Errata Sheet

YTM32B1HA0

This document details the silicon errata known at the time of publication for the YTM32B1HA0 device. For additional information, please see YTM32B1HA0 Reference Manual or Data sheet.

Revision History

Rev.No.	Date	Substantive Change(s)
1.6	2025-09-03	Added SCU errata E120002 Updated workaround for SPI errata E403002 Added MPWM errata E505004 Added ADC errata E600007 Replenish internal channels 32-39 to high channels for ADC errata E600002 Removed CMU errata E715002
1.5	2025-06-26	E0005 ADC errata ID is updated to E600002 E0007 I2C errata ID is updated to E407001 E0008 GPIO errata ID is updated to E300001 E0009 GPIO errata ID is updated to E300002 E0011 MPWM errata ID is updated to E505001 E0012 eTMR errata ID is updated to E503003 Removed E0013 eTMR errata E0014 FlexCAN errata ID is updated to E401001.D E0015 MPWM errata ID is updated to E505002 E0016 CLOCK errata ID is updated to E120001 E0017 FlexCAN errata ID is updated to E401002 E0019 eTMR errata ID is updated to E503005 E0022 LINFlexD DMA errata ID is updated to E406002 Added DMA errata E010002 Added SCU errata E120001 Added EFM(flash) errata E220004.C Added GPIO errata E300004 Added SPI errata E403001, E403002, E403003 Added LINFlexD errata E406001 Added QSPI errata E410001, E410002, E410003, E410004 Added LPTMR errata E502001 Added eTMR errata E503006 Added MPWM errata E505003 Added RTC errata E509001 Added ADC errata E600003, E600005 Added HCU errata E702003 Added CMU errata E715002 Added FMU errata E721001
1.4	2024-10-28	Updated E0014 and E0017 FlexCAN Added E0019 eTMR Added E0022 LINFlexD DMA
1.3	2024-08-16	Added E0009 GPIO Added E0011 MPWM Added E0012 eTMR Added E0013 eTMR Added E0014 FlexCAN Added E0015 MPWM Added E0016 CLOCK Added E0017 FlexCAN
1.2	2024-02-27	Added E0007 I2C Added E0008 GPIO
1.1	2023-12-29	Added Workarounds 3 for E0005 ADC
1.0	2023-12-02	Initial version

Table 1: Errata Table

Erratum ID	Erratum Title
E010002	[DMA] DMA REQEN register modify write may interfere with other channel(s)
E120001	[SCU] Very low probability of PLL lock failure when PLL REFDIV = 1
E120002	[SCU] Writing 1 to the OUTRNG flag bit fails to clear this bit.
E220004.C	[EFM] OTA fail if MCU power off when Boot-Swap
E300001	[GPIO] An unexpected interrupt will be generated when using filter after GPIO is configured as low-level or falling edge triggering
E300002	[GPIO] Write GPIO registers may clear interrupt flags
E300004	[GPIO] Slow transition edges (both rising/falling) on GPIO inputs can induce spurious interrupts
E401001.D	[FlexCAN] Enhanced Rx FIFO may lose a frame when a message buffer C/S write occurs or the message buffer is locked
E401002	[FlexCAN] Cannot use MB0-MB7 for reception when Enhanced Rx FIFO is enabled
E403001	[SPI] RXFIFO overflow flag logic error
E403002	[SPI] TXCFG register read value keep old after writing
E403003	[SPI] In SPI master mode, enabling both PCS continuous mode and TX mask simultaneously will result in con tinuous clock output
E406001	[LINFlexD] UARTSR[SDF] flag can not be cleared
E406002	[LINFlexD] LINFlexD UART mode may send error data
E407001	[I2C] In I2C slave mode, turning on tx stall may cause I2C master arbitration to fail
E410001	[QSPI] LUT pointer jump error when fault occurs
E410002	[QSPI] TXFIFO underflow flag error set when TXFIFO is not empty during device accessing
E410003	[QSPI] QSPI output 0 when TXFIFO is empty
E410004	[QSPI] Sample data error under parallel mode
E502001	[LPTMR] When polling to read CCF, there is a small probability of reading incorrect flag information
E503003	[eTMR] The channel 2-7 DMA requests of eTMR1 and eTMR2 will not be cleared by DMA done
E503005	[eTMR] The channel flag will be set if channel capture event occurs before counter enable
E503006	[eTMR] QD mode enters one more downward overflow interrupt
E505001	[MPWM] When channel DMA mode is enabled, channel overflow interrupt can not be generated
E505002	[MPWM] MPWM won't generate DMA request or trigger when PWM duty set to 100%
E505003	[MPWM] When using software loading, the PWM may be inaccurate by one period.
E505004	[MPWM] MPWM configured in capture mode and capture value over-write is disabled. DMA/interrupt request lo gic and CHxF flag clear logic result in self-locking.
E509001	[RTC] Alarm/Seconds/Overflow flag will not be set again after write 1 clear
E600002	[ADC] One sequence which includes low ADC channels and high ADC channels can lead to inaccurate results
E600003	[ADC] Only one external trigger will also cause trigger loss error interrupt
E600005	[ADC] When the ADC bus clock frequency differs greatly from the function clock frequency, the problem of multiple interrupts may occur
E600007	[ADC] TMU Trigger Source ADC0_COCO for Monitoring ADC Conversion Status
E702003	[HCU] Engines can't read input fifo data when output fifo is full
E721001	[FMU] FOSU timeout counter restarts when a new fault detected during fault output in bi-stable mode

E010002: [DMA] DMA REQEN register modify write may interfere with other channel(s)

Description 描述

When user adds new channel request, in order not to lose original channel request, normally first read REQEN[CH], and then OR with the new channel, finally write to REQEN register. However, one case is that the original channel transfer has completed in the period of reading and writing REQEN regisetr, thus, the original channel request will be enabled again, an unexpected transfer will be triggered.

当用户添加新的通道传输请求时，为了不丢失原有通道的传输请求，一般会先读取 REQEN[CH]，再或上新加的通道，再写入 REQEN[CH]。然而有一种情况是在读取之后，写入之前的这段时间原有通道的传输正好完成，这样就又会使能原有通道的传输请求，那么就会触发意外的 DMA 传输。

Workaround 规避方案

User needs to pause the original channel transfer before operating REQEN register, at this time, it is necessary to wait until the channel active is low (i.e., no dma channel is busy), and then execute the BSET operation on the REQEN register. After the operation is completed, clears PAUSE.

用户在对 REQEN 寄存器位操作前需置位 CTRL[PAUSE] 来暂停原有通道的传输，此时需要等到通道 active 为低（即没有 DMA 通道忙），然后再进行对 REQEN 寄存器进行位操作。操作完成之后清除 CTRL[PAUSE]。

E120001: [SCU] Very low probability of PLL lock failure when PLL REFDIV = 1

Description 描述

When the reference clock of PLL is divided by 2(SCU register PLL_CTRL[REFDIV] = 1), there is a very low probability that the PLL will fail to lock when repeatedly switched. However, when the divider value is configured to other values, the PLL can lock normally.

当 PLL 的参考时钟被配置为 2 分频时（SCU 寄存器 PLL_CTRL[REFDIV]=1），反复开关 PLL 时有极低概率会出现无法锁定的状况。而当分频配置为其他值时，PLL 则都能够正常锁定。

Workaround 1 规避方案 1

Avoid configuring the REFDIV field of SCU register PLL_CTRL as 1. Other divider value can be used instead.

避免将 SCU 寄存器 PLL_CTRL 的 REFDIV 域配置为 1，可以采用其它分频值。

Workaround 2 规避方案 2

If REFDIV=1 must be used, after the PLL lock timeout, turn off the PLL and then turn it on again.

如果必须使用 REFDIV=1 时，在 PLL 锁定超时之后，关闭 PLL，然后重新开启 PLL。

E120002: [SCU] Writing 1 to the OUTRNG flag bit fails to clear this bit.

Description 描述

When OUTRNG flag bit is set, writing 1 to the OUTRNG flag bit fails to clear this bit.

当 OUTRNG 标志位置位后，对 OUTRNG 标志位写入 1 的操作无法清除该位。

Workaround 规避方案

Repeat the following sequence until the OUTRNG flag bit reads 0:

1. Write 1 to the OUTRNG bit;
2. Read back the OUTRNG bit value;
3. If the bit is not cleared, repeat the operation.

执行以下操作直至 OUTRNG 位被成功清除：

1. 向 OUTRNG 位写入 1；
2. 读取 OUTRNG 位的值；
3. 若该位不为 0，则重复上述操作。

E220004.C: [EFM] OTA fail if MCU power off when Boot-Swap

Description 描述

EFM supports on-the-air (OTA) updates of the application program stored in PFlash memory. In OTA mode, the PFlash is evenly divided into two parts: Side A (with lower logical addresses) and Side B (with higher logical addresses). The MCU executes the application from Side A.

After MCU reset, the EFM hardware initialization process reads the first three addresses of the BOOT_NVR sector. If at least two of the read data equal SWAP_TAG (0x5A5A5A5A5A5A5A5A), the hardware sets STS[BOOT_INFO] to 1, indicating PFlash logical address remapping mode. Otherwise, STS[BOOT_INFO] maintains its reset value of 0.

EFM 支持在线更新 PFlash 中的应用程序，称之为 OTA(On-The-Air) 模式。在 OTA 模式下，PFlash 存储空间等分成两份，分别为 A 面（PFlash 低逻辑地址）和 B 面（PFlash 高逻辑地址）。MCU 从 A 面执行应用程序。

MCU 复位后的 EFM 硬件初始化过程会读 BOOT_NVR 扇区的前三个地址，如果读回的数据有至少两个等于 SWAP_TAG（0x5A5A5A5A5A5A5A5A），硬件会将 STS[BOOT_INFO] 置 1，表明 PFlash 逻辑地址重映射模式。否则 STS[BOOT_INFO] 保持复位值 0。

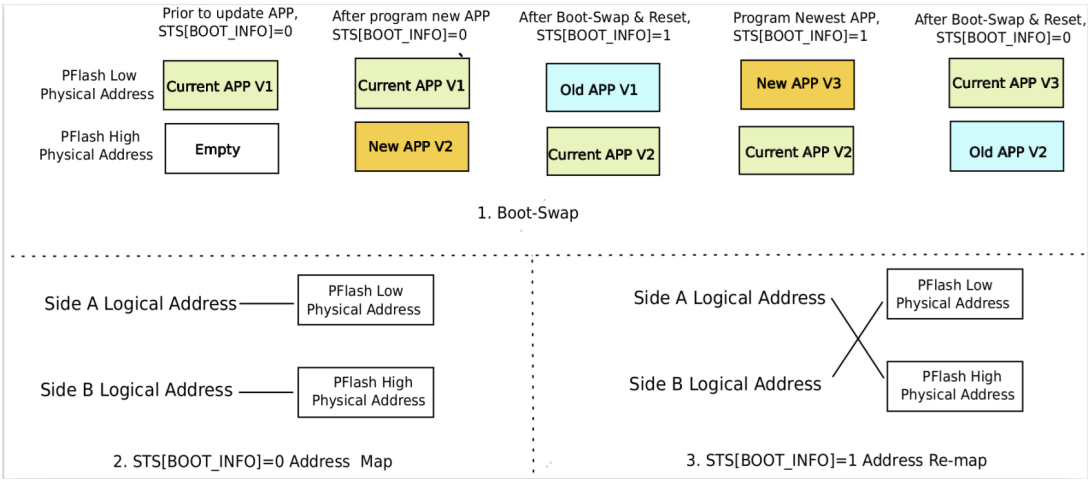

Figure 1: EFM update application code On-The-Air

If a new version of the application is detected, the software needs to update the application code using the following steps:

1. Erase PFlash Side B.
2. Program the new version of the application to Side B.
3. Execute the EFM BOOT_SWAP command.
4. Reset the MCU. At this point, the physical Flash arrays corresponding to the A/B Side logical addresses are swapped, and the Flash array containing the newest version of the application code is remapped to Side A.

如果检测到新版本应用程序，软件需要用以下步骤更新应用程序：

1. 擦除 PFlash B 面。
2. 将新版本应用程序烧写到 B 面。
3. 执行 EFM BOOT_SWAP 命令。
4. 复位 MCU，此时 A/B 面逻辑地址对应的 Flash 物理模块交换，新版本应用程序所在 Flash 模块重映射到了 A 面地址。

Analysis of EFM hardware operation failure when “Execute the EFM BOOT_SWAP command”:

- When EFM receives a BOOT_SWAP command with STS[BOOT_INFO]=0, it programs SWAP_TAG (0x5A5A5A5A5A5A5A5A) to the first three addresses of the BOOT_NVR sector in sequence. If the MCU power off while programming SWAP_TAGs, the number of successfully written SWAP_TAGs may be less than two. After MCU reset, STS[BOOT_INFO] remains 0, logical address remapping failed, the Flash physical array containing the new version application is still mapps to Side B, and the MCU executes the old version application from Side A by default. This application code update flow failed. Even if the update process is restarted from steps 1 to 4, when it reaches step 3 to program SWAP_TAG, the first three addresses of the BOOT_NVR sector are not empty, and the hardware will refuse to execute the Flash program operation, so the MCU cannot update the application in PFlash anymore.
- When EFM receives a BOOT_SWAP command with STS[BOOT_INFO]=1, it erases the BOOT_NVR sector. If the MCU power off during the erasure of the BOOT_NVR sector, the first three addresses of BOOT_NVR may not be completely erased, yet at least two addresses no longer equal SWAP_TAG. After MCU reset, STS[BOOT_INFO]=0, the Flash physical array containing the newest version application is mapped to Side A, and the MCU executes the application from Side A by default. This application code version updates successfully. However, the next time a new version of the application is detected, the update process needs to program SWAP_TAGs, it will find that the first three addresses of the BOOT_NVR sector are not empty, and the hardware will refuse to execute the Flash program operation, so the MCU cannot update the application in PFlash anymore.

其中 “执行 EFM BOOT_SWAP 命令” 时 EFM 硬件操作功能失效分析：

- EFM 在 STS[BOOT_INFO]=0 时收到 BOOT_SWAP 命令，EFM 会向 BOOT_NVR 扇区的前三个地址依次烧写 SWAP_TAG（0x5A5A5A5A5A5A5A5A）. 如果烧写 SWAP_TAG 时芯片掉电，烧写成功的 SWAP_TAG 数可能少于两个，MCU 复位后 STS[BOOT_INFO]=0，逻辑地址重映射失败，新版本应用程序所在 Flash 物理模块还是映射到 B 面，MCU 默认从 A 面执行的还是旧版本应用程序，本次应用程序更新失败。就算重新发起第一步到第四步的流程尝试更新应用程序，到第三步烧写 SWAP_TAG 时，BOOT_NVR 扇区前三个地址不为空，硬件会拒绝执行烧写 Flash 操作，所以 MCU 无法再更新 PFlash 中的应用程序。
- EFM 在 STS[BOOT_INFO]=1 时收到 BOOT_SWAP 命令，EFM 会擦除 BOOT_NVR 扇区。如果擦除 BOOT_NVR 扇区时芯片掉电，BOOT_NVR 为首三个地址没有擦除干净，却至少两个地址的数据不再等于 SWAP_TAG，MCU 复位后 STS[BOOT_INFO]=0，新版本应用程序所在 Flash 物理模块映射到了 A 面，MCU 默认从 A 面执行应用程序，本次应用程序更新是成功了。但是下一次检测到新版

本的应用程序时，程序更新过程需要烧写 SWAP_TAG，届时会发现 BOOT_NVR 扇区前三个地址不为空，硬件会拒绝执行烧写 Flash 操作，所以 MCU 无法再更新 PFlash 中的应用程序。

Workaround 规避方案

1. Erase PFlash Side B.
2. Program the new version of the application to Side B.
3. Execute the EFM BOOT_SWAP command.
    1. In the execution of BOOT_SWAP command, wait for EFM STS[DONE] to be set. Check the EFM STS[FAIL] register. If STS[FAIL]=1, the software needs to perform the following operations:
    2. First, write 0x484130 to the EFM register FAC_KEY (EFM offset address 0x304), and read back FAC_KEY equals 0x80000000.
    3. Use the ERASE_SECTOR command to erase the BOOT_NVR sector (starting address 0x10003800, sector size 2KB).
    4. Read EFM STS[BOOT_INFO], if STS[BOOT_INFO]=0, execute the BOOT_SWAP command again.
    5. Write 0x0 to the FAC_KEY register, and read back FAC_KEY equals 0.
4. Reset the MCU, at this point, the physical Flash modules corresponding to the A/B Side logical addresses are swapped, and the Flash module containing the new version of the application is remapped to Side A.

1. 擦除 PFlash B 面。
2. 将新版本应用程序烧写到 B 面。
3. 执行 EFM BOOT_SWAP 命令。
    1. 在执行 BOOT_SWAP 命令后等待 EFM STS[DONE] 置位后，检查 EFM STS[FAIL] 寄存器，如果 STS[FAIL]=1，软件需要执行以下操作:
    2. 软件先向 EFM 寄存器 FAC_KEY（EFM 偏移地址 0x304）写入 0x484130，回读 FAC_KEY 等于 0x80000000。
    3. 用 ERASE_SECTOR 命令擦除 BOOT_NVR 扇区（起始地址 0x10003800，sector 大小 2KB）。
    4. 再次执行 BOOT_SWAP 命令。
    5. 向寄存器 FAC_KEY 写 0x0，读回 FAC_KEY 等于 0。
4. 复位 MCU，此时 A/B 面逻辑地址对应的 Flash 物理模块交换，新版本应用程序所在 Flash 模块重映射到了 A 面地址。

E300001: [GPIO] An unexpected interrupt will be generated when using filter after GPIO is configured as low-level or falling edge triggering

Description 描述

When the GPIO module is configured as input mode with the filter, and interrupt trigger condition is low-level or falling edge, an unexpected interrupt will be generated.

当 GPIO 配置成带有过滤器的输入模式，中断触发为低电平或者下降沿触发时，此时会发生意外中断。

Workaround 1 规避方案 1

A possible approach is to first configure the GPIO Filter, then clear the flag, and finally set up the GPIO interrupt. The prerequisite for this method to work is ensuring that the internal state of the GPIO Filter is completely initialized before clearing the flag, meaning that it has already sent a low-level pulse to the internal GPIO pin, and the flag has been set. The initialization of the GPIO Filter's internal state can be accelerated by providing a higher clock to the GPIO Filter (the IPC Function Clock of the GPIO). After the GPIO Filter is initialized, you can choose the GPIO Filter Clock Source according to your needs.

可以先配置 GPIO 过滤器，再清除标志位，最后配置 GPIO 的中断。这种方法能修复的前提是在清除标志位之前要保证 GPIO 过滤器内部状态初始化完成，已经发送了低电平脉冲给 GPIO 内部引脚，且 flag 已经被置起。可以通过给 GPIO 过滤器较高的 clock(GPIO 的功能时钟) 来加快过滤器内部状态初始化。GPIO 过滤器初始化完成后可以再根据需求选择过滤器的功能时钟源。

GPIO Filter(T_{Filter})can be determined by multiplying the period of Filter's functional clock($T_{FilterClkSrc}$) by the Digital Filter width(N_{DFW}) configured in GPIO_PCR[DFW]. The calculation formula is as follows:

GPIO 过滤器的过滤周期（T_{filter}）可以由过滤器功能时钟源的周期（$T_{FilterClkSrc}$）乘以宽度 GPIO_PCR[DFW]（N_{DFW}）得出，计算公式如下:

$$T_{Filter} = T_{FilterClkSrc} * N_{DFW}$$

Workaround 2 规避方案 2

Disable the GPIO Filter function.

关闭 GPIO Filter 功能。

E300002: [GPIO] Write GPIO registers may clear interrupt flags

Description 描述

When a write operation is performed on a set of GPIO module registers, the write operation also affects the interrupt flag register of the GPIO. This is equivalent to simultaneously performing a write operation on the interrupt flag register of the GPIO.

当对一组 GPIO 模块寄存器进行写操作时，该写操作会同时作用到 GPIO 的中断标志寄存器。行为上相当于同时对 GPIO 中断标志寄存器同时进行了一次写操作。

Reading from the GPIO register has no additional impact.

对于 GPIO 寄存器的读操作并不会有额外的影响。

Example 示例

when GPIOA_PIFR=0x00010001, writing 0x00000001 to GPIOA_PSOR, the PIFR register will become 0x00010000, that is, the lowest bit flag is cleared by mistake, but the 16th bit Flag has no effect because the written data corresponds to the bit 0. Writing to any register in another group, such as GPIOB, will not affect the state of GPIOA.

当 GPIOA_PIFR=0x00010001 时，对 GPIOA_PSOR 写入 0x00000001 时，PIFR 寄存器会变成 0x00010000，也就是最低位的 flag 被错误的清零了，但是第 16 位 Flag 因为写入数据对应比特位 0，则无影响。写入其它组任意寄存器，例如 GPIOB，并不会影响 GPIOA 的状态。

Workaround 规避方案

Before writing to the GPIO register, read the interrupt register status. For GPIO output operations, use the PSOR and PCOR registers for bitwise operations. When operating the PDOR, ensure that the corresponding bit of the GPIO input status pin is written to 0.

写入 GPIO 寄存器前先读取中断寄存器状态，对于 GPIO 输出操作采用 PSOR 和 PCOR 寄存器按位操作，操作 PDOR 时确保 GPIO 输入状态引脚对应比特写 0。

E300004: [GPIO] Slow transition edges (both rising/falling) on GPIO inputs can induce spurious interrupts

Description 描述

Slow transition edges (both rising/falling) on GPIO inputs can induce spurious interrupts.

当 GPIO 输入端的上升沿和下降沿过渡速度过缓时，可能引发伪中断。

Workaround 1 规避方案 1

Enable PCTRL digital Filter.

打开引脚数字滤波器功能。

Workaround 2 规避方案 2

The rise/fall input transition time of GPIO must be ensured to be less than 5μs.

GPIO 的上升/下降输入过渡时间必须保证小于 5 微秒。

E401001.D: [FlexCAN] Enhanced Rx FIFO may lose a frame when a message buffer C/S write occurs or the message buffer is locked

Description 描述

When the enhanced Rx FIFO is enabled and mailbox n is enabled (n is the index of the mailbox), an incoming frame that should be stored in the Enhanced Rx FIFO is in instead dropped if either of the following conditions are true:

1. Mailbox n is used as a receive, and during the period when the mailbox is locked, a message with an enhanced Rx FIFO match is received.
2. Mailbox n is used as a transmit, and a message with an enhanced Rx FIFO match is received while writing the C/S configuration of the mailbox n.

如果使能了增强型接收 FIFO 并且启用了邮箱 n（n 为邮箱的索引）, 当满足以下条件之一，增强型接收 FIFO 都有可能会出现丢包：

1. 邮箱 n 用作接收，在锁定邮箱的期间，接收到有增强型接收 FIFO 匹配的报文。
2. 邮箱 n 用作发送，在写邮箱 n 的 C/S 配置时，接收到有增强型接收 FIFO 匹配的报文。

Workaround 1 规避方案 1

Avoid affected mailboxes:

- CAN0/1: n = 6, 8, 16, 18, 26, 28, 34, 36, 38, 44, 46, 48, 54, 56, 58, 64, 66, 68, 74, 76, 78, 84, 86, 88, 94, 96, 104, 106, 114, 116, 124, 126
- CAN2: n = 0, 2, 4, 10, 12, 14, 20, 22, 24, 30, 32, 40, 42, 50, 52, 60, 62
- CAN3/4: n = 0, 2, 10, 12, 20, 22, 30, 32, 40, 50, 60

避免使用受影响的邮箱：

- CAN0/1：n = 6，8，16，18，26，28，34，36，38，44，46，48，54，56，58，64，66，68，74，76，78，84，86，88，94，96，104，106，114，116，124，126
- CAN2：n = 0，2，4，10，12，14，20，22，24，30，32，40，42，50，52，60，62
- CAN3/4：n = 0，2，10，12，20，22，30，32，40，50，60

Workaround 2 规避方案 2

Do not use the Enhanced Rx FIFO.

不使用增强型 Rx FIFO。

E401002: [FlexCAN] Cannot use MB0-MB7 for reception when Enhanced Rx FIFO is enabled

Description 描述

A message received into a receive message buffer may be lost without indication when the following sequence of conditions occurs:

1. The Enhanced Rx FIFO is enabled.
2. A message is received into any of MB0-MB7 and is not immediately read by the application.
3. The application services one or more messages in the Enhanced Rx FIFO, thereby clearing the CAN_ERFSR.ERFDA bit one or more times.
4. Another message is received into a full and unserviced receive message buffer in the range MB0-MB7.

In such a case, the CODE field of the affected message buffer should change from FULL to OVERRUN. However, depending on the message buffer number and the number of times CAN_ERFSR.ERFDA was cleared, the CODE field of the affected receive message buffer may remain FULL. Therefore, the application is unaware that a received message was overwritten and lost.

当以下条件序列发生时，接收到接收消息缓冲区的消息会在没有提示下丢失：

1. 增强型 Rx FIFO 使能。
2. 消息接收进入 MB0 到 MB7 中的任何一个，并且没有被立即读取。
3. 应用程序取走增强型 Rx FIFO 中的一条或多条消息，从而清除 CAN_ERFSR.ERFDA 位一次或多次。
4. 另外一条消息被接收到满的且未读走的 MB0-MB7 中的一个

在这种情况下，受影响的 MB 的 CODE 域应当从 FULL 变成 OVERRUN。然而根据 MB 号和 CAN_ERFSR.ERFDA 清零的次数，受影响的接收消息缓冲区会保持 FULL 状态。因此应用程序就不知道接收到的消息被覆盖丢失了。

Workaround 1 规避方案 1

Do not use MB0-MB7 when Enhanced Rx FIFO is enabled.

当使能增强型 Rx FIFO 时不使用 MB0 到 MB7。

Workaround 2 规避方案 2

If you do use MB0-MB7 when the Enhanced Rx FIFO is enabled, be sure to service any pending received message buffer interrupts for MB0-MB7, including reading message data, before servicing messages received in the Enhanced Rx FIFO.

如果在使能增强型 Rx FIFO 时要使用 MB0 到 MB7，那么确保在读取接收到增强型 Rx FIFO 的消息之前处理任何挂起的 MB0-7 的接收消息中断，包括读取消息数据。

E403001: [SPI] RXFIFO overflow flag logic error

Description 描述

When the FIFO depth is N, SPI will only set the overflow flag of the RXFIFO after receiving N+2 data.

当 FIFO 深度为 N 时，SPI 在接收 N+2 个数据之后接收 FIFO 的溢出标志才会置起。

Workaround 1 规避方案 1

Don’t use receive FIFO overflow flag and its interrupt.

不要使用接收 FIFO 溢出标志及其中断。

Workaround 2 规避方案 2

Users should follow this rule when using receive FIFO overflow flag and its interrupt.

用户按照这一规则去使用接收 FIFO 溢出标志及其中断。

E403002: [SPI] TXCFG register read value keep old after writing

Description 描述

When changing the TXCFG configuration, immediately reading back will return the old value and it won't be updated even if read continuously for a long time.

当改变 TXCFG 配置后，立即回读会发现读到的还是旧值，一直读也不会更新。

Workaround 规避方案

After writing the TXCFG register, wait for data synchronization(around 3 functional clock cycles), and then read any other register once before reading back the TXCFG register.

写完 TXCFG 寄存器之后，延时等待数据同步（约 3 个功能时钟周期）, 然后在回读 TXCFG 寄存器之前，先读一次其它任意寄存器。

E403003: [SPI] In SPI master mode, enabling both PCS continuous mode and TX mask simultaneously will result in continuous clock output

Description 描述

When the SPI module operates as master with both PCS continuous mode and TX Mask enabled, the master will continuously output clock until either PCS continuous mode or TX Mask (or both) is disabled, or until the RxFIFO is filled with data.

当 SPI 模块作为主机且同时使用连续模式和 Tx Mask 功能时，主机将一直输出 clock，直到关闭连续模式和 Tx Mask 功能中的一个或全部，或者 RxFIFO 被填满数据。

Workaround 1 规避方案 1

In SPI multi-line master mode, writing 0xFFFFFFFF to the DATA register can be used as an alternative to the TX Mask function.

SPI 主机在多线模式下，往 DATA 寄存器写入 0xFFFFFFFF 方式替代 Tx Mask 功能。

Workaround 2 规避方案 2

In SPI single-line or multi-line master mode, GPIO can be used as an alternative to the PCS continuous function.

SPI 主机在多线或者单线模式下，使用 GPIO 控制 PCS 来替代连续模式。

E406001: [LINFlexD] UARTSR[SDF] flag can not be cleared

Description 描述

UARTSR[SDF] flag bit can not be cleared during transfer process.

UARTSR[SDF] 标志位在传输过程中无法被清除。

Workaround 规避方案

It is not possible to clear the SDF during the transfer, it is recommended to clear it after the transfer is complete.

在传输过程中无法清除 SDF，建议在传输完成之后再清除。如果在传输过程中使能了 LINIER[SDIE]，那么进入中断后要把 LINIER[SDIE] 清除。

E406002: [LINFlexD] LINFlexD UART mode may send error data

Description 描述

In LINFlexD UART mode, when using DMA for data transmission, if the LINFlexD module clock and the core clock are not sourced from the same source or the frequency is too high, the actual transmitted data may be inconsistent with the expected transmitted data.

LINFlexD UART 模式在使用 DMA 进行数据传输时，如果 LINFlexD 模块时钟与内核时钟（系统时钟）不同源或者频率过高，可能出现实际传输的数据与预期发送的数据不一致。

Workaround 1 规避方案 1

Switch the peripheral clock of the LINFlexD module to the fast bus clock and perform frequency division to make the module clock of LINFlexD lower than 25 MHz.

切换 LINFlexD 模块的外设时钟为快速总线时钟，并分频使得 LINFlexD 的模块时钟低于 25MHz。

Workaround 2 规避方案 2

Switch the peripheral clock and Core clock of the LINFlexD module to the same source, such as using PLL as the clock, and then perform reasonable frequency division to keep the clock frequency of the LINFlexD module lower than 25 MHz.

切换 LINFlexD 模块的外设时钟和系统时钟同源，比如都是用 PLL 作为时钟，然后进行合理的分频，保持 LINFlexD 模块时钟频率低于 25MHz。

Workaround 3 规避方案 3

Use interrupt mode instead of DMA mode to transfer data.

使用中断方式代替 DMA 方式传输数据。

E407001: [I2C] In I2C slave mode, turning on tx stall may cause I2C master arbitration to fail

Description 描述

When an I2C module operates as a slave and the TX stall function is enabled, if the master sends a read command and the slave hasn't had time to write data to the TX FIFO, the slave will hold both the SCL and SDA lines low. This state is maintained until the TX FIFO is populated with data. However, if the most significant bit of the data written to the TX FIFO is 1, SDA line will be pulled up, and SCL line will be released after one cycle of I2C peripheral clock, at this time SCL line also will be pulled up. If the rising edge of SDA line occurs during SCL line's high level, it will be considered as a stop signal, which will cause the arbitration failure.

当 I2C 模块作为从机且打开 Tx Stall 功能的时候，主机发送读的命令，从机未来得及往 Tx FIFO 里填写数据，此时从机会将 SCL 和 SDA 线拉低，一直到 Tx FIFO 中被填写数据；若填写的数据最高位为 1，则会将 SDA 线拉高，然后延迟一个 I2C 外设时钟释放 SCL 线，此时 SCL 线会被拉高，如果 SDA 线拉高时的上升沿出现在 SCL 线高电平的时候，那么硬件认为是终止信号，导致仲裁失败，如图所示。

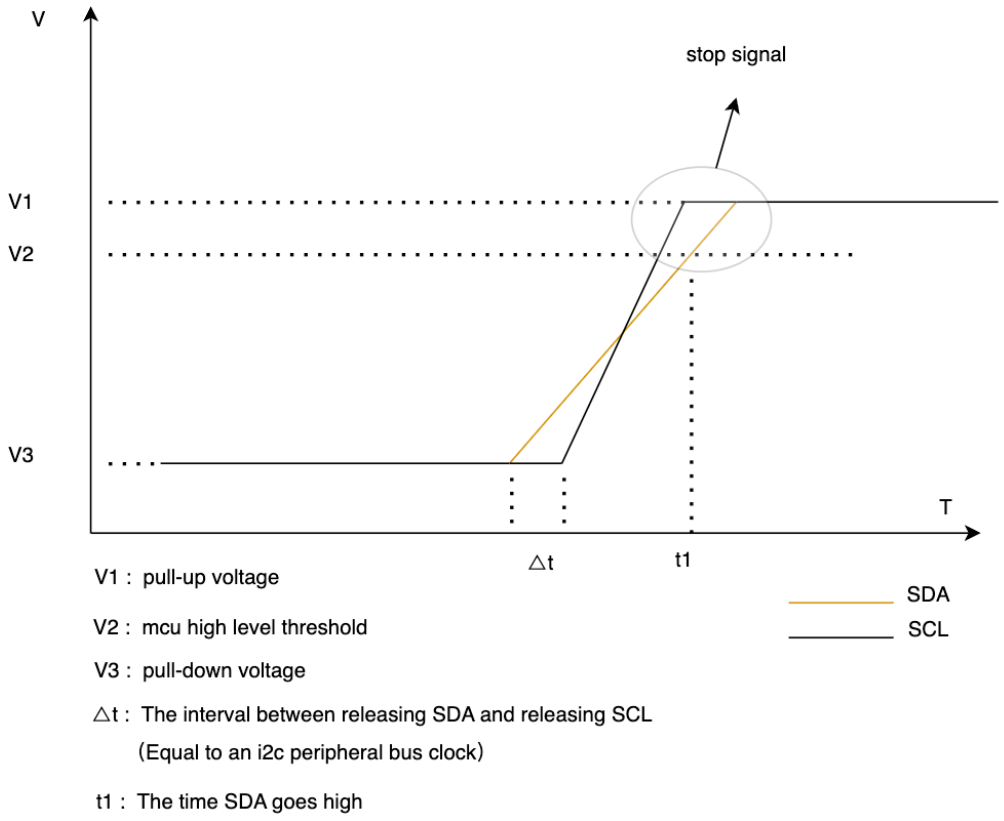

Figure 2: SDA SCL 时序说明

Workaround 1 规避方案 1

As an I2C slave, the module preloads data into the Tx Buffer before the master's read operation, thus preventing the Tx Stall feature from engaging and avoiding simultaneous

release of SCL and SDA from low states.

I2C 模块作为从机，在 I2C 主机进行读的操作的时候，从机提前将数据写入 Tx Buff，这样 Tx Stall 功能就不会生效，避免了 SCL 和 SDA 同时从低电平被释放。

Using this method requires the I2C to write to the Tx FIFO promptly, necessitating a higher priority for the I2C and ensuring it is not interrupted for extended periods.

用上面的方法可能需要 I2C 及时往 Tx FIFO 里写，这样就需要 I2C 有较高的优先级，且不能长时间被打断。

Workaround 2 规避方案 2

If the user still uses the Tx stall function, the highest bit of data sent is still 1, then the external circuit is required to satisfy the requirement that in standard mode (by adjusting the pull-up resistor R_p or the bus capacitor C_b) the SDA should be pulled high at least 250ns before SCL goes high, in fast mode the SDA should be pulled high at least 100ns before SCL goes high, and in upgraded fast mode the SDA should be pulled high at least 50ns before SCL goes high. SDA should be pulled high at least 50ns before SCL is high in fast-mode, and at least 50ns before SCL is high in fast-mode pulse.

若用户仍然使用 Tx stall 功能，此时发送数据的最高位仍然为 1，则需要外部电路满足在标准模式下（通过调整上拉电阻 Rp 或者总线电容 Cb）SDA 要在 SCL 为高前至少 250ns 拉高，在快速模式下 SDA 要在 SCL 为高前至少 100ns 拉高，在升级快速模式下 SDA 要在 SCL 为高前至少 50ns 拉高。

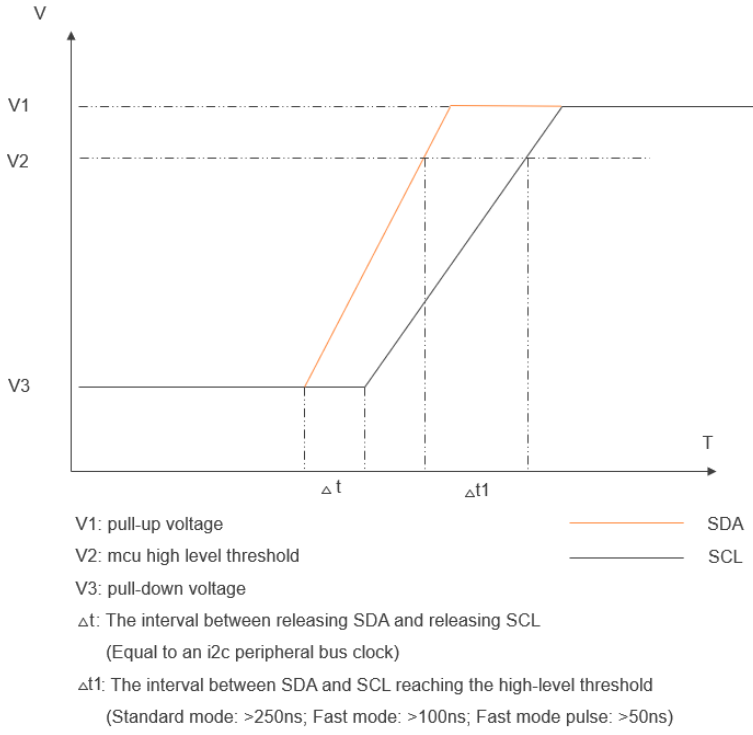

Figure 3: workaround 2

E410001: [QSPI] LUT pointer jump error when fault occurs

Description 描述

During the execution of the LUT in QSPI, if an illegal operation is detected, the pointer won't jump to the last one LUT, but to the third to last one. If the user has configured the third to last LUT, it will be executed, which may cause some unexpected result.

在 QSPI 执行 LUT 过程中，如果遇到非法操作，LUT 指针不会跳转到最后一个，而是跳转到倒数第三个，如果倒数第三个 LUT 用户有使用到，则会继续执行，可能会导致一些异常现象。

Workaround 规避方案

When detecting interrupts caused by illegal operations, users need to reconfigure the LUT and restart executing, discarding the execution result of the current instruction.

当遇到非法操作产生的中断时，用户需要重新配置 LUT 并从头开始执行，丢弃当前指令的执行结果。

E410002: [QSPI] TXFIFO underflow flag error set when TXFIFO is not empty during device accessing

Description 描述

During a QSPI communication process, when the number of data in TXFIFO doesn't meet the sending condition and the program instruction starts to be executed, TXFIFO will generate an underflow flag. Clear the flag and write data to TXFIFO until the sending condition is met. When pop data for transmission, a new underflow flag will be generated again, but the written data can be sent normally.

QSPI 一次通信过程中，当 TXFIFO 内数据个数不满足发送条件且开始执行发送指令时，TXFIFO 会产生一个下溢标志，清除标志并向 TXFIFO 写值直到满足发送条件，后续取数据发送时还会再次产生下溢标志，但是数据能正常取出并发送。

Workaround 1 规避方案 1

When an underflow error occurs, it indicates that invalid data has been written to external device. The user should terminate the operation and reprogram the address instead of continuing to write data to TXFIFO and maintain the communication.

当下溢错误发生时，说明已经有无效数据写入到外部存储中，用户应该终止本次操作并对该地址重新进行编程，而不是继续写数据来保持通信。

Workaround 2 规避方案 2

Do not use the underflow flag and its interrupt. If the user does not care about invalid data being written to external device, there is no need to worry about TXFIFO underflow.

不使用下溢标志及其中断，如果用户不在乎无效数据被写入外部存储中，则不用关心 TXFIFO 下溢。

E410003: [QSPI] QSPI output 0 when TXFIFO is empty

Description 描述

When TXFIFO is empty, if QSPI executes a write instruction, it will output a 128-bit value 0 before all 1.

当 TXFIFO 为空时，如果 QSPI 执行写指令，会在输出全 1 之前输出一个 128 位的 0 数据。

Workaround 规避方案

When an underflow error occurs, terminate the write operation and reprogram the address which have been programmed.

当产生下溢错误时，终止本次写操作，并对已编程的地址重新进行编程。

E410004: [QSPI] Sample data error under parallel mode

Description 描述

When QSPI works under parallel mode, because these two devices have slight differences in the data path delay, there is a low probability that the final sampled parallel data is error.

当 QSPI 使用并行模式通信时，由于两个存储设备的数据路径延时存在略微差异，会有较低几率导致最终采样的并行数据发生错位。

Workaround 规避方案

Before using parallel mode for communication, test whether there will be any data errors in parallel mode. If the sample data check errors, use DLL to adjust the clock sampling points to avoid data error gaps.

在使用并行模式进行通信前，先测试并行模式下数据是否会出错，如果数据出错，使用 DLL 调整时钟采样点来避开数据错位的间隙。

E502001: [LPTMR] When polling to read CCF, there is a small probability of reading incorrect flag information

Description 描述

When polling to read the CCF of LPTMR, there is a small probability of reading the wrong CCF. This situation occurs when the CCF is pulled up and the CCF is read exactly, and its probability of occurrence increases with the increase of the bus clock frequency/function clock frequency of LPTMR.

在轮询读取 LPTMR 的 CCF 时，有较小的概率会读到错误的 CCF，这种情况出现在 CCF 拉起时正好读 CCF，且它的发生概率随着 LPTMR 的总线时钟频率/功能时钟频率提高而提高。

Workaround 1 规避方案 1

After the interrupt is raised, reading the CCF again will not read the wrong CCF.

中断升起后再读 CCF 就不会读到错误的 CCF。

Workaround 2 规避方案 2

During polling, CCF can be read twice (to avoid reading metastable states). When the values of CCF read twice meet expectations, it is determined that CCF has risen.

在轮询时，可以读两次 CCF（避免读到亚稳态），当两次读的 CCF 的值都符合预期时，判定为 CCF 升起。

E503003: [eTMR] The channel 2-7 DMA requests of eTMR1 and eTMR2 will not be cleared by DMA done

Description 描述

Channel 2-7 of eTMR1 and eTMR2 will trigger DMA requests continuously since the DMA requests can not be cleared by DMA done. While all of channels of other eTMRs, channel 0-1 of eTMR1 and eTMR2 are normal.

eTMR1 和 eTMR2 的通道 2 到 7 会一直触发 DMA 请求，因为 DMA 传输完成后不会把 DMA 请求清掉。而其他 eTMR 的所有通道以及 eTMR1 和 eTMR2 的通道 0 和 1 都正常。

Workaround 规避方案

For problematic channels, user can use interrupt handler instead of DMA transfer.

对于有问题的通道，用户可以使用中断处理代替 DMA 传输。

E503005: [eTMR] The channel flag will be set if channel capture event occurs before counter enable

Description 描述

The channel flag will be set if channel capture event occurs before counter starts.

如果在计数器开启之前通道捕获事件发生，那么通道标志位会被置起。

So if interrupt or DMA enables before eTMR enables, it will cause the interrupt service routine or DMA request to respond immediately once the counter starts, and the first capture value is always 0.

所以如果中断或者 DMA 在 eTMR 使能之前使能了，这将导致一旦计数器开启中断处理或者 DMA 请求会立即响应，第一次的捕获值总是为 0.

Workaround 1 规避方案 1

Clear the channel flag before enabling the counter.

在开启计数器之前清掉通道事件的标志位。

Workaround 2 规避方案 2

Enable counter before enabling channel interrupt.

先使能 eTMR，打开计数器，再打开通道中断使能和中断服务。

E503006: [eTMR] QD mode enters one more downward overflow interrupt

Description 描述

When QD is enabled, if the counter starts counting down after the mode selection input, the counter does not really overflow downward, but it goes directly to the overflow interrupt.

当使能 QD 之后, 若模式选择输入后一开始便是向下计数此时计数器并未真正向下溢出但会直接进入溢出中断

Workaround 规避方案

If the user starts counting down at the beginning, it is recommended to change the polarity to count up at the beginning to avoid the problem of interrupting more than once at the beginning.

若用户在使用一开始就向下计数时, 建议改变极性使得一开始变为向上计数, 此时可避免一开始多进一次中断的问题。

E505001: [MPWM] When channel DMA mode is enabled, channel overflow interrupt can not be generated

Description 描述

When user enables the DMA mode of channel, the channel overflow interrupt will not be generated when counter overflow event occurs.

当用户使能了通道的 DMA 模式，该通道的计数器溢出中断将无法产生。

Workaround 1 规避方案 1

Replace the channel DMA mode with MPWM channel interrupt mode.

用 MPWM 通道的中断模式代替 DMA 模式。

Workaround 2 规避方案 2

Configure the prescaler of MPWM so that channel DMA transfer can finish in one period. In capture mode, the software can determine whether the counter overflows according to the capture value.

配置 MPWM 的分频器，使其能在一个周期内完成 DMA 的触发。在捕获模式下，可根据捕获值由软件判断计数器是否溢出。

Workaround 3 规避方案 3

Combine with other general purpose timers to complete the overflow interrupt count.

结合其他通用计时器完成溢出中断计数。

E505002: [MPWM] MPWM won't generate DMA request or trigger when PWM duty set to 100%

Description 描述

Without changing the polarity of MPWM.In PWM mode, when the PWM duty cycle is set to 100% (the CH_CMP value is 0), the MPWM cannot generate a DMA request or generate a hardware trigger to trigger other modules. For example, the MPWM module cannot trigger ADC conversion at this time. In Output Compare mode, when the compare value is set to 0 (the CH_CMP value is 0), the level on the channel will keep high or low, channel can not output normally. Moreover, at this time, the module will not generate a DMA request or a hardware trigger to trigger other modules.

在不改变 MPWM 的极性的情况下，PWM 模式下，当 PWM 占空比设置为 100%（CH_CMP 值为 0），MPWM 无法产生 DMA 请求或者产生硬件 Trigger 去触发其他模块，例如此时 MPWM 模块无法触发 ADC 转换。Output Compare 模式下，当设置输出比较值为 0(CH_CMP 值为 0)，通道上的电平会保持恒高或恒低，无法正确输出电平；而且此时无法产生 DMA 请求或者产生硬件 Trigger 去触发其他模块。

Root Cause 根本原因

The MPWM counter starts from 1 and counts up to the MOD, and the counter does not reach 0. And the conditions for MPWM to generate a DMA request or a hardware trigger in PWM mode and Output Comapre mode are when the counter value reaches CH_CMP. Therefore, when the CH_CMP value is 0, MPWM cannot generate a DMA request or a hardware trigger in PWM mode and Output Comapre mode.

MPWM 的计数器从 1 开始，一直计数至周期值 MOD，计数器不会出现 0。而 MPWM 在 PWM 模式与 Output Comapre 模式下产生 DMA 请求，或者产生硬件 Trigger 的条件是，计数器值计至 CH_CMP。所以当 CH_CMP 值为 0 时，MPWM 在 PWM 模式与 Output Comapre 模式下无法产生 DMA 请求与硬件 Trigger。

Workaround 1 规避方案 1

Avoid setting the CH_CMP value to 0.

避免 CH_CMP 值设置为 0。

Workaround 2 规避方案 2

Use another channel to trigger a DMA request or generate a hardware trigger, for example, when CH0 outputs 100% PWM, CH1 synchronously counts, and the CH_CMP value is set to 1 or the MOD value to trigger a DMA request or generate a hardware trigger.

使用另一个通道来触发 DMA 请求或产生硬件 trigger，例如当 CH0 输出 100%PWM 时，CH1 同步计数，但 CH_CMP 值设置为 1 或者 MOD 值，来触发 DMA 请求或产生硬件 trigger。

E505003: [MPWM] When using software loading, the PWM may be inaccurate by one period.

Description 描述

In PWM mode, when MPWM uses software trigger to load registers, if the newly loaded value is smaller than the current counter value at this time, then it will not update the PWM immediately, but wait for the overflow (0xffff) to update.

在 PWM 模式下,MPWM 使用软件触发加载寄存器时，如果此时新加载的值小于当前计数器的值，那么不会立即更新 PWM，而是等待溢出 (0xffff) 后更新。

Workaround 规避方案

Try not to use software to load registers, or if you use software to load the trigger need to ensure that the current load value is less than the current value of the counter, for example, after the TOF immediately after the load of the new value, the counter value is smaller, the specific use of the user needs to be evaluated.

尽量不选择使用软件来加载寄存器，或者如果使用软件加载触发器需要保证当前的加载时的值小于计数器当前值，例如在在发生 TOF 之后立即加载新的值，此时计数器的值较小，具体使用需用户评估。

E505004: [MPWM] MPWM configured in capture mode and capture value over-write is disabled. DMA/interrupt request logic and CHxF flag clear logic result in self-locking.

Description 描述

The CH_CAP[CAP] capture result of the MPWM must be read in order to start the next capture. Even if the flag STS1[CHxF] is cleared, it is necessary to wait for the reading of the CH_CAP[CAP] capture result before releasing the capture operation (capturing a new capture event). The clearing of the flag STS1[CHxF] needs to be completed through the W1C or DMA's done signal. The time difference between reading the CH_CAP[CAP] to release the capture operation and clearing the flag STS1[CHxF] leads to the self-locking phenomenon.

1. By using DMA to transfer the capture values, the general process is: capture event-1 (STS1[CHxF] is set, capture operation lock) -> DMA request -> CH_CAP[CAP] read operation (capture operation unlock) -> DMA returns the done signal (STS1[CHxF] is cleared to 0). If the capture operation is released between the "CH_CAP[CAP] read operation" and "DMA returns the done signal" and is in the unlock state, a new capture event-2 arrives, which will lock the capture operation and put it in the lock state. The corresponding "DMA returns the done signal" of the old capture event-1 will clear STS1[CHxF], and after clearing, capture event-2 will no longer generate DMA req, and thus there will be no CH_CAP[CAP] read operation. In this case, STS1[CHxF] is cleared (the STS1[CHxF] set from capture event-2), no new DMA req will be generated, there will be no CH_CAP[CAP] read operation, and the capture operation is locked and in the lock state, causing the self-locking phenomenon.

2. By using interrupts, the core reads the capture values, the general process is: capture event one (STS1[CHxF] is set, capture operation lock) -> interrupt service program -> CH_CAP[CAP] read operation (capture operation unlock) -> the core clears STS1[CHxF] through W1C. If the capture operation is released between the "CH_CAP[CAP] read operation" and "the kernel clears STS1[CHxF] through W1C" and is in the unlock state, a new capture event-2 arrives, which will lock the capture operation and put it in the lock state. The corresponding "the core clears STS1[CHxF] through W1C" of the old capture event-1 will clear STS1[CHxF], and after clearing, capture event-2 will no longer generate an interrupt, and thus there will be no CH_CAP[CAP] read operation. In this case, STS1[CHxF] is cleared (the STS1[CHxF] set from capture event-2), no new interrupt will be generated, there will be no CH_CAP[CAP] read operation, and the capture operation is locked and in the lock state, causing the self-locking phenomenon.

通过读取 MPWM 的 CH_CAP[CAP] 捕获结果才能开启下一次捕获，即使标志 STS1[CHxF] 被清除也需要等待读取 CH_CAP[CAP] 捕获结果才能释放捕获操作（即捕获新的捕获事件），标志 STS1[CHxF] 的清除需要通过 W1C 或者 DMA 的 done 信号完成，STS1[CHxF] 用来产生中断和 DMA req，读取 CH_CAP[CAP] 释放捕获操作和标志 STS1[CHxF] 的清除操作的时间差，导致自锁现象。

1. 通过 DMA 搬运捕获值，通常流程：捕获事件一（STS1[CHxF] 置起，捕获操作 lock）->DMA 请求->CH_CAP[CAP] 读操作（捕获操作 unlock）->DMA 返回 done

信号（STS1[CHxF] 被清 0）。假如 CH_CAP[CAP] 读操作和 DMA 返回 done 信号之间，捕获操作被释放处于 unlock 状态，新的捕获事件二到来，将使得捕获操作锁住并处于 lock 状态，旧的捕获事件一相应的 DMA 返回 done 信号将 STS1[CHxF] 清除，清除之后捕获事件二将不再产生 DMA req，进而无 CH_CAP[CAP] 读操作。这种情况下，STS1[CHxF] 被清除（含捕获事件二的 STS1[CHxF] 置位），将不再产生新的 DMA req，也无 CH_CAP[CAP] 读操作，且捕获操作锁住并处于 lock 状态，造成自锁现象。

2. 通过中断，内核执行读取捕获值，通常流程：捕获事件一（STS1[CHxF] 置起，捕获操作 lock）-> 中断服务程序->CH_CAP[CAP] 读操作（捕获操作 unlock）-> 内核通过 W1C 将 STS1[CHxF] 清 0。假如 CH_CAP[CAP] 读操作和内核通过 W1C 将 STS1[CHxF] 清 0 之间，捕获操作被释放处于 unlock 状态，新的捕获事件二到来，将使得捕获操作锁住并处于 lock 状态，旧的捕获事件一相应的内核通过 W1C 将 STS1[CHxF] 清 0，清除之后捕获事件二将不再产生中断，进而无 CH_CAP[CAP] 读操作。这种情况下，STS1[CHxF] 被清除（含捕获事件二的 STS1[CHxF] 置位），将不再产生新的中断，也无 CH_CAP[CAP] 读操作，且捕获操作锁住并处于 lock 状态，造成自锁现象。

Workaround 1 规避方案 1

By using interrupt service program, STS1[CHxF] can be cleared first, and then the CH_CAP[CAP] read operation can be executed, which can avoid the self-locking phenomenon.

通过使用中断服务程序中可以先清除 STS1[CHxF]，再执行 CH_CAP[CAP] 读操作，可以避免自锁现象。

Workaround 2 规避方案 2

The channel is configured as capture mode 3'b011, and the CHx_CFG[OVWR] is set to 1. There will be no self-locking phenomenon, and it can meet the functional requirements. When CHx_CFG[OVWR] is 0, the users cannot use DMA to transfer capture result.

通道配置为捕获模式 3b011，CHx_CFG[OVWR] 配置为 1，不会出现自锁现象，且能够满足功能需求。当 CHx_CFG[OVWR]=0 时，不能使用 DMA 搬取捕获结果。

E509001: [RTC] Alarm/Seconds/Overflow flag will not be set again after write 1 clear

Description 描述

After writing 1 to clear the alarm/seconds/overflow flag, the alarm/seconds/overflow flag will remain clear and will not be set again.

对 RTC 的 alarm/seconds/overflow 标志位进行写 1 清零后，RTC 的 alarm/seconds/overflow 标志位将会保持清除状态，且不会再次置起。

Workaround 规避方案

After clearing the alarm/seconds/overflow flag, execute a register access (either read or write). For example, change the operation of clearing the flag to: write 1 first and then write 0, or write 1 first and then read back.

在清除完 alarm/seconds/overflow 标志位后进行一次任意的寄存器访问（读或者写都可以）。例如，可以将清除标志位的操作改为：对标志位先写 1 再写 0 或者对标志位先写 1 再读一次。

E600002: [ADC] One sequence which includes low ADC channels and high ADC channels can lead to inaccurate results

Description 描述

The lower channels in this document are external channels 0 to 15, the upper channels are external channels 16 to 31, and the ADC internal channels（32 to 39）are also upper channels.

本文中的低通道是外部通道 0 至外部通道 15，高通道是外部通道 16 至外部通道 31，ADC 内部通道（32 to 39）也属于高通道。

When an ADC conversion sequence includes both the low 16 channels (ch0-ch15) and the high 24 channels (ch16-ch39), the conversion result of the last channel before each switch is abnormal.

当一个 ADC 转换序列中同时包含低 16 通道 (ch0-ch15) 与高 16 通道 (ch16-ch39)，每次切换前的最后一个通道转换结果异常。

In the channel sequence described in Example 1, Channel 16 belongs to the high 16 channels, while the rest belong to the low 16 channels. Consequently, when switching from the low 16 channels to the high 16 channels, the channel conversion result is abnormal (Channel 3). Similarly, when switching from the high 16 channels to the low 16 channels, the channel conversion result is abnormal (Channel 16).

在示例 1 中所描述的通道序列中，通道 16 属于高 16 通道，其余均为低 16 通道，则从低 16 通道切换至高 16 通道时的通道转换结果异常 (通道 3)，从高 16 通道切换至低 16 通道时的通道转换结果异常 (通道 16)。

In another special case, if the first and last channels of the sequence are not both in the lower 16 channels or the upper 16 channels at the same time, the conversion result of the last channel in the sequence is abnormal.

另外一种特殊情况，若序列的第一个通道与最后一个通道不同时为低 16 通道或高 16 通道，则该序列的最后一个通道转换结果异常。

In the situation described in Example 2, Channel 3, which is one of the lower 16 channels, switches to Channel 16, which is one of the upper 16 channels, resulting in an abnormal conversion result. The first channel in the sequence is Channel 0, which belongs to the lower 16 channels, and the last channel is Channel 19, which belongs to the upper 16 channels. Since they are not both low or high channels, the conversion result of Channel 19 is abnormal.

在示例 2 中所描述的通道序列中，作为低 16 通道之一的通道 3 切换至作为高 16 通道之一的通道 16，所以其转换结果异常；序列的第一个通道为通道 0，属于低 16 通道，序列的最后一个通道为通道 19，属于高 16 通道，两者不都是低通道或高通道，所以通道 19 的转换结果异常。

What's more, internal ADC channels belong to upper channels, like temperature sensor, VREFH and VREFL.

此外，内部通道也属于高通道，例如温感，正参考电压，负参考电压等等。

Example 示例

If the channel sequence is: 0, 1, 2, 3, 16, 4, 5, 6, then the conversion result of channel 3 and channel 16 will be abnormal.

若通道序列为：0，1，2，3，16，4，5，6，则通道 3 与通道 16 的转换结果异常。

If the channel sequence is: 0, 1, 2, 3, 16, 17, 18, 19, then the conversion result of channel 3 and channel 19 will be abnormal.

若通道序列为：0，1，2，3，16，17，18，19，则通道 3 与通道 19 的转化结果异常。

Root Cause 根本原因

When switching from the low channel to the high channel (from Channel 15 to Channel 16), after the sampling process of Channel 15 is completed, Channel 16 is enabled. At this point, the channel voltage of Channel 16 influences the reference voltage of Channel 15, resulting in abnormal conversion results for Channel 15.

在低通道切换至高通道的情况下 (从通道 15 到通道 16)，通道 15 采样完成后，通道 16 使能，此时通道 15 的参考电压被通道 16 的通道电压影响，导致通道 15 的转换结果异常。

Workaround 1 规避方案 1

In an ADC sequence, inserting a duplicated channel between high and low channel switches, and discarding the corresponding result, can be performed. For instance, if the channel sequence is: 0, 1, 2, 3, 16, 4; two duplicated channels can be inserted, resulting in the sequence: 0, 1, 2, 3, 3, 16, 16, 4; the results of the second conversion for channels 3 and 16 are discarded.

ADC 序列中，高低通道切换间插入一个重复通道，并丢弃该结果。例如，若通道序列为：0，1，2，3，16，4；则可插入两个重复通道，插入重复通道的序列为：0，1，2，3，3，16，16，4；丢弃第二次转换通道 3 与通道 16 的结果。

Workaround 2 规避方案 2

Mixing of high-pass and low-pass channels is eliminated, and two sequences are used for ADC conversion. One sequence consists entirely of high-pass channels, while the other consists entirely of low-pass channels. For instance, if the channel sequence is: 0, 1, 2, 3, 16, 17, 18, 19; it can be split into 2 sequences, each containing 4 channels. Sequence 1 would be: 0, 1, 2, 3; Sequence 2 would be: 16, 17, 18, 19.

取消高低通道的混用，用两个序列进行 ADC 转换，一个序列中全为高通道，一个序列中全为低通道。例如，若通道序列为：0，1，2，3，16，17，18，19；则可拆成 2 个序列，每个序列包含 4 个通道，序列 1 为：0，1，2，3；序列 2 为：16，17，18，19。

In the design of hardware circuit, attention should be paid to the problem of mixing high and low channels, and the ADC channels used should be low channels or high channels as far as possible. If it is necessary to mix high and low channels, the above workaround 1 can be used.

在硬件设计中，应注意混用高低通道的问题，ADC 通道应尽量选择低通道或高通道。如需混用高低通道，方案 1 可供参考。

E600003: [ADC] Only one external trigger will also cause trigger loss error interrupt

Description 描述

When using external triggers to trigger ADC conversions, normal triggering can also lead to misinterpretation as lost triggering; if the lost trigger interrupt is turned on, it can lead to false interrupt entry.

使用外部触发来触发 ADC 转换时，正常的触发也会导致误认为触发丢失；如果开启触发丢失中断则会导致误进入中断。

Workaround 规避方案

If a hardware trigger is used to convert the ADC, in continuous mode or single mode, the first trigger loss interrupt can be ignored and subsequent entries can be considered normal. In discontinuous mode, if a hardware trigger is used, the trigger error can be ignored if the trigger interval can be guaranteed to be greater than the actual conversion time of the ADC. Otherwise, it is recommended to use software trigger.

若使用硬件触发转换 ADC，在 Continuous 或者 Single 模式下，可将第一次触发丢失中断忽略，后续进入可认为正常。在 Discontinuous 模式下，如果使用硬件触发，若能保证触发间隔大于 ADC 实际转换时间，则可以忽略触发错误。否则建议使用软件触发。

E600005: [ADC] When the ADC bus clock frequency differs greatly from the function clock frequency, the problem of multiple interrupts may occur

Description 描述

When the ADC function clock and bus clock frequency gap is large (depending on the interrupt and handle the interrupt exit time is less than the bus clock and function clock multiples), if you open the corresponding interrupt, this time the interrupt flag bit is pulled high, and then enter the corresponding interrupt, the user in the exit of the interrupt will generally write 1 to clear the corresponding flag bit, and then at this time due to the difference between the frequency of the bus and the function clock frequency Then, due to the large difference between the bus frequency and the function clock frequency, it is not possible to synchronize the clearing to the function clock domain in time, so the flag bit is not effectively cleared, which leads to repeatedly entering the same interrupt after exiting the interrupt.

当 ADC 功能时钟与总线时钟频率差距较大时 (具体取决于进入中断且处理完中断退出时的时间是否小于总线时钟与功能时钟的倍数)，若开启相应中断，此时中断标志位拉高，然后进入相应中断，用户在退出中断时一般会写 1 清零来清除相应标志位，然后此时由于总线频率与功能时钟频率相差较大，未能及时同步清除到功能时钟域，所以标志位并未有效清除，从而导致在退出中断之后反复进入同一个中断。

Workaround 规避方案

If the user is not sure whether the processing time of each interrupt is less than the relationship between the bus clock and the function clock multiplier, it is recommended that the user check whether the corresponding flag bit is actually cleared when the interrupt service program exits, to prevent the same interrupt from repeatedly entering the problem.

若用户无法确定每次进中断处理的时间是否小于总线时钟与功能时钟倍数的关系，此时建议用户在中断服务程序退出时检查相应标志位是否确实被清除，防止同一中断反复进入的问题。

E600007: [ADC] TMU Trigger Source ADC0_COCO for Monitoring ADC Conversion Status

Description 描述

Upon completion of a channel conversion, the status flag STS[EOC] is set and can be cleared either by a write-1-to-clear (W1C) operation or by reading the conversion result register. The TMU trigger source ADC0_COCO is derived directly from the STS[EOC] status bit. When this trigger signal is output to a pad, it remains asserted at logic level 1 if no read operation or explicit W1C action is performed to clear the EOC status. This behavior prevents the trigger signal from being pulsed for each conversion completion, thereby hindering real-time monitoring of the ADC conversion process.

当通道转换完成后，状态标志位 STS[EOC] 会被置位，该标志可通过写 1 清除（W1C）操作或读取转换结果寄存器来清零。TMU 触发源 ADC0_COCO 直接源自 STS[EOC] 状态位。当该触发信号被输出到引脚时，如果未执行读取操作或显式的 W1C 操作来清除 EOC 状态，该信号将保持为逻辑电平 1。此行为导致触发信号无法在每次转换完成时产生脉冲，从而影响对 ADC 转换过程的实时监测。

Workaround 规避方案

To ensure the trigger signal behaves as a pulse for each conversion event, the STS[EOC] status bit must be cleared in a timely manner. This can be achieved by one of the following methods:

Performing a software write-1-to-clear (W1C) operation on the STS[EOC] bit, preferably within the ADC interrupt service routine (ISR) if interrupts are used. Initiating a DMA read operation to the conversion result register, which automatically clears the EOC status.

为确保触发信号在每次转换事件时表现为脉冲，必须及时清除 STS[EOC] 状态位。可采用以下方法之一：

在软件中执行写 1 清除（W1C）操作清除 STS[EOC] 位（若使用中断，建议在 ADC 中断服务例程中完成）；通过 DMA 对转换结果寄存器的读取操作自动清除 EOC 状态；

E702003: [HCU] Engines can’t read input fifo data when output fifo is full

Description 描述

When the output fifo is full, if the input fifo is not empty and any alogrithm engine is reading data from the input fifo, the engine will get wrong data and no available read request send to the input fifo. The use of AES/SM4 algorithm engine may be affected.

当输出 fifo 满时，如果此时输入 fifo 有数据且算法引擎要从输入 fifo 取数据，算法引擎会得到错误值且输入 fifo 不会被读。使用 AES/SM4 算法引擎时可能会受到影响。

Workaround 规避方案

When using AES/SM4 algorithm engine, configure the output fifo watermark less than the fifo depth, use the watermark flag to carry data from the output fifo, avoid the case that the output fifo is full.

在使用 AES/SM4 算法引擎时，配置输出 fifo 的 watermark 小于 fifo 深度，并根据 watermark 的标志来读取输出 fifo 数据，避免输出 fifo 满的情形出现。

E721001: [FMU] FOSU timeout counter restarts when a new fault detected during fault output in bi-stable mode

Description 描述

When the FOUT configuration of FMU is set to Bi-stable mode and a fault output indication has already started, if a new fault is detected and only it's fault output is enabled, the FOSU timeout counter will restart counting. Since the fault output is still reflecting the previous fault, there will be no fault output responding to the new fault, which may cause the FOSU timeout counter to fill up and reset.

当 FMU 的 FOUT 配置为 Bi-stable 模式且已经开始有故障输出时，如果再检测到一个新故障，且该故障只使能了故障输出，则 FOSU 的超时计数器会重新开始计数，由于故障输出还在反映前一次的故障，所以不会有故障输出响应新故障，有概率导致 FOSU 超时计数器计满并产生复位。

Workaround 规避方案

The response of the fault source should not only configure the fault output, but also enable interrupts, interrupt response can also turn off the FOUS timeout counter.

故障源的响应不要只配置故障输出，还要同时配置中断，中断响应也能关闭 FOUS 超时计数器。

Copyright and Contact

How to reach us:
Home Page: www.ytmicro.com