IRQ'
  write 0: no effect

- **PORT17** [*read/write*]: Port17 IRQ
  read 1: interrupt 'Port17 IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'Port17 IRQ'
  write 0: no effect

4.12.3 Interrupts

Port0 IRQ

HW prefix: pstats_port0
C prefix: PORT0
Trigger: high level

At least one of the counters on Port0 has overflown

Port1 IRQ

HW prefix: pstats_port1
C prefix: PORT1
Trigger: high level

At least one of the counters on Port1 has overflown

Port2 IRQ

HW prefix: pstats_port2
C prefix: PORT2
Trigger: high level

At least one of the counters on Port2 has overflown

Port3 IRQ

HW prefix: pstats_port3
C prefix: PORT3
Trigger: high level

At least one of the counters on Port3 has overflown

Port4 IRQ

HW prefix: pstats_port4
C prefix: PORT4
Trigger: high level

At least one of the counters on Port4 has overflown

Port5 IRQ

HW prefix: pstats_port5
C prefix: PORT5
Trigger: high level

At least one of the counters on Port5 has overflown

Port6 IRQ

HW prefix: pstats_port6
C prefix: PORT6
Trigger: high level

At least one of the counters on Port6 has overflown

Port7 IRQ

HW prefix: pstats_port7
C prefix: PORT7
Trigger: high level

At least one of the counters on Port7 has overflown

Port8 IRQ

HW prefix: pstats_port8
C prefix: PORT8
Trigger: high level

At least one of the counters on Port8 has overflown

Port9 IRQ

HW prefix: pstats_port9
C prefix: PORT9
Trigger: high level

At least one of the counters on Port9 has overflown

Port10 IRQ

HW prefix: pstats_port10
C prefix: PORT10
Trigger: high level

At least one of the counters on Port10 has overflown

Port11 IRQ

HW prefix: pstats_port11
C prefix: PORT11
Trigger: high level

At least one of the counters on Port11 has overflown

Port12 IRQ

HW prefix: pstats_port12
C prefix: PORT12
Trigger: high level

At least one of the counters on Port12 has overflown

Port13 IRQ

HW prefix: pstats_port13
C prefix: PORT13
Trigger: high level

At least one of the counters on Port13 has overflown

Port14 IRQ

HW prefix: pstats_port14
C prefix: PORT14
Trigger: high level

At least one of the counters on Port14 has overflown

Port15 IRQ

HW prefix: pstats_port15
C prefix: PORT15
Trigger: high level

At least one of the counters on Port15 has overflown

Port16 IRQ

HW prefix: pstats_port16
C prefix: PORT16
Trigger: high level

At least one of the counters on Port16 has overflown

Port17 IRQ

HW prefix: pstats_port17
C prefix: PORT17
Trigger: high level

At least one of the counters on Port17 has overflown

4.13 Routing Table Unit (RTU)

4.13.1 Memory map summary

H/W Addr	Type	Name	HW prefix	C prefix
0x0	REG	RTU Global Control Register	rtu_gcr	GCR
0x1	REG	Port Select Register	rtu_psr	PSR
0x2	REG	Port Control Register	rtu_pcr	PCR
0x3	REG	VLAN Table Register 1	rtu_vtr1	VTR1
0x4	REG	VLAN Table Register 2	rtu_vtr2	VTR2
0x5	REG	Ext: Control Register	rtu_rx_ctr	RX_CTR
0x6	REG	Ext: Fast Forward MAC[31:0]	rtu_rx_ff_mac_r0	RX_FF_MAC_R0
0x7	REG	Ext: Fast Forward MAC and control	rtu_rx_ff_mac_r1	RX_FF_MAC_R1
0x8	REG	Ext: CPU port mask	rtu_cpu_port	CPU_PORT
0x9	REG	Ext: Mirroring Ports Control Reg 0	rtu_rx_mp_r0	RX_MP_R0
0xa	REG	Ext: Mirroring Ports Control Reg 1	rtu_rx_mp_r1	RX_MP_R1
0x10	REG	Interrupt disable register	rtu_eic_idr	EIC_IDR
0x11	REG	Interrupt enable register	rtu_eic_ier	EIC_IER
0x12	REG	Interrupt mask register	rtu_eic_imr	EIC_IMR
0x13	REG	Interrupt status register	rtu_eic_isr	EIC_ISR
0x14	FIFOREG	FIFO 'UFIFO' data output register 0	rtu_ufifo_r0	UFIFO_R0
0x15	FIFOREG	FIFO 'UFIFO' data output register 1	rtu_ufifo_r1	UFIFO_R1
0x16	FIFOREG	FIFO 'UFIFO' data output register 2	rtu_ufifo_r2	UFIFO_R2
0x17	FIFOREG	FIFO 'UFIFO' data output register 3	rtu_ufifo_r3	UFIFO_R3
0x18	FIFOREG	FIFO 'UFIFO' data output register 4	rtu_ufifo_r4	UFIFO_R4
0x19	REG	FIFO 'UFIFO' control/status register	rtu_ufifo_csr	UFIFO_CSR
0x1a	FIFOREG	FIFO 'MFIFO' data input register 0	rtu_mfifo_r0	MFIFO_R0
0x1b	FIFOREG	FIFO 'MFIFO' data input register 1	rtu_mfifo_r1	MFIFO_R1
0x1c	REG	FIFO 'MFIFO' control/status register	rtu_mfifo_csr	MFIFO_CSR
0x100 - 0x1ff	MEM	Aging bitmap for main hashtable	rtu_aram	ARAM

4.13.2 Register description

RTU Global Control Register

HW prefix: rtu_gcr
HW address: 0x0
SW prefix: GCR
SW offset: 0x0

Control register containing global (port-independent) settings of the RTU.

31	30	29	28	27	26	25	24
-	-	-	-	RTU_VERSION[3:0]			
23	22	21	20	19	18	17	16
POLY_VAL[15:8]							
15	14	13	12	11	10	9	8
POLY_VAL[7:0]							
7	6	5	4	3	2	1	0
-	-	-	-	-	-	MFIFOTRIG	G_ENA

- **G_ENA** [*read/write*]: RTU Global Enable

Global RTU enable bit. Overrides all port settings.
0: RTU is disabled. All packets are dropped.
1: RTU is enabled.

- **MFIFOTRIG** [*read/write*]: MFIFO Trigger
  write 1: triggers a flush of MFIFO into the hash table (blocks the RTU for a few cycles)
  write 0: no effect
  read 1: MFIFO is busy
  read 0: MFIFO is idle

- **POLY_VAL** [*read/write*]: Hash Poly
  Determines the polynomial used for hash computation. Currently available: 0x1021, 0x8005, 0x0589

- **RTU_VERSION** [*read-only*]: Version
  Information about the version of RTU gateware

Port Select Register

HW prefix: rtu_psr
HW address: 0x1
SW prefix: PSR
SW offset: 0x4

Selects the RTU port to be controlled through the PCR register

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
N_PORTS[7:0]							
7	6	5	4	3	2	1	0
PORT_SEL[7:0]							

- **PORT_SEL** [*read/write*]: Port Select
  Selected Port for PCR register modifications

- **N_PORTS** [*read-only*]: Number of ports
  Number of RTU ports compiled in.

Port Control Register

HW prefix: rtu_pcr
HW address: 0x2
SW prefix: PCR
SW offset: 0x8

Register controlling the mode of the RTU port selected in PSR register.

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
B_UNREC	PRIO_VAL[2:0]			FIX_PRIO	PASS_BPDU	PASS_ALL	LEARN_EN

- **LEARN_EN** [*read/write*]: Learning enable
  1: enables learning process on this port. Unrecognized requests will be put into UFIFO
  0: disables learning. Unrecognized requests will be either broadcast or dropped.

- **PASS_ALL** [*read/write*]: Pass all packets
  1: all packets are passed (depending on the rules in RT table).
  0: all packets are dropped on this port.

- **PASS_BPDU** [*read/write*]: Pass BPDUs
  1: BPDU packets (with dst MAC 01:80:c2:00:00:00) are passed according to RT rules. This setting is independed from PASS_ALL state.
  0: BPDU packets are passed according to RTU rules only if PASS_ALL is set

- **FIX_PRIO** [*read/write*]: Fix priority
  1: Port has fixed priority of value PRIO_VAL. It overrides the priority coming from the endpoint
  0: Use priority from the endpoint

- **PRIO_VAL** [*read/write*]: Priority value
  Fixed priority value for the port. Used instead the endpoint-assigned priority when FIX_PRIO = 1

- **B_UNREC** [*read/write*]: Unrecognized request behaviour
  Sets the port behaviour for all unrecognized requests:
  0: packet is dropped
  1: packet is broadcast

VLAN Table Register 1

HW prefix: rtu_vtr1
HW address: 0x3
SW prefix: VTR1
SW offset: 0xc

31	30	29	28	27	26	25	24
-	-	-	-	-	UPDATE	PRIO[2:1]	
23	22	21	20	19	18	17	16
PRIO[0:0]	PRIO_OVERRIDE	HAS_PRIO	DROP	FID[7:4]			
15	14	13	12	11	10	9	8
FID[3:0]				VID[11:8]			
7	6	5	4	3	2	1	0
VID[7:0]							

- **VID** [*read/write*]: VLAN ID
- **FID** [*read/write*]: Filtering Database ID
  Assigns the VID to a particular filtering database
- **DROP** [*read/write*]: Drop
  1: drop all packets belonging to this VLAN
- **HAS_PRIO** [*read/write*]: Has user-defined priority
  1: VLAN has user-defined priority
- **PRIO_OVERRIDE** [*read/write*]: Override endpoint-assigned priority
  1: always take the priority from the PRIO field, regardless of the priority value assigned at the endpoint.
- **PRIO** [*read/write*]: Priority value
- **UPDATE** [*write-only*]: Force VLAN table entry update
  write 1: flush VTR1 and VTR2 registers to VLAN table entry designated in VTR1.VID

VLAN Table Register 2

HW prefix: rtu_vtr2
HW address: 0x4
SW prefix: VTR2
SW offset: 0x10

31	30	29	28	27	26	25	24
PORT_MASK[31:24]							
23	22	21	20	19	18	17	16
PORT_MASK[23:16]							
15	14	13	12	11	10	9	8
PORT_MASK[15:8]							
7	6	5	4	3	2	1	0
PORT_MASK[7:0]							

- **PORT_MASK** [*read/write*]: Port Mask
  Each bit corresponds to one RTU port, setting bit to 1 means that the port belongs to the VLAN

Ext: Control Register

HW prefix: rtu_rx_ctr
HW address: 0x5
SW prefix: RX_CTR
SW offset: 0x14

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	LEARN_DST_ENA	UREC_FW_CPU_ENA	HP_FW_CPU_ENA
15	14	13	12	11	10	9	8
PRIO_MASK[7:0]							
7	6	5	4	3	2	1	0
-	AT_FMATCH_TOO_SLOW	MR_ENA	FF_MAC_PTP	FF_MAC_LL	FF_MAC_SINGLE	FF_MAC_RANGE	FF_MAC_BR

- **FF_MAC_BR** [*read/write*]: Fast Forward for Broadcast
  The feature is:
  0: Disabled,
  1: Enabled.

- **FF_MAC_RANGE** [*read/write*]: Fast Forward for MAC Range
  The feature is:
  0: Disabled,
  1: Enabled.

- **FF_MAC_SINGLE** [*read/write*]: Fast Forward for MAC Single Entries
  The feature is:
  0: Disabled,
  1: Enabled.

- **FF_MAC_LL** [*read/write*]: Fast Forward for Link-Limited (Reserved) MACs
  The feature is:
  0: Disabled,
  1: Enabled.

- **FF_MAC_PTP** [*read/write*]: Fast Forward for PTP frames (PTP over IEEE 802.3 /Ethernet)
  The feature is:
  0: Disabled,
  1: Enabled.

- **MR_ENA** [*read/write*]: Port Mirror Enable
  Enable port mirroring as defined by proper configurition
  0: Disable,
  1: Enable.

- **AT_FMATCH_TOO_SLOW** [*read/write*]: Drop/Forward on FullMatch Full
  In case that a new Frame arrives on Ingress when the previous is still handed (FullMatch process, or SWcore):
  0: Drop currently processed frame (default),
  1: Broadcast currently processed frame.

- **PRIO_MASK** [*read/write*]: HP Priorities Mask
  Mask which defines which priorities of the Fast Forward traffic are considered High Priority (used also by SWcore)

- **HP_FW_CPU_ENA** [*read/write*]: HP forward to CPU
  Enables/disables forwarding of recognized HP frames to CPU (Network InterFace) - disabling forwarding can prevent flooding of switch CPU with unnecessary traffic, allowing forwarding can be enabled to snoop on network traffic). It uses HW-set (generic) mask which indicates port number of CPU - can be verified by reading
  0: Disabled [default] - does not forward HP frames to CPU,
  1: Enabled - forwards HP frames to CPU.

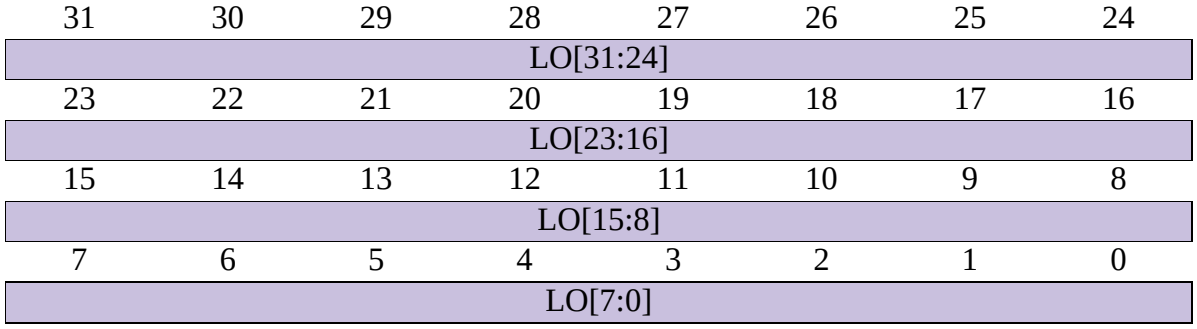
- **UREC_FW_CPU_ENA** [*read/write*]: Urecognized forward to CPU
  Allows to enable/disable forwarding of unrecognized frames (with unrecognized dstMAC) which are broadcast (when b_unrec enabled) CPU (Network InterFace) - disabled to prevent flooding of switch CPU with unnecessary traffic.It uses Link-Limited Frames Fast Forward Mask to know to which port CPU is connected.
  0: Disabled [default] - does not forward unrecognized braodcast (b_unrec) frames to CPU,
  1: Enabled - forwards unrecognized braodcast (b_unrec) frames to CPU.

- **LEARN_DST_ENA** [*read/write*]: Learn Destination MAC enable
  Allows to enable/disable learning based on Destination MAC address (works only if learning is enabled on a port, i.e. LEARN_EN=1) .
  0: Disabled [default] - frames with unrecognizd destinatin MAC do not trigger writes to UFIFO, i.e. ureq in software (unrecognized request),
  1: Enabled - frames with unrecognizd destinatin MAC trigger writes to UFIFO, i.e. ureq in software.

Ext: Fast Forward MAC[31:0]

HW prefix: rtu_rx_ff_mac_r0
HW address: 0x6
SW prefix: RX_FF_MAC_R0
SW offset: 0x18

Validated on write to RX_FF_MAC_R1



- **LO** [*read/write*]: Fast Forward MAC

Ext: Fast Forward MAC and control

HW prefix: rtu_rx_ff_mac_r1
HW address: 0x7
SW prefix: RX_FF_MAC_R1
SW offset: 0x1c

Double purpose on
write: low bigs: MAC bits [47:32]; high bits: MAC ID, single/range, valid,
read: low bits: max number of single entries (MAX ID), high bits: max number of range entries (MAX ID).

31	30	29	28	27	26	25	24
-	-	-	-	-	-	VALID	TYPE
23	22	21	20	19	18	17	16
ID[7:0]							
15	14	13	12	11	10	9	8
HI_ID[15:8]							
7	6	5	4	3	2	1	0
HI_ID[7:0]							

- **HI_ID** [*read/write*]: Fast Forward MAC
- **ID** [*read/write*]: Fast Forward entry index (single/range)
  Depending on the Single/Range bit:
  0: Index of the Fast Forward MAC for single Fast Forward MAC
  1: Index of the Fast Forward MAC for the Fast Forward MAC range (low bit 0 indicates lower range, low bit 1 indicates upper range, inclusive)
- **TYPE** [*read/write*]: Fast Forward MAC single/range entry
  Indicates what kind of entry is written
  0: Single Fast Forward MAC,
  1: Range Fast Forward MAC (low bit of MAC ID equal to 0 indicates lower range, low bit of MAX ID equal to 1 indicates upper range, inclusive)
- **VALID** [*read/write*]: Fast Forward MAC valid
  The value of the bit (only validated entries are used):
  0: Invalidates the entry,
  1: Validates the entry.

Ext: CPU port mask

HW prefix: rtu_cpu_port
HW address: 0x8
SW prefix: CPU_PORT
SW offset: 0x20

Link-Limited Frames Fast Forward Mask

31	30	29	28	27	26	25	24
MASK[31:24]							
23	22	21	20	19	18	17	16
MASK[23:16]							
15	14	13	12	11	10	9	8
MASK[15:8]							
7	6	5	4	3	2	1	0
MASK[7:0]							

- **MASK** [*read-only*]: CPU/LL Mask
  It is only for debugging purposes. The ID of the CPU port is set in HW using generic which produces the CPU/LL Mask.
  It is used for
  * forwarding of the Link-Limited traffic to CPU (if enabled by config)
  * enabling/disabling forwarding of HP traffic to CPU (HP_FW_CPU_ENA)
  * enabling/disabling forwarding of unrecognized broadcast to CPU (UREC_FW_CPU_ENA).

Ext: Mirroring Ports Control Reg 0

HW prefix: rtu_rx_mp_r0
HW address: 0x9
SW prefix: RX_MP_R0
SW offset: 0x24

select for the mask written using RX_MP_R1

31	30	29	28	27	26	25	24
MASK_ID[15:8]							
23	22	21	20	19	18	17	16
MASK_ID[7:0]							
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	RX_TX	DST_SRC

- **DST_SRC** [*read/write*]: DST/SRC Mirror port
  Defines whether destination or source mask is written to RX_MP_R1:
  0: Mirror port(s) - destination of the mirrored traffic,
  1: Mirrored port(s) - source of the mirrored traffic

- **RX_TX** [*read/write*]: RX/TX mirror port source
  Defines whether transmission or reception source mask is written to RX_MP_R1 (used only when DST_SRC bit is 1):
  0: Reception traffic mirror source,
  1: Transmission traffic mirror source.

- **MASK_ID** [*read/write*]: Mirrored Port MASK Index
  Index of the mirrored configuration (to be considered for implementation in future, curreantly only single config available)

Ext: Mirroring Ports Control Reg 1

HW prefix: rtu_rx_mp_r1
HW address: 0xa
SW prefix: RX_MP_R1
SW offset: 0x28

31	30	29	28	27	26	25	24
MASK[31:24]							
23	22	21	20	19	18	17	16
MASK[23:16]							
15	14	13	12	11	10	9	8
MASK[15:8]							
7	6	5	4	3	2	1	0
MASK[7:0]							

- **MASK** [*read/write*]: Mirror Port MASK
  MASK to define mirroring, depending on two lowest bits of select reg:
  00: port(s) which output mirrored traffic from the mirrored port(s)- destination of the mirrored traffic (egress only, disabled for ingress traffic and traffic other then from mirrored, source, port(s))
  10: port(s) whose ingress traffic is mirrored (reception source) - all the traffic received on this port(s) is forwarded to the mirror port(s)
  11: port(s) whose egress traffic is mirrored (transmision source) - all the traffic forwareded to this port(s) is also forwarded to the mirror port(s).

Interrupt disable register

HW prefix: rtu_eic_idr
HW address: 0x10
SW prefix: EIC_IDR
SW offset: 0x40

Writing 1 disables handling of the interrupt associated with corresponding bit. Writin 0 has no effect.

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
-	-	-	-	-	-	-	NEMPTY

- **NEMPTY** [*write-only*]: UFIFO Not Empty IRQ
  write 1: disable interrupt 'UFIFO Not Empty IRQ'
  write 0: no effect

Interrupt enable register

HW prefix: rtu_eic_ier
HW address: 0x11
SW prefix: EIC_IER
SW offset: 0x44

Writing 1 enables handling of the interrupt associated with corresponding bit. Writin 0 has no effect.

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
-	-	-	-	-	-	-	NEMPTY

- **NEMPTY** [*write-only*]: UFIFO Not Empty IRQ
  write 1: enable interrupt 'UFIFO Not Empty IRQ'
  write 0: no effect

Interrupt mask register

HW prefix: rtu_eic_imr
HW address: 0x12
SW prefix: EIC_IMR
SW offset: 0x48

Shows which interrupts are enabled. 1 means that the interrupt associated with the bitfield is enabled

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
-	-	-	-	-	-	-	NEMPTY

- **NEMPTY** [*read-only*]: UFIFO Not Empty IRQ
  read 1: interrupt 'UFIFO Not Empty IRQ' is enabled

read 0: interrupt 'UFIFO Not Empty IRQ' is disabled

Interrupt status register

HW prefix: rtu_eic_isr
HW address: 0x13
SW prefix: EIC_ISR
SW offset: 0x4c

Each bit represents the state of corresponding interrupt. 1 means the interrupt is pending. Writing 1 to a bit clears the corresponding interrupt. Writing 0 has no effect.

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
-	-	-	-	-	-	-	NEMPTY

- **NEMPTY** [*read/write*]: UFIFO Not Empty IRQ
  read 1: interrupt 'UFIFO Not Empty IRQ' is pending
  read 0: interrupt not pending
  write 1: clear interrupt 'UFIFO Not Empty IRQ'
  write 0: no effect

FIFO 'UFIFO' data output register 0

HW prefix: rtu_ufifo_r0
HW address: 0x14
SW prefix: UFIFO_R0
SW offset: 0x50

31	30	29	28	27	26	25	24
DMAC_LO[31:24]							
23	22	21	20	19	18	17	16
DMAC_LO[23:16]							
15	14	13	12	11	10	9	8
DMAC_LO[15:8]							
7	6	5	4	3	2	1	0
DMAC_LO[7:0]							

- **DMAC_LO** [*read-only*]: Destination MAC address least-significant part
  Bits [31:0] of packet destination MAC address

FIFO 'UFIFO' data output register 1

HW prefix: rtu_ufifo_r1
HW address: 0x15
SW prefix: UFIFO_R1
SW offset: 0x54

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
DMAC_HI[15:8]							
7	6	5	4	3	2	1	0
DMAC_HI[7:0]							

- **DMAC_HI** [*read-only*]: Destination MAC address most-significant part
  Bits [47:32] of packet destination MAC address

FIFO 'UFIFO' data output register 2

HW prefix: rtu_ufifo_r2
HW address: 0x16
SW prefix: UFIFO_R2
SW offset: 0x58

31	30	29	28	27	26	25	24
SMAC_LO[31:24]							
23	22	21	20	19	18	17	16
SMAC_LO[23:16]							
15	14	13	12	11	10	9	8
SMAC_LO[15:8]							
7	6	5	4	3	2	1	0
SMAC_LO[7:0]							

- **SMAC_LO** [*read-only*]: Source MAC address least-significant part
  Bits [31:0] of packet source MAC address

FIFO 'UFIFO' data output register 3

HW prefix: rtu_ufifo_r3
HW address: 0x17
SW prefix: UFIFO_R3
SW offset: 0x5c

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
SMAC_HI[15:8]							
7	6	5	4	3	2	1	0
SMAC_HI[7:0]							

- **SMAC_HI** [*read-only*]: Source MAC address most-significant part
  Bits [47:32] of packet source MAC address

FIFO 'UFIFO' data output register 4

HW prefix: rtu_ufifo_r4
HW address: 0x18
SW prefix: UFIFO_R4
SW offset: 0x60

31	30	29	28	27	26	25	24
-	-	-	-	-	-	HAS_PRIO	HAS_VID
23	22	21	20	19	18	17	16
PID[7:0]							
15	14	13	12	11	10	9	8
-	PRIO[2:0]			VID[11:8]			
7	6	5	4	3	2	1	0
VID[7:0]							

- **VID** [*read-only*]: VLAN Identifier
  VLAN ID of the packet (from the endpoint)

- **PRIO** [*read-only*]: Priority
  Priority value (from the endpoint)

- **PID** [*read-only*]: Port ID
  Identifier of RTU port to which came the request.

- **HAS_VID** [*read-only*]: VID valid
  1: VID value is valid
  0: packet had no VLAN ID

- **HAS_PRIO** [*read-only*]: PRIO valid
  1: PRIO value is valid
  0: packet had no priority assigned

FIFO 'UFIFO' control/status register

HW prefix: rtu_ufifo_csr
HW address: 0x19
SW prefix: UFIFO_CSR
SW offset: 0x64

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	EMPTY	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	USEDW[6:0]						

- **EMPTY** [*read-only*]: FIFO empty flag
  1: FIFO 'UFIFO' is empty
  0: FIFO is not empty

- **USEDW** [*read-only*]: FIFO counter
  Number of data records currently being stored in FIFO 'UFIFO'

FIFO 'MFIFO' data input register 0

HW prefix: rtu_mfifo_r0
HW address: 0x1a
SW prefix: MFIFO_R0
SW offset: 0x68

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
-	-	-	-	-	-	-	AD_SEL

- **AD_SEL** [*write-only*]: Address/data select
  1: AD_VAL contains new memory address
  0: AD_VAL contains data word to be written at current memory address. Then, the address is incremented

FIFO 'MFIFO' data input register 1

HW prefix: rtu_mfifo_r1
HW address: 0x1b
SW prefix: MFIFO_R1
SW offset: 0x6c

31	30	29	28	27	26	25	24
AD_VAL[31:24]							
23	22	21	20	19	18	17	16
AD_VAL[23:16]							
15	14	13	12	11	10	9	8
AD_VAL[15:8]							
7	6	5	4	3	2	1	0
AD_VAL[7:0]							

- **AD_VAL** [*write-only*]: Address/data value
  Value of new memory address (when AD_SEL = 1) or data word to be written (when AD_SEL = 0)

FIFO 'MFIFO' control/status register

HW prefix: rtu_mfifo_csr
HW address: 0x1c
SW prefix: MFIFO_CSR
SW offset: 0x70

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	EMPTY	FULL
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	USEDW[5:0]					

- **FULL** [*read-only*]: FIFO full flag
  1: FIFO 'MFIFO' is full
  0: FIFO is not full

- **EMPTY** [*read-only*]: FIFO empty flag
  1: FIFO 'MFIFO' is empty
  0: FIFO is not empty

- **USEDW** [*read-only*]: FIFO counter
  Number of data records currently being stored in FIFO 'MFIFO'

Aging bitmap for main hashtable

HW prefix:	rtu_aram
HW address:	0x100
C prefix:	ARAM
C offset:	0x400
Size:	256 32-bit words
Data width:	32
Access (bus):	read/write
Access (device):	read/write
Mirrored:	no
Byte-addressable:	no
Peripheral port:	bus-synchronous

Each bit in this memory reflects the state of corresponding entry in main hashtable:
0: entry wasn't matched
1: entry was matched at least once.
CPU reads this bitmap and subsequently clears it every few seconds to update the aging counters.

4.13.3 Interrupts

UFIFO Not Empty IRQ

HW prefix:	rtu_nempty
C prefix:	NEMPTY
Trigger:	low level

Interrupt active when there are some requests in UFIFO.