AN205499

F²MC-16FX Family, Watchdog Timer And Watchdog Reset

This application note describes the functionality of the Watchdog Timer and Watchdog Reset

1 Introduction

This application note describes the functionality of the Watchdog Timer and Watchdog Reset and gives some examples.

The Watchdog is an up-counting Timer which has to be cleared in a given time interval. Otherwise a reset is performed. It can be used for restarting an application, if the program gets stuck.

1.1 Key Features

- 3 Clock Sources (RC Clock, Sub Clock or Main Clock)
- Timer clear by `0x00` or Byte Pattern
- Timer stopped in Stop Mode
- Timer can be disabled only by Reset, if once activated
- Timer Interval using external 4 MHz Crystal: ~128 µs to ~4.19 s
- Timer Interval using 32.768 kHz Sub Clock: 15.6 ms to 512 s
- Timer Interval using RC Oscillator (2 MHz/100 kHz): ~256 µs/5.1 ms to ~8.4 s/168 s

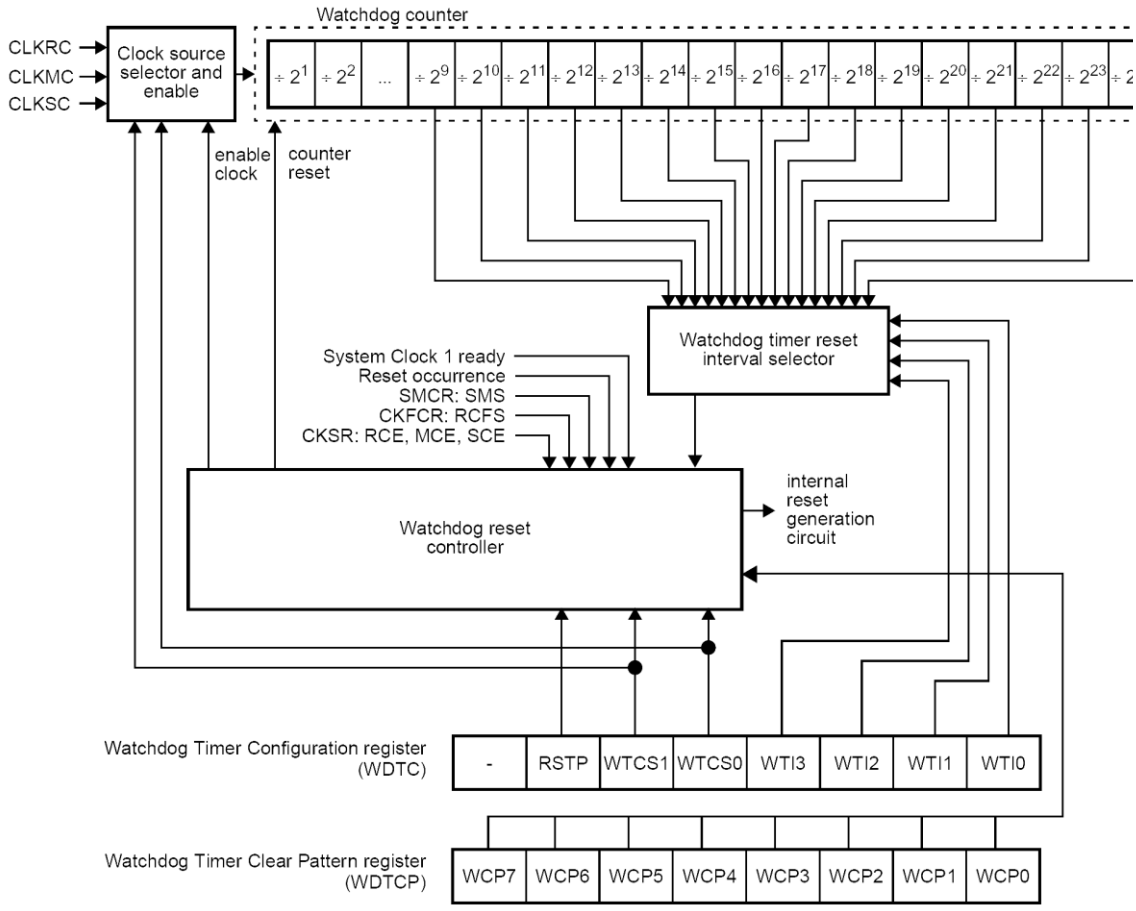
2 The Watchdog Timer

The basic functionality of the watchdog timer

2.1 Block Diagram

Figure 1 shows the internal block diagram of the Watchdog Timer.

Figure 1. Watchdog Timer block diagram



2.2 Registers

2.2.1 Watchdog Timer Configuration Register (WDTC)

Bit No.	Name	Explanation	Value	Operation
7	–	Undefined	-	-
6	RSTP	Watchdog Reset at Transition to Stop Mode	0	No Reset if transition to Stop Mode
			1	Reset if transition to Stop Mode
5, 4	WTCS1, WTCS0	Watchdog Timer Clock Selection	0, 0	CLKRC (RC frequency can be changed)
			0, 1	CLKRC (RC frequency change causes Reset)
			1, 0	CLKMC (Main Clock)
			1, 1	CLKSC (Sub Clock)
3, 2, 1, 0	WTI3, WTI2, WTI1, WTI0	Watchdog Timer Interval Selection	0, 0, 0, 0	2^9 / CLKWT
			0, 0, 0, 1	2^{10} / CLKWT
			0, 0, 1, 0	2^{11} / CLKWT
			0, 0, 1, 1	2^{12} / CLKWT
			0, 1, 0, 0	2^{13} / CLKWT
			0, 1, 0, 1	2^{14} / CLKWT
			0, 1, 1, 0	2^{15} / CLKWT
			0, 1, 1, 1	2^{16} / CLKWT
			1, 0, 0, 0	2^{17} / CLKWT
			1, 0, 0,1	2^{18} / CLKWT
			1, 0, 1, 0	2^{19} / CLKWT
			1, 0, 1,1	2^{20} / CLKWT
			1, 1, 0, 0	2^{21} / CLKWT
			1, 1, 0, 1	2^{22} / CLKWT
			1, 1, 1, 0	2^{23} / CLKWT
			1, 1, 1, 1	2^{24} / CLKWT

2.2.2 Watchdog Timer Clear Pattern register (WDTCP)

This 8-Bit Register is used for starting the Watchdog by writing a value to it. There are two ways of clearing the Watchdog.

- Clear by pattern: First writing a value unequal to 0x00. The next timer clear access has to be done by the inverted pattern (exclusive-or with 0xFF). Again the next clear has to be done by the original pattern, and so on. The corresponding software example is discussed in section 3.2.
- Clear by 0x00: Writing 0x00 to the WDTCP clears the watchdog. The corresponding software example is discussed in section 3.1.

Please note that both methods cannot be used together. Please also note that the Watchdog cannot be disabled once enabled. Only reset (any kind of) disables it.

2.2.3 Reset Cause and Clock Status Register (RCCSR/RCCSRC)

If a Watchdog reset has occurred it is indicated by the WRST-Bit (13) of the Reset Cause and Clock Status Register (RCCSR/RCCSRC). This register can be accessed at two addresses. A read access to address 0x040B (RCCSRC) clears all bits of the register after reading while a read access to address 0x040D (RCCSR) does not change the status of the register. The corresponding software example is discussed in section 3.3.

3 Watchdog Timer Examples

Following sections show sample code for Watchdog

3.1 Timer Clear by 0x00

```
void InitWatchdog(void)
{
    WDTC = 0x20;    // Clock Source: Main Clock, Interval=2^9/CLKWT=128 µs @ 4MHz
                    // of CLKMC
    WDTCP = 0x00;   // Clear by 0x00, enable Watchdog
}

void main(void)
{
    . . .

    InitWatchdog(); // Start Watchdog Timer

    . . .

    WDTCP = 0x00;   // Reset Watchdog Timer

    . . .
}
```

3.2 Timer Clear by Pattern

```
unsigned char wd_pattern;

void InitWatchdog(void)
{
    WDTC = 0x20;           // Clock Source: Main Clock, Interval=2^9/CLKWT=128 µs
                           // @ 4MHz of CLKMC
    wd_pattern = 0x55;
    WDTCP = wd_pattern;    // Clear by pattern, enable Watchdog
}

void WatchdogReset(void)
{
    wd_pattern ^= 0xFF;    // make complementary data
    WDTCP = wd_pattern;
}
```

```
void main(void)
{
    . . .

    InitWatchdog();        // Start Watchdog Timer

    . . .

    WatchdogReset();       // Reset Watchdog Timer

    . . .
}
```

3.3 Determining Watchdog Reset

```
void DetermineWtReset(void)
{
       unsigned char reset_cause;

       reset_cause = RCCSRC;      // Read reset cause, also clear it

       if (reset_cause & 0x20)    // Is reset caused by watchdog overflow?
       {
              // Reset caused by watchdog overflow,
              // Take appropriate action
       }
}
```

4 Additional Information

Information about Cypress Microcontrollers can be found on the following Internet page:
http://www.cypress.com/cypress-microcontrollers

The software example related to this application note is:

96340_wdtmr_wdreset s

It can be found on the following Internet page:

http://www.cypress.com/16lx

5 Document History

Document Title: AN205499 - F²MC-16FX Family, Watchdog Timer and Watchdog Reset

Document Number:002-05499

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	05/02/2006	Initial release
			12/18/2006	Reviewed the document and updated with review findings
			02/20/2007	Updated with re-review findings
			08/13/2007	Add more information about RCCSR, typos fixed
*A	5063392	NOFL	03/29/2016	Migrated Spansion Application Note MCU-AN-300206-E-V13 to Cypress format
*B	5869194	AESATP12	08/31/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709