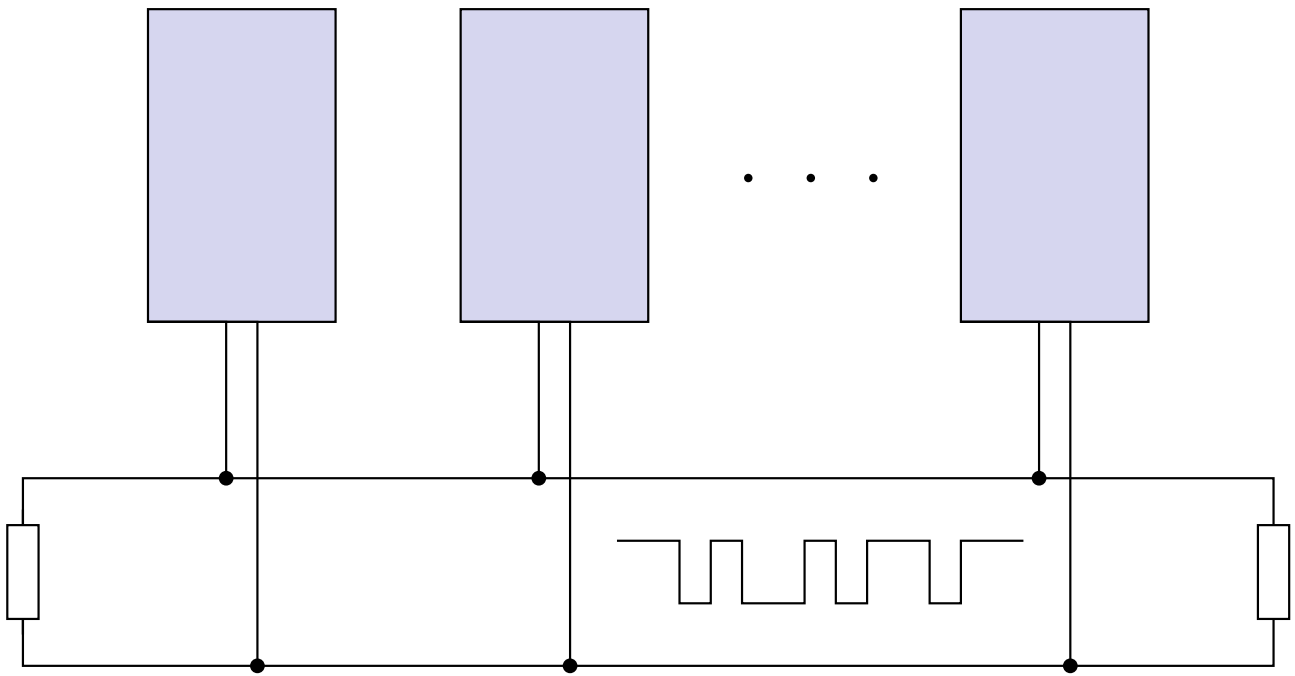
CAN und Linux im praktischen Einsatz

Linux Stammtisch
8. Juni 2012
Lutz Wirsig



Inhalt

Grundlagen

Gliederung

Grundlagen

CAN-Bus Entwicklungsziele, Verwendung

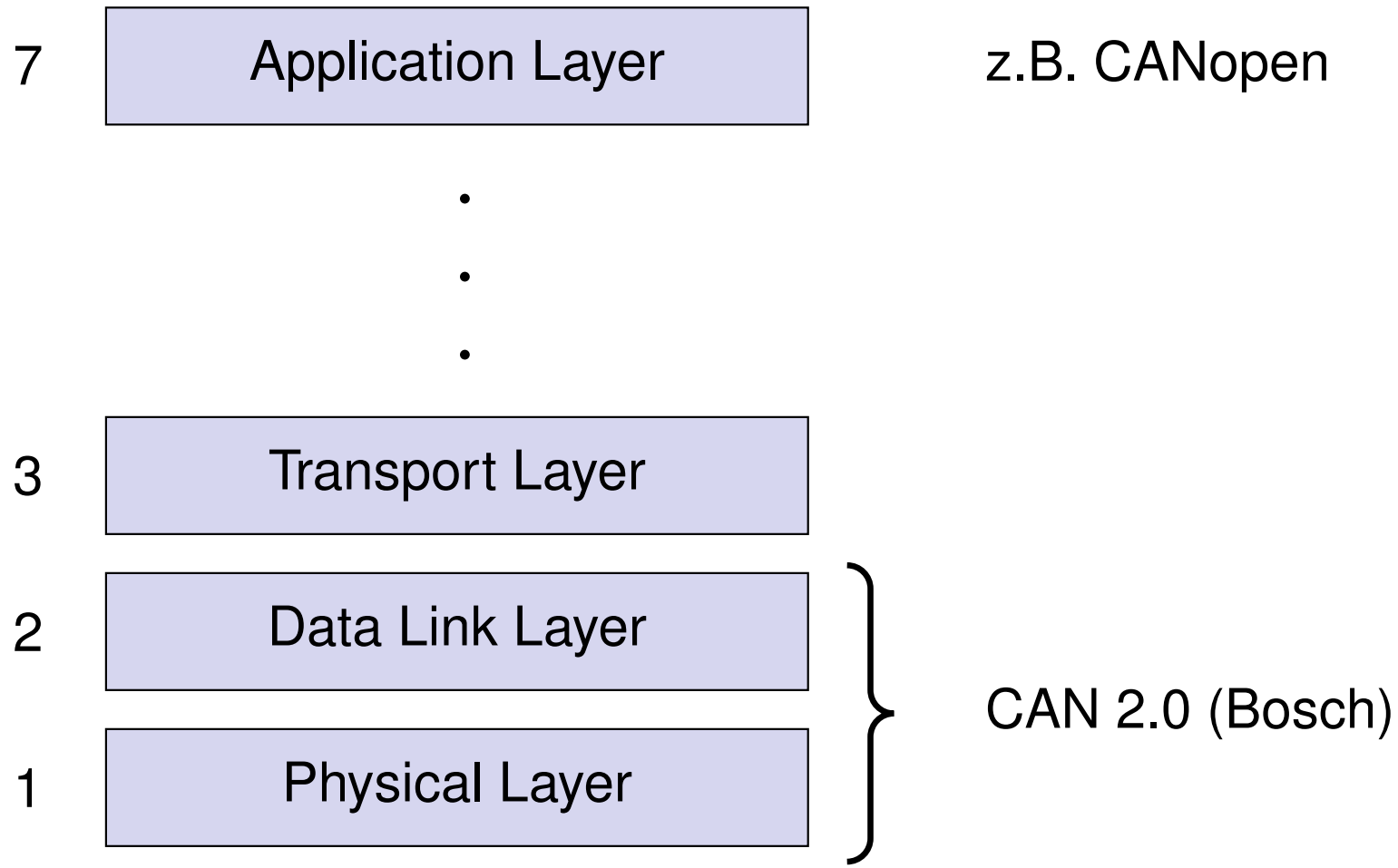
- einfache Verkabelung durch preiswerte verdrillte Zweidrahtleitung
- einfache nachträgliche Erweiterung des Netzwerkes
- kein Einfluß auf Netzwerk bei Ausfall eines Busteilnehmers
- Multi Master Netzwerksystem
- bidirektionale Datenrichtung
- Verwendung in Autombilindustrie, Luft-und Raumfahrt, Automatisierungstechnik und Medizintechnik

Grundlagen

Gliederung

Grundlagen

Einordnung in ISO 7-Schichtenmodell

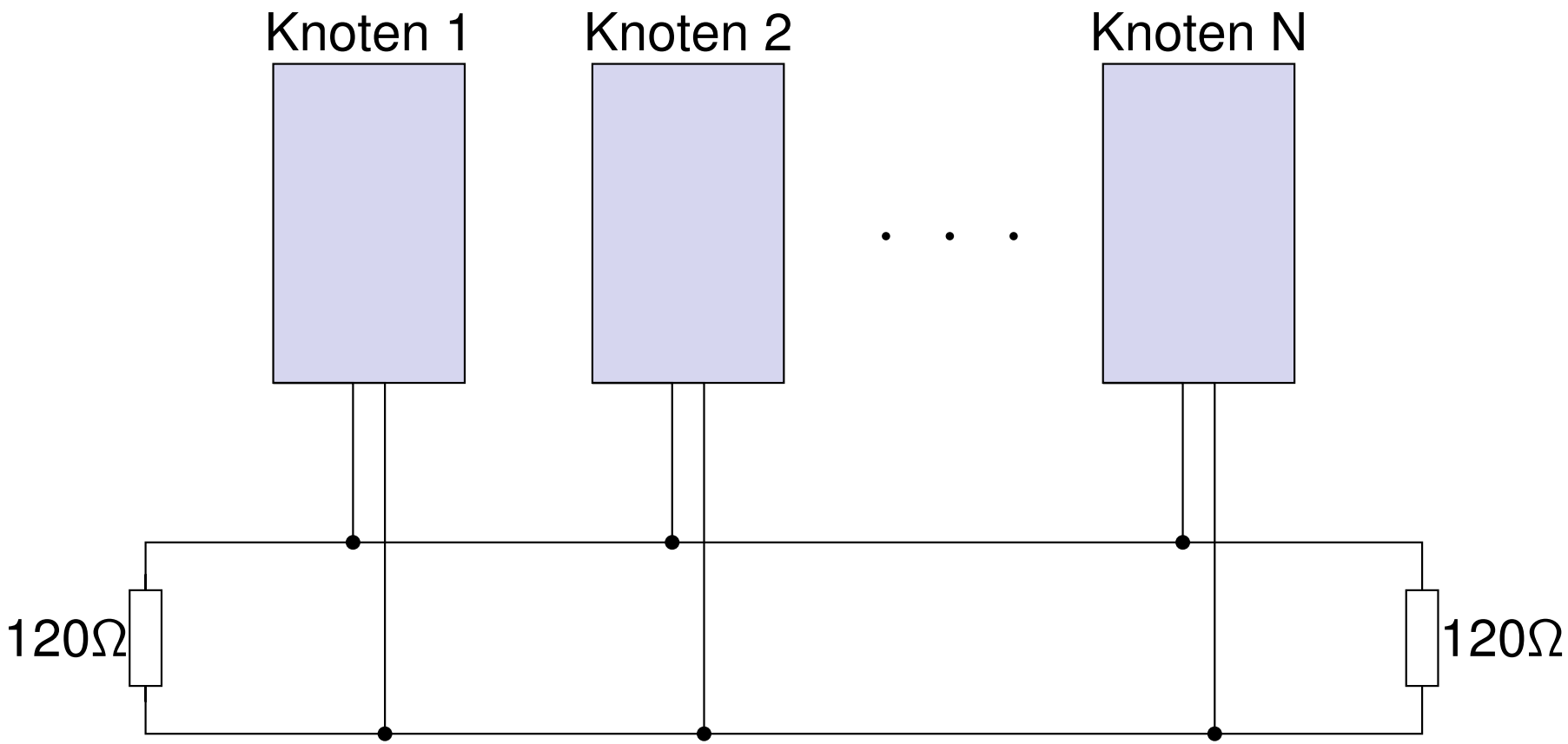


Grundlagen

Gliederung

Grundlagen

Bustopologie High Speed CAN

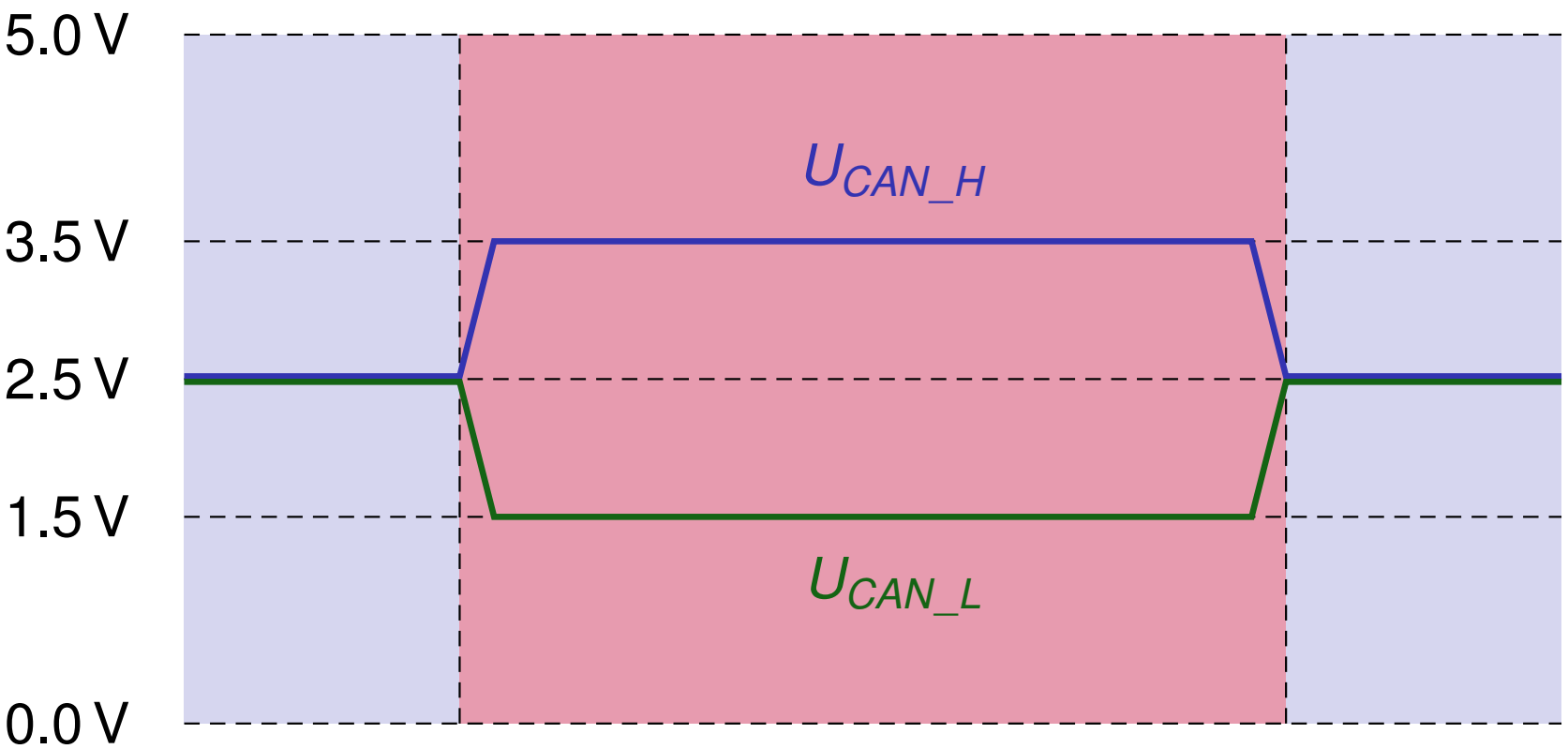


Grundlagen

Gliederung

Grundlagen

Signalpegel High Speed CAN (ISO-IS 11898)

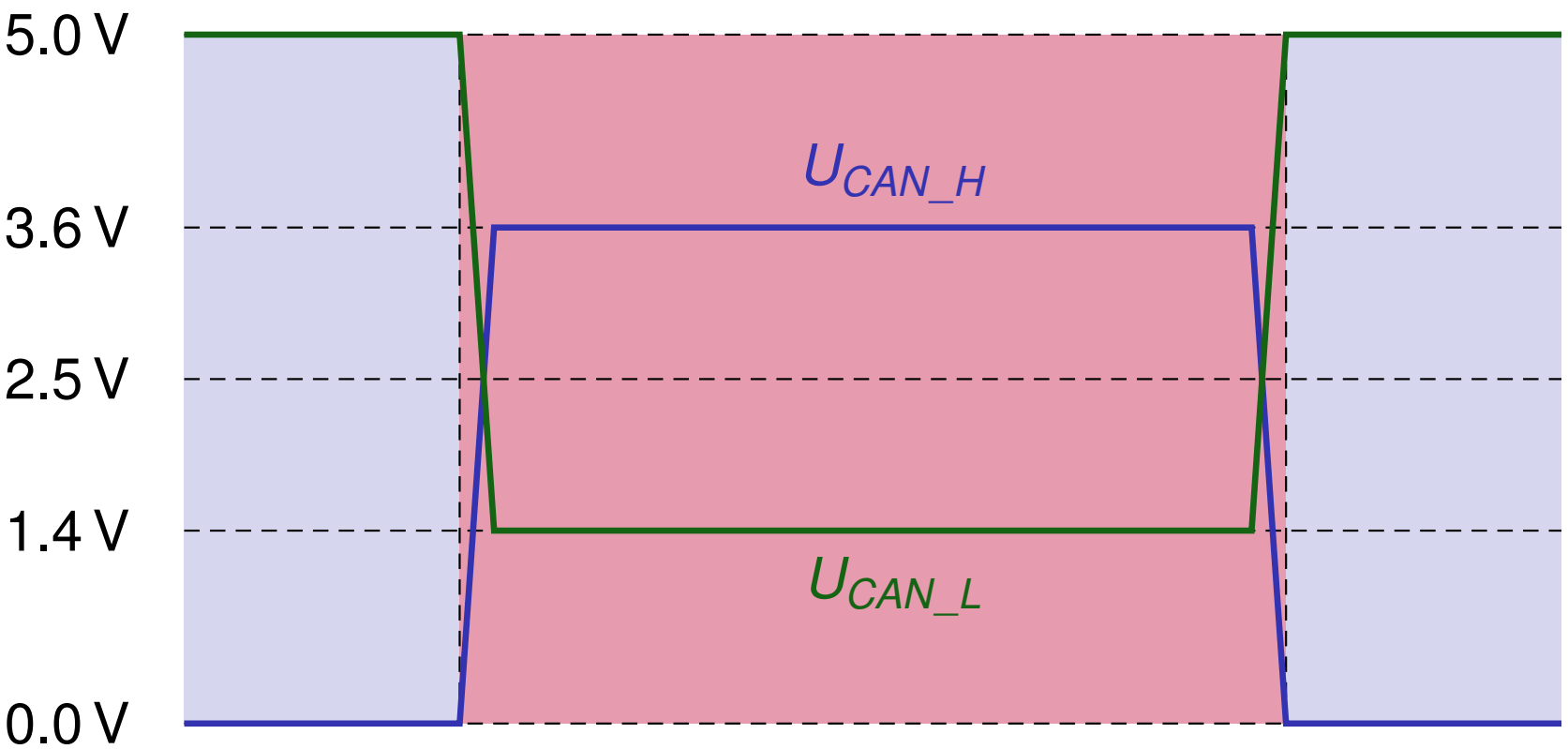


Rezessiv (binäre „1“)

Dominant (binäre „0“)

Grundlagen

Signalpegel Low Speed CAN (ISO-IS 11519)



Rezessiv (binäre „1“)

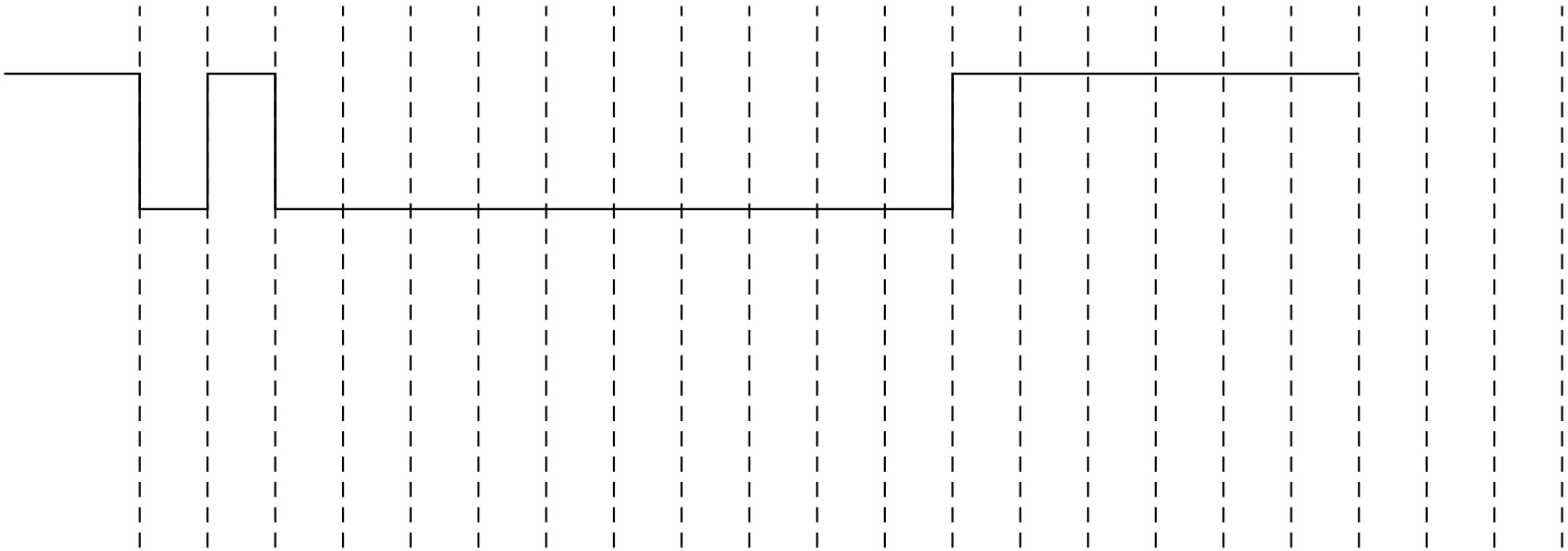
Dominant (binäre „0“)

Grundlagen

Gliederung

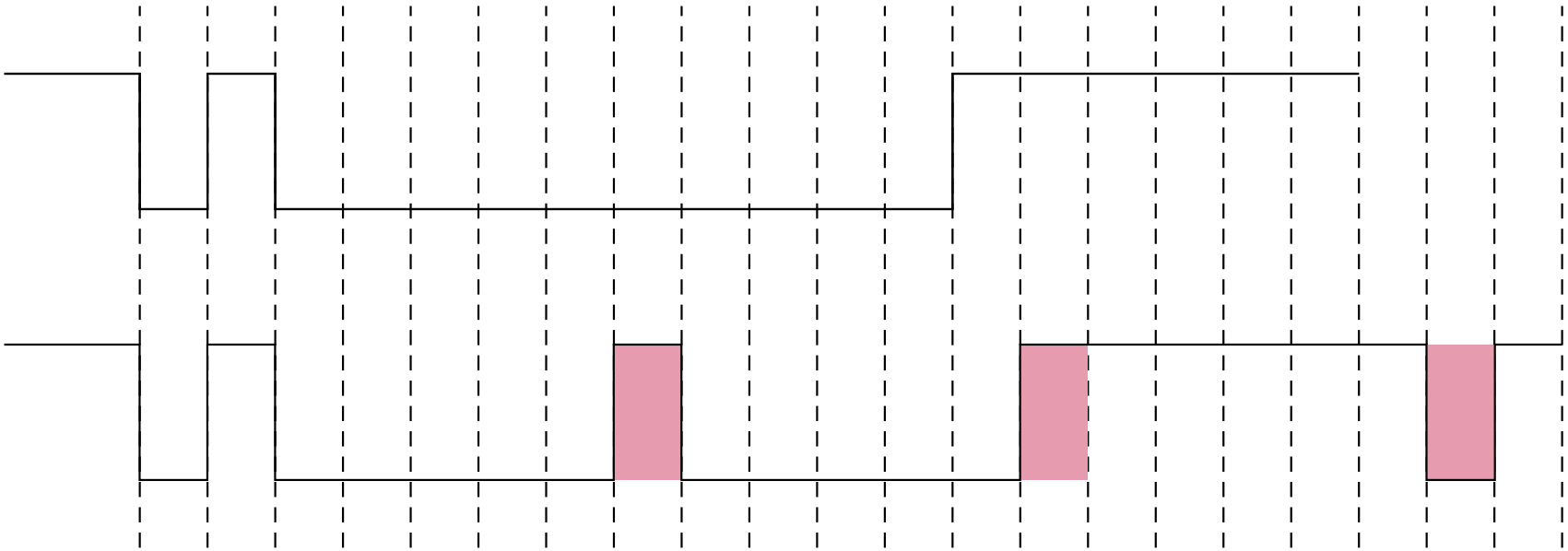
Grundlagen

Stuffbits



Grundlagen

Stuffbits



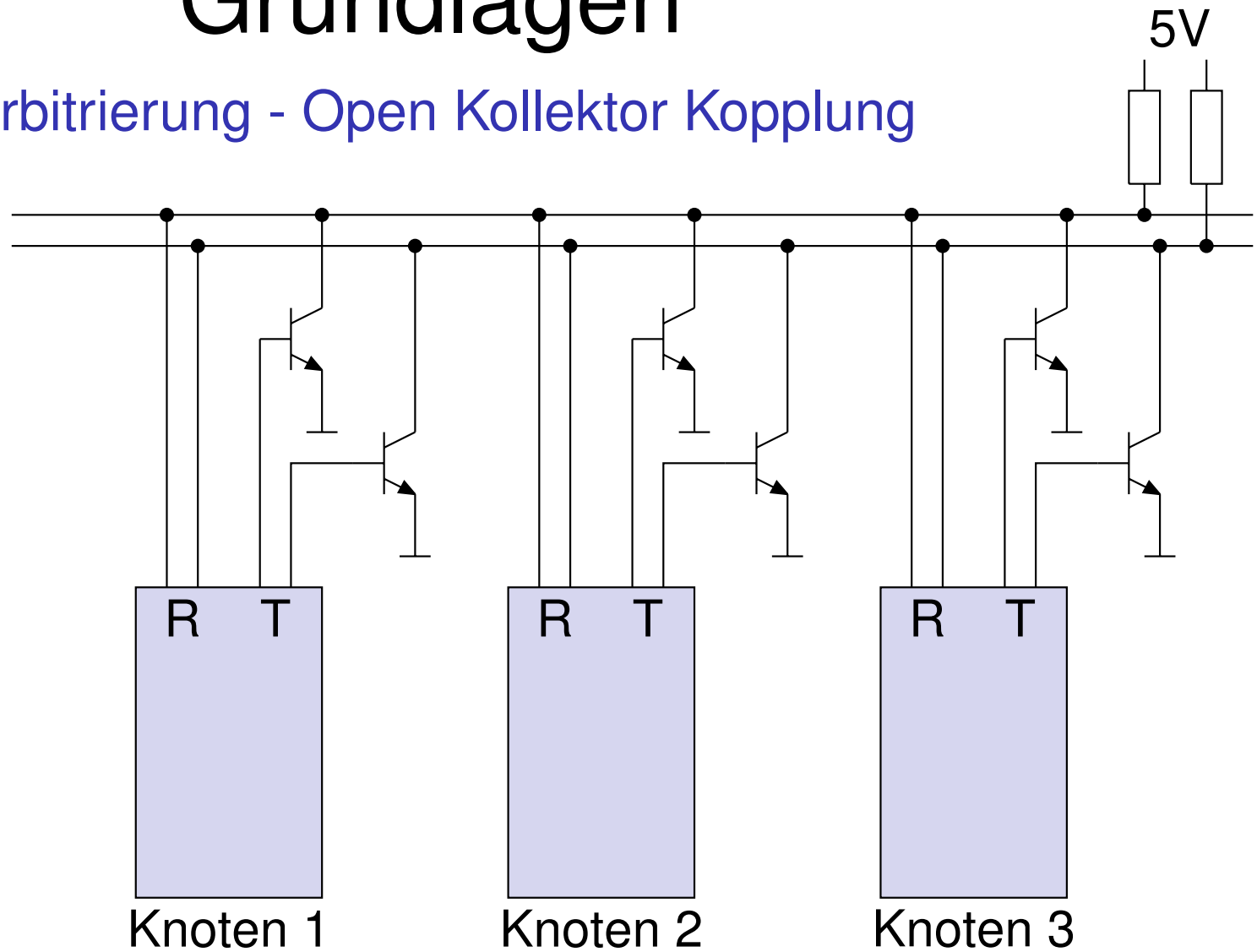
nach 5 aufeinanderfolgenden gleichwertigen Bits
wird ein anderwertiges Stuffbit eingeschoben

Grundlagen

Gliederung

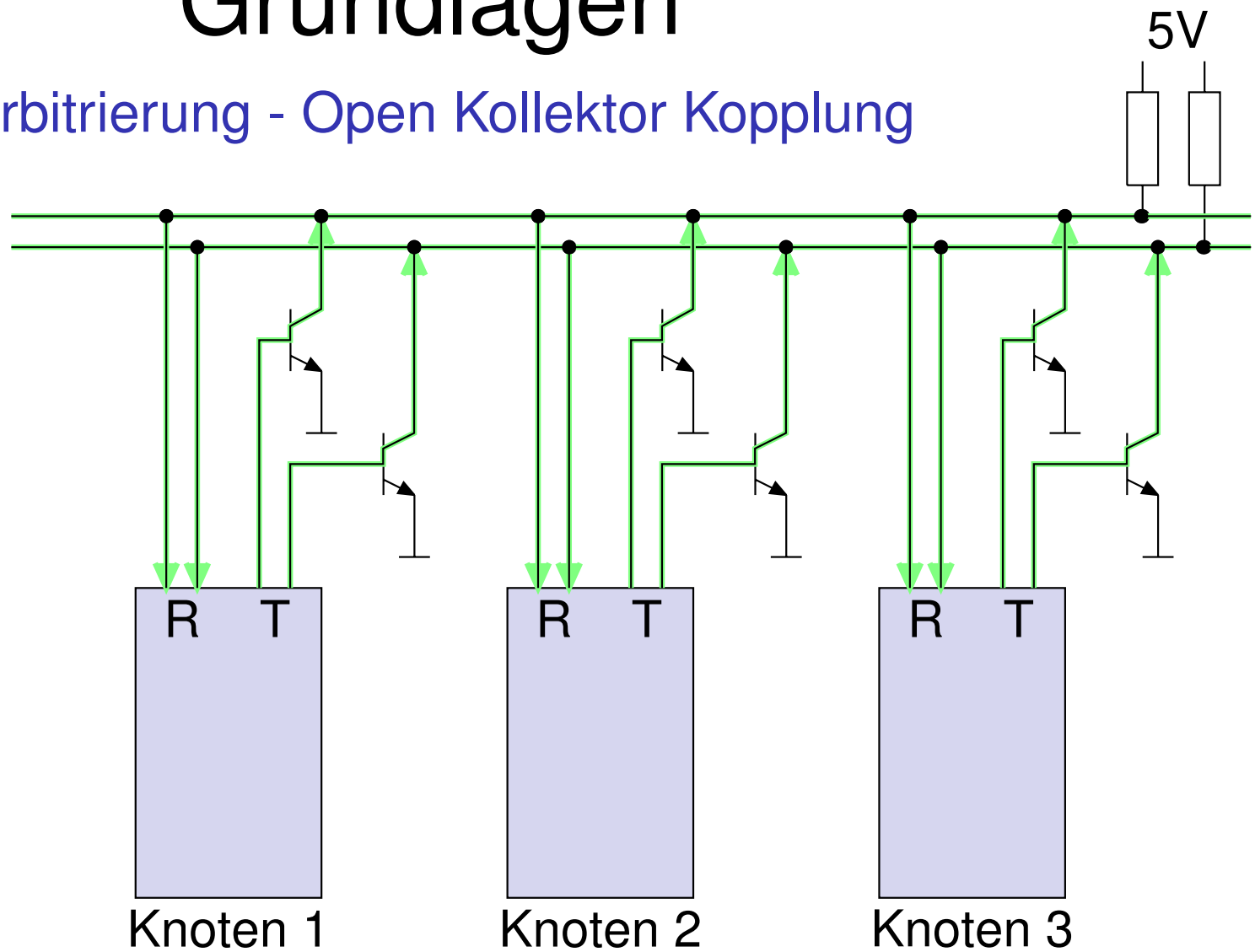
Grundlagen

Arbitrierung - Open Kollektor Kopplung



Grundlagen

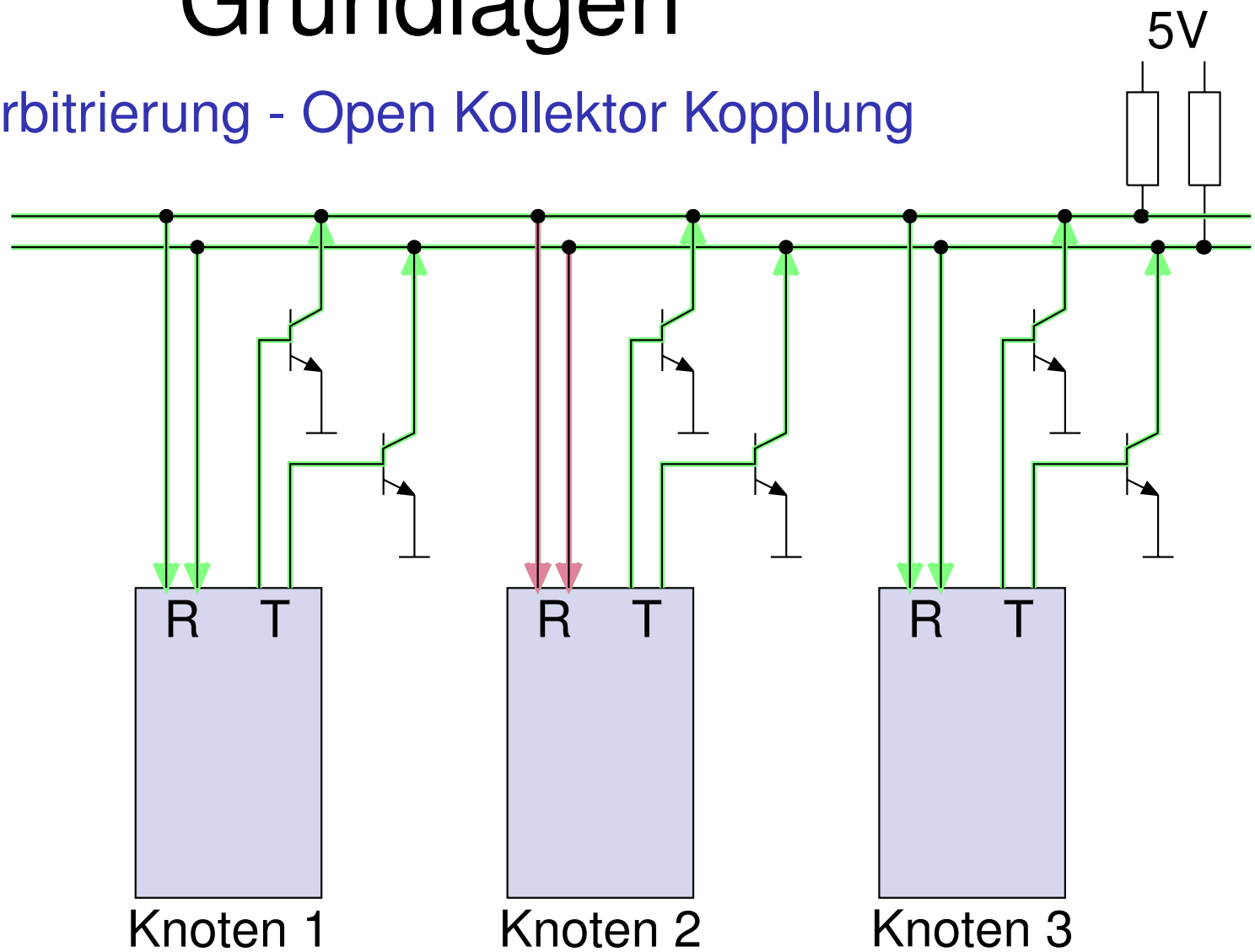
Arbitrierung - Open Kollektor Kopplung



Knoten sendet und liest Bus-Pegel ein

Grundlagen

Arbitrierung - Open Kollektor Kopplung

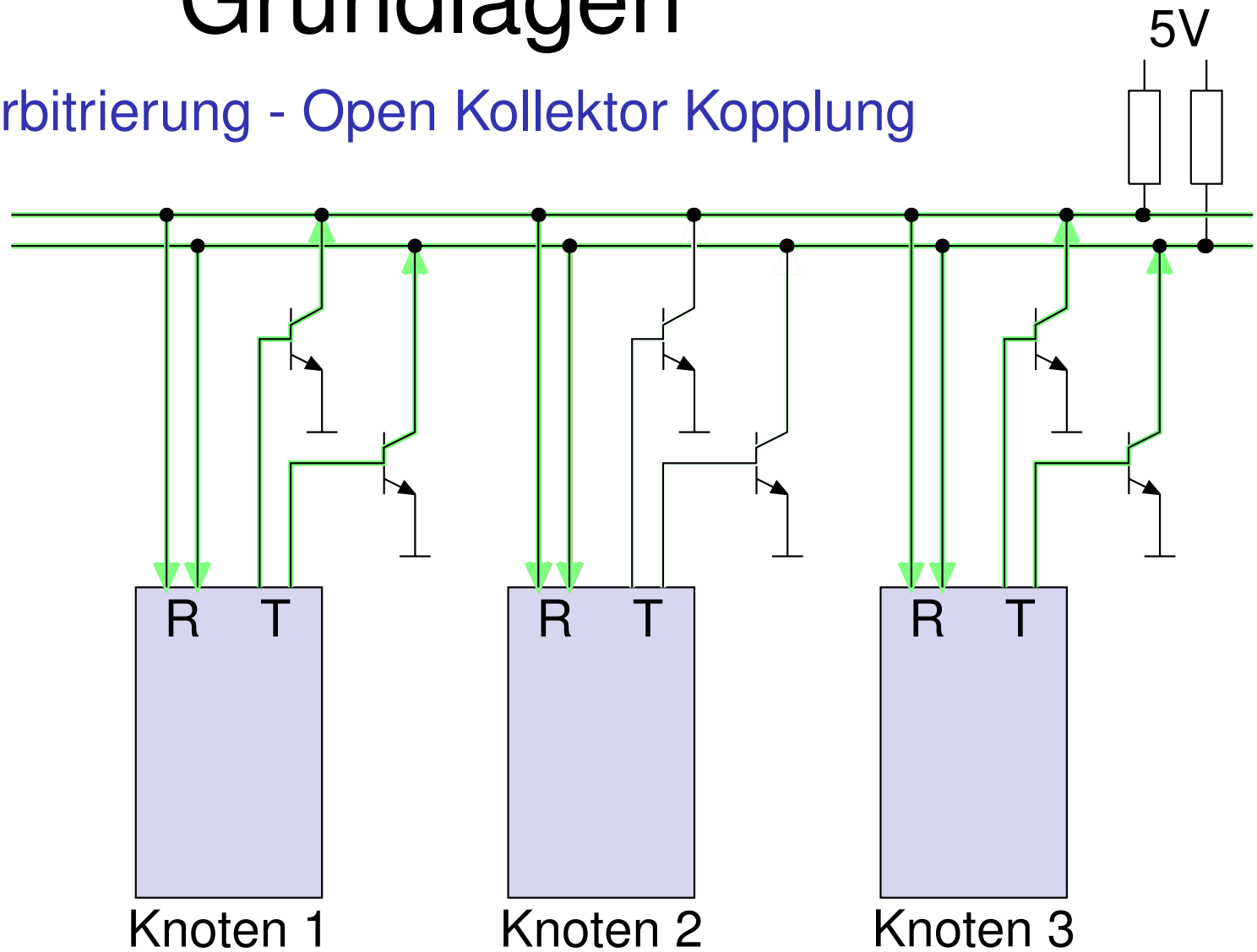


Knoten sendet und liest Bus-Pegel ein

Knoten liest dominaten Pegel, hat aber einen rezessiven gesendet

Grundlagen

Arbitrierung - Open Kollektor Kopplung

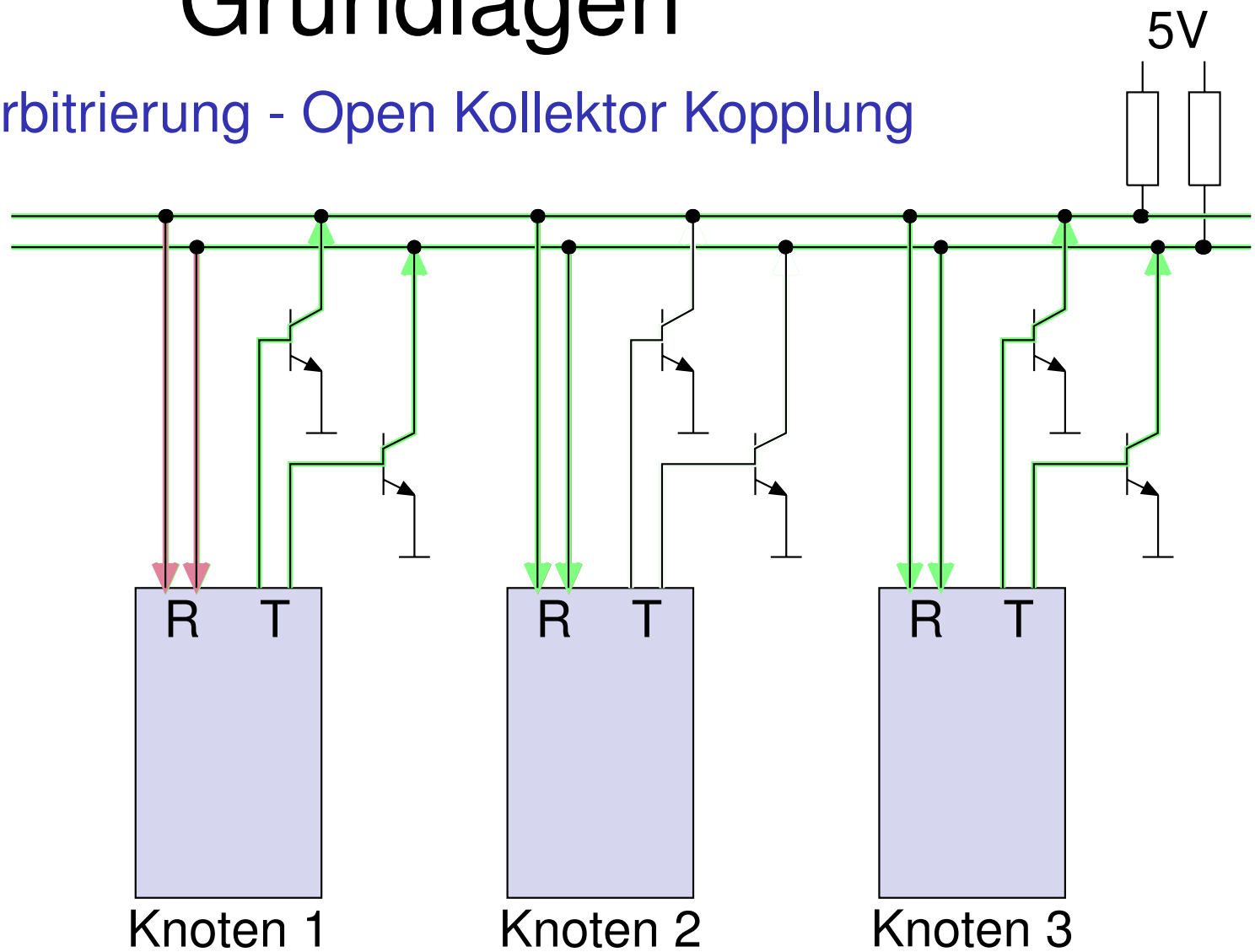


Knoten sendet und liest Bus-Pegel ein

Knoten liest dominaten Pegel, hat aber einen rezessiven gesendet

Grundlagen

Arbitrierung - Open Kollektor Kopplung

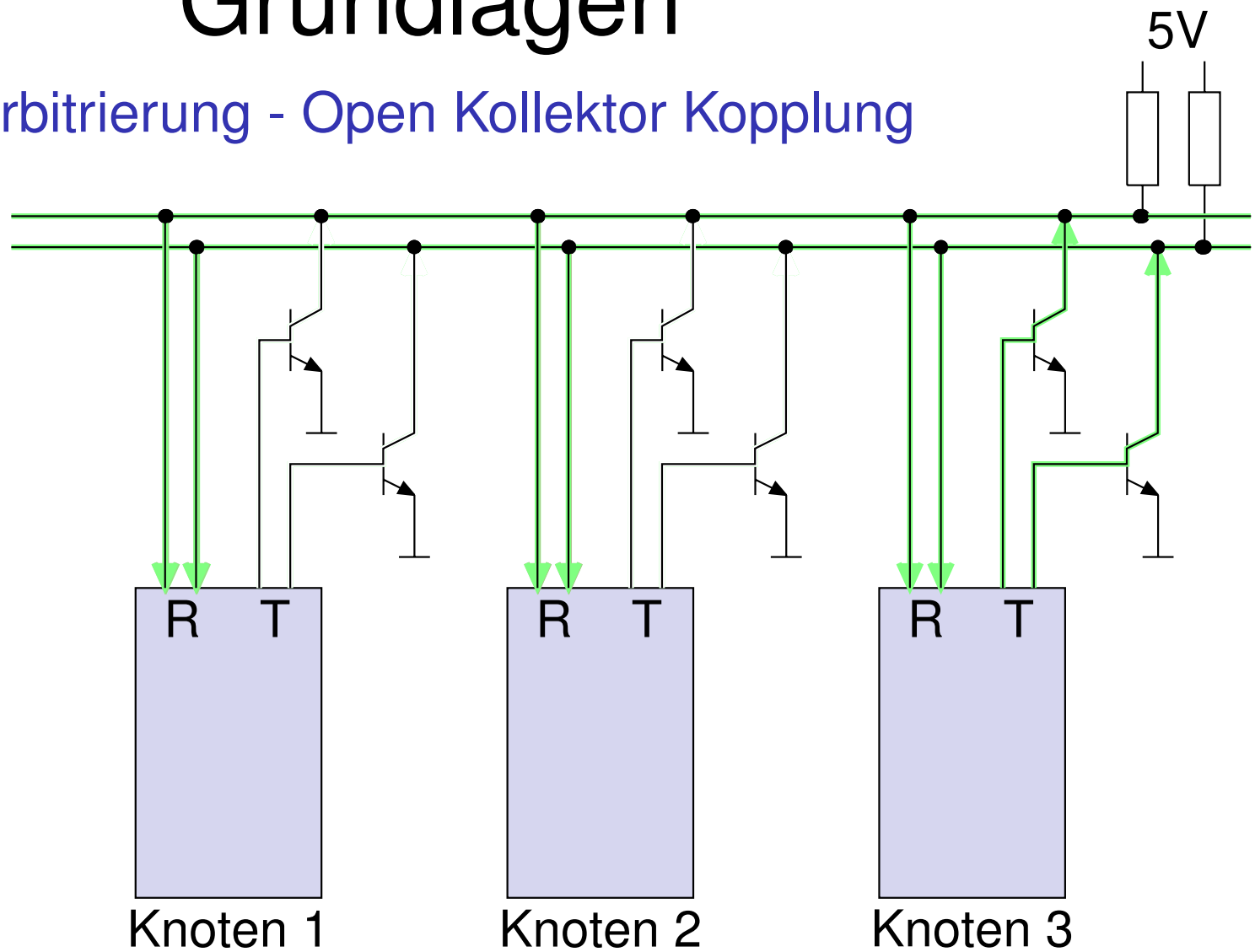


Knoten sendet und liest Bus-Pegel ein

Knoten liest dominaten Pegel, hat aber einen rezessiven gesendet

Grundlagen

Arbitrierung - Open Kollektor Kopplung

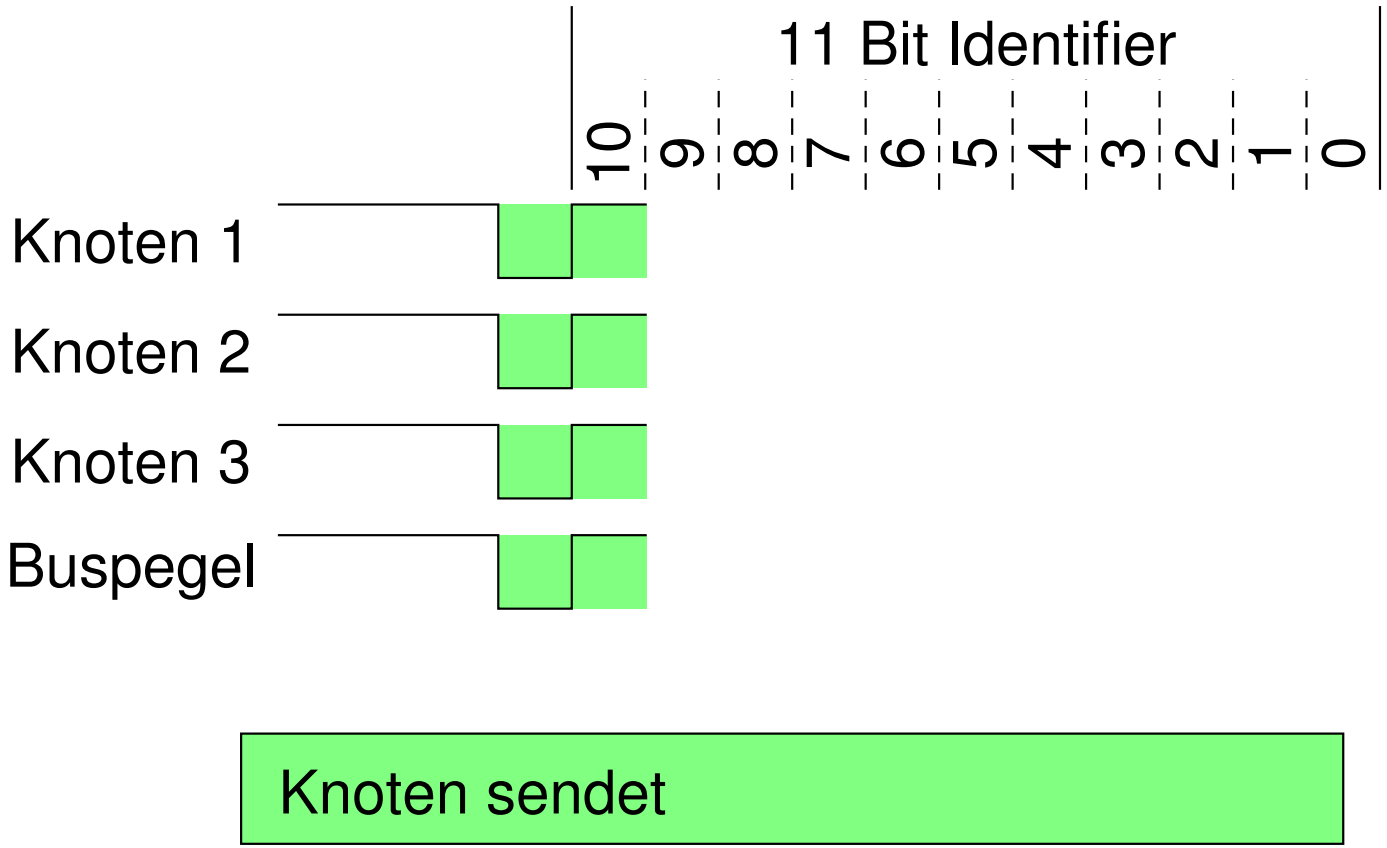


Knoten sendet und liest Bus-Pegel ein

Knoten liest dominaten Pegel, hat aber einen rezessiven gesendet

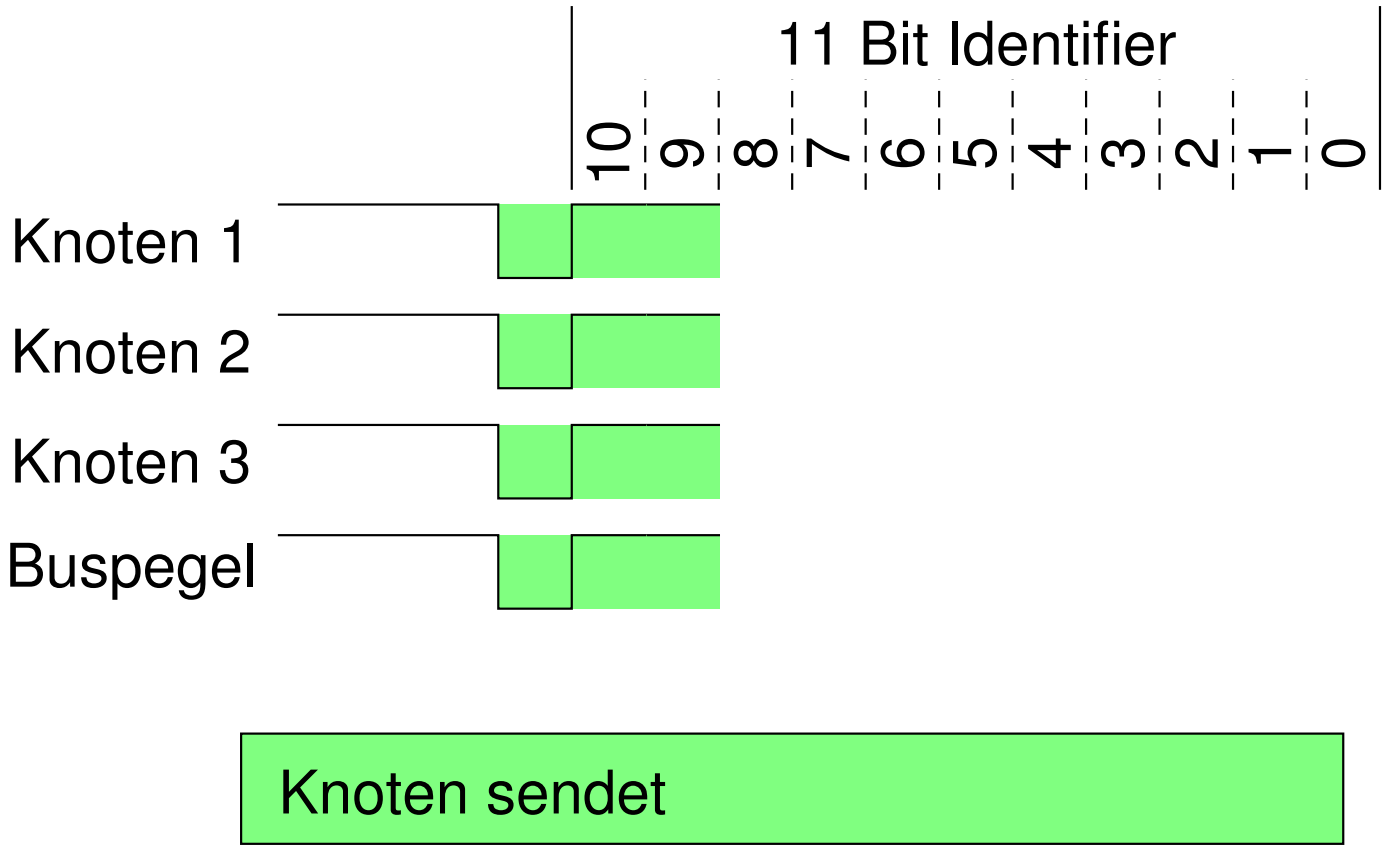
Grundlagen

Arbitrierung - Identifier Priorität



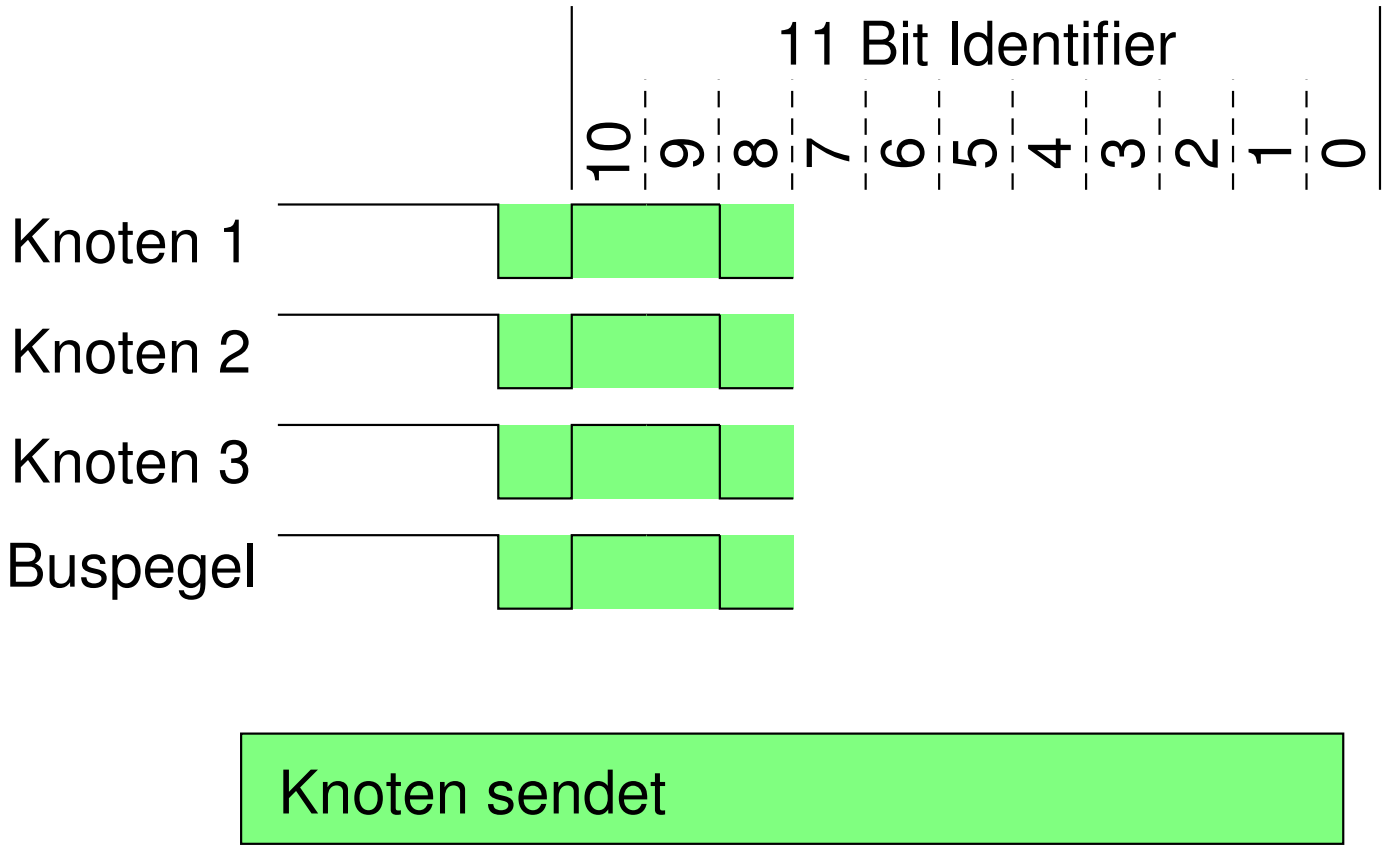
Grundlagen

Arbitrierung - Identifier Priorität



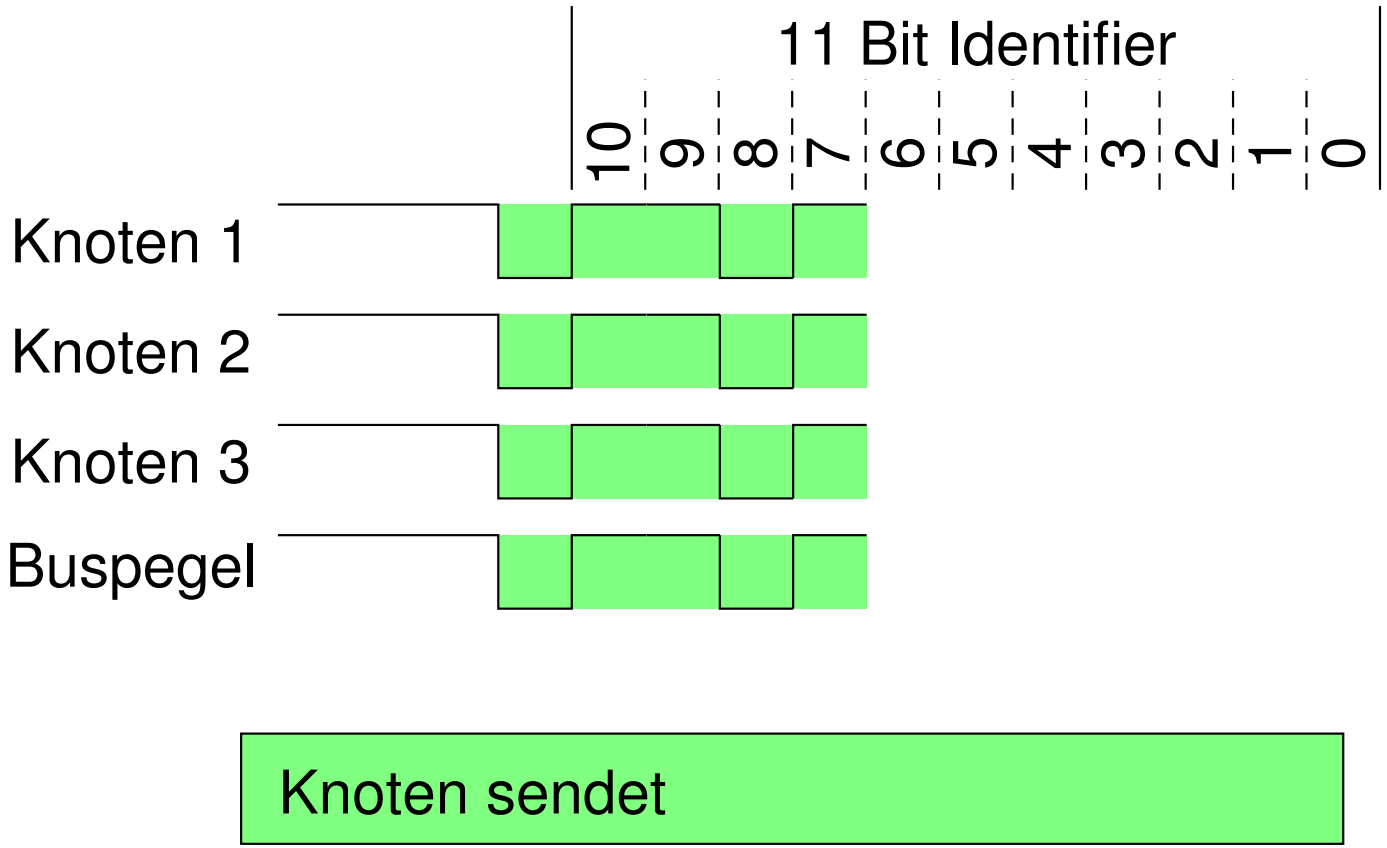
Grundlagen

Arbitrierung - Identifier Priorität



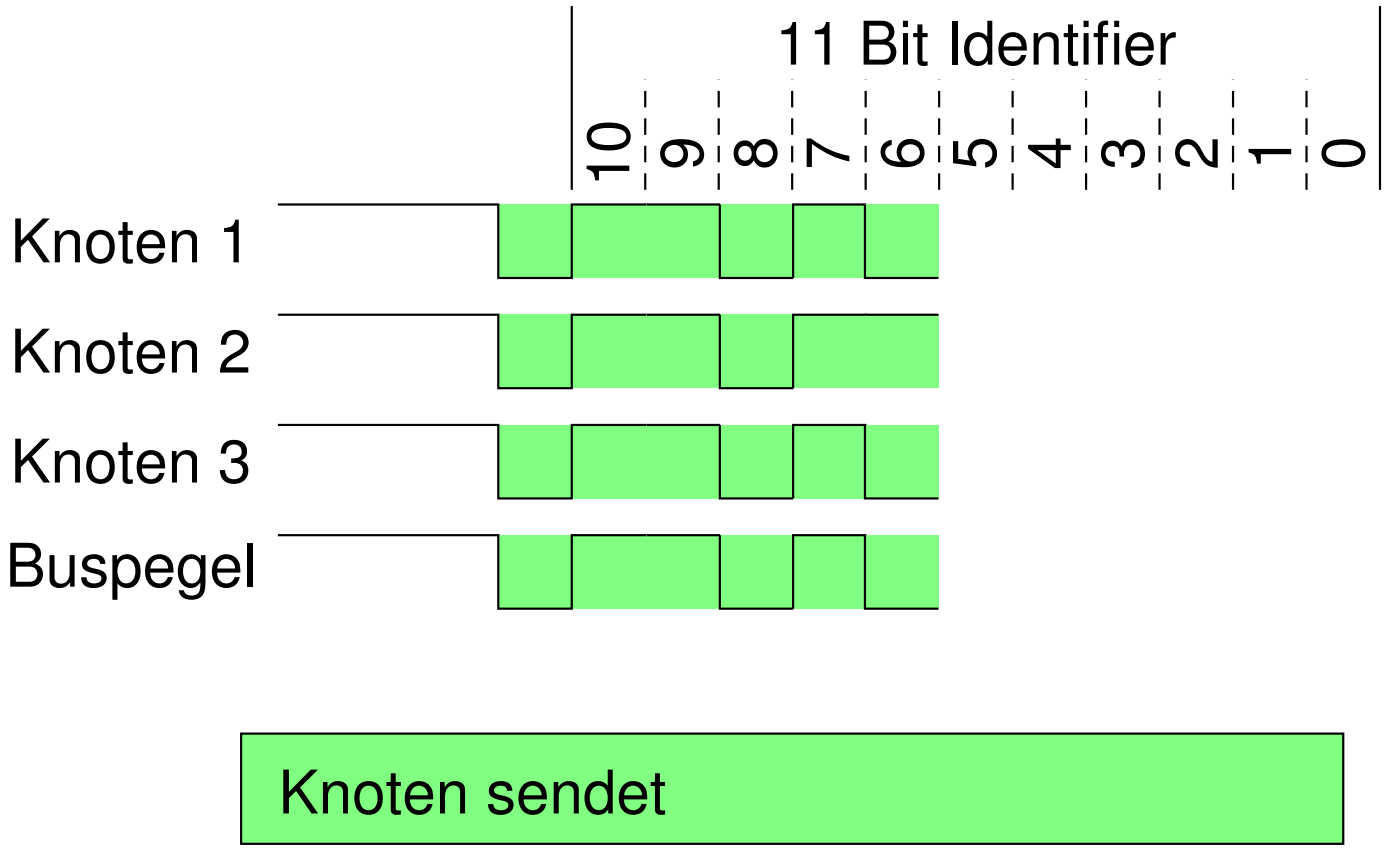
Grundlagen

Arbitrierung - Identifier Priorität



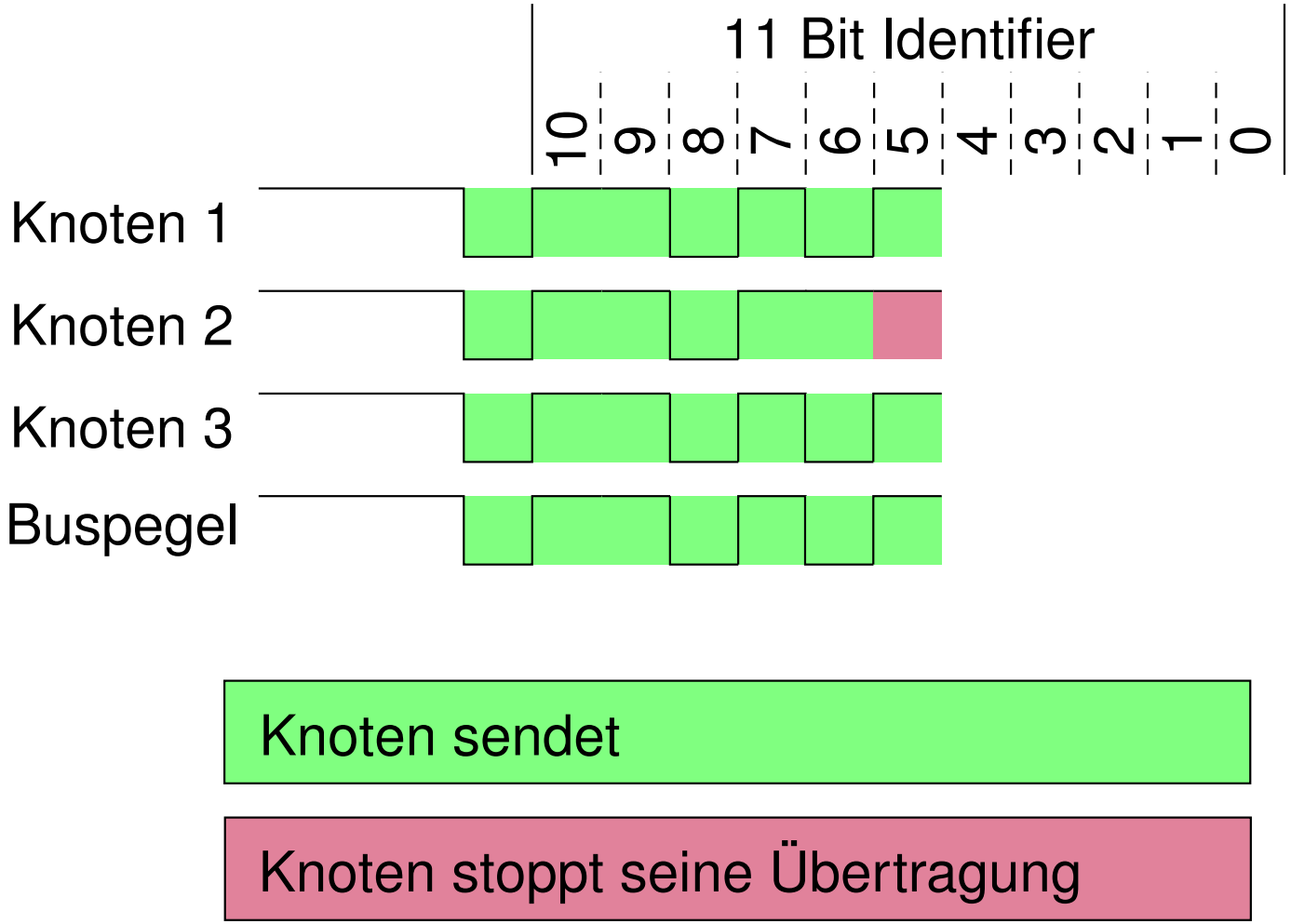
Grundlagen

Arbitrierung - Identifier Priorität



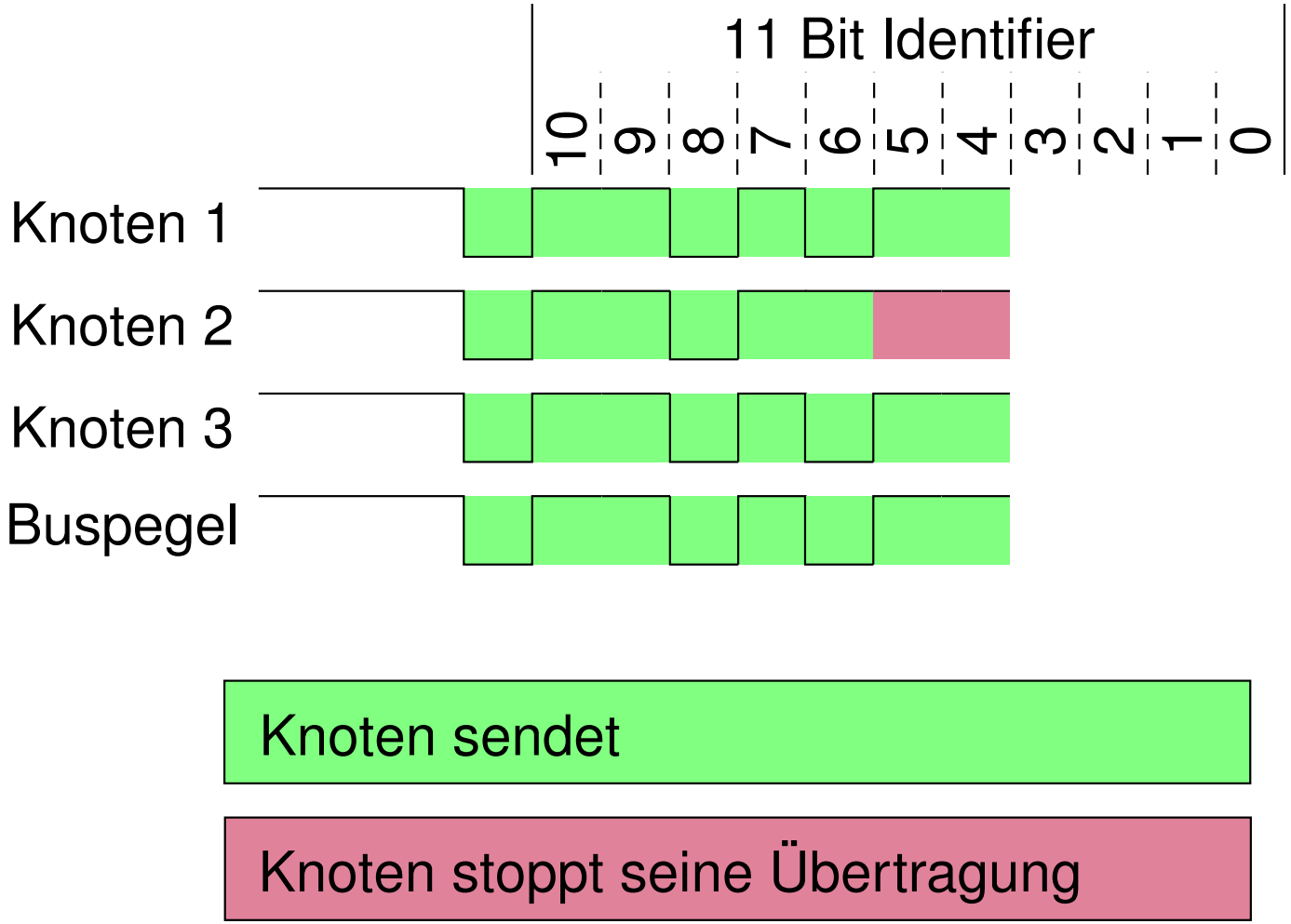
Grundlagen

Arbitrierung - Identifier Priorität



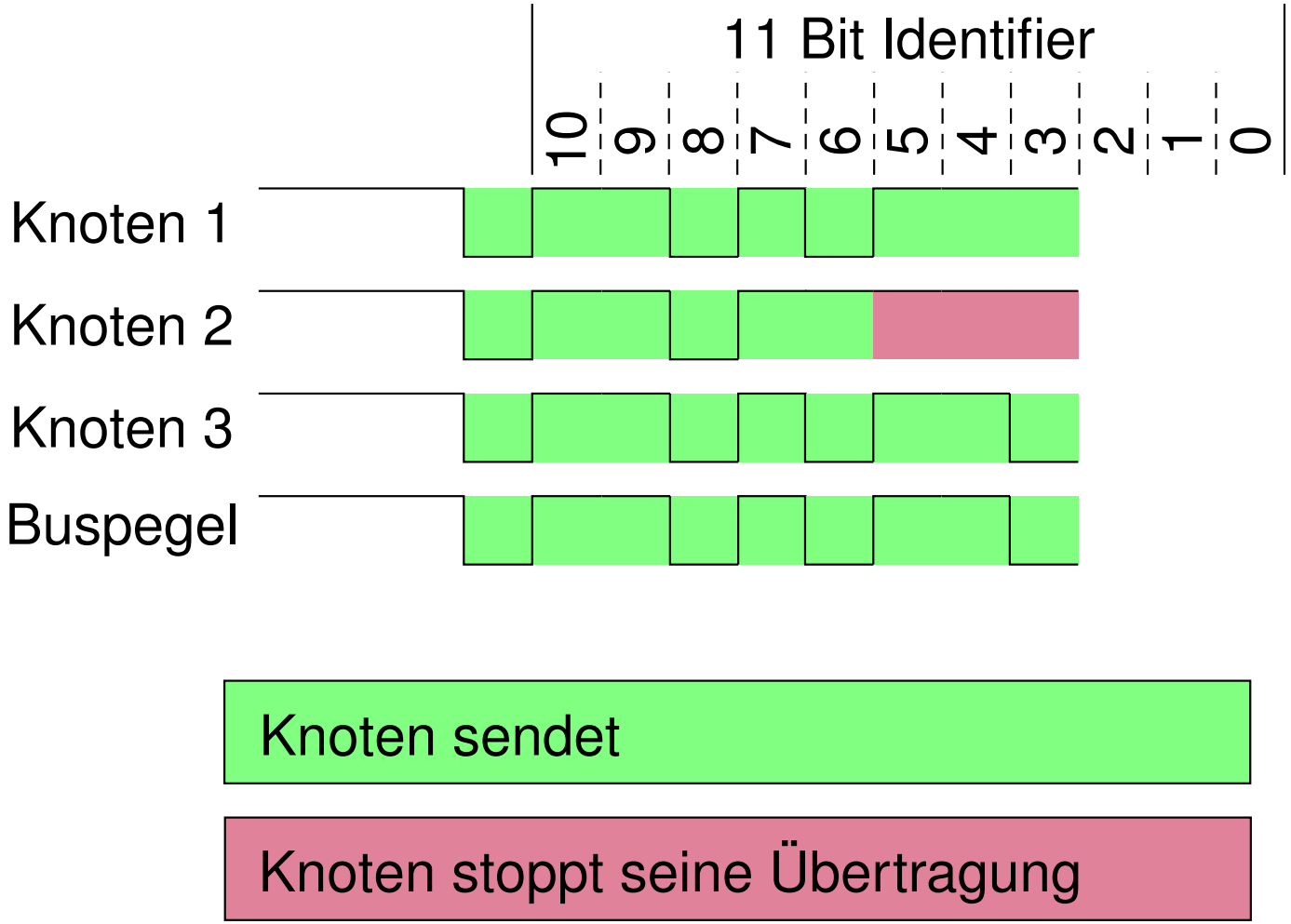
Grundlagen

Arbitrierung - Identifier Priorität



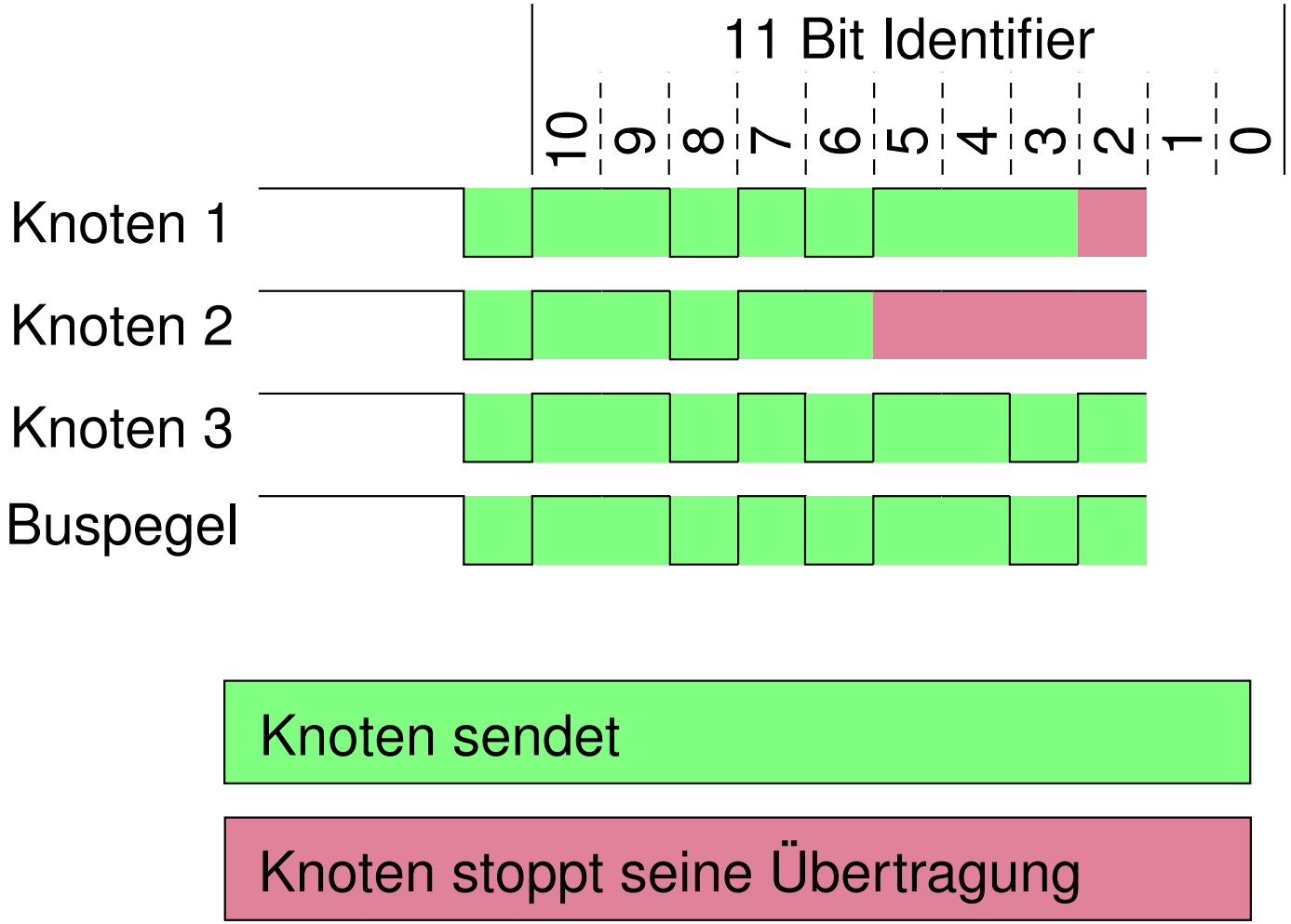
Grundlagen

Arbitrierung - Identifier Priorität



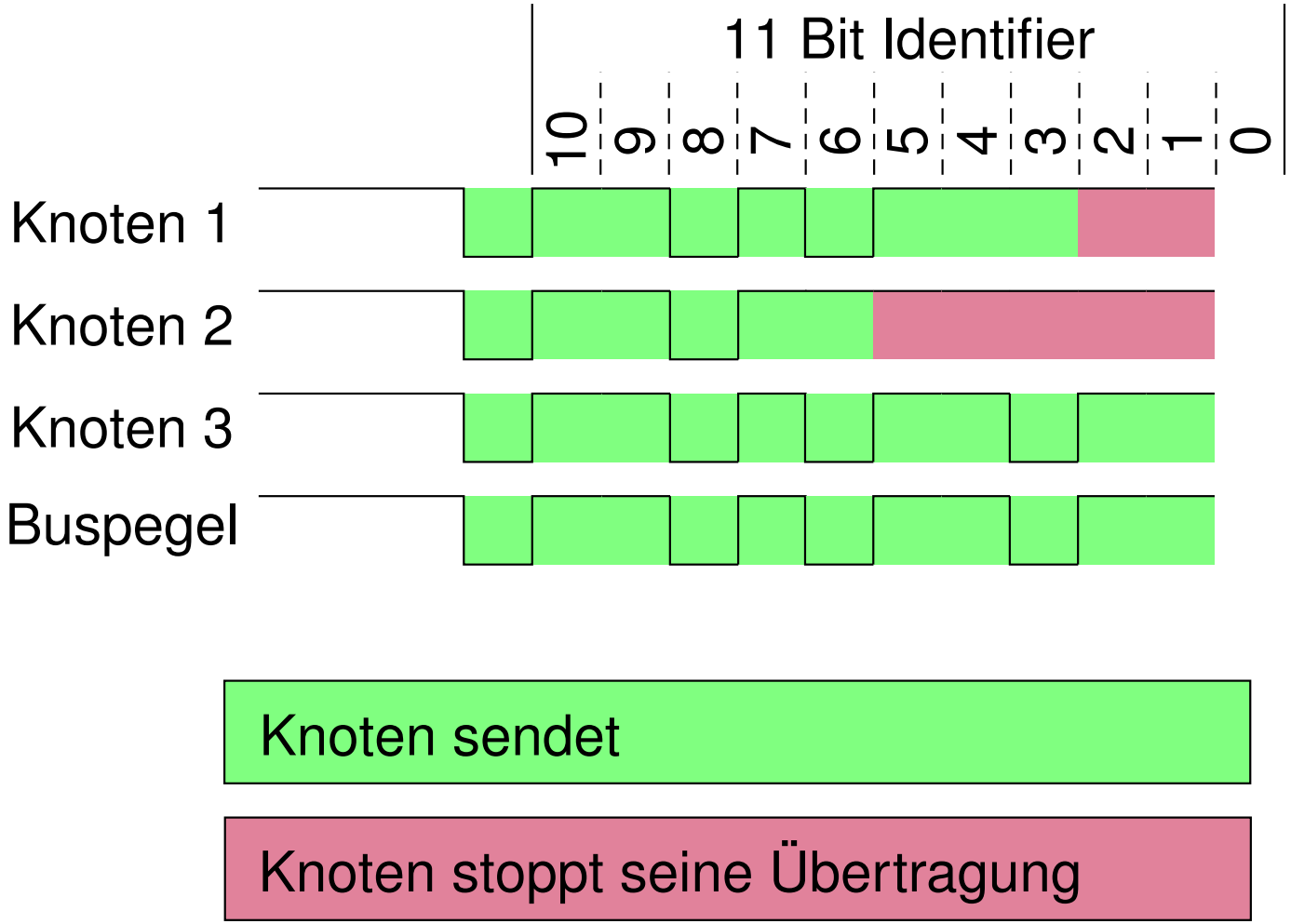
Grundlagen

Arbitrierung - Identifier Priorität



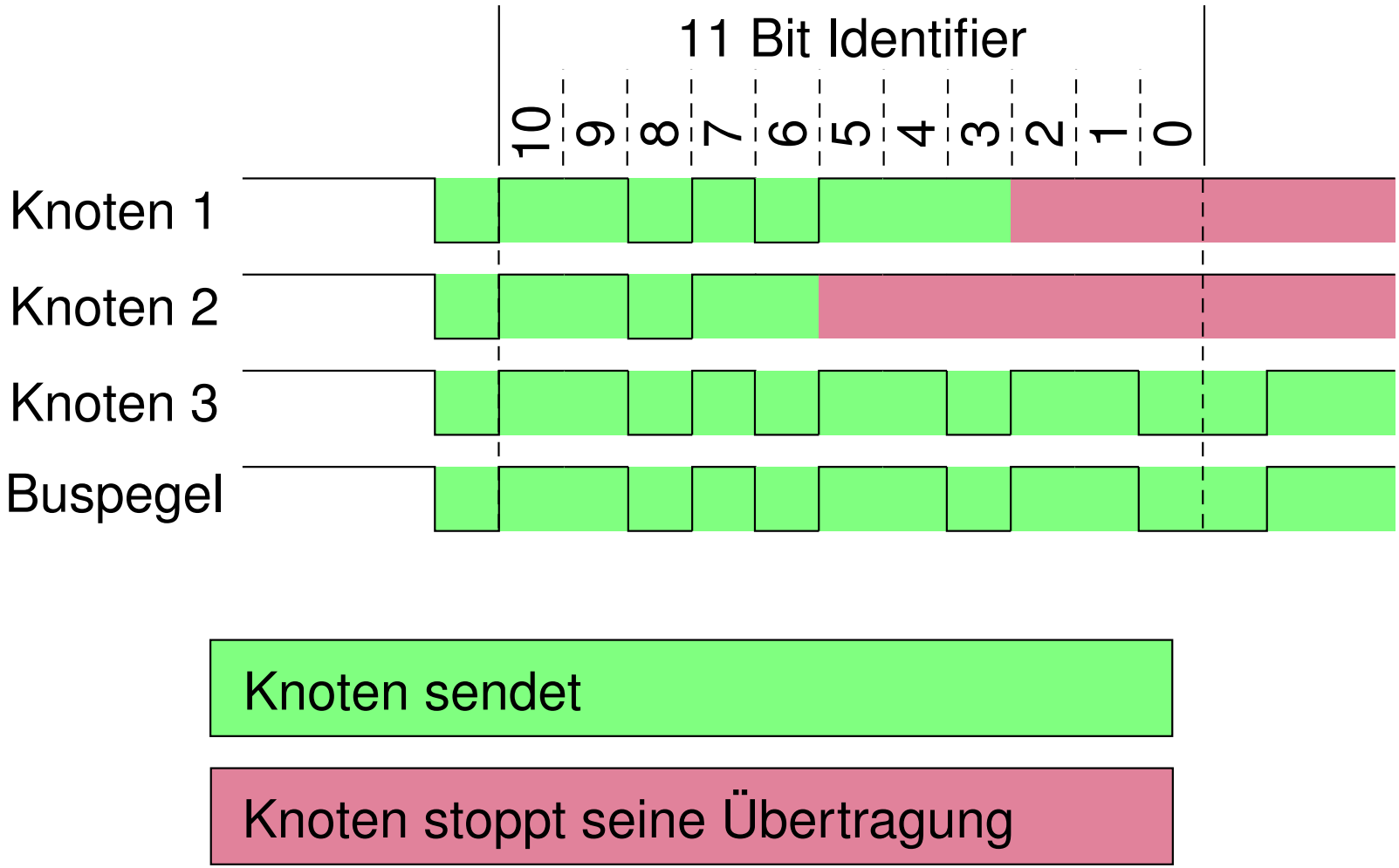
Grundlagen

Arbitrierung - Identifier Priorität



Grundlagen

Arbitrierung - Identifier Priorität

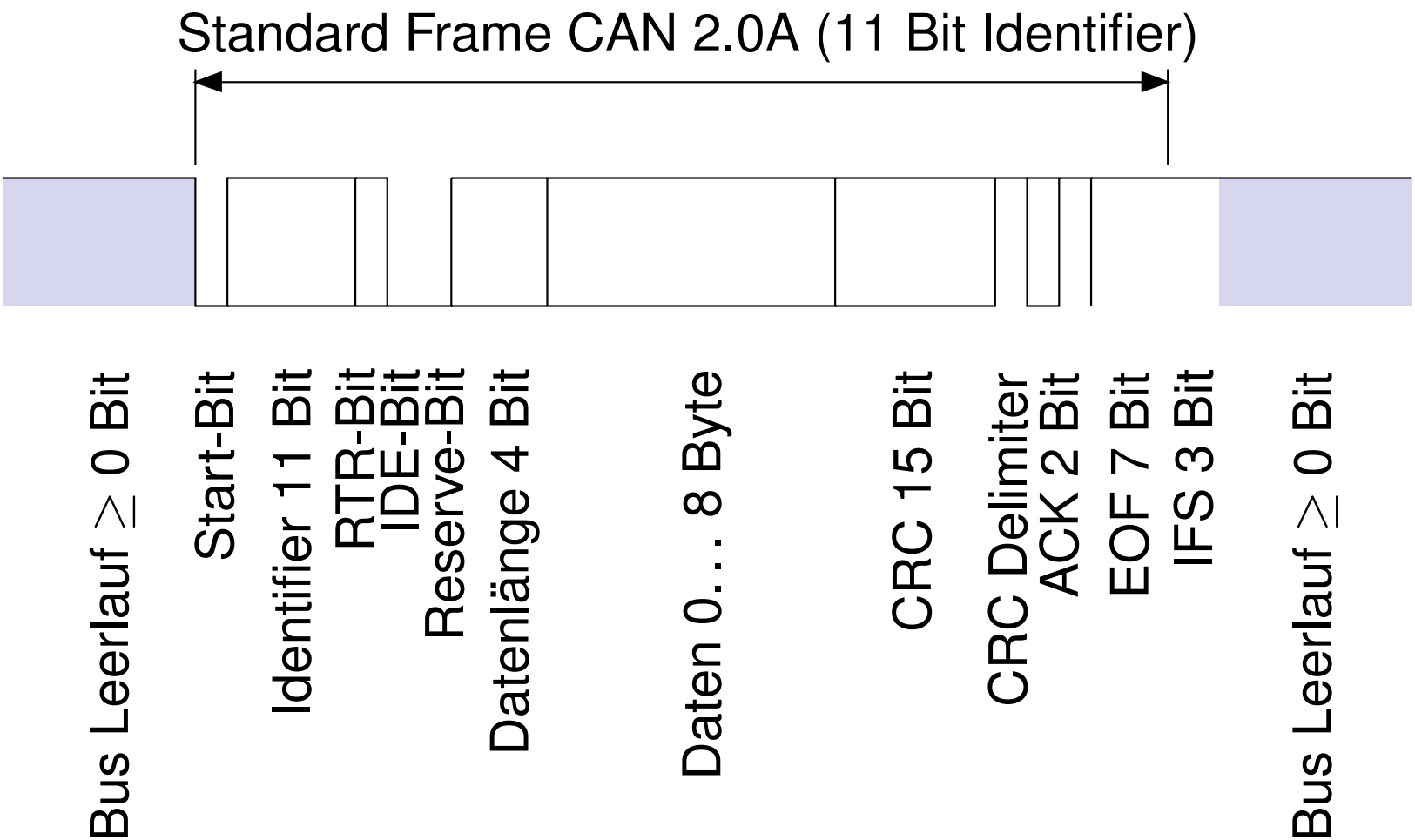


Grundlagen

Gliederung

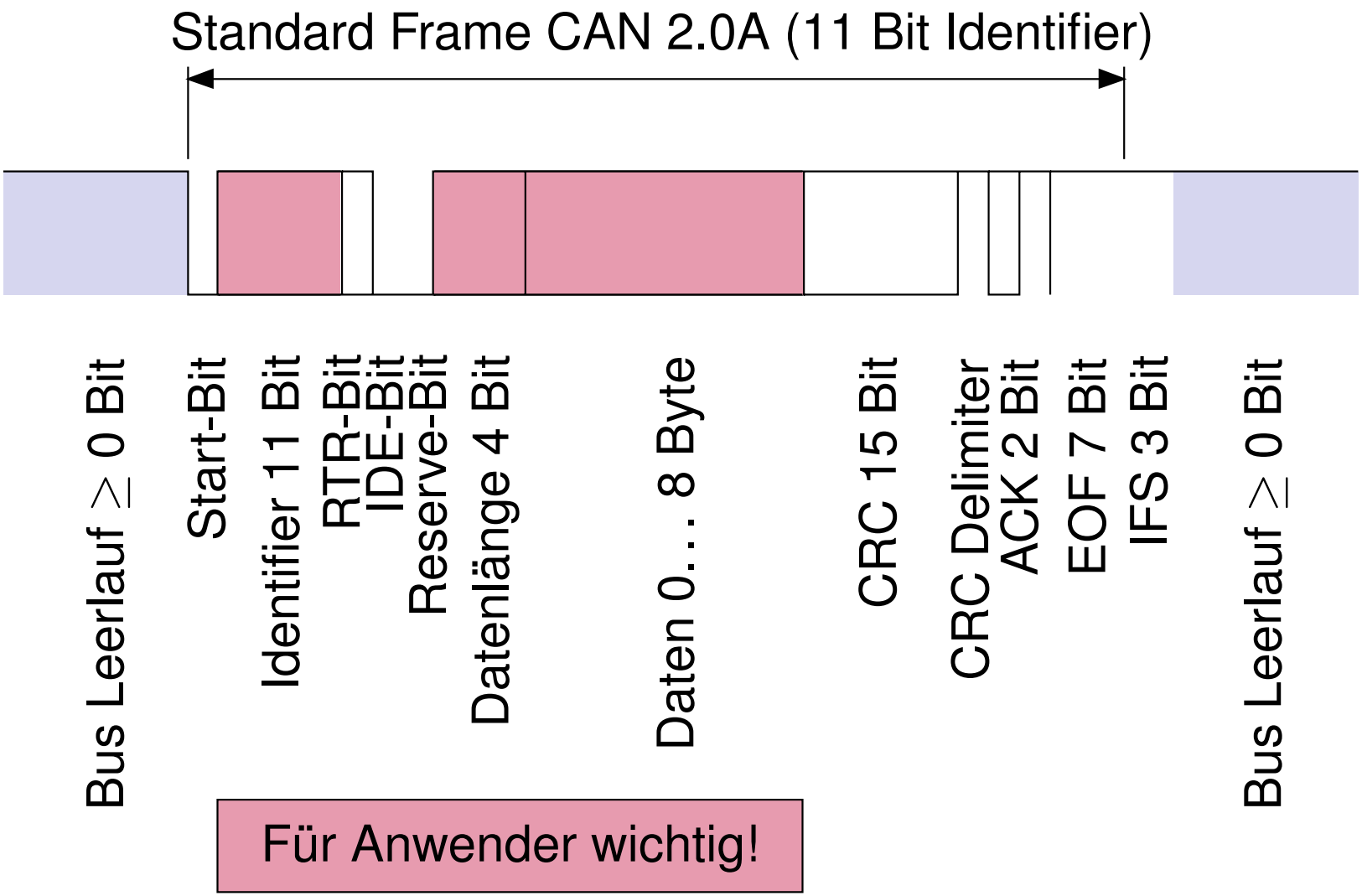
Grundlagen

Nachrichtenformat



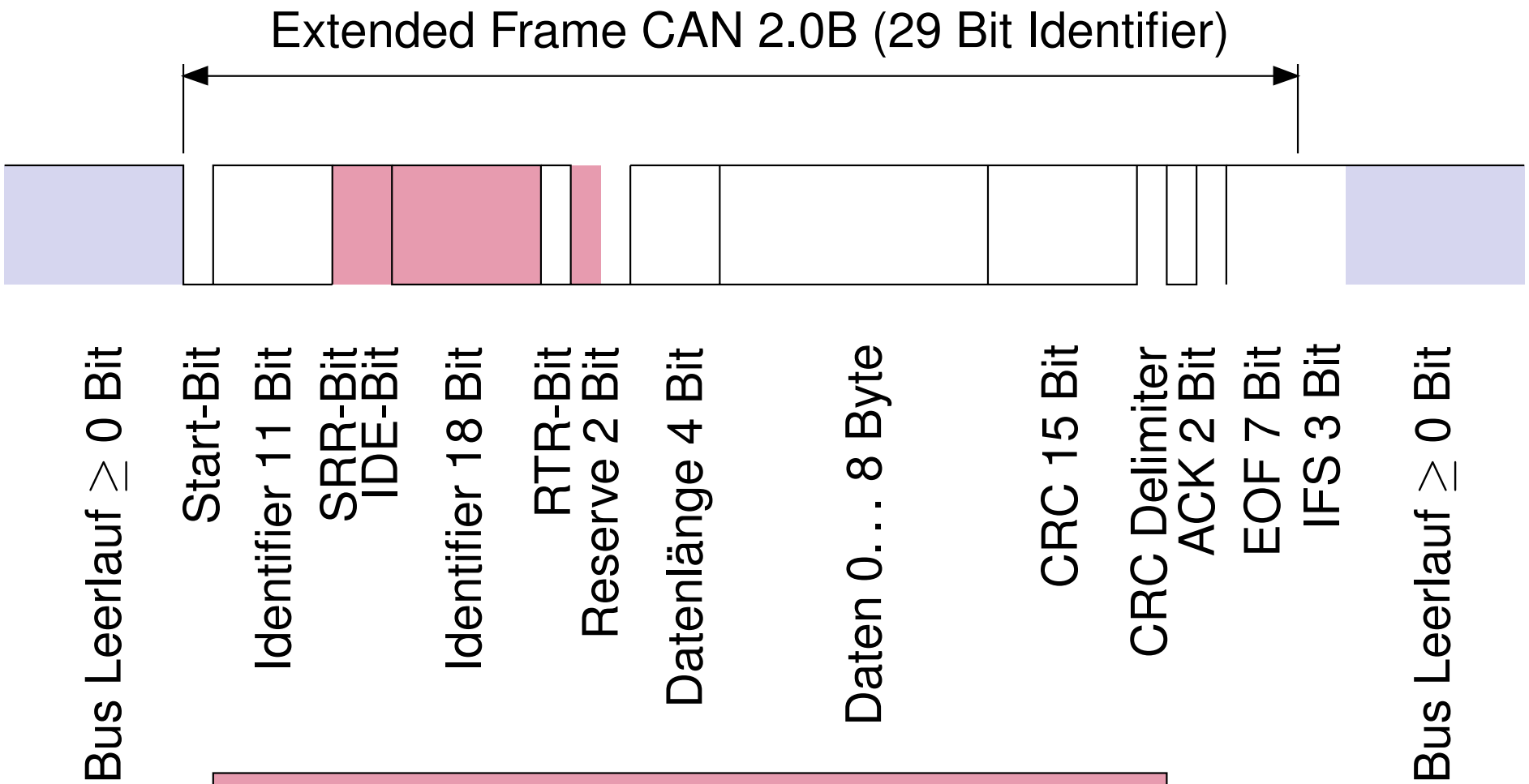
Grundlagen

Nachrichtenformat



Grundlagen

Nachrichtenformat



CAN unter Linux und Praxis

Gliederung

CAN unter Linux und Praxis

Kernelkonfiguration

- Linux Kernel version >= 2.6.32

```
CONFIG_CAN=m
CONFIG_CAN_RAW=m
CONFIG_CAN_BCM=m
CONFIG_CAN_DEV=m
CONFIG_CAN_CALC_BITTIMING=y

$ make modules
```

- Kernelmodule laden

```
$ sudo modprobe can_raw
$ sudo modprobe can_dev
```

- Modul mit Gerätetreiber laden, hier:

```
$ insmod systec_can.ko
```

CAN unter Linux und Praxis

Netzwerkkonfiguration

- CAN-Bus konfigurieren

```
$ ip link set can0 type can bitrate 500000
```

- optional Timings setzen (Vorsicht nur für Profis) (z.B. Werte für 125000 Baud)

```
$ ip link set can0 type can tq 500 prop-seg
6 phase-seg1 7 phase-seg2 2
```

- ...und Start

```
$ ifconfig can0 up
```

- Reset des CAN-Buses, z.B. notwendig nach Kurzschluß

```
$ ip link set can0 type can restart
```

- für die Statistik

```
$ ip -details -statistics link show can0
```

CAN unter Linux und Praxis

Gliederung

CAN unter Linux und Praxis

Werkzeuge

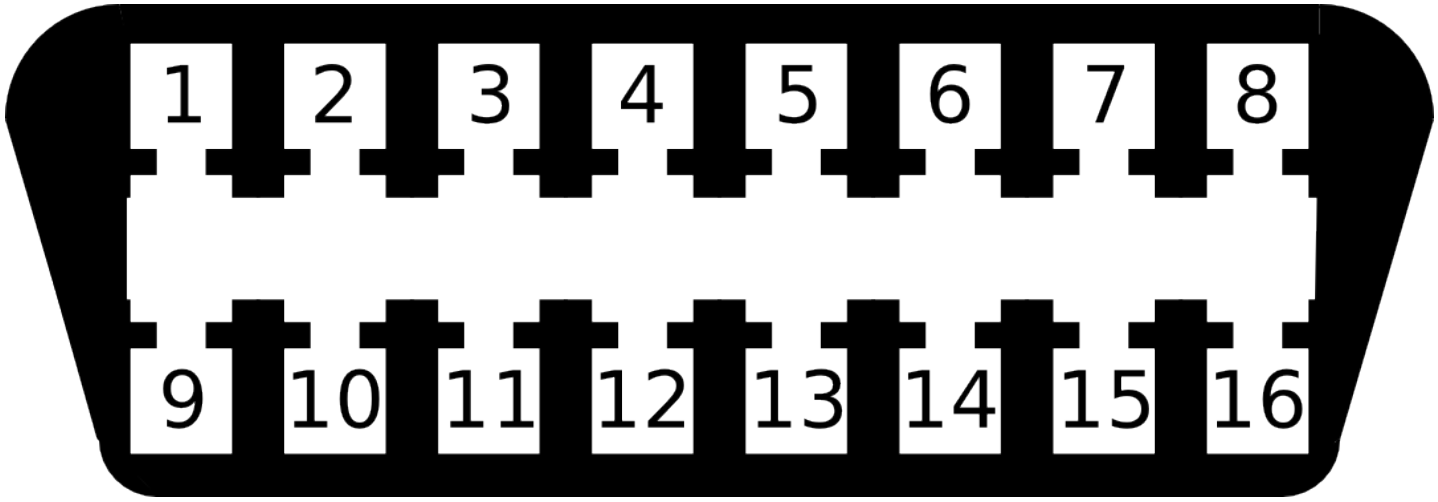
- Programme aus dem Paket can-utils
- cangen - generiert zufällig CAN-Pakete einmalig oder zyklisch
- cansend - sendet bestimmte CAN-Pakete
- candump - tcpdump für den CAN-Bussysteme
- canbusload - top für den CAN-Bussysteme
- Wireshark - Analyse vom Netzwerk
- ...

CAN unter Linux und Praxis

Gliederung

CAN unter Linux und Praxis

OBD Diagnosestecker ISO 15031-3



Pin 6 CAN_HIGH

Pin 14 CAN_LOW

CAN unter Linux und Praxis

Gesetzliche vorgeschriebene CAN-Identifier ISO 15765-4

Diagnoseanfrage-Identifier	Steuergerätantwort-Identifier
0x7DF	funktionale Anfrage
0x7E0	0x7E8
0x7E1	0x7E9
0x7E2	0x7EA
0x7E3	0x7EB
0x7E4	0x7EC
0x7E5	0x7ED
0x7E6	0x7EE
0x7E7	0x7EF

CAN unter Linux und Praxis

OBD Anfrage Nachrichtenlayout

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 7	Füllbyte z.B. `0x55`							
Byte 6	Füllbyte z.B. `0x55`							
Byte 5	Füllbyte z.B. `0x55`							
Byte 4	Füllbyte z.B. `0x55`							
Byte 3	Füllbyte z.B. `0x55`							
Byte 2	Parameter Identifier PID z.B. `0x05`							
Byte 1	Service Identifier SID z.B. `0x02`							
Byte 0	Anzahl der nachfolgenden Datenbytes z.B. `0x02`							

z.B. Kühlw.temperatur ID=`0x7DF` Daten=`0x 02 02 05 55 55 55 55 55`

CAN unter Linux und Praxis

OBD Antwort Nachrichtenlayout

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 7	Füllbyte z.B. `0xAA`							
Byte 6	Füllbyte z.B. `0xAA`							
Byte 5	Füllbyte z.B. `0xAA`							
Byte 4	Füllbyte z.B. `0xAA`							
Byte 3	Antwort-Werte z.B. `0x3E`							
Byte 2	Parameter Identifier PID z.B.`0x05`							
Byte 1	Service Identifier SID & `0x40` z.B.`0x42`							
Byte 0	Anzahl der nachfolgenden Datenbytes z.B.`0x03`							

z.B. Kühlw.temperatur ID=`0x???` Daten=`0x 03 42 05 3E AA AA AA AA`

CAN unter Linux und Praxis

Umrechnung der HEX-Werte in physikalische Werte

$$PH = \frac{HEX}{2^n - 1}(MAX - MIN) + MIN$$

PH	...	Physikalischer Wert
HEX	...	Hexadezimaler Wert
MAX	...	Maximaler Wert
MIN	...	Minimaler Wert
n	...	Datengröße in [Bit]

z.B. $HEX = \texttt{0x3E} = 62$, $n = 8$, $MIN = -40\,°\text{C}$, $MAX = +215\,°\text{C}$

$$PH = \frac{62}{2^8 - 1}(215\,°\text{C} - (-40\,°\text{C})) + (-40\,°\text{C}) = 22\,°\text{C}$$

Literatur

- CAN 2.0 Specifikation Bosch `http:www.semiconductors.bosch.demediapdfcanliteraturcan2spec.pdf`
- ISO 15765-4: Road vehicles — Diagnostics on Controller Area Networks (CAN) - Part 4 Requirements for emissions-related systems. 2005
- ISO 15031-3: Road vehicles - Communication between vehicle and external equipment for emissions-related diagnostics - Part 3 Diagnostic connector and related electrical circuits, specification and use
- OBD-II Pids `http:en.wikipedia.orgwikiOBD-II_PIDs`
- Zimmermann, W.; Schmidgall, R.: Bussysteme in der Fahrzeugtechnik. Vieweg, 2006

Fragen

Fragen?

Vielen Dank für Ihre Aufmerksamkeit!