ES0004
勘误手册

AT32F425使用限制

前言

该勘误表适用于雅特力科技的AT32F425系列芯片。该芯片系列集成了ARM®32位Cortex™-M4内核。

表 1. 芯片概览

涉及到的芯片	闪存存储器	芯片型号
AT32F425	64 K字节	AT32F425R8T7，AT32F425R8T7 -7， AT32F425C8T7，AT32F425C8U7， AT32F425K8T7，AT32F425K8U7-4， AT32F425F8P7
	32 K字节	AT32F425R6T7，AT32F425R6T7 -7， AT32F425C6T7，AT32F425C6U7， AT32F425K6T7，AT32F425K6U7-4， AT32F425F6P7

目录

表目录

1 AT32F425 芯片的使用限制

下表是所有已经发现的局限性概览：

表 2. 芯片局限性列表

章节	内容
1.1 CAN	1.1.1 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 1.1.2 32位宽标识符掩码模式下无法有效过滤标准帧的RTR位 1.1.3 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 1.1.4 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
1.2 GPIO	1.2.1 PD2在LEXT时钟开启后不能正常使用
1.3 I2S	1.3.1 SPI TI模式及三分频使能会影响I2S通讯 1.3.2 I2S PCM长帧只收模式下接收第一笔数据错位问题 1.3.3 I2S从发模式非连续通讯状态下误置位UDR标志问题 1.3.4 I2S 24位数据封装成32位帧格式接收异常问题
1.4 PWC	1.4.1 AHB分频后DEEPSLEEP模式无法被唤醒 1.4.2 DEEPSLEEP过度状态唤醒后的系统时钟无法正常切换 1.4.3 待机唤醒引脚使能时误置位SWEF标志问题
1.5 SPI	1.5.1 SPI从机TI模式下在使能SPI前异常置位CS脉冲标志 1.5.2 SPI从机硬件CS模式下SCK与Data同步异常
1.6 USART	1.6.1 USART的ROERR标志会异常置位
1.7 TMR	1.7.1 TMR产生的DMA请求的清除方式 1.7.2 未使能定时器时（TMREN = 0），刹车输入无效
1.8 CRM	1.8.1 PLL的倍频系数2倍3倍可能出错
1.9 ERTC	1.9.1 ERTC寄存器写入过程会占住APB总线约为4个ERTC clock时间

1.1 CAN

1.1.1 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。

1.1.2 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。

1.1.3 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 = 8;
  can_baudrate_struct.rsaw_size = CAN_RSAW_3TQ;
  can_baudrate_struct.bts1_size = CAN_BTS1_8TQ;
  can_baudrate_struct.bts2_size = CAN_BTS2_3TQ;

  …
}
```

1.1.4 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：

  当作为报文发送节点，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_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_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);
```

1.2 GPIO

1.2.1 PD2 在 LEXT 时钟开启后不能正常使用

- 问题描述：

LEXT时钟开启后PD2会被抢占，导致该IO口不能正常使用。正常LEXT使能后只应抢占PC14、PC15。

- 解决方法：

无。

1.3 I2S

1.3.1 SPI TI 模式及三分频使能会影响 I2S 通讯

- 问题描述：

I2S应用中，若SPI的TI模式或三分频功能被使能，此时I2S通讯将会出现异常。

- 解决方法：

此为异常使用，SPI的TI模式及三分频功能本不适用于I2S，在I2S应用中禁止使能SPI的TI模式或三分频功能。

1.3.2 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。

1.3.3 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。

1.3.4 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。

1.4 PWC

1.4.1 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。

1.4.2 DEEPSLEEP 过渡状态唤醒后的系统时钟无法正常切换

- 问题描述：
  当唤醒源在进DEEPSLEEP的过渡状态到达时，唤醒后的系统时钟可能无法再重新选择为HEXT或PLL。
- 解决方法：
  在DEEPSLEEP唤醒后，先延时等待约3个LICK时钟周期，随后再进行系统时钟的配置。

1.4.3 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);
```

1.5 SPI

1.5.1 SPI 从机 TI 模式下在使能 SPI 前异常置位 CS 脉冲标志

- 问题描述：

  SPI从机TI模式下，在没有使能SPI时，如果CS和SCK pin受到干扰，那么会产生帧格式错误并响应错误中断。

- 解决方法：

  TI模式与SPI的开关同步进行。即使能SPI时再使能TI模式，失能SPI时同时失能TI模式。

1.5.2 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。

1.6 USART

1.6.1 USART 的 ROERR 标志会异常置位

- 问题描述：

  作为接收方时，在STOP位期间检测到RX线为低电平，并且由此侦测到起始位时，会异常置起ROERR标志。这会导致当发送方的波特率偏大，且发送连续数据时，误置起ROERR标志。

- 解决方法：

  不使用ROERR标志判断接收数据是否溢出，且USART在DMA接收数据时不使能错误中断ERRIEN。

1.7 TMR

1.7.1 TMR 产生的 DMA 请求的清除方式

- 问题描述：

  TMR无法通过Reset/Set DMA/中断使能寄存器（TMRx_IDEN）中相应的DMA请求使能位的方式来清除。

- 解决方法：

  在开启DMA通道传送前，需复位TMR（即复位TMR的CRM时钟），再初始化配置TMR，确保能清除之前已挂起的DMA请求。

1.7.2 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。

1.8 CRM

1.8.1 PLL 的倍频系数 2 倍或 3 倍可能错误

- 描述：

  由于PLL的输出范围的限制，PLL输出时钟应大于等于16 MHz，所以在输入PLL的时钟频率较低时，采用倍频系数2倍或3倍可能出错。

- 解决方法：

  尽量避免使用PLL的2倍和3倍倍频系数。

1.9 ERTC

1.9.1 ERTC 寄存器写入过程会占住 APB 总线约为 4 个 ERTC clock 时间

- 问题描述：

  写ERTC寄存器需要与电池供电域进行约为4个ERTC CLK时间同步，该操作过程APB1会占住，在这期间APB1总线上的DMA传输也会暂停，操作完成后会自动释放APB1总线，DMA传输也会继续执行。

- 解决方法：

  初始化ERTC后，在满足ERTC的使用下，降低写ERTC寄存器频率，以减小对系统应用的影响。

2 版本历史

表 3. 文档版本历史

日期	版本	变更
2021.12.21	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” 增加CRM的“PLL的倍频系数2倍3倍可能错误”
2022.04.15	2.0.2	增加PWC的“待机唤醒引脚使能时误置位SWEF标志问题” 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“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” 增加“SPI从机硬件CS模式下CS下降沿不会做重同步”
2022.04.28	2.0.3	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
2022.09.21	2.0.4	增加“ERTC寄存器写入过程会占住APB总线约为4个ERTC clock时间”
2023.02.08	2.0.5	补充1.9.1章节描述
2023.08.03	2.0.6	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。