Nano102/112 Series Errata Sheet

Errata Sheet for 32-bit NuMicro™ Family

Document Information

Abstract	This errata sheet describes the functional problem known at the release date of this document.
Apply to	Nano102/112 Series.

Table of Contents

1 OVERVIEW 3

2 FUNCTIONAL PROBLEMS 4

2.1 User Config0 Bit 31 4

2.2 GPIO Debounce Function 5

2.3 User Config BOD Level Selection 9

2.4 BOD Reset Flag 10

2.5 HIRC Auto-Trim 11

1 Overview

Functional Problem	Description
User Config0 Bit 31	If the user config0 bit 31 is set as 0, Watchdog Timer (WDT) will be disabled after reset and cannot be enabled by software.
GPIO Debounce Function	GPIO Hardware debounce function cannot filter all noise.
User Config BOD Level Selection	BOD (Brown-out Detection) reset level set by user configuration does not always take effect.
BOD Reset Flag	BOD (Brown-out Detection) reset flag has no effect.
HIRC Auto-Trim	Auto-trim may fail in case that trim value spans from 0xFF to 0x100.

2 Functional Problems

2.1 User Config0 Bit 31

Description:

If Nuvoton related tools (e.g. ICP, ISP, NuGang Programmer, Nu-Link Driver for Keil or IAR) are used to program the user Config0 bit 31 and this bit is set to 0, Watchdog Timer (WDT) will be disabled after reset and cannot be enabled by software.

Problem:

If the User Config0 bit 31 is set as 0 by the above tools, it is intended to enable WDT after system reset, and software is not allowed to disable WDT during system operation. However, when the circuit is trying to enable WDT, there is no valid clock for WDT yet, so the WDT remains disabled state, and software cannot change its state during system operation unless this bit is set to 1 and system is reset.

Workaround:

If the system needs to use WDT, this bit must be set to 1.Therefore, if the Nano102/112 series is programmed by the following Nuvoton tools, please refer to the corresponding tool version or higher.

Tool	Version
ICP Programing Tool	V1.28.6386
ISP Programing Tool	V1.46
NuGang Programmer	V6.24
Nu-Link Driver for Keil	V1.28.6386
Nu-Link Driver for IAR	V1.28.6386

2.2 GPIO Debounce Function

Description:

GPIO de-bounce function failed.

Problem:

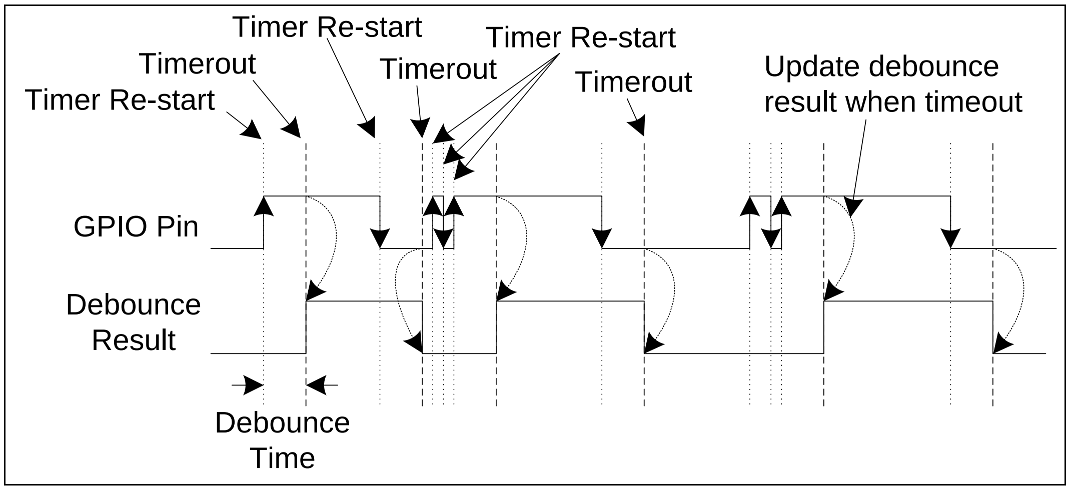
The GPIO de-bounce function cannot filter noise present at I/O pins completely.

Workaround:

Do not use the GPIO hardware debounce function. Use software debounce to filter unwanted input noise.

One software debounce method is to work with GPIO interrupt and a timer. The GPIO interrupt could be used to detect any I/O transient. The timer could be used to check if the I/O has transient within a period of time. By detect I/O transient with a period of time, user can skip all I/O transients shorter than specified period of time to achieve I/O debounce function.

The following figure shows the behavior of the software debounce flow. The timer is re-started whenever I/O transient. If there is no I/O transient within time-out period, the GPIO pin state at time-out will be assigned to a debounce result. If there is an I/O transient within time-out period, the timer will just be re-started without updating any debounce result.



A recommended software debounce steps are as follows:

1. Enable timer clock
2. Select timer clock source as LIRC (10 kHz)
3. Reset timer counter
4. Enable timer IRQ

5. Set GPIO pin as input
6. Enable GPIO pin interrupt with rising and falling interrupt type.
7. Enable GPIO IRQ
8. Prepare the GPIO IRQ handler:
   a. Reset timer counter
   b. Set timer time-out value (Debounce time)
   c. Start timer with one shot mode
   d. Clear GPIO interrupt flag
9. Prepare the timer IRQ handler:
   a. Get the debounce result
   b. Clear timer interrupt flag

The following code is an example to debounce pin PC.8. Timer 0 is used as the debounce timer. The debounce result is assigned to a global variable *g_u32Debounce*. PC.9 is used to monitor the result of *g_u32Debounce*.

```
#define DEBOUNCE_TIME           30     /* Debounce time in 0.1ms unit */

volatile uint32_t g_u32Debounce = 0;   /* Global variable for debounce result */

void DBNCE_Init(void)
{
    /* Enable Timer Clock Source */
    CLK->APBCLK |= CLK_APBCLK_TMR0_EN_Msk;

    /* Select Clock as LIRC 10kHz */
    CLK->CLKSEL1 = (CLK->CLKSEL1 & (~CLK_CLKSEL1_TMR0_S_Msk)) | CLK_CLKSEL1_TMR0_S_LIRC;

    /* Reset Timer */
    TIMER0->CTL = TIMER_CTL_SW_RST_Msk;
    while(TIMER0->CTL);

    /* Enable Timer IRQ */
    NVIC_EnableIRQ(TMR0_IRQn);


    /*****************************************************************
     Modify Here:
        All Debounce GPIO should be configured here.
```

```
          1. Set GPIO to be input mode
          2. Enable GPIO interrupt with rising + falling edge trigger.
          3. Enable GPIO IRQ
     ******************************************************************/

    /* Set GPIO Input */
    PC->PMD = (PC->PMD & (~(0x3<<8*2))) | (GPIO_PMD_INPUT << 8*2);

    /* Interrupt Type: Both Edge */
    PC->IMD |= (GPIO_IMD_EDGE << 8);
    PC->IER |= (1 << GP_IER_FIER8_Pos) | (1 << GP_IER_RIER8_Pos);

    /* Enable GPIO IRQ */
    NVIC_EnableIRQ(GPABC_IRQn);

}

void TMR0_IRQHandler(void)
{
    /*****************************************************************
     Modify Here:
       Debounce Ok when timer timeout.
       All GPIO debounce result should be return by global variable here.
     ******************************************************************/
     g_u32Debounce = PC8;

    /* Clear Timer Interrupt Flag */
    TIMER0->ISR = TIMER_ISR_TMR_IS_Msk;
}

void GPABC_IRQHandler(void)
{
    /* Reset Timer Counter */
    TIMER0->CTL = TIMER_CTL_SW_RST_Msk;
    while(TIMER0->CTL);

    /* Debounce Time */
    TIMER0->CMPR = DEBOUNCE_TIME - 1;

    /* Start Timer */
    TIMER0->IER |= TIMER_IER_TMR_IE_Msk;
    TIMER0->CTL = TIMER_CTL_TMR_EN_Msk | TIMER_ONESHOT_MODE;
    while(TIMER0->ISR & TIMER_ISR_TMR_IS_Msk);

    /*****************************************************************
     Modify Here:
       Clear relative GPIO interrupt flag here
     *****************************************************************/
    GPIO_CLR_INT_FLAG(PC, (1 << 8));
}

int main(void)
{
  /* Unlock Protected Registers */
  SYS_UnlockReg();

  /* Configure PC.9 as Output mode */
  PC->PMD = (PC->PMD & (~GP_PMD_PMD9_Msk)) | (GPIO_PMD_OUTPUT << GP_PMD_PMD9_Pos);

  DBNCE_Init();

  while(1)
  {
    PC9 = g_u32Debounce;
  }
}
```

2.3 User Config BOD Level Selection

Description:

BOD (Brown-out Detection) reset level set by user configuration does not always take effect.

Problem:

The BOD reset level in user configuration has no effect if the system reset source is POR (Power-on Reset) or BOD.

Workaround:

If the BOD reset function is required, configure BODCTL register to select the desired BOD reset level at the beginning of user program.

2.4 BOD Reset Flag

Description:

Starting from version F, the BOD (Brown-out Detection) reset flag, RSTS_BOD (RST_SRC[4]), is always read back as 0 even though the BOD function itself works properly.

Problem:

Since this bit is always read as 0, it cannot be used to tell if the system is reset by a BOD event or not.

Workaround:

Check RST_SRC register, and if none of the reset flag is set, the reset source must be BOD. Following code is an implementation supports all versions of Nano102/112 series MCUs.

```
int main(void)
{
    if((SYS->RST_SRC == 0) || (SYS->RST_SRC & SYS_RST_SRC_RSTS_BOD_Msk)) {
        /* System reset by BOD */
    }
    /* Clear system reset source flag */
    SYS->RST_SRC = SYS->RST_SRC;
}
```

2.5 HIRC Auto-Trim

Description:

This chip supports auto-trim function, HIRC trim, according to the accurate LXT (32.768 kHz crystal oscillator), which automatically gets accurate HIRC output frequency, 1% deviation within all temperature ranges.

An internal trim value determines the degree of HIRC trim. Increasing the trim value can obtain about 0.2% HIRC frequency adjustment. However, when the trim value steps from 0xFF to 0x100, it can induce a much larger change of HIRC frequency, about 2.0% ~ 2.5%.

Problem:

While performing auto-trim by writing a non-zero value to TRIM_SEL(IRCTRIMCTL[1:0]), if the trim step spans from 0xFF to 0x100, a valid trim value may not be found. For example, suppose that the best trim value is 0xFF plus one step (+ 0.2%) and the next step 0x100 is actually increased 2%, the auto-trim function can never succeed in this case.

Workaround:

Once falling into this condition, there is no workaround to make the auto-trim succeed. It is suggested to always check TRIM_FAIL_INT(IRCTRIMINT[1]). Once the auto-trim failed, an immediate retry is likely to be failed. In this situation, it is suggested to disable auto-trim until the temperature is obviously changed.

Revision History

Date	Revision	Description
2015.01.14	1.00	1. Initially issued.
2016.07.06	1.01	1. Added GPIO debounce issue and software debounce method. 2. Added BOD problem
2017.04.18	1.02	1. Added BOD reset flag issue.
2022.08.03	1.03	1. Added HIRC auto-trim issue.

Important Notice