

HK32L08x系列应用笔记

版本：V1.0.14

发布时间：2024-08-29

深圳市航顺芯片技术研发有限公司

<http://www.hsxp-hk.com>

前言

编写目的

本文档介绍了航顺MCU芯片应用开发的常见问题和注意事项，旨在帮助用户加快产品开发进程。

读者对象

本文适用于以下读者：

* 开发工程师
* 芯片测试工程师

修订记录

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 修订内容 |
| V1.0.0 | 2023-07-13 | 首次发布 |
| V1.0.1 | 2023-09-12 | 更新Lowpowerrun模式在HSI/PLL打开时进入失败 |
| V1.0.2 | 2023-10-01 | 新增使能systick后在Stop模式唤醒失败 |
| V1.0.3 | 2023-11-21 | 新增USB外设当SETUP事件产生时未自动将端点STAT\_TX置NAK导致枚举失败 |
| V1.0.4 | 2023-12-05 | 新增UART时钟使能后再配置串口GPIO有概率导致接收异常 |
| V1.0.5 | 2024-01-02 | 新增HK32L0H8RBT6在机台上电后，PA11输出异常 |
| V1.0.6 | 2024-01-02 | 新增VDD在非0V电压时，缓上电后工作异常 |
| V1.0.7 | 2024-01-19 | 新增TIM1电压比较输出重定向选择比较器时使用刹车功能不能将刹车输入极性配置为高电平有效 |
| V1.0.8 | 2024-01-25 | 新增USB端点被置为STALL状态之后的表现与USB标准协议不一致 新增无法使用USB DFU功能 新增USB在windows10操作系统下的HID模式传输大量数据时IN包无法应答 新增USB在PLLCLK=48M时进入Stop唤醒后无法正确枚举 新增USB的FRES位功能失效 新增USB的RESUME位功能失效 新增USB的PDWN位功能失效 新增USB的LP\_MODE位与S\*\*T功能不同 新增USB的ESOF中断与S\*\*T表现不一致 新增USB外设的缓冲区描述表与S\*\*T不同 |
| V1.0.9 | 2024-01-31 | 新增WKUP引脚极性配置和使能位配置后WUF都会置位导致不能进入Standby模式 |
| V1.0.10 | 2024-03-04 | 新增DVSQ模块连续做除法和开方除法计算错误 |
| V1.0.11 | 2024-05-13 | 新增I2C从字节控制模式下使能reload模式后，PECBYTE位仍然起作用 |
| V1.0.12 | 2024-07-24 | 新增DIV[3:0]配置成0时，LCD帧率与S\*\*T不一致 |
| V1.0.13 | 2024-08-21 | 新增在低温时standby唤醒功能异常 |
| V1.0.14 | 2024-08-29 | 新增ADC外部触发（EXTI线11）与S\*\*T不一致,新增DAC外部触发（EXTI线9）与S\*\*T不一致 |

目录

[1 简介 1](#_Toc136246985)

[2 应用 2](#_Toc136246986)

[3 缩略语 2](#_Toc136246987)

[4 重要提示 4](#_Toc136246988)

# 简介

本文档为航顺MCU芯片的应用笔记，适用于以下产品型号：

* HK32L08x系列
  + - HK32L088CBT6
    - HK32L084FBP6
    - HK32L084GBU6
    - HK32L084KBU6
    - HK32L088CBU6
    - HK32L084KBT6
    - HK32L084RBT6
    - HK32L084CBT6
    - HK32L088FBP6
    - HK32L088GBU6
    - HK32L088KBU6
    - HK32L088KBT6
    - HK32L088RBT6

用户若需获取更多相关技术文档，请联系深圳市航顺芯片技术研发有限公司。

# 应用

本章介绍了航顺MCU应用开发的常见问题和注意事项。

## 嵌入式Flash(FLASH)

### FLASH\_INT\_VEC\_OFFSET[14:0]不能完全覆盖FLASH容量

* **问题描述：**

 客户在使用IAP的时候，发现boot程序设置跳转APP的地址大于0x08004000的时候，在APP端出现hardfault(PC指向0xFFFFFFFE)情况。

* **根本原因：**

如下图 2‑1所示，传统的IAP流程如下，BOOT的中断向量表默认在BOOT程序的最前面，APP的中断向量表默认在APP程序的最前面，为了让程序跳转到APP能够正常的执行中断服务程序，我们的芯片仿照M3的架构，设计了一个FLASH\_INT\_VEC\_OFFSET寄存器，目的是把中断向量表偏移到指定的相对位置，使APP能够在FLASH\_INT\_VEC\_OFFSET寄存器偏移的相对位置找到中断向量表，最终使APP程序能够正常的响应中断。

/image/ptm/userfiles/system/images/20230713/image_$1679415894210838528.png

图 2‑1 传统IAP示意图

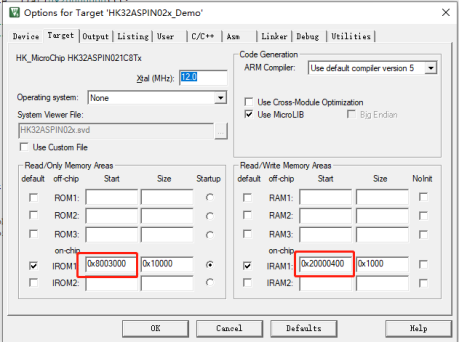
由于我们的芯片FLASH\_INT\_VEC\_OFFSET寄存器最大支持设置为[13:0]位，所以当设置偏移大于等于0x4000的时候，虽然BOOT能够正常的跳转到APP程序执行，但是只要存在中断服务程序的响应，APP这边的程序找不到中断向量表，就会报hardfault的错误，并且PC一定指向0xFFFFFFFE这个地址。

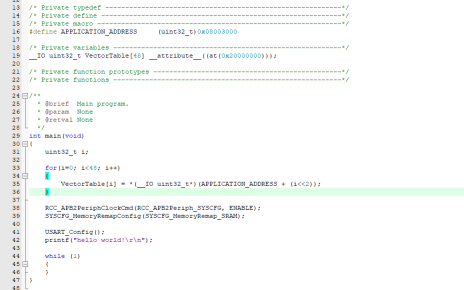
* **解决方案：**

•            方案一：

利用SYSCFG寄存器的MEM\_MODE[1:0]位，把APP的中断向量表映射到RAM的0地址区域，具体操作方法如下：

1. 首先明确BOOT跳转的APP地址
2. 定义首地址在0x20000000，大小为IVT的数组，用于存放APP的中断向量表地址
3. 修改RAM的其实地址，其实地址应该为0x20000000加上中断向量表的偏移
4. 把APP的中断向量表地址全部写入到上述定义的数组中
5. 利用SYSCFG寄存器的MEM\_MODE[1:0]位，把嵌入式RAM映射到0x00000000地址





•            方案二：

如下图 2‑2所示，把BOOT的IVT和APP的IVT放在FLASH的最前端，这样就不需要考虑FLASH\_INT\_VEC\_OFFSET寄存器偏移不够的问题。

/image/ptm/userfiles/system/images/20230713/image_$1679415894210838531.png

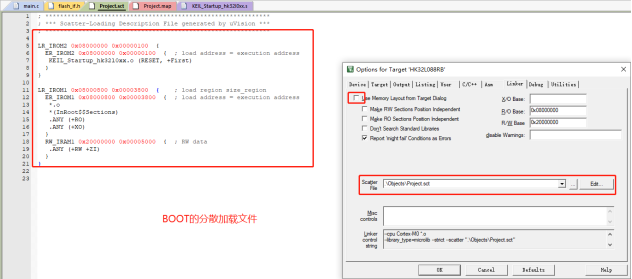
图 2‑2 改进IAP方案

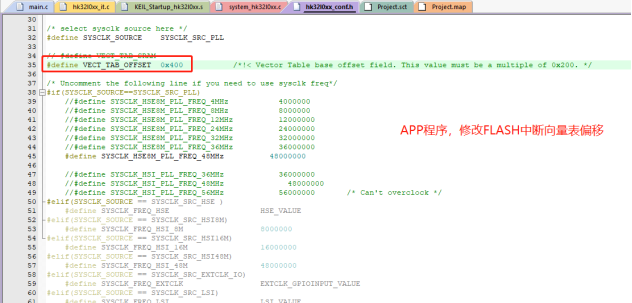
具体的实施如下：

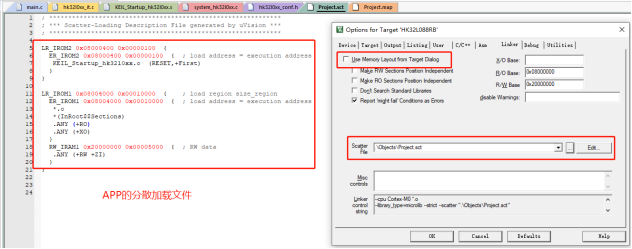
使用IDE工具为Keil的时候：

1. 明确BOOT的IVT和APP的IVT在固定的位置，比如BOOT的IVT固定在0x08000000起始地址的位置，APP的IVT固定在0x08000400起始地址的位置（以page分界）
2. 明确BOOT的跳转地址固定为APP的IVT的起始地址，这里BOOT的跳转地址设置为0x080000400
3. 修改BOOT的分散加载文件，必须把BOOT的IVT设置在0x08000000为起始地址的位置，BOOT的程序则需要设置到0x08000800的位置（因为0x08000400之后的一个page被APP的IVT使用）
4. 设置APP程序的FLASH\_INT\_VEC\_OFFSET，这里设置为0x400
5. 修改APP的分散加载文件，必须把APP的IVT设置在0x08000400为起始地址的位置，APP程序则需要设置到0x08004000的位置



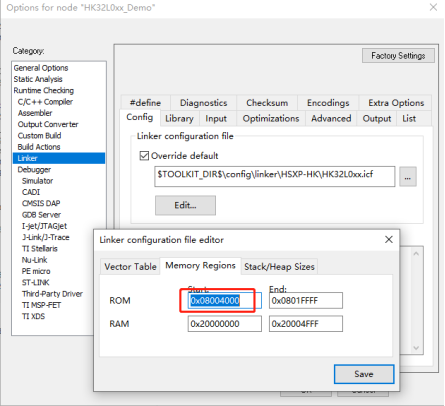
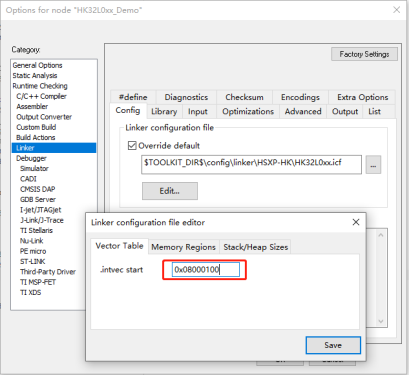






使用IDE工具为IAR的时候：

实现过程和Keil大致相同，但是IAR分散加载中断向量表不需要修改分散加载文件，只需要在可视化界面修改，具体如下（简单演示了一下APP的起始地址和IVT地址的修改过程）：



## 电源控制(PWR)

### Lowpowerrun模式在HSI/PLL打开时进入失败

* **问题描述：**

在使能HSI/PLL的情况下进入low power run模式，无法成功进入。

* **根本原因：**

设计原因

* **解决方案：**

在进入lower power run模式之前，软件关闭HSI和PLL。代码如下：

RCC\_HSICmd(DISABLE);

RCC\_PLLCmd(DISABLE);

### 使能systick后在Stop模式有概率唤醒失败

* **问题描述：**

在systick（或者任意非EXTI中断）使能后，有概率在进入lowpower\_stop模式后，EXTI无法正常唤醒。

* **根本原因：**

在进入低功耗过程中如果发生一次非EXTI中断，此时可能PMU已经进入低功耗，但是CPU被唤醒，因为CPU已经被唤醒不会再响应EXTI唤醒信号，导致PMU一直在低功耗中无法起来。（该问题只在normal\_stop和lowpower\_stop下存在）

* **解决方案：**

方案一：在进入 STOP 前 (即配置 WFI/WFE 前)，停掉相应的外设防止产生中断.

方案二：在进入 STOP 前 (即配置 WFI/WFE 前)， 在NVIC配置里面关闭除了要使用的 EXTI 中断线外的所有中断，直到唤醒后再打开. (推荐)



### 外部唤醒PIN使能后，WUF唤醒标志异常置位

* **问题描述：**

在外部唤醒PIN非唤醒电平时，配置WKUP引脚极性和使能位之后，唤醒标志位WUF会置位，若此时进入Standby模式，会立即唤醒。

* **根本原因：**

设计原因

* **解决方案：**

在配置完WKUP引脚极性和使能位之后，需要清除一次WUF标志，然后再进入Standby模式

### 低温时standby唤醒功能异常

* **问题描述：**

在低温时（低于0度），芯片进入standby后有可能无法唤醒，也无法复位，必须重新上电才能恢复正常。

* **根本原因：**

设计原因

* **解决方案：**

无

## 复位和时钟控制(RCC)

### HSE误使能后MCU跑飞

* **问题描述：**

客户在使用HSI时，误使能了HSE，但是外部没有接HSE电路和晶体，导致在外部有电机干扰时，出现程序跑死。

* **根本原因：**

HSE管脚接地或悬空，在使能HSE后，HSE Ready仍然会置位。

在配置sysclk为HSI-PLL之前，先enable了HSE，然后等待HSE ready，如果没有等到HSE ready会timeout然后配置sysclk为HSI-PLL。

因为板子上没有接外部晶体，同时在上电的时候外部有干扰发生，HSE会输入毛刺误触发HSE ready，这样在timeout之前就可能等到HSE ready，然后sysclk就被软件错误配置为了HSE-PLL，但实际上HSE没有输入，这样芯片就跑死了

* **解决方案：**

客户如果不使用HSE，建议将它disable。

### CSS在HSE异常时不能产生NMI中断

* **问题描述：**

使能HSE时，如果用户想要检测HSE是否正常工作，可以打开CSS的HSE时钟安全系统来帮助我们检测。

当CSS功能开启后，CSS模块电路会使用HSI时钟来测量HSE的波形，所以使用前需要保证HSION并HSIRDY。当电路没有外部晶体时，由于上电的干扰发生，会输入毛刺误触发HSE ready，此时默认的CSS检测阈值无法正确产生中断。如果此时直接或间接使用HSE振荡器作为系统时钟（间接是指它用作PLL输入时钟，PLL时钟用作系统时钟），那么系统会跑飞进入Hard Fault。

* **根本原因：**

CSS对HSE时钟波形异常检测能力不够。CSS功能只能检测HSE频率的有无，并不能精确的测试HSE的频率。

* **解决方案：**

用户可以在使用CSS的HSE时钟安全系统时，将RCC\_HSECTL中的CSS\_THRESHOLD[6:0]寄存器进行配置为最小为1，CSS模块判断的最低HSE频率大约为4M。

### RCC\_APB1ENR寄存器的LCDEN位不可读

* **问题描述：**

LCDEN位写1或者写0，控制功能正常，但是软件去读到的值始终为0。

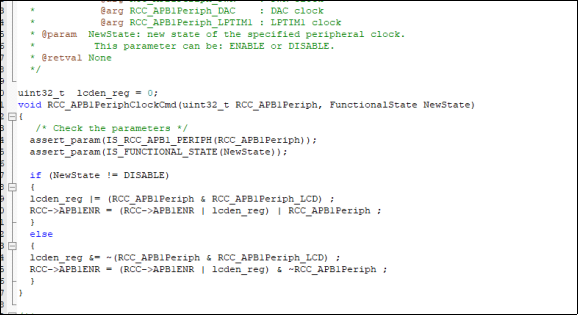
这样会导致，如果有客户需要对RCC\_APB1ENR寄存器的其他位进行写操作，需要软件读回该寄存器，修改后，重新写回。由于读取到的LCDEN一直为0，导致写回时，LCDEN位被置0，误关闭了LCD模块。

* **根本原因：**

设计原因

* **解决方案：**

设计一个全局变量来保存该寄存器的配置值。推荐采用库函数。



### VDD在非0V电压时，缓上电工作异常

* **问题描述：**

VDD掉电到1V再缓上电到3.3V后，MCU无法正常工作

* **根本原因：**

设计原因

* **解决方案：**

1. 在SystemInit开始处加延迟，等VDD上电稳定后再执行后面的函数
2. 在SystemInit开始处配置PWR\_CR.VOS[1:0]为1，提高Vcore到1.65V，等VDD上电稳定后再修改PWR\_CR.VOS[1:0]回0
3. 修改Bor到等级四及其以上

## 系统配置(SYSCFG)

### FlashOption的nRST\_STOP位在NormalStop模式下功能失效

* **问题描述：**

客户需要使用Normal Stop模式产生系统复位时，在Flash Option选项字中设置nRST\_STOP位后，进入Normal Stop模式，但是没有产生系统复位，也没有进入Normal Stop模式。

* **根本原因：**

设计原因

* **解决方案：**

无

如果客户需要低功耗管理产生系统复位时，建议用Low Power stop模式。可调用以下库函数进行配置：PWR\_EnterSTOPMode(PWR\_Regulator\_LowPower,PWR\_STOPEntry\_WFI);

### HK32L0xx在Nrst脚悬空时上电，PA9/PA10和PA11/PA12数字功能发生互换

* **问题描述：**

HK32L0xx在Nrst脚上未接对地电容和上拉电阻时，上电后，PA9/PA10和PA11/PA12数字功能发生互换。（比如本来PA11输出的PWM信号被PA9输出，PA11上测不到信号）。

* **根本原因：**

设计原因。

* **解决方案：**

1. 方案一：建议用户的nRST使用10KΩ的上拉电阻，并配置0.1uF的对地电容。
2. 方案二：库函数已经加上软件补丁，在上电后对相应寄存器进行软件重载。

## 除法开方运算单元(DVSQ)

### DVSQ模块连续做除法和开方除法计算错误

* **问题描述：**

在使用芯片硬件除法开方单元做连续的除法和开方运算时，除法计算错误。

* **根本原因：**

设计原因

* **解决方案：**

在需要做连续除法开方运算时，做完开方运算后，或者除法运算前  将DVSQRT时钟复位；

具体如下：

sqrt\_result = Dvsq\_Sqrt(100);

RCC->AHBRSTR |= RCC\_AHBRSTR\_DVSQRST;

RCC->AHBRSTR &= ~RCC\_AHBRSTR\_DVSQRST;

DSPresult = Dvsq\_Divsion(a,b,&c);

添加库函数方便调用：

void HK\_Dvsq\_Rst(void)

{

RCC->AHBRSTR |= RCC\_AHBRSTR\_DVSQRST;

        RCC->AHBRSTR &= ~RCC\_AHBRSTR\_DVSQRST;

}

## 模拟/数字转换(ADC)

### ADC外部触发（EXTI线11）与S\*\*T不一致

* **问题描述：**

该系列使用ADC外部触发（EXTI线11）时，除了ADC寄存器（详见ADC外部触发章节），还需要配置EXTI寄存器（必须把EXTI\_IMR或者EXTI\_EMR bit11使能，然后通过EXTI\_RTSR或者EXTI\_FTSR寄存器配置触发沿），而ST不需要配置EXTI相关寄存器。

* **根本原因：**

设计原因

* **解决方案：**

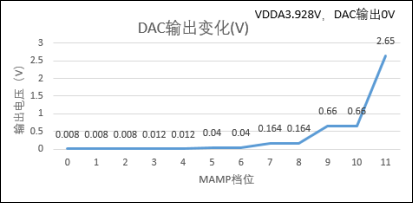
无

## 数模转换器(DAC)

### DAC输出在关闭波形生成情况下修改MAMP[3:0]为非零时输出有偏差

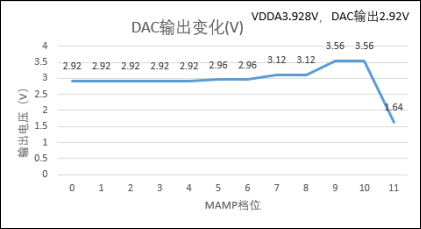
* **问题描述：**

初始化DAC功能时，若关闭波形生成，即不使能产生噪声或三角波（DAC\_CR\_WAVE1[1:0] == 2b’00），但是噪声模式下的屏蔽位或在三角波模式下的波形幅值没有设置为0（DAC\_CR\_MAMP != 4b’0000），则DAC实际输出的电压会在设定输出值（DAC\_DHR值）上有偏差，偏差电压会因DAC\_CR\_MAMP寄存器的值不同有差异。



当设定输出电压与偏差电压相加超过VDDA电压时，更高的设定输出值配置后，实际输出电压则变为设定输出电压与偏差电压相之和减VDDA电压的差值。

举例：MAMP第11档时：1.64V = 设定输出电压（2.92V）+ 偏差电压（2.65V）-3.928V = 1.642V.

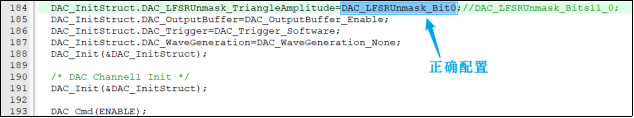


* **根本原因：**

设计原因

* **解决方案：**

配置DAC的输出不为三角波和噪声波时，DAC\_CR\_MAMP寄存器应配置为0值，代码如下：



### DAC外部触发（EXTI线9）与S\*\*T不一致

* **问题描述：**

该系列使用DAC外部触发（EXTI线9）时，除了DAC寄存器（详见DAC外部触发章节），还需要配置EXTI寄存器（必须把EXTI\_IMR或者EXTI\_EMR bit9使能，然后通过EXTI\_RTSR或者EXTI\_FTSR寄存器配置触发沿），而ST不需要配置EXTI相关寄存器。

* **根本原因：**

设计原因

* **解决方案：**

无

## 段码液晶显示控制器(SegmentLCD)

### LCDRAM在更新时锁死

* **问题描述：**

需要显示的数据写入LCD\_RAM后，将UDR位置1，请求更新，更新完成后UDD标识位1表示此次更新完成。可以进行下一次需要显示的数据写入。

测试发现在UDD位置1后，LCD\_RAM仍然处于写保护状态，导致后续数据写入无效，并且LCD\_RAM会一直处于写保护状态。

* **根本原因：**

由于双时钟域逻辑设计问题，LCD处于RTC域，寄存器处于系统域导致。

* **解决方案：**

第一次写入UDR位后，UDD置位，加入适量延时再写入UDR位。

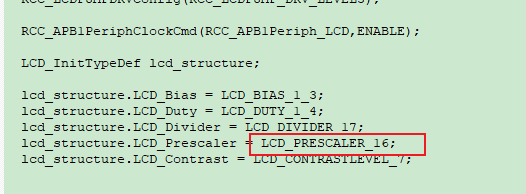
添加延时时间公式如下：

/image/ptm/userfiles/system/images/20230713/image_$1679415893321646080.png

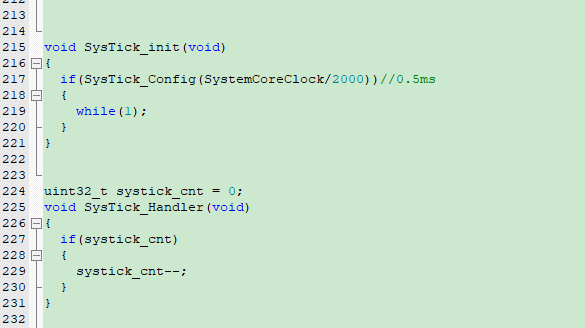
ck\_ps是LCDCLK经过预分频器分频后的时钟，ck\_ps = LCDCLK/2PS[3:0],详细请参考用户手册LCD章节LCD\_FCR寄存器PS[3:0]位。

代码示例如下：

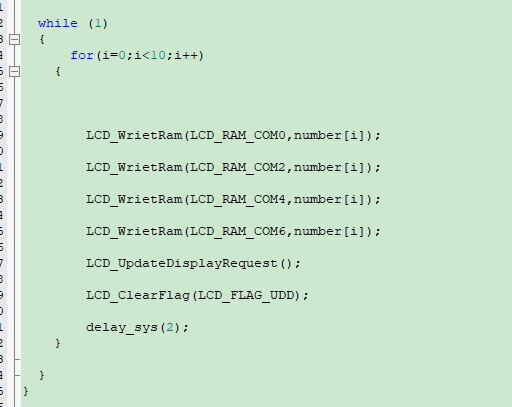
 LCD配置预分频为LCD\_PRESCALER\_16,LCDCLK = 32768Hz



此时，ck\_ps = 32768/24 = 2048Hz ，需要的延时delay\_time = 2/2048≈1ms，用systick将延时设置为0.5ms。



在每次更新显示完成后添加延时：



### DIV[3:0]配置成0时，LCD帧率与S\*\*T不一致

* **问题描述：**

DIV[3:0]配置成0时实际分频是31，而S\*\*T这样配置分频是16

* **根本原因：**

设计原因

* **解决方案：**

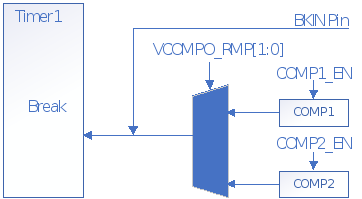
无

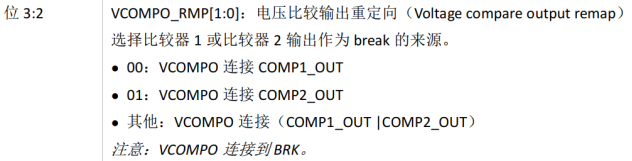
## 高级定时器(TIM1)

### TIM1的Break输入不能同时与两个比较器输出断开

* **问题描述：**

当TIM1使能Break功能，并且选择外部BKIN pin作为Break来源，用户准备将两个比较器作为其他目的使用，使能两个比较器后，发现比较器在工作时，会导致两个比较器的输出会误触发TIM1 Break。





* **根本原因：**

两个比较器的输出，VCOMPO\_RAMP[1:0]寄存器没有一种选择，将两个比较器的输出同时与TIM1 Break断开。

* **解决方案：**

无

注意：

如果使用外部Pin作为Break输入，需要使用两个比较器功能模块，无法解决；

如果使用两个比较器输出信号，作为Break输入时，没有问题；

如果使用外部信号作为Break输入时，需要使用一个比较器功能模块，配置TIM1\_OR 寄存器的VCOMPO\_RMP[1:0]控制Break连接到一个空闲的比较器；

### TIM1电压比较输出重定向选择比较器时使用刹车功能不能将刹车输入极性配置为高电平有效

* **问题描述：**

当配置比较器输出作为TIM1的刹车输入信号时，不能配置刹车输入极性为高，即将TIM1\_BDTR.BKP配置为1，否则会一直刹车

* **根本原因：**

设计原因

* **解决方案：**

当配置比较器输出作为TIM1的刹车输入信号时，必须将TIM1\_BDTR.BKP配置为0，若比较器输出为高，则刹车，若比较器输出为低，则不刹车。

## 通用定时器(TIM2/3)

### TIM3TI\_RAMP默认为0导致编码器计数器只增不减

* **问题描述：**

客户在使用TIM3作为编码器接口，由于TI\_RAMP寄存器位默认为0，TIM3\_TI1连接到USB\_SOF，导致当外部编码器信号没有正常输入到TIM3 TI1，从而编码器的计数器表现为只增不减。



* **根本原因：**

相对于HK32F030芯片，TIM3新增了TIM3\_OR寄存器，其中TI1\_RMP位，专门控制着TIM3\_CH1(PA6/PC6/PD2/PB4)与TIM3\_TI1的通断。默认是0，选择USB\_SOF，外部编码器信号没有正常输入到TIM3 TI1，导致编码器的计数器表现为只增不减。

* **解决方案：**

使用编码器功能，此位要置1。（此控制方式与S\*\*T\*\*M32L0xx芯片相同）

## 低功耗定时器(LPTIM1/2/3)

### LpTIM使能后低功耗模式不能工作

* **问题描述：**

当LPTIM使能后，Lprun/Lpstop/Normal Stop模式不能工作。

* **根本原因：**

设计原因

* **解决方案：**

无

如果客户要使用Lprun/Lpstop/Normal Stop模式，将LPTIM一直复位；

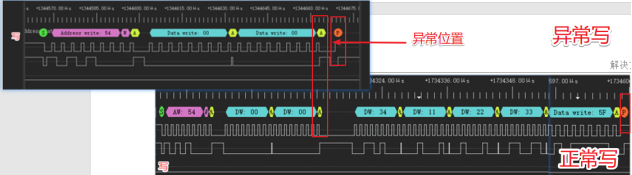
如果客户要使用LPTIM，就不能使用Lprun/Lpstop/Normal Stop模式；

## I2C接口(I2C)

### I2C的工作频率与系统时钟频率不一致导致异常

* **问题描述：**

系统时钟是48M, I2C为HSI8M时钟，此种组合会导致I2C向EEPROM写数据时，地址发送完成后，直接发送停止信号，而没有发送数据（如下图异常位置）。



* **根本原因：**

由于跨时钟域同步需要时间， 两个事件同步的时间有差别， 导致 NBYTES 配置还没有生效 (即TCR没有来得及被clear 掉)，但是 AUTOEND 又设置上了。  此时内部核心控制看到 AUTO\_END =1, 且 TC =1 , 就认为传输结束（此时还没有发送数据）。从而导致通讯异常。

* **解决方案：**

方案一：先配置NBYTES =非0值，再check TCR/TC=0后, 再配置AUTOEND=1。

方案二：先配置NBYTES =非0值，延迟至少Fpclk/Fi2c = 6周期（如果两者频率差得多, 那就需要延迟更多）再配置AUTOEND=1。

### I2C从字节控制模式下使能reload模式后，PECBYTE位仍然起作用

* **问题描述：**

I2C作为从机发送数据时，使能从字节控制后，在reload模式下想发送带PEC校验的数据时，每次reload字节数的最后一个字节都会传输PEC校验(使能从字节控制后，在reload下PECBYTE不应该生效)。

* **根本原因：**

设计原因

* **解决方案：**

在使能从字节控制后，如果要使用reload模式，不使能PEC校验，在结束reload模式后，再打开PEC校验。

## 通用同步异步收发器(USART)

### USART的SmartCard模式发送数据异常

* **问题描述：**

在使用USART smartcard 模式，发送多包数据时，每包的第一个字节有概率会出现数据发送长度少半个baud bit的情况。

* **根本原因：**

在发送多个字节之间的时间间隔不固定，发送的数据，每个字节都有概率出现Start Bit被异常扣除0.5Bit。

* **解决方案：**

方案一：优先推荐使用SmartCard模式换成普通串口的半双工模式，但是会少NACK和guard time的功能。

方案二：不能背靠背发送数据，需要在第一个字节发送前，重新reset初始化usart，再发送。

## 通用异步收发器(UART3/4)

### UART时钟使能后再配置串口GPIO有概率导致接收异常

* **问题描述：**

对串口进行两次初始化后，上位机发送一包数据（10个字节）到MCU中断内接收时，概率性出现以下两种错误情形：

1. 接收数据完全和发送数据不一致且接收数据量和发送数据量不同。

2. 一包数据的前几个字节接收不到，后面的数据接收正常。

* **根本原因：**

设计原因

* **解决方案：**

初始化时，配置GPIO之前关闭串口时钟，然后配置串口的GPIO，再打开串口时钟可以规避此问题。

void HK\_EVAL\_COMInit(uint32\_t baudrate)

{

    GPIO\_InitTypeDef GPIO\_InitStructure;

    UART\_InitTypeDef UART\_InitStructure;

    NVIC\_InitTypeDef NVIC\_InitStructure;

    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_UART1, DISABLE); //修改代码

    UART\_DeInit(UART1);

    /\* Enable GPIO clock \*/

    RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOA, ENABLE);

    /\* Connect PXx to USARTx\_Tx \*/

    GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource9, GPIO\_AF\_1);

    /\* Connect PXx to USARTx\_Rx \*/

    GPIO\_PinAFConfig(GPIOA,GPIO\_PinSource10,GPIO\_AF\_1);

    /\* Configure USART Tx as alternate function push-pull \*/

    GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9 | GPIO\_Pin\_10;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_10MHz;

    GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

    GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

    GPIO\_Init(GPIOA, &GPIO\_InitStructure);

     /\* Enable USART clock \*/

    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_UART1, ENABLE);  //修改代码

    UART\_InitStructure.UART\_BaudRate = baudrate;

    UART\_InitStructure.UART\_WordLength = UART\_WordLength\_8b;

    UART\_InitStructure.UART\_StopBits = UART\_StopBits\_1;

    UART\_InitStructure.UART\_Parity = UART\_Parity\_No;

    UART\_InitStructure.UART\_Mode = UART\_Mode\_Rx | UART\_Mode\_Tx;

    /\* USART configuration \*/

    UART\_Init(UART1,&UART\_InitStructure);

    NVIC\_InitStructure.NVIC\_IRQChannelPriority=3;

    NVIC\_InitStructure.NVIC\_IRQChannel = UART1\_IRQn;

    NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

    NVIC\_Init(&NVIC\_InitStructure);

    UART\_ITConfig(UART1, UART\_IT\_RXNE, ENABLE);//开启接收中断

    UART\_ClearFlag(UART1, UART\_FLAG\_TC);

    /\* Enable USART \*/

    UART\_Cmd(UART1, ENABLE);

}

## 串行外设接口(SPI/I2S)

### I2S的MCK时钟不能输出

* **问题描述：**

I2S的MCK时钟不能输出。

* **根本原因：**

设计原因

* **解决方案：**

无

### SPI不能配置SPI\_CR2寄存器的DS[3:0]位问题

* **问题描述：**

SPI设置8位/16位 debug 时 显示CR2 ->DS寄存器值都是0, 客户用逻辑分析仪看过只能设置成8bit。

* **根本原因：**

设计原因

* **解决方案：**

在SPI的初始化之前加上\*(uint32\_t\*)0x40013024 = 0xdeadbeef，这个只存在M0系列，M3系列不需要加。

## 通用串行总线全速设备接口(USB)

### USB外设当SETUP事件产生时未自动将端点STAT\_TX置NAK导致枚举失败

* **问题描述：**

1. 问题表现：设备枚举阶段（建立通信阶段）失败，同样的代码在S\*\*T上可以成功导致兼容性问题。

2. 问题产生条件：客户代码在某些时候误操作端点为有效（valid）时，在某些PC下会枚举失败导致无法识别USB设备。

* **根本原因：**

设计原因

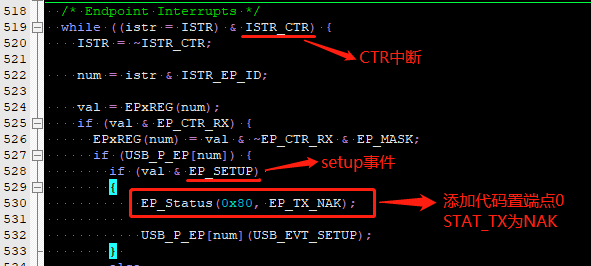
根本原因：

1. 在枚举阶段每次SETUP事件来临时USB外设都未自动将端点STAT\_TX状态置NAK而S\*\*T会，导致接着IN paket来临时，端点状态可能在早些时候被用户置位为了有效所以直接将端点数据送出，软件还未来得及刷入正确的数据导致。

2. 在某些PC下或者加了HUB又表现正常原因是SETUP过后IN Packet到达的时间较晚（主机驱动差异，或者加了HUB延时更长），设备软件有更多时间将正确的数据刷入端点缓冲区所以此时数据正常。

* **解决方案：**

在USB CTR中断中判断有SETUP事件发生时添加一个手动置位端点STAT\_TX为NAK的操作，这里只将端点0的发送状态置NAK，因为SETUP事件只在枚举阶段使用，并只用端点0；如图代码仅供参考：



### USB外设的缓冲区描述表与S\*\*T不同

* **问题描述：**

当USB外设使用时,从0x40006000地址开始的512字节被用来存放USB外设的缓冲区描述表和端点与PC交换的数据包。当最大编号端点仅用TX未用RX时,缓冲区描述表所占用的空间大小与S\*\*T系列不一致, 具体如下:

|  |  |  |  |
| --- | --- | --- | --- |
| S\*\*T实际数据收发缓存区起始地址计算举例 |  |  |  |
| 使用到的端点(端点方向) | Buffer Description Table 占用空间 | 可用于端点通讯数据缓存的最小起始地址 | 十六进制显示 |
| 0(TX+RX),1(TX) | 16\*1.5/2 | 12 | 0x0C |
| 0(TX+RX),1(RX) | 16\*2/2 | 16 | 0x10 |
| 0(TX+RX),1(TX),5(TX) | 16\*5.5/2 | 44 | 0x2C |
| 0(TX+RX),1(TX),7(RX) | 16\*8/2 | 64 | 0x40 |

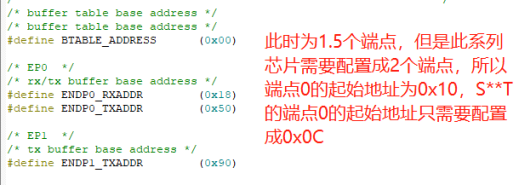
|  |  |  |  |
| --- | --- | --- | --- |
| HK实际数据收发缓存区起始地址计算举例 |  |  |  |
| 使用到的端点(端点方向) | Buffer Description Table 占用空间 | 可用于端点通讯数据缓存的最小起始地址 | 十六进制显示 |
| 0(TX+RX),1(TX) | 16\*1.5/2 | 16 | 0x10 |
| 0(TX+RX),1(RX) | 16\*2/2 | 16 | 0x10 |
| 0(TX+RX),1(TX),5(TX) | 16\*5.5/2 | 48 | 0x30 |
| 0(TX+RX),1(TX),7(RX) | 16\*8/2 | 64 | 0x40 |

* **根本原因：**

设计原因

* **解决方案：**

 只配置半个端点的时候(TX)，需要考虑39A是一个端点对齐的问题，所以需要多加4个字节，如果是配置整个端点(TX和RX)，与S\*\*T配置一致即可。



### USB的ESOF中断与S\*\*T表现不一致

* **问题描述：**

 USB在复位的时候会出现ESOF中断，一次复位产生10次ESOF中断，与S\*\*T不兼容。

* **根本原因：**

  设计原因

* **解决方案：**

 通过软件屏蔽ESOF中断。

/image/ptm/userfiles/system/images/20230722/image_$1682588657644945409.png

### USB的LP\_MODE位与S\*\*T功能不同

* **问题描述：**

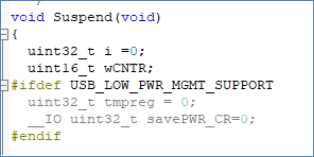
 在某些应用中，为了释放USB总线，USB主机会主动的Suspend设备，Suspend设备的时候，为了降低功耗，软件会置位LP\_MODE，如果软件置位了LP\_MODE，就会导致下一次Wake up的时候不能唤醒设备。

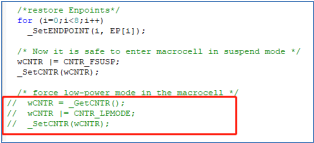
* **根本原因：**

  设计原因

* **解决方案：**

 在Suspend函数中，屏蔽置位LP\_MODE位的代码。





### USB的PDWN位功能失效

* **问题描述：**

本系列芯片的PDWN无效。

* **根本原因：**

 设计原因

* **解决方案：**

 无。

### USB的RESUME位功能失效

* **问题描述：**

 本系列芯片的RESUME无效。

* **根本原因：**

 设计原因

* **解决方案：**

 无。

### USB的FRES位功能失效

* **问题描述：**

 本系列芯片的FRES无效。

* **根本原因：**

 设计原因

* **解决方案：**

 无。

### USB在PLLCLK=48M时进入Stop唤醒后无法正确枚举

* **问题描述：**

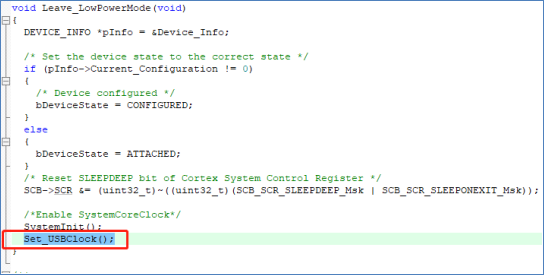
  当PLLCLK=48M，USB挂起之后进入Stop模式，再唤醒会导致USB枚举不成功。

* **根本原因：**

  设计原因

* **解决方案：**

  在函数Leave\_LowPowerMode()里面加上函数Set\_USBClock()来重新配置USB的时钟分频。



### USB在windows10操作系统下的HID模式传输大量数据时IN包无法应答

* **问题描述：**

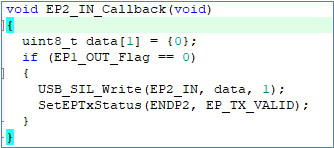
 当使用自定义HID模式发送接收数据的时候，如果IN包数据量过大(一次数据大于64字节，会分几次发送)，会导致设备无法回复主机。

* **根本原因：**

 设计原因

* **解决方案：**

通过当前未使用的端点一直发送IN包的数据给PC，致使PC不主动挂起设备。



### 无法使用USB DFU功能

* **问题描述：**

USB的DFU 固件更新功能无法使用

* **根本原因：**

设计原因

* **解决方案：**

无

### USB端点被置为STALL状态之后的表现与USB标准协议不一致

* **问题描述：**

  USB端点被设置为STALL状态之后，不能进行正常的通信。

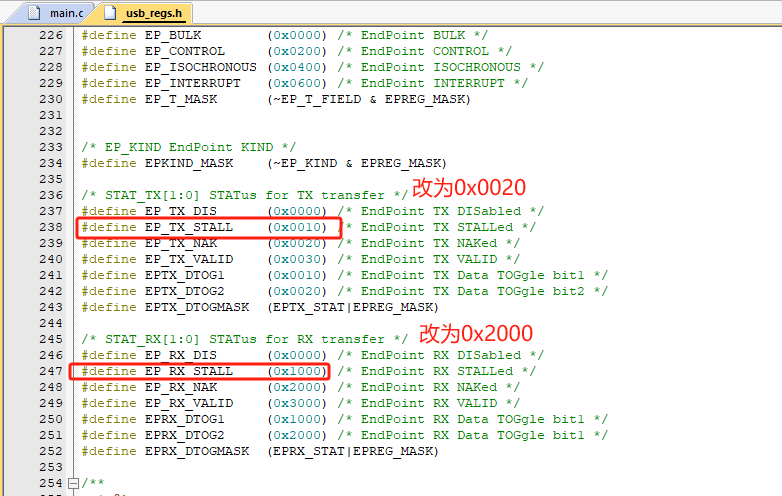
* **根本原因：**

  设计原因

* **解决方案：**

方案一：不要手动设置端点为STALL状态；即是修改函数库：把STALL位修改为VALID位；

标准库举例：



方案二：如果端点必须设置STALL状态或者被硬件强制设置STALL状态，需要在下次主机发送数据包的时候，手动清除STALL状态。

# 缩略语

| 缩写 | 全称 | 中文描述 |
| --- | --- | --- |
| ADC | Analog-To-Digital Converter | 模拟数字转换器 |
| AES | Advanced Encryption Standard | 高级加密标准 |
| AHB | Advanced High-Performance Bus | 高级高性能总线 |
| APB | Advanced Peripheral Bus | 外围总线 |
| CAN | Controller Area Network | 控制器局域网络 |
| CRC | Cyclic Redundancy Check | 循环冗余校验码 |
| DAC | Digital-To-Analog Converter | 数字模拟转换器 |
| DCMI | Digital Camera Memory Interface | 数字相机接口 |
| DMA | Direct Memory Access | 直接存储器访问 |
| ECB | Extended Interrupts and Events Controller | 中断和事件控制器 |
| EXTI | Extended Interrupts and Events Controller | 中断和事件控制器 |
| FM | Fast Mode | 全速模式 |
| GPIO | General Purpose Input Output | 通用输入输出 |
| HSE | High Speed External（Clock Signal） | 高速外部（时钟信号） |
| I2C | Inter-Integrated Circuit | I2C总线 |
| I2S | Inter-IC Sound | I2S总线 |
| IWDG | Independent Watchdog | 独立看门狗 |
| LSB | Least Significant Bit | 最低有效位 |
| LSE | Low-Speed External（Clock Signal） | 低速外部（时钟信号） |
| LSI | Low-Speed Internal（Clock Signal） | 低速内部（时钟信号） |
| LVD | Low Voltage Detect | 低电压检测 |
| MCU | Microcontroller Unit | 微控制单元 |
| MSB | Most Significant Bit | 最高有效位 |
| MSPS | Million Samples Per Second | 每秒百万次采样 |
| NVIC | Nested Vectored Interrupt Controller | 嵌套矢量中断控制器 |
| PDR | Power-Down Reset | 掉电复位 |
| PLL | Phase Locked Loop | 锁相环 |
| POR | Power-On Reset | 上电复位 |
| PVD | Power Voltage Detect | 电压检测 |
| PWM | Pulse Width Modulation | 脉宽调制 |
| QSPI | Queued Serial Peripheral Interface | 队列串行外围接口 |
| RCC | Reset and Clock Control | 复位时钟控制 |
| RISC | Reduced Instruction Set Computing | 精简指令集计算机 |
| RTC | Real Time Clock | 实时时钟 |
| SAI | Serial Audio Interface | 串行音频接口 |
| SDIO | Secure Digital Input and Output | 安全数字输入输出接口 |
| SPI | Serial Peripheral Interface | 串行外设接口 |
| SWD | Serial Wire Debug | 串行线调试 |
| TRNG | True Random Number Generator | 真随机数生成器 |
| USART | Universal Synchronous Asynchronous Receiver Transmitter | 通用同步/异步收发器 |
| WWDG | Window Watchdog | 窗口看门狗 |

# 重要提示

和其他航顺商标均为深圳市航顺芯片技术研发有限公司的商标。本文档提及的其他商标或注册商标，由各自的所有人持有。

在未经深圳市航顺芯片技术研发有限公司同意下，不得以任何形式或途径修改本公司产品规格和数据表中的任何部分以及子部份。深圳市航顺芯片技术研发有限公司在以下方面保留权利：修改数据单和/或产品、停产任一产品或者终止服务不做通知；建议顾客获取最新版本的相关信息，在下定订单前进行核实以确保信息的及时性和完整性。所有的产品都依据订单确认时所提供的销售合同条款出售，条款内容包括保修范围、知识产权和责任范围。

深圳市航顺芯片技术研发有限公司保证在销售期间，产品的性能按照本公司的标准保修。公司认为有必要维持此项保修，会使用测试和其他质量控制技术。除了政府强制规定外，其他仪器的测量表没有必要进行特殊测试。

顾客认可本公司的产品的设计、生产的目的不涉及与生命保障相关或者用于其他危险的活动或者环境的其他系统或产品中。出现故障的产品会导致人身伤亡、财产或环境的损伤（统称高危活动）。人为在高危活动中使用本公司产品，本公司据此不作保修，并且不对顾客或者第三方负有责任。

深圳市航顺芯片技术研发有限公司将会提供与现在一样的技术支持、帮助、建议和信息，（全部包括关于购买的电路板或其他应用程序的设计，开发或调试）。特此声明，对于所有的技术支持、可销性或针对特定用途，及在支持技术无误下，电路板和其他应用程序可以操作或运行的，本公司将不作任何有关此类支持技术的担保，并对您在使用这项支持服务不负任何法律责任。

**所有版权©深圳市航顺芯片技术研发有限公司**

**深圳市航顺芯片技术研发有限公司**

联系电话：0755-83247667

网址：www.hsxp-hk.com