

HK32F103x8xBT6A系列应用笔记

版本：V1.0.11

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 | 2023-07-22 | 首次发布 |
| V1.0.1 | 2023-11-21 | 新增读BKP\_DR寄存器与S\*\*T设计不兼容 新增RTC校准时钟和秒脉冲输出与S\*\*T设计不兼容 新增RTC唤醒定时器重载值写0导致唤醒功能异常 新增清除TEF标志位导致侵入检测功能异常 新增备份寄存器BKP\_DR11~42复位异常 |
| V1.0.2 | 2023-12-01 | 新增SMBus中SMBALERT中断与S\*\*T设计不兼容 |
| V1.0.3 | 2023-12-20 | 更新USB端点被置为STALL状态之后的表现与USB标准协议不一致 |
| V1.0.4 | 2024-01-09 | 新增ADC2在快速交替模式下发生采样数据错乱 |
| V1.0.5 | 2024-03-13 | 新增LSE驱动强度配置无效 |
| V1.0.6 | 2024-03-19 | 新增芯片内部参考电压为0.8V和S\*T（1.2V）存在差异 |
| V1.0.7 | 2024-06-04 | 新增CAN过滤器设计与S\*\*T存在差异 |
| V1.0.8 | 2024-06-07 | 更新SPISlave模式在无时钟时向DR寄存器连续写入数据与S\*\*TM32F103有区别 |
| V1.0.9 | 2024-07-22 | 更新通过DEV\_ID无法区分S\*\*T32F103&HK32F103xx&HK32F103xxA芯片,更新GPIO软件循环定时时间与S\*\*T32F103不一致,更新FLASH在读保护模式下SectorErase会导致部分代码被擦除,更新CACHE在打开的情况下FLASH写及擦除操作时产生错误,更新PVD档位与S\*\*T32F103档位不一致,更新VBAT有电在快速跌落VDD时RTC会被复位,更新Stop模式在HPRE配置为4分频及更高分频系数时功耗异常,更新HSE-25MHz在使用12pF电容时无法起振,更新使用HSE或者HSE-PLL作为系统时钟在使用S\*\*T库函数时程序跑飞,更新IO输入正弦波通过Timer输入捕获测量正弦波频率计数不正确,更新施密特特性的默认状态与S\*\*T不一致,更新在对DMA/HASH/FSMA寄存器进行连续读写操作时读失效,更新AWD唤醒STOP功能失效,更新ADC常规转换在手动切换通道下读取到的数据错误,更新ADC1/2在同步注入模式下ADC2工作状态不能单独配置,更新VREFINT电压值与S\*\*T不同,更新ADC常规通道和注入通道同时刻被触发时的处理机制与S\*\*T不一致,更新ADC在同时开启JAUTO和CONT模式下常规通道和注入通道没有连续转换。,更新ADC注入组间断模式EOC和JEOC的表现形式与S\*\*T不一致,更新DAC输出在对输出BUFF进行操作时会出现输出异常,更新DACIO输出配置与S\*\*T表现不一致,更新DAC输出在连入负电压时输出异常,更新TIM1在作为ADC注入组触发信号来源时与S\*\*T表现不一致,更新TIMPWM输出在手动调节TIMx\_CCRx寄存器时输出异常,更新TIM+DMA控制PWM在发送的数据以0开始时会丢失数据,更新TIM3\_CH1选择PC6在SDIO打开时不能输出PWM信号,更新TIM2-5作为编码器在选择为双沿模式下与S\*\*T表现不一致,更新在使用RTCAlarm唤醒Stop模式或Standby模式后RTCALRF标志位不能置位,更新RTC计数器在Standby唤醒以后计数器概率被清零,更新RTC-&gt;ALRH和RTC-&gt;ALRL寄存器在读功能上与S\*\*T不同,更新USB外设的缓冲区描述表与S\*\*T不同,更新USB的ESOF中断与S\*\*T表现不一致,更新USB在S\*\*T的HAL库上枚举失败,更新USB在CDC模式下使用HAL库上表现异常,更新USB在HID模式下使用HAL库时枚举失败,更新USB的LP\_MODE位与S\*\*T功能不同,更新USB的PDWN位功能失效,更新USB的RESUME位功能失效,更新USB的FRES位功能失效,更新USB在PLLCLK=48M时进入Stop唤醒后无法正确枚举,更新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,更新USB在Camera模式下PC端相机软件未置顶时数据异常,更新CAN在Sleep状态下初始化不成功,更新SPI1/SPI3在I2C1使能时失效,更新SPI在通信时MOSI表现形式与S\*\*T表现不一致,更新I2C在作为MasterRX时NACK不能及时发出,更新I2C时钟在关闭以后对应的Pin未释放,更新SmartCard的Guardtime设置无效,更新SmartCard无法检测NACK,更新USARTIDLE中断与S\*\*TM32F103的区别,更新FSMCNADV信号在QSPIBank1打开时不能输出,更新SDIO在反复初始化SDCard时会概率出现初始化失败,更新使用S\*\*Tlink连接JTAG口连接不成功 |
| V1.0.10 | 2024-08-26 | 更新USB外设的缓冲区描述表与S\*\*T不同,更新USB在CDC模式下使用HAL库上表现异常,更新USB在S\*\*T的HAL库上枚举失败,更新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 |
| V1.0.11 | 2024-08-29 | 新增DAC外部触发（EXTI线9）与S\*\*T不一致,新增ADC外部触发（EXTI线11）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：

* HK32F103x8xBT6A系列
  + - HK32F103CET6A
    - HK32F103TCU6A
    - HK32F103CCU6A
    - HK32F103CBT6A
    - HK32F103CCT6A
    - HK32F103R8T6A
    - HK32F103RBT6A
    - HK32F103V8T6A
    - HK32F103VBT6A
    - HK32F103C8T6A

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。

## 存储器和总线构架

### 通过DEV\_ID无法区分S\*\*T32F103&HK32F103xx&HK32F103xxA芯片

* **问题描述：**

 有一些客户需要通过识别S\*\*T32F103与第一版HK32F103xx芯片和第二版HK32F103xxA芯片，来实现应用程序或更高级后台软件的兼容处理。

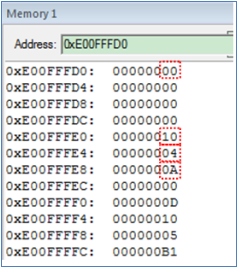
* **根本原因：**

设计差异，HK32F103xx和HK32F103xxA芯片，DEV\_ID相同。

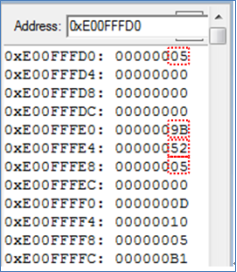
* **解决方案：**

因为DEV\_ID相同，推荐通过读取下图中4个地址的低字节，进行判断。

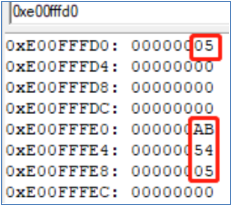
S\*\*T32F103RBT6：



HK32F103RB：



HK32F103xxA：

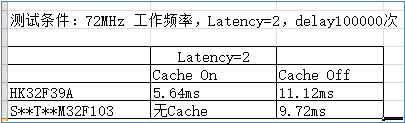


## 嵌入式Flash(FLASH)

### GPIO软件循环定时时间与S\*\*T32F103不一致

* **问题描述：**

用户用软件循环产生定时，用GPIO翻转模拟时序，发现与S\*\*T32F103不一致。



* **根本原因：**

 设计差异

* **解决方案：**

 关闭Cache，此系列效率低一点，打开Cache，此系列效率明显提高。可根据实际情况打开或者关闭Cache。

### FLASH在读保护模式下SectorErase会导致部分代码被擦除

* **问题描述：**

 当该系列芯片在读保护作用下，客户用Keil + S\*\*T-Link进行下载，虽然提示下载不成功，但是芯片内部的部分代码被擦除，客户反馈需要要保持设备原有功能正常。

 Keil+Jlink没有该问题。Jlink，会在下载前，主动去除读保护。

* **根本原因：**

 设计差异，该系列芯片在读保护模式下，debugger选择Sector Erase，会将前4Kbytes 以外的空间擦除掉。

* **解决方案：**

方案一：如果客户用S\*\*T-Link先去掉读保护，再下载或者仿真。

方案二：客户调试时，采用Chip Erase。

### CACHE在打开的情况下FLASH写及擦除操作时产生错误

* **问题描述：**

 CACHE打开的情况下，FLASH写及擦除操作时产生错误。

* **根本原因：**

 设计原因

* **解决方案：**

方案一：在写FLASH->AR寄存器后,操作FLASH->CR寄存器前执行 CACHE->CTL|= 0x0800指令清一下CACHE。该系列对应的库函数已经做了对应修改，如果直接调用该系列库函数操作Flash则无需额外修改。

方案二：在调用flash操作（擦除或编程）前关掉cache，操作之后再打开。

 \*(uint32\_t \*)0x400220D0=0x0;//关闭cache

 \*(uint32\_t \*)0x400220D0=0x1;//开启cache

## 电源控制(PWR)

### PVD档位与S\*\*T32F103档位不一致

* **问题描述：**

PVD档位与S\*\*T32F103档位不一致。

* **根本原因：**

 设计差异。

* **解决方案：**

 无

### VBAT有电在快速跌落VDD时RTC会被复位

* **问题描述：**

 VBAT有电，快速跌落VDD，RTC会被复位。

* **根本原因：**

设计原因：VDD 跌落速度过快时, 硬件掉电检测在电压跌落至危险电压前来不及反应。

* **解决方案：**

在VDD增加电容减缓掉电速度，以便给检测电路足够的动作时间。

### Stop模式在HPRE配置为4分频及更高分频系数时功耗异常

* **问题描述：**

 在进入Stop模式之前，如果HSE处于打开状态，当切换到HSI作为系统时钟，且设置4分频或者更高分频系数时，进入停止模式以后，HSE未被硬件关闭，导致功耗很高。

* **根本原因：**

设计原因

* **解决方案：**

 在进入停止模式之前，通过软件关闭HSE



## 备份寄存器(BKP)

### 备份寄存器BKP\_DR11~42复位异常

* **问题描述：**

用户对备份寄存器BKP\_DR11~42写非零值后，当侵入检测事件发生或者备份域复位时，这部分寄存器不能正常复位。

* **根本原因：**

设计原因

* **解决方案：**

无

### 清除TEF标志位导致侵入检测功能异常

* **问题描述：**

当侵入检测事件发生后，在TAMPER引脚还是有效电平时，清除TEF标志位后发现后续的侵入检测事件无法再触发。

* **根本原因：**

设计原因

* **解决方案：**

方案一：当侵入检测事件发生后，只能在TAMPER引脚电平无效时清除TEF；使能tamper功能前，需要清除一次TEF标志位。

方案二：当侵入检测事件触发后，需要失能再使能侵入检测功能一次；使能tamper功能前，需要清除一次TEF标志位。

### 读BKP\_DR寄存器与S\*\*T设计不兼容

* **问题描述：**

读BKP\_DR寄存器前需要使能BKP时钟，S\*\*T读BKP\_DR寄存器不需要

* **根本原因：**

设计原因

* **解决方案：**

方案一：读BKP\_DR寄存器前需要使能BKP时钟

## 复位和时钟控制(RCC)

### HSE-25MHz在使用12pF电容时无法起振

* **问题描述：**

 客户使用25MHz以上频率HSE，起振电容12pF，不起振。相同电路S\*\*T32F103可以起振。

* **根本原因：**

 设计差异。

* **解决方案：**

 增加起振电容到22pF。

### 使用HSE或者HSE-PLL作为系统时钟在使用S\*\*T库函数时程序跑飞

* **问题描述：**

 客户使用HSE时候，发现存在使用HSE或者HSE+PLL作为系统时钟时，程序跑飞问题。

* **根本原因：**

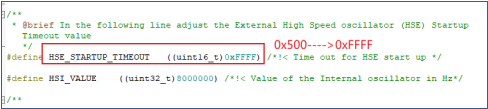
 设计差异，在HSE未起振时就进行了系统时钟切换，导致程序跑飞。

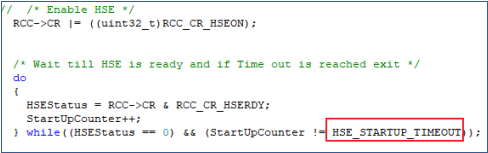
* **解决方案：**

方案一：使用该系列芯片对应的库函数；

方案二：如果使用S\*\*T的库函数按照如下步骤修改代码：

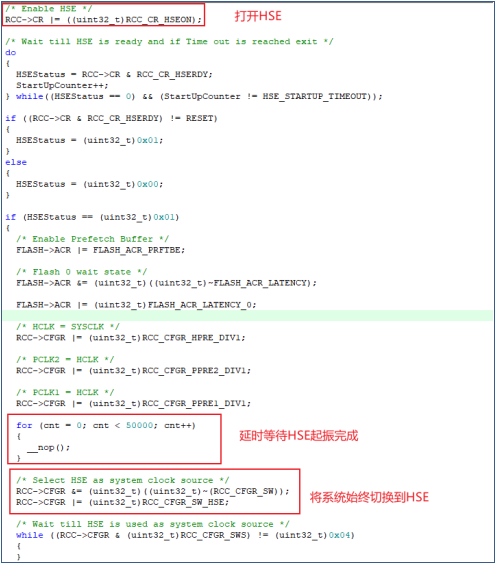
步骤1：将HSE\_STARTUP\_TIMEOUT由0x500->0xFFFF，该宏定义用于HSE起振超时判断。





步骤2：

如果使用HSE作为系统时钟，那么就在HSE打开之后，切换系统时钟之前增加延时。



如果使用HSE+PLL作为系统时钟，那么在HSE和PLL均打开以后切换系统时钟之前，增加延时操作，以等待HSE起振。



### LSE驱动强度配置无效

* **问题描述：**

上电后直接配置  BKP\_LSE\_CRL.IOP[1:0]时，LSE驱动能力没有变化（LSE晶振OUT脚幅值没有变化）

* **根本原因：**

设计原因

* **解决方案：**

如客户需要手动调整LSE驱动能力，需要先写

BKP->LSE\_CTL &= ((uint32\_t)0xFFFF7F3F);

关闭AGC/AUTO\_IOP/NFBYP后，再调整BKP\_LSE\_CRL.IOP[1:0]。

## 通用和复用功能I/O(GPIO和AFIO)

### IO输入正弦波通过Timer输入捕获测量正弦波频率计数不正确

* **问题描述：**

 函数发生器生成1kHz正弦波，通过Timer测量，频率计数不正确，达到100多kHz。IO开启施密特或者不开启，都有相同问题。

* **根本原因：**

 IO输入设计问题。

* **解决方案：**

建议被测信号整形成方波输入，或者芯片内部使能Timer数字滤波。

### 施密特特性的默认状态与S\*\*T不一致

* **问题描述：**

 GPIO的施密特特性默认为关闭状态，和S\*\*T不一致，所以需要手动打开。

* **根本原因：**

 设计差异。

* **解决方案：**

在需要开启施密特特性的IO上，手动开启施密特特性，GPIOx\_IOSEN寄存器的说明参考本系列的用户手册。

GPIOA\_IOSEN&=~GPIO\_Pin\_1; //开启PA1的施密特特性

GPIOA\_IOSEN|= GPIO\_Pin\_1;  //关闭PA1的施密特特性

GPIOB\_IOSEN&=~GPIO\_Pin\_5; //开启PB5的施密特特性

GPIOB\_IOSEN|= GPIO\_Pin\_5;  //关闭PB5的施密特特性

## DMA控制器(DMA)

### 在对DMA/HASH/FSMA寄存器进行连续读写操作时读失效

* **问题描述：**

对同一寄存器进行write-read连续操作时，由于write需要一个时钟周期才能生效，导致read的值为write之前的值。

* **根本原因：**

由于write需要一个时钟周期才能生效。读操作时write的值还没有生效。

* **解决方案：**

 write-read中间插入其他操作，如Nop。

举例说明：

    DMA1\_Channel3->CCR |= 0x00000001;

    while (DMA1\_Channel3->CCR & 0x00000001) ;

执行结果：while读出的第一次DMA1\_Channel3->CCR寄存器的值最后一个bit为0；

修改建议：两个操作中间插入一个Nop。

 DMA1\_Channel3->CCR |= 0x00000001;

Nop();

    while (DMA1\_Channel3->CCR & 0x00000001);

## 模拟/数字转换(ADC)

### AWD唤醒STOP功能失效

* **问题描述：**

ADC的AWD唤醒功能，进入STOP模式,通过RTC定时打开ADC的时钟进行AWD检测。

在设置了AWD电压阈值后，进入STOP模式，若第一次采集电压在设置的阈值范围内，会导致接下来的RTC信号无法再次触发ADC采样比较，AWD功能也就失效了，无法从STOP模式唤醒。

* **根本原因：**

设计原因

* **解决方案：**

无

### ADC常规转换在手动切换通道下读取到的数据错误

* **问题描述：**

当ADC在常规采样模式下，使用并配置一个通道，手动切换通道，软件启动采样，然后通过查询法读取EOC标记，当EOC置位以后读取DR数据，读到的数据是上一次采样的结果。

* **根本原因：**

设计原因，在常规通道采样时，由于EOC标志置位后，数据寄存器DR没有被及时正确更新。DR寄存器要么是上一个采样的数据或者是上一个通道的采样数据。

* **解决方案：**

 在读取DR数据寄存器之前，增加7个NOP指令。代码如下。



### ADC1/2在同步注入模式下ADC2工作状态不能单独配置

* **问题描述：**

在ADC1 和ADC2 工作在同步注入模式，ADC2 的常规通道不能启动转换。

* **根本原因：**

在ADC1 和ADC2 工作在同步注入模式，ADC2 作为从模式，配置好了以后ADC2 的工作状态受ADC21控制，所以后面去配置常规通道转换不能正常软件启动。这是我们芯片与Ｓ\*\*Ｔ芯片的差异。

* **解决方案：**

先定义和启动ADC2 的常规通道的工作状态，再配置主从模式的ADC工作方式，这样就能正常使用。

### VREFINT电压值与S\*\*T不同

* **问题描述：**

该系列芯片的内部参考电压，与HK32F103RBT6、S\*\*T，内部参考基准不同，从1.2V改成了0.8V。

* **根本原因：**

设计不同

* **解决方案：**

方案一：不需要带入0.8V，取V REFINT\_CAL地址处存储的值进行反推；

方案二：直接带入0.8V，就会和S\*\*T不同。

### ADC常规通道和注入通道同时刻被触发时的处理机制与S\*\*T不一致

* **问题描述：**

 当ADC的工作模式为规则通道和注入通道结合采样的时候，如果同时触发规则组和注入组的采样，此时规则通道的触发信号会被忽略， S\*\*T芯片在这种情况下会先触发注入通道再触发规则通道。

* **根本原因：**

设计差异。

* **解决方案：**

 调整软件使触发信号不在同时刻触发即不在同1个时钟周期触发。

### ADC在同时开启JAUTO和CONT模式下常规通道和注入通道没有连续转换。

* **问题描述：**

ADC配置成JAUTO和CONT模式，ADC的规则通道和注入通道只能被触发一次，不能连续转换；

* **根本原因：**

注入状态的转移条件中没有CONT信号，设计上的bug。

* **解决方案：**

    ADC只能在常规通道非CONT模式下使用注入通道的JAUTO模式。

### ADC注入组间断模式EOC和JEOC的表现形式与S\*\*T不一致

* **问题描述：**

此模式通过设置ADC\_CR1寄存器的JDISCEN位激活，SCAN位为0。在一个外部触发事件后，该模式按通道顺序逐个转换ADC\_JSQR寄存器中选择的序列。

一个外部触发信号可以启动ADC\_JSQR寄存器选择的下一个通道序列的转换，直到序列中所有的转换完成为止。总的序列长度由ADC\_JSQR寄存器的JL[1:0]位定义。

S\*\*T：

n=1，被转换的通道 = 1、2、3

第一次触发：通道1被转换，并且产生EOC和JEOC事件

第二次触发：通道2被转换，并且产生EOC

第三次触发：通道3被转换，并且产生EOC

第四次触发：通道1被转换，并且产生EOC和JEOC事件

HK：

n=1，被转换的通道 = 1、2、3

第一次触发：通道1被转换，并且产生EOC和JEOC事件

第二次触发：通道2被转换，并且产生EOC和JEOC事件

第三次触发：通道3被转换 ，并且产生EOC和JEOC事件

第四次触发：通道1被转换，并且产生EOC和JEOC事件

如果配置SCAN为1：

n=1，被转换的通道 = 1、2、3

第一次触发：

第二次触发：

第三次触发：通道3被转换 ，并且产生EOC和JEOC事件

第四次触发：

* **根本原因：**

设计差异。

* **解决方案：**

无。

### ADC2在快速交替模式下发生采样数据错乱

* **问题描述：**

（1）ADC快速交替模式下，ADC2先启动转换，当ADC1转换完成后，ADC1->DR寄存器的低16位存储的为ADC1的转换数据，高16位存储的为ADC2的转换数据。

（2）当ADC1和ADC2配置为双通道时，读取ADC1->DR寄存器，ADC1的两个通道数据是正确的，ADC2两个通道的数据在正确了几个数据以后就开始发生错乱。

* **根本原因：**

ADC1在快速交替模式下，采样和转换时间本应该是14cycle。但是由于设计错误，设计为15cycle。 从而造成BUG。

* **解决方案：**

后面的芯片设计中，修改上面的设计错误

### 芯片内部参考电压为0.8V和S\*T（1.2V）存在差异

* **问题描述：**

客户使用HK32F103CBT6A替换老款HK32F103CBT6,使用内部参考电压，计算的数值与老款差别较大，导致电流电压判断错误。

* **根本原因：**

设计的芯片内部参考电压为0.8V跟S\*T存在差异。

* **解决方案：**

HK32F103xxA/HK32F39Axx的内部参考电压是0.8V，跟老的HK103和S\*T103的参考电压（1.2V）不一样。所以计算的时候要改成0.8V。

### 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

## 数字/模拟转换(DAC)

### DAC输出在对输出BUFF进行操作时会出现输出异常

* **问题描述：**

DAC在使用过程中，当输出一个较低电压，例如0.41V，当对DAC->CR.BOFF1 bit进行操作时，不管是从0->1,还是从1->0，均会出现一个高于实际DAC输出很高的电压值，且不会自行恢复，需要手动给一个trig信号或者将DAC->CR.BOFF1回到上一次触发之前的状态才能恢复正常。

举例说明：

DAC配置为软件触发，DAC->CR.BOFF1 = 0，输出一个0.41V电压(DHR12R1设置为0x1FF)，从PA4输出，当正常输出以后配置DAC->CR.BOFF1 = 1，此时电压会跳变到2.9V左右，且不会自行恢复，当给DAC\_SWTRIGR.SWTRIG1配置为1以后即可自行恢复。

* **根本原因：**

设计原因

* **解决方案：**

方案一：DAC BUFF输出功能，在程序上电初始化配置完毕以后，程序运行中不要更改。

方案二：如果需要在程序运行中更改，且对输出电压的短暂跳变不敏感，那么可通过更改触发方式为DAC\_Trigger\_T1\_TRGO等循环自动触发方式或者在DAC->CR.BOFF1 bit发生改变以后手动给一次触发信号。

### DACIO输出配置与S\*\*T表现不一致

* **问题描述：**

DAC功能和S\*\*\*有个差异，初始化DAC功能的时候我们的芯片一定要配置成模拟输入才能正常工作。S\*\*\*的芯片IO配置成任意模式，DAC都能正常工作。

* **根本原因：**

设计差异

* **解决方案：**

 IO配置为模拟输入

### DAC输出在连入负电压时输出异常

* **问题描述：**

当支持模拟功能的GPIO被连入一个-0.8V左右的电压时，DAC输出会异常（2V变为0.8V左右），纯数字的GPIO没有这个问题。

* **根本原因：**

经过跟DE讨论：这个负电压串进芯片里面后，导致里面有些MOS管不能正常关断，没有彻底关断的MOS管的驱动电流不大，如果DAC输出不带buffer，则会将DAC的输出拉下来了；如果DAC使能输出buffer，因为buffer的输出电流较大，就不会发生异常。

* **解决方案：**

 方案一：不要在GPIO上面输入低于-0.3V的电压（参照本系列对应的数据手册）。

方案二：如果没有低功耗的要求，使用DAC时，建议开启DAC输出buffer。

### 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

## 高级控制定时器(TIM1和TIM8)

### TIM1在作为ADC注入组触发信号来源时与S\*\*T表现不一致

* **问题描述：**

ADC注入通道配置为定时器触发时（比如TIM1的CC4事件触发），只要TIM1的Counter enable，CC4E不置1，MOE也不置1就会触发ADC转换了，这说明我们的芯片TIMER的trigger信号只要TIM1的Counter enable就输出了，而S\*\*T芯片必须是CC4E置1，MOE也置1才能触发ADC转换。

 如果ADC的转换数据一直在有效使用的情况下，（比如说一直用 DMA把数据搬运到有效参与系统运行的场合），如果这样配置我们芯片会在被认为是TIMER没有触发转换的时候（CC4E置0，MOE也置0）产生很多无效的数据影响了系统正常运行。

* **根本原因：**

设计原因

* **解决方案：**

 在不使用ADC转换数据时要将TIMER的Counter disable。要正常使用时再Enable TIMER Counter。

### TIMPWM输出在手动调节TIMx\_CCRx寄存器时输出异常

* **问题描述：**

 当实时调整TIMx\_CCRx来控制输出的PWM时，当TIMx\_CCRx寄存器从不为0配置为0时，TIM的通道输出会在计数器计满TIMx\_ARR时出现毛刺（一个计数周期）。

* **根本原因：**

 设计原因

* **解决方案：**

 当需要配置TIMx\_CCRx从不为0到0时，需要配置TIMx\_CCMRx.OCxPE=0,并且判断当前TIMx\_CNT寄存器的值大于TIMx\_CCRx的值时，配置TIMx\_CCRx为0。

### TIM+DMA控制PWM在发送的数据以0开始时会丢失数据

* **问题描述：**

 当采用DMA方式设置TIMx\_CCRx来控制输出的PWM时，当传输的DMA数据头部或数据中出现0数据，会丢掉0后的两个数据。

* **根本原因：**

 设计原因

* **解决方案：**

 如果DMA发送的数据是以0开始，可以把数据头固定为0，0，1代替0；

 如发送数据为：0x0,0x3d,0x1c,0x20,0x0; 替换为0x0,0x0,0x1,0x3d,0x1c,0x20,0x0;

 如果DMA发送的数据中间存在0的方式，没有解决方案。

## 通用定时器(TIMx)

### TIM3\_CH1选择PC6在SDIO打开时不能输出PWM信号

* **问题描述：**

应用场景同时使用SDIO和TIM3,并且TIM3\_CH1配置为PC6，通过PC6输出PWM信号失败。

* **根本原因：**

设计原因，SDIO启动时，SDIO的所有IO(SDIO\_CLK,SDIO\_CMD,SDIO\_D0~~SDIO\_D7）都被SDIO所控制，不能被用作其他复用功能，当TIM3\_CH1配置为PC6时，SDIO\_D6与PC6是同一管脚，这就导致TIM3\_CH1无法输出PWM。

* **解决方案：**

 更换其他IO口

### TIM2-5作为编码器在选择为双沿模式下与S\*\*T表现不一致

* **问题描述：**

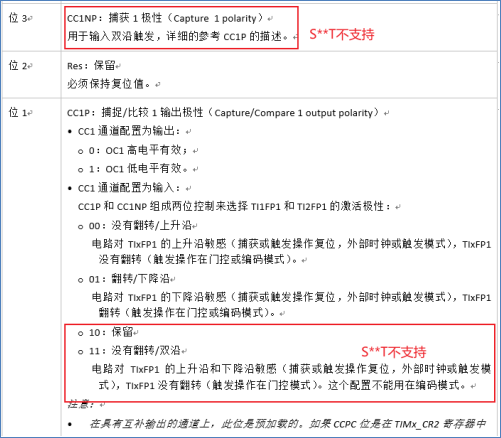
使用TIM2，3，4，5作为编码器功能时，选择双沿模式时，和S\*\*T行为不一致。

/image/ptm/userfiles/system/images/20230722/image_$1682588657292623872.png

* **根本原因：**

 S\*\*T的TIM2，3，4，5不支持双沿模式。

TIM2，3，4，5作为编码器输入时，TIMx\_CCER寄存器存在差异，S\*\*T没有定义CC1NP，所以不支持双沿模式。



* **解决方案：**

使用单边沿模式。

## I2C接口(I2C)

### I2C在作为MasterRX时NACK不能及时发出

* **问题描述：**

 I2C在作为Master RX时，接收完数据以后，NACK无法在最后一个字节接收完成以后及时发出。

* **根本原因：**

设计原因

* **解决方案：**

 用中断或者DMA方式

### I2C时钟在关闭以后对应的Pin未释放

* **问题描述：**

客户需要通过开关SPI和I2C的RCC 时钟来实现SPI（MOSI/MISO/SCS/SCLK）和I2C(SCL/SDA/SMBA)的时分复用，其中I2C\_SMBA和SPI\_MOSI信号配置为同一个Pin PB5。I2C切换到SPI模式后，SPI MOSI输出功能异常。

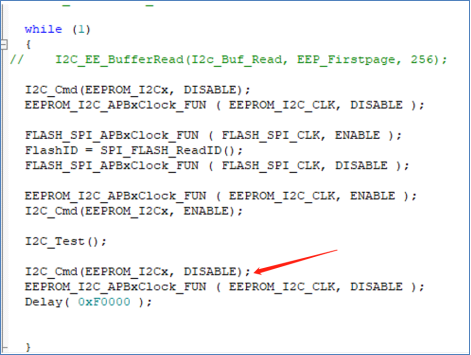
相同使用方式，S\*\*T的SPI可以正确读取数据。

* **根本原因：**

I2C切换成SPI功能后，SBMA信号没有释放，占用了SPI\_MOSI的输出，导致SPI\_MOSI功能不输出。

I2C的SMBA引脚以及SCL/SDA管脚的释放条件是，I2C RCC clock关闭，同时I2C 模块使能位（I2Cx\_CR1的PE位）为0。S\*\*T只需要判断关闭I2C RCC CLK，I2C管脚就被释放了。

* **解决方案：**

建议按照下面的代码，在关闭I2C时钟之前，先关闭I2C\_CR1的PE位。

### SMBus中SMBALERT中断与S\*\*T设计不兼容

* **问题描述：**

S\*\*T\*\*MF1 SMBus中SMBALERT中断是边沿触发，而我们是电平触发。

* **根本原因：**

设计原因

* **解决方案：**

无

## 通用同步异步收发器(USART)

### SmartCard的Guardtime设置无效

* **问题描述：**

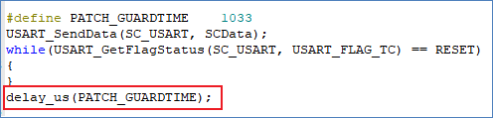
       SmartCard的Guard time设置无效，导致SmartCard通信时可以激活不能通信。

* **根本原因：**

       设计原因，GTPR寄存器GT[7:0]设置无效。

* **解决方案：**

在每次数据发送完成以后添加延时，达到Guard time设置效果；



### SmartCard无法检测NACK

* **问题描述：**

 在Smartcard模式下是单线半双工通信，主机发送数据若发生错误，从器件会拉低stopbit以便告知主机数据出错，所以主机发送的数据需要被回读，该系列芯片在发送数据后RXNE标志无法置起，不能回读数据，无法检测NACK。

* **根本原因：**

 设计原因，主机发送数据时将内部接收器和发送器断开了，导致发送数据无法回读至接收器。

* **解决方案：**

 无

### USARTIDLE中断与S\*\*TM32F103的区别

* **问题描述：**

配置了IDLE中断，芯片上电后会立刻进入IDLE中断，S\*\*TM32F103配置了IDLE中断，上电后在第一次数据来之前不会进入IDLE中断。

* **根本原因：**

设计差异。

* **解决方案：**

上电不使能IDLE中断，在RXNE中断后再打开IDLE中断。

## 串行外设接口(SPI/I2S)

### SPISlave模式在无时钟时向DR寄存器连续写入数据与S\*\*TM32F103有区别

* **问题描述：**

芯片的SPI工作在在Slave模式，在Master时钟来临之前,向DR寄存器写入两次数据(例如0x55,0x01)，在下一次Master来时钟时，DR寄存器发送出来的值缺是0x00,而不是0x01（S\*\*T的为最后一次数据即0x01）

* **根本原因：**

我们的SPI设计为：在等待时钟期间，向DR写入一次数据后，后续再向DR寄存器内写入的数值无法写入，而S\*\*T的设计为：等待时钟期间，向DR写入一次数据后，后续每次写入数据都会覆盖前一次数据。

* **解决方案：**

通过软件避免在这种情况下多次向DR寄存器中写入值。

### SPI1/SPI3在I2C1使能时失效

* **问题描述：**

 SPI1/SPI3 的SCK、MISO、MOSI配置成PB3、PB4、PB5；单独使用I2C1或者单独使用SPI1/SPI3都能正常工作，如果同时使能时，SPI1/SPI3通信不正常。

* **根本原因：**

 I2C\_SMBUS引脚和SPI1/SPI3的MOSI配置成了同一个引脚PB5，而在I2C时钟使能的时候I2C\_SMBUS作为I2C的功能引脚被拉高，导致SPI无法使用。

* **解决方案：**

方案一：分时复用，在使用SPI时失能I2C1时钟和I2C1外设使能位（PE位）。

方案二：SPI复用时不用PB5引脚，换成其它pin。

### SPI在通信时MOSI表现形式与S\*\*T表现不一致

* **问题描述：**

 SPI通信时，MOSI在两个数据间，会被外部上下拉控制，会出现上升或下拉曲线，而不会像S\*\*T一样一直为低。

* **根本原因：**

 设计差异，MOSI在空闲时会保持上一个数据的MSB或者LSB（由CR1的LSBFIRST位决定）；SPI释放MOSI之后，总线无驱动，变为高阻态，为外部上下拉控制。

由设计决定，但保证通信正常。

* **解决方案：**

无

## Quad-SPI接口(QUADSPI)

### FSMCNADV信号在QSPIBank1打开时不能输出

* **问题描述：**

应用场景同时使用FSMC和QSPI Bank1。FSMC会用到NADV信号，NADV信号与QSPI Bank2的数据信号IO3是同一个管脚。在QSPI Bank1打开时，导致NADV信号不输出的状态。

* **根本原因：**

设计原因QSPI Bank1和QSPI Bank2单独的使能信号无效。只要使能了QSPI，QSPI Bank1和QSPI Bank2就会同时使能，就造成QSPI Bank1的数据信号IO0/1/2/3和QSPI Bank2的数据信号IO0/1/2/3都被QSPI控制。不能被用作其他复用功能。

* **解决方案：**

 使用FSMC时，把QSPI1 disable。

 需要使用QSPI1时，Enable QSPI1。

 再次使用FSMC时，又把QSPI1 disable。

### QSPI-DMA接收问题

* **问题描述：**

QSPI-DMA方式接收数据时，QSPI分频系数PSC>2时，DMA无法正确搬运最后一个数据，总是读到0

* **根本原因：**

设计原因

* **解决方案：**

1. 暂时解决办法：设置QSPI FIFO的阈值FTHRES[3:0]≥1； 2. DE仿真得知，在高分频时，使用No-Shift的采样方式，低分频时，使用Half\_cycle-Shift的采样方式。

## SDIO接口(SDIO)

### SDIO在反复初始化SDCard时会概率出现初始化失败

* **问题描述：**

SDIO在程序里反复初始化SD card时，会概率出现初始化失败的情况。

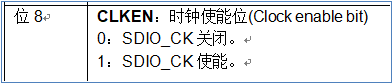
* **根本原因：**

SDIO接口每次对SD card进行初始化时，CMD都会出现一段不需要的低电平。

此时，SDCLK出现一个时钟，SD Card就会误认为通信开始，但是解析CMD/SDCLK命令却不正确。导致SD Card没有响应SDIO接口，从而初始化失败。

* **解决方案：**

 每次程序结束后，设置SDIO\_CLKCR寄存器CLKEN=0, 关闭Clock。下次再初始化SD card时，CMD为低电平时，就不会误出现一个SDCLK。



### SDIO在工作频率大于12MHz时会出现通信异常

* **问题描述：**

SDIO在工作频率大于12MHz时会出现通信异常，导致数据错误。

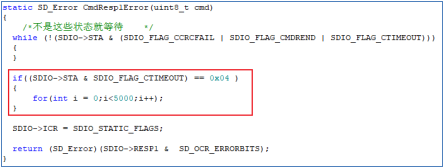
* **根本原因：**

设计原因，在软件清除CTIMEOUT标记时，置位逻辑仍然有效，导致STA寄存器中CTIMEOUT位无法清除。

* **解决方案：**

按照如下步骤：

步骤1：SDIO\_FLAG\_CTIMEOUT 在清空中断标志之前延时等待一下；



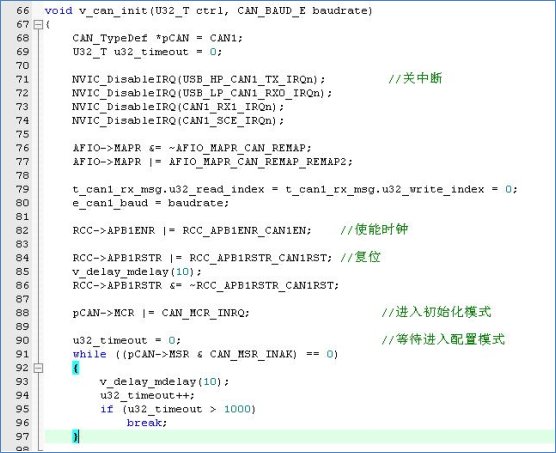
步骤2：建议用户SDIO工作频率最大配置为12MHz.

## 控制器局域网(bxCAN)

### CAN在Sleep状态下初始化不成功

* **问题描述：**

客户CAN初始化代码如下，CAN在Sleep状态下，软件使能初始化请求，INRQ始终无法置位成功；



* **根本原因：**

设计差异，由于CAN模块的复位后的初始状态为Sleep模式，要对CAN初始化，必须先清除CAN的Sleep模式，否则无法操作寄存器。S\*\*T的芯片在不清除Sleep模式，仍可以操作寄存器。

* **解决方案：**

 按照手册或者例程，软件先退出Sleep模式，再进行初始化。

### CAN过滤器设计与S\*\*T存在差异

* **问题描述：**

HK32F39A的CAN1和CAN2都有各自独立的过滤器寄存器，且每个CAN都有0~13共14个过滤器可用，因此CAN1和CAN2的过滤器使用时需要单独配置寄存器。S\*\*T32F103 CAN1和CAN2共用0~27共28个过滤器，且过滤器寄存器只存在于CAN1上，因此只需要配置CAN1的过滤器寄存器，CAN1和CAN2就都能够使用过滤器。



* **根本原因：**

设计原因

* **解决方案：**

若HK32F39A需要使用CAN2的过滤器，需要单独配置CAN2的过滤器寄存器。若S\*\*T32F103需要使用CAN2的过滤器，则只需要配置CAN1的过滤器寄存器。

## USB全速设备接口(USB)

### USB外设的缓冲区描述表与S\*\*T不同

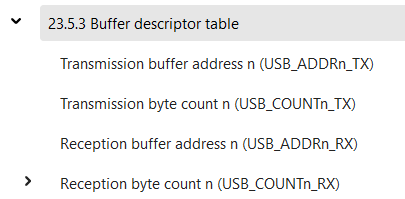
* **问题描述：**

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:

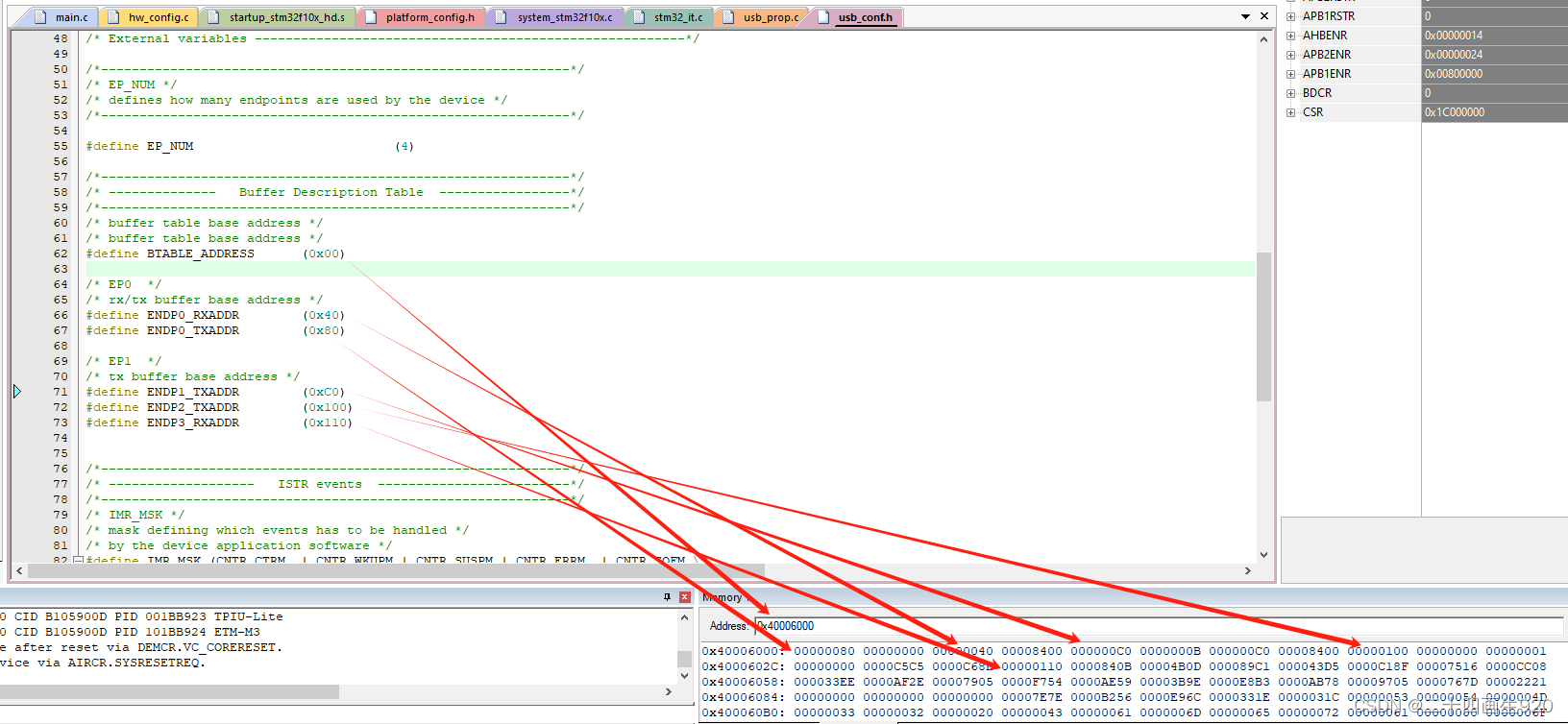
|  |  |  |  |
| --- | --- | --- | --- |
| S\*\*T实际数据收发缓存区起始地址计算举例 |  |  |  |
| 使用到的端点(端点方向) | Buffer Description Table 占用空间 | 可用于端点通讯数据缓存的最小起始地址 | 十六进制显示 |
| 0(TX+RX),1(TX) | 12 | 12 | 0x0C |
| 0(TX+RX),1(RX) | 16 | 16 | 0x10 |
| 0(TX+RX),1(TX),5(TX) | 44 | 44 | 0x2C |
| 0(TX+RX),1(TX),7(RX) | 64 | 64 | 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) | 12 | 16 | 0x10 |
| 0(TX+RX),1(RX) | 16 | 16 | 0x10 |
| 0(TX+RX),1(TX),5(TX) | 44 | 48 | 0x30 |
| 0(TX+RX),1(TX),7(RX) | 64 | 64 | 0x40 |

Buffer Description Table 占用空间如下图所示，一个完整的端点分为TX和RX，TX分为ADDR和COUNT寄存器，RX也分为ADDR和COUNT寄存器，32位对齐，所以这里一个完整的端点占用8个字节



下图是应用实例：

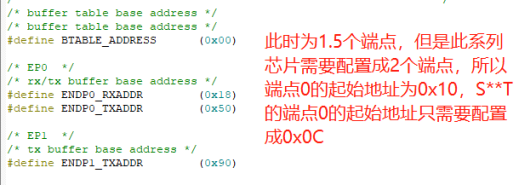


* **根本原因：**

 本系列芯片设计与S\*\*T不一致，S\*\*T按照半个端点(TX或者RX)为基本单位，本系列芯片是按照一个端点(TX和RX)为基本单位。

* **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：**

 设计差异

* **解决方案：**

 通过软件屏蔽ESOF中断。

/image/ptm/userfiles/system/images/20230722/image_$1682588657644945408.png/image/ptm/userfiles/system/images/20230722/image_$1682588657644945409.png

### USB在S\*\*T的HAL库上枚举失败

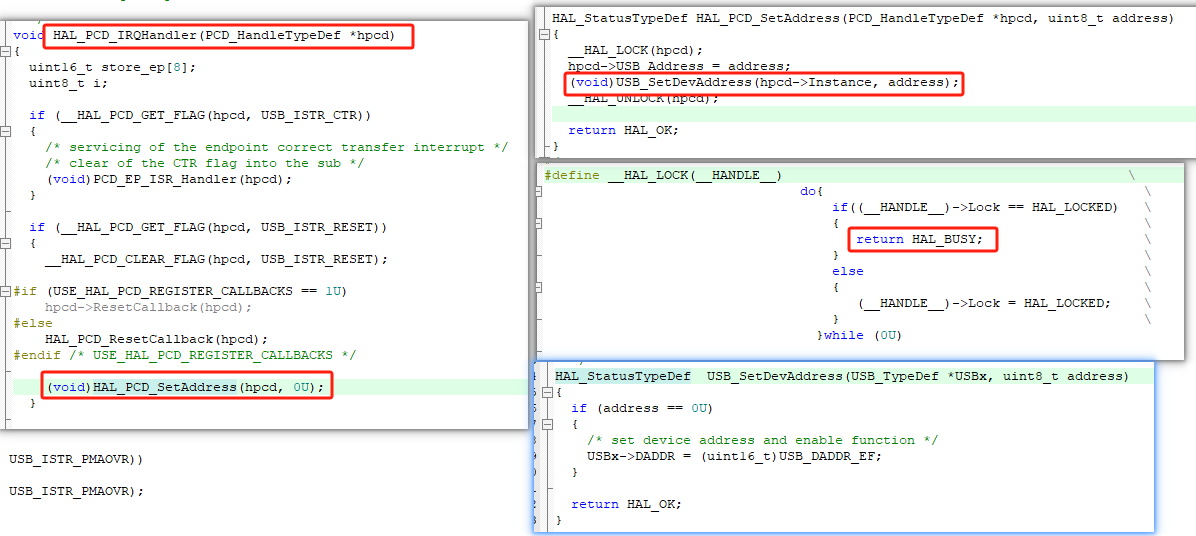
* **问题描述：**

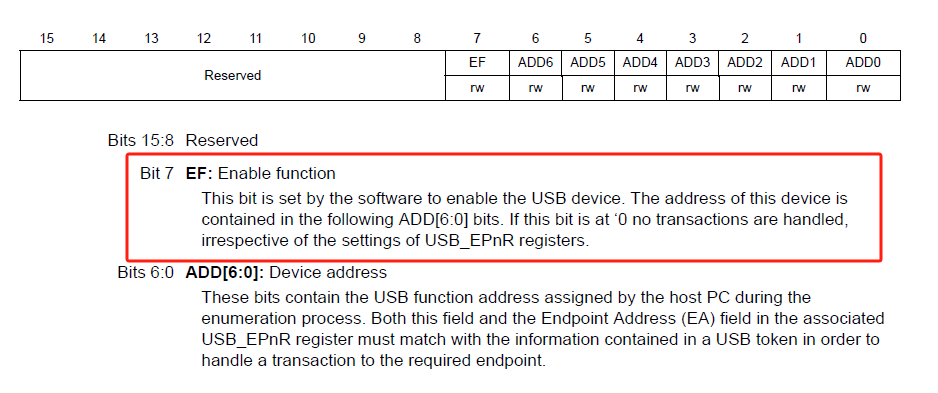
USB使用S\*\*T的HAL库时，出现枚举失败。

* **根本原因：**

如下图所示，HAL库复位设置地址之后，会执行HAL\_PCD\_SetAddress函数，这个函数内部会执行\_\_HAL\_LOCK函数，可看出\_\_HAL\_LOCK如果在被锁定的情况下，HAL\_PCD\_SetAddress会直接返回HAL\_BUSY状态，导致USB\_SetDevAddress函数不能执行。

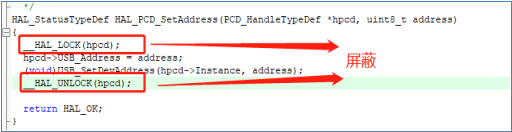
可能的原因是，USB端点状态机制可能与S\*\*T不一致（比如收到SETUP的时候，S\*\*T的端点会自动变成NAK，而HK的不会），在某一处调用了\_\_HAL\_LOCK函数，而没有解锁造成的。





* **解决方案：**

通过软件屏蔽lock，具体操作如下。



### USB在CDC模式下使用HAL库上表现异常

* **问题描述：**

USB在CDC模式下使用S\*\*T的HAL库的时候，即使按照应用笔记“USB在S\*\*T的HAL库上枚举失败”把lock锁屏蔽，也会出现枚举失败。

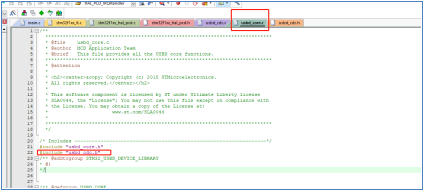
* **根本原因：**

由于在设置配置值的时候，HAL库会执行USB的初始化操作，重新初始化非控制端点(标准库初始化非控制端点在复位的时候)，HAL库初始化非控制端点的时候由于时序问题，导致USB不能完全初始就收到下一包数据，导致出错(这里的解决方式是将非控制端点放在复位的时候初始化)。

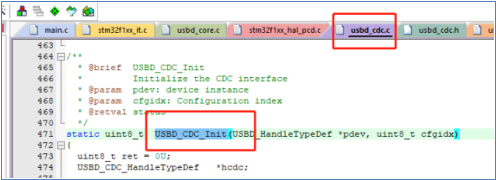
* **解决方案：**

通过应用笔记“USB在S\*\*T的HAL库上枚举失败”的lock屏蔽，再按照如下步骤修改代码可以解决问题：

 步骤1：在usbd\_core.c添加头文件#include "usbd\_cdc.h"，如下



 步骤2：修改usbd\_cdc.c的USBD\_CDC\_Init函数，具体如下(建议全部复制粘贴，下面提供USBD\_CDC\_Init函数的全部修改代码)：



static uint8\_t  USBD\_CDC\_Init(USBD\_HandleTypeDef \*pdev, uint8\_t cfgidx)

{

  uint8\_t ret = 0U;

  USBD\_CDC\_HandleTypeDef   \*hcdc;

  pdev->pClassData = USBD\_malloc(sizeof(USBD\_CDC\_HandleTypeDef));

  if (pdev->pClassData == NULL)

  {

    ret = 1U;

  }

  else

  {

    hcdc = (USBD\_CDC\_HandleTypeDef \*) pdev->pClassData;

    /\* Init  physical Interface components \*/

    ((USBD\_CDC\_ItfTypeDef \*)pdev->pUserData)->Init();

    /\* Init Xfer states \*/

    hcdc->TxState = 0U;

    hcdc->RxState = 0U;

    /\* Prepare Out endpoint to receive next packet \*/

    USBD\_LL\_PrepareReceive(pdev, CDC\_OUT\_EP, hcdc->RxBuffer,

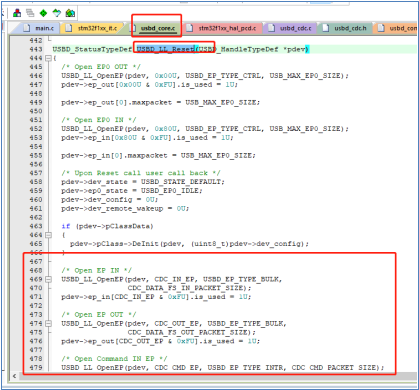
                           CDC\_DATA\_FS\_OUT\_PACKET\_SIZE);

  }

  return ret;

}

<3>在usbd\_core.c的USBD\_LL\_Reset函数下面添加以下代码(建议复制粘贴整段代码)：



USBD\_StatusTypeDef USBD\_LL\_Reset(USBD\_HandleTypeDef \*pdev)

{

  /\* Open EP0 OUT \*/

  USBD\_LL\_OpenEP(pdev, 0x00U, USBD\_EP\_TYPE\_CTRL, USB\_MAX\_EP0\_SIZE);

  pdev->ep\_out[0x00U & 0xFU].is\_used = 1U;

  pdev->ep\_out[0].maxpacket = USB\_MAX\_EP0\_SIZE;

  /\* Open EP0 IN \*/

  USBD\_LL\_OpenEP(pdev, 0x80U, USBD\_EP\_TYPE\_CTRL, USB\_MAX\_EP0\_SIZE);

  pdev->ep\_in[0x80U & 0xFU].is\_used = 1U;

  pdev->ep\_in[0].maxpacket = USB\_MAX\_EP0\_SIZE;

  /\* Upon Reset call user call back \*/

  pdev->dev\_state = USBD\_STATE\_DEFAULT;

  pdev->ep0\_state = USBD\_EP0\_IDLE;

  pdev->dev\_config = 0U;

  pdev->dev\_remote\_wakeup = 0U;

  if (pdev->pClassData)

  {

    pdev->pClass->DeInit(pdev, (uint8\_t)pdev->dev\_config);

  }

  /\* Open EP IN \*/

  USBD\_LL\_OpenEP(pdev, CDC\_IN\_EP, USBD\_EP\_TYPE\_BULK,

                 CDC\_DATA\_FS\_IN\_PACKET\_SIZE);

  pdev->ep\_in[CDC\_IN\_EP & 0xFU].is\_used = 1U;

  /\* Open EP OUT \*/

  USBD\_LL\_OpenEP(pdev, CDC\_OUT\_EP, USBD\_EP\_TYPE\_BULK,

                 CDC\_DATA\_FS\_OUT\_PACKET\_SIZE);

  pdev->ep\_out[CDC\_OUT\_EP & 0xFU].is\_used = 1U;

  /\* Open Command IN EP \*/

  USBD\_LL\_OpenEP(pdev, CDC\_CMD\_EP, USBD\_EP\_TYPE\_INTR, CDC\_CMD\_PACKET\_SIZE);

  pdev->ep\_in[CDC\_CMD\_EP & 0xFU].is\_used = 1U;

  return USBD\_OK;

}

### USB在HID模式下使用HAL库时枚举失败

* **问题描述：**

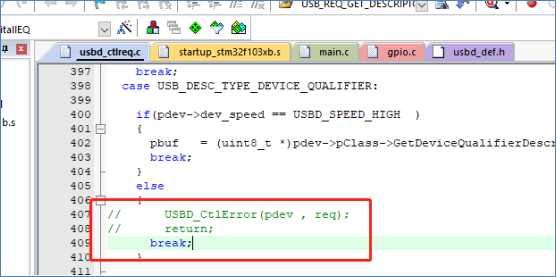
USB在HID模式下使用HAL库的时候，有些电脑会在端点设置STALL状态之后的下一条setup请求命令时乱码，导致枚举失败。

* **根本原因：**

 与USB标准规定不符，USB端点在设置STALL状态之后的，如果主机主动发送下一条指令，USB硬件会主动把端点的状态改为VALID（标准规定在端点设置成STALL之后，如果主机发送下一条指令，USB硬件应该把端点的状态设置为NAK），导致端点被设置成STALL之后的SETUP请求的数据乱码，乱码的数据分为两部分：一部分为本次要发送的数据，另一部分为上一次的数据。

* **解决方案：**

 屏蔽掉软件设置STALL的代码，例如在HAL库的HID类，屏蔽下图处。



### USB的LP\_MODE位与S\*\*T功能不同

* **问题描述：**

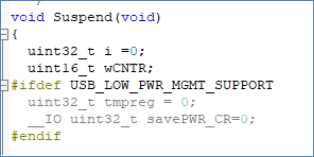
在某些应用中，为了释放USB总线，USB主机会主动的Suspend设备，Suspend设备的时候，为了降低功耗，软件会置位LP\_MODE，如果软件置位了LP\_MODE，就会导致下一次Wake up的时候不能唤醒设备。

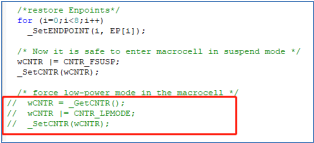
* **根本原因：**

与S\*\*T不兼容，S\*\*T的LP\_MODE位是低功耗模式，但是本系列芯片的LP\_MODE位和PDWN位功能相似，为关闭phy的电源。

* **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：**

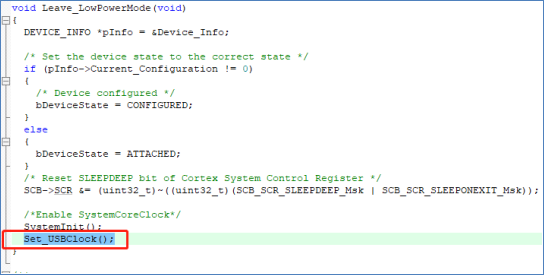
当PLLCLK=48M，USB挂起之后进入Stop模式，再唤醒会导致USB枚举不成功。

* **根本原因：**

 S\*\*T在置位了RCC\_APB1ENR寄存器的USBEN位后，重新初始化时钟的时候，RCC\_CFGR寄存器的USBPRE位会保持不变，但是本系列芯片会自动清零此位(USBPRE位为0表示USB1.5分频)。

* **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：**

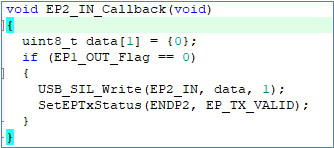
 当使用自定义HID模式发送接收数据的时候，如果IN包数据量过大(一次数据大于64字节，会分几次发送)，会导致通信异常。

* **根本原因：**

 在windows10系统的USB机制中，当PC一直在向设备请求IN包数据，如果设备没有准备好数据（回复NAK），一段时间之后，PC会主动挂起设备，导致下一次有IN包数据发送的时候设备不能够被正常唤醒(windows7不会的原因是因为windows7在没有收到IN包数据的时候不会主动挂起设备)，导致通信异常。

* **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。



### USB在Camera模式下PC端相机软件未置顶时数据异常

* **问题描述：**

 USB作为Camera的时候(同步传输)，当PC端关闭照相机软件，再打开之后，PC不能够正确的读取摄像头数据；如果相机软件不置顶，也会出现不能正确读取数据的情况。

* **根本原因：**

 设计原因，因为关闭或未置顶PC端的照相机软件的时候，PC端会主动挂起设备，在PC端照相软件再次打开时，该系列芯片不能够成功被唤醒。

* **解决方案：**

 置顶PC端的照相机软件。

### 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。

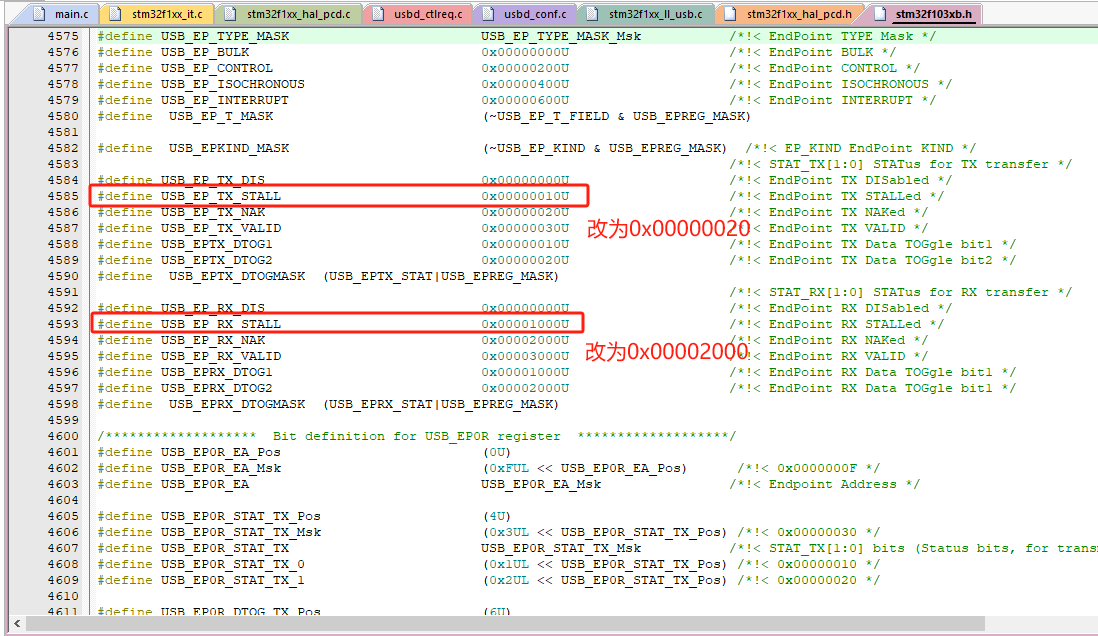
* **根本原因：**

 设计原因，USB设备端点被置为STALL状态之后，如果主机（HOST）干预（除了SOF包之外的所有包），USB设备会自动设置为VALID状态，导致后续不能正常工作。标准协议规定设置了STALL状态之后，如果主机干预，端点需要修改为NAK状态。

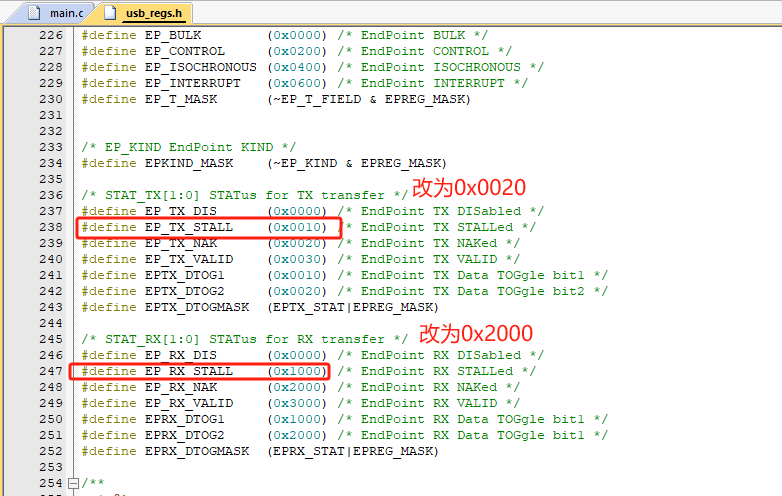
* **解决方案：**

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；

HAL库举例



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。

### 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时，在某些PC下会枚举失败导致无法识别USB设备。

* **根本原因：**

属设计原因

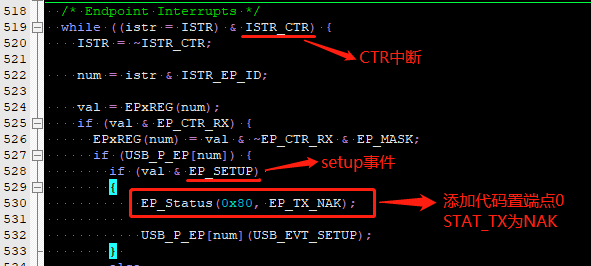
根本原因：

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：**

1. 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：



## 实时时钟(RTC)

### 在使用RTCAlarm唤醒Stop模式或Standby模式后RTCALRF标志位不能置位

* **问题描述：**

 使用RTC Alarm通过EXTI 17唤醒Stop以及使用RTC Alarm唤醒Standby模式时, RTC 的ALRF标志位不能正确置位。

* **根本原因：**

设计原因，RTC Alarm 唤醒时, RTC Alarm 标志位来不及置位。

* **解决方案：**

如果是RTC Alarm唤醒 Standby, 暂无标志位可以表明是否是 Alarm唤醒的。

如果是 RTC Alarm通过 EXTI17 唤醒 Stop , 那么在 EXTI 17 IRQ中断服务程序中不判断Alarm标志位，能够进入到 EXTI17 IRQ handler表明一定是 Alarm发生，另外最好再清除一次Alarm标志位, 因为如果是在RUN 模式下触发 EXTI17 那Alarm标志是可以正确置位的。

### RTC计数器在Standby唤醒以后计数器概率被清零

* **问题描述：**

使用RTC计数器计数时，在Standby唤醒以后发现RTC计数器有概率被清零。

* **根本原因：**

Standby唤醒使能时钟后，芯片设计期望的流程为：把RTC Domain的计数器值同步到VCORE Domain的影子寄存器，然后程序进入RTC配置模式，把VCORE Domain新的配置值配置到RTC Domain。

从Standby唤醒后如果程序执行很快，会立即进入RTC配置模式，此时VCORE Domain的影子寄存器值还未从RTC Domain同步过来，影子寄存器中为0，这样写入RTC Domain计数器的值就是0。

* **解决方案：**

方案一：在打开RTC时钟之后增加延时，延时>LSE周期+两个PCLK，再进入RTC配置模式。

方案二：程序进行RTC配置前先等待RTC\_CRL.RSF寄存器为1。

### RTC-&gt;ALRH和RTC-&gt;ALRL寄存器在读功能上与S\*\*T不同

* **问题描述：**

S\*\*T的RTC->ALRH和RTC->ALRL虽然标明是只写寄存器，但是可以读出数值，而我们的不可读，读数永远只能读到0xFFFFFFFF。

* **根本原因：**

设计差异

* **解决方案：**

通过全局变量或者static变量来存储RTC->ALRH和RTC->ALRL的值。

### RTC唤醒定时器重载值写0导致唤醒功能异常

* **问题描述：**

当RTC的自动唤醒定时器重载值写0时，只能从低功耗唤醒一次，无法周期性唤醒。

* **根本原因：**

当RTC的自动唤醒定时器重载值写0时，唤醒中断只会触发一次，设计原因。

* **解决方案：**

无，自动唤醒定时器重载值不能写0。

### RTC校准时钟和秒脉冲输出与S\*\*T设计不兼容

* **问题描述：**

当RTC校准值CAL[6:0]非0，分别配置PC13为校准时钟输出或者秒脉冲输出时，实际校准时钟输出的是校准后的输出，秒脉冲输出的是校准前的时钟，这一点和S\*\*T的相反，S\*\*T校准时钟输出的是校准前的时钟，秒脉冲输出的是校准后的时钟。

* **根本原因：**

设计原因。

* **解决方案：**

无

## 调试支持(DBG)

### 使用S\*\*Tlink连接JTAG口连接不成功

* **问题描述：**

 用S\*\*T link连接JTAG口，扫描不到设备，连接不成功。

* **根本原因：**

 S\*\*T link设计用于调试S\*\*T芯片，而HK的JTAG内部模块和S\*\*T有差别，导致S\*\*T link识别不是S\*\*T的芯片，连接不成功。

* **解决方案：**

 用SWD接口或换Jlink调试器。

# 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