AN0045
入门指南

AT32 WDT WWDT入门指南

前言

本入门指南详细阐述了如何使用 AT32 看门狗（WDT）和窗口看门狗（WWDT）。

注：本应用笔记对应的代码是基于雅特力提供的V2.x.x 板级支持包（BSP）而开发，对于其他版本BSP，需要注意使用上的区别。

支持型号列表：

支持型号	AT32 全系列

目录

表目录

图目录

1 看门狗简介

看门狗通常用来提高系统的稳定性。当因为一些特殊的情况导致程序跑飞，或者运行逻辑错误，而没有及时喂狗时，看门狗会将 MCU 重新复位，以达到自动从异常中恢复的效果。建议用户在所有应用中都使用看门狗，以提高系统稳定性。

AT32 单片机有两个看门狗：看门狗（WDT）和窗口看门狗（WWDT）：

- 看门狗（WDT）：一个 12 位的递减计数器，当计数器从某个值递减到 0 的时候，系统会产生复位，如果在计数器递减到 0 之前刷新了递减计数器，那么就不会产生复位。
- 窗口看门狗（WWDT）：一个 7 位的递减计数器，当计数器从某个值递减到 0x3F 的时候，系统会产生复位，如果在规定时间刷新了计数器（窗口时间内），那么就不会产生复位。

1.1 各个型号差异

各型号的窗口看门狗（WWDT）相同，程序兼容。

各型号的看门狗（WDT）基本功能相同，只是各个型号之间，可能去掉了更高级的窗口功能或者低功耗下可选的停止运行功能，其余功能相同并且程序兼容。

表 1. 各型号看门狗（WDT）差异

型号	窗口功能	DEEPSLEEP、STANDY 模式下可选停止运行
AT32F403xx	×	×
AT32F403Axx	×	×
AT32F407xx	×	×
AT32F413xx	×	×
AT32F415xx	×	×
AT32F421xx	×	×
AT32F425xx	√	√
AT32F435xx	√	√
AT32F437xx	√	√
AT32L021xx	√	√

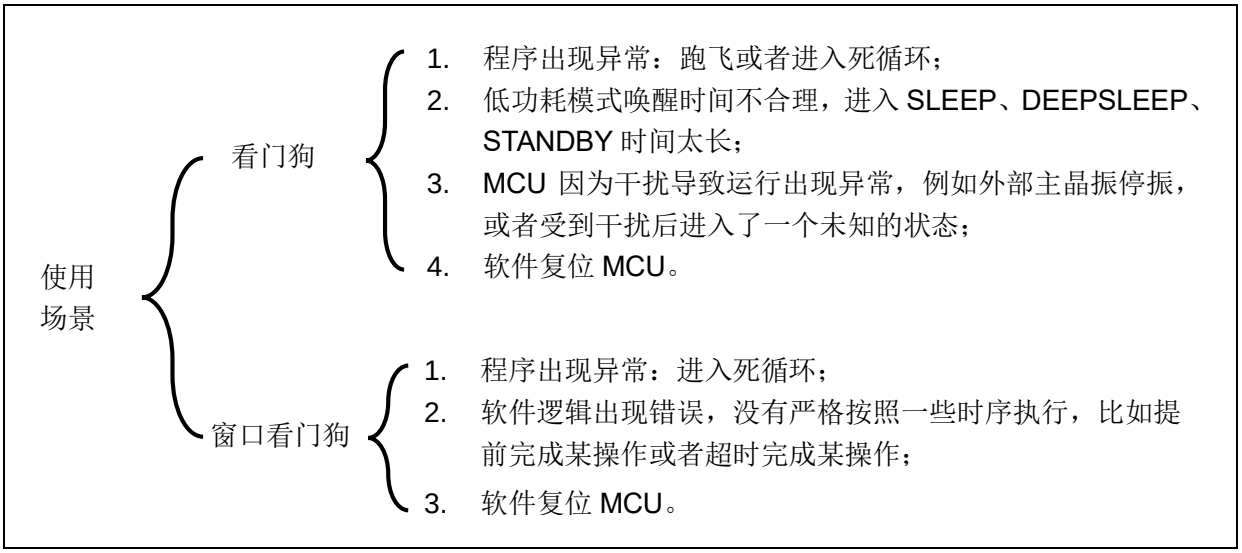
√：表示支持该功能，且功能相同。

×：表示不支持该功能。

1.2 使用场景对比

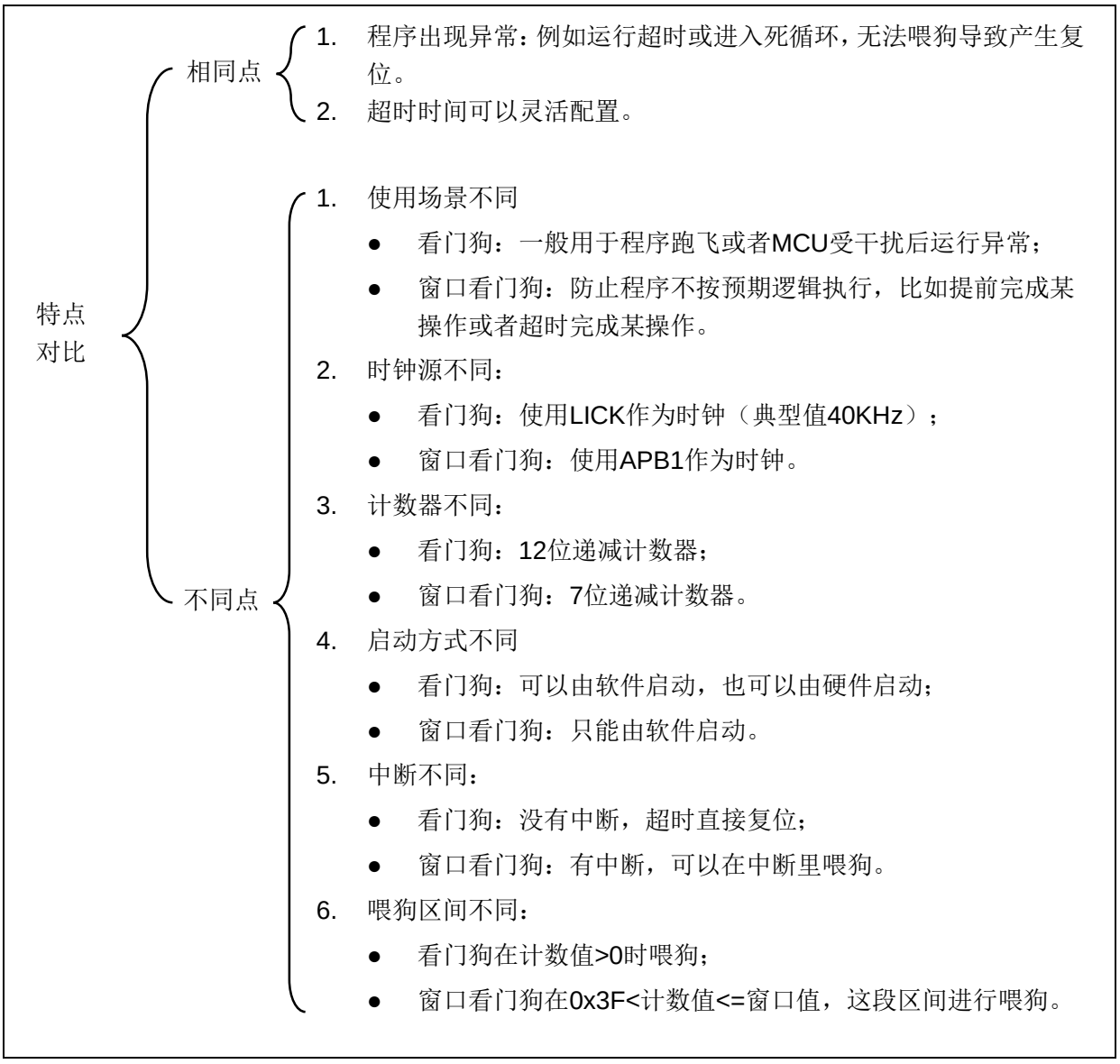
看门狗（WDT）和窗口看门狗（WWDT）作为两种不同类型的看狗，有着不用的适用环境。

图 1. WDT 与 WWDT 使用场景对比



1.3 特点对比

图 2. WDT 与 WWDT 特点对比



2 看门狗 WDT

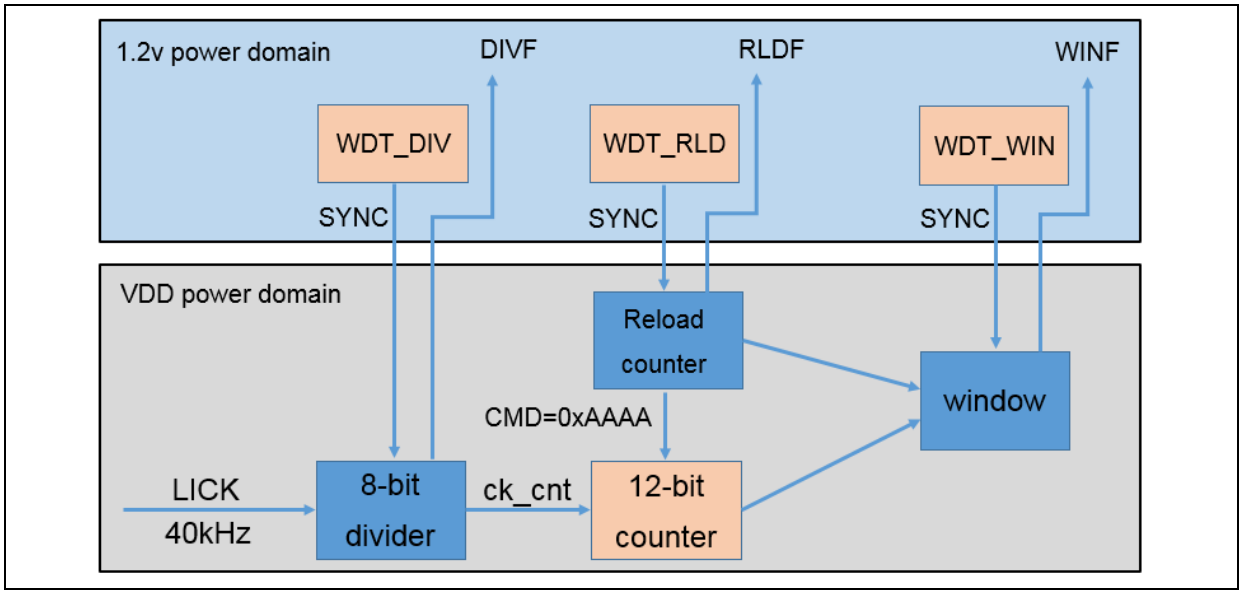
2.1 寄存器访问

状态寄存器

看门狗功能位于两个不同的区域，寄存器部分位于 1.2V 电压域，计数逻辑部分位于 VDD 电压域，所以看门狗能够在 SLEEP、DEEPSLEEP、STANDBY 模式下运行。

对看门狗寄存器的写操作位于 1.2V 电压域，所以当写了寄存器之后，还需要将寄存器值同步到 VDD 电压域。每一个寄存器都有一个同步标志指示同步操作是否完成。每一次同步时间最多需要 4 个 LICK 时钟，大约 125us。当写了寄存器之后对应的同步标志自动置 1，当同步完成了之后标志自动清 0，在同步标志清零之前，不允许再写此寄存器。

图 3. 看门狗框图



RLDF：当该位为 1 时，表示重装载值的同步正在进行中；当为 0 时，表示该过程执行完成。

DIVF：当该位为 1 时，表示预分频器值的同步正在进行中；当为 0 时，表示该过程执行完成。

WINF：当该位为 1 时，表示窗口值的同步正在进行中；当为 0 时，表示该过程执行完成。

标志获取函数：

```
flag_status wdt_flag_get(uint16_t wdt_flag);
```

寄存器写保护

看门狗寄存器受到写保护，在写寄存器前需要先解锁写保护，写命令寄存器CMD = 0x5555解锁写保护。当写一个其他值，将重新开启读保护。受读保护的寄存器如下表所示：

表 2. 看门狗寄存器

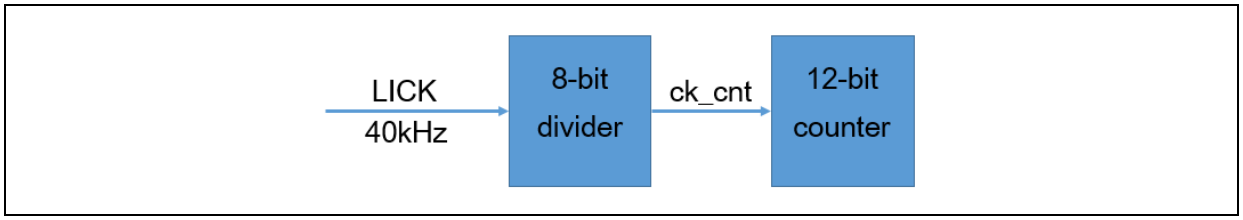
寄存器	简称	同步完成标志	是否受写保护	解锁方式
命令寄存器	WDT_CMD	-	否	-
预分配寄存器	WDT_DIV	DIVF	是	写 CMD=0x5555
重载寄存器	WDT_RLD	RLDF	是	写 CMD=0x5555
状态寄存器	WDT_STS	-	否	-
窗口寄存器	WDT_WIN	WINF	是	写 CMD=0x5555

寄存器解锁写保护函数：

```
void wdt_register_write_enable( confirm_state new_state);
```

2.2 时钟结构

图 4. 看门狗时钟



看门狗计数器由LICK时钟驱动，经过8位的预分频器得到递减计数器时钟。LICK是内部RC时钟，其典型值为40kHz，范围为30kHz~60kHz之间（详情请见对应型号的数据手册）。所以超时时间也是在一定区间内，使用时应注意在超时时间配置上应该留有余量，如果需要获得较为精确的看门狗超时时间，可以先通过定时器测量出LICK频率，然后再根据实际的LICK频率计算超时时间。

通过寄存器DIV[2：0]配置配置不同的预分频值，可配置预分频值为4、8、16、32、64、128、256。

$$f_{ck\_cnt} = \frac{f_{LICK}}{\text{分频值}}$$

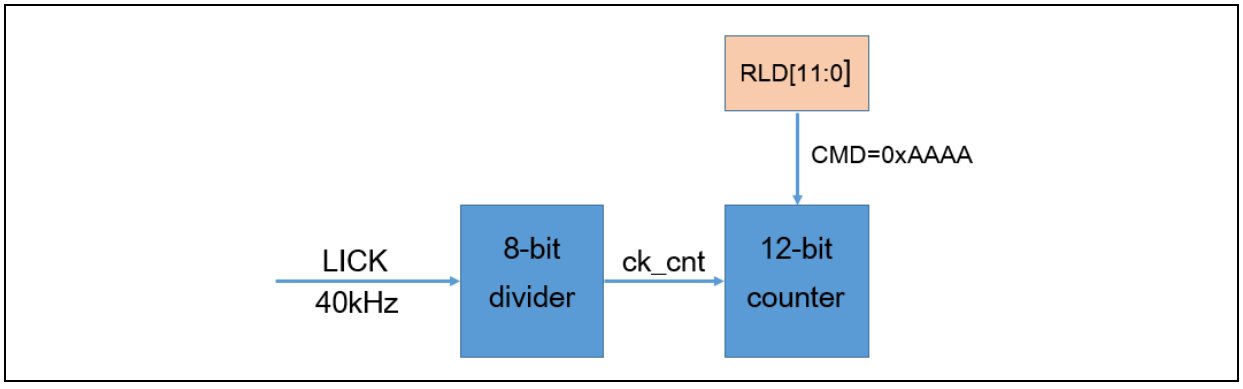
分频设置函数：

```
void wdt_divider_set(wdt_division_type division);
```

2.3 计数器

看门狗的计数器是一个12位的递减计数器，最大值为0xFFF。当开启看门狗后，计数值将从设定的值开始递减，当递减到0时，产生系统复位。

图 5. 看门狗重载



计数值通过重载寄存器RLD设置，在分频值确定的情况下，该值的大小决定了看门狗复位的时间长短，每当往命令寄存器WDT_CMD写入0xAAAA时，该寄存器的值便会更新到递减计数器中（此操作通常称为喂狗），喂狗的操作需要在计数器递减到0之前进行，不然会发生复位。

看门狗复位时间计算如下：

$$\text{超时时间} = (\text{RLD} + 1) \times \frac{\text{分频值}}{f_{LICK}}$$

表 3. 看门狗复位时间（LICK = 40kHz）

分频值	最短复位时间（ms） RLD[11：0] = 0x000	最长复位时间（ms） RLD[11：0] = 0xFFF
4	0.1	409.6
8	0.2	819.2
16	0.4	1638.4
32	0.8	3276.8
64	1.6	6553.6
128	3.2	13107.2
256	6.4	26214.4

重载值设置函数：

```
void wdt_reload_value_set(uint16_t reload_value);
```

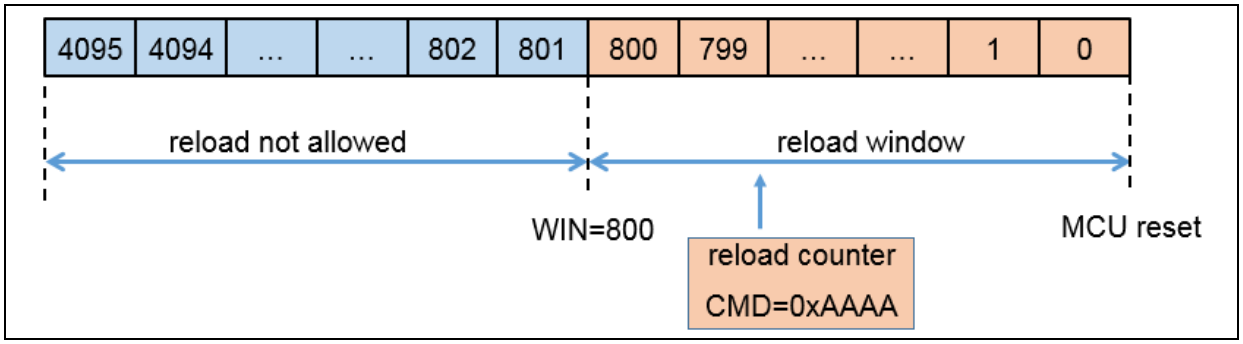
重载看门狗计数器（喂狗）函数：

```
void wdt_counter_reload(void);
```

2.4 窗口功能

当 WIN[11：0]设置为非默认值（0xFFF）将开启窗口功能。当在计数值大于窗口值时重载计数器值将会产生系统复位，例如将 WIN 值设置成 800 时允许重载的窗口时间如下图所示。

图 6. 窗口功能



窗口设置函数：

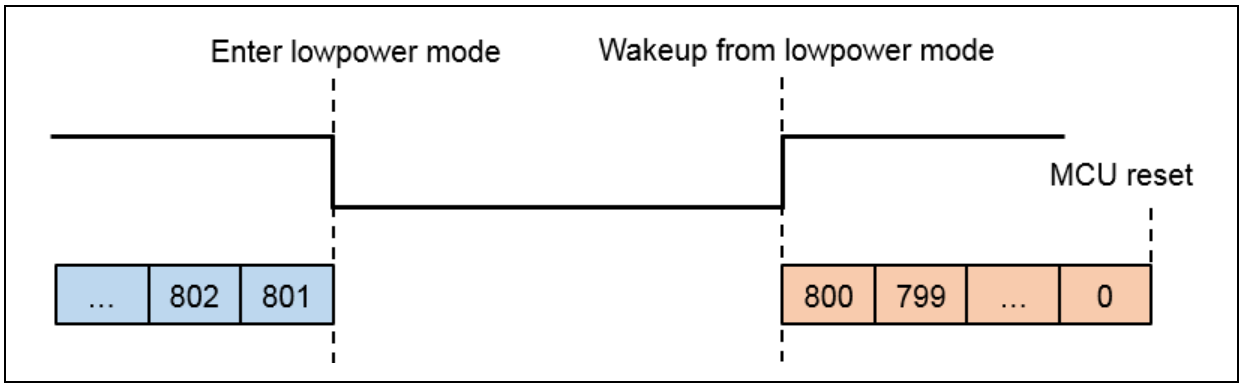
```
void wdt_window_counter_set(uint16_t window_cnt);
```

2.5 低功耗停止计数

看门狗能够在 SLEEP、DEEPSLEEP、STANDBY 模式下运行，用户可选择进入 DEEPSLEEP、STANDBY 模式后计数器是否停止计数，可由用户系统数据区中的 nWDT_DEPSLP、nWDT_STDBY 位配置。

如果设置了停止计数，当进入了 DEEPSLEEP、STANDBY 模式后，看门狗计数器停止递减，意味着看门狗在这两种低功耗模式下不会发生复位，当从这两种模式唤醒后，计数器从进入时的值继续递减。

图 7. 低功耗停止计数功能



用户系统数据擦除函数：

```
flash_status_type flash_user_system_data_erase(void);
```

用户系统数据配置函数：

```
flash_status_type flash_ssb_set(uint8_t usd_ssb);
```

低功耗停止功能使用示例：

```
/* 用户系统数据擦除 */
flash_user_system_data_erase();
/* 在DEEPSLEEP和STANDBY下停止计数 */
flash_ssb_set(USD_WDT_ATO_DISABLE | USD_DEPSLP_NO_RST | USD_STDBY_NO_RST |
FLASH_BOOT_FROM_BANK1 | USD_WDT_DEPSLP_STOP | USD_WDT_STDBY_STOP);
```

2.6 启动看门狗

看门狗启动方式分为硬件启动和软件启动，当看门狗启动了之后不能被关闭，除非发生复位。

软件启动方式

向命令寄存器写入 0xCCCC，启用看门狗。

看门狗软件使能函数：

```
void wdt_enable(void);
```

硬件启动方式

硬件启动则需通过配置用户系统数据区的nWDT_ATO_EN位来实现，使能硬件看门狗后，看门狗将在上电复位后自动开始运行。

硬件启动看门狗使用示例：

```
/* 用户系统数据擦除 */
flash_user_system_data_erase();
/* 使能硬件看门狗 */
flash_ssb_set(USD_WDT_ATO_ENABLE | USD_DEPSLP_NO_RST | USD_STDBY_NO_RST |
FLASH_BOOT_FROM_BANK1 | USD_WDT_DEPSLP_CONTINUE |
USD_WDT_STDBY_CONTINUE);
```

2.7 使用方法

看门狗一般用于检测程序跑飞或者死循环，比如一个正常的程序运行完的时间是10ms，可以设置看门狗超时的时间为20ms，当程序运行完便立即进行喂狗操作，这样便不会产生复位，超过20ms还未喂狗时，说明产生了故障，此时会复位MCU。

例如：

要设置WDT超时时间为20ms，那么可以设置预分频值为4，计数值为200

$$\text{超时时间} = \text{RLD} \times \frac{\text{预分频系数}}{f_{LICK}} = 200 \times \frac{4}{40000} = 20ms$$

配置步骤：

1. 禁止寄存器写保护

```
wdt_register_write_enable(TRUE);
```

2. 设置预分频值为4

```
wdt_divider_set(WDT_CLK_DIV_4);
```

3. 设置重载值为200

```
wdt_reload_value_set(200 - 1);
```

4. 启用看门狗

```
wdt_enable();
```

5. 在应用程序中重载计数器

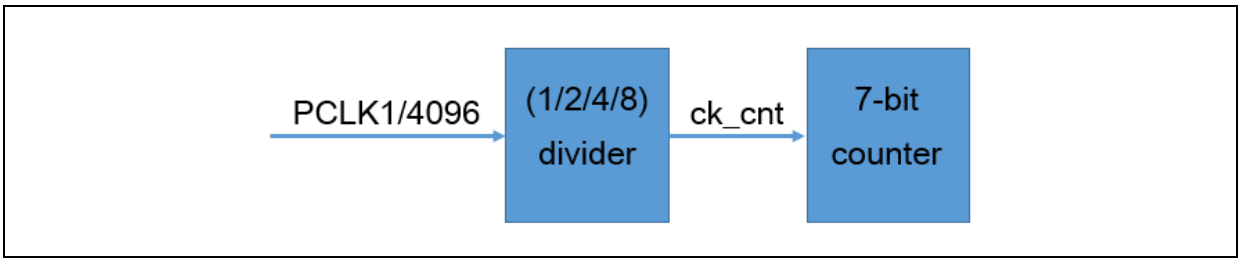
```
wdt_counter_reload();
```

3 窗口看门狗 WWDT

窗口看门狗（WWDT）主要作用是用来检测软件逻辑是否按照预期执行，其喂狗时间是一个有上下范围内，可以通过相关的寄存器，设定其上限时间和下限时间，喂狗的时间不能过早也不能过晚（当递减计数器的值小于0x40，或者当递减计数器在窗口外被刷新时产生复位）。

3.1 时钟结构

图 8. 窗口看门狗时钟



窗口看门狗时钟由APB1时钟分频而来，由于APB1_CLK的精确性，因此窗口看门狗时间精度很高。APB1时钟先经过4096分频后，再送到预分频器，最后提供给7位递减计数器CNT[6:0]。可以配置不同的预分频值来获得不同的时钟，通过DIV[1：0]可配置预分频值取值范围为1、2、4、8。

$$f_{ck\_cnt} = \frac{f_{PCLK}}{4096 \times 2^{DIV[1:0]}}$$

分频设置函数：

```
void wwdt_divider_set(wwdt_division_type division);
```

3.2 计数器

窗口看门狗的计数器是一个7位的递减计数器，最大值为0x7F，当开启看门狗后，计数值将从设定的值开始递减，当递减到0x3F时，产生系统复位。

$$\text{超时时间} = (\text{CNT}[5:0] + 1) \times \frac{4096 \times 2^{DIV[1:0]}}{f_{PCLK}}$$

表 4. 窗口看门狗复位时间（PCLK = 72MHz）

分频值	最短复位时间（ms） CNT [6：0] = 0x40	最长复位时间（ms） CNT [6：0] = 0x7F
1	56.5μs	3.64ms
2	113.5μs	7.28ms
4	227.5μs	14.56ms
8	455μs	29.12ms

计数值设置函数：

```
void wwdt_counter_set(uint8_t wwdt_cnt);
```

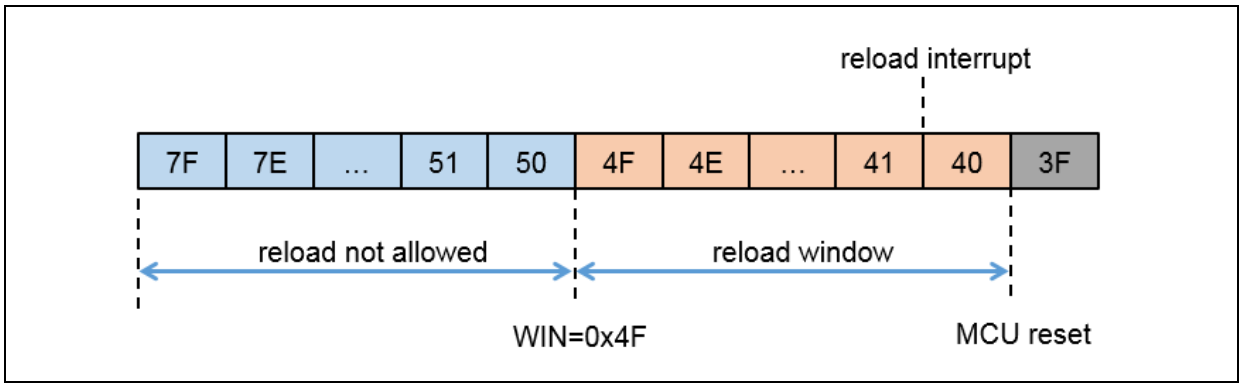
3.3 窗口功能

窗口的值（WIN[6：0]）可以自由设定，最大值为（0x7F），最小值必须大于下窗口的0x40，所以

取值范围为64~127（即：0x40~0x7F）；只有当递减计数器的值小于等于窗口值时，才允许刷新递减计数器，否则将会产生复位。

为了便于喂狗，应用程序也可以利用重载计数器中断（RLDIEN）进行喂狗。当递减计数器到达0x40时，则产生中断，在相应的中断服务程序中重新设置计数器。

图 9. 窗口功能



如上图所示当配置窗口值为0x4F时，不允许刷新的窗口为0x7F~0x50，允许刷新的窗口为0x4F~0x40。

重载标志清除函数：

```
void wwdt_flag_clear(void);
```

重载标志获取函数：

```
flag_status wwdt_flag_get(void);
```

重载中断使能函数：

```
void wwdt_interrupt_enable(void);
```

窗口设置函数：

```
void wwdt_window_counter_set(uint8_t window_cnt);
```

3.4 看门狗使能

设置WWDTEN=1使能窗口看门狗，当窗口看门狗被打开后不能被关闭，直到复位。为了避免使能看门狗后立即发生复位，在使能看门狗时，应该同时配置看门狗计数值。

窗口看门狗使能函数：

```
void wwdt_enable(uint8_t wwdt_cnt);
```

3.5 使用方法

窗口看门狗一般用于检测逻辑运行是否正常，比如一个正常的程序执行完的时间是10ms，当程序在10ms以前执行完说明出现了逻辑错误，可以设置看门狗窗口值为9ms，当程序在9ms以前进行喂狗时，说明程序产生了故障，此时会产生一个复位。

例如：

当PCLK1=36MHz时，要设置WWDT超时时间为9ms，那么可以设置预分频值为4，总的分频为4x4096=16384。计数值为127，窗口值为108，此时从计数值减到窗口值时间约为9.1ms。

$$窗口时间= \ (\mathrm{CNT}-\mathrm{WIN})\times\frac{4096\times 2^{DIV[1:0]}}{f_{PCLK1}}=((127-108)+1)\times\frac{4096\times 2^{2}}{36MHz}=9.1ms$$

$$复位时间= \ (\mathrm{CNT}-0\mathrm{x3F})\times\frac{4096\times 2^{DIV[1:0]}}{f_{PCLK1}}=(127-63)\times\frac{4096\times 2^{2}}{36MHz}=29.1ms$$

所以允许喂狗时间为9.1~29.1ms，不允许喂狗时间为0~9.1ms。

配置步骤：

1. 开启窗口看门狗APB1时钟

```
crm_periph_clock_enable(CRM_WWDT_PERIPH_CLOCK, TRUE);
```

2. 设置预分频值为4，总的分频为4096x4=16384

```
wwdt_divider_set(WWDT_PCLK1_DIV_16384);
```

3. 设置窗口值为108

```
wwdt_window_counter_set(108);
```

4. 启用看门狗

```
wwdt_enable(127);
```

5. 在应用程序中重载计数器

```
wwdt_counter_set(127);
```

备注：需要在0x3F<递减计数器<=窗口值执行

4 案例 看门狗 WDT 使用

4.1 功能简介

演示看门狗（WDT）功能使用。

4.2 资源准备

1) 硬件环境:

对应产品型号的 AT-START BOARD

2) 软件环境

project\at_start_f4xx\examples\wdt\wdt_reset

注：所有project都是基于keil 5而建立，若用户需要在其他编译环境上使用，请参考AT32xxx_Firmware_Library_V2.x.x\project\at_start_xxx\templates中各种编译环境（例如IAR6/7,keil 4/5）进行简单修改即可。

4.3 软件设计

1) 配置流程

- 初始化看门狗
- 在主程序中喂狗

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
  /* 初始化系统时钟 */
  system_clock_config();

  /* 初始 START 板子资源 */
  at32_board_init();

  /* 获取看门狗复位标志 */
  if(crm_flag_get(CRM_WDT_RESET_FLAG) != RESET)
  {
    /* 从看门狗复位 */
    crm_flag_clear(CRM_WDT_RESET_FLAG);

    at32_led_on(LED4);
  }
  else
  {
    /* 从其他模式复位 */
    at32_led_off(LED4);
  }

  /* 解锁写保护 */
```

```
  wdt_register_write_enable(TRUE);

  /* 设置看门狗分频 */
  wdt_divider_set(WDT_CLK_DIV_4);

  /* 设置看门狗重载值 */
  wdt_reload_value_set(3000 - 1);

  /* 使能看门狗 */
  wdt_enable();

  while(1)
  {
    /* 重载看门狗 */
    wdt_counter_reload();

    at32_led_toggle(LED3);

    delay_ms(200);

    if(at32_button_press() == USER_BUTTON)
    {
      while(1);
    }
  }
}
```

4.4 实验效果

- 正常运行时看门狗不会复位，当按下按键后，停止喂狗，导致 MCU 复位。
- 复位后，如果检查到是看门狗复位则 LED4 点亮，否则 LED4 不亮。

5 案例 窗口看门狗 WWDT 使用

5.1 功能简介

演示窗口看门狗（WWDT）功能使用。

5.2 资源准备

1) 硬件环境:

对应产品型号的 AT-START BOARD

2) 软件环境

project\at_start_f4xx\examples\wwdt\wwdt_reset

注：所有project都是基于keil 5而建立，若用户需要在其他编译环境上使用，请参考AT32xxx_Firmware_Library_V2.x.x\project\at_start_xxx\templates中各种编译环境（例如IAR6/7,keil 4/5）进行简单修改即可。

5.3 软件设计

1) 配置流程

- 初始化窗口看门狗
- 在主程序中喂狗

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
  /* 初始化系统时钟 */
  system_clock_config();

  /* 初始 START 板子资源 */
  at32_board_init();

  /* 获取窗口看门狗复位标志 */
  if(crm_flag_get(CRM_WWDT_RESET_FLAG) != RESET)
  {
    /* 从窗口看门狗复位 */
    crm_flag_clear(CRM_WWDT_RESET_FLAG);

    at32_led_on(LED4);
  }
  else
  {
    /* 从其他模式复位 */
    at32_led_off(LED4);
  }

  /* 使能窗口看门狗时钟 */
```

```
  crm_periph_clock_enable(CRM_WWDT_PERIPH_CLOCK, TRUE);

  /* 设置窗口看门狗分频 */
  wwdt_divider_set(WWDT_PCLK1_DIV_16384);

  /* 设置窗口值 */
  wwdt_window_counter_set(0x6F);

  /* 使能窗口看门狗 */
  wwdt_enable(0x7F);

  while(1)
  {
    at32_led_toggle(LED3);

    delay_ms(6);

    /* 重载看门狗 */
    wwdt_counter_set(0x7F);

    if(at32_button_press() == USER_BUTTON)
    {
      while(1);
    }
  }
}
```

5.4 实验效果

- 正常运行时窗口看门狗不会复位，当按下按键后，停止喂狗，导致 MCU 复位。
- 复位后，如果检查到是窗口看门狗复位则 LED4 点亮，否则 LED4 不亮。

6 文档版本历史

表 5. 文档版本历史

日期	版本	变更
2021.12.15	2.0.0	最初版本
2022.06.06	2.0.1	修正flash_ssb_set函数

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独力负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。