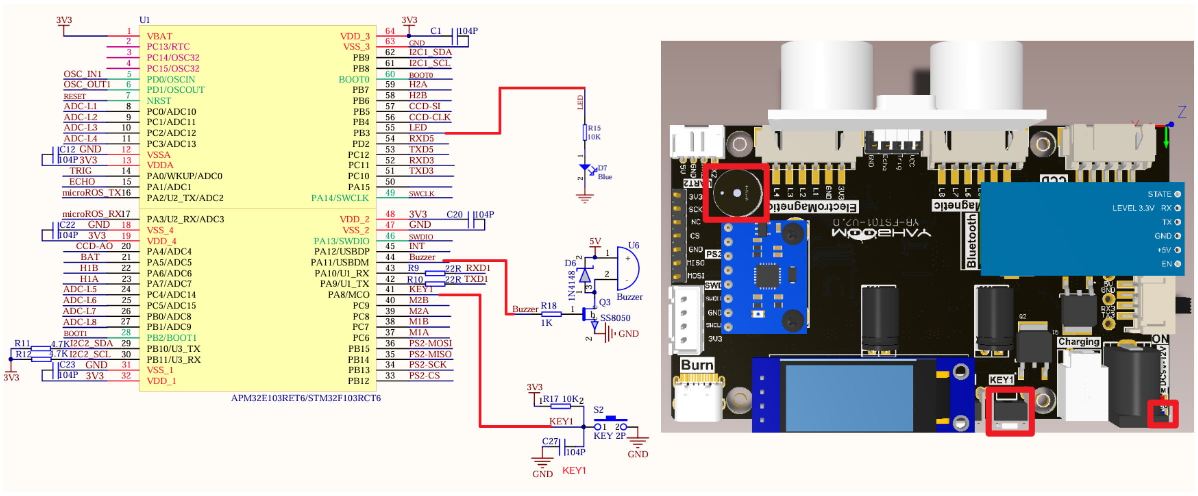
Window Watchdog (WWDG)

The tutorial combines LED, buzzer, and KEY to demonstrate the hardware fault detection function of the window watchdog (WWDG).

Hardware connection



Peripherals	Development board	Description
LED	PB3	The anode of the LED is connected to the development board PB3, and the cathode is connected to the development board GND
KEY1	PA8	One end of the KEY is connected to the PA8 pin, and the other end is connected to GND
Buzzer (active buzzer)	PA11	

Control principle

The STM32F103RCT6 has two built-in watchdogs (independent watchdog and window watchdog), which are mainly used for system fault detection and recovery.

Watchdog	Function
Independent Watchdog	Used to detect whether the system is running normally

Watchdog	Function
Window Watchdog	Used to detect system failures

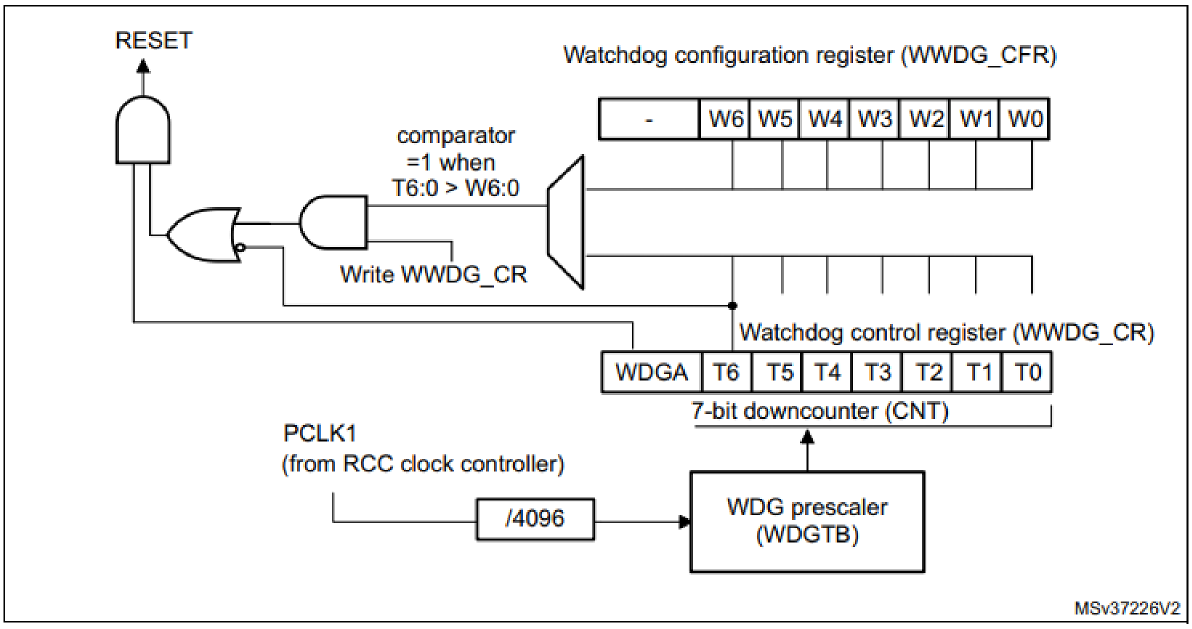
- **Window Watchdog**

The window watchdog (WWDG) is driven by the clock obtained by dividing the APB1 clock. It detects abnormal late or early operations of the application by configuring the time window.

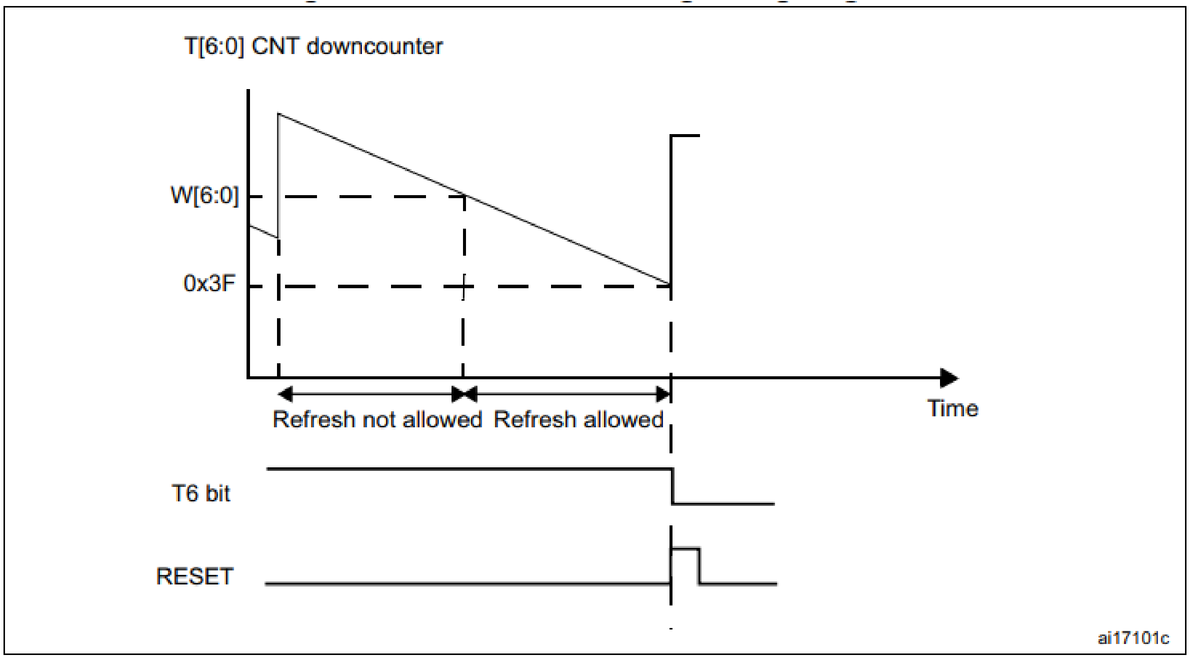
Window Watchdog	Status
Feed the watchdog when the count value > window upper limit	Reset
Feed the watchdog when the window upper limit > count value > window lower limit	Do not reset
Count value < window lower limit	Reset

```
The main function of the window watchdog is to monitor the working status of the
system and prevent deadlock or infinite loop caused by software errors or
unexpected events
```

- **Independent watchdog feature**
- Free-running down counter
- Conditional reset
- A reset is generated when the down counter is reloaded outside the window
- A reset is generated when the down counter value is less than 0x40
- If the watchdog is enabled and interrupts are enabled
- When the down counter is equal to 0x40, an early wake-up interrupt (EWI) is generated to reload the counter to avoid WWDG reset



- **Window watchdog timeout**: PCLK1=36MHz



WDGB	Minimum timeout	Maximum timeout
0	113us	7.28ms
1	227us	14.56ms

```
The frequency division number set in the tutorial is 8, the window value is 43,
and the count value is 54
```

$$T_{WWDG(ms)} = T_{PCLK1} * 4096 * 2^{WDGTB} * (T[5:0]+1)$$

T_{WWDG}: WWDG timeout time

T_{PCLK1}: APB1 time interval in ms

Example: Counter minus 1 time (PCLK1=36MHz, frequency division number is 8)

$$T_{WWDG(ms)} = T_{PCLK1} * 4096 * 2^{WDGTB} * (T[5:0]+1) = \frac{1}{36000} * 4096 * 2^3 * (1) = 0.910ms$$

Software Configuration

Software Code

Configure the Window Watchdog (WWDG) hardware fault detection function, no need to configure specific pins.

```
Product supporting materials source code path: Attachment → Source Code Summary →
1.Base_Course → 13.WWDG
```

Control Function

The tutorial only briefly introduces the code, you can open the project source code to read it.

> **IWDG_Start**

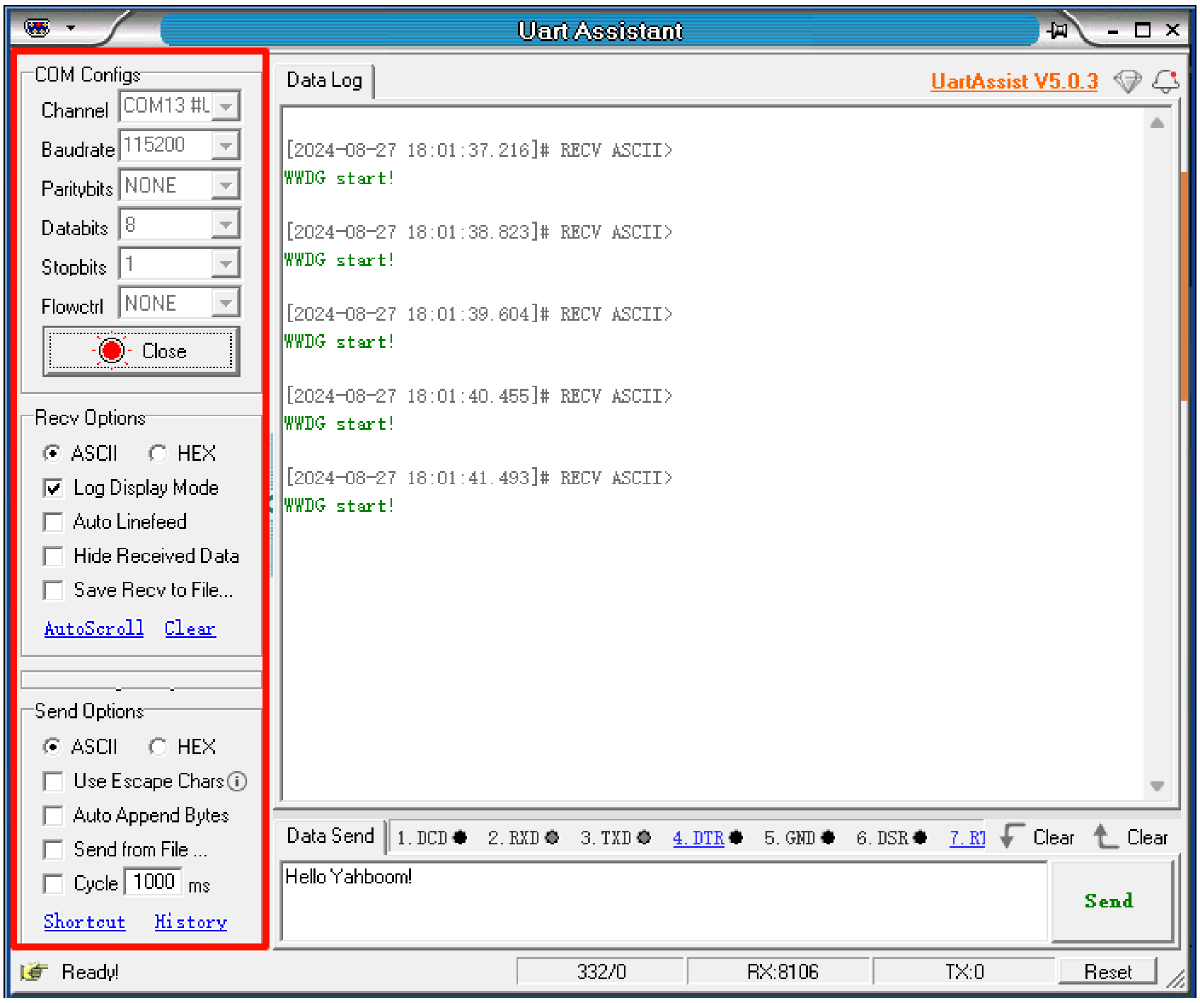
```
void WWDG_Start(void)
{
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_WWDG, ENABLE);
    WWDG_SetPrescaler(WWDG_Prescaler_8);
    WWDG_SetWindowValue(43|0x40);
    WWDG_Enable(54|0x40);
}
```

Experimental phenomenon

The WWDG.hex file generated by the project compilation is located in the OBJ folder of the WWDG project. Find the WWDG.hex file corresponding to the project and use the FlyMcu software to download the program into the development board.

After the program is downloaded successfully: Pressing KEY1 will trigger the dog not to be fed (equivalent to reset: LED turns off, buzzer beeps, serial port prints WWDG start!), if KEY1 is not pressed, the dog will be fed continuously (LED is always on)

```
When using the serial port debugging assistant, you need to pay attention to the
serial port settings. If the settings are wrong, the phenomenon may be
inconsistent
```



COM Configs
Channel: COM13 #L
Baudrate: 115200
Paritybits: NONE
Databits: 8
Stopbits: 1
Flowctrl: NONE

Recv Options: [✓ ASCII] HEX
[✓] Log Display Mode
[ ] Auto Linefeed
[ ] Hide Received Data
[ ] Save Recv to File...

Send Options: [✓ ASCII] HEX
[ ] Use Escape Chars
[ ] Auto Append Bytes
[ ] Send from File ...
[ ] Cycle 1000 ms

```
[2024-08-27 18:01:37.216]# RECV ASCII>
WWDG start!

[2024-08-27 18:01:38.823]# RECV ASCII>
WWDG start!

[2024-08-27 18:01:39.604]# RECV ASCII>
WWDG start!

[2024-08-27 18:01:40.455]# RECV ASCII>
WWDG start!

[2024-08-27 18:01:41.493]# RECV ASCII>
WWDG start!
```

Hello Yahboom!

332/0 RX:8106 TX:0