1 Járműipari hálózatok

1.1 Járműipari kommunikációs technológiák összehasonlítása

A járműelektronika viharos fejlődése az utóbbi évtizedekben egyre több új, elektronikusan támogatott funkció megjelenését eredményezte. A vezérlőkészülékek, valamint az egyéb járműelektronikai berendezések rohamos fejlődése a 70-es évek végén kezdődött és a precíz adatfeldolgozáson alapuló végrehajtás nagymértékű javulást eredményezett a gépjármű üzemében. A mai számítógépek struktúrájához hasonló, digitális jeleket feldolgozó mikrovezérlők működési paraméterei a kezdeti mechanikus, majd elektromechanikus rendszerekkel egyáltalán nem, de még a digitálist megelőző analóg, vagy hibrid kialakítású elektronikus vezérlők és szabályozók hatásosságával is nehezen vethető össze. A digitális rendszerek óriási előnye, hogy nagymennyiségű, jórészt próbapadi mérésekből származó (jellegmezők) adat tárolására képesek, így kifogástalan vezérlések és szabályozások valósíthatók meg a működő motorok majd minden üzemállapotában, úgy a gyújtás, mint a keverékképzés területén.

Először a korszerű motormenedzsment, majd az aktív és passzív biztonsági elektronika, végül a karosszéria, a komfort és a kommunikáció vezérlői jelentek meg a járművekben. Az elektronikával felszerelt korszerű gépjárművek egyre magabiztosabban teljesítették a szigorodó környezetvédelmi követelményeket, valamint gazdaságosabbá, biztonságosabbá és komfortosabbá váltak.

A gondosan elkészített mikroprocesszoros egységek eleinte autonóm módon működtek, vagyis saját érzékelőikkel és végrehajtóikkal tartottak kapcsolatot. A sokasodó és egymást átfedő funkciók (pl. az automatikus váltóműködés alatti motornyomaték redukálás, vagy ASR és motorvezérlő együttműködés, stb.) megjelenése az autonóm egységek egymással való kommunikációját, azaz egyre több adat és információ cserét igényelt. A biztonsági, komfort és kényelmi (pl. telekommunikációs) berendezések szakadatlan korszerűsítése ezt az igényt tovább fokozta, ezért a digitális vezérlők együttműködése is egyre szorosabbá vált.

A kapcsolatokat fizikailag megvalósító csatlakozók, valamint kábelerek száma törvényszerűen növekedett, s ez sok problémát okozott.

A korábban egységesen, de döntően még ma is használatos megoldás az, hogy a villamos és elektronikai egységeket szigetelt kábelerekkel csatlakoztatják egymáshoz, melyeket sablonokban gondos tervek alapján még a beszállítónál kábelkorbácsokká alakítanak és a járműgyártónál ezt szerelik, napról-napra körülményesebben, a karosszéria megfelelő üregeibe.

A nehezen kezelhető, nem csak súlyra, de méterre is tetemes (egy felsőkategóriás luxus járműnél csupán a kábelezés ≈100 kg tömegű, míg teljes hossza megközelítheti a 2-2.5 km értéket is) „kábeldzsungel" a sok vezetőér és csatlakozó miatt egyre megbízhatatlanabbá válik. Mindezt tetézi, hogy a nagy járműgyártók nem csak egyféle, hanem különböző modellsorozatokat készítenek, melyekhez sokszor több száz féle kábelkialakítást használnak a gyártósorokon. A kábelkötegek előállítása, szerelése, javítása is egyre költségesebb, időigényesebb, s ha mindez még a megbízhatóság csökkenését is eredményezi (csatlakozók kontakthibái, kötéslazulások, vezetéktörések stb., vagyis érintkezési bizonytalanságok miatt), akkor ez a helyzet, mely a 20.-ik század utolsó évtizedeire kialakult, sürgős felülvizsgálatra szorul.

A járműelektronikákat tervező szakemberek elvileg két út közül választhattak, vagy megpróbálják az egyes részterületekért felelős vezérlők funkcióit egyetlen központi vezérlő számítógépben egyesíteni és ezzel legalább a közöttük szükséges külső villamos kapcsolatokat megspórolni, vagy a sok-sok kábelér helyett minél kevesebb, lehetőleg egyetlen összeköttetést (buszvonalat) használni a szükségszerűen soros formátumba rendezendő információk továbbítására.

A legtöbb szakember a második utat tartotta járhatónak. Kezdetben tehát majdnem minden ismert márka képviselői (elsősorban a németek, a franciák és az amerikaiak) megpróbáltak valamilyen saját szabvány szerint működő buszrendszert összeállítani, és fejlesztéseiket a lehető leghamarabb nemzetközileg is elismertetni. Az akkoriban még lefedetlen, óriásinak ígérkező piactól ugyanis az elsőként érkező tetemes hasznot remélhetett.

A nemzetközi szabványosítás azonban nehéz feladat, így összesen négy megoldás maradt talpon, a VAN, a J1850 SCP, a J1850 DLC és az általunk ismertetni kívánt CAN rendszer, melyet a járműelektronika tervezésével és gyártásával foglalkozó szakemberek (többéves munkával, 1983-ra) elviekben kidolgoztak.

Az járműiparban a folyamatos a biztonsági, kényelmi, környezetvédelmi és minőségi elvárások növekedése. A fejlődésben komoly szerepet kapnak az elektronikus vezérlőegységek (ECU), melyek egymással történő kommunikációja régóta nélkülözhetetlen funkció a járművekben. Ez a folyamatos fejlődés megkívánja a kommunikációs technológiák megbízhatóságának, adatátviteli sebességének és az adatok mennyiségének növelését is. A fejlett szabályozó és biztonsági rendszerek (beleértve a különféle érzékelőket, beavatkozókat és vezérlőegységeket) igényeinek nagy részét a jelenleg használt Controller Area Network (CAN) szabvány is kielégíti, azonban a növekvő sávszélesség igény miatt ma már több CAN hálózat (akár 5-7) is szükséges egy átlagos gépjárműben. Mindezen okok miatt egyre komolyabb igény mutatkozott egy új technológia iránt, amely kielégíti az új igényeket. A beszállítók és az autógyártók végül a 2000-es évek elején megalakították a FlexRay konzorciumot és kifejlesztettek egy új technológiát, amely először 2008-ban jelent meg sériagyártású járműben.

A FlexRayt jelenleg a legmagasabb technológiai színvonalat képviselő rendszerek esetében használják, és még jó ideig nem fogja teljese mértékben kiszorítani a másik két fontos járműipari kommunikációs technológiát, a CAN-t és a LIN-t. Ennek okai a költségek optimalizálásában és az átállás nehézségében keresendőek. A 1. táblázat nagyvonalú összehasonlítást ad a három említett technológiáról.

1. táblázat: Kommunikációs technológiák összehasonlítása

Technológia	LIN	CAN	FlexRay
Sebesség	40 kbit/s	1 Mbit/s	10 Mbit/s
Költségek	$	$$	$$$
Vezetékek száma	1	2	2 vagy 4
Alkalmazási terület	Kényelmi berendezések (motoros tükrök, klímaberendezés,	Hajtáslánc (motor, váltó, ABS stb.)	Magas technológiai színvonalú hajtáslánci

	elektromos ülések stb.)		elemek, biztonsági berendezések („Drive-by-wire”, aktív felfüggesztés, adaptív tempomat)

Ennek alapján elmondható, hogy a műszakilag és gazdaságilag legoptimálisabb rendszereket a három technológia együttes használata adja.

1.2 LIN

A járműipari hálózatok 1980-as évekbeli felfutásával egy időben jelentek meg a gépjárművekben a szabványos soros kommunikációs megoldások, mint például az UART (Universal Asynchronous Receiver/Transmitter) alapú rendszerek. Ezen megoldások igen gyorsan elterjedtek, egyszerűségük, és relatíve alacsony költségeik miatt. A járműipari hálózatok elterjedésével rengeteg – jellemzően egy gyártóhoz köthető – UART alapú kommunikációs megoldás és protokoll fejlődött ki egymással párhuzamosan, ám ezek – bár voltak köztük hasonlóságok -, csak speciális esetben, vagy egyáltalán nem voltak egymással kompatibilisek. A szabványosítás iránti igény ezen a terülten is hamar felmerült, miután a szabványos soros kommunikáció egyszerűbbé és olcsóbbá teszi a tervezést, illetve a gyártást, és átjárhatóságot biztosít. E célból jött létre a Local Interconnect Consortium az Audi, BMW, DaimlerChrysler, Volkswagen és Volvo, mint autóipari gyártók, illetve a Volcano Technologies és Motorola, mint autóipari elektronikai gyártók összefogásával. (5)

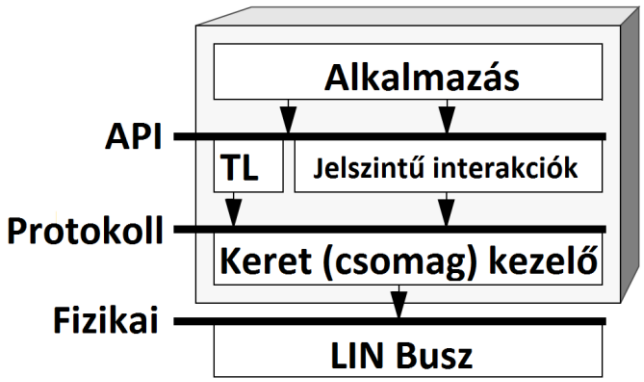
A LIN busz főbb tulajdonságai a következők (6):

- Egy Master (mester), több Slave (szolga) koncepció
- Alacsony költségű UART alapú integrált áramköri megoldás, egyszerű állapotgép alapú működés
- Kvarc, illetve rezonátor nélküli szinkronizációs lehetőség a szolgák esetében
- Determinisztikus jelátvitel előre számítható jelterjedési idővel
- Alacsony költségű egyvezetékes megoldás

- 20kbit/s sebesség
- Jel érzékelésen alapuló interakció
- Újrakonfigurálhatóság
- Szállítási réteg és diagnosztikai támogatás

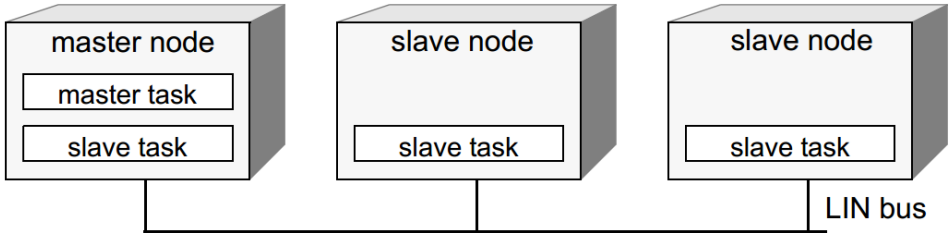
A LIN így egy olyan költséghatékony kommunikációs busz, amely megfelelően használható olyan területeken, ahol nem szükséges a CAN sávszélessége és sokoldalúsága.

A LIN szabvány a kommunikációt négy rétegre osztja, amelyet három interfész köt össze. A fizikai réteg és az alkalmazás között a keretkezelő és a jelfeldolgozó szolgáltat kapcsolatot. Sokszor a jelek és keretek kezelése egyben, szoftveresen megvalósítva alkot közös interfészt az alkalmazás felé. (1. ábra)



1. ábra: A LIN kommunikáció rétegei

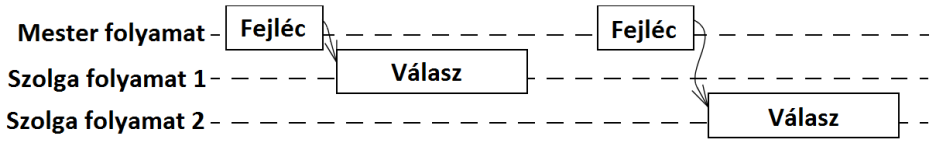
Egy LIN klaszter egy mester és több szolga folyamatból áll. A mester csomópont mind mester, mind szolga feladatokat is ellát, míg a többi csomópont csak szolga folyamatokat végez. A 2. ábra egy egy mesterből és két szolgából álló LIN klasztert ismertet. Azt, hogy a csomagok (keretek) mikor kerülnek továbbításra a buszon, a mester folyamat dönti el, a kerethez tartozó adatokat pedig a szolga folyamatok szolgáltatják. Szolga folyamat a LIN buszon nem kezdeményezhet kommunikációt.



2. ábra: LIN klaszter példa

1.2.1 Keretek

Egy adatkeret a mesterfolyamat által szolgáltatott fejlécből (header) és a szolgafolyamat által szolgáltatott válaszból (response) áll. A fejléc break szignálból, szinkronizációs mezőből, és keretazonosítóból áll. A keret azonosítója egyértelműen definiálja a keretet. Az a szolgafolyamat, amely a keretazonosító alapján a válasz szolgáltatására kötelezett, „kitölti" a keretet a válasszal, amely adatmezőből, és ellenőrzőösszegből (checksum) áll. Az a szolgafolyamat, akinek a keretazonosító alapján az adatok szólnak ellenőrzi a checksum-ot, és kiolvassa az adatokat. A folyamatot a 3. ábra ismerteti.

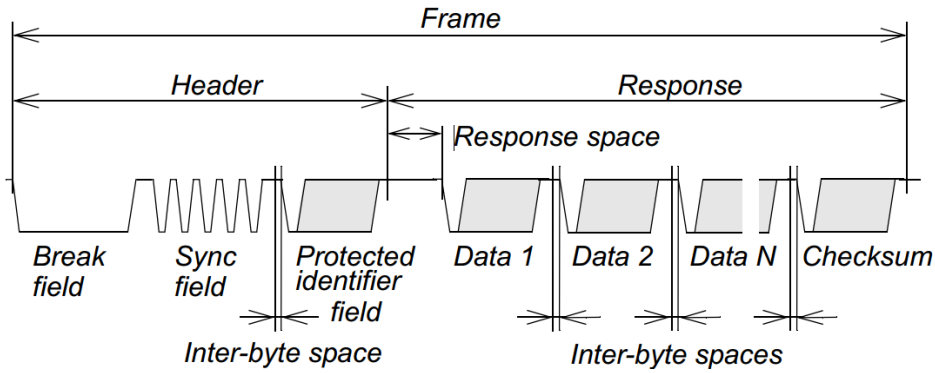


3. ábra: LIN Keretek

Ez a protokoll a következő előnyös tulajdonságokat eredményezi:

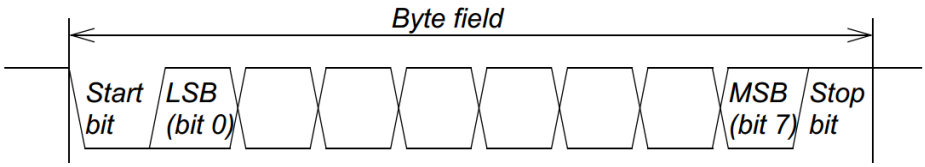
- Flexibilitás: A LIN klaszterhez tetszőlegesen lehet (szolga-) csomópontokat hozzáadni a többi szolgacsomópont szoftveres vagy hardveres átkonfigurálása nélkül.
- Üzenetcímzés: A CAN-hez hasonlóan az üzenet címzettjét a keret azonosítója definiálja.
- Multicast: Egyszerre több csomópont is felhasználhatja a keretek adatait.

A keret felépítését a 4. ábra ismerteti.



4. ábra: A LIN Keret felépítése

A kereten belül minden byte továbbítása – leszámítva a break jelet – a 5. ábra leírása alapján működik. A byte mező egy zéró (domináns) start bittel kezdődik, majd az adat LSB-vel kezdve, majd egy 1 értékű (recesszív) stop bittel záródik le.



5. ábra: A byte mező formátuma a LIN keretben

A keret a fejléccel, a *break jel* lefutó élével kezdődik. Ezt értelemszerűen mindig a mester folyamat generálja. A break jel legalább 13 bithossz hosszúságú. A break lezárása felfutó éllel történik, amely a magas jelet legalább névleges bitidő hosszúságú.

Ezután következik a *szinkronizációs mező*, amely egy 0x55 (bin 01010101) byte átvitelét jelenti. Ilyen break-sync formában egy szolga folyamat akkor is tudja érzékelni a keret kezdetét, ha történetesen adatmezőre számítana.

A *védett azonosító* (protected identifier) mező egy olyan byte mező, amely két almezőből áll: a keretazonosítóból és a hozzá tartozó paritásból. A keretazonosító hat bitből áll, így 0..63 értéket vehet fel, és háromféle csoportba sorolható:

- 0..59 (0x3B) alkalmazandó a jeltovábbító keretekhez;

- 60 (0x3C) mester igény (master request) és 61 (0x3D) szolgaválasz (slave response) a diagnosztikai keretek azonosítói; illetve
- 62 (0x3E) és 63 (0x3F) fenntartott azonosítók későbbi protokollbővítéshez.

A paritás a keretazonosító bitjeiből számítandó a következő módon:

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$$

$$P1 = \neg(ID1 \oplus ID3 \oplus ID4 \oplus ID5)$$

Egy keret egytől nyolc byte-ig terjedő adatot szállíthat, de a keret hossza előre definiált kell, hogy legyen. A LIN keret utolsó eleme az ellenőrzőösszeg, amely két alapvető formájú lehet:

- Klasszikus ellenőrzőösszeg, amely az adatbyte-ok átvitellel együtt vett összege invertálva; diagnosztikai keretek, illetve a LIN 1.x protokollt használó szolgák esetében alkalmazandó.
- Továbbfejlesztett ellenőrzőösszeg (Enhanced checksum), amely a védett azonosítót is beleveszi az összeg számolásába, amelyet a LIN 2.x protokollt alkalmazó eszközök használnak.

Az ellenőrzőösszegek használata értelemszerűen keretenként előre definiált kell, hogy legyen.

1.2.2 Időzítés, hozzáférés, kerettípusok

A LIN mester folyamata folyamatosan karbantartja az időzítési táblát (schedule table). Így minden keret egy meghatározott keret pozícióban keletkezik (frame slot). A keret pozíciók hossza megegyezik a leghosszabb keret elméleti hosszával, plusz egy keret-közi állandó idővel. Mivel a közeghozzáférést egyedül a mesterfolyamat vezérli, ezért konfliktus csak speciális esetben alakulhat ki. A LIN hálózatban különböző kerettípusok léteznek:

1.2.2.1 Feltétel nélküli keret (Unconditional frame)

A feltétel nélküli kertetek egyszerű jeleket szállítanak, keretazonosítójuk a fentebb említett 0 és 59 közötti értékeket vehetnek fel. Átvitelük a hozzájuk tartozó frame slot-ban történik, a keret fejlécét értelemszerűen a

mester írja, míg az adatbeírásra kötelezett szolga folyamat kitölti az adatmezőket, és a fogadásra kötelezett szolgák értelmezik azt.

1.2.2.2 Eseményvezérelt keret (Event triggered frame)

Az eseménykeretek célja a LIN klaszter sávszélességének optimalizálása, amikor több szolga figyelése történhet egyszerre ritkán fellépő események esetében. Az eseményvezérelt keretek adatmezőjét több szolga is kitöltheti. A mester a feltételes keret fejlécét beírja a hálózatra, amelynek adatmezőjét kitölti az a szolgafolyamat, ahol az azonosítónak megfelelő esemény keletkezik. Értelemszerűen ritkán előfordulhat, egyszerre két szolga is fel akarja használni a keretet, ekkor a mester folyamat érzékeli a konfliktust, és a két szolgának külön-külön küld egy feltétel nélküli keretet, amelybe már csak azok írják be a saját adataikat.

1.2.2.3 Szórványosan megjelenő keret (Sporadic frame)

A szórványosan megjelenő keretek olyan adatokat hordoznak, amelyek nem ütemes rendszerességgel kell, hogy megjelenjenek a hálózaton. Ennek megfelelően több szórványos keret foglalhatja el ugyanazt a frame pozíciót az időzítési táblában, és a mester folyamat dönti el, hogy mikor melyiket küldi ki a hálózatra.

1.2.2.4 Diagnosztikai keret

A diagnosztikai keretek a szállítási réteg információját továbbítják a hálózaton. Ezeknek a kereteknek a fentebb említett két azonosítója lehet: a 60, amelyet mester igény (master request) és a 61, amelyet szolga válasz (slave response) kereteknek nevezünk.

1.2.2.5 Fenntartott keretek (Reserved Frames)

A LIN2.x klaszterben a 62,63 azonosítójú keretek nem használhatóak, későbbi protokollbővítésre vannak fenntartva.

1.2.3 Jelek kezelése

Egy keretben két típusú adat kerülhet továbbításra:

- Jel (Signal), amely egy skalár, vagy byte tömb, amely a keret adatrészébe van csomagolva. a jelek pozíciója egy adott keretazonosítón belül állandó.
- Diagnosztikai üzenetek, amelyek két fentartott azonosítójú keretben közlekedhetnek.

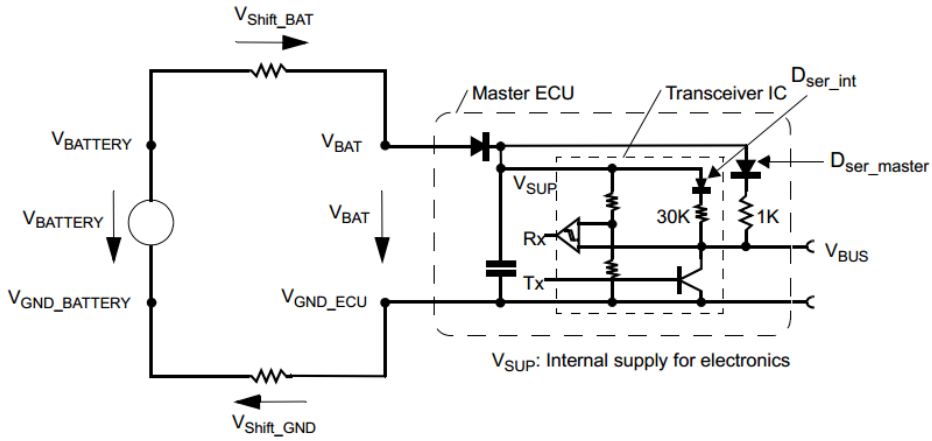
A skalár jel hossza 1-től 16 bitig terjedhet. Az egy bites skalár értéket logikai, „boolean” jelnek nevezzük. A 2 és 16 bit közötti skalár értékeket előjelnélküli egésznek (unsigned integer) tekintjük. A byte tömb 1 és 8 byte közötti tömböt határozhat meg.

Minden jelnek pontosan egy szolgáltatója van, tehát mindig ugyanaz a csomópont határozza meg az értékét. Értelemszerűen a jel fogadására tetszőleges számú csomópont iratkozhat fel. Minden jelhez tartozik egy kezdeti érték (initial value), amely a hálózat felépítésétől kezdve a jel értékének tekintendő, amíg abban változás nem lép fel. A jelek kezelése legkisebb egység a kommunikációban, azaz egy skalárt nem lehet csak részben módosítani, ami igaz a byte tömbökre is.

A jelek átvitele az (LSB first, MSB last) elv alapján történnek, azaz a legkisebb jelentőségű bit kerül beírásra először. A skalár jelek átnyúlhatnak a byte-ok határain, a kereten belüli elrendezésükre nincs megkötés. A byte tömböket azonban a keret byte-jaihoz igazítva, az első byte-al kezdve kell elhelyezni. Tetszőleges számú jel helyezhető el egy keretben, amíg azok nem fedik át egymást. Egy jel több keretben is szerepelhet, amíg ugyanaz a forrása.

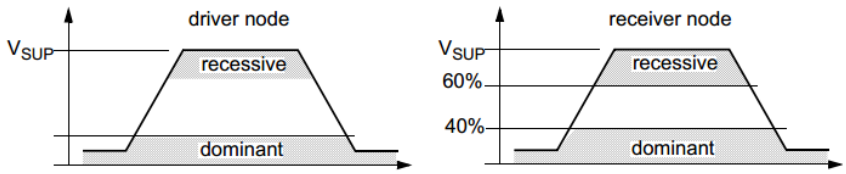
1.2.4 A LIN fizikai rétege

A busz meghajtó és receiver áramkör kialakítása az ISO 9141(7) szabványon alapul (6. ábra).



6. ábra: LIN busz fizikai réteg meghajtása

A bitek megfelelő továbbításához szükséges, hogy a jelek feszültségszintje a megfelelő legyen. Ezt ismerteti a 7. ábra. A buszon a domináns jel a 0, míg a recesszív jel a logikai 1.



7. ábra: A busz feszültségszintjei

1.3 CAN (8)(9)(10)

Ahogy az a fejezet bevezetőjében is ismertetésre került az autóiparban erős a szabványosítás iránti igény. Ennek az elvárásnak először a CAN szabvány felelt meg. Amennyiben egy adatkommunikációs rendszernek a szabványosítását tűzik ki célul, akkor úgy a felépítését, mint a működését részleteiben is precízen kell definiálni. A megválaszolandó lényegesebb kérdések általában a következők.

Hogyan kell a rendszer egyes elemeit a kommunikációs hálózathoz fizikailag (elektromos és logikai szempontból) csatlakoztatni, azaz milyen a rendszer hálózati topológiája?

Hogyan kell a továbbítandó adatokat az adatátvivő buszvonalra (villamos vagy fénykábel, rádiófrekvenciás összeköttetést biztosító szabad tér) csatolni és a megfelelő állomás(ok)hoz eljuttatni? Hogyan vannak a logikai szintek, a csatlakozó felületek, stb. kialakítva?

Milyen szabályok szerint történik az adatcsere a rendszer résztvevői között? Hogyan ismerhetők fel, kerülhetők el illetve korrigálhatók az átvitel alatt fellépő hibák? Milyen felépítésű az adatátviteli protokoll?

Hogyan történik a buszvonal használati jogának kiosztása a rendszer résztvevői között? Milyen elv szerint oldható fel a konfliktus, amennyiben több résztvevő egyszerre akar üzenetet küldeni ugyanazon a kábelen?

Az ilyen és hasonló kérdések megválaszolásával a CAN rendszert kialakító Bosch cég mérnökei sem maradtak adósak, és az általuk

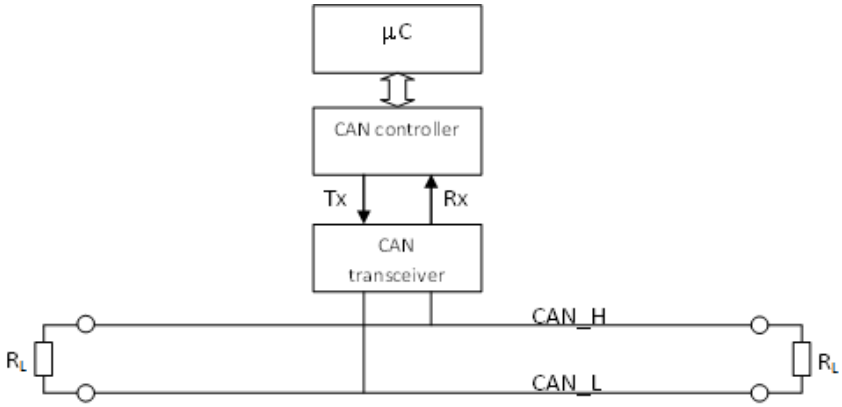
eredetileg autóipari felhasználásra kifejlesztett, többször módosított, majd szabadalmaztatott buszrendszer ma már egyéb ipari automatizálási területeken, így a gyógyászatban vagy akár a háztartási gépek elektronikájában is egyre gyakrabban tűnik fel, ezért feltétlenül indokolt az iránta megnyilvánuló fokozott érdeklődés. A CAN busz megbízhatóan működik elektromágneses zajjal terhelt környezetben, előnyösen használható valósidejű rendszerekben, egyszerű használat jellemzi, és ami az alkalmazások nagy részénél döntő: alacsony az állomásokra eső kommunikációs költség. Ezek az érvek azt támasztják alá, hogy a CAN jól használható ipari környezetben intelligens érzékelők és beavatkozók hálózatba kapcsolására is.

A CAN hálózatot leíró szabványok a fizikai és az adatkapcsolati réteget írják le, erős hangsúlyt fektetve ez utóbbira, melyet további alrétegekre bontanak:

- MAC (Media Access Control: Közeg hozzáférés vezérlés)
- LLC (Logical Link Control: Logikai kapcsolatvezérlés)

1.3.1 Felépítés

Fizikailag a CAN rendszer egy kétvezetékes aszimmetrikus buszt jelent, erre csatlakoznak az állomások soros vonalon (80. ábra). A vezeték sodrott érpár, mely lehet árnyékolt vagy árnyékolatlan is. A CAN kontrollert egy adóvevő (transceiver) áramkör illeszti a buszra, mely a kontroller TTL szintű jeleit alakítja át a busz differenciális jeleire. A transceiver másik fontos feladata, hogy a buszon lévő állomások között huzalozott ÉS kapcsolatot hozzon létre. Ennek az ütközések elkerülésében van döntő szerepe.



8. ábra: A CAN busz fizikai felépítése

A busz impedancia-illesztéséről lezárásokkal kell gondoskodni, ennek szabványos értéke R_L=120Ω. Ez a fizikai felépítés legfeljebb 1Mbit/s adatátviteli sebességet tesz lehetővé. Az adott rendszer sebességét természetesen korlátozza a busz hossza és a transceiverek késleltetése.

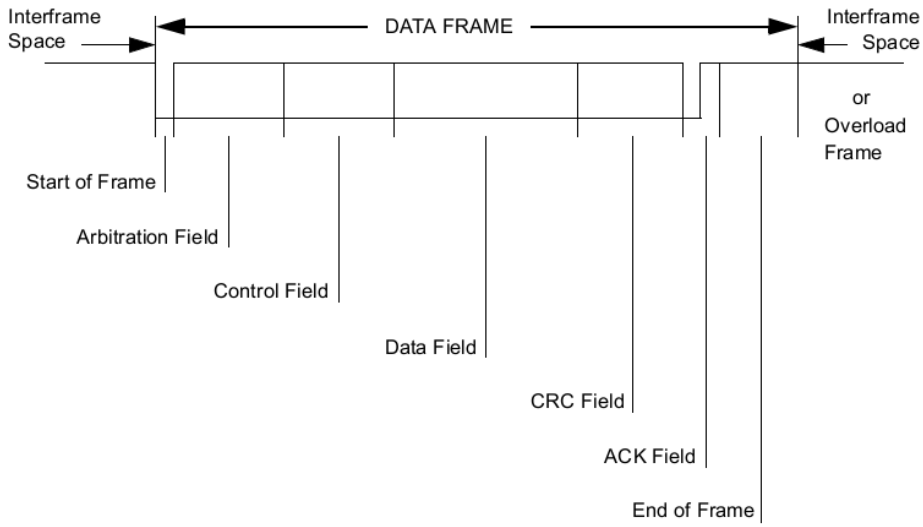
Az átvitel elektromos paramétereit az ISO11898 szabvány részletesen specifikálja. A maximális adatátviteli sebességek az alábbi táblázatban láthatók, melyből kiderül, hogy 500 m-en 125 kbit/s sebességre képes a CAN hálózat.

2. táblázat: Maximális adatátviteli sebességek a CAN hálózaton a kábelhossz függvényében

Kábelhossz (m)	Adatátvitelisebsség (kbit/s)
30	1000
100	500
250	250
500	125
1000	62,5

1.3.2 Üzenetformák és működés

Rátérhetünk a CAN rendszerben alkalmazott tényleges üzenetforma bemutatására és a kialakított protokollra. Az üzenetet, mint ahogy ez a soros adatátviteleknél egyébként is szokásos, keretformátumba foglalták. A teljes üzenetet alkotó bitsorozat, az adott kereten belül mezőkre osztott. Az egyes mezőkben elhelyezkedő bitcsoportok protokolláris, hibafelismerő, ill. adatátviteli feladatokat teljesítenek.



9. ábra: A CAN üzenet keret felépítése(11)

1.3.2.1 SOF (Start of Frame)

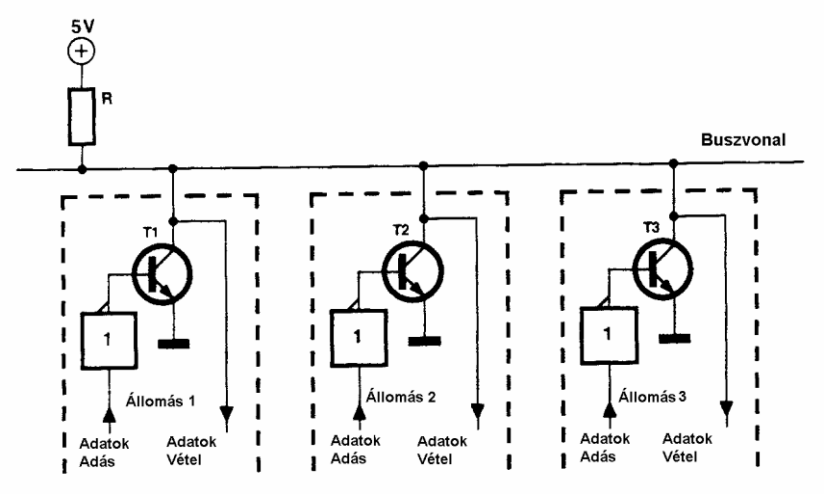
A buszvonal alaphelyzetben magas (1) szinten van és amennyiben valamelyik résztvevő használni kívánja, azzal kezdi az adást, hogy a vonalat alacsony, azaz (0) szintre húzza. Ez a feszültség szintváltozás azt eredményezi, hogy a többiek elkezdik figyelni a vonalat és egyúttal saját órajelük össze-szinkronozását is megkezdik az éppen adó egységgel. Igaz ugyan, hogy egy buszrendszeren belül csak egyféle bitátviteli sebesség használata szokásos, de az ehhez szükséges órajelet minden résztvevő saját maga állítja elő, nem járhatnak teljesen együtt. Soros, egyvonalas adatátviteli rendszerek esetében a magas szintről (1) alacsony (0) szintre lefutó, vagy 0-ról 1-re felfutó (feszültség) él (egyes esetekben mindkettő) használatos a többi résztvevő órajelének szinkronozásához. Ahány él van egy üzenetben, annyiszor történik meg a résztvevők órajelének szinkronozása. A mindenkori adó által a buszvonalra ültetett üzenet (bitfolyam) leolvasásához ugyanis megfelelő szinkronitás kell, vagyis alapkövetelmény, hogy a vevő saját órajeléből pontosan akkor képezze az olvasó impulzust, amikor az egységnyi bitérték a bemenetén megjelenik. Ellenkező esetben, vagyis ha nincs megfelelő gyakorisággal szinkronozás, előfordulhat, hogy a beérkező és az olvasó impulzusok időben elcsúsznak egymáshoz viszonyítva (fázishiba), ami hamis üzenetvételt eredményezhet.

A legnagyobb bitsebességet lehetővé tevő NRZ kód éppen emiatt hátrányos, hiszen egymásután több azonos szint (0-s vagy 1-es)

átvitelekor nincs élszerű feszültség változás, tehát szinkronozás sem történik. A szinkronizálást a CAN-IC BTL (Bus Timing Logic) időzítő logikája végzi, mégpedig úgy, hogy akár egyetlen bitidőn belül képes az olvasási idő beállítására. Az un. kemény szinkronozás a START bit lefutó élére indul, de üzenetolvasás közben is folyamatos utószinkronozás történik, figyelemmel a vevőket összekötő kábel jelterjedési idejére, valamint az esetleges fáziseltérésre.

Többek között a szinkronozás meghatározott időnkénti biztosítására vezették be a bitbeültetési szabályt (Bitstuffing).

A START bit fontossága a szinkronizálási folyamat megindításához már belátható, de érdemes azt a kiindulási alaptételt is kimondani, hogy ebben a rendszerben a domináns bitek a meghatározók. Ez azért lényeges, mert ha a buszvonal szintje 1-es (recesszív) akkor mindig átírható 0-ra (domináns), de fordítva nem. Ennek az elvnek felhasználásával valósíthatók meg különféle (pl. adásigény, átviteli hiba stb.) jelzések a rendszeren belül, hiszen elég a buszvonalat bármikor (egy vagy több ütemjelnyi időre) átírni 0-ra. Az átírást fizikailag a buszvonalat recesszív szinten tartó ellenállás osztóra kapcsolódó tranzisztorok realizálják. A megoldás elvét szabad kollektoros tranzisztorok által létrehozott ÉS (AND) kapcsolat realizálja.

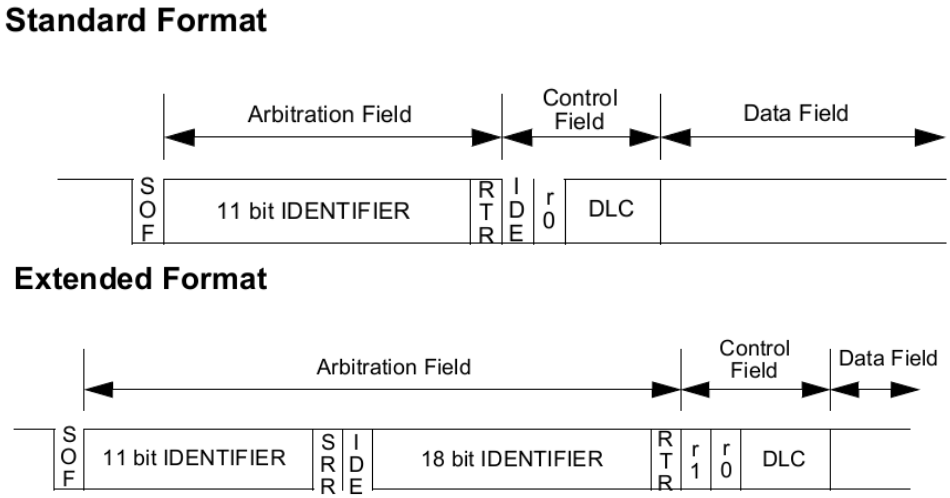


10. ábra:Szabad kollektoros tranzisztorok ÉS kapcsolata a buszvonalon

Mivel a CAN rendszerben nincsenek elsőbbséget (prioritást) élvező résztvevők, ezért nagyon könnyen fennáll a lehetősége annak, hogy egyszerre több résztvevő is szeretne a buszon üzenetet továbbítani. Megállapodás szerint prioritása nem az egyes résztvevőknek, hanem a legfontosabb leadandó üzenetnek van, tehát ez az alapelv dönti el minden adáskezdetnél a buszhasználati jogosultságot.

1.3.2.2 ARBITRATION (döntési) bitmező

A buszhasználat jogosultságának eldöntésére szolgál a startbitet követő ún. arbitrációs mező. Ugyanis az itt található 11+1 bit (CAN 2.0A), vagy 11+2+18+1 bit (CAN 2.0B) mindenkori értéke dönti el a jogosultságot, mégpedig úgy, hogy minél kisebb ez a számérték, annál nagyobb a buszhasználati prioritás.



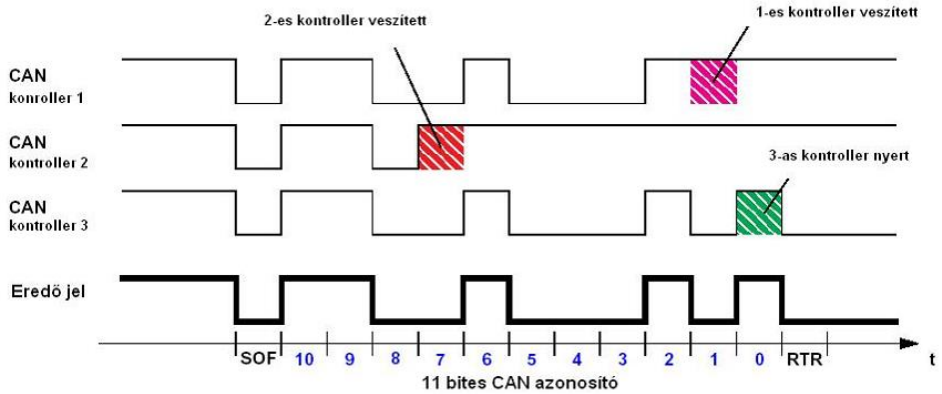
11. ábra: Arbitrációs mezők kialakítása (standard és extended ID esetén) (11)

Mint említettük, a rendszertervezők nem a résztvevőket (Multi Master elv), hanem az üzenetek tartalmát részesítik előnyben, így mindig a legmagasabb prioritású üzenet kerül a vonalra. Mivel a legmagasabb értékű üzenet arbitrációs bitsorozatának számszerű értéke a legkisebb az alkalmazott (kettes) számrendszerben, ezért az ilyen üzenetet adó résztvevő fogja legtöbbször alacsony szintre húzni a buszvonalat. Ehhez az üzenethez rendelt arbitrációs mező ugyanis - a többi jelentkezőjével összehasonlítva - a legtöbb 0-át (domináns bitet) tartalmazza a magasabb helyiértékeken.

Az üzenetek prioritási sorrendjét előre megállapítják és az adott rendszer-összeállítást ennek megfelelően programozzák. Belátható tehát, hogy egy újabb állomásnak a már meglévő rendszerbe illesztésekor, ha az csak vevőként üzemel, nem szükséges a szoftvert módosítani, hiszen nincs újabb üzenet, ami a korábbi prioritási sorrend megváltoztatását igényelné.

A buszhasználati jogosultság eldöntésének valós folyamatát (három adni kívánó állomás esetében) a következő ábra szemlélteti. Megfigyelhető, hogy a különböző arbitrációs kódok ellenére a buszvonalon csak egyetlen, az elsőbbséget (prioritást) élvező kódja jelenik meg, ami éppen a Multi-Master elv lényege!

A magyarázatot természetesen a korábban említett alapszabály adja, hiszen a busz recesszív bitje mindig átírható dominánsra, míg fordítva ez nem lehetséges.



12. ábra: A buszhasználati jogosultság eldöntésének folyamata(12)

Tisztázandó a következő probléma: hogyan dönti el a buszon lévő vevők sokasága, hogy melyiküknek szól az üzenet, hiszen nincs direkt, hanem csak az arbitrációs kódban foglalt, tartalommal történő címzés. A megoldás az, hogy minden résztvevő minden üzenetet egyidejűleg megkap, de csak a beérkezett üzenet nyugtázása után kezdi megvizsgálni, hogy ténylegesen "neki" szól-e a vett üzenet.

Ehhez olyan memóriát használnak, melybe gyártáskor beírják azon üzenetek vagy üzenetcsoportok arbitrációs kódját, melyet el kell fogadnia. Miután az összes résztvevő figyeli a vonalat (amelyik meg megszerezte, éppen üzenetet küld), az üzenet arbitrációs mezőjével a buszhasználati

jogosultság eldöntésén kívül a címzés is megoldható, még ha az ismertetett indirekt formában is.

Egyetlen esetben történhetne csak ütközés két, a buszt használni igyekvő résztvevő között, ha mindketten ugyanazt az arbitrációs bitsorozatot ültetik a vonalra. Ez a határeset azért fordulhat elő, mert a rendszerben nem csak adatokat lehet egy üzenetben továbbítani (Data Frame), hanem adatkérést (Remote Frame) is végre lehet hajtani. Az adatkérő üzenetkeretből csak az adatmező hiányzik.

Ezért, ha az egyik résztvevő éppen megkezdi azon adatok továbbítását, melyet a másik résztvevő egyidejűleg kérne, akkor az arbitrációs mezők azonossága miatt (tartalmi azonosság!), nem dönthető el, hogy ki jogosult a busz használatára.

A döntést a lezáró RTR bit (Remote Transmission Request, adatátviteli kérelem) hozza meg, hiszen ha adattovábbítás van, akkor ez 0, míg ha kérelem, akkor 1 értékű. Márpedig, ahogy korábban említettük az 1-es értékű (recesszív) bit mindig átírható 0-ás (domináns) bit-re, vagyis az adatküldőnek lesz igaza, tehát övé a busz. Belátható, hogy a standard CAN 2.0A rendszer 11 arbitrációs bitje összesen 211 = 2048 féle azonosító kódot tenne lehetővé, de ebből „csupán” 2032 realizálható, tehát ennyi résztvevő, vagy pontosabban ennyiféle üzenet kapcsolódhatna elméletileg a buszvonalra. A valóságban a tényleges szám jóval kisebb, de a jövő kihívásait figyelembe véve, a kiterjesztett CAN 2.0B rendszerben az arbitrációs mező bitszámát 29 bitre növelték, ami 536.870.912 re emelhetné a buszon küldhető különféle üzenetek számát. A felhasznált érték itt is sokkal kevesebb az elviekben lehetségesnél, hiszen a túl sok résztvevő, vagy üzenetfajta azt eredményezné, hogy az alacsony prioritásúak szinte sosem kapnának buszhasználati jogosultságot.

1.3.2.3 CONTROL (ellenőrző) bitmező

A következő Control bitmező egy ellenőrző mező. Az ebben elhelyezkedő 6 bit tartalmazza a rendszerkódot, valamint azt az összegkódot, melyet az üzenetben soron következő adatmezőben található bájtok képeznek. Mivel a CAN rendszerben 0-8 byte között változhat az adatmező kitöltöttsége, ezért az adathossz előzetes megadására 4 bit elegendő. A maradék 2 bit közül az első (IDE) domináns (0) értéke azt tudatja a résztvevőkkel, hogy az alap CAN 2.0A rendszerben használt 11 bites arbitrációs formáról van szó, míg ennek recesszív (1) értéke a kibővített 29 bites CAN 2.0B rendszer üzenetére utal. A második bit (r0) egy fenntartott (reserved) bit,

melynek a későbbi továbbfejlesztéseknél lesz jelentősége, értéke jelenleg domináns, azaz (0).

A kontroll mező a hibafelismerésben is fontos szerepet játszik, hiszen közli a vevővel, hogy hány adat bájt tartalmaz az üzenet. Ha a vett üzenet adat byte száma eltér a kontroll mezőben megadott értéktől, akkor az átvitel közben hiba történt.

1.3.2.4 DATA (adat) bitmező

A következő mező az ún. adatmező. Ebben bájtokban összefogott adatbitek találhatók, mégpedig úgy, hogy számuk rugalmasan változhat, azaz 0-8 bájt (0-64 bitet) tartalmazó adatmezők fordulhatnak elő. A változó hosszúságú adatmező, az egész üzenet hosszát is megváltoztatja, hiszen egy adatkérés nyilván 0 adatbájtot tartalmaz, így az ilyen üzenet lényegesen (éppen 64 bitidővel) rövidebb, mint a teljes, 8 adatbájtot kihasználó adattovábbítás, ezért az üzenetek átviteléhez szükséges idő is változik az adatmező hosszának függvényében. A gyakorlatban az adatmező legtöbbször 2-4 bájtból áll, ezért az üzenetek többnyire rövidek.

1.3.2.5 CRC (Cyclic Redundancy Check) bitmező

A következő, 15+1 bitet tartalmazó mező a ciklikus redundancia vizsgálathoz szükséges kódot (CRC) tartalmazza, amit mindig az adó képez, az üzenet előre rögzített mezőinek bitképe alapján. A hibajelzésre kiválóan, de hibajavításra már kevésbé használható CRC.

A CAN üzenetek CRC-15-ös (Cyclic Redundancy Check) védelemmel vannak ellátva. Ez a védelem garantálja a 6-os Hamming távolságot, azaz 5 hibás bit esetéig bármely hálózati eszköz felfedezi a hibát egy 113 információs bitet tartalmazó üzenetben. Mivel a jelenlegi üzenetek átlagosan csupán 83 információs bitet tartalmaznak a hiba felismerésének valószínűség még nagyobb. A számítások azt mutatják, hogy a maradék hiba valószínűsége $\lambda_v \approx 3x10^{-5}$. Gyakorlatilag ez azt jelenti, hogy átlagos adatátviteli sebesség és üzenethossz esetén egy gépjármű teljes üzemi élettartama alatt (kb. 3000 óra) az adott hibavalószínűség mellett kb. 300 db a fel nem ismert hibás üzenetek száma, holott ennyi idő alatt megközelítőleg 10.000.000 üzenet átvitelére kerül sor az adatbuszon. A hibafelismerést természetesen még további megoldások segítik, melyek együttes alkalmazásával rendkívül kedvező értékek érhetők el.

A folyamat lényegét összefoglalva: az adó a buszra feltett üzenetből, míg a vevők a kapott üzenetből képzenek, szigorúan azonos szabályok alapján, egy a 15 bitből álló vizsgáló (CRC) összeget. Mivel az adó ezt a CRC ellenőrző összeget üzenetében, a most tárgyalt mezőben elküldi, a vevők pedig maguk képzik a beérkezett üzenetből, csupán a kettő összehasonlítása marad hátra. Amennyiben egyezés van, a vétel hibátlan volt, ha nincs, hibajelzést kell generálnia. A +1 bit mindig recesszív (1) és ez jelzi a CRC mező végét. Feltétlenül érdemes megjegyezni, hogy a leírt CRC képzési procedúra viszonylag egyszerűen megvalósítható, nem kell tehát komplikált elektronikai megoldásokra gondolni. Az adatátviteleknél talán éppen ezért terjedt el a CRC használata, mert az elérhető eredmény kitűnő, ugyanakkor a realizálás egyszerű.

1.3.2.6 ACKNOWLEDGEMENT (nyugtázó) bitmező

A következő ún. nyugtázó mező 1+1 bitből áll és arra szolgál, hogy a vett üzenet hibás vagy hibátlan voltát visszajelezze az üzenetet küldőnek. A két eredetileg recesszív bit közül jelzőfunkciója csak az elsőnek van. Mivel az adó által küldött üzenetet minden vevő figyeli, és ha bármelyik hibátlannak tartja, akkor ezt az első (ACK-Slot) bitet 1-es értékről átírja 0-ra, azaz dominánsra. Mint tudjuk, a buszon a recesszív bit mindig átírható dominánsra, így az adó, figyelve saját adását azonnal észleli, hogy legalább egy vevő hibátlannak ítélte a leadott üzenetet. Hibás üzenet esetén a vevő recesszív állapotban hagyja az ACK-Slot bitet, amiről az adó felismeri, hogy egyetlen vevőhöz sem jutott el kifogástalan üzenet, így azt meg kell ismételni. A második bit mindig recesszív és csupán a nyugtázó mező végét jelzi.

1.3.2.7 EOF (End of Frame, keret vége) bitmező

Az üzenet keretformátuma egy üzenet vége mezővel záródik, ami 7 recesszív bitet tartalmaz, erről ismeri fel a mindenkori vevő, hogy az üzenet véget ért. Ebben a mezőben a bitbeültetési szabályt nem érvényesítik.

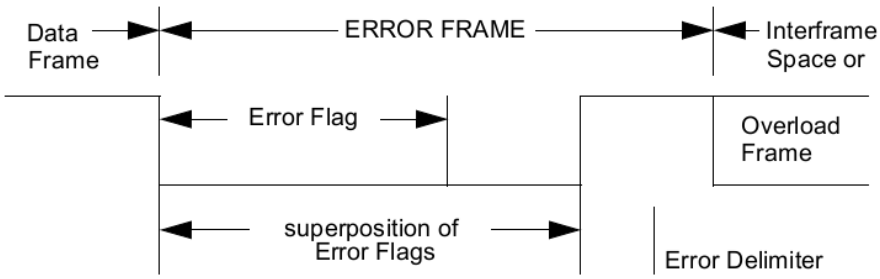
A következő üzenet megkezdéséig az előző üzenetet vevő egységeknek egy megadott idő szükséges a feldolgozáshoz, de legalábbis az adatok eltároláshoz. A szükséges minimális idő legalább három bitidő, amit egy közbenső, (Interframe Space) mező biztosít. A következő üzenet kezdetéig természetesen lényegesen több idő is eltelhet, hiszen ez a mindenkori résztvevők üzenetküldési aktivitásától függ.

Az üzenetek közül a két alapvető fontosságú, adatküldő illetve adatkérő üzenet felépítésével már megismerkedtünk.

1.3.2.8 ERROR (hiba) üzenet

A hibaüzenet az egész rendszer működésére kihatással van. Amennyiben legalább egy vevő "menetközben" felismeri a továbbított üzenet hibás voltát, akkor leadja a hibaüzenetet. Az üzenet egy hibajelzéssel kezdődik (Error Flag), ami egymásután 6 domináns bitnek a buszra jutását jelenti. Minden vevő azonnal felismeri, hogy az un. "bitbeültetési" szabály sérült, hiszen legfeljebb 5 azonos bit követheti egymást a szigorú előírások szerint. Amennyiben más vevők is "egyetértenek" a hibajelzéssel, akkor beültetnek még egy hibajelzést az üzenetbe, ezért a hibaüzenet minimum 6 és maximum 12 domináns bitet tartalmazhat, majd 8 recesszív bittel zárul. Ezzel jelzett a hibaüzenet vége, ami, mint a korábban megismert üzeneteknél láttuk, szintén kivétel a bitbeültetési szabály alól.

A hibajelzés (6-12 domináns bit) megjelenésekor az üzenetet minden vevő automatikusan elveti, azaz érvénytelen üzenetként kezeli. A hibaüzenet keret kialakítása a következő ábrán látható.



13. ábra: A CAN hibaüzenet keret felépítése(11)

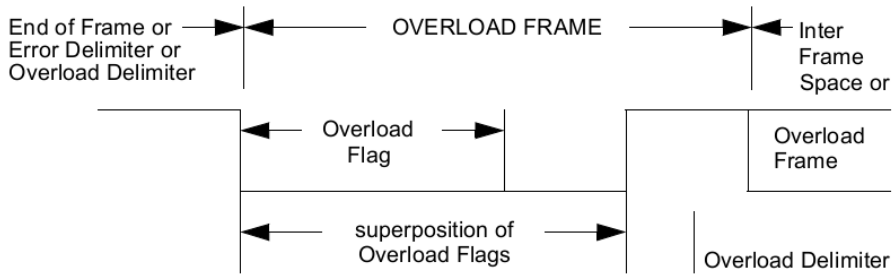
Maga az üzenetet adó is küldhet hibaüzenetet, hiszen az adás alatt saját kiadott bitsorozatát is folyamatosan hasonlítja össze a buszon lévővel (monitoring üzem), és eltérés esetén azonnal hibaüzenetet küld.

1.3.2.9 OVERLOAD (túlterheltség) üzenet.

Ez az üzenet összesen két bitmezőből áll, nevezetesen a túlterheltség jelző és az üzenet vége bitsorozatból. Ilyen üzenet két esetben jelenhet meg a buszon, egyrészt akkor, ha valamelyik vevő feldolgozó processzora még egy kis időt kér a korábban kapott adatok kezeléséhez, vagyis nincs még készen új üzenet vételére, másrészt akkor, ha az üzenetkeretek közötti

szünetidőt (Interframe Space) biztosító 3 recesszív bit valamelyike helyett domináns bitet észlel. Bármelyik esetről is van szó, a túlterheltségi üzenet a következő üzenet buszra helyezését gátolja. A túlterheltséget jelző üzenet a hibajelzéshez hasonlóan 6 domináns bitet juttat a buszra, majd ezt az üzenetet is 8 recesszív bit zárja. A túlterheltség üzenetet az Interframe Space első recesszív bitjének átírásával kezdi az egyik résztvevő, mire a többi, érzékelve a szünetidőben fellépő domináns bitet, leadja saját túlterheltség üzenetét is, ami így összesen 7 domináns bitre növekszik, majd az üzenet végét jelző 8 recesszív bit jelenik meg, melyekre természetesen nem érvényes a bitbeültetési szabály.

Belátható, hogy a túlterheltség üzenet domináns bitjei nem írják át az üzenet maradék részét, így nem is teszik azt használhatatlanná, hiszen a túlterheltség jelzés csak az üzenetek között fenntartott „pihenési" időben indítható. Ezzel elérhető, hogy az üzenetet feldolgozó egy kis „szusszanáshoz" jusson. A túlterheltséget jelző üzenetkeret kialakítását az alábbi ábra mutatja.



14. ábra: A CAN túlterheltségi üzenet keret felépítése(11)

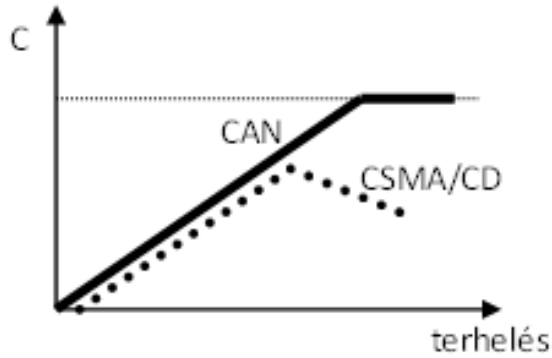
Néhány mondat erejéig érdemes ismertetni a jóval kevésbé frekventált egységek információ cseréjéhez alkalmazott, ún. multiplex üzemmódról. Ezzel a lassú üzemmóddal olyan egységek közötti információcserét valósítanak meg, mint a fedélzeti műszerek, a világítás, a komfortelektronika vagy a kommunikációs berendezések alapműködtetése. Mint ismeretes, a felsorolt egységek egyike sem igényel ún. valós idejű beavatkozást.

Ilyenkor egy külön buszvonalat építenek ki a korábban említett egységek között, alacsonyabb adatátviteli sebességgel (szokásos érték: 30-50 Kbit/sec), melyre a résztvevők a kissebességű CAN rendszer protokollja szerint adják fel az üzeneteiket.

A CAN rendszer tervezésekor ugyanis, kis és nagysebességű adatátviteli változatokat alakítottak ki. A szabványosított, kifejezetten alacsonysebességű CAN rendszernél (ISO11519-2) az adatátviteli sebesség 5–125 Kbit/s között választható, míg a nagysebességű CAN rendszernél (ISO11898), úgy az alacsony, mint a nagysebességű átvitel megvalósítható (5 Kbit/s - 1 Mbit/s) ezért kedveltebb és gyakrabban alkalmazott forma.

1.3.3 Összefoglalás

A CAN busz igény szerinti allokáció osztályába tartozik, hozzáférése nem destruktív. Ez azt jelenti, hogy a buszon mindig csak hasznos információ közlekedik, nincsenek törött keretek, mint a CSMA/CD (Carrier Sense Multiple Access with Collision Detection: Vivőérzékeléses többszörös hozzáférés ütközésérzékeléssel) esetében. Másrészt senki sem tartja fel feleslegesen az adatforgalmat, ha nem kíván adni, mint ahogy a token ringnél, vagy a fix időosztásos rendszereknél történik.



15. ábra: A CSMA/CD és a CAN busz csatornakapacitása a terhelés függvényében

A CAN busznál ez nem fordulhat elő, elégtelen átviteli kapacitás esetén is a maximális csatornakapacitásnak megfelelő mennyiségű kérést feldolgozza a rendszer a prioritásának megfelelő sorrendben. A legmagasabb prioritású keretek átvitele tehát még ekkor is biztosítható. A CAN a rendelkezésre álló kapacitást hatékonyan használja ki, mivel kicsik a buszallokációs rések.

A CAN buszos rendszer decentralizált, vagyis az arbitrációt nem egy központi (master) egység végzi, melynek a meghibásodása megbéníthatná a kommunikációt. A CAN busz mindaddig működőképes, amíg az

adatátviteli csatorna fizikailag ép, illetve valamely állomás meghibásodása révén nem kerül végleg domináns állapotba.

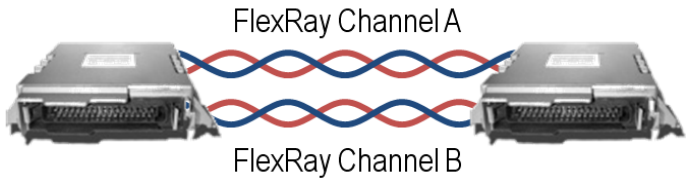
A fentiek alapján elmondható, hogy a CAN elméletileg minden tekintetben kielégíti az elvárásokat, továbbá sok pozitív tapasztalatról tudni mind a járműves, mind pedig az ipari felhasználások területén. Továbbá nagyon sok gyártó termékei közül lehet választani, ennek megfelelően az eszközök árai is alacsonyak.

1.4 A FlexRay technológia

1.4.1 A FlexRay Fizikai rétege

1.4.1.1 Hálózat kialakítása

A FlexRay tervezésénél is alapvető szempont volt, hogy alacsony költséggel nyújtson nagy teljesítményű és megbízható adatátvitelt járműipari környezetben. A FlexRay hálózatban árnyékolatlan csavart érpárok kötik össze az egyes csomópontokat. A hálózat lehet egy- vagy kétcsatornás, ami egy vagy két független érpárt jelent. A differenciális adatátvitel biztosítja, hogy az elektromos zaj árnyékolás nélkül se okozzon zavarokat az átvitelben.

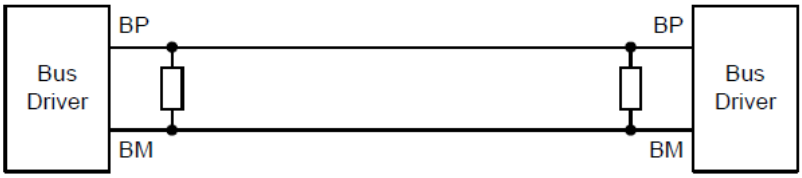


16. ábra: Kétcsatornás FlexRay hálózat (forrás: http://ni.com)

A kétcsatornás kivitel használható hibatűrő (fault-tolerant) rendszerek kialakítására, vagy növelhető a sávszélesség a segítségével. A legtöbb elsőgenerációs FlexRay Hálózat csak egy csatornát használ, azonban a jövőben várható a kétcsatornás rendszerek fokozottabb elterjedése is.

A FlexRay busz végeit ellenállásokkal le kell zárni (17. ábra), és ezt elég csak a busz végein lévő eszközöknél megtenni. A túl sok lezárás (hasonlóan a lezárás hiányához) hibát okozhat a hálózat működésében. Míg a hálózatok kialakítása változhat, addig a vezeték impedanciája tipikusan 80 és 110 ohm között lehet., és a lezárás is megegyezik ezekkel

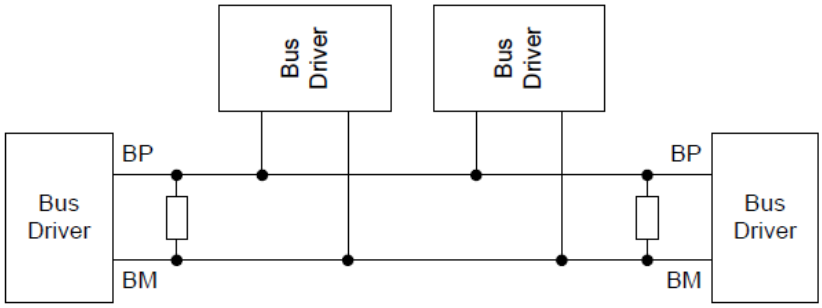
az impedancia értékekkel. A PC-kben használatos FlexRay eszközök általában tartalmaznak lezáró ellenállást.



17. ábra: A lezáró ellenállások elhelyezés egy pont-pont hálózaton

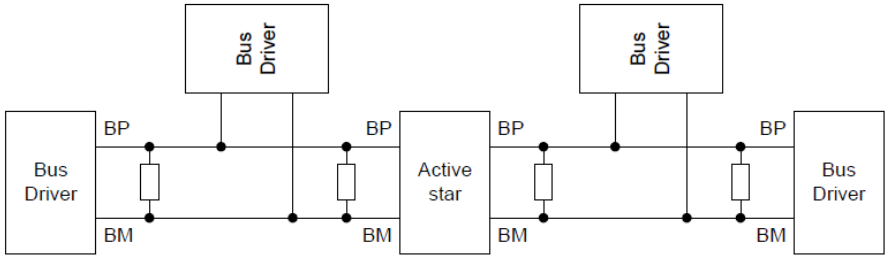
A differenciális adatátvitelnek megfelelően a BP-vel (busz plusz) jelöljük a pozitív busz feszültséget, és BM-mel (busz mínusz) a negatívat. A „Bus Driver" (BD) elnevezésű egységek az ECU-k FlexRay meghajtó részeit reprezentálják.

A járműipari kommunikációs technológiák esetén a legjellemzőbb a buszrendszerű kialakítás. Mind a Flexray, mind pedig a CAN és a LIN alapvetően a párhuzamos huzalozású, sokcsatlakozós (multi-drop) rendszerű buszt támogatja.



18. ábra: A lezáró ellenállások elhelyezés egy sokcsatlakozós busz hálózaton

Ebben az esetben a fenti pont-pont hálózatot bővítjük további eszközökkel (18. ábra), ahol további lezáró ellenállások használata már nem szükséges.



19. ábra: A lezáró ellenállások elhelyezés aktív csillag hálózaton

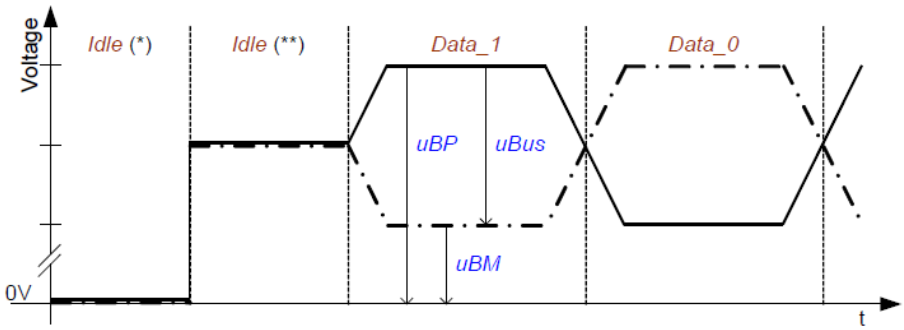
A FlexRay technológia lehetővé teszi a kialakítható aktív csillag hálózat is, ebben az esetben a csillagpont elveiben egy kétirányú hálózati jelismétlőként (repeater) funkcionál. Ennél az elrendezésnél a csillagpont mindkét oldalát a busz hálózatnak megfelelően kell lezárni. A Flexray hálózat lehetséges topológiáit a 1.4.3 fejezetben tárgyaljuk.

1.4.2 Elektromos jelátvitel

A fizikai jelátvitel szempontjából a busznak három különböző állapota lehet, melyek szabványos elnevezései a következők:

- Data_0,
- Data_1,
- Idle.

A 20. ábra mutatja az elméleti jelszinteket a buszon. Az előző fejezetben bevezetett jelölésekhez alkalmazkodva uBP és uBM reprezentálja a vezetékek feszültségét a földpotenciálhoz képest. A busz feszültsége ennek megfelelően: uBus = uBP- uBM.



*: az összes eszköz (és aktív csillagpont is) alacsony fogyasztású üzemmódban van

**: egy eszköz sincs (és aktív csillagpont sem) alacsony fogyasztású üzemmódban

20. ábra: Elméleti elektromos jelátvitel

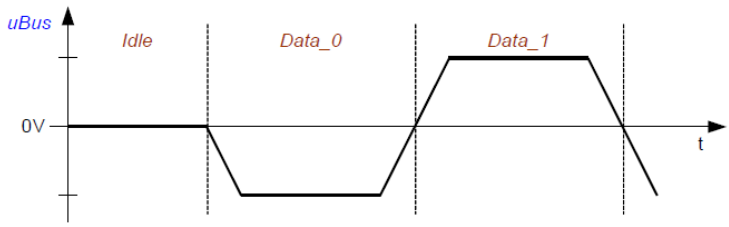
„Idle” módban sem a BP sem a BM vezeték nincs aktívan meghajtva. A busz meghajtók (BD) mindkét vezeték feszültségét ugyanarra az adott feszültségszintre állítja a működési módtól függően.

3. táblázat: Feszültségszintek „Idle” módban

Leírás	**Min**	**Max**	**Jellemző**
Feszültségszint a vezetékeken normál módban	1,8 V	3,2 V	2,5 V
Feszültségszint a vezetékeken alacsony fogyasztású módban	-0,2 V	0,2 V	0 V

Fontos megjegyezni, hogy a feszültség források belső ellenállása szignifikánsan nagyobb az alacsony fogyasztású módban. Továbbá abban az esetben, ha egyes BD-ek alacsony fogyasztású módban vannak, míg mások nem, akkor a buszon mért feszültségszint 2,5 V alatt lesz.

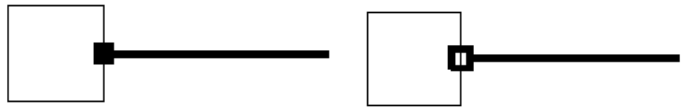
A Data_1 állapothoz pozitív feszültségkülönbség, míg a Data_0-hoz negatív tartozik. Az állapotok összefoglalását (a buszfeszültséget az idő függvényében ábrázolva) a 21. ábra mutatja.



21. ábra: Busz állapotok összefoglalása

1.4.3 Hálózati topológiák

A FlexRay esetén a jármű kialakításától és a FlexRay felhasználásától függően sokféle topológia kialakítható, melyeket a következő alfejezetekben mutatunk be.

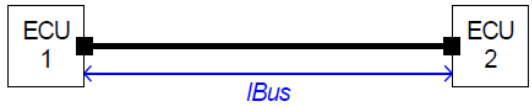


22. ábra: A lezárások jelölése (bal: lezárt kábelvég, jobb: lezáratlan kábelvég)

A továbbiakban a szükséges lezárásokat a 22. ábra szerint jelöljük.

1.4.3.1 Pont-pont

A legegyszerűbb kialakítású hálózat. A vezeték maximális hossza (lBus) függ az EMC zavarástól és a vezeték típusától.

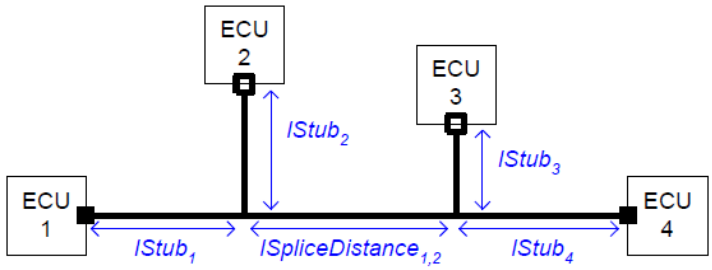


23. ábra: Pont-pont kialakítású FlexRay hálózat

Általános szabályként a szabvány kimondja, hogy mindig a lehető legrövidebb hosszra kell törekedni, és a teljes, kumulált vezetékhossz nem lehet több mint 24 m. Ezek a szabályok a többi topológiára is érvényesek.

1.4.3.2 Sokcsatlakozós (multi-drop) busz

Klasszikus buszrendszerű hálózati topológia, a két végén elhelyezett lezáró ellenállásokkal.

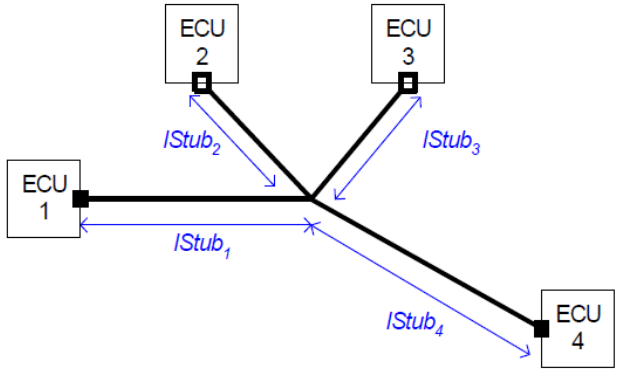


24. ábra: Busz kialakítású FlexRay hálózat

A fenti ábrán látható változat a minimális kiépítést mutatja. A csomópontok száma nincs maximálva, azonban a vezeték típusát és a maximális összesített vezetékhosszt itt is figyelembe kell venni.

1.4.3.3 Passzív csillag

A passzív csillag kialakítású FlexRay hálózat tulajdonképpen a buszrendszerű hálózat egy speciális esete. Annyi megszorítást tartalmaz, hogy egy csomópontot tartalmaz, és minden eszköz ahhoz csatlakozik.



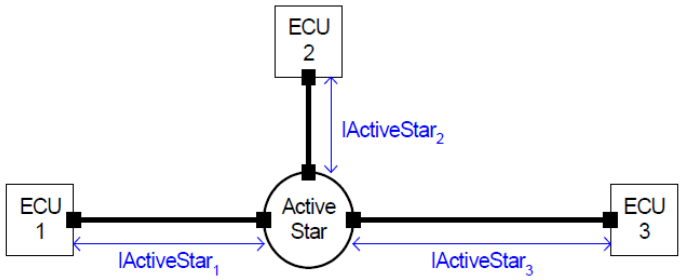
25. ábra: Busz kialakítású FlexRay hálózat

A lezárás itt is két csomópontban történik, a példában az ECU1 és ECU4 egységekben.

1.4.3.4 Aktív csillag

Az aktív csillag kialakítású hálózat tulajdonképpen n számú pont-pont hálózatot jelent az ECU-k és a csillagpont (Active Star) között. A csillagponti eszköz feladata az adatfolyamok továbbítása egyik ágtól az

összes többi felé. Az eszköz mindenághoz tartozóan különálló adatküldő és fogadó egységet tartalmaz, ebből következően az ágak elektromosan le vannak választva egymástól.

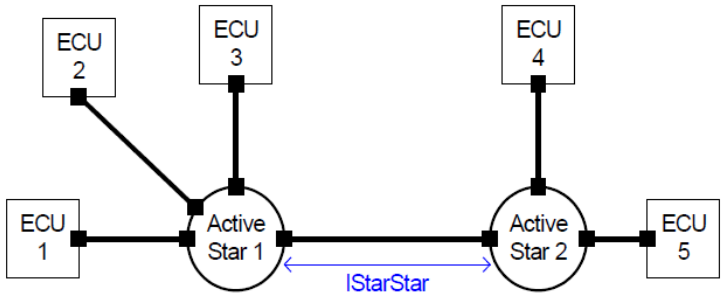


26. ábra: Aktív csillag kialakítású FlexRay hálózat

Abban az esetben, ha csak két ágat tartalmaz az aktív csillag hálózat, akkor tulajdonképpen egy jelismétlő (repeater vagy hub) funkciót lát el, hogy a pont-pont hálózat maximális vezetékhossza növelhető legyen.

1.4.3.5 Kaszkád aktív csillag

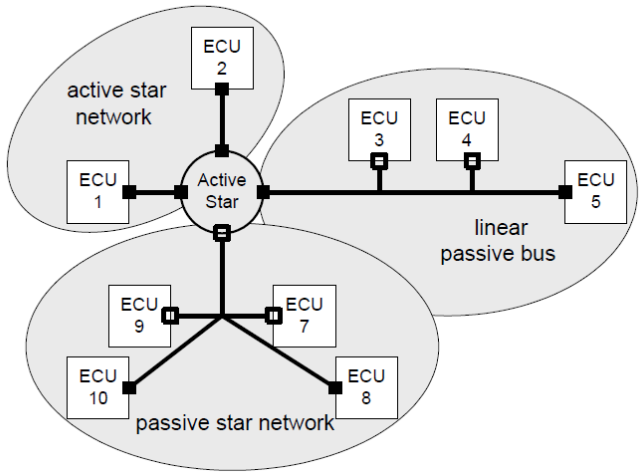
A kaszkád akítv csillag hálózati topológia két aktív csillagpontot tartalmaz. Ezt a hálózati kialakítást a 2,5 és az 5 Mbit/s-os sebességet alkalmazó hálózatokban lehet használni. 10 Mbit/s esetén nem szabad két aktív csillagpontot a hálózatba illeszteni, mivel az aszimmetrikus késleltetés túlzott mértékben megnövekszik. A két csillagponti eszköz pont-pont kapcsolatban van egymással.



27. ábra: Kaszkád aktív csillag kialakítású FlexRay hálózat

1.4.3.6 Hibrid topológiák

Az aktív csillag hálózatok használata esetén, az egyes ágak kialakíthatók busz, vagy passzív csillag topológiában is.



28. ábra: Példa a hibrid topológiájú FlexRay hálózatra

Az ilyen hálózati kialakításokat a FlexRay szabvány hibrid topológiáknak nevezi.

1.4.4 A FlexRay protokoll

A FlexRay protokoll egyedülálló módon egy „idő-triggerelt” protokoll, amely lehetőséget biztosít determinisztikus adatok kiszámítható időn belüli (mikroszekundum pontossággal) fogadására, de emellett a CAN-hez hasonlóan dinamikus, eseményvezérelt adatok kezelésére is képes. FlexRay ezt a hibrid működést, azaz a statikus és dinamikus keretek továbbítását egy előre beállított kommunikációs ciklussal valósítja meg, amely egy előre meghatározott helyet biztosít a statikus és dinamikus adatoknak. Ezt a helyet a hálózattal együtt kell konfigurálnia a hálózat tervezőjének. Míg a CAN csomópontoknak csak arra van szükségük a kommunikációhoz, hogy ismerjék a megfelelő átviteli sebességet, addig a FlexRay csomópontoknak ismerniük kell az összes hálózati elem konfigurációját ahhoz, kommunikálni tudjanak.

Mint minden „multi-drop” busz, egyszerre csak egy eszköz küldhet adatot a buszra egyidőben, ellenkező esetben az adatok hibásak lesznek. Többféle módszer is létezik ennek kiküszöbölésére. A CAN hálózat

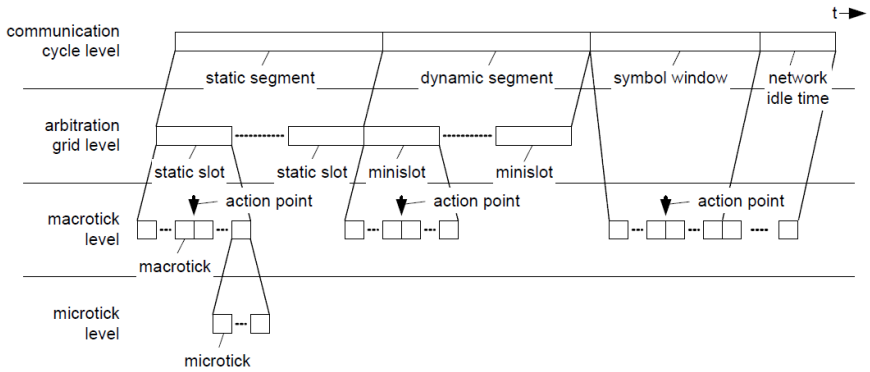
például arbitrációs módszert használ, amely prioritás alapján dönti el, hogy mely eszköz küldhet az adott pillanatban. Bár a módszer rugalmas és könnyű bővíteni, ez a technika nem teszi lehetővé a magas adatátviteli sebességet, és nem garantálja az időben történő megérkezését az adatoknak. FlexRay a több csomópontot az időosztásos többszörös hozzáférés (Time Division Multiple Access: TDMA) módszerével kezeli. Minden FlexRay hálózati eszköz órája szinkronban van egymással, és egy adott időrésben írhatnak a buszra. Mivel ebben a rendszerben az időzítés konzisztens, így a FlexRay garantálni tudja az adatok determinizmusát a hálózaton. Ez számos előnnyel jár azon rendszerek számára, amelyek működéséhez elengedhetetlen az adok, pontos, valós-idejű időzítése, mint például az összetettebb szabályozásokat megvalósító ECU-k.
Beágyazott rendszerekben kialakított hálózatoknál nem elvárás, hogy az egyes eszközök automatikusan felderítsék a hálózatot és megfelelően átkonfigurálják magukat. A jellemző eljárás az, hogy a gyártás során beírásra kerül egy zárt konfiguráció az eszközökbe, amely a termék életciklusa során már nem változik. Így a konfigurációt a tervezés során ki lehet alakítani, így csökkentve a további fejlesztési költségeket, és növelve a megbízhatóságot.

Egy FlexRay hálózat helyes működéséhez minden csomópont megfelelően be kell konfigurálni. A FlexRay szabvány lehetővé teszi különböző típusú hálózatok használatát, és megengedi a tervezőknek, hogy a bizonyos feltételek mellett, beállítsák a szükséges hálózati sebességet, a determinisztikus és a dinamikus adatmennyiséget, valamint egyéb paramétereket. Minden FlexRay hálózat eltérő lehet, ezért minden egyes eszközt fel kell programozni a megfelelő hálózati paraméterekkel, mielőtt részt vesznek a kommunikációban

A hálózati konfiguráció megkönnyítése érdekében, kialakításra került egy szabványosított formátum a szükséges paraméterek tárolására és továbbítására Field Bus Exchange Format (FIBEX) néven.

1.4.4.1 Közeghozzáférés vezérlés (Media Access Control)

A FlexRay protokoll közeghozzáférés vezérlése (MAC) ismétlődő kommunikációs ciklusokon (communication cycle) alapul. Egy kommunikációs cikluson belül kétféle módon érhető el a közeg. Az egyik a statikus időosztásos többszörös hozzáférés, míg a másik az ún. „minislot” alapú megoldás.



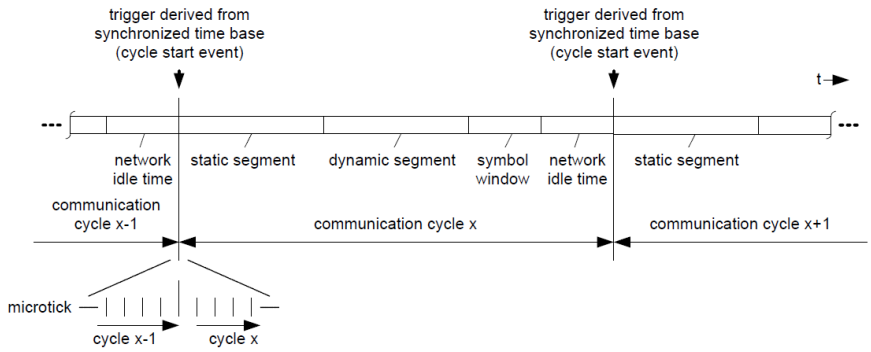
29. ábra: Egy kommunikációs ciklus felépítése és időzítés hierarchiája

A közeghozzáférés legmagasabb szintjén a kommunikációs ciklus áll, amely a következő részeket tartalmazza:

- statikus szegmens (static segment),
- dinamikus szegmens (dynamic segment),
- szimbólum ablak (symbol window),
- hálózati holtidő (network idle time).

Az egyes szegmensek a következő szinten időrésekre (slot) vannak osztva, amelyek egy-egy eszköz számára vannak fenntartva. Ezek az időrések ún. „macrotickekből” (ütemjelekből) állnak, amelyek minden egyes hálózati eszköznél szinkronban vannak egymással. Ehhez a FlexRay vezérlők aktív szinkronizációt folytatnak egymással és állítják az órájukat. A „macrotickek” hossza konfigurálható egy adott hálózatban, azonban a gyakorlatban legtöbbször 1 ms időtartamot szoktak használni. Mivel a „macrotickek” szinkronizáltak, így természetesen a hozzájuk kapcsolódó adat is szinkronizáltan továbbítódik.

A statikus szegmens szolgál a determinisztikus adatküldésre, míg a dinamikus szegmens a CAN-hez hasonló esemény-alapú adatok küldésére használható. A szimbólum ablak hálózat karbantartási és vezérlési jelzésekre használatos, például a hálózat indítása során. Végül a holtidő az óraszinkronizálásra van fenntartva, ilyenkor nincs adatküldés. Gyakorlatilag a másik három rész által fel nem használt „macrotickeket” tartalmazza. Ez idő alatt az eszköz kiszámolhatja és elvégezheti a szükséges órakorrekciót.



30. ábra: A kommunikáció periodikus működése

1.4.4.1.1 Statikus szegmens

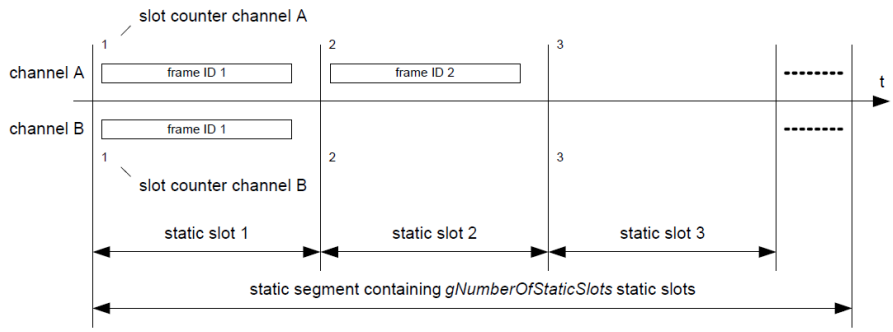
A statikus szegmensen belül statikus időosztásos többszörös hozzáférési technikával irányítják az adatátvitelt. A szegmensen belül minden időrés (slot) egyforma, statikusan konfigurált időtartammal és hosszal rendelkezik.

A szabvány a sztatikus szegmensen belüli kommunikációval kapcsolatban a következő fontos megszorításokat teszi:

Ha az eszköz rendelkezik egy vagy több „kulcs időrésessel” (key slot), akkor az eszköznek adatkeretet kell küldeni az összes kulcs időrésében, az összes használt csatornán és az összes kommunikációs ciklusban.

A többi időrésben küldhet vagy egyik, vagy mindkét csatornán.

Egy adott kommunikációs ciklusban csak egy eszköz küldhet adatot egy adott azonosítóval (frame ID) egy adott csatornán. Azonban eltérő kommunikációs ciklusokban megengedett, hogy több eszköz küldjön ugyanazzal az azonosítóval, ugyanazon a csatornán.

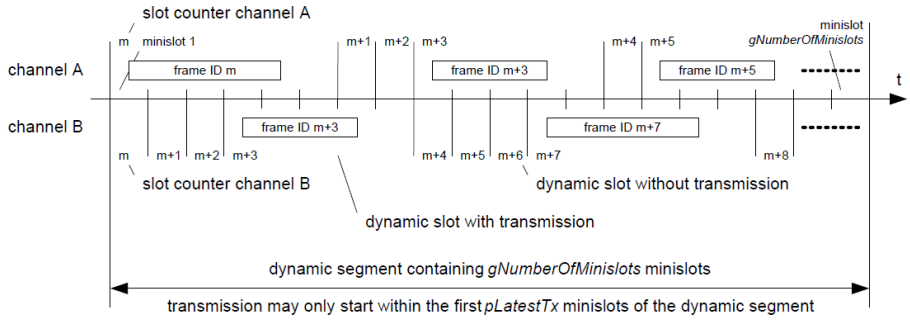


31. ábra: A statikus szegmens felépítése

Tehát a megfelelően konfigurált eszközök mindig a hozzájuk rendelt időrésben küldenek adatokat, ezzel garantálva a pontos időzítést. Amennyiben például egy ECU kikerül a hálózatból, mert lekapcsol, esetleg meghibásodik, akkor a hozzárendelt időrés üresen marad, ez a többi ECU működését és adatátviteli sebességét nem befolyásolja.

1.4.4.1.2 Dinamikus szegmens

A legtöbb beágyazott hálózat általában kisebb számban tartalmaz nagy sebességigényű üzenteket, míg nagyobb számban alacsonyabb sebességigényű, kevésbé kritikus üzeneteket. Azért, hogy a kommunikációs ciklus ne tartalmazzon túlságosan nagy számban statikus időréseket, ezért a FlexRay szabvány lehetőséget nyújt az ún. dinamikus szegmens használatára, amely megfelelő az alkalmanként küldendő üzenetek számára.



32. ábra: A dinamikus szegmens felépítése

A dinamikus szegmensben az ún. mini-időrés (minislot) alapú eljárás használatos. Azaz a dinamikus szegmens mini időrésekből (minislot) épül

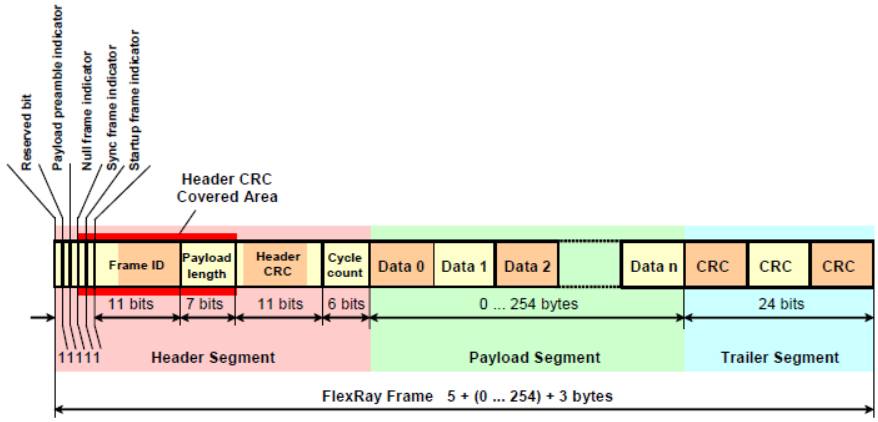
fel, amelyek általában egy „macrotick” hosszúak. A dinamikus szegmensben az időrések hossza változhat, annak érdekében, hogy alkalmazkodni lehessen az adatkereketek változó hosszához. A keretek hossza különböző lehet a különböző időrésekben egyazon kommunikációs cikluson belül, és szintén különböző lehet azoknál az időréseknél, amelyek ugyanolyan azonosítóval rendelkeznek különböző kommunikációs ciklusokban. Azonban a szegmens hossza fix, így behatárolt a dinamikus adatok teljes mérete.

Az adatok priorizálását megoldandó a „minislotok” előre hozzárendelődnek minden egyes adatkerethez, amely jogosult a dinamikus szegmensben történő adattovábbításra. A magasabb prioritású adat a dinamikus szegmens elejéhez közelebb lévő „minislotot” kapja.

1.4.4.2 Adatformátum

A FlexRay protkoll használata során az eszközök keretekbe foglalják a küldendő információt. A szabvány szerint egy adatkeret a következő 3 részből kell álljon:

- fejléc (Header Segment),
- hasznos adat (Payload Segment),
- lezárás (Trailer Segment).



33. ábra: A FlexRay adatkeret felépítése

Az adatokat minden esetben ennek megfelelő sorrendben is kell továbbítani a hálózaton.

1.4.4.2.1 Fejléc (Header Segment)

A FlexRay „header segment” 5 bájtból áll, amely tartalmazza a státusz biteket, a keret azonosítóját, az hasznos bájtok számát, valamint egy CRC ellenőrző kódot.

A státusz bitek csoportját a következő elemekből áll:

„Reserved bit” (1 bit): Későbbi felhasználásra lefoglalt bit. A küldő fél nullára kell, hogy állítsa, a fogadó fél pedig eldobja a tartalmát.

„Payload preamble indicator” (1 bit): Jelzi, hogy a hasznos bájtok tartalmaznak-e ún. opcionális hálózati menedzsment vekort. Ha a keret a statikus szegmensben továbbítódik, akkor a bit jelzi (értéke 1), hogy hálózati menedzsment vektor található a „payload segment” elején. Amennyiben a dinamikus szegmensben helyezkedi el a keret, úgy a bit jelzi az üzenetazonosító (message ID) jelenlétét a „payload segment” elején. Amennyiben ezek az elemek (hálózati menedzsment vektor vagy üzenetazonosító) nem szerepelnek az adatkeretben, úgy a bit értéke 0.

„Null frame indicator” (1 bit): Nulla értékkel jelzi, hogy az adott keret tartalmaz-e hasznos adatot. A FlexRay szabvány ún. „null frame”-nek hívja azt az adatkertet, amelynek a „payload segment”-je nem tartalmaz hasznos adatot. Ebben az esetben a hasznos bájtok mindegyike nulla, valamint a „payload preamble indicator” bit is 0 értékű. A „null frame”-et a FlexRay protokoll egyes speciális esetekben használja. Amennyiben a bit értéke 1, úgy a keret hasznos adatokat tartalmaz.

„Sync frame indicator” (1 bit): A bit 1-es értékkel jelzi, ha az adott keret ún. „sync frame”, azaz szinkronizációs keret. Az ilyen típusú adatkeretet az eszközök az óraszinkronizálásnál használják.

Startup frame indicator (1 bit): A bit 1-es értékkel jelzi, ha az adott keret ún. „startup frame”, azaz indító keret. Az ilyen típusú adatkertet az eszközök a bekapcsolás után egyszer küldik le a hálózatra. Az indító keretben a „sync frame indicator” bitet is mindig 1-be kell állítani, azaz az indító keret mindig szinkronizációs keret is egyben.

A következő rész az üzenetazonosító (frame ID), amely azt az időrést azonosítja, amelyben a keretet továbbítani kell. Egy azonosító legfeljebb egyszer használható egy csatornán, egy adott kommunikációs cikluson

belül. Az azonosító hossza 11 bit és 1-2047-ig vehet fel értéket, a 0-s azonosító érvénytelen.

Ezt követi a hasznos adathossz (payload length) mező, amely 7 bit hosszú, és a hasznos bájtok számát kettővel elosztva tárolja. A mező értékének állandó és megegyezőnek kell lennie minden egyes keret esetében a staikus szegmensen belül egy adott kommunikációs ciklus esetén. A dinamikus szegmensen belül ez eltérő lehet, és akár ciklusról ciklusra változhat. A dinamikus szegmensben megengedett, hogy az értékeke az egyes csatornákon is eltérőek legyenek.

A fejlécet tartalmaz egy CRC kódot (header CRC), amely a szinkronizációs keret, az indító keret jelzőbitjeiből, az üzenetazonosítóból, valamint a hasznos adathossz értékekből számolódik. Adatküldésnél a CRC-t nem kell a kommunikációs vezérlőnek kiszámolnia, hanem az már a konfiguráció során kiszámításra kerül, mivel a számításhoz használt adatok konstansok. Természetesen a fogadott adatkereteknél ellenőrzési célokból kiszámítandó, mégpedig a következő CRC polinom segítségével:

$$X^{11}+X^9+X^8+X^7+X^2+1 = (X+1)*(X^5+X^3+1)*(X^5+X^4+X^3+X+1)$$

A generáláshoz használt regiszter kezdeti vektora 0x01A értékű kell legyen. A 11 bites CRC polinom egy BCH (31,20) hibajavító kódot generál, amelynek minimális Hamming távolsága 6. A kódszó tartalmazza a 20 bitnyi védendő adatot, és a 11 bites CRC-t.

Végül a fejlécet egy 6 bites ciklusszámláló zárja le, amely az eszköz által számolt kommunikációs ciklusok értékét tartalmazza.

1.4.4.2.2 Hasznos adat (Payload Segment)

A FlexRay ún. „Payload” szegmense 0-254 bájtot tartalmazhat. Az adatnak két bájtos szavakból (word) kell állnia, tehát csak páros számú bájt méretű lehet. Az adatok indexálása a fejléc szegmens után, nulla indexeléssel kezdődik. Az adatátvitel a 0. bájttal kezdődik, úgy hogy a bájton belül az MSB az első.

A dinamikus szegmens adatmezőjének első két bájtja opcionálisan üzenet azonosítóként (message ID) használható, melyet a fent leírt „Payload preamble indicator” bit jelez. A statikus szegmens 0-12-es bájtjai pedig

hálózati menedzsment vektorként használhatók, melyet ugyanez a bit jelez.

Az hasznos adat védelméről a keret végén (Trailer Segment) lévő 24-bites CRC gondoskodik, amely 248 bájtos adathosszig 6-os, efölött 4-es Hamming távolságot biztosít.

1.4.4.2.3 Záró szegmens (Trailer Segment)

A FlexRay keret végén található szegmens egy mezőt tartalmaz, amely egy 24-bites CRC az egész keretre (Header Segment + Payload Segment), annak összes mezőjét beleértve.

1.4.5 Hálózat és adatleírás

A Flexray hálózatok tervezéséhez, konfigurációjához, monitorozásához és adattartalmának leíráshoz a Field Bus Exchange Format (FIBEX) a jelenleg elfogadott iparági szabvány. A szabványt, az autógyárak és beszállítók által létrehozott, az Association for Standardisation of Automation and Measuring Systems nevű szervezet fejleszti és gondozza. A FIBEX célja alapvetően a Flexray és MOST buszrendszerek szabványos leírása, azonban a 3.1-es verziótól (jelneleg 4.0-nál tart) a CAN, TTCAN, LIN és Ethernet rendszereket is támogatja.

Ábrajegyzék

Nincs ábrajegyzék-bejegyzés.

Táblázatjegyzék

Irodalomjegyzék

Felhasznált irodalom

1. **International Telecommunication Union.** http://www.itu.int/ITU-T/. [Online]

2. **International Organization for Standardization.** http://www.iso.org/iso/home.html. [Online]

3. —. ISO/IEC 7498-1:1994. *Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model.* 1994.

4. **Kónya, László.** *Számítógép-hálózatok.* Budapest : LSI Oktatóközpont, 2002.

5. *Evolution of local interconnect network (LIN) solutions.* **Ruff, M.** Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th : ismeretlen szerző, 6-9 Oct. 2003., 5. kötet, old.: 3382- 3389.

6. **LIN Consortium.** *LIN Specification Package Revision 2.1.* 2006.

7. **International Organization for Standardization.** ISO9141. *Road vehicles - Diagnostic systems - Requirement for interchange of digital information.* 1989.

8. —. ISO 11898-1. *Road vehicles - Controller area network (CAN) - Part1: Data link layer and physical signalling.* 2003.

9. —. ISO 11898-3. *Road vehicles - Controller area network (CAN) - Part 3: Low-speed, fault-tolerant, medium-dependent interface.* Switzerland : International Organization for Standardization, 2006.

10. —. ISO 11898-2. *Road vehicles - Controller area network (CAN) - Part 2: High-speed medium access unit.* Switzerland : International Organization for Standardization, 2003.

11. **BOSCH.** CAN Specification, Version 2.0. 1991.

12. **Szalay, Zsolt.** Gépjármű Elektronika (jegyzet). Budapest : BME Gépjárművek Tanszék, 2005.