AN12799

How to Use CAN-FD to Transfer Data on LPC54018

Rev. 0 — 23 March 2020 Application Note

Contents

1 Introduction

The LPC54018 features two CAN controllers supporting CAN-FD. The LPC54018 SDK provides the mcan_interrupt_transfer example to demonstrate how to use CAN to transfer data. Based on this example, this application note describes the use of CAN-FDs bit rate switch and transmitter delay compensation features. Enabling these two features can improve the throughput and eliminate the bit error caused by transmitter delay.

2 CAN-FD

CAN-FD is defined in the international standard ISO 11898-1:2015. For quick start on using CAN-FD, this section introduces some key features of CAN-FD and aims at users who are familiar with CAN.

2.1 Differences between CAN and CAN-FD

There are two key differences between classical CAN and CAN-FD. The first is that CAN-FD can use much higher bit rates than classical CAN. Bit rate of classical CAN is limited to 1 Mbps. CAN-FD does not have a theoretical limit, but in practice it is limited by the transceivers. The second key difference is the increased amount of data per CAN message. Classical CAN is limited to eight bytes. CAN-FD limit is increased eight-fold to 64 bytes per message.

With the increased amount of data per CAN message, CAN-FD frames need higher bit rate to decrease the delay time in the communication and increase real-time performance. The CAN-FD frames can reach higher bit rates by enabling bit rate switch feature.

On another hand, the bit rate is higher, the bit time is shorter. In order to enable a data phase bit time, which is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN-FD frame is limited by the transmitter delay.

2.2 Bit rate switch

In CAN-FD frame, control phase, data phase, and CRC phase are transmitted with a higher bit rate than the beginning and the end of the frame.

The bit rate switch feature is enabled by setting the BRSE bit in the CCCR register. When enabling the bit rate switch, we also need to set arbitration phase bit rate (before enabling bit rate switch) and data phase bit rate (after enabling bit rate switch) correctly. The arbitration phase bit rate is set by the NBTP register, the data phase bit rate is set by DBTP register.

2.3 Transmitter delay compensation

The protocol unit of the MCAN has implemented a delay compensation mechanism to compensate the transmitter delay. The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay It is described in detail in the new ISO11898-1. It is enabled by setting the TDC bit in the DBTP register. The TDCO field in the TDCR register is used for setting the transmitter delay compensation offset. The offset value defines the distance measured between the delay

from m_can_tx to m_can_rx and the secondary sample point. The transmitter delay compensation offset and secondary sample point are show in following figure.

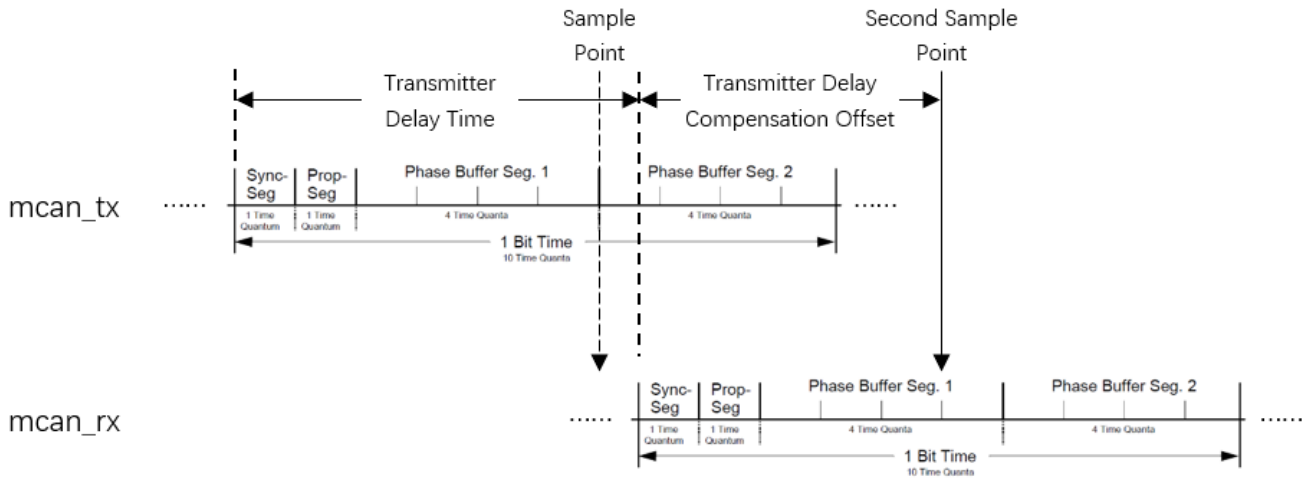

Figure 1. MCAN transmitter delay compensation offset and secondary sample point

3 Demonstration

This section provides a brief introduction on how to use CAN-FD to transfer data and enable bit rate switch and transmitter delay compensation. It is based on the LPC54018 SDKs mcan_interrupt_transfer example.

3.1 Hardware environment

- **Boards**
  - Two LPCXpresso54018 boards (OM40003)
  - Two Dual CAN-FD transceiver shields (OM13099)
- **Miscellaneous**
  - Two Micro USB cables
  - One 120 Ohms terminated CAN cable
  - Personal Computer
- **Boards Setup**

CAN shields were connected to evaluation boards. The CAN0 interfaces on each shield were connected using the terminated CAN cable. Connect the micro USB cable between PC and J8 link on the evaluation board for loading and running a demo. This is also used for UART communication to UART terminal on PC. The boards setup and CAN connection are showen in following figure.

Figure 2. Boards setup and CAN connection

3.2 Software environment

- **Tool chain**
  - MCUXpresso10.3.0 or above
- **Software package**
  - lpcxpresso54018_mcan_interrupt_transfer_canfd.zip
- **UART terminal program**
  - PuTTy, or similar one

3.3 Using CAN-FD steps overview

This application note provides a mcan_interrupt_transfer_canfd software package. It demonstrates the use of CAN-FD to transfer data. In this example, the following steps are required:

- Set the system clock
- Set the MCAN clock
  - Divide the system clock for MCAN module

- Initialize MCAN
  - Enable MCAN clock
  - Reset the MCAN module
  - Configure the MCAN Control Register, enable CAN-FD and bit rate switch
  - Set arbitration phase bit rate and data phase bit rate
  - Enable transmitter delay compensation
- Set Message RAM
- Set message ID filter configuration and element
- Set the configuration of Rx FIFO and Tx buffer
- Enter MCAN normal mode
- Transfer data
  - Configure TX frame data and send
  - Receive data

On the LPC54018 SDKs mcan_interrupt_transfer example, it uses the classical CAN nodes on bus by default. The data phase bit rate setting needs the same with the arbitration phase bit rate setting. They should also comply classical CAN bus protocol.

But in the example of this document, it uses the CAN-FD nodes on bus. The CAN-FD arbitration phase bit rate is set to 1 Mbps and the data phase bit rate is set to 5 Mbps. It must enable the bit rate switch and transmitter delay compensation features.

The follow sections introduce some key steps for using CAN-FD.

3.4 Config CAN-FD Bit Rate Switch

Enabling the CAN-FD Bit Rate Switch can increase the throughput. In the example, it calls the MCAN_SetBaudRate() function to set arbitration phase bit rate and calls the MCAN_SetBaudRateFD() function to set data phase bit rate. The MCAN clock is set to 60 MHz.

3.4.1 Set arbitration phase bit rate to 1 Mbps

As per the CAN specification, the nominal bit rate is the number of bits per second transmitted in the absence of resynchronization by an ideal transmitter. The relationship between nominal bit rate and the nominal bit time is NOMINAL BIT TIME = 1 / NOMINAL BIT RATE. So, if the arbitration phase bit rate is set to 1 Mbps, the arbitration phase bit time is 1μs.

The time quantum (t_q) is a fixed unit of time derived from the MCAN clock period. There exists a programmable prescaler with integral values ranging from at least 1 to 32. Starting with the MCAN clock period, the t_q can have a length of

t_q = m * MCAN clock period = m / MCAN clock

with m the value of the prescaler.

In the MCAN_SetBaudRate() function, we need to define a variable whose type is mcan_timing_config_t for setting arbitration phase bit time. The structure mcan_timing_config_t is defined as Fig 3.

```
typedef struct _mcan_timing_config
{
    uint16_t preDivider; /*!< Clock Pre-scaler Division Factor. */
    uint8_t rJumpwidth;  /*!< Re-sync Jump Width. */
    uint8_t seg1;        /*!< Data Time Segment 1. */
    uint8_t seg2;        /*!< Data Time Segment 2. */
} mcan_timing_config_t;
```

Figure 3. MCAN protocol timing characteristic configuration structure

The prescaler m is equivalent to (preDivider + 1). The t_q can have a length of

t_q = (preDivider + 1) / 60 MHz = (preDivider + 1) / 60 (μs)

with the structure element preDivider.

The total number of t_q in an arbitration phase bit time can be programmed in the range of 4 to 385 time quanta. The length of

Arbitration phase bit time = MCAN_TIME_QUANTA_NUM_ARBIT * t_q.

We define a macro MCAN_TIME_QUANTA_NUM_ARBIT as 20 in the example. The arbitration phase bit time is 1 μs, t_q is 1/20 μs, preDivider is 2.

The macro

MCAN_TIME_QUANTA_NUM_ARBIT = 1 + (seg1 + 1) + (seg2 + 1)

where the structure elements seg1 and seg2 stands for phase buffer segment 1 and 2 minus one respectively. We set the element seg1 to the value 13, and the element seg2 to 4 in the example. Calling the updated MCAN_SetBaudRate() function, we finish setting the arbitration phase bit rate to 1 Mbps.

3.4.2 Set data phase bit rate to 5 Mbps

Setting the data phase bit rate is similar to setting the arbitration phase bit rate. The data phase bit rate is set in the MCAN_SetBaudRateFD() function.

One of the differences is that the total number of t_q in a data phase bit time may be programmed in the range of 4 to 49 time quanta.

In the example, we define a macro MCAN_TIME_QUANTA_NUM_DATA as 12 as the total number of t_q in a data phase bit time. And the element seg1 is 7, the element seg2 is 2. The element preDivider is 4. Calling the updated MCAN_SetBaudRateFD() function, we finish setting the data phase bit rate to 5 Mbps, which is the highest bit rate, that the onboard transceiver can support.

3.4.3 Enable Bit Rate Switch

Setting the MCAN CCCR register's BRSE bit to 1 enables the Bit Rate Switch for CAN-FD. In this application note's example, control phase, data phase, and CRC phase of the CAN-FD frame are transmitted at the bit rate of 5 Mbps, while the other phases of a CAN-FD frame are transmitted at the bit rate of 1 Mbps.

3.5 Config Transmitter Delay Compensation

In this application note's example, it defines a function to enable the transmitter delay compensation for CAN-FD.

3.5.1 MCAN_SetTransmitterDelayCompensationFD

Set the TDC bit in the DBTP register to 1 to enable the transmitter delay compensation. The TDCO field in the TDCR register sets the transmitter delay compensation offset

This function sets the second sample point at the middle of the mcan_rx bit time.

The transmitter delay compensation offset = MCAN_TIME_QUANTA_NUM_DATA / 2

where the macro MCAN_TIME_QUANTA_NUM_DATA is defined bysetting the data phase bit rate step.

3.6 Steps and results

On this CAN-FD demo, there are two nodes on the bus to transmit and receive data. One node is selected as A,and the another is selected as B. Press any key on the node A's terminal console to trigger one-shot transmission. The node B receives this one-shot transmission data and sends it back to node A. Node A receives the data and this one-shot transmission finish.

The basic steps are:

1. Hardware setup
   - See section 3.1 for boards setup and CAN connection
2. Build and download
   - Import this demo's software package to the MCUXpresso IDE and build.
   - Download the executable file using debugger.
3. Setup for UART terminal programs
   - Check the COM number, which is simulated as LPC-LinkII in Device Manager, on PC.
   - Open two UART terminal programs on PC and connect one evaluation board with one UART terminal program. Configure the communication protocol as 115200 + 8 + N + 1.
4. Run
   - Reset two evaluation boards by pressing the SW1(reset) button on each board. One evaluation board selects as node A, another selects as node B. Press any key on the node A's terminal console to trigger one-shot transmission. The message appears on the node A terminal is as shown in Figure 4 and the message appears on the node B terminal is as shown in Figure 5.

```
Please select local node as A or B:
Note: Node B should start first.
Node:A
Press any key to trigger one-shot transmission

Received Frame ID: 0x123
Received Frame DATA: 0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xa 0xb 0xc 0xd 0xe
 0xf 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1d 0x1e
0x1f 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2a 0x2b 0x2c 0x2d 0x2e
0x2f 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3a 0x3b 0x3c 0x3d 0x3e
0x3f
Press any key to trigger the next transmission!
```

Figure 4. Messages printed on the node A terminal

```
Please select local node as A or B:
Note: Node B should start first.
Node:B
Start to Wait data from Node A

Received Frame ID: 0x321
Received Frame DATA: 0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xa 0xb 0xc 0xd 0xe
 0xf 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1d 0x1e
0x1f 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2a 0x2b 0x2c 0x2d 0x2e
0x2f 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3a 0x3b 0x3c 0x3d 0x3e
0x3f
Wait Node A to trigger the next transmission!
```

Figure 5. Messages printed on the node B terminal

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2020. **All rights reserved.**

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 23 March 2020

Document identifier: AN12799

arm