The Configuration of the CAN Bit Timing

6th International CAN Conference
2nd to 4th November, Turin (Italy)

Florian Hartwich, Armin Bassemir
Robert Bosch GmbH, Abt. K8/EIS
Tübinger Straße 123, 72762 Reutlingen

Abstract:

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detaile#d knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

The purpose of this paper is to describe the CAN bit synchronization algorithm and the parameters which have to be considered for the proper calculation of the CAN bit time.

Some critical cases of resynchronizations will be demonstrated by waveforms simulated in the VHDL CAN Reference Model's testbench.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

1. Introduction

CAN supports bit rates in the range of lower than 1 kBit/s up to 1000 kBit/s. Each member of the CAN network has its own clock generator, usually a quartz oscillator. The timing parameter of the bit time (i.e. the reciprocal of the bit rate) can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods (f_{osc}) may be different.

The frequencies of these oscillators are not absolutely stable, small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 1). The Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 1). The length of the time quantum ($t_{q)}$, which is the basic time unit of the bit time, is defined by the CAN controller's system clock f_{sys} and the Baud Rate Prescaler (BRP) : t_q = BRP / f_{sys}. Typical system clocks are : $f_{sys} = f_{osc}$ or $f_{sys} = f_{osc}/2$.

The Synchronization Segment Sync_Seg is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync_Seg and the Sync_Seg is called the phase error of that edge. The Propagation Time Segment Prop_Seg is intended to compensate for the physical delay times within the CAN network. The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample Point. The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

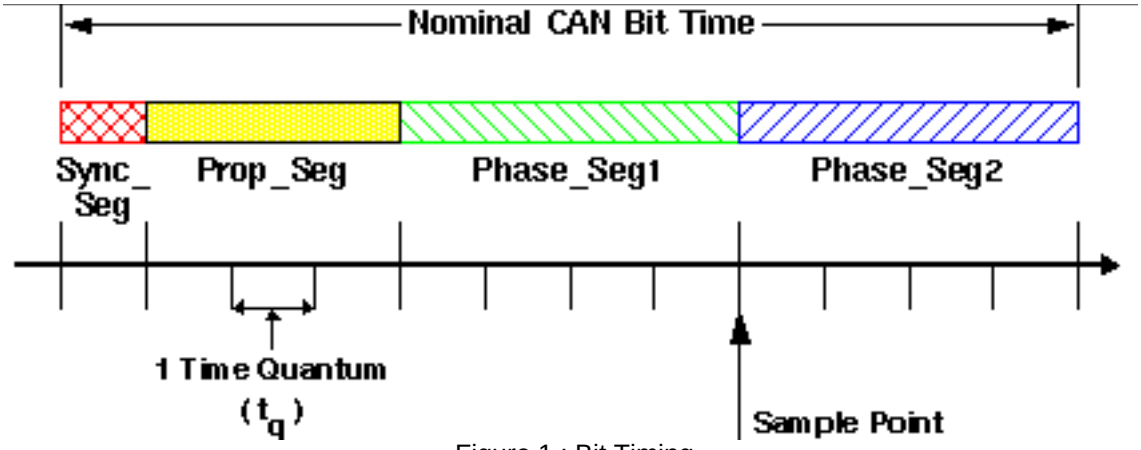

Figure 1 : Bit Timing

Parameter	Range	Remark
BRP	[1 .. 32]	defines the length of the time quantum t_q
Sync_Seg	1 t_q	fixed length, synchronization of bus input to system clock
Prop_Seg	[1 .. 8] t_q	compensates for the physical delay times
Phase_Seg1	[1 .. 8] t_q	may be lengthened temporarily by synchronization
Phase_Seg2	[1 .. 8] t_q	may be shortened temporarily by synchronization
SJW	[1 .. 4] t_q	may not be longer than either Phase Buffer Segment
This table describes the minimum programmable ranges required by the CAN protocol		

Table 1 : Parameters of the CAN Bit Time

A given bit rate may be met by different bit time configurations, but for the proper function of the CAN network the physical delay times and the oscillator's tolerance range have to be considered.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

2. Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus will be out of phase with the transmitter of that bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's non-destructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in Figure 2 shows the phase shift and propagation times between two CAN nodes.

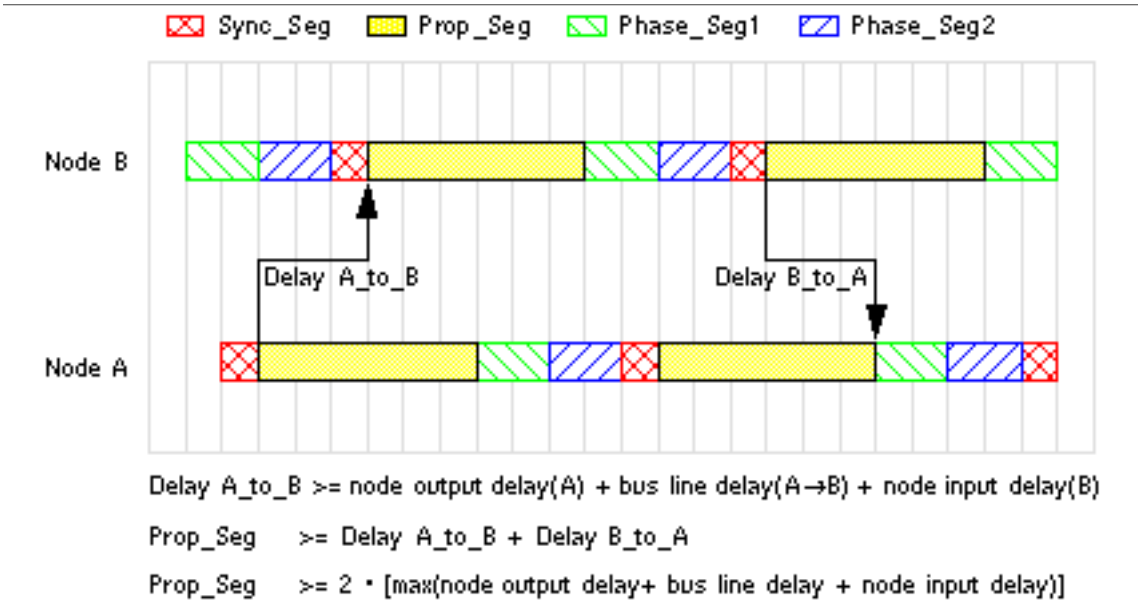

Figure 2 : The Propagation Time Segment

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A_to_B) after it has been transmitted, B's bit timing segments are shifted with regard to A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay(B_to_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

The error occurs only when two nodes arbitrate for the CAN bus that have oscillators of opposite ends of the tolerance range and that are separated by a long bus line; this is an example of a minor error in the bit timing configuration (Prop_Seg to short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 t_q, requiring a longer Prop_Seg.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

3. Phase Buffer Segments and Synchronization

The Phase Buffer Segments (Phase_Seg1 and Phase_Seg2) and the Synchronization Jump Width (SJW) are used to compensate for the oscillator tolerance. The Phase Buffer Segments may be lengthened or shortened by synchronization.

Synchronizations occur on edges from recessive to dominant, their purpose is to control the distance between edges and Sample Points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous Sample Point. A synchronization may be done only if a recessive bit was sampled at the previous Sample Point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise the distance between edge and the end of Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist : Hard Synchronization and Resynchronization. A Hard Synchronization is done once at the start of a frame; inside a frame only Resynchronizations occur.

- Hard Synchronization

  After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

- Bit Resynchronization

  Resynchronization leads to a shortening or lengthening of the bit time such that the position of the sample point is shifted with regard to the edge.

  When the phase error of the edge which causes Resynchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

  When the phase error of the edge which causes Resynchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

When the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of Hard Synchronization and Resynchronization are the same. If the magnitude of the phase error is larger than SJW, the Resynchronization cannot compensate the phase error completely, an error of (phase error - SJW) remains.

Only one synchronization may be done between two Sample Points. The synchronizations maintain a minimum distance between edges and Sample Points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize "hard" on the edge transmitted by the "leading" transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The "leading" transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently "take the lead" and that are differently synchronized to the previously "leading" transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that "takes the lead" in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator's clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator's tolerance range.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

The examples in Figure 3 show how the Phase Buffer Segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a “late” edge, the lower drawing shows the synchronization on an “early” edge, and the middle drawing is the reference without synchronization.

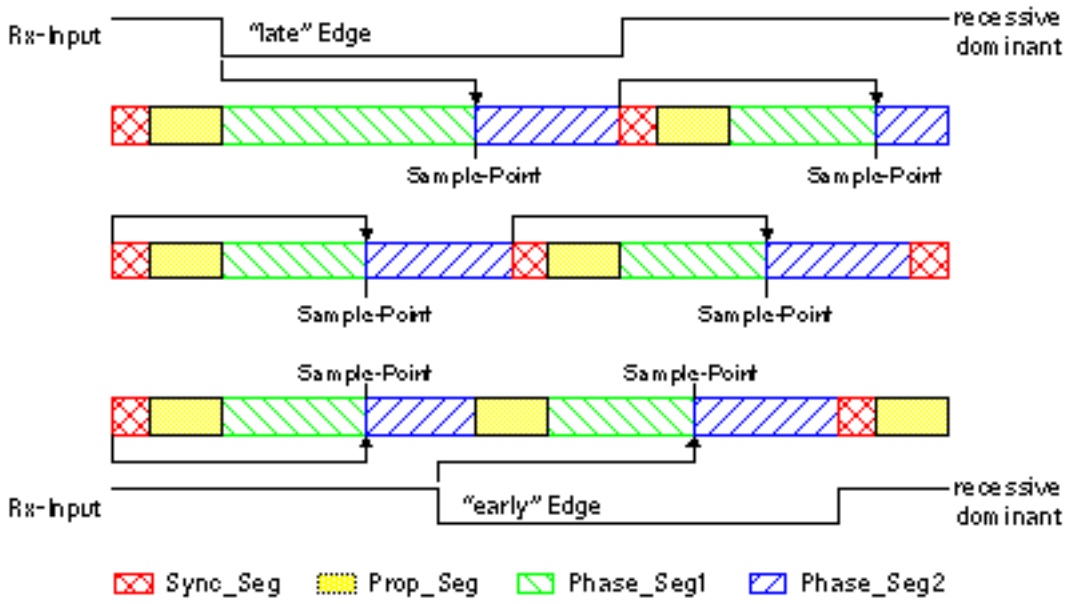

Figure 3 : Synchronization on “late” and “early” Edges

In the first example an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is “late” since it occurs after the Sync_Seg. Reacting to the “late” edge, Phase_Seg1 is lengthened so that the distance from the edge to the Sample Point is the same as it would have been from the Sync_Seg to the Sample Point if no edge had occurred. The phase error of this “late” edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example an edge from recessive to dominant occurs during Phase_Seg2. The edge is “early” since it occurs before a Sync_Seg. Reacting to the “early” edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the Sample Point is the same as it would have been from an Sync_Seg to the Sample Point if no edge had occurred. As in the previous example, the magnitude of this “early” edge’s phase error is less than SJW, so it is fully compensated.

The Phase Buffer Segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

In these examples, the bit timing is seen from the point of view of the CAN implementation’s state machine, where the bit time starts and ends at the Sample Points. The state machine omits Sync_Seg when synchronizing on an “early” edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in Figure 4 show how short dominant noise spikes are filtered by synchroniza-tions. In both examples the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the Synchronization Jump Width is greater than or equal to the phase error of the spike’s edge from recessive to dominant. Therefore the Sample Point is shifted after the end of the spike; a recessive bus level is sampled.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

In the second example, SJW is shorter than the phase error, so the Sample Point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

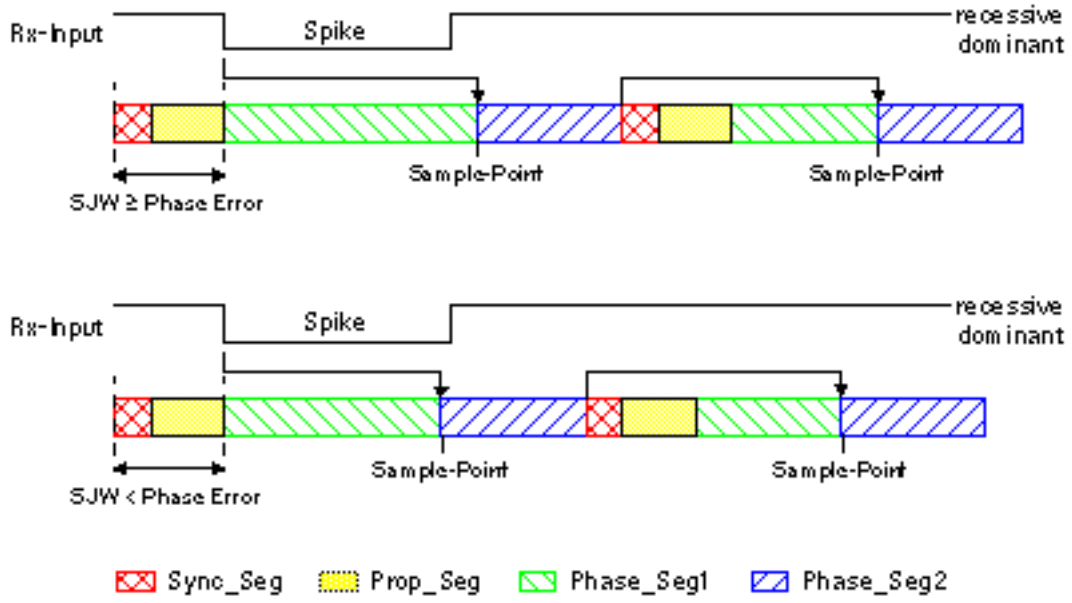

Figure 4 : Filtering of Short Dominant Spikes

4. Oscillator Tolerance Range

The oscillator tolerance range was increased when the CAN protocol was developed from version 1.1 to version 1.2 (version 1.0 was never implemented in silicon). The option to synchronize on edges from dominant to recessive became obsolete, only edges from recessive to dominant are considered for synchronization. The only CAN controllers to implement protocol version 1.1 have been Intel 82526 and Philips 82C200, both are superseded by successor products. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator's frequency f_{osc} around the nominal frequency f_{nom} with $(1 - df) \bullet f_{nom} \le f_{osc} \le (1 + df) \bullet f_{nom}$ depends on the proportions of Phase_Seg1, Phase_Seg2, SJW, and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met) :

$$\text{I:} \quad df \le \frac{min(\text{Phase_Seg1}, \text{Phase_Seg2})}{2 \bullet (13 \bullet \text{bit_time} - \text{Phase_Seg2})}$$

$$\text{II:} \quad df \le \frac{\text{SJW}}{20 \bullet \text{bit_time}}$$

It has to be considered that SJW may not be larger than the smaller of the Phase Buffer Segments and that the Propagation Time Segment limits that part of the bit time that may be used for the Phase Buffer Segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8 µs) with a bus length of 40 m.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

5. Configuration of the CAN Protocol Controller

In most CAN implementations, the bit timing configuration is programmed in two register bytes. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one register, SJW and BRP are combined in the other register (see Figure 5).

In these bit timing registers, the four components TSEG1, TSEG2, SJW, and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, e.g. SJW (functional range of [1..4]) is represented by only two bits.

Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

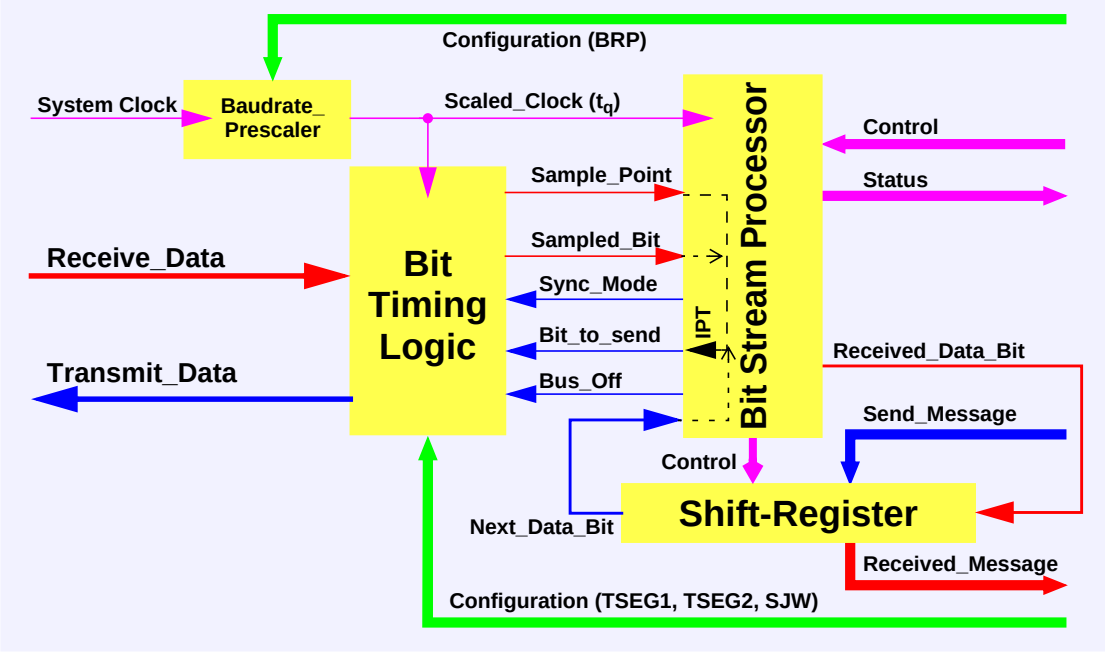

Figure 5 : Structure of the CAN Protocol Controller

The data in the bit timing registers are the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRP) defines the length of the time quantum, the basic time unit of the bit time; the Bit Timing Logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the BTL state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the Bit Stream Processor (BSP) state machine is evaluated once each bit time, at the Sample Point.

The Shift Register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the Sample Point and processes the sampled bus input bit. The time after the Sample point that is needed to calculate the next bit to be sent (e.g. data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT).

The IPT is application specific but may not be longer than 2 t_q. Its length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect the bus timing.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

6. Measurement of Node Delay Times

While the measurement of the bus line delay is trivial, the determination of the node output delay and the node input delay (see Figure 6) requires a more elaborate approach. The delays are measured between the node's protocol controller state machine and node's connection to the CAN bus line.

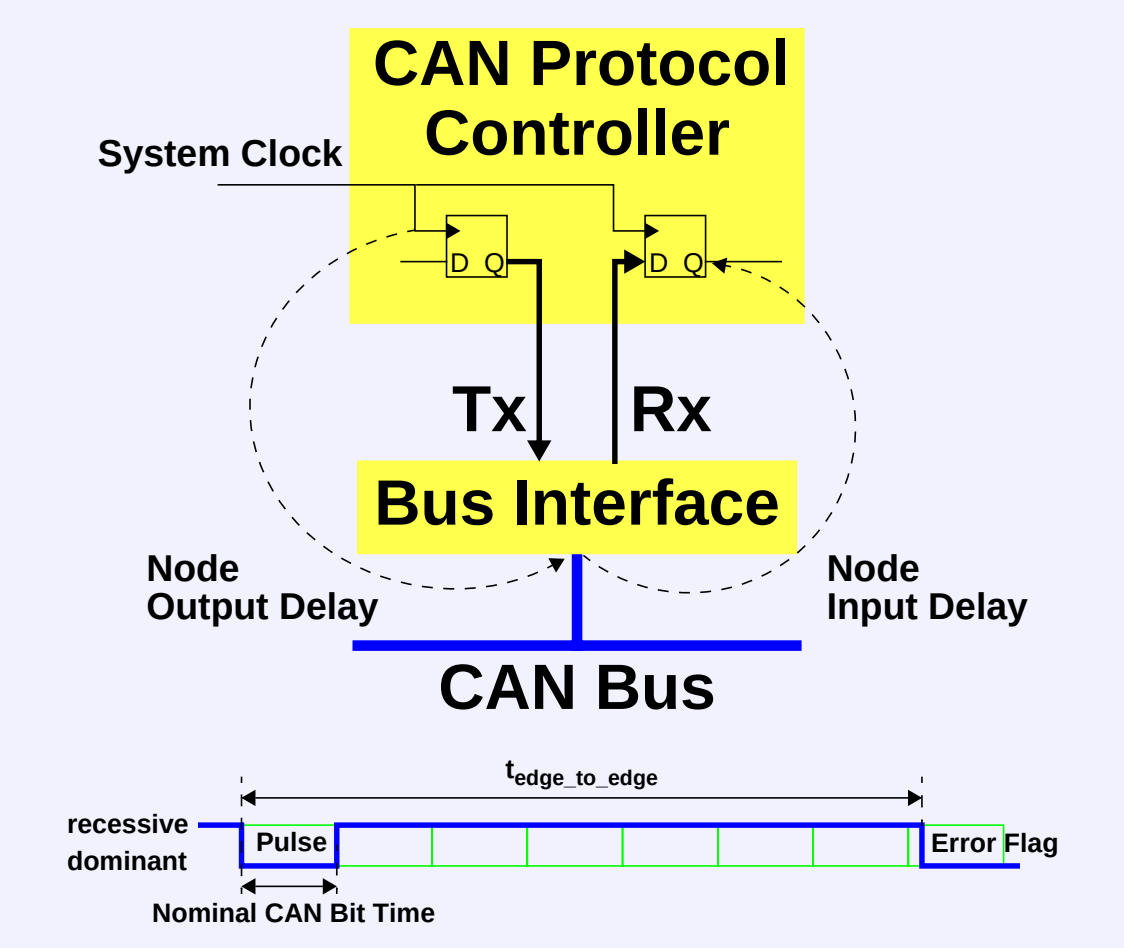

Figure 6 : Input and Output Delay Times.

For the calculation of the CAN bit timing configuration, only the sum of the node output delay and the node input delay, t_{node}, needs to be known. The measurement is done by applying a pulse of dominant bus level (length one nominal CAN bit time) at the CAN bus input of an error active CAN node that is in idle state. The CAN node will regard the dominant bit as Start of Frame and will perform a hard synchronization. At the sixth recessive bit after the dominant bit, the CAN node will detect a stuff error and will respond with an active error flag.

The time measured from the beginning of the externally applied dominant bit to the beginning of the dominant active error flag is $t_{edge_to_edge}$.

The actual length of $t_{edge_to_edge}$ is composed of the node output delay, the node input delay, of a multiple of the nominal CAN bit time, and of the clock synchronization delay (see Figure 7). The clock synchronization delay depends on the clock phases of the pulse generator and the CAN node. In the CAN bit time, this clock synchronization delay is compensated by Sync_Seg, so it has to be eliminated when measuring t_{node}. This is done by adjusting the clock phases.

The clock phases of pulse generator and CAN node have to be adjusted in order to minimize $t_{edge_to_edge}$. Then t_{node} is calculated by : $t_{node} = \min(t_{edge_to_edge}) - 7 \bullet$ nominal CAN bit time.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

The waveforms of Figure 7 have been generated by simulating Bosch's IP CAN module in the VHDL CAN Reference Model's testbench. RECEIVE_DATA and TRANSMIT_DATA are the Rx and Tx signals between the CAN controller and its bus interface, BUSMON in the input signal synchronized to the system clock CLK, BUSMOND is BUSMON sampled at the Sample Point (rising CLK edge when SAMPLE_POINT is '1'), BUSDRIVE is the next bit to be sent, and REC is the CAN Protocol Controller's Receive Error Counter.

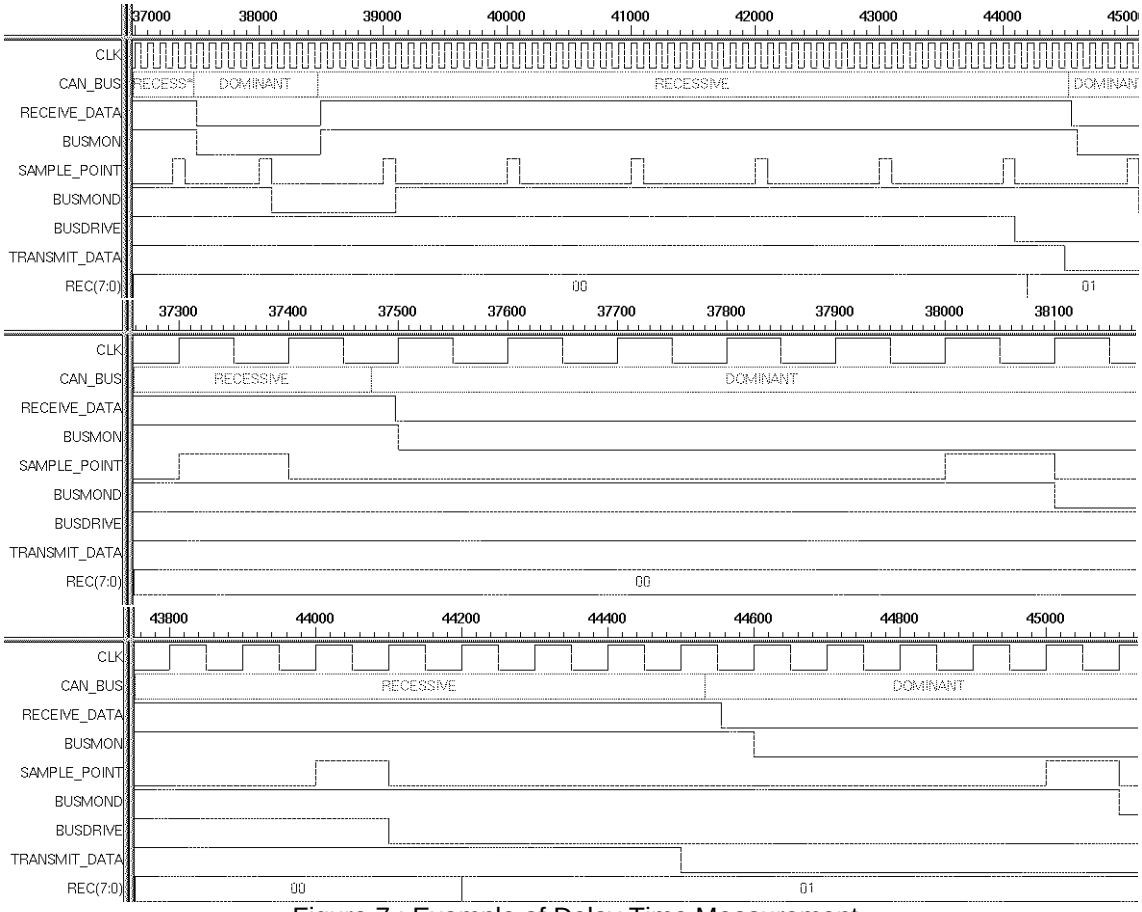

Figure 7 : Example of Delay Time Measurement

The first waveform shows the whole test sequence. A dominant pulse is applied and causes a Hard Synchronization. After the sixth consecutive recessive bit has been sampled, a stuff error is detected, the Receive Error Counter is incremented, and an active error flag is started.

The second waveform shows the delay time between the CAN bus line and RECEIVE_DATA. Between RECEIVE_DATA and BUSMON is only the register's setup time.

The third waveform shows the delay time between TRANSMIT_DATA and the CAN bus line. BUSDRIVE, the output of the BSP state machine, changes immediately at the sample point. The Information Processing Time is 0 tq. The time between BUSDRIVE and TRANSMIT_DATA is Phase_Seg2.

In this example, $t_{edge_to_edge}$ is (44533 - 37475) ns = 7058 ns. With a nominal CAN bit time of 1000 ns, t_{node} is 7058 ns - 7000 ns = 58 ns.

The fourth waveform in Figure 8, a measurement with unadjusted clock phases, shows the additional delay between RECEIVE_DATA and BUSMON caused by the synchronization of the CAN bus input signal to the system clock CLK. The pulse generator's clock phase has to be shifted about 45 ns in order to minimize $t_{edge_to_edge}$.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.

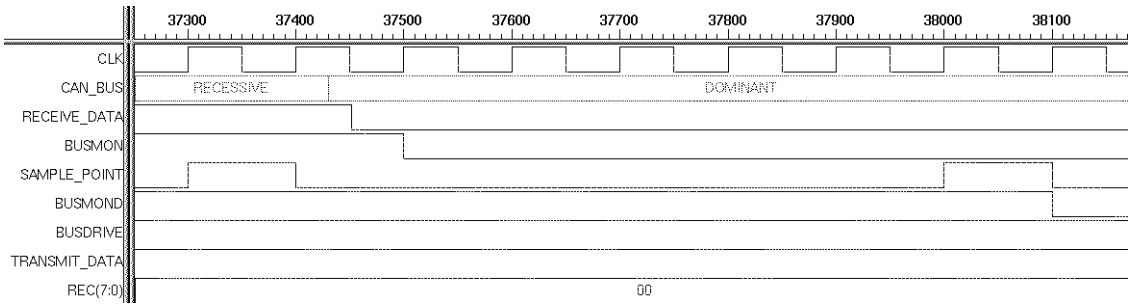

Figure 8 : Measurement with Maladjusted Clock Phases.

7. Conclusion

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 8 to 25 time quanta, the length of the time quantum t_q is defined by the Baud Rate Prescaler with t_q = (Baud Rate Prescaler)/f_{sys}. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The Sync_Seg is 1 t_q long (fixed), leaving (bit time – Prop_Seg – 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of [0..2] t_q.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in section 4.

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by that node with the lowest tolerance range.

In most implementations, the resulting configuration is written into two register bytes :

- Bit Timing Register 0 = (Synchronization Jump Width - 1) • 64 + (Baud Rate Prescaler - 1)
- Bit Timing Register 1 = (Phase_Seg2 - 1) • 16 + (Phase_Seg1 + Prop_Seg - 1)

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

© This document is the exclusive property of the ROBERT BOSCH GmbH. Without their consent it may not be reproduced or given to third parties.