FlexRay

A state of the art vehicle bus

Tuesday, 02.12.2008
Embedded Systems Lecture
Chair of Professor Finkbeiner
Saarbrücken

Michael Gerke

Overview

- The FlexRay protocol (who, why, what)
- General setting
- Architecture of FlexRay bus controller
- Initialization
- Time management (TDMA)
- Synchronization
- Message types and format
- Bit level message transfer
- Fault tolerance, error scenarios
- Summary and sources

The FlexRay Protocol

FlexRay is a vehicle bus:
it enables embedded devices in cars to communicate with each other.

The FlexRay consortium:
- BMW
- Bosch
- Daimler
- Freescale (formerly Motorola)
- General Motors
- NXP Semiconductors (formerly Philips)
- Volkswagen

Verifying FlexRay

Verification of FlexRay is a part of the Verisoft Project, which is headed by the Deutsche Zentrum für Luft- und Raumfahrttechnik and receives funding from the Bundesministerium für Bildung und Forschung.

The Verisoft Consortium consists of:
AbsInt, BMW, DFKI, Infeneon, MPI, OFFIS, OneSpin Solutions, Saarland University, T-Systems, TU-Darmstadt, TU-München, University of Koblenz-Landau

FlexRay History and Uses

- Developed since 2000, based on BMWs ByteFlight protocol
- Version 1.1 is used in BMWs X5 2006 car (pneumatic damping)
- Current version is 2.1 from 2005
- New version is expected by the end of the year
- Use for drive by wire in BMWs X6 2009 car


BMW X5 (2006)


BMW X6 (2009)

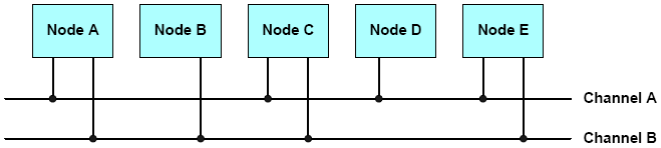
The FlexRay Goals

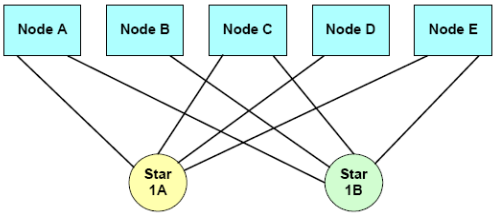
Goals of the FlexRay design:

- High data rates (up to 10 megabits/second)
- Time- and event triggered communication
- Fault tolerance and redundancy

FlexRay Network Topologies

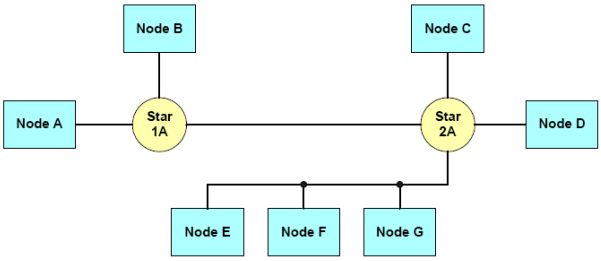
Dual channel bus





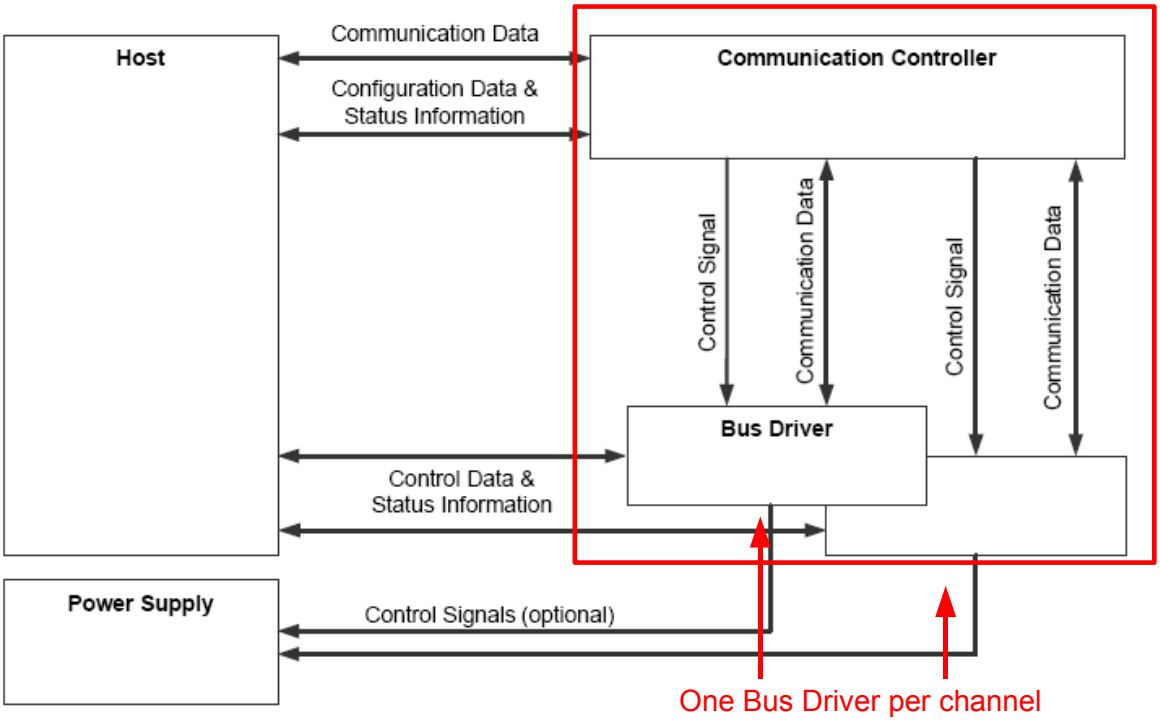
Dual channel star architectures

Mixed architectures

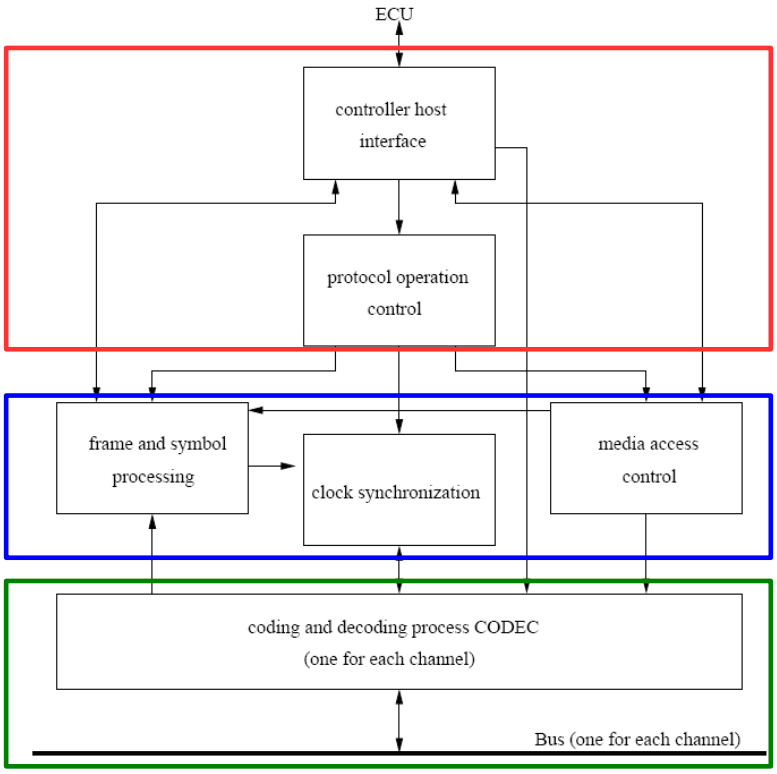


→ up to 2 channels supported

FlexRay Node



Architecture of a Bus Controller



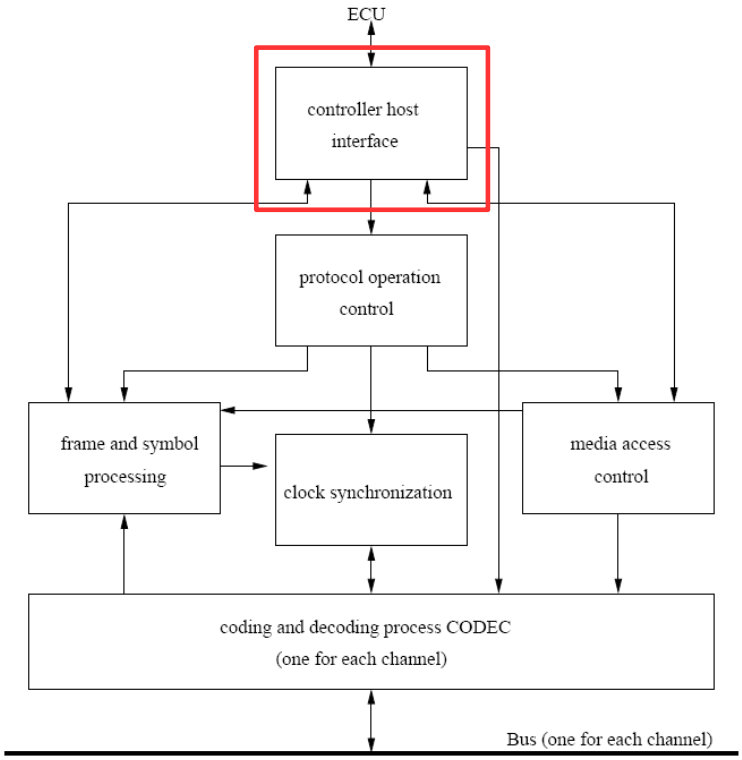
Roughly divided into 3 layers:

Distribution and control layer (1 for each node)

Communication layer (1 for each channel)

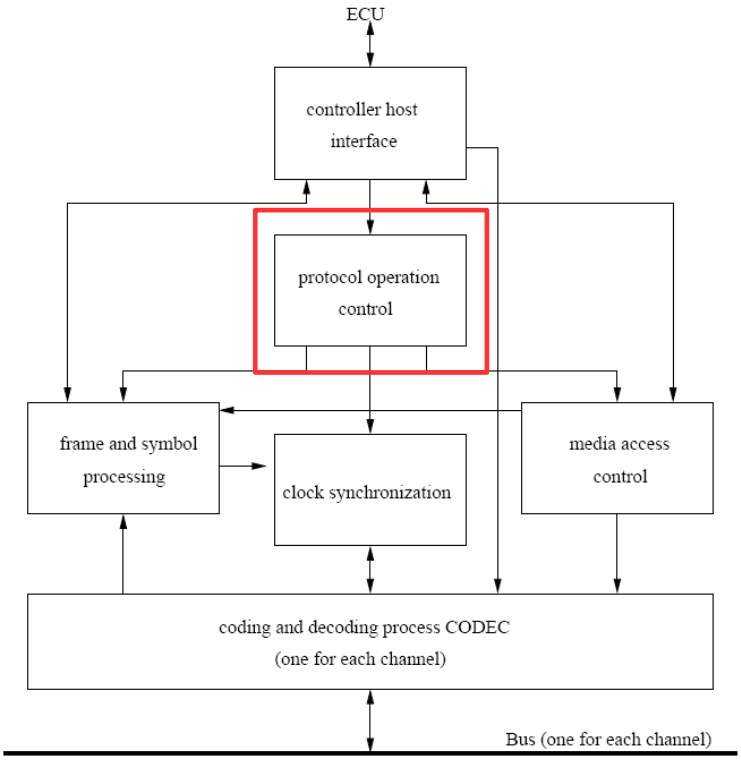
Bitlevel layer (1 for each channel)

Controller Host Interface



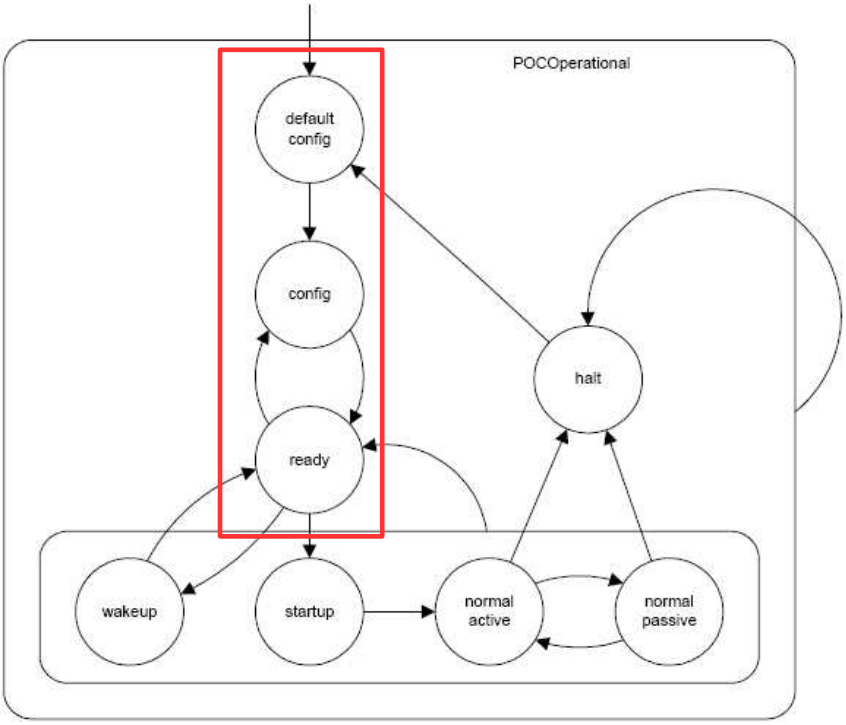
- well defined interface that controls all communication between host and bus driver
- offers host control over configuration (depending on phase)
- offers aggregated status reports to the host
- forwards the hosts commands to the relevant subprocesses
- buffers incomming and outgoing communication

Protocol Operation Control



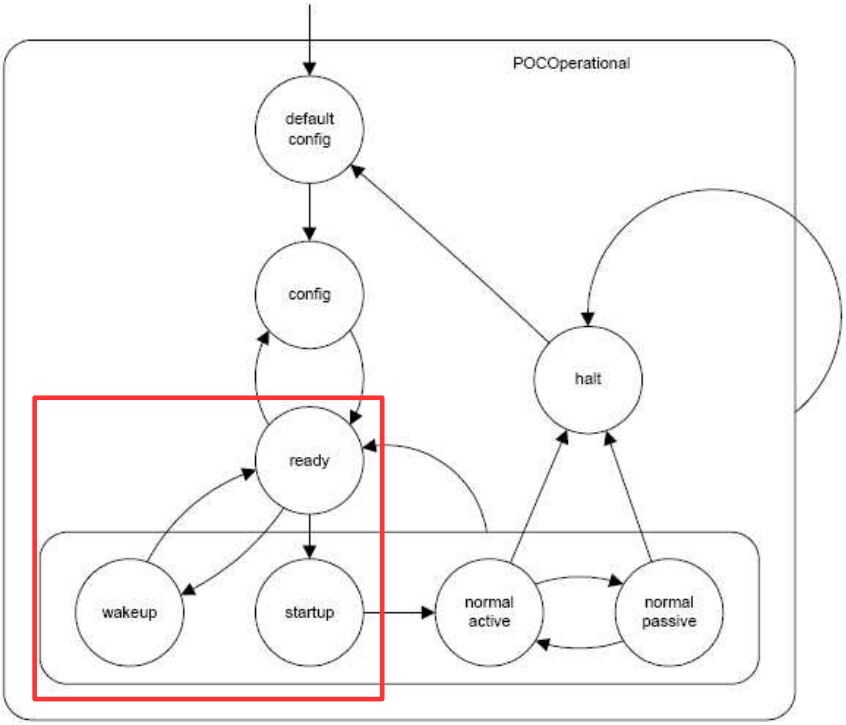
- controls status of the other subprocesses
- controls overall protocol behavior

Protocol Operation Control



Configuration

Protocol Operation Control

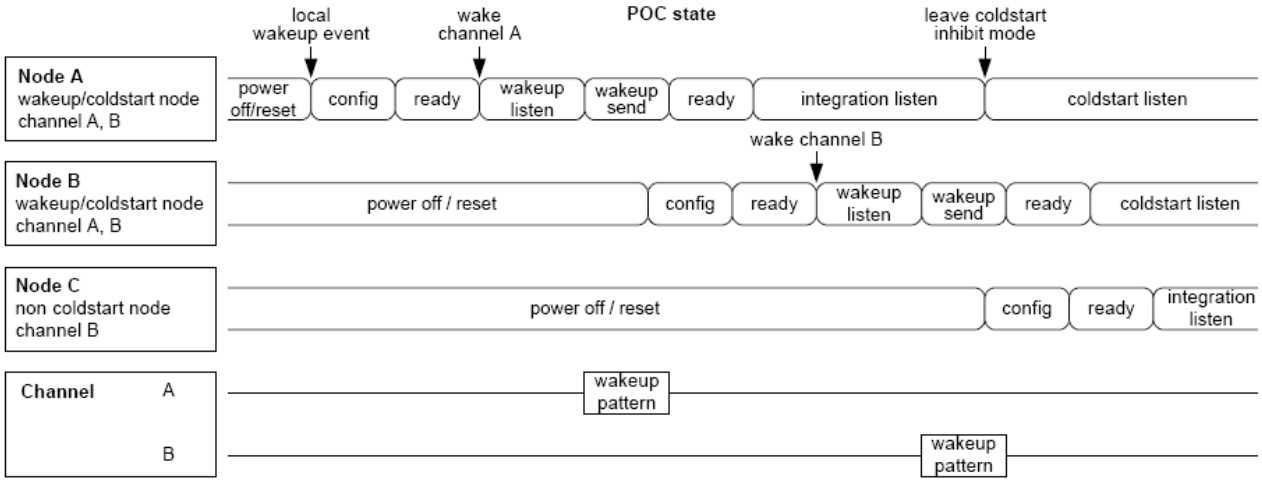


Initialize Communication:

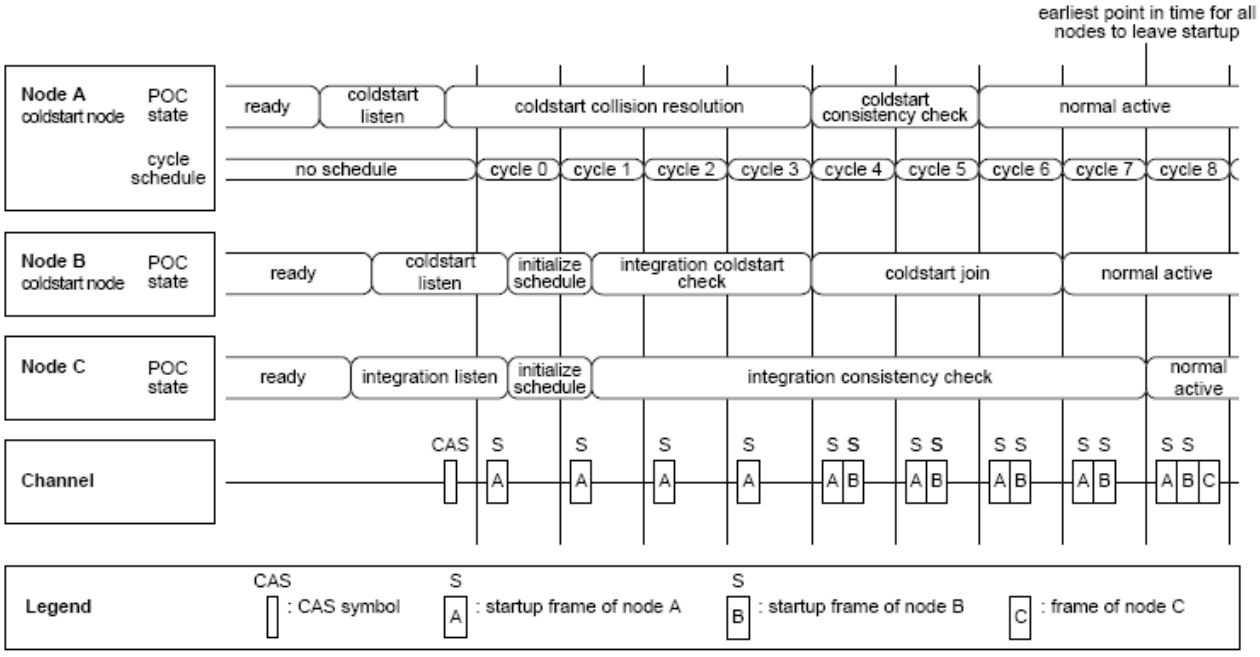
- startup nodes wakeup cluster
- other nodes integrate

Wakeup Example

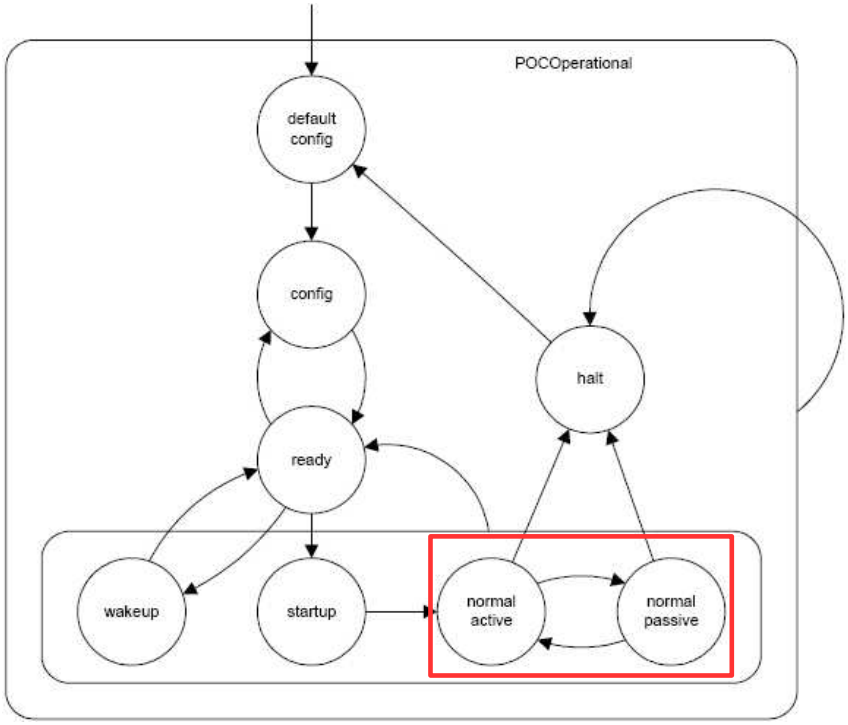
- One node wakes up cluster, the rest integrates



Startup: Example

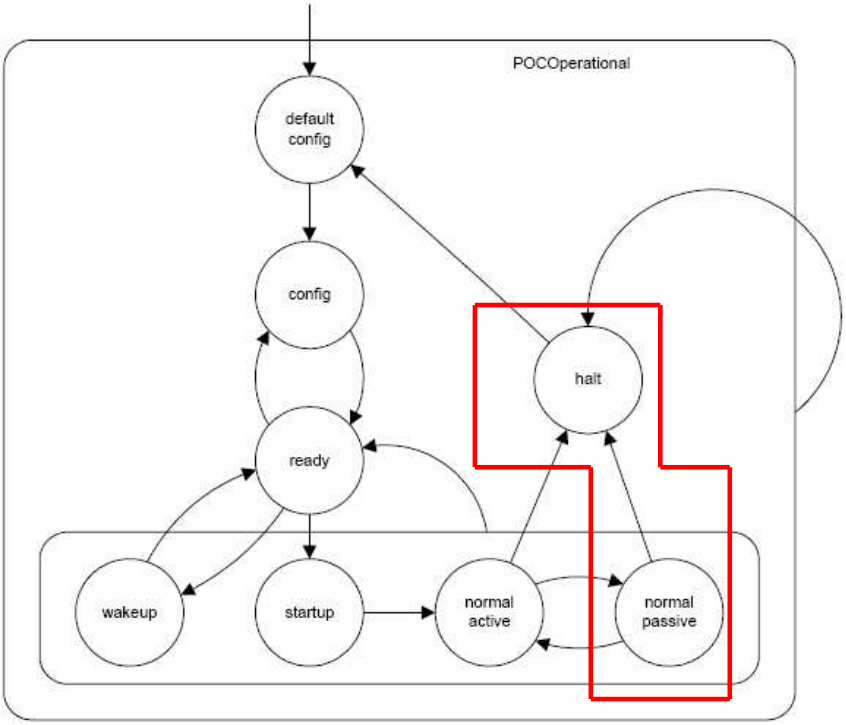


Protocol Operation Control



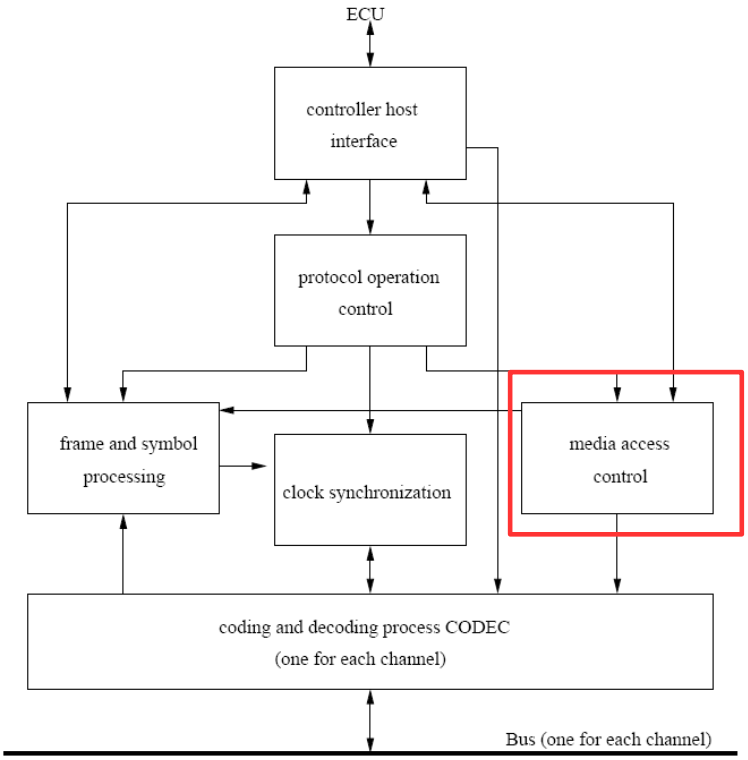
Normal operation

Protocol Operation Control



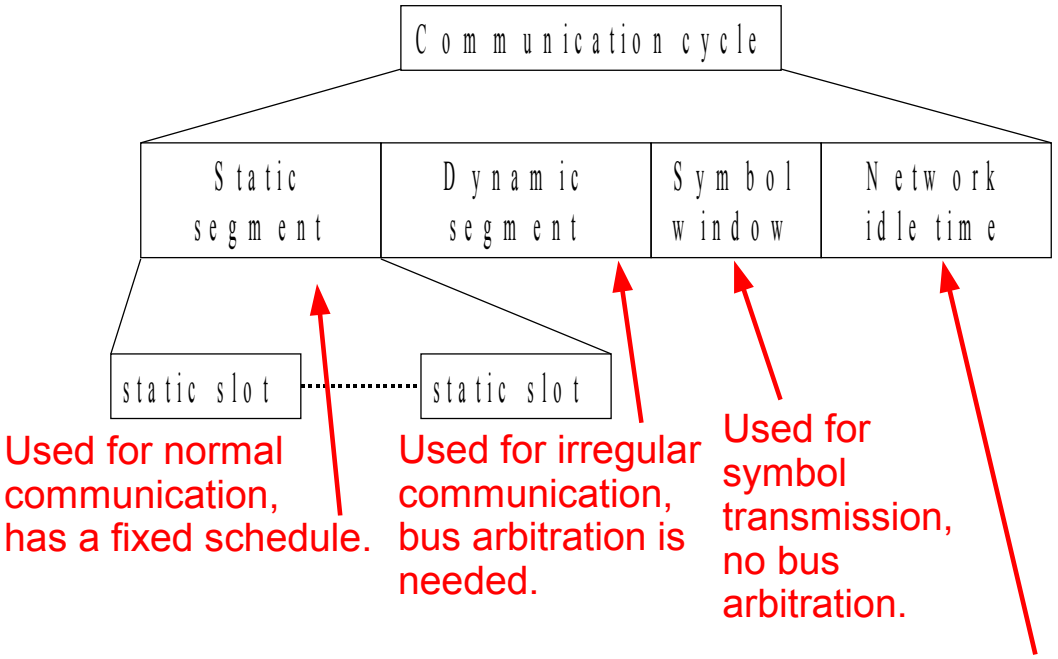
Error handling

Media Access Control

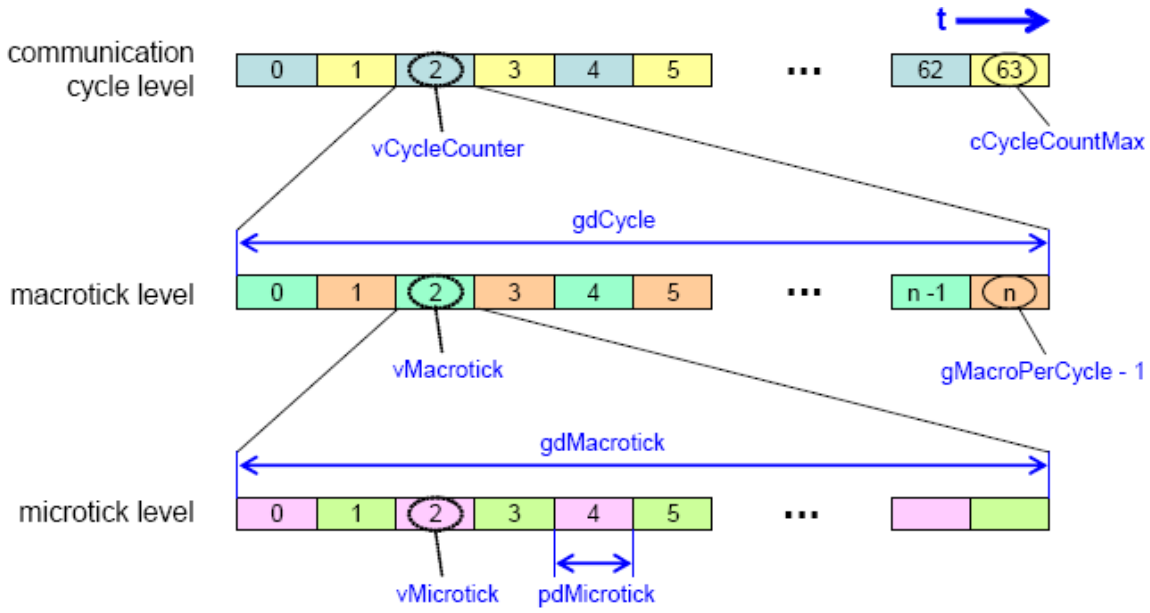


- controls access to the bus
- uses a Time Division Multiple Access (TDMA) scheme
- guarantees compliance with schedule
- assembles frame header

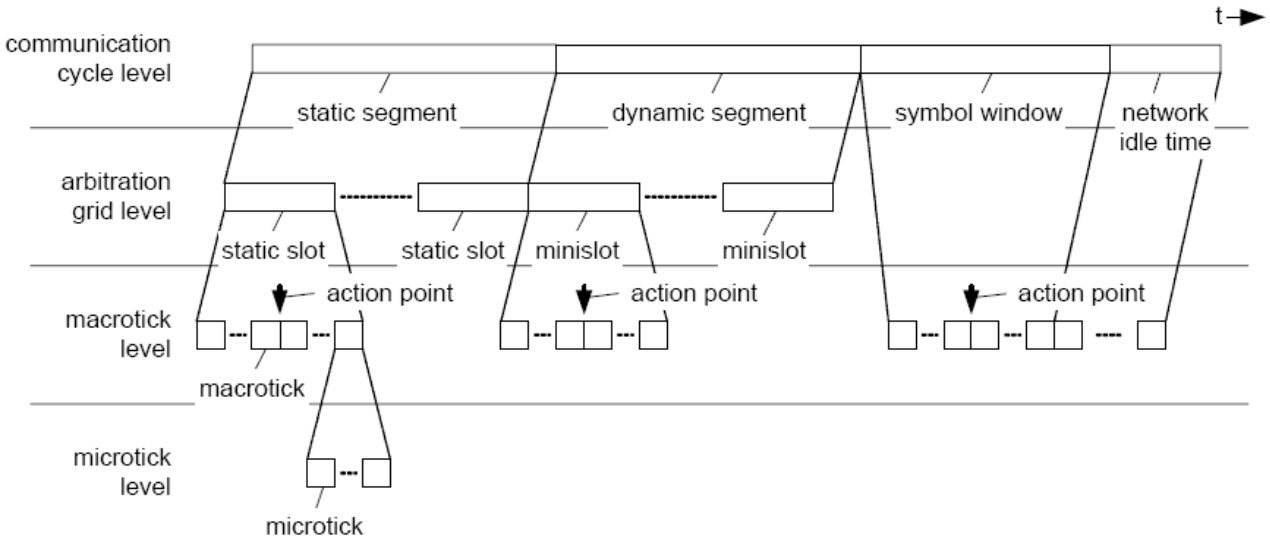
Time Division Multiple Access



Timing of Cycles



Timing Hierarchy

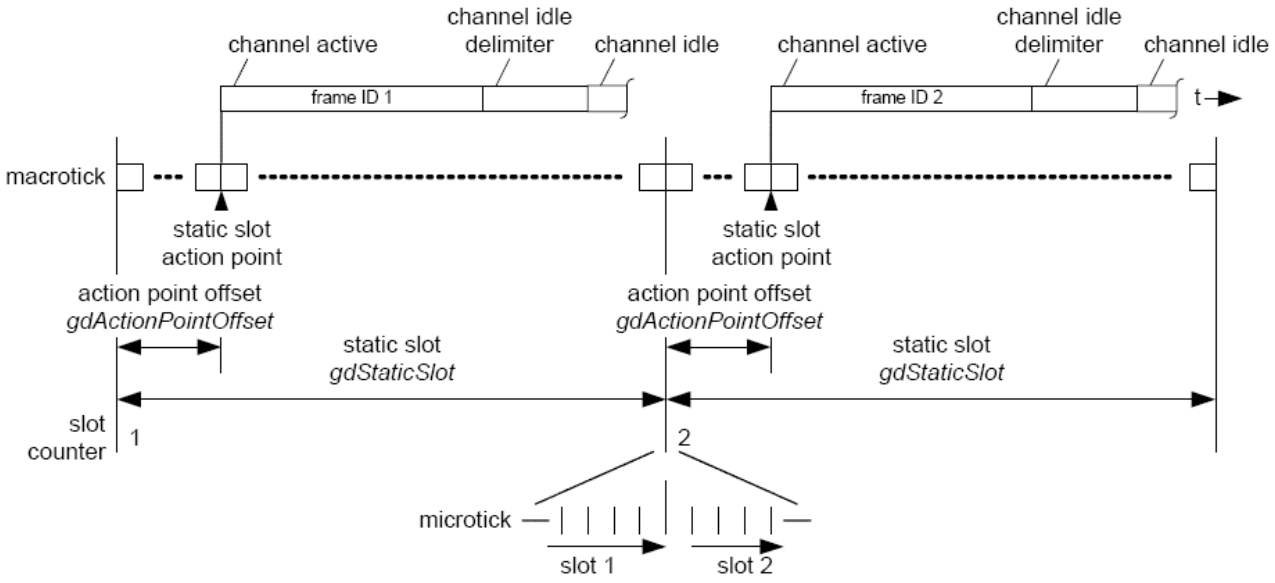


Media access control (MAC)

Slot usage:

- short idle time
- transfer of the frame
- long idle time

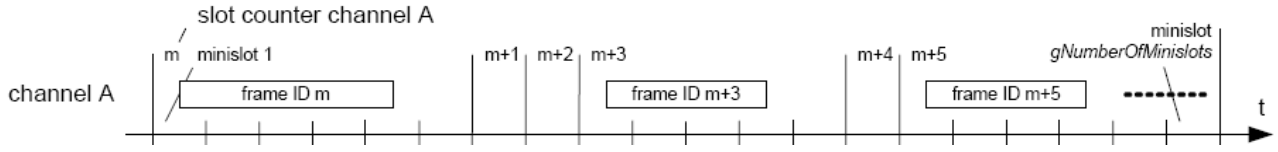
Static Segment



- fixed length slots
- each ECU sends in its own slot

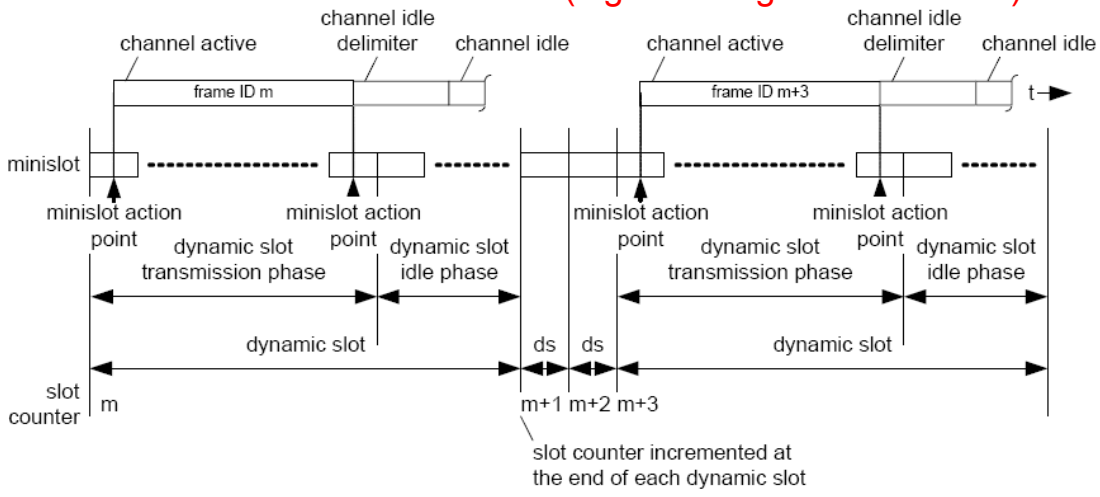
→ fixed communication rate, reliable but unflexible

Dynamic Segement

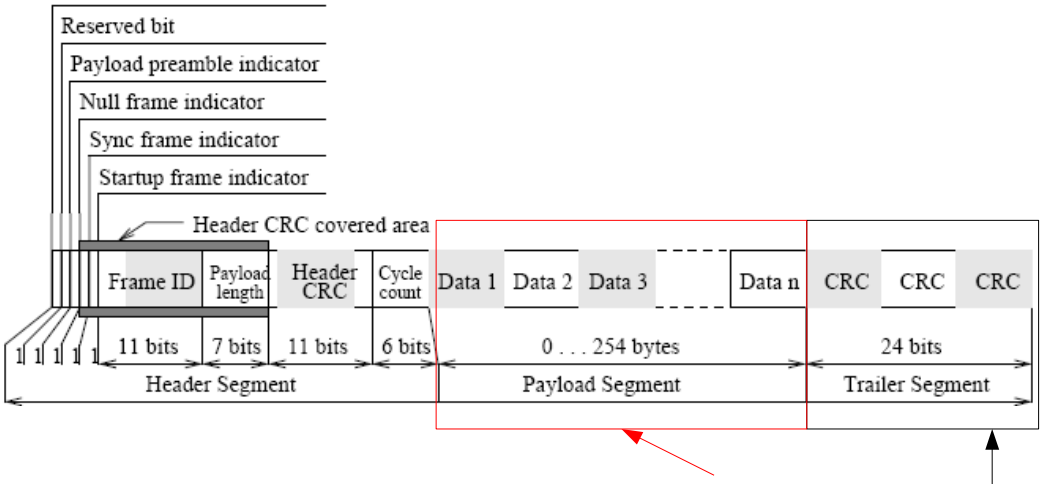


- small minislots as a basis
- dynamic slots, one for each ECU (longer if communication occurs)
- maximum number of minislots → ID is priority

→ more flexible but less reliable (high IDs might be excluded)

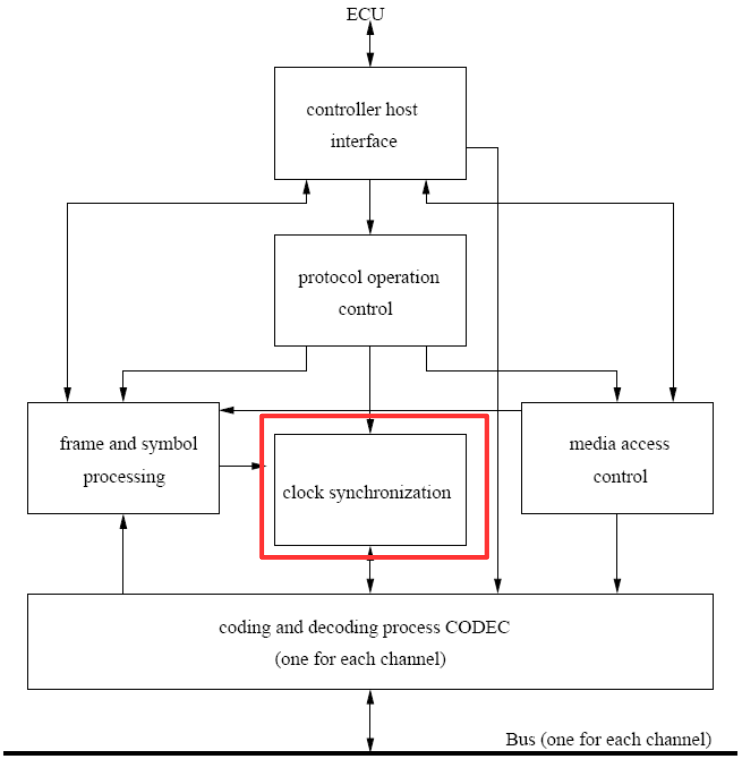


FlexRay Message Frame



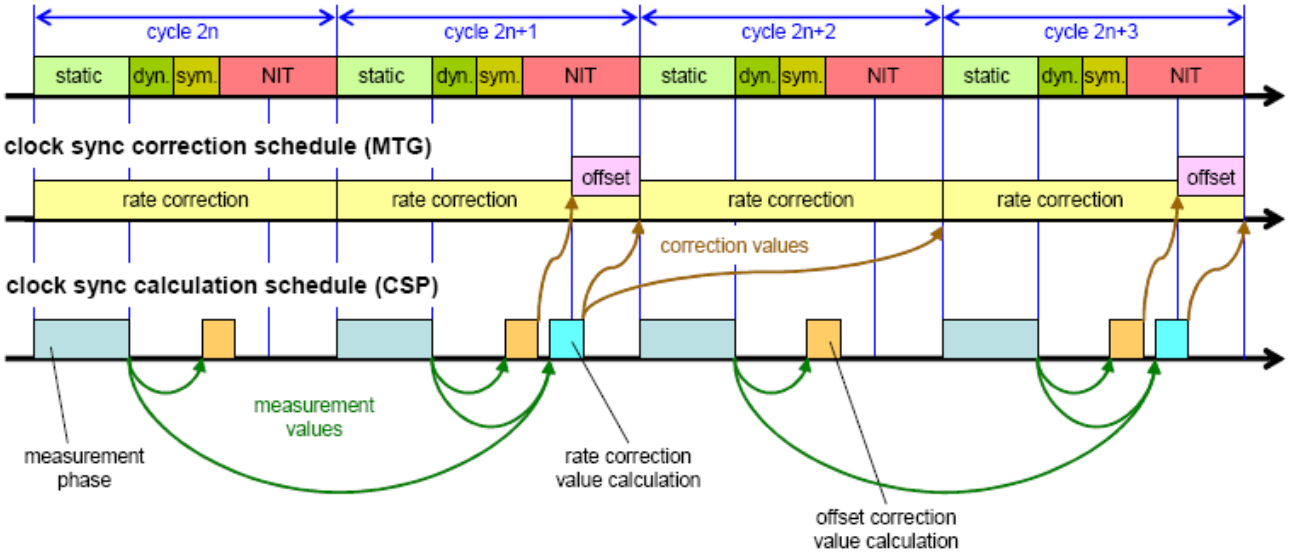
- Message in payload section: up to 254 bytes
- CRC used to notice transmission errors

Clock Synchronization



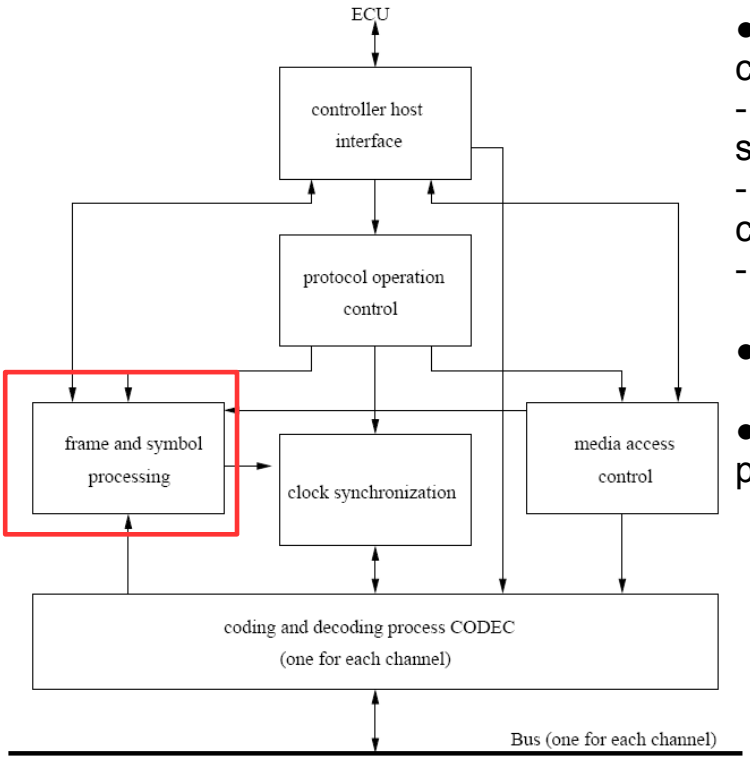
- tries to establish a shared view of time (to a certain extend)
- uses sync frames to get an impression of the time assumptions of other nodes
- adjusts the length of communication cycles

Clock Synchronization



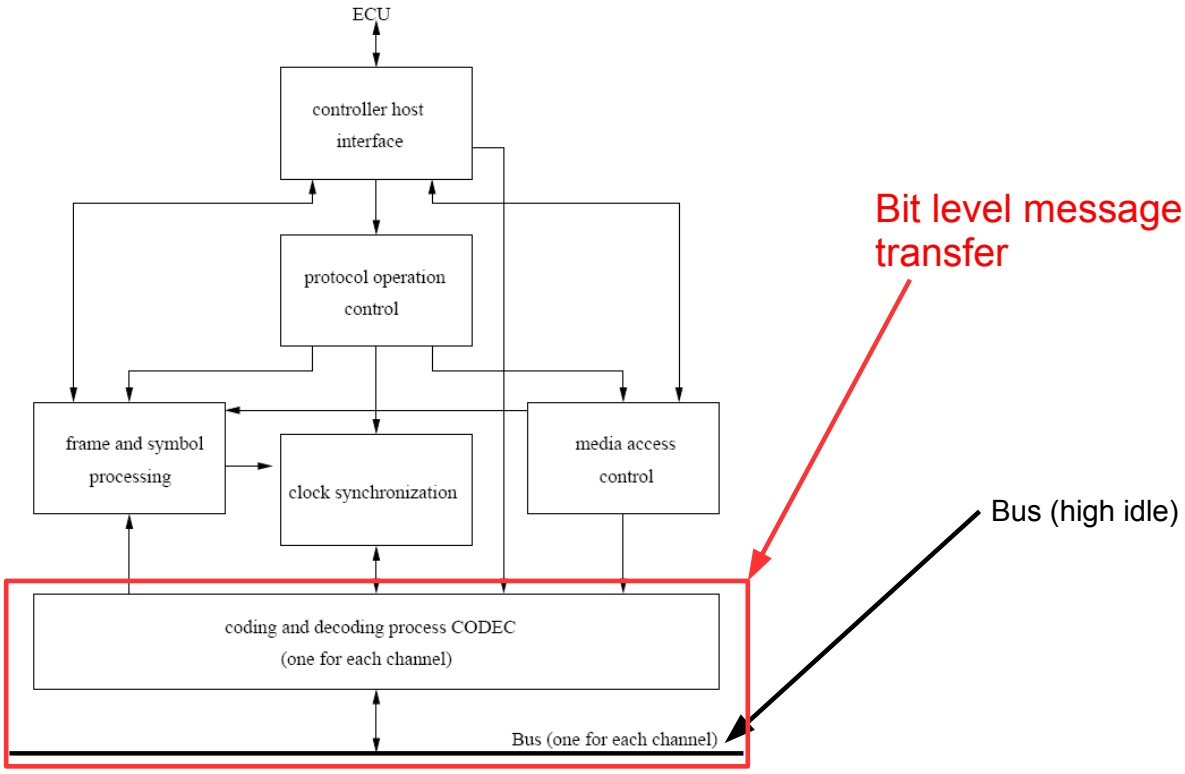
- rate correction adjusts the numbers of microticks per macrotick
- offset is an adjustment to the number of macroticks in the current cycle

Frame and Symbol Processing

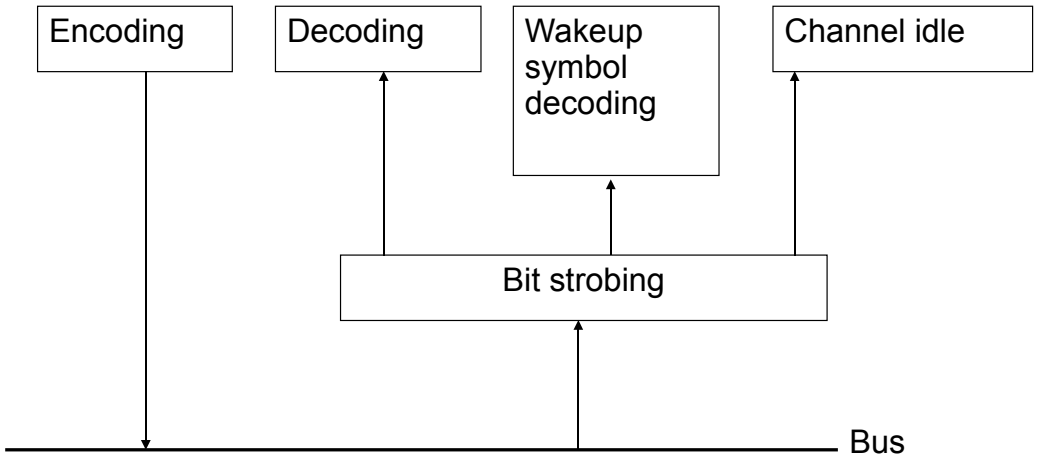


- controls integrity of communication:
  - Checks for appropriate segment
  - Checks length of communication elements
  - Checks for correct header
- Reports errors
- Dismantles frame and passes on the payload

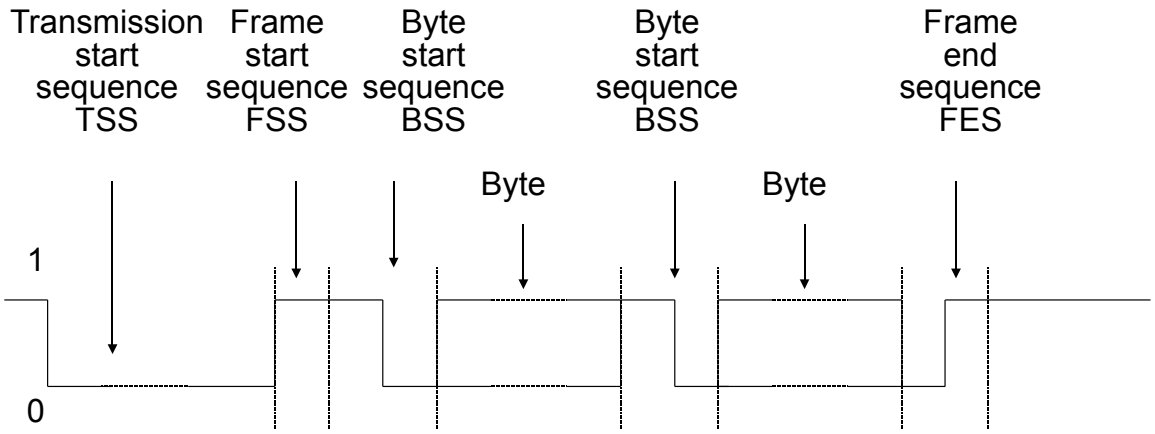
The FlexRay Controller



Coding and Decoding



Encoding (ENC)
Frames



Frame coding in static segment

Encoding (ENC)
Symbols

Collision avoidance and

media access test symbol (CAS/MTS):

$TSS+0^{30}$

Wakeup symbol (WUS):

$0^{15\text{-}60}+idle^{45\text{-}180}$

sent in a wakeup pattern (WUP): $WUS^{2\text{-}63}$

Wakeup symbol decoding (WUSDEC)

- Waits for a WUP like pattern:
  $0^x+1^y+0^x$
  produced by two consecutive WUS

- Reports to protocol operation control(POC)

Channel idle (IDET)

Whenever 1^{11} is received,
a channel idle recognition point (CHIRP)
is reported.

Decoding (DEC)

- Handles reception of CAS/MTS and frames
- Checks for errors

Frames:

- reassembles the message
- checks format
- checks CRC (frame “fingerprint”)

CAS/MTS:

- checks length of LOW signal

FlexRay Voting and Strobing

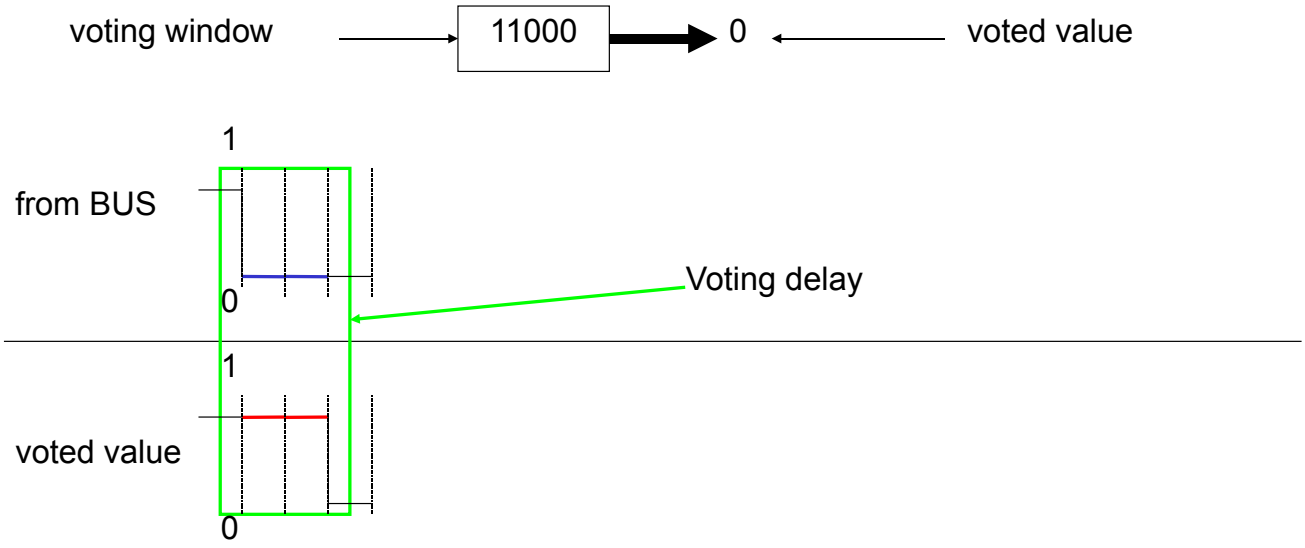
The bitstream has a structured format, and:

- Every bit is sent 8 times
- Just the 5^{th} bit of this byte is considered (“strobed”)
- Majority vote over the last 5 bits

→ Redundancy is used to provide fault tolerance
→ A means of low level synchronization is provided

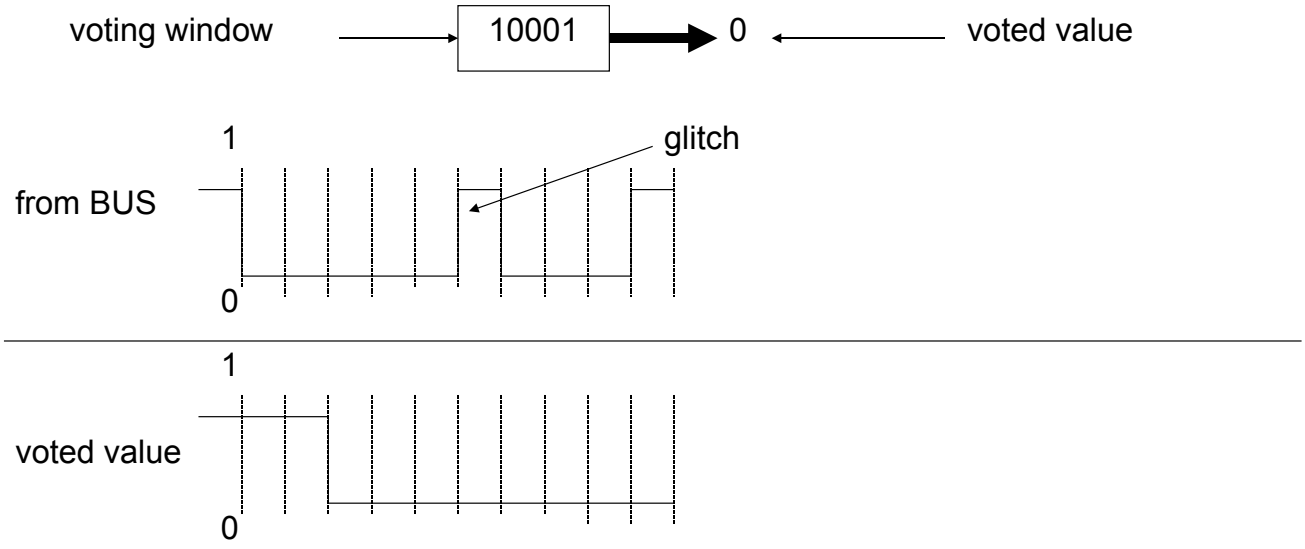
Bit strobing: Majority voting

Majority voting over last 5 bit samples



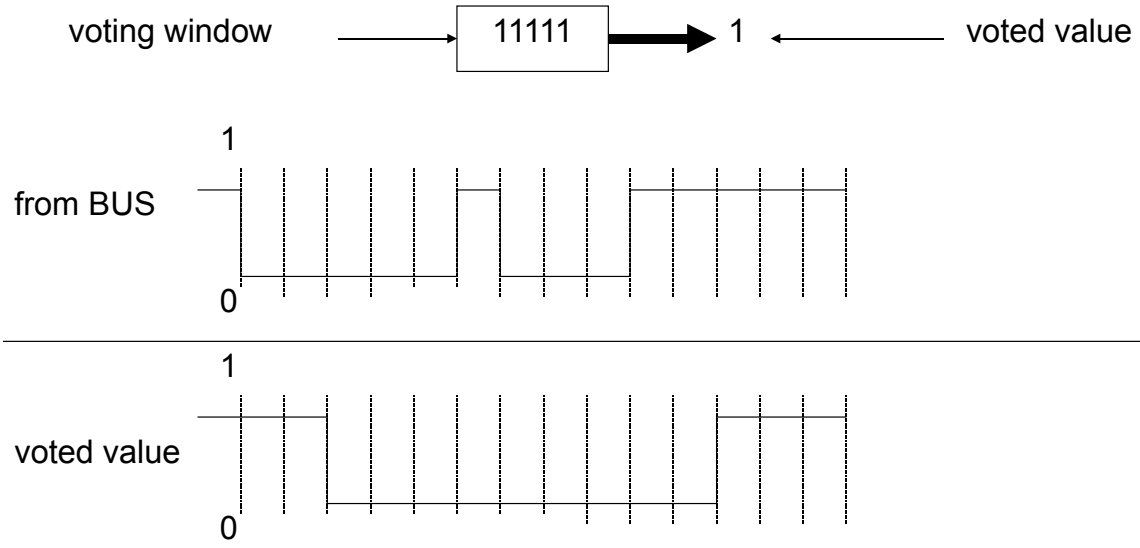
Bit strobing: Majority voting

Majority voting over last 5 bit samples



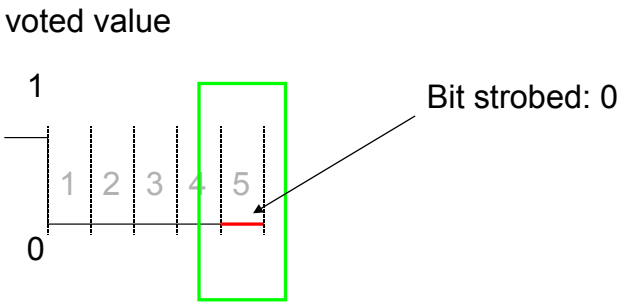
Bit strobing: Majority voting

Majority voting over last 5 bit samples



Bit strobing: Strobing

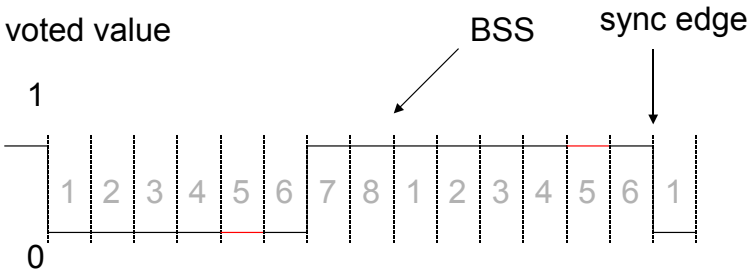
- Every 5th sample out of the 8 samples „strobed“
- Low-level synchronization of strobecounter in BSS



Strobecounter: 5

Bit strobing: Synchronization

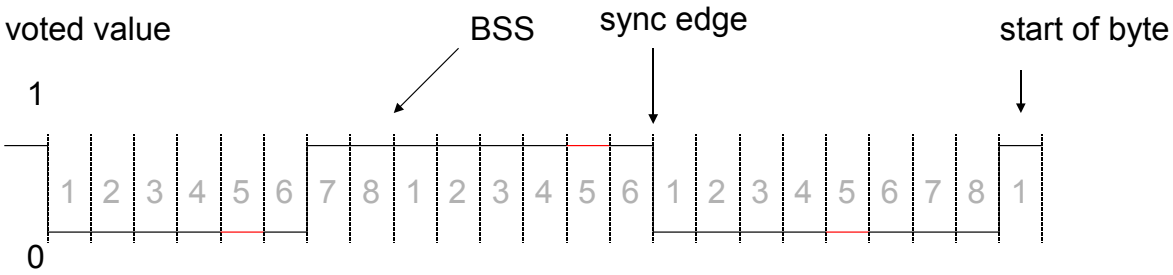
- Every 5th sample out of the 8 samples „strobed“
- Low-level synchronization of strobecounter in BSS



Strobecounter: 1

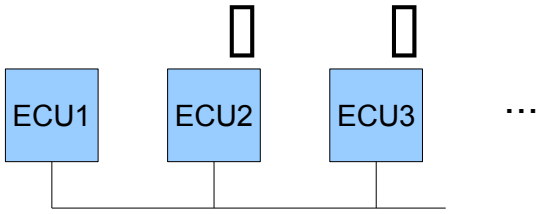
Bit strobing: Synchronization

- Every 5th sample out of the 8 samples „strobed“
- Low-level synchronization of strobecounter in BSS



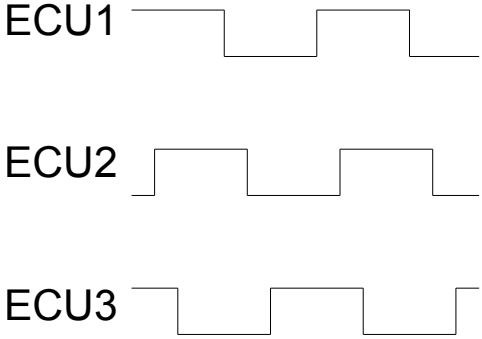
Strobecounter: 1

Why all this “Overhead”?



Node = ECU
(Electronic Control Unit)

- Each ECU has its own oscillator



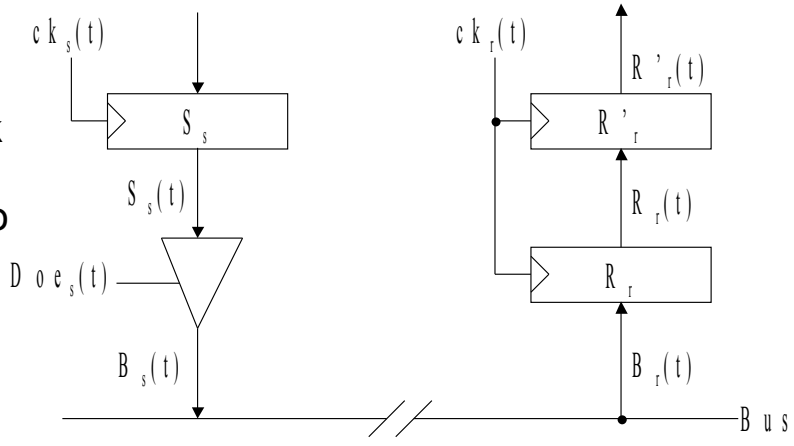
- Oscillators may deviate at most 0.15% from standard

→ Clock edges will ***not*** be at the same time in all ECUs.

Asynchronous Clocks

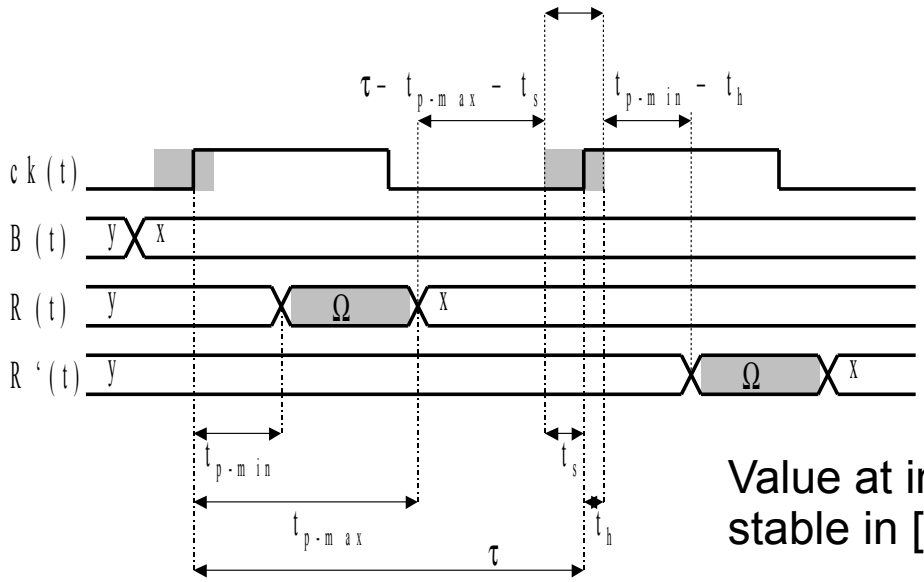
Two ECUs with ***separate*** oscillators are connected

- Digital setting inside ECUs
- The clock signals ck are asynchrounous
- All signals relative to time t
- Time is continuous



Lower indices: X_s = sender's X and X_r = receiver's X

Register Semantics



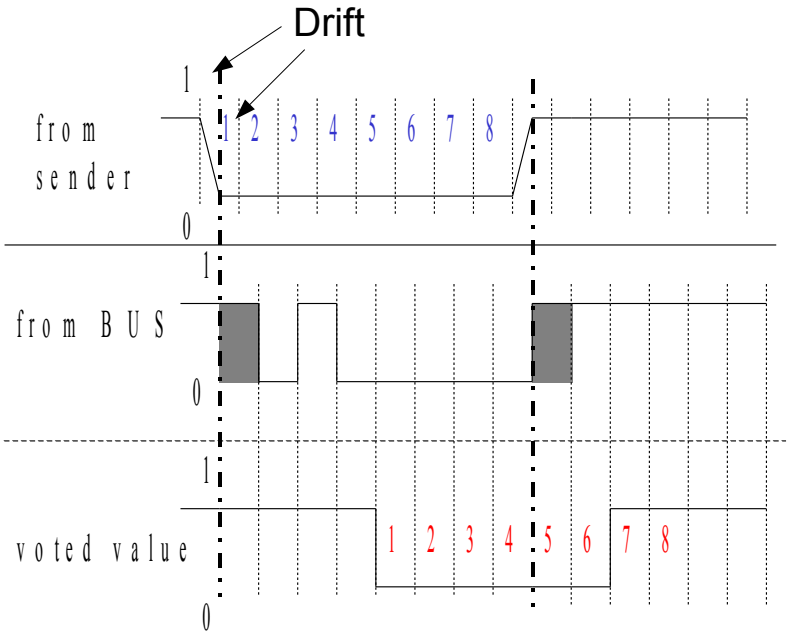
Value at input has to be stable in [ck-t_s,ck+t_h]

τ =cycle time

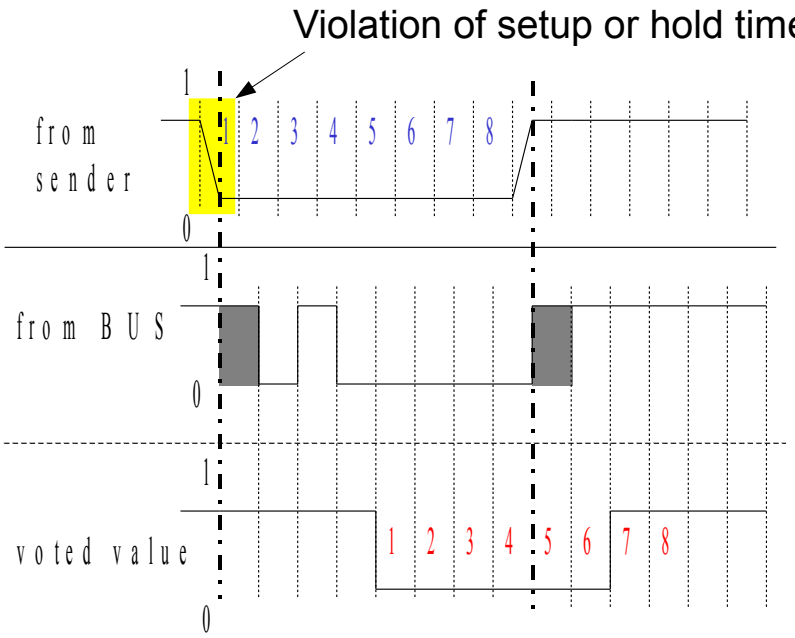
$t_{p\text{-}min/max}$ =delay of voltage change and signal propagation

$t_{s/h}$ =hold and setup times of the register

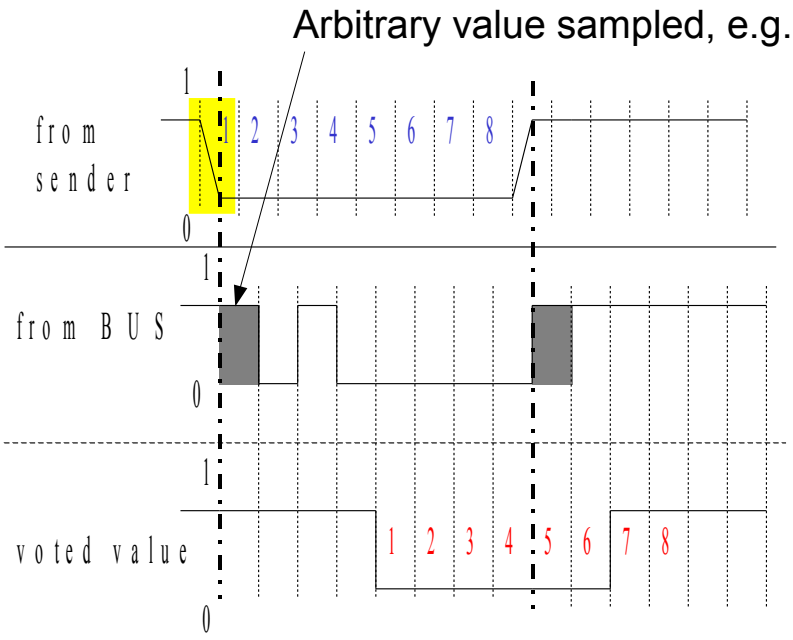
Error: Late Synchronization



Error: Late Synchronization

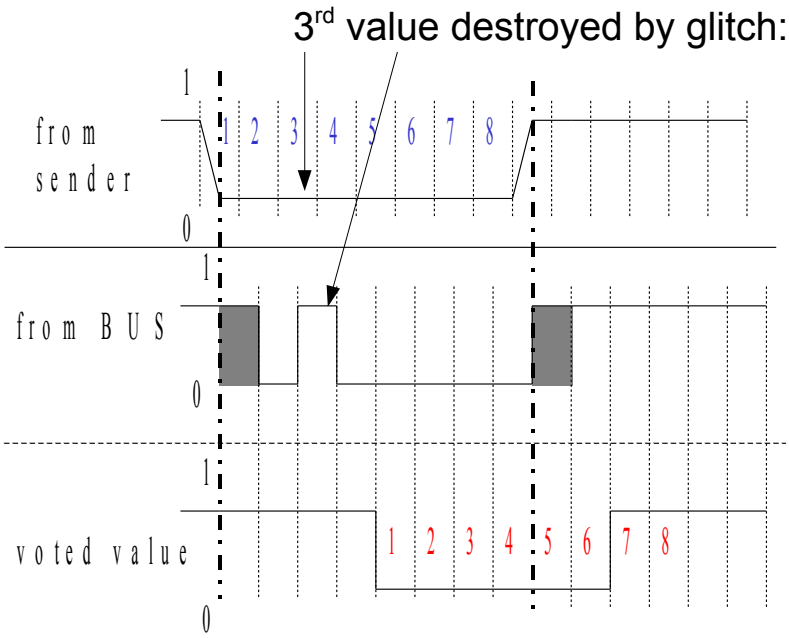


Error: Late Synchronization



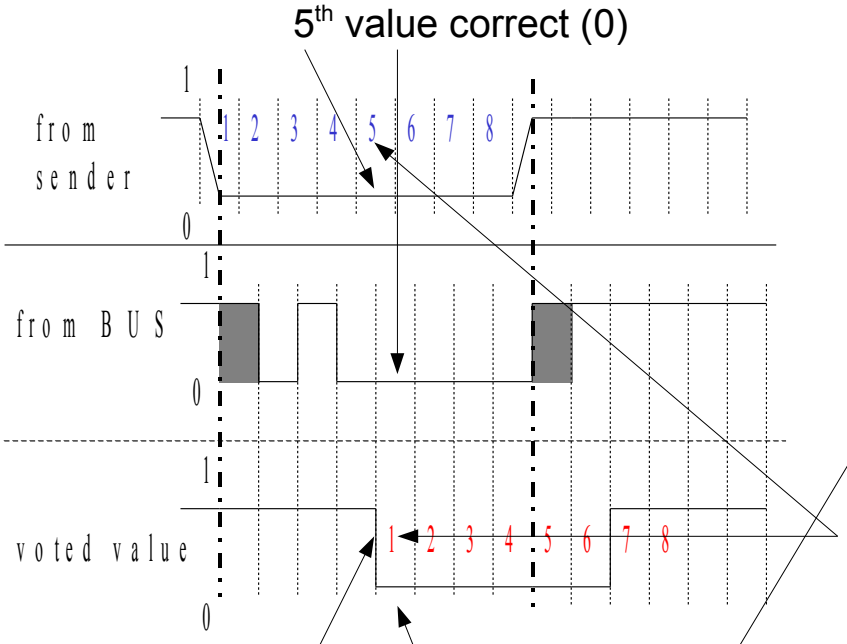
Voting window:
<1,1,1,1,1>

Error: Late Synchronization



Voting window:
<1,1,1,0,1>

Error: Late Synchronization



Voting window:
<1,0,1,0,0>

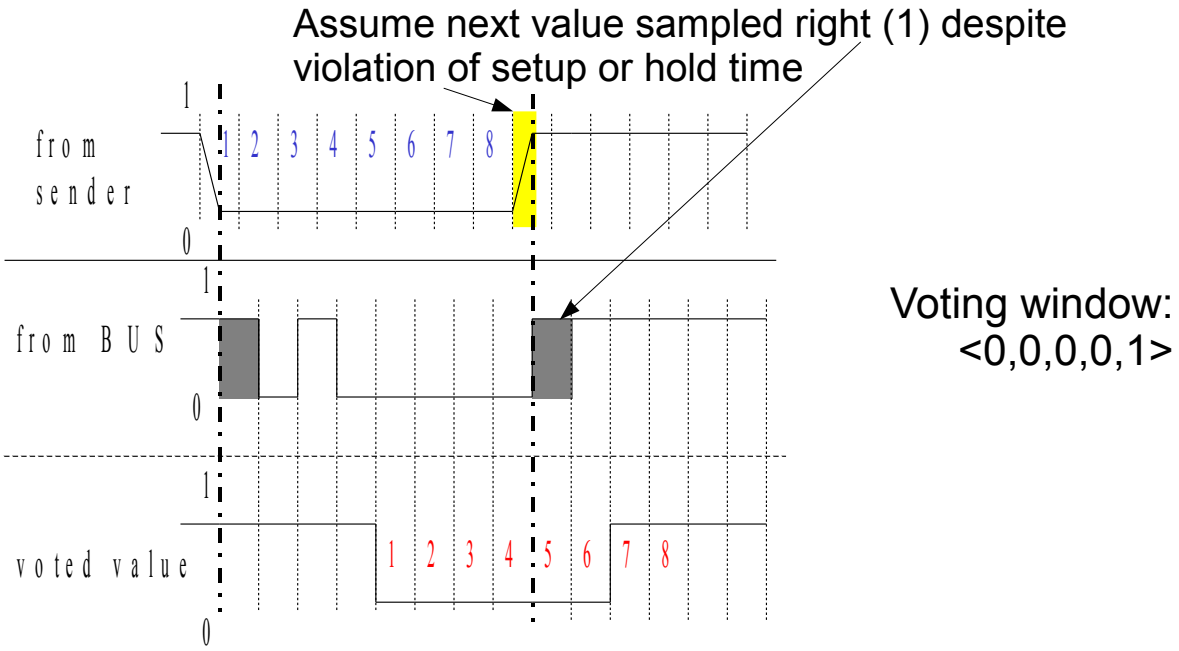
→ Counter offset 4
instead of 2

→Counter late by 2

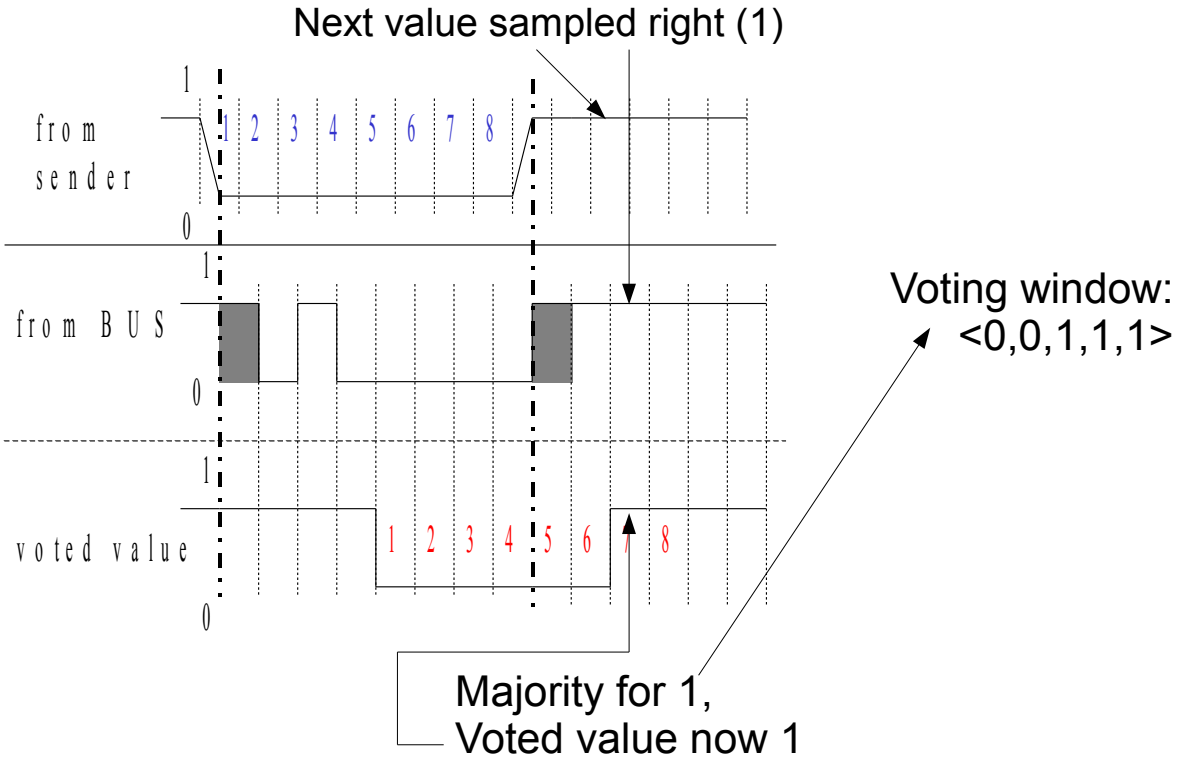
Synchronization edge

Majority for 0,
Voted value now 0

Error: Late Synchronization



Error: Late Synchronization



The FlexRay Communication Protocol

- specified for a dependable automotive network

Basic characteristics:
- synchronous and asynchronous frame transfer
- guaranteed frame latency and jitter during synchronous transfer
- prioritization of frames during asynchronous transfer
- multi-master clock synchronization
- error detection and signaling
- error containment on the physical layer through the use of a bus guardian device
- scalable fault tolerance

Sources

FlexRay Consortium, *FlexRay Communications System Protocol Specification Version 2.1*, 2005, available at http://www.flexray.com/

Sven Beyer, Peter Böhm, Michael Gerke, Mark Hillebrand, Tom In der Rieden, Steffen Knapp, Dirk Leinenbach and Wolfgang J. Paul, *Towards the Formal Verification of Lower System Layers in Automotive Systems*. In *ICCD '05*, pages 317-324. IEEE Computer Society, 2005

Closely Related Work

(***very*** detailed descriptions, just to satisfy your curiosity)

Coding / Decoding:
Michael Gerke, *Implementation of Frame and Symbol Transmission in a Time Triggered Serial Bus Architecture*, Bachelor Thesis, Universität des Saarlandes, March 2007, available at http://www-wjp.cs.uni-sb.de/publikationen/Ger07.pdf

Clock Synchronization:
Peter Böhm, *Implementation of the High-Level Components of a Bus Controller for a Time-Triggered Serial Bus*, Bachelor Thesis, Universität des Saarlandes, July 2006, available at http://www-wjp.cs.uni-sb.de/publikationen/Boe06.pdf