AN1798

CAN Bit Timing Requirements

by Stuart Robb
East Kilbride, Scotland.

1 Introduction

The Controller Area Network (CAN) is a serial, asynchronous, multi-master communication protocol for connecting electronic control modules in automotive and industrial applications. A feature of the CAN protocol is that the bit rate, bit sample point and number of samples per bit are programmable. This gives the system engineer the opportunity to optimise the performance of the network for a given application. This paper examines the relationship and constraints between the bit timing parameters, the reference oscillator tolerance, and the various signal propagation delays in the system.

2 CAN Bit Timing Overview

2.1 CAN Bit Structure

The Nominal Bit Rate of the network is uniform throughout the network and is given by:

$$f_{\mathrm{NBT}} = \frac{1}{\mathrm{t}_{\mathrm{NBT}}} \tag{1}$$

where t_{NBT} is the Nominal Bit Time. As defined in [1], a bit is divided into four separate non-overlapping time segments called SYNC_SEG, PROP_SEG, PHASE_SEG1 and PHASE_SEG2. These are illustrated in Figure 1.

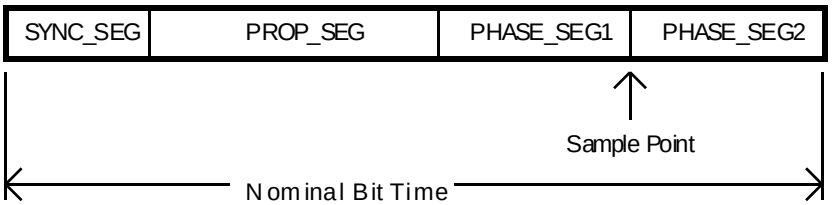

Figure 1 CAN Bit Segments

The sample point indicated in Figure 1 is the position of the actual sample point if a single sample per bit is selected. If three samples per bit are selected, the sample point indicated in Figure 1 marks the position of the final sample.

The period of the Nominal Bit Time (NBT) is the sum of the segment durations:

$$t_{NBT} = t_{SYNC\_SEG} + t_{PROP\_SEG} + t_{PHASE\_SEG1} + t_{PHASE\_SEG2} \quad (2)$$

Each of these segments is an integer multiple of a unit of time called a Time Quantum, t_Q. The duration of a Time Quantum is equal to the period of the CAN system clock, which is derived from the microcontroller (MCU) system clock or oscillator by way of a programmable prescaler, called the Baud Rate Prescaler.

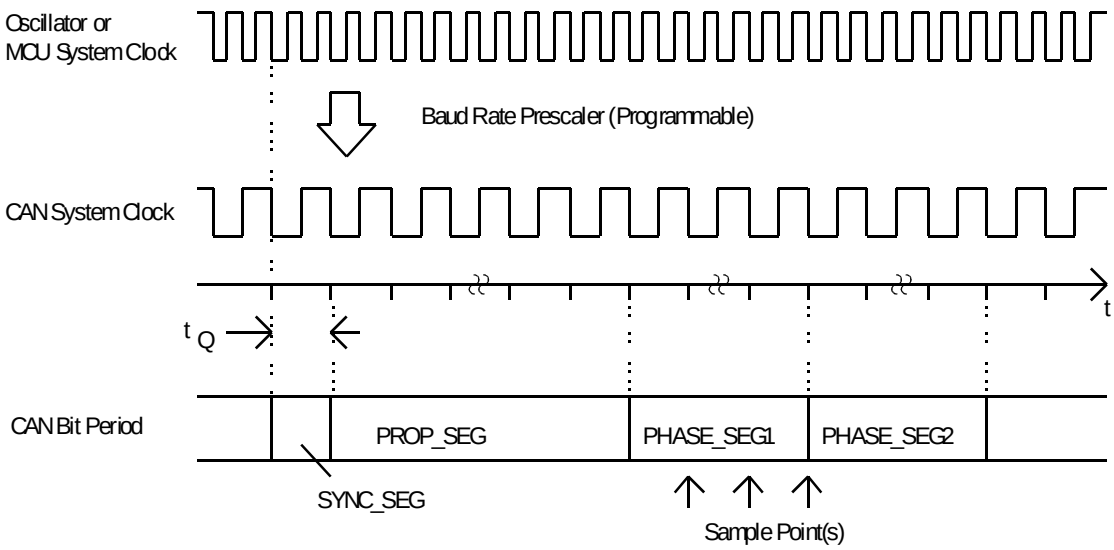

Figure 2 Relationship between CAN System Clock and CAN Bit Period

The duration of the synchronisation segment, SYNC_SEG, is not programmable and is fixed at one Time Quantum. The duration of the other segments are programmable, either individually or with two values, t_{SEG1} and t_{SEG2} where:

$$t_{SEG1} = t_{PROP\_SEG} + t_{PHASE\_SEG1} \quad (3)$$

$$t_{SEG2} = t_{PHASE\_SEG2} \quad (4)$$

The duration of the propagation segment PROP_SEG may be between 1 and 8 Time Quanta. The duration of the segment PHASE_SEG1 may be between 1 and 8 Time Quanta if one sample per bit is selected and may be between 2 and 8 Time Quanta if three samples per bit are selected. If three samples per bit are chosen, the most frequently sampled value is taken as the bit value. The duration of segment PHASE_SEG2 must be equal to PHASE_SEG1, unless PHASE_SEG1 is less than the Information Processing Time (IPT), in which case PHASE_SEG2 must be equal to the Information Processing Time. This is summarised in Table 1.

Segment	Duration
SYNC_SEG	$t_{SYNC\_SEG} = 1\ t_Q$
PROP_SEG	$t_{PROP\_SEG} = 1, 2 \ldots 8\ t_Q$
PHASE_SEG1	$t_{PHASE\_SEG1} = 1, 2 \ldots 8\ t_Q$
PHASE_SEG2	$t_{PHASE\_SEG2} = MAX(IPT, t_{PHASE\_SEG1})$

Table 1

Note: the function MAX(,) returns the larger of the two arguments.

The Information Processing Time is equal to 2 Time Quanta, except for the following circumstances [2]:

TOUCAN Module: IPT = 3 T_Q if the Baud Rate Prescaler = 1 (MCU system clock equals CAN system clock.)

MCAN Module: IPT = 3 T_Q if 3 samples per bit are selected.

From Table 1, it would appear that the minimum number of Time Quanta per bit is 5. However, many CAN controllers require a minimum of 8 Time Quanta per bit, as stipulated in [1]. The maximum number of Time Quanta per bit is 25.

2.2 Synchronisation Segment

For each CAN node, the nominal start of each bit is the beginning of the SYNC_SEG segment. For nodes that are transmitting, the new bit value is transmitted from the beginning of the SYNC_SEG segment. For receiving nodes, the start of the received bit is expected to occur during the SYNC_SEG segment. Due to the propagation delay of the transmitted signal through the physical interface to the bus and along the bus itself, the SYNC_SEG segment of receiving nodes will be delayed with respect to the SYNC_SEG segment of the transmitting node(s). This is illustrated in Figure 3. The actual delay will vary depending on the distance between the transmitting and receiving nodes being considered.

2.3 Propagation Segment

The existence of the propagation delay segment, PROP_SEG, is due to the fact that the CAN protocol allows for non-destructive arbitration between nodes contending for access to the bus, and the requirement for in-frame acknowledgement. In the case of non-destructive arbitration, more than one node may be transmitting during the arbitration field. Each transmitting node samples data from the bus in order to determine whether it has won the arbitration, and also to receive the arbitration field in case it loses arbitration. When each node samples each bit, the value sampled must be the logical superposition of the bit values transmitted by each of the nodes arbitrating for bus access. In the case of the acknowledge field, the transmitting node transmits a recessive bit but expects to receive a dominant bit, i.e. a dominant value must be sampled at the sample point(s). The propagation delay segment, PROP_SEG, exists to delay the earliest possible sample of the bit by a node until the transmitted bit values from all the transmitting nodes have reached all of the nodes.

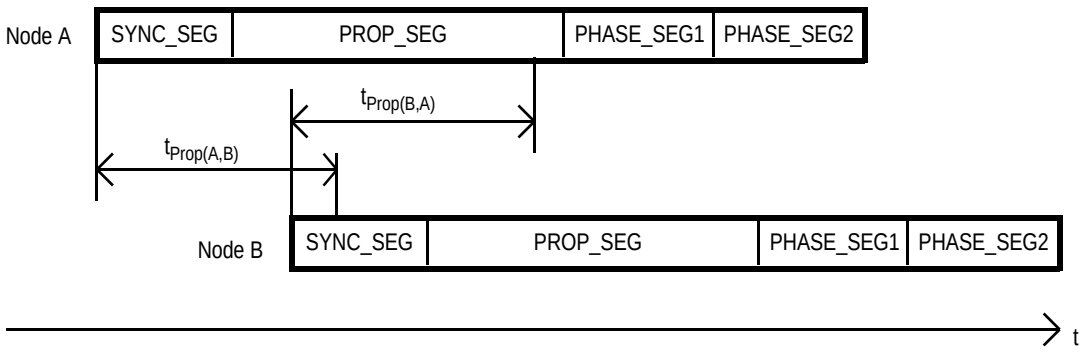

Figure 3 Propagation delay between nodes

Figure 3 shows the propagation delay between two nodes, and shows that the bit value transmitted by Node A is received by Node B after a time $t_{Prop(A,B)}$, and the bit value transmitted by Node B is received by Node A after a time $t_{Prop(B,A)}$, before the end of Node A's propagation segment, thus ensuring that Node A will correctly sample the bit value. Node B will also correctly sample the bit value, even although Node B's sample point lies beyond the end of Node A's bit time, because of the propagation delay between Node A and Node B. Time $t_{Prop(A,B)}$ consists of the sum of the propagation delay through Node A's bus driver plus the propagation delay along the bus from Node A to Node B plus the propagation delay through Node B's bus receiver:

$$t_{Prop(A,B)} = t_{Tx(A)} + t_{Bus(A,B)} + t_{Rx(B)} \quad (5)$$

2.4 Synchronisation

All CAN nodes on a network must be synchronised while receiving a transmission, i.e. the beginning of each received bit must occur during each nodes SYNC_SEG segment. This is achieved by means of synchronisation. Synchronisation is required due to phase errors between nodes which may arise due to nodes having slightly different oscillator frequencies, or due to changes in propagation delay when a different node starts transmitting. Two types of synchronisation are defined, hard synchronisation and re-synchronisation. Hard synchronisation is performed only at the beginning of a message frame, when every CAN node aligns the SYNC_SEG of its current bit time to the recessive to dominant edge of the transmitted Start-Of-Frame bit. Re-synchronisation is subsequently performed during the remainder of the message frame, whenever a change of bit value from recessive to dominant occurs outside of the expected SYNC_SEG segment.

For CAN nodes which are transmitting, the value of a bit is transmitted on the CAN bus at the start of the transmitting nodes SYNC_SEG segment, and the bit value is transmitted until the end of the PHASE_SEG2 segment. All nodes which are active receive the data on the bus (including transmitting nodes) and any changes in bit value are expected to occur during the SYNC_SEG segment. If a recessive to dominant bit value transition is detected outside of a receiving nodes SYNC_SEG segment, then that node will re-synchronise to the edge. If a recessive to dominant bit value transition is detected after the SYNC_SEG segment, but before the sample point, then this is interpreted as a late edge. The node will attempt to re-synchronise to the bit stream by increasing the duration of its PHASE_SEG1 segment of the current bit by the number of Time Quanta by which the edge was late, up to the re-synchronisation jump width limit. The effect of this is

that the next sample point is delayed until it is the programmed number of Time Quanta after the actual edge (if the required delay does not exceed the re-synchronisation jump width limit). Conversely, if a recessive to dominant bit value transition is detected after the sample point but before the SYNC_SEG segment of the next bit, then this is interpreted as an early bit. The node will now attempt to re-synchronise to the bit stream by decreasing the duration of its PHASE_SEG2 segment of the current bit by the number of Time Quanta by which the edge was early, by up to the re-synchronisation jump width limit. Effectively, the SYNC_SEG segment of the next bit begins immediately (if the edge is early by no more than the re-synchronisation jump width limit).

The number of Time Quanta by which a bit period may be extended or shortened due to re-synchronisation is limited by a programmable value called the re-synchronisation jump width (RJW or SJW). The re-synchronisation jump width must be programmed to a valid value. The re-synchronisation jump width cannot exceed 4 Time Quanta and it also must not exceed the number of Time Quanta in the PHASE_SEG1 segment. The minimum value for the re-synchronisation jump width is 1 Time Quantum.

In order to minimise the maximum time between recessive to dominant edges, and hence maximise the number of opportunities for resynchronisation, the CAN protocol uses bit stuffing. After every occurrence of five consecutive bits of equal value, an extra stuff bit of opposite value is inserted into the bit stream. Bit stuffing is implemented in Data Frames and Remote Frames, from the Start-Of-Frame bit to the end of the Cyclic Redundancy Check field.

2.5 Oscillator Tolerance

Typically, the CAN system clock for each CAN node will be derived from a different oscillator. The actual CAN system clock frequency for each node, and hence the actual bit time, will be subject to a tolerance. Ageing and variations in ambient temperature will also affect the initial tolerance. The CAN system clock tolerance is defined as a relative tolerance:

$$\Delta f = \frac{|f - f_{\mathrm{N}}|}{f_{\mathrm{N}}} \tag{6}$$

where f is the actual frequency and f_N is the nominal frequency.

3 Bit Timing Requirements

To ensure effective communication, the minimum requirement for a CAN network is that two nodes, each at opposite ends of the network with the largest propagation delay between them, and each having a CAN system clock frequency at the opposite limits of the specified frequency tolerance, must be able to correctly receive and decode every message transmitted on the network. This requires that all nodes sample the correct value for each bit.

3.1 Propagation Delay

The minimum time for the propagation delay segment to ensure correct sampling of bit values is given by:

$$t_{PROP\_SEG} = t_{Prop(A,B)} + t_{Prop(B,A)} \quad (7)$$

where nodes A and B are at opposite ends of the network, i.e. the propagation delay is a maximum between nodes A and B. From equation (5), this gives:

$$t_{PROP\_SEG} = 2(t_{Bus} + t_{Tx} + t_{Rx}) \quad (8)$$

where t_{Bus} is the propagation delay of the signal along the longest length of the bus between two nodes, t_{Tx} is the propagation delay of the transmitter part of the physical interface and t_{Rx} is the propagation delay of the receiver part of the physical interface. If the propagation delay of the transmitters and receivers on a network is not uniform, the maximum delay values should be used in equation (8).

The minimum number of Time Quantum that must be allocated to the PROP_SEG segment is therefore:

$$PROP\_SEG = ROUND\_UP\left(\frac{t_{PROP\_SEG}}{t_Q}\right) \quad (9)$$

where the function ROUND_UP( ) returns the argument rounded up to the next integer value.

3.2 Oscillator Tolerance Requirements

In the absence of bus errors due to, for example, electrical disturbances, bit stuffing guarantees a maximum of 10 bit periods between re-synchronisation edges (5 dominant bits followed by 5 recessive bits will then be followed by a dominant bit). This represents the worst case condition for the accumulation of phase error during normal communication. The accumulated phase error must be compensated for by re-synchronisation following the recessive to dominant edge and therefore the accumulated phase error must be less than the programmed Re-synchronisation Jump Width (t_{RJW}). The accumulated phase error is due to the tolerance in the CAN system clock, and this requirement can be expressed as [1]:

$$(2 \times \Delta f) \times 10 \times t_{NBT} < t_{RJW} \tag{10}$$

Real systems must operate in the presence of electrical noise which may induce errors on the CAN bus. In the event of an error being detected, an Error Flag is transmitted on the bus. In the case of a local error, only the node that detects the error will transmit the Error Flag. All other nodes receive the Error Flag and then transmit their own Error Flags as an echo. If the error is global, all nodes will detect it within the same bit time and will therefore transmit Error Flags simultaneously. A node can therefore differentiate between a local error and a global error by detecting whether there is an echo after its Error Flag. This requires that a node can correctly sample the first bit after transmitting its Error Flag.

An Error Flag from an Error Active node consists of 6 dominant bits, and there could be up to 6 dominant bits before the Error Flag, if for example, the error was a stuff error. A node must therefore correctly sample the 13^{th} bit after the last re-synchronisation. This can be expressed as [1]:

$$(2 \times \Delta f) \times (13 \times t_{NBT} - t_{PHASE\_SEG2}) < MIN(t_{PHASE\_SEG1}, t_{PHASE\_SEG2}) \tag{11}$$

where the function MIN(,) returns the smaller of the two arguments. Thus there are two clock tolerance requirements which must be satisfied. It should be noted that at high bit rates (small Nominal Bit Time), the CAN clock tolerance is specified over a relatively short time: $10 \times t_{NBT}$ in the case of equation (10), and $13 \times t_{NBT} - t_{PHASE\_SEG2}$ in the case of Equation (11). This is important for systems which derive the CAN clock from a Phase Locked Loop circuit for which the relative accuracy decreases over short time periods due to output jitter.

4 Selection of Bit Timing Values

The selection of bit timing values involves consideration of various fundamental system parameters. The requirement of the PROP_SEG value imposes a trade-off between the maximum achievable bit rate and the maximum propagation delay, due to the bus length and the characteristics of the bus driver circuit. The maximum achievable bit rate is also influenced by the tolerance of the CAN clock source. The highest bit rate can only be achieved with a short bus length, a fast bus driver circuit and a high frequency high tolerance CAN clock source. In many systems, the bus length will be the least variable system parameter which will impose the fundamental limit on bit rate. However the actual bit rate chosen may involve a trade-off with other system constraints, such as cost.

4.1 Step-by-Step Calculation of Bit Timing Parameters

The following steps provide a method for determining the optimum bit timing parameters which satisfy the requirements for proper bit sampling.

Step 1: Determine minimum permissible time for the PROP_SEG segment.
Obtain the maximum propagation delay of the physical interface for both the transmitter and the receiver from the manufacturers data sheet. Calculate the propagation delay of the bus by multiplying the maximum length of the bus by the signal propagation delay of the bus cable. Use these values to calculate t_{PROP_SEG} using equation (8).

Step 2: Choose CAN System Clock Frequency
As the CAN system clock is derived from the MCU system clock or oscillator, the possible CAN system clock frequencies will be limited to whole fractions of the MCU system clock or oscillator by the prescaler. The CAN system clock is chosen so that the desired CAN bus Nominal Bit Time (NBT) is an integer number of time quanta (CAN system clock periods) from 8 to 25.

Step 3: Calculate PROP_SEG duration.
From equation (9), the number of time quanta required for the PROP_SEG segment are calculated. If the result is greater than 8, go back to Step 2 and choose a lower CAN system clock frequency.

Step 4: Determine PHASE_SEG1, PHASE_SEG2
From the number of time quanta per bit obtained in Step 2, subtract the PROP_SEG value calculated in Step 3 and subtract 1 t_Q for SYNC_SEG. If the remaining number is less than 3 then go back to Step 2 and select a higher CAN system clock frequency. If the remaining number is an odd number greater than 3 then add one to the PROP_SEG value and recalculate. If the remaining number is equal to 3 then PHASE_SEG1 = 1 and PHASE_SEG2 = 2 and only one sample per bit may be chosen. Otherwise divide the remaining number by two and assign the result to PHASE_SEG1 and PHASE_SEG2.

Step 5: Determine RJW
RJW is chosen as the smaller of 4 and PHASE_SEG1

Step 6: Calculate required oscillator tolerance from equations (10) and (11).
In the case of PHASE_SEG1 > 4 t_Q, it is recommended to repeat steps 2 to 6 with a larger value for the prescaler, i.e. smaller T_Q period, as this may result in a reduced oscillator tolerance requirement. Conversely, if PHASE_SEG1 < 4 t_Q, it is recommended to repeat steps 2 to 6 with a smaller value for the prescaler, as long as PROP_SEG ð 8, as this may result in a reduced oscillator tolerance requirement. If the prescaler is already equal to 1 and a reduced oscillator tolerance is still required, the only option is to consider using a higher frequency for the prescaler clock source.

4.2 Example 1

Calculate the bit segments for the following system constraints:

Bit rate = 1M bit per second

Bus length = 20m

Bus propagation delay = 5×10^{-9} sm^{-1}

Physical Interface (PCA82C250) transmitter plus receiver propagation delay = 150ns at 85C

MCU oscillator frequency = 8MHz

Step 1: Physical delay of bus = $20 \times 5 \times 10^{-9}$ = 100ns

$$t_{PROP\_SEG} = 2(100ns + 150ns) = 500ns$$

Step 2: A prescaler value of 1 gives a CAN system clock of 8MHz and a Time Quantum of 125ns. This will give 1000 / 125 = 8 time quanta per bit.

Step 3:

$$\text{PROP\_SEG} = \text{ROUND\_UP}\left(\frac{500\text{ns}}{125\text{ns}}\right) = \text{ROUND\_UP}(4) = 4$$

Step 4: From 8 time quanta per bit, subtract 4 for PROP_SEG and 1 for SYNC_SEG. This leaves 3 which is the absolute minimum, so PHASE_SEG1 = 1 and PHASE_SEG2 = 2.

Step 5: RJW is the smaller of 4 and PHASE_SEG1, in this case 1

Step 6: From equation (10):

$$\Delta f < \frac{\text{RJW}}{20 \times \text{NBT}} = \frac{1}{20 \times 8} = 0.00625$$

From equation (11):

$$\Delta f < \frac{\text{MIN(PHASE\_SEG1,PHASE\_SEG2)}}{2(13 \times \text{NBT} - \text{PHASE\_SEG2})} = \frac{1}{2(13 \times 8 - 2)} = 0.00490$$

The required oscillator tolerance is the smaller of these values, i.e. 0.0049 (0.49%) over a period of 12.75µs (12.75 bit periods).

In this case the prescaler = 1 so no reduction in oscillator tolerance can be made without using a higher MCU oscillator frequency. Also PHASE_SEG1 =1 so only one sample per bit is possible.

In summary:

Prescaler	= 1
Nominal Bit Time	= 8
PROP_SEG	= 4
PHASE_SEG1	= 1
PHASE_SEG2	= 2
RJW	= 1
Oscillator tolerance	= 0.49%

4.3 Example 2

Calculate the bit segments for the following system constraints:

Bit rate = 125k bit per second

Bus length = 50m

Bus propagation delay = 5×10^{-9} sm^{-1}

Physical Interface (PCA82C250) transmitter plus receiver propagation delay = 150ns at 85C

MCU oscillator frequency = 8MHz

Step 1: Physical delay of bus = $50 \times 5 \times 10^{-9}$ = 250ns

$$t_{PROP\_SEG} = 2(250ns + 150ns) = 800ns$$

Step 2: A prescaler value of 4 gives a CAN system clock of 2MHz and a Time Quantum of 500ns. This will give 8000 / 500 = 16 time quanta per bit.

Step 3:

$$\text{PROP\_SEG} = \text{ROUND\_UP}\left(\frac{800ns}{500ns}\right) = \text{ROUND\_UP}(1.6) = 2$$

Step 4: From 16 time quanta per bit, subtract 2 for PROP_SEG and 1 for SYNC_SEG. This leaves 13. Therefore PHASE_SEG1 = 6 and PHASE_SEG2 = 6 and the remaining bit is added to PROP_SEG, i.e. PROP_SEG = 3.

Step 5: RJW is the lesser of 4 and PHASE_SEG1, in this case 4

Step 6: From equation (10):

$$\Delta f < \frac{\text{RJW}}{20 \times \text{NBT}} = \frac{4}{20 \times 16} = 0.0125$$

From equation (11):

$$\Delta f < \frac{\text{MIN(PHASE\_SEG1,PHASE\_SEG2)}}{2(13 \times \text{NBT} - \text{PHASE\_SEG2})} = \frac{6}{2(13 \times 16 - 6)} = 0.01485$$

The required oscillator tolerance is the smaller of these values, i.e. 0.0125 (1.25%).

As PHASE_SEG1 > 4, repeat Steps 2 to 6 with a larger prescaler value:

Step 2: A prescaler value of 8 gives a CAN system clock of 1MHz and a Time Quantum of 1000ns. This will give 8000 / 1000 = 8 time quanta per bit.

Step 3:

$$\text{PROP\_SEG} = \text{ROUND\_UP}\left(\frac{800\text{ns}}{1000\text{ns}}\right) = \text{ROUND\_UP}(0.8) = 1$$

Step 4: From 8 time quanta per bit, subtract 1 for PROP_SEG and 1 for SYNC_SEG. This leaves 6. Therefore PHASE_SEG1 = 3 and PHASE_SEG2 = 3.

Step 5: RJW is the smaller of 4 and PHASE_SEG1, in this case 3

Step 6: From equation (10):

$$\Delta f < \frac{\text{RJW}}{20 \times \text{NBT}} = \frac{3}{20 \times 8} = 0.01875$$

From equation (11):

$$\Delta f < \frac{\text{MIN}(\text{PHASE\_SEG1}, \text{PHASE\_SEG2})}{2(13 \times \text{NBT} - \text{PHASE\_SEG2})} = \frac{3}{2(13 \times 8 - 3)} = 0.01485$$

The required oscillator tolerance is the smaller of these values, i.e. 0.01485 (1.485%) over 101µs (12.625 bit times). This is a significant increase in the oscillator tolerance requirement, so the chosen values are:

Prescaler	= 8
Nominal Bit Time	= 8
PROP_SEG	= 1
PHASE_SEG1	= 3
PHASE_SEG2	= 3
RJW	= 3
Oscillator tolerance	= 1.485%

5 References

1. Bosch CAN Specification Version 1.2 1990
2. Freescale BCANPSV2.0/D

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor

Freescale Semiconductor, Inc.