CAN-FD Protocol

The main limitation of the traditional CAN network is the restricted bandwidth, which is the maximum of 1 Mbit/s on a classical CAN bus. The CAN-Flexible Data Rate nothing but the CAN-FD protocol is a new standard developed by Bosch in the year of 2011 aiming to increase the bandwidth of the CAN bus while retaining the core characteristics of the traditional CAN bus. Then after one year of lab tests, they have released it in the year of 2012. The bandwidth restriction in CAN arises due to the arbitration feature available in the CAN protocol, which is used in the standard CAN.

On a CAN bus, multiple nodes are allowed to transmit at the same time before the higher priority message wins the arbitration. Suppose the sender A does at ID 7. In order for the arbitration scheme to work for all communication on the bus. Then the signals from a node must be able to propagate through the entire length of the CAN bus and back again. In order to ensure corresponding bits are compared in the arbitration scheme, even for the nodes, which are furthest away from each other. However, once the higher priority node gains access to the CAN bus only one node will be transmitting data. By utilizing this characteristic of the CAN bus it is possible to transmit data at a higher rate once there is only one node transmitting. The bus is the main idea behind the new standard called CAN-FD protocol.

What Is The Need of CAN-FD Protocol?

The automotive technologies are changing in every year as the customer expectation is increasing. If you will look onto the vehicle there are so many features are adding in automotive ECU. To prevent this they want a protocol that can send a bigger data packet in a single frame and also the data rate is 5-10 Mbps. In the current times, only the CAN protocol mostly used for ECU communication. It is nothing like there is no other protocol like FlexRay, MOST, etc. available but the CAN is better than them if you will compare them with cost and safety. Even if you also can get a better data rate but it will be high cost.

There is the main reason also that there are most of the vehicles are now having CAN protocol. The OEM does not want to change it which will cost them both from customer and development. Even if the OEM will change that since there is no way of option. then what Robot BOSCH will do if everyone will remove the CAN? How they will earn the money?

Then the Robot BOSCH started to research and development of advanced CAN protocol called CAN-FD protocol. The CAN-FD protocol has a lot of advanced features over the standard classical CAN protocol. Let us discuss below the difference between the CAN protocol and CAN-FD protocol.

Properties of CAN-FD Protocol

- Prioritization of messages:
- Guarantee of latency times;
- Configuration flexibility:
- Multicast reception with time synchronization:
- System wide data consistency.
- Multimaster
- Error detection and signalling.
- Automatic retransmission of corrupted messages as soon as the bus is idle again.
- Distinction between temporary errors and permanent failures of nodes and autonomous switching off of defect nodes.
- Compatibility with CAN protocol, every CAN FD node is able to receive and to transmit CAN messages according to ISO 18188-1

Safety in CAN-FD Protocol

In order to achieve the utmost safety of data transfer, powerful measures for error detection, signalling and self-checking are implemented in every CAN FD node.

- Error Detection: For detecting errors the following measures have been taken;
- Monitoring (transmitters compare the bit levels to be transmitted with the bit levels detected on the bus)
- Cyclic Redundancy Check
- Bit Stuffing
- Message Frame Check
- Performance of Error Detection
  The error detection mechanisms have the following properties:
- all global errors are detected.
- all local errors at transmitters are detected.
- up to 5 randomly distributed errors in a message are detected.
- burst errors of length less than CRC Sequence in a message are detected.
- errors of any odd number in a message are detected. Total residual error probability for undetected corrupted messages: less than message error rate * 4.7 * 10-11.

CAN-FD in OSI Layer

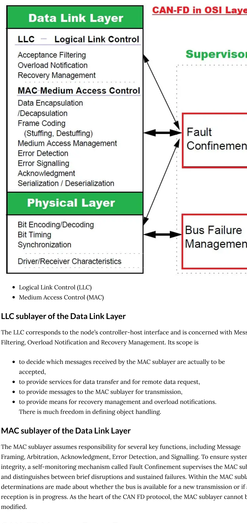
CAN-FD Physical Layer Design

The CAN-FD protocol physical layer design is different from standard classical CAN protocol. The CAN-FD controller is not that much change but in transceiver design, it is totally different to make the compatibility of new high bandwidth and payload support.

The Physical Layer is responsible for managing bits and determining the method of transmitting a signal. This involves describing Bit Timing, Bit Encoding, and Synchronization. However, the characteristics of the electrical driver/receiver for the Physical Layer are not specified in order to permit customization of transmission medium and signal level implementations for their specific applications.

CAN-FD Data link layer

The Data Link Layer handles frames and consists of the two sublayers:



- Logical Link Control (LLC)
- Medium Access Control (MAC)

LLC sublayer of the Data Link Layer

The LLC corresponds to the node's controller-host interface and is concerned with Message Filtering, Overload Notification and Recovery Management. Its scope is

- to decide which messages received by the MAC sublayer are actually to be accepted,
- to provide services for data transfer and for remote data request,
- to provide messages to the MAC sublayer for transmission,
- to provide means for recovery management and overload notifications.
  There is much freedom in defining object handling.

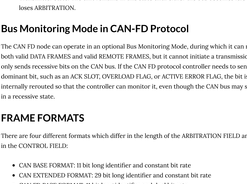
MAC sublayer of the Data Link Layer

The MAC sublayer assumes responsibility for several key functions, including Message Framing, Arbitration, Acknowledgment, Error Detection, and Signalling. To ensure system integrity, a self-monitoring mechanism called Fault Confinement supervises the MAC sublayer and distinguishes between brief disruptions and sustained failures. Within the MAC sublayer, determinations are made about whether the bus is available for a new transmission or if a reception is in progress. As the heart of the CAN FD protocol, the MAC sublayer cannot be modified.

CAN-FD Message Frame Format

The CAN-FD protocol is having its own frame format. Though there is not a lot of changes, some changes are there to support the new features as well as how it can also work with standard classical CAN network. Except for the Data frame, other frames are the same so let us discuss the CAN-FD data frame. If you will look at the below figure there are some fields have different. I would like to request you that if you have any doubts about the data frame please go in the current times, only the CAN protocol mostly used for ECU communication. It is nothing like there is no other protocol like FlexRay, MOST, etc. available but the CAN is better than them if you will compare them with cost and safety. Even if you also can get a better data rate but it will be high cost. comment section. I will clear your doubts.

Extended Data Length (EDL):



CAN – FD Arbitration

Data Consistency in CAN-FD Protocol

The host prepares messages for transmission, which are then transferred through the node's controller-host interface and LLC sublayer of the Data Link Layer to the MAC sublayer responsible for Message framing. These messages may be stored in a shared memory, and preserving data consistency of transmitted messages from this shared memory can be accomplished by at least one of two methods:

- Before transmission begins, the MAC sublayer must fill a temporary buffer with the entire message to be transmitted.
- While transferring the message to the MAC sublayer, the LLC sublayer is responsible for checking data errors. If any data error is found, the transmission cannot be initiated. If the transmission has already begun when a data error is detected, the node must switch to Bus Monitoring Mode. In such cases, receiving nodes will not receive a valid message.

Different Operation Modes in CAN-FD Protocol

The operational states of a CAN FD unit are categorized into four, namely: Integrating, Idle, Receiver, and Transmitter, and each of them is described as follows:

- **Integrating Mode:** During the start of the controller or during bus_off recovery, a unit is said to be in the Integrating state, during which it remains idle until it detects eleven consecutive recessive bits. Once it detects the required number of recessive bits, the unit switches to the Idle state.
- **Idle Mode:** When a unit is ready to receive or transmit data, it is in the Idle state. In this state, the unit is waiting for a START OF FRAME signal and can switch to either the Receiver or Transmitter state depending on the requirements of the communication.
- **Receiver Mode:** If a unit detects activity on the CAN bus and is not functioning as a Transmitter, it operates in the Receiver state.
- **Transmitter Mode:** When a unit initiates a message transmission, it is operating in the Transmitter state. The unit remains in this state until either the bus becomes idle or it loses ARBITRATION.

Bus Monitoring Mode in CAN-FD Protocol

The CAN FD node can operate in an optional Bus Monitoring Mode, during which it can receive both valid DATA FRAMES and valid REMOTE FRAMES, but it cannot initiate a transmission and only sends recessive bits on the CAN bus. If the CAN FD protocol controller needs to send a dominant bit, such as an ACK SLOT, OVERLOAD FLAG, or ACTIVE ERROR FLAG, the bit is internally rerouted so that the controller can monitor it, even though the CAN bus may still be in a recessive state.

FRAME FORMATS

There are four different formats which differ in the length of the ARBITRATION FIELD and in the CONTROL FIELD:

- CAN BASE FORMAT: 11 bit long identifier and constant bit rate
- CAN EXTENDED FORMAT: 29 bit long identifier and constant bit rate
- CAN FD BASE FORMAT: 11 bit long identifier and dual bit rate
- CAN FD EXTENDED FORMAT: 29 bit long identifier and dual bit rate

FRAME TYPES

Message transfer is manifested and controlled by four different frame types:
A DATA FRAME carries data from a Transmitter to the Receivers. There are four subtypes of DATA FRAME in CAN FD:

- DATA FRAME in CAN BASE FORMAT
- DATA FRAME in CAN EXTENDED FORMAT
- DATA FRAME in CAN FD BASE FORMAT
- DATA FRAME in CAN FD EXTENDED FORMAT

Recent Posts

Understanding Inheritance in CPP Programming Language
Objects of a Class in CPP Language
Classes in CPP Programming Language
How Crystal Oscillators Work in Microcontrollers: Ensuring Precise Timing and Stable Performance
Download and Install Turbo C++ Compiler

Archives

Select Month

Embedded Research Forum

Table Of Contents

8051 Microcontroller
8085 Microprocessor
8086 Microprocessor
About Us
Account
Adaptive AUTOSAR
Advanced driver assistance systems (ADAS)
Arduino
ARINC Protocol
ARM Microcontroller
Artificial Intelligence
Assembly Language
Automotive Architecture
Automotive BAP Protocol
Automotive ECU
Automotive Protocols
Automotive Safety
AUTOSAR
AUTOSAR ECM
AVR Microcontroller
Basic Electronics
Basic Understanding Of VLSI
BlueTooth Protocol
Boot Loader
ByteFlight Protocol
C Plus Plus (CPP) Tutorial
C Language
CAN Protocol
CAN-FD Protocol
CAN TP Protocol
CaneLayer
Canoe
CAPL Language
CrbbOS/RT
CMSIS RTOS
Contact Us
Contiki RTOS
Cookie Policy
CPU Design
Dashboard
Disclaimer
DMC
DoCAN Protocol
DoIP Protocol
DSRC Protocol
eCos RTOS
Edit
Embedded Linux
EtherNet Protocol
FlexRay Protocol
FlexRay Transport Protocol (ISO 10681-2)
Free RTOS
Guest Post
Home
HTTP (Hypertext Transfer Protocol): An Overview of the Internet's Most Widely Used Protocol
HwSw Interface
I2C Protocol
INTEGRITY Operating System
IoT
ISO-15031 Protocol
K-Line Protocol
KWP-2000 Protocol
LIN Protocol
Linux Basics
Linux Device Driver
Linux IPC
Linux Kernel
Linux System Architecture
Login
Long Range Wide Area Network (LoRaWAN) Protocol
Mastering MPI (3C Protocol: A Comprehensive Guide to Efficient Communication Between Devices
Mbed OS: Arm's Open-Source OS for IoT Devices
Message Queuing Telemetry Transport (MQTT) Protocol
Microcontroller
MODBUS Protocol
MOST Protocol
Motor Design
Nucleus RTOS
OBD-II
Operating System
Order Received
OSAL
OSEK
Payment
Power Electronics
PowerPC Processor
Privacy Policy
QNX RTOS
Raspberry-Pi
Register
Resistor
RIOT Operating System
Robotics
RT-Thread RTOS
RTLinux RTOS
Robotics Concept
SAE J1708 Protocol
SAE J1939 Protocol
SENT Protocol
SPI Communication Protocol: A Comprehensive Guide to Serial Peripheral Interface
Subscription
Terms and Conditions
Thank You
ThreadX RTOS: A Lightweight and Scalable RTOS for Embedded Devices
TinyOS
Transmission Control Protocol (TCP/IP)
UART Protocol
uC/OS RTOS
UDS Protocol
Understanding of Internet Protocol (IP) - The Backbone of the Internet
USB Protocol
User Datagram Protocol (UDP)
V2X Communication
VxWorks RTOS: A High-Performance Real-Time Operating System for Embedded Systems
Wi-Fi Protocol
Windows CE: An Evolution in GUI Based OS
XBEE Protocol
XCP Protocol
ZepPur RTOS

Automotive Electronics
Computer Science
Electronics Technology
Linux System
Programming Language
C
C++
Python
Robotics Technology
VLSI