ECO and Workarounds for Bugs in ESP32

Version 1.7

About This Guide

This document details hardware errata and workarounds in the ESP32.

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 Section 3.1; Added a bug in Section 3.8.
2017-06	V1.3	Added items 3.9 and 3.10
2018-02	V1.4	Corrected typos in the register names in Section 3.3.
2018-02	V1.5	Added Section 3.11.
2018-05	V1.6	Overall update.
2018-05	V1.7	Added Section 3.12.

Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe *here*.

Certification

Download certificates for Espressif products from *here*.

Table of Contents

1. Chip Revision

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

Table 1-1. Chip Revision

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

2. Errata List

Table 2-1. Errata Summary

Section	Title	Affected revisions
Section 3.1	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, under certain conditions read and write errors occur.	0
Section 3.3	When the CPU accesses peripherals and writes a single address repeatedly, some writes may be lost.	0
Section 3.4	The Brown-out Reset (BOR) function does not work. The system fails to boot up after BOR.	0
Section 3.5	The CPU crashes when the clock frequency switches directly from 240 MHz to 80/160 MHz.	0
Section 3.6	GPIO pull-up and pull-down resistors for pads with both GPIO and RTC_GPIO functionality can only be controlled via RTC_GPIO registers.	0/1
Section 3.7	Audio PLL frequency range is limited.	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 external SRAM in a certain sequence, read and write errors can occur.	1
Section 3.10	When each CPU reads certain different address spaces simultaneously, a read error can occur.	0/1
Section 3.11	When certain RTC peripherals are powered on, the inputs of GPIO36 and GPIO39 will be pulled down for approximately 80 ns.	0/1
Section 3.12	When the LEDC is in decremental fade mode, a duty overflow error can occur.	0/1

3. Errata Descriptions and Workarounds

3.1. A spurious watchdog reset occurs when ESP32 is powered up or wakes up from Deep-sleep.

Workarounds:

When waking from Deep-sleep, this bug is worked around automatically in ESP-IDF V1.0 and newer.

During initial power-up the spurious watchdog reset cannot be worked around, but ESP32 will boot normally after this reset.

Workaround Details:

To work around the watchdog reset when waking from Deep-sleep, the CPU can execute a program from RTC fast memory. This program must clear the illegal access flag in the cache MMU as follows:

1. Set the PRO_CACHE_MMU_IA_CLR bit in DPORT_PRO_CACHE_CTRL1_REG to 1.
2. Clear this bit.

Fixes:

This issue is fixed in silicon revision 1.

3.2. When the CPU accesses external SRAM through cache, under certain conditions read and write errors occur.

Details:

Access to external SRAM through cache will cause read and write errors if these operations are pipelined together by the CPU.

Workarounds:

There is no automatic workaround available in software.

Workaround Details:

If accessing external SRAM from a revision 0 ESP32, users must ensure that access is always one-way—only a write or a read can be in progress at a single time in the CPU pipeline.

The MEMW instruction can be used: insert `__asm__("MEMW")` after any read from external PSRAM that may be followed by a write to PSRAM before the CPU pipeline is flushed.

Fixes:

This issue is fixed in silicon revision 1.

3.3. When the CPU accesses peripherals and writes a single address repeatedly, some writes may be lost.

Details:

Some ESP32 peripherals are mapped to two internal memory buses (AHB & DPORT). When written via DPORT, consecutive writes to the same address may be lost.

Workarounds:

This issue is automatically worked around in the drivers of ESP-IDF V1.0 and newer.

Workaround Details:

When writing the same register address (i.e., FIFO-like addresses) in sequential instructions, use the equivalent AHB address not the DPORT address.

(For other kinds of register writes, using DPORT registers will give better write performance.)

Registers	DPORT Address	AHB (Safe) Addresses
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
GPIO_ENABLE1_W1TC_REG	0x3ff44034	0x60004034

Fixes:

This issue is fixed in silicon revision 1.

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.

Fixes:

This issue is fixed in silicon revision 1.

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

Workarounds:

This issue is automatically worked around in ESP-IDF V2.1 and newer.

Workaround Details:

When switching frequencies, use intermediate frequencies as follows:

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

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

Fixes:

This issue is fixed in silicon revision 1.

3.6. GPIO pull-up and pull-down resistors for pads with both GPIO and RTC_GPIO functionality can only be controlled via RTC_GPIO registers.

Details:

For these pads, the GPIO pull-up and pull-down configuration register fields are non-functional.

Workarounds:

This issue is automatically worked around when using GPIO drivers in ESP-IDF V2.1 or newer.

Workaround Details:

Use RTC_GPIO registers for both GPIO and RTC_GPIO functions.

3.7. Audio PLL frequency range is limited.

Details:

When configuring the Audio PLL, configuration registers sdm0 & sdm1 are not used. This limits the range and precision of PLL frequencies which can be configured.

For chip revision 0, the Audio PLL frequency is calculated in hardware as follows:

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

For chip revision 1 onwards this bug is fixed and the Audio PLL frequency is calculated in hardware as follows:

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

Workarounds:

The particular hardware frequency calculation is automatically accounted for when setting Audio PLL frequency via the I2S driver in ESP-IDF V3.0 and newer. However, the range and precision of available Audio PLL frequencies is still limited when using silicon revision 0.

Fixes:

This issue is fixed in silicon revision 1.

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.

Details:

If the ESP32 reads from the flash chip before it is ready, invalid data can cause booting to fail until a Watchdog Timer reset occurs. This can occur on power-on and on wake from Deep-sleep, if the ESP32 VDD_SDIO is used to power the flash chip.

Workarounds:

1. Replace the flash chip with one with a fast start-up time (<800 μs from power-on to ready to read). This works around the issue for both power-on and wake from Deep-sleep.
2. When waking from Deep-sleep, this issue is automatically worked around in ESP-IDF V2.0 and newer (the delay to wait can be configured if necessary). In this workaround, the CPU executes from RTC fast memory immediately after waking and a delay is added before continuing to read the program from flash.

3.9. When the CPU accesses the external SRAM in a certain sequence, read & write errors can occur.

Details:

This error can occur when the CPU executes the following instructions to access external SRAM:

```
store.x at0, as0, n
```

```
load.y at1, as1, m
```

In the pseudo-assembly instructions above, `store.x` represents an x-bit write operation, while `load.y` represents a `y-bit` read operation. `as0+n` and `as1+m` represent the same address in external SRAM.

- The instructions can be sequential or contained within the same pipeline (less than four intermediate instructions, and no pipeline flushes.)
- When x>=y, the data write may be lost. (NOTE: when both the `load` and the `store` refer to 32-bit values, the write is only lost if an interrupt occurs between the first and second instructions.)
- When x <y, data writes may be lost and invalid data may be read.

Workarounds:

This bug is automatically worked around when external SRAM use is enabled in ESP-IDF V3.0 and newer.

Workaround Details:

- When x>=y, insert four `nop` instructions between `store.x` and `load.y`.
- When x <y, insert a `memw` instruction between `store.x` and `load.y`.

3.10. When each CPU reads certain different address spaces simultaneously, a read error can occur.

Details:

Running in dual-core CPU mode, when one CPU bus reads address space A (0x3FF4_0000 ~ 0x3FF7_FFFF), while the other CPU bus reads address space B (0x3FF0_0000 ~ 0x3FF1_EFFF), an incorrect read can be generated on the CPU reading address space A.

Workarounds:

This issue is automatically worked around in ESP-IDF V3.0 and newer.

Workaround Details:

Either of the following workarounds can be used:

- When either CPU reads address space A, prevent the other CPU bus from reading address space B via locks and interrupts.
- Before reading address space A, disable interrupts and insert a read from address space B on the same CPU (read a non-FIFO register, e.g., 0x3ff40078).

3.11. When certain RTC peripherals are powered on, the inputs of GPIO36 and GPIO39 will be pulled down for approximately 80 ns.

Details:

Powering on the following RTC peripherals will trigger this issue:

- SARADC1
- SARADC2
- AMP
- HALL

Workarounds:

When enabling power for any of these peripherals, ignore input from GPIO36 and GPIO39.

3.12. When the LEDC is in decremental fade mode, a duty overflow error can occur.

Details:

This issue may happen when the LEDC is in decremental fade mode and LEDC_DUTY_SCALE_HSCH*n* is 1. If the duty is $2^{LEDC\_HSTIMERx\_DUTY\_RES}$, then the next one should be $2^{LEDC\_HSTIMERx\_DUTY\_RES} - 1$, however, the next duty is actually $2^{LEDC\_HSTIMERx\_DUTY\_RES+1}$, which indicates a duty overflow error. (HSCH*n* refers to high-speed channel with *n* being 0-7; HSTIMER$_x$ refers to high-speed timer with *x* being 0-3.)

For low-speed channels, the same issue may also happen.

Workarounds:

This issue is automatically worked around in the LEDC driver since the ESP-IDF commit ID b2e264e and will be part of the ESP-IDF V3.1 release.

Workaround Details:

When using LEDC, avoid the concurrence of following three cases:

1. The LEDC is in decremental fade mode;
2. The scale register is set to 1;
3. The duty is $2^{LEDC\_HSTIMERx\_DUTY\_RES}$ or $2^{LEDC\_LSTIMERx\_DUTY\_RES}$.