FlexRay International Workshop

16th and 17th April, 2002
Munich

Protocol Overview

Mr. Florian Bogenberger - Motorola
Dr. Bernd Müller - Bosch
Mr. Thomas Führer - Bosch

Protocol Overview

- **Basic Concepts & Motivations**
  - **Static and Dynamic Message Segments**
  - **Cluster & Node Configuration**
  - **Topologies**
  - **Node Architecture - The ‘Checks and Balances’ Trio**
  - **Never-Give-Up Strategy**
  - **Distributed Clock Synchronisation**
- Protocol Mechanisms

Static and Dynamic Segments

- Periodic statically scheduled message transfer is a benefit for automotive applications, especially distributed control loops with replication
- Static message segment
  - deterministic communication behavior
  - state message semantic
  - required support for distributed control and closed-loop control functions
  - benefits for design and simulation of distributed functions
- Spontaneous message transfer in a dynamic segment to allow
  - burst transmissions
  - diagnosis information
  - ad hoc messages in general

Static and Dynamic Segments

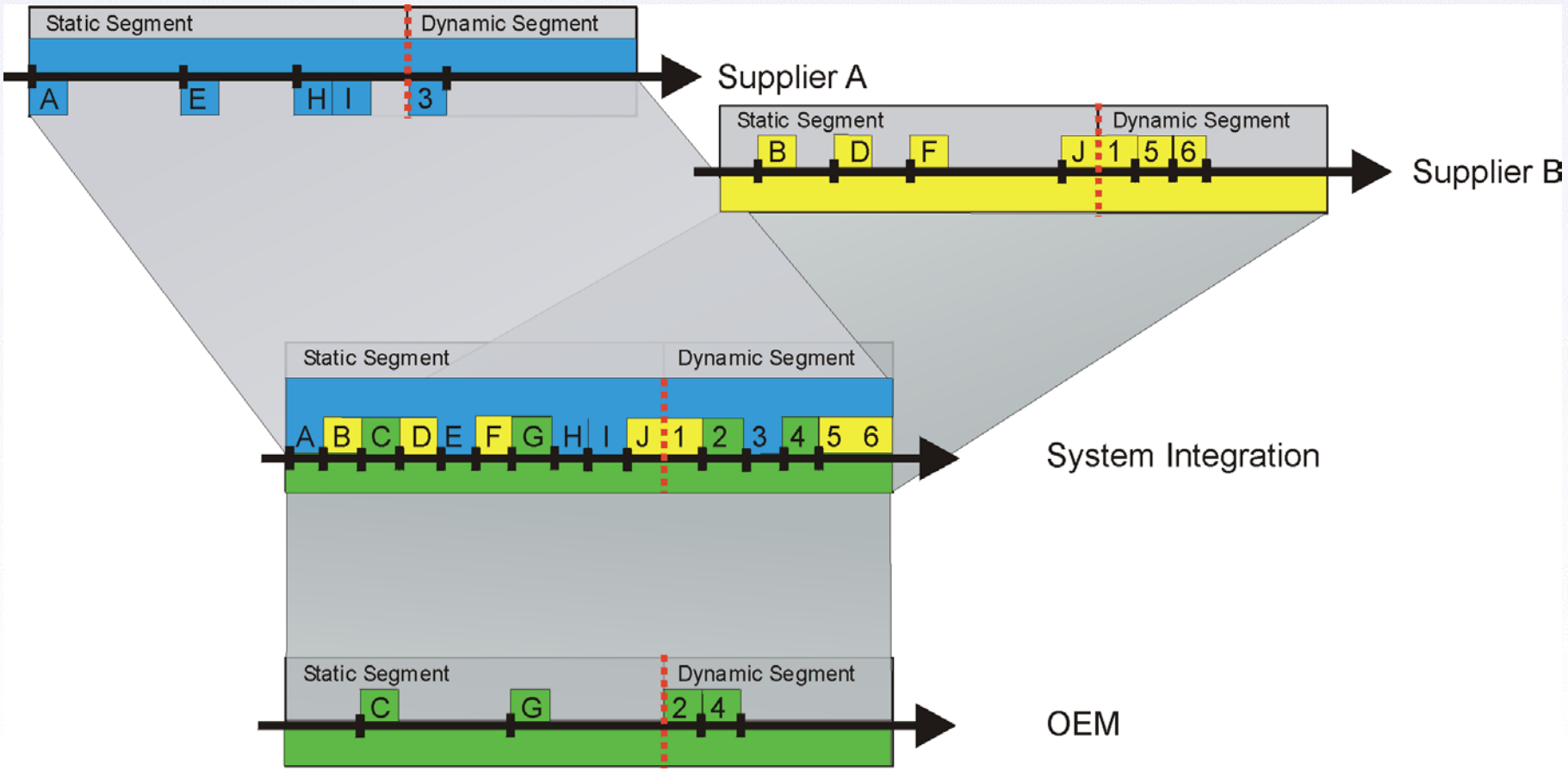
- The dynamic segment
  - is handled as one big static segment
  - allows network wide arbitration and
  - is limited in time and bandwidth consumption

FlexRay supports offline-defined segment
for time triggered messages and
for a scaleable number of spontaneous messages

Cluster and Node Configuration

- Functions are distributed in a system, however not all nodes need data signals from all messages
  - ⇒ no requirement to know about all messages in the cluster
  - ⇒ minimize memory needed for configuration
- Over lifetime and over different platforms functions are modified
  - by merging from a set of several ECUs to a subset of ECUs
  - by moving between ECUs
  - ⇒ mapping between ECUs and messages changes
  - ⇒ avoid reprogramming of the whole network by avoiding unnecessary coupling between messages and ECUs

FlexRay nodes are configured on a "need-to-know base"



FlexRay supports automotive needs for system integration

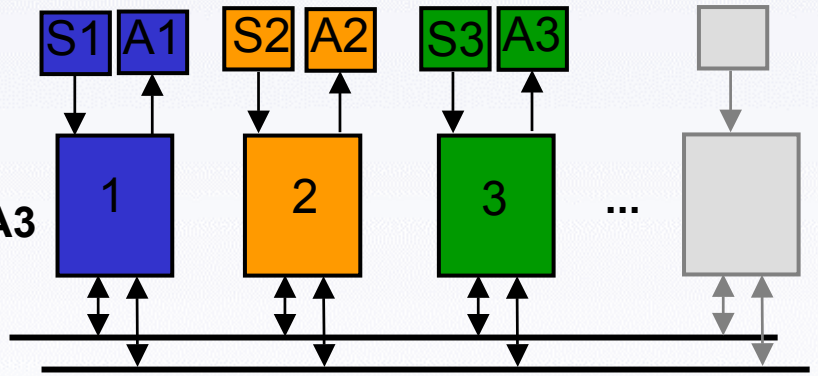
Cluster and Node Configuration

- Functions need different communication bandwidth
  - Several sending possibilities within one cycle.
  - Distributed control loops shall be closed within one cycle.
- Application agreement protocols shall be supported by the communication system
  - Data consistency within the application achieved by an end-to-end application agreement.
  - Multiple sending slots for one node allow to finalize agreement within one cycle.

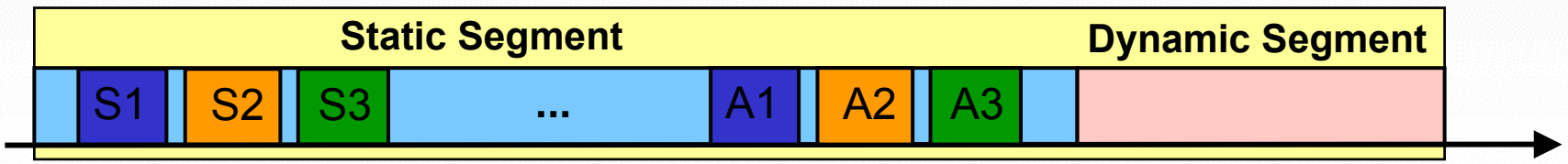
FlexRay nodes are scheduled on a “need-to-send base”

Application Agreement Protocol Example: 2 of 3 Voting

1. Step: Nodes 1, 2, 3 send signals S1, S2, S3
2. Step: 2 of 3 voting of input signals
3. Step: Application calculation
4. Step: Sending of application results A1, A2, A3
5. Step: 2 of 3 voting of results



FlexRay Com. Schedule



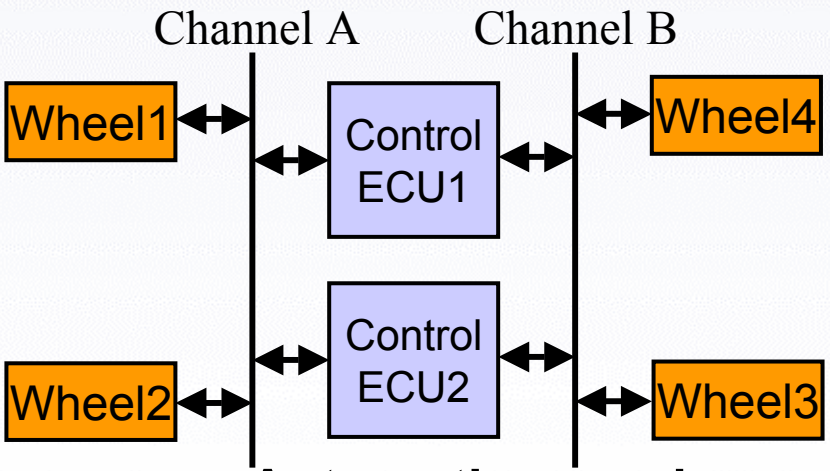
➡ **Finalize agreement within one communication cycle.**

Topologies - Single & Dual Channel

- Dual channel
  - supports concurrent transmission of redundant data
  - use for non-redundant parallel transmission for higher data rates
  - tolerates one faulty channel
- Single channel
  - reduces wire-harness
  - avoids spatial proximity of remaining second channel
  - most automotive experience
  - automotive costs

Topologies - Mixed Connectivity

- Mixed connectivity in dual channel systems
  - allows nodes to be either on channel A or B
  - use ‘redundancy by system topology’
  - allows cost efficient solutions



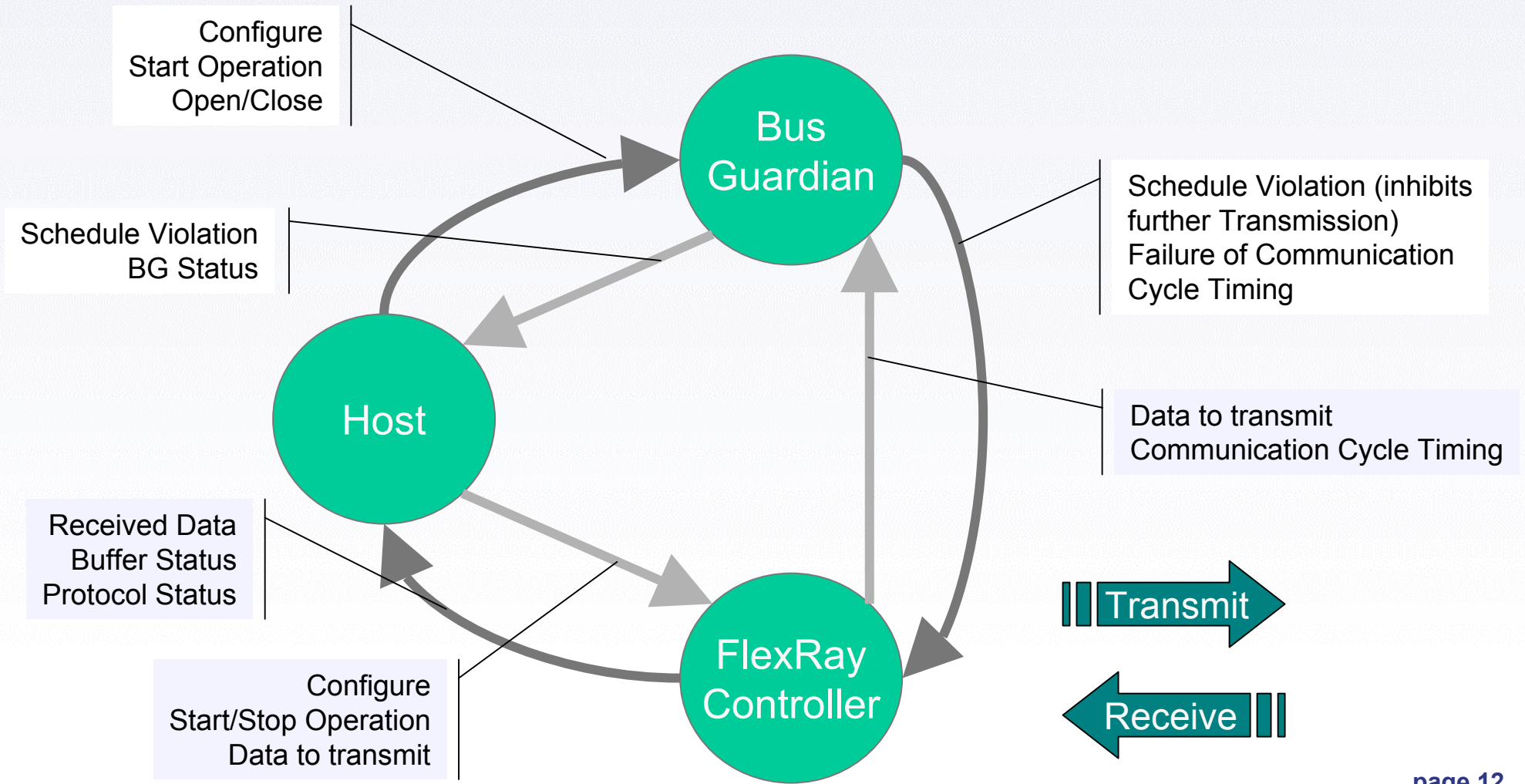
Automotive requirements for future system architectures are efficiently supported

Topologies - Bus & Star

- Star
  - best suited technology for high speed networks
  - different degrees of intelligence possible
    - with/without protocol knowledge
    - can protect against concurrent media access
    - limits the error domain of not correctly working subnetworks
- Bus
  - passive medium
  - no active components within the channel
  - most automotive experience
  - automotive costs

Node Architecture

The 'Checks and Balance' Trio



‘Never-give-up’ - Strategy (1)

- Background
  - An unavailable communication system inhibits distributed rescue mechanisms
  - Restarting a node in a running system implies more than just restarting its communication
- Basic assumptions
  - The communication system is only one part of a larger system serving specific applications
  - Errors can occur in all parts of the complete system
  - Some errors can be recognized, diagnosed and handled only on the application layer
  - End-to-end agreement and interactive consistency protocols detect communication errors ‘en-passant’

‘Never-give-up’ - Strategy (2)

- Approach
  - maintain data transmission as long as communication among other nodes is not jeopardized, i.e.
    - as long as fault tolerant clock synchronization works
    - as long as key operation and transmission related sanity checks are passed successfully
  - maintain data reception as long as possible, i.e.
    - as long as fault tolerant clock synchronization works
    - as long as key operation related sanity checks are passed successfully

Stopping communication is a critical decision which must be made by the application whenever possible

Distributed Clock Synchronization

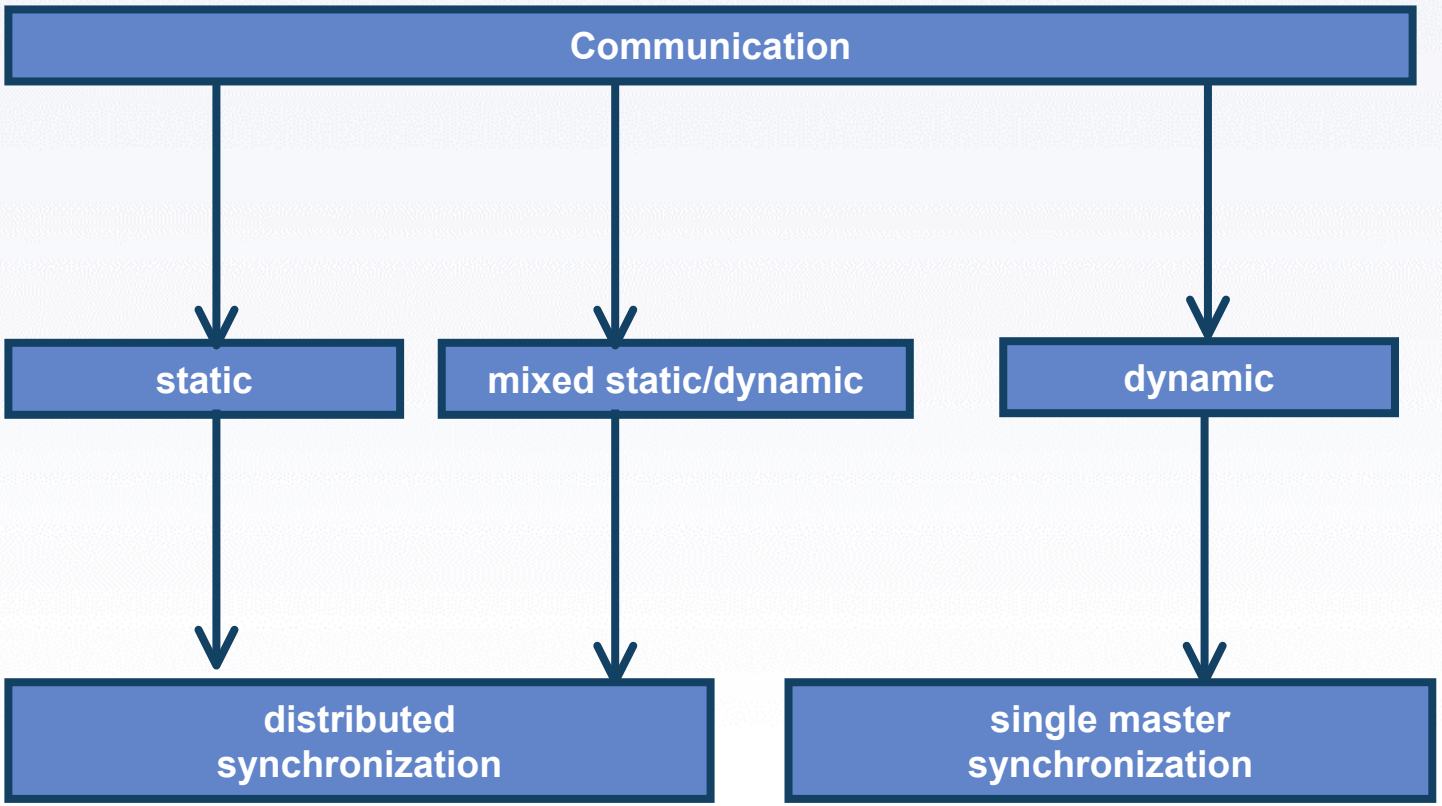
- Future systems running distributed control algorithms require a highly synchronous operation of network nodes
- Clock synchronization is thus a pre-requisite for these systems
- Automotive environment imposes very stringent hard real-time requirements on the clock synchronization algorithm
- Automotive target costs for oscillators have to be met

FlexRay provides a distributed, fault tolerant and precise clock synchronization service

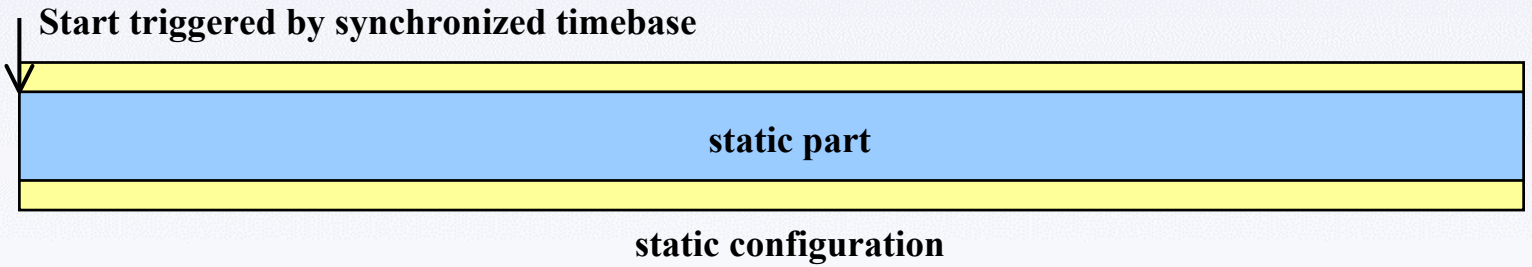
Protocol Mechanisms

- **Media Access**
  - **FlexRay Modes**
  - **Configurations**
  - **Communication Cycle**
  - **Static Part**
  - **Dynamic Part**
- Frame Format & Coding
- Clock Synchronisation
- Startup
- Error Management

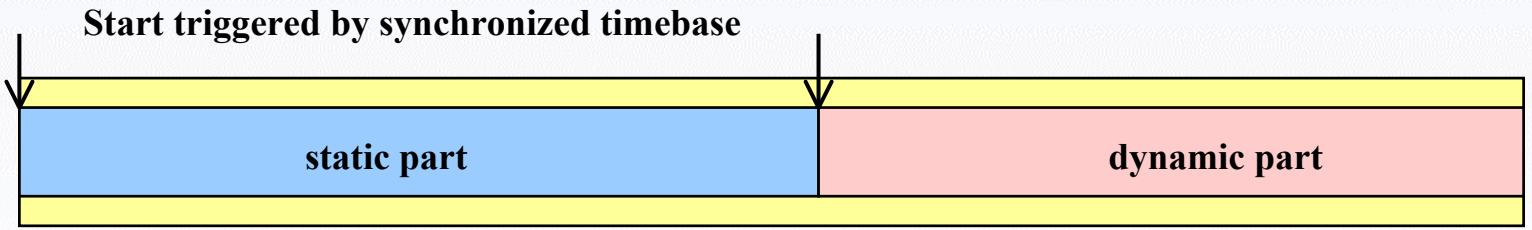
FlexRay Modes



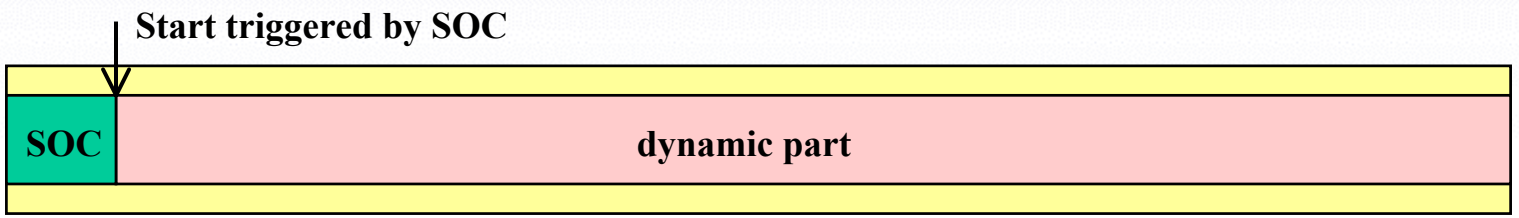
Configurations



static configuration

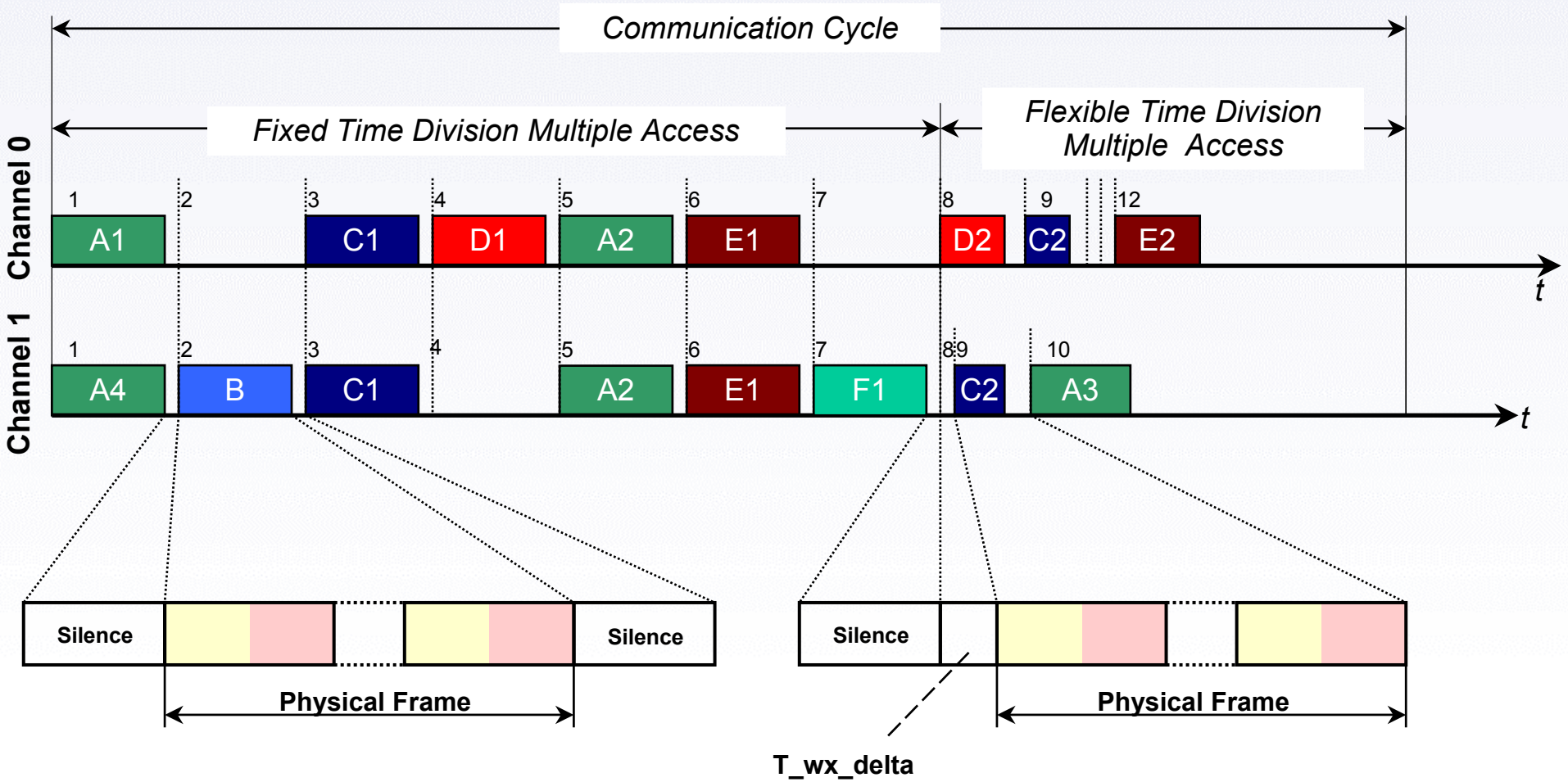


mixed configuration



dynamic configuration

Communication Cycle



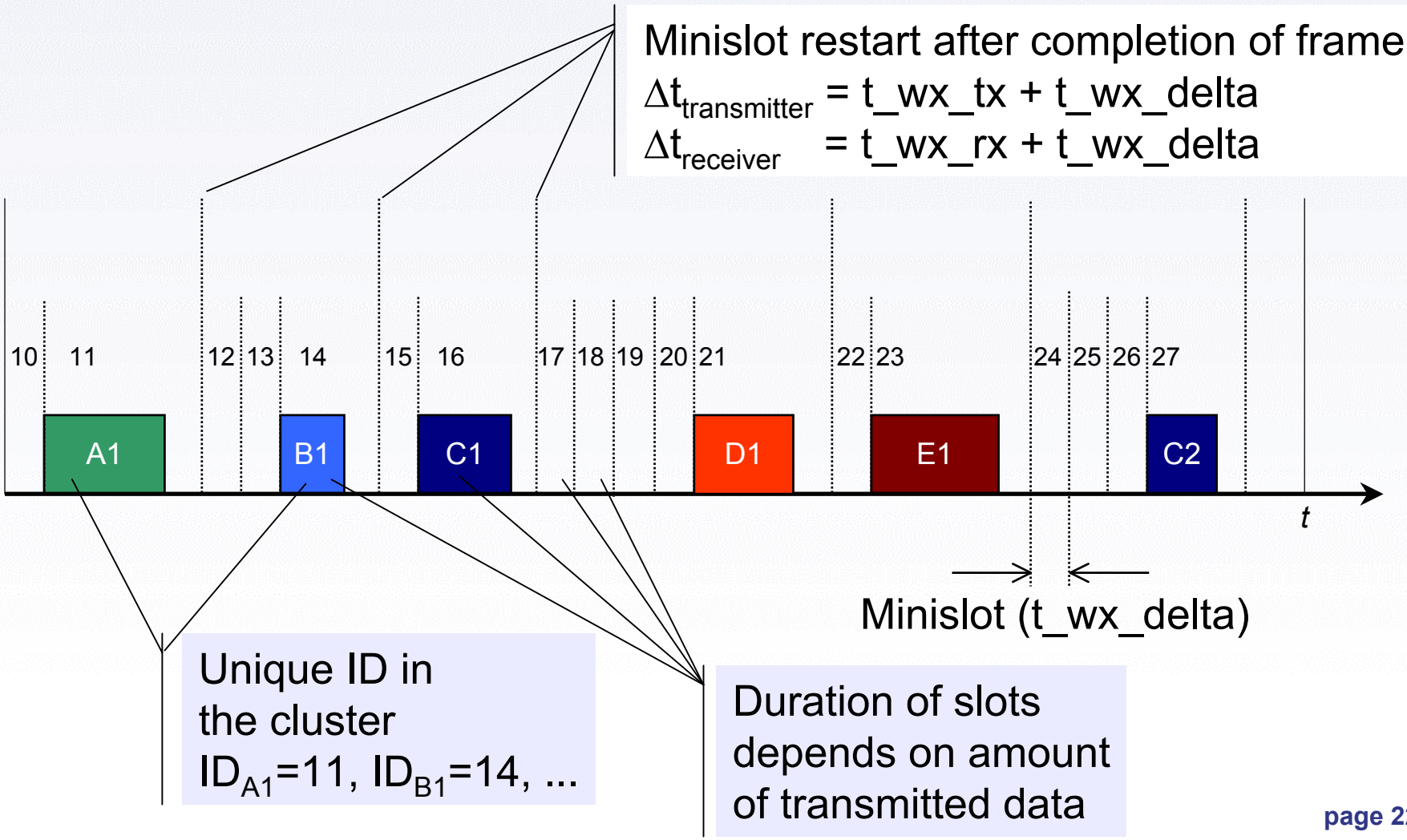
Static Part - Characteristics (1)

- Structure
  - static part divided into static slots of equal duration
  - slot duration defined on a per cluster basis
  - Slot timing identical on both channels
  - each slot identified by an unique slot ID
  - slot start determined by global time
- Configuration
  - static bandwidth allocation
    - slot assigned statically on a per channel basis to a node for transmission
    - specific (time) slots reserved for each node
    - static configuration eliminates run-time contention
  - up to 16 slots assignable per node

Static Part - Characteristics (2)

- Transmission
  - frames transmitted in slots where frame ID matches slot ID
  - all frames have equal length
  - frame content may match or differ within one slot on different channels
  - “Data Update” Bit is set if data was updated since previous cycle
  - slot remains empty if no frame is configured for it
- Bus Guardian interaction
  - access enabled for transmission during assigned slots
  - access disabled for transmission during unassigned slots

Dynamic Part - Minislotting



Dynamic Part - Characteristics (1)

- Structure
  - dynamic bandwidth allocation for each node
    FTDMA - Flexible Time Division Media Access
  - unique IDs matching the (mini-)slot number
    ⇒ no collision, no concurrent transmission
  - if available bandwidth is smaller than sum of all frames to be transmitted those with the higher IDs wait for the next communication cycle
  - start of dynamic part
    - in mixed configuration: based on the global time
    - pure dynamic mode: triggered by SOC

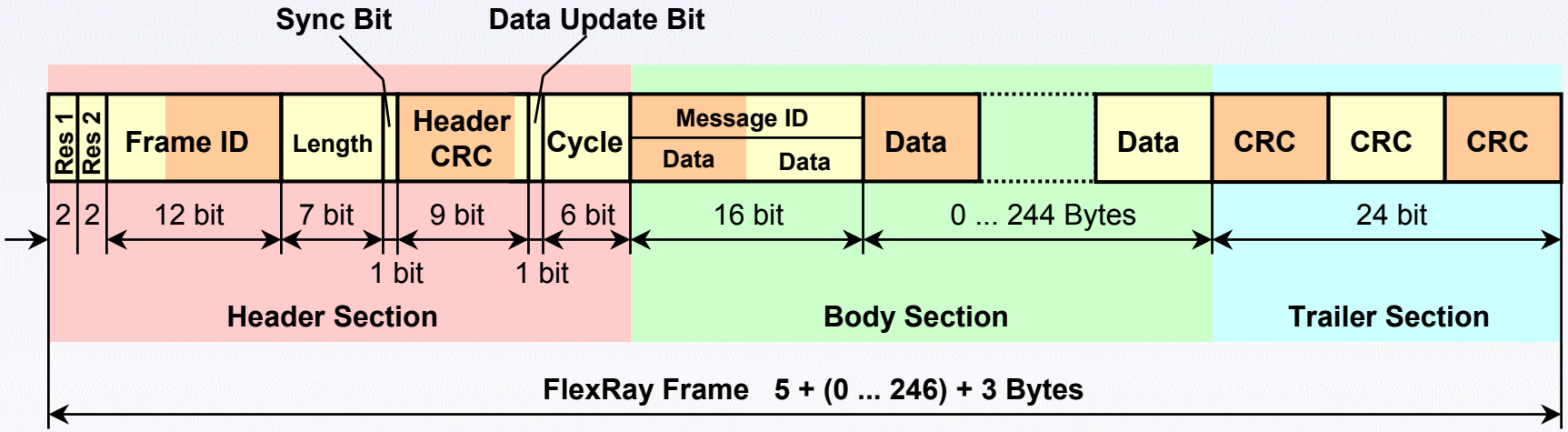
Dynamic Part - Characteristics (2)

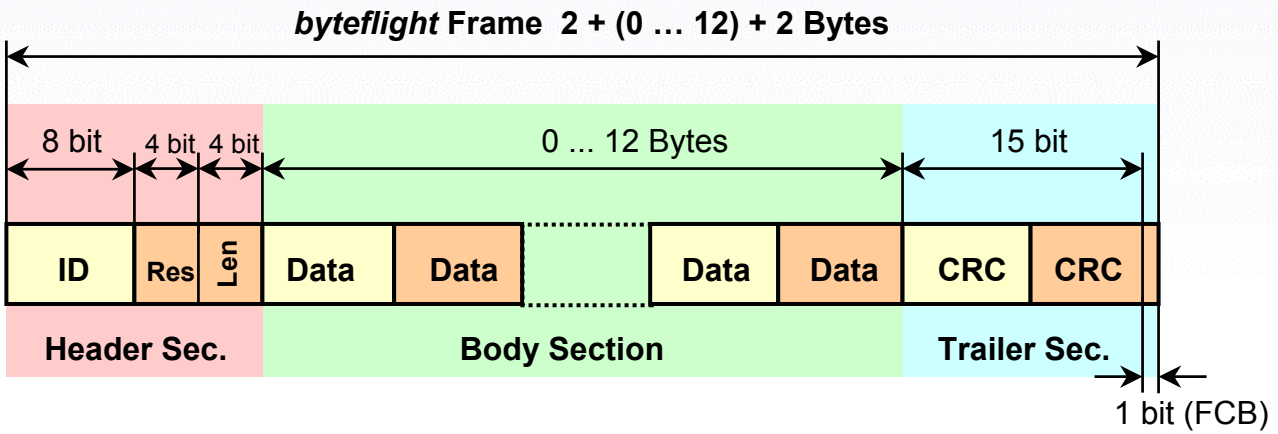
- Configuration
  - only ‘new’ data is transmitted
  - can be reconfigured during normal operation
  - channel sharing allowed for dual and single channel nodes
  - no sync frames allowed
  - to prevent interaction with static part frame IDs are larger than $ID_{max,static}$
- Transmission
  - can be different on both channels
  - determined by minislot counter derived from local clock
- Bus Guardian interaction
  - no protection within dynamic part
  - static part protected against dynamic part
    - BG treats dynamic part like one large static transmission slot

Protocol Mechanisms

- Media Access
- **Frame Format & Coding**
- Clock Synchronisation
- Startup
- Error Management

Frame Format





FlexRay Frame Format (1)

- Res1, Res2
  - reserved for future use
- Frame ID
  - determines transmission slot (matches slot ID)
  - legal values: 1 – 4095, 0 is illegal
- Length
  - number of words in payload section
  - legal values: 0 – 123, values > 123 are treated as error
- Sync Bit
  - “0” for normal frames
  - “1” indicates sync frame to be used for clock synchronisation
  - must be set on both channels, only dual channel nodes can set this bit
  - at most one sync frame per node

FlexRay Frame Format (2)

- Header CRC
  - protects Length and Sync Bit with hamming distance of 6
  - configured by host of transmitting communication controller
  - checked by receiving communication controller
- Data Update Bit
  - “1” indicates that data was updated by host
  - “0” indicates that data is identical to previous communication cycle
  - automatically set by controller when host releases the frame for transmission
  - automatically reset by controller after transmission

FlexRay Frame Format (3)

- Cycle (Counter)
  - denotes current communication cycle number
  - automatically incremented by controller
  - wraps around at global, pre-configured maximum
  - must be identical for all frames transmitted in one communication cycle
  - supports synchronization to periods which are a multiple of the communication cycle
- Message ID
  - optional - 0 or 16 bit
  - receiver filterable data (no impact on transmission)
  - frame is stored in message buffer only if message ID matches filter
- Data
  - padding supported for small message buffers
- CRC
  - protects the whole frame with hamming distance >= 6

byteflight Frame Format

Mutually exclusive to FlexRay Frame Format

Frame ID

- legal values: 1-255, 0 is an error

Reserved

- Length
  - legal values: 0 - 12, 13-15 is an error

Data

- CRC
  - protects the whole *byteflight* frame with hamming distance >= 6
- FCB
  - Frame Completion Bit - fills frame to byte boundary

Coding

- Currently supporting NRZ 8N1 (one start and stop bit for each byte)
- Frame Start Sequence (FSS)
  - each frame starts with a FSS of 8 bit of value “0”
  - FSS has no start/stop bits
  - FSS may be shortened by star coupler and/or receiver

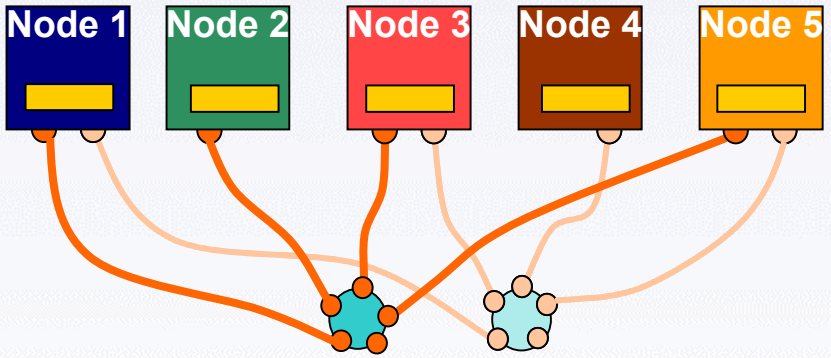
Protocol Mechanisms

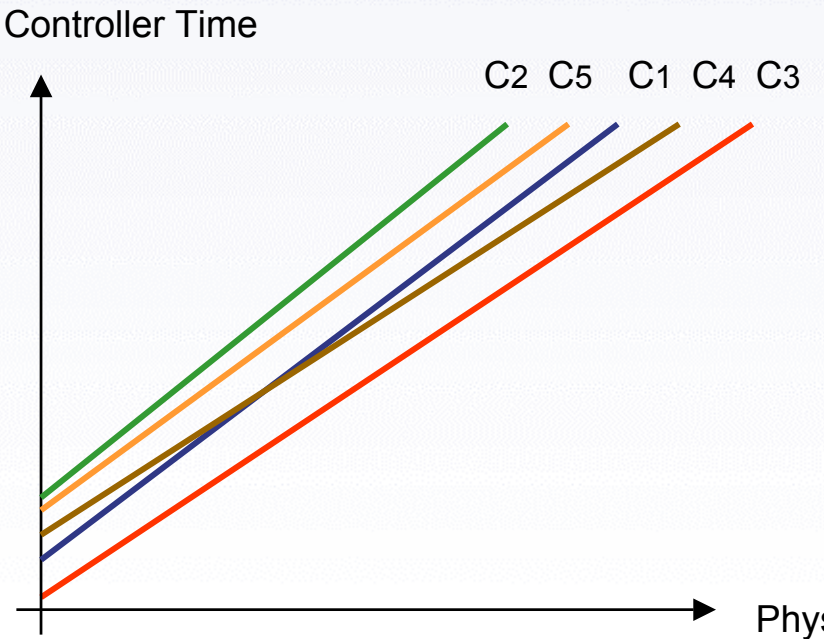
- Media Access
- Frame Format & Coding
- **Clock Synchronisation**
  - **Requirements for the FlexRay Clock Synchronisation**
  - **Time representation in FlexRay**
  - **Introduction of the Offset Correction mechanism**
  - **Introduction of the Rate Correction mechanism**
  - **Conclusion**
- Startup
- Error Management

Requirements

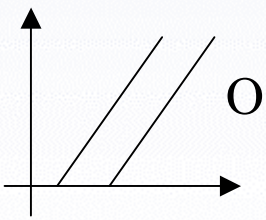
- Scaleable number of controllers.
- High precision clock synchronisation.
- Goal is 1 µs in reasonable scenarios.
- Efficient use of bandwidth.
- Fault-tolerance for up to two asymmetric faults.
- High robustness to survive several cycles without synchronisation.
- Tolerance of crystal drifts in automotive environments.

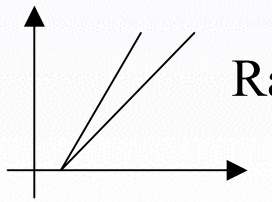
Clocks in a distributed system





Clocks have differences in:





Time representation

- For each controller there are four potentially different time units:
  - Bit time
  - Clock tick
  - Microtick
  - Macrotick
- Bit time is only important for bit representation.
- Clock tick is the physical basis for other units but of no relevance for the logical time representation.
- The microtick is a fixed multiple of the clock tick. It is the smallest unit for time (difference) measurements.
- The microtick is a completely controller-local time unit. Microtick length is influenced by drift of oscillators.

Macrotick and Global Time

- Intention is that the macrotick is a network constant, the relevant network time unit.
- Network wide global time is measured in macroticks.
- Each cycle has the same integer number of macroticks.
- The cycles are numbered.
- Global time is a pair (Cycle_Count, Macrotick_Count).

Macrotick and Global Time

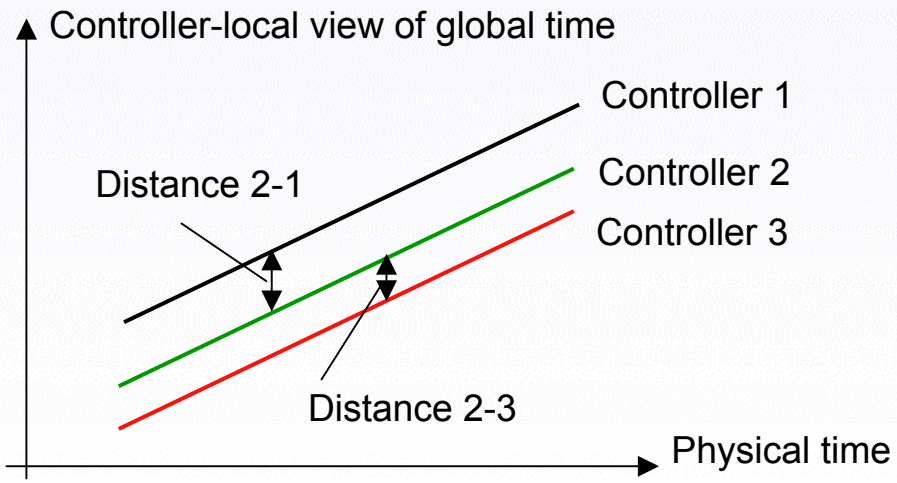
- The ratio macrotick/microtick is a controller-local value.
- This value is influenced by clock synchronisation.
- It is not usually an integer (!).
- It may be advantageous to distinguish between:
  - the abstract macrotick as the network time unit (e.g. 1 µs)
  - the local view of the macrotick length during one cycle
  - the local macrotick length of one given macrotick

Macrotick

- Within a controller there are three relevant values
  - Number of macroticks per cycle (network constant): ***k***
  - Default number of microticks per macrotick (controller local configuration parameter): ***n***
  - Number of additional microticks per cycle: ***d***
- So during one cycle there are ***n * k + d*** microticks and ***k*** macroticks.
- The average macrotick length is ***n + d / k*** microticks.
- The hardware has to distribute the additional ***d*** microticks uniformly over the ***k*** macroticks.
- By manipulation of ***d*** it is possible to increase or decrease the rate of the global time.
- Value ***d*** is controller-local and is influenced by clock synchronisation.

Clock Synchronisation: Offset Correction

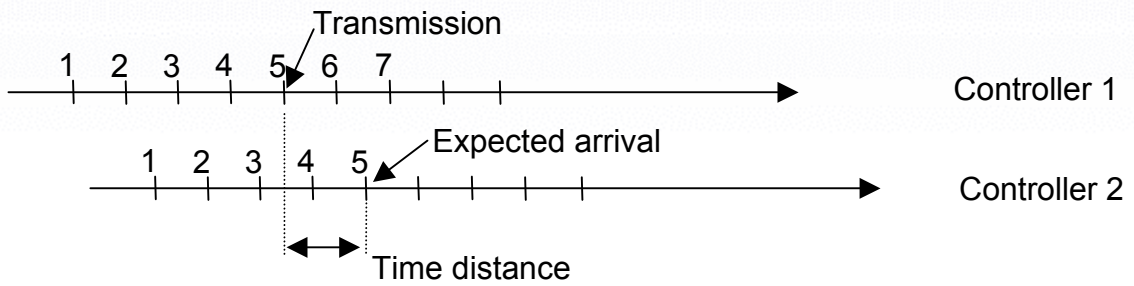
- Assume that all clocks in all nodes run with the same rate.
- Ignoring granularity we than get a set of parallel lines in the following diagram.



- Offset correction tries to make all these distances small.

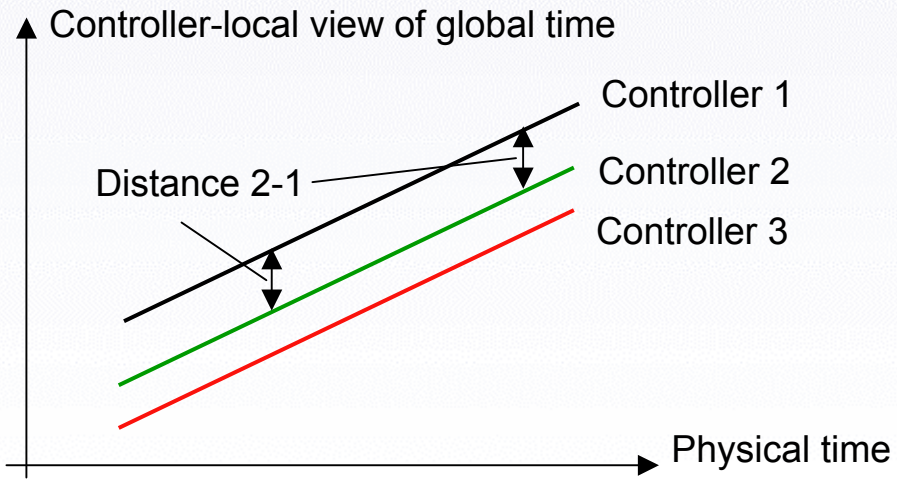
Clock Synchronisation: Offset Correction

- A node starts the transmission of its messages when its local view of global time reaches predefined values.
- This value can be calculated from the identifier of the message.
- A second node “expects” the transmission of this message when his local view of global time reaches the same value.
- By measuring the difference between the actual arrival time and the expected arrival time the node can measure the “global time distance” to the other controller.



Clock Synchronisation: Offset Correction

- If all clocks run with the ***same*** rate a controller is able to collect all distance information from observing the set of relevant messages.
- Only predefined messages are used - so called Sync Frames.

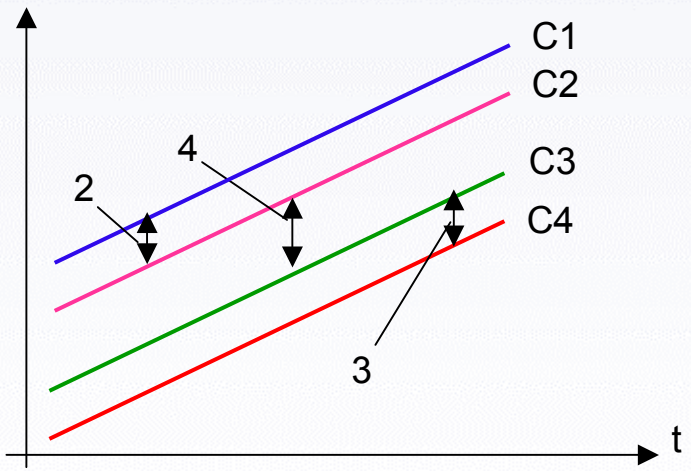


- The measurement is done with microtick resolution.

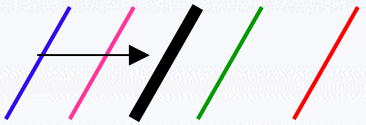
Clock Synchronisation: Offset Correction

- After collecting all relative distances the controller can calculate the resulting distance to an average line.
- For fault-tolerance and hardware reasons not the average but a fault tolerant midpoint (FTM) is used.
- FTM algorithm of order ***j***:
  - Sort the time distances (signed numbers of microticks).
  - Remove ***j*** largest values and ***j*** smallest values.
  - Out of remaining values choose largest and smallest and build the average over those two values.

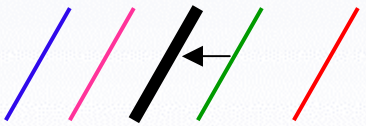
Offset Correction, an example



Controller1: Ø, -2, -6, -Ø → -4



Controller3: Ø, 4, 0, -Ø → 2



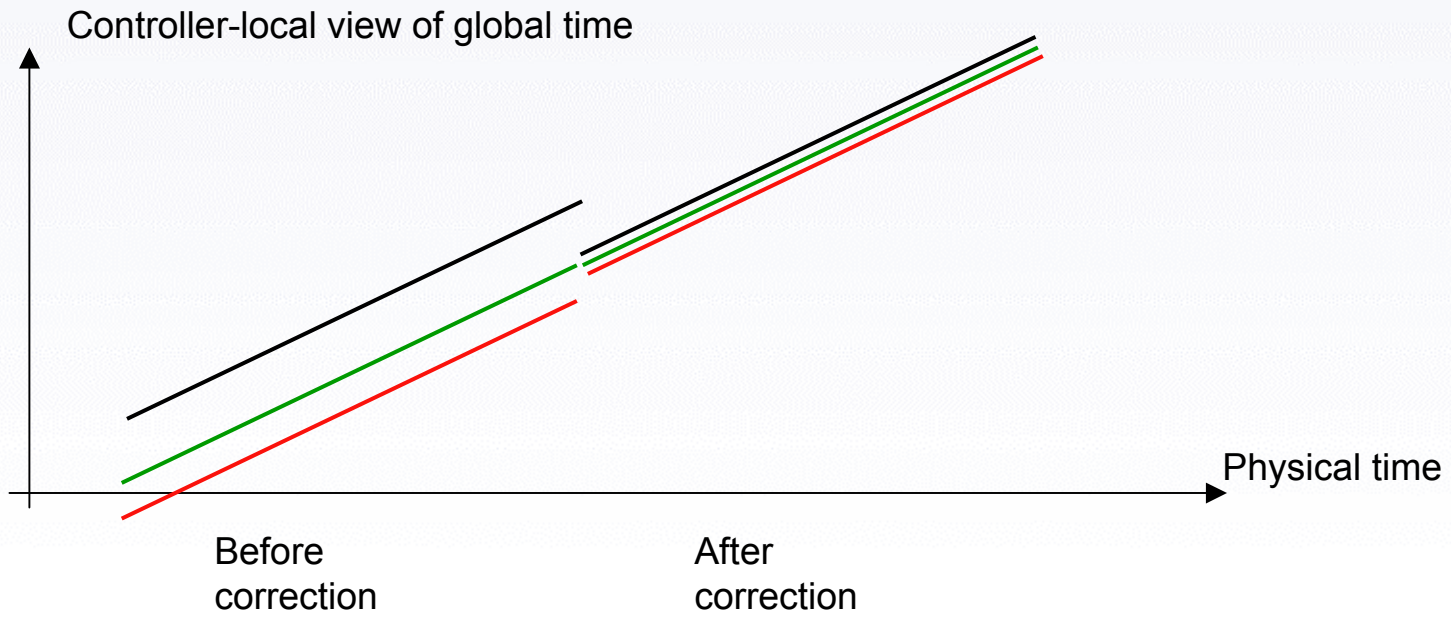
Clock Synchronisation: Offset Correction

- Using a FTM algorithm, the maximum effect of a fault is strongly limited.

- Systematic errors:
  - Run time.
  - Resolution, granularity.
  - Rate differences.
  - Higher order phenomena (non-linearities in the oscillator).

Clock Synchronisation: Offset Correction

- At the beginning of (every second) cycle a node adjusts its own view of the global time according to the calculated offset correction term.



Clock Synchronisation: Rate Correction

- Offset correction requires (sufficiently) equal rates.
- In automotive context over an ageing period of 10 years rate deviations of +/- 250 ppm have been observed.
- Including a safety factor a few thousand ppm have to be tolerated.
- 2000 ppm over a communication cycle of, e.g., 20 ms results in deviations of about 40 µs besides the fact that the offset correction then is not working properly.
- 1 µs is synchronisation target.
- Physical observation shows that rate correction is sufficient
  - Higher order phenomena are either extremely small or have very large time constants (can be modelled by change of rate).

Conclusion: Rate correction shall be implemented.

Clock Synchronisation: Rate Correction

- Physical length of one controller-local cycle is directly connected with rate.
- Measurement of the cycle length of controller 1 in units of controller 2 equals a measurement of the relative rates.
- Making all cycle lengths equal means synchronising all rates.

Idea:

- Measuring of cycle length is done by measuring the time difference between the reception of the same message in two successive cycles.

Clock Synchronisation: Rate Correction

- Run time error removed by design.
- The offset correction systematically influences the measurement.

Conclusion:

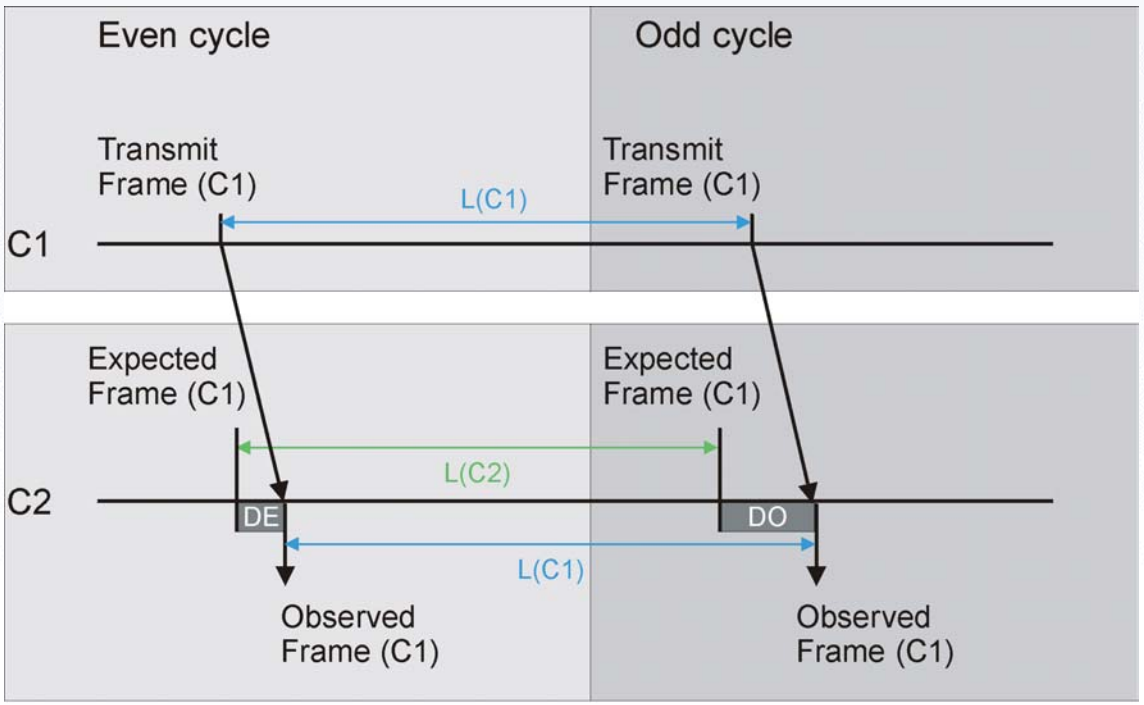
- Make offset correction only every second cycle.
- Distinguish between even and odd cycles (last bit of cycle count).

Clock Synchronisation: Rate Correction

- Rate correction is performed, using the measurements of an even/odd cycle, at the beginning of an even cycle.
- Measurement of cycle lengths is done over even/odd cycles.
- For a controller it is sufficient to measure the differences of other cycle lengths to its own cycle length.
- The differences are measured with microtick resolution.
- They are collected and by way of an FTM algorithm transformed into a resulting rate correction term.
- The rate correction term is an integer number of microticks that increases/decreases the own cycle length.
- It is added to the ***d***-value of the macrotick representation.
- After one rate correction all rates are synchronised.

Clock Synchronisation: Rate Correction

- Measure cycle length by measuring the time differences between the reception of the same message in two successive cycles.



$$L(C1) = L(C2) - DE + DO$$

Clock Synchronisation: Rate Correction

- Using a FTM algorithm, the maximum effect of a fault is strongly limited.

- Systematic errors:
  - Run time jitter.
  - Granularity.
  - Non-linearities in the oscillator and so microticks do not have equal length.

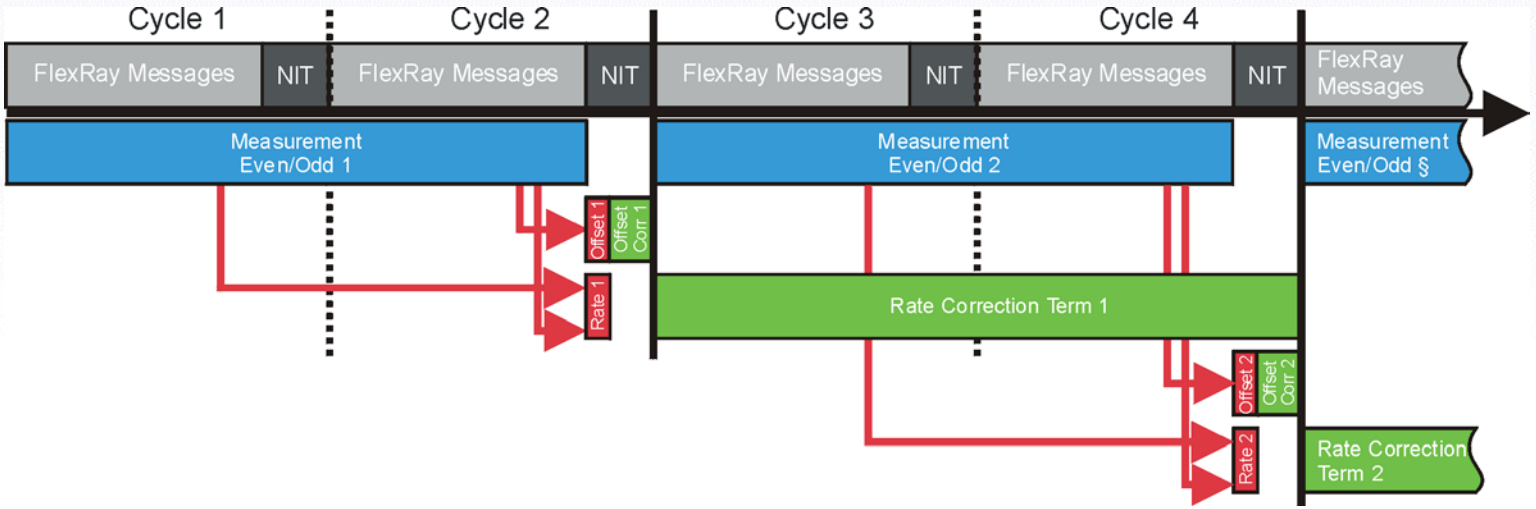
Clock Synchronisation: Rate Correction

- Advantages:
  - Fulfils automotive requirements.
  - Fault robustness.
    - Stable rate correction after one successfull measurement and calculation.
    - Robust over several cycles without new measurements/calculations.
  - Extremely strong fault detection capabilities.

- Disadvantages:
  - Implementation effort.
  - Longer intitialisation time.
  - Handling of even/odd cycles.

Principle of Clock Synchronisation

- Split the algorithm of clock synchronisation into four major parts:
  - Measurement phase
  - Calculation
  - Offset correction
  - Rate correction



Conclusion

- FlexRay Clock Synchronisation uses both rate and offset correction.
- The algorithm is clearly structured and easy to implement.
- Provides required automotive precision (1 µs target reached).
- Provides globally synchronised Global Time.
- Tolerates large quartz tolerances over lifetime.
- Allows to run over several cycles without clock correction.
- Allows to meet automotive target costs for oscillators.

Protocol Mechanisms

- Media Access
- Frame Format & Coding
- Clock Synchronisation
- **Start-up**
  - **Two start-up modes**
  - **Fault tolerant, distributed start-up**
  - **Master driven start-up**
- Error Management

Start-up - Two Modes

- Static or mixed configuration: fault tolerant, distributed Start-up
- Pure dynamic configuration: master driven Start-up

Distributed Start-up (1) – General

- Only nodes that are allowed to transmit sync frames can try to initiate the Startup (sync nodes)
- Every node listens before becoming active
  - at least for a given listen timeout L_1 without noise
  - L_1 is restarted whenever activity is detected on the media
  - at most for a given listen timeout L_2
  - L_2 is not restarted when activity is detected on the media
- During listen time the node tries to integrate, however …
- … when this fails it will try to initiate a Start-up (Coldstart)

Distributed Start-up (2) – Integration

- Every node that cannot initiate Start-up follows the following integration path:
  - receive a valid sync frame (assume ID=x) in even cycle
  - initialize local cycle counter with cycle counter in sync frame
  - measure time till reception of sync frame with ID=x in odd cycle
  - calculate initial rate correction
  - calculate start of next communication cycle based on ID and reception time of sync frame
  - check if majority of sync frames has matching cycle counter and perform clock synchronization
  - start communication in next communication cycle with even cycle count

Distributed Start-up (3) – Initiation

- Become active only after L_1 or L_2 has elapsed
- Transmit SOC and first own sync frame
- If meanwhile SOC or other sync frame is received, return to Listen; else transmit two further sync frames
  - unique distance SOC - sync frame resolves any possible collision
- Transmit further sync frames and receive also other sync frames
- check if majority of sync frames has matching cycle counter and perform clock synchronization
- start communication in next communication cycle with even cycle count

Start-up in pure dynamic mode

- only one node has authority to transmit SOC (Sync master)
- sync master transmits SOC one all channels
- slaves integrate based on SOC reception
- master continues transmitting SOC unless host stops this

Protocol Mechanisms

- Media Access
- Frame Format & Coding
- Clock Synchronisation
- Startup
- **Error Management**
  - **FlexRay dedicated degradation concept**
  - **Protocol (Error) States**
  - **FlexRay Status Vectors**

Error management

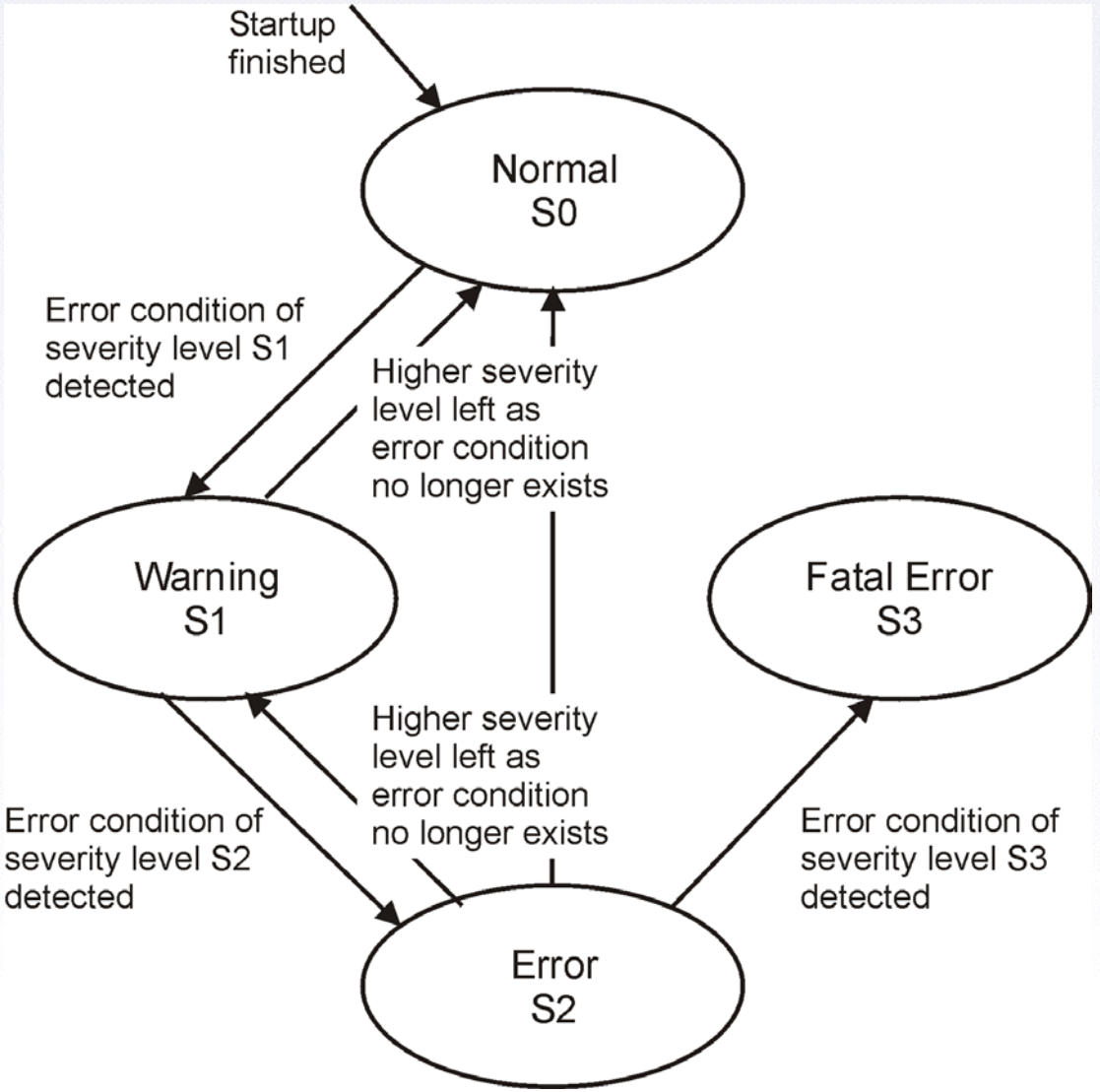
- The philosophy of FlexRay is the ‘never-give-up’ strategy and
- robustness against transient faults.
- So the FlexRay error handling model and its error detection mechanisms
  - attempt to avoid faulty behavior in the precence of errors.
  - shall support a dedicated degradation concept.

Dedicated degradation concept

- The degradation within FlexRay is closely related to the severity
  - an error can cause in the system or
  - a repetition of the same error within a period of time can cause.
- Four degradation levels based on the severity classification of a possible error are defined
  - Normal operation (S0)
  - Warning (S1)
  - Error (S2)
  - Fatal Error (S3)
- FlexRay specifies the handling and classification of an upcoming error as the error gets a severity level.

Dedicated degradation concept

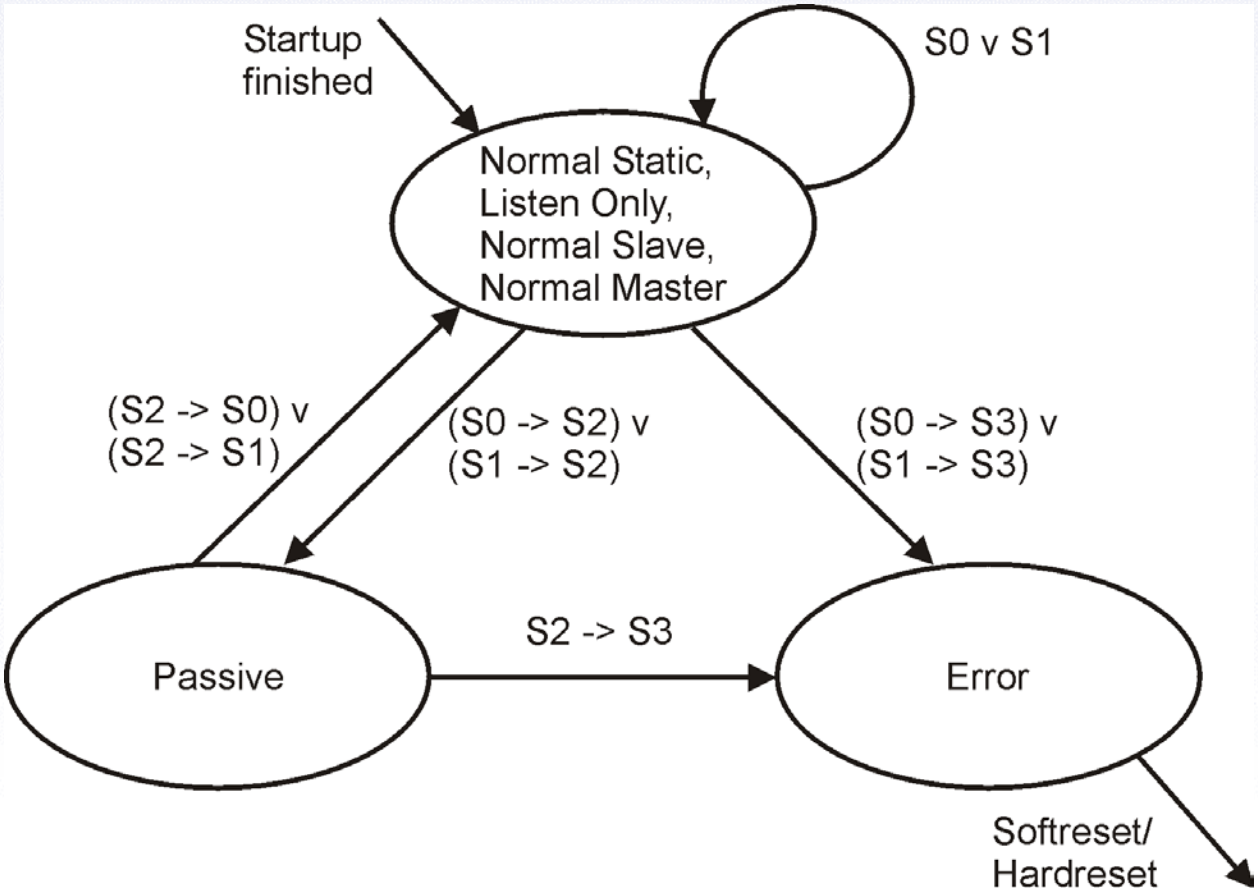
- Normal
  - sending and receiving, full operation within CC and BG.
- Warning
  - continue full operation within CC and BG, notify host.
- Error
  - stop transmission, keep CC and BG synchronized and notify host.
- Fatal Error
  - operation is stopped, all pins set into a safe state, notify host, BG closes media access.
- Error management shall allow to go into normal operation if the error condition (severity level) no longer exists.


Startup
finished
Normal
S0
Error condition of
severity level S1
detected
Higher severity
level left as
error condition
no longer exists
Warning
S1
Fatal Error
S3
Higher severity
level left as
error condition
no longer exists
Error condition of
severity level S2
detected
Error condition of
severity level S3
detected
Error
S2

Protocol States

- The error management philosophy and its the degradation model together with the severity levels influence the general protocol states.
- Degradation reflected
  - Normal
    - Normal Static (FlexRay)
    - Listen Only (FlexRay)
    - Normal Slave (Pure dynamic FlexRay, byteflight)
    - Normal Master (Pure dynamic FlexRay, byteflight)
  - Passive
  - Error

Protocol (error) states



Channels and Frames

- FlexRay defines dedicated error detection mechanisms for
  - channel specific errors and
  - frame specific errors.
- Channel specific error detection and host information allows overall on every traffic / disturbances on the media.
- Frame specific error detection and host information allows dedicated observation of frame status in which a node is interested
- Approach fulfills the ‘need-to-know’ philosophy.

Channel Status Error Vector (CSEV)

- Every event/error related to FlexRay frames on each communication channel is detected and reported to the host.
  - Bit Coding/CRC Error.
  - Slot mismatch.
  - Cycle counter mismatch.
  - Length mismatch.
- The CSEV
  - is mapped into the diagnosis interface of the FlexRay CC.
  - can be configured to be an interrupt source (bit-by-bit enable vector).
  - is reset by the host explicitly.

Frame Status Error Vector (FSEV)

- Every event/error related to explicitly received frames is detected and reported to the host.
  - Bit Coding/CRC Error.
  - Slot mismatch of any frame.
  - Cycle counter mismatch.
  - Missing frame.
  - Null frame.
  - Length mismatch.
- The FSEV
  - is updated after the end of the expected frame.
  - can be configured to be an interrupt source (bit-by-bit enable vector).
  - Interrupt source definition covers all frames in a CC.

FlexRay International Workshop

www.flexray-group.com