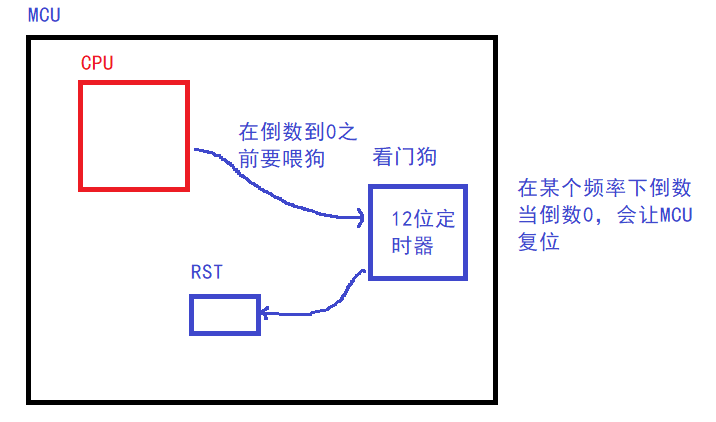
# STM32 第八天

## 独立看门狗

### 1、独立看门狗概述

在由单片机构成的微型计算机系统中，由于单片机的工作常常会受到来自外界电磁场的干扰，造成程序的跑飞，而陷入死循环，程序的正常运行被打断，由单片机控制的系统无法继续工作，会造成整个系统的陷入停滞状态，发生不可预料的后果，所以出于对单片机运行状态进行实时监测的考虑，便产生了一种专门用于监测单片机程序运行状态的模块或者芯片，俗称“看门狗”(watchdog) 。

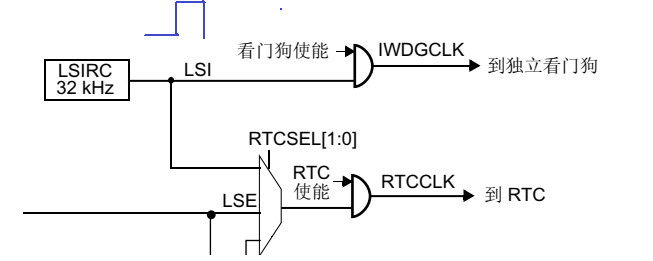
看门狗（ watchdog timer）,是一个定时器（12位）电路, 一般有一个输入,叫喂狗(kicking the dog or service the dog),一个输出到MCU的RST端,MCU正常工作的时候,每隔一段时间输出一个信号到喂狗端,给WDT清零,如果超过规定的时间不喂狗,(一般在程序跑飞时，不在程序正常的状态),WDT 定时超过,就会给出一个复位信号到MCU,使MCU复位. 防止MCU死机. 看门狗的作用就是防止程序发生死循环，或者说程序跑飞。



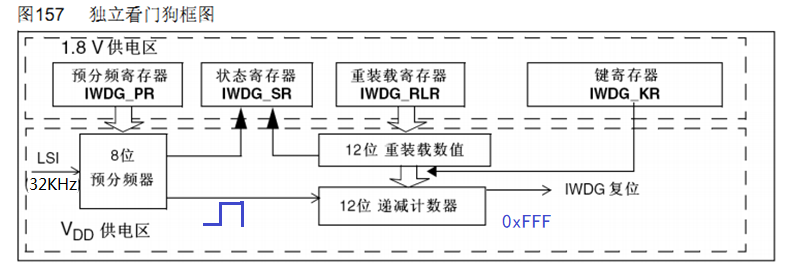
### 2、STM32独立看门狗

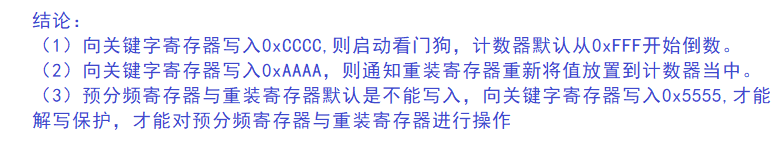
独立看门狗（IWDG)由专用的低速时钟（LSI)驱动，即使主时钟发生故障它仍有效。

独立看门狗适合应用于需要看门狗作为一个在主程序之外 能够完全独立工作，并且对时间精度要求低的场合。



STM32F4 的独立看门狗由内部专门的 32Khz 低速时钟 （LSI） 驱动，即使主时钟发生故障，它也仍然有效。这里需要注意独立看门狗的时钟是一个内部RC时钟，所以并不是准确的32Khz，而是在15~47Khz 之间的一个可变化的时钟，只是我们在估算的时候，以 32Khz 的频率来计算，独立看门狗对时间的要求不是很精确，所以，时钟有些偏差都是接受的范围。





### 3、独立看门狗操作步骤

独立看门狗操作步骤需要添加的库函数文件：stm32f4xx\_iwdg.c

1、 取消寄存器写保护：

IWDG\_WriteAccessCmd();

2、设置独立看门狗的预分频系数，确定时钟:

IWDG\_SetPrescaler();

3、设置看门狗重装载值，确定溢出时间:

IWDG\_SetReload();

4、使能看门狗

IWDG\_Enable();

5、应用程序喂狗:

IWDG\_ReloadCounter();

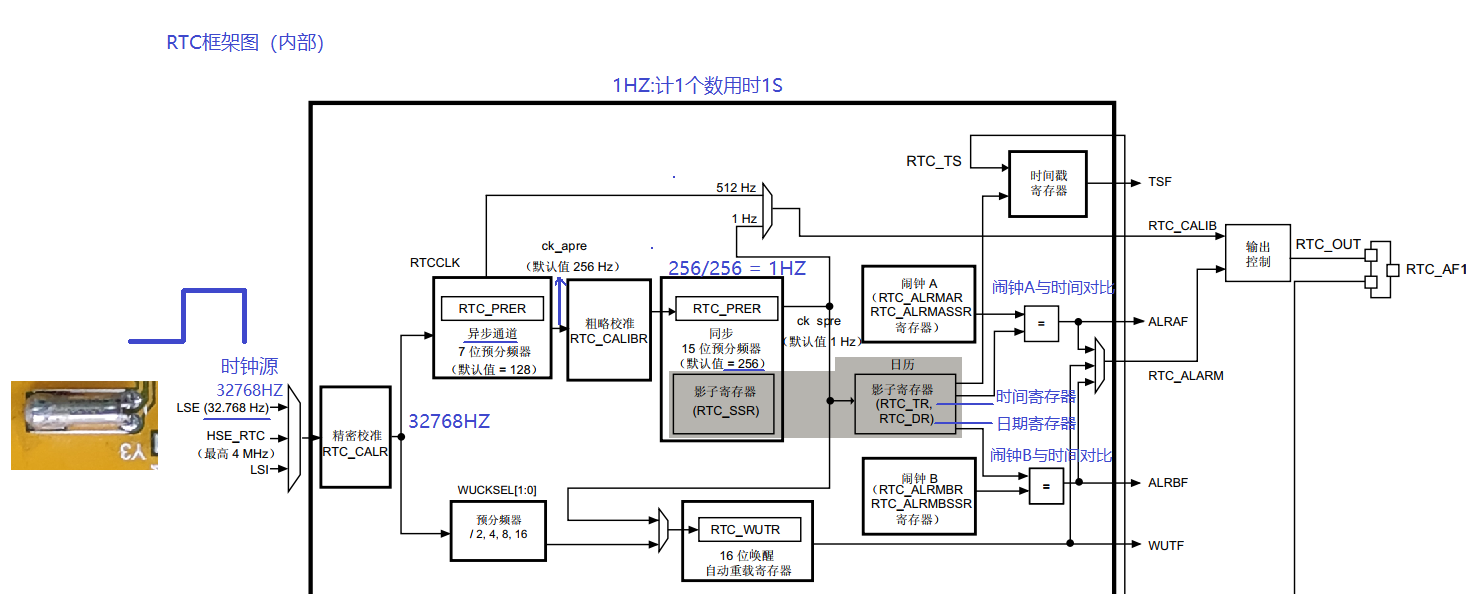
## RTC

### 1、RTC时间

RTC (Real Time Clock)：实时时钟

STM32 的 RTC 外设，实质是一个掉电后还继续运行的定时器。RTC是个独立的BCD定时器/计数器。提供一个日历时钟，两个可编程闹钟中断，以及一个具有中断功能的周期性可编程唤醒标志。RTC还包含用于管理低功耗模式的自动唤醒单元。

两个32位寄存器包含二进码十进制格式(BCD)的秒，分钟，小时（12或24小时制），星期几，日期，月份和年份。此外，还可以提供二进制的亚秒值。系统可以自动将月份的天数补偿为28，29（闰年），30,31天。并且还可以进行夏令时补偿。



选择外部时钟源，当主时钟发生故障时，RTC还能正常运行；且当主电源发生故障，RTC由钮扣电池进行独立供电

**RTC时间与日期配置流程：**

RTC时间与日期配置流程所需要的库函数文件：stm32f4xx\_rtc.c与stm32f4xx\_pwr.c

1、使能PWR时钟：RCC\_APB1PeriphClockCmd();

2、使能后备寄存器访问: PWR\_BackupAccessCmd();

3、配置RTC时钟源，使能RTC时钟：

RCC\_RTCCLKConfig();

RCC\_RTCCLKCmd();

如果使用LSE，要打开LSE：RCC\_LSEConfig(RCC\_LSE\_ON);

4、 初始化RTC(同步/异步分频系数和时钟格式)：RTC\_Init ();

5、 设置时间：RTC\_SetTime ();

6、设置日期：RTC\_SetDate()；

### 2、RTC闹钟

RTC 中断

所有 RTC 中断均与 EXTI（外部中断控制） 控制器相连。

要使能 RTC 闹钟中断，需按照以下顺序操作：

1. **将 EXTI 线 17 配置为中断模式并将其使能，然后选择上升沿有效。**

2. 配置 NVIC 中的 RTC\_Alarm IRQ 通道并将其使能。

3. 配置 RTC 以生成 RTC 闹钟（闹钟 A 或闹钟 B）。

**闹钟A配置流程**

1、RTC已经初始化好相关参数。

2、关闭闹钟：RTC\_AlarmCmd(RTC\_Alarm\_A,DISABLE);

3、配置闹钟参数：RTC\_SetAlarm();

4、开启闹钟：RTC\_AlarmCmd(RTC\_Alarm\_A,ENABLE);

5、开启配置闹钟中断：

RTC\_ITConfig();

EXTI\_Init();

NVIC\_Init();

6、编写中断服务函数：RTC\_Alarm\_IRQHandler();

## 三、函数说明

**void IWDG\_WriteAccessCmd(uint16\_t IWDG\_WriteAccess);**

函数功能：是否使能 IWDG\_PR and IWDG\_RLR寄存器

返回值：无

uint16\_t IWDG\_WriteAccess：是否使能

IWDG\_WriteAccess\_Enable

IWDG\_WriteAccess\_Disable

**void IWDG\_SetPrescaler(uint8\_t IWDG\_Prescaler)**

函数功能：设置看门预分频器

返回值：无

uint8\_t IWDG\_Prescaler：分频系数

IWDG\_Prescaler\_4

IWDG\_Prescaler\_8

IWDG\_Prescaler\_16

IWDG\_Prescaler\_32

IWDG\_Prescaler\_64

IWDG\_Prescaler\_128

IWDG\_Prescaler\_256

**void RCC\_RTCCLKConfig(uint32\_t RCC\_RTCCLKSource)**

**void IWDG\_SetReload(uint16\_t Reload)**

函数功能：设置重载值

返回值：无

uint16\_t Reload：重载值寄存器的值，范围：0~0xFFF

**void RCC\_RTCCLKConfig(uint32\_t RCC\_RTCCLKSource)**

函数功能：RTC时钟源配置

返回值：无

uint32\_t RCC\_RTCCLKSource：时钟源选择

RCC\_RTCCLKSource\_LSE

RCC\_RTCCLKSource\_LSI

RCC\_RTCCLKSource\_HSE\_Div2

RCC\_RTCCLKSource\_HSE\_Div3

RCC\_RTCCLKSource\_HSE\_Div4

RCC\_RTCCLKSource\_HSE\_Div5

RCC\_RTCCLKSource\_HSE\_Div6

RCC\_RTCCLKSource\_HSE\_Div7

RCC\_RTCCLKSource\_HSE\_Div8

RCC\_RTCCLKSource\_HSE\_Div9

RCC\_RTCCLKSource\_HSE\_Div10

RCC\_RTCCLKSource\_HSE\_Div11

RCC\_RTCCLKSource\_HSE\_Div12

RCC\_RTCCLKSource\_HSE\_Div13

RCC\_RTCCLKSource\_HSE\_Div14

RCC\_RTCCLKSource\_HSE\_Div15

RCC\_RTCCLKSource\_HSE\_Div16

RCC\_RTCCLKSource\_HSE\_Div17

RCC\_RTCCLKSource\_HSE\_Div18

RCC\_RTCCLKSource\_HSE\_Div19

RCC\_RTCCLKSource\_HSE\_Div20

RCC\_RTCCLKSource\_HSE\_Div21

RCC\_RTCCLKSource\_HSE\_Div22

RCC\_RTCCLKSource\_HSE\_Div23

RCC\_RTCCLKSource\_HSE\_Div24

RCC\_RTCCLKSource\_HSE\_Div25

RCC\_RTCCLKSource\_HSE\_Div26

RCC\_RTCCLKSource\_HSE\_Div27

RCC\_RTCCLKSource\_HSE\_Div28

RCC\_RTCCLKSource\_HSE\_Div29

RCC\_RTCCLKSource\_HSE\_Div30

RCC\_RTCCLKSource\_HSE\_Div31

**ErrorStatus RTC\_Init(RTC\_InitTypeDef\* RTC\_InitStruct);**

函数功能：RTC设置

返回值

成功：SUCCESS

失败：ERROR

RTC\_InitTypeDef\* RTC\_InitStruct：RTC结构体

typedef struct

{

uint32\_t RTC\_HourFormat; //小时制选择

uint32\_t RTC\_AsynchPrediv; //异步通道分频器

uint32\_t RTC\_SynchPrediv; //同步通道分频器

}RTC\_InitTypeDef;

**ErrorStatus RTC\_SetTime(uint32\_t RTC\_Format, RTC\_TimeTypeDef\* RTC\_TimeStruct);**

函数功能：RTC时间设置

返回值

成功：SUCCESS

失败：ERROR

uint32\_t RTC\_Format：存储格式

RTC\_Format\_BIN：二进制存储

RTC\_Format\_BCD：BCD编码存储

RTC\_TimeTypeDef\* RTC\_TimeStruct：时间结构体

typedef struct

{

uint8\_t RTC\_Hours; //时

uint8\_t RTC\_Minutes; //分

uint8\_t RTC\_Seconds; //秒

uint8\_t RTC\_H12; //上/下午

}RTC\_TimeTypeDef;

**ErrorStatus RTC\_SetDate(uint32\_t RTC\_Format, RTC\_DateTypeDef\* RTC\_DateStruct);**

函数功能：RTC日期设置

返回值

成功：SUCCESS

失败：ERROR

uint32\_t RTC\_Format：存储格式

RTC\_Format\_BIN：二进制存储

RTC\_Format\_BCD：BCD编码存储

RTC\_DateTypeDef\* RTC\_DateStruct:日期结构体

typedef struct

{

uint8\_t RTC\_WeekDay; //星期

uint8\_t RTC\_Month; //月

uint8\_t RTC\_Date; //日

uint8\_t RTC\_Year; //年

}RTC\_DateTypeDef;

/\*\*

\* @brief Set the specified RTC Alarm.

\* @note The Alarm register can only be written when the corresponding Alarm

\* is disabled (Use the RTC\_AlarmCmd(DISABLE)).

\* @param RTC\_Format: specifies the format of the returned parameters.

\* This parameter can be one of the following values:

\* @arg RTC\_Format\_BIN: Binary data format

\* @arg RTC\_Format\_BCD: BCD data format

\* @param RTC\_Alarm: specifies the alarm to be configured.

\* This parameter can be one of the following values:

\* @arg RTC\_Alarm\_A: to select Alarm A

\* @arg RTC\_Alarm\_B: to select Alarm B

\* @param RTC\_AlarmStruct: pointer to a RTC\_AlarmTypeDef structure that

\* contains the alarm configuration parameters.

\* @retval None

\*/

void RTC\_SetAlarm(uint32\_t RTC\_Format, uint32\_t RTC\_Alarm, RTC\_AlarmTypeDef\* RTC\_AlarmStruct)

函数说明：闹钟设置

返回值：无

uint32\_t RTC\_Format：存储格式

RTC\_Format\_BIN：二进制存储

RTC\_Format\_BCD：BCD编码存储

uint32\_t RTC\_Alarm：选择闹钟

RTC\_Alarm\_A：闹钟A

RTC\_Alarm\_B：闹钟B

RTC\_AlarmTypeDef\* RTC\_AlarmStruct:闹钟结构体

typedef struct

{

RTC\_TimeTypeDef RTC\_AlarmTime; //时间设置

uint32\_t RTC\_AlarmMask; //掩码位

uint32\_t RTC\_AlarmDateWeekDaySel; //选择日期还是星期设置闹钟

uint8\_t RTC\_AlarmDateWeekDay; //日期还是星期

}RTC\_AlarmTypeDef;

typedef struct

{

uint8\_t RTC\_Hours; //时

uint8\_t RTC\_Minutes; //分

uint8\_t RTC\_Seconds; //秒

uint8\_t RTC\_H12; //上/下午

}RTC\_TimeTypeDef;

## 四、技术应用

### 看门狗



### RTC

小天才手表



公交刷卡机

