ES0002
勘误手册

AT32F403A/407使用限制

前言

该勘误表适用于雅特力科技的 AT32F403A/407 系列中的 A、B 版本芯片。该芯片系列集成了 ARM™ 32 位 Cortex®-M4 内核。
表 2 列出了所有的产品型号。表 1 列出了本文涉及产品的识别：

- 通过芯片封装上产品标识下的版本号

表 1. 芯片的识别

销售型号	标注在芯片上的版本代码
AT32F403A/407	“A”
	“B”

1. 产品容量信息和器件唯一ID寄(UID基地址：0x1FFF F7E8)中的Bit[78:76] Mask_Version指明芯片的版本号，即通过地址0x1FFFF7F1的Bit[6:4]获知版本号，比如
   A版：0b000
   B版：0b001

2. 关于在不同芯片封装上识别版本号，请参 2 产品上硅版本号标示。

表 2. 芯片概览

涉及到的芯片	闪存存储器	芯片型号
AT32F403A/407	1024 K字节	AT32F403ACGU7, AT32F403ACGT7, AT32F403ARGT7, AT32F403AVGT7, AT32F407RGT7, AT32F407VGT7, AT32F407AVGT7
	512 K字节	AT32F403ACEU7, AT32F403ACET7, AT32F403ARET7, AT32F403AVET7, AT32F407RET7, AT32F407VET7
	256 K字节	AT32F403ACCU7, AT32F403ACCT7, AT32F403ARCT7, AT32F403AVCT7, AT32F407RCT7, AT32F407VCT7 , AT32F407AVCT7

目录

表目录

图目录

1 AT32F403A/407 芯片的使用限制

下表是所有已经发现的局限性概览：

表 3. 芯片局限性列表

章节	内容	A版	B版
1.1 ADC	1.1.1 使用 ADC 主从模式时，程序卡在判断 CAL 控制位为零的循环中	Fail	Fixed
	1.1.2 ADC 主从普通短位移且反复模式下的转换效率异常	Fail	Fixed
	1.1.3 ADC 主从普通短位移模式在 ADC2 校准期间触发 ADC1 导致数据乱序	Fail	Fixed
1.2 CAN	1.2.1 CAN 通讯数据域期间出现位填充错误会导致下一帧数据错位问题	Fail	Fail
	1.2.2 32 位宽标识符掩码模式下无法有效过滤标准帧的 RTR 位	Fail	Fail
	1.2.3 CAN 在有窄脉冲干扰 BS2 段的条件下有概率会发送非预期的报文	Fail	Fail
	1.2.4 CAN 总线在人为或异常断开后执行邮箱的取消发送命令无效	Fail	Fail
1.3 GPIO	1.3.1 USBFS 的 SOF 作为 TMR2_IS1 输入时导致 TMR2 IO 复用异常	Fail	Fixed
1.4 I2C	1.4.1 作为主机时，当从机释放延展了之后，发送的第一个 clk 可能会变短	Fail	Fixed
	1.4.2 I2C 作为主机时特定条件下 START 发不出来	Fail	Fixed
	1.4.3 在 APB 时钟小于等于 4MHz 时，I2C 做从机无法在 400kHz 的速度下通讯	Fail	Fail
	1.4.4 I2C 在通信开始前，当总线上出现 BUSERR 时序，会误检测到 BUSERR	Fail	Fail
1.5 I2S	1.5.1 I2S 的 CK 线在受到干扰后，通信就无法再自动恢复正常	Fail	Fixed
	1.5.2 I2S 飞利浦协议在特定条件下通讯起始帧数据异常	Fail	Fixed
	1.5.3 I2S PCM 长帧只收模式下接收第一笔数据错位问题	Fail	Fail
	1.5.4 I2S 从发模式非连续通讯状态下误置位 UDR 标志问题	Fail	Fail
	1.5.5 I2S 24 位数据封装成 32 位帧格式接收异常问题	Fail	Fail
1.6 PWC	1.6.1 AHB 分频后 DEEPSLEEP 模式无法被唤醒	Fail	Fail
	1.6.2 Systick 中断误唤醒 DEEPSLEEP	Fail	Fail
	1.6.3 进 DEEPSLEEP 过程瞬间被立即唤醒时异常	Fail	Fixed
	1.6.4 待机唤醒引脚使能时误置位 SWEF 标志问题	Fail	Fixed
1.7 CRM	1.7.1 进入 DEEPSLEEP 模式后 CLKOUT 可能有时钟输出问题	Fail	Fail
1.8 TMR	1.8.1 TMR1/TMR8 用 DMA 访问 0x4C 偏移地址可能导致请求异常	Fail	Fixed
	1.8.2 TMR 在编码器模式下的溢出事件	Fail	Fail
	1.8.3 次 TMR 无法接收由外部触发的主 TMR 复位的输出信号	Fail	Fixed
	1.8.4 未使能定时器时（TMREN = 0），刹车输入无效	Fail	Fail
1.9 USART	1.9.1 特定条件下间隔帧被异常多发一次	Fail	Fixed
	1.9.2 特定条件下智能卡模式使能后的一个空闲帧宽度内无法接收数据	Fail	Fixed
1.10 WWDT	1.10.1 使用 WWDT 中断时，无法清除 RLDF 标志	Fail	Fixed
1.11 RTC	1.11.1 设置 RTC 计数值时，实际值可能是设置值+1	Fail	Fail
1.12 FLASH	1.12.1 sLib 安全库区配置在非零等待区时，特殊情况下可能导致程序出错	Fail	Fail
	1.12.2 擦除零等待区域时，擦除期间执行代码可能导致程序出错	Fail	Fail
	1.12.3 擦除 SPIM 时，擦除期间出现 cpu 读 flash 操作可能导致程序出错	Fail	Fail
1.13 USB	1.13.1 USB 连接会广播数据的 HUB 时，特殊情况下可能出现异常	Fail	Fixed

章节	内容	A版	B版
	1.13.2 USB IN 传输使用部分端点号通信异常	Fail	Fixed
1.14 SPI	1.14.1 SPI 从机硬件 CS 模式下 CS 下降沿不会做重同步	Fail	Fail
1.15 EXINT	1.15.1 EXINT line 一次软件触发会产生两次中断响应	Fixed	Fail

1.1 ADC

1.1.1 使用 ADC 主从模式时，程序卡在判断 CAL 控制位为零的循环中

- 描述：

  在使用ADC主从模式时， ADC1的校准会等待ADC2上电后执行，若单使能ADC1并校准的话，程序会卡在等待校准结束while(adc_calibration_status_get(ADC1))命令处。

- 解决方法：

  修改ADC使能和校准函数顺序，如下示例

  - 原ADC使能及校准顺序

```
adc_enable(ADC1, TRUE);
adc_calibration_init(ADC1);
while(adc_calibration_init_status_get(ADC1));
adc_calibration_start(ADC1);
while(adc_calibration_status_get(ADC1));

adc_enable(ADC2, TRUE);
adc_calibration_init(ADC2);
while(adc_calibration_init_status_get(ADC2));
adc_calibration_start(ADC2);
while(adc_calibration_status_get(ADC2));…
```

  - 修改后ADC使能校准顺序

```
adc_enable(ADC1, TRUE);
adc_enable(ADC2, TRUE);
adc_calibration_init(ADC1);
while(adc_calibration_init_status_get(ADC1));
adc_calibration_start(ADC1);
while(adc_calibration_status_get(ADC1));
adc_calibration_init(ADC2);
while(adc_calibration_init_status_get(ADC2));
adc_calibration_start(ADC2);
while(adc_calibration_status_get(ADC2));…
```

- 改版记录：

  已于硅版本B修正。

1.1.2 ADC 主从普通短位移且反复模式下的转换效率异常

- 描述：

ADC主从普通短位移模式，且配置反复模式时，ADC2的采样转换被控制在ADC1转换完毕后执行，而不是转换起始时序上位移7个ADCCLK，因此导致此模式下整体转换效率降低。

- 解决方法：

  更换使用主从普通长位移模式（相邻两个转换起始时序上位移14个ADCCLK）。

- 改版记录：

  已于硅版本B修正。

1.1.3 ADC 主从普通短位移模式在 ADC2 校准期间触发 ADC1 导致数据乱序

- 描述：

  ADC主从普通短位移模式下，在ADC2校准期间触发ADC1，此时ADC1会保持转换到ADC2校准完毕。随后的触发将会导致ADC转换数据乱序。

- 解决方法：

  禁止校准期间的触发。可通过以下方式实现

  软件触发——控制在校准完毕后执行；

  外部触发——控制在校准完毕后再使能触发源。

- 改版记录：

  已于硅版本B修正。

1.2 CAN

1.2.1 CAN 通讯数据域期间出现位填充错误会导致下一帧数据错位问题

- 问题描述：

  如果CAN因为外部干扰导致通讯的数据域期间出现位填充错误，此时会按照期望停止接收当前帧数据并回馈错误到总线上，但是下一帧通讯报文会出现数据错序，且随后的报文又自动恢复正常的现象。

- 解决方法：

  方法1：开启CAN的错误类型记录中断号对应的错误中断（中断优先级需设定为最高），在CAN错误类型记录中断的中断函数内检测到出现位填充错误时，复位CAN（可只复位CAN寄存器，其相关的GPIO等、NVIC不需复位），并在CAN错误中断函数内完成CAN的重新初始化。

  此方法适用于期望快速完成CAN的初始化，以保障CAN及时参与通讯，避免过多CAN数据丢失的场景。

  以CAN1为例，其典型示例代码如下：

```
/* 开启 CAN 的上次错误中断号对应的错误中断并设定中断最高优先级 */
  nvic_irq_enable(CAN1_SE_IRQn, 0x00, 0x00);
  can_interrupt_enable(CAN1, CAN_ETRIEN_INT, TRUE);
  can_interrupt_enable(CAN1, CAN_EOIEN_INT, TRUE);
/* 中断服务函数 */
void CAN1_SE_IRQHandler(void)
{
  __IO uint32_t err_index = 0;
```

```
  if(can_flag_get(CAN1,CAN_ETR_FLAG) != RESET)
  {
    err_index = CAN1->ests & 0x70;
    can_flag_clear(CAN1, CAN_ETR_FLAG);
    if(err_index == 0x00000010)
    {
      can_reset(CAN1);
      /* 调用CAN初始化函数 */
    }
  }
}
```

注意事项

a) CAN错误类型记录中断的优先级需设定为最高；

b) 由于CAN初始化存在耗时，出现问题后CAN不能及时恢复参与通讯，因此存在丢数据的现象。

方法2：开启CAN的错误类型记录中断号对应的错误中断（中断优先级需设定为最高），在CAN错误类型记录中断的中断函数内检测到出现位填充错误时，复位CAN（可只复位CAN寄存器，其相关的GPIO等、NVIC不需复位），及记录复位事件，并通过在其他低优先级中断或main函数内重新进行CAN初始化。

此方法适用于可接受CAN不能及时参与通讯，需严格保障CAN的重新初始化不影响其他应用逻辑的实现场景。

以CAN1为例，其典型示例代码如下：

```
/* 开启 CAN 的上次错误中断号对应的错误中断并设定中断最高优先级 */
  nvic_irq_enable(CAN1_SE_IRQn, 0x00, 0x00);
  can_interrupt_enable(CAN1, CAN_ETRIEN_INT, TRUE);
  can_interrupt_enable(CAN1, CAN_EOIEN_INT, TRUE);
/* 中断服务函数 */
__IO uint32_t can_reset_index = 0;
void CAN1_SE_IRQHandler(void)
{
  __IO uint32_t err_index = 0;
  if(can_flag_get(CAN1,CAN_ETR_FLAG) != RESET)
  {
    err_index = CAN1->ests & 0x70;
    can_flag_clear(CAN1, CAN_ETR_FLAG);
    if(err_index == 0x00000010)
    {
      can_reset(CAN1);
      can_reset_index = 1;
```

```
    }
  }
}
```

应用再在期望的地方（例如main函数内）查询can_reset_index是否置位，置位后调用CAN初始化函数。

注意事项

a) CAN错误类型记录中断的优先级需设定为最高;

b) 由于CAN初始化及其他应用中断存在耗时，出现问题后CAN不能及时恢复参与通讯，因此存在丢数据的现象。

方法3：开启CAN的错误类型记录中断号对应的错误中断（中断优先级需设定为最高），在CAN错误类型记录中断的中断函数内检测到出现位填充错误时，强制发送一帧标识符优先级最高的无效报文。

此方法适用于不期望消耗时间去复位CAN，CAN总线上的所有报文标识符均为已知，且CAN各个节点有严格按照标识符过滤条件来接收报文的实现场景。

以CAN1为例，其典型示例代码如下：

```
/* 强制发送一帧标识符优先级最高的无效报文 */
static void can_transmit_data(void)
{
    uint8_t transmit_mailbox;
    can_tx_message_type tx_message_struct;
    tx_message_struct.standard_id = 0x0;
    tx_message_struct.extended_id = 0x0;
    tx_message_struct.id_type = CAN_ID_STANDARD;
    tx_message_struct.frame_type = CAN_TFT_DATA;
    tx_message_struct.dlc = 8;
    tx_message_struct.data[0] = 0x00;
    tx_message_struct.data[1] = 0x00;
    tx_message_struct.data[2] = 0x00;
    tx_message_struct.data[3] = 0x00;
    tx_message_struct.data[4] = 0x00;
    tx_message_struct.data[5] = 0x00;
    tx_message_struct.data[6] = 0x00;
    tx_message_struct.data[7] = 0x00;
    can_message_transmit(CAN1, &tx_message_struct);
}
/* 开启 CAN 的上次错误中断号对应的错误中断并设定中断最高优先级 */
  nvic_irq_enable(CAN1_SE_IRQn, 0x00, 0x00);
  can_interrupt_enable(CAN1, CAN_ETRIEN_INT, TRUE);
  can_interrupt_enable(CAN1, CAN_EOIEN_INT, TRUE);
/* 中断服务函数 */
```

```
void CAN1_SE_IRQHandler(void)
{
  __IO uint32_t err_index = 0;
  if(can_flag_get(CAN1,CAN_ETR_FLAG) != RESET)
  {
    err_index = CAN1->ests & 0x70;
    can_flag_clear(CAN1, CAN_ETR_FLAG);
    if(err_index == 0x00000010)
    {
      can_transmit_data();
    }
  }
}
```

注意事项

a) CAN错误类型记录中断的优先级需设定为最高;

b) 此方法仅适用于发送FIFO优先级由报文标识符决定的场景；

c) 此方法无效报文的标识符可修改，但一定要确保其标识符的优先级是CAN总线上最高的，且不会被其他节点当做正常报文接收。

- 改版记录：

  无。

1.2.2 32 位宽标识符掩码模式下无法有效过滤标准帧的 RTR 位

- 问题描述：

  当CAN的过滤器模式设置为32位宽标识符掩码模式时，在标准帧的过滤过程中，RTR位（即远程帧标识符）无法被有效过滤。

  即同时满足如下使用条件时，需要采用解决方法描述进行处理：

  1. 过滤器模式选用32位宽标识符掩码模式
  2. 进行标准帧过滤且不期望接收符合过滤条件的远程帧

- 解决方法：

  方法1：使用软件补充RTR位的过滤。即32位宽标识符掩码模式下过滤标准帧时，使用软件判断RTR位（远程帧标识符）的状态，来决定该帧报文是否被应用需要。参考示例：

```
void CAN1_RX0_IRQHandler(void)
{
  can_rx_message_type rx_message_struct;
  if(can_flag_get(CAN1,CAN_RF0MN_FLAG) != RESET)
  {
    can_message_receive(CAN1, CAN_RX_FIFO0, &rx_message_struct);
    /* only store the data frame,discard the remote frame */
    if((rx_message_struct.id_type == CAN_ID_STANDARD) && (rx_message_struct.frame_type ==
CAN_TFT_DATA))
    {
      /* user store the receive data */
    }
  }
}
```

方法2：更换使用其他过滤模式。结合实际应用需求，使用其他过滤模式来替代，比如32位宽标识符列表模式、16位宽标识符掩码模式或16位宽标识符列表模式。

- 改版记录：

  无。

1.2.3 CAN 在有窄脉冲干扰 BS2 段的条件下有概率会发送非预期的报文

- 问题描述：

  当CAN总线上有大量的窄脉冲干扰（脉冲宽度小于1tq），CAN节点发送时有一定概率发出非预期报文的情况，例如将数据帧发送成远程帧，将标准帧发送成扩展帧，或数据段出现错误。

- 解决方法：

  设置同步跳跃宽度RSAW = BTS2段宽度，以避免出现发出非预期错误的现象。

  需要注意的是，在设置RSAW = BTS2后，在CAN总线有大量干扰的情况下，CAN总线的通信效率会降低。

```
static void can_configuration(void)
{
  …

  /* can baudrate, set baudrate = pclk/(baudrate_div *(3 + bts1_size + bts2_size)) */
  can_baudrate_struct.baudrate_div = 10;
  can_baudrate_struct.rsaw_size = CAN_RSAW_3TQ;
  can_baudrate_struct.bts1_size = CAN_BTS1_8TQ;
  can_baudrate_struct.bts2_size = CAN_BTS2_3TQ;

  …
}
```

- 改版记录：

  无。

1.2.4 CAN 总线在人为或异常断开后执行邮箱的取消发送命令无效

- 问题描述：

  CAN作为报文发送节点，若同时满足如下条件，则在CAN错误被动中断内执行邮箱的取消发送命令将会无效，使得断开CAN总线时刻邮箱内的待发报文的发送并未被实际取消，其会在等待后续CAN总线恢复后重新发送出来。

  1. 人为或异常断开CAN总线（CANH/L）
  2. 自动重传功能有开启

- 解决方法：

  使能CAN的错误被动中断，在其中断函数内关闭自动重传，并在报文发送函数内重新开启自动重传。代码实现如下

  1） 在CAN的初始化时使能错误被动中断

```
nvic_irq_enable(CAN1_SE_IRQn, 0x00, 0x00);
can_interrupt_enable(CAN1, CAN_EPIEN_INT, TRUE);
can_interrupt_enable(CAN1, CAN_EOIEN_INT, TRUE);
```

  2） 在CAN的错误被动中断内关闭自动重传功能

```
void CAN1_SE_IRQHandler(void)
{
  if(can_flag_get(CAN1,CAN_EPF_FLAG) != RESET)
  {
    CAN1->mctrl |= (uint32_t)(1<<4);
    can_flag_clear(CAN1, CAN_EPF_FLAG);
  }
}
```

  3） 在CAN的报文发送函数内重新开启自动重传功能

```
CAN1->mctrl &= (uint32_t)~(1<<4);
```

- 改版记录：

  无。

1.3 GPIO

1.3.1 USBFS 的 SOF 作为 TMR2_IS1 输入时导致 TMR2 IO 复用异常

- 描述：

  使用USBFS的SOF作为TMR2_IS1输入（内部连接）时，会导致TMR2的输入输出通道受影响，其中输入只能使用No Remap的默认IO，输出则都无法使用。

- 解决方法：

  无。

- 改版记录：

  已于硅版本B修正。

1.4 I2C

1.4.1 作为主机时，当从机释放延展了之后，发送的第一个 clk 可能会变短

- 描述：

  作为主机时，在从机延展SCL的地方，当从机释放SCL总线时，主机发出的第一个clk可能会变短，可能导致从机无法正常接收数据。

- 解决方法：

  在从机延展SCL的地方，增加延时，避免在从机释放SCL延展后立即发送数据。

- 改版记录：

  已于硅版本B修正。

1.4.2 I2C 作为主机时特定条件下 START 发不出来

- 描述：

  作为主机时，在发送RESTART的时候，如果从机拉低了总线，可能会造成START发不出来。

- 解决方法：

  在从机延展SCL的地方，增加延时，避免在从机释放SCL延展后立即发送数据。

- 改版记录：

  已于硅版本B修正。

1.4.3 在 APB 时钟小于等于 4MHz 时，I2C 做从机无法在 400kHz 的速度下通讯

- 描述：

  在APB时钟小于等于4MHz时，I2C做从机无法在400kHz的速度下通讯。

- 解决方法：

  APB时钟增加到8MHz或者I2C速度降到100kHz运行。

- 改版记录：

  无。

1.4.4 I2C 在通信开始前，当总线上出现 BUSERR 时序，会误检测到 BUSERR

- 描述：

  I2C同时满足下列条件时，会检测到BUSERR条件，导致不能正常通信

  条件1　I2C使能

  条件2　通信开始前

  条件3　总线上出现BUSERR时序

- 解决方法：

  通讯开始前检查BUSERR标志是否置起，如果置起，清除标志后便可正常通讯。

  也可以开启错误中断，当BUSERR标志置起后在中断里清除。

- 改版记录：

无。

1.5 I2S

1.5.1 I2S 的 CK 线在受到干扰后，通信就无法再自动恢复正常

- 描述：

  I2S的CK与WS信号在芯片内部未做同步约束，导致最终实际通讯时clock线上出现干扰时，该干扰将被I2S当做CK信号进行处理，且后续不会自动恢复正常。

- 解决方法：

  根据期望的音频协议及I2SCLKPOL配置，对WS及CK脚进行内部或者外部上下拉处理。且如果检测到通信出错时，可通过关闭并重新使能I2S来实现恢复。

- 改版记录：

  已于硅版本B修正。

1.5.2 I2S 飞利浦协议在特定条件下通讯起始帧数据异常

- 问题描述：

  I2S飞利浦协议、主收从发、I2SCLKPOL为High配置条件下，通讯的第一帧数据左声道对应的WS信号下降沿不会被真正输出出来，可能导致有些设备无法接收到该左声道数据。

- 解决方法：

  根据期望的音频协议及I2SCLKPOL配置，对WS及CK脚进行内部或者外部上下拉处理。

- 改版记录：

  已于硅版本B修正。

1.5.3 I2S PCM 长帧只收模式下接收第一笔数据错位问题

- 问题描述：

  当PCLK分频系数大于1，I2S配置PCM长帧标准只收模式，若配置I2SCPOL = 0，且在使能I2S前SCK线上异常保持为高电平时，此时接收到的第一笔数据会出现错位。

- 解决方法：

  根据 I2SCLKPOL 配置，对 SCK 脚进行对应的内部或者外部上下拉处理。

- 改版记录：

  无。

1.5.4 I2S 从发模式非连续通讯状态下误置位 UDR 标志问题

- 问题描述：

  I2S从发模式，不连续通讯时，虽在通讯起始前有写入待发数据，但还是会异常置位UDR标志。

- 解决方法：

  结合协议特点，I2S 从发模式建议使用 DMA 或中断等高效的数据传输方式，保障通讯连续。

- 改版记录：

  无。

1.5.5 I2S 24 位数据封装成 32 位帧格式接收异常问题

- 问题描述：

  I2S在24位数据封装成32位帧格式时，8个无效CLK对应的数据会被接收方当做正常数据接收。

- 解决方法：

  解法一：收发双方采用相同的 24 位数据封装成 32 位帧格式的方式；
  解法二：采用软件处理，在此帧格式条件下，丢弃 8 个无效 CLK 对应的数据。

- 改版记录：

  无。

1.6 PWC

1.6.1 AHB 分频后 DEEPSLEEP 模式无法被唤醒

- 问题描述：

  如果将AHB做分频配置后，任何唤醒源唤醒DEEPSLEEP模式都会存在无法唤醒的情况。

- 解决方法：

  使用DEEPSLEEP模式时，不能对AHB进行分频。

  即进DEEPSLEEP模式前，将AHB分频修改为不分频，唤醒后再按照期望设定AHB的分频。

- 改版记录：

  无。

1.6.2 Systick 中断误唤醒 DEEPSLEEP

- 问题描述：

  若进DEEPSLEEP前未关闭Systick或Systick中断时，进DEEPSLEEP后Systick将保持运行，且随后产生的Systick中断会唤醒DEEPSLEEP。

- 解决方法：

  进DEEPSLEEP前关闭Systick或Systick中断。

- 改版记录：

  无。

1.6.3 进 DEEPSLEEP 过程瞬间被立即唤醒时异常

- 问题描述：

  当唤醒源在进DEEPSLEEP模式的过渡状态（约3个LICK时钟周期）出现时，DEEPSLEEP唤醒后有可能会出现漏执行指令问题。

- 解决方法：

  DEEPSLEEP唤醒后，延时3个LICK时钟周期再执行应用程序（详情请参考文档FAQ0114）。

- 改版记录：

  已于硅版本B修正。

1.6.4 待机唤醒引脚使能时误置位 SWEF 标志问题

- 问题描述：

  当待机唤醒引脚使能前，该引脚被用作GPIO通用推挽输出并输出高或通用上拉输入时，此时在待机唤醒引脚使能会会立即误置位SWEF标志。

- 解决方法：

  如果之前有使用待机唤醒引脚作为普通 IO，则在使能待机唤醒引脚前将其 IO 重新初始化为下拉输入或模拟输入。参考示例：

```
gpio_init_type gpio_init_struct;

/* enable the button clock */
crm_periph_clock_enable(CRM_GPIOA_PERIPH_CLOCK, TRUE);

/* set default parameter */
gpio_default_para_init(&gpio_init_struct);

/* configure wakeup pin as input with pull-down */
gpio_init_struct.gpio_drive_strength = GPIO_DRIVE_STRENGTH_STRONGER;
gpio_init_struct.gpio_out_type  = GPIO_OUTPUT_PUSH_PULL;
gpio_init_struct.gpio_mode = GPIO_MODE_INPUT;
gpio_init_struct.gpio_pins = USER_BUTTON_PIN;
gpio_init_struct.gpio_pull = GPIO_PULL_DOWN;
gpio_init(GPIOA, &gpio_init_struct);

/* enable wakeup pin - pa0 */
pwc_wakeup_pin_enable(PWC_WAKEUP_PIN_1, TRUE);
```

- 改版记录：

  无。

1.7 CRM

1.7.1 进入 DEEPSLEEP 模式后 CLKOUT 可能有时钟输出问题

- 问题描述：

  当DEEPSLEEP_DEBUG位设置为0,CLKOUT配置输出系统时钟，在进入DEEPSLEEP模式后,CLKOUT脚上还会有时钟输出,频率为LICK时钟的频率。

- 解决方法：

  在进入DEEPSLEEP模式前，将CLKOUT的时钟输出配置为NOCLK，等退出DEEPSLEEP模式后再配置为输出系统时钟。

- 改版记录：

  无

1.8 TMR

1.8.1 TMR1/TMR8 用 DMA 访问 0x4C 偏移地址可能导致请求异常

- 问题描述：

  TMR（TMR1/TMR8）发出DMA请求时，当前所属地址总线的低8位地址是0x4C，且DMA传输操作未改变当前TMR所属APB总线地址。在此情况下单次TMR的DMA请求在撤销后会再次被置起，可能导致DMA传输异常。

- 解决方法：

  使用其他TMR1/8以外的定时器实现上述功能。

- 改版记录：

已于硅版本B修正。

1.8.2 TMR 在编码器模式下的溢出事件

- 问题描述：

  在编码器模式计数时，如Counter是在0和PR之间来回计数，此上溢或下溢时TMR的OVFIF事件不会置位。

- 解决方法：

  方法1：需占用当前使用编码器TMR的C3IF、C4IF通道为输出模式，设C3DT = AR、C4DT = 0，并使能C3IF、C4IF中断。

  在中断中判断“C3IF事件 & 向下计数”，则此时发生了“下溢”；

  在中断中判断“C4IF事件 & 向上计数”，则此时发生了“上溢”；

  注此解法有限制：如编码器计数的输入信号频率太快，需反复进中断并由软件处理，可能会来不及处理。可应用于编码器的外部输入信号频率不太快的情况。

  方法2：换用有增强模式的定时器(Counter能由16bit扩展为32bit的位宽来计数)，以增加编码器检测正反转的计数范围，将Counter的初值设为PR/2，不让定时器产生溢出。

  注此解法有限制：编码器计数的正反转，只能在一定范围内去转动，如总往一个方向转也会产生溢出。可应用于编码器检测的正反转是在一定范围内去转动的情况。

- 改版记录：

  无。

1.8.3 次 TMR 无法接收由外部触发的主 TMR 复位的输出信号

- 问题描述：
- 在同时满足如下配置条件时，次TMR无法接收到复位信号，导致次TMR无法被触发复位。

  1 主TMR的次模式配置成复位模式，且次模式的触发源为外部信号输入

  2 当主TMR的输出信号为复位信号同时送给次TMR，次TMR的次模式也配置成复位模式

- 解决方法：

  将主TMR的输出信号由复位信号换成溢出信号，即可在主TMR复位时，次定时器也能被触发复位。

- 改版记录：

  已于硅版本B修正。

1.8.4 未使能定时器时（TMREN = 0），刹车输入无效

- 问题描述：

  未使能定时器时（TMREN = 0），刹车输入无效，从而导致刹车输入无法触发刹车事件或中断。

  例如：当使用单周期模式时，一个周期的计数完成后硬件会自动将TMREN清0，此时刹车输入由于上述原因将被屏蔽，从而导致输出使能位（OEN）无法被清零、刹车标志无法置位。

- 解决方法：

  无。

- 改版记录：

  无。

1.9 USART

1.9.1 特定条件下间隔帧被异常多发一次

- 问题描述：
  在串口接收静默条件下执行发送间隔帧命令，若在间隔帧发送结束的同时又出现了唤醒条件，此时发送间隔帧的控制位（即USART_CTRL1寄存器的bit0）无法被硬件有效的清除。导致随后会强制再发送一帧间隔帧。
- 解决方法：
  无。
- 改版记录：
  已于硅版本B修正。

1.9.2 特定条件下智能卡模式使能后的一个空闲帧宽度内无法接收数据

- 问题描述：
  若先使能串口，再使能智能卡模式，由于在智能卡模式使能时硬件会强制发送一帧空闲帧，导致在该空闲帧发送期间，串口无法接收数据（在该空闲帧发送完毕后可正常接收）。
- 解决方法：
  更改初始化顺序为：先使能智能卡模式，后使能串口。
- 改版记录：
  已于硅版本B修正。

1.10 WWDT

1.10.1 使用 WWDT 中断时，无法清除 RLDF 标志

- 描述：
  在使用WWDT的中断时，在进入中断程序中CNT = 0x40时无法清除RLDF标志，所以进入中断后需要先进行喂狗操作，再清除RLDF标志。
- 解决方法：
  在WWDT中断处理函数中，先进行喂狗操作，再清除RLDF标志。

```
void WWDT_IRQHandler(void)
{
  wwdt_counter_set(127);
  wwdt_flag_clear();
}
```

- 改版记录：
  已于硅版本B修正。

1.11 RTC

1.11.1 设置 RTC 计数值时，实际值可能是设置值+1

- 描述：
  设置RTC计数值时，实际值可能是设置值+1。

- 解决方法：

  设置RTC计数值时，先设置分频值。

```
rtc_wait_config_finish();
rtc_divider_set(32767);

rtc_wait_config_finish();
rtc_counter_set(100);
```

- 改版记录：

  无。

1.12 FLASH

1.12.1 sLib 安全库区配置在非零等待区时，特殊情况下可能导致程序出错

- 问题描述：

  当sLib设置在非零等待区，程序运行中出现cpu访问i-code(or d-code)，并快速切bus id的行为，由于内部访问flash区域需要时间，所以可能导致使用切换后的bus id访问sLib，判断bus id不匹配导致回复错误的指令(or 数据)给cpu，最终导致程序出错。

- 解决方法：

  将sLib区域配置在零等待区（ZW）。

- 改版记录：

  无。

1.12.2 擦除零等待区域时，擦除期间执行代码可能导致程序出错

- 问题描述：

  当擦除的是零等待区，在擦除期间程序可以继续执行，如果程序有零等待区取指，然后又有非零等待区的取指，则会读出错误的非零等待区数据，最终导致程序出错。

  举例：擦除零等待区期间中断可以响应，如果中断处理函数调用过程中包含非零等待区和零等待区的取指操作，那么可能导致程序出错

- 解决方法：

  原理是在擦除零等待区域时，需保证擦除期间的所有执行代码全都位于同一区域（全在零等待区或者全在非零等待区）。

  为实现上述原理：可在擦除前关闭中断使能，擦除完成再打开中断使能，并且保证擦除函数相关代码编译到同一区域。

- 改版记录：

  无。

1.12.3 擦除 SPIM 时，擦除期间出现 cpu 读 flash 操作可能导致程序出错

- 问题描述：

  当擦除SPIM期间有cpu read flash情况，会导致该笔read指令被误判断为读SPIM，数据读错，最终导致程序出错。

  举例：擦除SPIM函数本身编译在非零等待区，擦除时该函数本身执行就会从非零等待区取指，即

有read flash操作，那么可能导致程序出错

- 解决方法：

  原理是在擦除SPIM时，需保证擦除期间的所有执行代码不能有read flash的行为。

  为实现上述原理：可在擦除前关闭中断使能，擦除完成再打开中断使能，并且保证擦除函数相关代码编译到零等待区或者RAM。

- 改版记录：

  无。

1.13 USB

1.13.1 USB 连接会广播数据的 HUB 时，特殊情况下可能出现异常

- 问题描述：
  连接到会广播数据的HUB，同时HUB的端口有接其它设备，此时有概率收到主机发给其它设备的数据，在特殊情况下，其它设备的数据会影响USB导致枚举异常。
- 解决方法：

  将APB1的时钟降到48M及以下
- 改版记录：

  已于硅版本B修正。

1.13.2 USB IN 传输使用部分端点号通信异常

- 问题描述：
  使用USB IN传输时，在部分地址和端点号组合下，USB IN传输可能出现异常，表现为IN传输数据不正确或者异常回复NAK状态。
- 解决方法：

  IN 传输可使用端点号2/6/12/14。
- 改版记录：

  已于硅版本B修正。

1.14 SPI

1.14.1 SPI 从机硬件 CS 模式下 CS 下降沿不会做重同步

- 问题描述：

  SPI 从机硬件 CS 模式下，不会在每个 CS 下降沿进行数据传输的起始 CLK 同步。
- 解决方法：

  解法一：严格约束从机CS线，在通讯完毕后及时拉高CS；

  解法二：开启CRC校验，当检测到CRC校验错误时，复位SPI并重新进行握手通讯。
- 改版记录：

  无。

1.15 EXINT

1.15.1 EXINT line 一次软件触发会产生两次中断响应

- 问题描述：

当使用 EXINT line 的软件触发功能时，会在每个软件触发命令执行后立即产生两次该 EXINT 中断线的中断响应。

- 解决方法：

  软件触发对应的中断函数内，将EXINT标志清除命令连续执行两次。

  可使用最新版BSP库的标志清除函数（如下），或参考此方法自行修改代码。

```
void exint_flag_clear(uint32_t exint_line)
{
        if((EXINT->swtrg & exint_line) == exint_line)
        {
            EXINT->intsts = exint_line;
            EXINT->intsts = exint_line;
        }
        else
        {
            EXINT->intsts = exint_line;
        }
}
```

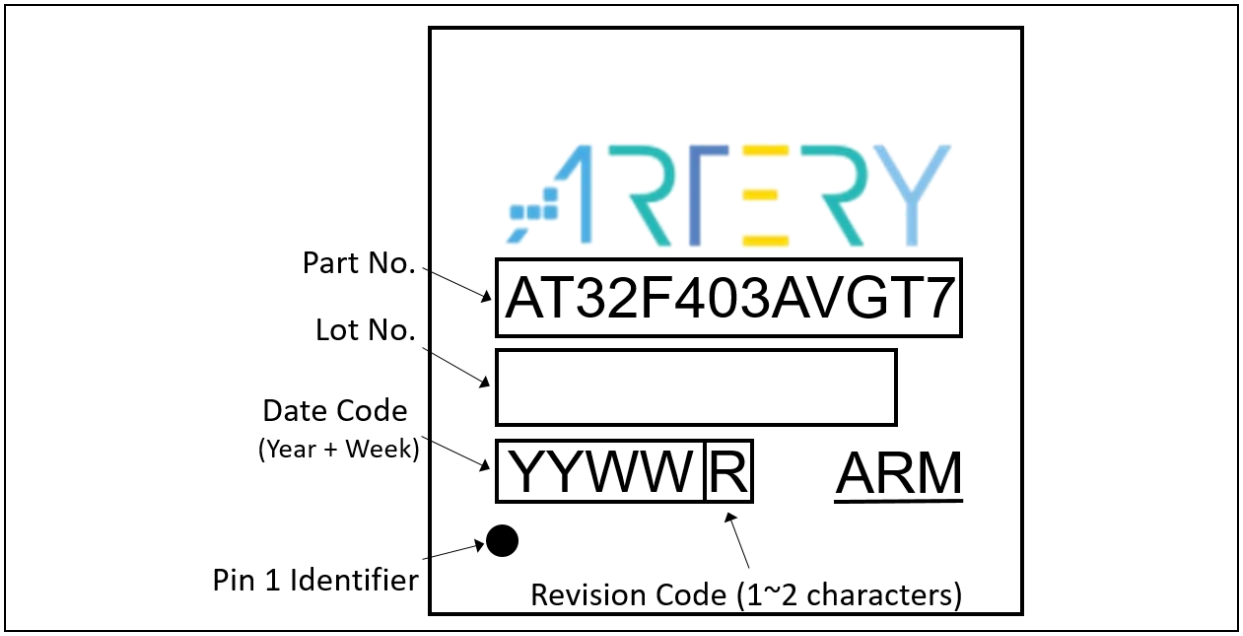
- 改版记录：

  无。

2 产品上硅版本号标示

下图显示了 AT32F403A 芯片上硅版本标示的位置，标出的部分是 R (Revision Code)的第 1 码。’A’表示该芯片硬件版本为硅版本 A。

图 1. 丝印标记(封装俯视图)



3 版本历史

表 4. 文档版本历史

日期	版本	变更
2021.5.27	2.0.0	最初版本
2022.03.01	2.0.1	增加CAN的“32位宽标识符掩码模式下无法有效过滤标准帧的RTR位”
2022.03.30	2.0.2	增加RTC的“设置RTC计数值时，实际值可能是设置值+1”
2022.04.07	2.0.3	增加CAN在有窄脉冲干扰BS2段的条件下有概率会发送非预期的报文 增加FLASH相关勘误描述 增加USB相关勘误描述
2022.04.15	2.0.4	增加“I2S PCM长帧只收模式下接收第一笔数据错位问题” 增加“I2S从发模式非连续通讯状态下误置位UDR标志问题” 增加“I2S 24位数据封装成32位帧格式接收异常问题” 修改描述“I2C作为主机时特定条件下START发不出来”以避免目录过长 增加“SPI从机硬件CS模式下CS下降沿不会做重同步”
2022.04.28	2.0.5	增加“32位宽标识符掩码模式下无法有效过滤标准帧的RTR位”的解法示例 增加“待机唤醒引脚使能时误置位SWEF标志问题”的解法示例 增加“在APB时钟小于等于4MHz时，I2C做从机无法在400kHz的速度下通讯”
2022.07.20	2.0.6	增加EXINT的“EXINT line一次软件触发会产生两次中断响应”
2022.09.06	2.0.7	增加“I2C在通信开始前，当总线上出现BUSERR时序，会误检测到BUSERR”
2022.10.24	2.0.8	更新1.12.2章节描述“擦除零等待区域时，擦除期间执行代码可能导致程序出错”
2023.03.08	2.0.9	增加“次TMR无法接收由外部触发的主TMR复位的输出信号”
2023.08.03	2.0.10	增加“未使能定时器时（TMREN = 0），刹车输入无效” 修改“CAN通讯数据域期间出现位填充错误会导致下一帧数据错位问题”描述 增加“CAN总线在人为或异常断开后执行邮箱的取消发送命令无效”

重要通知 – 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。