

---

# **SpartanMC**

## ***Basic Timer (Timer)***

---

---

---

# Table of Contents

<b>1. Module parameters</b> .....	<b>1</b>
<b>2. Peripheral Registers</b> .....	<b>2</b>
2.1. Timer Register Description .....	2
2.2. TIMER_CTRL Register .....	2
2.3. TIMER_DAT Register .....	3
2.4. TIMER_VALUE Register .....	3
2.5. TIMER C-Header for Register Description .....	3



# List of Figures

1 Timer block diagram ..... 1



## List of Tables

1	TIMER module parameters .....	1
1	TIMER registers .....	2
1	TIMER_CTRL register layout .....	2
1	TIMER_DAT register layout .....	3
1	TIMER_VALUE register layout .....	3





## Basic Timer (Timer)

The Basic Timer module can be used to divide the system clock frequency to a user defined periodic signal required by the application. For this purpose the Basic Timer provides a configurable prescaler. If enabled, the prescaler allows the usage of all powers of two between 2 and 256 as prescaler value. The input of prescaler block could be connected to the system clock, a dedicated DCM output or to the output (data register) of a previous timer module. The output of the prescaler is used as input of an 18 bit counter which counts up to a programmable value and restarts afterwards.

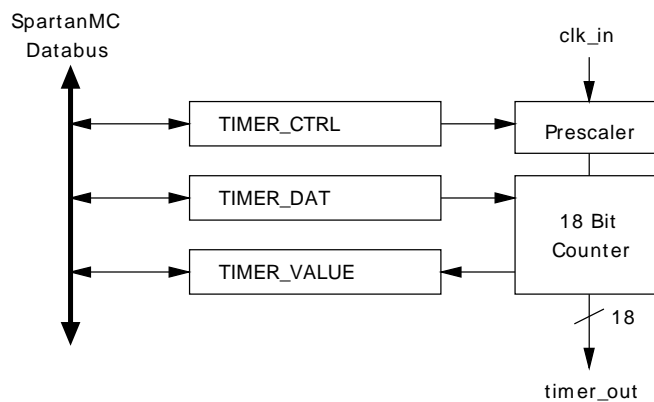


Figure 1: Timer block diagram

The basic timer could be used for the following peripherals: Real Time Interrupt (RTI), Watchdog, Capture-, Compare-Module and Pulse-Accumulator-Module. The capture- and compare-module require on their input a complete 18 bit counter register output. While all other modules (RTI, Pulse-Accumulator, Watchdog and additional basic timer) require only one single output bit for their clock input.

### 1. Module parameters

Table 1: TIMER module parameters

Parameter	Default Value	Description
BASE_ADR	0x10	Start address of the memory mapped peripheral registers. The value is taken as offset to the start address of the peripheral memory space. <b>This parameter is set by jConfig automatically.</b>

## 2. Peripheral Registers

### 2.1. Timer Register Description

The timer peripheral provides three 18 bit registers which are mapped to the SpartanMC address space e.g.  $0x1A000 + \text{BASE\_ADR} + \text{Offset}$ .

**Table 2: TIMER registers**

Offset	Name	Access	Description
0	TIMER_CTRL	read/ write	Configuration of the timer.
1	TIMER_DAT	read/ write	Maximum value of the 18 bit counter.
2	TIMER_VALUE	read/ write	Current counter value.

### 2.2. TIMER\_CTRL Register

**Table 3: TIMER\_CTRL register layout**

Bit	Name	Access	Default	Description
0	TI_EN	read/ write	0	If set to one the timer is enabled.
1	TI_PRE_EN	read/ write	0	If set to one the prescaler is enabled.
2-4	TI_PRE_VAL	read/ write	000	Sets the prescaler value :  000 = $2^1$ 001 = $2^2$ 010 = $2^3$ 011 = $2^4$ 100 = $2^5$ 101 = $2^6$ 110 = $2^7$ 111 = $2^8$
5-17	x	read	0	Not used.

**Table 3: TIMER\_CTRL register layout**

## 2.3. TIMER\_DAT Register

**Table 4: TIMER\_DAT register layout**

Bit	Name	Access	Default	Description
0-17	Max Counter	read/ write	x	Register for the maximum counter value.

## 2.4. TIMER\_VALUE Register

**Table 5: TIMER\_VALUE register layout**

Bit	Name	Access	Default	Description
0-17	Main Counter	read/ write	0	Register for the current counter value. The content of this 18 bit register is used as timer_output and could be connected to other peripherals e.g. capture- or compare-logic. A single bit of this register could also be used to cascade multiple timers.

## 2.5. TIMER C-Header for Register Description

```

#ifndef __TIMER_H
#define __TIMER_H

#ifdef __cplusplus
extern "C" {
#endif

#define TIMER_EN      (1<<0)
#define TIMER_PRE_EN  (1<<1)
#define TIMER_PRE_VAL (1<<2)      // *0 fuer 2^1 bis *7 fuer
2^8
#define TIMER_PRE_2   (TIMER_PRE_VAL * 0)
#define TIMER_PRE_4   (TIMER_PRE_VAL * 1)
#define TIMER_PRE_8   (TIMER_PRE_VAL * 2)
#define TIMER_PRE_16  (TIMER_PRE_VAL * 3)
#define TIMER_PRE_32  (TIMER_PRE_VAL * 4)
#define TIMER_PRE_64  (TIMER_PRE_VAL * 5)
#define TIMER_PRE_128 (TIMER_PRE_VAL * 6)
#define TIMER_PRE_256 (TIMER_PRE_VAL * 7)

typedef struct timer {

```

```
    volatile unsigned int control;  
    volatile unsigned int limit;  
    volatile unsigned int value;  
} timer_regs_t;  
  
#ifdef __cplusplus  
}  
#endif  
  
#endif
```