

ESP32

勘误表及解决办法



版本 1.8
乐鑫信息科技
版权 © 2018

关于本文档

本文收录了 ESP32 芯片的硬件问题并给出解决方法。

发布说明

日期	版本	发布说明
2016-11	V1.0	首次发布。
2016-12	V1.1	修订章节 3.2 中 MEMW 指令。
2017-04	V1.2	修改章节 3.1 的描述； 增加章节 3.8。
2017-06	V1.3	增加章节 3.9、3.10。
2018-02	V1.4	修正章节 3.3 中前五个寄存器的名称里的笔误。
2018-02	V1.5	增加章节 3.11。
2018-05	V1.6	整体更新。
2018-05	V1.7	增加章节 3.12。
2018-12	V1.8	增加章节 3.13: ESP32 CAN 相关问题。

文档变更通知

用户可通过[乐鑫官网](#)订阅技术文档变更的电子邮件通知。

证书下载

用户可通过[乐鑫官网](#)下载产品证书。

目录

1. 芯片修订	1
2. ESP32 勘误表	2
3. 问题描述和解决方法.....	4
3.1. 芯片上电或 Deep-sleep 醒来后，会随机发生一次看门狗复位。	4
3.2. CPU 使用 cache 访问外部 SRAM 时，特定条件下会发生读写错误。	4
3.3. CPU 访问外设时，如果连续不间断地写同一个地址，会出现数据丢失的现象。	5
3.4. Brown-out Reset（欠压复位）功能在当前版本无法工作，复位之后芯片无法起来。	6
3.5. CPU 频率从 240 MHz 直接切换到 80/160 MHz 会卡死。	6
3.6. 同时有 GPIO 和 RTC_GPIO 功能的 pad 的上拉下拉电阻只能由 RTC_GPIO 的上拉下拉寄存器控制。	6
3.7. Audio PLL 使用频率有限制。	7
3.8. 由于 flash 启动的速度慢于芯片读取 flash 的速度，芯片上电或 Deep-sleep 醒来后，会随机发生一次看门狗复位。	7
3.9. CPU 在访问外部 SRAM 时会小概率发生读写错误。	8
3.10. 双核 CPU 在读不同地址空间时可能会发生错误。	8
3.11. 当一些 RTC 外设的电源打开时，GPIO36 和 GPIO39 的数字输入会被拉低约 80 ns。	9
3.12. LEDC 递减渐变，duty 值溢出错误。	9
3.13. ESP32 CAN 相关问题.....	10
3.13.1. 处于复位模式或总线关闭恢复状态时的接收错误计数器 (REC) 数值仍会变化。	10
3.13.2. 总线关闭恢复期间，错误状态位未被冻结。	10
3.13.3. 总线关闭恢复后发送的数据出错。	10
3.13.4. CPU 读取中断寄存器信息时可能导致发送中断信号丢失。	11
3.13.5. 接收到错误的帧可能导致下一次接收到的数据字节无效。	11
3.13.6. 仲裁失败后，帧间间隔期间的第 3 bit 上的显性位不会被当做帧起始 (SOF) 信号。	11

- 3.13.7. 当错误界定符的第 8 bit 为显性时, CAN 控制器不能进入被动错误状态。12
- 3.13.8. ESP32 CAN 在仲裁失败后的帧间间隔期间等待了挂起时间。12
- 3.13.9. 当 CAN 控制器作为发送器在仲裁段发生填充错误时, 在随后的错误帧或过载帧中发生的错误将不会使 TEC 的数值增加。12



1.

芯片修订

用户可以根据 ESP32 的 eFuse chip_version bit 来读取芯片的版本。详情请参考 [ESP32 技术参考手册](#) 中的 eFuse 控制器章节。

表 1-1. 芯片修订

芯片版本	发布日期
0	2016-09
1	2017-02



2.

ESP32 勘误表

表 2-1. ESP32 勘误表

章节	概要	影响版本
章节 3.1	芯片上电或 Deep-sleep 醒来后，会随机发生一次看门狗复位。	0
章节 3.2	CPU 使用 cache 访问外部 SRAM 时，特定条件下会发生读写错误。	0
章节 3.3	CPU 访问外设时，如果连续不间断地写同一个地址，会出现数据丢失的现象。	0
章节 3.4	Brown-out Reset（欠压复位）功能在当前版本无法工作，复位之后芯片无法起来。	0
章节 3.5	CPU 频率从 240 MHz 直接切换到 80/160 MHz 会卡死。	0
章节 3.6	同时有 GPIO 和 RTC_GPIO 功能的 pad 的上拉下拉电阻只能由 RTC_GPIO 的上拉下拉寄存器控制。	0/1
章节 3.7	Audio PLL 的频率范围有限制。	0
章节 3.8	由于 flash 启动慢于芯片读取 flash 的速度，芯片上电或 Deep-sleep 醒来后，会随机发生一次看门狗复位。	0/1
章节 3.9	CPU 在访问外部 SRAM 时会小概率发生读写错误。	1
章节 3.10	双核 CPU 在读不同地址空间时可能会发生错误。	0/1
章节 3.11	当一些外设的电源打开时，GPIO36 和 GPIO39 的数字输入会被拉低约 80 ns。	0/1
章节 3.12	LEDC 递减渐变，duty 值溢出错误。	0/1
章节 3.13	ESP32 CAN 相关问题	
章节 3.13.1	处于复位模式或总线关闭恢复状态时的接收错误计数器 (REC) 数值仍会变化。	0/1
章节 3.13.2	总线关闭恢复期间，错误状态位未被冻结。	0/1
章节 3.13.3	总线关闭恢复后发送的数据出错。	0/1
章节 3.13.4	CPU 读取中断寄存器信息时可能导致发送中断信号丢失。	0/1
章节 3.13.5	接收到错误的帧可能导致下一次接收到的数据字节无效。	0/1
章节 3.13.6	仲裁失败后，帧间间隔期间的第 3 bit 上的显性位不会被当做帧起始 (SOF) 信号。	0/1
章节 3.13.7	当错误界定符的第 8 bit 为显性时，CAN 控制器不能进入被动错误状态。	0/1
章节 3.13.8	ESP32 CAN 在仲裁失败后的帧间间隔期间等待了挂起时间。	0/1



章节	概要	影响版本
章节 3.13.9	当 CAN 控制器作为发送器在仲裁段发生填充错误时，在随后的错误帧或过载帧中发生的错误将不会使 TEC 的数值增加。	0/1



3. 问题描述和解决方法

3.1. 芯片上电或 Deep-sleep 醒来后，会随机发生一次看门狗复位。

解决方法：

Deep-sleep 醒来后的看门狗复位在 ESP-IDF V1.0 及更高版本中自动绕过。

芯片上电的看门狗复位无法使用软件绕过，但复位后 ESP32 正常启动。

详细解决方法：

Deep-sleep 醒来后，CPU 可以立即执行 RTC fast memory 中的一段程序。RTC fast memory 中的这段程序通过清除 cache MMU 的非法访问标志从而绕过 Deep-sleep 醒来后的看门狗复位，具体为：

1. 将 DPORT_PRO_CACHE_CTRL1_REG 寄存器的 PRO_CACHE_MMU_IA_CLR 比特置 1。
2. 将该比特清零。

修复：

此问题已在芯片版本 0 中修复。

3.2. CPU 使用 cache 访问外部 SRAM 时，特定条件下会发生读写错误。

描述：

使用 cache 访问外部 SRAM 时，如果这些操作需要 CPU 同时处理，则可能发生读写错误。

解决方法：

这个问题无法使用软件自动绕过。

详细解决方法：

对于版本 0 ESP32，CPU 使用 cache 访问外部 SRAM 时，只能够进行单向操作，即只能够单纯的进行写 SRAM 操作，或者单纯的进行读 SRAM 操作，不能交替操作。

使用 MEMW 指令：在读操作之后，加上 `__asm__("MEMW")` 指令，然后在 CPU 流水线被清空前再发起写操作。

**修复:**

此问题已在芯片版本 0 中修复。

3.3. CPU 访问外设时，如果连续不间断地写同一个地址，会出现数据丢失的现象。

描述:

一些 ESP32 外设映射到两条内部存储器总线（AHB 和 DPORT）。当通过 DPORT 写入时，对相同地址的连续写入可能会出现数据丢失的现象。

解决方法:

此问题在 ESP-IDF V1.0 及更高版本中自动绕过。

详细解决方法:

当连续写同一个地址（即类似 FIFO 的地址）时，使用 AHB 地址而不是 DPORT 地址。（对于其他类型的寄存器写入，使用 DPORT 地址可能写性能更好。）

寄存器名称	DPORT 地址	AHB (安全) 地址
UART_FIFO_REG	0x3ff40000	0x60000000
UART1_FIFO_REG	0x3ff50000	0x60010000
UART2_FIFO_REG	0x3ff6E000	0x6002E000
I2S0_FIFO_RD_REG	0x3ff4F004	0x6000F004
I2S1_FIFO_RD_REG	0x3ff6D004	0x6002D004
GPIO_OUT_REG	0x3ff44004	0x60004004
GPIO_OUT_W1TC_REG	0x3ff4400c	0x6000400c
GPIO_OUT1_REG	0x3ff44010	0x60004010
GPIO_OUT1_W1TS_REG	0x3ff44014	0x60004014
GPIO_OUT1_W1TC_REG	0x3ff44018	0x60004018
GPIO_ENABLE_REG	0x3ff44020	0x60004020
GPIO_ENABLE_W1TS_REG	0x3ff44024	0x60004024
GPIO_ENABLE_W1TC_REG	0x3ff44028	0x60004028
GPIO_ENABLE1_REG	0x3ff4402c	0x6000402c
GPIO_ENABLE1_W1TS_REG	0x3ff44030	0x60004030



寄存器名称	DPORT 地址	AHB (安全) 地址
GPIO_ENABLE1_W1TC_REG	0x3ff44034	0x60004034

修复:

此问题已在芯片版本 0 中修复。

3.4. Brown-out Reset (欠压复位) 功能在当前版本无法工作, 复位之后芯片无法起来。

解决方法:

无。

修复:

此问题已在芯片版本 1 中修复。

3.5. CPU 频率从 240 MHz 直接切换到 80/160 MHz 会卡死。

解决方法:

此问题在 ESP-IDF V2.1 及更高版本中自动绕过。

详细解决方法:

建议使用以下两种模式:

(1) 2 MHz <-> 40 MHz <-> 80 MHz <-> 160 MHz

(2) 2 MHz <->40 MHz <->240 MHz

修复:

此问题已在芯片版本 1 中修复。

3.6. 同时有 GPIO 和 RTC_GPIO 功能的 pad 的上拉下拉电阻只能由 RTC_GPIO 的上拉下拉寄存器控制。

描述:

这些 pad 的 GPIO 上拉下拉配置寄存器字段不能使用。

解决方法:

ESP-IDF V2.1 及更高版本的 GPIO 驱动自动绕过此问题。



详细解决方法:

GPIO 和 RTC_GPIO 都使用 RTC_GPIO 寄存器。

3.7. Audio PLL 使用频率有限制。

描述:

当配置 Audio PLL 频率时，不会用到配置寄存器 `sdm0` 和 `sdm1`，这样就限制了 PLL 频率可以配置的范围和精度。

版本 0 ESP32 芯片的 Audio PLL 频率公式如下:

$$f_{\text{out}} = \frac{f_{\text{xtal}}(\text{sdm2}+4)}{2(\text{odiv}+2)}$$

版本 1 及之后的 ESP32 芯片已修复此问题，Audio PLL 频率公式如下:

$$f_{\text{out}} = \frac{f_{\text{xtal}}(\text{sdm2} + \frac{\text{sdm1}}{2^8} + \frac{\text{sdm0}}{2^{16}} + 4)}{2(\text{odiv}+2)}$$

解决方法:

在 ESP-IDF V3.0 及更高版本中通过 I2S 驱动程序设置 Audio PLL 频率时，会自动考虑相关的频率公式。但是对于版本 0 ESP32 芯片，Audio PLL 的使用频率仍然有限制。

修复:

此问题已在芯片版本 1 中修复。

3.8. 由于 flash 启动的速度慢于芯片读取 flash 的速度，芯片上电或 Deep-sleep 醒来后，会随机发生一次看门狗复位。

描述:

如果 ESP32 在 flash 可读之前就进行读取，则无效数据会使启动失败，这时会发生看门狗复位。如果 ESP32 VDD_SDIO 用作 flash 电源，则芯片上电和 Deep-sleep 醒来时都可能发生看门狗复位。

解决方法:

- (1) 更换更快的 flash，要求 flash 上电到可读的时间小于 800 μs 。这种方法可以绕过芯片上电和 Deep-sleep 醒来时的看门狗复位。



(2) Deep-sleep 醒来后的看门狗复位问题在 ESP-IDF V2.0 及更高版本中自动绕过（延迟时间可以根据需要配置）。具体方式是从 Deep-sleep 醒来后首先读取 RTC fast memory 中的指令，等待一段时间，然后再读取 flash。

3.9. CPU 在访问外部 SRAM 时会小概率发生读写错误。

描述：

CPU 在执行下面汇编指令访问外部 SRAM 时会小概率发生错误：

```
store.x at0, as0, n  
load.y at1, as1, m
```

其中 `store.x` 表示 x 位写操作，`load.y` 表示 y 位读操作，且 `as0+n` 和 `as1+m` 访问的外部 SRAM 的地址相同。

- 指令可以是连续的，也可以包含在同一个流水线中（少于 4 个中间指令，并且没有流水线刷新）。
- $x \geq y$ 时，写数据会丢失。（注意：当 `load` 和 `store` 都是 32-bit 值时，写数据只有在第一个和第二个指令之间发生中断时才发生。）
- $x < y$ 时，写数据会丢失，且读数据错误。

解决方法：

当外部 SRAM 在 ESP-IDF V3.0 及更高版本中启用时，此问题自动绕过。

详细解决方法：

- $x \geq y$ 时，在 `store.x` 和 `load.y` 之间插入 4 个 `nop` 指令。
- $x < y$ 时，在 `store.x` 和 `load.y` 之间插入 `memw` 指令。

3.10. 双核 CPU 在读不同地址空间时可能会发生错误。

描述：

双核情况下，一个 CPU 的总线在读 A (`0x3FF4_0000 ~ 0x3FF7_FFFF`) 地址空间，而另一个 CPU 的总线在读 B (`0x3FF0_0000 ~ 0x3FF1_EFFF`) 地址空间，读 A 地址空间的 CPU 可能会发生错误。

解决方法：

此问题在 ESP-IDF V3.0 及更高版本中自动绕过。

详细解决方法：

以下两种方法都可以使用：



- 一个 CPU 在读 A 地址空间时，通过加锁和中断的方式来避免另一个 CPU 发起对 B 地址空间的读操作。
- 一个 CPU 在读 A 地址空间之前，加一个此 CPU 读 B 地址空间（非 FIFO 地址空间，如 0x3ff40078）操作，并且要保证读 B 地址空间操作和读 A 地址空间操作是原子的。

3.11. 当一些 RTC 外设的电源打开时，GPIO36 和 GPIO39 的数字输入会被拉低约 80 ns。

描述：

打开以下 RTC 外设的电源会发生此问题：

- SARADC1 传感器
- SARADC2 传感器
- AMP 传感器
- HALL 传感器

解决方法：

当用户决定把用于控制以上传感器的电源域打开时，应当忽略来自 GPIO36 和 GPIO39 的输入。

3.12. LEDC 递减渐变，duty 值溢出错误。

描述：

在配置 LEDC 为递减渐变且 $LEDC_DUTY_SCALE_HSCH_n$ 为 1 的情况下，当 duty 值为 $2^{LEDC_HSTIMER_x_DUTY_RES}$ 时，下一次 duty 变化应该为 $2^{LEDC_HSTIMER_x_DUTY_RES} - 1$ ，但是实际上 duty 值等于 $2^{LEDC_HSTIMER_x_DUTY_RES+1}$ ，即出现 duty 值溢出的错误。（ $HSCH_n$ 代表高速通道， n 为 0-7； $HSTIMER_x$ 代表高速定时器， x 为 0-3。）

对于低速通道，存在同样的问题。

解决方法：

此问题在 ESP-IDF commit ID 为 b2e264e 及以后版本的 LEDC 驱动中已自动绕过，并将于 ESP-IDF V3.1 中发布。

详细解决方法：

使用 LEDC 的过程中，应避免以下三个条件同时成立：

1. LEDC 启动递减渐变功能；



2. LEDC 渐变过程中 scale 寄存器设置为 1；
3. LEDC 递减渐变开始时刻或者过程中的某一时刻，duty 值为 $2^{\text{LEDC_HSTIMER}_x_DUTY_RES}$ 或 $2^{\text{LEDC_LSTIMER}_x_DUTY_RES}$ 。

3.13. ESP32 CAN 相关问题

3.13.1. 处于复位模式或总线关闭恢复状态时的接收错误计数器 (REC) 数值仍会变化。

描述：

当 CAN 控制器处于复位模式（即 RESET_MODE 位置 1 或由于总线关闭）或总线关闭恢复状态时，接收错误计数器 (REC) 的数值仍会变化，这会引发以下问题：

- 错误状态位可能发生改变，进而触发错误报警限制中断。
- REC > 0 可能导致 CAN 控制器无法从总线关闭状态恢复。

解决方法：

进入复位模式时，应将 LISTEN_ONLY_MODE 置位，此时 REC 数值不会变化。退出复位模式前或总线关闭恢复完成时，再恢复正常的操作模式。

3.13.2. 总线关闭恢复期间，错误状态位未被冻结。

描述：

当 CAN 控制器处于总线关闭恢复过程中时，必须等待总线上出现 128 次总线空闲信号（连续 11 个隐性位），才能再次进入主动错误状态。剩余的总线空闲信号由发送错误计数器 (TEC) 指示。由于错误状态位在总线恢复期间未冻结，因此当发送错误计数器低于用户设置的错误报警限制数值（默认值为 96）时，错误状态位将发生变化，从而导致在总线关闭恢复完成之前触发错误报警限制中断。

解决方法：

在总线关闭恢复过程中，错误报警限制中断并不一定指示恢复过程已完成。用户需检查 STATUS_NODE_BUS_OFF 位来验证恢复过程是否完成。

3.13.3. 总线关闭恢复后发送的数据出错。

描述：

总线关闭恢复完成后，CAN 控制器下一次发送的数据可能出错（即不符合 CAN 数据帧格式）。

**解决方法：**

一旦通过错误报警限制中断检测到总线关闭恢复完成，CAN 控制器应先进入复位模式来复位控制器的内部信号，随后退出复位模式。

3.13.4. CPU 读取中断寄存器信息时可能导致发送中断信号丢失。**描述：**

CPU 通过读取 INTERRUPT_REG 寄存器来复位 CAN 控制器的中断信号。如果在同一个 APB 时钟周期内 CAN 控制器刚好产生发送中断信号，则发送中断信号丢失。

解决方法：

数据等待发送完成期间（即发送请求已发起），每一次读取 INTERRUPT_REG 后，用户都应检查 STATUS_TRANSMIT_BUFFER 位。如果 STATUS_TRANSMIT_BUFFER 置位而 CAN_TRANSMIT_INT_ST 没有置位，则说明发送中断信号丢失。

3.13.5. 接收到错误的帧可能导致下一次接收到的数据字节无效。**描述：**

当 CAN 控制器接收数据帧时，如果在数据段或 CRC 字段中发生位错误或填充错误，则下一次接收到的数据可能发生字节移位或丢失。因此，下一次接收的数据帧（包括由验收滤波器滤出的数据帧）应视为无效。

解决方法：

用户可以通过置位 INTERRUPT_BUS_ERR_INT_ENA 并在接收到总线错误中断时，读取 ERROR_CODE_CAPTURE_REG 来检测错误类型及错误位置。如果符合错误产生条件（在数据段或 CRC 字段发生位错误或填充错误），可以采用以下两种解决方法：

- CAN 控制器可以发送 0 字节的空数据帧来复位 CAN 控制器的内部信号。建议给空数据帧分配一个不会被任何 CAN 总线上的节点接收的 ID。
- 硬件复位 CAN 控制器（需要保存并恢复当前寄存器的数值）。

3.13.6. 仲裁失败后，帧间间隔期间的第 3 bit 上的显性位不会被当做帧起始 (SOF) 信号。**描述：**

CAN2.0B 协议规定帧间间隔期间的第 3 bit 上的显性位应当被当做帧起始 (SOF) 信号。因此，CAN 节点应在下一个 bit 上接收或发送（即参与竞争仲裁）ID 字段。

当 ESP32 CAN 控制器失去仲裁并且下一个帧间间隔期间检测到第 3 位为显性时，ESP32 CAN 控制器不会将其视作 SOF 并且不会参与竞争仲裁（即，不会重传数据）。

**解决方法：**

无。

3.13.7. 当错误界定符的第 8 bit 为显性时，CAN 控制器不能进入被动错误状态。**描述：**

当 CAN 控制器发送数据并且 TEC 的值为 120 ~ 127 时，发送错误帧会使 TEC 增加 8 并且 CAN 控制器会进入被动错误状态（CAN 2.0B 协议规定 $TEC \geq 128$ 时，CAN 节点应进入错误被动状态）。但是，如果错误界定符的第 8 bit 为显性时，TEC 仍会增加 8，而 CAN 控制器不会进入被动错误状态。再次发送错误帧后 CAN 控制器才会进入被动错误状态。注意，由于错误界定符的第 8 bit 为显性，CAN 控制器仍会产生协议规定的过载帧。

解决方法：

无。

3.13.8. ESP32 CAN 在仲裁失败后的帧间间隔期间等待了挂起时间。**描述：**

CAN2.0B 协议规定作为发送器并处于被动错误态的 CAN 节点应在随后的帧间间隔内等待挂起时间。但是，作为接收器的被动错误态 CAN 节点是不需要等待挂起时间的。

当 CAN 控制器处于被动错误态并且失去仲裁（转为接收器）时，它仍将在随后的帧间间隔中等待挂起时间。这导致 CAN 控制器的重传延后，如果在其等待帧间间隔挂起时间期间，另一个节点发送数据，则 CAN 控制器不会参与竞争仲裁。

解决方法：

无。

3.13.9. 当 CAN 控制器作为发送器在仲裁段发生填充错误时，在随后的错误帧或过载帧中发生的错误将不会使 TEC 的数值增加。**描述：**

CAN2.0B 协议规定，CAN 控制器发送数据时如果在仲裁字段中检测到填充错误，应发送错误帧，但是 TEC 不应增加（错误计数规则 3 中的特例 2）。CAN 控制器能够满足这一规定。

但是，若在随后发生的错误帧或过载帧中遇到以下两种情况时，TEC 数值将不能按照 CAN2.0B 协议要求实现增加：

- 在主动错误标志或过载标志期间发生位错误（规则 4）。
- 在主动错误标志、被动错误标志或过载标志之后检测到过多的显性位（规则 6）。



解决方法:

无。



乐鑫 IOT 团队
www.espressif.com

免责声明和版权公告

本文中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。

文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。蓝牙标志是 Bluetooth SIG 的注册商标。文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

版权归© 2018 乐鑫所有。保留所有权利。