

## PPS Handler Module Description

	Organisatie / Organization	Datum / Date
<b>Auteur(s) / Author(s):</b>  Eric Kooistra	ASTRON	
<b>Controle / Checked:</b>  Gijs Schoonderbeek	ASTRON	
<b>Goedkeuring / Approval:</b>  Andre Gunst	ASTRON	
<b>Autorisatie / Authorisation:</b>  <b>Handtekening / Signature</b> Andre Gunst	ASTRON	

© ASTRON 2013  
All rights are reserved. Reproduction in whole or in part is prohibited without written consent of the copyright owner.

UniBoard

**DESP**

Doc.nr.: ASTRON-RP-1374  
Rev.: 0.1  
Date:  
Class.: Public

## Distribution list:

---

Group:	Others:
Andre Gunst (AG, ASTRON) Gijs Schoonderbeek (GS, ASTRON) Eric Kooistra (EK, ASTRON) Daniel van der Schuur (DS, ASTRON) Harm-Jan Pepping (HJP, ASTRON)	

## Document history:

---

Revision	Date	Author	Modification / Change
0.1	2013-01-29	Eric Kooistra	Draft.

## Table of contents:

---

1	Introduction.....	4
2	Software interface .....	4
3	Hardware interface .....	5
4	Design .....	6
5	Implementation .....	7
6	Verification.....	7

## Terminology:

---

DDIO	Double Data rate IO
DDR	Double Data Rate
DP	Data Path
HDL	Hardware Description Language
IO	Input Output
IOE	IO Element
LSBit	Least Significant bit
MM	Memory Mapped
MSBit	Most Significant bit
PPS	Pulse Per Second
PPSH	PPS Handler
SI	Std_logic (1 bit)
Slv	Std_logic_vector (bit vector)
ST	Streaming
TB	Test Bench
UNB	UniBoard
Wi	Word Index

## References:

---

1. UNB = [https://svn.astron.nl/UniBoard\\_FP7/UniBoard/trunk/](https://svn.astron.nl/UniBoard_FP7/UniBoard/trunk/)

## 1 Introduction

This document describes the PPS Handler firmware module that contains VHDL components for capturing an external periodic time pulse into the system clock domain of the FPGA. Typically the periodic pulse is a pulse per second (PPS) that is locked to the system clock.

The PPS Handler (PPSH) module is located at \$UNB/Firmware/modules/ppsh [1].

## 2 Software interface

The PPSH can be controlled and monitored via the memory-mapped (MM) interface. The register map is shown Table 1 and the register fields are described in Table 2.

31	24	23	16	15	8	7	0	wi	
----- ----- ----- -----									
toggle[31], stable[30]				xxx	capture_cnt = [n:0]				0
edge[31],				xxx	expected_cnt = [n:0]				1
----- ----- ----- -----									

**Table 1: PPSH MM register map**

Register field	Width	R/W	Description
Toggle	1	R	Toggles on each PPS rising edge.
Stable	1	R	When '1' then the PPS <i>capture_cnt</i> was equal to <i>expected_cnt</i> for every PPS interval since the last read access to wi = 0. Else at least one PPS interval had a different <i>capture_cnt</i> value.
Edge	1	W	When '0' then clock in PPS with the rising edge of the system clock, else use the falling edge of the system clock.
Capture_cnt	30	R	Detected number of system clock cycles between two PPS.
Expected_cnt	31	W	Expected number of system clock cycles between two PPS.

**Table 2: PPSH MM register fields**

The actual width of the *capture\_cnt* and *expected\_cnt* is  $\text{ceil\_log2}(g\_st\_clk\_freq)$ , so sufficient to fit the largest supported system clock frequency value in Hz.

The Software/python/peripherals/pi\_ppsh.py Python peripheral script provides methods for accessing the PPSH registers and the util\_pphs.py provides usage examples.

### 3 Hardware interface

The mms\_ppsh interface parameters and ports are given in respectively Table 3 and Table 4.

Generic	Type	Description
g_cross_clock_domain	boolean	Use false when <i>mm_clk</i> and <i>st_clk</i> are the same, else use true to cross the clock domain
g_st_clk_freq	natural	System clock <i>st_clk</i> frequency in Hz

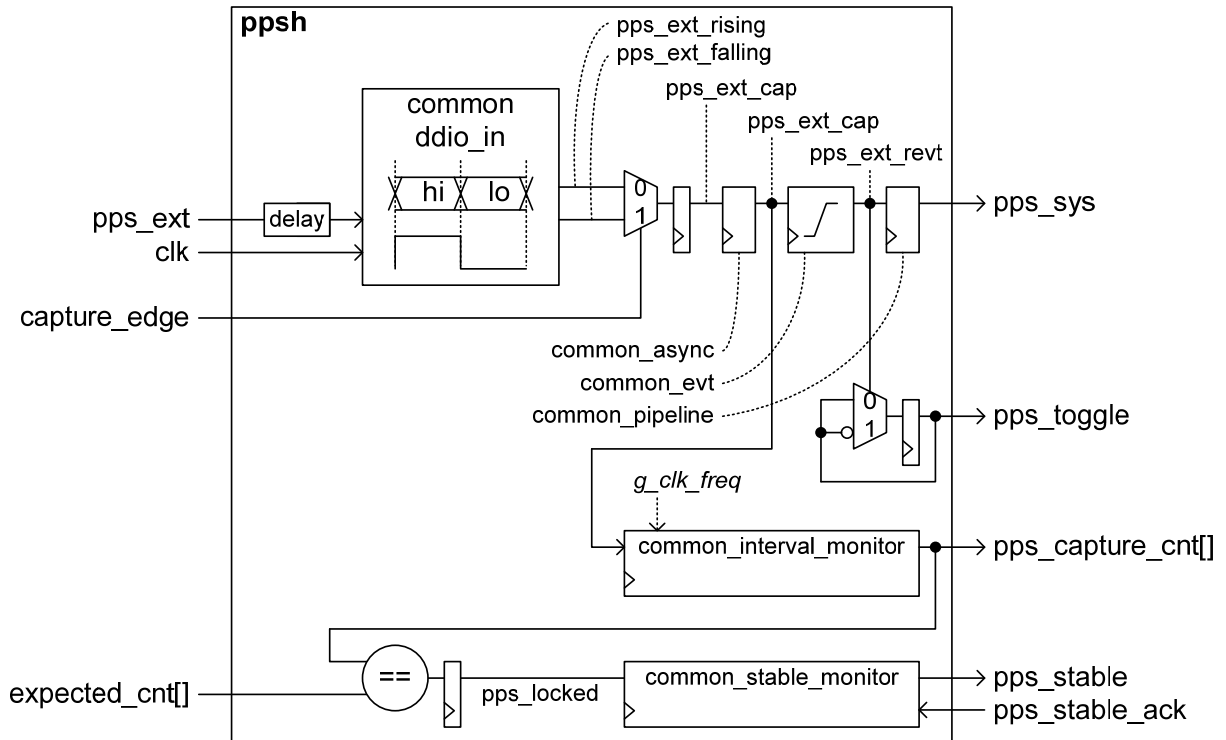
**Table 3: mms\_ppsh parameters**

Signal	IO	Type	Description
<b>MM clock domain</b>			
mm_rst	IN	std_logic	Reset in mm_clk domain
mm_clk	IN	std_logic	MM clock
reg_mosi	IN	t_mem_mosi	MM interface for ppsh
reg_miso	OUT	t_mem_miso	
pin_pps	OUT	slv	PIO support for backwards compatibility with pin_pps on ctrl_unb_common. The pin_pps did not support <i>stable</i> yet and is defined as: <i>toggle</i> & '0' & RESIZE_UVEC( <i>capture_cnt</i> , 30)
<b>ST clock domain</b>			
pps_ext	IN	std_logic	External periodic sync pulse with unknown but constant rising-edge phase to <i>st_clk</i> . The pulse width must be $\geq 1$ <i>st_clk</i> cycle. The falling edge is not used for the timing.
st_rst	IN	std_logic	Reset in system clock <i>st_clk</i> domain
st_clk	IN	std_logic	System clock domain clock
pps_sys	OUT	sl	The <i>pps_ext</i> pulse detected in the in <i>st_clk</i> domain The <i>pps_sys</i> pulse width is 1 <i>st_clk</i> cycle.

**Table 4: mms\_ppsh IO**

## 4 Design

Figure 1 shows the block diagram of the PPSH that is used inside mms\_ppsh.



**Figure 1: Block diagram of ppsvh.vhd**

The *pps\_ext* signal is assumed to be locked to the system clock *clk* and arrives with unknown but stable phase offset to the system clock. The PPSH module can be set to capture the PPS pulse using the rising edge or the falling edge of the system clock via the *capture\_edge* control input. This allows choosing the maximum time offset (i.e. eye-opening) between the rising edge of the PPS and the edge of the clock.

The *common\_interval\_monitor* in the PPSH module counts the number of system clock pulses that occur between every two external pulses

When the external pulse period and phase relation with the system clock are stable, then the number of clock pulses per period remains fixed. The *common\_stable\_monitor* in the PPSH module reports stable PPS for as long as the *capture\_cnt* equals the *expected\_cnt*. And not stable if one or more *capture\_cnt* were not expected. The stable monitor interval restarts after each *pps\_stable\_ack* pulse.

## 5 Implementation

The ppsh uses a `common_ddio_in` component to clock in the `pps_ext` via a double data rate (DDR) component that is located near the IO pin in the IO Element (IOE). The DDR component also eases the selection of which edge of the `clk` is used via `capture_edge`.

The input delay element in IOE shown in Figure 1 is not instantiated in the ppsh VHDL. Typically using either the rising edge or falling edge of the system clock provides sufficient margin for reliable capture of the external PPS, so therefore the input delays are set to 0. The input delay can be set via an input delay constraint for the PPS pin in the synthesis script. For example as done in `$UNB\designs\unb_common\src\tcl\COMMON_NODE_general_pins.tcl`:

```
set_instance_assignment -name D2_DELAY 0 -to PPS
set_instance_assignment -name D3_DELAY 0 -to PPS
set_instance_assignment -name D1_DELAY 0 -to PPS
```

## 6 Verification

The `tb_ppsh.vhd` and `tb_mms_ppsh.vhd` are self-check test benches that verify the correct behavior of the PPSH module.