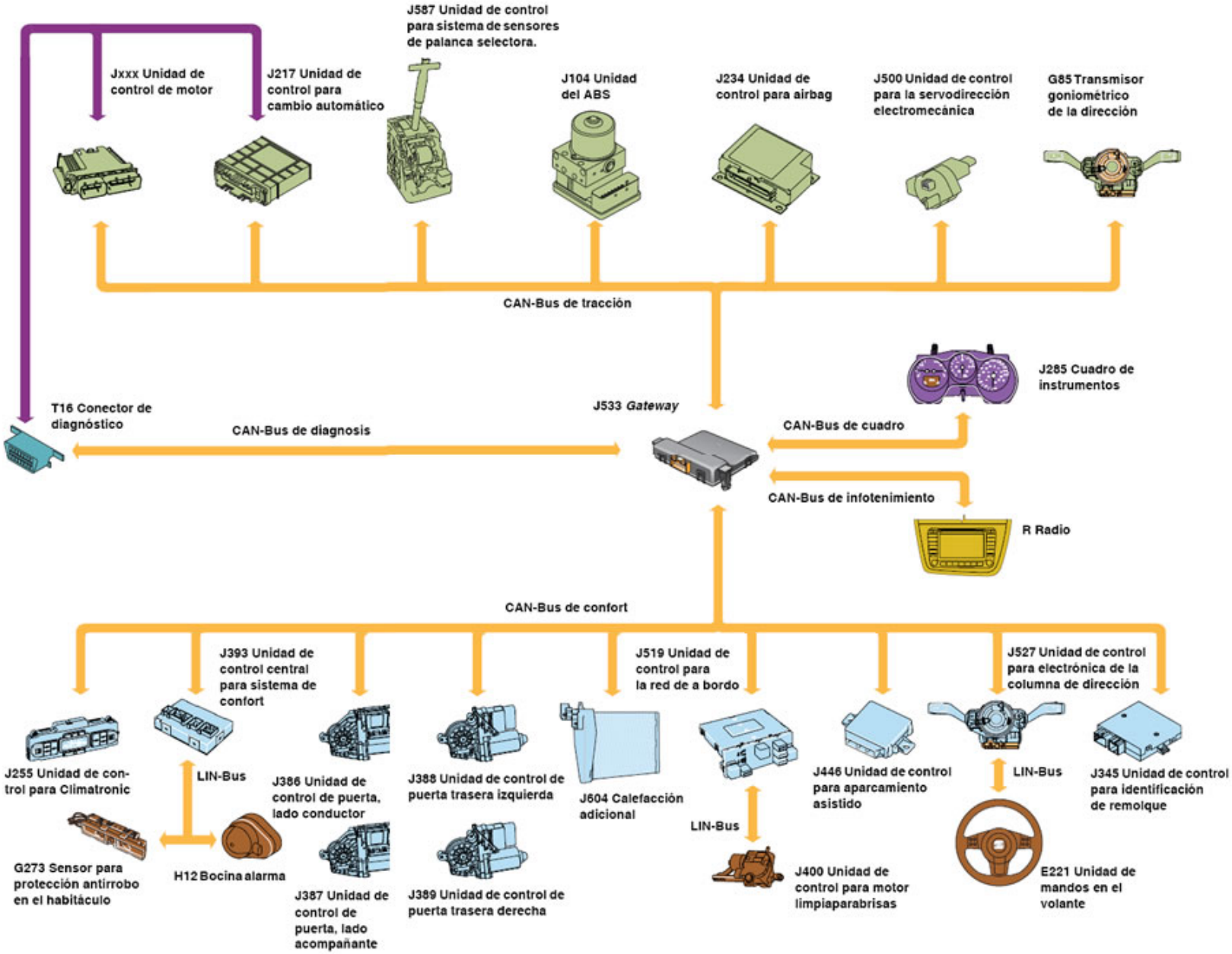
Systemy wbudowane - wykład do samodzielnej pracy

Przemek Błaśkiewicz

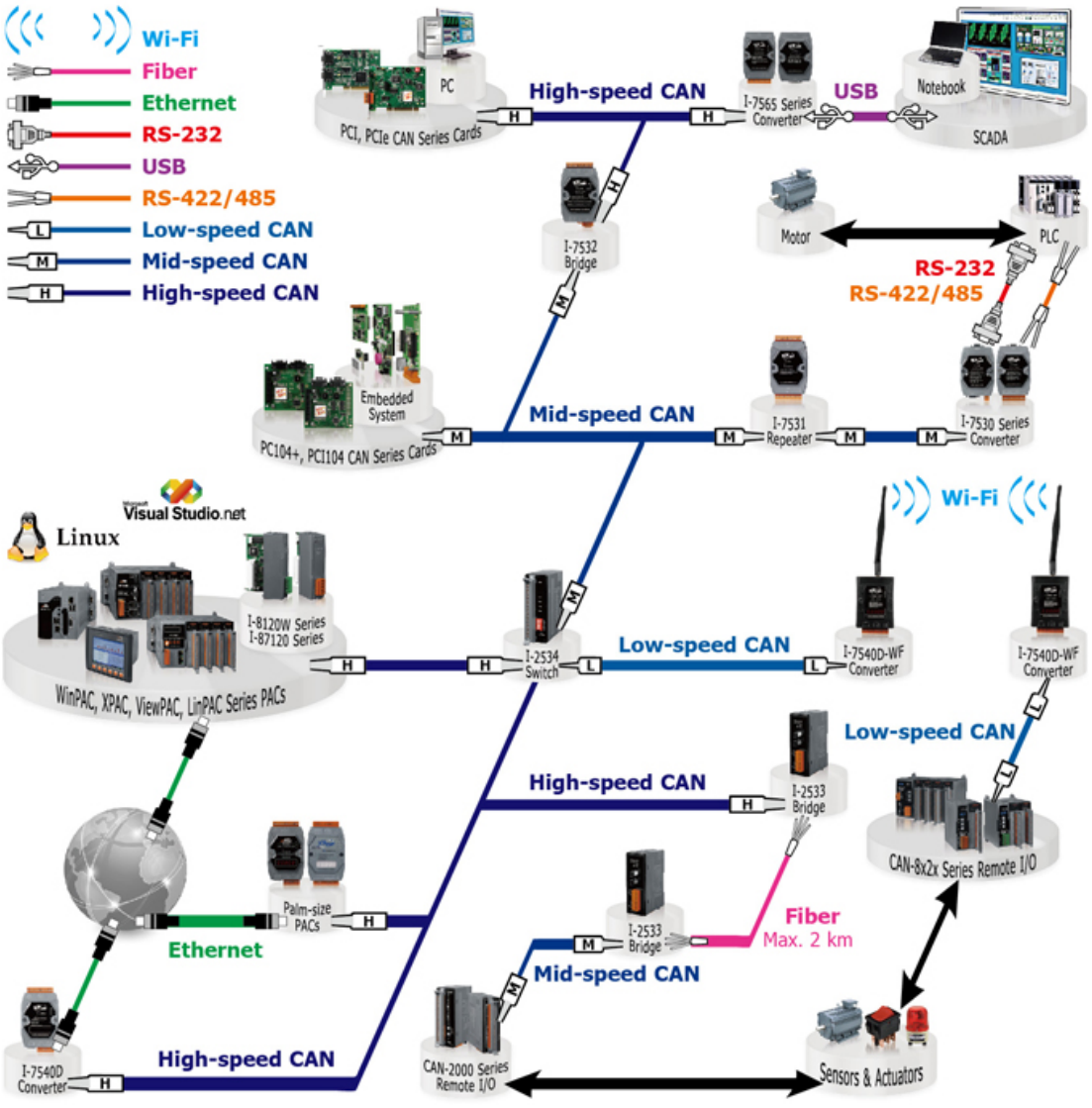
13 marca 2020

CAN – controller area network

CAN – controller area network

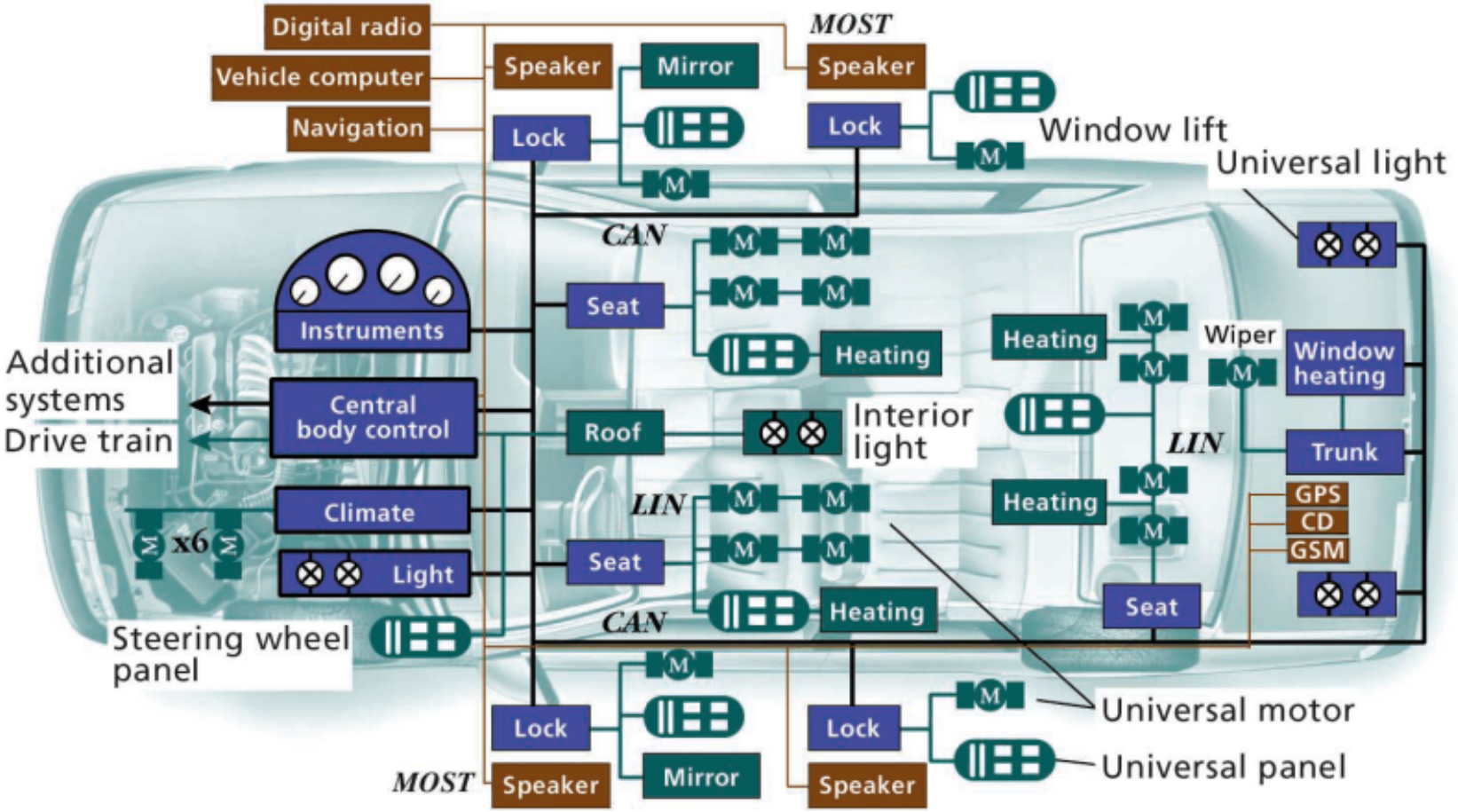


CAN – controller area network



♠ http://www.icpdas.com/root/product/solutions/industrial_communication/fieldbus/can_bus/can_intro.html

CAN – controller area network



CAN Controller area network
GPS Global Positioning System
GSM Global System for Mobile Communications
LIN Local interconnect network
MOST Media-oriented systems transport

♠ Smart Vehicles, Technologies and Main Applications in Vehicular Ad hoc Networks

CAN - specyfikacja

- transmisja typu broadcast (semi dupleks) - multi-master
- transmisja asynchroniczna, szeregowa, oparta na ramkach
- CSMA/CA - carrier sense multiple acces / collision avoidance
- detekcja/mitygacja błędów transmisji/urządzeń
- przepustowość od 10kbps (do 5 km) do 1Mbps na odległość do $\approx$ 25 m (ok. 50% zawartości ramki to dane)
- standard CAN definiuje *warstwę fizyczną i łącza danych*
  - co to jest "bit", jakie medium transmisyjne)
  - kontrola dostępu do medium (MAC) i obsługa błędów (*error detection/confinement*, kształt wiadomości/ramki)
  - na tym zbudowane warstwy obiektów (rodzaje wiadomości, filtrowanie, routing) i aplikacji (np. CANOpen, DeviceNet, etc.)

CAN – fizyczność i łącze danych

Warstwa fizyczna

Definiuje poziomy sygnałów (napięcia) oraz jaki jest stan dominujący na linii. Określa sposób kodowania poszczególnych bitów (bo nie wszędzie logiczna "1"to 5V) oraz określa synchronizację magistrali.

Warstwa łącza danych

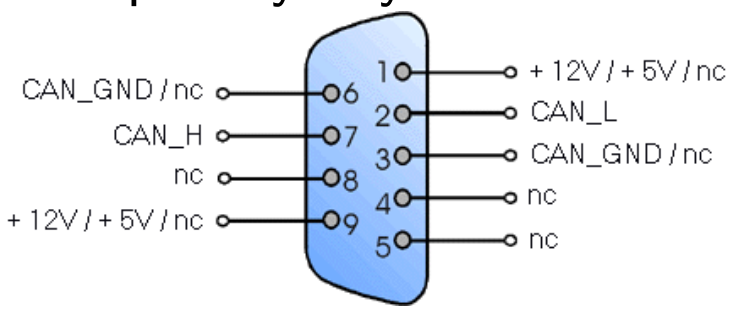
Definiuje strukturę ramki, sposoby sygnalizacji błędów i mechanizm MAC (dostępu do medium).

CAN - opis fizyczny

- wymagany jest MAC (medium-access-control) na poziomie bitowym
- bity dominujące (ich nadanie ma pierwszeństwo) i recesywne
- mechaniczne aspekty (wtyki, okablowanie) nie określone w standardzie

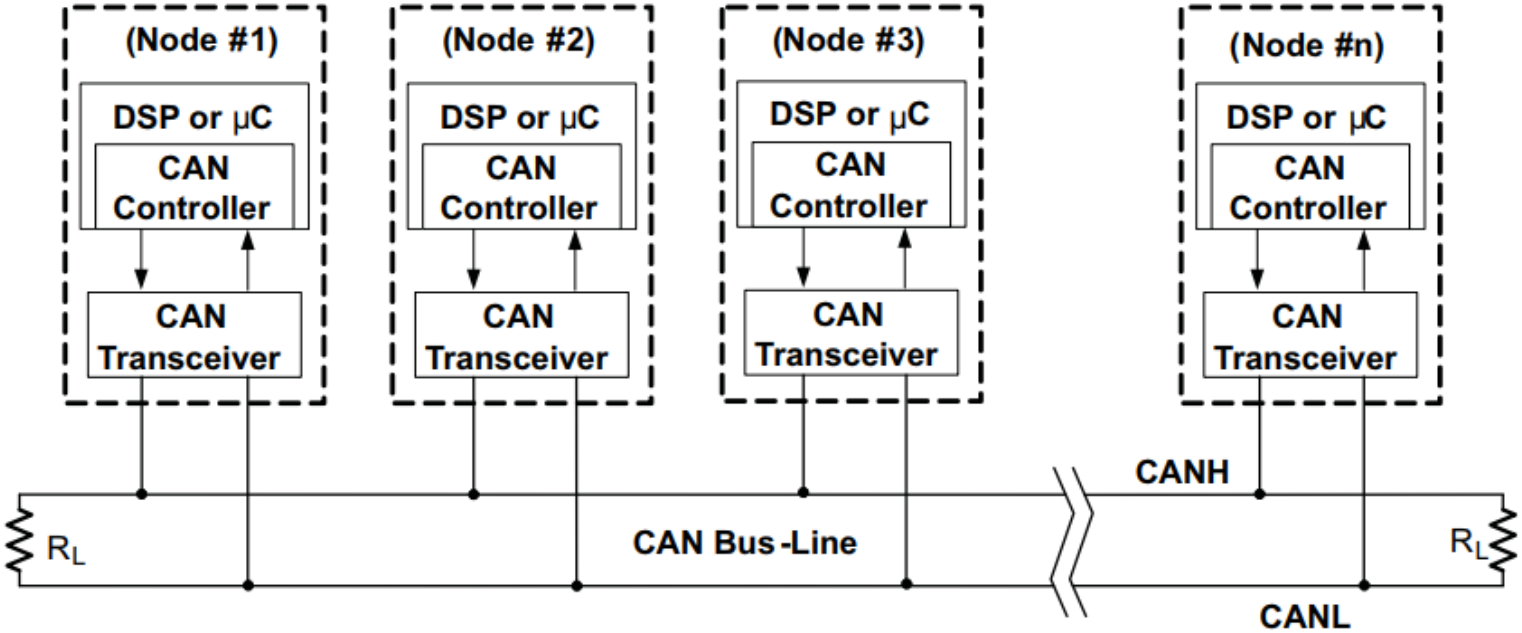
często:

- 9-cio pinowy wtyk D-sub: CAN-Lo, GND, CAN-Hi, CAN-V+



- wspólna szyna zasilająca +5V (3.3V opcjonalnie) oraz GND

CAN - magistrala



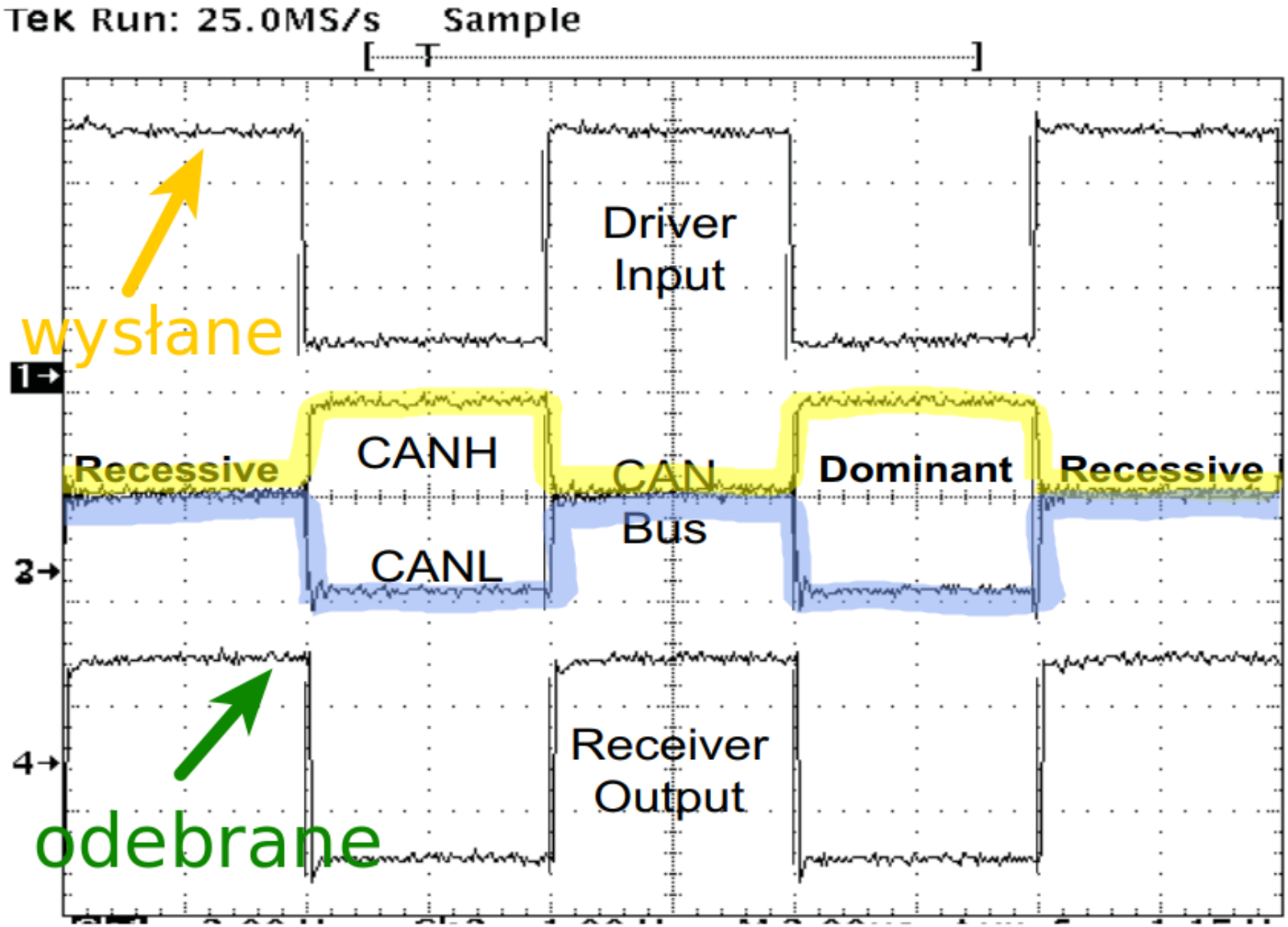
♠ Sygnalizacja różnicowa (*differential signalling*)

Sygnał "bez niczego"jest utrzymywany w okolicach 2.5V. CAN-Hi jest podciągany o 1V w górę, a CAN-Lo o 1V w dół, co daje 2V różnicy między parą linii sygnałowych.

♠ Zastanów się

Siedzicie z kolegą w dwóch zamkniętych pudłach (każdy w swoim). Łączą was dwa sznurki, które możecie napinać i luzować. Jak zbudujecie bezgłośną komunikację?

CAN - magistrala

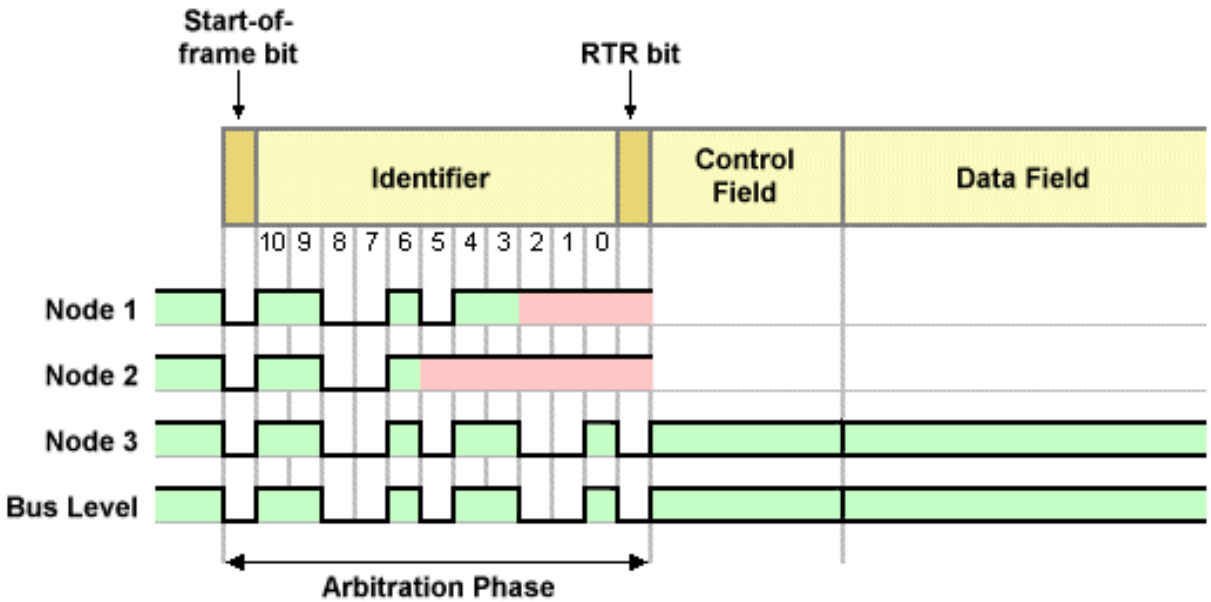


♠ Steve Corrigan: *Introduction to the Controller Area Network (CAN)*, SLOA101A, www.ti.com

CAN - magistrala

- Linia “wysłane” i “odebrane” to *logiczne bity* wysyłane i odebrane
- Linia żółta to jedna z dwóch linii magistrali CAN - “podciągana” o 1V w górę
- Linia niebieska to druga z linii różnicowych - “ściągana” o 1V w dół
- Obszar oznaczony “recessive” to okres, kiedy żadne urządzenie nie manipuluje liniami CANH i CANL - jest interpretowane jako bit **1**
- Obszar “dominant” jest przykładem wysyłania bitu **0**
- *Zauważmy*, że więcej niż jedno urządzenie na magistrali może “podciągnąć” i "ściągnąćĆANH i CANL w tym samym czasie...

CAN - arbitraż magistrali ♠



- Trzy urządzenia nadają jednocześnie. Dla bitów 10-6 wszystkie nadają to samo.
- Urządzenie 2 nadaje swój 5. bit jako "1", ale jest to bit recesywny, a inne urządzenia nadają wtedy "0". Urządzenie 2. przegrywa i powstrzymuje się od nadawania.
- Ostatecznie wygrywa urządzenie 3 (bo ma *niższy* identyfikator!)

CAN - mechanizmy wykrywania błędów

CAN zaprojektowano do zadań specjalnych. Ma działać i "naprawiać" się sam.

Wyrkywanie na poziomie wiadomości:

- CRC (15 bitów + 1 bit odstępu)
- ACK (1 bit potwierdzenia + 1 bit odstępu)
- bity recesywne: CRC-delim, ACK-delim, SOF, EOF: muszą być, ale łatwo je ktoś może zniszczyć, jeśli działa źle

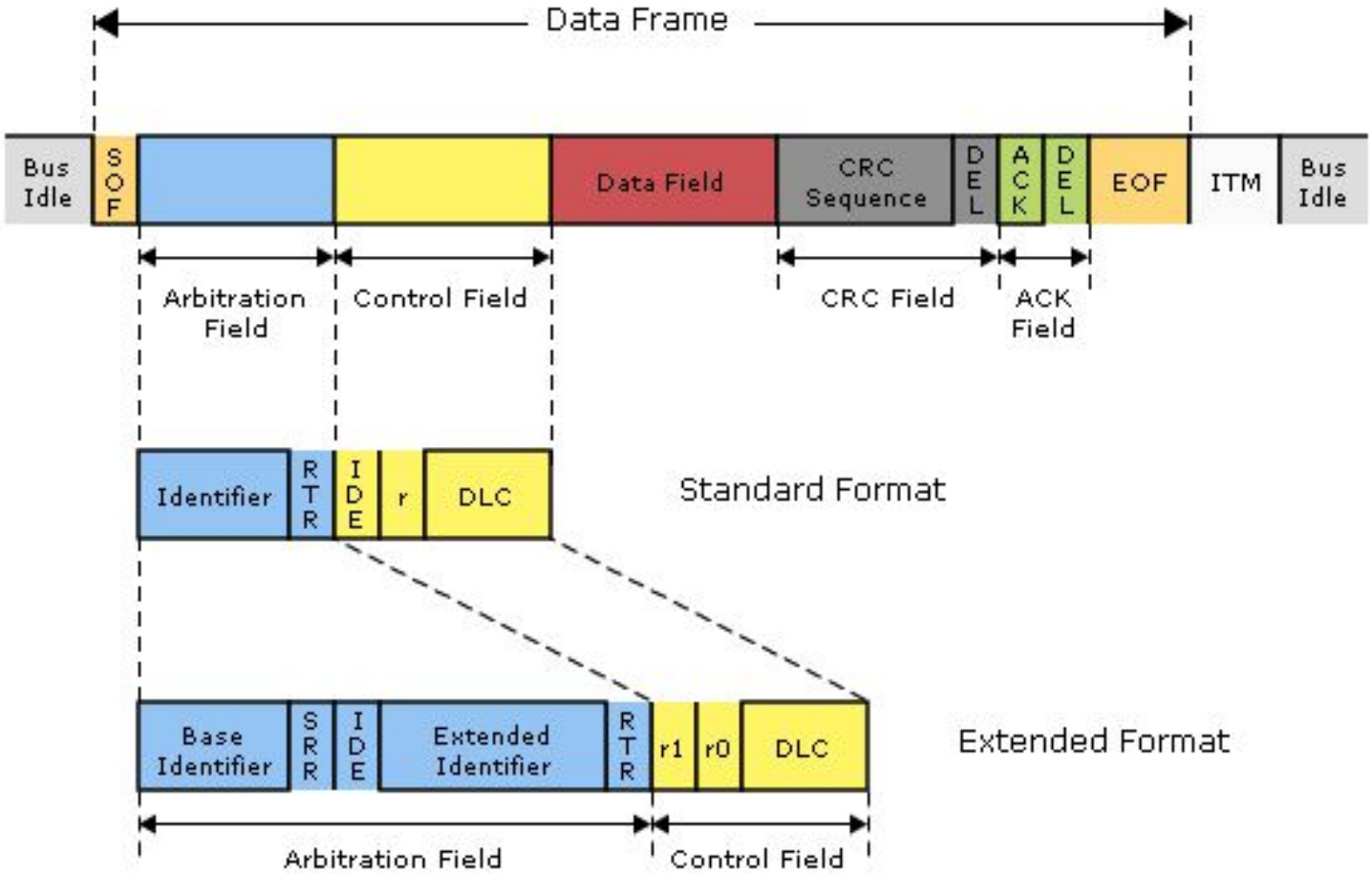
Wykrywanie na poziomie bitów:

- urządznia cały czas monitorują magistralę danych, sprawdzając niezgodności (np. brak ACK)
- bit stuffing (opis poniżej)

CAN - łącze danych

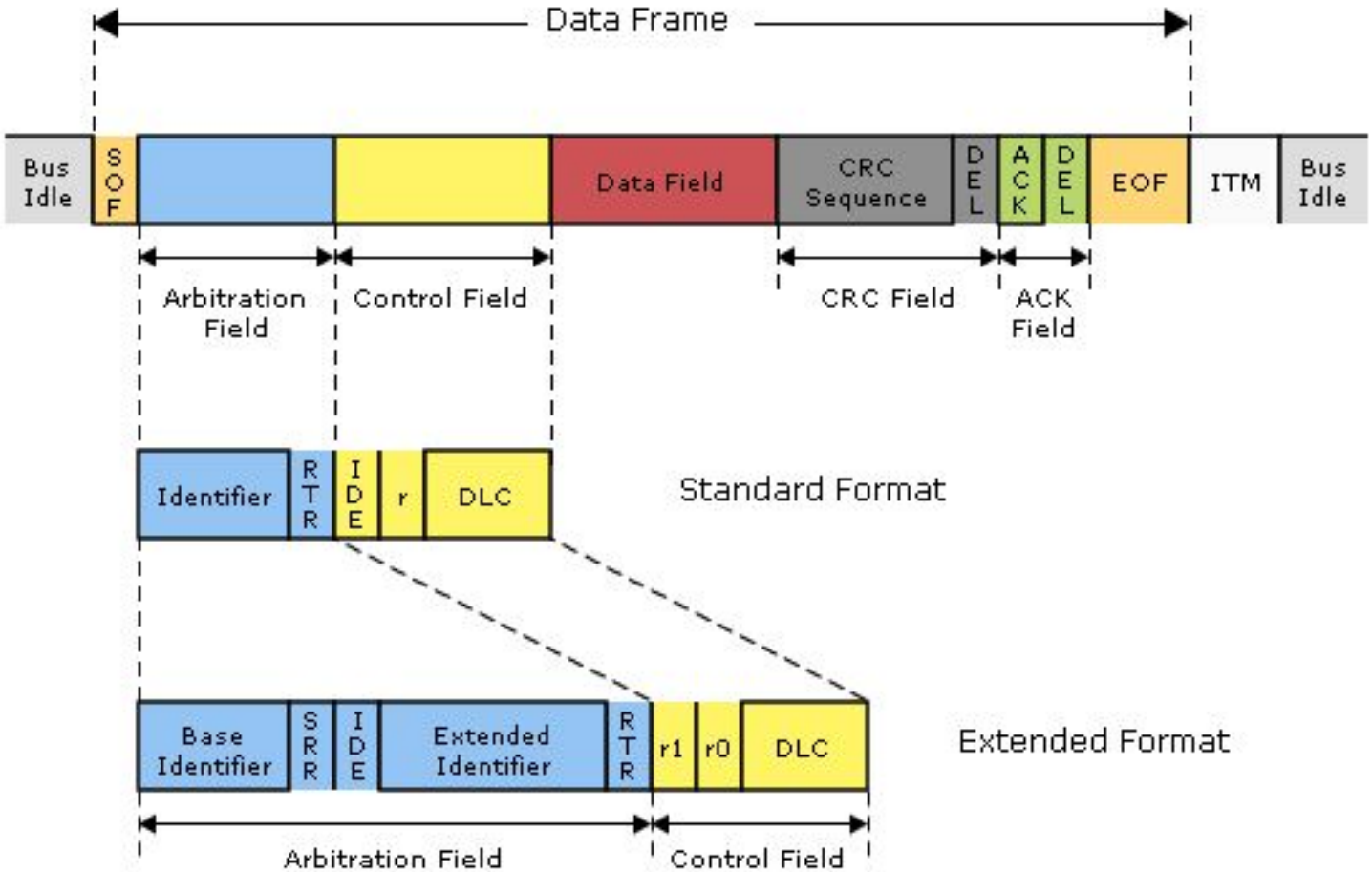
Typy ramek:

- danych – faktycznie przesyła dane
- zdalnego wywołania – żądanie przesłania danych
- błędów – w przypadku wystąpienia błędu
- przepełnienia – do uzyskania opóźnienia między ramką danych lub zdalnego wywołania



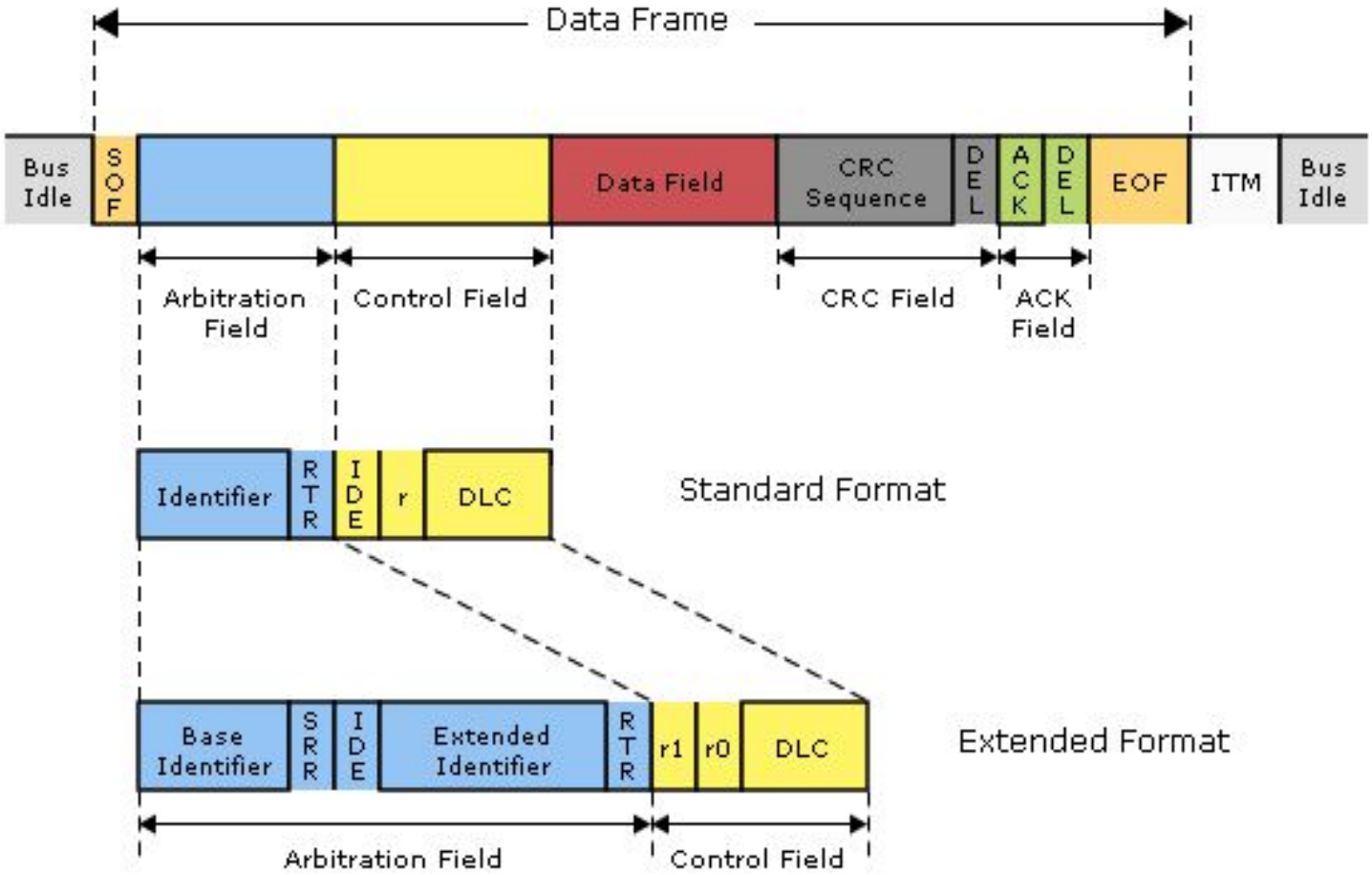
Format standard

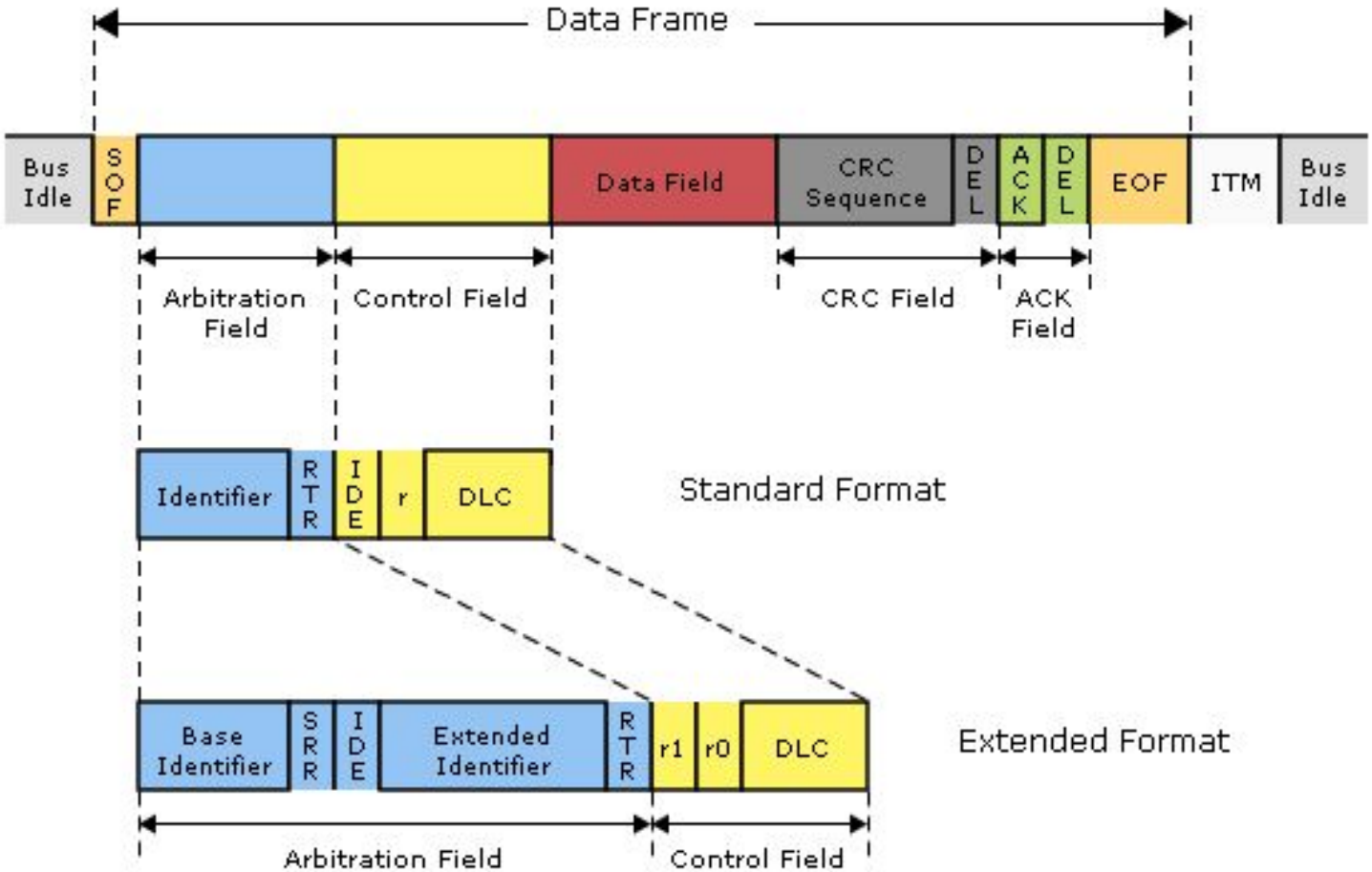
RTR	*remote transmission request* – 0 dla ramek danych, 1 dla zdalnego wywołania
IDE	*id extension* – 0 dla ramek formatu podstawowego, 1 dla rozszerzonego
DLC	*data length* – długość danych (0-8 bajtów)
CRC-del	recesywny bit (2)
ACK-slot	odbiorca ma szansę wysłać 0 (dominujący), bo nadawca wysyła 1 (recesywny)
ACK-del	recesywny bit (1)
EOF	*end of frame* - recesywny bit (1)



Format extended

SRR	*substitute remote request* – recesywny bit (1)
IDE	*identifier extension bit* – recesywny bit (1)


Data Frame
Bus Idle
SOF
Data Field
CRC Sequence
DEL
ACK
DEL
EOF
ITM
Bus Idle
Arbitration Field
Control Field
CRC Field
ACK Field
Identifier
RTR
IDE
r
DLC
Standard Format
Base Identifier
SRR
IDE
Extended Identifier
RTR
r1
r0
DLC
Extended Format
Arbitration Field
Control Field



Ramka zdalna

RTR $= 1 \rightarrow$ przegrywa arbitraż kiedy jednocześnie jest ramka danych z tym samym identyfikatorem. To ma sens, bo jeśli wywołuję kogoś, kto właśnie chce się dostać na magistralę, to jest szansa, że zaraz usłyszę coś, o co miałem poprosić.

Ramka błędu

Podstawowa ramka błędu

Składa się z 6 bitów dominujących (ERROR FLAG) oraz 8 bitów recesywnych (ERROR DELIMITER). Bity dominujące (dla ERROR FLAG) wysyła stacja będąca aktywną (tj. dla której nie określono błędnego działania). Bity recesywne wysyła stacja w stanie *passive error*.

Jest to świadome naruszenie zasady bit-stuffing (poniżej). Każdy węzeł, który rozpoznaje taki błąd, wysyła na magistralę swoją ramkę błędu.
Dlatego w sumie bitów dominujących (ERROR FLAG) może być 6-12, plus 8 bitów recesywnych.
Po czym następuje przerwa międzyramkowa (3 bity).

Error confinement

CAN ma wbudowany mechanizm odłączania się niedziałających urządzeń *już na poziomie protokołu komunikacji*. Każde urządzenie ma liczniki TxE, RxE (błędy przy nadawaniu, odbiorze).

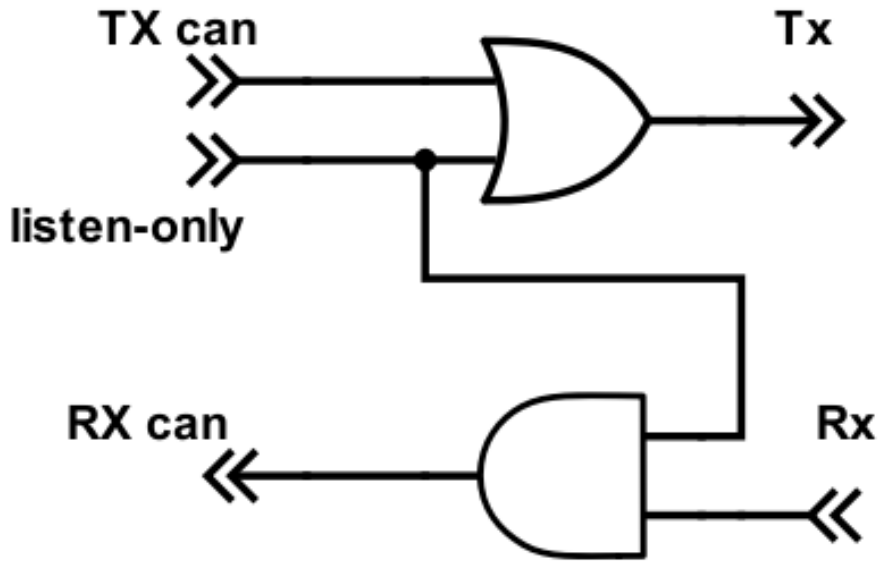
1. błąd przy nadawaniu $\rightarrow$ TxE += 8 (TxE -=1 jeśli bezbłędnie)
2. błąd przy odbiorze $\rightarrow$ RxE += 1 (RxE -=1 jeśli bezbłędnie)
3. TxE $>$ 127 lub RxE $>$ 127 $\rightarrow$ urządzenie wchodzi w stan *passive error*
4. TxE $>$ 255 lub RxE $>$ 255 $\rightarrow$ urządzenie wchodzi w stan *bus-off*
5. po RESET oraz poprawnym przesłaniu 128 x 11 bitów $\rightarrow$ stan *active error*

Auto-detekcja *bitrate* (prędkości nadawania)

Możliwości:

1. jest jakiś domyślny (default) bit-rate
2. automatyczna detekcja
   - pomiar czasu trwania pojedynczego bitu (jak znaleźć pojedynczy bit?)
   - cichutkie podsłuchiwanie (Ale niektórzy nie potrafią...)

Listen-only dla nieumiejących ♠



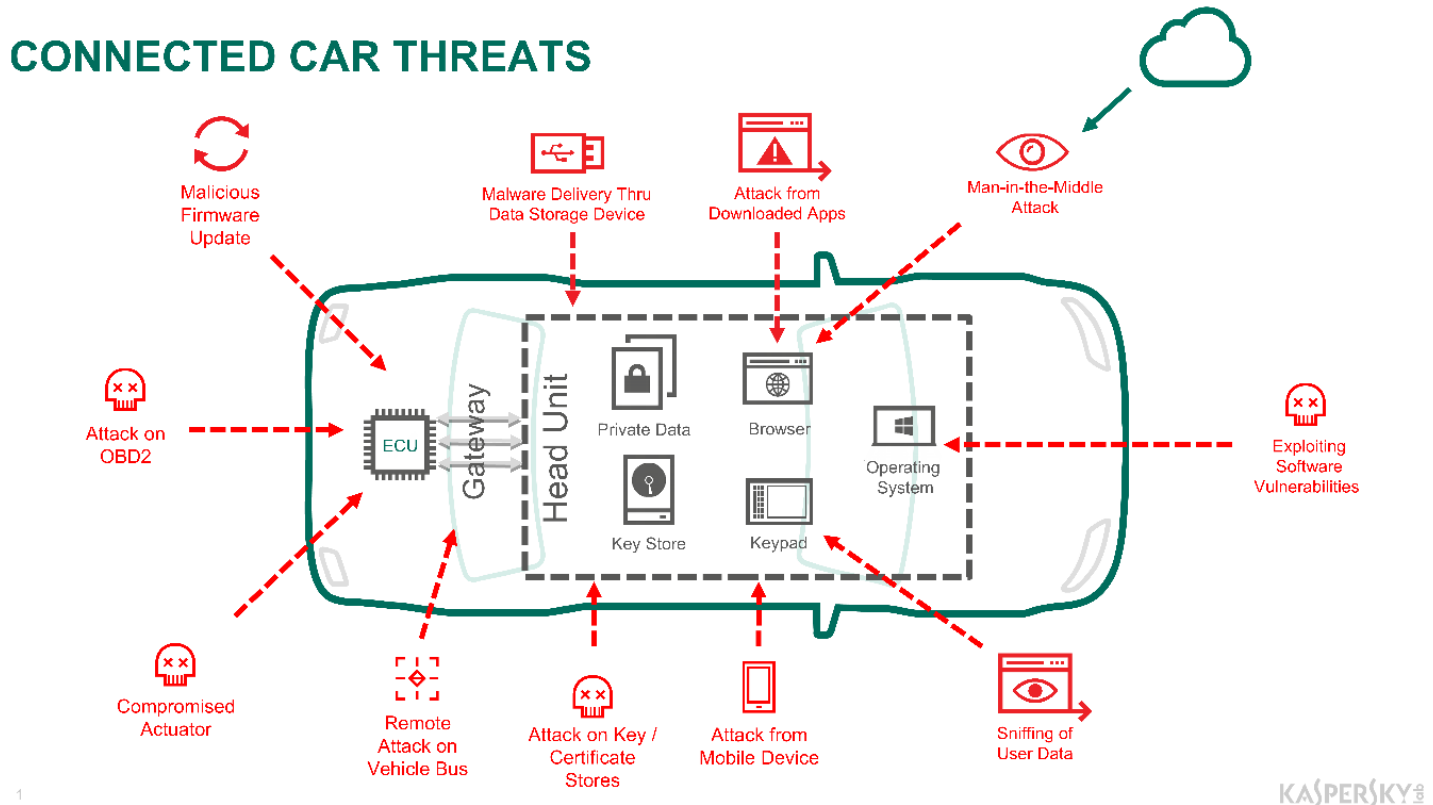
listen-only $= 1$ wchodząc do OR powoduje zawsze nadawanie bitu 1. Jednocześnie przy odbiorze nie zagłusza strumienia bitów wejściowych (bramka AND).

Bit stuffing ♠

Po każdych 5 takich samych bitach automatycznie wtawiany jest bit przeciwny:

- by utrzymać synchronizację między węzłami
- nie dotyczy pól CRC, ACK
- 6 taki sam bit $\rightarrow$ błąd aktywnej stacji

Bezpieczeństwo...



♠ https://www.itnewsbuzz.com/ ↩

↪ /nmw2017-kaspersky-avl-software-functions-gmbh-pave-way-for-secure-by-design-connected-cars/

♠ www.kaspersky.com/blog/connected-car-apps-revisited/18548/