**OVERVIEW and PROTOCOL**

1. Concepts

Controller Area Network (CAN or CAN Bus) is a serial, high-speed, half-duplex (only transmit or receive at a time), two-wire network technology. CAN was originally designed for the automotive industry, but now CAN has also become a common standard in industrial automation and other industries.

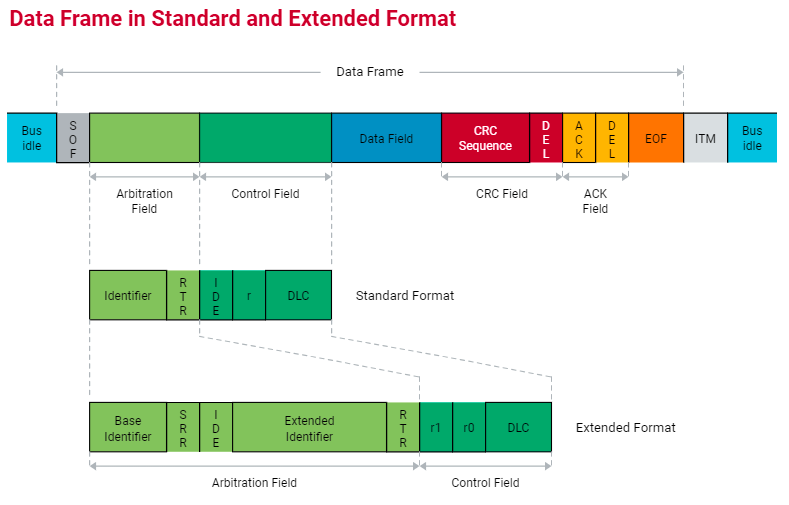
2. Advantages of CAN

* Basic, low cost: CAN bus has only 2 wires, making it easier to connect control modules together when compared to the traditional way. Accompanied by many benefits of ease of installation and ease of repair and maintenance when there is a problem.
* Create a common protocol so that different vendors can develop interoperable control modules
* Priority of messages: each message transmitted from a node or station on the CAN bus has priority. When multiple messages are sent to the bus at the same time, the message with the highest priority is transmitted. Messages with lower priority are paused and retransmitted when the bus is free. The determination of message priority is based on the message structure (structure) and the arbitration mechanism specified in the CAN standard.
* Flexible configuration: allows to configure bit time, synchronization time, length of data transmitted, data received, ...
* Multipoint data reception with time synchronization: a message can be received by many different nodes on the bus at the same time. All nodes on the bus can see the message being transmitted on the bus, depending on the configuration at each node, the node will decide whether to accept this message or not.
* Multiple masters (multimaster)
* Error detection and signaling: Each message is accompanied by a CRC (Cyclic Redundancy Code) code to perform error checking. If an error occurs, the receiving node ignores the error message and transmits an error frame onto the CAN bus. Each node in the bus has its own error management counter to determine its own error state. If the error occurs too much, a node can automatically disconnect from the bus. There are also several other types of errors that can be detected with the CAN standard.
* Automatic retransmission of error messages when the bus is idle: A message transmitted to the bus if there is an error will not be lost, but the node transmitting this message will keep it and automatically retransmit this message when the CAN bus idle until successful. This helps to ensure data integrity in the bus.

3. Types Frame

CAN data is transmitted as frames. There are 4 different types of Frames, which are:

* Data frame: A data frame carries data from a transmitter to the receivers.
* Remote frame: A remote frame is transmitted by a node to request the transmission of the data frame with the same identifier, except missing data field.
* Error frame: An error frame is transmitted by a node on detecting a bus error.
* Overload frame: An overload frame is used to provide for an extra delay between the preceding and the succeeding data or remote frames.



* Start Of Frame Field – SOF: For both CAN 2.0 formats, the start field is the position of the first bit in the frame. This field occupies 1 bit of data. This first bit is a Dominant Bit (logical level 0) marking the start of a Data Frame.
* Arbitration Field: Arbitration field format is different with standard format and extended format:

+ Standard format: Size of Arbitration field is 12 bit include 11 bit ID and 1 bit RTR

+ Extended format: Size of arbitration field is 32 bit include 29 bit ID, 1 bit SRR, 1 bit IDE and 1 bit RTR.

+ RTR (remote transmission request):

The bit used to distinguish whether the frame is a Data Frame or a Remote Frame.

If it is a Data Frame, this bit is always 0 (Dominant Bit).

If it is a Remote Frame, this bit is always 1 (Recessive Bit).

This bit position is always after the ID bit.

In case if the Data Frame and Remote Frame with the same ID are sent at the same time, the Data Frame will take precedence.

+ Bit SRR (Substitute Remote Request):

This bit is only available in the extended frame.

This bit has a value of 1 (Recessive Bit).

Compared to the corresponding position in the standard frame, this bit coincides with the position of the RTR bit, so it is also called the Substitute bit.

Assuming there are two nodes transmitting at the same time, one transmitting the standard Data Frame and the other transmitting the Extended Data Frame with the same ID, the Node transmitting the standard frame will win the arbitration because to the position after the ID, the standard frame is the RTR bit. = 0, and the extended frame is bit SRR = 1. Thus, the standard frame prevails over the extended frame when having the same ID.

+ Bit IDE (Identifier Extension):

This is the bit that distinguishes between the standard and extended framework types: IDE = 0 specifies the standard framework, IDE = 1 specifies the extended framework.

This bit is located in the preemption field with extended frames and in the control field with standard frames.

* Control Field:

+ Standard format: Include IDE, r0 and DLC (Data Length Code)

+ Extended format: r1, r0 and DLC

Bit r0, r1 (two reserved bits)

Although these two bits must be transmitted as Recessive Bits by the transmitter, the receiver is not interested in this 2-bit value. The receiver may receive combinations of 00, 01, 10, or 11 of r1 and r0 but not consider it an error but ignore it and receive the message normally.

DLC (Data Length Code)

Has a length of 4 bits that specify the number of bytes of the Data Frame's data field can only carry a value from 0 to 8 corresponding to a data field with 0 to 8 data bytes. The Data Frame may have no data bytes when DLC = 0. Values greater than 8 are not allowed. The figure below shows the types of bit codes that DLC can contain to specify the number of bytes of the data field.

* Data Field:

This field has a length of 0 to 8 bytes depending on the value of the DLC in the control field.

* Cyclic Redundancy Check Field – CRC:

The check field or CRC field consists of 16 bits and has two parts:

CRC Sequence: Consists of 15 bits of Sequence CRC

CRC Delimiter: is a Recessive Bit that separates the CRC field from the ACK field

The CRC check code is best suited for frames where the bit sequence to be tested is less than 127 bits in length, which is suitable for detecting bus errors. Here, the total bit from the start field (SOF) to the data field (Data Field) is up to 83 bits (standard format frame) and 103 bits (extended format frame).

=> The CRC field protects the information in the Data Frame and Remote Frame by adding redundant check bits at the beginning of the transmitted frame. At the beginning of the receive frame, it calculates the same CRC as the transmitter when it received the data and compares that result with the CRC Sequence it received, if different then there is an error, if it is the same, it is received correctly. from the SOF field to the data field.

* Acknowledge Field – ACK:

Has a length 2 bit and consists of ACK slot and ACK Delimiter.

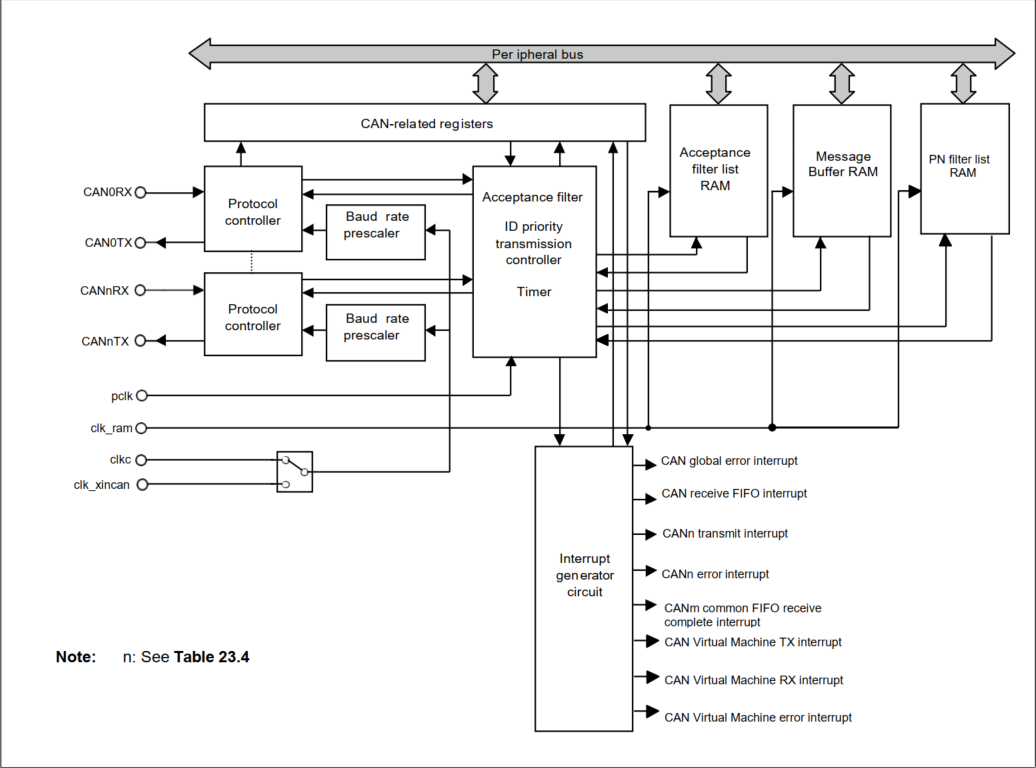
ACK Slot: has a length of 1 bit, a data transmission node will set this bit to Recessive. When one or more Nodes receive the correct message value (no error and matched CRC Sequence has been compared), it will report back to the transmitter by transmitting a Dominant Bit at the ACK Slot position to overwrite Recessive. Transmitter bit.

ACK Delimiter: 1 bit in length, it is always a Recessive Bit. Thus, we see that the ACK Slot is always placed between two Recessive Bits, CRC Delimiter and ACK Delimiter.

* End Of Frame Field (EOF)

The EOF field is a field that indicates the end of a Data Frame or Remote Frame. This field contains 7 Recessive Bits.

**Block diagram CANFD:**



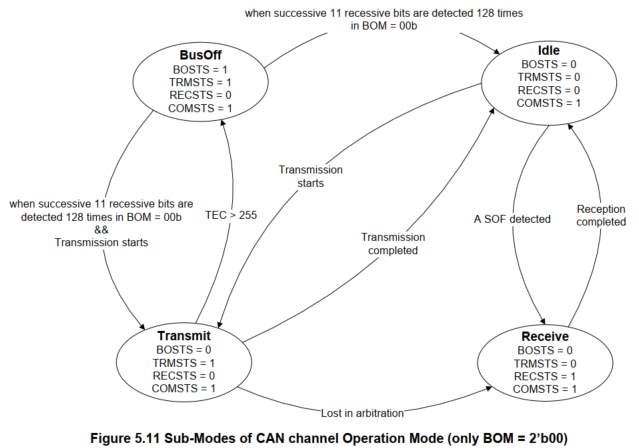
**TX/RX:** Input/Output pins of the CAN module

**Protocol controller:** Handles CAN protocol processing such as bus arbitration, bit timing at transmission and reception, stuffing, error handling, etc.

**Acceptance filter list RAM:**This RAM is used to store the message acceptance filtering entries for all channels. Each acceptance filter entry has an individual ID, data length code, data field, message pointer for upper layer application usage and message direction pointer. The AFLRAM is divided in two parts to accelerate the AFL access process.

**Message Buffer RAM:**This RAM is used to store messages after reception or for transmission using a normal Message Buffer  
or a FIFO. Each message entry has an individual ID, data length code, data field, message pointer for  
upper layer application usage and a time stamp.  
**Acceptance filter:**Performs filtering of received messages. The entries in the Acceptance filter list RAM are used for the  
filtering process.  
**Timer:**Two timers  
• Reception Timestamp function  
• Transmission separation time for FIFO Buffers  
**Interrupt generator:**Generates several types of global and channel interrupts

CAN channel operation mode:



Transmit:

Start: Register **CFDCnCTR** - channel n control registers 🡪 Using **CFDCnSTS** – Channel n status registers 🡪 register CFDCnFDCFG - channel n CAN-FD configuration register (if FD only enable setting 0001 0000 0000 0000 0000 0000 0000 0000 (0x10000000))

Stop: Register **CFDCnCTR** - channel n control registers 🡪 channel sleep request enable.

Send CAN data:

Prepare the buffer:

Send Data buffer: **CFDTMCi -** TX Message Buffer Control Registers i

**CFDTMTRSTSf -** TX Message Buffer Transmission Request Status Register f

In CAN, the transmitter of any message will only try to retransmit if it loses arbitration on the bus, or potentially if no other CAN node on the bus provides an ACK within the correct window.  Once these two basic criteria are met, the transmitter will broadcast its message without caring about the state of the receiver node's message buffers.

The reception process of the CAN node has to take care to empty any receive buffers before the next available message cycle.  Since the LPC1700 series CAN peripheral has a double receive buffer, the controller is able to receive a message that is presently coming over the bus, while the CPU is processing a message which was received previously.

If the receive buffers have not been processed and cleared by the CPU and a new message arrives, the CAN controller flags a data overrun event and discards the new message.  A status flag is set to identify this event, and an optional interrupt can be generated.

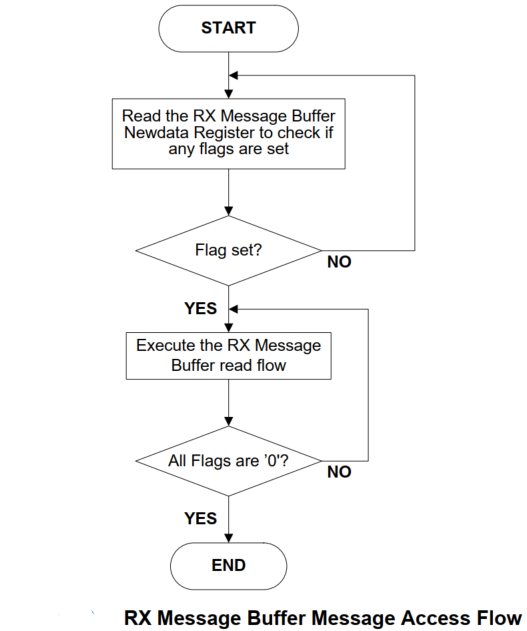
You should ensure that received messages are processed timely.  Also, your CAN message strategy might allow critical messages to be transmitted periodically to help guarantee delivery.

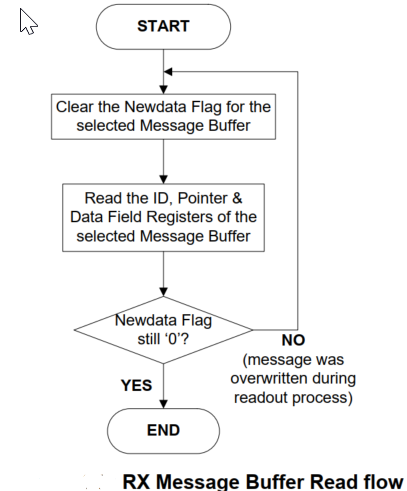
**OPERATIONs:**

1. **Reception - Message storage in RX Message Buffers:**

When a message is successfully received and stored in a RX Message Buffer, the corresponding Newdata Flag is set in the RX Message Buffer Newdata Register.  
The CAN Message can be read from the corresponding RX Message Buffer.

If a new message is stored into a RX Message Buffer before the previous message in this Message Buffer can be read, then the original message is overwritten. There is no mechanism for preventing a new message from overwriting the current message in the RX Message Buffer. If such a loss of messages is not acceptable, then RX FIFO should be used for storing related messages.





1. **Reception - Message storage in FIFO Buffers:**

When the received message is stored in one or more RX FIFO Buffers or Common FIFO Buffers configured in RX Mode or GW Mode, then the Message counter value is incremented in the corresponding RX FIFO Status Registers or Common FIFO Status Registers.

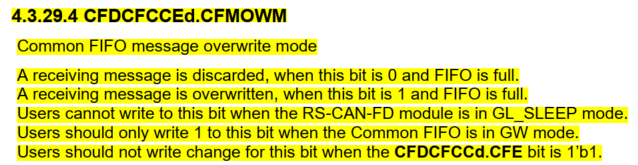
Depending upon the configuration of the FIFO Buffers, an Interrupt may also be generated.

When all the messages stored in the FIFO are read, then the FIFO Empty flag is set.  
If a new message is stored into the FIFO when the FIFO Message count matches the FIFO Depth (FIFO Full condition), the FIFO Message Lost flag is set and the new message will be lost (no overwrite of already stored messages will take place).

An appropriate value can be configured as warning level to generate an interrupt before the FIFO full condition occurs to avoid loss of a Message due to Overrun condition.

In GW mode, when a transmit/receive FIFO buffer is trying to receive a new message while the transmit/receive FIFO buffer is already full of data, the oldest data of the buffer will be overwritten with the message received or the message will be discarded.

And **CFDCFSTSd.CFMLT** bit is set to 1.



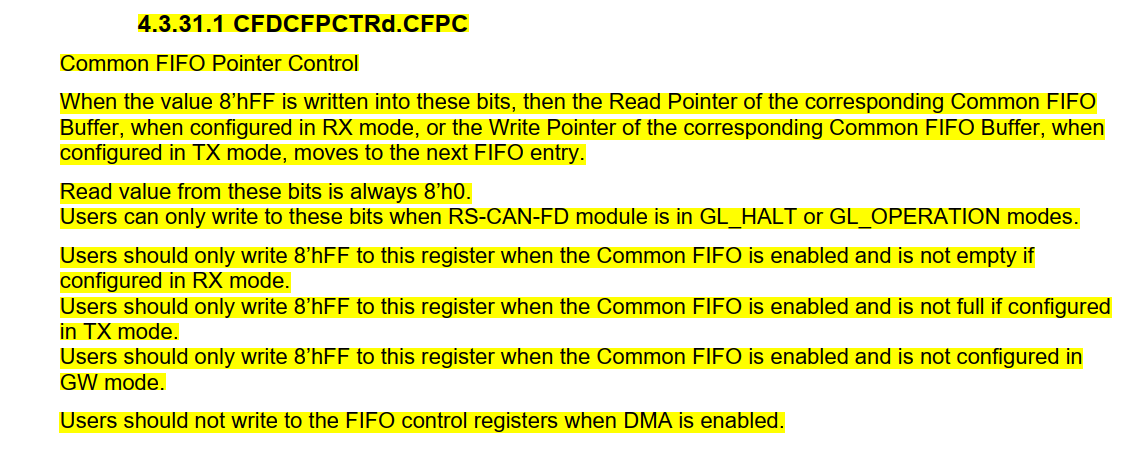


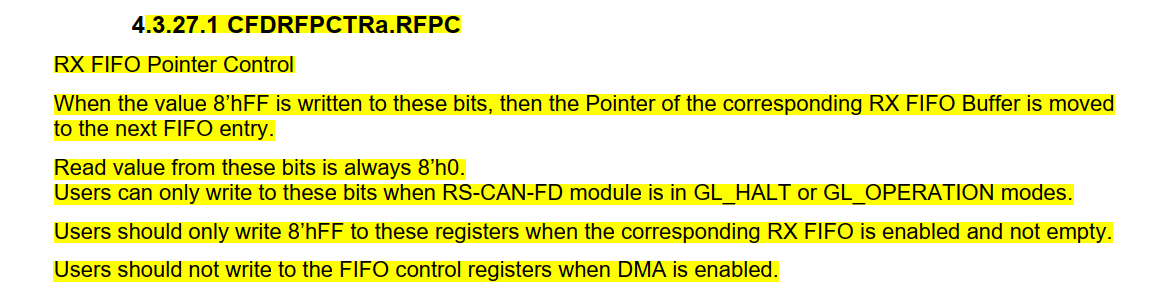
When **CFDCFCCEd.CFMOWM**=0:  
When writing of data is required due to reception of a new message while a transmit/receive FIFO buffer is full of data, the received message will be discarded. And **CFDCFSTSd.CFMLT** bit is set to 1.  
When **CFDCFCCEd.CFMOWM**=1:  
When writing of data is required due to reception of a new message while a transmit/receive FIFO buffer is full of data, the oldest data in the buffer will be overwritten with the received message.  
The read pointer of the transmit/receive FIFO buffer simultaneously moves to the next oldest message. Then, **CFDCFSTSd.CFMOW** bit is set to 1, which notifies that the oldest message has been overwritten with the received message.

n addition, in a case a CAN bus error or arbitration-lost for the transmitting message occurs in transmit/receive FIFO buffer full, the transmitting message is lost and re-transmission for the message is not performed. Then the read point moves to the next message automatically.  
Users should not write change for this bit when the **CFDCFCCd.CFE** bit is 1’b1.

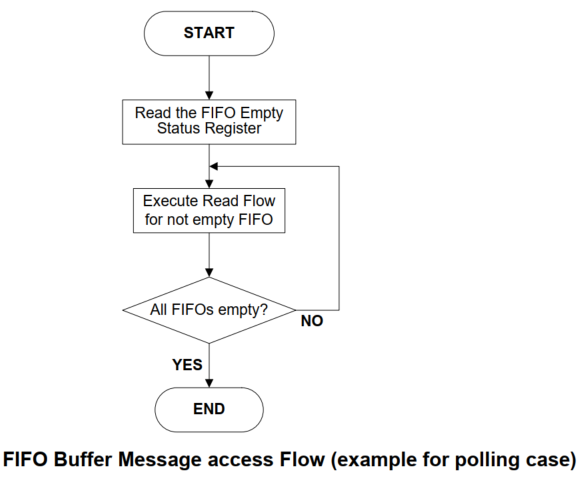
Common FIFO can set interrupt, when CAN frame reception is completed.  
Common FIFO can set interrupt, when FIFO is in full status in RX mode or GW mode.

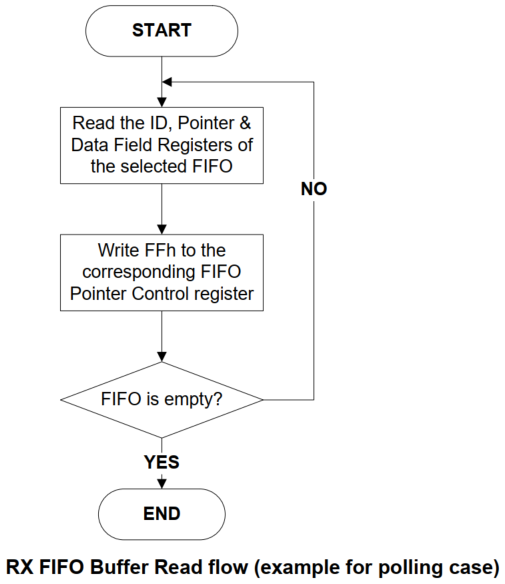
The RX FIFO Buffers and the Common FIFO Buffers configured in RX or GW Mode can be disabled at any time by clearing the **CFDRFCCa.RFE** or **CFDCFCCd.CFE** bit in the RX FIFO Configuration / Control Registers and the Common FIFO Configuration / Control Registers. When the **CFDRFCCa.RFE** or **CFDCFCCd.CFE** bit is cleared, then the message read and write pointers of the FIFO are cleared and are no longer active. Hence, all messages in the FIFO Buffers will be lost and no further messages can be stored into the FIFO.  
When the RX FIFO Buffers or Common FIFO Buffers configured in RX Mode is assigned as DMA channel then the SW should not access the FIFO Access Register of this FIFO buffer or write 8’hFF to the FIFO Pointer Control Register (**CFDCFPCTRd.CFPC** or **CFDRFPCTRa.RFPC**), because this could lead to unintended FIFO message decrement. The DMA channel will control the FIFO decrement by automatically.





Note: If the interrupt flag is set for a FIFO Buffer and then the FIFO is disabled, then the interrupt flag will not be cleared automatically. The interrupt flag should be cleared before disabling the FIFO.





1. **Transmission – Normal**

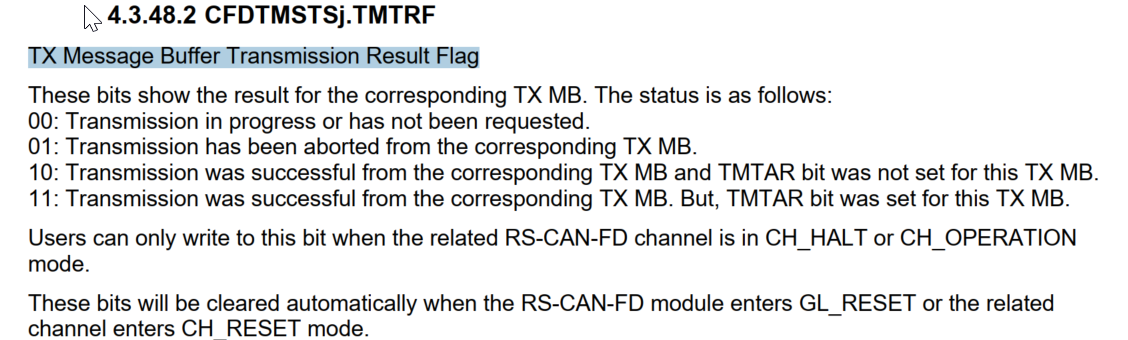
* ***Regular transmission mode***

If the Message Buffer is placed in regular transmission mode, the data frame or remote frame set in that Message Buffer can be transmitted.

Completion of regular transmission can be checked through the related TX Message Buffer Transmission Result Flag bits (CFDTMSTSj.TMTRF) in the TX Message Buffer Status Registers. These bits are set to 2’b10 or 2’b11 when the regular transmission is successful.

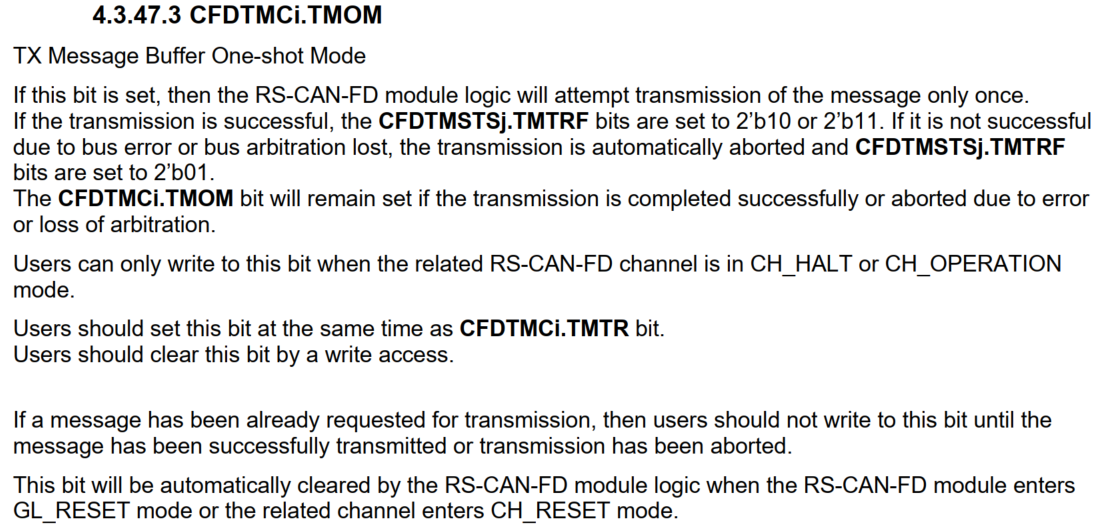
When arbitration is lost or an error occurs, message transmission will be attempted further if no transmission abort request is set for this transmission Message Buffer.

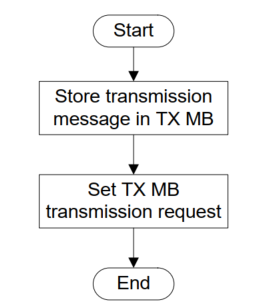
New internal transmission arbitration for this channel will be performed considering all Message Buffers with transmission request.



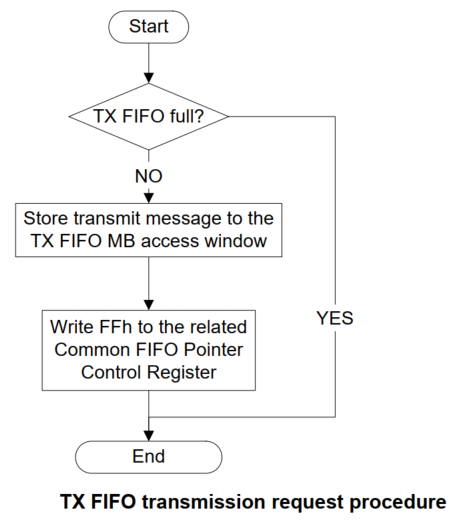
* ***One shot transmission mode***

When the **CFDTMCi.TMOM** bit of the TX Message Buffer Control Registers is set for a transmission Message Buffer, then the Message Buffer is placed in one-shot transmission mode and attempts to transmit a message only once.





1. **TX FIFO operation**



When the value 8’hFF is written into the corresponding FIFO Pointer Control Register, then the Message Count of the related FIFO is incremented by 1.

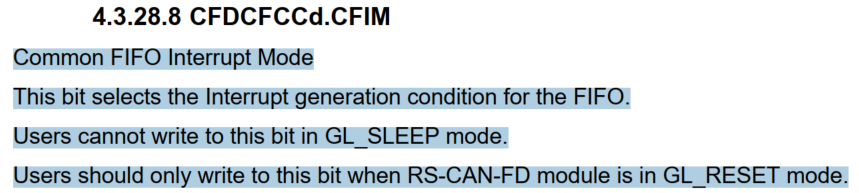
If the Message count matches the FIFO Depth, then the FIFO Full flag is set.

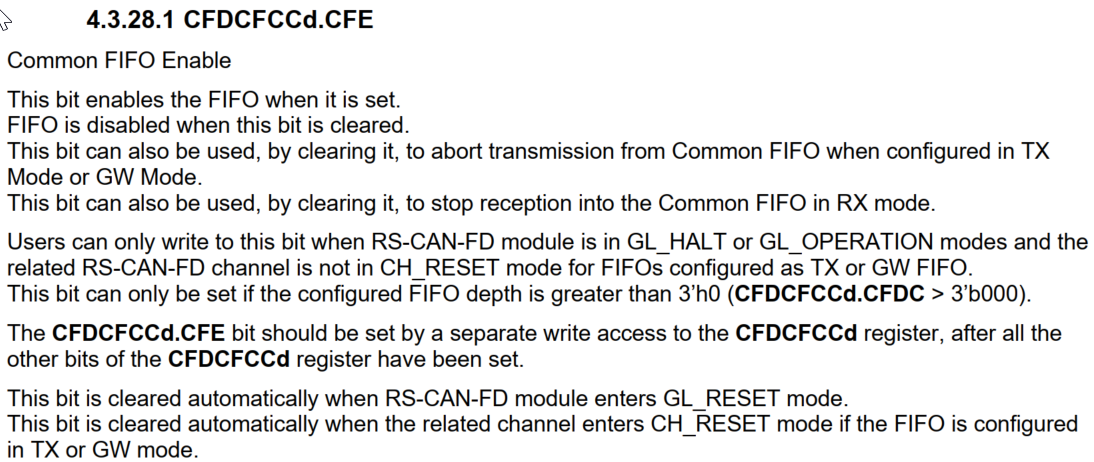
When a message is successfully transmitted from the TX FIFO, the Message Count value is decremented by 1.

When all the messages from the FIFO are transmitted, the FIFO Empty flag is set.

The Interrupt generation conditions for the TX FIFO buffers can be configured by configuring the **CFDCFCCd.CFIM** bit in the corresponding Common FIFO Configuration / Control Registers.  
If **CFDCFCCd.CFIM** bit is 1’b0, then interrupt is generated when last message is successfully transmitted from the TX FIFO buffer.  
If **CFDCFCCd.CFIM** bit is 1’b1, then interrupt is generated for every successfully transmitted message from the TX FIFO Buffer.  
Common FIFO can set interrupt, when CAN frame transmitted is completed.

The Common FIFO Buffers configured in TX Mode can be disabled by clearing the **CFDCFCCd.CFE** bit.





1. **GW FIFO Operation**

If a new message is stored into the FIFO when the FIFO Message count matches the FIFO Depth (FIFO Full condition), the FIFO Message Lost flag is set and the new message will be lost (no overwrite of already stored messages will take place).

In GW mode, when a transmit/receive FIFO buffer is trying to receive a new message while the transmit/receive FIFO buffer is already full of data, the oldest data of the buffer will be overwritten with the message received or the message will be discarded. The behavior is determined by setting **CFDCFCCEd.CFMOWM** bit.

When CFDCFCCEd.CFMOWM=0:  
When writing of data is required due to reception of a new message while a transmit/receive FIFO buffer is full of data, the received message will be discarded. And CFDCFSTSd.CFMLT bit is set to 1.

When CFDCFCCEd.CFMOWM=1:  
When writing of data is required due to reception of a new message while a transmit/receive FIFO buffer is full of data, the oldest data in the buffer will be overwritten with the received message. The read pointer of the transmit/receive FIFO buffer simultaneously moves to the next oldest message. Then, CFDCFSTSd.CFMOW bit is set to 1, which notifies that the oldest message has been overwritten with the received message.

The Interrupt generation conditions for the GW FIFO buffers can be configured by configuring the **CFDCFCCd.CFIM**

1. **Test mode**

* Channel specific test modes:

Each CAN channel can be configured into following test mode:

Basic test mode

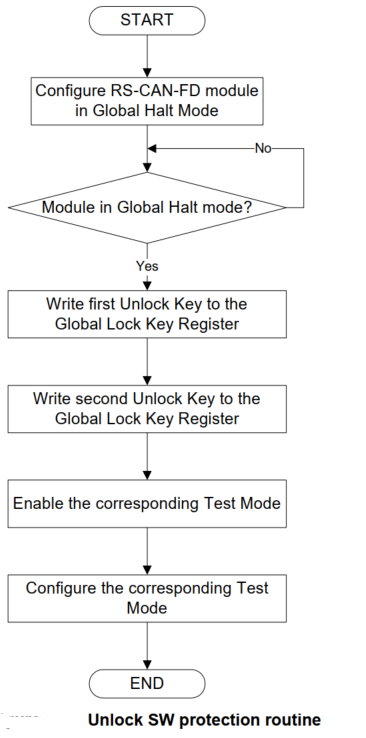
Listen-only mode

Self test mode 0 (External loop back mode)

Self test mode 1 (Internal loop back mode)

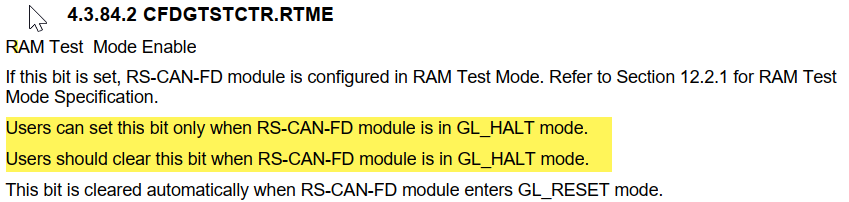
Restricted Operation mode

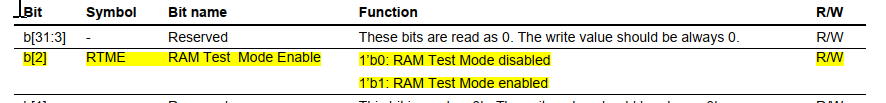
* Global test modes:



The RS-CAN-FD module can be configured into following test modes:  
RAM Test Mode  
Internal CAN Bus Communication Mode  
CRC Error Test.

+ Ram test mode: The RS-CAN-FD module can be configured in RAM Test Mode by setting the **CFDGTSTCTR.RTME** bit.





1. **Flexible CAN mode**

This is a mode in which it is possible to connect the CAN modules of 2 channels to a single CAN driver.

In Flexible CAN mode, each channel performs communication processing independently.  
However, when one of the channels transmits, the other channel will not return an acknowledge bit.

Flexible transmission buffer assignment configured in **CFDGFTBAC** register (section 4.3.82 - Global Flexible transmission buffer assignment Configuration Register ) and Flexible CAN mode configured in **CFDGFCMC (**section 4.3.80 - Global Flexible CAN mode Configuration Register**)** register should not be used simultaneously.