W7100A Application Note

Watchdog Timer Interrupt

Version 1.0.1e

http://www.wiznet.co.kr

© Copyright 2013 WIZnet Co.,Ltd. All rights reserved.

Table of contents

© Copyright 2013 WIZnet Co.,Ltd. All rights reserved.

1 Introduction

This document includes the use of Watchdog timer interrupt and example codes of the W7100A

2 Watchdog Timer

The Figure 1 is the structure of Watchdog Timer. Set the WD1 and WD0 bit of CKCON register to select the Timeout period. If RWT bit (WDCON.0, Reset Watchdog Timer bit) is set to '1' before the timeout occurs, the watchdog timer is initialized and timeout does not occur. If the timeout occurs before RWT bit is set, the watchdog timer interrupt or Reset will be carried out. If you set the EWDI bit (EIE.4, Enable Watchdog Timer Interrupt) in advance, it is possible to create the Watchdog Interrupt. If the Interrupt occurs, WDIF bit is set to '1'.

Do not use the watchdog timer reset, because it can cause the abnormal operation of the RESET. Instead, we recommend you to use watchdog timer interrupt. You must set 'Timed access register (TA = 0xAA; TA = 0x55;)' before setting the watchdog timer registers.

For more details, refer to example source codes.

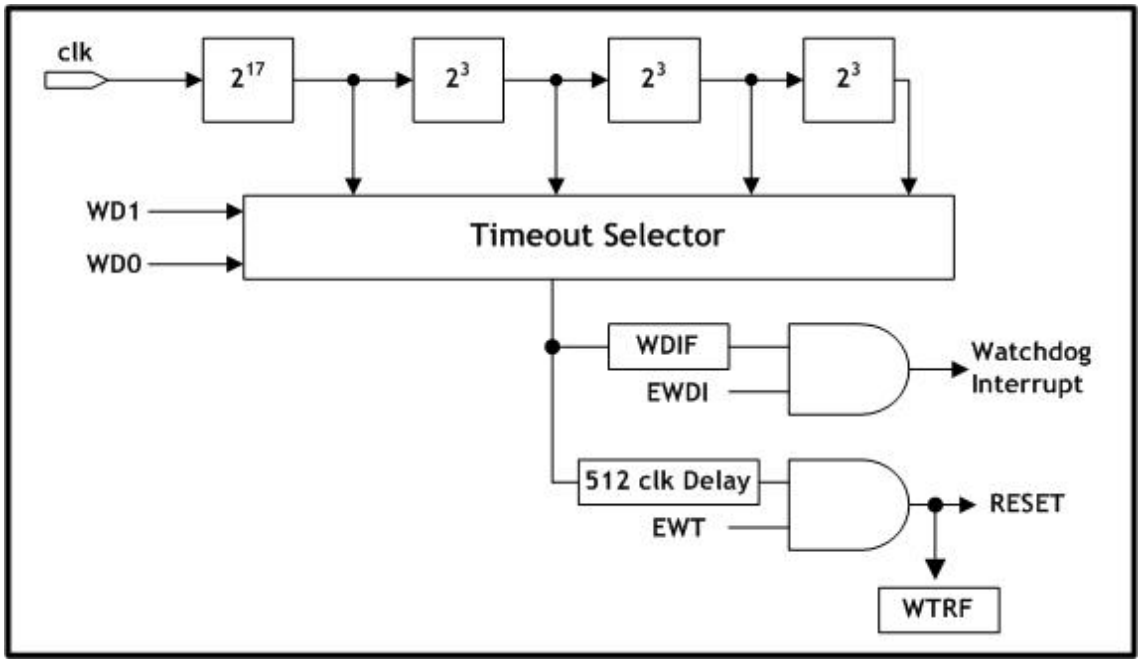

Figure 1. Watchdog Timer Structure

3 Watchdog Timer for Interrupt Application

This example shows how to enable watchdog timer interrupt and implements software reset in watchdog time interrupt service routine. This code will operate in the same way of watchdog timer reset.

3.1 Main Routine

```
void main()
{
        Init_iMCU();                    // Initialize iMCUW7100
        Init_Network();                 // Initialize Network Configuration
        wd_init();                      // Watchdog Timer Initialization
        EA = 1;                         // Global interrupt enable

        while(1)
        {
                printf(".");
                // Watchdog timer reset function (refer to section 3.4 Reset Timer)
                // if you want to avoid watchdog reset occur, call this timer reset function.
                /* wd_resetTimer(); */
        }
}
```

3.2 WDTimer Initialization

```
/* Watchdog timer Initialization */
void wd_init(void)
{
        uint8 tmpEA;
        tmpEA = EA;                                     // Backup global interrupt status
        EA = 0;                                         // Global interrupt disable
        CKCON |= 0x80;                                  // CKCON.7-6 = WD[1:0] : Watchdog Interval
        TA=0xAA;TA=0x55; EWT = 0;                       // Disable Watchdog Timer Reset (WDCON)
        TA=0xAA;TA=0x55; WDIF = 0;                      // Watchdog Interrupt flag clear (WDCON)
        EWDI = 1;                                       // Enable Watchdog Timer Interrupt (EIE)
        EA = tmpEA;                                     // Global interrupt restoration
}
```

First, set the watchdog timeout interval using the WD bits (CKCON.7-6, Watchdog Interval). The watchdog timer interrupt will be enabled as above. Clear the EWT bit

(WDCON.1, Enable Watchdog Timer Reset) and WDIF bit (WDCON.3) to initialize, and set EWDI bit (EIE.4) as '1' to enable the watchdog timer interrupt.

3.3 WDTimer Interrupt Service Routine

```
/* Watchdog timer Interrupt Service Routine */
void wd_isr(void) interrupt 12
{
        uint8 tmpEA;
        tmpEA = EA;                     // Backup global interrupt status
        EA = 0;                         // Global interrupt disable
        TA=0xAA; TA=0x55; WDIF = 0;     // Watchdog interrupt flag clear
        chip_reset();                   // Chip reset function (refer to section 3.5 Sys Reset)
        EA = tmpEA;                     // Global interrupt restoration
}
```

The hardware logic will automatically set WDIF bit whenever the interrupt occurs. This bit must be cleared by user software during exiting the interrupt service routine operates or before another interrupt generates.

3.4 Reset Timer

```
/* Watchdog timer reset function */
void wd_resetTimer(void)
{
        TA=0xAA;TA=0x55; RWT = 1;       // Reset Watchdog timer bit
}
```

If RWT is set before the timeout occurs, the timer will start over. If the timeout occurs without RWT being set, the watchdog interrupt occurs. The hardware logic will automatically clear RWT after software sets it.

3.5 System Reset

```
/* Reset function (Assembly code) */
void chip_reset(void)
{
#pragma ASM
                POP   ACC           // pop return address
                POP   ACC
                CLR   A             // push 0 as new
                PUSH ACC            // return address to stack
```

```
                    PUSH ACC
                    RETI                    // execute return of interrupt
#pragma ENDASM
}
```

The software reset code of jump to address 0[1] is not allowed in Keil µvision compiler. So, reset code must be written in assembly code as above. In order to, compile a file that contains the assembly reset code in Keil µvision compiler, follow the below steps.

1. Find 'Options' – [Properties] tab of the file including assembly reset codes.
2. Enable the 'Generate Assembler SRC File' checkbox.
3. Enable the 'Assemble SRC File' checkbox.

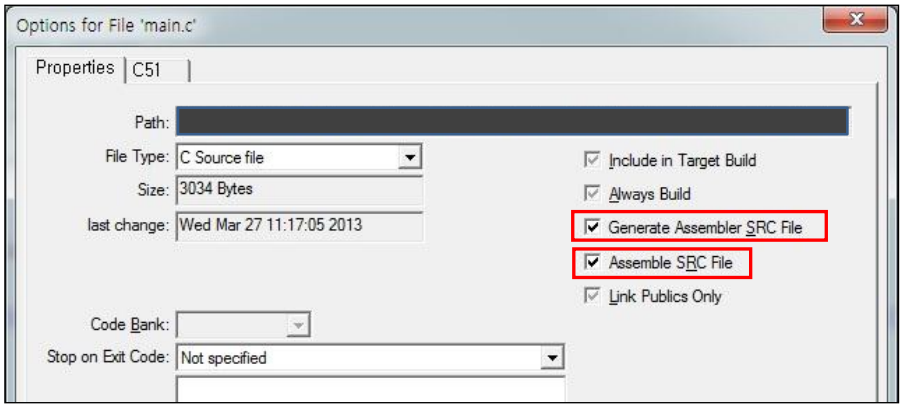

Figure 2. Options for Keil µvision project

[1] ((void (*)())0x0000)();

4 Demonstration

The example code will operates as follows:

First, this example performs initialization process and prints out the network configuration information to serial terminal using UART. And then it prints dot('.') to serial terminal repeatedly using printf function in while loop.

The user can verify that the reset performs repeatedly by a watchdog timer.

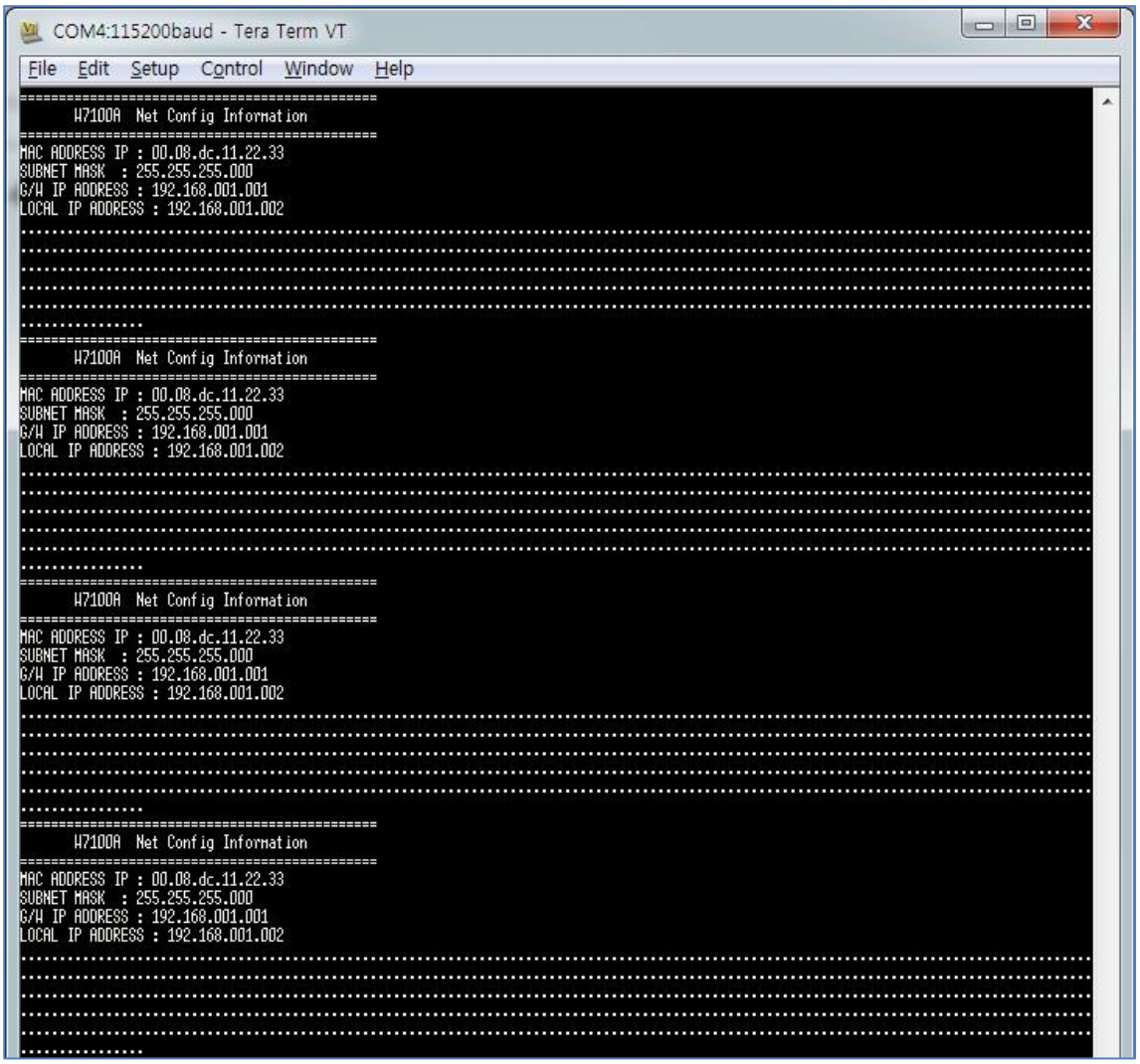

Figure 3. Watchdog Timer Interrupt Application Demonstration

Document History Information

Version	Date	Descriptions
Ver. 1.0.0	Apr. 2013	Application note released

Copyright Notice

Copyright 2013 WIZnet Co.,Ltd. All Rights Reserved.

Technical Support: support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr
For more information, visit our website at http://www.wiznet.co.kr