SpartanMC

Timer Real Time Interrupt Module (timer-rti)

Table of Contents

List of Figures

List of Tables

Timer Real Time Interrupt Module (timer-rti)

The Timer RTI module can be used to divide the system clock frequency to a user defined periodic signal required by the application. For this purpose the Timer RTI provides a configurable prescaler. If enabled, the prescaler allows the usage of all powers of two between 2 and 32768 as prescaler value. The input of the prescaler block can be connected to the system clock, a dedicated DCM output or to the output of a previous timer module. The output of the prescaler could be connected to another timer module or could be used to generates an interrupt which for the application.

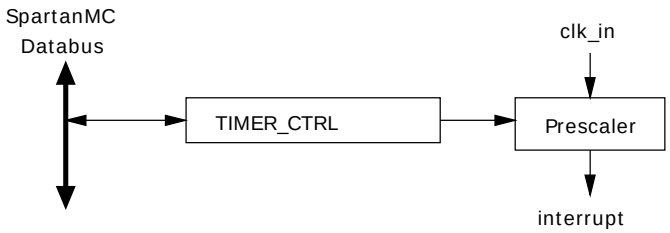

Figure 1: Timer RTI block diagram

Note: The timer RTI module could be used as stand alone peripheral or in connection with another timer module (used as input).

1. Interrupts

The peripheral generates a cyclic interrupt signals on the maximum value of the timer period.

2. Module Parameters

Table 1: Timer RTI 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 RTI Register Description

The timer RTI peripheral provides one 18 bit registers which are mapped to the SpartanMC address space e.g. 0x1A000 + BASE_ADR + Offset.

Table 2: TIMER RTI registers

Offset	Name	Access	Description
0	RTI_CTRL	read/ write	Specify the operation mode. (An access on this register clears the counter value)

3.2. RTI_CTRL Register

Table 3: RTI_CTRL register layout

Bit	Name	Access	Default	Description
0	RTI_EN	read/ write	0	If set to one the timer RTI logic is enabled.
1	RTI_EN_INT	read/ write	0	If set to one the timer RTI interrupt is enabled.
2-5	RTI_PRE_VAL	read/ write	0000	Specify the prescaler value: 0000 = 2^0 0001 = 2^1 0010 = 2^2 0011 = 2^3 0100 = 2^4 0101 = 2^5 0110 = 2^6 0111 = 2^7 1000 = 2^8 1001 = 2^9 1010 = 2^10 1011 = 2^11 1100 = 2^12 1101 = 2^13 1110 = 2^14

Bit	Name	Access	Default	Description
				1111 = 2^15
6-17	x	read	0	Not used.

Table 3: RTI_CTRL register layout

3.3. RTI C-Header for Register Description

```
#ifndef __RTI_H
#define __RTI_H

#ifdef __cplusplus
extern "C" {
#endif

#define RTI_EN         (1 << 0)
#define RTI_EN_INT      (1 << 1)
#define RTI_PRE_VAL      (1 << 2) // *0 fuer 2^0 bis *15 fuer
2^15

#define RTI_PRE_1      (RTI_PRE_VAL * 0)
#define RTI_PRE_2      (RTI_PRE_VAL * 1)
#define RTI_PRE_4      (RTI_PRE_VAL * 2)
#define RTI_PRE_8      (RTI_PRE_VAL * 3)
#define RTI_PRE_16      (RTI_PRE_VAL * 4)
#define RTI_PRE_32      (RTI_PRE_VAL * 5)
#define RTI_PRE_64      (RTI_PRE_VAL * 6)
#define RTI_PRE_128      (RTI_PRE_VAL * 7)
#define RTI_PRE_256      (RTI_PRE_VAL * 8)
#define RTI_PRE_512      (RTI_PRE_VAL * 9)
#define RTI_PRE_1024   (RTI_PRE_VAL * 10)
#define RTI_PRE_2048   (RTI_PRE_VAL * 11)
#define RTI_PRE_4096   (RTI_PRE_VAL * 12)
#define RTI_PRE_8192   (RTI_PRE_VAL * 13)
#define RTI_PRE_16384   (RTI_PRE_VAL * 14)
#define RTI_PRE_32765   (RTI_PRE_VAL * 15)

typedef struct rti {
   volatile unsigned int ctrl;
} rti_regs_t;

#ifdef __cplusplus
}
#endif
```

```
#endif
```