CTU CAN FD

Errata For Version 2.5

June 20, 2025

Document Version	Date	Change description
0.1	27-8-2024	Initial version describing errata for release 2.5
0.2	27-2-2025	Added Issue 5 - MODE[BMM]. Added Notes on what has been fixed.
0.3	6-6-2025	Mark Issue 5 as resolved.
0.4	20-6-2025	Elaborate on the Workaround of Issue 5

Introduction

This document describes issues found in CTU CAN FD design.

At the time of writing this document, CTU CAN FD is primarily revisioned on CTU FEE Giltlab:

`https://gitlab.fel.cvut.cz/canbus/ctucanfd_ip_core`

Second mirror repository is on Github:

`https://github.com/Blebowski/CTU-CAN-FD/tree/2v5-lts`

The document classifies the issues into several severities:

- Very low - Do not affect FW, integrated system, communication on CAN bus with other nodes nor CAN Compliance tests result.
- Low - Potential impact on FW and system is minimal. Has no impact on communication on CAN bus with other nodes. Has no impact on CAN Compliance tests result.
- Medium - May require a FW workaround. Has no impact on CAN bus communication with other nodes. May affect CAN Compliance test result.
- High - Likely will require FW workaround to operate correctly. May have impact on CAN bus communication with other nodes. Affects CAN Compliance test result.
- Very High - Either requires FW workaround, or no such FW workaround is possible. Affects CAN bus communcication (may result in undefined behavior), and definitely will affect CAN Compliance test result.

Version 2.5

This section describes issues found in the CTU CAN FD version 2.5.

Git Tag: v2.5

Git Hash: 859a1495

The v2.5 GIT tag was used to create a 2v5-lts GIT branch to maintain 2v5 release. 2v5-lts branch contains equal HW design as v2.5 GIT tag, but with testbench improvements to achieve higher code coverage in CTU CAN FD verification. 2v5-lts is a branch where potential ECO fixes for issues described in this section can be applied.

Git Tag: v2.5.1

Git Hash: 27efaafa

The v2.5.1 GIT tag was used to 2v5-lts branch. This tag contains fixes of Issue 2, Issue 3 and Issue 4. This tag still contains Issue 1 and Issue 5. Issue 1 is resolved in the 2v5-lts branch after the v2.5.1 commit. Issue 5 is not yet resolved.

Issue 1

Type: Documentation inconsistency

Severity: Very low

Description: MODE[SAM] (self-acknowledge mode) is present in the datasheet in 2v5-lts branch, but this mode is not implemented in CTU CAN FD HW.

Consequence: None

Fixed in later development: Currently, the Self-acknowledge mode is implemented in "451-add-loopback-flag-to-frame-in-rx-buffer-2" branch towards 2.6 release in GIT commit: `https://gitlab.fel.cvut.cz/canbus/ctucanfd_ip_core/-/commit/0075b8e1a2cdefb5ce89eb825525ab218e26fa94`

Workaround: None needed. MODE[SAM] behaves as a R/W bit with no effect. It is enough to make it reserved and remove the Datasheet section with Self-acknowledge mode. Simply, Self-acknowledge mode is not implemented in 2.5

ECO fixable: Not applicable

Next steps: Will push a fix in 2v5-lts branch that makes the datasheet documentation constistent with real HW.

Fixed in: `https://gitlab.fel.cvut.cz/canbus/ctucanfd_ip_core/-/commit/a4809536e83cac2f56e432899d807096fcf7f4` - A build PDF of Datasheet is commited.

Issue 2

Type: Bug

Severity: Low

Description: An incorrect reset source is used for registers holding STATUS[TXPE] and STATUS[TXDPE] bits.

Consequence: When either of STATUS[TXPE], STATUS[TXDPE] bits is set as a result of parity error in TXT Buffers, then issuing Soft reset (writing MODE[RST] = 1) will not clear these flags. The correct behavior is, that issuing Soft reset shall clear all bits of Register map into their reset value.

Fixed in later development: Not yet, planned for v2.6 release.

Workaround: None should be needed in FW.

ECO_fixable: Definitely yes

Next steps: Will fix in the 2.6. If needed, RTL change can be pushed into 2v5-lts for ECO.

Fixed in: 2.5.1

Issue 3

Type: Bug

Severity: Medium

Description: CTU CAN FD does not detect bit error when bit error occurs during last bit of CRC of CAN FD frame with Secondary sampling point enabled, and configured to be between regular sampling point and end of the last CRC bit. According to section 11.3.3 of [1], if a bit error is detected with secondary sampling point, the IUT shall react to this bit error in the follow-up regular sample point. When such bit error occurs in the last bit of CRC, and the Secondary sample point is configured to be "later" than regular sample point (the SSP for last CRC bit occurs from SP of last bit of CRC until SP of CRC delimiter), then CTU CAN FD will not detect this bit error, and will continue with regular transmission.

Consequence: Consequence on real bus operation is minimal. It is a single bit where CTU CAN FD does not detect local disturbance, thus the error detection capabilites are only negligibly worsened. Further, [1] defines, that from Sample point of CRC delimiter, the IUT may optionally decide to "cut-off" secondary sample points for previous bits that are still "on-the-fly". The CTU CAN FD bug only extends this period by one more bit.
No FW workaround is needed, and communication with other nodes on CAN bus is not affected beyond negligible reduction in error detection capabilites.
Compliance tests defined in [2] have a test that is close to the scenario above. The test is 8.8.2.4. This test invokes bit error in last bit of CRC and delays the received bit sequence by 1 or 2 Data bit times (***d***). The IUT is configured to have SSP offset similar to "regular" data sample point (see test 8.8.2.4). Together with measured Transmitter delay, the overall position of SSP from start of CRC delimiter bit will be:
SSP pos = ***d*** + SSP offset
Then the sample point detecting the bit error will occur in the sample point of BRS (at the moment of switching bit-rate back to nominal). [1] defines that bit errors that are detected by SSP of previous bits beyond this moment, shall be ignored (only optionally can be reacted to), see figures 25 and 26 in [1]. The 8.8.2.4 test verifies this scenario, and it expects that IUT shall ignore such bit error, not transmit any error frame. CTU CAN FD passes this test, and does not transmit any error frame.
The issue with CTU CAN FD is when transmitter delay (***d***) would be small (e.g. just 1 Data time quanta), and SSP offset would be configured similar to regular sample point. Then, if the Secondary sampling point is enabled (SSP_CFG=SSP_SRC_MEAS_N_OFFSET), and bit error occurs in the last bit of CRC, CTU CAN FD will not recognize bit error in the sample point of last CRC bit (since SSP offset is configured equal to regular sample point). The CTU CAN FD shall recognize such error in the next regular sample point (sample point of CRC delimiter), and react with error frame from start of ACK bit. CTU CAN FD does not detect such error, nor transitts the error frame. This is a bug.
Compliance tests do not verify such scenario, and thus the bug was not discovered. If Certification does only execute tests according to [2], then the bug will possibly not result in CAN Compliance certification failing. If the Certification authority does "extra tests" to verify such functionality, this issue can affect CAN Compliance certification results. Other compliance tests do not verify Secondary sample point in the last bit of CRC.

Fixed in later development: Not yet, planned for v2.6 release.

Workaround: None is possible on FW level. The issue does not occur when SSP is disabled (SSP_CFG=NO_SSP), but such setting may be too limiting since CAN FD nodes are required to implement secondary sample point.

ECO fixable: Likely yes, the change should be couple of gates in output decoder of Protocol control FSM. I estimate incremental synthesis will be able to do ECO. Plausibility depends on spare gates used in the design (amount, location, etc...).

Next steps: Will be fixed in 2.6. If needed, RTL change can be pushed into 2v5-lts for ECO.

Fixed in: 2.5.1

Issue 4

Type: Bug

Severity: Medium

Description: When CTU CAN FD operates in Loopback mode (SETTINGS[ILBP] = 1) and a bit error occurs during last bit of DLC when CTU CAN FD is transmitting a CAN Frame, then CTU CAN FD will not revert the write pointer in RX Buffer FIFO, and result in inconsistent RX Buffer FIFO write pointer.

Consequence: In this scenario, the write pointer of RX Buffer FIFO will get corrupted, and will require RX Buffer FIFO Flush to become consistent again. The bug does not affect communication on CAN bus with other nodes. The bug does not affect CAN Compliance certification.

Fixed in later development: Fixed in "451-add-loopback-flag-to-frame-in-rx-buffer-2" branch towards 2.6 release in GIT commit: `https://gitlab.fel.cvut.cz/canbus/ctucanfd_ip_core/-/commit/46cffe5b1ff77c299bc0067c3fb2f92c74`

Workaround: There are several possible options on FW level (mutually exclusive):

1. Not use Loopback mode (keep SETTINGS[ILBP] = 0).
2. If using Loopback mode (SETTINGS[ILBP] = 1), then upon any error frame that occurs in Contrrol field, issue RX Buffer Flush (writing COMMAND[RRB]=1). Error frame occurence can be recognized by BEI interrupt. If Interrupt service routine of detects such interrupt, it can read ERR_CAPT register to determine type and position of Error. If it detects Bit Error during Control field (see description of ERR_CAPT register), it may issue Flush. If CAN frames are properly read by FW, this work-around will probably not cause any CAN frames to be dropped.

ECO fixable: Likely yes, the change should be coupl of gates in output decoder of RX Buffer FSM. I estimate incremental synthesis will be able to do ECO. Plausibility depends on spare gates used in the design (amount, location, etc...) .

Next steps: Will be fixed in 2.6. If needed, RTL change can be pushed into 2v5-lts for ECO.

Fixed in: 2.5.1

Issue 5

Type: Bug

Severity: Medium

Description: When CTU CAN FD operates in MODE[BMM] = 1 (Bus Monitoring Mode), it will fail to receive frames when there are multiple nodes on the bus that are sending ACKs to each other. CTU CAN FD will be able to receive the frame only if there is only one other node sending the CAN Frame.

Consequence: Bus Monitoring Mode is not working properly. The use-case of recording what is happening on the CAN bus without affecting it is compromised. The bug does not affect CAN Compliance certication

Fixed in later development: Fixed in master branch towards 2.7 release in GIT commit: `https://gitlab.fel.cvut.cz/canbus/ctucanfd_ip_core/-/commit/386337e8dcbb2eddfcbd48b84ebabb4310ff2e1c`

ECO fixable: Likely yes. It is just a couple of gates.

Next steps: Can be ported to 2v5-lts and tagged with v2.5.2. An ECO can be made from v2.5.2. Alternatively, MODE[BMM] can be removed from the documentation.

Fixed in: 2.5.2 - Applied fix to be ECOed

Workaround in SW

If CTU CAN FD version 2.5 or 2.5.1 was manufactured, the MODE[BMM] bit does not work correctly. For these versions, it is recommended not to include MODE[BMM] in documentation of product that contains CTU CAN FD. It is still possible to get the functionality of "passive CAN bus logging", via SW workaround. To get this functionality, the integrating system can apply following workaround:

- Set MODE[ROM] = 1. Restricted operation mode make sure that CTU CAN FD does not get "lost" when it detected error condition. Since CTU CAN FDs shall not "affect the bus", it shall not transmit Error frames nor Overload frames.
- Set MODE[ACF] = 1. Since CTU CAN FD shall not "affect the bus", it shall not send dominant ACK bits.
- Not transmit any CAN frames.

These countermeasures should be sufficient for CTU CAN FD not to affect the bus. If there is not enough confidence, you can disconnect CAN_TX pin from physical propagation to the bus. There are number of different ways of how to do this:

- Switch the MCU IO pin to another functionality
- Use physical layer transceiver with "Bus monitoring" functionality.

If it is desirable to be able to record also CAN frames without acknowledge, you can set:

- MODE[STM]=1

Bibliography

[1] ISO 11898-1, 2015, Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling

[2] ISO 16845-1, 2016, Road vehicles — Controller area network (CAN) conformance test plan — Part 1: Data link layer and physical signalling