Automotive Software Engineering

Lecture 2 - Communication

Prof. Dr. Wolfram Hardt
Owes Khan
Norbert Englisch

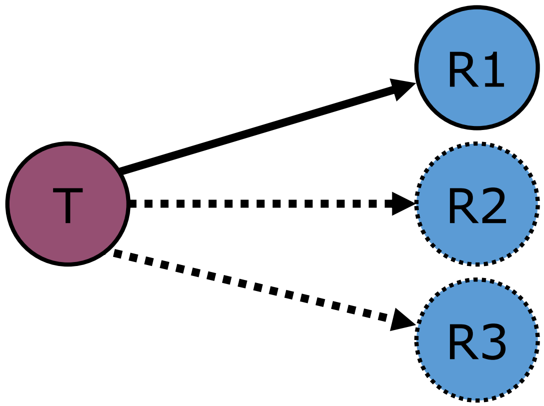
Contents

- Basics
- Controller Area Network (CAN)
- Technical Backgrounds
- Practical Labs

Communication

- (abstract) **declaration**

  Communication is the transmission of information from one transmitter to at least one receiver.

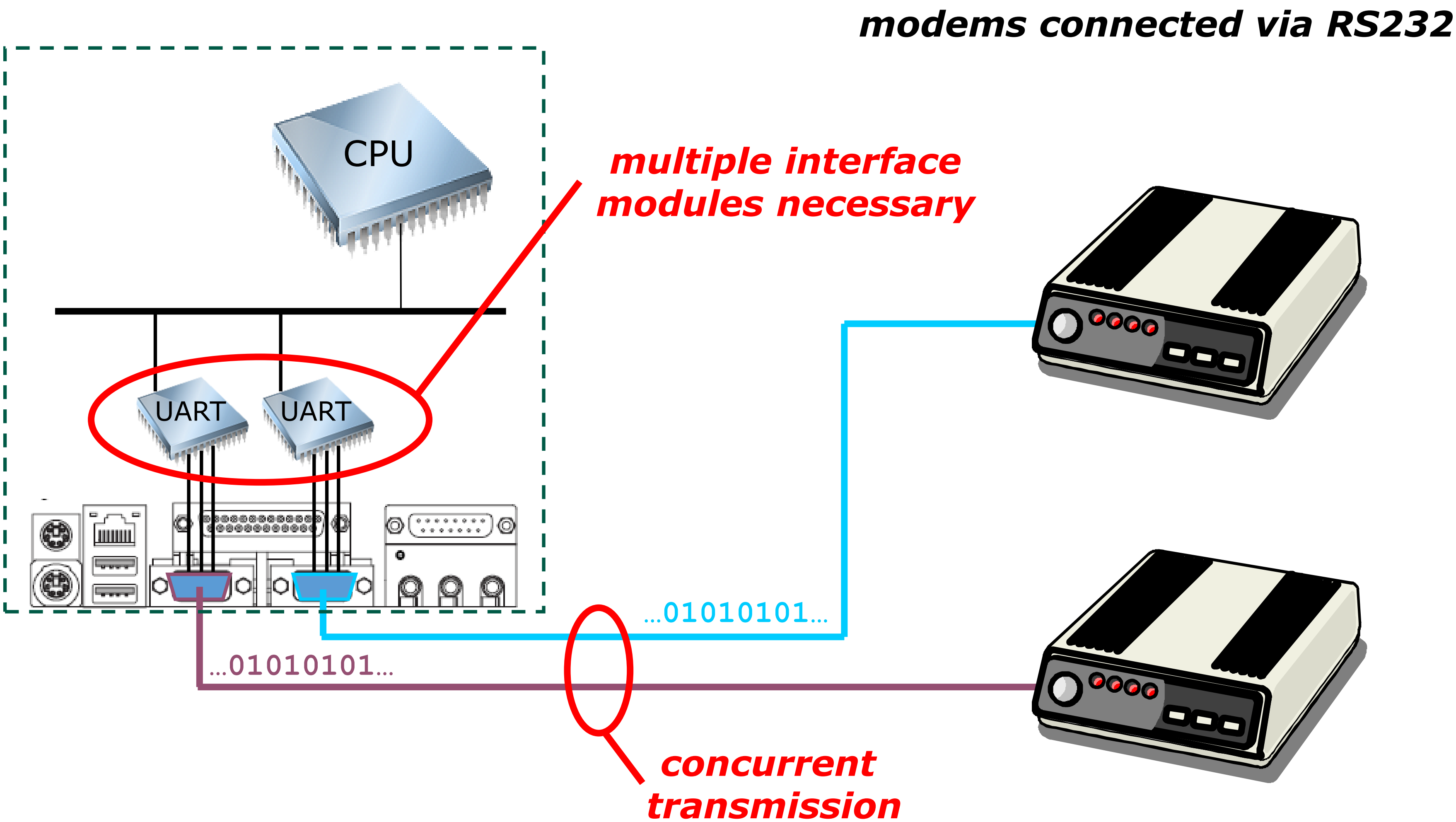


- transmitted kind of information depends on abstraction level (e.g. transmit "a website" → ASCII signs → bits)

Point-To-Point Communication

- an interface module converts internal signals of the system in standardised signals of the interface
- a communication is called point-to-point (P2P), if it is limited to one transmitter and ONE receiver by (electrical, mechanical and/or functional) properties of interface module
  - intermediate stations (router, hubs, …) are not allowed
- advantages:
  - no device addressing and arbitration techniques necessary
  - Real-time properties can easily be fulfilled
  - in many cases no data or packet formatting necessary
- disadvantages:
  - many dedicated interface modules and wires are necessary to communicate with several devices

Example


modems connected via RS232
CPU
UART
UART
multiple interface modules necessary
...01010101...
...01010101...
concurrent transmission

Bus Systems

- **definition**[1]

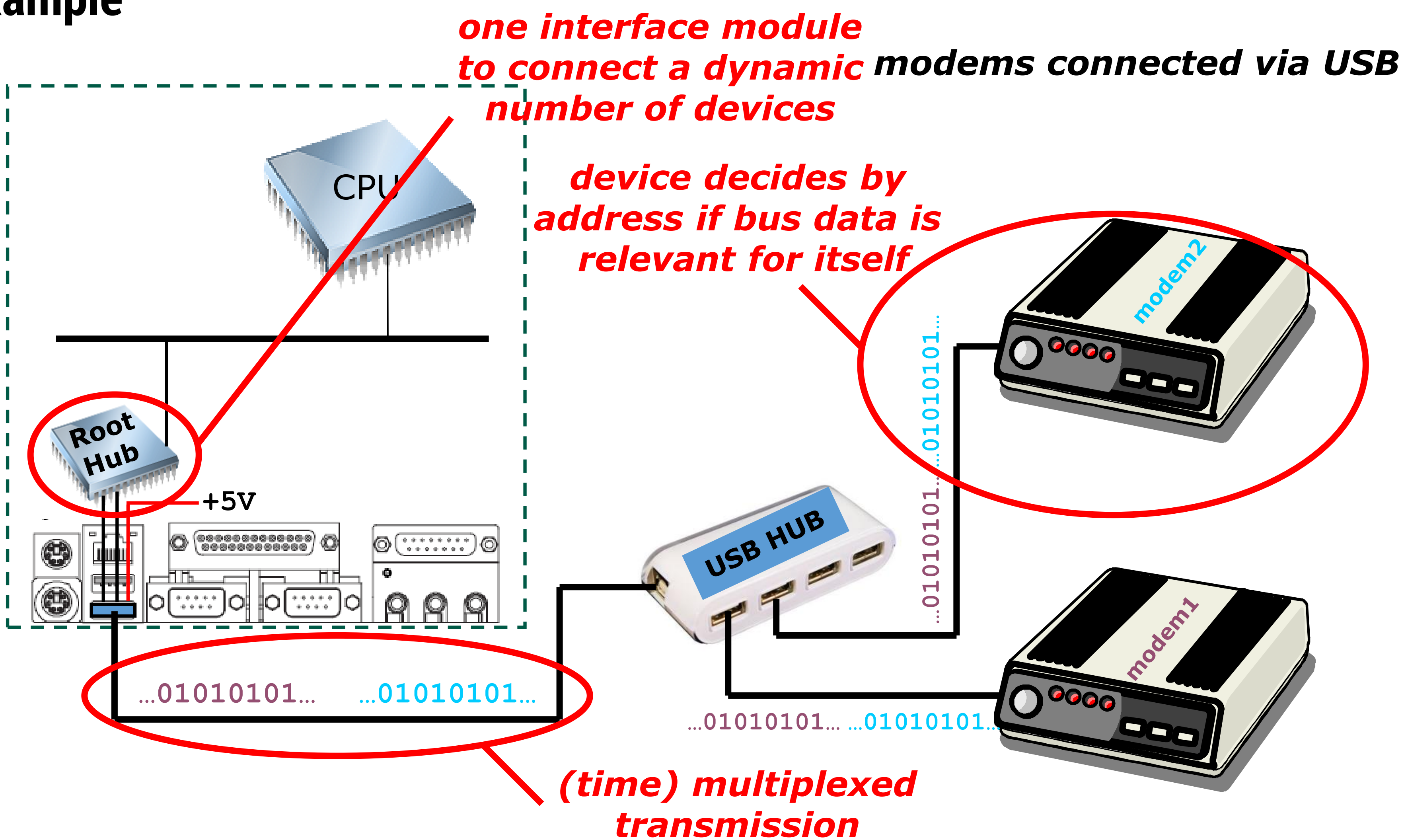
  A bus is a multi-conductor line, which allows data- and information-interchange between different system components […]. It connects all according components of a system […]. The information-interchange between the components is realised by multiplexing.

  → one transmitter (at one time), many (possible) receivers

- advantages
  - one interface module per device to communicate with all other devices
  - multi-/broadcasting possible
- disadvantages
  - slower communication speed (due to multiplexing)
  - solve fairness problems in arbitrating

[1] *Bernd Schürmann: Grundlagen der Rechnerkommunikation. Friedr. Vieweg & Sohn Verlag, Wiesbaden 2004*

Example


one interface module to connect a dynamic number of devices
modems connected via USB
CPU
device decides by address if bus data is relevant for itself
Root Hub
+5V
USB HUB
modem2
modem1
...01010101... ...01010101...
...01010101... ...01010101...
...01010101... ...01010101...
(time) multiplexed transmission

ISO/OSI Model

- define different layers of abstraction in communication

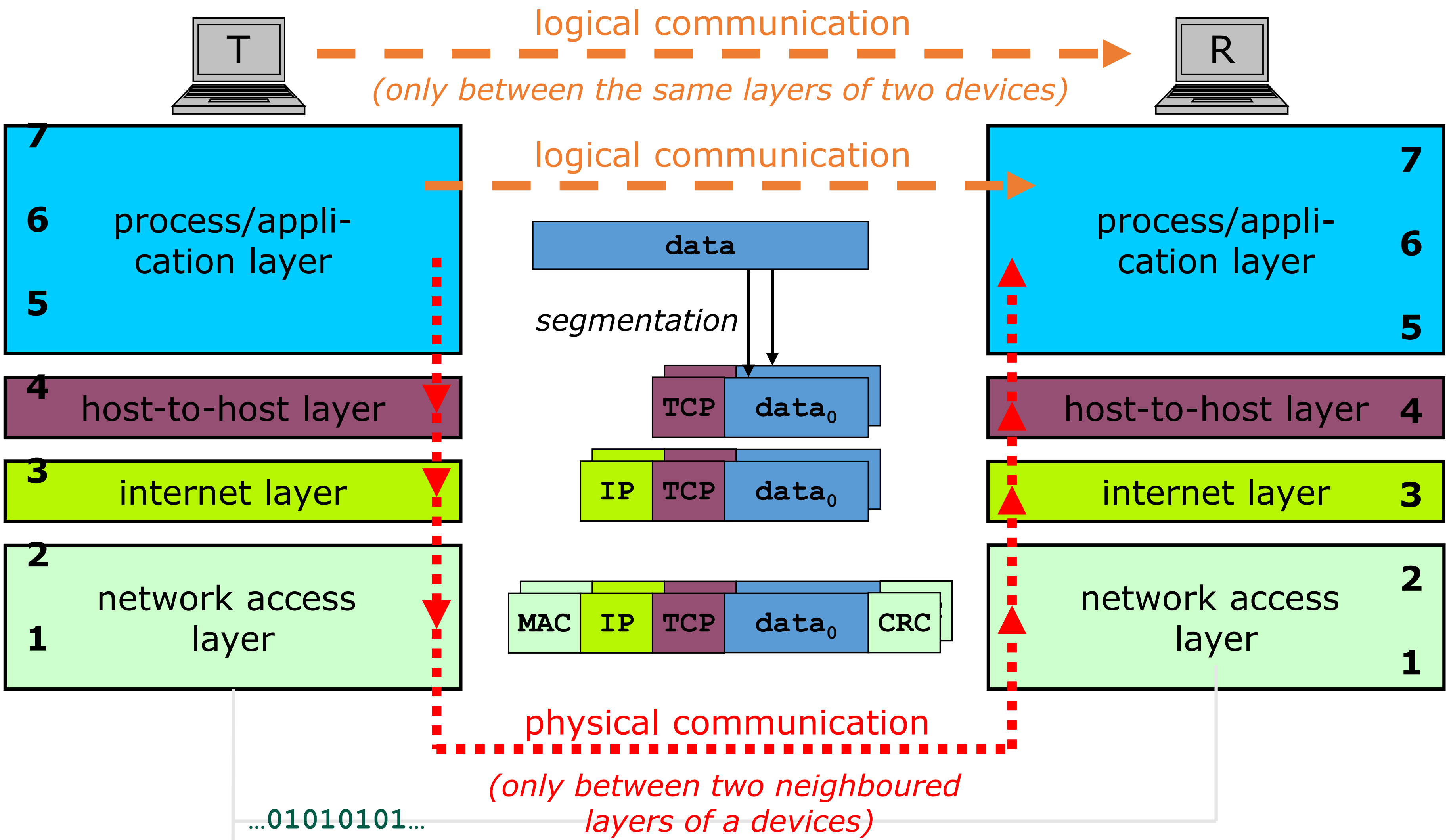
7	application layer	*mail client*
6	presentation layer	*de/encryption, compression*
5	session layer	*session control (start, stop, …)*
4	transport layer	*segmentation, packet ordering*
3	network layer	*data routing*
2	data link layer	*point-to-point transmission*
1	physical layer	*electrical modulation, cabling*

application

transportation

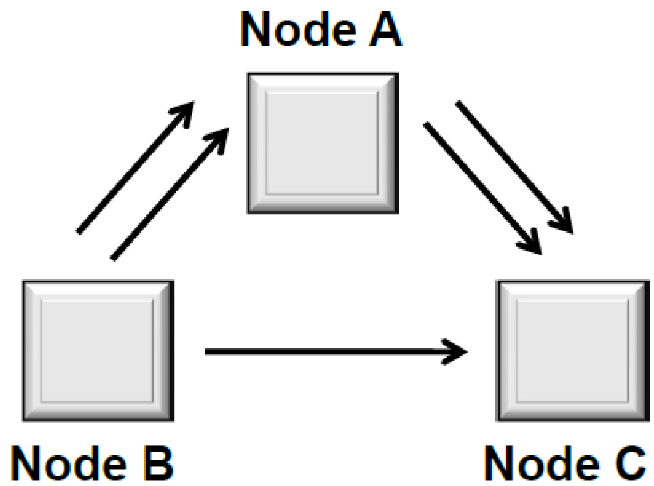
…01010101…

Example: TCP/IP Communication

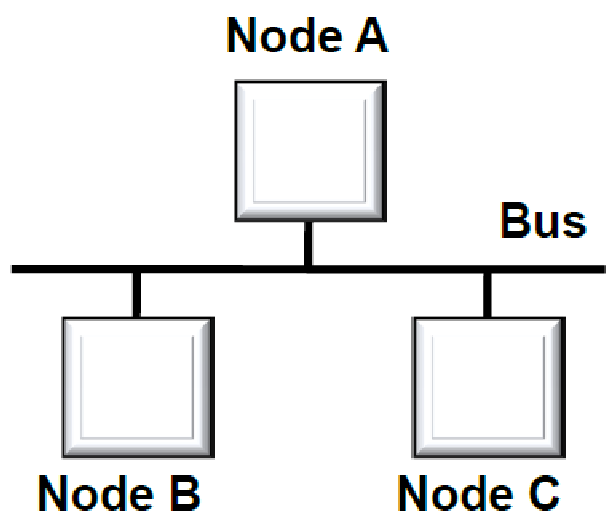


CAN Bus

Logical system architecture



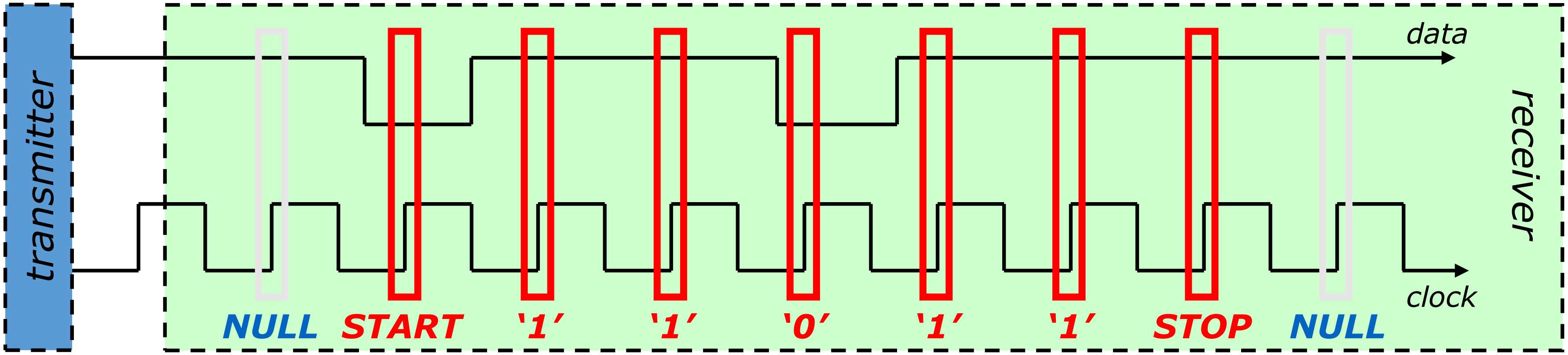
Technical system architecture



- CAN
- FlexRay
- Ethernet
- etc.

Synchronous Communication

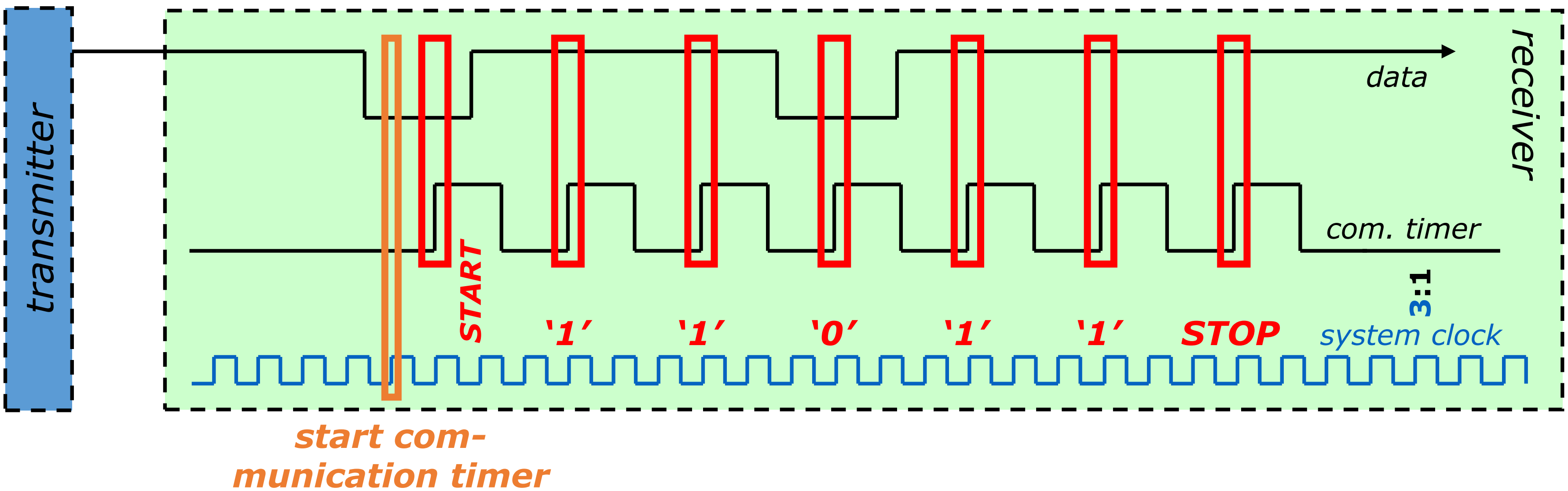
- sampling time, synchronisation between transmitter and receiver during communication and time between two transmissions are defined
- usually a common clock or self-synchronising code is used for synchronisation (→ master/slave relation necessary)



- advantage: no resynchronisation necessary → higher data rate

Asynchronous Communication

- no common time base → information about sampling times and necessities for receiver (e.g. baud rate, ratio between communication and system clock, …), any time between two transmissions
- synchronisation necessary to detect beginning of a transmission (e.g. dedicated handshake wire(s), start bit, …)

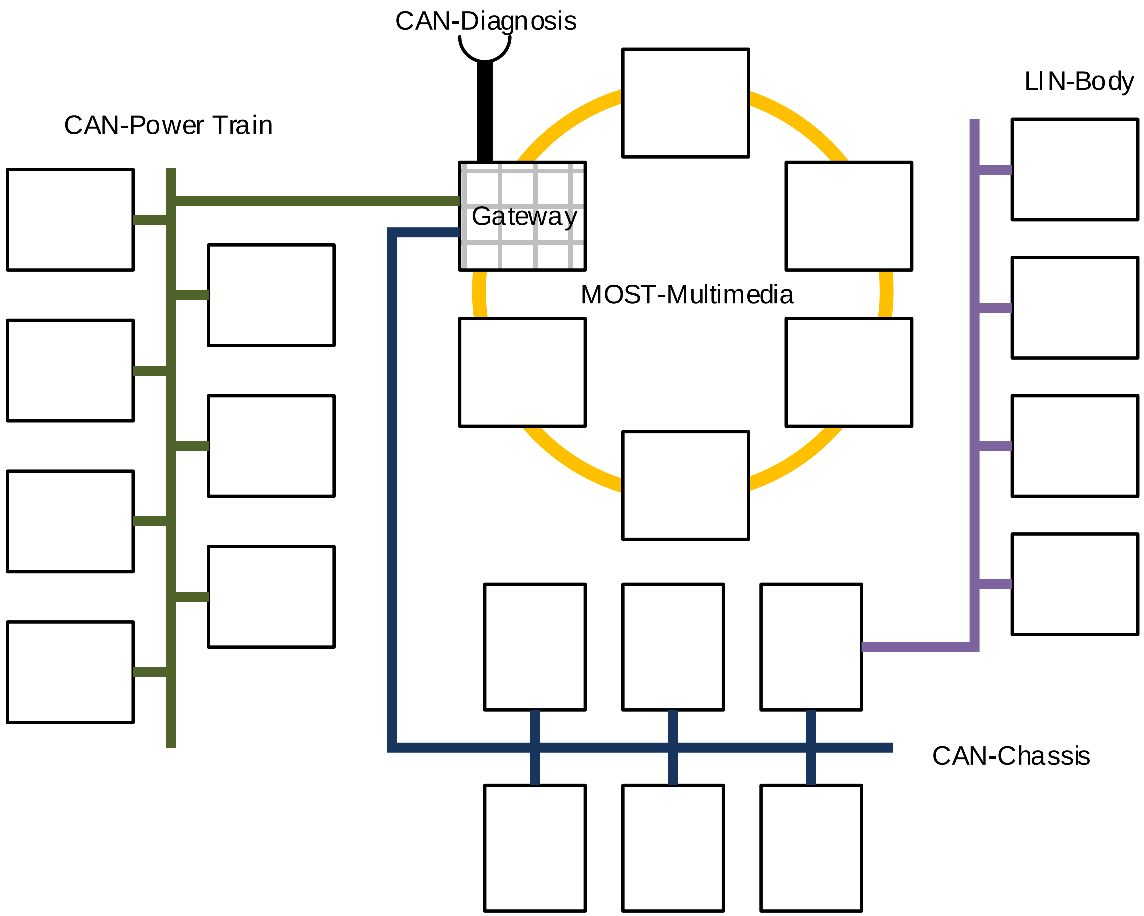


- advantage: no master/slave differentiation necessary

Automotive Networking

	Class A	Class B	Class C	Class C+	Class D
Data rate	<10 Kbit/s	<125 Kbit/s	<1 Mbit/s	<10 MBit/s	>10 Mbit/s
App-lication	Sensor-Actuator-Networking	Networking in the comfort area	Networking in the drive and chassis	Networking in the drive and chassis (X-By-Wire)	Networking in telematics and multimedia
Example	LIN	CAN	CAN	Flexray, TT-CAN	MOST

Typical Car Network



Contents

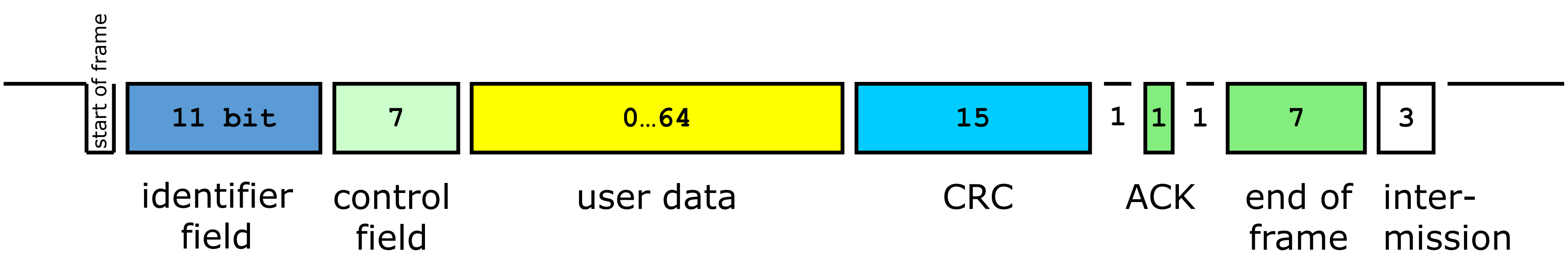
- Basics
- Controller Area Network (CAN)
- Technical Backgrounds
- Practical Labs

Controller Area Network (CAN)

- ISO 11898 standardised fieldbus (interconnection of sensors, actors and controller units) for asynchronous, serial communication, realtime conditions feasible
- released by Bosch and Intel in 1987, aim: reduce number of wiring harnesses in automotive domain
- high speed (1 MBit/s) and low speed/ fault tolerant (125 kBit/s) modes available
- theoretically unlimited number of bus nodes possible, up to 100 with common interface units
- message oriented broadcast bus → no receiver address in messages, all nodes can “hear” all transmissions

Functional Definition (I)

- CAN specification covers data link and physical layer
- typically realised as a line topology, star and ring topologies possible (with restrictions)
- four types of message frames defined
  - data frame: containing up to 8 byte of user data
  - remote frame: requesting the transmission of specific user data
  - error frame: transmitted by any node detecting an error
  - overload frame: inject a delay between data and/or remote frames
- e.g. data frame (CAN 2.0A)



Functional Definition (II)

- identifier field marks the content of a message (not the address of transmitter or receiver)
  - e.g. temperature, voltage, commands for actors, …
  - sensor marking possible (e.g. by marking the content with the ID: “temperature of sensor 1”, …)

- every bus node reads the message and “decides” if the content of the message is relevant for itself

- two identifier field formats defined
  - base frame format: 11 bit (CAN 2.0A)
  - extended base frame format: 29 bit (CAN 2.0B)
  - base frame format has to be accepted, extended base frame format can be accepted but has to be tolerated by every bus node

Functional Definition (III)

- problem
  - one serial wire, many nodes

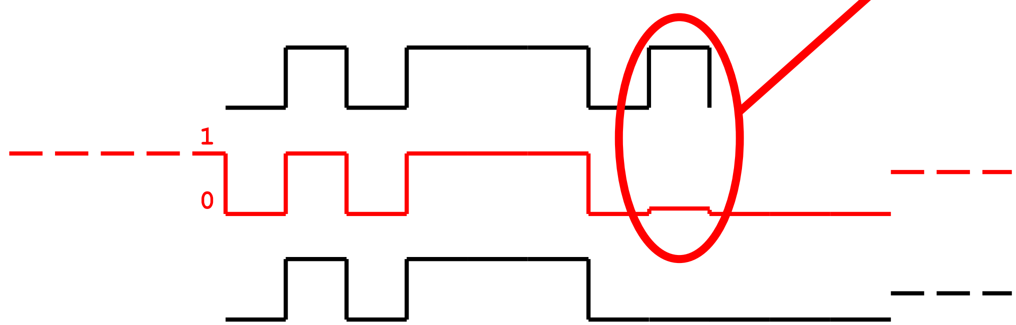
  → how to prevent conflicts in accessing the bus?
- solution
  - CAN uses bitwise arbitration based on the identifier fields
  - requires a transmission medium which allows a hard (dominant) and a soft (recessive) bus state
- example (2 messages at same time)
  (bus with recessive state ‘1’ and dominant state ‘0’)

bus state != message state
→ sender of message1 stops arbitration, message2 wins

$ID_{message1}$: 01011101000

resulting bus state

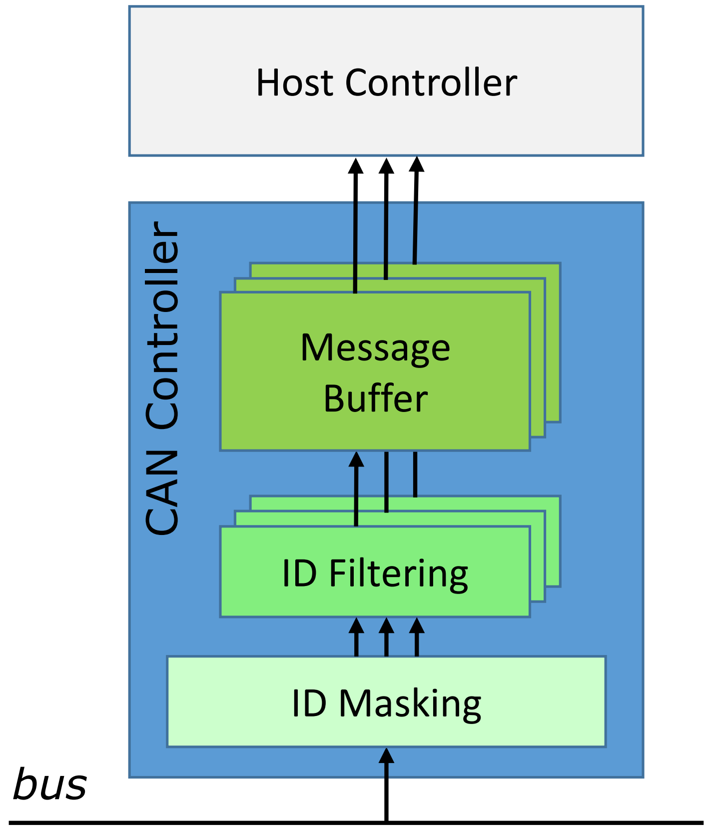
$ID_{message2}$: 01011100000



→ message prioritisation by identifier assignment possible

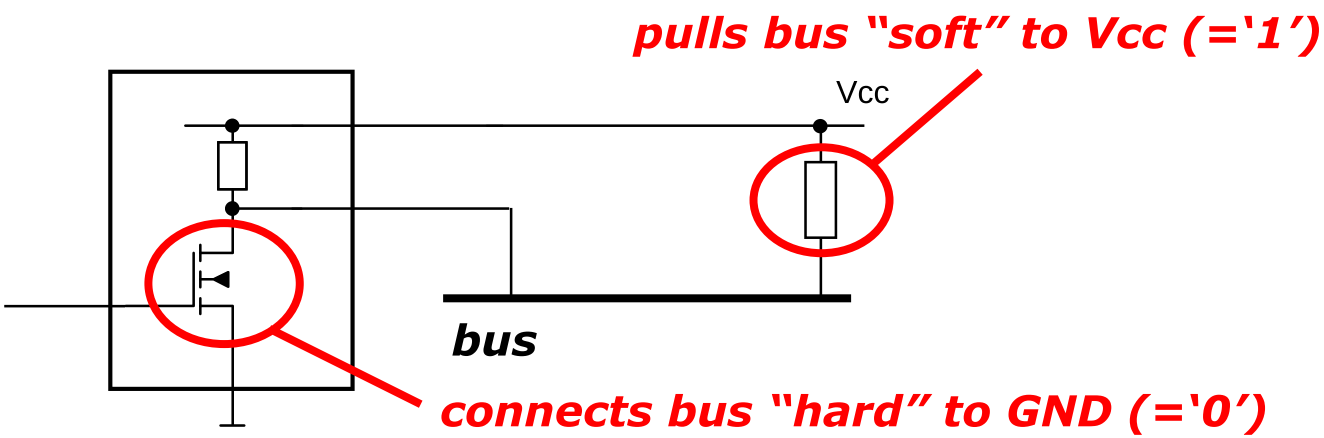
Functional Definition (IV)

- all nodes listen to all messages
  → masking and filtering to prevent overload of host controller



Electrical Definition -I

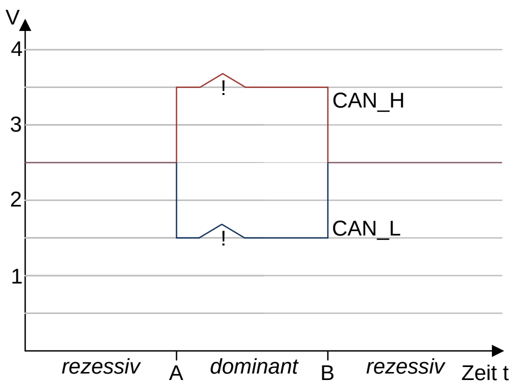
- CAN is not limited to a single physical layer → many different specifications based on electrical and optical mediums specified
- important is the support of a dominant and a recessive bus state, e.g. by pull-up/down nets

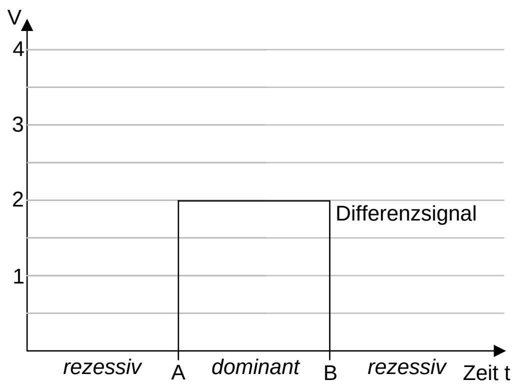


- widespread physical layers are
  - RS-485
  - ISO 11898-2:2003 – High-Speed medium access unit

Electrical Definition -II

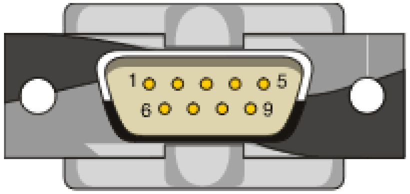
- Example: Voltage levels of the Highspeed CAN and resulting difference signal





Mechanical Definition

- several mechanical definitions exist - depending on the used physical layer and mediums
- e.g. ISO 11898/ CAN in Automation (CiA) DS102-1
  - usage of a D-SUB9 connector



1… not connected
2… CAN_L
3… CAN_GND
4… not connected
5… not connected
6… CAN_GND
7… CAN_H
8… not connected
9… not connected

  - cable length recommendation

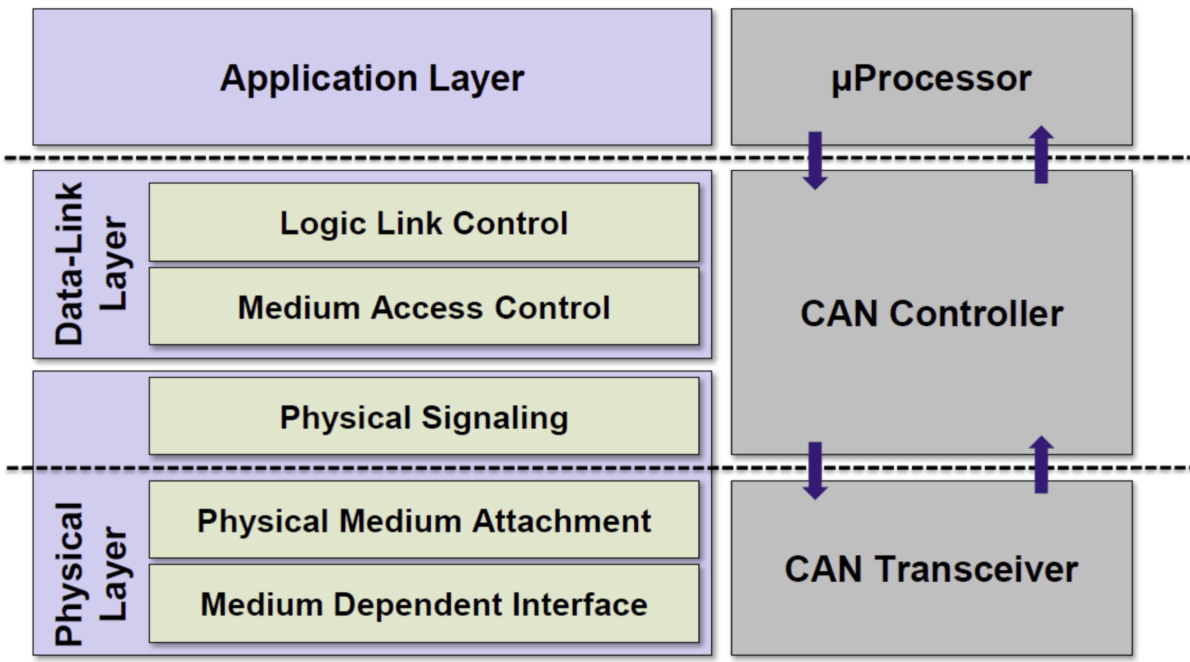
baudrate	bit time	cable length	baudrate	bit time	cable length
1000 kBit/s	1 µs	40 m	100 kBit/s	10 µs	400 m
500 kBit/s	2 µs	80 m	50 kBit/s	20 µs	800 m
250 kBit/s	4 µs	160 m	20 kBit/s	50 µs	2000 m
125 kBit/s	8 µs	320 m	10 kBit/s	100 µs	4000 m

Contents

- Basics
- Controller Area Network (CAN)
- Technical Backgrounds
- Practical Labs

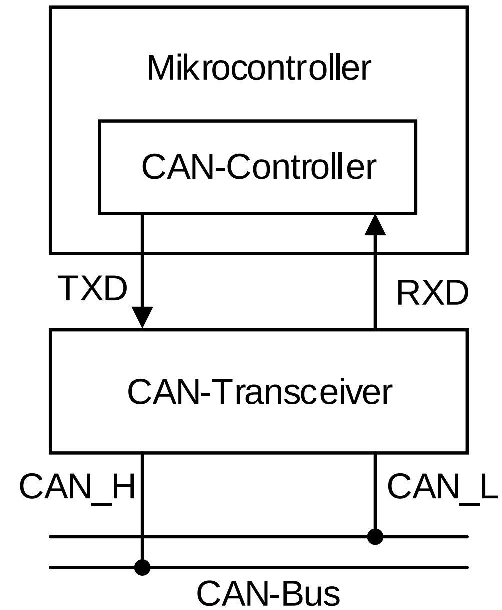
Stack Levels and Implementation

Common Concept



Source: Introduction to the Controller Area Network (CAN), Steve Corrigan, Application Report, 2008, TI

ST Power Architecture
(SPC560P50L5)



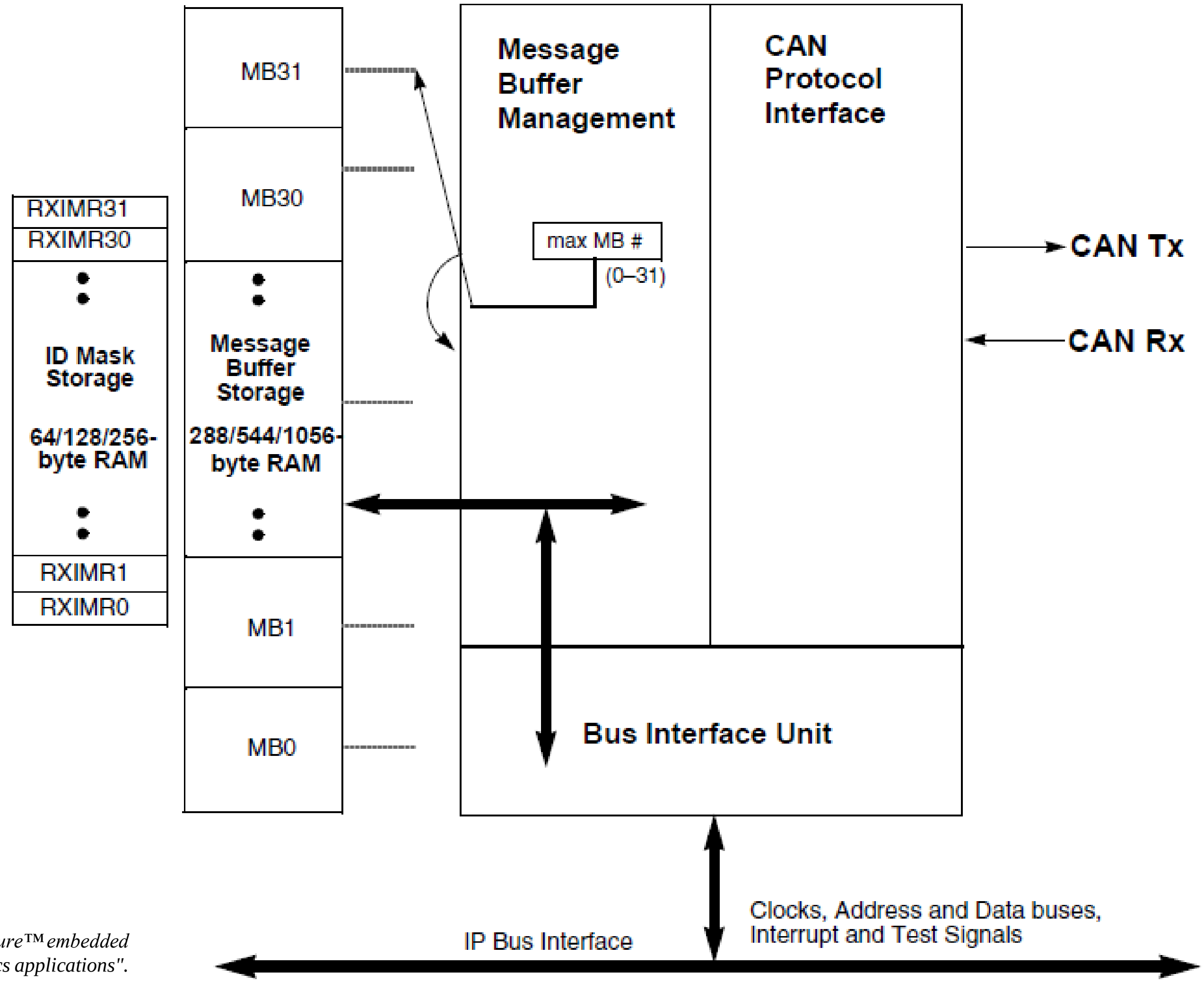
Communication Core / Controller

- (virtual) registers for exchange between core and CAN controller
    - controller states
    - controller modes
    - interrupts
    - filter
    - buffer
    - errors

Offset from FlexCAN_BASE (0xFFFC_0000)	Register	Access	Reset value
0x0000	Module Configuration Register (MCR)	R/W	0xUUU0_0000
0x0004	Control Register (CTRL)	R/W	0x0000_0000
0x0008	Free Running Timer (TIMER)	R/W	0x0000_0000
0x000C	Reserved		
0x0010	Rx Global Mask (RXGMASK)	R/W	0x0000_0000
0x0014	Rx Buffer 14 Mask (RX14MASK)	R/W	0x0000_0000
0x0018	Rx Buffer 15 Mask (RX15MASK)	R/W	0x0000_0000
0x001C	Error Counter Register (ECR)	R/W	0x0000_0000
0x0020	Error and Status Register (ESR)	R/W	0x0000_0000
0x0024	Reserved		
0x0028	Interrupt Masks 1 (IMASK1)	R/W	0x0000_0000
0x002C	Reserved		
0x0030	Interrupt Flags 1 (IFLAG1)	R/W	0x0000_0000
0x0034–0x005F	Reserved		
0x0060–0x007F	Serial Message Buffers (SMB0–SMB1) – Reserved	x[1]	U
0x0080–0x017F	Message Buffers MB0–MB15	R/W	U[2]
0x0180–0x027F	Message Buffers MB16–MB31	R/W	U[2]
0x0280–0x087F	Reserved		
0x0880–0x08BF	Rx Individual Mask Registers RXIMR0–RXIMR15	R/W	0x0000_0000
0x08C0–0x08FF	Rx Individual Mask Registers RXIMR16–RXIMR31	R/W	0x0000_0000

Source: "32-bit MCU family built on the Power Architecture™ embedded category for automotive chassis and safety electronics applications". ST Microelectronics, RM0022, Doc ID 14891, Rev. 3.

Buffer Management



Source: "32-bit MCU family built on the Power Architecture™ embedded category for automotive chassis and safety electronics applications". ST Microelectronics, RM0022, Doc ID 14891, Rev. 3.

Buffer

- used for TX (sending) and RX (receiving) setup

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>0x0</td><td colspan="4"></td><td colspan="4">CODE</td><td></td><td>SRR</td><td>IDE</td><td>RTR</td><td colspan="4">LENGTH</td></tr>
<tr><td>0x4</td><td colspan="3">PRIO</td><td colspan="11">ID (Standard/Extended)</td><td colspan="2">ID (Extended) [17:16]</td></tr>
<tr><td>0x8</td><td colspan="8">Data Byte 0</td><td colspan="8">Data Byte 1</td></tr>
<tr><td>0xC</td><td colspan="8">Data Byte 4</td><td colspan="8">Data Byte 5</td></tr>
</table>

<table>
<tr><th></th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0x0</td><td colspan="16">TIME STAMP</td></tr>
<tr><td>0x4</td><td colspan="16">ID (Extended) [15:0]</td></tr>
<tr><td>0x8</td><td colspan="8">Data Byte 2</td><td colspan="8">Data Byte 3</td></tr>
<tr><td>0xC</td><td colspan="8">Data Byte 6</td><td colspan="8">Data Byte 7</td></tr>
</table>

Source: "32-bit MCU family built on the Power Architecture™ embedded category for automotive chassis and safety electronics applications". ST Microelectronics, RM0022, Doc ID 14891, Rev. 3.

Contents

- Basics
- Controller Area Network (CAN)
- Technical Backgrounds
- Practical Labs

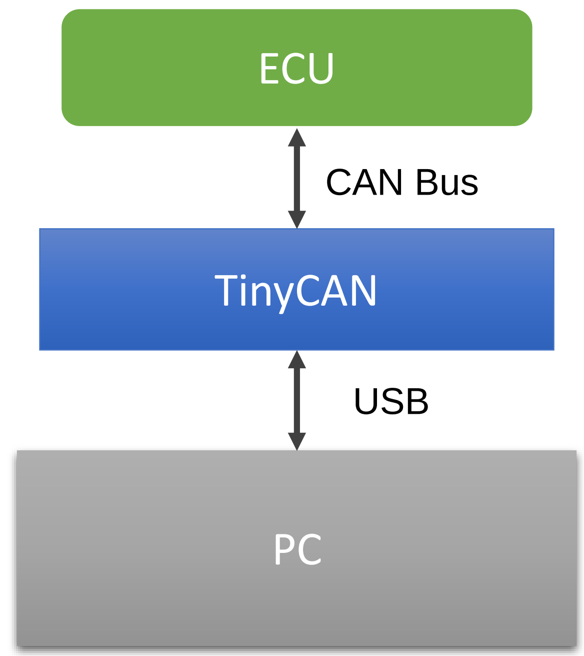
Practical Lab 2

- setup timer
    - interrupt setup
    - interrupt management

- setup pins for CAN transmission (**`RXD, TXD`**)

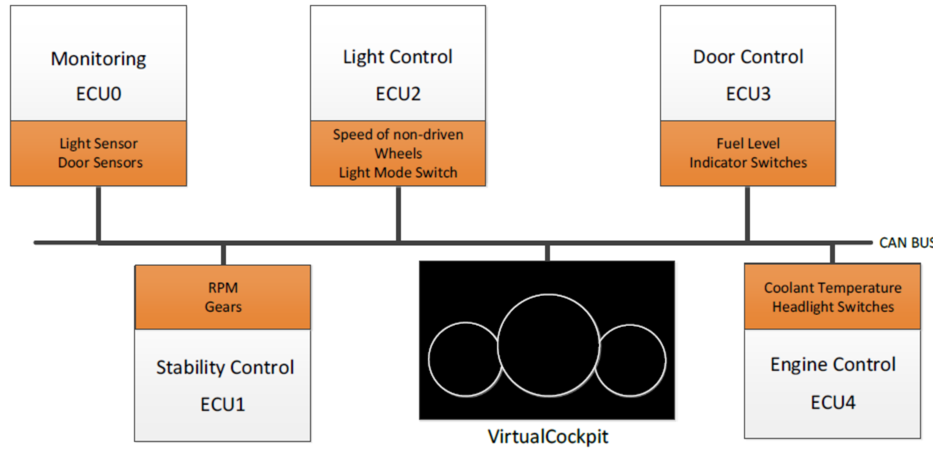
- setup sending buffer (**`CAN0.BUF[4]`**)
- send messages periodically

- setup receiving buffer (**`CAN0.BUF[0]`**)
- setup ID masking and ID filtering



Practical Lab 3

- ECU network (CAN)



- team work
  - team = 10 persons,
    1 team leader
  - 5 workgroups per team
  - each workgroup
    responsible for 1 ECU
    with decicated functionality

- Working of ECUs similar to lab 1 and 2 but with a different API

- **important: elect a team leader for your group**

Masking Filter

- Masking Value - Specificaiton of bits to be checked against recieved message ID
- Acceptance Value – Specification of bit values to be recieved
- CAN Message Ids to be received
- Bit Correspondence
- Example:

	Bit 3 MSB	Bit 2	Bit 1	Bit 0 LSB
Receiving ID 1	0	1	0	1
Receiving ID 2	1	1	0	0
Masking Value	0	1	1	0
Acceptance Value	X	1	0	X

X = Dont care (0 or 1)

TinyCAN Interface