Technical note

SPC5x CAN errors management and bus off recovery

Introduction

This technical note aims at helping to detail the CAN errors management and the CAN bus off recovery procedure for the CAN/CAN FD controllers embedded on all SPC5x automotive powertrain/chassis/car body microcontrollers. The reader is supposed to be familiar with CAN/CAN FD protocol and internals of the device on SP5x MCU families.

This document provides a description of both CAN error state machine and transmit error counter (TEC)/receive error counter (REC). This document will also detail the CAN bus off recovery procedure providing a very basic example. This is applicable to the standard CAN and CAN FD.

1 CAN errors overview

The CAN protocol is designed for safe data transfers and provides mechanisms for error detection, signaling and self-diagnostics, including measures for fault confinement, which prevent faulty nodes from affecting the status of the network. The general measures for error detection are based on the capability of each node of monitoring broadcast transmissions over the bus, whether it is a transmitter or a receiver, and of signaling error conditions resulting from several sources. Corrupted messages are flagged by any node detecting an error. Such messages can be aborted and can be retransmitted automatically too.

The CAN protocol defines different types of errors and errors management techniques as showed in the next paragraphs.

More details about the CAN internals can be found inside the RM0452 reference manual of the device.

1.1 Error types

There are 5 different error types:

- **Bit error**: a unit that is sending a bit on the bus also monitors the bus. A bit error must be detected at that bit time, when the bit value that is monitored is different from the bit value that is sent. Exceptions are the recessive bits sent as part of the arbitration process or the ACK slot.
- **Stuff error:** a stuff error is detected at the 6th consecutive occurrence of the same bit in a message field that is subject to stuffing.
- **CRC error:** if the CRC computed by the receiver differs from the one stored in the message frame.
- **Form error:** when a fixed form bit field contains one or more illegal bits.

Note: For a receiver when a dominant bit, during the last bit of end of frame, is not treated as a form error then it is the possible cause of inconsistent message omission and duplicates.

- **Acknowledgement error:** detected by a transmitter if a recessive bit is found on the ACK slot.

1.2 CRC checks

The CAN manages the CRC check sum of a received message detecting if it is incorrect, this means the CRC of an incoming message does not match with the CRC calculated from the received data.

The CRC portion of the frame is obtained by selecting the input polynomial as stream of bits from the START OF FRAME bit (included) to the data field (if present) followed by 15 zeros. This polynomial is divided by the generator:

X15 + X14 + X10 + X8 + X7 + X4 + X3 + 1.

The remainder of the division is the CRC sequence portion of the frame.

1.3 Acknowledgement

The protocol requires that receivers acknowledge reception of the message by changing the content of the ACK field. The ACK field is 2-bit and contains the ACK slot and the ACK delimiter. In the ACK field the transmitting station sends two recessive bits. All receiver nodes that detect a correct message (after CRC checks), inform the sender by changing the recessive bit of the ACK slot into a dominant bit.

1.4 Bit stuffing

The protocol employs an NRZ type transmission method which requires bit stuffing and due to bit stuffing there cannot be more than 5 bits with the same polarity in the sequence arriving at the receiver. If six consecutive bits of the same value are found between the start of frame and the included CRC delimiter, the bit stuffing rule is violated and acknowledged a stuff error.

2 Fault confinement

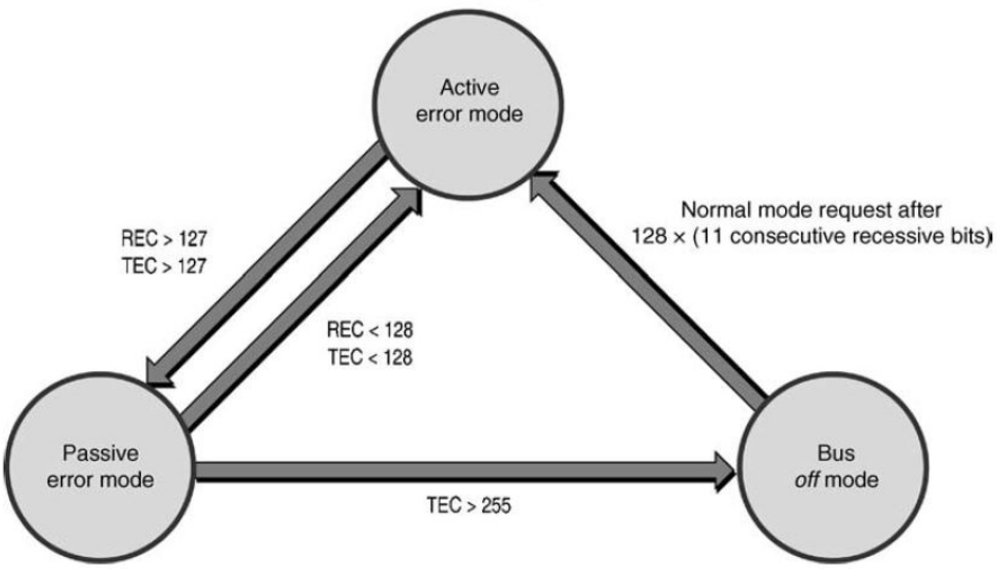
The CAN protocol has a fault confinement system so that each node sensing the bus can signal other nodes of wrong messages being transmitted and identify his own faulty condition so that they cannot affect the bus with their outputs.

The protocol assigns each node to one of the following three states:

- **Error active** units in this state are assumed to function properly. Units can transmit on the bus and signal errors with an ACTIVE ERROR FLAG.
- **Error passive** units are suspected of faulty behavior (in transmission or reception). They can still transmit on the bus, but their error signaling capability is restricted to the transmission of a PASSIVE ERROR FLAG.
- **Bus off** units are very likely to be corrupted and cannot have any influence on the bus. (e.g., their output drivers are switched off). Only a transmitting node can pass to bus off state.

Units change their state according to the value of two integer counters: TRANSMIT ERROR COUNT and RECEIVE ERROR COUNT, which give a measure of the likelihood of a faulty behavior on transmission and reception, respectively. The state transitions are represented in the Figure 1.

Figure 1. CAN state machine



The transition from bus off to active error state is subject to the additional condition that 128 occurrences of 11 consecutive recessive bits have been monitored on the bus (for example the node must correctly see on the bus a long idle condition to be enabled to start synchronizing again with other nodes).

The counters are updated according to the rules showed in the Figure 2:

Figure 2. CAN counter nodes

Node as RECEIVER	change	when node
RECEIVE ERROR COUNT	+ 1	detects an error unless a BIT ERROR during an ACTIVE ERROR FLAG or an OVERLOAD FLAG
	+ 8	detects a dominant bit as the first bit after sending an ERROR FLAG
	+ 8	detects a BIT ERROR while sending an ACTIVE ERROR FLAG or an OVERLOAD FLAG
	- 1	successful reception of a message if RECEIVE ERROR COUNT $\leq$ 127,
	any n 119 $\leq$ n $\leq$ 127	successful reception of a message if RECEIVE ERROR COUNT ¿ 127
	+ 8	node detects more than 7 consecutive dominant bits after sending an ACTIVE ERROR FLAG, PASSIVE ERROR FLAG or OVERLOAD FLAG (+8 for each additional 8 dominant bits)
Node as TRANSMITTER	**change**	**when node**
TRANSMIT ERROR COUNT	+ 8	sends an ERROR FLAG (with some exceptions, see [?])
	+ 8	detects a BIT ERROR while sending an ACTIVE ERROR FLAG or an OVERLOAD FLAG
	- 1	successful transmission of a message
	+ 8	detects more than 7 consecutive dominant bits after sending an ACTIVE ERROR FLAG, PASSIVE ERROR FLAG or OVERLOAD FLAG (+8 for each additional 8 dominant bits)

Note: In terms of pure software management, as detailed in the Section 3 Bus off management in MCAN controllers, the application must take care about the bus off event while other errors are managed by the controller updating related TEC and REC counters values to reduce the controller capability if communication problems are detected. The application can also modify his behavior in case TEC/REC values lead to passive error state, but this depends on the application needs.

3 Bus off management in MCAN controllers

Once the transmit error counter field in ECR register has reached the value of 255 or an overflow condition, then the MCAN controller enters in bus off status.

Note:

1. *The TEC field has 8-bit and so, it could happen that on the base of counter values the addiction of 8 units, due to the last error before entering bus off condition, will cause the counter overflow and so the counter is not updated also if the bus off condition is detected.*
2. *The bus off condition can be easily reached for debugging reasons with a short circuit between CAN-H and CAN-L lines while the controller is transmitting messages.*

The recovery from bus off is not automatically managed by the controller and so the user must start this doing simple operations.

The bus off recovery sequence consists of 128 occurrences of 11 consecutive recessive bits. MCAN controllers start sensing the bus looking for the recovery sequence when the INIT bit of control register (CCCR) is reset by the user. The bus off recovery sequence cannot be shortened by setting or resetting CCCR[INIT].

Summarizing, if the device raises a bus off condition, CCCR[INIT] is set stopping all bus activities. Once CCCR[INIT] has been cleared again by the software, the device will then wait for 129 occurrences of bus idle (129 x 11 consecutive recessive bits) before resuming on normal operation. At the end of the bus off recovery sequence, the error management counters will be reset, and so PSR.BO, ECR.TEC, and ECR.REC.

The small portion of code below shows the bus off recovery inside the ISR:

```
/* BUSOFF interrupt management */
if (canp->mcan->IR.B.BO == 1U) {
            /* Start bus off status exit sequence */
            canp->mcan->CCCR.B.INIT = 0;
}
```

Appendix A Acronyms, abbreviations and reference documents

Table 1. Acronyms and abbreviations

Terms	Description
CAN	Controller area network
FD	Flexible data rate
CAN-H / CAN-L	CAN high and low signals
ISR	Interrupt service routine

Table 2. Reference documents

Document name	Document title
RM0452	SPC58 H Line - 32 bit Power Architecture automotive MCU Triple z4 cores 200 MHz, 10 MBytes Flash, HSM, ASIL-D
Datasheets of SPC58EHx and SPC58NHx families.	

Revision history

Table 3. **Document revision history**

Date	Revision	Changes
07-Jul-2021	1	Initial release.

Contents