Budapesti Műszaki- és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Rendszertervezés házi feladat

Autós Body rendszer tervezése

Bartakovics Tamás

Békéssy László

Horváth Zsolt

Kelemen Tibor

Kővári Balázs

Patonai Balázs

Sass Péter

KONZULENS

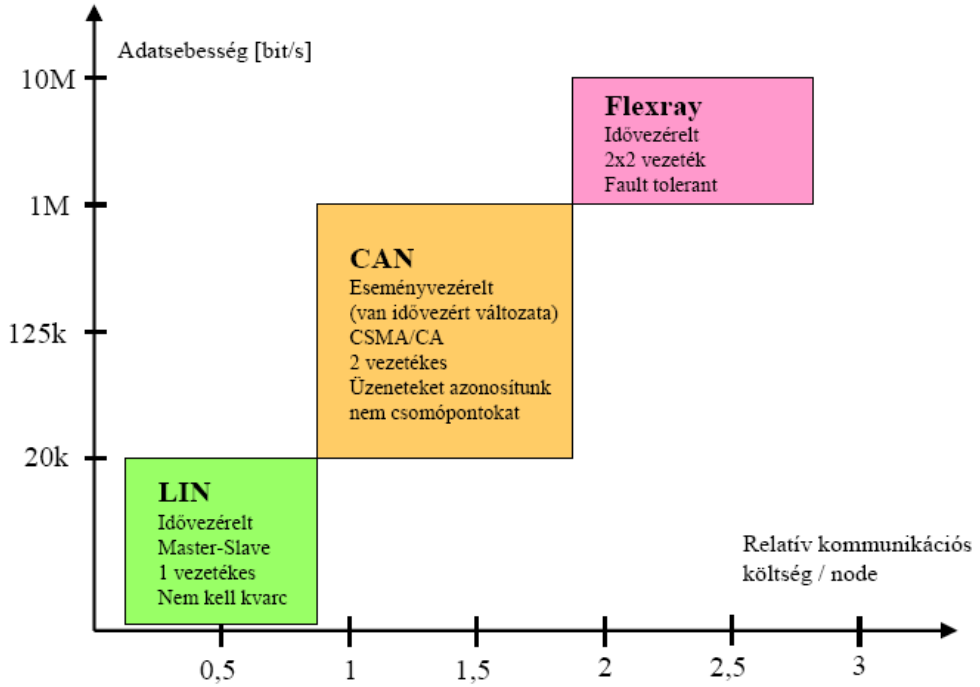
Scherer Balázs

BUDAPEST

2009

Bevezetés

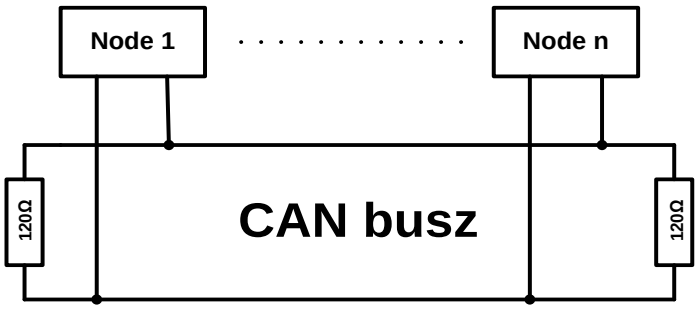
Mára, a XXI. században az autók szinte minden egységének működését ún. ECU-k – Electronic Control Unit – vezérlik, szabályozzák. Egy átlagos mai autóban 60-80 ECU van, melyek egymással szoros együttműködésben állnak. Szükségszerű például, hogy a motorvezérlő tudjon a jármű sebességéről, a környezeti hőmérsékletről és arról, hogy melyik sebességfokozatban van a váltó. Az ECU-knak kommunikálniuk kell egymással, ahhoz, hogy feladatukat maradéktalanul elvégezhessék, a kommunikációs protokoll pedig meg kell feleljen az autóipar szigorú előírásainak. A különböző ECU-k ma már általában CAN, LIN esetleg FlexRay protokollon keresztül kommunikálnak egymással. A három protokoll összehasonlítása az *1.ábrán* látható.



1. ábra – Autóipari hálózatok összehasonlítása

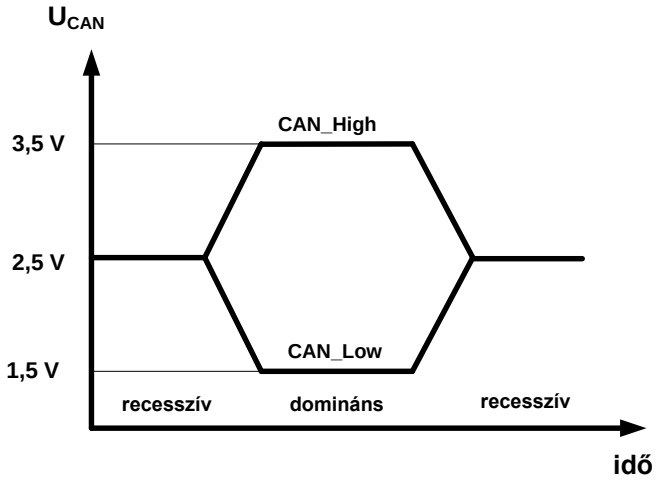
CAN (Controller Area Network)

A Robert Bosch GmbH 1983-ban kezdte el kidolgozni a CAN kommunikációs protokoll alapjait, melynek első verzióját 1986-ban publikálták. A protokoll 1993-ban került ISO szabványosításra, piaci megjelenése is nagyjából ehhez az évszámhoz köthető. Az elnevezés egyes források szerint kezdetben a Car Area Network szókapcsolatra utalt, manapság azonban a CAN eszközök csupán egyharmada található az autókban, így az elnevezés megváltozásáért valószínűleg más iparágak érdeklőse a felelős. A maradék kétharmad orvosi műszerekben és ipari automatizálási berendezésekben található.
A hálózat topográfiája tetszőleges, de általában busz elrendezést használnak. A *2.ábrán* a tipikus CAN busz topológia látható. A több master egység miatt többszörös hozzáférésre, nem destruktív üzenetkezelésre (CSMA/CA) van szükség. Az üzenetek prioritással rendelkeznek, azaz a fontosabb üzenetek előbb kerülnek kiküldésre. Ezt huzalozott ÉS kapcsolattal oldották meg. Az elérhető maximális adatsebesség *1 Mbit/s*, az áthidalható távolság *40-500 m*, mely a kívánt sebességtől függ.



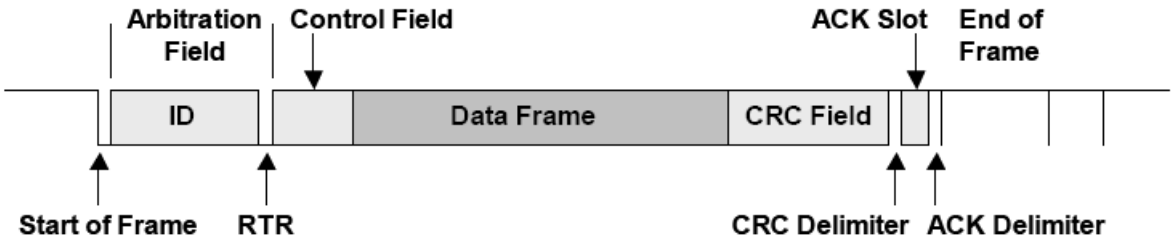
2. ábra – CAN busztopológia

A CAN buszon tipikusan csavart érpáron keresztül differenciális jelátvitel történik. A *3. ábrán* a CAN fizikai rétege van ábrázolva.



3. ábra – CAN fizikai rétege

NRZ bitkódolással rendelkezik bitbeszúrással, illetve bitkiejtéssel. Rövid, változó hosszúságú keretei vannak: 0-64 bit hosszú adatmező, 0-8 adatbyte. A CAN protokollnak négyféle keretformátuma van: adatkeret (Data Frame), hibakeret (Error Frame), távoli keret (Remote Frame), túlcsordulás keret (Overload Frame). A CAN üzenet keretformátuma a *4.ábrán* látható.



4. ábra – CAN keretformátuma

Az adatkeret jelentése, hogy „itt egy keret, aki akarja használja fel”. Az adatkeret mezői a következők: keret kezdetét jelző Start of Frame bit, *11 bit* hosszúságú ID mező, *1 bit* RTR (Remote Transfer Request), *4+2 bit* hosszú control mező – mely az adatmező hosszát adja meg -, *0-64 bit* hosszú adatmező, *15 bit* CRC mező, illetve a CRC Delimiter, ACK (Acknowledge) Slot, ACK Delimiter, EOF (End of Frame), melyek mindegyike *1 bit*.

Az arbitráció az ID+RTR alapján történik, huzalozott ÉS kapcsolaton alapul. A versengő állomások bitről bitre egyszerre hajtják meg a buszt, és az összehasonlítja az adott és vett biteket. Ha a kettő különböző, akkor abbahagyja az adást. Teház az a node kapja meg a buszt, melynek az arbitrációs mezőjében először szerepel 0.

A hibakeret jelentése, hogy a keret hibás. *6 db* azonos értékű bitből, és *8 db* 1 értékű Error Delimiter bitből áll. Ha 0, akkor Error Active, ha 1 akkor Error Passive állapotról van szó. Az Error Passive állapot fokozottabb, elővigyázatosabb állapotot jelent. A CAN ötféle hibadetektálást definiál:

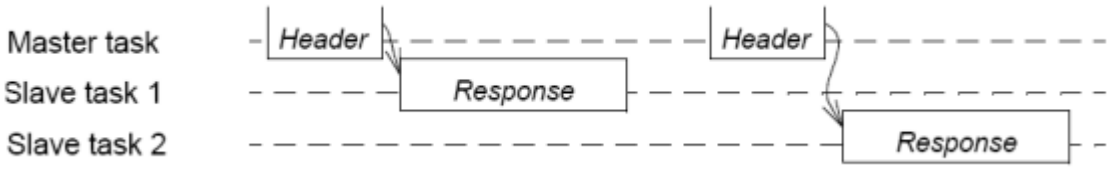
- Bit Monitoring: ha az adott és a vett bit nem egyezik (arbitráción kívül),
- Bit Stuffing: ha a bitbeszúrás/bitkiejtés sérül,
- Frame Check: ha a keret rögzített bitjei eltérnek,
- Acknowledgement Check: ha nem történt nyugtázás,
- Cyclic Redundancy Chec: CRC hiba esetén.

A Remote Frame-mel egy node egy másik node-t tud felszólítani üzenet elküldésére. Ez abban különbözik az adatkerettől, hogy az RTR bitet recesszívbe kell állítani, ezért a Remote Frame kisebb prioritású a Data Frame-nél.

Az Overload Frame-et általában a slavek küldik, abban az esetben, ha túlterheltek és időre van szükségük a következő üzenet fogadásához.

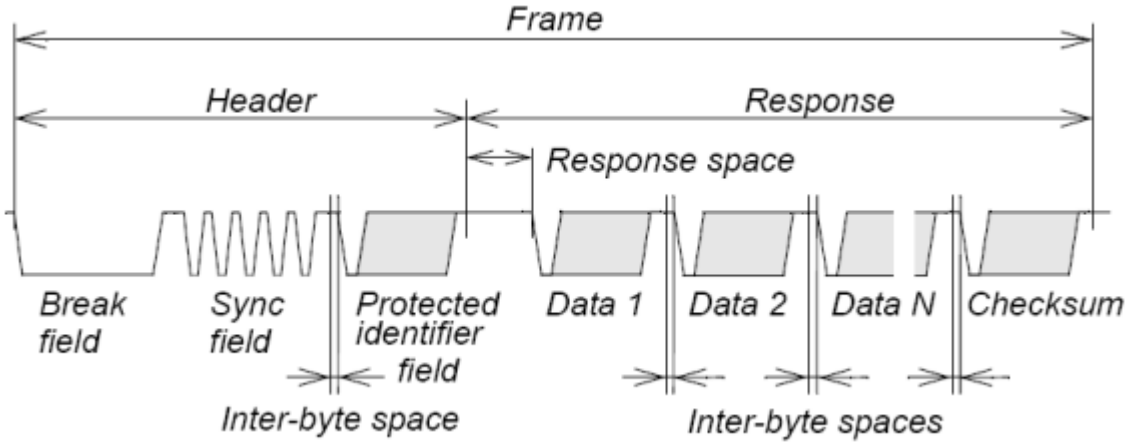
LIN (Local Interconnect Network)

A LIN hálózat egy egyszerű, kis sebességű autón belüli – azaz onboard – kommunikációs hálózat. Megalkotásánál a fő cél az volt, hogy olcsóbb legyen a CAN-nél, illetve a többi hasonló célú protokollnál. Gerinchálózatként általában a CAN-t használják. Első verzióját a Motorola cég dolgozta ki 1999-ben. Ennek sikerén felbuzdulva 2000-ben létrehozták a LIN Konzorciumot az Audi, a BMW, a Daimler Chrysler, a Volkswagen, illetve a Volvo közreműködésével. Az első elterjedt verzió az 1.2 volt, melyet 2000 év végén publikáltak. A fizikai réteg javítását hozó 2.0 verziót 2003 szeptemberében publikálták, míg a legújabb 2.1 verzió lényegi változást nem hozott, csak a dokumentáción javítottak. A LIN szabványok az interneten ingyenesen hozzáférhetők.
A LIN hálózat aszinkron soros kommunikációra épül. Egyvezetékes kommunikáció, félduplex átvitellel. Az üzeneteket mindig a master indítja, a slave node-ok csak a master kérésére használhatják a buszt. Egy master, illetve több slave node található egy hálózatban. Működés szempontjából a rendszerben egy master task és több slave task létezik. A master mindkét títpusú taszkot végrehajthatja, míg a slave-ek csak slave taszkokat hajthatnak végre. Azt, hogy mikor milyen üzenet kerül elküldésre, a master taszk szabja meg. A masterben van egy ütemező tábla, hogy mikor melyik slave-et kell lekérdeznie. Ez determinisztikussá teszi a működést, így nincs ütközés, versengés a busz használatáért.
Az adatok itt is keretben kerülnek elküldésre. A keret a master által kiadott headerből, illetve a slave válaszként kiadott adatból áll. Ez látható az *5. ábrán*.



5. ábra – LIN hálózat kommunikációja

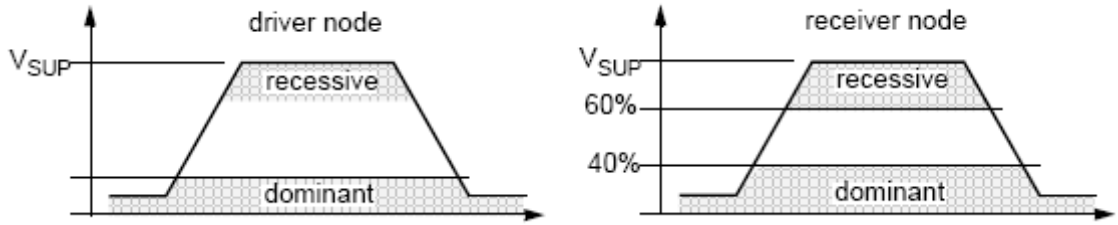
A header egy breakből, egy szinkronizációs szekvenciából, és egy azonosítóból áll. A megszólított slave válasza adatmezőből, illetve checksum mezőből tevődik össze. A LIN keret felépítését mutatja a *6. ábra.*



6. ábra – LIN keret felépítése

Egy üzenet célját az ID egyértelműen azonosítja. Broadcast rendszer, azaz egy üzenetet akár több node is használhat. A maximálisan elérhető azonosítók száma 64. A LIN nincs olyan nyugtázás, mint a CAN-ben, a nyugtázást tulajdonképpen az adatmező meglétével lehet azonosítani.

A LIN kommunikáció sebessége *1-20 kbit/s* lehet, a maximális vezetékhossz 40. Bitkódolása NRZ, a domináns érték a 0, a recesszív érték az 1. A LIN jelszintek a *7. ábrán* láthatóak.



***7.* ábra – LIN jelszintek**

A feladat megvalósítása:

Az első ajtók vezérlése CAN buszon keresztül történik, míg a hátsó ajtóké LIN buszon. A kormányon lévő gombok megnyomásával CAN üzeneteket küldünk a buszon keresztül. Az egyes ajtókat a tesztösszeállításunkban 1-1 MITMOT reprezentálja. Két MITMOT az első ajtókat, kettő a hátsó ajtókat, illetve egy a CAN-LIN gateway-t. Az első ajtókat reprezentáló kártyákra a DOORx_CAN_slave projekt került letöltésre, míg a hátsó ajtókat reprezentáló kártyákra a DOORx_LIN_slave nevű projekt. A gateway-t megvalósító kártyára a CAN_LIN_gateway nevű projekt került. A feladat megvalósításához és teszteléséhez a tanszéken tervezett MITMOT-ot használtuk. A programkódok megírása során felhasználtuk a tanszéken előre megírt api-kat.

DOOR1_CAN_slave:

Ennek a programnak az a feladata, hogy a CAN buszon érkezett üzenetek alapján magára ismerjen az egység és bekapcsolja a központi zárat illetve fel- vagy lehúzza az ablakot. A központi zár egy egységes paranccsal működik, hiszen valamennyi slave-nek egyszerre kell ki- és bekapcsolnia a központi zárat. Az ablakemelés viszont slave-nként külön történik, így először azonosítani kell a megcímzett slave-t. Az a kormány által küldött üzenet 4. bájtjában találhatóak a kormány nyomógombjairól szóló információk. A felső négy bit az ajtó azonosítója tehát azt jelenti, hogy melyik ajtón akarjuk fel- vagy lehúzni az ablakot. A harmadik bit az ablak leengedését jelenti, a negyedik bit az ablak felhúzását. A második bit pedig a központi zár aktiválását.
CAN üzenet vételekor az egyes slave-k ellenőrzik, hogy központi zár aktiválását kérik az üzenetben illetve, hogy nekik szól-e az üzenet. Ha neki szól az üzenet, ő az aktív slave akkor figyeli, hogy fel- vagy lehúzásról szóló információt kap. Központi zár aktiválását kérő üzenetben azonnal be- vagy kikapcsolja a központi zárat.
A központi zár állapotát a slave-ken a 4-es számú LED szimbolizálja. Az ablakemelő motor vezérlése úgy történik, hogy egyik bemenetével aktiváljuk, elindítjuk a motort, egy másik bemenetén keresztül pedig a forgás irányát tudjuk állítani. Ezt a motorvezérlést szimbolizálandó a slave-n található 3-as LED a motor engedélyezését jelzi, a 2-es LED arra utal, hogy az ablakot leengedjük, az 1-es LED pedig arra, hogy felhúzzuk.

CAN_LIN_gateway

Ennek a programnak az a feladata, hogy a CAN buszon érkező üzeneteket LIN buszra továbbítsa. Feladata, hogy a kormányról érkező üzenetek 4. bájtja alapján azonosítsa, hogy melyik ajtónak szól az üzenet, ha valamelyik hátsó ajtónak (azaz LIN buszon lévő ajtónak) akkor az üzenetet továbbítja a LIN buszra, különben ne tegyen vele semmit. Mint azt már fent leírtuk az azonosítás a 4. bájt felső 4 bitje azonosítja az ajtót. A központi zárral kapcsolatos üzeneteket valamennyi slave-nek továbbítja.

DOOR2_LIN_slave

Ez a program a LIN üzenet alapján felemeli vagy leengedi az ablakot, illetve kezeli a központi zárat. Mindig csak az a slave kap üzenetet a gateway-től amelyiknek mozgatnia kell az ablakát. A központi zárra vonatkozó üzeneteket mindkét hátsó ajtó megkapja. A slave-n található 3-as LED a motor engedélyezését jelzi, a 2-es LED arra utal, hogy az ablakot leengedjük, az 1-es LED pedig arra, hogy felhúzzuk. A 4-es LED a központi zárat reprezentálja. Ebben az alkalmazásban a központi zár felfutó élre érzékeny. Tehát a központi zár állapota minden felfutó él érzékelésekor megváltozik.