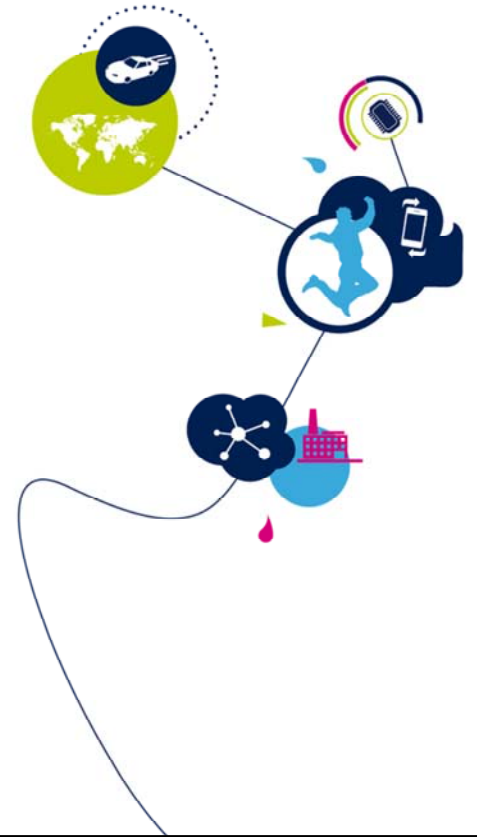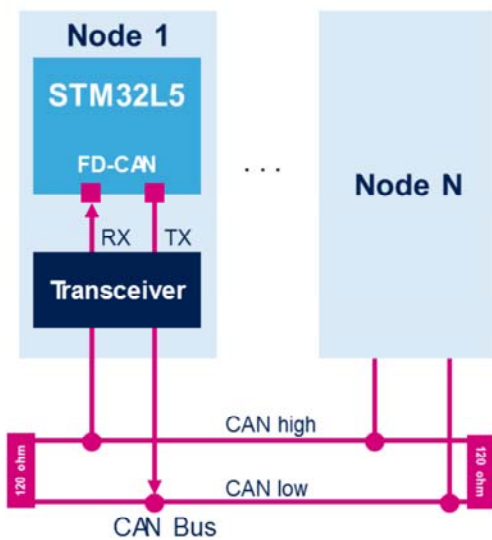# STM32L5- FD-CAN

Flexible Data-rate Controller Area Network interface
Revision 1.0

life.augmented

Hello and welcome to this presentation of the Flexible Data rate Controller Area Network interface.
It covers the main features of this interface, which is widely used to connect the microcontroller to a CAN network.

- An FD-CAN controller provides a communication interface with an external CAN transceiver via two pins

- One FD-CAN controller in the STM32L5

Application benefits

- Multi-master concept
- Object-oriented communication
- Real-time capability
- Low message transfer latency
- System wide message consistency

Node 1

STM32L5

FD-CAN

. . .

Node N

RX   TX

Transceiver

CAN high

CAN low

CAN Bus

120 ohm

120 ohm

The Flexible Data-rate Controller Area Network is a standard serial differential bus broadcast interface that enables the microcontroller to communicate with external devices connected to the same network bus.

The FD-CAN interface is highly configurable, enabling nodes to easily connect using just two wires.

Applications benefit from a multi-master concept with message priority, object-oriented communication (no node addressing, but content identification), real-time capability with low message transfer latency and system wide message consistency (error detection & management mechanism).

The STM32L5 microcontroller embeds one FD-CAN controller.
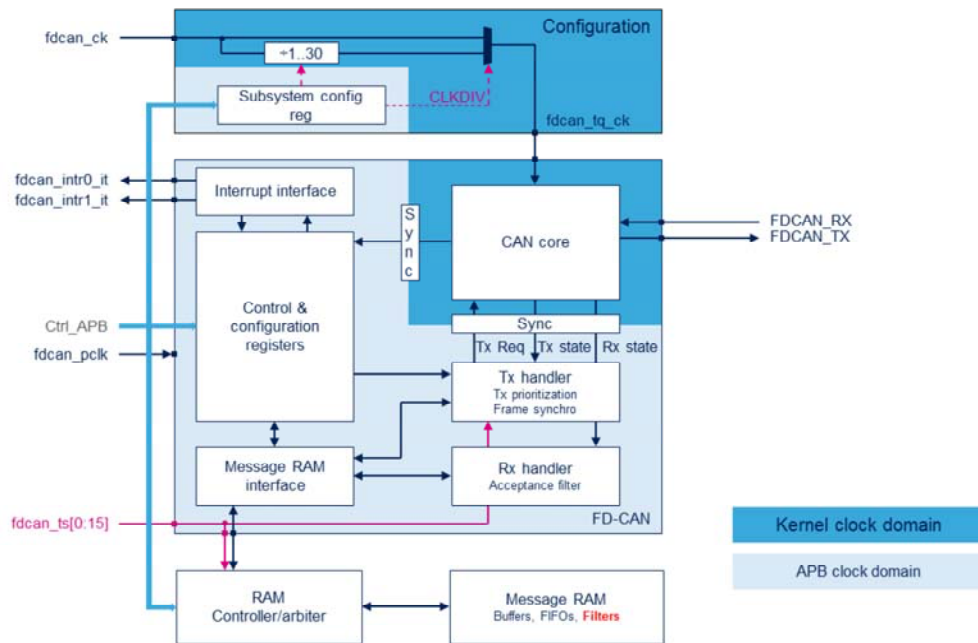
The FDCAN controller supports both the Basic Extended CAN protocol versions 2.0 A and B with a maximum bit rate of 1 Mbit/s, as well as the CAN FD protocol version 1.0 with up to 64 data bytes and a data bit rate of up to 8 Mbit/s.

The CAN core contains the Protocol Controller and receive / transmit shift registers.

It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

The Tx handler controls the message transfer from the Message RAM to the CAN core while the Rx handler controls the transfer of received messages from the CAN core to the external Message RAM.

Two clock domains are implemented: the APB bus interface and the CAN core kernel clock and therefore synchronization blocks are required between these two domains.

A shared 0.8-Kbyte Message RAM memory is available. This RAM is used to contain the filters, buffers and FIFOs.

Name	Type	Description	
fdcan_ck	Digital input	CAN subsystem kernel clock input	
fdcan_pclk	Digital input	CAN subsystem APB interface clock input	
fdan_intr0_it	Digital output	FDCAN interrupt0	
fdan_intr1_it	Digital output	FDCAN interrupt1	
fdcan_ts[0:15]	-	External timestamp vector ➢ This timestamp is provided by a timer contained in the FDCAN block	
FDCAN_RX	Digital input	FDCAN receive pin	Alternate function of GPIOs
FDCAN_TX	Digital output	FDCAN transmit pin	
Ctlr_APB	Digital input/output	Single APB slave interface with multiple psel for configuration, control and RAM access	

The CAN subsystem I/O signals and pins are detailed in this table.

Two clocks are provided to the FD-CAN unit:
- fdcan_ck, the kernel clock used to obtain the bit rate
- fdcan_pclk, which is the APB clock used to access memory-mapped registers and message RAM.

Two interrupt outputs enable the FD-CAN unit to report events to the Cortex-M33 processor.

An external 16-bit timestamp input port can be used by the FD-CAN unit to timestamp the transmission or the reception of a message.

This timestamp is provided by a timer contained in the FD-CAN block.

FDCAN_RX and FDCAN_TX have to be connected to the transceiver.

Finally, the APB slave interface is internally split into three parts, each of them having a dedicated chip-select:

configuration, control and RAM access.

- CAN protocol versions 2.0 A and B and CAN FD protocol version 1.0
    - Compliance with ISO 11898-1:2015.
        - CAN FD with max. 64 data bytes supported
        - Error-checking improvement, based on checksum field up to CRC 21 bits
        - Bit rates:
            - Arbitration Bit Rate up to 1 Mbit/s
            - Data Bit Rate up to 8 Mbit/s

- Supports
    - 2 maskable interrupts
    - Power-down support
    - CAN error logging
    - AUTOSAR and J1939 support
    - Separate signaling on reception of High Priority Messages

The FD-CAN controller conforms with the CAN protocol version 2.0 part A, B and ISO 11898-1: 2015 and CAN FD protocol with maximum 64 data bytes supported. Maximum bit rate in FD mode is 8 Megabits per second. The controller also supports 2 independent maskable interrupts, each one having 24 fully configurable interrupt flags.
The controller has a power-down mode.
It supports error logging, AUTOSAR, J1939 and separate signaling on reception of high-priority messages.

- The message RAM contains:
  - Two Receive FIFOs of three payloads each
  - Configurable Transmit FIFO / queue of three payloads (up to 64 Bytes per payload)
  - Configurable Transmit Event FIFO
  - 11-bit filter
  - 29-bit filter

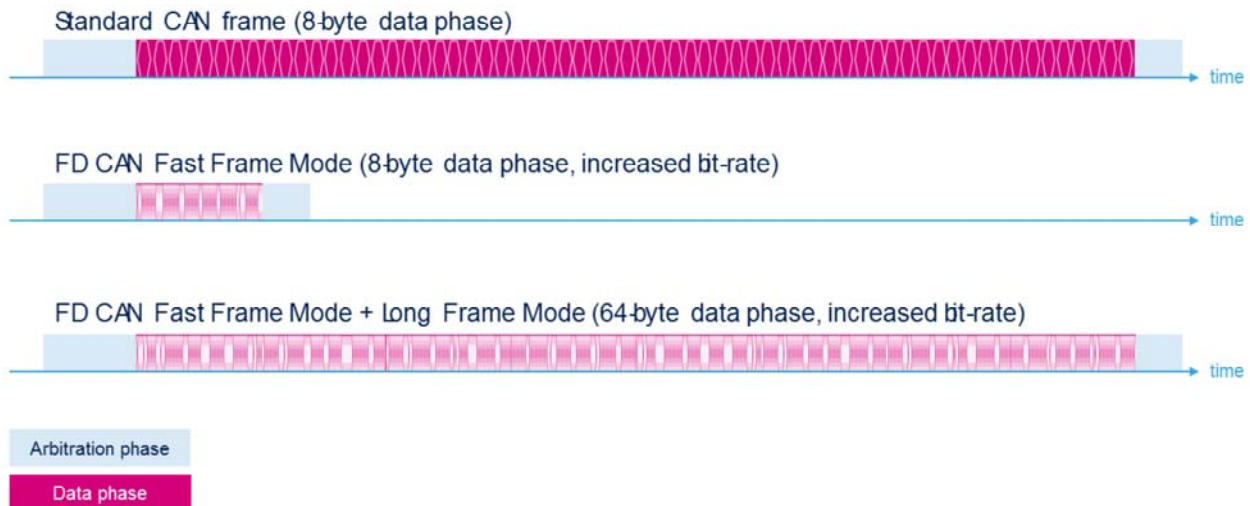| |
|---|
| 11-bit filter |
| 29-bit filter |
| Rx FIFO0 3 messages |
| Rx FIFO1 3 messages |
| Tx Buffers/TXFIFO 3 messages |
| Tx Events FIFO 3 events |

3 received messages can be stored in each of the two Rx FIFOs. The acceptance filter selects the FIFO to use. 3 messages to transmit can be stored as part of the message RAM configured either as a Tx FIFO or as 3 separate Tx buffers.
Each entry of the Rx FIFO and Tx FIFO or Tx buffers supports the maximum message size, 64 Bytes of payload.
The Tx Event FIFO stores Tx timestamps together with the corresponding Message ID.

Flexible Datarate (FD) CAN enhancements

Standard CAN frame (8-byte data phase)

FD CAN Fast Frame Mode (8-byte data phase, increased bit-rate)

FD CAN Fast Frame Mode + Long Frame Mode (64-byte data phase, increased bit-rate)

Arbitration phase
Data phase

There are two variants in the FD-CAN protocol:
1. Long Frame Mode (LFM), where the data field of a CAN frame may be longer that eight bytes, up to 64 bytes.
2. Fast Frame Mode (FFM), where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame. This high data rate is typically 8 Megabits per second.
Fast Frame Mode can be used in combination with Long Frame Mode.
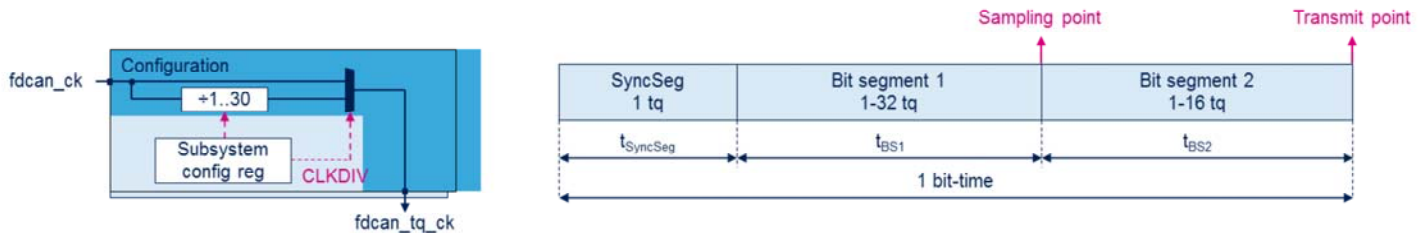
Bit timing

- The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point
  - bit time = tSyncSeg + tBS1 + tBS2
    - The length of tBS1 and tBS2 is programmable
  - Nominal bit time
    - time quantum (tq) = (FDCAN_NBTP[NBRP] + 1) * $t_{fdcan\_tq\_clk}$
  - Data bit time
    - time quantum (tq) = (FDCAN_DBRP[NBRP] + 1) * $t_{fdcan\_tq\_clk}$

The bit timing logic monitors the serial bus line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.
Regarding FD-CAN, the data bit-time and the nominal bit-time have two separate definitions, depending on the time quantum value.
The time quantum is the basic timing unit, obtained from the configuration unit and equal to tfdcan_tq_ck multiplied by a ratio from 1 to 512, programmed in the FDCAN_NBTP register for nominal bit-time and equal to tfdcan_tq_ck multiplied by a ratio from 1 to 32, programmed in the FDCAN_DBRP register for data bit-time.
The bit time is split into 3 segments: the synchronization segment, the bit segment 1 and the bit segment 2.
Each of these segments is an integer multiple of the time

quantum.

The duration of BS1 and BS2 is independently programmable for nominal bit time and data bit time. The data bit time applies when operating in FD mode and data are transmitted at the high data rate.
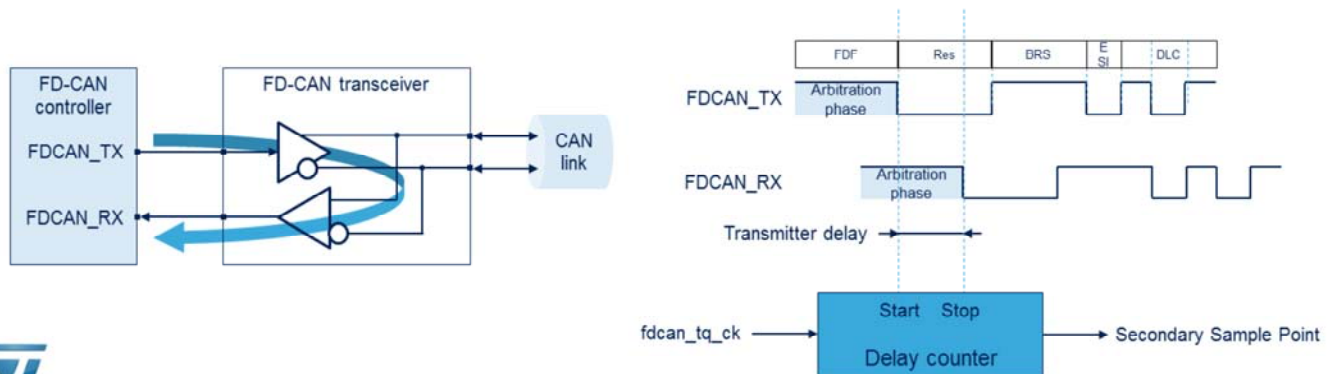
In order to adjust the on-chip bus clock, the CAN controller may shorten or prolong the length of a bit by an integral number of quanta.

The maximum value of these bit time adjustments are termed the Synchronization Jump Width, SJW, which is programmable from 1 to 4 time quanta.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay.

It is activated by setting the TDC bit in the DBTP register. The received bit is compared against the transmitted bit at the Secondary Sample Point.

The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset.

The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit, e.g. half of the bit time in the data phase.

FD-CAN operating modes
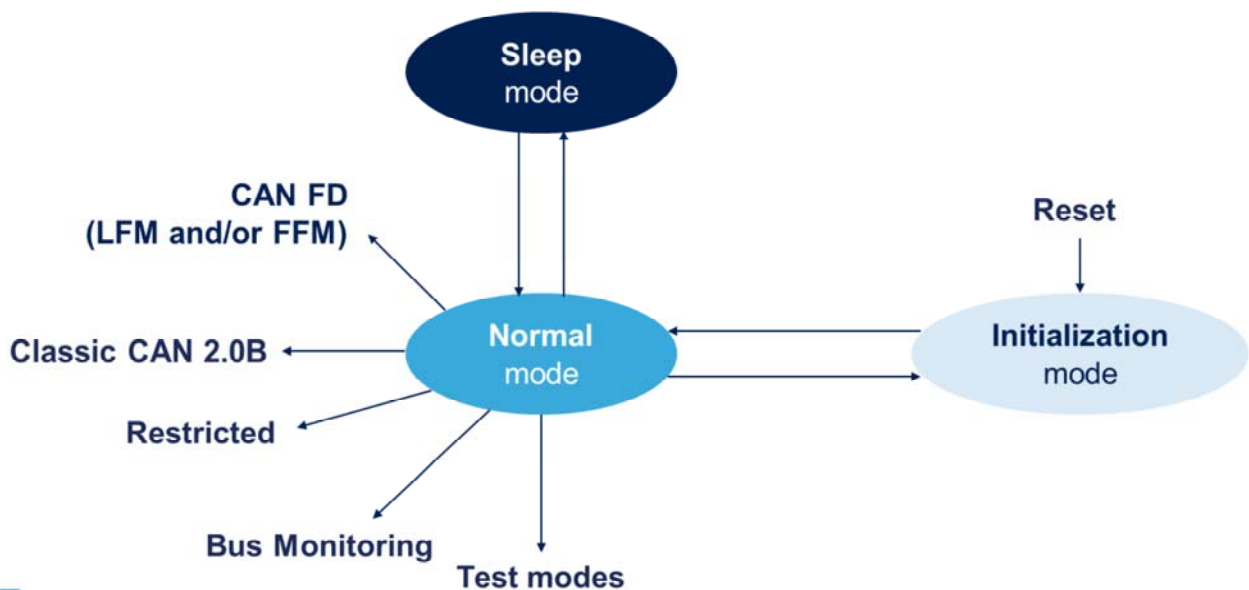
The FD-CAN has three main operating modes: Initialization, Normal and Sleep.

After a hardware reset, the FD-CAN enters Initialization mode via software.

In this mode:

- The peripheral must be configured (bit timings and RAM allocation). In the 'Bit timing' configuration, the rate is set then the sampling point is adjusted according to the actual serial bus-line.
- The CAN controller then synchronizes itself with the CAN bus by waiting for 11 consecutive recessive bits.

When the CAN is in Normal mode, the user can select different specific sub-modes:

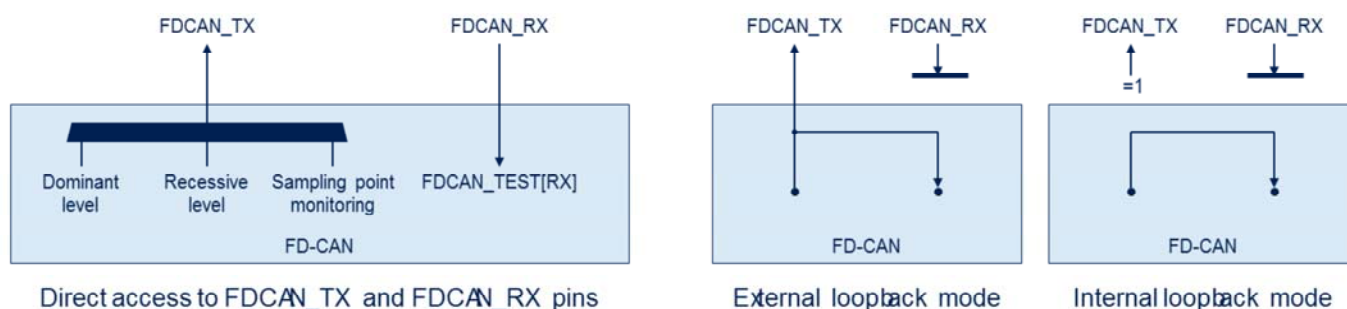- Classic CAN mode compatible with CAN

specification 2.0B

- FD CAN mode: it can be Long Frame and/or Fast Frame mode, named respectively LFM and FFM
- Restricted mode: the controller is able to receive data frames and acknowledge them, but does not send frames. It can be used in applications that adapt themselves to different CAN bit rates.
- Bus Monitoring mode: the controller is able to receive data frames (but cannot acknowledge them). It can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.
- Test modes detailed in next slide.

Upon a CPU request, the FD-CAN is put in Sleep mode which operates at a lower power, when bus idle state is detected.

FDCAN_TX   FDCAN_RX

Dominant | Recessive | Sampling point | FDCAN_TEST[RX]
level | level | monitoring

FD-CAN

Direct access to FDCAN_TX and FDCAN_RX pins

External loopback mode

Internal loopback mode

To enable write access to the FDCAN_TEST register, the Test bit in CCCR register must be set to 1, thus enabling the configuration of test modes and functions.

In test mode, software can control the state of the FDCAN_TX pin and can read the state of FDCAN_RX.

Through the FDCAN_TEST register, software can control the FDCAN_TX output: force dominant level, force recessive level, monitor the sample point.

The actual value at pin FDCAN_RX can be read from the RX bit in the FDCAN_TEST register. Both functions can be used to check the CAN bus physical layer.

These test modes should be used for production tests or self test only.

Furthermore, the FD-CAN controller supports two

loopback modes that are entered through control bits in the FDCAN_TEST and FDCAN_CCCR registers.
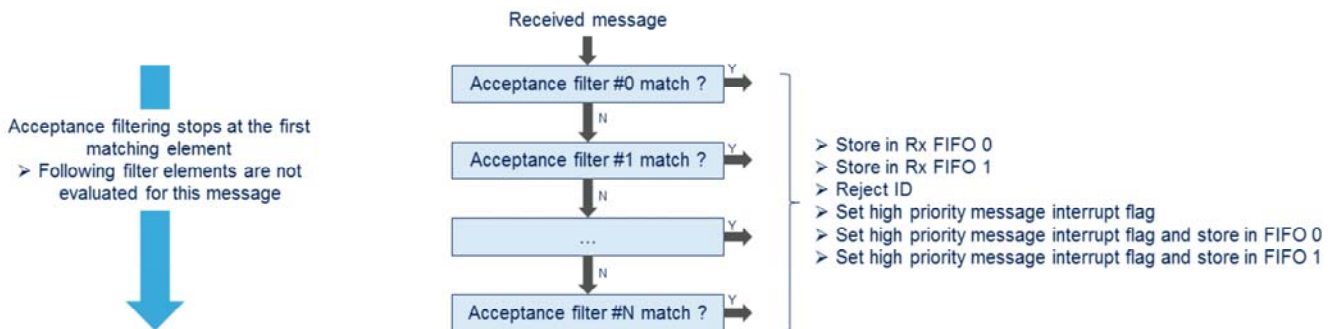
In external loopback mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data / remote frame) in Loop Back mode.

Internal loopback mode can be used for a hot self-test, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins.

In this mode, the FDCAN_RX pin is disconnected from the FDCAN and the FDCAN_TX pin is held recessive.

Acceptance filter

- The FD-CAN controller offers the possibility to configure two sets of acceptance filters located in message RAM, one for standard identifiers and another for extended identifiers
  - Each set has 28 entries : 28x 11-bit filter entries+ 28x 29-bit filter entries
  - These filters can be assigned to Rx FIFO 0 or Rx FIFO 1

Acceptance filtering stops at the first matching element
➢ Following filter elements are not evaluated for this message

Received message

Acceptance filter #0 match ?
Acceptance filter #1 match ?
...
Acceptance filter #N match ?

➢ Store in Rx FIFO 0
➢ Store in Rx FIFO 1
➢ Reject ID
➢ Set high priority message interrupt flag
➢ Set high priority message interrupt flag and store in FIFO 0
➢ Set high priority message interrupt flag and store in FIFO 1

The FD-CAN controller offers the possibility to configure two sets of acceptance filters, one for standard 11-bit identifiers and another for 29-bit extended identifiers. Each filter element is configurable for acceptance or rejection filtering

• Each filter element can be enabled/disabled individually

• Filters are checked sequentially, execution stops with the first matching filter element.

Software configures the number of active filter instances, the maximum is 28.

Acceptance filtering is started after the complete identifier has been received.

After acceptance filtering has completed, and if a matching Rx FIFO has been found, the Message Handler starts writing the received message data in 32-bit portions to the matching Rx FIFO.

- Each filter element can be configured as
  - Range filter (from - to)
    - $MessageID \in [SF1ID, SF2ID]$ for standard ID
    - $MessageID \in [EF1ID, EF2ID]$ for extended ID if EFT=0
    - $MessageID$ & $XIDAM[EIDN] \in [EF1ID, EF2ID]$ for extended ID if EFT=11
  - Filter for one or two dedicated IDs
    - $MessageID = SF1ID$ || $MessageID = SF2ID$ for standard ID
    - $MessageID$ & $XIDAM[EIDN] = EF1ID$ || $MessageID$ & $XIDAM[EIDN] = EF2ID$ for extended ID
  - Classic bit mask filter
    - $MessageID$ & $SF2ID == SF1ID$ & $SF2ID$ for standard ID
    - $MessageID$ & $XIDAM[EIDN]$ & $EF2ID == EF1ID$ & $EF2ID$ for extended ID

Each filter element can be configured as
- Range filter (from - to)
- Filter for one or two dedicated IDs
- Classic bit mask filter.
Regarding extended ID, the Extended ID AND Mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.
To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Message ID filtering
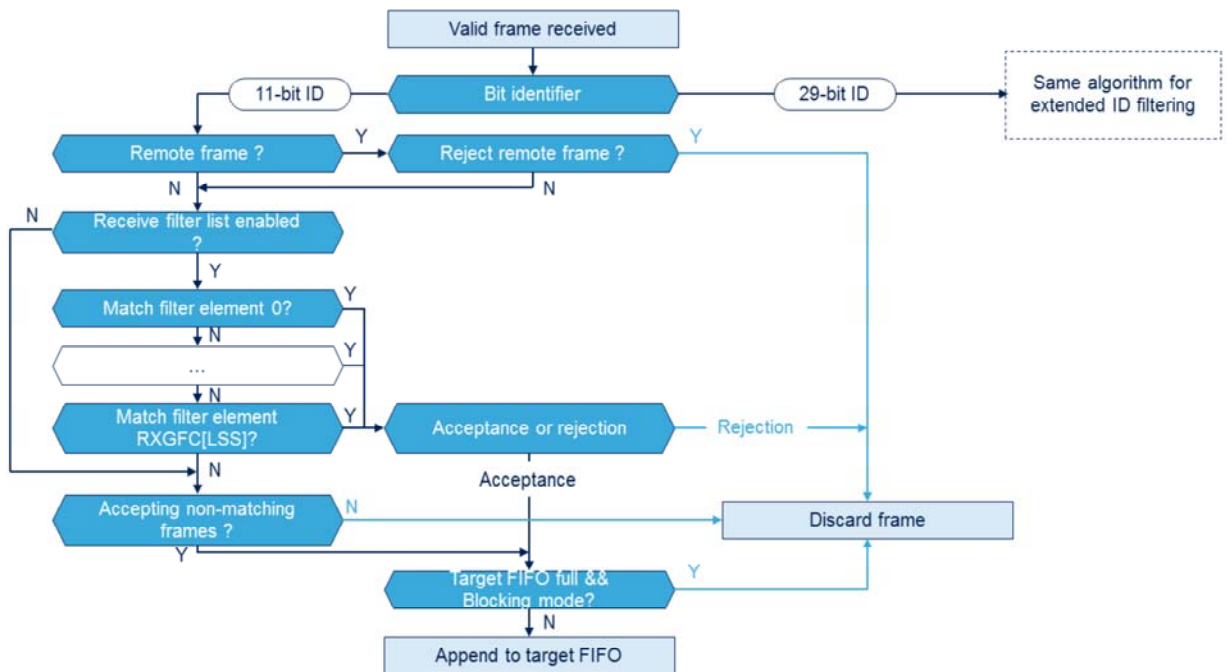
This algorithm describes the filtering sequence of frames received with a standard ID.

A similar algorithm is used to handle frames received with an extended ID, however the configuration of these two algorithms is done independently.

The First step is accepting or rejecting the remote frames.

Then when the receiver list is disabled, the filter elements are bypassed. Otherwise, the first matching element determines whether the frame is accepted or rejected.

When the receiver filter is disabled or no filtering elements have matched, the frame is either accepted or rejected.

At last, when the frame is accepted and the targeted Rx FIFO is not full, this frame is appended to the Rx FIFO. When the Rx FIFO is full and blocking mode is selected,
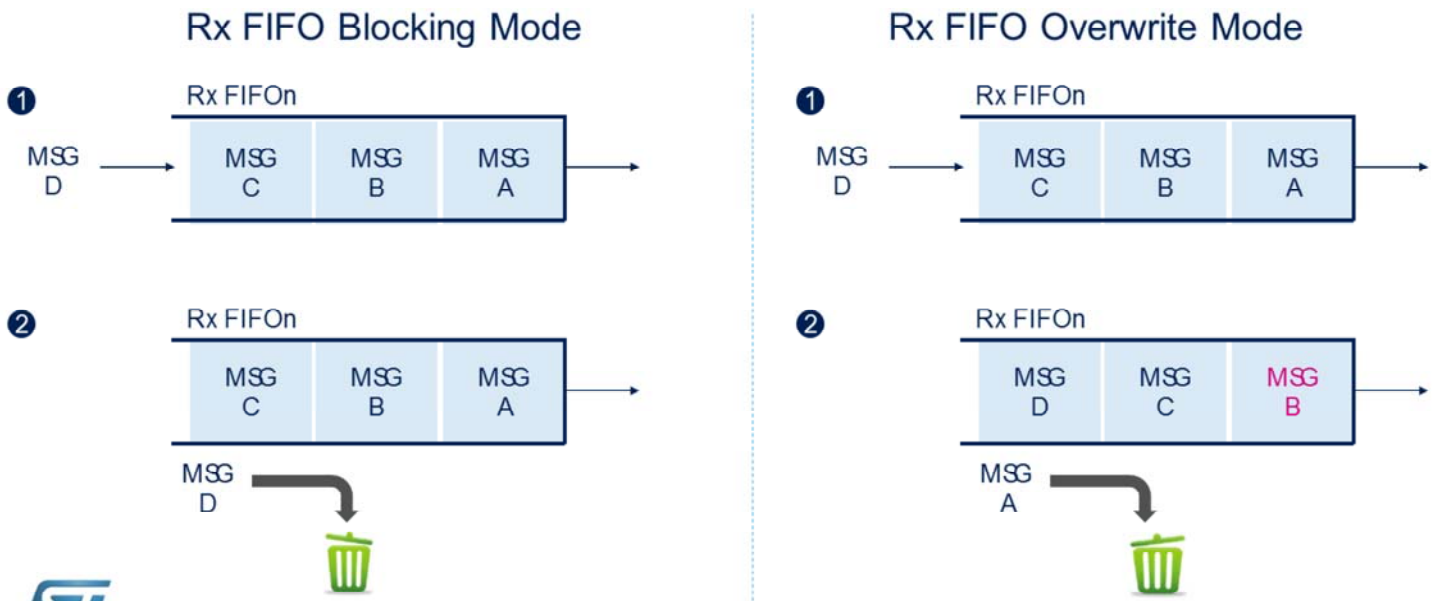
then the frame is discarded.

- The two Rx FIFOs implement the same synchronization mechanisms between hardware and software
  - When the FD-CAN controller writes a new message into Rx FIFOn, the read only Put index field is updated
  - Software has to indicate how many messages it has read in order to update the Get index

Rx FIFO0

Acceptance filters

Rx FIFO1

Rx FIFO 0 and Rx FIFO 1 can hold three elements each. Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element.
The read only registers FDCAN_RXF0S and FDCAN_RXF1S provide the following information:
- Position of the Put index
- Position of the Get index
- Number of pending messages
- FIFO full condition.

Rx FIFO Blocking Mode

❶ Rx FIFOn

MSG D → | MSG C | MSG B | MSG A | →

❷ Rx FIFOn

| MSG C | MSG B | MSG A | →

MSG D →🗑

Rx FIFO Overwrite Mode

❶ Rx FIFOn

MSG D → | MSG C | MSG B | MSG A | →

❷ Rx FIFOn

| MSG D | MSG C | MSG B | →

MSG A →🗑

The Rx FIFO blocking mode is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached, no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented.

If a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled.

In Rx FIFO overwrite mode, when an Rx FIFO full condition is signaled, the oldest message is discarded and the next message is accepted, as shown in the sequence on the right.

Put and get indexes are both incremented by one.

Tx FIFO / Tx Buffer operation

Tx FIFO mode
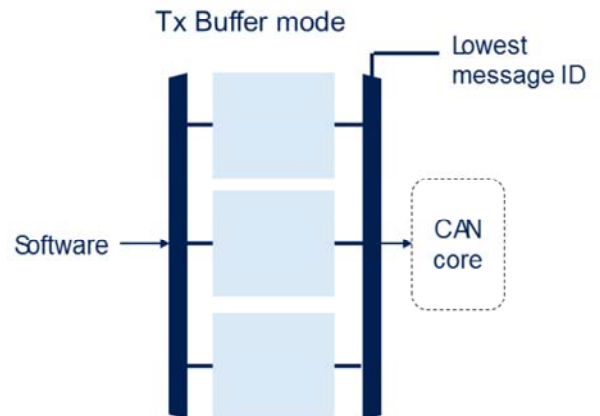
Tx FIFO

Software → [| |] → CAN core

- Software has to indicate how many messages it has written to the FIFO in order to update the Put index

- When the FD-CAN controller reads a message, it updates the Get index

Transmit cancellation — NA

Tx Buffer mode

Lowest message ID

Software → CAN core

- The three buffers are handled independently of each other
  - Software selects the buffer to which the next message is written
  - The FD-CAN controller indicates which buffers are available

Transmit cancellation — ✓

Three Tx Buffers can be set up for message transmission.
Either the Tx FIFO mode is chosen, in which all messages are transmitted in the same order that have been prepared by software.
Or the Tx Queue mode is chosen, in which the three message buffers are handled independently of each other.
Messages stored in the Tx Queue are transmitted starting with the message with the highest priority.
The FD-CAN controller supports transmit cancellation.
To cancel a requested transmission from a Tx Queue Buffer, software has to write a 1 to the corresponding bit position of register TXBCR.
Transmit cancellation is not intended for Tx FIFO operation.

Tx Event handling

- To support Tx event handling, the FD-CAN has implemented a Tx Event FIFO
  - After the FD-CAN controller has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element

To support Tx event handling, the FD-CAN has implemented a Tx Event FIFO.
The purpose of the Tx Event FIFO is to decouple handling transmit status information from transmit message handling.
A Tx Buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx Event FIFO.
This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx Buffer can be used for a new message immediately after successful transmission.
There is no need to save transmit status information from a Tx Buffer before overwriting that Tx Buffer.
If a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag is set.

An FD-CAN controller peripheral provides two independent interrupt lines.
This slide shows the complete list of possible interrupt events.

Mode	Description
Run	Active.
Sleep	Active ➢ Peripheral interrupts cause the device to exit Sleep mode
Low-power run	Active
Low-power sleep	Active ➢ Peripheral interrupts cause the device to exit Low-power sleep mode
Stop 0/Stop 1/Stop 2	Not available
Standby	
Shutdown	

Here is an overview of the FD-CAN sub-system low-power configuration modes.

The device is not able to perform any communications in Stop or Standby modes.

It is important to ensure that all CAN traffic is completed before the peripheral enters Stop or Standby modes.

- While the CPU Core is in Debug mode:
  - FD-CAN remains in its normal functioning mode
  - Registers of the type "reset on read" or "set on read" are disabled; reading them does not affect their value

While the CPU Core is in Debug mode (i.e. stopped at a breakpoint), then

- FD-CAN remains in its normal functioning mode. In particular, reception continues as normal and this may lead to reception overrun errors when FIFOs or buffers are full.
- Registers of the type "reset on read" or "set on read" are disabled; reading them does not affect their value.

CAN related peripherals

- Refer to these other peripherals:
  - **Reset and clock controller** (RCC) for more information about the CAN clock control and enable/reset
  - **Nested vectored interrupt controller (NVIC)** for more information about the mapping of the FD-CAN's interrupts
  - **General-purpose I/Os** (GPIO) for more information about the FD-CAN's input and output pins
  - **Debug Support** (DBG) for more information about the FD-CAN's behavior when the CPU is halted

For additional information, refer to the training for these peripherals which may affect FD-CAN behavior:

• Reset and clock controller (RCC) for more information about the CAN clock control and enable/reset.

• Interrupts for more information about the mapping of the FD-CAN's interrupts.

• General-purpose I/Os (GPIO) for more information about the FD-CAN's input and output pins.

• Debug Support (DBG) for more information about the FD-CAN's behavior in debug mode.

- For more details, please refer to the following resources:
  - Application note AN3154: Description of the CAN protocol used in the STM32 boot loader
  - Application note AN3364: Migration and compatibility guidelines for STM32 microcontroller applications
  - Application note AN5348: FDCAN peripheral on STM32 devices
  - Web (connection examples, available monitoring tools, and more)

Application notes covering the CAN topic are available on www.st.com.
To learn more about the CAN interface, you can also visit a wide range of web pages discussing the CAN communication protocol and bus monitoring tools.
Many digital oscilloscopes support direct reading and analysis of data transmitted over the CAN bus.