Datasheet

CC-SYSTICK-APB Documentation

1.2

Scope

This document describes the CC-SYSTICK-APB IP core. Module features and configuration registers are described. The document contains integration guide that covers synthesis options and instantiation example for easy implementation in customer's environment.

Contents

1. Systick Module

1.1 Functionality

- Incrementing mode,
- overflow interrupt (OVF).

1.2 Overview

Systick module can be used as a interrupt source for tasks that need to be executed regularly. For example allows an OS to carry out context switching to support multiple tasking.

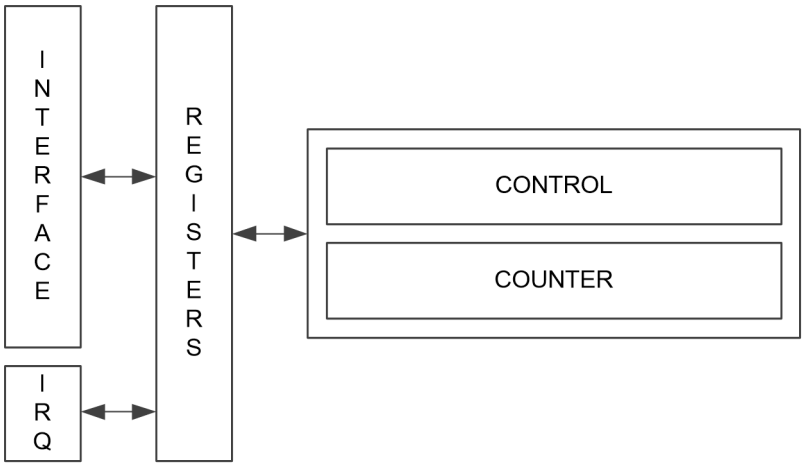


Figure 1.1. Systick block diagram.

Figure 1.1 presents the block diagram of the Systick module. It is composed of configuration registers and main counter register. Prescaler register counts peripheral clock cycles (PCLK) and generates events for main COUNT register (1.4.3). The Systick module has configurable period and works in incrementing mode only.

1.3 Interrupts

The Systick module has one interrupt source.

1.3.1 Systick Interrupt

The Systick Interrupt is signalized by IF flag in IRQF register (1.4.6). Systick interrupt occurs when COUNT (1.4.3) reaches PER value (1.4.4). The interrupt is cleared after reading IRQF register or by writing one in IF bit (1.4.6).

1.4 Configuration Registers

1.4.1 Registers List

The core is controlled through registers mapped into memory address space. Not implemented locations are read as zeros.

Address Offset	Register	Name
0x00	CTRL	Control Register
0x04	COUNT	Count Register
0x08	PER	Period Register
0x0C	PRES	Prescaler Register
0x10	IRQF	Interrupt Flags Register
0x14	IRQMAP	Interrupt Mapping Register

1.4.2 Control Register

Address: 0x00

31	30	...	...	...	...	9	8
DBG_STOP		...	...	...	...		
R/W	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
						IE	EN
R	R	R	R	R	R	R/W	R/W
0	0	0	0	0	0	0	0

EN *Systick Enable*

0 Systick is disabled. Clock to the module is stopped.

1 Systick is enabled. Clock to the module is supplied.

IE *Systick Interrupt Enable*

0 Interrupt disabled.

1 Interrupt enabled.

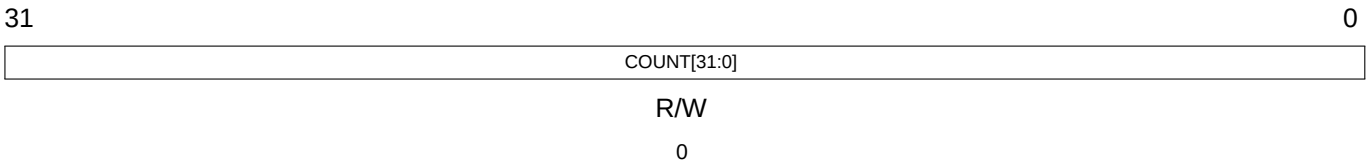
DBG_STOP *Systick stop in debug mode*

0 Systick is counting in debug mode.

1 Systick is not counting in debug mode.

1.4.3 Count Register

Address: 0x04

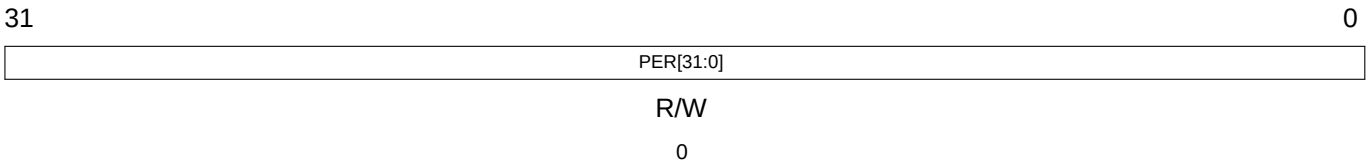


COUNT[31:0] *Current Count*

Current value of Systick counter.

1.4.4 Period Register

Address: 0x08



PER[31:0] *Systick Period*

Systick counter period. Reaching PER value sets interrupt flag (IF, 1.4.6) and clears COUNT register (1.4.3).

1.4.5 Prescaler Register

Address: 0x0C

31	0
PRES[31:0]	
R/W	
0	

PRES[31:0] *Systick Prescaler*

Systick module prescaler. COUNT register (1.4.3) is incremented every each $PRES + 1$ PCLK clock cycles.

1.4.6 Interrupt Flags Register

Address: 0x10

31	30	...	...	...	...	9	8
		...	...	...	...		
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
							IF
R	R	R	R	R	R	R	R/W
0	0	0	0	0	0	0	0

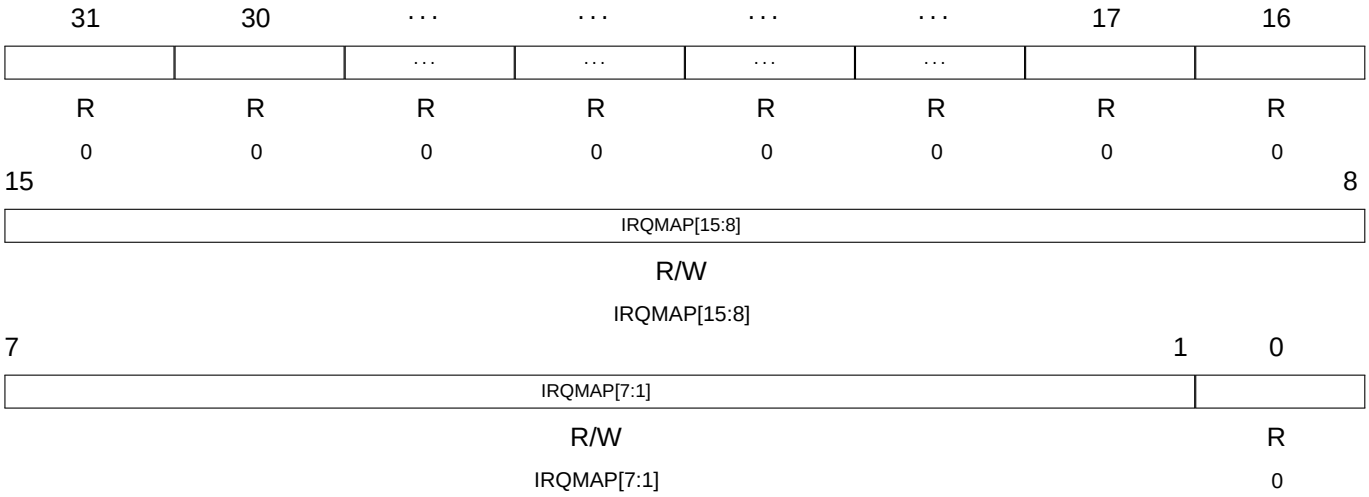
IF *Interrupt Flag*

1 Systick reached PER value.

Bit is cleared after register readout or by writing one to this position.

1.4.7 Interrupt Mapping Register

Address: 0x14



IRQMAP[15:1] *Interrupt Mapping*

Each set bit represents the interrupt number that will be passed to interrupt controller. It is allowed to set more than one bit.

1.5 Implementation

1.5.1 Design Structure

The synthesible RTL IP core part (*SYSTICK/rtl* folder) utilizes Verilog 2005 HDL. The testbench part (*SYSTICK/tb* folder) uses SystemVerilog language.

```
SYSTICK
├── beh
├── rtl
│   ├── APB_SYSTICK.v
│   ├── SYSTICK_config.v
│   ├── SYSTICK_defines.v
│   └── SYSTICK.v
├── tb
│   ├── APB
│   │   ├── tb_APB_SYSTICK_init.v
│   │   └── tb_APB_SYSTICK_reg_access_tasks.v
│   ├── common
│   │   ├── tb_SYSTICK_other_tasks.v
│   │   ├── tb_SYSTICK_read_config_tasks.v
│   │   ├── tb_SYSTICK_write_config_tasks.v
│   │   └── timescale.v
│   ├── run
│   │   └── ncvlog_apb_systick.sh
│   ├── tests
│   │   ├── tb_interrupt_MAPPING_test.sv
│   │   └── tb_SYSTICK_INCREMENT_test.sv
│   └── tb_APB_SYSTICK.sv
└── compile.list
```

1.5.2 Simulation Flow

The IP Core is provided with self-checking testbench to verify the correct operation of the IP prior to use in a design. To run the simulation using Cadence® Incisive® Enterprise Simulator run *ncvlog_apb_systick.sh* script located in *SYSTICK/tb/run* folder. The simulation should end with reporting no errors.

1.5.3 Clock and Reset

The CC-SYSTICK-APB utilizes a fully synchronous design with one positive edge clocking domain and negative asynchronous reset assertion. External reset synchronizer has to be used to ensure synchronous reset deassertion.

1.5.4 Constraints

In most cases only module output ports are registered. Therefore, it is a good practice to reserve the entire clock cycle for module inputs combinational logic and set minimal input delay (*set_input_delay* command). Registered outputs leave the entire clock cycle for external logic (*set_output_delay* command).

1.5.5 Configuration Options

The table below shows the generic parameters of the core.

Generic name	Description	Range	Default
systick_width	Configure width of systick count register	1:32	32
prescaler_width	Configure width of systick prescaler register	1:32	8
default_interrupt_MAPPING	Reset value of interrupt_MAPPING register	0:32767	0

1.5.6 Signals Description

Signal name	Description	I/O	Active	Type
PCLK	Synchronous clock	I	rising	clock
PRESETn	Asynchronous reset	I	low	reset
PSEL	APB peripheral select	I	high	comb.
PENABLE	APB bus enable	I	high	comb.
PADDR[4:2]	APB bus address	I	data	comb.
PWRITE	APB bus write	I	high	comb.
PWDATA[31:0]	APB bus write data	I	data	comb.
PREADY	APB bus ready	O	high	const.
PRDATA[31:0]	APB bus read data	O	data	reg.
systick_interrupt	Systick interrupt	O	high	reg.
interrupt_MAPPING[15:1]	Interrupt mapping vector	O	data	reg.
clock_request	Clock request signal	O	high	reg.
debug_mode	Debug mode indicator (1 - core is halted)	I	high	comb.

1.5.7 Instantiation

```
icg
icg_stick_u (
    .E(stick_PSEL|stick_clock_request),
    .clk(PCLK),
    .gclk(stick_clk),
    .scan_enable(scan_enable));

APB_SYSTICK #(
    .systick_width(CFG_SYSTICK_WIDTH),
    .prescaler_width(CFG_SYSTICK_PRES),
    .default_interrupt_MAPPING(CFG_DEF_INT_MAPPING)
APB_SYSTICK_u (
    .PCLK(stick_clk),
    .PRESETn(rst),
    .PSEL(stick_PSEL),
    .PENABLE(PENABLE),
    .PADDR(PADDR[4:2]),
    .PWRITE(PWRITE),
    .PWDATA(PWDATA),
    .PREADY(stick_PREADY),
    .PRDATA(stick_PRDATA),
    .systick_interrupt(stick_interrupt),
    .interrupt_MAPPING(stick_interrupt_MAPPING),
    .clock_request(stick_clock_request),
    .debug_mode(debug_amba_stall_req));

assign  stick_irq_mapped  = stick_interrupt_MAPPING & {15{stick_interrupt}};
```


1.6 Revision History

Doc. Rev.	Date	Comments
1.2	11-2018	Editorial corrections in 1.5.7 Instantiation section.
1.1	09-2018	Updated module instantiation in 1.5.7 Instantiation section.
1.0	10-2017	First Issue.

ChipCraft Sp. z o.o. Dobrzańskiego 3 lok. BS073, 20-262 Lublin, POLAND www.chipcraft-ic.com

©2018 ChipCraft Sp. z o.o. CC-SYSTICK-APB-Doc_112018.

ChipCraft®, ChipCraft logo and combination of thereof are registered trademarks or trademarks of ChipCraft Sp. z o.o. All other names are the property of their respective owners.

Disclaimer: ChipCraft makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. ChipCraft does not make any commitment to update the information contained herein.