

3. 16 看门狗实验

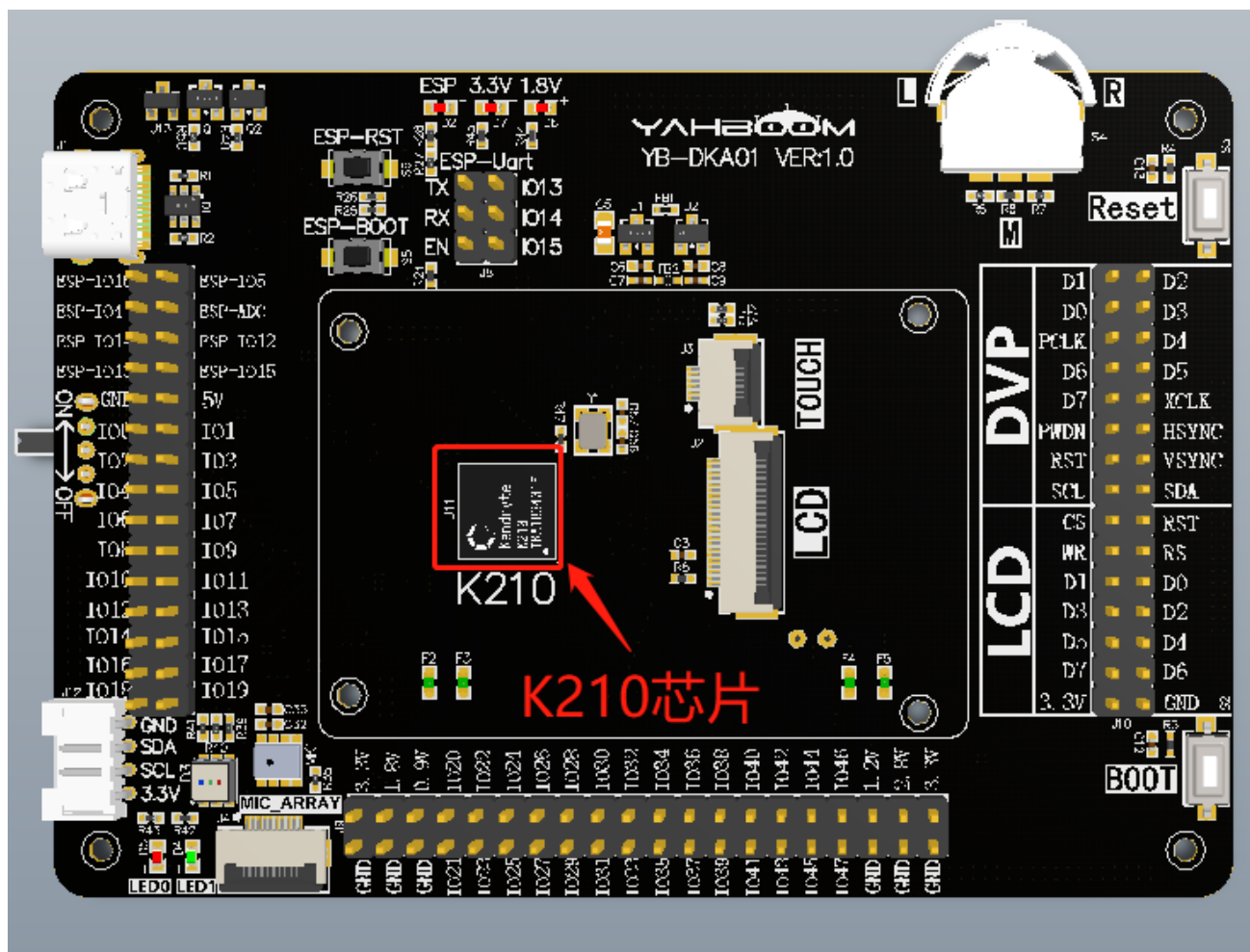
一、实验目的

本节课主要学习 K210 的看门狗超时复位的功能。

二、实验准备

1. 实验元件

K210 芯片中的看门狗。



2. 元件特性

WDT 是外围总线（APB）的一种从外设，并且也是“同步化硬件组件设计”的组成部分，具有两个 WDT，分别为 WDT0 和 WDT1 看门狗定时器，主要包含的模块

有一个 APB 从接口，一个当前计数器同步的寄存器模块，一个随着计数器递减的中断/系统重置模块和逻辑控制电路，一个同步时钟域来为异步时钟同步做支持。

看门狗定时器支持如下设置：

- APB 总线宽度可配置为 8、16 和 32 位
- 时钟计数器从某一个设定的值递减到 0 来指示时间的计时终止
- 可选择的外部时钟使能信号，用于控制计数器的计数速率
- 一个时钟超时 WDT 可以执行以下任务：
 - 产生一个系统复位信号
 - 首先产生一个中断，即使该位是否已经被中断服务清除，其次它会产生一个系统复位信号
- 占空比可编程调节
- 可编程和硬件设定计数器起始值
- 计数器重新计时保护
- 暂停模式，仅当使能外部暂停信号时
- WDT 偶然禁用保护
- 测试模式，用来进行计数器功能测试（递减操作）
- 外部异步时钟支持。当该项功能启用时，将会产生时钟中断和系统重置信号，即使 APB 总线时钟关闭的情况下。

4. SDK 中对应 API 功能

对应的头文件 `wdt.h`

WDT 看门狗在开发单片机中作用巨大，可以在程序出现死机的情况自动重启系统，而不需要手动操作。

为用户提供以下接口：

- wdt_init: 配置看门狗参数，启动看门狗，不使用中断的话，将 on_irq 设置为 NULL。返回值为看门狗实际超时时间，一般比设置的时间稍微大一些。
- wdt_start(0.6.0 后不再支持，请使用 wdt_init)
- wdt_stop: 关闭看门狗。
- wdt_feed: 重置看门狗计时器，俗称喂狗。
- wdt_clear_interrupt: 清除中断，如果在中断函数中清除中断，看门狗不会重启。

三、实验原理

看门狗其实就是一个需要在设定一定时间内被复位的计数器，如果没有按时复位，则会强制系统复位。在看门狗启动前需要配置超时时间，当看门狗启动后，计数器开始自动计数，经过一定时间，如果没有被复位，计数器溢出就会对 CPU 产生一个复位信号使系统重启（俗称“被狗咬”）。要保证系统正常运行时，需要在看门狗超时时间内重置看门狗计数器（俗称“喂狗”）。

四、实验过程

1. 首先在系统启动的时候答应一次“system start!”提示，可以清楚地知道系统什么时候重启过。times 用于记录喂狗的次数。然后初始化系统中断以及使能全局中断。

```
/* 打印系统启动信息 */  
printf("system start!\n");  
/* 记录feed的次数 */  
int times = 0;  
  
/* 系统中断初始化 */  
plic_init();  
sysctl_enable_irq();
```

2. 配置看门狗的参数，使用的是看门狗 WDT0，设置超时时间为 2 秒，中断函数函数为 wdt0_irq_cb，返回值是看门狗实际超时的时间，一般会比设置的时间稍微大一些。这里注意中断函数不是超时的时候调用的，而是实际超时时间的一

半调用的。比如这里设置了 2 秒超时时间，实际超时时间约 2.58 秒，如果在 1.29 秒之前没有喂狗，则会调用中断回调函数。

```
/* 启动看门狗，设置超时时间为2秒后调用中断函数wdt0_irq_cb */
int timeout = wdt_init(WDT_DEVICE_0, 2000, wdt0_irq_cb, NULL);

/* 打印看门狗实际超时的时间 */
printf("wdt timeout is %d ms!\n", timeout);
```

3. 在 WDT0 的中断回调中打印系统超时的信息，默认 WDT_TIMEOUT_REBOOT 为 1，看门狗超时重启，如果把 WDT_TIMEOUT_REBOOT 设置为 0，则重启只会打印提示，不会重启。

```
#define WDT_TIMEOUT_REBOOT 1

int wdt0_irq_cb(void *ctx)
{
    #if WDT_TIMEOUT_REBOOT
        printf("%s:The system will reboot soon!\n", __func__);
        while(1);
    #else
        printf("%s:The system is busy but not reboot!\n", __func__);
        wdt_clear_interrupt(WDT_DEVICE_0);
    #endif
    return 0;
}
```

4. 前五次每隔 1 秒钟喂狗一次，所以在五秒之后没有喂狗，系统约过 2.6 秒后重启。

```
while(1)
{
    sleep(1);
    if(times++ < 5)
    {
        /* 打印feed的次数 */
        printf("wdt_feed %d times!\n", times);

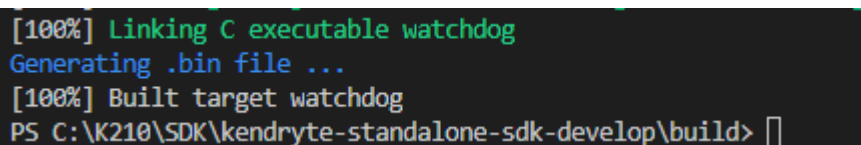
        /* 重置看门狗的计时器，重新开始计时 */
        wdt_feed(WDT_DEVICE_0);
    }
}
```

5. 编译调试，烧录运行

把本课程资料中的 watchdog 复制到 SDK 中的 src 目录下，然后进入 build 目录，运行以下命令编译。

```
cmake .. -DPROJ=watchdog -G "MinGW Makefiles"
```

```
make
```



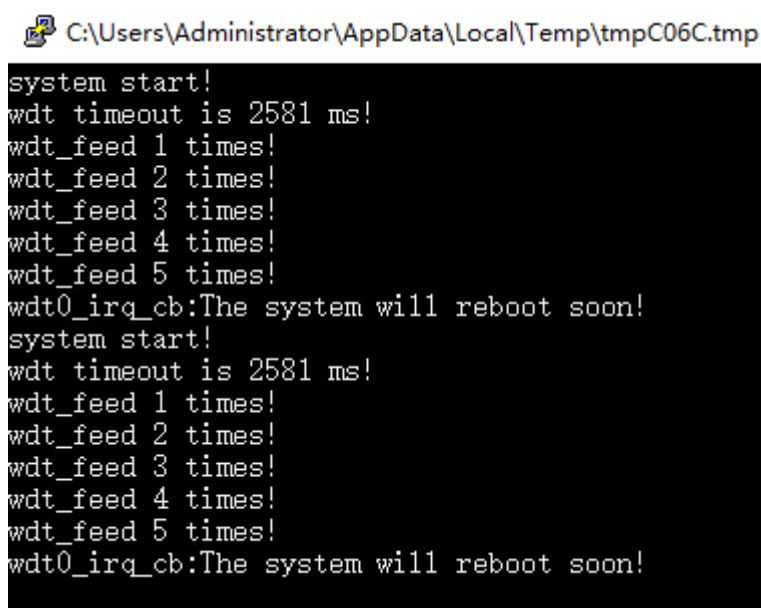
```
[100%] Linking C executable watchdog
Generating .bin file ...
[100%] Built target watchdog
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build>
```

编译完成后，在 build 文件夹下会生成 watchdog.bin 文件。

使用 type-C 数据线连接电脑与 K210 开发板，打开 kflash，选择对应的设备，再将程序固件烧录到 K210 开发板上。

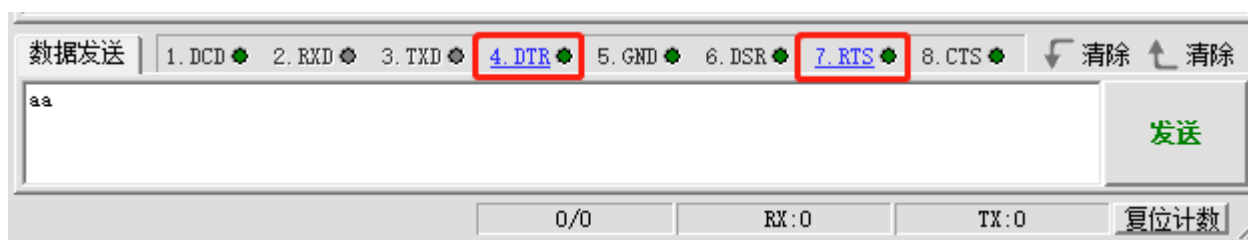
五、实验现象

烧录完成固件后，系统会弹出一个终端界面，如果没有弹出终端界面的可以打开串口助手显示调试内容。



```
C:\Users\Administrator\AppData\Local\Temp\tmpC06C.tmp
system start!
wdt timeout is 2581 ms!
wdt_feed 1 times!
wdt_feed 2 times!
wdt_feed 3 times!
wdt_feed 4 times!
wdt_feed 5 times!
wdt0_irq_cb:The system will reboot soon!
system start!
wdt timeout is 2581 ms!
wdt_feed 1 times!
wdt_feed 2 times!
wdt_feed 3 times!
wdt_feed 4 times!
wdt_feed 5 times!
wdt0_irq_cb:The system will reboot soon!
```

打开电脑的串口助手，选择对应的 K210 开发板对应的串口号，波特率设置为 115200，然后点击打开串口助手。注意还需要设置一下串口助手的 DTR 和 RTS。在串口助手底部此时的 4. DTR 和 7. RTS 默认是红色的，点击 4. DTR 和 7. RTS，都设置为绿色，然后按一下 K210 开发板的复位键。



可以看到串口助手打印系统启动的信息，并且每隔一秒打印一次喂狗的次数，当过了 5 秒后不再喂狗，系统再过 2.6 秒后重启系统。



六、实验总结

1. 看门狗的作用是当没有在设定的时间内喂狗，则系统会发送中断使系统强制重启。
2. 看门狗必须在系统正常运行的情况下喂狗，这样系统异常时就能够及时重启系统。
3. 看门狗的定时中断是实际超时时间的一半，需要在这个时间内喂狗。

附：API

对应的头文件 `wdt.h`

wdt_init

描述

配置参数，启动看门狗。不使用中断的话，将 `on_irq` 设置为 `NULL`。

函数原型

```
uint32_t wdt_init(wdt_device_number_t id, uint64_t time_out_ms,  
plic_irq_callback_t on_irq, void *ctx)
```

参数

参数名称	描述	输入输出
<code>id</code>	看门狗编号	输入
<code>time_out_ms</code>	超时时间（毫秒）	输入
<code>on_irq</code>	中断回调函数	输入
<code>ctx</code>	回调函数参数	输入

返回值

看门狗超时重启的实际时间（毫秒）。与 `time_out_ms` 有差异，一般会大于这个时间。在外部晶振 26M 的情况下，最大超时时间为 330 毫秒。

wdt_start

描述

启动看门狗。

函数原型

```
void wdt_start(wdt_device_number_t id, uint64_t time_out_ms,  
plic_irq_callback_t on_irq)
```

参数

参数名称	描述	输入输出
id	看门狗编号	输入
time_out_ms	超时时间（毫秒）	输入
on_irq	中断回调函数	输入

返回值

无

wdt_stop

描述

关闭看门狗。

函数原型

```
void wdt_stop(wdt_device_number_t id)
```

参数

参数名称	描述	输入输出
id	看门狗编号	输入

返回值

无。

wdt_feed

描述

喂狗。

函数原型

```
void wdt_feed(wdt_device_number_t id)
```

参数

参数名称	描述	输入输出
id	看门狗编号	输入

返回值

无。

wdt_clear_interrupt

描述

清除中断。如果在中断函数中清除中断，则看门狗不会重启。

函数原型

```
void wdt_clear_interrupt(wdt_device_number_t id)
```

参数

参数名称	描述	输入输出
id	看门狗编号	输入

返回值

无。

数据类型

相关数据类型、数据结构定义如下：

- wdt_device_number_t

wdt_device_number_t

描述

看门狗编号。

定义

```
typedef enum _wdt_device_number
{
    WDT_DEVICE_0,
    WDT_DEVICE_1,
    WDT_DEVICE_MAX,
} wdt_device_number_t;
```

成员

成员名称	描述
WDT_DEVICE_0	看门狗 0
WDT_DEVICE_1	看门狗 1