Datasheet

CC-TIMER-APB Documentation

1.2

Scope

This document describes the CC-TIMER-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. Timer Module

1.1 Functionality

- Incrementing and decrementing modes,
- double-buffered period and prescaler,
- double-buffered Capture/Compare channels,
- interrupt generation:
  - overflow interrupt (OVF),
  - capture error interrupt (ERR),
  - Capture/Compare events,
- three capture modes:
  - standard,
  - period capture,
  - pulse width capture,
- two PWM modes:
  - single Slope PWM,
  - dual Slope PWM.

1.2 Overview

Configurable Timer module can be used to measure time period, waveform generation with configurable duty cycle or for monitoring and detecting external events and their duration.

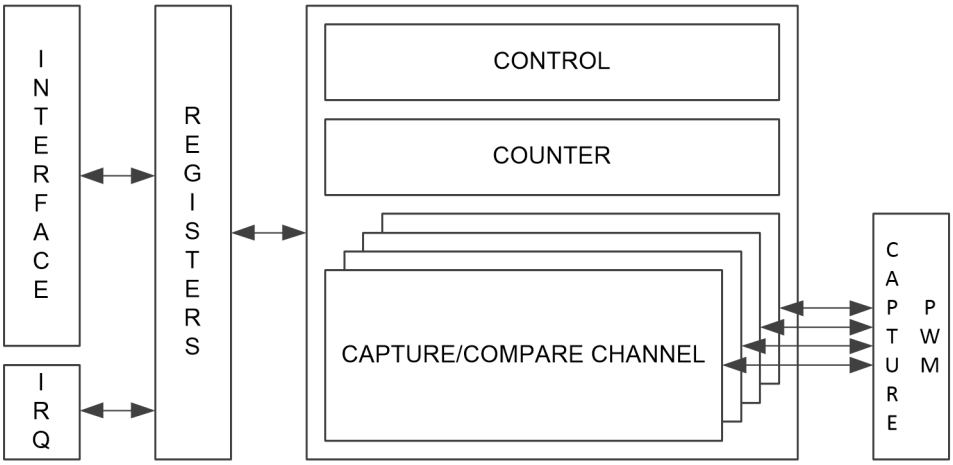


Figure 1.1. Timer block diagram.

Figure 1.1 presents the block diagram of the Timer module. It is composed of configuration registers, main counter register and a set of Capture/Compare channels. Prescaler register counts peripheral clock cycles (PCLK) and generates events for main COUNT register. The Timer module has configurable period and can work in incrementing or decrementing mode. Capture/Compare channels, along with the main counter, allow waveform generation and measuring the time duration of external events. Capture/Compare channels can work only in one mode at a time. All Capture/Compare channels work in Capture or Compare mode or they are disactivated.

1.2.1 Definitions

Name	Description
BOTTOM	Timer COUNT register equals 0h.
MAX	The maximum value that can be stored in COUNT register (0xFFFFFFFF in 32-bit Timer).
PERIOD/TOP	Timer period stored in period register (PER).
UPDATE	Update of internal registers when timer reaches BOTTOM or PERIOD/TOP value (depending on incrementing or decrementing mode).
CAPTURE	Configured rising or falling edge of Capture input.

1.3 Double Buffering

Period register (PER, 1.10.3), Capture/Compare data register (CCDATA, 1.10.14) and prescaler register (PRES, 1.10.5) are double-buffered. Each of buffer registers (PERBUF (1.10.4), CCDATABUF (1.10.15) and PRESBUF (1.10.6)) has a dedicated buffer valid flag (CC*n*BV) located in the buffer valid flags register (BUFVD) that indicates if stored data is actual. Buffer valid flags of period buffer register (PERBUF), Capture/Compare data buffer register (CCDATABUF) and prescaler buffer register (PRESBUF) are set during register write event. The UPDATE event moves buffered data to the main registers and clears corresponding flags. Figure 1.2 shows double buffering mechanism on Capture/Compare buffer data register example.

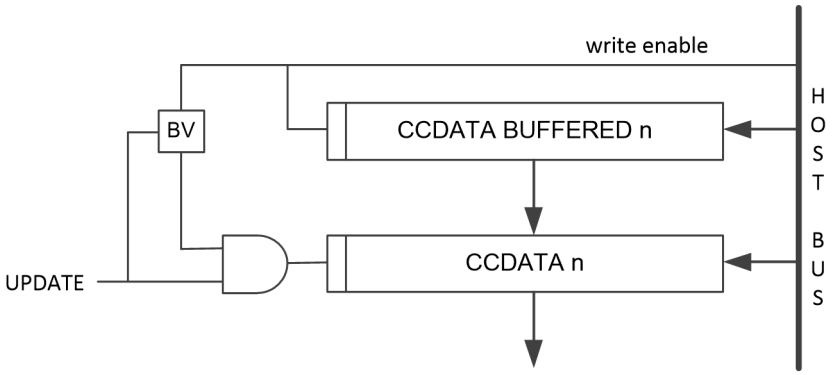


Figure 1.2. Double buffering of PRES, PER and CCDATA registers.

When Timer works in Capture mode, the CC*n*BV flag is set on CAPTURE event (Figure 1.3). In Capture mode registers CCDATA and CCDATABUF forms a FIFO. When CCDATA register is empty, COUNT sample is stored in CCDATA register and CC*n*IF flag is set in interrupt flags register (IRQF, 1.10.9). If CCDATA stores unread data, COUNT sample is stored in CCDATABUF register and corresponding CC*n*BV flag is set. Readout of CCDATA register moves CCDATABUF content to CCDATA register and the corresponding CC*n*BV flag is cleared. When new COUNT sample arrives when both CC*n*IF and CC*n*BV flags are set error interrupt is signaled (ERRIF).

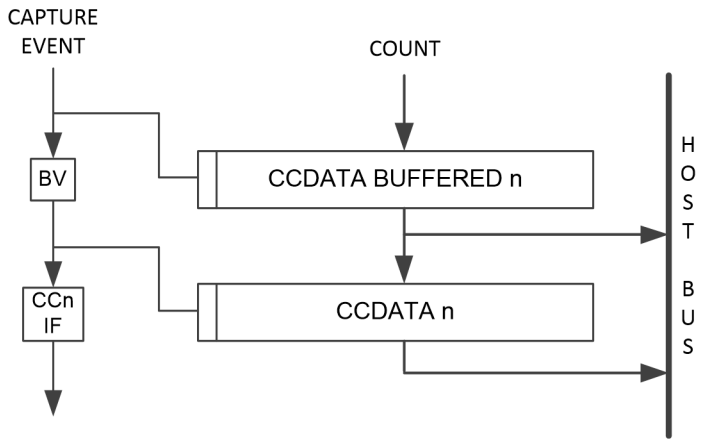


Figure 1.3. Double buffering of CCDATA register.

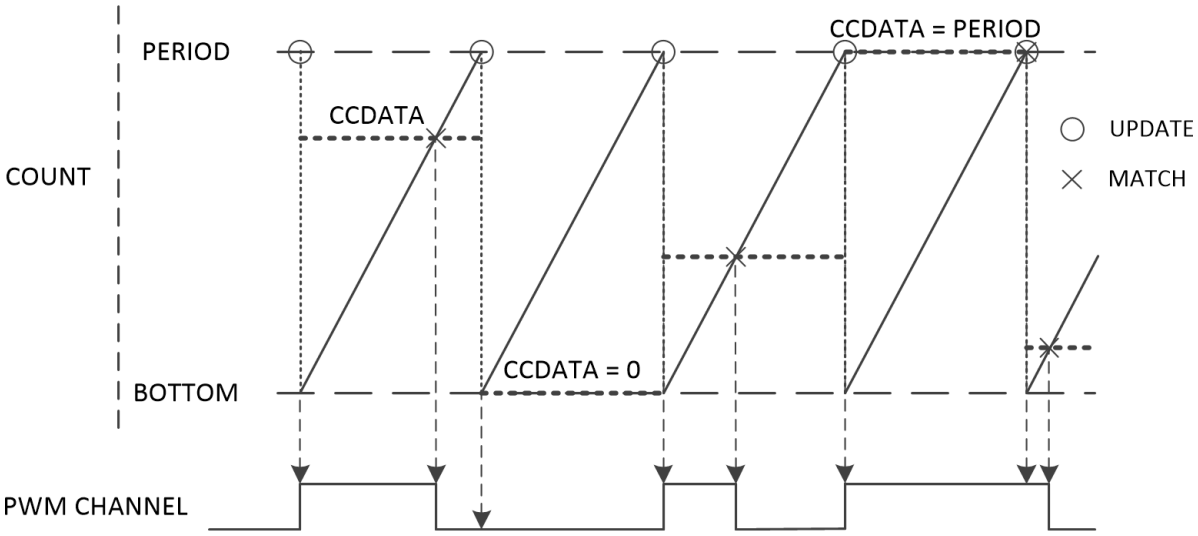


Figure 1.4. Change of the period value without buffering.

The Timer module allows to access buffered registers and to override buffering mechanism. Figure 1.4 shows change of the period value without buffering. The new value is applied immediately.

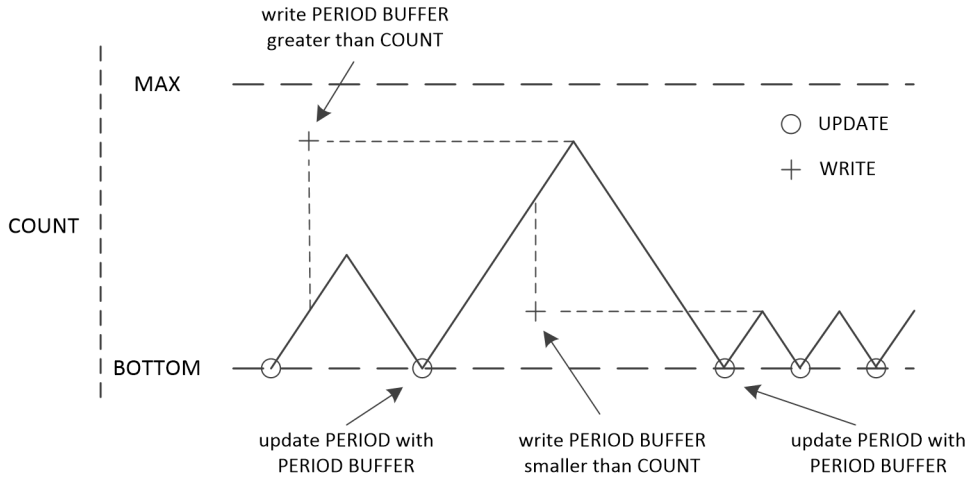


Figure 1.5. Change of the period value with buffering.

Figure 1.5 depicts the same change (Capture/Compare channel in Dual Slope PWM mode) with buffering. The new period value is applied on UPDATE event to ensure convenient transition.

1.4 Counting Modes

The Timer module counts peripheral clock cycles (PCLK). The counting scheme comprises of two levels. The first one is prescaler counter that directly counts PCLK clock cycles up to the prescaler register (PRES, 1.10.5) value and gets reset. The COUNT (1.10.7) register counts prescaler overflow events and is incremented after every $PRES+1$ PCLK clock cycles. The PRES register can be additionally double-buffered. The PRES register is written with PRESBUF (1.10.6) register on UPDATE event, but only if PRESBUF register holds valid data.

1.5 Counter Mode

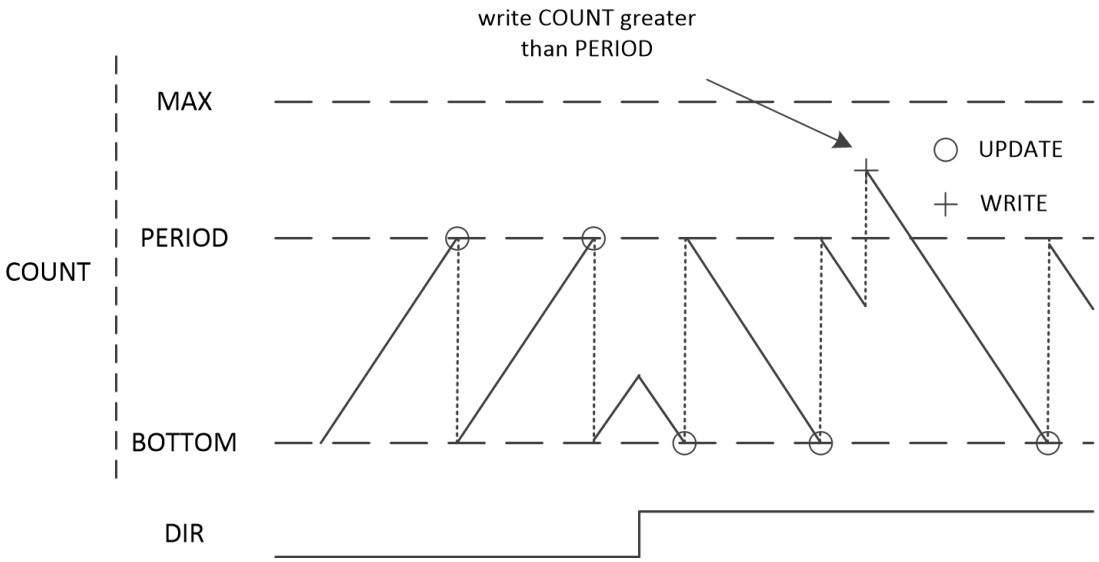


Figure 1.6. Counter mode.

In Counter mode, module increments or decrements count register up or down to PERIOD/TOP or BOTTOM value (Figure 1.6) depending on DIR bit in CTRL register (1.10.2). In incrementing mode, reaching PERIOD/TOP value will cause:

- UPDATE event,
- overflow indication (OVFIF flag in IRQM (1.10.8) register),
- zero of COUNT register on next prescaler overflow.

In case of decrementation, reaching BOTTOM value will cause:

- UPDATE event,
- overflow indication (OVFIF flag in IRQM (1.10.8) register),
- load COUNT register with period (PER) value on next prescaler overflow.

1.6 Capture Modes

Capture/Compare channels can be used for detecting and stamping external events (rising or falling edges) on dedicated Capture inputs. On Capture event the value of COUNT register is stored in FIFO composed of CCDATA (1.10.14) and CCDATABUF (1.10.15) registers. There are three capture modes.

1.6.1 Input Capture

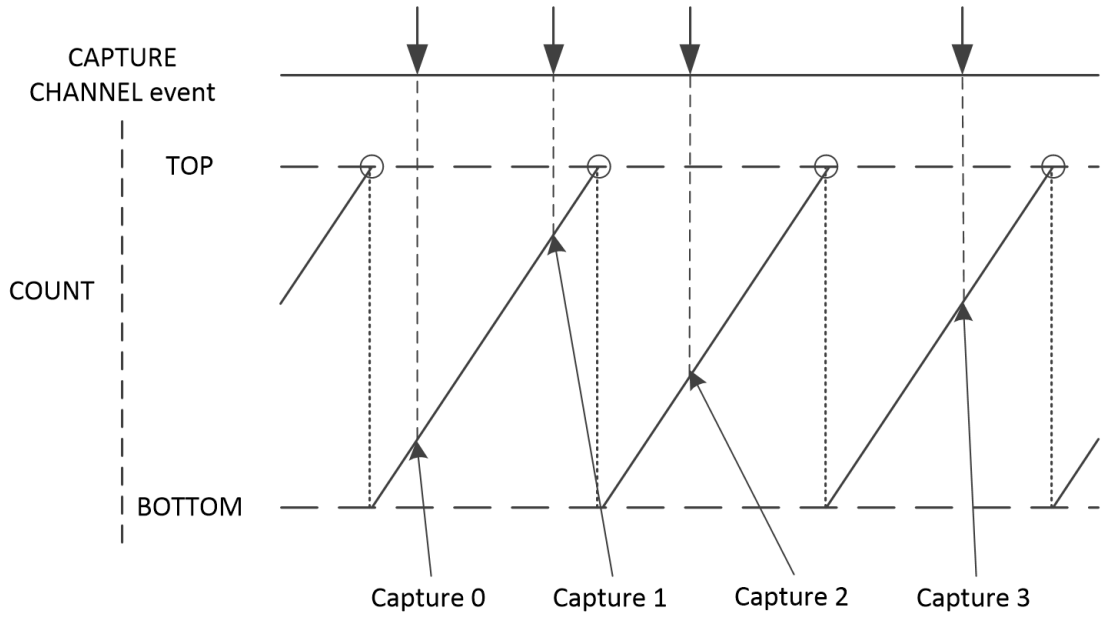


Figure 1.7. Input Capture mode.

In Input Capture mode Capture/Compare channels detect rising or falling edges of dedicated Capture input lines. On Capture event the value of COUNT register is stored in CCDATA register and CC*n*IF flag is set. The Timer continues to count to BOTTOM or PERIOD/TOP value and is zeroed (Figure 1.7).

1.6.2 Frequency Capture

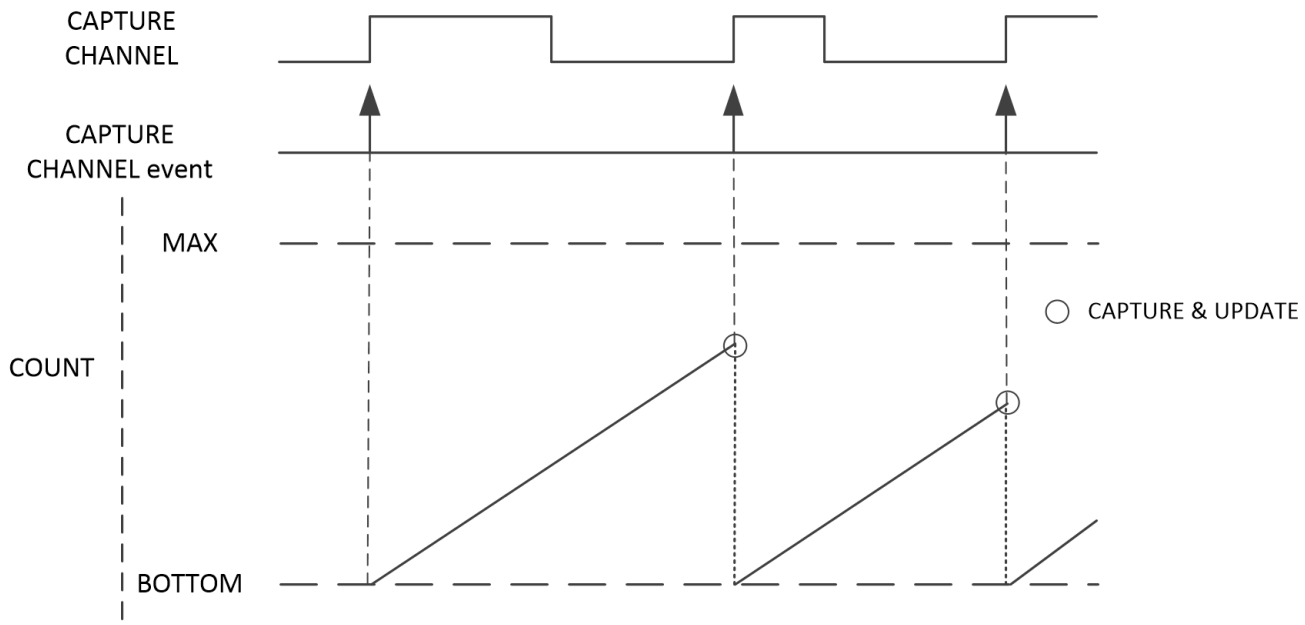


Figure 1.8. Frequency Capture mode.

In Frequency Capture mode the rising edge on Capture input stores COUNT value in CCDATA FIFO and zeros the Timer COUNT register. This allows to direct measure the signal period (Figure 1.8). The maximum frequency that can be detected is limited by the PCLK clock. Because all of Capture/Compare channels utilize the same COUNT register, only one channel can work in Frequency Capture mode. Otherwise, The COUNT register will be zeroed by every rising edge of all active Frequency Capture channels. In Frequency Capture mode, the Timer counts to the MAX value, period register (PER) is not taken into account. The OVFIF flag is set when COUNT is reset and holds BOTTOM value.

1.6.3 Pulse-Width Capture

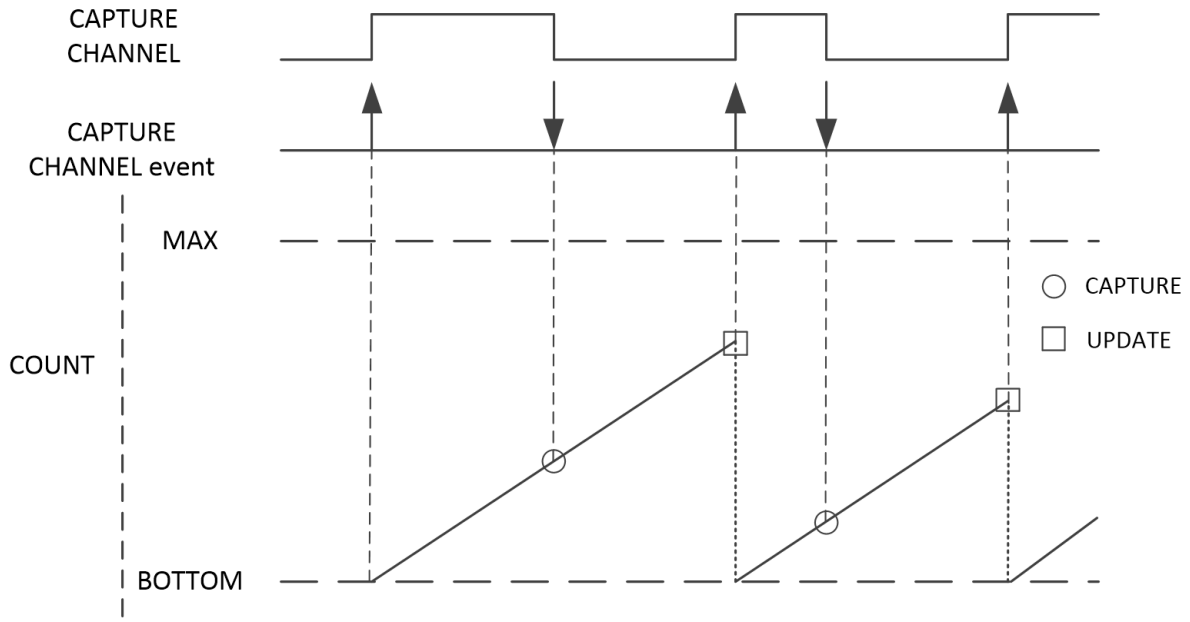


Figure 1.9. Tryb Pulse Width Capture.

In Pulse-Width Capture mode, rising edge of Capture input resets COUNT register, and falling edge stores its value in CCDATA FIFO (1.9). Because all of Capture/Compare channels utilize the same COUNT register, only one channel can work in Pulse-Width Capture mode. Otherwise, The COUNT register will be zeroed by every rising edge of all active Frequency Capture channels. In Pulse-Width Capture mode, the Timer counts to the MAX value, period register (PER) is not taken into account. The OVFIF flag is set when COUNT is reset and holds BOTTOM value.

1.7 Compare Modes

Capture/Compare channels can be used to generate waveforms on dedicated PWM output lines. In this mode, the active channels compare Timer COUNT register with CCDATA value. When both values are equal, channel indicates Compare Match event and CC*n*IF flag is set. Compare Match event is also used to update the state of PWM output line. The CCDATA registers are buffered with CCDATABUF registers to ensure convenient transition of PWN output lines. Each of Capture/Compare channels has its own CCDATA and CCDATABUF registers so they can work independently as long as they share the same period value.

1.7.1 Single Slope PWM

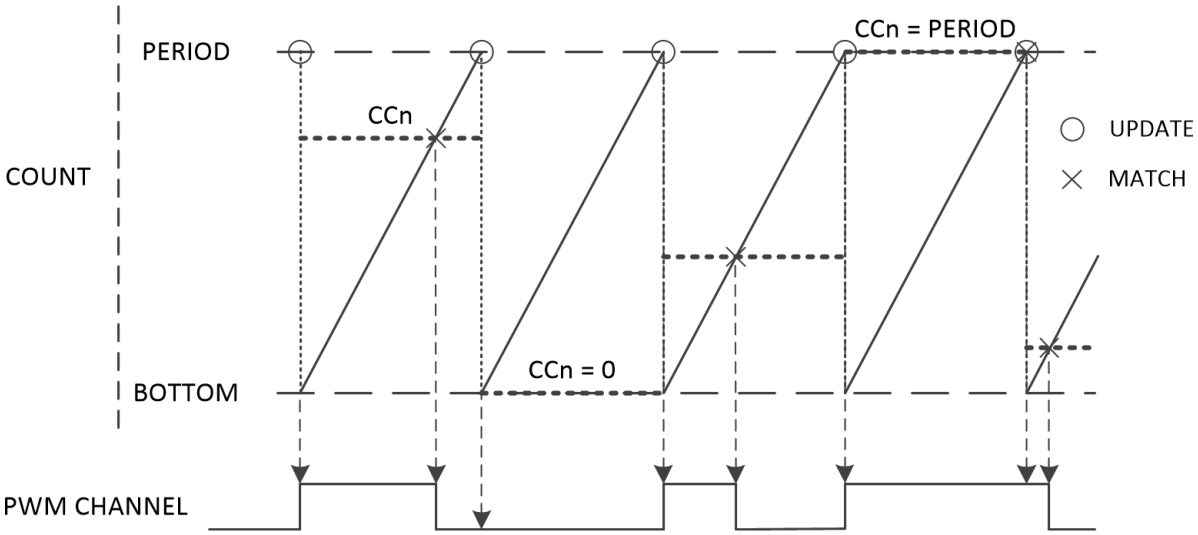


Figure 1.10. Single Slope PWM mode.

In Single Slope PWM mode, the period of generated waveform is controlled by period register and equals $(PER + 1) * (PRES + 1)$ PCLK clock cycles. CCDATA (1.10.14) registers are used to control generated waveform duty cycle. The Timer counts from BOTTOM to PERIOD/TOP or from PERIOD/TOP down to BOTTOM depending on DIR bit in CTRL register (Figure 1.12). PWM output is set high when counter reaches PERIOD/TOP value. The PWM output is set low on Compare Match event. The PWM is set high for $(CCDATA+1)*(PRES+1)$ PCLK clock cycles.

1.7.2 Dual Slope PWM

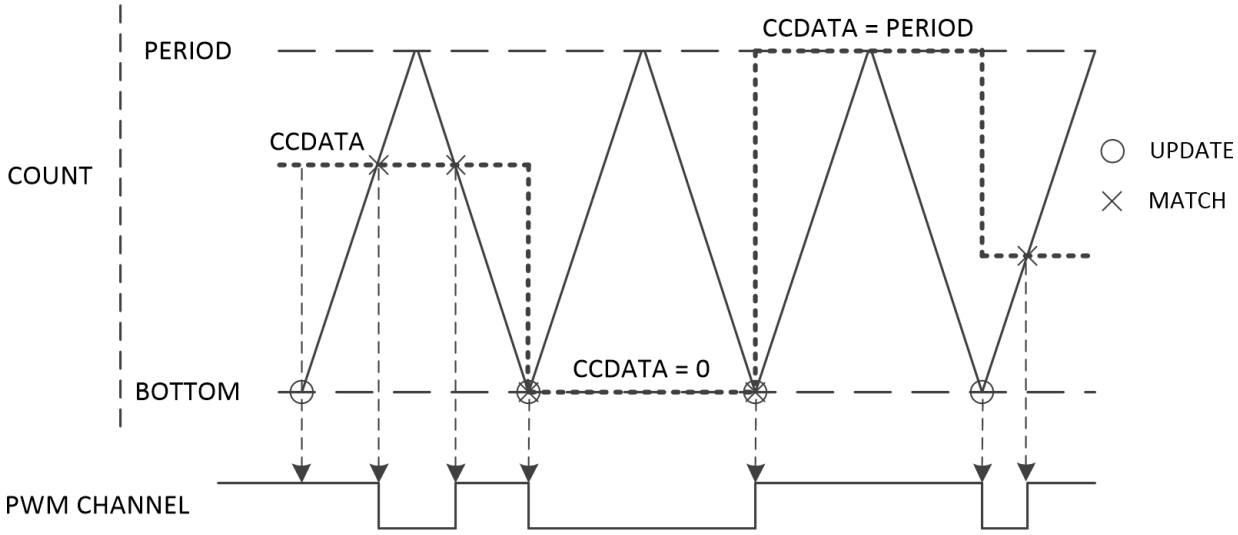


Figure 1.11. Dual Slope PWM mode.

In Dual Slope PWM mode, the period of generated waveform is controlled by period register and equals $(2 * PER + 1) * (PRES + 1)$ PCLK clock cycles. CCDATA (1.10.14) registers are used to control generated waveform duty cycle. The Timer counts from BOTTOM to PERIOD/TOP value and then decremented to BOTTOM (Figure 1.11) The PWM output is set as follows:

- high - when reaching BOTTOM value,
- low - on Compare Match event when counter is incrementing,
- high - on Compare Match event when counter is decrementing.

The PWM is set high for $2 * CCDATA$ PCLK clock cycles.

1.7.3 PWM Output Pads

Capture/Compare PWM output enable signals are all activated after writing EN bit in Timer control register (CTRL, 1.10.2). Because of that PWM output lines do not directly share the same pad with Capture inputs. Pad sharing is achieved with alternative function feature embedded in the GPIO controller (see CC-GPIO core).

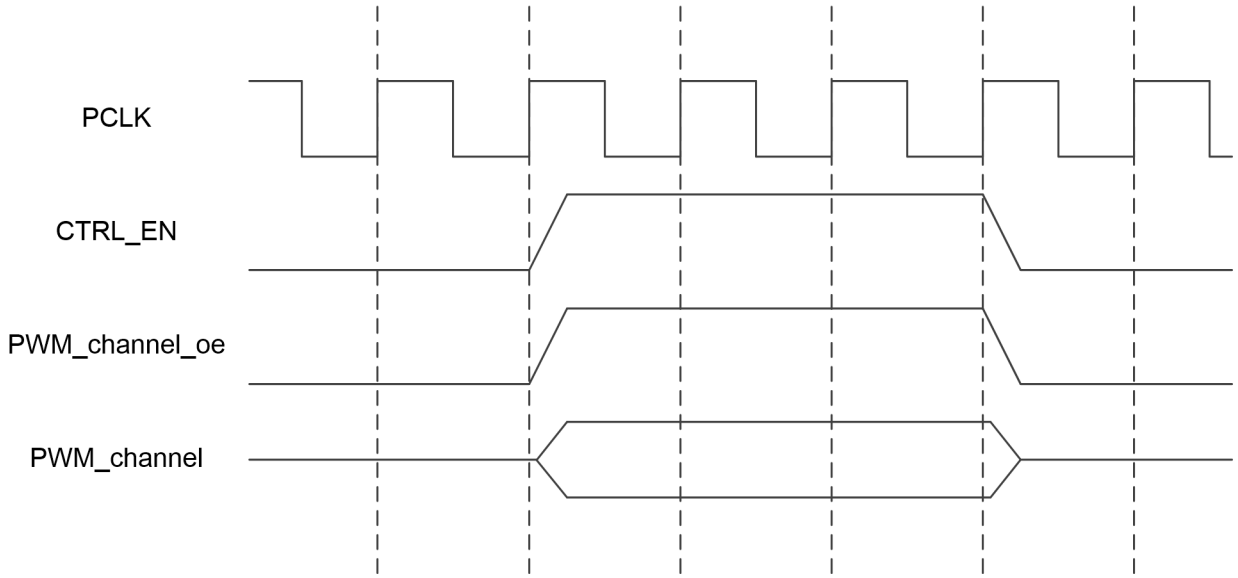


Figure 1.12. PWM pad output enable timing.

1.8 Commands

The Timer module provides a set of commands for instant change of module behaviour.

Command	Description
UPDATE	Generates UPDATE event
RESTART	Clears COUNT and PRES registers
RESET	Resets the entire Timer module

1.9 Interrupts

The Timer module has three interrupt sources.

1.9.1 Overflow Interrupt

The Overflow Interrupt is signalized by OVFIF flag in IRQF register. Depending on configuration overflow interrupt occurs when COUNT reaches PERIOD/TOP or BOTTOM value. The interrupt is cleared after reading IRQF register or by writing one in OVFIF bit.

1.9.2 Error Interrupt

The Error Interrupt is signalized by ERRIF flag in IRQF register. It is set only in Capture mode, when Capture event occurs with both CC*n*IF and CC*n*BV flags set. The interrupt is cleared after reading IRQF register or by writing one in ERRIF bit.

1.9.3 Capture/Compare Interrupt

The Capture/Compare Interrupt is signalized by CC*n*IF flag in IRQF register. It is set on Compare Match event in Compare mode or Capture event in Capture mode. The interrupt is cleared after reading IRQF register or by writing one in CC*n*IF bit.

1.10 Configuration Registers

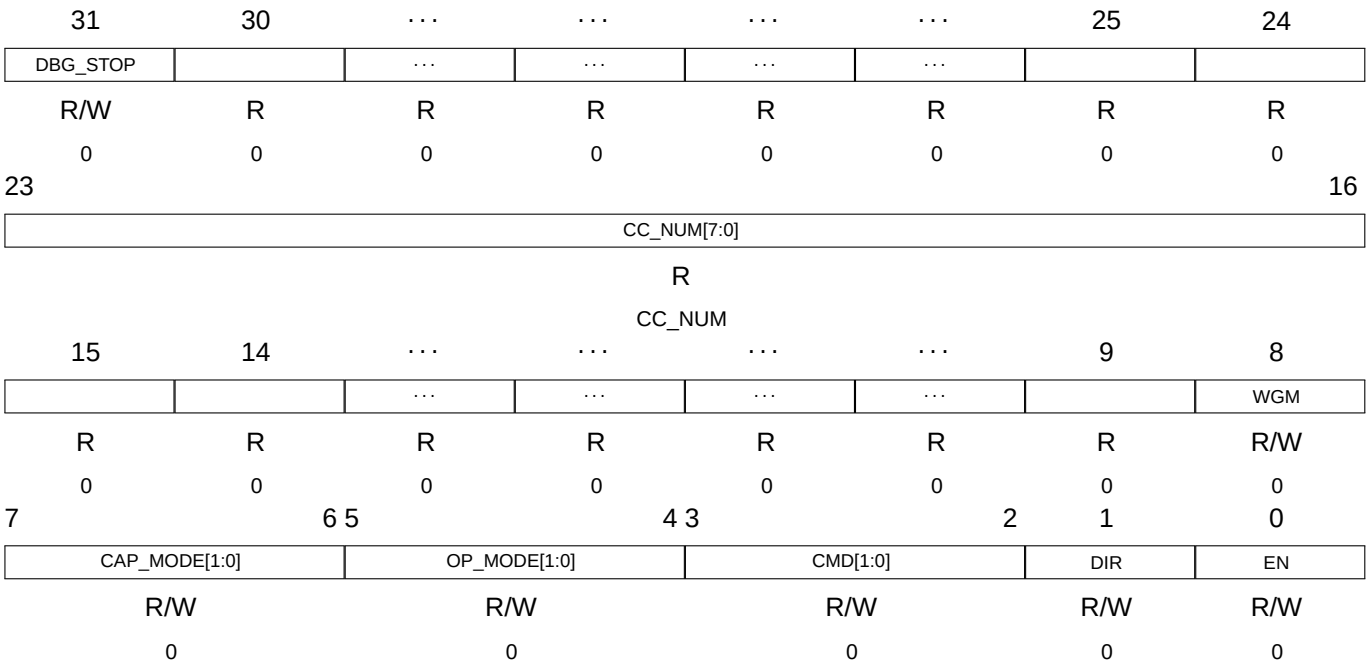
1.10.1 Registers List

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

Address Offset	Register	Name
General Registers		
0x00	CTRL	Control Register
0x04	PER	Period Register
0x08	PERBUF	Period Buffer Register
0x0C	PRES	Prescaler Register
0x10	PRESBUF	Prescaler Buffer Register
0x14	COUNT	Count Register
0x18	IRQM	Interrupt Mask Register
0x1C	IRQF	Interrupt Flags Register
0x20	IRQMAP	Interrupt Mapping Register
0x24	BUFVD	Buffer Valid Register
0x28	COMPDEF	Compare Output Default Register
Capture/Compare Channels Registers		
0x2C + 0x0C*channelIndex	CCCTRL	Channel Control Register
0x30 + 0x0C*channelIndex	CCDATA	Channel Data Register
0x34 + 0x0C*channelIndex	CCDATABUF	Channel Data Buffer Register

1.10.2 Control Register

Address: 0x00



EN *Timer Enable*

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

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

DIR *Counter Direction*

0 Timer is incrementing.

1 Timer is decrementing.

Setting is valid only for:

- OP_MODE = 00 (TIMER)
- OP_MODE = 10 (WAVEFORM GENERATOR) or WGM = 0 (SINGLE SLOPE)

CMD[1:0] *Module Command*

Module Command:

CMD[1:0]	Command
00	NO OP
01	RESTART
10	UPDATE
11	RESET

OP_MODE[1:0] *Operating Mode*

Timer operating mode:

OP_MODE[1:0]	Operating Mode
00	TIMER
01	CAPTURE
10	WAVEFORM GENERATION
11	RESERVED

CAP_MODE[1:0] *CAPTURE Mode*

Timer CAPTURE mode (valid only for OP_MODE = 01 (CAPTURE)):

CAP_MODE[1:0]	CAPTURE Mode
00	NORMAL
01	PERIOD
10	PULSE WIDTH
11	RESERVED

WGM *Waveform Generator Mode*

Waveform Generator Mode:

0 Single Slope.

1 Dual Slope.

CC_NUM[7:0] *Capture/Compare Channels Number*

Number of implemented Capture/Compare channels.

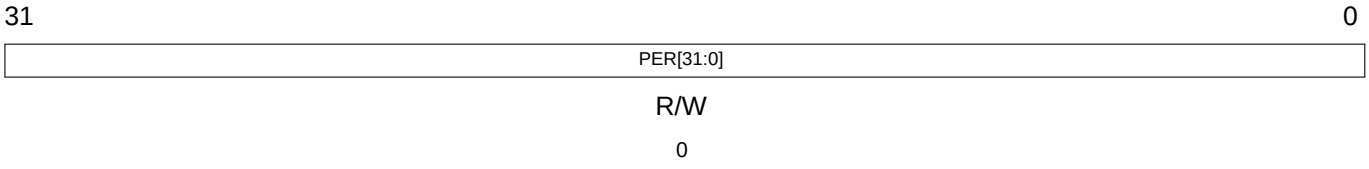
DBG_STOP *Timer stop in debug mode*

0 Timer is counting in debug mode.

1 Timer is not counting in debug mode.

1.10.3 Period Register

Address: 0x04



PER[31:0] *Counter Period*

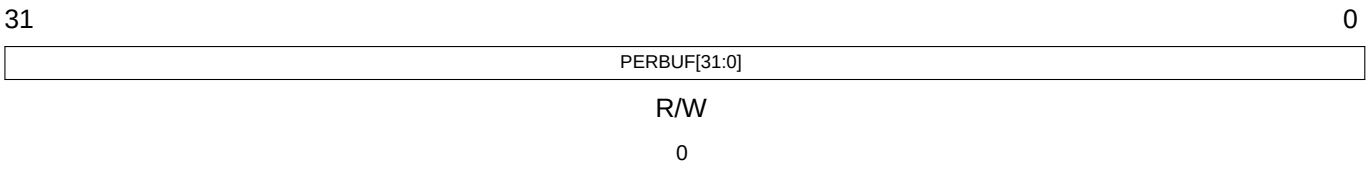
Timer counter period (TOP value). Behaviour depends on DIR bit in CTRL register.

INCR Reaching PER value generates overflow (OVF) and clears COUNT register,

DECR Reaching 0 value generates overflow (OVF) event and COUNT register is set with PER value.

1.10.4 Period Buffer Register

Address: 0x08

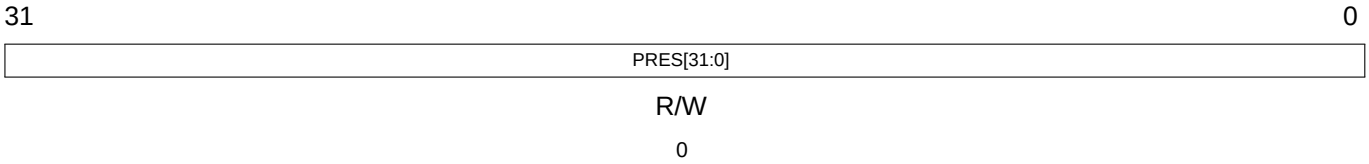


PERBUF[31:0] *Counter Period Buffer*

Period register buffer. If PERBUF register stores valid data (written by the user) its content is written to PER register on UPDATE event.

1.10.5 Prescaler Register

Address: 0x0C

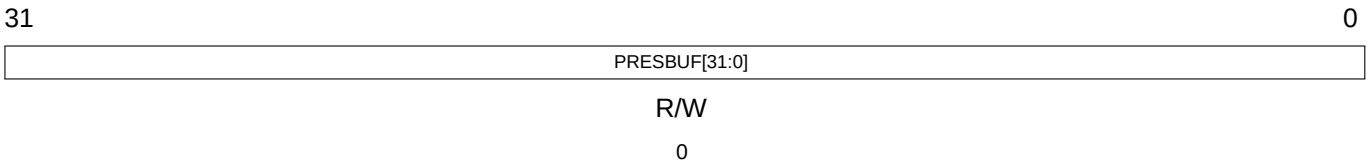


PRES[31:0] *Counter Prescaler*

Timer module prescaler. COUNT register is incremented or decremented every each $PRES + 1$ PCLK clock cycles.

1.10.6 Prescaler Buffer Register

Address: 0x10

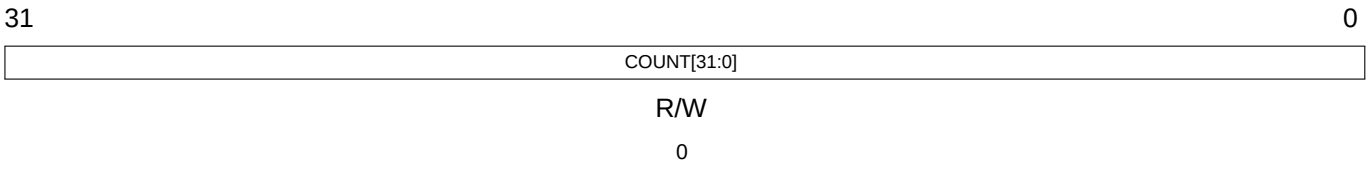


PRSBUF[31:0] *Counter Prescaler Buffer*

Prescaler register buffer. If PRESBUF register stores valid data (written by the user) its content is written to PRES register on UPDATE event.

1.10.7 Count Register

Address: 0x14

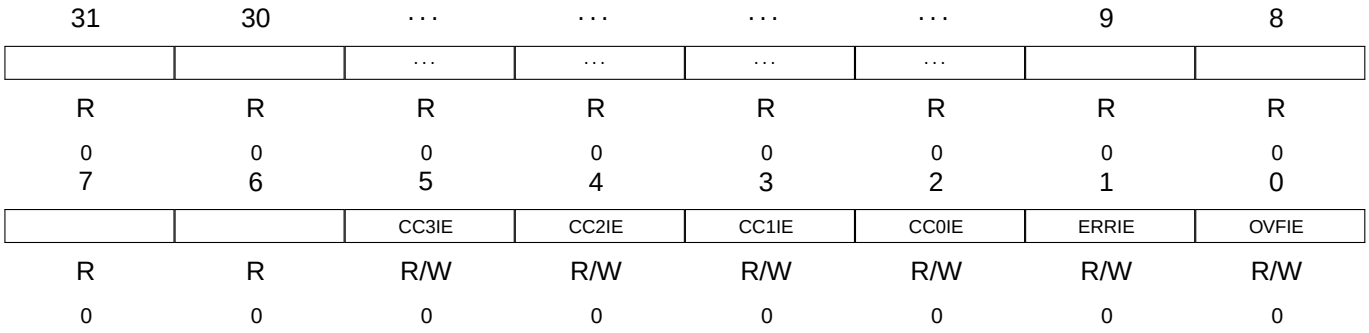


COUNT[31:0] *Current Count*

Current value of Timer counter.

1.10.8 Interrupt Mask Register

Address: 0x18



OVFIE *Overflow Interrupt Enable*

0 Overflow interrupt disabled.

1 Overflow interrupt enabled.

ERRIE *Error Interrupt Enable*

0 Error interrupt disabled.

1 Error interrupt enabled.

CCnIE *Capture/Compare n Channel Interrupt Enable*

0 Capture/Compare interrupt of *n* channel disabled.

1 Capture/Compare interrupt of *n* channel enabled.

1.10.9 Interrupt Flags Register

Address: 0x1C

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
		CC3IF	CC2IF	CC1IF	CC0IF	ERRIF	OVFIF
R	R	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

OVFIF *Overflow Interrupt Flag*

1 Counter reached PERIOD or 0 value (depending on DIR configuration).

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

ERRIF *Error Interrupt Flag*

1 CCDATA FIFO overflow detected.

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

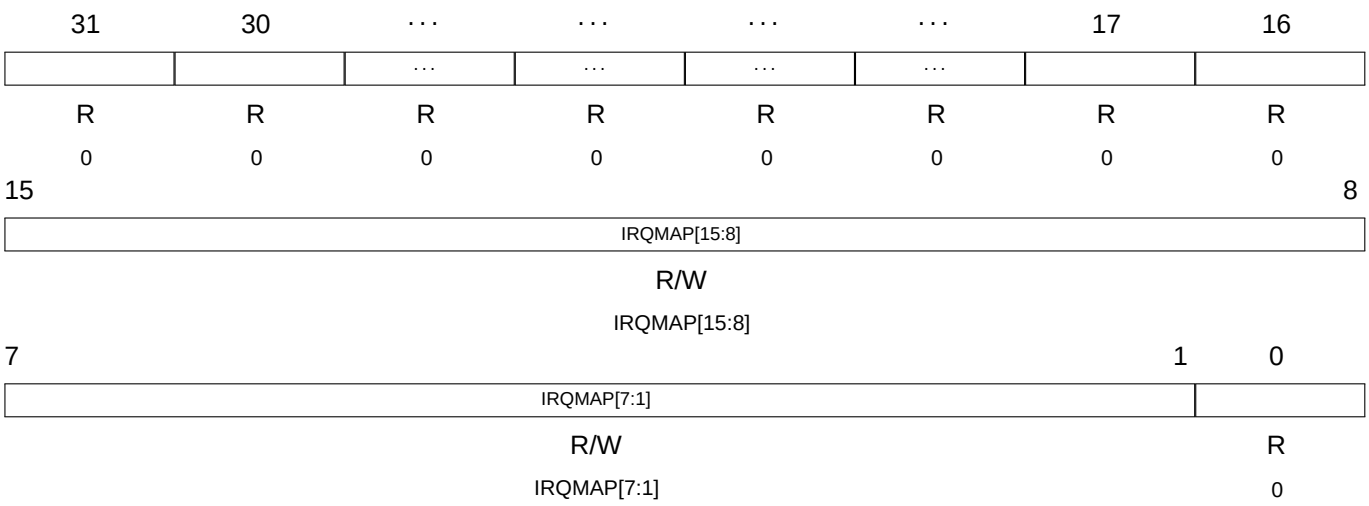
CCnIF *Capture/Compare n Channel Interrupt Flag*

1 Capture/Compare event is detected.

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

1.10.10 Interrupt Mapping Register

Address: 0x20

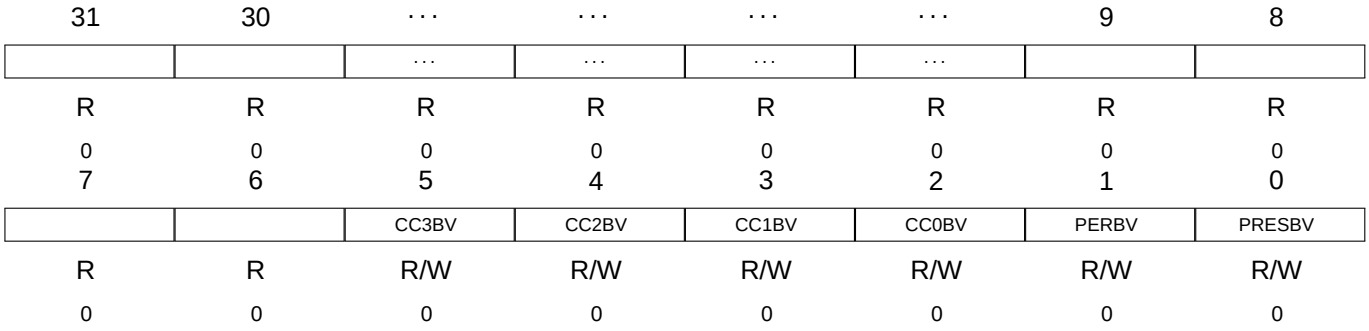


IRQMAP[15:1] *Interrupt Mapping*

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

1.10.11 Buffer Valid Register

Address: 0x24



PRESBV *Prescaler Buffer Valid*

0 Prescaler buffer register does not hold valid data.

1 Prescaler buffer data is valid.

PERBV *Period Buffer Valid*

0 Period buffer register does not hold valid data.

1 Period buffer data is valid.

CCnBV *Capture/Compare n Channel Buffer Valid*

Behaviour depends on Timer mode.

COMPARE Bit set means that CCDATABUF value was not written into CCDATA register,

CAPTURE Bit set means that CCDATABUF stores valid data sample. CCDATABUF readout will clear this bit.

1.10.12 Compare Output Default Register

Address: 0x28

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
		CC3DEF	CC2DEF	CC1DEF	CC0DEF		
R	R	R/W	R/W	R/W	R/W	R	R
0	0	0	0	0	0	0	0

CCnDEF *Compare n Channel Default Value*

Default state of PWM output when channel is inactive or is not working in COMPARE mode.

1.10.13 Channel Control Register

Address: 0x2C + 0x0C*channelIndex

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
					CC_MODE[1:0]		CC_EN
R	R	R	R	R	R/W		R/W
0	0	0	0	0	0		0

CC_EN *Capture/Compare Channel n Enable*

0 Channel is inactive.

1 Channel is active.

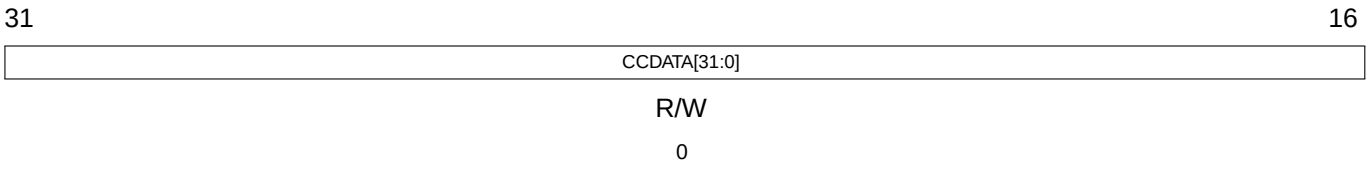
CC_MODE[1:0] *Capture Mode*

Configuration of operating signal edges in Capture mode:

CC_MODE[1:0]	Mode
00	RISING EDGE
01	FALLING EDGE
10	BOTH EDGES
11	RESERVED

1.10.14 Channel Data Register

Address: 0x30 + 0x0C*channelIndex



CCDATA[31:0] *Capture/Compare n Data*

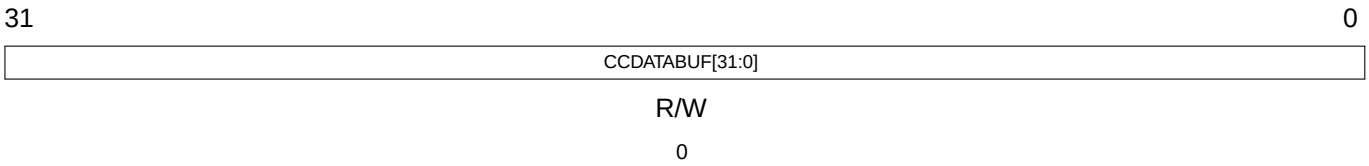
Behaviour depends on Timer mode.

CAPTURE Value sampled on Capture event.

COMPARE Configured value in Compare Match mode.

1.10.15 Channel Data Buffer Register

Address: 0x34 + 0x0C*channelIndex



CCDATABUF[31:0] *Capture/Compare n Buffer*

Behaviour depends on Timer mode.

CAPTURE Value sampled on Capture event when CCDATA stores valid data.

COMPARE Buffer for CCDATA register. If CCDATABUF register stores valid data (written by the user) its content is written to CCDATA register on UPDATE event.

1.11 Implementation

1.11.1 Design Structure

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

```
COMMON
├── rtl
│   ├── DFF_en.v
│   ├── edge_detector.v
│   └── synchronizer.v
TIMER
├── beh
├── rtl
│   ├── APB_TIMER.v
│   ├── TIMER_config.v
│   ├── TIMER_defines.v
│   └── TIMER.v
├── tb
│   ├── APB
│   │   ├── tb_APB_TIMER_init.v
│   │   └── tb_APB_TIMER_reg_access_tasks.v
│   ├── common
│   │   ├── tb_TIMER_other_tasks.v
│   │   ├── tb_TIMER_read_config_tasks.v
│   │   ├── tb_TIMER_write_config_tasks.v
│   │   └── timescale.v
│   ├── run
│   │   └── ncvlog_apb_timer.sh
│   ├── tests
│   │   ├── tb_interrupt_MAPPING_test.sv
│   │   ├── tb_TIMER_CAPTURE_NORMAL_test.sv
│   │   ├── tb_TIMER_CAPTURE_PERIOD_test.sv
│   │   ├── tb_TIMER_CAPTURE_PULSE_WIDTH_test.sv
│   │   ├── tb_TIMER_COMPARE_DS_test.sv
│   │   ├── tb_TIMER_COMPARE_SS_FE_test.sv
│   │   ├── tb_TIMER_COMPARE_SS_RE_test.sv
│   │   ├── tb_TIMER_DECREMENT_test.sv
│   │   └── tb_TIMER_INCREMENT_test.sv
│   └── tb_APB_TIMER.sv
└── compile.list
```

1.11.2 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_timer.sh* script located in *TIMER/tb/run* folder. The simulation should end with reporting no errors.

1.11.3 Clock and Reset

The CC-TIMER-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.

1.11.4 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).

By default module capture inputs are synchronized using Synchronizer2 module located in the synchronizer.v file. If possible, they should be replaced with integrated 2FF synchronizers from the target technology library. Otherwise, max delay (*set_max_delay* command) of 10% to 20% of one destination clock cycle should be set between synchronizer stages. Do not use dynamic FFs to implement synchronizer module.

1.11.5 Configuration Options

The table below shows the generic parameters of the core.

Generic name	Description	Range	Default
timer_width	Configure width of timer count register	1:32	32
prescaler_width	Configure width of timer prescaler register	1:32	8
cc_channels_number	Number of Capture/Compare channels	0:4	4
address_width	Must be set accordingly to the number of Capture/Compare channels - $clog2(11 + cc\_channels\_number * 3)$	4:5	5
default_interrupt_MAPPING	Reset value of interrupt_MAPPING register	0:32767	0

The table below shows the define parameters of the core (TIMER_config.v file).

Define name	Description	Default
TIMER_INPUTS_SYNCHRONIZATION	Comment to remove capture input synchronizing flip flops	defined

1.11.6 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[address_width+1:2]	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.
interrupt_OVF	Overflow interrupt	O	high	reg.
interrupt_ERR	Error interrupt	O	high	reg.
interrupt_CC	Capture/Compare interrupt	O	high	reg.
interrupt_MAPPING[15:1]	Interrupt mapping vector	O	data	reg.
capture_channels [cc_channels_number:0][1]	Capture mode input	I	data	reg./comb.[2]
PWM_channels [cc_channels_number:0][3]	PWM waveform generation output	O	data	reg.
PWM_channels_oe [cc_channels_number:0][3]	PWM output enable	O	high	reg.
clock_request	Clock request signal	O	high	reg.
debug_mode	Debug mode indicator (1 - core is halted)	I	high	comb.

[1] The actual usable width of capture_channels signal is [cc_channels_number-1:0]. capture_channels[cc_channels_number] bit is not used and its purpose is to avoid generating negative index when mcc_channels_number = 0.

[2] Depending on TIMER_INPUTS_SYNCHRONIZATION setting. Defined value will insert input synchronization flip-flops.

[3] The actual usable width of PWM_channels and PWM_channels_oe signals is [cc_channels_number-1:0]. [cc_channels_number] bit is hardcoded as zero to avoid generating negative index when mcc_channels_number = 0.

1.11.7 Instantiation

```
icg
    icg_tmr_u ( .E(tmr_PSEL|tmr_clock_request),
                .clk(PCLK),
                .gclk(tmr_clk),
                .scan_enable(scan_enable));

APB_TIMER   #(  .timer_width(CFG_TMR_WIDTH),
                .prescaler_width(CFG_TMR_PRES),
                .cc_channels_number(CFG_CHANNELS_NUM),
                .address_width(CFG_TMR_REG_LOG),
                .default_interrupt_MAPPING(CFG_DEF_INT_MAPPING)

APB_TIMER_u  (  .PCLK(tmr_clk),
                .PRESETn(rst),
                .PSEL(tmr_PSEL),
                .PENABLE(PENABLE),
                .PADDR(PADDR[CFG_TMR_REG_LOG+1:2]),
                .PWRITE(PWRITE),
                .PWDATA(PWDATA),
                .PREADY(tmr_PREADY),
                .PRDATA(tmr_PRDATA),
                .interrupt_OVF(tmr_interrupt_OVF),
                .interrupt_ERR(tmr_interrupt_ERR),
                .interrupt_CC(tmr_interrupt_CC),
                .interrupt_MAPPING(tmr_interrupt_MAPPING),
                .clock_request(tmr_clock_request),
                .capture_channels({1'b0,tmr_capture_channels[CFG_CHANNELS_NUM-1:0]}),
                .PWM_channels(     {tmr_PWM_channels_dummy,
                                    tmr_PWM_channels[CFG_CHANNELS_NUM-1:0]}),
                .PWM_channels_oe(  {tmr_PWM_channels_oe_dummy,
                                    tmr_PWM_channels_oe[CFG_CHANNELS_NUM-1:0]}),
                .debug_mode(debug_mode));

               assign  tmr_irq         = tmr_interrupt_OVF | tmr_interrupt_ERR
                                       | (|tmr_interrupt_CC);
               assign  tmr_irq_mapped  = tmr_interrupt_MAPPING & {15{tmr_irq}};

io_pad_model      #( .IO_NUM(CFG_CHANNELS_NUM))
  capture_pad_model( .core_input(tmr_capture_channels[CFG_CHANNELS_NUM-1:0]),
```


```
                    .core_output(1'b0),
                    .IO_pad(capture_channels_pad[CFG_CHANNELS_NUM-1:0]),
                    .output_enable(1'b0));

io_pad_model      #( .IO_NUM(CFG_CHANNELS_NUM))
  PWM_pad_model    ( .core_input(1'b0),
                    .core_output(tmr_PWM_channels[CFG_CHANNELS_NUM-1:0]),
                    .IO_pad(PWM_channels_pad[CFG_CHANNELS_NUM-1:0]),
                    .output_enable(tmr_PWM_channels_oe[CFG_CHANNELS_NUM-1:0]));
```

1.12 Revision History

Doc. Rev.	Date	Comments
1.2	11-2017	Corrected 1.11.6 Signals Description and 1.11.7 Instantiation sections.
1.1	09-2017	Updated 1.11.4 Constraints section. Added description of synchronizers implementation. Added footnotes describing dummy input and output signals in 1.11.6 Signals Description section.
1.0	05-2017	First Issue.

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

 CC-TIMER-APB-Doc_112017.