MCP2510

MCP2510 Rev. AA Silicon Errata Sheet

The MCP2510 parts you have received conform functionally to the Device Data Sheet (DS21291C), except for the anomalies described below.

All of the problems listed here will be addressed in future revisions of the MCP2510 silicon.

1. Module: Configuration Mode State Machine

The device may not enter Normal mode after a valid request to enter Normal mode (by writing the REQOP2:REQOP0 bits to '100').

Work Around

After requesting Normal mode, always read the OPMODE2:OPMODE0 bits (CANSTAT register) to confirm that the device entered into Normal mode. If these bits do not reflect Normal mode, reset the device to Configuration mode by writing the REQOP2:REQOP0 bits to '100' (CANCTRL register). Then request Normal mode (by writing the REQOP2:REQOP0 bits to '000'). Read the OPMODE2:OPMODE0 bits (CANSTAT register) to confirm that the device is in Normal mode.

2. Module: Configuration Mode State Machine

Configuring the REQOP2:REQOP0 bits (CANCTRL register) to enter Listen Only mode ('011') from any other mode may cause the device to enter Sleep Mode.

Work Around

First, configure the REQOP2:REQOP0 bits (CANCTRL register) to enter Loopback Mode ('010'). Then, configure the REQOP2:REQOP0 bits to enter Listen Only mode ('011').

3. Module: Error Flags

The TXWAR bit (EFLG register) will be set when the TEC register increments from 96 to 97 (decimal) and from 224 to 225 (decimal). The first condition is specified in the data sheet, but the second is not a specified operation.

Note 1: The EWARN bit (EFLG register) is automatically set when the TXWAR bit becomes set.

2: If the ERRIE bit (CANINTE register) is set, an interrupt will be generated. The microcontroller's Interrupt Service Routine (ISR) will need to address this.

Work Around

None

4. Module: Bus Timing Logic

Transmission of a pending message may not start immediately after the 11 recessive bit times following the previous message frame's ACK slot bit. A delay of between zero and two additional bit times may occur before transmission of the message.

This may only occur when a message is queued for transmission (by setting the TXREQ bit (TXBnCTRL register) for the buffer) while receiving a message. If the message is received (i.e. transferred to a receive buffer) by passing the mask and filter values, a delay of between zero and two additional bit times occurs before the pending message is transmitted. If the received message is rejected (based on the mask and filter values), no additional bit time delay occurs.

This additional message transmission delay is a function of the total number of OSC1 clocks per bit time. By setting the total number of OSC1 clocks per bit time to 26 or greater, this delay will always be zero and the issue does not exist. If the number of OSC1 clocks per bit time is between 18 and 24, the delay will be one bit time, and if the number of OSC1 clocks per bit time is less than 18, the delay will be two bit times.

Work Around

Provide 26 or more OSC1 clocks per bit time. To provide 26 or more OSC1 clocks per bit time, the user should do one or more of the following:

- Use a higher speed oscillator for a given nominal bit time (this allows the value in BRP5:BRP0 to be increased, thereby increasing the time of TQ)
- Increase the BRP5:BRP0 value
- Increase the number of TQ per nominal bit time

The number of OSC1 clocks per bit time is determined by two things:

- The baud rate prescaler setting
- The total number of TQ in a bit time

The baud rate prescaler (BRP5:BRP0) bits control the number of OSC1 clocks per TQ. The minimum number of OSC1 clocks per TQ is 2 (for BRP5:BRP0 = 000000). This number can be increased by increasing the baud rate prescaler setting in the CNF1 register. A BRP5:BRP0 value of 000001 doubles the number of OSC1 clocks per TQ to 4, a BRP5:BRP0 value of 000010 doubles this again to 8, etc.

The total number of TQ in a bit time is a simple multiplier to the number of OSC1 clocks per TQ which yields the number of OSC1 clocks that occur during a bit time. This is shown in Equation 1.

EQUATION 1: Calculation of OSC1 Clocks

$$\#OSC1\ clocks = 2 \bullet (BRP5:BRP0 + 1) \bullet \frac{\#\ T_Q}{bit\ time}$$

Table 1 shows common CAN bus speeds and how to set up the other parameters to meet 26 OSC1 clocks per bit time, so no additional delays occur.

5. Module: Receive Logic

Data from a Received message may be corrupted when three messages (Message #1, #2, and #3) are received (back-to-back-to-back). This corruption occurs when all messages are destined for the same receive buffer, and message #2 and #3 both match the same filter.

Since Message #1 is not serviced until message #3 begins to be received into the Message Acceptance Buffer (MAB), Message #2 (which is currently in the MAB), starts to be overwritten by the reception of Message #3). Once the RXnIF bit is cleared, the contents of the MAB is loaded into the Receive Buffer. The value loaded into the Receive Buffer will contain part of message #2 and part of Message #3.

Work Around

The RXnOVR bit (EFLG register) should be checked before accepting each message. If this bit is set, the message that is currently in the receive buffer should be discarded (since the data could be corrupted), and the microcontroller can then respond appropriately.

TABLE 1: CAN BUS SPEEDS AND OSC1 CLOCKS PER BIT TIME

CAN Bus Speed (Bits/Second)	Minimum OSC1 Frequency (MHz)	Baud Rate Prescaler (BRP5:BRP0)	# of TQ's per Bit Time	Clocks per Bit Time	Comment
125K	4	000000	16	32	No additional delay occurs
125K	4	000001	8	32	No additional delay occurs
250K	4	000000	8	16	Additional delay occurs
250K	8	000000	16	32	No additional delay occurs
250K	8	000001	8	32	No additional delay occurs
500K	8	000000	8	16	Additional delay occurs
500K	16	000000	16	32	No additional delay occurs
500K	16	000001	8	32	No additional delay occurs

6. Module: Receive Logic

Invalid data will be transmitted any time there are two messages (Message #1 and #2) received and one message (Message #3) transmitted (back-to-back-to-back). This invalid data will be transmitted when Message #1 and Message #2 are destined for the same receive buffer, and if the 1st received message has not been serviced (i.e. RXnIF cleared) before the start of transmission of Message #3.

Since Message #1 does not get serviced until message #3 is being transmitted, the receive flag is cleared and message #2 begins moving into the receive buffer. At this point, the transmitting message malfunctions and erroneous data is transmitted.

Work Around

Check for a transmitting message before clearing the RXnIF bit (CANINTF register) and wait to clear the receive interrupt until after the appropriate TXREQ bit is cleared (signifying the message is done transmitting). This can be done via an SPI™ Read Status command.

7. Module: Receive Buffer 0

The Receive Buffer 0 FILHIT0 bit (RXB0CTRL register) does not function. This bit is stuck clear, and can not be used to determine which filter match occurred to receive the Message into Receive Buffer 0.

Work Around

None

8. Module: Receive Buffer 0

When the device is in Listen Only mode, the Receive Buffer 0 RX0OVR bit (EFLG register) is set whenever an error occurs on the CAN bus.

Work Around

None

9. Module: SPI Interface

The Output Valid from Clock Low timing specification (T_V) for the SPI interface may exceed the maximum specification shown in the data sheet. The new tested specification is shown in Table 3.

This can occur when the SPI "Read" command (0x03) is used for sequentially reading multiple bytes from certain registers. These registers are shown in Table 2. Only sequential reads of these registers will cause the T_V specification to be exceeded. Byte reads will not affect the T_V specification.

Note:	Sequential reads of all other SRAM registers, including the Transmit and Receive Buffers, are unaffected.

Work Around

There are two potential work arounds that can be implemented:

Ensure that the SPI clock frequency is less than 4.25 MHz.

If the SPI clock is greater than 4.25 MHz, check the Data Setup Time (T_{SU}) requirements of the microcontroller's SPI port to determine if T_V is an issue.

If T_V is an issue, use byte reads (single byte) when reading multiple bytes of the registers listed in Table 2. Typical applications configure the MCP2510 by programming most of these registers during initialization and subsequent sequential reads are not required. However, if it becomes necessary to read these registers, they may be read individually (as opposed to sequentially) without violating the Data Sheet T_V specification.

TABLE 2: REGISTERS AFFECTED

Register	
Name	**Address**
RXFnSIDH RXFnSIDL RXFnEID8 RXFnEID0	0x00 - 0x0B 0x10 - 0x1B
RXMnSIDH RXMnSIDL RXMnEID8 RXMnEID0	0x20 - 0x27
CNF2	0x29
CNF1	0x2A
CANSTAT	0xnE (1)
CANCNTL	0xnF (1)
TEC	0x1C
REC	0x1D
CANINTE	0x2B
CANINTF	0x2C
EFLG	0x2D

Note 1: $0 \le n \le 7$

SPI™ is a trademark of Motorola Inc.

TABLE 3: TESTED TIMING SPECIFICATIONS

Parm No.	Sym.	Characteristic	Specification				Units	Condition
			Tested		Data Sheet			
			Min	Max	Min	Max		
—	TV	Output Valid from Clock Low						When sequentially reading multiple bytes from registers shown in Table 2.
			—	115	—	90	nS	VDD = 4.5V to 5.5V (ITEMP)
			—	115	—	115	nS	VDD = 4.5V to 5.5V (ETEMP)
			—	180	—	180	nS	VDD = 3.0V to 4.5V

10. Module: Receive Buffer 1

The Receive Buffer 1 RXRTR bit (RXB1CTRL register) does not function properly. After the RXRTR bit has been set by an RTR event, it can not be cleared. This includes the reception of subsequent non-RTR messages into Receive Buffer 1.

If a CAN bus system does not use RTR messages, then this is not an issue.

Work Around

If a CAN bus system uses Remote Transmission Request messages for Receive Buffer 1. The RTR status can be read directly from the appropriate Receive Buffer 1 register (instead of the RXB1CTRL register).

For a Standard RTR Frame, the RTR status is indicated by the SRR bit (RXB1SIDL register). For an Extended RTR Frame, the RTR status is indicated by the RTR bit (RXB1DLC register).

11. Module: CAN Engine

The MCP2510, as a receiver, may detect a Start Of Frame (SOF) after a long bus idle time and then generate an error on the message, causing the message to be retransmitted. There is only a small window within the bit time in which this can occur and after the initial error is sent by the MCP2510, the resent message will be received successfully and normal bus communications will resume. The following sections detail the conditions that must occur for the MCP2510 to error, including the portion of the bit time affected and the bus idle time required for the error to occur.

CAUSE OF RECEIVE ERROR

A receive error will occur when the MCP2510, as a receiver, detects a SOF (recessive-to-dominant) at a certain position before the sample point. As alluded to earlier, each node operates asychronously to it's own bit timing which is a function of the oscillator. During a bus idle, the nodes do not have any recessive-to-dominant edges to synchronize to and soon get out of synchronization with each other. This is not usually a problem because the SOF hard synchronizes the receivers to the message.

Figure 1 illustrates the window in which the MCP2510 will mis-sample the SOF and generate an error later in the message. This Window of Susceptibility (WS) occurs 1TQ before the sample point and varies in width depending on the Baud Rate Prescaler (BRP) and the number of Time Quanta (TQ) in the bit and is discussed in detail later.

Probability of Receive Error

The probability that the MCP2510 will mis-sample a SOF and generate an error depends on two main factors:

1. Bit Time Affected?

As Figure 1 illustrates, there is a small timeframe within a bit time in which a recessive-to-dominant edge will cause the SOF to be incorrectly sampled. This window of Susceptibility as a percentage of the bit time is a function of the Baud Rate Prescaler (BRP) and number of Time Quanta (TQn) and is defined as:

$$WS = \left\lfloor \frac{(BRP + 0.5)}{2(TQn)(BRP)} \right\rfloor$$

The WS can also be regarded as the "probability" that the SOF will occur in the region of the bit time that causes an error, providing enough bus idle time has elapsed to allow the nodes to become out of synchronization (bus idle requirements are covered next).

Figure 2 graphs the WS (Bit Time Affected) for all of the BRP and number of TQ settings. As the graph shows, the worst case WS occurs when the MCP2510 is configured to TQ = 8 and BRP = 1. In this case, the percentage of the bit time affected is about 9.3%. This means that, providing sufficient oscillator drift has occurred during bus idle, there is a 9.3% probability that the MCP2510 will incorrectly sample the SOF and generate an error frame. (See item 2 below).

2. How Much Bus Idle Time is Required to Create the Risk?

While the Window of Susceptibility exists for each bit time, the probability of the MCP2510 incorrectly sampling the SOF depends on how far the MCP2510, as a receiver, has become out of synchronization with the transmitting node. Recall in Figure 1 that for the MCP2510 to incorrectly sample the SOF, the recessive to dominant edge of the SOF must occur 1TQ before the sample point within a window depending on the BRP setting and number of TQs.

During long bus idle times, CAN nodes tend to get out of synchronization due to less than perfect oscillator tolerances. Eventually, the nodes may get far enough out of synchronization to allow the SOF to occur in the WS of the MCP2510's internal bit time, causing an error.

Two factors must be known to calculate how long it will take for the nodes to get far enough out of synchronization to enable the probability of a node incorrectly sampling the SOF. The two factors are the oscillator tolerances and the sample point position within the bit time. Figure 3 charts the number of consecutive recessive bits required before the probability of mis-sampling is activated. The chart uses oscillator tolerances from 0.01% (100 ppm) for crystal oscillators to 1.50% (15000 ppm) for ceramic resonators.

Ramifications:

- Since the receive error causes the message to be retransmitted, no data is lost.
- The latency time from the original transmit until a valid received message is increased by the error frame delay.
- Bus traffic increases slightly, but since the receive error issue can only occur after long bus idle times, the bus loading is very light to begin with and will not significantly affect message throughput.

FIGURE 1: HARD SYNCHRONIZATION

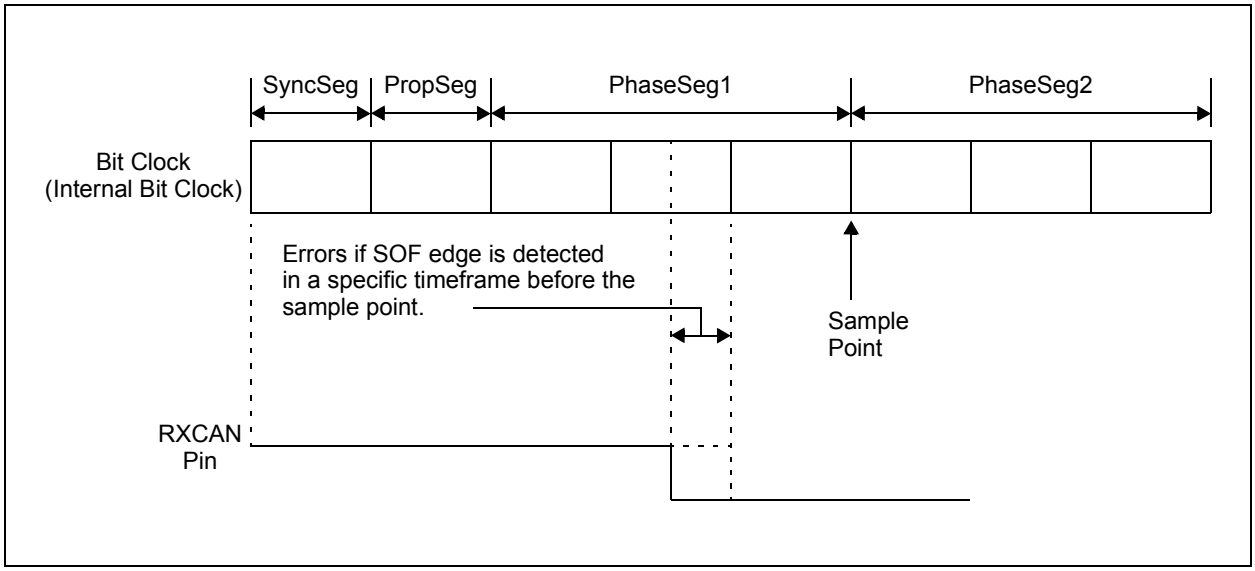


FIGURE 2: AFFECTED BIT TIME

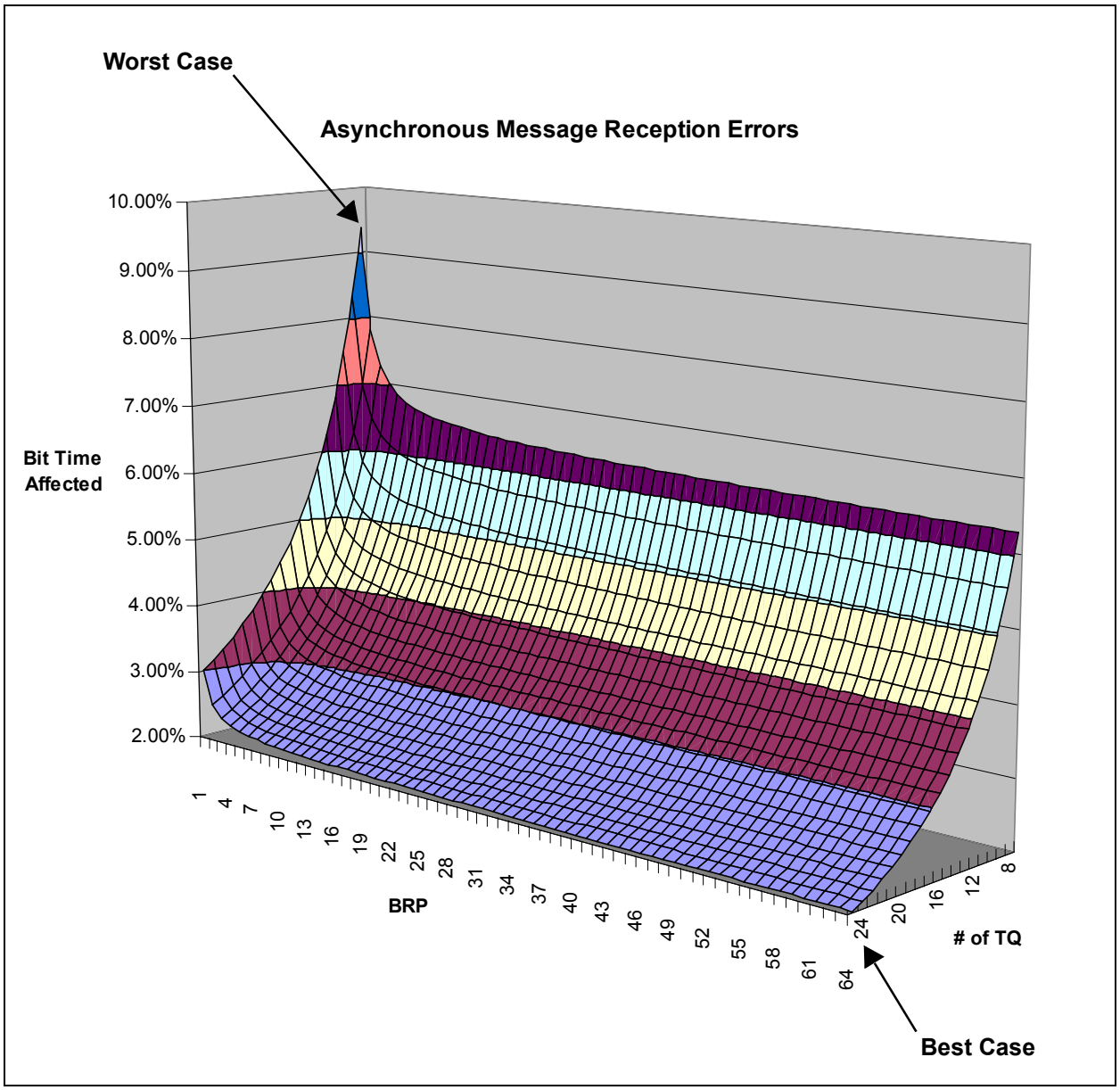
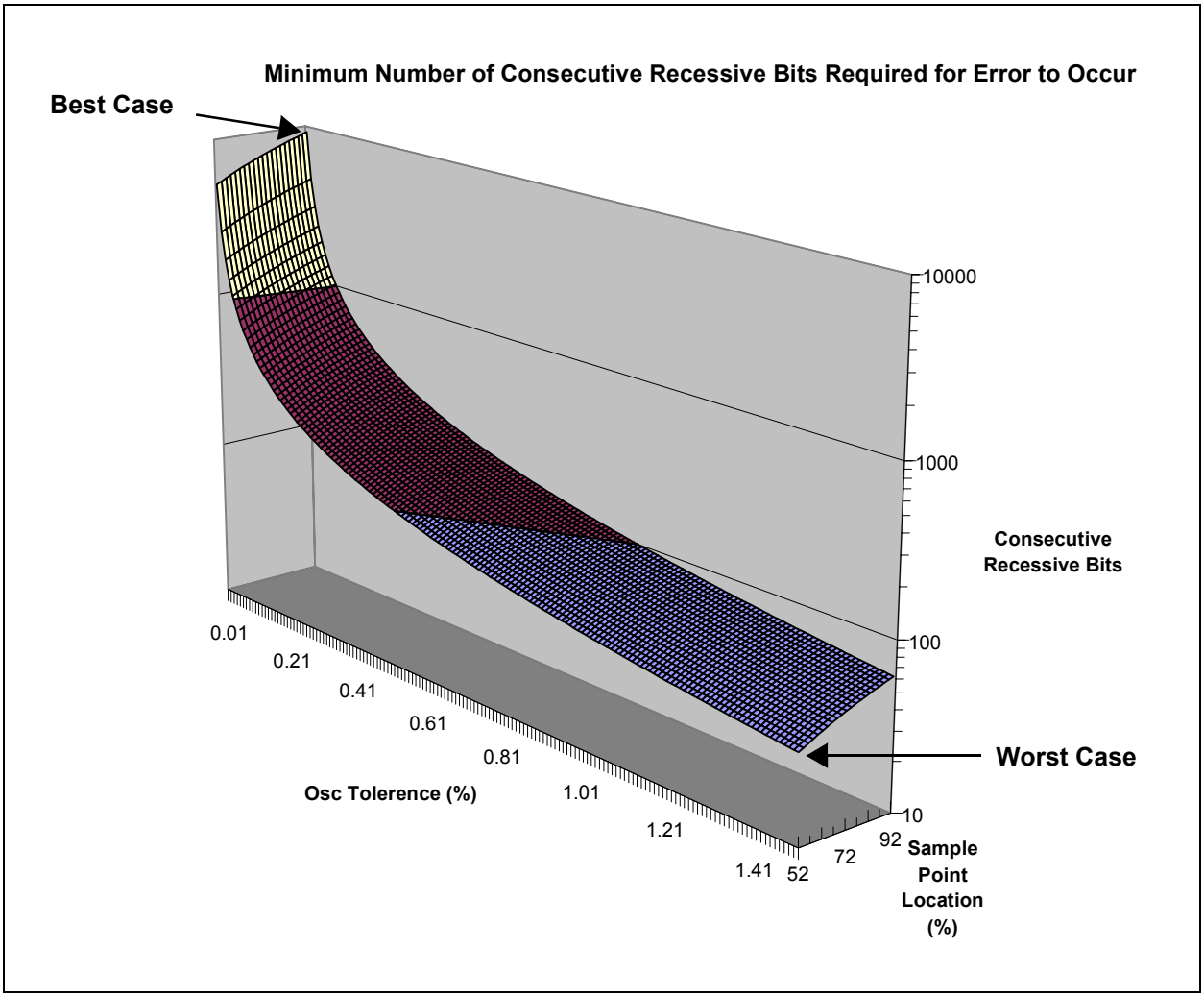


FIGURE 3: CONSECUTIVE RECESSIVE BITS REQUIRED TO ENABLE PROBABILITY



Work Around

It is possible to virtually eliminate the potential for false errors, even in very light bus loads with long bus idle times, simply by using crystal oscillators with low ppm tolerances and manipulating the bit timing.

Example:

Required CAN bit rate = 500 kb/s
Fosc = 16 MHz

Using a 100 ppm crystal puts the consecutive recessive bit requirement to activate the probability at approaching 10,000 bit times (20 ms).

There are two ways to get 500 kb/s:

1. #TQ = 8, BRP = 2.
   This puts the probability approaching **8%**
2. #TQ = 16, BRP = 1.
   This puts the probability around **3.5%**

12. Module: CAN Engine

Under certain circumstances, the CAN module will not transition to “error passive” due to a bus error.

As a transmitter: If the CAN module is “error active” and attempts to drive a dominant error on the bus, and properly samples a dominant state on the bus, the TEC error counter increments and the device transitions from “error active” to “error passive” (TEC ≥128).

If the device attempts to drive a dominant error on the bus and samples recessive states on the bus, it will not enter “error passive” even if the error counter exceeds the value for the “error passive” state. The module will however, transition to the “bus off” state if the transmit error counter exceeds the “bus off” threshold (TEC ≥255).

As a receiver: If the CAN module is “error active” and attempts to drive a dominant error on the bus, and properly samples a dominant state on the bus, the TEC error counter increments and the device transitions from “error active” to “error passive” (TEC ≥128).

If the device attempts to drive a dominant error on the bus and samples recessive states on the bus, it will not enter “error passive” even if the error counters exceed the value for the “error passive” state.

Normal operation will resume once the dominant error frame is successfully sent (i.e., sampled correctly on the receive input).

Work Around

None

13. Module: CAN Engine

Under one specific circumstance, the CAN engine may not receive a message on the bus. The CAN protocol states that if a receiver detects a dominant state during the third bit of Inter-Frame Space (IFS), it will interpret it as a SOF. This scenario can only occur if the oscillator tolerances between nodes are large enough to allow the nodes to get out of synchronization during the error frame.

During normal communications, the MCP2510 CAN engine interprets this situation correctly. However, if this scenario occurs immediately following an error frame, the CAN module will not recognize the SOF occurring during the third bit of the IFS (and will in fact, completely ignore the message)

Work Around

None.

14. Module: CAN Engine

The device may send a standard remote frame instead of the intended standard data frame under one specific configuration.

The device may randomly send a standard remote frame instead of the intended standard data frame, if the CAN bit timing is configured with phase segment 2 equal to 2TQ. This phenomenon does not occur while sending extended data frames or if phase segment 2 is greater than 2TQ.

Work Around

Any of the following work arounds will solve the issue:

- Set Phase Segment 2 > 2TQ.
- If receiving messages from a MCP2510 that transmits standard data frames and phase segment 2 = 2TQ, check if RTR bit was set indicating a remote frame.
- Use extended data frames.

CLARIFICATIONS/CORRECTIONS TO THE DATA SHEET:

In the Device Data Sheet (DS21291C), the following clarifications and corrections should be noted.

None

REVISION HISTORY

Rev. D Document

1: 3rd revision of this document.

NOTES:

NOTES: