3.16 Watchdog experiment

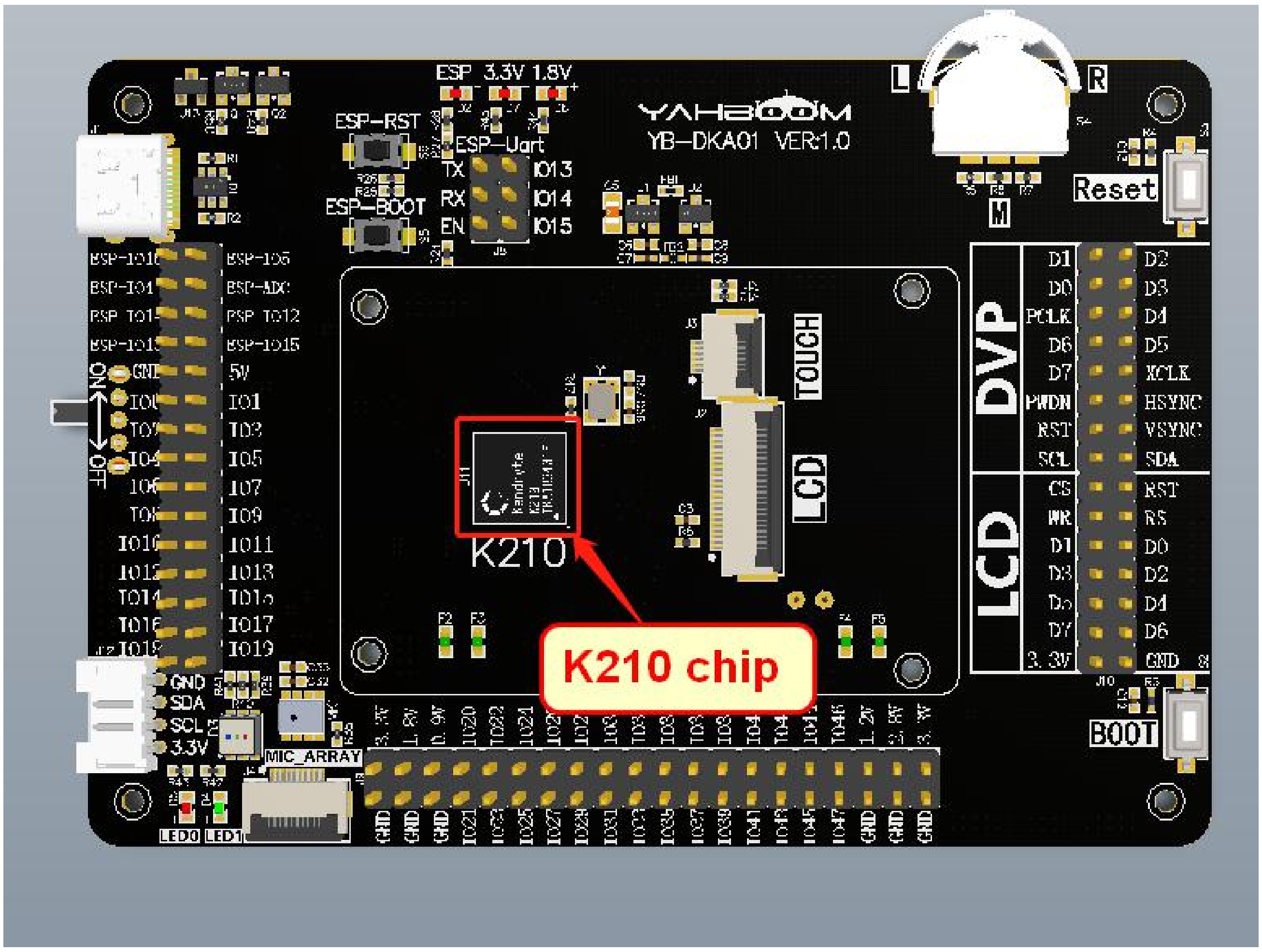
1. Experiment purpose

In this lesson, we mainly learn watchdog timeout reset function of K210.

2.Experiment preparation

2.1 components

watchdog of K210



2. Component characteristics

WDT is a slave peripheral of the peripheral bus (APB), which are respectively WDT0 and WDT1 watchdog timers.

It mainly includes an APB slave interface, a register module for current counter synchronization, an interrupt/system reset module and logic control circuit that decrease with the counter, and a synchronous clock domain to support asynchronous clock synchronization.

The watchdog timer supports the following settings:

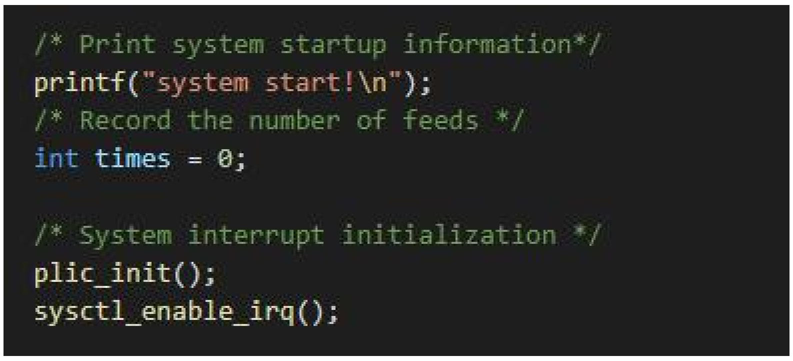
- APB bus width can be configured as 8, 16 and 32 bits
- The clock counter indicates the end of time counting by decrementing from a set value to 0
- Optional external clock enable signal to control the counting rate of the counter
- A clock timeout WDT can perform the following tasks:

– Generate a system reset signal
– First, generate an interrupt, even if the bit has been cleared by the interrupt service. Then, it will generate a system reset signal
• Duty cycle adjustment can be adjustment by programming
• Setting counter start value by programming or hardware
• Counter re-time protection
• Pause mode, only when the external pause signal is enabled
• WDT accidentally disabled protection
• Test mode for counter function test (Decrement operation)
• Support external asynchronous clock. When this function is enabled, a clock interrupt and system reset signal will be generated, even if the APB bus clock is turned off

2.4 SDK API function
The header file is wdt.h
WDT watchdog can automatically restart the system when the program crashes, without manual operation.
We will provide following interfaces to users:
• wdt_init: Configure watchdog parameters, start the watchdog. If interrupts are not used, and set on_irq to NULL. The return value is the actual timeout time of the watchdog, which is generally slightly larger than the set time.
• wdt_start (No longer supported after 0.6.0, please use **wdt_init**)
• wdt_stop: Close the watchdog.
• wdt_feed: Reset the watchdog timer.
• wdt_clear_interrupt: Clear the interrupt. If the interrupt is cleared in the interrupt function, the watchdog will not restart.

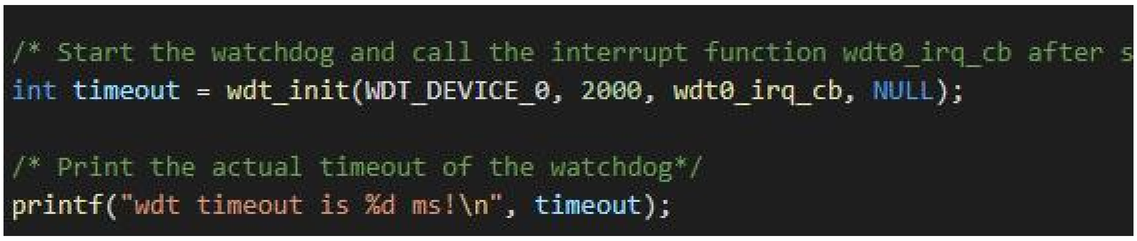
3. Experimental principle
The watchdog is actually a counter that needs to be reset within a set period of time. If it is not reset on time, it will force the system to reset.

4. Experiment procedure
4.1 First, it will print "system start!" when the system starts. times is used to record the number of restarts of the watchdog. Then initialize the system interrupt and enable the global interrupt.

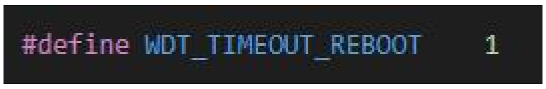


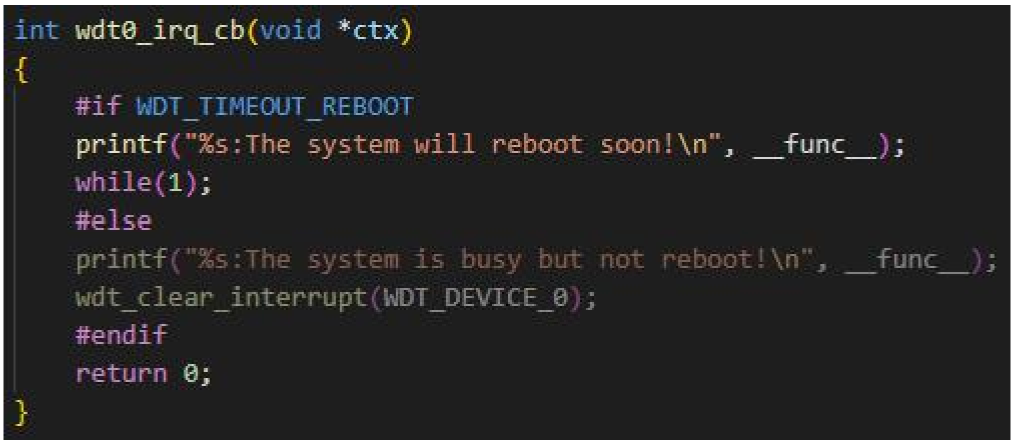
```
/* Print system startup information*/
printf("system start!\n");
/* Record the number of feeds */
int times = 0;

/* System interrupt initialization */
plic_init();
sysctl_enable_irq();
```

4.2 Configure the watchdog parameters, we use the watchdog WDT0, set the timeout time to 2 seconds, the interrupt function function is wdt0_irq_cb. Return value is the actual timeout time of the watchdog, which is generally slightly larger than the set time.

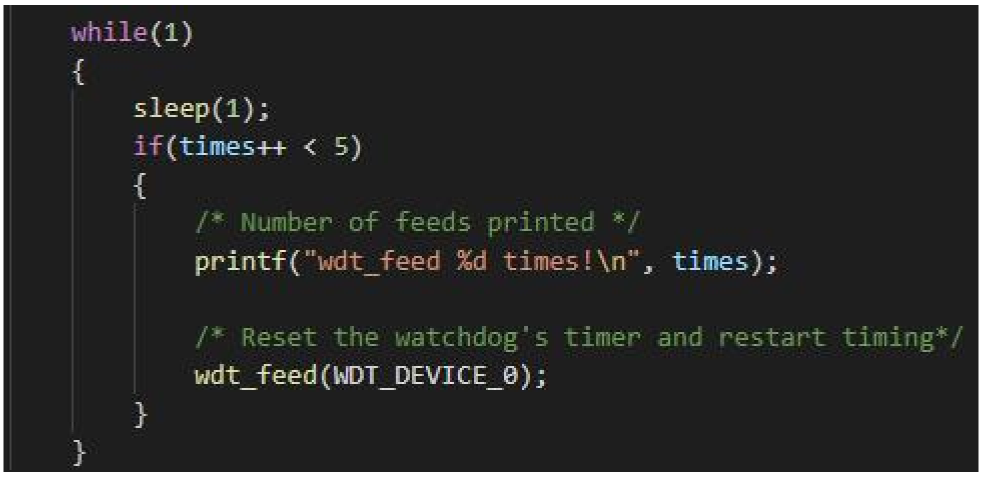

```
/* Start the watchdog and call the interrupt function wdt0_irq_cb after s
int timeout = wdt_init(WDT_DEVICE_0, 2000, wdt0_irq_cb, NULL);

/* Print the actual timeout of the watchdog*/
printf("wdt timeout is %d ms!\n", timeout);
```

4.3 Print the system timeout information in the interrupt callback of WDT0. The default WDT_TIMEOUT_REBOOT is 1, and the watchdog restarts after timeout. If WDT_TIMEOUT_REBOOT is set to 0, the restart will only print a prompt and will not restart.


```
#define WDT_TIMEOUT_REBOOT    1
```

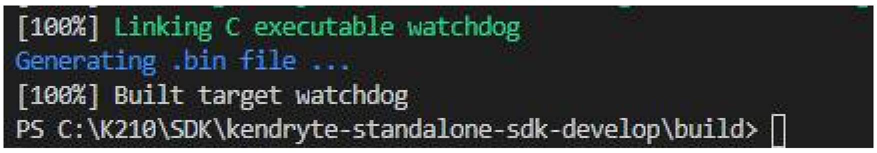

```
int wdt0_irq_cb(void *ctx)
{
    #if WDT_TIMEOUT_REBOOT
    printf("%s:The system will reboot soon!\n", __func__);
    while(1);
    #else
    printf("%s:The system is busy but not reboot!\n", __func__);
    wdt_clear_interrupt(WDT_DEVICE_0);
    #endif
    return 0;
}
```

4.4 The watchdog was restarted every 1 second for the first five times, and the system restarted after about 2.6 seconds. start.


```
    while(1)
    {
        sleep(1);
        if(times++ < 5)
        {
            /* Number of feeds printed */
            printf("wdt_feed %d times!\n", times);

            /* Reset the watchdog's timer and restart timing*/
            wdt_feed(WDT_DEVICE_0);
        }
    }
```

4.7 Compile and debug, burn and run
Copy the watchdog to the src directory in the SDK.
Then, enter the build directory and run the following command to compile.

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

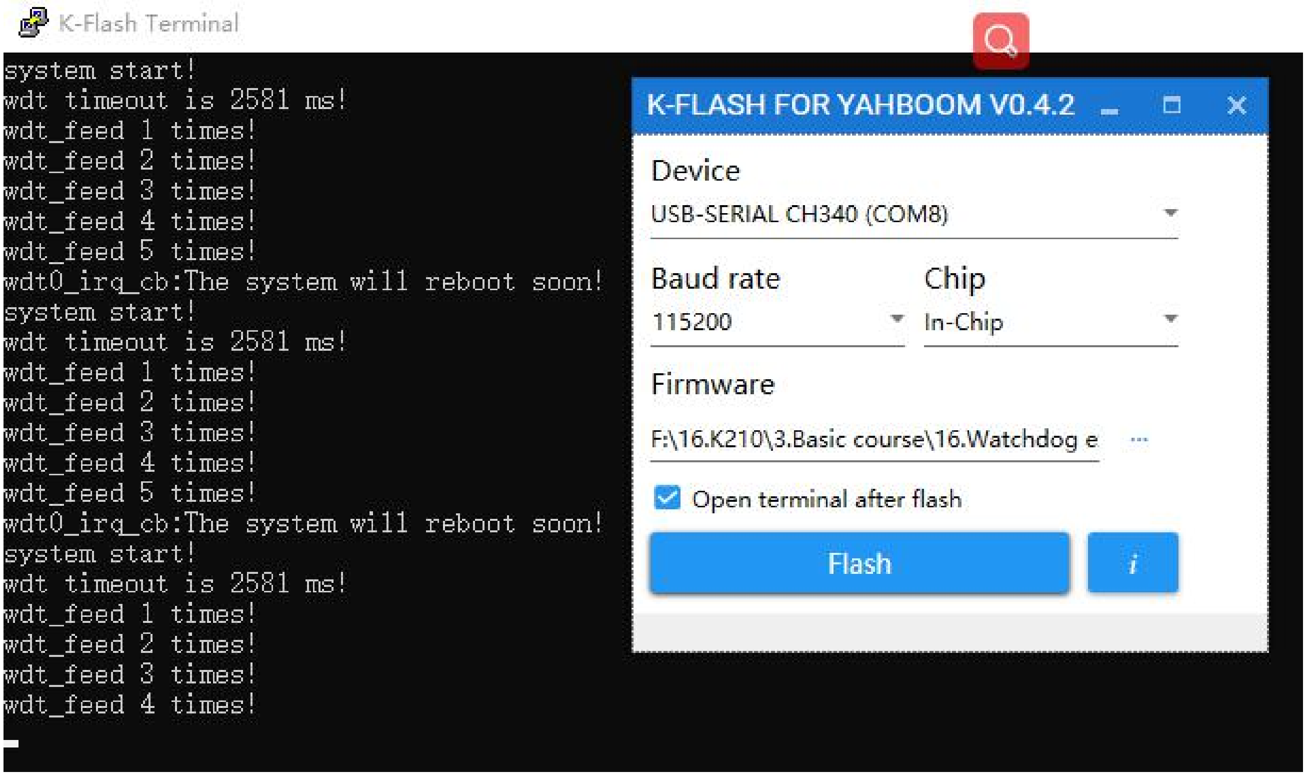


```
[100%] Linking C executable watchdog
Generating .bin file ...
[100%] Built target watchdog
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build>
```

After the compilation is complete, the **watchdog.bin** file will be generated in the build folder.
We need to use the type-C data cable to connect the computer and the K210 development board. Open kflash, select the corresponding device, and then burn the **watchdog.bin** file to the K210 development board.

5.Experimental phenomenon

After the firmware is burned, a terminal interface will pop up. If there is no terminal interface, you can open the serial port assistant to display the debugging content.



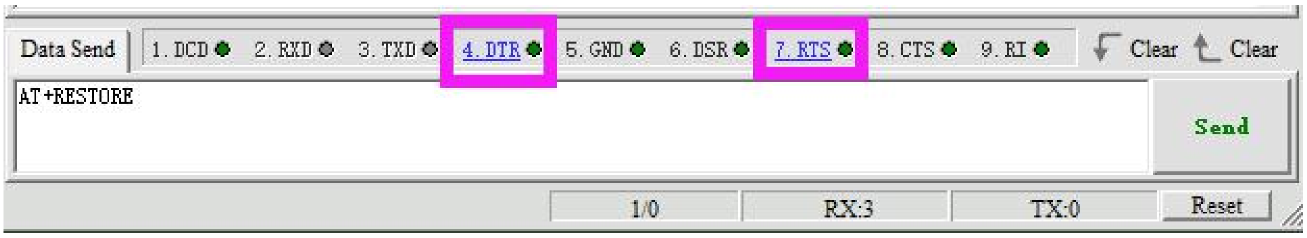
K-Flash Terminal

```
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!
system start!
wdt timeout is 2581 ms!
wdt_feed 1 times!
wdt_feed 2 times!
wdt_feed 3 times!
wdt_feed 4 times!
```

K-FLASH FOR YAHBOOM V0.4.2

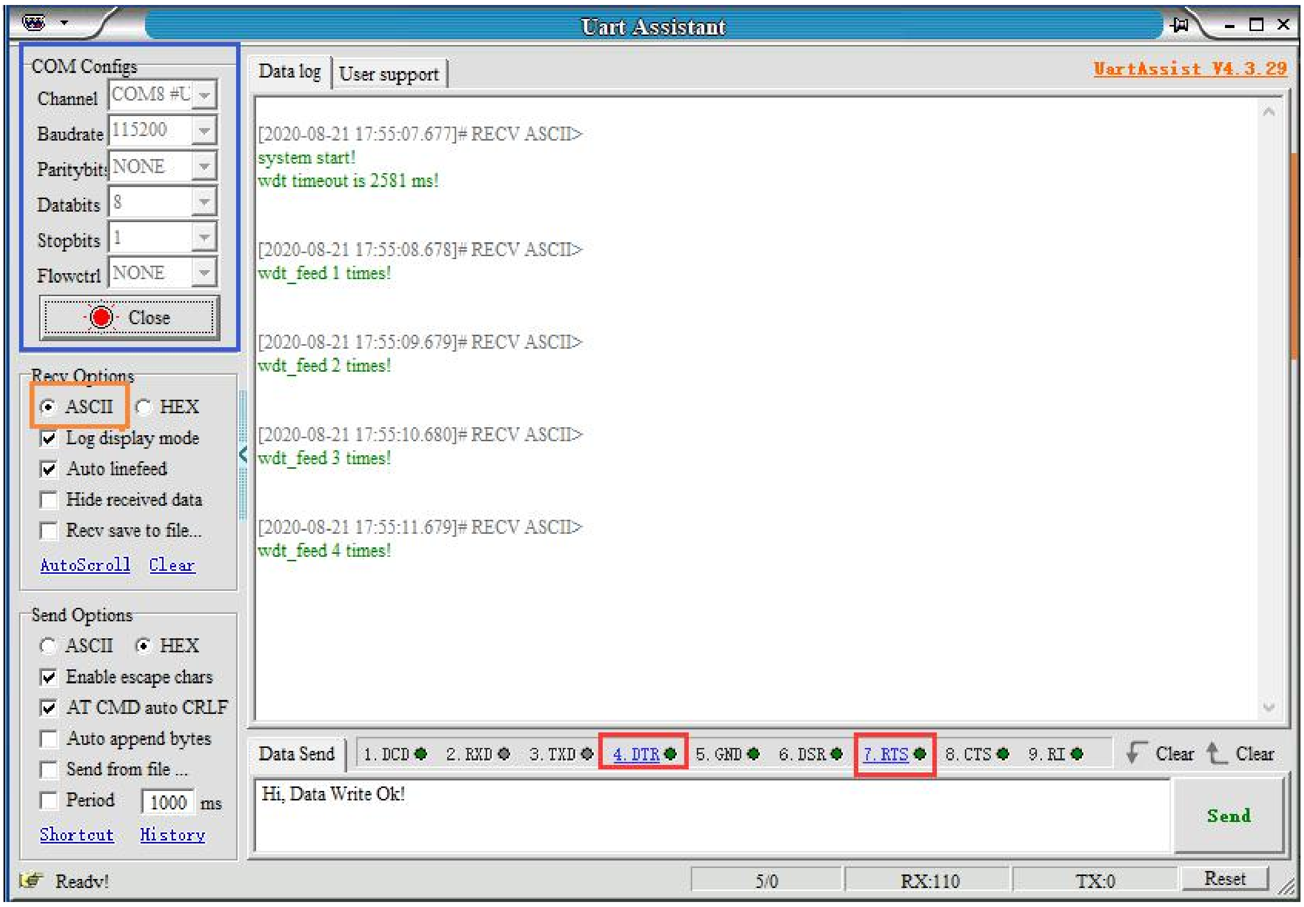
Device: USB-SERIAL CH340 (COM8)
Baud rate: 115200
Chip: In-Chip
Firmware: F:\16.K210\3.Basic course\16.Watchdog e
☑ Open terminal after flash

Close this terminal interface. Open the serial port assistant of the computer, select the corresponding serial port number of the corresponding K210 development board, set the baud rate to 115200.
Then, click to open the serial port assistant.
!Note: you also need to set the DTR and RTS of the serial port assistant.
Click 4.DTR and 7.RTS to set them to green.



Data Send
1. DCD ● 2. RXD ● 3. TXD ● 4. DTR ● 5. GND ● 6. DSR ● 7. RTS ● 8. CTS ● 9. RI ●
Clear
Clear
AT+RESTORE
Send
1/0
RX:3
TX:0
Reset

Open the serial port assistant, we can see the serial assistant print the information of the system startup, and print the number of restart watchdog every second. After 5 seconds, the system will stop restarting watchdog, and restart the system after 2.6 seconds.



```
[2020-08-21 17:55:07.677]# RECV ASCII>
system start!
wdt timeout is 2581 ms!

[2020-08-21 17:55:08.678]# RECV ASCII>
wdt_feed 1 times!

[2020-08-21 17:55:09.679]# RECV ASCII>
wdt_feed 2 times!

[2020-08-21 17:55:10.680]# RECV ASCII>
wdt_feed 3 times!

[2020-08-21 17:55:11.679]# RECV ASCII>
wdt_feed 4 times!
```

6. Experiment summary

6.1 The function of the watchdog is that when the watchdog is not restart within the set time, the system will send an interrupt to force the system to restart.

6.2 The watchdog must restart the dog under the normal operation of the system, so that the system can be restarted timely if the system is abnormal.

6.3 The watchdog timing interruption is half of the actual timeout period, we need to restart watchdog during this time.

Appendix -- API

Header file is wdt.h

wdt_init

Description: Configure the parameters to start the watchdog.

If no interrupt is used, set on_IRq to NULL.

Function prototype: **uint32_t wdt_init(wdt_device_number_t id, uint64_t time_out_ms, plic_irq_callback_t on_irq, void *ctx)**

Parameter:

Parameter name	Description	Input/Output
id	WDT number	Input
time_out_ms	Time-out period(ms)	Input
on_irq	Interrupt callback function	Input
ctx	Callback function arguments	Input

Return value:
The actual time (milliseconds) that the watchdog timed out to restart.
This is different from time_out_MS, which is generally greater than this time.
With an external crystal oscillator of 26M, the maximum timeout is 330 ms.

wdt_start
Description: Start WDT
Function prototype: **void wdt_start(wdt_device_number_t id, uint64_t time_out_ms, plic_irq_callback_t on_irq)**
Parameter:

Parameter name	Description	Input/Output
id	WDT number	Input
time_out_ms	Time-out period(ms)	Input
on_irq	Interrupt callback function	Input

Return value: No

wdt_stop
Description: Close WDT
Function prototype: **void wdt_stop(wdt_device_number_t id)**
Parameter:

Parameter name	Description	Input/Output
id	WDT number	Input

Return value: No

wdt_feed
Description: Restart WDT
Function prototype: **void wdt_feed(wdt_device_number_t id)**
Parameter:

Parameter name	Description	Input/Output
id	WDT number	Input

Return value: No

wdt_clear_interrupt
Description: Clear interrupts.If the interrupt is cleared in the interrupt function, the watchdog does not restart.
Function prototype: **void wdt_clear_interrupt(wdt_device_number_t id)**

Parameter:

Parameter name	Description	Input/Output
id	WDT number	Input/Output

Return value: No

Data type

The related data types and data structure are defined as follows:

- wdt_device_number_t

wdt_device_number_t

Description: Restart WDT number

Define

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

member

Member name	**Description**
WDT_DEVICE_0	WDT 0
WDT_DEVICE_1	WDT 1