CX32L003 Application Note 3

DeepSleep Mode AWK Wake-Up Causes The Chip Reset

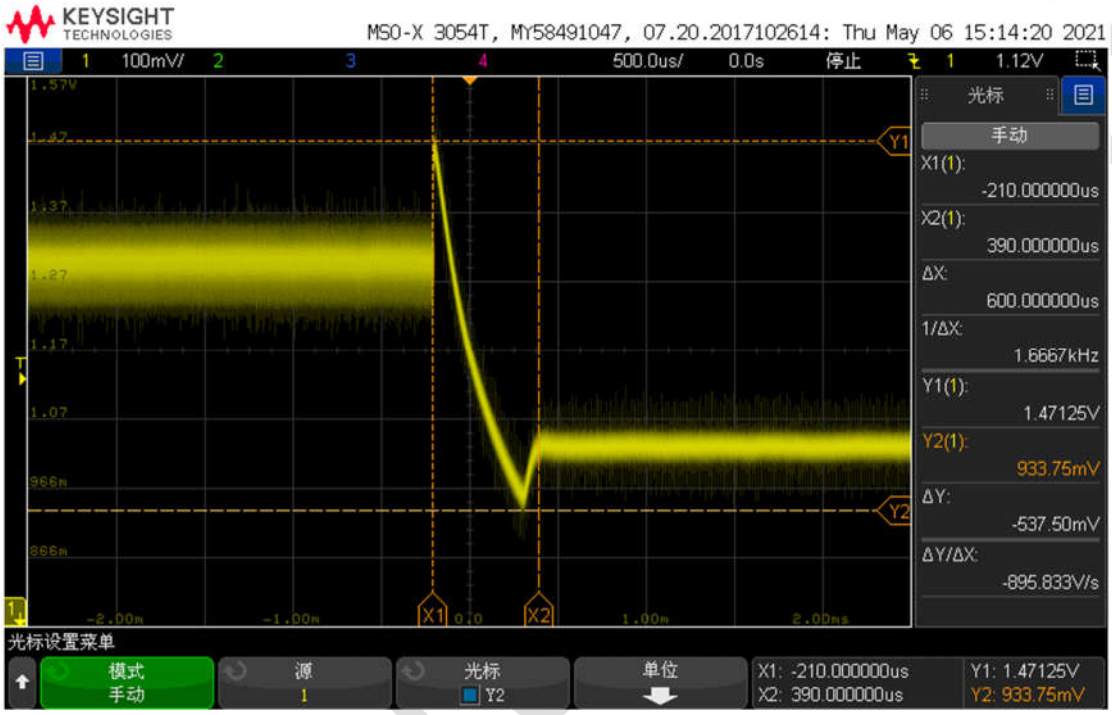
Rev.0.1, Date:2021/6/30

1. 问题描述:

在进入 Deepsleep 后，通过 AWK 唤醒芯片，会出现芯片复位的现象。

2. 原因分析:

在进入 Deepsleep 的时候，Vcore 电压上会有个上升和下降再到稳定（进入 Deepsleep Mode）的过程。如下图所示:



在电压峰值的时候，会干扰到 AWK 的计数寄存器，导致寄存器溢出，从而触发唤醒。然而此时 Vcore 电压不稳，容易造成芯片复位。

3. 解决办法:

在进入 Deepsleep Mode 之前主动关掉外设时钟，并且将时钟源切换到 LIRC。（目的是降低峰值电压）。

4. 修订历史

Revision	Date	ChangeLog
0.1	2021/6/30	Initial Version

附，相关代码如下：

```
while (1)
{

    all_clock_off();
    wait_100nop();
    HAL_PWR_EnterDEEPSLEEPMode(); // 关闭相关外设后进入睡眠
    wait_100nop();
    all_clock_on();
```

```
void all_clock_off(void)
{
    __HAL_RCC_GPIOA_CLK_DISABLE() ;
    __HAL_RCC_GPIOB_CLK_DISABLE() ;
    __HAL_RCC_GPIOC_CLK_DISABLE() ;
    __HAL_RCC_GPIOD_CLK_DISABLE() ;

    __HAL_RCC_UART0_CLK_DISABLE() ;
    __HAL_RCC_UART1_CLK_DISABLE() ;
    __HAL_RCC_TIM1_CLK_DISABLE() ;
    __HAL_RCC_ADC_CLK_DISABLE() ;

    RCC_ClkInitTypeDef  RCC_ClkInitStruct = {0};
    RCC_OscInitTypeDef  RCC_OscInitStruct = {0};

    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HIRC | RCC_OSCILLATORTYPE_LIRC;
    RCC_OscInitStruct.HIRCState = RCC_HIRC_OFF;
    RCC_OscInitStruct.HIRCCalibrationValue = RCC_HIRCCALIBRATION_24M;
    RCC_OscInitStruct.LIRCState = RCC_LIRC_ON;
    RCC_OscInitStruct.HIRCCalibrationValue = RCC_LIRCCALIBRATION_32K;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct)  != HAL_OK)
    {
        Error_Handler();
    }

    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_PCLK;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_LIRC;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_HCLK_DIV1;
    RCC_ClkInitStruct.APBCLKDivider = RCC_PCLK_DIV1;


    if( HAL_RCC_ClockConfig (&RCC_ClkInitStruct)!=HAL_OK )
    {
        Error_Handler();
    }
}
```

```
void wait_100nop(void)
{
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
    __nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
}
```

```
void all_clock_on(void)
{
    __HAL_RCC_GPIOA_CLK_ENABLE() ;
    __HAL_RCC_GPIOB_CLK_ENABLE() ;
    __HAL_RCC_GPIOC_CLK_ENABLE() ;
    __HAL_RCC_GPIOD_CLK_ENABLE() ;

    __HAL_RCC_UART0_CLK_ENABLE() ;
    __HAL_RCC_UART1_CLK_ENABLE() ;
    __HAL_RCC_TIM1_CLK_ENABLE() ;
    __HAL_RCC_ADC_CLK_ENABLE() ;
    RCC_ClkInitTypeDef  RCC_ClkInitStruct = {0};
    RCC_OscInitTypeDef  RCC_OscInitStruct = {0};

    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HIRC | RCC_OSCILLATORTYPE_LIRC;
    RCC_OscInitStruct.HIRCState = RCC_HIRC_ON;
    RCC_OscInitStruct.HIRCCalibrationValue = RCC_HIRCCALIBRATION_24M;
    RCC_OscInitStruct.LIRCState = RCC_LIRC_OFF;
    RCC_OscInitStruct.HIRCCalibrationValue = RCC_LIRCCALIBRATION_32K;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }

    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_PCLK;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HIRC;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_HCLK_DIV1;
    RCC_ClkInitStruct.APBCLKDivider = RCC_PCLK_DIV1;

    uwTickFreq = HAL_TICK_FREQ_1KHZ;
    if( HAL_RCC_ClockConfig (&RCC_ClkInitStruct)!=HAL_OK )
    {
        Error_Handler();
    }
}
```