SPC58x configuring CAN and CAN-FD bit timing parameters

Introduction

This technical note details how to configure the bit timing and baud-rate for the CAN peripheral available on all ST automotive SPC58x microcontrollers.

The document provides a practical description of the configuration registers and some examples that aim to speed up the peripheral setup. The configuration steps are shared among the SPC58 microcontroller family excepting for minor changes especially related to the CAN protocol clock initialization. The differences found for the protocol clock configuration are mainly due to different clock tree; these are detailed inside each microcontroller's reference manual (see Section 5 Reference documents).

This document and related examples will focus on the CAN bit timing and baud-rate configuration of SPC584Cx/SPC58ECx 32-bit MCU.

1 CAN overview

SPC584Cx/SPC58ECx has eight CAN instances embedded in two different subsystems as documented in the device reference manual. All the CAN controllers into the same subsystem will share resources like RAM memory, clock, etc.

Each CAN subsystem consists of the following major blocks:

- Modular CAN cores: The registers of the CAN module can be accessed using the generic slave interface (GSI). The peripheral GSI module enables acts as a request from each master.
- CAN-RAM arbiter: it is an additional logic for arbitration between the requests for the RAM access by the various CAN controllers.
- SRAM: the CAN subsystem interfaces with an external RAM using this interface, it is the SRAM.
- ECC controller: it contains the logic to compute and validate the correction code on the SRAM memory.

1.1 Peripheral clocks

As described in the device's reference manual, each CAN instance inside a CAN subsystem needs two different clocks:

- Host clock
- Protocol clock

Note: refer to the device's reference manual for this clock routing and details (see Section 5 Reference documents)

Note: to achieve a stable function of the peripheral, the host clock must always be faster or equal to the CAN protocol clock.

1.2 Protocol clock selection

This paragraph details the protocol clock. It`s the clock used by the CAN controller to define the time quanta period and define the bit timing (Baudrate).

In this microcontroller family it is possible to choose the protocol clock source between:

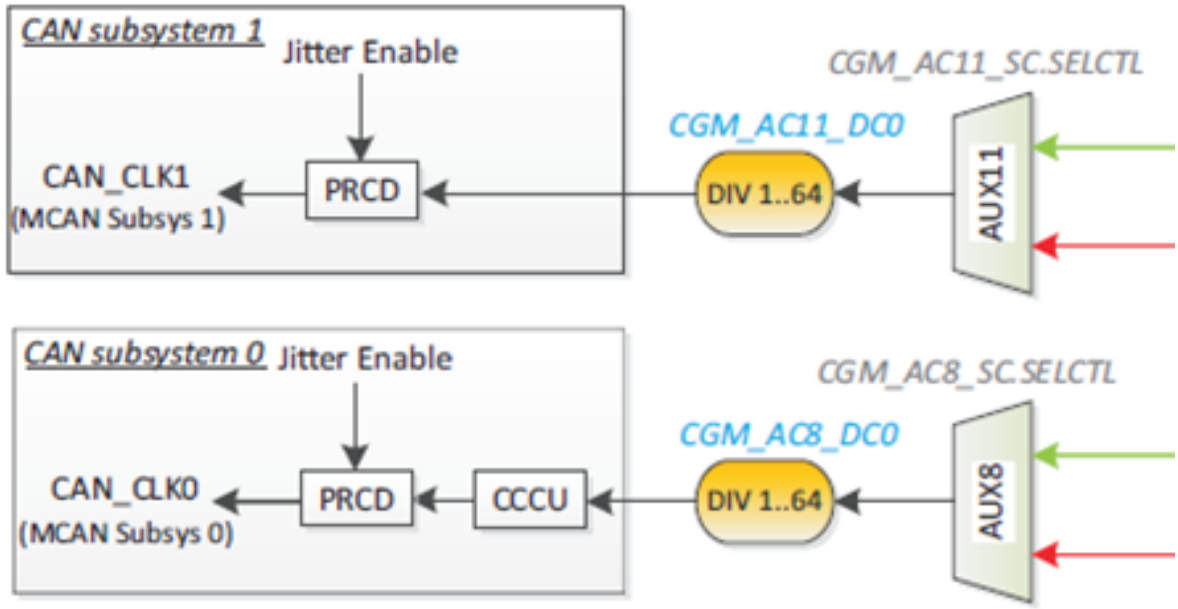
- XOSC
- PLL0

The desired configuration is made through the clock generation module (CGM) registers.

In this example, the registers for CAN subsystem 1 and 2 are respectively CGM_AC8 and CGM_AC11.

See the following figure.

Figure 1. **CAN clock schema**



If it is desired to setup the protocol clock for subsystem zero, User must define the divider in CGM_AC8_DC0 register and setup the XOSC as CAN protocol clock in CGM_AC8_SC register as below:

Supposing to choose the 40 MHz oscillator XOSC this register must be programmed:

```
// Select XOSC as Source clock
MC_CGM.AC8_SC.B.SELCTL = 0x01; // 1 it is the default value
// enable the divider
MC_CGM.AC8_DC0.B.DE = 0x01
// divide input clock by 2
MC_CGM.AC8_DC0.B.DIV = 0x01; // the divider is intended one plus the value wrote in register
```

Using the above setup, the protocol clock provided to CAN will have a frequency of 20 MHz that is the XOSC frequency (40 MHz) divided by 2.

2 Baud rate configuration

This chapter describes how to configure bit timing parameters according to the baud rate value. It aims to provide configuration examples for both CAN Standard and CAN-FD.

On the base of the protocol clock configuration, in this example the clock is considered running at 20 MHz.

It is mandatory to know that the CAN controller divides each bit in an integer number of time-quanta (Tq). The time-quanta time is the period of protocol clock:

Tq = 1/protocol clock

Acting on baud-bate pre-scaler (NBRP field of NBTP register) it is possible to define the number of Tq per bit.

For example, if it is needed a **1M baud-rate**, the **bit-time will be 1 µs**. If the CAN controller has a Protocol clock of 20 MHz, defining a baud rate pre-scaler equals to “1”, then the Tq will be 50 ns. Dividing the bit time by the Tq time (protocol clock period) the result is 20 time-quanta per bit.

Using a baud-rate pre-scaler equals to “2”, the frequency of protocol clock will be divided by 2 and 10 MHz as protocol clock will be obtained. So, following the same previous formula, the number of Tq per bit will be 10 instead of 20.

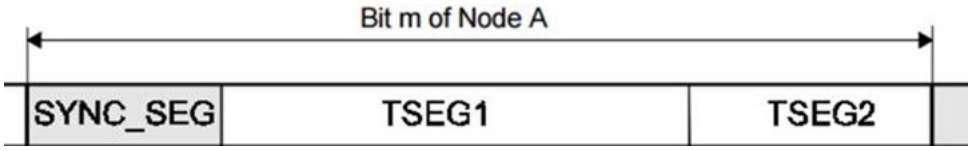
Note: all the pre-scalers are acceptable for a selected baud-rate. In the previous example, using a pre-scaler value of 3, the protocol clock will be 6.666 MHz (20 MHz / 3) and the Tq period will be about 152 ns. Dividing the bit time (1 µs at 1 Mbit/s) for the Tq period the result isn`t an integer number. Since CAN protocol the number of Tq inside a bit must be integer, then the pre-scaler 3 cannot be used for 1 M baud-rate having a protocol clock (coming from CGM clock generation module) of 20 MHz.

3 Bit timing configuration

Configuring the bit timing registers, it is possible to define the position of the sample point for all the bits that the controller gets on the bus and the baud-rate as well.

For each bit, three sections are available, as shown in the following figure.

Figure 2. Bit timing



Each section is composed by a configurable number of time-quanta, except for SYNC_SEG that is always composed by 1 time-quanta.

The sum of SYNC_SEG, TSEG1 AND TSEG2 is equal to the total number of time-quanta that compose the bit.

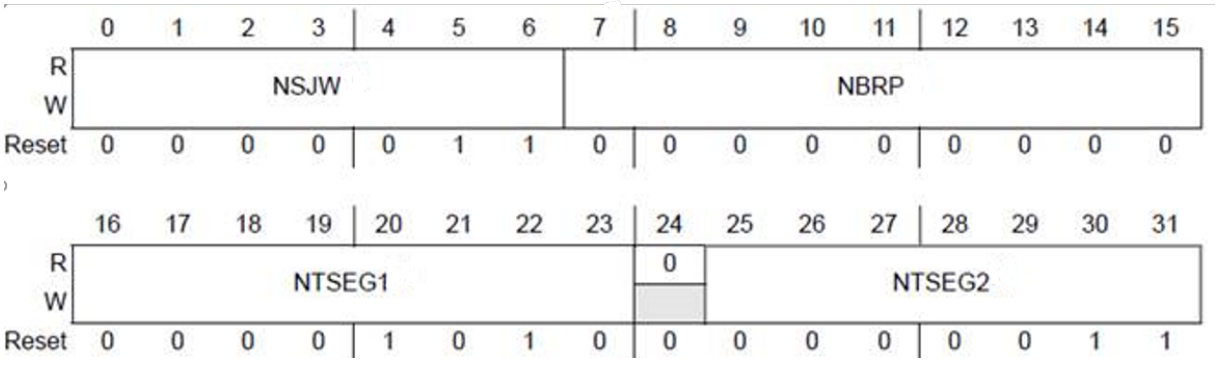
The sample point is where the controller samples the bit and is placed at the end of TSEG1 (or beginning of TSEG2).

Configuring the length of segments TSEG1 and TSEG2 (both expressed in number of time-quanta (Tq)) it is defined the positioning of the sample point. It must be programmed in the registers NBTP (for arbitration part of the message) and DBTP for the data. In CAN-FD mode both registers must be programmed, while in standard CAN mode it is needed to only configure the NBTP register.

3.1 Standard CAN example

Supposing it is needed to configure the CAN controller with a 1Mbit/ baud-rate and a sample point of 75% the NBTP register will be programmed as shown below.

Figure 3. NBTP register



The NSJW field is used for the Resynchronization and generally it should <= NTSEG2 value.

Supposing to have 20 MHz as protocol clock, and no other pre-scaler must be applied on this frequency, so the NBRP field will be set to 0. In this case the time-quanta will be 1/20000000 = 50 ns. The bit time (the length of a bit) at 1 Mbit/s is 1 µs.

Dividing the bit time period by the time-quanta period it is seen that one bit contain exactly 20 time-quanta.

Note: if the division result is not an integer value user has to try another pre-scaler and so, the sum of segments length NTSEG1 + NTSEG2 = 19 (this because SYNC_SEG is always 1 Tq).

Supposing to sample each bit at 75%, then the sum SYNC_SEG + NTSEG1 should be the 75% of total TQ number per bit that in our case (20 total Tq per bit) is 15 time-quanta.

Considering one time-quanta of SYNC_SEG, then the NTSEG1 must be 14 Tq and the remaining 5 Tq are the length of NTSEG2.

At the end, NBTP field values are:

- NSJW = 4 (NTSEG2 value)
- NBRP to 0
- NTSEG1 to 13
- NTSEG2 to 4

Note: from device's reference manual: the actual interpretation by the hardware of this value is such that one more than the value programmed here is used (see Section 5 Reference documents)

Considering one time-quanta of SYNC_SEG, then the DTSEG1 must be 5 Tq and the remaining 4 Tq are the length of DTSEG2.

The DBTP register fields are:

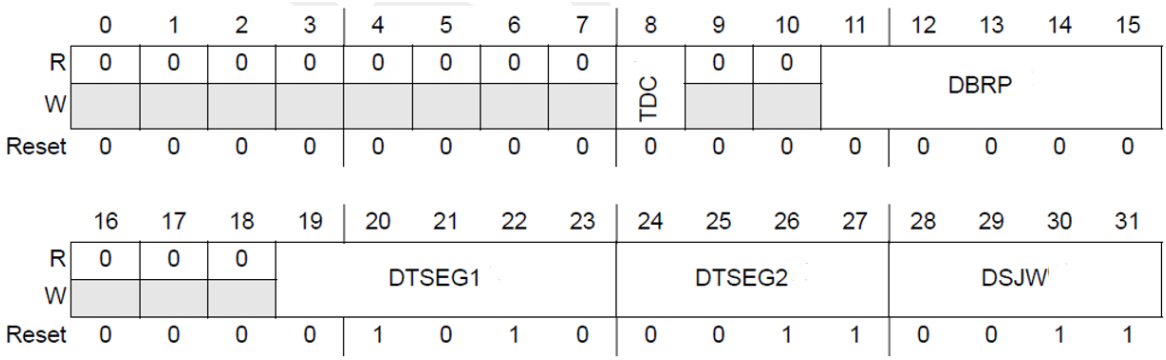
- TDC to 1 (because baud-rate > 1M)
- DSJW to 3 (DTSEG2 value)
- DBRP to 0
- DTSEG1 to 4
- DTSEG2 to 3

Note: From device's reference manual: the actual interpretation by the hardware of this value is such that one more than the value programmed here is used. (see Section 5 Reference documents)

3.2 CAN-FD example

This paragraph shows how to configure the bit timing and the baud-rate in case of CAN-FD. The software must configure two registers: NBTP and DBTP, former is for the arbitration part of a message and the latter one for the data part of the message. The NBTP register programming does not change from the previous example. So below it is showed how to configure the baud-rate for the data part of a message. It is also detailed the configuration of transceiver delay compensation for CAN-FD mode (not available on standard CAN).

Figure 4. DBTP register



This register has an extra bit: Transceiver delay compensation (TDC). It enables the use of transceiver delay compensation and must be set for baud-rate greater than 1 M. Using the TDC, the Software application can also configure a secondary sample point.

The first sample point is to define the segments length in the bit timing configuration (DTSEG1 and DTSEG2), while the second sample point depends on transceiver delay compensation value.

Supposing to configure the baud-rate of the message data part at **2 M** and the **two sample** points at 60% and 80% of the bit length. Below it will be showed how to program the registers:

- TDC to 1 (because baud-rate > 1 M)
- DSJW to this is used for the resynchronization and generally it could be equal to DTSEG2 value (maximum synch width).

Supposing to have 20 MHz as protocol clock, and no other pre-scaler must be applied on this frequency, so the DBRP field will be set to 0. The time-quanta period will be 1/20000000 = 50 ns. The bit time (the length of a bit) at 2 Mbit/s is 500 ns.

Dividing the bit time period by the time-quanta period, it can be noticed that one bit contains exactly 10 time-quanta.

Note: if the division result is not an integer value user has to try another pre-scaler and so DTSEG1 + DTSEG2 = 9 (this because SYNC_SEG is always 1 Tq).

Supposing that it is needed a primary sample point at 60% of bit length then User can easily calculate that is 6 Tq on a bit length of 10 Tq.

Considering one time-quanta of SYNC_SEG, then the DTSEG1 must be 5 Tq and the remaining 4 Tq are the length of DTSEG2.

The DBTP register fields are:

- TDC to 1 (because baud-rate > 1 M)
- DSJW to3 (DTSEG2 value)
- DBRP to 0
- DTSEG1 to 4
- DTSEG2 to 3

Note: from device's reference manual: the actual interpretation by the hardware of this value is such that one more than the value programmed here is used (see Section 5 Reference documents)

4 Transceiver delay compensation

After configuring the baud-rate and the primary sample points, the transceiver delay compensation needs to be calculated. The transceiver delay compensation has been added to CAN-FD protocol to compensate CAN transceiver's loop delay at baud-rate greater than 1M, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver. There is a relationship between TDCO (transceiver delay compensation offset) and secondary sample point:

Note: this value is declared inside the transceiver specification and often called: loop delay.

Secondary sample point position% = TDCO * (protocol_clock_period (ns) * (bitrate(Mbits) / 1000))

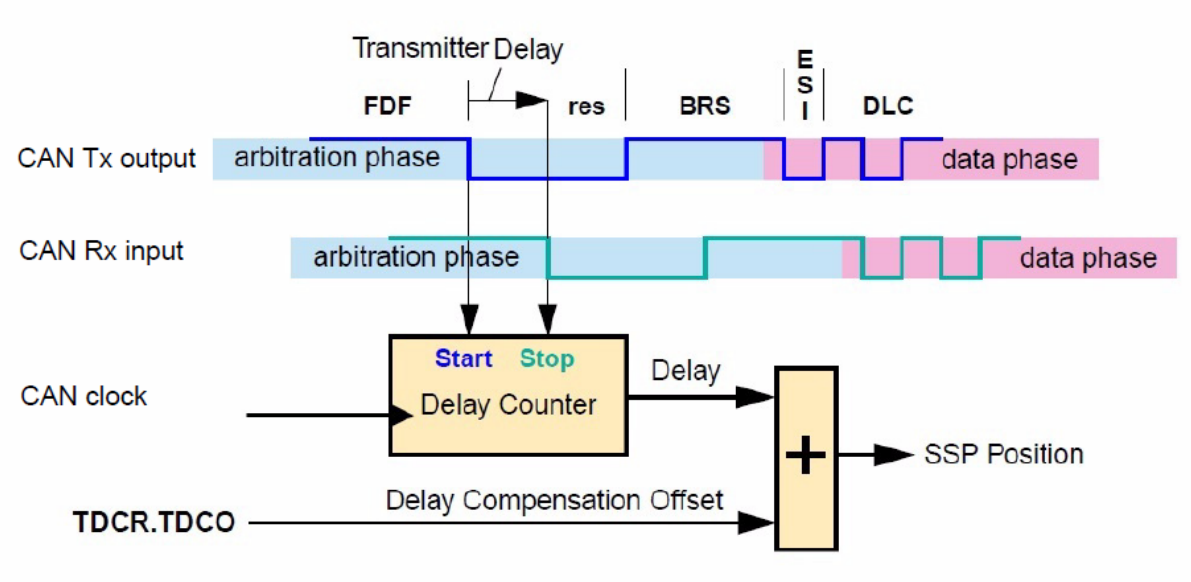
So: TDCO = secondary sample point position% / (protocol_clock_period (ns) * (bitrate(Mbits) /1000))

In this example:

TDCO = 0.80 / (50 * 0.002) = 8

Figure 5. Transceiver delay timing shows how the transceiver delay compensation works. In a glance, the controller calculates the delay between transmit and receive signals and tries to compensate that by using TDCO value. This is to sample, at the correct secondary sample point, where the controller checks the transmission errors

Figure 5. Transceiver delay timing



5 Reference documents

- RM0407, *SPC584Cx/SPC58ECx 32-bit MCU family built on the Power Architecture for automotive body electronics applications reference manual*
- SPC58EHx, SPC58NHx datasheet

6 Acronyms and abbreviations

Table 1. Acronyms

Abbreviation	Complete name
CAN	Controller area network
FD	Flexible data rate
TQ	Time quanta
TDCO	Transceiver delay compensation offset (field of TDCR register)

Revision history

Table 2. **Document revision history**

Date	Version	Changes
02-Nov-2020	1	Initial release.

Contents