SpartanMC

Timer Compare Module (timer-cmp)

Table of Contents

List of Figures

List of Tables

Timer Compare Module (timer-cmp)

The timer compare module is used to generate variable frequencies or programmable duty cycles by comparing an internal value to a given timer value.

Note: **The timer compare module always requires a basic timer module as input. Hence, it can not work autonomously.**

(Otherwise, a basic timer could be used as input for multiple capture moduls.)

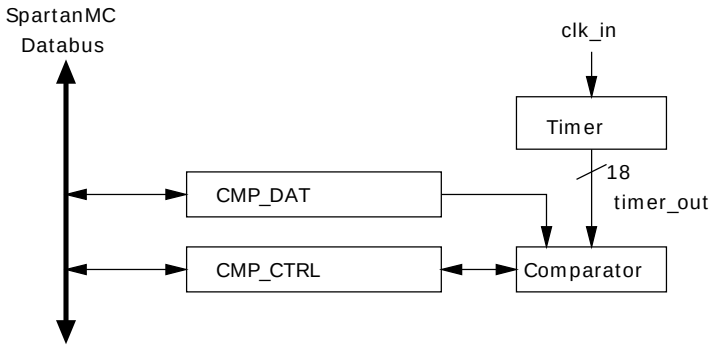

Figure 1: Timer compare module block diagram

1. Usage and Interrupts

If the programmend value of the compare register equals the current timer value the timer compare module triggers an event. These events could be the generation of an interrupt or the switching of the output pin (set, reset, or negate). In case of an interrupt generation, the interrupt is cleared on each access to the modules registers. In case the module output pin is used, the compare module contains a control register which specifies the behavior of this pin.

2. Module parameters

Table 1: TIMER Compare module parameters

Parameter	Default Value	Descripton
BASE_ADR		Start address of the memory mapped peripheral registers. The value is taken as offset to the start address of the peripheral memory space. **This parameter is set by jConfig automatically.**

3. Peripheral Registers

3.1. Timer Compare Register Description

The timer compare module provides tow 18 bit registers which are mapped to the SpartanMC address space e.g. 0x1A000 + BASE_ADR + Offset.

Table 2: Timer Compare registers

Offset	Name	Access	Description
0	CMP_CTRL	read/ write	Specify the operation mode. (An access on this register clears the interrupt flag)
1	CMP_DAT	read/ write	Compare value for the 18 bit counter of the basic timer modules.

3.2. Compare Control Register

Table 3: CMP_CTRL register layout

Bit	Name	Access	Default	Description
0	CMP_EN	read/ write	0	If set to one the compare logic is enabled.
1	CMP_EN_INT	read/ write	0	If set to one the interrupt is enabled.
2-4	CMP_MODE	read/ write	000	Operation mode (if bit 4 = 0): 000 = Output remains constant 001 = Set output (After trigger event the oputput is always set to 1). 010 = Clear output (After trigger event the oputput is always set to 0). 011 = Toggle output after trigger event
4	OUT_TYP	read/ write	0	If the fourth bit of the operation mod register is set to 1 the output pin switches two times per period. Firstly, on each zero crossing and secondly on the configured maximum value (COMP_DAT). This mechanism enables the usage of the compare module for pulse width modulation (PWM).
2-4	CMP_MODE	read/ write	000	Opertaion mode (if bit 4 = 1): 100 = Output remains constant 101 = Output is set to 1 if timer value equals COMP_DAT -- output is set to 0 if timer value equals 0. 110 = Output is set to 0 if timer value equals COMP_DAT -- output is set to 1 if timer value equals 0.

Bit	Name	Access	Default	Description
				111 = Output is set to 1 if timer value equals COMP_DAT -- output is set to 0 if timer value equals 0.
5	CMP_EN_OUT	read/ write	0	If set to one the comparator output is enabled.
6	CMP_VAL_OUT	read	0	Comparator output bit.
7-17	x	read	0	Not used.

Table 3: CMP_CTRL register layout

3.3. Compare Value Register

Table 4: CMP_DAT register layout

Bit	Name	Access	Default	Description
0-17	CMP_DAT	read/ write	x	18 bit compare value

3.4. TIMER_CMP C-Header for Register Description

```
#ifndef __COMPARE_H
#define __COMPARE_H

#ifdef __cplusplus
extern "C" {
#endif

#define COMPARE_EN      (1 << 0)   // Compare Enable
#define COMPARE_EN_INT   (1 << 1)   // Compare Interrupt Enable
#define COMPARE_MODE      (1 << 2)   // Mode Bit 0

#define COMPARE_NON_FRQ   (COMPARE_MODE * 0) // Ausgang bleibt
gleich
#define COMPARE_SET_OUT   (COMPARE_MODE * 1) // Ausgang
setzen(=1)
#define COMPARE_CLEAR_OUT (COMPARE_MODE * 2) // Ausgang
zurücksetzen(=0)
#define COMPARE_TOGGLE_OUT (COMPARE_MODE * 3) // Ausgang
negieren
#define COMPARE_NON_IMP   (COMPARE_MODE * 4) // Ausgang bleibt
gleich
#define COMPARE_C0_N1   (COMPARE_MODE * 6) // Ausgang auf 0
wenn Timer = CMP_DAT ist -- Ausgang auf 1 wenn Timer = 0 ist.
```

```
#define COMPARE_C1_N0   (COMPARE_MODE * 7) // Ausgang auf 1
wenn Timer = CMP_DAT ist -- Ausgang auf 0 wenn Timer = 0 ist.

#define COMPARE_EN_OUT   (1 << 5)   // Compare Output Enable
#define COMPARE_VAL_OUT   (1 << 6)   // Compare Output Value

typedef struct cmp {
   volatile unsigned int CMP_CTRL; // (r/w)
   volatile unsigned int CMP_DAT; // (r/w)
} compare_regs_t;

#ifdef __cplusplus
}
#endif

#endif
```