AX Watchdog 使用说明

文档版本：V1.0

发布日期：2024/01/25

目 录

前 言 3

修订历史 4

1 概述 5

2 特点 6

3 功能描述 7

4 工作方式 8

5 寄存器描述 9

5.1 common_sys 寄存基描述 9

5.2 periph_sys 寄存器描述 9

5.3 看门狗内部寄存器描述 10

6 内核配置 21

6.1 设备树配置 21

6.2 功能选择 22

权利声明

注意

您购买的产品、服务或特性等应受商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非商业合同另有约定，本公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

前 言

适用产品

AX620E

适读人群

- 软件开发工程师
- 技术工程师

符号与格式定义

符号/格式	说明
`xxx`	表示您可以执行的命令行。
☞ 说明/备注：	表示您在使用产品的过程中，我们向您说明的事项。
！ 注意：	表示您在使用产品的过程中，需要您特别注意的事项。

修订历史

文档版本	发布时间	修订说明
V0.5	2024/01/11	文档初版
V1.0	2024/01/25	根据评审意见优化文档描述

1 概述

芯片支持 3 个 WDT（Watchdog Timer），从 WDT0 到 WDT2，其中 WDT0 和 WDT2 支持复位系统，目前 WDT0 和 WDT2 给 CPU 专用，用来防止系统锁死。程序开始运行后，WDT 开始倒计开始。如果程序运行正常，过一段时间 CPU 应发出指令让 WDT 从初始值重新开始倒计时。如果 WDT 倒数到 0 时（即 timeout）则 WDT 向 CPU 发出中断，WDT 从初始值开始重新倒计时，第二次倒数到 0，CPU 还没有发出命令，则认为系统故障，WDT 发出系统复位。目前的 Linux 使用了 WDT0 和 WDT2，默认配置在中断中进行喂狗，同时支持配置内核线程喂狗，或者选择一个 WDT 由应用进行喂狗，配置方式见 1.6 节。

首先，WDT 从超时周期值（0x000F_FFFF，默认值）开始倒计时（默认以 32K 的时钟，每个 clk 减 1，项目中选择的是 24M 的时钟），超时周期值可通过寄存器 TORR 配置。WDT 会在发生第一次超时，首先产生一个到 CPU 的中断，第二次超时产生系统复位。CPU 可向寄存器 INTR_EOI 写 1 清除中断，如果 WDT 发生超时前，CPU 向寄存器 KICK 先写入 0x61696370 后再写入 0，则 WDT 重新计数，即又从超时周期值开始倒计时。

2 特点

- 内部具有一个 32bit 减法计数器，其中高 16bit 可编程，低 16bit 固定为 0xFFFF。
- 支持超时周期值可通过寄存器配置。
- 支持写入寄存器来喂狗。
- 支持超时中断产生。
- 支持复位信号产生。
- 支持清除中断。

3 功能描述

WDT 的基本功能为监控系统运行。系统总线给 WDT 配置寄存器参数值，WDT 定时发出中断请求给系统，并在系统没有响应中断的情况下，发出 WDT 复位信号，使系统复位，达到监控系统运行的目的。

WDT 在系统中的功能框图如下：

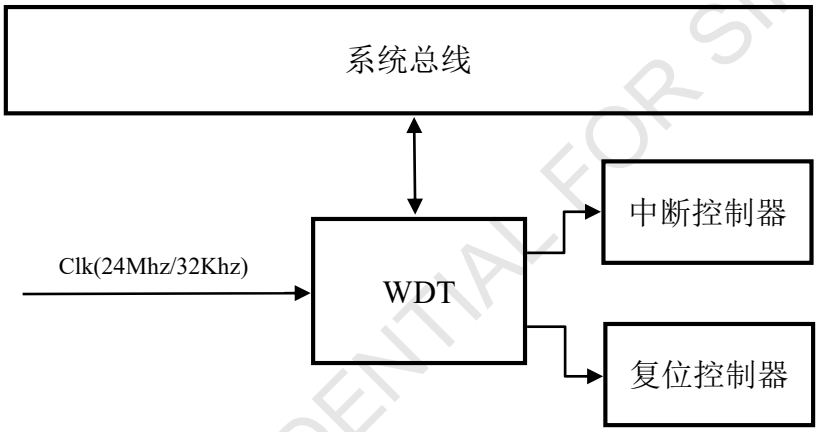

图3-1 WDT 在系统中的功能框图

说明

WDT 的运行基于 32bit 的减法计数器，从计数初始值（默认为 0x000F_FFFF，寄存器可配）开始递减计数，计数值在每个计数时钟的上升沿减 1。当计数值第一次递减到 0 时，WDT 将产生一个中断，然后计数器又开始从计数初始值递减计数。如果当计数器第二次递减到 0 时，系统仍未进行喂狗操作（也称 kicking the dog 或者 service the dog），则 WDT 产生系统复位信号，使整个系统复位；如果当计数器第二次递减到 0 前，系统进行了喂狗，则计数器又开始从计数初始值递减计数，如此循环。

32K 时钟可配置时间范围：2.05s*2 ~ 134217.73s*2。

24M 时钟可配置时间范围：2.74ms*2 ~ 178.96s*2。

4 工作方式

首先，WDT 从超时周期值（0x000F_FFFF，默认值）开始倒计时（默认以 32K 的时钟，每个 clk 减 1，项目中选择的是 24M 的时钟），超时周期值可通过寄存器 TORR 配置。

WDT 会在发生第一次超时，首先产生一个到 CPU 的中断，第二次超时产生系统复位。CPU 可向寄存器 INTR_EOI 写 1 清除中断，如果 WDT 发生超时前，CPU 向寄存器 KICK 先写入 0x61696370 后再写入 0，则 WDT 重启，即又从超时周期值开始倒计时。

! 注意：当时钟为 32Khz 时，KICK 寄存器写入 0x61696370 后需要延时最少 40us 再写入 0。

5 寄存器描述

注：_set, _clr 寄存器位为位操作寄存器，方便置位对应的寄存器位

例如 clk_comm_wdt_sel_set,(0x0421_0FFF +0x28)寄存器的 bit12，对此寄存器位写 1，其他 bit 全写 0，则可以把 (0x0421_0FFF+0x00)的第 12 位，即 clk_comm_wdt_sel 置 1。

例如 clk_comm_wdt_sel_clr,(0x0421_0FFF+0x2C)寄存器的 bit12，对此寄存器位写 1，其他 bit 全写 0，则可以把 (0x0421_0FFF+0x00)的第 12 位，即 clk_comm_wdt_sel 清 0。

5.1 common_sys 寄存基描述

common_sys 的基地址：0x0234_0000

表5-1 common_sys 寄存基描述

偏移地址	名称	位域	描述	说明
0xA8	abort_cfg	[9] [7]	wdt2 abort enable wdt0 abort enable	0: default 关闭复位系统功能 1: 使能复位系统功能
0xAC	abort_cfg_set	[9] [7]	见_set 描述	见_set 描述
0xB0	abort_cfg_clr	[9] [7]	见_clr 描述	见_clr 描述

5.2 periph_sys 寄存器描述

periph_sys 的基地址：0x0487_0000

表5-2 periph_sys 寄存基描述

偏移地址	名称	位域	描述	说明
0x00	clk_mux_0	[20] [19]	Wdt2 clk_wdt_sel Wdt0 clk_wdt_sel	0: 32K, default 1: 24M
0x4	clk_eb_0	[15] [14]	clk_wdt2_eb clk_wdt0_eb	clock clk_wdt enable control: 1'b0 : disable 1'b1 : enable
0x10	clk_eb_3	[20] [19]	pclk_wdt2_eb pclk_wdt0_eb	Clock pclk_wdt gate enable control: 1'b0 : disable 1'b1 : enable
0x24	sw_rst_3	[3] [2] [1] [0]	wdt2_sw_rst wdt2_sw_prst wdt0_sw_rst wdt0_sw_prst	Software reset for signal : rst_wdt2_n 1'b1 : disable 1'b0 : enable 对 wdt 内部寄存器进行复位；

5.3 看门狗内部寄存器描述

WDT0 和 WDT2 是 CPU 专用，其基地址：0x4840000，0x6040000。

WDT1 是给 RISCV 专用，其基地址：0x2230000。

WDT 内部的寄存器描述：

WDT_EN

偏移地址：0x00

表5-3 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:1	RSVD_WDT_EN	R	For reserved	Reserved
0	WDT_EN	R/W	WDT enable	0：WDT disabled 1：WDT enabled

TORR

偏移地址：0x0C

表5-4 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:16	RSVD_WDT_TORR	R	For reserved	Reserved
15:0	torr	R/W	Timeout period	Timeout range high 16bit reg. Low 16bit is tied to 0xffff

TORR_START

偏移地址：0x18

表5-5 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:1	RSVD_TORR_START	R	For reserved	Reserved
0	torr_start	R/W	Torr start	Clk 为 24Mhz 时 write1,then write0, to set torr. Clk 为 32Khz 时 write1,间隔大于 40us, then write0, to set torr.

CCVR

偏移地址：0x24

表5-6 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:0	CCVR	R	该寄存器保存当前计数值	当前计数值

KICK

偏移地址：0x30

表5-7 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:0	KICK	R/W	CLk 为 24Mhz 时： kick the dog,write 0x6169_6370, then write 0 CLk 为 32Khz 时： kick the dog,write 0x6169_6370, 大于间隔 40us,then write 0	看门狗重新计数

INTR_EIO

偏移地址：0x3C

表5-8 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:1	RSVD_INTR_EIO	R	For reserved	Reserved
0	intr_eio	W1C	Clean interrupt	写 1，清理中断

INTR_MASK

偏移地址：0x48

表5-9 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:1	RSVD_INTR_MASK	R	For reserved	Reserved
0	intr_mask	R/W	Interrupt mask	写 1，屏蔽中断

INTR_EN

偏移地址：0x54

表5-10 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:1	RSVD_INTR_EN	R	For reserved	Reserved
0	intr_en	R/W	Interrupt enable	写 1，使能中断

INTR_RAW_STATUS

偏移地址：0x60

表5-11 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:1	RSVD_INTR_RAW_STATUS	R	For reserved	Reserved
0	intr_raw_status	R/W	Interrupt raw status	未使能中断时的中断状态

INTR_STATUS

偏移地址：0x6C

表5-12 WDT 内部的寄存器描述

位域	名称	访问	描述	说明
31:1	RSVD_INTR_STATUS	R	For reserved	Reserved
0	intr_status	R/W	Interrupt status	使能中断后的中断状态

6 内核配置

6.1 设备树配置

```
wdt0: watchdog@4840000 {
        compatible = "axera,ax-wdt";
        reg = <0x0 0x4840000 0x0 0x400>,
              <0x0 0x4870000 0x0 0x400>;
        interrupts = <GIC_SPI 149 IRQ_TYPE_LEVEL_HIGH>; //支持中断喂狗功能
        cpu_flag = <0>;
        clock-frequency = <24000000>; //正常工作模式下的clk，可配置为24M和32K
        /* keep-alive
          休眠模式下的clk，可配置为24M和32K，需要休眠唤醒功能支持，
          目前休眠时默认开启wdt0，175 S中断喂狗一次，此时间尽量不
          要修改，如需修改需要和休眠唤醒功能同步确认，确保wdt重启
          时间大于休眠时间
        */
        keep-alive = <24000000 175>;
        clk_set = <0xA8 0xAC 19>;
        aclk = <0xB0 0xB4 14>;
        pclk = <0xC8 0xCC 19>;
        arst = <0xf0 0xF4 1>;
        prst = <0xf0 0xF4 0>;
        status = "disabled"; //默认关闭
};
```

在 dtsi 中，看门狗默认是关闭的，在项目的 dts 文件中按需修改是否启用看门狗。

```
&wdt0 {
        status = "okay";
};
      wuqianlong, 上周 • [
&wdt2 {
        status = "okay";
};
```

6.2 功能选择

1. 内核中断喂狗：默认配置 wdt0 和 wdt2 为中断喂狗

2. wdt0 和 wdt2 都配置为内核线程喂狗：如下，注释 dtsi 中的中断号，即切换成内核线程喂狗

```
--- a/linux-4.19.125/arch/arm64/boot/dts/axera/AX620E.dtsi
+++ b/linux-4.19.125/arch/arm64/boot/dts/axera/AX620E.dtsi
@@ -376,7 +376,7 @@
                        compatible = "axera,ax-wdt";
                        reg = <0x0 0x4840000 0x0 0x400>,
                              <0x0 0x4870000 0x0 0x400>;
-                       interrupts = <GIC_SPI 149 IRQ_TYPE_LEVEL_HIGH>;
+                       // interrupts = <GIC_SPI 149 IRQ_TYPE_LEVEL_HIGH>;
                        cpu_flag = <0>;
                        clock-frequency = <24000000>;
                        keep-alive = <24000000 175>;
@@ -391,7 +391,7 @@
                        compatible = "axera,ax-wdt";
                        reg = <0x0 0x6040000 0x0 0x400>,
                              <0x0 0x4870000 0x0 0x400>;
-                       interrupts = <GIC_SPI 150 IRQ_TYPE_LEVEL_HIGH>;
+                       // interrupts = <GIC_SPI 150 IRQ_TYPE_LEVEL_HIGH>;
                        cpu_flag = <1>;
                        clock-frequency = <24000000>;
                        keep-alive = <24000000 175>;
```

3. 配置某一个 wdt 由应用喂狗：由于需要关闭内核线程喂狗，所以另一个 wdt 需要配置为中断模式（版本默认配置，无需修改）。以 wdt2 配置为中断喂狗，wdt0 由上层应用喂狗（wdt0 对应的设备节点有两个/dev/watchdog 和/dev/watchdog0），如下配置流程(详细见下图)

1.在 deconfig 文件中关闭内核线程喂狗功能，

2.在 dtsi 文件中关闭 wdt0 的中断喂狗功能。

注意：目前两个 wdt 在 kernel 中都是默认打开的，180S 内不进行喂狗，系统会重启。

```
diff --git a/linux-4.19.125/arch/arm/configs/axera_AX630C_emmc_arm32_k419_defconfig b/linux-4.19.125/arch/arm/configs/axera_AX630C_emmc_arm32_k419_defconfig
index f8c4f7f4f..0bc19bae0 100644
--- a/linux-4.19.125/arch/arm/configs/axera_AX630C_emmc_arm32_k419_defconfig
+++ b/linux-4.19.125/arch/arm/configs/axera_AX630C_emmc_arm32_k419_defconfig
@@ -1560,9 +1560,9 @@ CONFIG_THERMAL_EMULATION=y
 CONFIG_AXERA_THERMAL=y
 CONFIG_WATCHDOG=y
 CONFIG_WATCHDOG_CORE=y
 CONFIG_WATCHDOG_NOWAYOUT=y
-CONFIG_WATCHDOG_HANDLE_BOOT_ENABLED=y
+# CONFIG_WATCHDOG_HANDLE_BOOT_ENABLED is not set
 # CONFIG_WATCHDOG_SYSFS is not set

 #
 # Watchdog Device Drivers
diff --git a/linux-4.19.125/arch/arm64/boot/dts/axera/AX620E.dtsi b/linux-4.19.125/arch/arm64/boot/dts/axera/AX620E.dtsi
index 7a9760924..2007f68dd 100644
--- a/linux-4.19.125/arch/arm64/boot/dts/axera/AX620E.dtsi
+++ b/linux-4.19.125/arch/arm64/boot/dts/axera/AX620E.dtsi
@@ -375,9 +375,9 @@
                wdt0: watchdog@4840000 {
                        compatible = "axera,ax-wdt";
                        reg = <0x0 0x4840000 0x0 0x400>,
                              <0x0 0x4870000 0x0 0x400>;
-                       interrupts = <GIC_SPI 149 IRQ_TYPE_LEVEL_HIGH>;
+                       // interrupts = <GIC_SPI 149 IRQ_TYPE_LEVEL_HIGH>;
                        cpu_flag = <0>;
                        clock-frequency = <24000000>;
                        keep-alive = <24000000 175>;
                        clk_set = <0xA8 0xAC 19>;
```

4. wdt 默认是不能被应用层关闭的，如需要支持应用层关闭 wdt 的功能，修改如下

```
diff --git a/linux-4.19.125/arch/arm/configs/axera_AX630C_emmc_arm32_k419_defconfig b/linux-4.19.125/arch/arm/configs/axera_AX630C_emmc_arm32_k419_defconfig
index f8c4f7f4f..cc56339cc 100644
--- a/linux-4.19.125/arch/arm/configs/axera_AX630C_emmc_arm32_k419_defconfig
+++ b/linux-4.19.125/arch/arm/configs/axera_AX630C_emmc_arm32_k419_defconfig
@@ -1559,9 +1559,9 @@ CONFIG_THERMAL_EMULATION=y
 # CONFIG_GENERIC_ADC_THERMAL is not set
 CONFIG_AXERA_THERMAL=y
 CONFIG_WATCHDOG=y
 CONFIG_WATCHDOG_CORE=y
-CONFIG_WATCHDOG_NOWAYOUT=y
+# CONFIG_WATCHDOG_NOWAYOUT is not set
 CONFIG_WATCHDOG_HANDLE_BOOT_ENABLED=y
 # CONFIG_WATCHDOG_SYSFS is not set
```