SpartanMC

Timer Watchdog Module (timer-wdt)

Table of Contents

List of Figures

List of Tables

Timer Watchdog Module (timer-wdt)

The timer watchdog module can be used for system monitoring purposes. Typically, the application has to clear the watchdog counter at regular intervals otherwise it generates an system reset or interrupt. The timer watchdog module requires two clock inputs: On the one hand CLK_1 which is used as timer input for the watchdog counter, on the other hand CLK_X which garantees the functionality of this module during system reset. (CLK_X has to be completly independend of the remaining SoC design.)

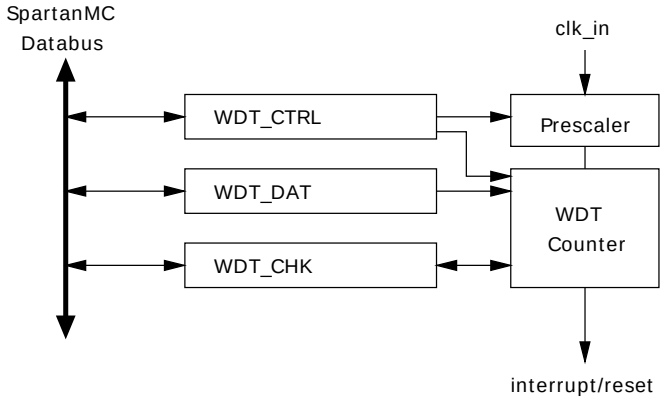

Figure 1: Watchdog timer block diagram

Note: The timer watchdog moudule can be used as stand alone peripheral or in connection with an basic timer module (used as counter clock input).

1. Usage

During the operation of the watchdog timer peripheral the value in WDT_DAT register is incremented continuously. If the value in WDT_DAT reaches a configured maximum value the peripheral performs a globel reset or an interrupt (with maximum priority). The reset of the watchdog counter is perfomend by writing a specific data word to WDT_CHK register. To determine the alert status of the watchdog module after a system reset bit 5 (ALARM bit) of WDT_CTRL has to be read .

2. Module Parameters

Table 1: Timer watchdog 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.**
WDT_RESET_PIN	0x12345	Code word to clear the watchdog timer.

3. Interrupts

If the watchdog counter reaches its maximum value an interrupt can be generated. The interrupt can be cleared by writing the WTD_CTRL register.

4. Peripheral Registers

4.1. Timer Watchdog Register Description

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

Table 2: Timer watchdog registers

Offset	Name	Access	Description
0	WDT_CTRL	read/ write	Specify the operation mode of the watchdog timer. (Each write access clears the ALARM bit)
1	WDT_DAT	read/ write	Maximum value of watchdog timer.
2	WDT_CHK	read	If read it contains the current value of the watchdog timer.
2	WDT_CHK	write	**Clears the watchdog timer if written with the configured code word.**

4.2. WDT_CTRL Register

Table 3: WDT_CTRL register layout

Bit	Name	Access	Default	Description
0	WDT_EN	read/ write	0	If set to one the watchdog timer is enabled.
1	WDT_EN_PRE	read/ write	0	If set to one the prescaler is enabled.
2-4	WDT_PRE_VAL	read/ write	000	Specify 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	WDT_ALARM	read/ write	0	Determines a watchdog alert. Set to null on each write access to this register.
6-17	x	read	0	Not used.

Table 3: WDT_CTRL register layout

4.3. WDT_DAT Register

Table 4: WDT maximum value register layout

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

4.4. WDT_CHK Register

Table 5: WDT counter register layout

Bit	Name	Access	Default	Description
0-17	Main Counter	read/ (write)	0	If read it contains the current watchdog counter value. If written with WDT_RESET_PIN it clears the watchdog counter value.

4.5. WDT C-Header for Register Description

```
#ifndef WDT_H_
#define WDT_H_

#define WDT_EN (1 « 0)
#define WDT_EN_INT (1 « 1)
#define WDT_PRE_VAL (1 « 2) // *0 fuer 2^1 bis *7 fuer 2^8

#define WDT_PRE_2 (WDT_PRE_VAL * 0)
#define WDT_PRE_4 (WDT_PRE_VAL * 1)
#define WDT_PRE_8 (WDT_PRE_VAL * 2)
#define WDT_PRE_16 (WDT_PRE_VAL * 3)
#define WDT_PRE_32 (WDT_PRE_VAL * 4)
#define WDT_PRE_64 (WDT_PRE_VAL * 5)
#define WDT_PRE_128 (WDT_PRE_VAL * 6)
#define WDT_PRE_256 (WDT_PRE_VAL * 7)

#define WDT_ALARM (1 « 5)

typedef struct wdt {
   volatile unsigned int control; // (r/w)
   volatile unsigned int limit;   // (r/w)
   volatile unsigned int val_rst; // (r = val / w PIN = rst)
} wdt_t;

#endif /* WDT_H_ */
```