SPC1168 WDT 使用指南

概述

看门狗定时器（WDT）可以在系统因软件错误运行失败的情况下，重新恢复系统的控制，以增加应用的可靠性。WDT 在计数器计数到一个给定的超时数值时，可以产生系统复位或者一个中断。SPC11X8/SPC11X8 有两个 32 位看门狗定时器。WDT 的寄存器可由 CPU 通过 AHB 总线进行控制。

注意： 本文档主要以 SPC1168 为例进行介绍。

目录

图片列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-06-08	CanChai	Outdated	首次发布。
C/0	2024-03-26	Jiali Zhou	Released	修改排版格式。

术语或缩写

术语或缩写	描述
/	/

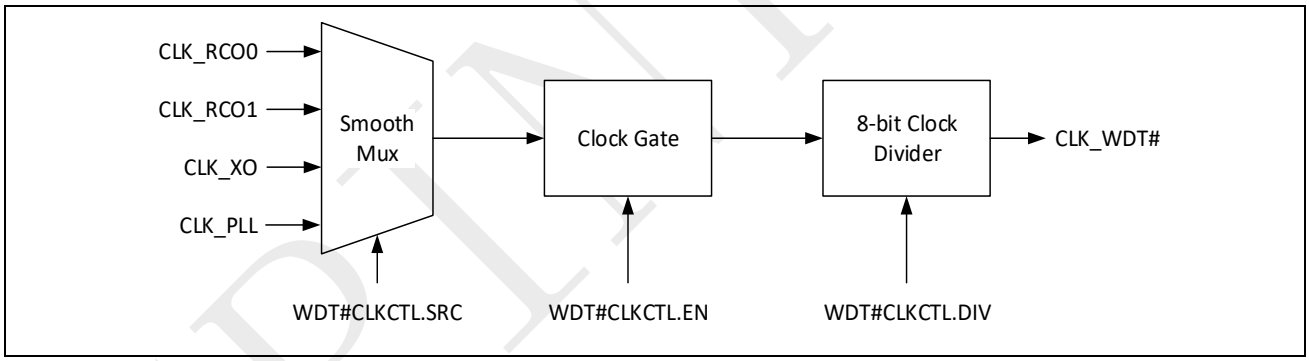
1 WDT 描述

1.1 WDT 特性

SPC11X8/SPC11X8 器件内置 2 个看门狗，每一个看门狗都有如下特性：

- APB 总线寄存器接口
- 专属 WDT 时钟，其时钟可以选择内部 RC 振荡器、外部振荡器或者锁相环时钟，由相应的平滑选择、门控和分频产生，如图 1-1 所示。
- 32 位向下计数器
- 当发生计数超时事件时，可配置产生复位或者中断
- 当第一次超时发生时，只是会产生一个中断。只有在第二次超时发生时，上一次的中断没有被清除且系统复位功能被使能，那么 WDT 会产生一个系统复位请求
- 在调试模式下，看门狗计数器可以被冻结或者自由运行

图 1-1：WDT 时钟控制框图



1.2 WDT0 与 WDT1 区别

WDT0 与 WDT1 的寄存器的特性及基本架构完全相同，使用时只需要注意以下区别：

- WDT0 产生的中断信号接到了 NVIC 的 NMI 中断，而 WDT1 产生的中断信号接到了 NVIC 的 WDT1 中断。
- WDT0 的时钟源默认选择的是 RCO0；WDT1 的时钟源默认选择的是 RCO1。使用者可根据实际需要修改时钟源。

2 WDT 代码实例

2.1 WDT 超时后产生中断

在本示例中，WDT 将根据所设定的时间（500ms）计时，超时后产生中断。在中断服务程序中需要清除 WDT 中断标志。

如下 WDT 配置步骤可供开发者参考：

- 调用 WDT_Init()初始化看门狗
- 调用 NVIC_EnableIRQ()使能 NVIC 中断，如果使用 WDT0，则不需要该步骤，因为 WDT0 的中断请求信号被连接到 NMI 中断
- 调用 WDT_EnableInt()，以启动计数器和使能中断
- 中断服务函数中清除 WDT 中断标志

Example Code

```
int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /*Disable flash write access after flash operation had done*/
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO34/35 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);
```

```
  /* Init WDT0 and set count time to 500ms */
  WDT_Init(WDT0, 500);

  /* Enable the counter and the interrupt */
  WDT_EnableInt(WDT0);

  while(1)
  {

  }
}


/* In our solution, WDT0 INT connected to SYS NMI */
void NMI_Handler()
{
  /* Clear INT, So that SYS would restart */
  WDT_ClearInt(WDT0);

  /* Set WDT0 Stop count and disable the INT */
  WDT_DisableInt(WDT0);

  printf("WDT INT test OK\n");
}
```

如果需要使 WDT 按所设定的时间周期性地产生中断，NMI_Handler 函数可修改为：

Example Code

```
/* In our solution, WDT0 INT connected to SYS NMI */
void NMI_Handler()
{
  /* Clear INT, So that SYS would restart */
  WDT_ClearInt(WDT0);

  printf("WDT INT test OK\n");
}
```

2.2 WDT 超时后产生复位

在本示例中，WDT 将根据所设定的时间（500ms）计时，超时后产生中断，本例程只在第一次进中断后清中断标志，之后再进中断都不清中断标志。WDT 如果在超时发生时，上一次的中断没有被清除且系统复位功能被使能，那么 WDT 会产生一个系统复位请求，运行结果就是每隔 1500ms 会复位 MCU。

如下 WDT 配置步骤可供开发者参考：

- 调用 WDT_Init()初始化看门狗（函数内部使能了系统复位请求）
- 调用 NVIC_EnableIRQ()使能 NVIC 中断，如果使用 WDT0，则不需要该步骤，因为 WDT0 的中断请求信号被连接到 NMI 中断
- 调用 WDT_EnableInt()，以启动计数器和使能中断
- 中断服务函数中不清除 WDT 中断标志

Example Code

```
uint16_t iFlag = 0;  /* Used for judging if INT had happened */

int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO34/35 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);
```

```
  /* Init WDT1 */
  WDT_Init(WDT1, 500);

  /* Enable the NVIC INT of WDT1 */
  NVIC_EnableIRQ(WDT1_IRQn);

  /* Enable the counter and the interrupt */
  WDT_EnableInt(WDT1);

  while(1)
  {

  }
}

void WDT1_IRQHandler()
{
  iFlag++;

  /* Do not clear INT after the first INT to make SYS reset */
  if(iFlag <= 1)
  {
    WDT_ClearInt(WDT1);
  }
}
```

2.3 WDT 超时前进行喂狗

用户程序可以在 WDT 超时前清零 WDT 计数器，从而避免进入中断或产生复位。

在本示例中，WDT 将根据所设定的时间（500ms）计时，在超时之前重新喂狗以清零计数器，使其不进入中断。

如下 WDT 配置步骤可供开发者参考：

- 调用 WDT_Init()初始化看门狗
- 调用 NVIC_EnableIRQ()使能 NVIC 中断，如果使用 WDT0，则不需要该步骤，因为 WDT0 的中断请求信号被连接到 NMI 中断
- 调用 WDT_EnableInt()，以启动计数器和使能中断

- 中断服务函数中清除 WDT 中断标志
- 在合适位置调用 WDT_FeedDog()清零计数器

Example Code

```
int iFlag = 0;
uint32_t u32count = 100;

#define  TimeMs  500  /*500ms*/

int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO34/35 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Init WDT1 and set count time to 500ms */
  WDT_Init(WDT1, TimeMs);

  /* Enable the NVIC INT of WDT1 */
  NVIC_EnableIRQ(WDT1_IRQn);

  /* Enable the counter and the interrupt */
  WDT_EnableInt(WDT1);

  while(u32count--)
```

```
  {
      /* Wait for 60ms, less then TimeMs, to prevent WDT1 process INT to
CPU */
      Delay_Ms(30);

      /* Feed dog */
      WDT_FeedDog(WDT1);

      /* Disable INT at the last count */
      if(u32count == 1)
        WDT_DisableInt(WDT1);
  }

  if(iFlag)
    printf("WDT Feed DOG test FAIL\n");
  else
    printf("WDT Feed DOG test PASS\n");

  while(1)
  {

  }
}

void WDT1_IRQHandler()
{
  iFlag++;

  /* Clear INT, So that SYS would restart */
  WDT_ClearInt(WDT1);

  /* Set WDT1 Stop count and disable the INT */
  WDT_DisableInt(WDT1);
}
```

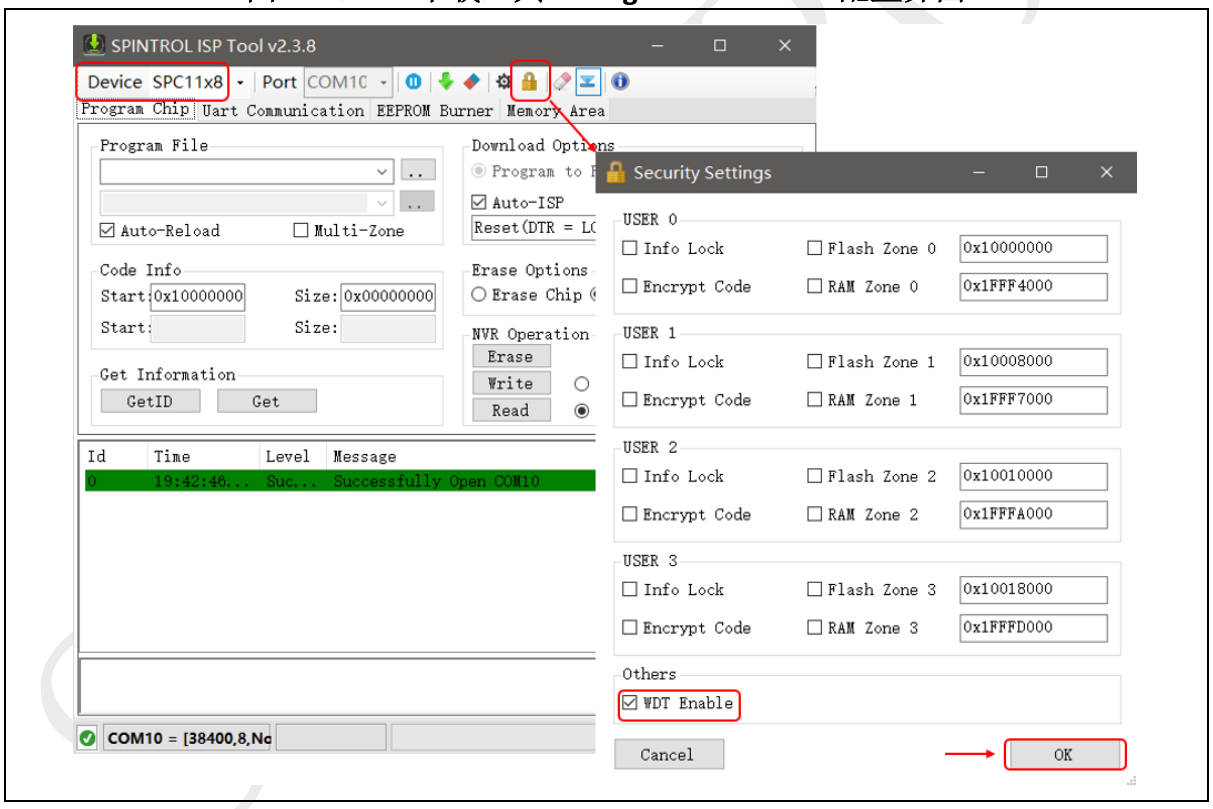
2.4 硬件开启看门狗

看门狗除了通过设置控制寄存器进行软件启动方式外，还有硬件启动方式。

在 Flash 存储器中存在 NVR 存储区块，该块中包含有配置字（Configuration Words）存储区，在配置字存储区中的 WDT_ENABLE 配置字可以实现 WDT 的开机使能，其地址为 0x11000700。当 WDT_ENABLE 字段为 0xFFFFFFFF 时，则当芯片启动时关闭 Watchdog，若为其他值，则当芯片启动时启动 Watchdog（WDT0 和 WDT1）。硬件启动 WDT 时，默认的超时时间大约是 2 秒（即触发复位的超时时间为 4 秒）。

SPINTROL 提供的 ISP Tool 下载工具可以帮助用户设置 Configuration Words，界面如图 2-1 所示，本节以 SPC11x8 为例。勾选“WDT Enable”后，点击“OK”。

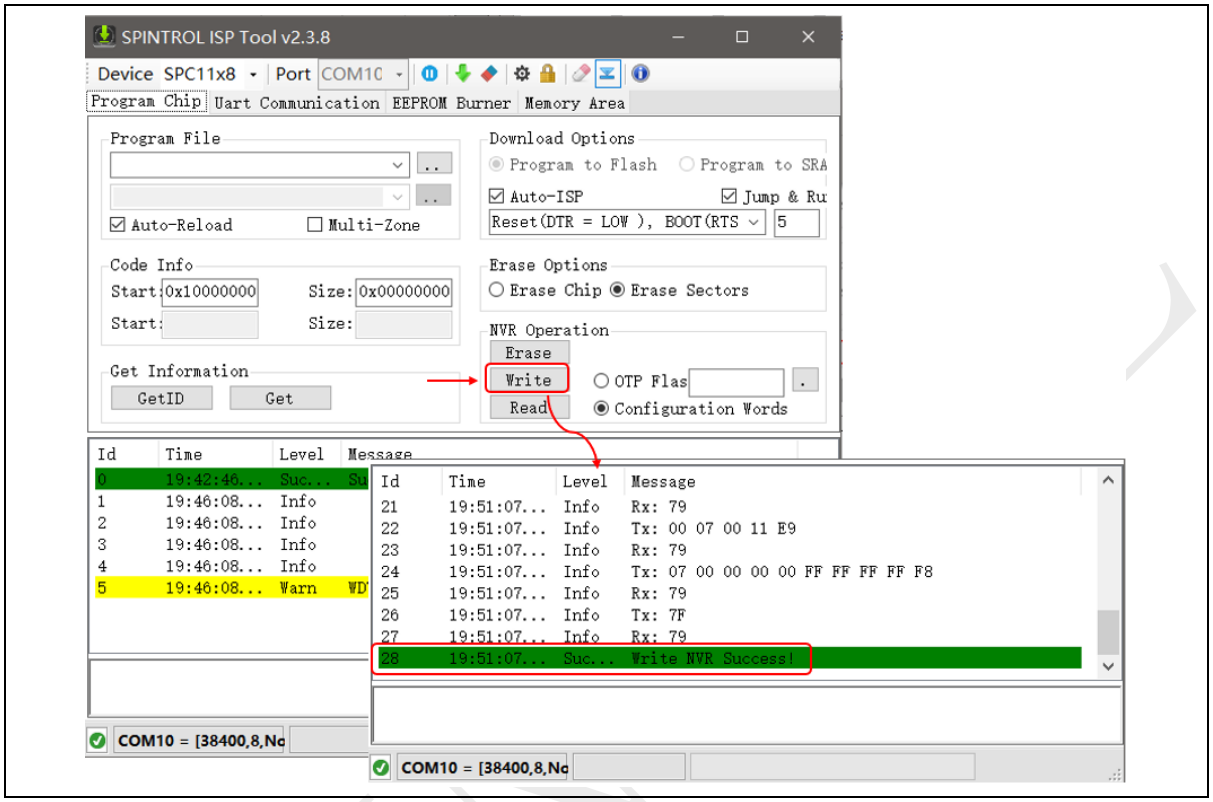
图 2-1：ISP 下载工具 Configuration Words 配置界面



注意： 使用 ISP 工具设置配置字时，需将 BOOT 和 TRSTn 均接低电平，即工作在 ISP（In System Programming）模式，启动引导程序使用 UART 接口对 Flash 存储器进行重新编程。在这个过程中，GPIO34 被配置为 UART_TXD 功能；GPIO35 被配置为 UART_RXD 功能。

如图 2-2 所示，点击“Write”按钮，即可将非 0xFFFFFFFF 的字段写入 WDT_ENABLE 配置字。当芯片启动时就会硬件启动 Watchdog。

图 2-2：Write NVR 界面



注意： 新写入的配置字需要在芯片复位后才能生效。

硬件启动 WDT 后，如果不喂狗，大约 4 秒后芯片将复位。用户可在合适位置加入喂狗代码或者在中断函数内清除中断标志，避免复位。参考代码如下：

Example Code

```
int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();
```

```
  /*
   * Init the UART
   *
   * 1.Set the GPIO34/35 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Enable write access to the protected WDT registers */
  WDT_WALLOW(WDT1);

  /* Enable write access to the protected WDT registers */
  WDT_WALLOW(WDT0);

  while(1)
  {
      /* Feed dog */
      WDT_FeedDog(WDT1);
  }
}


void NMI_Handler(void)
{
  /* Clear INT, So that SYS would restart */
  WDT_ClearInt(WDT0);
}
```