FlexRay

Content Owner

Responsible: Ismael Calderon

FlexRay is a recent automotive protocol designed to be faster than CAN, although it is also more expensive. The key factor which makes FlexRay the best option in some contexts is that it guarantees the delivery of the messages. It is commonly used for higher-end applications, such as cruise control, suspension and some safety functions.

Table of contents

Overview

Features

- Multi-master serial communication technology.
    - up to 22 nodes.
- Physical layer: Optical/electrical, 1 or 2 unshielded twisted pair wires (= channels).
    - Support for single- and dual-channel configuration.
- Topology: flexible (bus, passive star, active star).
- Node architecture: 2x Transciever (one per channel) + Controller + Microprocessor (Host).
- Medium access: Event- and time-triggered messages
    - Deterministic. Message latency predetermined.
- Data rate: Up to 10 Mbit/s per channel.
- Data length: up to 254 bytes.
- Error handling:
    - Cyclic Redundancy Check.
    - Bus guardian.

Key concepts

- Synchronization.
- Static & Dynamic segment.

Scroll down to find more about these concepts.

Pros and Cons

Advantages

- Fast (10x faster than classic CAN).
- Reliable.

Disadvantages

- Expensive.
- Complex (steep learning curve, compared to other technologies).

Use

Higher-end applications, real-time systems and safety critical:

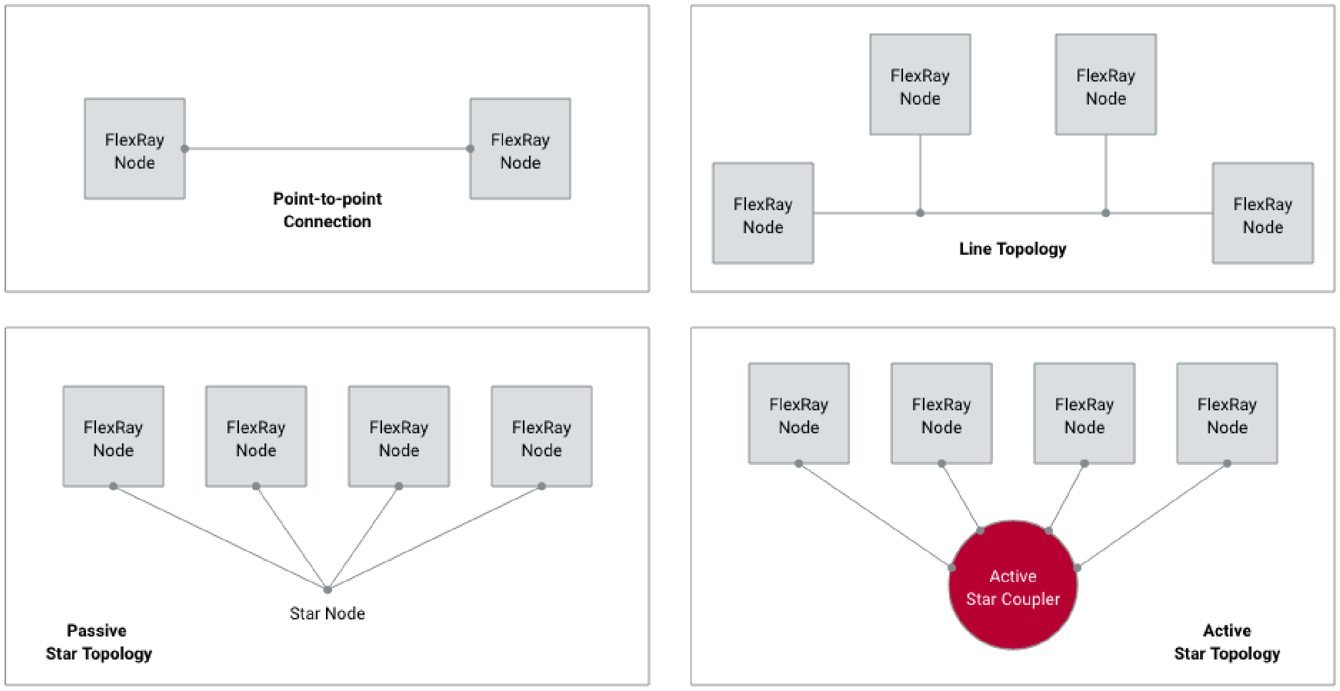
- Drive-by-wire.
- Active suspension.
- Adaptive cruise control system.
- Steer-by-wire.
- Brake-by-wire.

Details

Network architecture

FlexRay communication is not restricted to any specific physical topology. A simple point-to-point connection is just as feasible as a line topology or star topology. In addition, the system designer can choose between single channel (with or without redundancy) or dual channel communication.

- In the case of point-to-point connection, two FlexRay nodes are directly connected.
- In the case of three FlexRay nodes, the FlexRay nodes are interconnected via a central but passive star node.
- With at least four FlexRay nodes, it can be chosen between passive star topology and line topology.

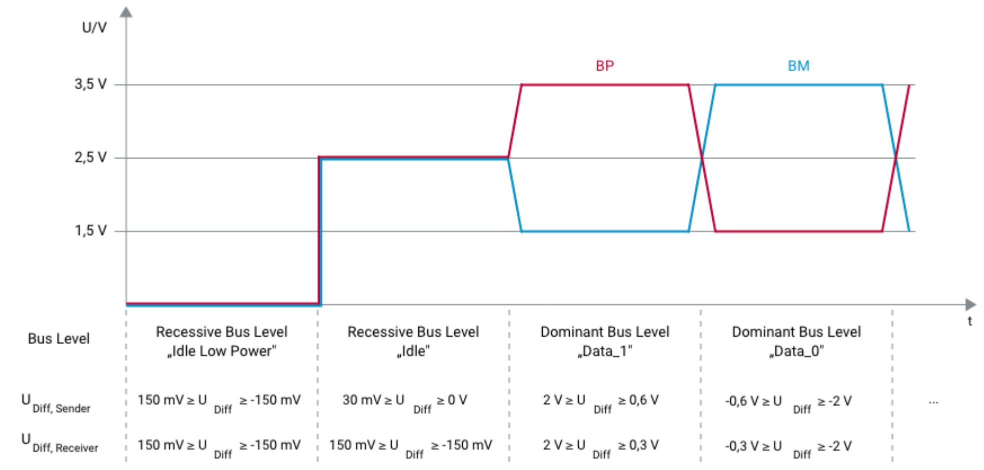


Wire length between the two needs should not exceed 24 m when the communication channels of the FlexRay cluster are operated at 10 Mbit/s. A reduction in data rate makes it possible to increase the distance between two FlexRay nodes. Regarding number of nodes supported, no more than 22 FlexRay nodes should be connected to a line.

At a physical level, communication within a FlexRay cluster is based on differential signal transmission. Therefore, the FlexRay bus consists of the two lines Bus Plus (BP) and Bus Minus (BM).

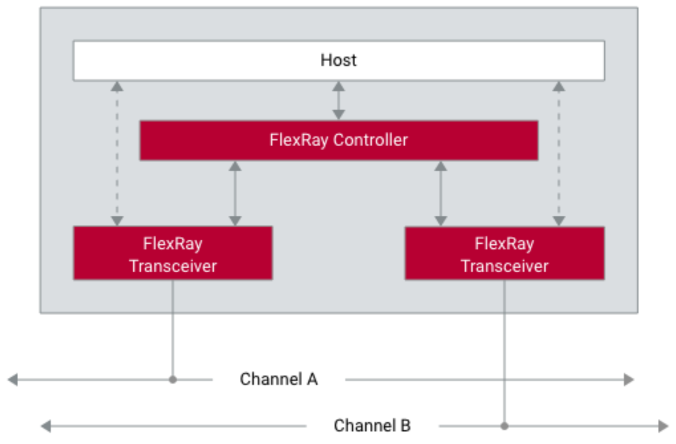
The FlexRay Electrical Physical Layer Specification defines four bus states, two of which are recessive and two dominant. The recessive bus state has a differential voltage of 0 volts and the dominant state has a differential voltage not equal to 0 volts:

- Recessive
  - Idle: In the Idle bus state, both BP and BM are driven to a nominal 2.5 V, giving a 0 V differential voltage.
  - Idle Low Power: If a node is in its low power state (Standby, Sleep, Go-To-Sleep), the Idle Low Power bus state is used, which also has a 0 V differential voltage, but the two bus wires are in this case at a nominal 0 V level.
    NOTE: The Idle state is a defined length of time used by each node to maintain clock synchronization.
- Dominant
  - Data_0 = Logical 0 bit: In the dominant Data_0 bus state (which represents logical state 0 or LOW), BP is driven to 1.5 V and BM is driven to 3.5 V, giving a differential voltage of –2.0 V.
  - Data_1 = Logical 1 bit: In the Data_1 bus state (which represents logical state 1 or HIGH), BP is 3.5 V and BM is 1.5 V, giving a differential voltage of +2.0 V.



FlexRay nodes

Since FlexRay is designed as a multimaster network, and analog to CAN, all nodes must be capable of both sending and receiving at any time. This is achieved with the following internal architecture:

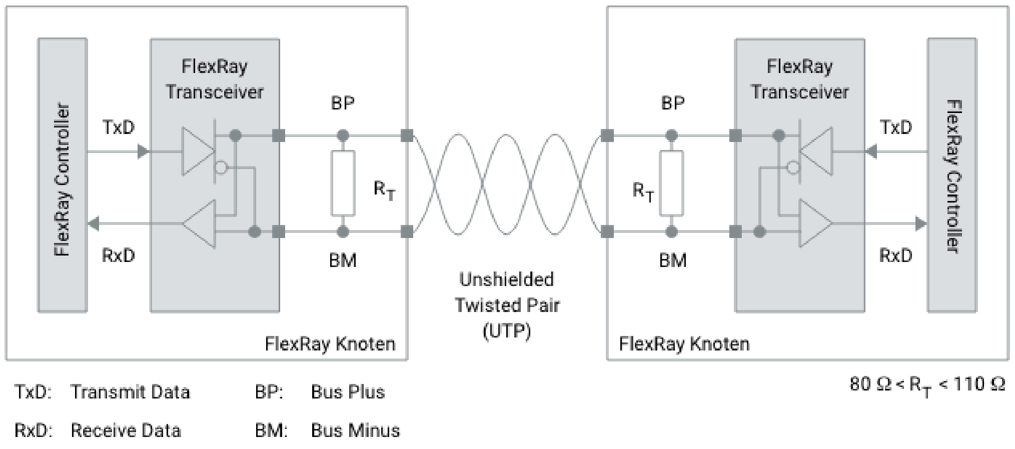


The FlexRay (Communication) Controller, often called CC, is an electronic component in a node that is responsible for implementing the protocol aspects of the FlexRay communications system.

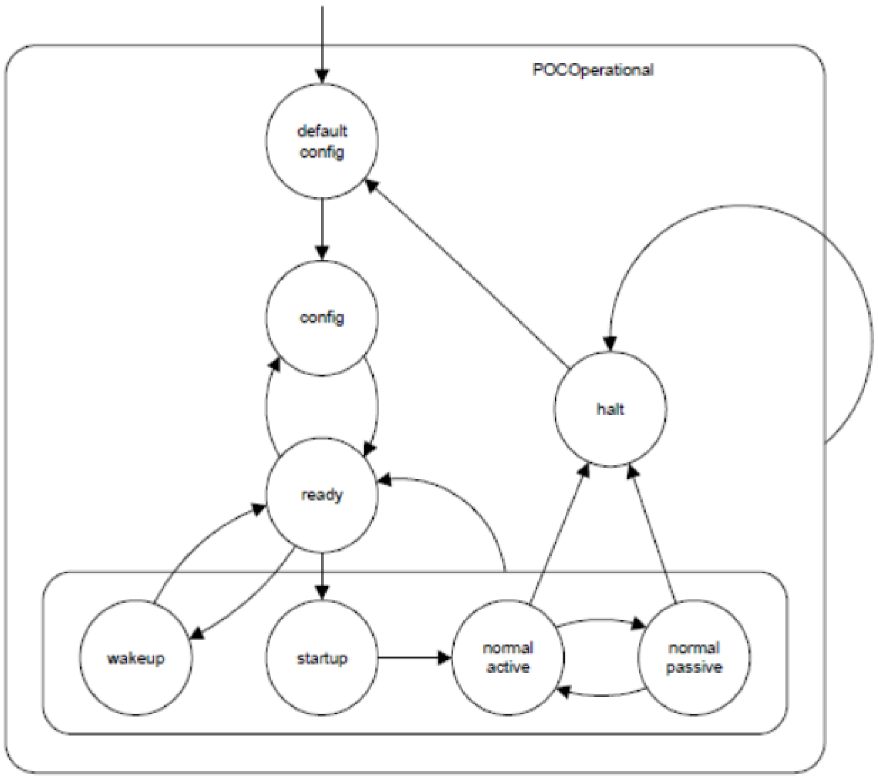
Note that a FlexRay node requires two separate transceivers for the two channels, so that they can be:

- configured and used to handle different communication if used as independent channels.
- used for redundancy, as an additional safety mechanism (the two channels must contain the same info).

Devices connected via FlexRay need good oscillators, and must resync periodically. This makes them more complex and expensive.



In the following diagram we can see the state machine for the *Protocol Operation Control (POC)*, which is the FlexRay state control logic within each FlexRay node, that makes possible to implement this lifecycle concept:



As seen in the diagram above, FlexRay controller has several states:

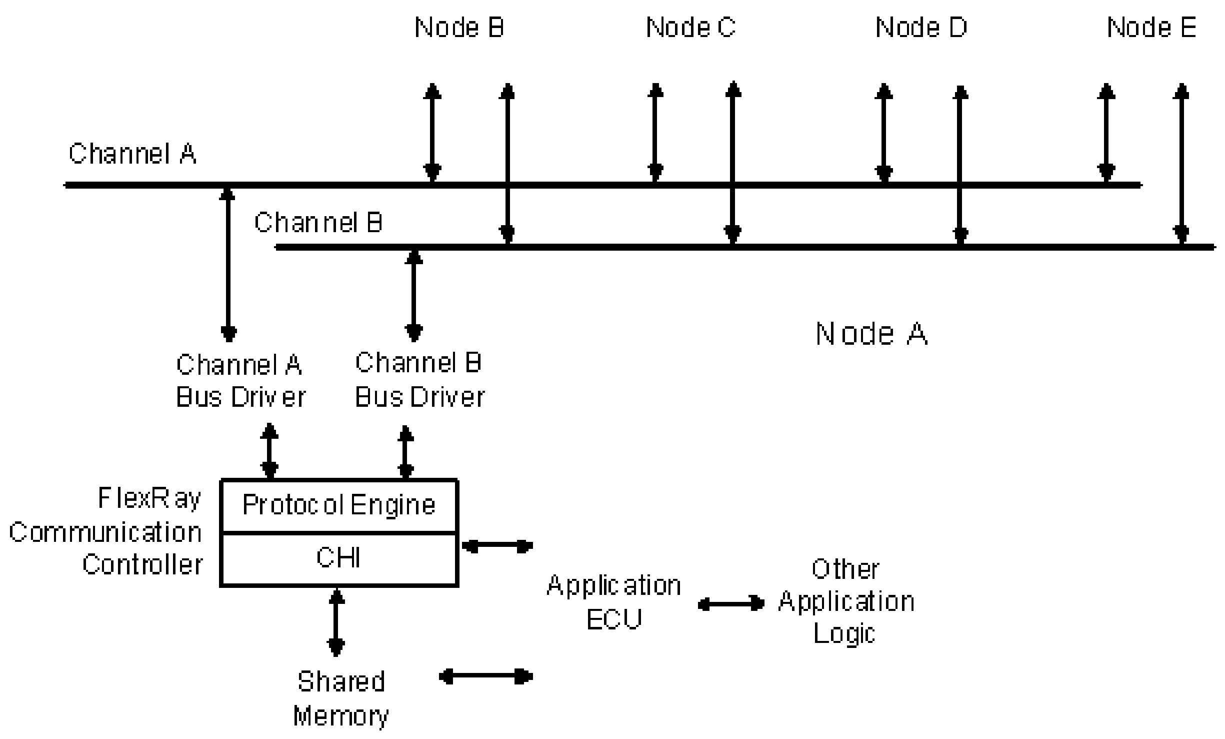
1. *Default config*: Frame communication is stopped, all node configuration memory is accessible, and physical layer pins are set to their inactive state. Default settings hardcoded in the firmware are applied.
2. *Config*: Frame communication is stopped, all node configuration memory is accessible, and physical layer pins are set to their inactive state. Customizable settings - not hardcoded - are applied.
3. *Ready*: Node is ready to send a wakeup, coldstart a cluster, integrate into an ongoing cluster, or be configured again.
4. *Wakeup:* Node is in wakeup phase.
5. *Startup:* Node is in startup phase

6. *Normal Active*: Normal operation state following a successful startup. Node is synchronized to the cluster, allowing continued frame transmission without disrupting other nodes. If synchronization problems occur, POC can transition to *Normal Passive*.
7. *Normal Passive*: Node can receive frames, but not transmit frames due to synchronization problems to the cluster. If synchronization improves, POC can transition back to *Normal Active*. If synchronization problems persist, POC transitions to *Halt*.
8. *Halt:* POC halts the node in preparation for reinitialization.
    a. A startup/wakeup routine is also requiredafter a fatal error, because the halt state of the failure model can only be left through thedefault config state.

For details about lifecycle, please check the Network Management section within this document.

Communication

A FlexRay communication system, usually called FlexRay cluster, is a group of FlexRay nodes and a physical transmission medium (FlexRay Bus) interconnecting all of them. This FlexRay cluster may be based on any of a number of different physical topologies, as explained in the previous section, and can make use of one or two channels, configured indepedently or as redundancy mechanism.



FlexRay is a proocol relying strongly on synchronization. Communication won't take place unless the nodes within a FlexRay cluster are synchronized. The synchronization mechanism is described later in this document.

Data communication in a FlexRay cluster is periodical and is based on a schedule. The concept is somehow similar to the LIN schedule tables, although more complex and flexible. In FlexRay we do not talk about schedule tables, but about communication cycles. The concept of communication cycle is described in detail below.

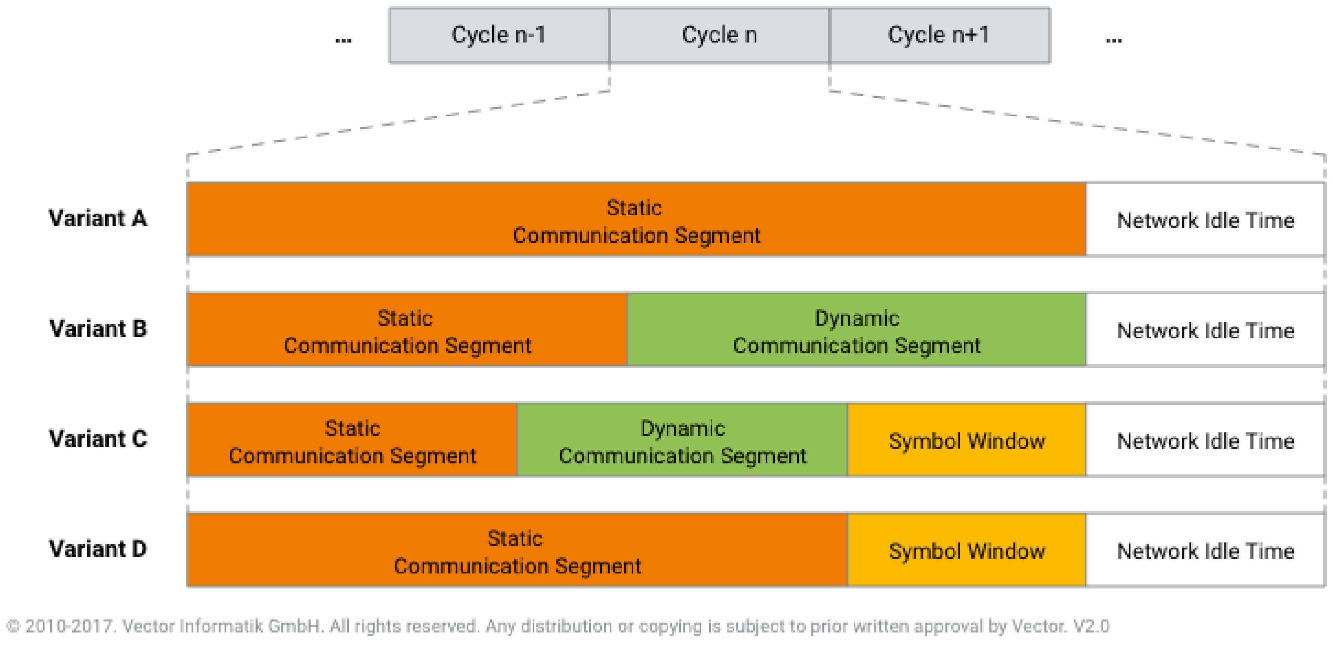
Communication cycle

In order to use the bandwidth efficiently, FlexRay may divide the communication cycle into 4 different sections:

- a static segment: used for cyclic data transfer, deterministic behavior.
- a dynamic segment: used for data transfer on demand. It serves to transmit event-triggered messages.
- a symbol window: used to transmit symbols like CAS (= Collision Avoidance Symbol).
- a synchronization phase, called NIT (= Network Idle Time): used to synchronize the local clocks of all nodes in the FlexRay cluster. No data communication in this phase, only sync.

The duration of a cycle is fixed when the network is designed, but is typically around 1-*5* ms. The smallest unit of FlexRay time is a “macrotick” and the FlexRay controllers use the idle time to synchronize themselves by adjusting their local clock so that the macrotick occurs at the same point in time on every node in the network.

Within a communication cycle, only the static segment and the synchronization phase are mandatory. With that in mind, several configurations are possible for a FlexRay communication cycle, as seen in the image below:


...
Cycle n-1
Cycle n
Cycle n+1
...
Variant A
Static Communication Segment
Network Idle Time
Variant B
Static Communication Segment
Dynamic Communication Segment
Network Idle Time
Variant C
Static Communication Segment
Dynamic Communication Segment
Symbol Window
Network Idle Time
Variant D
Static Communication Segment
Symbol Window
Network Idle Time
© 2010-2017. Vector Informatik GmbH. All rights reserved. Any distribution or copying is subject to prior written approval by Vector. V2.0

Which variant to use is something that needs to be defined during the network design phase.

As mentioned above, both static and dynamic segments are used to transfer data, but with different purposes and thus, different approaches as well. The following information is applicable to both the static and the dynamic segments:

- Segment is divided in time slots that are assigned to devices and their messages
  - A communication slot is an interval of time during which access to a communication channel is granted exclusively to a specific node for the transmission of a frame with a frame ID corresponding to the slot.
  - FlexRay distinguishes between static communication slots and dynamic communication slots:
    - When talking about static segment, the term *slot* is used.
    - When it comes to dynamic segment, we use the term *minislot.*
- Each node owns exclusively one or more slots.
- Per slot and channel exactly one frame is transmitted.
- Two nodes may share one slot on different channels.
- A slot may be empty (not owned by anyone).
- Scheduling is static and cannot be changed during runtime.

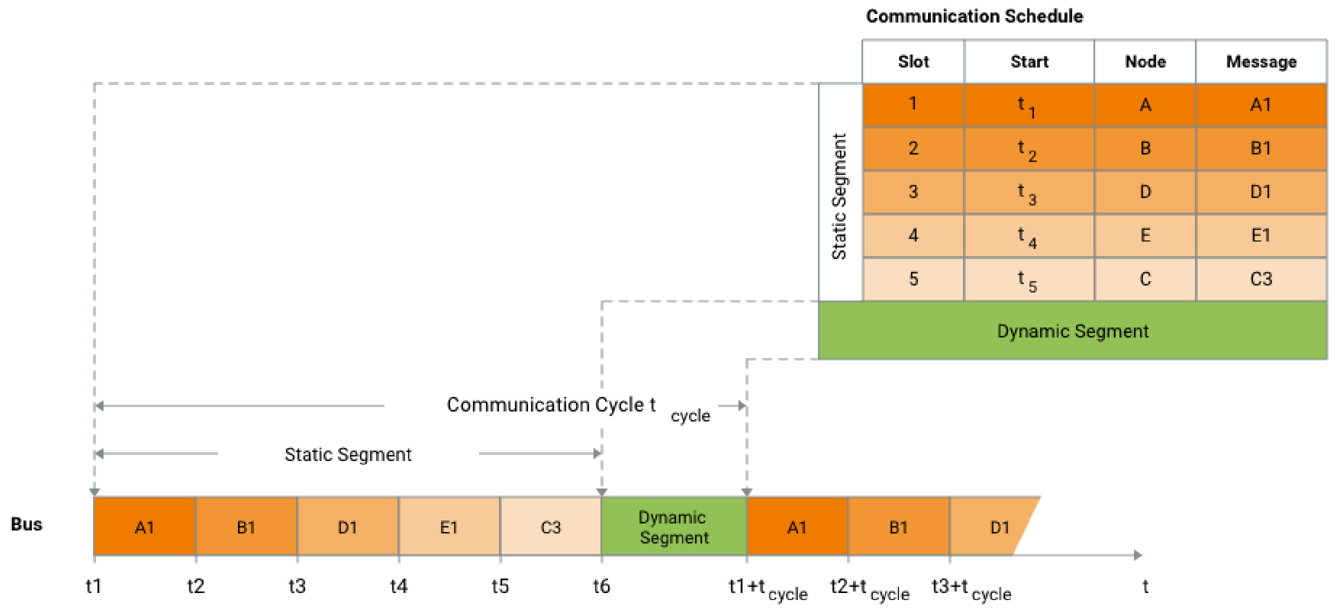
The two communication segments are described more in detail later in this document.

Using this communication principle, all the devices may send critical info in the guaranteed static segment slot and additional sporadic data depending on their priority in the dynamic segment.

Nodes can define several different cyclic offsets, as long as they won't cause any collisions.

e.g. if node A wants to send 14.3.4 and node B wants to send 14.7.8, this would cause a collision, because both *3 mod 4* and *7 mod 4* equals 3.

On receiving the message, the node must split the data into slots and process those indicated by the "update bits", which are a series of boolean values to tell the receiver which cycles have new info to process.



Synchronization

At least 2 devices are needed in the synchronization process. There is no time master, but not all nodes are capable of perform the synchronization process.

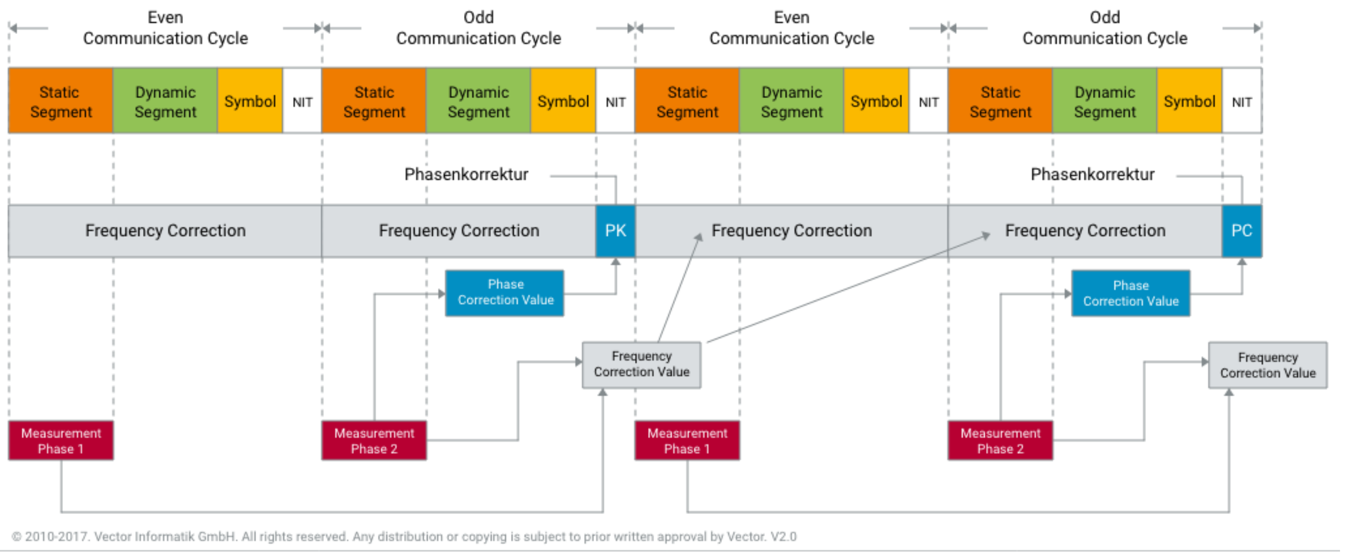
All sync-capable nodes synchronize to all sync-capable nodes.

To start a FlexRay cluster, at least 2 different nodes are required to send startup frames. The action of starting up the FlexRay bus is known as a cold-start and the nodes sending the startup frames are usually known as cold-start nodes. The startup frames are analogous to a start trigger, which tells all the nodes on the network to start. A retry mechanism for the initial synchronization at startup is included.

Once the network is started, all nodes must synchronize their internal oscillators to the network's macrotick. This can be done using two or more synchronization nodes. These can be any two separate nodes on the network that pre-designated to broadcast special sync frames when they are first turned on. Other nodes on the network wait for the sync frames to be broadcast, and measure the time between successive broadcasts in order to calibrate their internal clocks to the FlexRay time.

The sync frames are designated in the FIBEX configuration for the network. A static slot that is used by a node to transmit sync and startup frames is called key slot.

Once the network is synchronized and running, the network idle time (NIT) is measured and used to adjust the clocks from cycle-to-cycle to maintain tight synchronization.



Timing

The representation of time in the FlexRay protocol is based on a hierarchy that includes microticks and macroticks.

As described in the FlexRay specs:

- Macrotick (MT): an interval of time derived from the cluster-wide clock synchronization algorithm. A macrotick consists of an integral number of microticks. The actual number of microticks in a given macrotick is adjusted by the clock synchronization algorithm.
    - The macrotick represents the smallest granularity unit of the global time.
- Microtick (T): an interval of time derived directly from the CC's oscillator (possibly through the use of a prescaler). The microtick is not affected by the clock synchronization mechanisms, and is thus a node-local concept.
    - IMPORTANT! Different nodes can have microticks of different duration.

A communication cycle is made up of a defined number of macroticks, which are assigned to the individual segments. The macroticks are composed of a number of microticks.

Medium access

In the FlexRay protocol, media access control is based on a recurring communication cycle. Within one communication cycle FlexRay offers the choice of t wo media access schemes. These are a static timedivision multiple access (TDMA) scheme, and a dynamic mini-slotting based scheme (FTDMA):

Static segment

In the static segment all communication slots are of identical, statically configured duration and all frames are of identical, statically configured length.

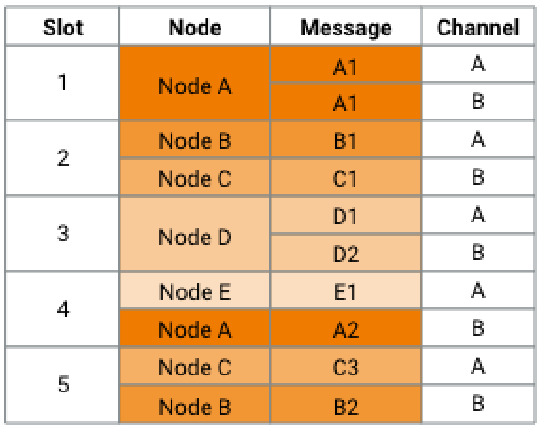
Key data:

- Synchronous communication.
- Time-triggered messages.
- TDMA = Time-Division Multiple Access: time divided in equally-sized slots and point of the time when a frame is transmitted on the channel fixed.
- All slots and frames in the static segment have the same size.
- Deterministic: all messages planned to be sent in the static segment are sent in each cycle, and the timing is known in advance.

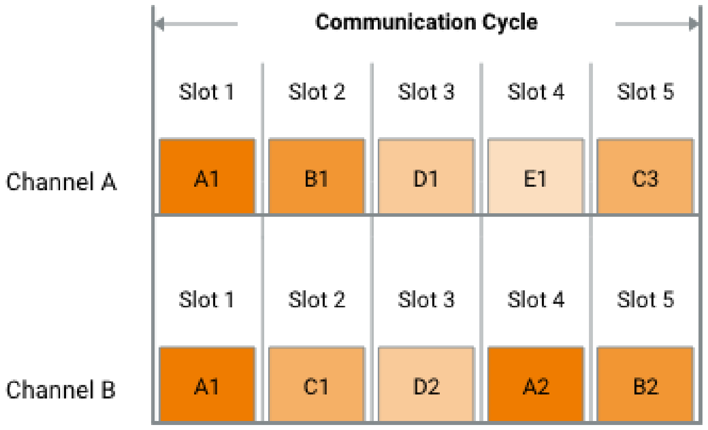
For communication within the static segment the following applies:

- If a node has a key slot or key slots, that node shall transmit a frame in the key slot(s) on all connected channels in all communication cycles.
- In slots other than the key slot(s), frames may be transmitted on either channel, or on both.
- Every slot is reserved for a specific device$^{(*)}$, so that all the devices can communicate on every cycle.
    - $^{(*)}$ Empty slots are allowed, i.e. not all slots must be assigned. If a slot is unassigned, the time reserved for it will be used anyway, but nothing will be transmitted.
    - Devices must always send data on their reserved slot, and if they have no information to send, a special "Null Frame" is sent in its place.
- In a given communication cycle, no more than one node shall transmit a frame with a given frame ID on a given channel.
    - It is allowed, however, for different nodes to transmit frames with the same frame ID on the same channel in different communication cycles.

- No collisions possible (unless network wrongly configured), because everybody knows when to send data.



Slot	Node	Message	Channel
1	Node A	A1	A
		A1	B
2	Node B	B1	A
	Node C	C1	B
3	Node D	D1	A
		D2	B
4	Node E	E1	A
	Node A	A2	B
5	Node C	C3	A
	Node B	B2	B



This static segment part of the communication cycle allows a deterministic behavior of the protocol, where you can predict when what will be sent. The disadvantage of the static segment is that, since the device has exclusive access to the bus, when it has nothing to send, it wastes bandwidth.

Dynamic segment

The dynamic segment is the portion of the communication cycle where the media access is controlled via a mini-slotting scheme, also known as Flexible Time Division Multiple Access (FTDMA). During this segment access to the media is dynamically granted on a priority basis to nodes with data to transmit.

Key data:

- Event-triggered messages prioritized based on its position within the segment: first messages have higher priority.
- Event-driven communication takes place.
- Used for low-priority data for the transmission of diagnostic information.
    - Network Management also sent here, in the first slot of the dynamic segment.
- Frames may have different lengths.
    - Length of a dynamic slots depends on the frame length.
    - Empty slots have the size of exactly one minislots.
- FTDMA = Flexible TDMA: Time is divided in equally-sized minislots that compose a slot and the point in time assigned to each slot is flexible, depending of the amount of information to transmit.
    - The dynamic segment is composed by a fixed number of macroticks, divided up into minislots
    - There might or might not be enough time for all dynamic messages to be sent
        - If message is sent, minislot expands into a message transmission
            - Each message can take up as much time as it needs, pushing the next slots forward in time.
        - If message isn’t sent, minislot elapses unused as a short idle period
        - All transmitters watch whether a message is sent so they can count minislots
    - When dynamic segment time is up, unsent messages wait for next cycle
        - Not possible to predict when a certain message planned for the dynamic segment will be sent, other than the first one.
        - The number of messages finally sent in the dynamic segment in a certain cycle depends on the amount and length of the messages need to be sent.
- Non-deterministic. Each node can decide whether to use the slots assigned to it or not, depending on its needs.
    - But to avoid affecting the deterministic data transmission of the static segment, the dynamic time segment always has the same length.

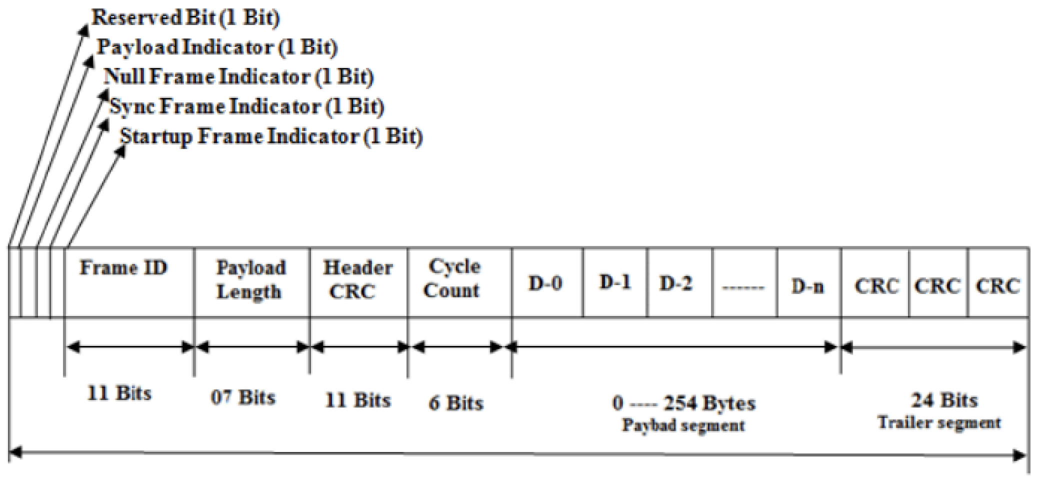
For communication within the dynamic segment the following applies:

- It takes the last part of the cycle, and it is not initially divided into slots.
- The dynamic time segment, if it is implemented, always follows the static segment.
- Every device, starting from the lowest ID, may send extra information of variable length (up to a total of 254 bytes per frame).
- This way, the dynamic segment can contain data of any number of devices, divided into variable length slots.

The downside of the dynamic segment is that, if a device has more data (messages) than what fits into one time slot, it needs to wait until it has exclusive access to the bus.

Frame structure

Each FlexRay message is composed of three parts: header, payload and trailer:



Frame ID

The frame ID defines the slot in which the frame should be transmitted. Each frame that may be transmitted in a cluster has a frame ID assigned to it.

The frame ID is 11 bits long, ranging from 1 to 2047. Note that zero is not included in this range, thus **not** a valid ID.

A frame ID is used no more than once on each channel in a communication cycle, in different slots. How to make the slot assigment unambiguous and identify in which slot(s) the ID is being sent?

- A node can set which frames carry its data with double-dotted number that the receiver must read and interpret in order to know to which message belongs the received frame.
- The format is X.Y.Z, where
  - X is the slot carrying the info
  - Y stands for the initial cycle offset from which the message is sent
  - Z represents the cyclic repeat offset (Z = 2^n).

For example, a FlexRay ID of 14.3.4 would mean that frame with ID 14 is sent for the first time in slot 3 with a repetition cycle of 4, i.e. every 4th slot. All of these slots - 14.3.4, 14.7.4, 14.11.4, etc. - belong to the same node (message with ID 14 is sent by one, and only one node!).

Slot multiplexing

With this name it is referred to the technique of assigning, for a given channel, slots having the same slot identifier to different nodes in different communication cycles.

A specific communication slot of a specific communication cycle can be identified by the pair of a slotnumber and a communication cycle number.

Slot multiplexing is allowed by the protocol in the static segment for slots which are notconfigured as key slots. The configuration of assignment must ensure that transmission in the cluster is conflict-free.

Slot multiplexing is allowed for all slots in the dynamic segment. It is up to the application or the configuration to ensure that transmission in the cluster is conflict-free.

Frame header

The Header is 5 bytes (40 bits) long and includes the following fields:

- Status Bits (5 bits): Flags
  - One initial reserved bit, not used so far.
  - PPI = Payload Preamble Indicator
    The payload preamble indicator indicates whether or not an optional vector is contained within the payloadsegment of the frame transmitted:
    - If the frame is transmitted in the static segment the payload preamble indicator indicates thepresence of a network management vector at the beginning of the payload.

        - If the frame is transmitted in the dynamic segment the payload preamble indicator indicates thepresence of a message ID at the beginning of the payload.
    - NF = Null Frame indicator
      The null frame indicator indicates whether or not the frame is a null frame, i.e. a frame that contains no useable data in the payload segment of the frame. Nodes that receive a null frame may still use some informationrelated to the frame.
        - If the null frame indicator is set to zero then the payload segment contains no valid data. All bytes inthe payload segment are set to zero, and the payload preamble indicator is set to zero.
        - If the null frame indicator is set to one then the payload segment contains data.
    - SYNC = Sync frame indicator
      The sync frame indicator indicates whether or not the frame is a sync frame, i.e. a frame that is utilized forsystem wide synchronization of communication. The clock synchronization mechanism makes use of the sync frame indicator.
        - If the sync frame indicator is set to zero then no receiving node shall consider the frame for synchro-nization or synchronization related tasks.
        - If the sync frame indicator is set to one then all receiving nodes shall utilize the frame for synchronization if it meets other acceptance criteria.
    - SU = Startup indicator
      The startup frame indicator indicates whether or not a frame is a startup frame. Startup frames serve aspecial role in the startup mechanism. Only coldstart nodes are allowed to transmit startup frames, and this flag must only be set to one in the sync frames of coldstart nodes (i.e all valid startup frames are also sync frames).
        - If the startup frame indicator is set to zero then the frame is not a startup frame.
        - If the startup frame indicator is set to one then the frame is a startup frame.
- Frame ID (11 bits)
- Payload Length (7 bits)
- Header CRC (11 bits)
- Cycle Count (6 bits): Cyclic Redundancy Check calculated over the sync frame indicator, the startup frame indicator, the frame ID, and the payload length.

Frame payload

The FlexRay payload field contains the actual data transferred by the frame. It contains 0 to 254 bytes (0 to 127 two-byte words) of data.

Each FlexRay frame contains processing data units (PDU) where ach consists of one or more signals. An update bit can be defined for each PDU, which informs recipients whether at least onesignal in the PDU was updated or not.

Note that a signal can be contained in more than one PDU and thus, appear in more than one frame.

Frame trailer

The trailer contains three 8-bit CRC (Cyclic Redundancy Check) used to detect errors. This CRC is calculated over the header segment **and** the payload segment of the frame and it includes all fields in these segments.

Note that the header includes also a CRC calculation. That means that two different CRCs are used to protect the data integrity in FlexRay.

Frame types

Data frames

A data frame is the standard type of frames to transmit application data. It can transport a maximum payload of 254 bytes. Frame layout is explained in the previous section.

Null frames

A "Null Frame Indicator" set to zero indicates the presence of a (formal) Null Frame, or an invalid frame. "Formal Null Frame may not indicate an 'invalid data frame'".

> e.g., Null frames used by the synchronisation algorithms are conceptually 'valid' entities. If the Null Frame Indicator is set to zero, other nodes know the data segment is not to be used.

If a slot is configured with a cyclic frame and the repetition is greater than 1, null frames will be present in this slot when the cycle does not meet the required repetition.

> e.g., Sync frame set for slot one with repetition 2 or 3 ->
> (cycle n) valid (cycle n+1) null (cycle n+2)   valid ...
> (cycle n) valid (cycle n+1) null (cycle n+2)   null (cycle n+3)   valid ...

The idea behind is that if a slot in the static segment is configured to a given node, either valid content is sent or a null frame.

Null frames will be seen in the static segment when collisions occur.

Null frames shall not be present in the dynamic segment. A null frame in the dynamic segment goes against the principle of its use: if you do not have anything to transmit, nothing will be sent in the dynamic segment, not even a null frame.

A null frame with the sync indicator set will be used at the start of a cluster initialisation.

Zero-length frames

Null Frames and data frames with zero length payload are not the same. A data frame with zero length payload can be sent as a valid data frame.

Sync frames

A sync frame is a frame whose header segment contains an indicator that the deviation measured between the frame's arrival time and its expected arrival time should be used by the clock synchronization algorithm.

See Synchronization section for more details.

Startup frame

A startup frame is a frame whose header segment contains an indicator that integrating nodes may use time-related information from this frame for initialization during the startup process. Startup frames are always also sync frames.

See Startup section for more details.

Error handling

The FlexRay network provides scalable fault-tolerance by allowing single or dual-channel communication. For security-critical applications, the devices connected to the bus may use both channels for transferring data. However, it is also possible to connect only one channel when redundancy is not needed, or to increase the bandwidth by using both channels for transferring non-redundant data.

Within the physical layer, FlexRay provides fast error detection and signaling, as well as error containment through an independent Bus Guardian.

Bus Guardian

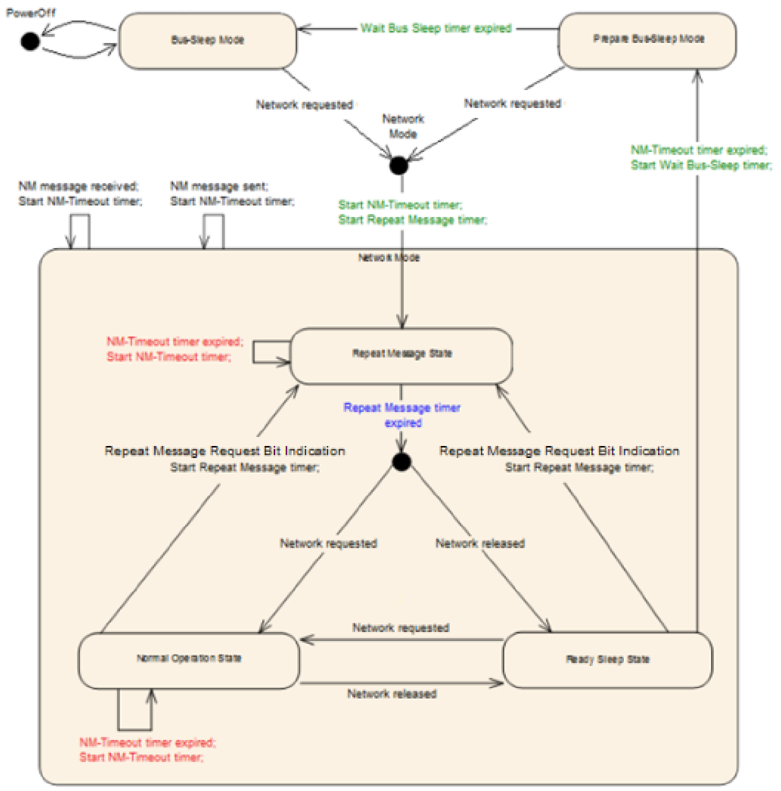
The Bus Guardian is a mechanism on the physical layer that protects a channel from interference caused by communication that is not aligned with the cluster’s communication schedule.

To implement its functionality, the bus guardian must know the communication schedule and the time in the FlexRay cluster. Ideally, the bus guardian does not rely on the local time base generated by the FlexRay controller, but instead generates its time base independent of the FlexRay controller. This is the only way a bus guardian can assure that a FlexRay node can only send in its time slots, because in addition to checking the time slots themselves, all errors of the FlexRay controller’s clock are detected as well.

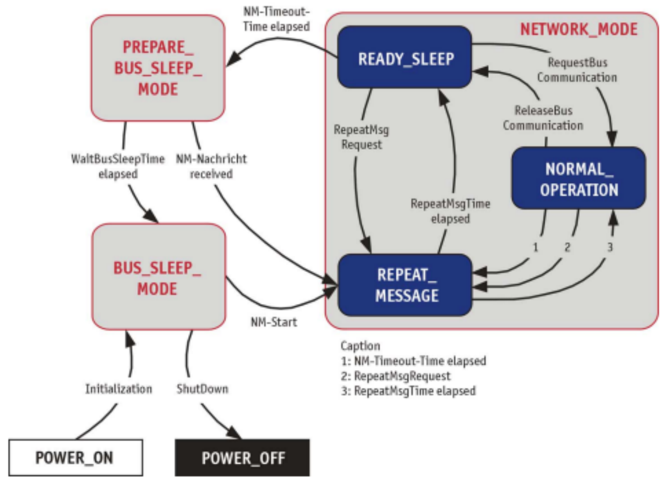
However, this means that the bus guardian must be equipped with nearly the same functions as the FlexRay controller, giving the bus guardian a similar level of complexity, which would result in increasing costs for FlexRay communication. For that reason, not in use in the real world.

Network management

The FlexRay lifecycle is summarized in the picture below, extracted from the AUTOSAR specification:


PowerOff
Bus-Sleep Mode
Wait Bus Sleep timer expired
Prepare Bus-Sleep Mode
Network requested
Network Mode
Network requested
NM-Timeout timer expired; Start Wait Bus-Sleep timer;
NM message received; Start NM-Timeout timer;
NM message sent; Start NM-Timeout timer;
Start NM-Timeout timer; Start Repeat Message timer;
Network Mode
NM-Timeout timer expired; Start NM-Timeout timer;
Repeat Message State
Repeat Message timer expired
Repeat Message Request Bit Indication Start Repeat Message timer;
Repeat Message Request Bit Indication Start Repeat Message timer;
Network requested
Network released
Network requested
Normal Operation State
Ready Sleep State
Network released
NM-Timeout timer expired; Start NM-Timeout timer;

Since the image quality is not good, here another view of the same diagram:



The different states we have are:

- *Bus Sleep Mode*: the purpose of the Bus-Sleep state is to reduce power consumption in the node, when no messages are to be exchanged. Transmission and reception capabilities can be switched off if supported by hardware.
- *Prepare Bus Sleep Mode*: the purpose of this state is to ensure that all nodes have time to stop their network activity before the *Bus Sleep* state is entered.
  - Bus activity is calmed down (i.e. queued messages are transmitted in order to empty all TX-buffers) and finally there is no activity on the bus in this state.
- *Network Mode,* which in turn consists of three internal:

- *Repeat Message*: it ensures, that any transition from *Bus Sleep* or *Prepare Bus Sleep* to the *Network Mode* becomes visible for the other nodes on the network. Additionally, it ensures that any node stays active for a minimum amount of time.
- *Normal Operation*: it ensures that any node can keep the NM-cluster awake as long as the network functionality is required.
- *Ready Sleep*: it ensures that any node in the NM-cluster waits with the transition to the *Prepare Bus Sleep Mode* as long as any other node keeps the NM-cluster awake.

In order to control the transition between states, NM messages are used. These are described in detail in AUTOSAR Network Management Protocol.

Wakeup

Before any NM message can be sent, the system has to be started or woken up (if sleeping), bringing the system to *Bus Sleep Mode.* The bus driver has the ability to wake up the other components of its node when it receives a wakeup pattern on its channel. So, at least one node in the cluster needs an external wakeup source.

The host completely controls the wakeup procedure. The communication controller provides the host the ability to transmit a special wakeup pattern on each of its available channels separately.

The wakeup pattern must not be transmitted on both channels at the same time. This is done to prevent a faulty node from disturbing communication on both channels simultaneously with the transmission. The host must configure, which channel the communication controller shall wake up. The communication controller ensures that ongoing communication on this channel is not disturbed.

The wakeup pattern then causes any fault-free receiving node to wake up if it is still asleep. Generally, the bus driver of the receiving node recognizes the wakeup pattern and triggers the node wakeup. The communication controller needs to recognize the wakeup pattern only during the wakeup (for collision resolution) and startup phases.

Wakeup pattern consists of a preconfigured number of the wakeup symbol. A wakeup symbol is composed of n bits that are transmitted at a LOW level followed by m bits of ‘idle’. The wakeup pattern is sent only on one channel at a time.

Startup

Before communication startup can be performed, the cluster has to be awake. The transition from "sleep" to "wake" does not need necessarily to be initialited by a sstartup-capable node.

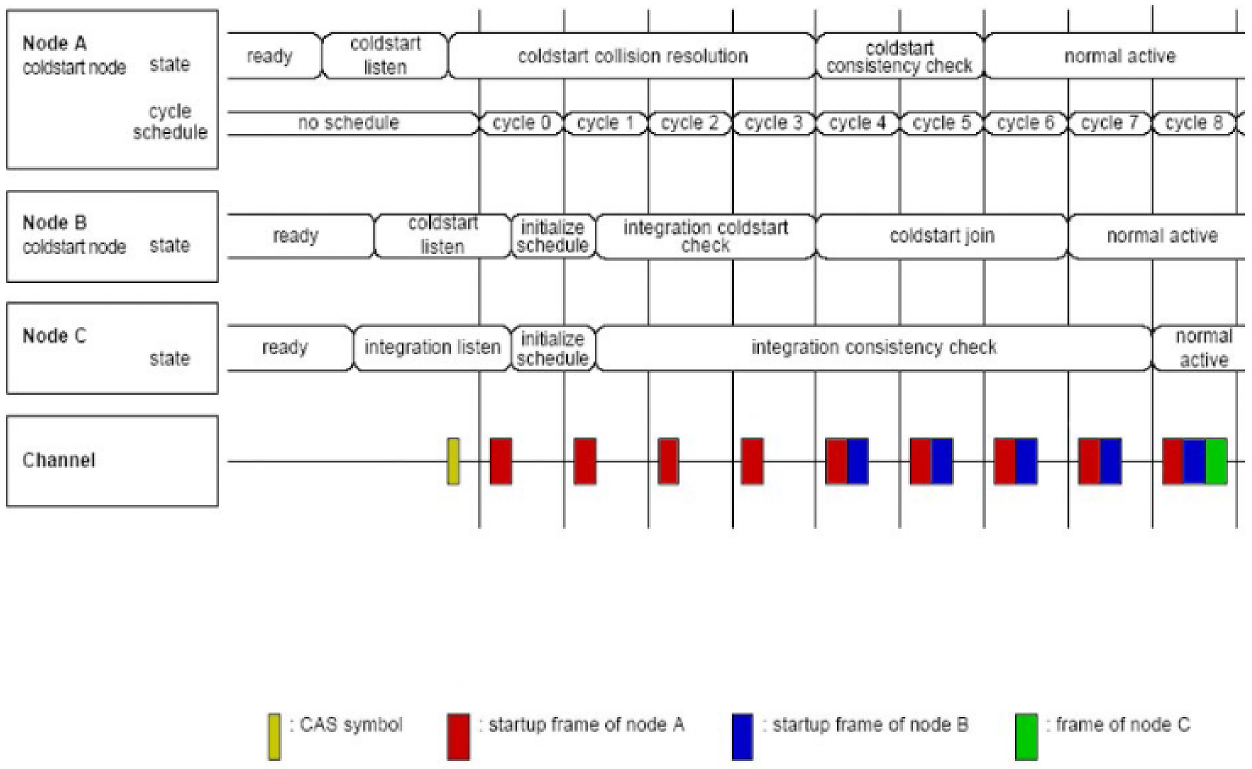
The startup is performed on all channels synchronously. The action of initiating a startup process is called a coldstart. Only a limited number of nodes may initiate a startup, theyare called the coldstart nodes. A coldstart attempt begins with the transmission of acollision avoidance symbol (CAS). Only the coldstart node that has transmitted the CAS can transmit startup frames in the first four cycles after the CAS. It is then joined firstly by the other coldstart nodes and afterwards by all other nodes.

Each startup frame shall also be a sync frame; therefore, each coldstart node will also be a sync node. At least two fault-free coldstart nodes are necessary for the cluster to startup because non-coldstart nodes require at least two startup frames from distinct nodes for integration. For systems with 3 or more nodes, ideal number is 3 coldstart nodes, since a bigger number can lead to problems with forming a single group or cluster (the communication workflow might end up being done only between the nodes that always react faster).

A coldstart node that actively starts the cluster is also called a leading coldstart node. A coldstart node that integrates upon another coldstart node is also called a following coldstart node. During startup, a node may only transmit startup frames. Any coldstart node shall wake up the cluster or determine that it is already awake before entering.

The system startup consists of two logical steps:

1. In the first step dedicated coldstartnodes start up.
2. In the second step, the other nodes join the coldstart nodes.



Once the cluster is in normal active state, it can start sending data frames, including NM messages.

Sleep

The network management service defined in AUTOSAR supports an application-level functional network management strategy, where functions may sign on or off to other functions within the system without knowledge of their physical location, i.e. without knowing which node actually performs the function.

Network Management is a big topic, but the basics can be summarized in the following points:

- One of the nodes in the network acts as NM master. It controls when the bus goes to sleep and wakes up from a logical point of view.
- The NM master sends NM status messages as long as there is information to exchange on the bus by one or more nodes, keeping this way the bus awake.
- Each node sends NM request messages cyclically, informing this way the NM master what they needs in terms of communication are.
- Once a node has no need anymore to send data on the bus, it stops sending NM messages.
- Based on the NM information received from the different nodes, the NM master decides whether to keep the bus awake or not
- If the NM master stops sending NM status messages, communication on the bus must stop within a predefined, configurable time
    - First the applicative messages (i.e. messages with application data)

The channel(s) enter(s) then in idle condition, where no node is transmitting, as perceived by each individual node in the network. Note that detection of channel idle occurs some time after all nodes have actually stopped transmitting (due to idle detection times, channel effects, ringing, etc.).

Best practices

Collect here all the points you consider important to keep in mind with regards to the functionality or technology explained in order to use it correctly.

TBD

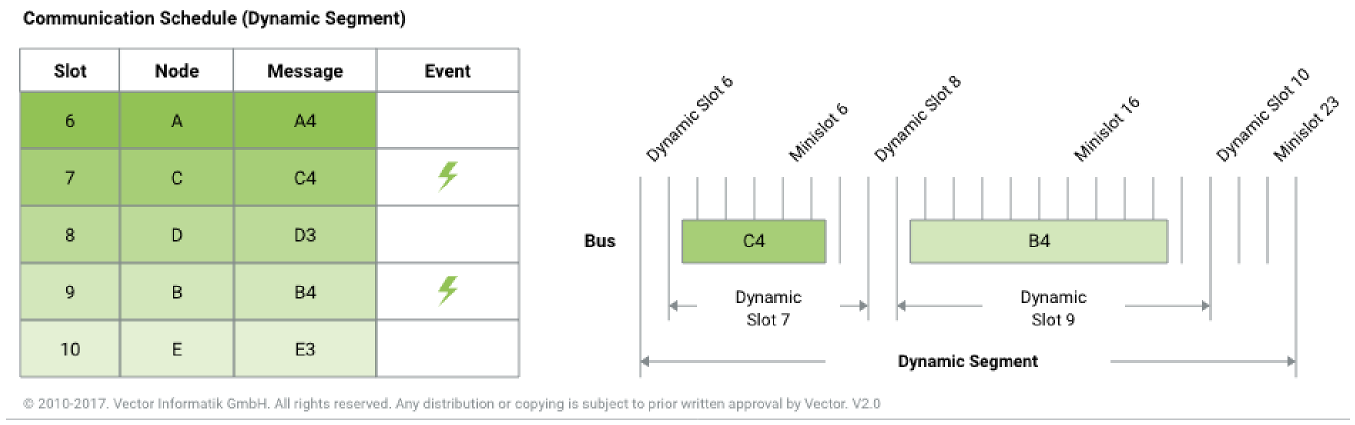
Additional resources

- FlexRay overview:

- https://en.wikipedia.org/wiki/FlexRay
- https://elearning.vector.com/mod/page/view.php?id=379
- https://www.ti.com/lit/ml/sprt718/sprt718.pdf?ts=1597392727088&ref_url=https%253A%252F%252Fwww.google.com%252F

File	Modified
PDF File FlexRay.pdf	29 Nov 2019 by Arantxa Fernandez
PNG File FR TDMA principle.png	29 Nov 2019 by Arantxa Fernandez
PDF File FlexRayISOTransportLayer_Specification.pdf	29 Nov 2019 by Arantxa Fernandez
PDF File FlexRay™ Electrical Physical Layer Conformance Test Specification V2.1 Rev. A.pdf	29 Nov 2019 by Arantxa Fernandez
PDF File FlexRay™ Protocol Conformance Test Specification V2.1.1.pdf	29 Nov 2019 by Arantxa Fernandez
PDF File FlexRay™ Protocol Conformance Test Specification Version 3.0.1.pdf	29 Nov 2019 by Arantxa Fernandez
PDF File FlexRay™ Protocol Specification V2.1 Rev. A Errata V1.pdf	29 Nov 2019 by Arantxa Fernandez
PDF File FlexRay™ Protocol Specification V2.1 Rev.A.pdf	29 Nov 2019 by Arantxa Fernandez
PDF File FlexRay™ Protocol Specification Version 3.0.1.pdf	29 Nov 2019 by Arantxa Fernandez
XML File FBX412_SP2021_18KW43_V16_A_FlexRay_V118_lokal.xml	29 Nov 2019 by Arantxa Fernandez
HTML File NK_SP2021_18KW43_V16_A_FlexRay_V118.html	29 Nov 2019 by Arantxa Fernandez
PNG File FR topologies.png	12 Aug 2020 by Arantxa Fernandez
PNG File FR bus.png	12 Aug 2020 by Arantxa Fernandez
PNG File FR bus level.png	12 Aug 2020 by Arantxa Fernandez
PNG File FR communication principle.png	12 Aug 2020 by Arantxa Fernandez
PNG File FR node.png	12 Aug 2020 by Arantxa Fernandez
PNG File FR synchronization.png	12 Aug 2020 by Arantxa Fernandez
PNG File FR static segment.png	12 Aug 2020 by Arantxa Fernandez
PNG File FR dynamic segment.png	12 Aug 2020 by Arantxa Fernandez
PNG File FR cycle variants.png	12 Aug 2020 by Arantxa Fernandez
GIF File FR cluster.gif	12 Aug 2020 by Arantxa Fernandez
PNG File FR_Frame.png	14 Aug 2020 by Arantxa Fernandez
PNG File FR_ProtocolOperationControl.png	14 Aug 2020 by Arantxa Fernandez
PNG File FR_Startup_StateTransitions.png	14 Aug 2020 by Arantxa Fernandez
PNG File FR_NM_StateDiagram.png	14 Aug 2020 by Arantxa Fernandez
PNG File FR_Lifecycle.png	01 Jun 2021 by Arantxa Fernandez

Drag and drop to upload or browse for files
Download All



Communication Schedule (Dynamic Segment)

Slot	Node	Message	Event
6	A	A4	
7	C	C4	⚡
8	D	D3	
9	B	B4	⚡
10	E	E3	