

前言

该勘误表适用于雅特力科技的AT32F413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
AT32F413	256 K字节	AT32F413RCT7, AT32F413CCT7, AT32F413CCU7, AT32F413KCU7
	128 K字节	AT32F413RBT7, AT32F413CBT7, AT32F413CBU7, AT32F413KBU7
	64 K字节	AT32F413C8T7

目录

1	AT32F413 芯片的使用限制	5
1.1	GPIO	7
1.1.1	FT（5V 容忍）引脚浮空输入时维持在中间电平	7
1.2	ADC.....	7
1.2.1	ADC 普通通道组序列转换期间更改抢占通道配置将会导致普通通道组转换数据错序	7
1.2.2	ADC 抢占通道组转换结束标志无法清除及置位问题	7
1.3	CAN.....	7
1.3.1	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	7
1.3.2	32 位宽标识符掩码模式下无法有效过滤标准帧的 RTR 位	10
1.3.3	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	11
1.3.4	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.....	12
1.4	CRM.....	12
1.4.1	进入 DEEPSLEEP 模式后 CLKOUT 可能有时钟输出问题.....	12
1.5	I2S.....	12
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	12
1.5.2	I2S 飞利浦协议在特定条件下通讯起始帧数据异常	13
1.5.3	I2S PCM 长帧只收模式下接收第一笔数据错位问题.....	13
1.5.4	I2S 从发模式非连续通讯状态下误置位 UDR 标志问题	13
1.5.5	I2S 24 位数据封装成 32 位帧格式接收异常问题	13
1.6	PWC.....	13
1.6.1	VDD 大于 PVM 阈值时使能 PVM 会误置位一次 PVM 事件	13
1.6.2	AHB 分频后 DEEPSLEEP 模式无法被唤醒	13
1.6.3	Systick 中断可误唤醒 DEEPSLEEP 问题.....	15
1.6.4	特殊条件下的 DEEPSLEEP 唤醒问题.....	15
1.6.5	待机唤醒引脚使能时误置位 SWEF 标志问题	15
1.7	SPI	16
1.7.1	SPI 已置位的接收数据传输 DMA 请求无法通过读 DT 寄存器清除	16
1.7.2	SPI 从机硬件 CS 模式下 CS 下降沿不会做重同步	16
1.8	TMR	16

1.8.1	外部时钟模式 B 加挂起模式条件下挂起模式失效.....	16
1.8.2	TMR 产生的 DMA 请求的清除方式.....	16
1.8.3	TMR 在编码器模式下的溢出事件.....	16
1.8.4	TMR1/TMR8 用 DMA 访问 0x4C 偏移地址可能导致请求异常.....	17
1.8.5	次 TMR 无法接收由外部触发的主 TMR 复位的输出信号.....	17
1.8.6	未使能定时器时 (TMREN = 0)，刹车输入无效.....	17
1.9	USART.....	17
1.9.1	USART 在 IrDA 模式下无法正常接收.....	17
1.9.2	USART 配置完后立即清除 TDC 位，无法正常发送数据.....	18
1.9.3	USART 仅能透过读取数据寄存器清除 RDBF 标志位.....	18
1.9.4	USART 在静默状态下用 DMA 接收仍能收到数据.....	18
1.10	WWDT.....	18
1.10.1	使用 WWDT 中断时，无法清除 RLDF 标志.....	18
1.11	WDT.....	18
1.11.1	使能 WDT 后立即进入 Standby 模式，会发生复位.....	18
1.11.2	使能 WDT 后立即进入 deepsleep 模式，WDT 会未使能.....	19
1.12	RTC.....	19
1.12.1	设置 RTC 计数值时，实际值可能是设置值+1.....	19
1.13	FLASH.....	20
1.13.1	擦除零等待区域时，擦除期间执行代码可能导致程序出错.....	20
1.13.2	擦除 SPI 时，擦除期间出现 cpu 读 flash 操作可能导致程序出错.....	20
1.14	USB.....	20
1.14.1	USB 连接会广播数据的 HUB 时，特殊情况下可能出现异常.....	20
1.14.2	USB IN 传输使用部分端点号通信异常.....	21
1.15	I2C.....	21
1.15.1	在 APB 时钟小于等于 4MHz 时，I2C 做从机无法在 400kHz 的速度下通讯.....	21
1.15.2	I2C 在通信开始前，当总线上出现 BUSERR 时序，会误检测到 BUSERR.....	21
2	版本历史.....	22

表目录

表 1. 芯片概览	1
表 2. 芯片局限性列表	5
表 3. 文档版本历史	22

1 AT32F413 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 GPIO	1.1.1 FT (5V容忍) 引脚浮空输入时维持在中间电平
1.2 ADC	1.2.1 ADC普通通道组序列转换期间更改抢占通道配置将会导致普通通道组转换数据错序
	1.2.2 ADC抢占通道组转换结束标志无法清除及置位问题
1.3 CAN	1.3.1 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
	1.3.2 32位宽标识符掩码模式下无法有效过滤标准帧的RTR位
	1.3.3 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
	1.3.4 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
1.4 CRM	1.4.1 进入DEEPSLEEP模式后CLKOUT可能有时钟输出问题
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
	1.5.2 I2S飞利浦协议在特定条件下通讯起始帧数据异常
	1.5.3 I2S PCM长帧只收模式下接收第一笔数据错位问题
	1.5.4 I2S从发模式非连续通讯状态下误置位UDR标志问题
	1.5.5 I2S 24位数据封装成32位帧格式接收异常问题
1.6 PWC	1.6.1 VDD大于PVM阈值时使能PVM会误置位一次PVM事件
	1.6.2 AHB分频后DEEPSLEEP模式无法被唤醒
	1.6.3 Systick中断可误唤醒DEEPSLEEP问题
	1.6.4 特殊条件下的DEEPSLEEP唤醒问题
	1.6.5 待机唤醒引脚使能时误置位SWEF标志问题
1.7 SPI	1.7.1 SPI已置位的接收数据传输DMA请求无法通过读DT寄存器清除
	1.7.2 SPI从机硬件CS模式下CS下降沿不会做重同步
1.8 TMR	1.8.1 外部时钟模式B加挂起模式条件下挂起模式失效
	1.8.2 TMR产生的DMA请求的清除方式
	1.8.3 TMR在编码器模式下的溢出事件
	1.8.4 TMR1/TMR8用DMA访问0x4C偏移地址可能导致请求异常
	1.8.5 次TMR无法接收由外部触发的主TMR复位的输出信号
	1.8.6 未使能定时器时 (TMREN = 0)，刹车输入无效
1.9 USART	1.9.1 USART在IrDA模式下无法正常接收
	1.9.2 USART配置完后立即清除TDC位，无法正常发送数据
	1.9.3 USART仅能透过读取数据寄存器清除RDBF标志位
	1.9.4 USART在静默状态下用DMA接收仍能收到数据
1.10 WWDT	1.10.1 使用WWDT中断时，无法清除RLDF标志
1.11 WDT	1.11.1 使能WDT后立即进入Standby模式，会发生复位
	1.11.2 使能WDT后立即进入deepsleep模式，WDT会未使能
1.12 RTC	1.12.1 设置RTC计数值时，实际值可能是设置值+1
1.13 FLASH	1.13.1 擦除零等待区域时，擦除期间执行代码可能导致程序出错
	1.13.2 擦除SPIM时，擦除期间出现cpu读flash操作可能导致程序出错
1.14 USB	1.14.1 USB连接会广播数据的HUB时，特殊情况下可能出现异常
	1.14.2 USB IN传输使用部分端点号通信异常

章节	内容
1.15 I2C	1.15.1在APB时钟小于等于4MHz时，I2C做从机无法在400kHz的速度下通讯
	1.15.2 I2C在通信开始前，当总线上出现BUSERR时序，会误检测到BUSERR

1.1 GPIO

1.1.1 FT（5V 容忍）引脚浮空输入时维持在中间电平

- 问题描述：
5 V容忍管脚在浮空输入模式时，仍有约小于10 μ A的上拉电流能力，以致管脚可能维持约2.0 V。
- 解决方法：
引脚外部加下拉电阻（200 k Ω 或以下）解决。

1.2 ADC

1.2.1 ADC 普通通道组序列转换期间更改抢占通道配置将会导致普通通道组转换数据错序

- 问题描述：
ADC 普通通道组序列及反复转换模式下，在普通通道组转换过程中更改抢占通道组通道配置将导致普通通道组转换数据错序。
例如：普通通道组序列转换 1、2、3、4 通道，在通道 2 转换期间更改抢占通道组通道配置，将会导致普通通道组通道 2 重复转换两次，最终普通通道转换数据为 1、2、2、3、4。
- 解决方法：
同时使用多通道普通及抢占转换时，配置完抢占通道组后，不能再去切换其配置。

1.2.2 ADC 抢占通道组转换结束标志无法清除及置位问题

- 问题描述：
当ADC“清除抢占通道组转换结束标志PCCE”与“普通通道组转换结束CCE事件”同时发生时，会出现PCCE标志无法被及时清除，且下一次PCCE标志无法及时置位问题。
- 解决方法：
在原清除PCCE标志命令处再执行一次该标志的清除命令。参考如下

```
/* 修改前 */  
adc_flag_clear(ADC1, ADC_PCCE_FLAG);  
/* 修改后 */  
adc_flag_clear(ADC1, ADC_PCCE_FLAG);  
adc_flag_clear(ADC1, ADC_PCCE_FLAG);
```

- 改版记录：
无。

1.3 CAN

1.3.1 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

- CAN错误类型记录中断的优先级需设定为最高；
- 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;
        }
    }
}
```

应用再再期望的地方（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

- CAN错误类型记录中断的优先级需设定为最高;
- 此方法仅适用于发送FIFO优先级由报文标识符决定的场景;
- 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。

1.3.2 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:
 - 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。

1.3.3 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;

    ...
}
```

1.3.4 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_EOIEIN_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);
```

1.4 CRM

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

- 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。

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

- 问题描述:

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。

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 SCK 脚进行内部或者外部上下拉处理。

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。

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。

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。

1.6 PWC

1.6.1 VDD 大于 PVM 阈值时使能 PVM 会误置位一次 PVM 事件

- 问题描述：
当VDD大于PVM阈值时，使能PWC的电压监测功能，会立即产生一次非期望的PVM事件。
- 解决方法：
在PVM初始化期间清除掉该非预期的PVM事件。

1.6.2 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。

1.6.3 Systick 中断可误唤醒 DEEPSLEEP 问题

- 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。

1.6.4 特殊条件下的 DEEPSLEEP 唤醒问题

- 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）。

1.6.5 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时会立即误置位 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);
```

1.7 SPI

1.7.1 SPI 已置位的接收数据传输 DMA 请求无法通过读 DT 寄存器清除

- 问题描述：
示例，在使用SPI全双工实现分时收发应用中，SPI发送期间置位的无效数据接收传输的DMA请求，将无法通过读DT寄存器进行清除。
- 解决方法：
在SPI接收对应的DMA通道处于关闭状态下，使用关闭SPI使能命令替代原读DT寄存器命令，随后再在期望的通讯起始处开启SPI使能。

1.7.2 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。

1.8 TMR

1.8.1 外部时钟模式 B 加挂起模式条件下挂起模式失效

- 问题描述：外部时钟模式 B 下，配置了挂起模式，挂起模式不起作用（无论挂起电平是高电平，还是低电平，Timer 都在计数）
- 解决方法：
无。

1.8.2 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。

1.8.3 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。

1.8.4 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。

1.8.5 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。

1.8.6 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 USART 在 IrDA 模式下无法正常接收

- 问题描述:
USART 的 IrDA 模式下, 当 USART 波特率配置为小于等于 38400 时, USART 无法正常接收到数据。
- 解决方法:
USART 波特率避免配置小于等于 38400。

1.9.2 USART 配置完后立即清除 TDC 位，无法正常发送数据

- 问题描述：
在初始化 USART 后，立即清 TDC flag，然后判断 TDC 置起，再发送数据，此时数据无法正常发出。
- 解决方法：
按照正常的逻辑来说，不应该软件去清除 TDC 标志。TDC 标志置起，表明数据发送完毕。若将 TDC 清除的话，只有再发数据后，TDC 才能置起，不然 TDC 标志会一直处于 0 状态。

1.9.3 USART 仅能透过读取数据寄存器清除 RDBF 标志位

- 问题描述：
普通异步通讯模式下，读数据寄存器非空标志 RDBF 的清除方式有二，其一为通过对 USART_DT 寄存器的读操作清除，另一个为直接对状态寄存器的 RDBF Bit 写 0 来清除，B 版仅能透过对 USART_DT 进行读操作清除。
- 解决方法：
无。

1.9.4 USART 在静默状态下用 DMA 接收仍能收到数据

- 问题描述：
USART 在静默状态（地址匹配唤醒模式）下，向 RX 端发送数据，此时 RDBF 并未置位，但仍能产生 DMA 的接收请求且 DT 数据寄存器也会接收数据，此时静默模式会失效。
- 解决方法：
无。

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();
}
```

1.11 WDT

1.11.1 使能 WDT 后立即进入 Standby 模式，会发生复位

- 描述：
使能 WDT（WDT_CMD = 0Xcccc）后立即进入 standby 模式，会立即发生复位。
- 解决方法：
使能 WDT 后加入一段延时约几个 us，再进入 Standby 模式。

1.11.2 使能 WDT 后立即进入 deepsleep 模式，WDT 会未使能

- 描述：
使能WDT（WDT_CMD = 0xCCCC）后立即进入deepsleep模式，WDT会未使能。
- 解决方法：
使能WDT后加入一段延时约30us，再进入deepsleep模式。

1.12 RTC

1.12.1 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);
```

1.13 FLASH

1.13.1 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。

1.13.2 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。

1.14 USB

1.14.1 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。

1.14.2 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。

1.15 I2C

1.15.1 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。

1.15.2 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。

2 版本历史

表 3. 文档版本历史

日期	版本	变更
2021.11.22	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”
2022.04.15	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 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	2.0.4	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“FT（5V容忍）引脚浮空输入时维持在中间电平” 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”
2022.09.06	2.0.5	增加“I2C在通信开始前，当总线上出现BUSERR时序，会误检测到BUSERR”
2022.09.27	2.0.6	增加“ADC抢占通道组转换结束标志无法清除及置位问题” 更新1.13.1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	2.0.7	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	2.0.8	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。

© 2023 雅特力科技 保留所有权利