**第十三章 软件狗**

本章主要介绍STM32中的两种看门狗类型，分别为独立看门狗和窗口看门狗。在本章中，我们将完成两个实验，一是验证不及时喂狗，系统将复位重启的实验，二是验证窗口看门狗的功能的实验。本章分为如下几个部分：

13.1 看门狗简介

13.2 硬件设计

13.3 软件设计

13.4 下载验证

13.1 看门狗简介

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)，分为独立看门狗和窗口看门狗。

表13.1.1 对比表

|  |  |  |
| --- | --- | --- |
| 对比点 | 独立看门狗 | 窗口看门狗 |
| 时钟源 | LSI(40KHz或32KHz) | PCLK1或PCLK3 |
| 复位条件 | 递减计数到0 | 计数值大于W[6:0]值喂狗或减到0x3F |
| 中断 | 没有中断 | 计数值减到0x40可产生中断 |
| 递减计数器  位数 | 12位（最大计数范围：4096~0） | 7位（最大计数范围：127~63） |
| 应用场合 | 防止程序跑飞，死循环，死机 | 检测程序时效，防止软件异常 |

* 独立看门狗(Independent watchdog)，IWDG

本质：能够产生**系统复位信号**的计数器。

特性：递减的计数器，时钟由独立的RC振荡器提供（可在待机和停止

模式下运行），看门狗被激活后，当递减计数器计数到0x000时产生

复位。

喂狗：在计数器计到0之前，重装载计数器的值，防止复位。

作用：主要用于检测外界电磁干扰，或**硬件**异常导致的程序跑飞问题。

应用：在一些需要高稳定性的产品中，并且对时间精度要求较低的场合。

**CNT**

**递减计数器**

**RELOAD**

**重装载寄存器**

CNT = RELOAD

**复位**

CNT==0？

**PSC**

**预分频器**

IWDG CLK

Src CLK

喂狗

图13.1.1 IWDG工作原理

**CPU必须及时喂狗，否则系统复位重启！**

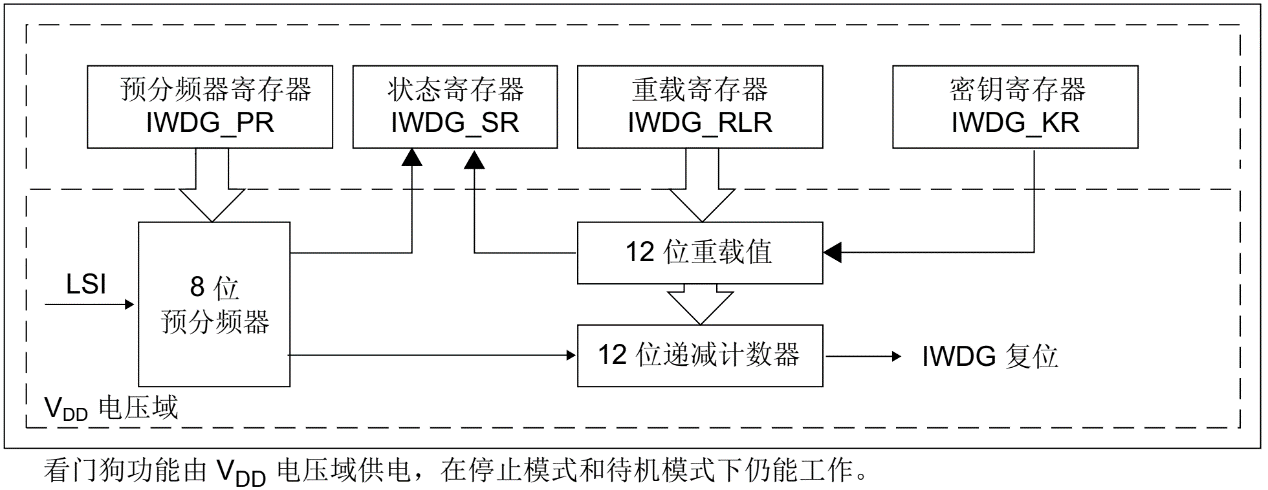
****

图13.1.2 IWDG框图

独立看门狗时钟： 独立看门狗是由专门的低速总线进行驱动，即**LSI总线** （LSI时钟并不精准，F1用**时钟频率40KHz**），它可以在主时钟故障的情况下仍然可以工作。启用独立看门狗后，LSI时钟会自动开启。

计数器：独立看门狗的计数器是一个12位的递减计数器，最大值为0XFFF，当计数器减到0时，会产生一个复位信号：IWDG\_RESET，让程序重新启动运行，如果在计数器减到0之前刷新了计数器的值的话，就不会产生复位信号，重新刷新计数器值的这个动作我们俗称喂狗。

重装载寄存器：重装载寄存器是一个 12 位的寄存器，里面装着要刷新到计数器的值，这个值的大小决定着独立看门狗的溢出时间。超时时间 Tout = (4\*2^prer) / 40 \* rlr，prer是预分频器寄存器的值，rlv 是重装载寄存器的值。

键寄存器：键寄存器（IWDG\_KR）写 0xCCCC 来启动看门狗是属于软件启动的方式，一旦独立看门狗启动，它就关不掉，只有复位才能关掉。

表13.1.3 键寄存器

|  |  |
| --- | --- |
| 键值 | 键值作用 |
| 0xAAAA | 把RLR的值重装载到CNT |
| 0x5555 | 允许访问IWDG\_PR和IWDG\_RLR寄存器 |
| 0xCCCC | 启动IWDG |

**IWDG溢出时间计算：**

是看门狗溢出时间 是看门狗的时钟源频率

是看门狗预分频系数 IWDG\_PR的值

rlr是看门狗重装载值

**操作步骤：**①取消寄存器写保护（向IWDG\_KR写入0x5555）；

②设置独立看门狗的预分频系数和重装载值；

③重载计数值喂狗（向IDWG\_KR写入0xAAAA）；

④启动看门狗（向IWDG\_KR写入0xCCCC）；

⑤应用程序喂狗。

* 窗口看门狗(Window watchingdog)，WWDG

本质：能产生**系统复位信号**和**提前唤醒中断**的计数器

特性：递减的计数器。当递减计数器值从 0x40减到0x3F时复位（即T6位跳变到0），计数器的值大于W[6:0]值时喂狗会复位，提前唤醒中断：当递减计数器等于 0x40 时可产生。

喂狗：在窗口期内重装载计数器的值，防止复位。

作用：用于监测单片机程序运行时效是否精准，主要检测**软件**异常。

应用：需要精准检测程序运行时间的场合。

计数器初始值



可产生中断

**窗口下限值,**

**产生复位**

W[6:0] ≥窗口期＞0x3F

窗口期，可以喂狗

非窗口期，喂狗则会复位

**窗口上限值**

计数值递减

图13.1.4 WWDG工作原理

注意：W[6:0]必须大于窗口下限值0x3F，否则无窗口期

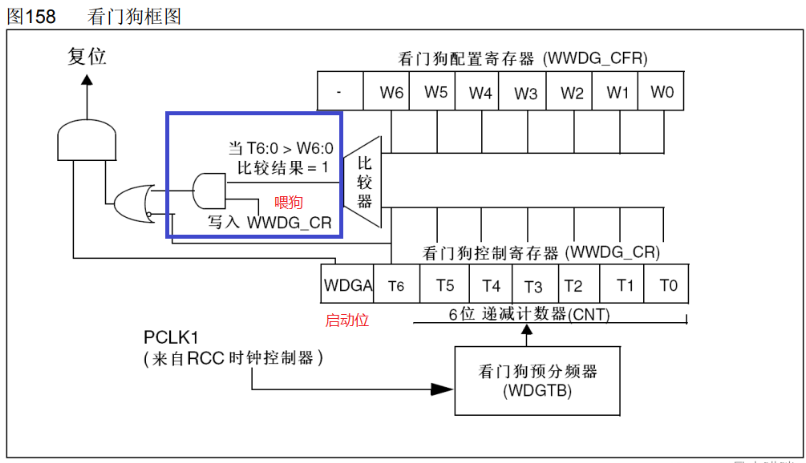


图13.1.5 看门狗框图

时钟源：F103：PCLK1( 36MHz )。



图13.1.6 WWDG最短最长超时时间

产生复位信号的逻辑分析：

第一条途径：

* 将WWDG模块启动，也就是将WDGA位置1；
* 计数器中的值T6:0与上窗口中的值W6:0通过比较器进行比较，当T6:0大于W6:0时产生的结果是1，这个结构与喂狗后的结果通过与门后的结果仍然是1；
* 上面产生的1通过或门后还是1，并且与WDGA产生的1经过与门产生复位信号。

第二条途径：

* 将WWDG模块启动，也就是将WDGA位置1；
* 当计数器中的T6位变成0以后，通过或非门变成1；
* 产生的1与WDGA产生的1经过与门后产生复位信号。

WWDG的喂狗可以在中断中实现的，该模块有提前唤醒中断功能。

**WWDG超时时间计算：**

是WWDG超时时间（没喂狗） F是WWDG的时钟源频率

4096是WWDG固定的预分频系数 T[5:0]是WWDG计数器低6位

是WWDG\_CFR寄存器设置的预分频系数值

在实验代码中注释部分，详细解释了如何计算非窗口期时间和窗口期时间。

**操作步骤**：①使能看门狗时钟: RCC\_APB1PeriphClockCmd()；

②设置分频系数: WWDG\_SetPrescaler()；

③设置上窗口值: WWDG\_SetWindowValue()；

④开启提前唤醒中断并分组(可选):

WWDG\_SetWindowValue()、NVIC\_Init()：

⑤使能看门狗: WWDG\_Enable()；

⑥喂狗: WWDG\_SetCounter()；

⑦编写中断服务函数: WWDG\_IRQHandler(),中断函数中还需要清除中断标志位WWDG\_ClearFlag();。

13.2 硬件设计

本章需要用到的硬件资源有：

1）极风STM32开发板

2）STLINK下载器

下面介绍一下STM32开发板和STLINK下载器的连接，STLINK的3.3V、SWCLK、SWDIO、GND分别连在STM32开发板的3V3、CLK、DIO、GND上。总体连接实物图如图13.2.1所示：

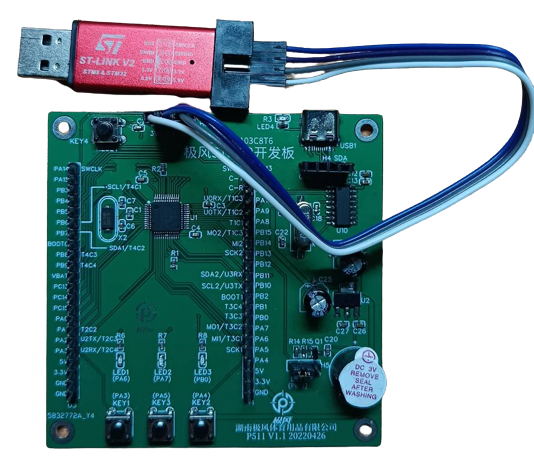


图13.2.1 总体连接实物图

13.3 软件设计

* 验证不及时喂狗，系统将复位重启的实验(IWDG)

1. 程序实现的功能

验证不及时喂狗，系统将复位重启的实验——按KEY1键来喂狗，每隔1s至少喂狗一次，此时LED1常亮，打开串口，可看见Yes，喂狗成功；如果没有按KEY1键，那么LED1闪烁，因为初始状态LED1是灭的，打开串口，可看见No，未喂狗。

1. 程序的实现

打开**实验 1 IWDG\motor.uvprojx**，我们可以看见工程中拥有的主要文件有iwdg.c、iwdg.h、led.c、led.h、key.c、key.h、main.c。iwdg.c 文件存放 iwdg 驱动代码，led.c 文件存放 led 驱动代码，key.c 文件存放 key驱动代码，main.c 文件存放应用代码。这里我们主要介绍iwdg.c、iwdg.h和main.c文件。

打开iwdg.c文件，代码如下：

#include "iwdg.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数全称:

\* void IWDG\_Init(uint8\_t prer,uint16\_t rlr)

\*

\* 辅助解释：

\* prer:分频数

\* 分频因子=4\*2^prer.最大值只能是256!

\* rlr:重装载寄存器值

\* 溢出时间计算:Tout=((4\*2^prer)\*rlr)/40 (ms).

\*

\* 函数作用:

\* 初始化独立看门狗

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void IWDG\_Init(uint8\_t prer,uint16\_t rlr)

{

IWDG\_WriteAccessCmd(IWDG\_WriteAccess\_Enable); //取消写保护

IWDG\_SetPrescaler(prer); //设置IWDG预分频值

IWDG\_SetReload(rlr); //设置IWDG重装载值

IWDG\_ReloadCounter(); //按照IWDG重装载寄存器的值重装载IWDG计数器

IWDG\_Enable(); //使能IWDG

}

下面我们看看头文iwdg.h的代码，代码中包含一些头文件定义，数据类型、函数声明，代码如下：

#ifndef \_\_IWDG\_H

#define \_\_IWDG\_H

#include "stdlib.h"

#include "stm32f10x.h"

/\* 初始化 \*/

void IWDG\_Init(uint8\_t prer,uint16\_t rlr);

#endif

最后，我们在main函数里编写应用代码，main.c文件如下：

#include "stm32f10x.h"

#include <string.h>

#include <stdlib.h>

#include "delay.h"

#include "usart.h"

#include "led.h"

#include "key.h"

#include "iwdg.h"

int main(void)

{

uint8\_t key;

uart\_init(115200);

LED\_Init(); //初始化LED

KEY\_Init(); //按键初始化

delay\_ms(200); //延时200ms再初始化看门狗,LED1的变化"可见"

IWDG\_Init(4,625); // Tout=((4\*2^4)\*625)/40 (ms) = 1000 ms溢出时间为1s

GPIO\_SetBits(GPIOA, GPIO\_Pin\_6);

printf("No!\r\n"); //未喂狗

while(1)

{

key = KEY\_Scan(); /\* 得到键值 \*/

if(key == 1) /\* 按下KEY1 \*/

{

IWDG\_ReloadCounter();//如果按下KEY1,则喂狗

printf("Yes!\r\n");//喂狗

}

}

}

至此，实验 1 IWDG的软件设计部分就完成了。

* 验证窗口看门狗的功能的实验

（1）程序实现的功能

验证窗口看门狗的实验——方法① 通过LED1来指示是否喂狗成功：若LED1先灭500ms后一直亮，表示喂狗成功；若LED1闪烁表示喂狗失败。方法② 在中断服务函数中进行喂狗：若看见LED2闪烁，表示喂狗成功；若在中断服务函数中增加延时（由于设置的WDGTB = 3，最小超时值为910us），所以延时1ms后喂狗会失败，LED2会不亮。

（2）程序的实现

打开**实验2 WWDG\motor.uvprojx**，我们可以看见工程中拥有的主要文件有wwdg.c、wwdg.h、led.c、led.h、key.c、key.h、main.c。iwdg.c 文件存放 iwdg 驱动代码，led.c 文件存放 led 驱动代码，key.c 文件存放 key驱动代码，main.c 文件存放应用代码。这里我们主要介绍wwdg.c、wwdg.h和main.c文件。

打开iwdg.c文件，代码如下：

#include "wwdg.h"

#include "led.h"

#include "delay.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数全称:

\* void WWDG\_Init(void)

\*

\* 辅助解释：

\* 超时时间计算:Tout=((4096\*2^WDGTB)\*（T[5:0]+1）)/Fwwdg

\* ((4096\*8)\*（127 - 95))/36000 = 29.13ms 非窗口期，喂狗产生复位，127是0x7F

\* ((4096\*8)\*（127 - 63))/36000 = 58.25ms，初始化之后经过58.25ms到达窗口期下限值，63是0x3F

\* 所以可以得到在20.13ms到58.25ms之间喂狗不产生复位

\*

\* 函数作用:

\* 初始化窗口看门狗

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WWDG\_Init(void)

{

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_WWDG,ENABLE);//使能APB1时钟

WWDG\_SetPrescaler(WWDG\_Prescaler\_8);//预分频系数设为8

WWDG\_SetWindowValue(0x5F);//上窗口值为0x5F，95是0x5F

WWDG\_Enable(0x7F);//使能WWDG

NVIC\_InitStructure.NVIC\_IRQChannel = WWDG\_IRQn;//WWDG中断

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 2; //设置抢占优先级

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 3; // 设置响应优先级

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; // 使能中断通道

NVIC\_Init(&NVIC\_InitStructure); // 调用NVIC初始化函数

WWDG\_ClearFlag();//清除提前唤醒中断标志位

WWDG\_EnableIT();//开启中断

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数全称:

\* void WWDG\_IRQHandler(void)

\*

\* 辅助解释：

\* 当计数值为0x40时可产生中断

\*

\* 函数作用:

\* 中断服务函数

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WWDG\_IRQHandler(void)

{

//delay\_ms(1);//由于设置的WDGTB = 3,最小超时值为910us，所以延时1ms后，喂狗会失败，LED2会不亮

// WWDG\_SetCounter(0x7F); //设置计数值

// WWDG\_ClearFlag();//清除提前唤醒中断标志位

// LED2\_Turn(); //LED2翻转，喂狗成功可以看见LED2闪烁

}

下面我们看看头文wwdg.h的代码，代码中包含一些头文件定义，数据类型、函数声明，代码如下：

#ifndef \_WWDG\_H

#define \_WWDG\_H

#include "stdlib.h"

#include "stm32f10x.h"

/\* 初始化 \*/

void WWDG\_Init(void);

#endif

最后，我们在main函数里编写应用代码，main.c文件如下：

#include "stm32f10x.h"

#include <string.h>

#include <stdlib.h>

#include "stdio.h"

#include "delay.h"

#include "led.h"

#include "wwdg.h"

int main(void)

{

LED\_Init(); //初始化LED

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//设置中断优先级分组为组2

delay\_ms(500);

WWDG\_Init(); //初始化窗口看门狗

GPIO\_SetBits(GPIOA, GPIO\_Pin\_6); //LED1灯亮

while(1)

{ //若LED1先灭500ms后一直亮，表示喂狗成功

//若LED1闪烁表示喂狗失败

delay\_ms(30);//理想下延时在29.13ms~58.25ms之间（窗口期）

//但是执行设置计数值还需要时间，所以在30ms~56ms内能够喂狗成功

WWDG\_SetCounter(0x7F);//设置计数值

}

}

至此，实验2 WWDG的软件设计部分就完成了。

13.4 下载验证

在代码编译成功下载程序完成后，在实验1 IWDG中按KEY1键来喂狗，每隔1s至少喂狗一次，此时LED1常亮，打开串口，可看见Yes，喂狗成功；如果没有按KEY1键，那么LED1闪烁，因为初始状态LED1是灭的，打开串口，可看见No，未喂狗。在实验2 WWDG中可以看见LED1先灭500ms后一直亮，表示喂狗成功；若在中断服务函数中喂狗（需要注释掉main函数中的喂狗函数），LED2闪烁则喂狗成功。