带 32 位 MCU 和高精度 ADC 的 SoC 产品

----SD93Fxxx 开发指南（一）

一、前言

您在使用新款芯片和新的开发环境进行开发的过程中是否会遇到以下问题：如何快速建工程？如何快速上手编程？不了解芯片功能？实现功能和预期不符？

本开发指南将手把手帮工程师们解决以上问题，了解芯片的各个模块，对照手册熟悉各个模块的寄存器配置。同时，分享一些常见的错误用法，避免您走一些弯路，从而提高开发效率。

二、程序描述

示例程序是以 SD93F115-JBS 芯片编写，包含了按键中断、RTC 中断、SDADC、SARADC、RTC、DAC、低压检测、LCD 显示、休眠测试等多个基础功能综合的一个程序，配合按键中断进入不同的子程序进行演示。

附件 SD93F115-JBS_TEST 是已经编写好的示例程序，其主体思路：先对芯片进行系统初始化，再对需要使用的功能进行配置，然后进入主循环通过按键中断选择置起相应的测试标志位，进入相应的测试项实现功能演示。

最后按照《软件篇》的方法将程序下载至芯片中。

三、main 函数

main 函数的整体框架，如图 3-1：

```c
#include "define.h"

int  main(void)
{
    Sys_MyInit();//系统初始化
    FLAG_Init();//标志位初始化
    IWDG_MyInit();//独立看门狗初始化
    KEY_TestInit();//按键中断初始化
    PWM1_MyInit();//PWM初始化
    SDADC_MyInit();//SDADC初始化
    SARADC_MyInit();//SARADC初始化
    UART1_MyInit(115200);//波特率115200
    GPIO_MyInit();//P83指示灯初始化
    RTC_MyInit();//RTC初始化
    LCD_MyInit();//LCD初始化


    LCD_DisplayData(1,0x01);//第1个数据显示‘1’
    LCD_DisplayData(2,0x02);//第2个数据显示‘2’
    LCD_DisplayData(3,0x03);//第3个数据显示‘3’
    LCD_DisplayData(4,0x04);//第4个数据显示‘4’
    LCD_DisplayData(5,0x05);//第5个数据显示‘5’
    LCD_DisplayData(6,0x06);//第6个数据显示‘6’


    while(1)
    {
        IWDG_ReloadCounter();//喂狗

        if(PWM1_FLAG)
            PWM1_TEST();//PWM1测试

        if(DAC_FLAG)
            DAC_TEST();//DAC测试

        if(LBT_FLAG)
            LBT_TEST();//低压检测

        if(SDADC_FLAG)
            SDADC_TEST();//SDADC测试

        if(SARADC_FLAG)
            SARADC_TEST();//SARADC测试

        if(STOP_FLAG)
            STOP_TEST();//休眠测试

        LED_1;
        delay_ms(50);
        LED_0;
        delay_ms(50);
    }
```

图 3-1 main 函数

如上图所示，main 函数首先运行的是 Sys_MyInit()系统初始化，因为芯片上电后的默认状态下基本只有 IHRC 和 ILRC 在工作，大部分的外设都需要打开对应时钟后，才可以重新配置相应模块，后期自行开发时有需要用到的外设可参照应用手册 13.7 节--时钟与复位寄存器优先开启时钟。

由于 IHRC 是校准过的，只需从 RCC_IHRC_CLBR 寄存器将校准数据存入 RCC_IHRC_CR 寄存器即可完成校准动作，模拟部分电源 BG、ACM 和 AVDDR 默认上电是关闭的，后续也会用到，直接调用库函数打开，完成对芯片的初步配置。如下图 3-2 所示：

```c
#include "define.h"

void Sys_MyInit(void)
{
    RCC_LoadIHRCALValue();  //载入IHRC的校准值
    RCC_LoadILRCALValue(); //写入ILRC的经验值

    PWR_BGCCCmd(ENABLE);//使能BG
    PWR_ACMCmd(ENABLE);//使能ACM
    PWR_AVDDRCmd(ENABLE);//使能AVDDR
}
```

图 3-2 系统初始化函数

系统初始化之后，进行 FLAG_Init()标志位初始化，该函数里的标志位是自定义的一些变量，在后续程序执行时会使用到，而在初始运行函数时，这些变量可能是随机数据，需要进行变量清零。如下图 3-3 所示：

```c
void FLAG_Init(void)//自定义标志位初始化
{
    SDADC_FLAG =0;   //SDADC测试标志位
    SARADC_FLAG =0; //SARADC测试标志位
    RTC_FLAG =0;     //RTC测试标志位
    DAC_FLAG=0;      //DAC测试标志位
    LBT_FLAG =0;     //低压检测标志位
    PWM1_FLAG =0;    //PWM1测试标志位
    STOP_FLAG =0;    //休眠模式测试标志位

    DAC_count =0;//DAC计数
    PWM1_count =0;//PWM1计数
}
```

图 3-3 自定义标志位初始化函数

自定义标志位初始化之后，为保证程序运行正常，在编写其他函数前，需要先编写一个 IWDG_Init()独立看门狗初始化函数（脱离主函数单独计数的一个寄存器，属于硬件范畴），它能在程序跑飞的情况下，复位芯片，使程序重新开始运行。独立看门狗结构图如下图 3-4：

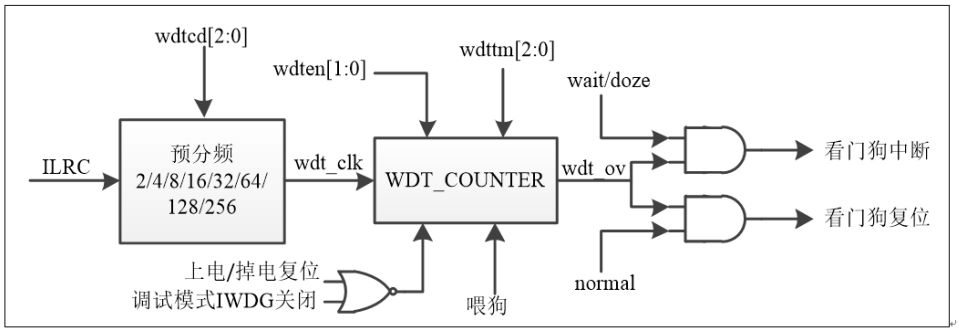

图 3-4 独立看门狗结构图

从结构上看，独立看门狗是以低频时钟 ILRC 的分频为时钟源，当 WDT_COUNTER 计数时间超过设置值时溢出，芯片便会产生中断或复位，注意一旦独立看门狗使能后软件就无法关闭 IWDG 和 ILRC，除非产生复位。

独立看门狗初始化函数如图 3-5 所示：

```c
#include "define.h"

void IWDG_MyInit(void)
{
    //独立看门狗配置
    IWDG_SetPrescaler(IWDG_Prescaler_8);//设置独立看门狗预分频器
    IWDG_SetReload(IWDG_Reload_23040);//设置独立看门狗重载值
    IWDG_Cmd(IWDG_Mode_Normal);//使能独立看门狗，只在正常模式下工作
    //独立看门狗复位时间计算：t=重载值*1/（ILRC/预分频值）    在这里=23040*1/（32k/8）=5.76 秒

}
```

图 3-5 独立看门狗初始化函数

这里直接调用库函数，将独立看门狗预分频器设置 8 分频，溢出时间设置为 23040，最后开启看门狗选择正常模式工作，待机模式不工作，在此配置下计算的复位时间 t=5.76 秒。

因为开启了独立看门狗，但又不希望程序在正常运行时发生复位，所以需要在正常运行的程序期间加入‘喂狗’操作，防止芯片在正常情况下复位，尤其是在执行循环过程中是需要特别注意进行‘喂狗’操作的，建议溢出时间不要设的太小。

独立看门狗初始化后，还对按键中断、PWM、SDADC、SARADC、UART、RTC 和 LCD 显示等功能也进行了初始化，在后面的章节会详细介绍。

为了更直观的看到程序是否正常运行，先在 LCD 屏上显示‘123456’，然后用一个 GPIO 做数字输出来驱动外部 LED 灯进行闪烁，GPIO 初始化如下图 3-6：

```c
#include "define.h"

void GPIO_MyInit(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;

    RCC_APBPeriphClockCmd(RCC_APBPeriph_GPIO,ENABLE);

    //P83初始化
    GPIO_InitStruct.GPIO_Mode=GPIO_Mode_OUT|GPIO_Mode_DI;//数字输出
    GPIO_InitStruct.GPIO_Pin =GPIO_Pin_3;
    GPIO_InitStruct.GPIO_PTSR =GPIO_PTSR_STRONG;
    GPIO_InitStruct.GPIO_PuPd =GPIO_PuPd_UP;
    GPIO_Init(GPIO8,&GPIO_InitStruct);
}
```

图 3-6 GPIO 初始化函数

首先打开 GPIO 外设时钟使能，将 P83 口设置为数字输出，配置为大电流模式（最大电流可以达到 12mA），为避免其他 GPIO 引脚悬空造成未知影响，将所有没有使用的 GPIO 全部配置为输入上拉模式。

宏定义了 P83 引脚的输出，使得控制 LED 灯的状态更清楚，然后在 main 函数的 while 循环里每隔 50ms 把 P83 状态翻转，即可控制 LED 灯的亮灭，最终程序运行 PCB 板的初步现象就是 LCD 显示‘123456’，LED 灯不停闪烁，等待按键中断的来临。

main.c Sys.c FLAG.c IWDG.c GPIO.c define.h

```
43  #include "STOP.h"
44
45  /**********定义标志位******************/
46  uint8_t Frequency_count;//频率溢出计数
47  uint8_t DAC_count;//DAC自增计数
48  uint32_t PWM1_count;//PWM1自增计数
49
50  uint8_t SDADC_FLAG;//测试SDADC标志
51  uint8_t SARADC_FLAG;//测试SARADC标志
52  uint8_t RTC_FLAG;//测试RTC标志
53  uint8_t DAC_FLAG;//测试DAC标志
54  uint8_t LBT_FLAG;//低压检测标志
55  uint8_t PWM1_FLAG;//测试PWM1标志
56  uint8_t STOP_FLAG;//测试休眠标志
57
58  /**********定义IO******************/
59  #define     LED_1       GPIO8->ODR |=(1<<3)      //P83输出1
60  #define     LED_0       GPIO8->ODR &=~(1<<3)     //P83输出0
61
62  #endif
```

图 3-7 P83 引脚输出宏定义

四、总结：

本文介绍了 Main 函数的整体运行流程，重点在于编程时要优先开启需要使用的外设时钟，再配合库函数的使用可以有效减少开发过程中遇到的问题，芯片的其他功能将会在后续章节逐一分享。

附件：

示例程序：

SD93F115-JBS_TEST.zip