如何在 KF32L/LS 系列中在低功耗 STOP1 模式下保持 LCD 显示及 RTC 唤醒刷新

前言

KF32L/LS 提供多种模式供用户在不同工作场景下使用。包含两种运行模式、两种休眠模式及三种低功耗模式。

本应用笔记将以 KF32L530 为例介绍如果低功耗模式下的运行及唤醒流程。以及如何在 STOP1 模式下使用 RTC 及 LCD。

本应用笔记使用的 KF32 IDE 与 KF32Lxxx 外设固件库及代码例程可以从 ChipON 官方网站 www.chipon-ic.com 下载。

目录

1 KF32L/LS 系列的主要特性

KF32L/LS 系列提供多种工作模式，内置两个电压调节器：主电压调制器 MR 和低功耗电压调制器 LPR，在不同模式下，可根据不同的需求开启或关闭调节器。内置灵活的电压结构，可以灵活的配置关闭不需要的外设电压以达到省电的目的。

微控制器的功耗主要为动态功耗和休眠静态功耗，KF32L/LS 系列提供的低至 60uA/MHz 的运行动态功耗及低至 0.2uA 的 shutdown 模式。

KF32L/LS 系列提供一个带有侵入检测功能的备份区，此备份区可以用于保存数据。寄存器组在 VDD 电源被切断时，仍然可以通过 VBAT 维持供电。备份域内寄存器只会在初始上电复位时被复位，不会因为 VDD 掉电上电而复位。

2 工作模式与电压域的关系

KF32L/LS 内置两个电压调节器，主电压调制器 MR 和低功耗电压调制器 LPR，提供的电流大小也不一致，根据不同功耗模式规格将 1.2V 域分为如下 5 类：Vcore、Vflash、Vsram、Vlpsram 及 Vpheri。外设域包含的具体内容请参考用户手册中 VREG 电源域分类章节。

Pow-Mode/Pow- domain	Regulator		Vcore	Vflash		Vsram		Vpheri	VDD33	VBKP
	MR	LPR	CPU &PHERI_A &SRAM_A &IO_A	FLASH_1.2V	FLASH_1.8V	DP-SRAM_A	LP-SRAM	LP-PHERI &GPIOA	Main_Power	Backup_Power
Run	ON	OFF	ON	ON	ON	ON	ON	ON	ON	ON
Sleep	ON	OFF	ON	ON	ON	ON	ON	ON	ON	ON
LPRUN	OFF	ON	ON	ON	ON	ON	ON	ON	ON	ON
LPSleep	OFF	ON	ON	ON	ON	ON	ON	ON	ON	ON
Run_SRAM	OFF/ON	OFF/ON	ON	OFF	OFF	ON	ON	ON	ON	ON
Stop0	OFF	ON	ON	OFF	OFF	ON	ON	OFF/ON	ON	ON
Stop1	OFF	ON	OFF	OFF	OFF	ON	ON	OFF/ON	ON	ON
Standby0 (with LP-SRAM)	OFF	ON	OFF	OFF	OFF	ON	ON	OFF	ON	ON
Standby1 (without LP-SRAM)	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON
Shutdown	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

3 RTC 运行在 STOP1 模式下

RTC 为了降低功耗，使用了内部低速时钟源、外部低速时钟源及外部低速时钟源的 128 分频中的一种作为 RTC 的时钟源。RTC 运行在 STOP1 模式下需要开启节拍中断，使用节拍中断在 STOP1 低功耗模式下唤醒。

KF32Lxxx 外设固件库中的 kf32l_basic_rtc.c 中定义了 RTC 初始化的配置函数。在例程代码 KF32L530_LP_LCD_RTC_STOP1 中，RTC_Init()为 RTC 的初始化代码，此初始化代码中包含两部分。一部分为 RTC 的基础配置，另一部分为节拍中断的配置。

RTC 配置前需要先使能是关闭复位寄存器，使能 RTC 时钟，例程代码如下：

```
RTC_Reset_Config();
```

RTC 的基础配置为配置‘RTC_InitTypeDef’结构体，配置时钟源及初始时间，例程代码如下：

```
RTC_InitTypeDef RTC_InitStructure;
/* 设置 RTC 时钟源 */
RTC_InitStructure.m_ClockSource = BKP_RTC_INTLF;      //内部低频 32KHz
/* 设置 RTC 时间格式 */
RTC_InitStructure.m_HourFormat = RTC_HOUR_FORMAT_24;  //24 小时制
/* 设置时钟 */
RTC_InitStructure.m_TimeStruct.m_Hours = 1;
/* 设置分钟 */
RTC_InitStructure.m_TimeStruct.m_Minutes = 59;
/* 设置秒钟 */
RTC_InitStructure.m_TimeStruct.m_Seconds = 55;
/* 设置周 */
RTC_InitStructure.m_DateStruct.m_WeekDay = RTC_WEEKDAY_SUNDAY_DEC;
/* 设置年 */
RTC_InitStructure.m_DateStruct.m_Year = 20;
/* 设置月 */
RTC_InitStructure.m_DateStruct.m_Month = RTC_MONTH_JANUARY_BCD;
/* 设置日 */
RTC_InitStructure.m_DateStruct.m_Day = 31;
RTC_Configuration(RTC_TIME_FORMAT_BIN,&RTC_InitStructure);//不使用 BCD 编码，日期时间初始化配置，该函数结束时会退出配置模式。
```

RTC 节拍中断的配置及 RTC 使能，RTC 的节拍中断属于外部中断，挂在外部中断向量中的 EINT19TO17 中，例程配置代码如下：

```
RTC_Config_Mode_Enable (TRUE);//进入 RTC 配置模式
//********外部中断(EINT)配置信息结构体,开放外部 int17-19 中断
EINT_InitTypeDef EINT_17to19;
RTC_Clock_Calibration_Config((int8_t)0x0);//配置 RTC 时钟校正值。
```

```
/*******外部中断(EINT)配置信息结构体*/
EINT_17to19.m_Mask=TRUE; //对应外部中断使能
EINT_17to19.m_Rise=TRUE; //上升沿触发
EINT_17to19.m_Line=INT_EXTERNAL_INTERRUPT_17; //外部中断线选择
EINT_17to19.m_Fall=FALSE;  //下降沿不触发
EINT_17to19.m_Source=INT_EXTERNAL_SOURCE_RTC;//中断源
INT_External_Configuration (&EINT_17to19);  //写入中断源信息
RTC_Clock_Calibration_Config((int8_t)0x0);//配置 RTC 时钟校正值。如果不使用校准功
能，则必须写入 0 值。
RTC_Time_Tick_Config(RTC_TIME_TICK_DIV_2);//配置时间节拍 1/2 秒
RTC_Clear_Time_Tick_INT_Flag();
RTC_Enable(TRUE);//设置 RTC 使能
RTC_Config_Mode_Enable (FALSE);//退出配置模式
INT_Interrupt_Enable(INT_EINT19TO17,TRUE);//开放 EIE 的 RTS 中断
INT_All_Enable (TRUE);//开放系统总中断
```

注：上述配置将 RTC 的节拍中断定为 500ms,即每 500ms 产生一次中断,因为没有开启 RTC 的节拍中断,当产生中断时，不会进入中断服务函数，若需要进入中断服务函数，需要在清除节拍中断标志位后将节拍中断标志开启。（进入中断服务函数需要时间,此时间会增加运行功耗）

4 LCD 运行在 STOP1 模式下

KF32Lxxx 外设固件库中的 kf32l_basic_lcd.c 中定义了 RTC 初始化的配置函数。在例程代码 KF32L530_LP_LCD_RTC_STOP1 中，LCDdisplay_Init()为 LCD 的初始化代码。

LCD 配置前需要先使能再关闭复位寄存器，使能 LCD 时钟，例程代码如下：

```
LCD_Reset();
```

LCD 的基础配置为配置‘LCD_InitTypeDef’结构体，配置时钟源、时钟分频及使用的端口引脚，例程代码如下：

```
LCD_InitStructure.m_SegPin =
LCD_SEG_PIN_7|LCD_SEG_PIN_8|LCD_SEG_PIN_9|LCD_SEG_PIN_32|LCD_SEG_PIN_33 \
|LCD_SEG_PIN_37|LCD_SEG_PIN_40|LCD_SEG_PIN_43;
LCD_InitStructure.m_SegPinEn = LCD_SEG_PIN_SEG;
LCD_InitStructure.m_CommonPort = LCD_COMMON_PORT_DIV_4;
LCD_InitStructure.m_Analog = TRUE;
LCD_InitStructure.m_ClockSource = LCD_SOURCE_HALF_INTLF;
LCD_InitStructure.m_SourcePrescaler = LCD_SOURCE_DIVIDE_2;
LCD_InitStructure.m_LCDPrescaler = LCD_PRESCALER_2;
LCD_InitStructure.m_VoltageSelect = LCD_VOLTAGE_INTERNAL;
LCD_Configuration(&LCD_InitStructure);
LCD_IO_Enable(TRUE);
/*STOP 下允许 LCD 低功耗配置*/
```

```
PM_CTL2 |= PM_CTL2_LCDCLKLPEN|PM_CTL2_LCDLPEN;
LCD_Cmd_Enable(TRUE);
```

注：配置低功耗下 LCD 保持运行，需要在 LCD 使能前将 PM_CTL2 寄存器的 bit2 和 bit6 置 1。

5 低功耗模式下 IO 的配置

例程代码 KF32L530_LP_LCD_RTC_STOP1 中，低功耗 IO 的例程配置代码如下：

```
/* 低功耗 IO 配置 */
GPIOA_PMOD = 0xFFFFFFFF;
GPIOB_PMOD = 0xFFFFFFFF;
GPIOC_PMOD = 0xFFFFD7FF;
GPIOD_PMOD = 0xFFFFFFFF;
GPIOE_PMOD = 0xFFFFFFFF;
GPIOF_PMOD = 0xFFFFFFFF;
GPIOG_PMOD = 0xFFFFFFFF;
GPIOH_PMOD = 0xFFFFFFFF;
GPIOB_PUR =  0x00000000;
```

LCD 的引脚为模拟引脚，进入休眠前应将 GPIOx_PMOD 端口方向控制寄存器对应引脚配置为模拟模式。此处使用寄存器直接操作，提高代码效率，减少运行时间。

注：1.PC5、PC6 为编程口，需要将这两个引脚配置为数字输出模式。这两个引脚在配置低功耗模式时，如果外界上拉或者外部信号为高电平，会增加上电时判断是否为编程信号的时间，会增加功耗。

2.PB3 为上电启动的引脚选择，默认为上拉配置，需要在进入低功耗需要将此引脚上拉功能关闭。

3.开发板上的 EXCON1 接口，为了满足多种外设，在 LCD 信号线上串联了 560R 电阻，会导致 LCD 存在“鬼影”，需要将电阻改为小电阻或短接直连。

6 KF32 系列微控制器上电过程解析

上电的启动顺序：

（其中代码包含 RAM 初始化+用户代码。）

正常的低功耗启动顺序：

优化后的低功耗启动顺序：

（其中代码为用户代码，不包含 RAM 再初始化）

优化后低功耗启动与正常模式下有以下三个区别：

1.进入低功耗模式前将启动模式改通过 SYS_MEMCTL 寄存器的 MEMM<1:0>修改为从 Flash 启动，将减少 4.68ms 左右的启动时间；

2.P12 及 P18 上电电流脉冲较大，如果频繁进入低功耗再唤醒，会导致 P12、P18 重复上电，功耗会增加很多。可以选择保持 P12，通过保持 DRAM 不掉电 P12 将不会被关闭。若低功耗下唤醒的时间较短，可保持 P12 不掉电，即保持 DRAM 不关闭，若低功耗下唤醒的时间较长，可以关闭 DRAM 节省此部分的功耗。

3.RAM 在第一次上电的时候进行初始化，再次上电的时候可以不进行 RAM 初始化。RAM 初始化会将全局变量全部初始化。全局变量越多，初始化的时间就越长。初始化 RAM 的代码在工程的 config 文件夹的 startup.c 中，函数名为：**void startup()**；启动流程的修改在 config 文件夹的 vector.c 中，将第七行的 startup 修改为 main，将在上电的时候从 main 函数启动。

7 备份区的操作

此应用笔记中使用的 RTC、LCD、电源 PM 操作，都属于备份区的操作。

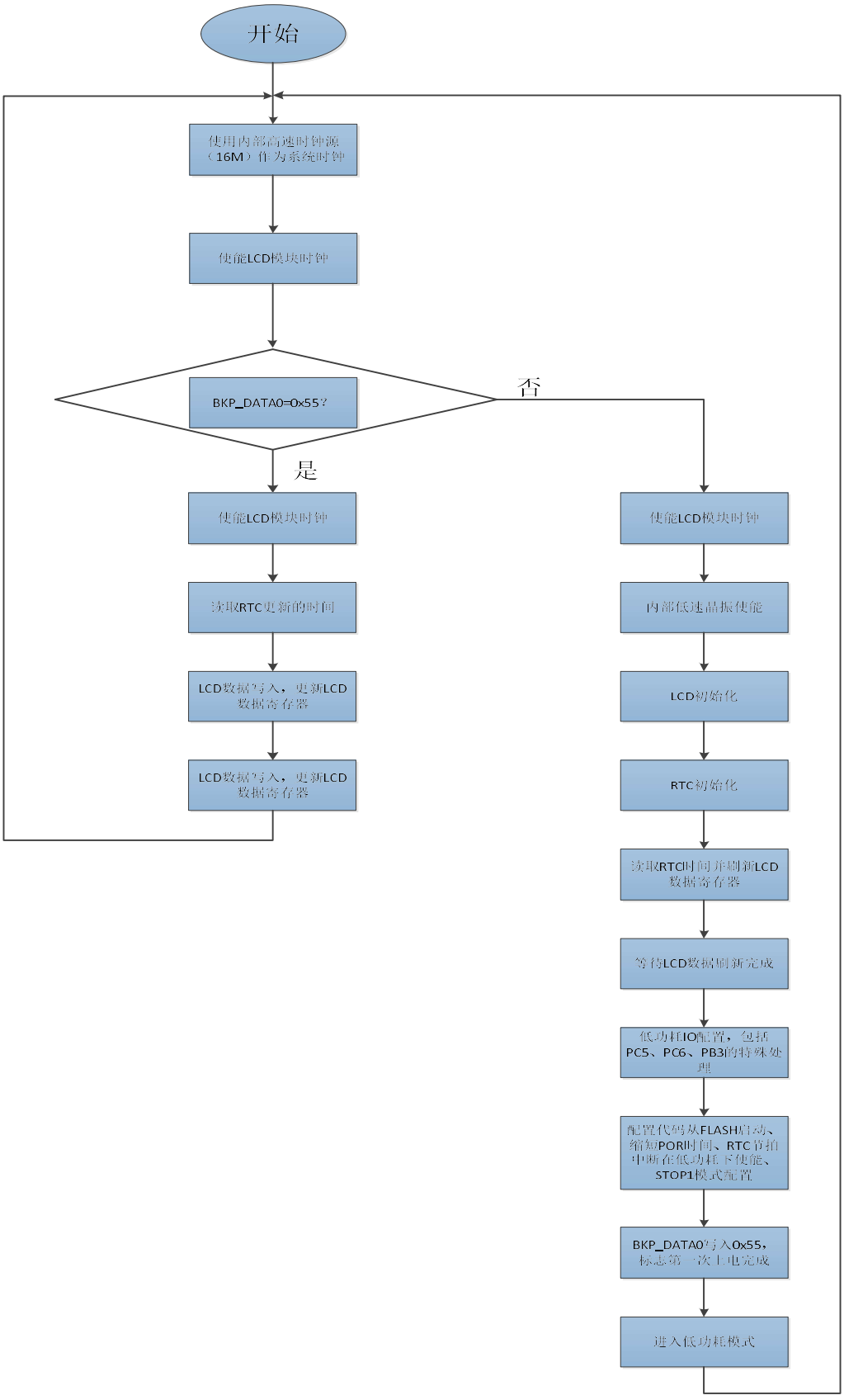
备份区有 32 个 32 位的数据寄存器，例程代码中使用了 BKP_DATA0 数据寄存器，首次上电进入低功耗前将写入 0x55，读出非 0x55 将代表非首次上电。

对备份区的操作请查看数据手册中的备份区章节中的寄存器组的读写小节。

8 软件流程图

例程代码 KF32L530_LP_LCD_RTC_STOP1 的软件流程图如下。

本应用笔记使用的 KF32 IDE 与 KF32Lxxx 外设固件库及代码例程可以从 ChipON 官方网站 www.chipon-ic.com 下载。


开始
使用内部高速时钟源（16M）作为系统时钟
使能LCD模块时钟
BKP_DATA0=0x55?
否
是
使能LCD模块时钟
读取RTC更新的时间
LCD数据写入，更新LCD数据寄存器
LCD数据写入，更新LCD数据寄存器
使能LCD模块时钟
内部低速晶振使能
LCD初始化
RTC初始化
读取RTC时间并刷新LCD数据寄存器
等待LCD数据刷新完成
低功耗IO配置，包括PC5、PC6、PB3的特殊处理
配置代码从FLASH启动、缩短POR时间、RTC节拍中断在低功耗下使能、STOP1模式配置
BKP_DATA0写入0x55，标志第一次上电完成
进入低功耗模式

9 版本历史

文档版本历史

日期	版本	变更
2020 年 1 月 10 日	1	初始版本。