ECO and Workarounds for Bugs in ESP32

Version 1.3

About This Guide

This document details the bugs in the ESP32. The structure is as follows:

Chapter	Title	Content
Chapter 1	Chip Revision	Identifying the chip revision.
Chapter 2	Bug List	Summary of all the bugs.
Chapter 3	Bug Descriptions and Workarounds	Detailed description of each bug and possible workarounds.

Release Notes

Date	Version	Release notes
2016-11	V1.0	Initial release.
2016-12	V1.1	Modified the MEMW command in Section 3.2.
2017-04	V1.2	Changed the description of the bug in Section 3.1; Added a bug in Section 3.8.
2017-06	V1.3	Added bugs in Section 3.9 and 3.10

Table of Contents

1. Chip Revision

The chip revision is identified by the eFuse bit. Details can be found in Chapter eFuse in *ESP32 Technical Reference Manual*.

Table 1-1. Chip Revision

Chip revision	Release date
0	2016-09
1	2017-02

2. Bug List

Table 2-1 provides a summary of the bugs.

Table 2-1. Bug List

Section	Title	Affected revisions
Section 3.1	Due to the cache MMU bug, a spurious watchdog reset occurs when ESP32 is powered up or wakes up from Deep-sleep.	0
Section 3.2	When the CPU accesses external SRAM through cache, sometimes random read and write errors occur.	0
Section 3.3	When the CPU accesses peripherals and writes one address repeatedly, random data loss occurs.	0
Section 3.4	The Brown-out Reset (BOR) function is not functional. The system fails to boot up after BOR.	0
Section 3.5	The CPU crashes when the clock frequency switches from 240 MHz to 80/160 MHz.	0
Section 3.6	The pull-ups and pull-downs for the pads with both GPIO and RTC_GPIO functions can only be controlled by the RTC_GPIO registers. For these pads, the GPIO pull-up and pull-down configuration fields are non-functional.	0/1
Section 3.7	There is a limit on the frequency range for the audio PLL.	0
Section 3.8	Due to the flash start-up time, a spurious watchdog reset occurs when ESP32 is powered up or wakes up from Deep-sleep.	0/1
Section 3.9	When the CPU accesses the external SRAM, a rare error occurs.	1
Section 3.10	When the dual-core CPU accesses different address spaces, a random error occurs.	0/1

3. Bug Descriptions and Workarounds

3.1. Due to the cache MMU bug, a spurious watchdog reset occurs when ESP32 is powered up or wakes up from Deep-sleep.

Description:

The spurious power-up watchdog reset cannot be bypassed with software.

However, when ESP32 wakes up from Deep-sleep, the watchdog reset can be bypassed with software.

Workarounds:

After waking up from Deep-sleep, the CPU will read an instruction from the RTC fast memory, and then execute the boot program from this memory. This program needs to clear the illegal access flag in the cache MMU by:

- setting the PRO_CACHE_MMU_IA_CLR bit in DPORT_PRO_CACHE_CTRL1_REG to 1, and then,
- clearing the bit.

3.2. When the CPU accesses external SRAM through cache, sometimes random read and write errors occur.

Description:

The errors cannot be bypassed with software.

For the current version of ESP32, the access to external SRAM by CPU through cache is limited to a one-way operation, that is, the write operation only or the read operation only. Alternative operations are not available.

Workarounds:

- Clear the pipeline after the read operation and then initiate the write operation.
- Use MEMW command. Add `__asm__("MEMW")` command after the read operation, and then initiate the write operation.

3.3. When the CPU accesses peripherals and writes one address repeatedly, random data loss occurs.

Workarounds:

The FIFO-related addresses and some of the GPIO-related addresses need to be changed as follows:

Registers	Original addresses	Changed addresses
UART_FIFO	0x3ff40000	0x60000000
UART1_FIFO	0x3ff50000	0x60010000
UART2_FIFO	0x3ff6E000	0x6002E000
I2S0_FIFO	0x3ff4F004	0x6000F004
I2S1_FIFO	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
GPIO_ENABLE1_W1TC_REG	0x3ff44034	0x60004034

3.4. The Brown-out Reset (BOR) function does not work. The system fails to boot up after BOR.

Workarounds:

There is no workaround for this issue.

3.5. The CPU crashes when the clock frequency switches from 240 MHz to 80/160 MHz.

Workarounds:

We recommend the following two switching modes:

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

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

3.6. The pull-ups and pull-downs for the pads with both GPIO and RTC_GPIO functions can only be controlled by the RTC_GPIO registers. For these pads, the GPIO pull-up and pull-down configuration fields are non-functional.

Workarounds:

Use RTC_GPIO registers for both GPIO and RTC_GPIO functions.

3.7. There is a limit on the frequency range for the audio PLL.

Description:

For the current version of chip, the frequency formula is as follows:

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

For the later versions of the chips with the bug fixed, the frequency formula is:

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

3.8. Due to the flash start-up time, a spurious watchdog reset occurs when ESP32 is powered up or wakes up from Deep-sleep.

Description:

The spurious power-up watchdog reset cannot be bypassed with software. However, it can be bypassed by using a flash chip with fast start-up time.

The Deep-sleep reset can be bypassed by replacing the flash chip (workaround 1) or with software (workaround 2).

Workarounds:

1. Replace the flash chip with one with fast start-up time (<800 µs from boot to being read).
2. After waking up from Deep-sleep, the CPU will read an instruction from the RTC fast memory to wait for a while before executing the boot program. In this way, the flash start-up time will be longer than that spent by the CPU from boot to reading flash.

3.9. When the CPU accesses the external SRAM, a rare error occurs.

Description:

When the CPU executes the following assembly instructions to access the external SRAM, a rare error can occur:

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

In the assembly instructions, `store.x` represents the x-bit write operation, while `load.y` represents the y-bit read operation. `as0+n` and `as1+m` access the same address of the external SRAM.

- When x>=y, random write-data loss occurs. (NOTE: when both the `load` and the `store` refer to 32-bit values, the problem only occurs when an interrupt happens.)
- When x <y, random write-data loss and read-data error occur.

Workarounds:

- When x>=y and random write-data loss occurs, insert four `nop` instructions between `store.x` and `load.y`.
- When x <y and both random write-data loss and a read-data error occur, insert the `memw` instruction between `store.x` and `load.y`.

3.10. When the dual-core CPU accesses different address spaces, a random error occurs.

Description:

In the case of a dual-core CPU, when one CPU bus reads address space A (0x3FF0_0000 ~ 0x3FF1_EFFF), while the other CPU bus reads address space B (0x3FF4_0000 ~ 0x3FF7_FFFF), a random error will be generated on the CPU bus reading address space A.

Workarounds:

When one CPU bus reads address space A, prevent the other CPU bus from reading address space B by implementing locks and interrupts.

Espressif IoT Team

www.espressif.com