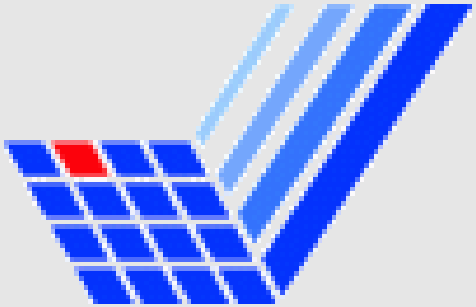
LIN Bus

Matthias Meier

matthias2.meier@udo.edu

PG AutoLab
Seminarwochenende 21.-23. Oktober 2007



Überblick

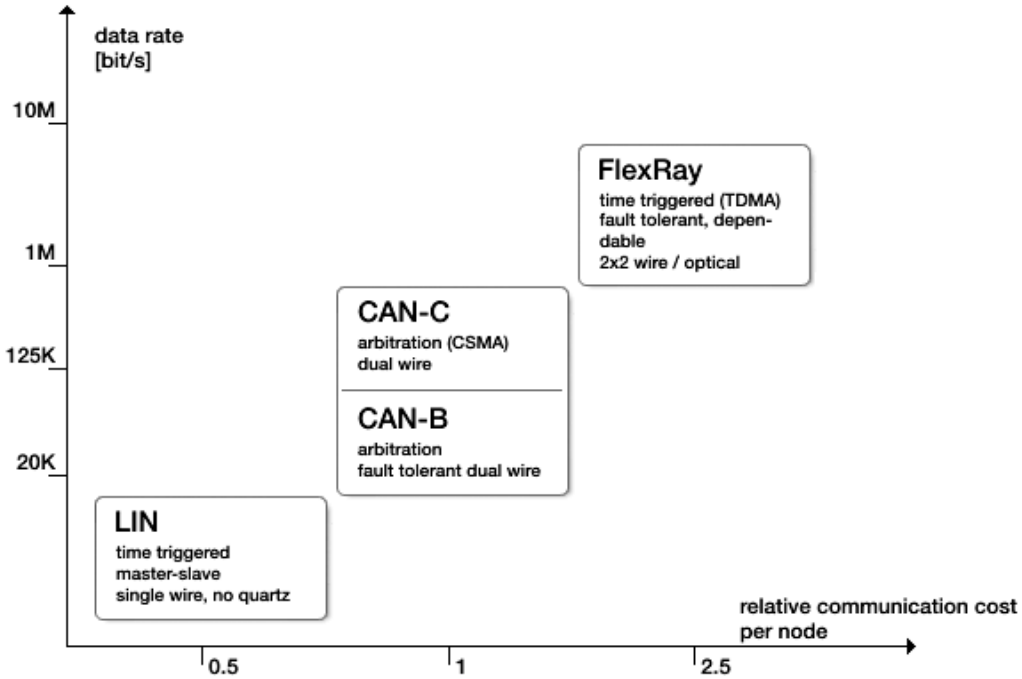
- Einführung
- Physical Layer
- Data Link Layer
- LIN Configuration Language
- Beispiel aus der Praxis

Überblick

- Einführung
- Physical Layer
- Data Link Layer
- LIN Configuration Language
- Beispiel aus der Praxis

Motivation

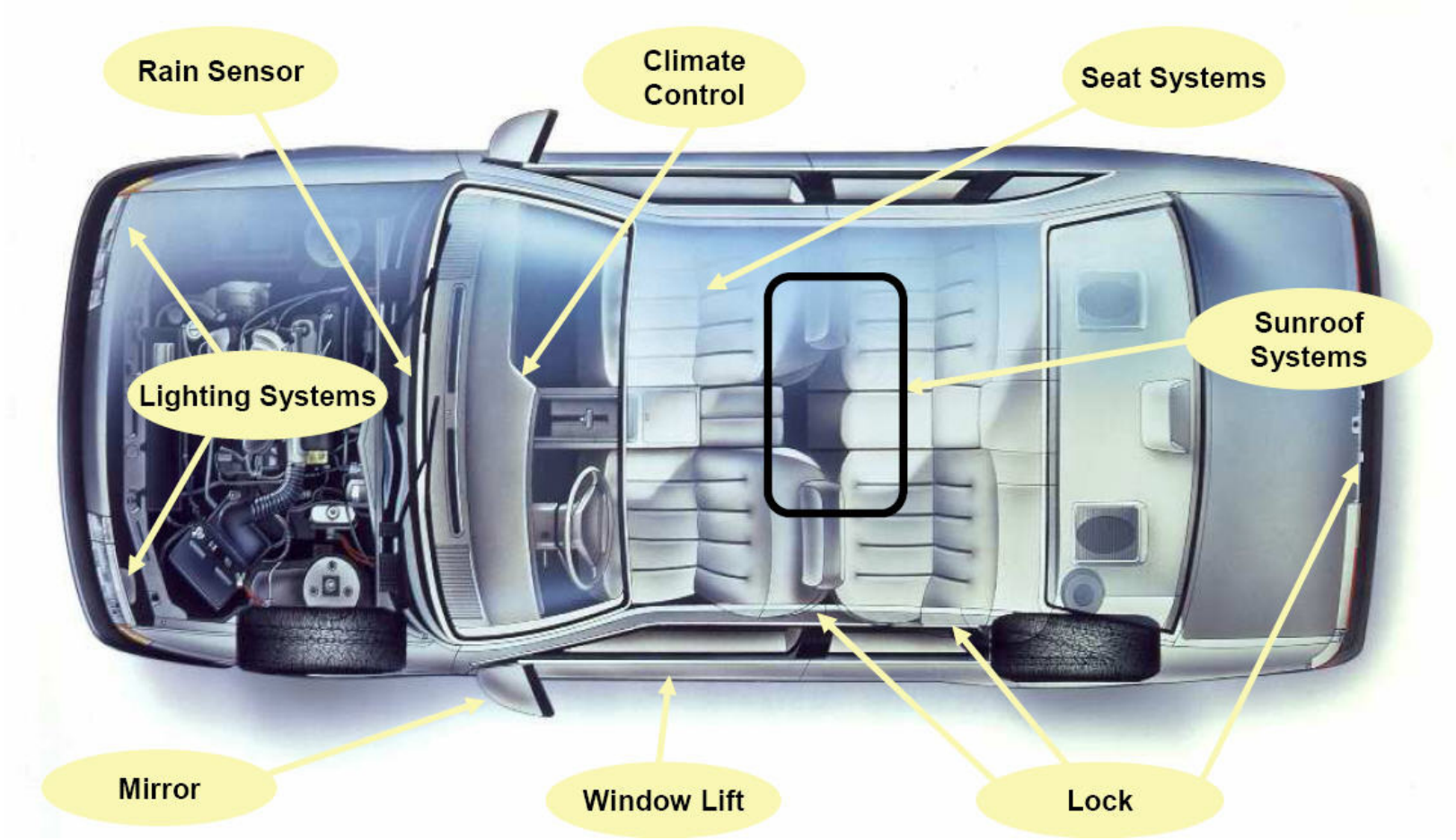
- einfaches & kostengünstiges Bussystem
- für Bereiche in denen Bandbreite & Vielseitigkeit vom CAN-Bus nicht benötigt wird
  - einfache Sensor & Aktor Anwendungen



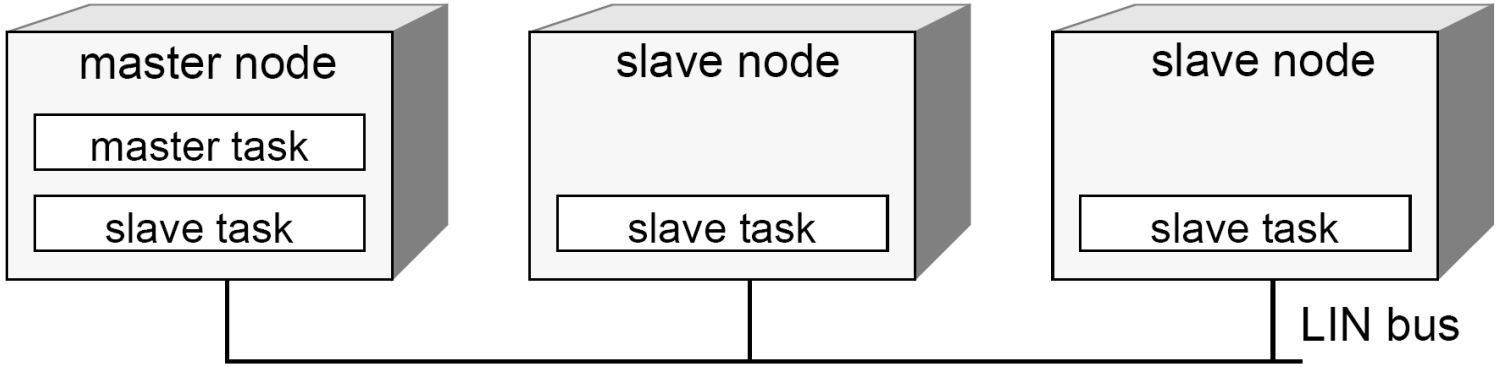
Eigenschaften

- seit 1998 von Konsortium entwickelt
- bidirektionale Ein-Draht-Leitung
- 20 kbit/s maximale Datenrate
- bis zu 16 Bus-Teilnehmer
- maximale Länge aller Busleitungen 40 m
- als Subbus eingesetzt
- basiert auf verbreiteten UART-Interfaces
  - "kostengünstige Standard Komponenten"
- Slaves synchronisieren sich selbst
  - werden keine Quarze oder Keramikresonatoren für Slaves benötigt
- vorhersagbares Verhalten

Einsatzgebiete



Aufbau eines LIN-Clusters



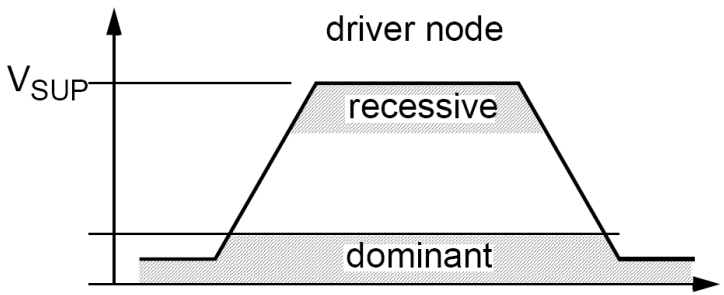
- Single-Master / Multiple-Slave Konzept
- Master Task steuert Kommunikation
- Kommunikation zwischen den Knoten läuft über eine Datenleitung
- Master Steuergerät dient meistens als Gateway zum in der Hierarchie höheren Netzwerk (z.B. zum CAN-BUS)

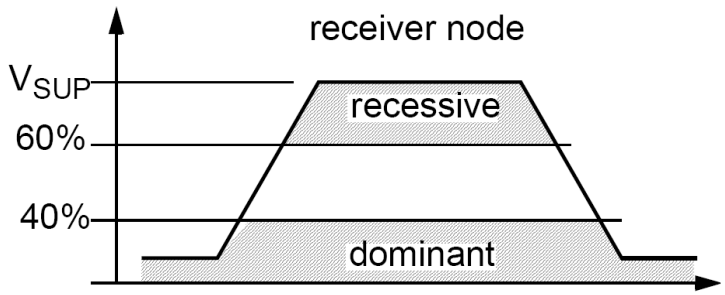
Überblick

- Einführung
- Physical Layer
- Data Link Layer
- LIN Configuration Language
- Beispiel aus der Praxis

Physical Layer

- 12 V Ein-Draht-Bus (basiert auf dem ISO 9141 Standard)
  - Datenrate wurde auf 20 kbit/s beschränkt
  - Flankensteilheit wurde angepasst
- Signale werden relativ zur Versorgungsspannung übertragen
  - V_{SUP}: rezessive Zustand (logische 1 wird übertragen)
  - 0 V: dominante Zustand (logische 0 wird übertragen)

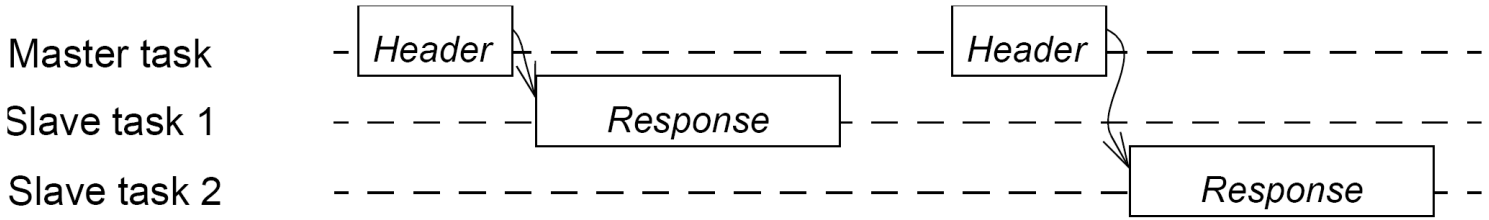




Überblick

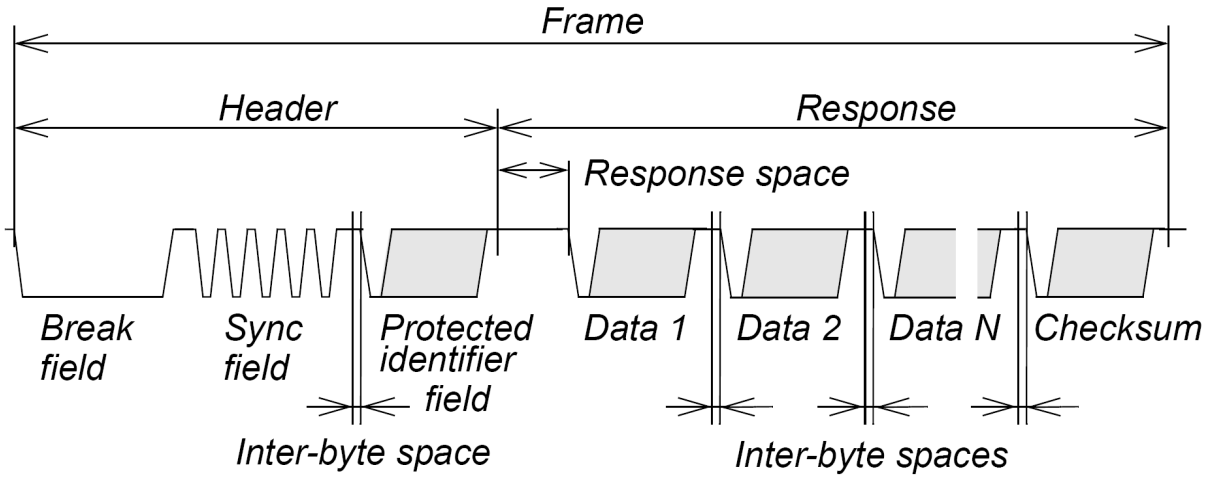
- Einführung
- Physical Layer
- Data Link Layer
- LIN Configuration Language
- Beispiel aus der Praxis

Frames



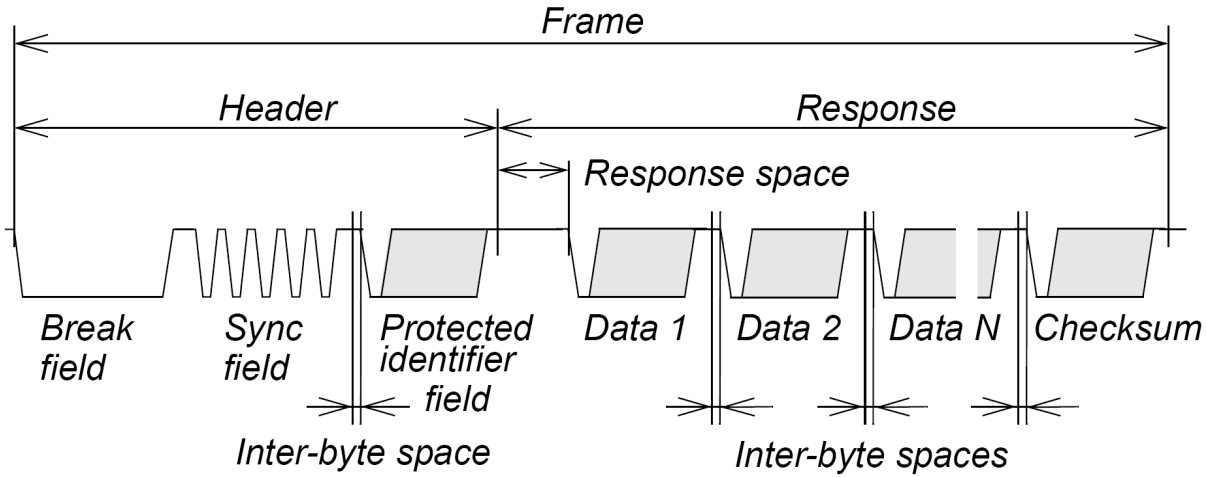
- Master Task entscheidet welcher Frame auf dem Bus übertragen werden soll
- Slave Task stellt die Daten zur Verfügung, die in dem Frame übertragen werden sollen
- Multicast: alle Slaves können die Daten empfangen und lesen die für sie relevanten

Frame (Header)



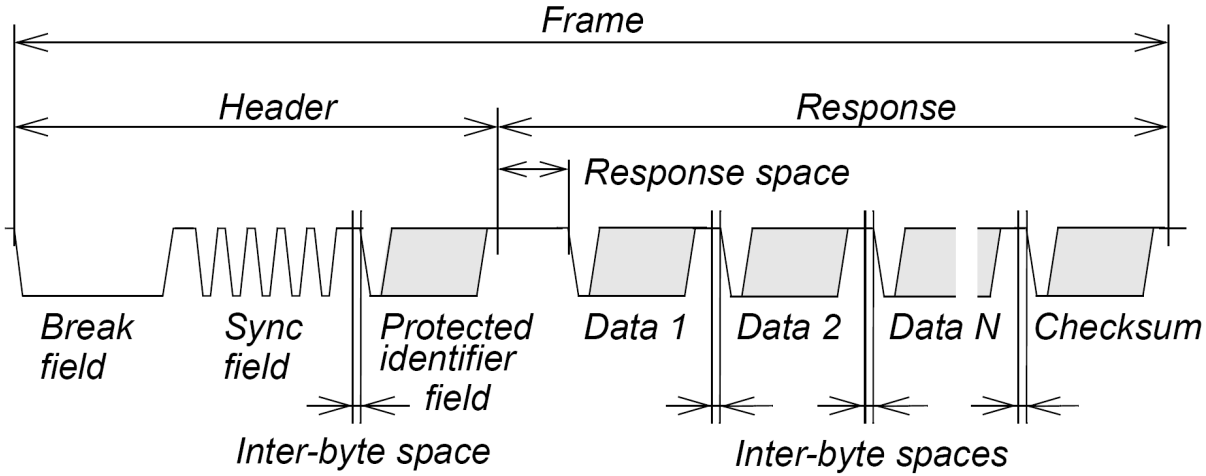
- Break Field
  - signalisiert einen neuen Frame
  - besteht aus mindestens 13 low-Bits und einem high-Bit
  - als einzige Bitfolge kein Standard UART-Zeichen und dadurch eindeutig von den Slaves als neuer Frame zu identifizieren

Frame (Header)



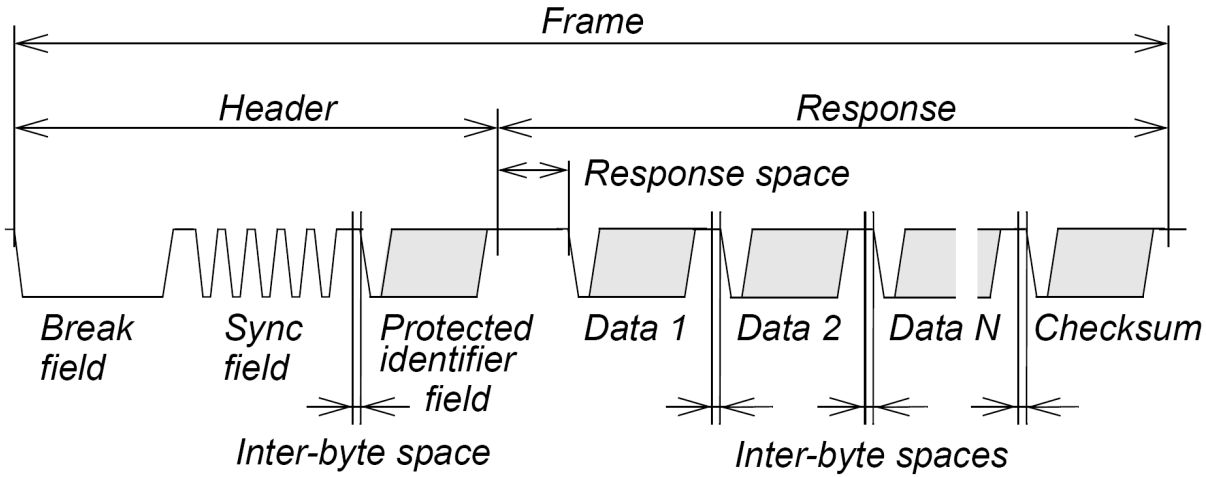
- Sync Field
  - dient der Bittakt-Synchronisation der Slaves
  - besteht aus alternierenden low-high-Bitfolge

Frame (Header)



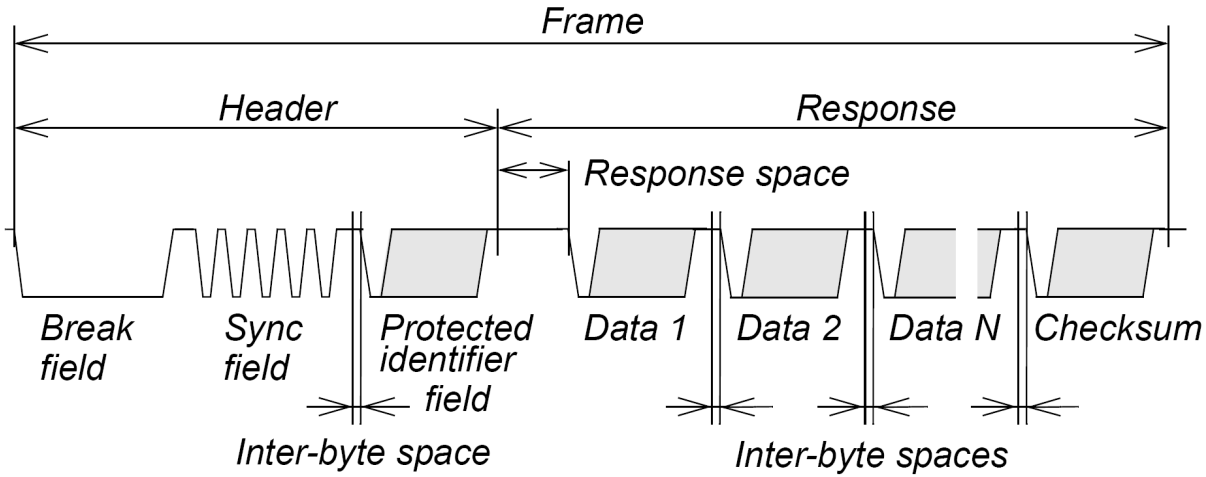
- Protected Identifier Field
  - adressiert nicht direkt die Slaves, sondern die gewünschte Datenbotschaft, die die Slaves zu senden haben (inhaltsbezogene Adressierung)
  - ersten 6 Bits Identifier und letzten beiden Paritätsbits
    - stehen damit 64 verschiedene Identifier zur Verfügung

Frame (Response)



- Data
  - in einem Frame können bis zu 8 Datenbytes übertragen werden

Frame (Response)



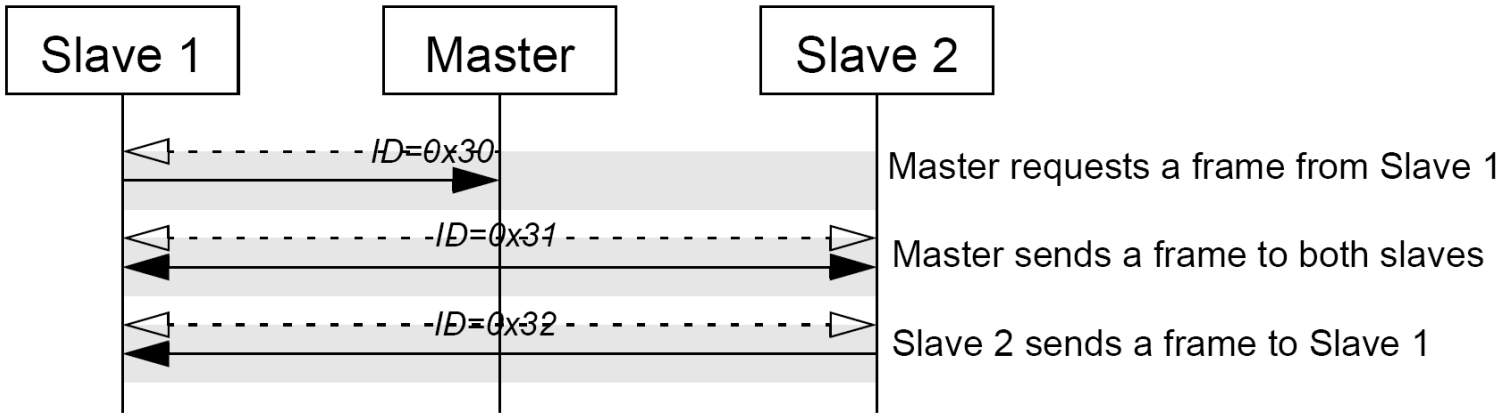
- Checksum
  - wird aus den Datenbytes und aus dem Identifier im Header berechnet

Scheduling-Tabelle

- gibt die Reihenfolge der Frames und das Intervall zwischen den Frames an
- im Master-Steuergerät abgelegt und vom Master Task immer wieder zyklisch abgearbeitet
- wird zur Entwicklungszeit festgelegt
- Master kann, falls nötig, zwischen verschiedenen Tabellen wählen, um sich z.B. den Umgebungsbedingungen anzupassen
  - können so die Reaktionszeit der einzelnen Komponenten entsprechend der Situation anpassen

Frametypen

- Unconditional Frames
  - "standard" Frames im LIN-Cluster
  - ständig periodisch zu sendende Frames
  - Übertragung der Frame-Response von einem festen Slave (der Slave sollte immer antworten)



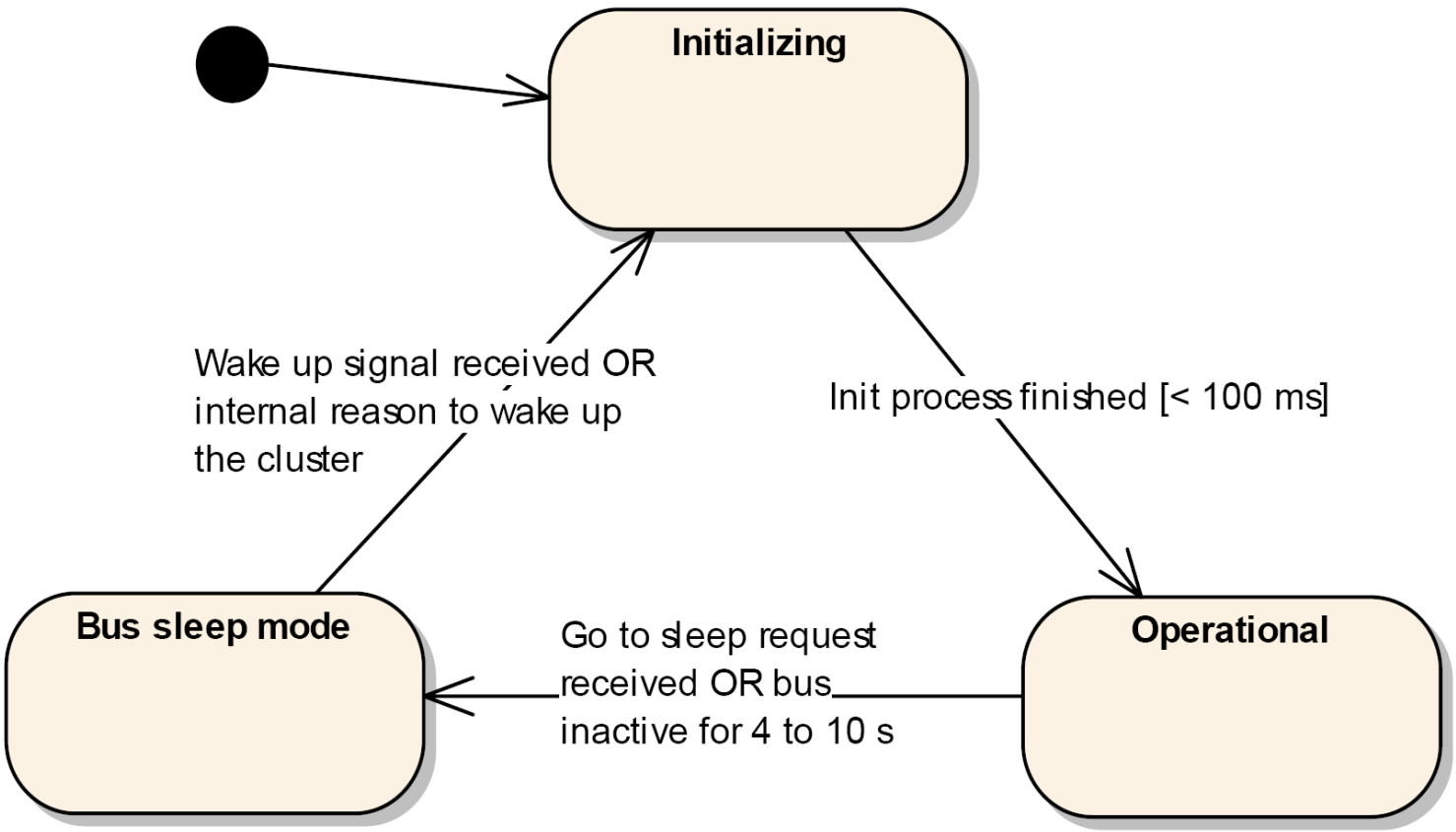
Frametypen

- Event Triggered Frames
  - kann mehrere Slave-Knoten mit einem einzigen Frame abfragen
  - Slave antwortet nur, wenn sich Daten innerhalb des Steuergerätes geändert haben
  - falls kein Slave Daten übertragen muss, bleibt der Bus nach dem Header frei
  - Vorgehen bei einer Kollision auf dem Bus:
    - Slaves brechen das Senden ab
    - Master wechselt zu einer kollisionslösenden Scheduling-Tabelle und arbeitet diese einmalig ab
  - Vorteile:
    - Antwortzeiten von selten eintretenden Events können so verbessert werden, ohne viel Bandbreite des Busses opfern zu müssen

Frametypen

- Sporadic Frames
  - eine Gruppe von Unconditional Frames, die sich einen Zeitschlitz teilen
  - werden nur versendet, wenn tatsächlich Daten im Master vorliegen oder der Master Daten von einem Slave benötigt
  - sonst sendet der Master keinen Header und der Slot bleibt frei
  - für den Fall, dass mehrere Frames übertragen werden sollen,
    - werden statische Prioritäten für diese Frames vergeben
    - Master versendet dann nur den Header mit dem höchstpriorisierten Frame
  - Vorteile:
    - bringen Dynamik in die Schedulling-Tabelle, ohne die Vorhersagbarkeit im Rest der Tabelle zu verlieren

Netzwerk Management (Sleep Modus)



Fehlererkennung & -behandlung

- wie erkennen die Slaves Fehler?
  - Sender muss das von ihm gesendete Signal mitlesen
    - bei einem festgestellten Fehler bricht Sender die Übertragung ab
  - Auswertung der Identifier-Paritätsbits und der Checksum
- wie werden Fehler mitgeteilt? (Response Error Bit)
  - einmal pro Zyklus teilt jeder Slave dem Master über einen gewöhnlichen Unconditional Frame mit, ob ein Fehler aufgetreten ist
- was macht der Master mit den Fehlerinformationen?
  - sammelt alle Error Bits & wertet eigene Fehlerüberwachung aus
  - auf dem Application Layer des Masters wird entsprechend reagiert

Überblick

- Einführung
- Physical Layer
- Data Link Layer
- LIN Configuration Language
- Beispiel aus der Praxis

LIN Configuration Language

- Konfigurationssprache aus der Spezifikation, um
  - Knoten,
  - Signale (zu übertragenden Daten),
  - Frames und
  - Scheduling-Tabellen zu spezifizieren.
- Sprache wird benutzt um die LIN Description File zu erstellen (LDF)
  - aus LDF kann mit Hilfe von Werkzeugen automatisch C-Code für die Steuergeräte erzeugt werden

LIN Configuration Language (LDF)

```
LIN_description_file;
LIN_protocol_version = “2.1”;
LIN_language_version = “2.1”;
LIN_speed = 19.2 kbps;

Nodes {
   Master: DCU, 5 ms, 0.1 ms;
   Slaves: FLWM, LMM, CPM, CEM, ...;
}

Signals {
   WindowButtons: 8, 0, DCU, FLWM;
   MirrorButtonsStatus: 2, 0, DCU, LMM;
   MirrorButtons: 4, 0, DCU, LMM;
   ...
   WaterTempLow: 8, 0, CPM, CEM;
   WaterTempHigh: 8, 0, CPM, CEM;
   CPMFuelPump: 7, 0, CPM, CEM;
   ...
}
```

LIN Configuration Language (LDF)

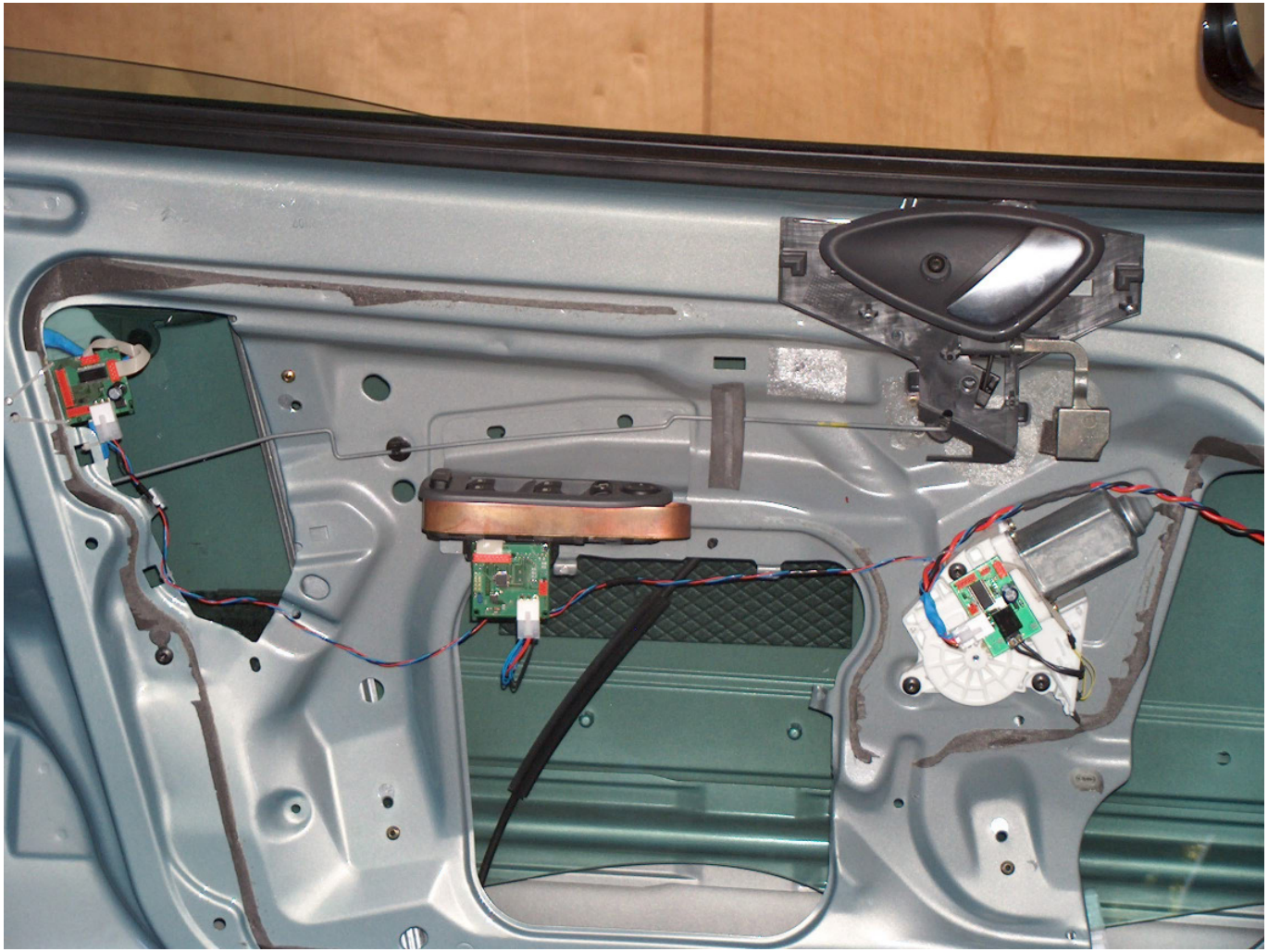
```
Frames {
   DCU_Frm1: 0x01, DCU, 1 {
     MirrorButtonsStatus, 0;
     MirrorButtons, 2;
   }
   CPM_Frm1: 0x02, CPM, 5 {
     WaterTempLow, 0;
     WaterTempHigh, 8;
     CPMFuelPump, 20;
   }
   ...
}

Schedule_tables {
   Normal_Schedule {
     DCU_Frm1 delay 15 ms;
     CPM_Frm1 delay 25 ms;
   }
   ...
}
```

Überblick

- Einführung
- Physical Layer
- Data Link Layer
- LIN Configuration Language
- Beispiel aus der Praxis

Beispiel: Autotür



Zusammenfassung

- einfaches & kostengünstiges Bussystem
- für Sensor & Aktor Anwendungen
- meist als Subsystem zum CAN-Bus (Master als Gateway)
- Master steuert die Kommunikation durch periodisches Senden von Headern und entsprechender Slave antwortet mit einer Datenbotschaft
- Identifier adressiert nicht die Knoten, sondern den Inhalt der Botschaften
- nur einfache Fehlererkennung
  - Fehlerbehandlung auf Anwendungsebene hoch gereicht (in LIN Spezifikation nicht weiter definiert)
- LIN Configuration Language zur Spezifikation des gesamten Bussystems

Fragen

... noch Fragen?

Literatur

- www.lin-subbus.org,
  LIN Specification Package Rev. 2.1
- W. Zimmermann / R. Schmidgall,
  Bussysteme in der Fahrzeugtechnik, Vieweg 2007
- http://www.vector-informatik.de/vi_lin_de,,2816.html
- http://www.hto.fh-deggendorf.de/komm/automotive/technik/techniklin.html
- http://prof.hti.bfh.ch/uploads/media/LIN-Bus.pdf
- http://www.passau.ihk.de/themen/innovation/Innovation/innovativost/netzwerke/itforum/veranstalt/einbetsystem/grzemba.pdf