

3. fejezet - CAN: Controller Area Network

Tartalom

[3.1. A CAN protokoll jellemzői és felépítése](#)

[3.1.1. A CAN protokoll jellemzői](#)

- [3.1.1.1. Több-mester \(Multimaster\)](#)
- [3.1.1.2. Üzenetközpontúság](#)
- [3.1.1.3. Nem-destruktív arbitrációs mechanizmus](#)
- [3.1.1.4. Üzenetszórás \(Broadcast\)](#)
- [3.1.1.5. Eseményvezérelt](#)
- [3.1.1.6. Távoli válaszkérés](#)
- [3.1.1.7. Rugalmasság](#)
- [3.1.1.8. Valós idejű megvalósíthatósági lehetőség](#)
- [3.1.1.9. Alacsony költség](#)
- [3.1.1.10. Megbízhatóság](#)
- [3.1.1.11. Hiba detektálás a kommunikációs médium szintjén](#)
- [3.1.1.12. Nyugtázas](#)
- [3.1.1.13. Teljes rendszerre nézve konzisztens üzenetátvitel](#)
- [3.1.1.14. Csomópontok közötti szinkronizáció](#)
- [3.1.1.15. Széles eszközválaszték](#)

[3.1.2. A CAN alkalmazási területei](#)

[3.1.3. Szabványosítás](#)

[3.1.4. A CAN protokoll felépítése](#)

- [3.1.4.1. Fizikai réteg](#)
- [3.1.4.2. Adatkapcsolati réteg](#)

[3.2. A CAN protokoll Fizikai rétege](#)

[3.2.1. CAN busz felépítése](#)

[3.2.2. CAN csomópont felépítése](#)

- [3.2.2.1. A CAN protokollvezérlők csoportosítása](#)

3.2.2.2. CAN csatlakozó

3.2.3. Arbitráció

3.2.4. Bitreprezentáció a CAN-en

3.2.4.1. Bitszint meghatározása

3.2.4.2. Bitkódolási technikák

3.2.4.3. Bitidőzítés

3.2.4.4. Bitsebesség és a buszhossz viszonya

3.2.4.5. Szinkronizálás

3.2.4.6. Oszcillátor

3.2.4.7. Bitidő paramétereinek kiszámítása egy példán keresztül

3.3. A CAN protokoll Adatkapcsolati rétege

3.3.1. CAN üzenetkeretek

3.3.1.1. Adathordozó üzenet

3.3.1.2. Adatkérő üzenet

3.3.1.3. Hibaüzenet

3.3.1.4. A túlcordulás üzenet

3.3.1.5. Üzenetek közötti mező

3.3.2. Üzenetek késleltetése

3.4. Hibakezelés a CAN hálózaton

3.4.1. Üzenet jóváhagyás

3.4.2. Hibatípusok, és Hibafelismerés

3.4.2.1. Bithiba – Bitellenőrzés

3.4.2.2. Kitöltési hiba – Bitbeszúrás ellenőrzés

3.4.2.3. CRC hiba – CRC ellenőrzés

3.4.2.4. Formai hiba – Üzenetkeret ellenőrzés

3.4.2.5. Nyugtázási hiba – Nyugtázás ellenőrzés

3.4.3. Hibafelismerési képesség

3.4.3.1. Számítási példa felderítetlen hibára

3.4.4. Hibaforrás megszüntetése, a CAN csomópont állapotgépe

3.5. CAN üzenet válaszideje

3.5.1. Adott m üzenet legrosszabb esetben vett válaszidejének analízise

3.5.2. Válaszidőt befolyásoló tényezők

3.5.3. CAN válaszidő jitterének minimalizálása

3.5.3.1. Bitbeszúrás minimalizálása az Arbitrációs mezőben

3.5.3.2. Bitbeszúrás minimalizálása az adatmezőben

A '80-as évek elején a Bosch mérnökei megvizsgálták a létező hálózati protokollokat a személyautókban történő felhasználhatóságuk szempontjából. Mivel az akkor használatos hálózati protokollok közül egyet sem találtak megfelelőnek, 1983-ban új buszrendszer tervezését kezdték meg, valamint hozzá illő hálózati protokoll fejlesztésébe fogtak. Fő céljuk a vezetékek számának csökkentése és a biztonság növelése volt, melyeket az 1986-os^[10] megszületésekor a CAN (Controller Area Network) protokoll sikeresen alkalmazott.

Napjainkra a CAN széles körben elterjedt kommunikációs protokollá vált az ipar számtalan területén. Ennek köszönhetően nagyon sok cég gyárt és forgalmaz hardver és szoftver eszközöket a CAN-hez kapcsolódóan.

3.1. A CAN protokoll jellemzői és felépítése

3.1.1. A CAN protokoll jellemzői

3.1.1.1. Több-mester (Multimaster)

3.1.1.2. Üzenetközpontúság

3.1.1.3. Nem-destruktív arbitrációs mechanizmus

3.1.1.4. Üzenetszórás (Broadcast)

3.1.1.5. Eseményvezérelt

3.1.1.6. Távoli válaszkérés

3.1.1.7. Rugalmasság

3.1.1.8. Valós idejű megvalósíthatósági lehetőség

3.1.1.9. Alacsony költség

3.1.1.10. Megbízhatóság

3.1.1.11. Hiba detektálás a kommunikációs médium szintjén

3.1.1.12. Nyugtázs

3.1.1.13. Teljes rendszerre nézve konzisztens üzenetátvitel

3.1.1.14. Csomópontok közötti szinkronizáció

3.1.1.15. Széles eszközválaszték

3.1.2. A CAN alkalmazási területei

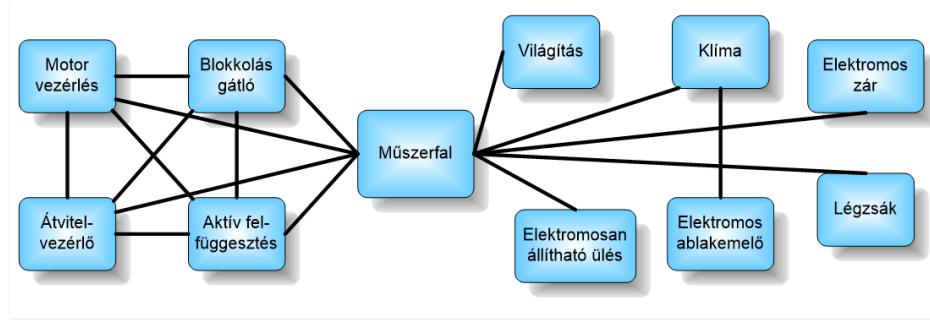
3.1.3. Szabványosítás

3.1.4. A CAN protokoll felépítése

3.1.4.1. Fizikai réteg

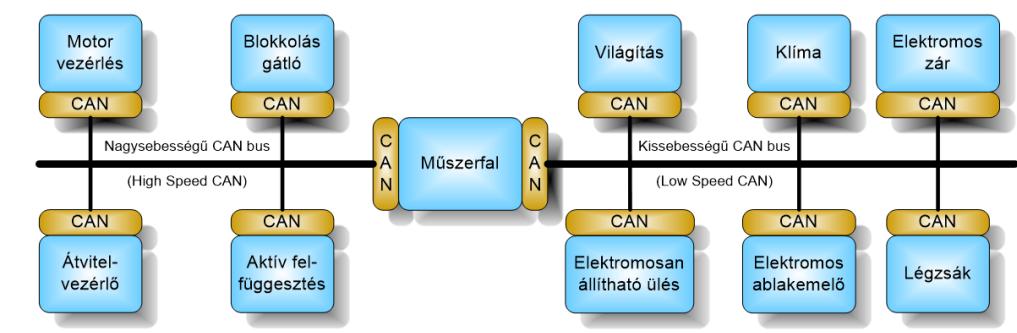
3.1.4.2. Adatkapcsolati réteg

A CAN olyan hálózatot reprezentál, amelyben a vezérlőeszközöket (pl.: mikrokontrollerek, DSP) egy soros buszrendszer köti össze, ezáltal alkalmas elosztott irányító rendszerek megvalósítására. A vezérlőeszközök a hálózat csomópontjai (node), amelyek üzenetkeretekkel/ üzenetekkel (message frame) kommunikálnak egymással. Az üzenetkeret fajtái szerint többféle felépítéssel és különféle mezőkkel rendelkezhetnek ([[3.3.1. szakasz](#)] fejezet).



3.1. ábra - A CAN előtt alkalmazott rendszerstruktúra

A rendszer kifejlesztését az motiválta, hogy az igények növekedésével, az egymással kommunikáló elektronikus eszközök illetve csomópontok száma egyre nőtt a különböző rendszereken (járművek, hajók, gyártósorok, repülőgépek) belül. A CAN megjelenése előtt ezeket a csomópontokat közvetlenül kötötték össze, ami a vezetékek bonyolult rendszerét eredményezte ([3.1. ábra]). Ezt a közvetlen összeköttetésen alapuló hálózatok többségét felváltotta a CAN, melynek kialakítását az [3.2. ábra] szemlélteti.



3.2. ábra - CAN alkalmazásával előálló rendszerfelépítés

3.1.1. A CAN protokoll jellemzői

- 3.1.1.1. Több-mester (Multimaster)
- 3.1.1.2. Üzenetközpontúság
- 3.1.1.3. Nem-destruktív arbitrációs mechanizmus
- 3.1.1.4. Üzenetszórás (Broadcast)
- 3.1.1.5. Eseményvezérelt
- 3.1.1.6. Távoli válaszkérés
- 3.1.1.7. Rugalmasság
- 3.1.1.8. Valós idejű megvalósíthatósági lehetőség
- 3.1.1.9. Alacsony költség
- 3.1.1.10. Megbízhatóság

- [3.1.1.11. Hiba detektálás a kommunikációs médium szintjén](#)
- [3.1.1.12. Nyugtázás](#)
- [3.1.1.13. Teljes rendszerre nézve konzisztens üzenetátvitel](#)
- [3.1.1.14. Csomópontok közötti szinkronizáció](#)
- [3.1.1.15. Széles eszközválaszték](#)

3.1.1.1. Több-mester (Multimaster)

Nincs kiválasztott busz-vezérlő (bus master), minden csomópont teljesen egyenrangú, képes az üzeneteit önállóan, bármely másik csomópont segítsége nélkül továbbítani az adatbuszon (data bus), ha az felszabadult. Egy csomópont leállása esetén az egész rendszer nem válik működésképtelenné, csak a teljesítőképessége csökken.

3.1.1.2. Üzenetközpontúság

Az üzenetek azonosítása nem a küldő vagy a fogadó csomópont címe alapján történik (mint általában a többi buszrendszerenél), hanem egyedi azonosító (identifier) alapján, amit az üzenetek a hordozott információ fontossága szerint kapnak. Így az üzenet azonosítója (Azonosító mezeje) határozza meg az üzenet prioritását, valamint közvetlenül szerepet játszik a buszáért való versengés eldöntésében is. E versengési folyamat az arbitráció (arbitration) ([[3.2.3. szakasz](#)] fejezet).

3.1.1.3. Nem-destruktív arbitrációs mechanizmus

A CAN ún. prioritásos CSMA/CD+CR^[11] médiaelérési technikát használ. Az adatbuszt elérni kívánó csomópontok várak a busz felszabadulásáig, majd megkezdik a kommunikációt, amely egy speciális Üzenet kezdete bittel ([[3.3.1.1. szakasz](#)] fejezet) indul és egyben szinkronizálja az összes kommunikációs partnert. Ezután történik az üzenetazonosító továbbítása. Több partner egyidejű adási szándéka esetén ebben a szakaszban történik az ütközés feloldása, bitszintű arbitrációval. Ezt a technikát nem-destruktív arbitrációs mechanizmusnak (non-destructive arbitration) nevezzük, mivel a „vesztes” csomópont úgy mond le busz-igényéről, hogy emiatt az átvitt magasabb prioritású üzenet nem sérül. Ez annyit jelent, hogy mindenmű késleltetés nélkül a legmagasabb prioritású üzenet továbbítódik a buszon.

3.1.1.4. Üzenetszórás (Broadcast)

A CAN buszon telefonkonferencia szerű kommunikáció zajlik. Ahogy a telefonkonferencia bármely résztvevője szabadon elmondhatja a mondanivalóját, amelyet minden más résztvevő hall, éppúgy bármelyik csomópont elküldheti az üzenetét a CAN buszon, és azt minden egyes, ugyanarra a buszra csatlakozó csomópont megkapja. Azonban a „bemondott” információ csak akkor értelmezhető, ha egyidőben csak egy résztvevő beszél. Hogy éppen melyik konferenciatagé ez a kiváltság, azaz melyik csomópont használhatja a buszt a saját üzenete elküldésére, azt az arbitráció folyamata hivatott eldönten.

A csomópontok az üzenetazonosítók alapján döntik el, hogy pufferelik-e azt későbbi kiértékelés céljából, vagy figyelmen kívül hagyják üzenetszűrés (message filtering) által. Az üzenetszűrőt a felhasználói alkalmazás állítja be. ([[3.2.2.1.1. szakasz](#)] fejezet)

3.1.1.5. Eseményvezérelt

A kommunikáció adott esemény bekövetkezésének (új információ generálódott egy csomópontban) hatására kezdődik el. Az új üzenettel rendelkező csomópont maga kezdi meg az átvitelt. Így jelentős kommunikációs időt takarít meg például azokhoz a rendszerekhez képest, amelyekben a csomópontok minden ciklusban adott időszekettel rendelkeznek, melyben az új információkat elküldhetik. Ugyanis ez

esetben, ha nincs új információja egy csomópontnak, akkor ez az időszelet kárba vész, míg esetlegesen egy másik, új információval rendelkező eszköznek várnia kell, amíg sorra kerül.

Lehetőség van ciklikus információtörzserére is, ekkor belső óra, vagy egy másik csomópont kezdeményezi a kommunikációt (ISO 11898-4: [[3.1.3. szakasz](#)] fejezet).

3.1.1.6. Távoli válaszkérés

Az eseményvezérelt kommunikációt kiegészítve a CAN lehetőséget biztosít ún. „távoli válaszkérő üzenetek” küldésére. Ezek segítségével egy fogadó csomópont kérheti a számára szükséges információ elküldését a megfelelő küldő csomóponttól. A kérés, és a válasz külön üzenetet képez. ([[3.3.1.2. szakasz](#)] fejezet) Föleg a csomópontok állapotának (aktív/inaktív) lekérdezésére használják ezt a technikát.

3.1.1.7. Rugalmasság

A csomópontokat dinamikusan rákapcsolhatjuk, illetve leválaszthatjuk a buszról anélkül, hogy a többi csomópont kommunikációját zavarnánk, így a rendszer rugalmasan alakítható.

- Egy rendszeren belül 32 csomópont lehet szabványos buszmeghajtók esetén, valamint 64-128 darab lehet alkalmazás-specifikus meghajtók esetén.
- Üzenetek száma a rendszerben: Standard üzenetformátum esetén 2^{11} (= 2048), Kiterjesztett üzenetformátum esetén 2^{29} (=536 870 912) darab különböző azonosítójú üzenet lehetséges.
- Adatmennyiség üzenetenként: 0-8 bajt. Ezek a rövid üzenetek elegendők a járművekben valamint beágyazott illetve automatizált gyártó rendszerekben történő kommunikációhoz, és egyben garantálják a lehető legrövidebb buszelérési időt a nagy prioritású üzenetek számára, valamint erős zavarású közigben történő kommunikáció esetén a zavaró jellel való összeütközés kisebb valószínűségét.
- Maximális üzenethossz: beszúrt bitekkel ([[3.2.4.1. szakasz](#)] fejezet) együttes 117 bit standard üzenetformátum esetén, 136 bit kiterjesztett üzenetformátum esetén.
- Bitráta: 5kbit/s és 1Mbit/s között programozható (a buszhossztól függően).

3.1.1.8. Valós idejű megvalósíthatósági lehetőség

Az adattovábbítás maximális sebessége 1Mbit/s (40m-es buszhossznál), az üzenetek rövidek, a késleltetési idő maximálva van, az arbitráció ([[3.2.3. szakasz](#)] fejezet) pedig gyors. Az utóbbi tulajdonságok alkalmassá teszik a CAN rendszert a valós idejű események (pl.: ABS, motorvezérlés) irányítására.

3.1.1.9. Alacsony költség

A kivitelezéshez szükséges eszközökre nagy igény van az ipar különböző területein, ezért a sorozatgyártás alacsony árat és kedvező teljesítmény-ár viszonyt eredményez. A csavart érpár, amelyet a CAN rendszereknél a leggyakrabban használnak, szintén olcsó, mert ez az egyik leggyakoribb buszfajta. A rendszer üzemeltetési költségének csökkentése érdekében a csomópontok átállhatnak ún. „alvó állapotba” (sleep mode), amely azt jelenti, hogy belső aktivitásuk megszűnik és lekapcsolják a buszmeghajtókat, ezáltal csökkentve a rendszer

áramfogyasztását. Az „alvó állapot” követi az ún. „ébredési fázist-t” (wake-up), aminek következtében a belső aktivitás újra indul. A rendszernek lehetősége van arra, hogy egy speciális azonosítóval rendelkező üzenet elküldésével aktiváljon egy csomópontot.

3.1.1.10. Megbízhatóság

Kifinomult hibadetektáló és hibakezelő mechanizmusokkal rendelkezik, mint például:

- 15 bites, 6-os Hamming-távolságú CRC-vel (Cyclic Redundancy Check), amely 5 hibás bit felismerését teszi lehetővé üzenetenként.
- Nem rendszeres hibák helyreállítása a hibás üzenetek automatikus újraküldésével.
- Ismétlődő hibák kiküszöbölése a hibás csomópont kikapcsolásával, ami determinisztikussá teszi a rendszer esetleges hibák utáni helyreállásának idejét.

Az elektromágneses interferenciákra alacsony az érzékenysége.

A rendszer garantálja, hogy a küldő-csomópont által elküldött adatok megegyeznek a fogadó-csomópontok által fogadott adatokkal. Átlagos terhelés mellett statisztikailag 1000 év alatt egy olyan hiba fordul elő, amelyet a rendszer nem észlel.

3.1.1.11. Hiba detektálás a kommunikációs médium szintjén

A CAN vezérlők (CAN controller) sok fajta vezetékhibát ismernek, és definiálnak: szakadás, testzárlat, egyéb zárlatok. A protokoll nem írja le, hogy mi a teendő a fenti hibák esetén, de az újabb CAN vezérlőkben legalább a fenti esetek egyikének kezelése implementálva van.

3.1.1.12. Nyugtázás

Az üzenetek globális Nyugtázó mezővel (Acknowledgement field) ([[3.3.1.1.6. szakasz](#)] fejezet) rendelkeznek, amely jelzi a küldő csomópontnak, hogy legalább egy kommunikációs partnerhez hibátlanul megérkezett az üzenet. Így a küldő információt kap arról, hogy még a buszhöz van-e csatlakoztatva, vagy sem. Az üzenetszórás-jellegű üzenettovábbítás következtében minden csomópont nyugtázó jellel válaszol, ha nem észleltek hibát.

3.1.1.13. Teljes rendszerre nézve konzisztens üzenetátvitel

A rendszer minden egyes elküldött üzenetre garantálja, hogy azt vagy minden csomópont elfogadja, vagy minden csomópont elutasítja. Ha legalább egy vevő hibát észlel a fogadás során, akkor egy Hibaüzenettel (Error frame) ([[3.3.1.3. szakasz](#)] fejezet) rögtön megszakítja az átvitelt, és jelzi a többi fogadó állomásnak, hogy hagyják figyelmen kívül az üzenetet, a küldőnek pedig, hogy küldje el ismét azt. Ez eredményezi a teljes üzenet-konzisztenciát a rendszerben, azaz vagy minden egyes csomópont megkapja ugyanazt az információt ugyanabban az időpillanatban, vagy egyik sem. Ez az elosztott rendszerekben igen fontos tulajdonság, mivel így garantálható, hogy a különböző mikrokontrollerek ne dolgozzanak ugyanahoz a változóhoz tartozó eltérő adatokon egyidőben.

3.1.1.14. Csomópontok közötti szinkronizáció

Üzenetek sikeres küldése és fogadása után az összes résztvevő csomópontban egy megszakítás (interrupt) generálódik, amit fel lehet használni az elosztott vezérlőrendszer óráinak beállításához. Az órák szinkronizáltsága elengedhetetlen feltétele az elosztott valós idejű alkalmazások működésének.

3.1.15. Széles eszközválaszték

Manapság a legtöbb mikrokontroller gyártó (a legnevesebbek pl. Intel, Motorola, Siemens, Philips) kínálatában megtalálhatók CAN chippek is, amelyek a nagy választék és az árverseny miatt meglehetősen olcsón beszerezhetők, és így gyorsan elterjednek.

3.1.2. A CAN alkalmazási területei

A fentieket összefoglalva tehát megállapíthatjuk, hogy a CAN ára alacsony, a kiépített rendszer megbízható, és valós idejű környezetben is alkalmazható, flexibilis protokoll. Ezen igen előnyös tulajdonságokat figyelembe véve érthető, hogyan válhatott az autóipari és az ipari automatizálási alkalmazások területén napjaink vezető protokolljává, amely egyre újabb és újabb területeket hódít meg, mint például orvosi elektronika, háztartási eszközök, épületautomatizálás, irodai automatizálás, vasúti rendszerek, hajózás, mezőgazdasági gépek, repülőgép-elektronika, PLC^[12], robotvezérlés, intelligens motorvezérlés, valamint őrtechnológia. Tehát a CAN igen széles körben használt rendszer, mely egyre újabb és újabb területeket hódít meg.

1998-ban 31 millió, 1999-ben 57 millió új CAN csomópontot helyeztek üzembe világszerte. A 2001-es évben ez a szám meghaladta a 200 milliót, majd 2003-ban a 350 milliót [14].

A globális elterjedéshez és használhatósághoz, a kompatibilitási problémák elkerüléséhez azonban szükség volt a protokoll szabványosítására.

3.1.3. Szabványosítás

Három évvel az első CAN vezérlő chippek megjelenése^[13] után, 1990-ben, a Bosch féle CAN specifikációt nemzetközi szabványosításra nyújtották be. Így született meg az ISO (International Standardization Organization) és a SAE (Society of Automotive Engineers) együttműködése során az ISO 11898 nemzetközi szabvány. A különböző megoldások egységesítéséhez, valamint a CAN további technikai fejlődésének biztosításához szükség volt egy – felhasználóból és gyártókból álló – semleges platformra. 1992 márciusában hivatalosan is megalakult a CAN in Automation (CiA) nemzetközi felhasználói és gyártói csoport. A CiA munkája során leszükítette a legalsó OSI réteg specifikációját vezeték, csatlakozó és Adó-vevő chip (Transceiver chip) ajánlásra, kidolgozta a CAL-t (CAN Application Layer), amely az ISO/OSI referencia modellhez képest a CAN-ból addig hiányzó Alkalmazási réteget pótolja. Később olyan további CAN Alkalmazási rétegek kidolgozásával foglalkoztak, mint a SDS (Smart Distributed System), DeviceNet stb.

1993-ra megjelent az ISO 11898-as CAN szabvány, amely a protokoll 11 bites azonosítójú, standard formátumú üzenetein túl a Fizikai réteget is definiálja, 1Mbit/s-os átviteli sebességgel (CAN Specification 1.2).

Az üzenetek fajtáinak növekedésével szükségessé vált a 29 bites azonosítójú, kiterjesztett formátumú üzenetek ([3.3.1. szakasz] fejezet) specifikálása, melyet a CAN Specification 2.0 definiál, amelyet az ISO 11898 kiegészítéseként rögzítettek. Maga a CAN Specifikáció 2.0 a Bosch mérnökei által már 1991 szeptemberében megalkotásra került, és az alábbi két fő fejezetből valamint függelékből áll:

- CAN Specifikáció 2.0 „A fejezet” (Part A), amely csak a standard formátumú üzeneteket definiálja. Ez magába foglalja a korábbi CAN Specifikáció 1.2-t (CAN Specification 1.2).
- CAN Specifikáció 2.0 „B fejezet” (Part B), pedig a standard – és a kiterjesztett formátumú üzeneteket együttesen specifikálja.
- CAN Specifikáció 2.0 Függelék útmutatást ad arra vonatkozólag, hogy hogyan érdemes megvalósítani a CAN protokollt úgy, hogy megfeleljen a szabvány A vagy B fejezetében leírtaknak.

Az átdolgozott CAN specifikációk szabványosítása napjainkban is folyik. Az alábbi szabványokat a következő feladatkörökre dolgozták ki:

- ISO 11898-1: a CAN Adatkapcsolati rétegének és a fizikai jelterjedésnek (physical signaling) a leírása.
- ISO 11898-2: a CAN nagysebességű Fizikai réteget jellemzi, amely leginkább az autóiparban és ipari vezérléseknel használatos (two-wire balanced signaling).
- ISO 11898-3: a CAN alacsony sebességű, hibatűrő Fizikai réteget rögzíti.
- ISO 11898-4: a CAN idővezérelt kommunikációja (TTCAN = Time-Triggered CAN), ahol a CAN Adatkapcsolati rétegen található rendszerőre ütemez az üzeneteket (messages).
- ISO 11898-5: a CAN nagysebességű Fizikai rétegének leírása alacsony energiaszintű módban (Low-Power Mode).
- ISO 11898-6: a CAN nagysebességű Fizikai rétegének leírása szelektív felébresztő funkció (selective wake-up function) esetén.
- ISO 11992-1: a teherautókra, trélerekre szabott CAN protokoll leírása.
- ISO 11783-2: 250kbit/s bitsebességű mezőgazdasági szabvány
- SAE J1939-11: 250kbit/s bitsebességű árnyékolt csavart érpár (STP – Shielded Twisted Pair) közeg leírása.
- SAE J1939-15: 250kbit/s bitsebességű árnyékolatlan csavart érpár (UTP – Unshielded Twisted Pair) közeg leírása, csökkentett réteggel.
- SAE J2411: Egyvezetékes CAN (SWC – Single-Wire CAN) megvalósításának leírása.

3.1.4. A CAN protokoll felépítése

3.1.4.1. Fizikai réteg

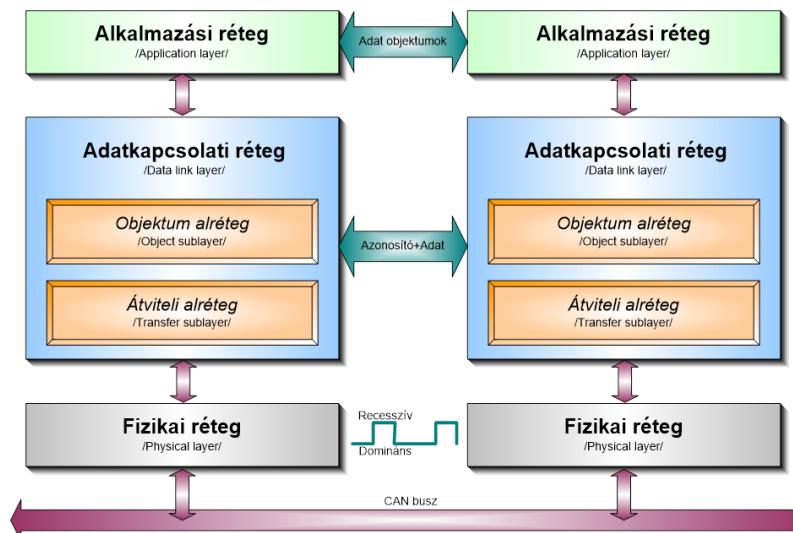
3.1.4.2. Adatkapcsolati réteg

Az adateszerét az ISO/OSI referencia modellból megismert módon egymásra épülő rétegek valósítják meg. minden réteg a közvetlenül alatta elhelyezkedő réteg szolgáltatásait felhasználva szolgáltatásokat nyújt a felette lévő rétegeknek. A valóságban a kommunikáció függőlegesen, logikailag viszont vízszintesen történik. minden réteg a társentitásával kommunikál, azaz a távoli rendszer azonos magasságban elhelyezkedő rétegevel. Ezt az elküldeni kívánt adat megfelelő „becsomagolásával” (megfelelő kerettel látja el azt) és lejjebb küldésével teszi meg, egészen a Fizikai rétegig. A fogadó rétegek felfelé továbbítják az adatot, minden szinten kicsomagolva azt. Ezzel az elgondolással a rétegek a feladataik alapján tisztán elkülöníthetők egymástól. minden réteg csak a közvetlen alsó és felső szomszédjait ismeri, és továbbítja azok üzeneteit módosítás és feldolgozás nélkül.

A CAN az ISO/OSI referencia modell hét rétegeből háromat definiál, a tervezés átláthatósága, valamint a megvalósítás hatékonysága és rugalmassága érdekében ([3.3. ábra]). A CAN protokoll a Fizikai és az Adatkapcsolati réteget definiálja, amelyet kiegészíthetnek magasabb szintű protokollok, melyeket az alkalmazási réteg ír le. Ilyen magasabb rendű protokollok:

- CANOpen
- Device Net

- Smart Distributed Systems (SDS)
 - J1939
 - CAN Application Layer (CAL)
 - CANKingdom
 - OSEK/VDX^[14]



3.3. ábra - A CAN protokoll felépítése a CAN Specifikáció 2.0 alapján

3.1.4.1. Fizikai réteg

A Fizikai réteg ([3.2. szakasz] fejezet) felelős a csomópontok közötti tényleges jeltovábbításért („bitek” továbbítása). A digitális jelek analóggá (és vissza) alakításán kívül ez a réteg végzi a busz paramétereinek megfelelő bit időzítést és szinkronizálást. A Fizikai rétegeknek az egész hálózaton belül azonosaknak kell lenniük.

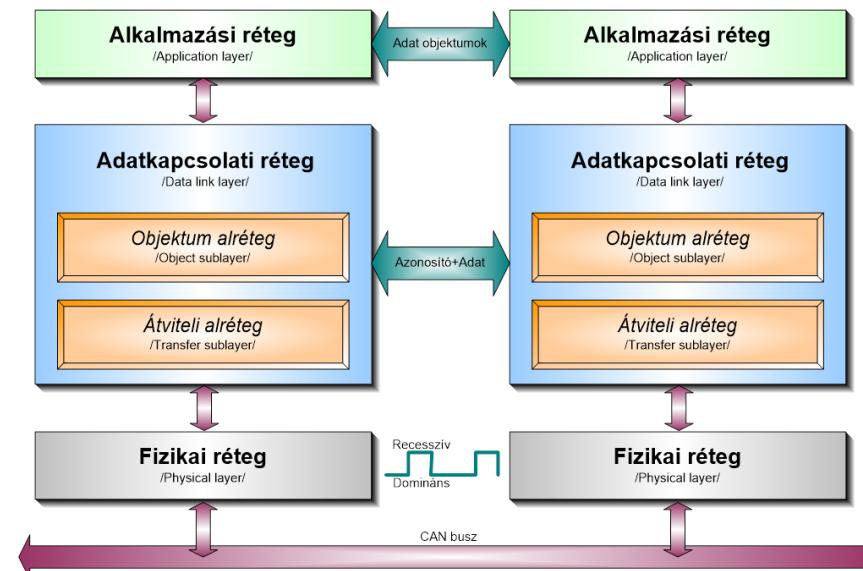
3.1.4.2. Adatkapcsolati réteg

A CAN Adatkapcsolati rétege ([3.3. szakasz] fejezet) a CAN Specifikáció 2.0 B része alapján Logikai kapcsolatvezérlésre (Logical Link Control) és Közeghuzzáférés vezérlésre (Medium Access Control) bontható. Illetve a CAN Specifikáció 1.2 (2.0 A) alapján a CAN Adatkapcsolati rétege Objektum alrétre (Object sublayer) és átviteli alrétre (Transfer sublayer) bontható.

Az Alkalmazási és Adatkapcsolati réteg közötti interfész képezi az Objektum alréteg, melynek feladata a busz felől kapott üzenetek szűrése (message filtering), azaz definiált feltételek alapján eldönti, melyeket fogadja el, és melyeket kell elvetenie. Ez a réteg végezi a túlcordulás

jelzését (overload notification), és kezeli a hibaállapotok felismerését, és a helyes működés visszaállítását.

Az Átviteli alréteg alkotja a CAN protokoll magját. Ez a réteg végzi el a megfelelő keretek alkotását, vezéri az arbitrációt, felismeri, jelzi és megszünteti a hibákat. Az ún. hiba elszigetelő entitás (Fault Confinement) felügyeli az Átviteli alréteg működését, ennek segítségével lehetséges az állandó meghibásodások megkülönböztetése az egyedi, ritkán fellépő hibáktól. Valamint e réteg dönti el, hogy vételi vagy adási folyamat indul-e.



3.4. ábra - A CAN protokoll felépítése a CAN Specifikáció 1.2 alapján

3.2. A CAN protokoll Fizikai rétege

3.2.1. CAN busz felépítése

3.2.2. CAN csomópont felépítése

3.2.2.1. A CAN protokollvezérlők csoportosítása

3.2.2.2. CAN csatlakozó

3.2.3. Arbitráció

3.2.4. Bitreprezentáció a CAN-en

3.2.4.1. Bitszint meghatározása

3.2.4.2. Bitkódolási technikák

3.2.4.3. Bitidőzítés

3.2.4.4. Bitsebesség és a buszhossz viszonya

3.2.4.5. Szinkronizálás

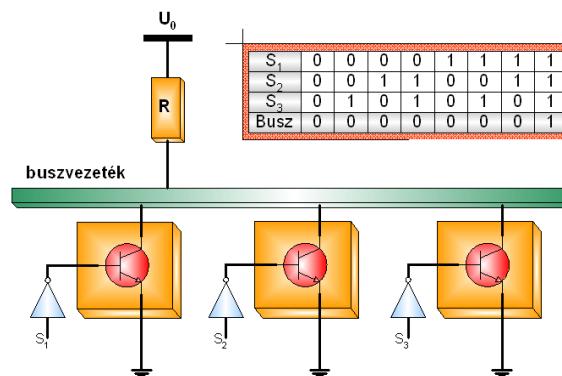
3.2.4.6. Oszcillátor

3.2.4.7. Bitidő paramétereinek kiszámítása egy példán keresztül

A Fizikai rétegnek egy-egy CAN hálózat egészére nézve azonosnak kell lennie. A CAN szabvány nem tesz kikötést a fizikai médium típusára, de jelenleg a csavart érpáron történő adatátvitel a legelterjedtebb, amit az ISO 11898 definiál. A két vezetéken átvitt jelek különbségei határozzák meg a busz logikai állapotát. Az egyiket CAN_magas (CAN_H), a másikat CAN_alacsony (CAN_L) vezetéknek nevezünk, a különbségi jel előjelének megfelelően. A busz minden végét ellenállással kell lezárni, hogy a vezetékek végéről történő jelvisszaverődés elkerülhető legyen. A lezáró ellenállás ajánlott nagysága 120 Ω (minimum 108 Ω , maximum 132 Ω). Ennek a megvalósításnak köszönhetően a rendszer érzéketlen az elektromágneses zavarásokra, valamint egyes zárlatok illetve szakadás okozta hibák esetén egyvezetékes módban továbbra is működőképes marad.

3.2.1. CAN busz felépítése

A CAN busz és a csomópontok a logikai jelszinteket tekintve gyakorlatilag az alábbi huzalozott-ÉS (wired-AND) konfigurációnak megfelelően viselkednek ([3.5. ábra]).



3.5. ábra - Logikai szintek huzalozott-ÉS szerkezetű megvalósítása

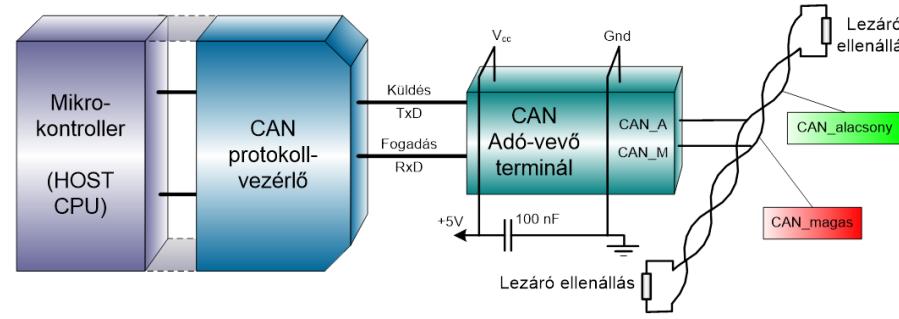
A csomópontok logikai 1-est továbbíthatnak a tranzisztor kikapcsolásával (U_0 feszültség mérhető buszvezetéken), illetve logikai 0-t a tranzisztor bekapcsolásával (0V a buszvezetéken). Ezt az elrendezést azért nevezük huzalozott-ÉS konfigurációknak, mert ahhoz, hogy logikai 1-es jelenjen meg a buszon, minden egyes csomópontnak logikai 1-est kell továbbítania. Más megfogalmazásban: ha akár csak egyetlen csomópont logikai 0-t tesz a buszra, akkor a busz logikai 0 állapotba kerül. Ez az oka, hogy a CAN rendszerben a 0-s bitet nevezük dominánsnak (dominant), és az 1-est recesszívnek (recessive).

3.2.2. CAN csomópont felépítése

3.2.2.1. A CAN protokollvezérlők csoportosítása

3.2.2.2. CAN csatlakozó

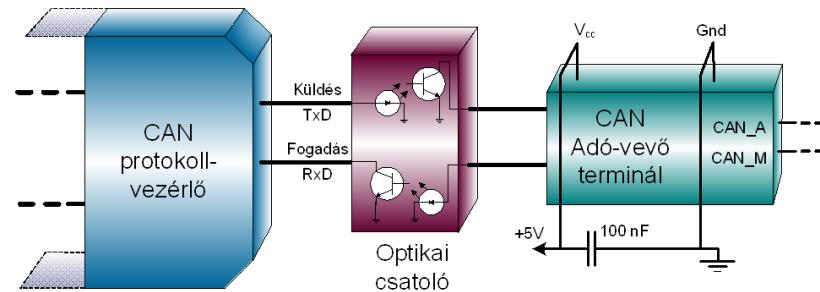
Egy általános CAN csomópont architektúrát szemléltet az [3.6. ábra], ahol a CAN adó-vevő terminál (Transceiver) teremti meg a kapcsolatot a CAN busz és a CAN protokollvezérlő (CAN protocol controller) között. A CAN vezérlő lehet a mikrokontrollerbe ágyazott, illetve attól különálló.



3.6. ábra - CAN csomópont architektúrák

Az RxD (Received Data) és TxD (Transmitted Data) jelek sorosan továbbítódnak, a CAN protokoll vezérlő ezeket használva továbbítja az információit. A CAN adó-vevő terminál a TxD jeleket alakítja át a busz differenciális jeleivé, illetve a busz-jeleket „fordítja le” a CAN protokoll vezérlő számára értelmezhető soros jelfolyammá (RxD). Bizonyos vezérlőkben ezeket a jeleket nem a földpotenciálhoz, hanem egy adott referencia-feszültséghez hasonlítják. Ez esetben 4 vonalra van szükség, Tx0, Rx0 (az adó-vevő illetve a kontroller oldali referencia-feszültségre kötve), valamint Tx1 és Rx1 (jelvezetékek).

Az előző esetben megismert közvetlen elektromos csatolás helyett lehetőség van optikai csatolás használatára is ([3.7. ábra]), így a CAN protokoll vezérlő elektromosan elszigetelhető a kommunikációs hálózattól, ezáltal megóvható a buszon esetlegesen keletkező túlfeszültségektől és kialakuló potenciálkülönbségektől.



3.7. ábra - Optikai csatolóval megvalósított összeköttetés

3.2.2.1. A CAN protokollvezérlők csoportosítása

3.2.2.1.1. Üzenetszűrés szerint

3.2.2.1.2. Azonosító mező hossza alapján

3.2.2.1.3. Az integráltság foka alapján

Ahány féle gyártó, annyi féle CAN protokoll implementáció található manapság az autóiparban. A különböző protokollvezérlő megvalósításokat a következő szempontok szerint lehet osztályozni:

- Az implementáció során alkalmazott puffer stratégia/üzenetszűrés szerint lehet BasicCAN vezérlőről (BasicCAN controller), valamint FullCAN vezérlőről (FullCAN controller) beszélni.
- Az üzenetek Azonosító mezejének hossza alapján lehetséges Standard CAN illetve Extended CAN megvalósítás.
- A CAN integráltsági foka alapján léteznek különálló CAN vezérlők (Stand-alone CAN controller) és beágyazott CAN vezérlők (Embedded CAN controller).

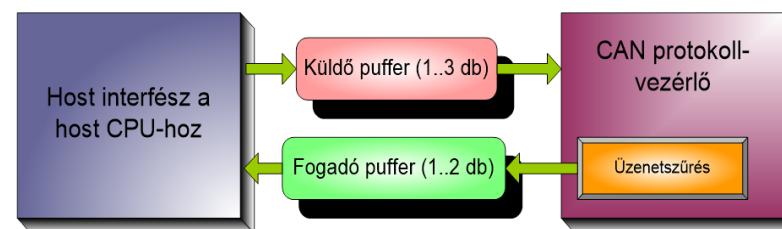
3.2.2.1.1. Üzenetszűrés szerint

A BasicCAN és a FullCAN vezérlők között a különbséget a pufferbe írás előtti üzenetszűrés (message filtering) jelenti, mely kizárolag az üzenet Azonosító mezeje alapján történik. A puffer interfésként szolgál a fogadott üzenetet elérni kívánó folyamat számára.

Ezen megfontolások alapján a mai implementációk az üzenetpufferek számában különböznek, habár az újabb FullCAN megvalósítások egyben képesek BasicCAN módban is működni.

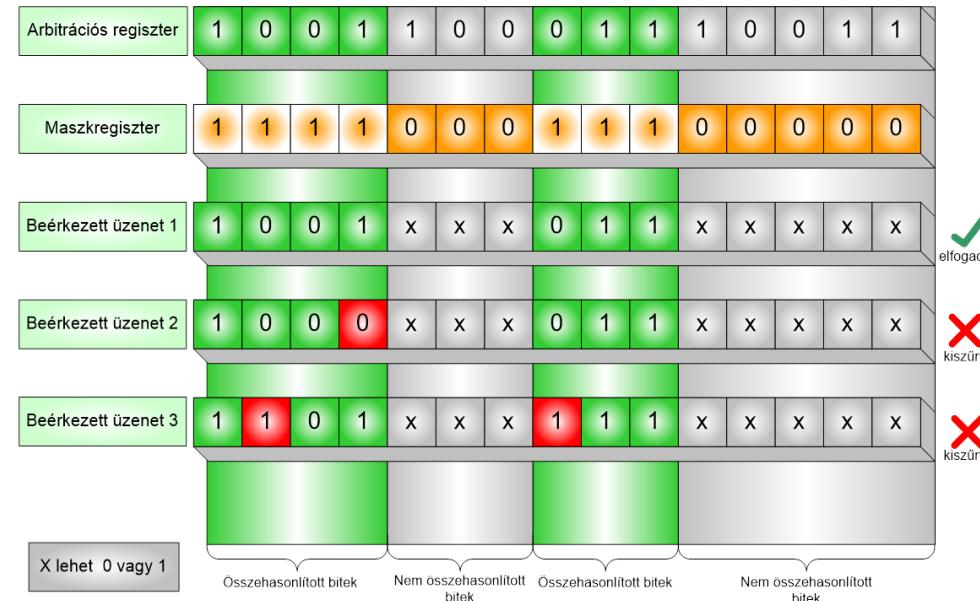
BasicCAN vezérlő:

A BasicCAN vezérlőknek 1-3 Küldő pufferük lehet a kimenő üzenetek küldésére, és 1 vagy 2 Fogadó pufferük van a bejövő üzenetek fogadására.



3.8. ábra - BasicCAN vezérlő

A BasicCAN felépítésű interfész egyfajta asszociatív memória-szűrként működik. Egy Elfogadási mintából (Arbitrációs regiszter) és egy Elfogadási maszkból (Maszkregiszter) áll ([3.8. ábra]), melyeket a felhasználó állít be a BasicCAN interfész konfigurálása során. minden üzenet Azonosító mezejét összehasonlítja a CAN vezérlő az Arbitrációs regiszterrel, és figyelmen kívül hagyja a maszk által kiszűrt biteket ([3.9. ábra]). Ha például a Maszkregiszter minden bitje 1-esre van állítva, akkor csak egyfélé Azonosító mezővel rendelkező üzenet lesz elfogadva. Ha a Maszkregiszter n-edik bitje 1, akkor összehasonlítja a beérkezett üzenet Azonosító mezejének n-edik bitjét az Arbitrációs regiszter n-edik bitjével. Ha azonban a Maszkregiszter n-edik bitje 0, akkor nem törödi vele („don't care”). Ha az üzenet Azonosító mezeje és az Arbitrációs regiszter bitenként megegyezik – amely biteknél a Maszkregiszterben összehasonlítás van – akkor a csomópont fogadja és eltárolja az üzenetet egy pufferbe, vagy kiszűri, azaz nem fogadja (beállításfüggő).



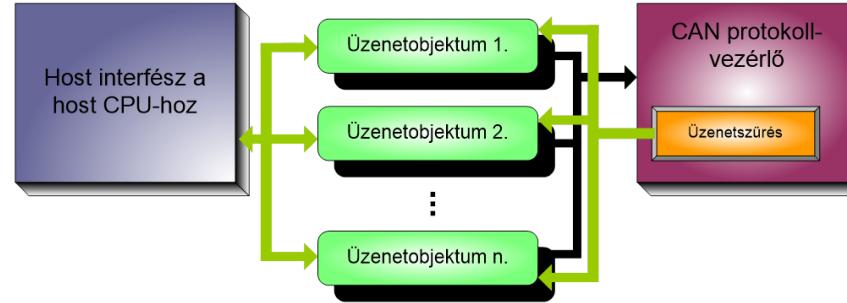
3.9. ábra - Az üzenetek szűrése

A beállításoknak megfelelő eltárolt üzenetek tehát a fogadó puffer ún. fogadási azonosító/adat regiszterében vannak, ahol a mikrokontroller azonnal elérheti őket. A BasicCAN leggyakoribb implementációja a FIFO (First-In-First-Out)^[15] szervezés, mivel több üzenet is megfelelhet a maszknak, és várhat köztes tárolásra a feldolgozás előtt.

Minden üzenet elfogadásakor a puffer-vezérlő egy megszakítással figyelmezteti a mikrokontrollert az új információ feldolgozására. Ha a mikrokontroller túl lassú, túlcordulás következhet be a fogadó pufferben, ez figyelmeztető jelzést generál. A BasicCAN sajátossága, hogy a fogadó puffer tartalmán a kiolvasás után a mikrokontroller még utólagos szűrést végez, hiszen a maszkból és mintából álló páros nem határozza meg egyértelműen az elfogadandó üzeneteket. Ezt megteheti, mert a pufferben az adatok az azonosítójukkal együtt tárolódnak. Így egy hardverszűrővel és az utólagos szűréssel tetszőleges számú virtuális fogadó puffer valósítható meg. Ez minimalizálja a hardver komplexitását. Küldéskor csak egy puffert használ a BasicCAN, ebbe írja bele az elküldendő adatokat, mielőtt ténylegesen elindítaná a küldési folyamatot.

Az Adatkérő üzenetre történő Adathordozó üzenet elküldését is a központi egységnek kell kezelnie, így magas bitsebesség esetén nagyon leterhelt, ezért a BasicCAN vezérlő használata csak korlátozott számú üzenettípus kezelése esetén, alacsony bitsebesség mellett ajánlott.

FullCAN vezérlő:

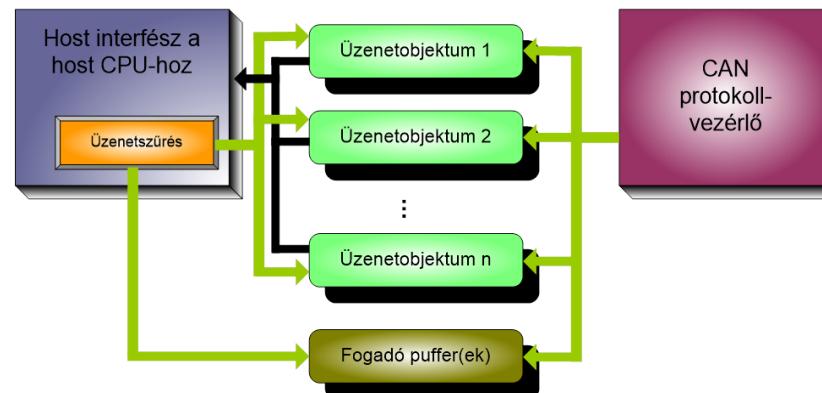


3.10. ábra - FullCAN vezérlő

Elképzelve egy olyan BasicCAN megvalósítást, ahol a fogadott üzenetazonosító maszk teljesen áttetsző, úgy belátható, hogy ebben az esetben nincs szükség a maszkot megvalósító regiszterre. Tehát minden egyes elfogadási mintának csak egy üzenet felel meg, így nincs szükség utólagos szűrésre. Azonban több puffer regisztert kell implementálni a hardverben, ezeket üzenetobjektumoknak (message object) nevezzük. A FullCAN vezérlők meghatározott számú (általában 15 vagy 16) üzenetobjektumot tartalmaznak, melyek mindegyike egy meghatározott azonosítójú üzenetet tárol. Az üzenetobjektumok felprogramozhatók fogadónak, vagy küldőnek is, mely a rendszer flexibilitását biztosítja.

Ha a CAN protokoll vezérlő kap egy üzenetet, és az Üzenetazonosítója egyezik valamelyik üzenetobjektum azonosítójával, akkor eltárolja ebben az üzenetobjektumban és ezt egy megszakítással (interrupt) jelzi a központi egység felé. A másik előnye az BasicCAN vezérlővel szemben, hogy az Adatkérő üzeneteket a FullCAN vezérlő automatikusan lekezeli. Ezáltal nem okoz problémát magas bitsebesség a nagyszámú, különböző típusú üzenetek hatékony kezelése.

Sok FullCAN vezérlő rendelkezik egy olyan üzenetobjektummal, amely úgy viselkedik, mint a BasicCAN vezérlő egy fogadó pufferje. Ezen üzenetobjektumra a BasicCAN bekezdésében leírtak szerint üzenetszűrés alkalmazható. Ez a tulajdonság különösen hasznos akkor, ha az üzenetobjektumok száma kevésnek bizonyul. ([3.11. ábra])



3.11. ábra - FullCAN vezérlő fogadó pufferrel kiegészítve

3.2.2.1.2. Azonosító mező hossza alapján

A CAN hálózaton az üzenet formátumát az Azonosító mező hossza dönti el. Ha az üzenet Azonosító mezeje 11 bites akkor standard formátumú-, ha viszont 29 bites, akkor kiterjesztett formátumú üzenetről van szó.

A CAN protokollvezérlők a protokoll verziók szerint 3 csoportba sorolhatók:

- CAN V2.0A eszközök/modulok: Ezek a CAN 2.0A fejezetében leírt Standard formátumú üzeneteket tudnak küldeni és fogadni, viszont CAN 2.0B fejezetben leírt Kiterjesztett formátumú üzenetek esetén Hibaüzenetet generálnak.
- CAN V2.0B Passzív eszközök/modulok: Ezek a CAN 2.0A fejezetében leírt Standard formátumú üzeneteket tudnak küldeni és fogadni, és CAN 2.0B fejezetben leírt Kiterjesztett formátumú üzenetek esetén nem generálnak Hibaüzenetet.
- CAN V2.0B Aktív eszközök/modulok: Ezek a CAN 2.0A fejezetében leírt Standard formátumú üzeneteket és a CAN 2.0B fejezetben leírt Kiterjesztett formátumú üzeneteket is tudnak küldeni és fogadni.

3.2.2.1.3. Az integráltság foka alapján

A CAN protokollvezérlők kezdetben ún. különálló CAN vezérlőegységek voltak. Mind a busztól, mind a központi egységtől jól el voltak különítve, így a felhasználó szabadon kombinálhatta a neki tetsző mikrokontrollert és CAN vezérlőt.

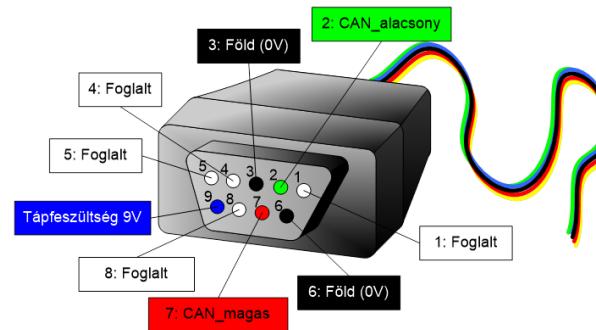
Néhány évvel később megjelentek a beágyazott CAN vezérlők, melyeknél a vezérlő már integrálva volt a mikrokontrollerbe. Előnyei között érdemes megemlíteni:

- a központi vezérlő könnyebben elérheti a puffereket
- a kisebb szilícium méretet
- a nagyobb megbízhatóságot
- és az alacsonyabb költséget.

Ma azonban már a beágyazott CAN vezérlőknek is olyan széles választéka áll rendelkezésre, hogy minden felhasználó megtalálhatja az alkalmazásnak megfelelő kombinációt.

3.2.2.2. CAN csatlakozó

A szabvány nem tesz megkötést a használandó csatlakozó típusára nézve, de a CiA DS102 definíciója szerint ajánlott a DIN41652 szabványnak megfelelő 9-tús D-Sub csatlakozó alkalmazása. A DS102 tartalmazza a csatlakozó tükiosztását ([[3.12. ábra](#)]) is.



3.12. ábra - DS102 szerinti csatlakozótű-hozzárendelés

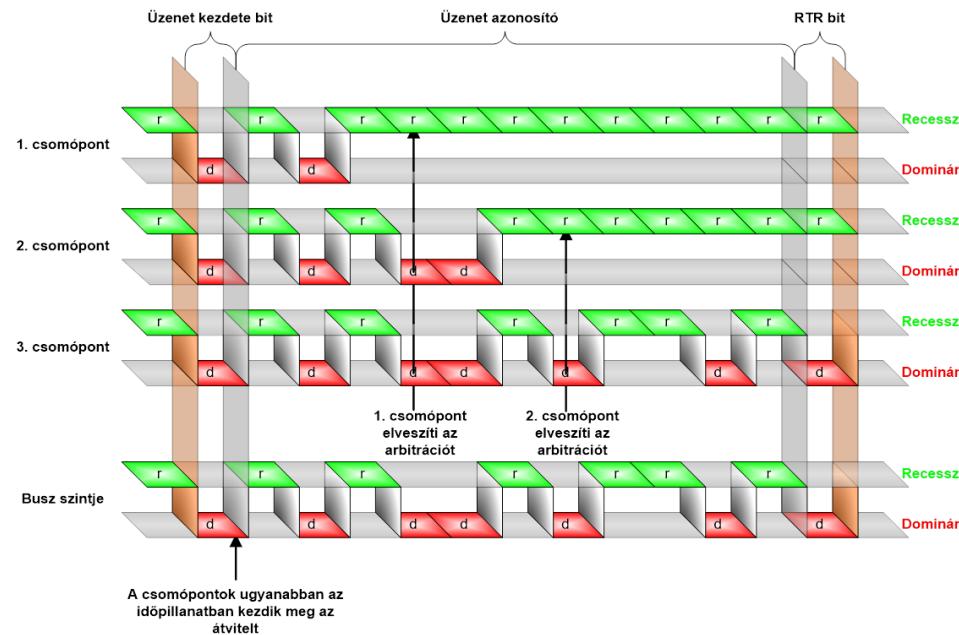
A 2-es és 7-es számú csatlakozótűk a csavart érpár vezetékeinek felelnek meg; a 3-as és 6-os számú tűk (utóbbi opcionális) a földpotenciálok; 1,4,5,8-as számúak foglaltak; a 9-es pedig opcionálisan a rendszer áramellátását biztosítja.

3.2.3. Arbitráció

Az adatok valós idejű feldolgozásához elengedhetetlen az üzenetek igen gyors továbbítása. Ehhez nem csak egy nagysebességű fizikai adatút szükséges, hanem gyors buszallokálás is, főleg amikor több csomópont akarja egyszerre megszerezni a buszt.

Hogy éppen melyik CAN csomópont használhatja a buszt üzenetei elküldésére, azt az arbitráció folyamata hivatott eldönten. A CAN rendszerben a csomópontok elosztott, tartalom alapú arbitrációt használnak. Mivel a csomópontokban az üzenet küldését generáló események nem szinkronizáltan következnek be, így előfordulhat, hogy több csomópont próbál meg egyidejűleg küldeni. Azonban egy csomópont akkor kezdheti meg üzenetének átvitelét, ha a busz szabad. A busz akkor tekinthető szabadnak, ha az Üzenetek utáni szünetet ([**3.3.1.5. szakasz**]. fejezet) nem szakította meg domináns bit, vagyis egy keret végén 11 recesszív bit érkezett. A küldésre váró üzenetek továbbítása az Üzenetek utáni szünet mezőt követő biten kezdődnek meg.

Ha több csomópont kezdi meg egyszerre a küldést, az ütközés feloldása a CAN keret Arbitrációs mezeje ([**3.3.1.1.2. szakasz**] fejezet) alatt üzenetrombolás nélkül megy végbe. Ennek kulcsa az előző fejezetben ismertetett huzalozott-ÉS megvalósítás. A huzalozott-ÉS logika mechanizmusának megfelelően a domináns szint a logikai 0-nak, a recesszív szint a logikai 1-nek felel meg. A domináns bit felülírja a recesszív bitet, de ez fordítva nem teljesül.



3.13. ábra - Az arbitráció folyamata

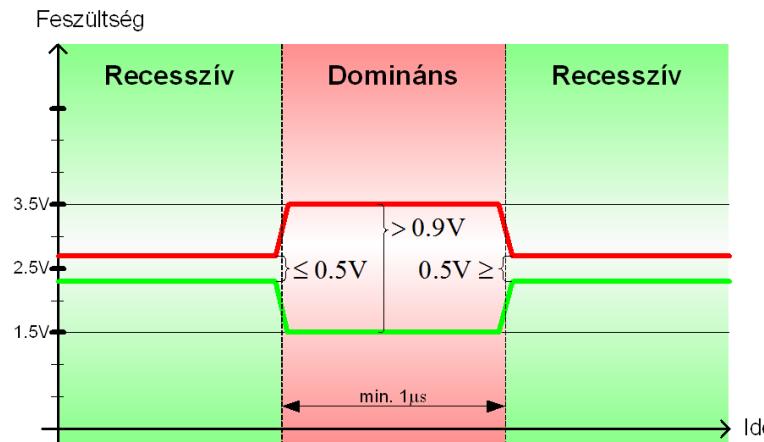
Az Arbitráció folyamata^[16] ([3.13. ábra]) akkor indul el, ha a busz szabad lesz (Szabad busz mező: [3.3.1.5. szakasz]. fejezet). minden olyan csomópont, amely recesszív bitet küld és domináns bitet vesz a buszról, elveszti az arbitrációt. Azok a csomópontok, amelyek elvesztik az arbitrációt, megszakítják a saját üzenetük küldését és automatikusan fogadóivá válnak annak az üzenetnek, amelynek a legnagyobb a prioritása a buszért való versenyben. A megszakított üzenetek újraküldését addig nem kezdhetik meg a csomópontok, amíg a busz újra szabaddá nem válik. A prioritásokat már a rendszer tervezéskor meg kell határozni, mivel ezután már nem lehet dinamikusan változtatni. A prioritást az üzenetazonosító határozza meg, oly módon, hogy az minél kisebb bináris szám, annál nagyobb az üzenet prioritása.

3.2.4. Bitreprezentáció a CAN-en

- 3.2.4.1. Bitszint meghatározása
- 3.2.4.2. Bitkódolási technikák
- 3.2.4.3. Bitidőzítés
- 3.2.4.4. Bitsebesség és a buszhossz viszonya
- 3.2.4.5. Szinkronizálás
- 3.2.4.6. Oszcillátor
- 3.2.4.7. Bitidő paramétereinek kiszámítása egy példán keresztül

3.2.4.1. Bitszint meghatározása

A bitszint meghatározása a CAN_magas és a CAN_alacsony vezetékek feszültségszint-különbségei alapján történik ([3.14. ábra]).

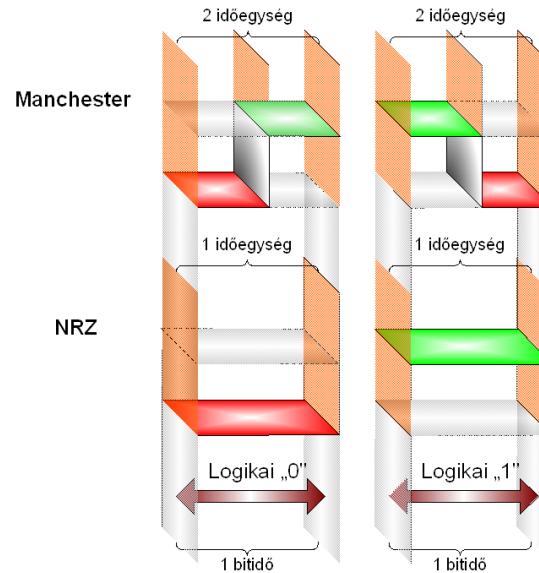


3.14. ábra - Bitszint meghatározása

Ha a CAN_magas és a CAN_alacsony vezeték feszültségszintjei között a különbség nagyobb, mint 0.9 V, akkor a bitszint domináns, ha kisebb, mint 0.5V akkor a bitszint recesszív lesz. Mivel a bitszintet a rendszer a feszültséggükönbségből határozza meg, ezért elektromágneses interferencia ellen – amely minden a két feszültségszintre egyformán hat – védve van a hálózat. A vezeték árnyékolásával a külső zavarások hatása tovább csökkenthető.

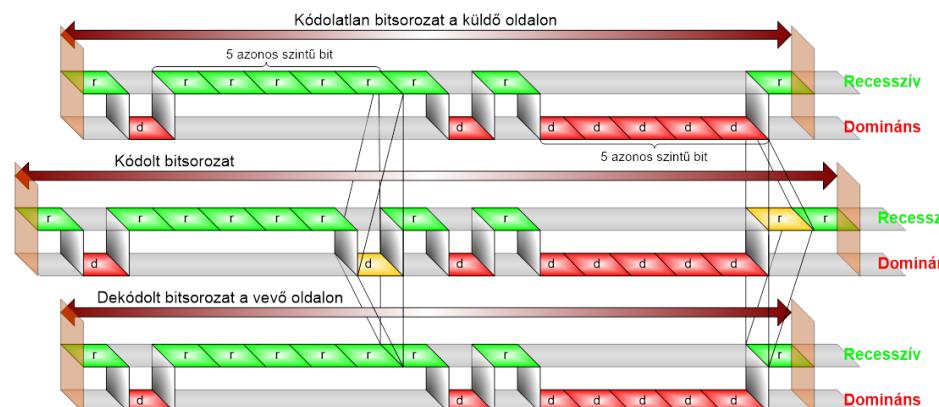
3.2.4.2. Bitkódolási technikák

A különböző bitkódolási, vagy más néven bitreprezentációs technikák (Manchester, NRZ, impulzushossz kódolás stb.) közötti fő eltérést az adja, hogy hány időszelet szükséges egy bit megjelenítéséhez. A CAN nullára vissza nem térő (NRZ = Non-Return-to-Zero) kódolást használ, mivel ez nyújtja a legnagyobb hatékonyságot. Ebben a megközelítésben a teljes bitidő alatt változatlan – vagy domináns, vagy recesszív – a jelszint. Ezzel szemben például a Manchester-kódolásnál az egy biten belüli jelszintváltás miatt két időegység szükséges egyetlen bit ábrázolásához ([3.15. ábra]).



3.15. ábra - Bitreprezentációs technikák

Látható, hogy az NRZ tömörebb adatátvitelt tesz lehetővé, azonban még a Manchester-kódolás esetén minden egyes bit átvitele a jelszintváltás miatt egyben szinkronizáció is, addig NRZ esetén a jelszint az átvitt információtól függően hosszabb időre változatlanul domináns illetve recesszív maradhat. Ebben az esetben is szükség van a szinkronizáció fenntartására. Ezt a feladatot látja el a bitbeszúrás módszere ([3.16. ábra]). Ez annyit jelent, hogy a küldő csomópont öt egymást követő azonos értékű küldendő bit után automatikusan beszúr egy ellentétes értékű bitet, amelyet a fogadó csomópontok az üzenet feldolgozása előtt automatikusan kivesznek. Fontos megemlíteni, hogy a bitbeszúrás módszere bizonyos mezőkre érvényesek, azaz a kommunikáció során nem minden esetben jelent/jelez hibát, ha ötnél több azonos jelszint jelenik meg a CAN buszon.



3.16. ábra - A CAN bitbeszúrási módszere

A bitbeszúrás (bit stuffing) módszere érvényes az Üzenet kezdete bit, az Arbitrációs, a Vezérlő- és az Adatmezőkre, valamint a CRC mezőre. A fennmaradó mezők (CRC-határoló, Nyugtázó, Üzenet vége bit), valamint a Hiba- és Túlcordulás üzenetek továbbítása bitbeszúrás nélkül történik. ([3.3.1. szakasz]. fejezet)

3.2.4.3. Bitidőzítés

3.2.4.3.1. Mintavételezési pont

3.2.4.3.2. Szinkronizáló szegmens

3.2.4.3.3. Terjedési időszegmens

3.2.4.3.4. Az 1. és 2. szinkron puffer szegmens

Egy CAN csomópont megfelelő bitrátán való kommunikációjának beállításához fontos ismerni a CAN specifikációban definiált alábbi három paraméter jelentését:

- Névleges bitráta (NBR = Nominal Bit Rate): az egy másodperc alatt átvitt bitek száma, amely megfelel a kívánt átviteli bitrátának.
- Névleges bitidő (NBI = Nominal Bit Time):

(16)

A CAN implementációk bitidőzítése a névleges bitidő paraméter értelmében adandó meg.

- Időkvantum: rögzített időegység, amely az oszcillátor-periódusból származik. Az implementációkban előre beállítható egy érték 1 és 32 között, amely azt adja meg, hogy az időkvantum hányszorosa a minimális időkvantumnak, amely megegyezik a CAN rendszer órajelének periódusidejével.

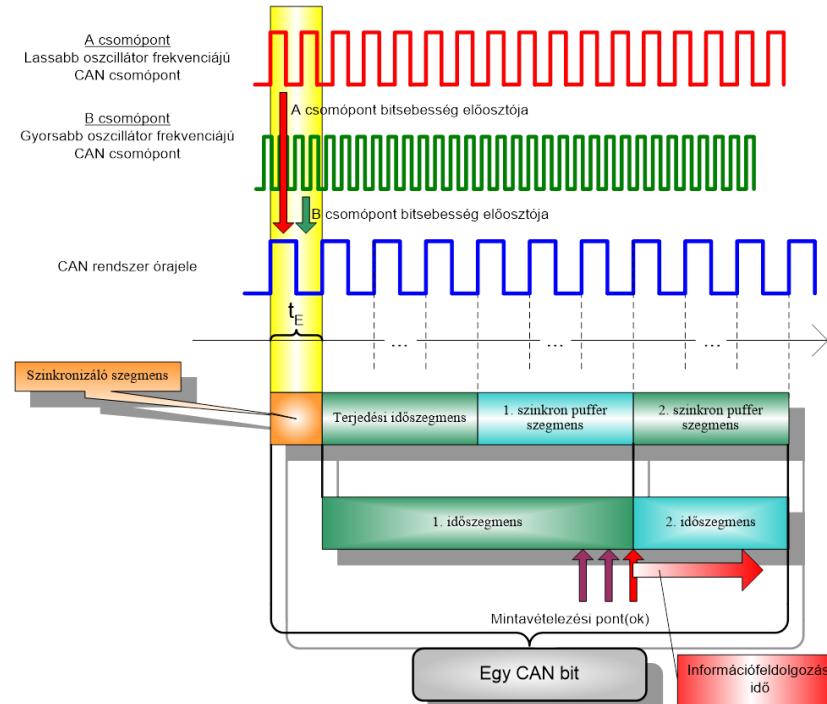
Egy adott bitráta beállításához szükség van az időkvantumnak, valamint ennek alapján a névleges bitidőnek a meghatározására. Az NBI négy darab nem átlapolódó időszegmenssel adható meg:

- Szinkronizáló szegmens (Synchronization segment) Szink_szeg
- Terjedési időszegmens (Propagation time segment) Terj_szeg
- 1. szinkron puffer szegmens (Phase buffer segment1) Szink_puff1
- 2. szinkron puffer szegmens (Phase buffer segment2) Szink_puff2

A hálózat névleges bitideje a fentiek szerint a következő:

(17)

Minden időszegmens felosztható időegységek (Time quantum) t_E egészszámú többszörösére. Az időszegmensek hosszának programozásával az NBI beállítható a kívánt értékre, azonban a Specifikáció tesz bizonyos megkötések a szegmensek hosszára.



3.17. ábra - CAN bit struktúrája

Az időegység időtartama megegyezik a CAN rendszer órajelének periódusával, amely a csomópont oszcillátor frekvenciából állítódik elő egy programozható bitsebesség előosztójával. minden bit minimum 8, maximum 25 időegységből áll. A bitidő és ebből következően a bitsebesség az időegység hosszának, és az időszegmensekben lévő időegységek számának programozásával állíthatók be a kívánt értékre. Egyes CAN modulok implementálásánál – a könnyebb programozhatóság érdekében – a Terjedési időszegmenst és a Szinkronizáló szegmenst egy időszegmensben egyesítik, amelyet 1. időszegmensnek neveznek. A 2. szinkron puffer szegmenst pedig a 2. időszegmensnek hívják. ([3.17. ábra])

3.2.4.3.1. Mintavételezési pont

A mintavételezési pont (sample point) a CAN buszon lévő bitnek az a pontja, ahol a buszsint leolvasása megtörténik, és amelyből a bit értéke fog generálódni. Egy biten belül lehet 1 vagy 3 mintavételezési pont. Ha 3 mintavételezési pontot van megadva, akkor a bit értéke a leggyakrabban mintavételezett érték lesz. Ez a pont a 1. szinkron puffer szegmens és a 2. szinkron puffer szegmens illetve az 1. és a 2. időszegmens határánál található. A mintavételezési ponttal kezdődik az információfeldolgozási idő (Information processing time), és addig tart, amíg az aktuális bitszint kiértékelése történik. ([3.17. ábra])

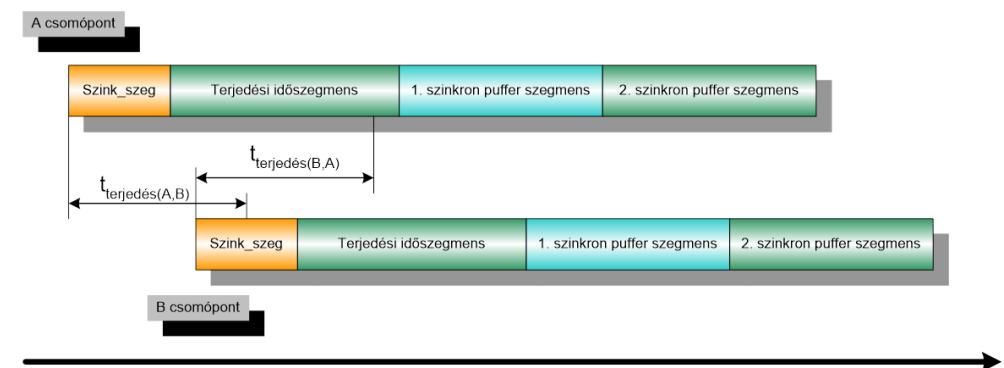
3.2.4.3.2. Szinkronizáló szegmens

A Szinkronizáló szegmens hossza nem programozható, hanem minden fixen egy időegység, a CAN buszon lévő bit első szegmense, és a CAN buszon lévő csomópontok közötti szinkronizálásra szolgál. A küldő-csomópontnak a következő elküldendő bit értékének küldését a Szinkronizáló szegmensen belül kell megkezdenie, illetve ha az előző bit és az elküldendő bit között szintváltás van (recesszív szintből domináns szintbe, vagy fordítva), akkor a szintváltás élének a Szinkronizáló szegmensbe kell esnie. Az elküldött bit fogadását a fogadó csomópontok a Szinkronizáló szegmens alatt kezdi meg. A küldési-időkésés következtében a fogadó csomópontok Szinkronizáló szegmense a küldő csomópont Szinkronizáló szegmenséhez képest készik. ([[3.18. ábra](#)])

3.2.4.3.3. Terjedési időszegmens

A Terjedési szegmens hossza programozható, 1-8 db időegységből állhat, és arra szolgál, hogy kompenzája a rendszerből adódó fizikai késleltetést. Mivel a CAN protokoll üzenetrombolás nélküli arbitrációt használ, és a nyugtázás az üzeneten belül történik, ezért minden csomópont miután elküldte a soron következő bitet, monitorozza az elküldött bitek logikai szuperpozícióját. A Terjedési szegmens azt biztosítja, hogy a csomópontok legkorábbi lehetséges mintavételezési pontját addig készítse, hogy az összes elküldött bit, amelyet a küldő csomópont küldött, elérjen mindegyik csomóponthoz. Hogy ez megvalósulhasson, ennek a szegmensnek kétszer olyan hosszúnak kell lennie, mint a buszon lévő két legtávolabbi csomópont közötti jelterjedési idő valamint a küldő és fogadó csomópontok belső késleltetéseinek összege.

A [[3.18. ábra](#)] két csomópont között lévő terjedési-idő késleltetést mutatja. A csomópont által küldött bitértéket a B csomópont $t_{terjedés(A,B)}$ idő múlva kapja meg, a B csomópont által küldött bitértéket az A csomópont $t_{terjedés(B,A)}$ idő múlva kapja meg az A csomópont terjedési időszegmensének vége előtt. Így az A csomópont is helyesen mintavételezi a bitértékét. A B csomópont még akkor is helyesen fogja mintavételezni a bitértékét, ha a B csomópont mintavételezési pontja az A csomópont által küldött bit után van a két csomópont közötti terjedési-idő késleltetés miatt.



[3.18. ábra - Terjedési-idő késleltetés két csomópont között](#)

A $t_{terjedés(A,B)}$ és hasonlóképpen az $t_{terjedés(B,A)}$ három részből tevődik össze:

- Küldési idő késleltetés: $t_{K(A)}$
- Busz késleltetési idő a két csomópont között: $t_{Busz(A,B)}$

- Fogadási idő késleltetés: $t_{F(B)}$

(18)

Ahhoz hogy biztosítató legyen a helyes mintavételezés, a Terjedési időszegmens minimum értékét a következőképpen kell megválasztani:

(19)

ahol az A és a B csomópont a hálózat két egymástól legtávolabb lévő csomópontja azért, hogy a köztük lévő késleltetés maximális legyen. A [???] képletből adódik:

(20)

ahol t_{Busz} a legnagyobb buszkésleltetési idő két csomópont között, t_F a fogadó-csomópont késleltetése a fizikai interfész miatt, és t_K a küldő-csomópont késleltetése a fizikai interfész miatt. Ha a t_K és a t_F nem egységes, akkor a CAN rendszeren belüli legnagyobb értékkel kell számolni.

Tehát, hogy minimum hány darab időegységet kell hozzárendelni a Terjedési időszegmenshez, azt a következő képlettel lehet kiszámolni:

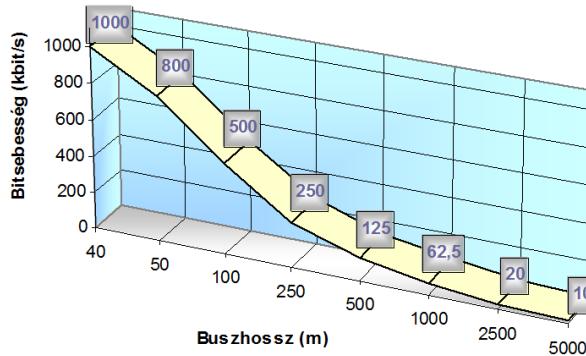
(21)

3.2.4.3.4. Az 1. és 2. szinkron puffer szegmens

Ez a két szegmens kompenzálja a szinkron hibákat. Az 1. szinkron puffer szegmens hossza programozható. 1-8 időegységből állhat, ha egy, 2-8 időegységből állhat, ha 3 mintavételezési pont van kijelölve bitenként. A 2. szinkron puffer szegmens hosszának meg kell egyeznie az 1. szinkron puffer hosszával, viszont ha az 1. szinkron szegmens hossza kisebb, mint az információfeldolgozási idő, akkor a 2. szinkron szegmens hosszának legalább egyenlőnek kell lennie az információfeldolgozási idővel. Ez azt is jelenti, hogy a két szegmens együttesen nem lehet hosszabb időtartamú, mint az információfeldolgozási idő kétszerese.

3.2.4.4. Bitsebesség és a buszhossz viszonya

A kommunikációs vonalakon terjedő jelek sebességére vonatkozó fizikai korlátok miatt (réz vezetékben az elektromos hullám terjedési ideje megközelítőleg 20cm/ns), valamint a CAN bitszintű arbitrációja miatt, ahogy a busz hossza nő, úgy csökken a megengedett maximális átviteli sebesség. E jelenséget szemlélteti a [[3.19. ábra](#)]. Az idő, amíg egy csomópont által elküldött bit eljut a legtávolabbi csomóponthoz, majd vissza, nem lehet hosszabb, mint a küldő csomópont bitidejének 2/3 része. A maradék 1/3-nyi bitidő elegendő arra, hogy minden csomópont eldöntse, hogy elveszítette-e a busz használatának jogát, vagy folytathatja-e a küldést.



3.19. ábra - Bitsebesség és a buszhossz viszonya

A maximum sebesség CAN rendszereken belül az 1Mbit/s, amely 40m hosszú busszal valósítható meg.

Ha hosszabb buszra lenne szükség, akkor számolni kell a bitsebesség csökkenésével, így 400 m-es buszhosszhoz már 100kbit/s sebesség párosul.

Ennél is hosszabb vezetékezés esetén akár az 1000m-es hossz is elérhető, ekkor azonban már csak 50kbit/s bitsebesség realizálható. Ilyen hosszú busz esetén már ajánlatos speciális meghajtókat és jelismétlőket alkalmazni.

3.2.4.5. Szinkronizálás

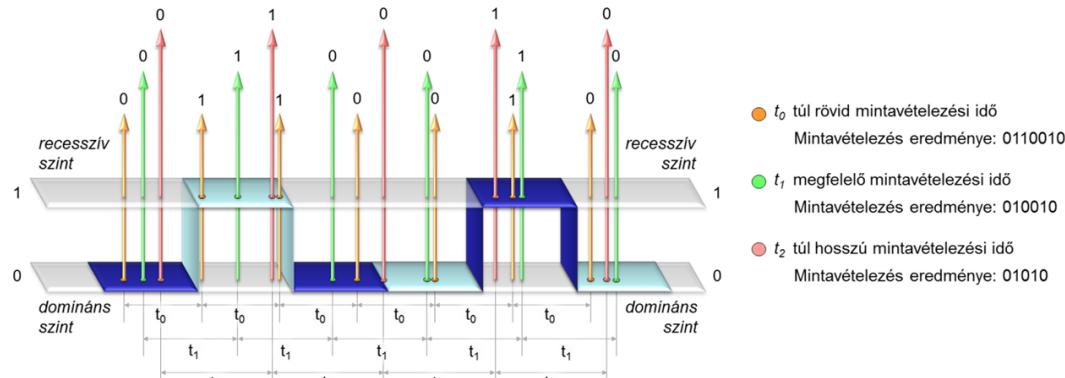
3.2.4.5.1. Az él szinkronhibája

3.2.4.5.2. Fixszinkronizálás

3.2.4.5.3. Újraszinkronizálás

3.2.4.5.4. Szinkronizálás szabályai

Soros buszrendszereken történő adatátvitel során a küldő oldalon az adatok párhuzamos-soros, míg vevő oldalon soros-párhuzamos átalakítása történik. A vevőnek megfelelő időpillanatokban kell mintát vennie a buszról ahhoz, hogy a helyes jelet alakítsa vissza párhuzamos formába. A helytelen mintavételezés következtében a vevő oldalon nem ugyanaz az üzenet áll elő, mint amit a küldő továbbított ([3.20. ábra]).



3.20. ábra - Mintavételezési idő helyes megválasztásának fontossága

Az ilyen, ún. szinkronhibák oka lehet az, hogy az egyes csomópontok oszcillátor frekvenciája kissé eltér egymástól, vagy az, hogy a különböző csomópontok t_k küldési idő késleltetése eltérő, ami a [???] szerint a terjedési idő megváltozását okozza. A CAN aszinkron mintavételezetet használ, vagyis minden csomópontnak saját órajel generátora van (szemben a szinkron esettel, amikor egy közös órajel hatására történik a mintavételezés), a szinkronhibák elkerülése érdekében tehát különösen fontos, hogy a küldő és fogadó csomópontok órai valamilyen módon szinkronizálják legyenek. Ehhez az információt a jelszint váltások adják. A bitbeszúrás módszere ([3.16. ábra]) biztosítja, hogy megfelelő időközönként mindenkorábban történjen bitszint változás.

3.2.4.5.1. Az él szinkronhibája

Élek detektálása úgy történik, hogy a csomópont folyamatosan, minden időkvantumban mintavételezi a buszt, és az aktuális jelszintet összehasonlíta az előző időkvantumban mért értékkal. Szinkronizáció csak recesszívból dominánsba való jelszint változáskor történik. Az él szinkronhibája (Phase Error of an edge) azt a pozíciót adja meg a Szinkronizáló szegmenshez képest, hogy az él, melyik időegységbe esik. A szinkronhiba értéke mindenkorábban van kifejezve, és az alábbi három eset különíthető el:

- szinkronhiba = 0, ha az él a Szinkronizáló szegmensbe esik
- szinkronhiba > 0, ha az él a Szinkronizáló szegmens előtt helyezkedik el
- szinkronhiba < 0, ha az él a Szinkronizáló szegmens után helyezkedik el

3.2.4.5.2. Fixszinkronizálás

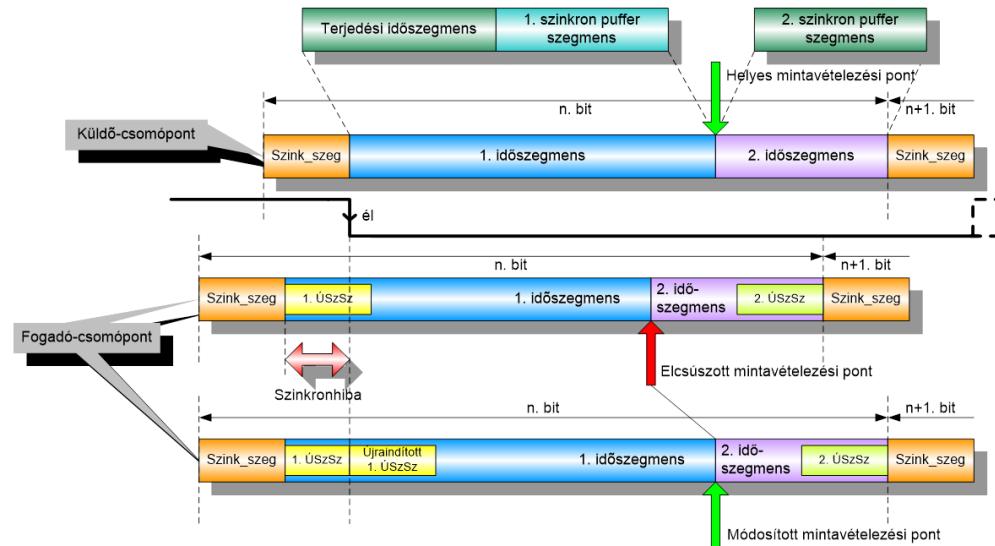
A fixszinkronizálás (Hard synchronization) csak az üzenetek elején történik, amikor minden csomópont az aktuális bitidőt újra indítja a Szinkronizáló szegmenssel úgy, hogy az elküldött üzenet kezdete bit recesszívból dominánsba ugró éle ebbe a szegmensbe essen.

3.2.4.5.3. Újraszinkronizálás

Az újraszinkronizálás (Resynchronization) az üzenet további részében történik, ha a bitérték recesszívről dominánsra változik. Az újraszinkronizálás a szinkronhiba értéke alapján, a Szinkron puffer szegmensek hosszának változtatásával történik. A Szinkron puffer

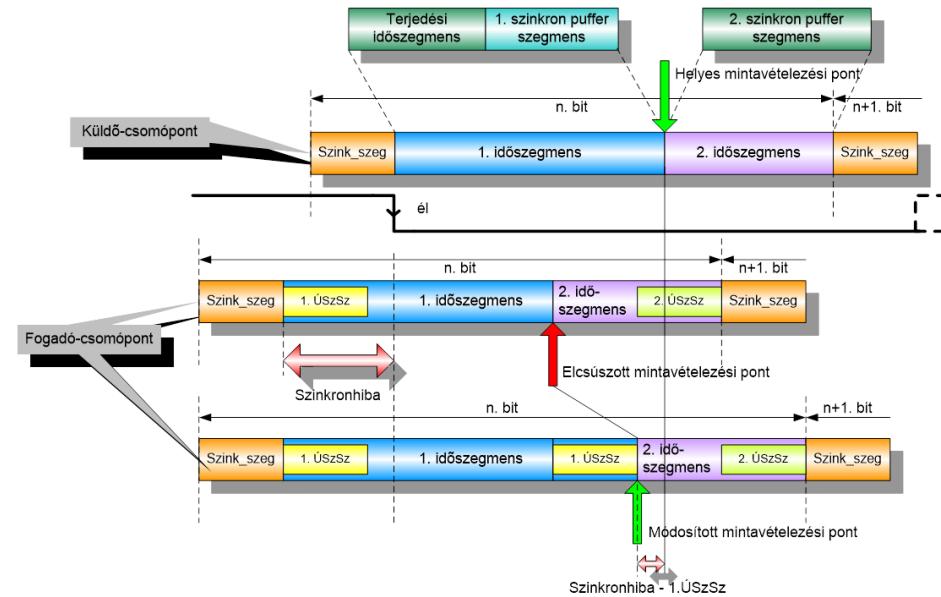
szegmensek növelésének és csökkentésének mértéke az újraszinkronizálási szélesség (Resynchronization jump width), amelynek az értéke legalább 1, és legfeljebb az 1. Szinkron puffer szegmens hossza, de nem lehet nagyobb 4-nél (azaz 1 és $\min(4, \text{szink_puffl})$ között programozható). A következő esetek fordulhatnak elő:

- Ha a szinkronhiba = 0, akkor a bit szinkronban van. Ez az optimális eset, ekkor az él hatására a vevőben elkezdődik az 1. Szinkron puffer szegmens, majd ennek a szegmensnek a végén mintavételezi a bitet. Ezután kezdődik a 2. szinkron puffer szegmens, amelynek lefutása után várható a következő bit megjelenése.
- Ha a szinkronhiba < 0 (fogadó csomópont órája gyorsabb, mint a küldőé), akkor az aktuális bithez tartozó 1. szinkron puffer szegmens hosszát növeli a szinkronhiba értékének megfelelően, és így később vesz mintát a bitból.
 - Ha a szinkronhiba az 1. újraszinkronizálási szélességnél kisebb vagy egyenlő, akkor az él hatására újraindul az 1. időszegmens ([[3.21. ábra](#)]).



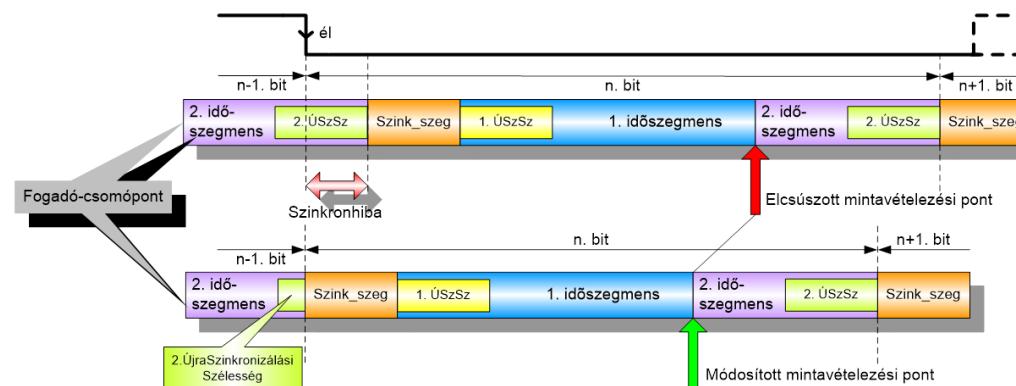
[3.21. ábra - Újraszinkronizálás, ha szinkronhiba < 0, és | szinkronhiba | < 1. újraszinkronizálási szélesség](#)

- - Ha a szinkronhiba az 1. újraszinkronizálási szélességnél nagyobb, akkor az 1. időszegmens (Terjedési idő szegmens + 1. szinkron puffer szegmens) végét hosszabbítja meg egy 1. újraszinkronizálási szélességnyi idővel. ([[3.22. ábra](#)])



3.22. ábra - Újraszinkronizálás, ha szinkronhiba < 0, és $| \text{szinkronhiba} | > 1$. újraszinkronizálási szélesség

- Ha a szinkronhiba > 0 (fogadó csomópont órája lassabb, mint a küldőé), akkor az aktuális bithez tartozó 2. szinkron puffer szegmens hosszát csökkenti a szinkronhiba értékének megfelelően, és így előbb vesz mintát a bitból. Az él hatására rögtön elkezdődik a következő bit, mégpedig a Szinkronizációs szegmensek elhagyva rögtön az 1. időszegmenssel. Ha a szinkronhiba nagyobb, mint a 2. újraszinkronizálási szélesség hossza, akkor is maximum egy 2. újraszinkronizálási szélességnyi idővel csökken a 2. szinkron puffer szegmens hossza. ([3.23. ábra])



3.23. ábra - Újraszinkronizálás, ha a szinkronhiba > 0

3.2.4.5.4. Szinkronizálás szabályai

A fixszinkronizálás és az újraszinkronizálás szabályai:

- Csak egy szinkronizálás megengedett egy bitidőn belül
- Az él akkor használható újraszinkronizáláshoz, ha az előző mintavételezési pontban vett értéke eltér a közvetlenül az él után következő buszszint értékétől
- Fixszinkronizálás akkor történhet, ha a Szabad busz mező alatt a bitérték recesszívből dominánsba vált.

3.2.4.6. Oszcillátor

3.2.4.6.1. Oszcillátor tolerancia követelmény

A CAN rendszer órajelét minden csomópont a saját oszcillátor frekvenciából származtatja ([3.17. ábra]), amely csomóponthonként kisebb, vagy nagyobb mértékben eltérhetnek egymástól. Az aktuális CAN rendszer órajele, és ebből következően a bitidő is minden csomópontnál alá van rendelve az oszcillátor toleranciának. A rendszer kora és a külső hőmérsékletváltozás is hatással van az kezdeti oszcillátor toleranciára. A CAN rendszer órajelének tolerancia definíciója:

(22)

ahol f az aktuális frekvencia és f_N a névleges frekvencia. Ahhoz hogy biztosítható legyen a hatékony kommunikáció, azt a minimum elvárást kell teljesítenie a CAN rendszernek, hogy annak a két csomópontnak, amelyek között a legnagyobb a késleltetés, és a CAN rendszer órajelének frekvenciájától való eltérésük a megadott oszcillátor tolerancia két határára esik, helyesen kell fogadnia és dekódolnia minden elküldött üzenetet.

3.2.4.6.1. Oszcillátor tolerancia követelmény

Mivel újraszinkronizálás csak recesszívből dominánsba futó élnél történik, az addig felhalmozódó szinkronhibának kisebbnek kell lennie az újraszinkronizálási szélesség értékénél. A felhalmozódó szinkronhiba a CAN rendszerben lévő oszcillátor toleranciának köszönhető. Ez a következő képlettel fejezhető ki:

(23)

6 egymást követő domináns bit kitöltési hibát ([3.4.2.2. szakasz] fejezet) eredményez, amelyet követően egy Aktív hibaüzenet ([3.3.1.3.1. szakasz]fejezet) generálódik, amely egy aktív 6 domináns bitból álló Hibajelző mezővel kezdődik. Ezért az utolsó újraszinkronizálás után a csomópontnak helyesen kell a 13. bit értékét kiolvasnia. Ez a következő módon fejezhető ki:

(24)

A fentiek szerint tehát két oszcillátor tolerancia követelmény van, amit be kell tartani.

3.2.4.7. Bitidő paramétereinek kiszámítása egy példán keresztül

Bitsebesség = 125kbit/s (\Rightarrow bitidő = 8000ns/bit)

Buszhossz = 50m

Buszkésleltetés = $5 \cdot 10^{-9}$ s/m

A fizikai interfész küldési és fogadási késleltetése összesen = 150ns 85°C-on

Oszcillátor frekvencia = 8MHz

- lépés: A Terjedési időszegmens minimum értékének kiszámítása a késleltetési idő maximumának kiszámításával.

Busz fizikai késleltetése: $t_{Busz} = 50m \cdot (5 \cdot 10^{-9} \text{ s/m}) = 250\text{ns}$

$t_{Terj_szeg} = 2 \cdot (250\text{ns} + 150\text{ns}) = 800\text{ns}$

- lépés: A CAN rendszer órajel frekvenciájának az előosztóval történő megválasztása úgy, hogy a bitek időegységeinek a száma 8 és 25 közé essen. Legyen 4 az előosztó. Ekkor a CAN rendszer órajel frekvenciája $8/4 = 2\text{MHz}$ és egy időegység $t_E = 1/2\text{MHz} = 500\text{ns}$ ebből következően $8000/500 = 16$ időegység bitenként.
- lépés: A [???] alapján kiszámolható a Terjedési időszegmens. Ha az eredmény nagyobb, mint 8, akkor vissza kell lépni a 2. lépéshoz és alacsonyabb CAN rendszer órajel frekvenciát kell választani.

$Terj_szeg = \text{Felkerekítés}(800\text{ns}/500\text{ns}) = \text{Felkerekítés}(1,6) = 2$

- lépés: Az 1. és 2. szinkron puffer szegmens meghatározása. A 2. lépésben kiszámolt bitenkénti időegységből levonunk 1 időegységet (Szink_szeg). Ha a fennmaradó érték kisebb, mint 3, akkor vissza kell lépni a 2. lépéshoz és nagyobb CAN rendszer órajel frekvenciát kell választani. Ha a fennmaradó érték páratlan és nagyobb, mint 3, akkor le kell vonni belőle 1-et. Ha a megmaradt összeg egyenlő 3-mal, akkor a Szink_puff1 = 1 és Szink_puff2 = 2 és csak egy mintavételezési pont választása engedélyezett. Egyébként a fennmaradó összeget meg kell felezni, és ezzel az értékkal lesz egyenlő az 1. és 2. szinkron puffer.

$16 - 1 - 3 = 13$ így Szink_puff1 = 6 és Szink_puff2 = 6 és a fennmaradó 1 időegységet hozzáadjuk a Terj_szeg-hez. Mivel Szink_puff1 > 4, ezért vissza kell lépni a 2. lépéshoz, és nagyobb előosztóval újraszámolni.

- lépés (újraszámolás): Legyen az előosztó 8. A CAN rendszer órajel frekvenciája $8/8 = 1\text{MHz}$ és egy időegység $t_E = 1/1\text{MHz} = 1000\text{ns}$ ebből következően $8000/1000 = 8$ időegység bitenként.
- lépés: A [???] alapján:

$Terj_szeg = \text{Felkerekítés}(800\text{ns}/1000\text{ns}) = \text{Felkerekítés}(0,8) = 1$

- lépés: $8 - 1 - 1 = 6$ így Szink_puff1 = 3 és Szink_puff2 = 3. Ha a Szink_puff1 > 4 lenne, akkor ismét vissza kellene lépni a 2. lépéshoz, és kisebb előosztóval újraszámolni.
- lépés: $t_{\text{Újraszinkronizálás ugrás hossz}} = \min(4, \text{Szink_puff1}) = 3$
- lépés: A [???] alapján:

A [??] felhasználásával:

A kívánt oszcillátor tolerancia a kisebb érték, azaz 0,014851 (1,4851%)

Összegezve:

Előosztó = 8	Szink_puff1 = 3
Névleges bitidő = 8	Szink_puff2 = 3
Terj_szeg = 1	tÚjraszinkronizálás ugrás hossz = 3
Oszcillátor tolerancia = 1,4851%	

3.3. A CAN protokoll Adatkapcsolati rétege

3.3.1. CAN üzenetkeretek

- 3.3.1.1. Adathordozó üzenet
- 3.3.1.2. Adatkérő üzenet
- 3.3.1.3. Hibaüzenet
- 3.3.1.4. A túlcordulás üzenet
- 3.3.1.5. Üzenetek közötti mező

3.3.2. Üzenetek késleltetése

3.3.1. CAN üzenetkeretek

- 3.3.1.1. Adathordozó üzenet
- 3.3.1.2. Adatkérő üzenet
- 3.3.1.3. Hibaüzenet
- 3.3.1.4. A túlcordulás üzenet
- 3.3.1.5. Üzenetek közötti mező

A CAN hálózatokon az adatátvitel üzenetkeretek használatával történik. Kétféle formátumú üzenetkeret/üzenet létezik. Ha az üzenet Azonosítómezeje 11 bites, akkor standard formátumú üzenetről (a CAN Specifikáció 2.0 „A” részében definiált), ha viszont 29 bites, akkor kiterjesztett (extended) formátumú üzenetről (a CAN Specifikáció 2.0 „B” részében definiált) lehet beszélni. Az előbbiből $2^{11}=2048$ db, az utóbbiból $2^{29}=536870912$ db különböző azonosítóval rendelkező üzenet alkotható.

A CAN buszon a következő üzenettípusok különböztethetők meg:

- Adathordozó üzenet (Data frame) ([[3.3.1.1. szakasz](#)]. fejezet)
- Adatkérő üzenet (Remote frame) ([[3.3.1.2. szakasz](#)] fejezet)
- Hiba üzenet (Error frame) ([[3.3.1.3. szakasz](#)] fejezet)
- Túlcordulás üzenet (Overload frame) ([[3.3.1.4. szakasz](#)] fejezet)

Csak az első két fajtájú, az Adathordozó és az Adatkérő üzenetekből van standard és kiterjesztett formátumú is, míg a Hiba és Túlcordulás üzenetekből csupán standard formátumú létezik!

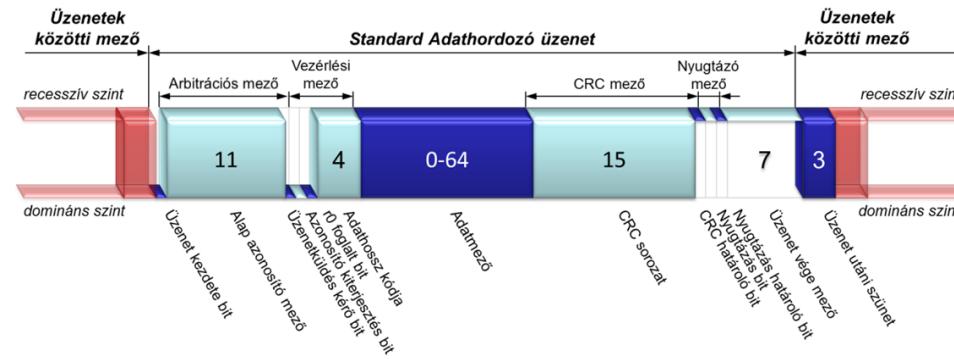
Ezek mellett fontos megemlíteni, hogy a CAN buszon még az üzenetek közötti szünetekben is megjelennek bitek ([[3.3.1.5. szakasz](#)]. fejezet)

3.3.1.1. Adathordozó üzenet

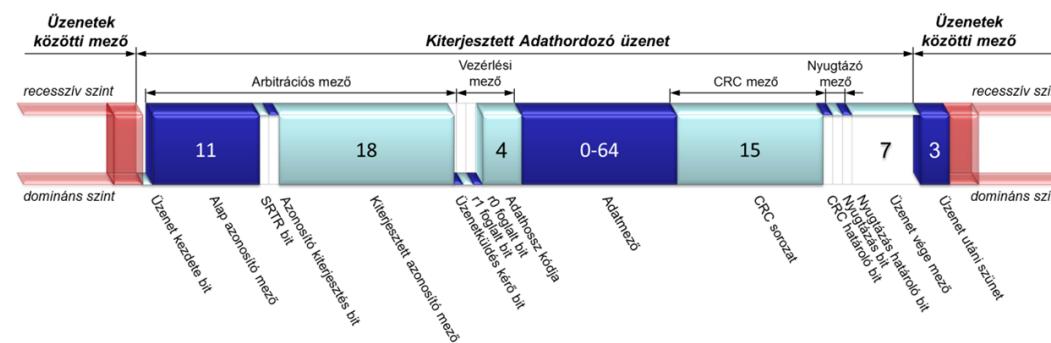
- 3.3.1.1.1. Üzenet kezdete bit
- 3.3.1.1.2. Az arbitrációs mező
- 3.3.1.1.3. Vezérlési mező
- 3.3.1.1.4. Adatmező
- 3.3.1.1.5. CRC mező
- 3.3.1.1.6. Nyugtázás mező
- 3.3.1.1.7. Üzenet vége mező

A CAN hálózaton az adatátvitelt ilyen Adathordozó üzenetek segítségével valósítják meg a csomópontok. Az adathordozó üzenet a következő mezőkből áll:

- Üzenet kezdete bit = SOF (Start of Frame)
- Arbitrációs mező (Arbitration field)
- Vezérlési mező (Control field)
- Adat mező (Data field)
- CRC mező (CRC field)
- Nyugtázás mező = ACK (Acknowledgement field)
- Üzenet vége mező = EOF (End of Frame)



3.24. ábra - Standard formátumú Adathordozó üzenet, ahol az Alapazonosító mező megegyezik az Azonosító mezővel



3.25. ábra - Kiterjesztett formátumú Adathordozó üzenet

3.3.1.1. Üzenet kezdete bit

Ez a rész jelzi az Adathordozó üzenet kezdetét egy darab domináns bittel. Az Üzenet kezdete bit segítségével minden csomópontnak szinkronizálnia kell magát az arbitrációt elsőként kezdő csomóponthoz. Egy csomópont csak akkor kezdheti az Arbitrációs szakaszt, ha a busz szabaddá válik, melyet általában 11 egymást követő recessív bit (Nyugtázás határoló bit + Üzenet vége mező + Üzenet utáni szünet) jelez.

3.3.1.2. Az arbitrációs mező

Az Arbitrációs mező standard formátumú Adathordozó üzenet esetén az Alap azonosító mezőből és az Üzenetküldés kérő bitból (RTR – Remote Transmission Request bit) áll. Kiterjesztett formátum esetén e két rész közé beékelődik az Üzenetküldés kérő bit helyettesítő (SRTR bit – Substitute RTR bit), az Azonosító kiterjesztés bit (Identifier Extension bit) és a Kiterjesztett azonosító mező. Az Alap azonosító mező standard formátum esetén maga az Azonosító mező (Identifier field), míg kiterjesztett formátum esetén az Alap, és a Kiterjesztett azonosító mezők együttesen alkotják az Azonosító mezőt (Identifier field).

Az Arbitrációs szakaszban dől el, hogy melyik csomópont nyeri el az adási jogot, azaz hogy melyik elküldésre váró üzenet fog a CAN buszon megjelenni, így az üzenetek prioritásának megfelelő megválasztása igen fontos.

- Alap-azonosító mező

Az Arbitrációs mező 1-11. bitjeit fogalja el az Alapazonosító mező. Standard formátumú üzenet esetén e rész lefedi a teljes 11 bites Azonosító mezőt, így standard üzeneteknél elégséges csupán az utóbbi megnevezést használni. Itt a bitek fordított sorrendben követik egymást a 10.-től a 0. bitig.

Ezzel szemben kiterjesztett formátum esetén a teljes Azonosító mezőnek csupán a felső 11 bitje szerepelhet az Alap azonosító mezőben, szintén fordított sorrendben a 28.-tól a 18. bitig. A fennmaradó 18 bit ($11+18 = 29$ bit) a Kiterjesztett azonosító mezőben kap majd helyet.

- Üzenetküldés kérés bit helyettesítő

Ez az SRTR bit az Arbitrációs mező 12. bitje, mely mindig recesszív, és csak kiterjesztett formátumú üzenetekben található meg.

- Azonosító kiterjesztés bit

Ha az Azonosító kiterjesztés (IDE = Identification Extension) bit – az Arbitrációs mező 13. bitje – recesszív, akkor ez egy kiterjesztett formátumú üzenetet jelöl.

Ellenkező esetben, ha a bit domináns, akkor csak standard formátumú üzenetről lehet szó. Ekkor azonban az Azonosító kiterjesztés bit már nem az Arbitrációs mezőhöz, hanem a Vezérlési mezőhöz tartozik.

Ezen Azonosító kiterjesztés bit gondoskodik arról, hogy – ha egy standard üzenet Azonosító mezeje és egy kiterjesztett üzenet Alapazonosító mezője azonos lenne, akkor – mindenül a standard üzenet prioritása legyen a magasabb.

- Kiterjesztett Azonosító mező

Ez a mező csupán kiterjesztett formátumú üzeneteknél létezik, és a 14-31. biteket foglalják el az Arbitrációs mezőből. Az Azonosító mező (29 bites) első 18 bitjét tartalmazza fordított sorrendben a 17.-től a 0. bitig.

- Üzenetküldés kérés bit

Az RTR bit értéke mindenül domináns. Standard formátum esetén ez a 12. bitje, míg kiterjesztett formátum esetén a 32. bitje az Arbitrációs mezőnek.

A bit akkor lehet recesszív, ha Adatkérő üzenetről van szó. Ezáltal ha egy Adathordozó üzenet és egy Adatkérő üzenet azonosítója megegyezik, akkor az Adathordozó üzenetnek lesz nagyobb a prioritása, azaz e két üzenet esetén megnyerné az arbitrációt.

3.3.1.3. Vezérlési mező

A Vezérlési mező hat bitet tartalmaz. Az első két bit mindenképpen domináns. Standard formátumú üzeneteknél e két bit az IDE bit és az r0 rövidítéssel jelzett foglalt bit, míg kiterjesztett formátumú üzeneteknél az r1, és az r0 foglalt bitek. minden foglalt bitet a küldő-csomópontoknak dominánsan kell elküldeniük, de a fogadó-csomópontok képesek ezeket recesszív bitként is fogadni. Ez a későbbi fejlesztések miatt lett így specifikálva.

A következő négy bit az Adathossz kód (DLC – Data Length Code), ami az Adatmezőben található adat bájtjainak a számát adja meg. Csupán a 0-8-ig terjedő kódok érvényesek, mivel nyolcnál nagyobb Adathossz kódot nem engedélyez a CAN specifikáció, hiába adna rá lehetőséget a négy bit. A legális kódok tehát alábbi táblázat szerint (D: domináns, R: recesszív):

3.1. TÁBLÁZAT - LEGÁLIS ADATHOSSZ-KÓDOK

Adatbájtok száma	Adathossz-kód bitjei			
	3.	2.	1.	0.
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D

3.3.1.4. Adatmező

Az Adatmező tartalmazza az üzenetben elküldendő adatokat. Az említettek szerint maximum nyolc bájtból állhat, és minden bájt első bitje a legnagyobb helyiértékű bit. Az Adatmező nem kötelező, akár nulla bit hosszúságú is lehet.

3.3.1.5. CRC mező

A 15 bites CRC sorozatból (CRC Sequence) és a CRC határoló bitból áll. A CRC mező segítségével ellenőrzi a fogadó fél a kapott bitfolyam hibátlanosságát. A 15 bit hosszúság 127 bitnél rövidebb üzenetek ellenőrzésére használható a leghatékonyabban. A bitbeszúrásoktól mentes SOF, Arbitrációs mező, Vezérlési mező, és Adatmező alapján számolja ki mind a küldő, mind a fogadó fél a CRC mezőt. A CAN hálózaton alkalmazott CRC kód Hamming-távolsága 6 bit, ami 5 véletlenül bekövetkező egyszeres hibát tud jelezni, vagy 15 bitnyi löketszerű hibát. (A hibaösszeg számításáról részletesen a [[3.4.2.3. szakasz](#)] fejezet ír.)

3.3.1.6. Nyugtázás mező

A Nyugtázás bitból (ACK Slot) és a Nyugtázás határoló bitból (ACK Delimiter) áll a 2 bites Nyugtázás mező. A Nyugtázás bitet a küldő csomópont minden recesszívre állítja, melyet minden olyan állomás, amelyik sikeresen ^[17]fogadta az üzenetet egy domináns bittel azonnal felülír, amelyet a küldő csomópont érzékel, mivel figyeli a CAN buszon megjelenő adatforgalmat.

A Nyugtázás bitet a recesszív Nyugtázás határoló bit követi, hogy egyértelműen elkülönítse a pozitív (domináns) nyugtát egy esetlegesen párhuzamosan kezdődő Hiba üzenettől.

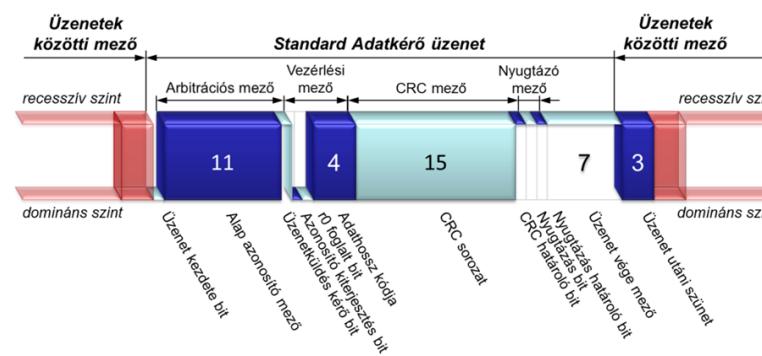
3.3.1.7. Üzenet vége mező

Minden Adathordozó és Adatkérő üzenetet 7 recesszív bitból álló sorozat zár le. A Nyugtázás határoló bittel együtt így nyolc recesszív bit jelzi az ilyen üzenetek végét. Azért van szükség ennyi bitre az Üzenet vége mezőben, hogy a hibás üzeneteket megfelelően lehessen jelezni.

A CAN buszon minden Adathordozó üzenetet szünetnek kell követnie, amelyet Üzenet utáni szünetnek (Intermission) neveznek. Ez 3 recesszív bitból áll, és csak e mezőben van lehetőség Túlcordulás üzenet ([[3.3.1.4. szakasz](#)] fejezet) küldésére.

3.3.1.2. Adatkérő üzenet

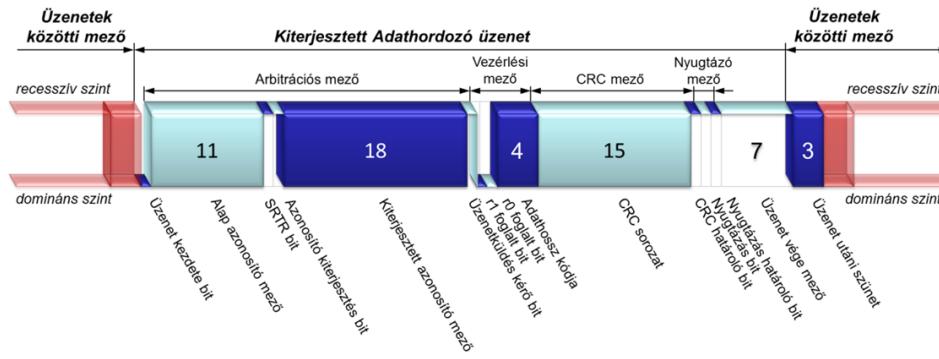
Minden csomópont kérheti a számára szükséges információ elküldését az adatot szolgáltató csomóponttól. Ehhez az igényelt Adathordozó üzenettel egyező azonosítójú Adatkérő üzenetet kell küldenie.



3.26. ábra - Standard formátumú Adatkérő üzenet

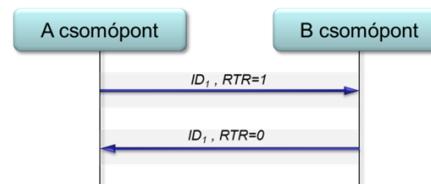
Az Adatkérő üzenet felépítése igen hasonlít az Adathordozó üzenetéhez. Formátumát tekintve szintén kétfélék, standard ([[3.26. ábra](#)]) és kiterjesztett formátumú Adatkérő üzenetek ([[3.27. ábra](#)]) különbözthetők meg.

Csupán három különbség van az Adatkérő és az Adathordozó üzenetek között. Az első, hogy adatkérésnél az Üzenetküldés kérő bit mindig recesszív szintet képvisel, ezzel biztosítva, hogy az arbitrációt az Adathordozó üzenet nyerje meg az Adatkérő üzenettel szemben. Ha ezen arbitrációs folyamat előfordulna, akkor az információt igénylő csomópont is jól jár, hiszen a kért Adathordozó üzenet kerül továbbításra. A második eltérés, hogy az adathossz-kód a kért üzenet Adatmezejére vonatkozik. A harmadik különbség pedig, hogy az Adatkérő üzenetnél az Adatmező üres, pontosabban e mező hiányzik az üzenetből.



3.27. ábra - Kiterjesztett formátumú Adatkérő üzenet

Az adatkérési ciklus lefolyásának elvét a [3.28. ábra] mutatja. Az „A” csomópont Adatkérő üzenetére a „B” csomópont azonos azonosítóval rendelkező Adathordozó üzenettel válaszol, amit természetesen nem csak az „A”, hanem az összes a buszon lévő csomópont megkap.



3.28. ábra - Adatkérési ciklus

3.3.1.3. Hibaüzenet

3.3.1.3.1. Az aktív hibaüzenet

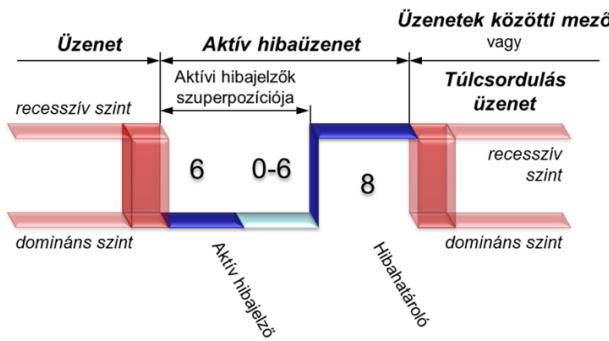
3.3.1.3.2. Passzív hibaüzenet

Bármilyen küldés, vagy fogadás (Adathordozó, Adatkérő, Hiba-, Túlcordulás üzenet) során fellépő hiba egy Hibaüzenetet generál, ami szándékosa megsérti a bitbeszúrás szabályait ([3.4.2.2. szakasz] fejezet), ezzel kényszerítve az adó csomópontot az újraküldésre.

Egy csomópont kétféle állapotban képes hibát jelezni, így két formája létezik az elküldhető Hibaüzenetnek: Aktív ([3.29. ábra]) és Passzív hibaüzenet ([3.30. ábra]). Mindkettő két tagból áll: egy Hibajelző (Error Flag) és egy Hibahatároló (Error Delimiter) mezőből.

3.3.1.3.1. Az aktív hibaüzenet

Egy csomópont Hiba aktív állapotban van, ha a saját hibaszámlálója a meghatározott érték alatt van. Hibát észlelve Aktív hibaüzenet küldésére képes. Ekkor a csomópont biztos benne, hogy hiba történt, és azt nem ő okozta.



3.29. ábra - Aktív hibaüzenet

A kezdeti hiba észlelésekor tehát egy (vagy több) Hiba aktív állapotú csomópont azonnal megszakítja a kommunikációt (kivéve CRC hiba esetén^[18]) úgy, hogy domináns biteket kezd el sugározni. Az első 6 domináns bit alkotja az Aktív hibajelző részt, az Aktív hibaüzenet első mezejét. Ezt követően recesszív bitet kezd el adni a csomópont.

Minden olyan Hiba aktív állapotban lévő csomópont, amely a kezdeti hibát nem érzékelte legkésőbb az Aktív hibajelző mező 6. domináns bitjénél hibát fog generálni, ugyanis ezen a ponton a bitbeszúrás szabálya sérül. Így legrosszabb esetben újabb 6 domináns bit fog a CAN buszon megjelenni, ezért ez a szakasz az Aktív hibajelzők szuperpozíciója. E szakasz hossza ismeretlen, 0-6 bit hosszságú lehet. Ha 0 bit hosszságú, akkor a kezdeti hibát egyszerre észlelte az összes Hiba aktív állapotú csomópont.

Ahogy a CAN buszt figyelik a csomópontok, az Aktív hibajelző 6 domináns bitet követően egy idő után recesszív bitet fog visszaolvassni minden csomópont. Ezt követően még 7 recesszív bitet sugároznak a csomópontok. Az Aktív hibaüzenet záró része tehát a 8 recesszív bitból álló Hibahatároló mező. Ezzel a módszerrel lehetséges válik egy csomópont számára, hogy érzékelje, vajon ő volt-e az első, aki hibajelést küldött, azaz elsőként észlelte a hibát. A hibás csomópontok elszigetelésénél ([[3.4.4. szakasz](#)] fejezet) fontos ez a mechanizmus.

Az Aktív hibaüzenet végén a busz ismét kész Adathordozó üzenet továbbítására. Így az a csomópont, amelyiknek adása meg lett szakítva, megkísérelheti az el nem küldött üzenet újra küldését.

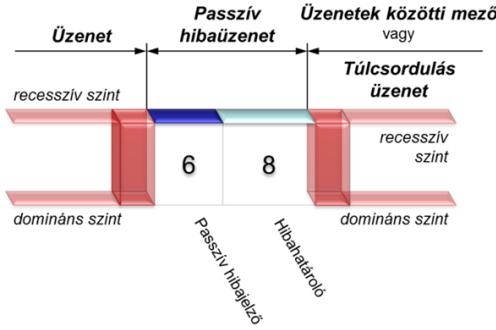
3.3.1.3.2. Passzív hibaüzenet

Egy csomópont Hiba passzív állapotban van, ha átlépte a kijelölt hibaszámláló értéket – és valószínűleg helyi meghibásodása van –, de még nem olyan súlyos a helyzet, hogy le kelljen válnia a buszról. E csomópont hiba észlelése során Passzív hibaüzenet küldésére képes.

A Passzív hibaüzenet első fele a Passzív hibajelző, amely 6 recesszív bitból áll. Ennek csak akkor van hatása, ha a Hiba passzív állapotú csomópont a megfelelő helyen kezdi el a Passzív hibaüzenet küldését. Ugyanis az Arbitrációs mezőben, illetve kevesebb, mint hat bittel a CRC sorozat vége előtt adni kezdett^[19] Passzív hibajelzőt nem érzékeli a többi csomópont.

Tehát ha egy nem buszbirtokló csomópont próbál Passzív hibajelzést adni, akkor annak nem lesz hatása a hálózat többi csomópontjára.

Hiba passzív állapotú csomópontoknak mindig ki kell várni a 6 azonos értékű recesszív bitet (Passzív hibajelző) a hiba detektálása után, hogy befejezettnek tekinthessék a hibajelzésüket, melyet a Hibahatároló 8 recesszív bit zár le, megegyezően az Aktív hibaüzenettel.



3.30. ábra - Passzív hibaüzenet

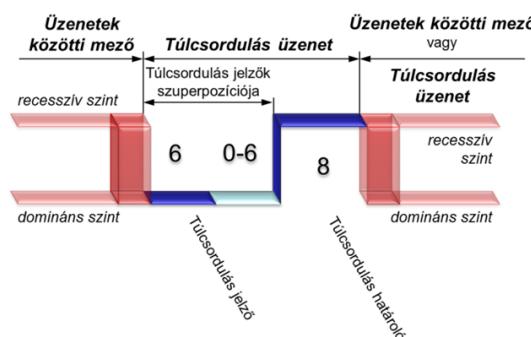
3.3.1.4. A túlcordulás üzenet

3.3.1.4.1. Túlcordulás mező

3.3.1.4.2. Túlcordulás-határoló

A Túlcordulás üzenetek ([3.31. ábra]) ugyanolyan formátuma van, mint az Aktív hibaüzenetnek. Egy csomópont három esetben küld Túlcordulás üzenetet:

- ha a fogadó csomópont késleltetni akarja a következő üzenet fogadását,
- ha a fogadó csomópont az üzenet közötti mező első két bitjén domináns bitet érzékel,
- ha a Hiba-, vagy Túlcordulás-határoló mező utolsó bitjén domináns bitet érzékel.



3.31. ábra - Túlcordulás üzenet

Túlcordulás üzenetet csak Hiba aktív állapotú fogadó csomópont küldhet, abban az esetben, ha nem kész a következő üzenet fogadására. Egymás után maximum kettő küldhető, és csupán az Üzenet utáni szünetben fordulhat elő. Szerkezeti felépítése hasonló az Aktív

hibaüzenethez, azonban nem kényszeríti ki az előző üzenet újraküldését.

3.3.1.4.1. Túlcordulás mező

A Túlcordulás jelzővel (Overload flag) kezdődik, amely 6 domináns bitból áll. Ezt követi a 0 és 6 között tetszőleges hosszú domináns bitekből álló Túlcordulás jelzők szuperpozíciója, amely a többi csomópont által generált Túlcordulás üzenetek átlapolódásából adódik össze, hasonlóan az Aktív hibaüzenethez.

3.3.1.4.2. Túlcordulás-határoló

A Túlcordulás-határoló (Overload Delimiter) tag 8 recesszív bitból áll, és a Túlcordulás üzenetet zárja le. Ugyanúgy generálódik, mint a Hibahatároló, azaz a Túlcordulás jelző befejezésekor recesszív biteket forgalmaz a csomópont, majd, ha azt is érzékel a buszon, akkor még 7 darab recesszív bitet küld, mellyel lezárja a Túlcordulás üzenetet.

3.3.1.5. Üzenetek közötti mező

3.3.1.5.1. Hiba aktív csomópont esetén

3.3.1.5.2. Hiba passzív csomópont esetén

Az Üzenetek közötti mező (Interframe Space) célja az Adathordozó és az Adatkérő üzenetek elkülönítése az öket megelőző üzenetektől. Ezzel szemben a Hiba- és a Túlcordulás üzenetek folyamatosan továbbítódnak, azaz nincs előttük ilyen Üzenetek közötti mező.

A Hibaüzenetekhez hasonlóan itt is elkülönülnek a Hiba aktív és a Hiba passzív állapotú csomópontok Üzenetek közötti mezői.

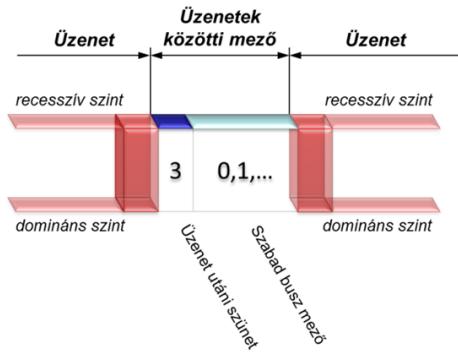
3.3.1.5.1. Hiba aktív csomópont esetén

Kötelezően az Üzenetek utáni szünettel (Intermission field) kezdődik, amely 3 recesszív bitból áll. Csak Túlcordulás üzenetet lehet küldeni közben, Adathordozó és Adatkérő üzenetek küldését nem lehet kezdeményezni.

Az Üzenetek utáni szünetet követi egy tetszőleges hosszúságú recesszív bitsorozat, a Szabad busz (Bus Idle) mező ([[3.32. ábra](#)]). Ha a busz szabaddá válik, és valamelyik csomópont küldeni akar egy üzenetet, akkor hozzáférhet a buszhoz. Az Üzenet utáni szünetet követő első biten lehet megkezdeni azoknak az üzeneteknek az elküldését, amelyeknek az átvitele függőben maradt egy másik nagyobb prioritású üzenet elküldése miatt.

Ha az Üzenet utáni szünet utolsó bitjén domináns bitet észlel egy küldésre várakozó csomópont, akkor azt egy másik csomópont SOF bitjének kell tekintenie^[20], és a küldésre várakozó csomópont a saját üzenetének azonosítóját kezdi el küldeni a következő biten, anélkül, hogy SOF bitet küldene, ezzel belépve a buszáért való versengésbe.

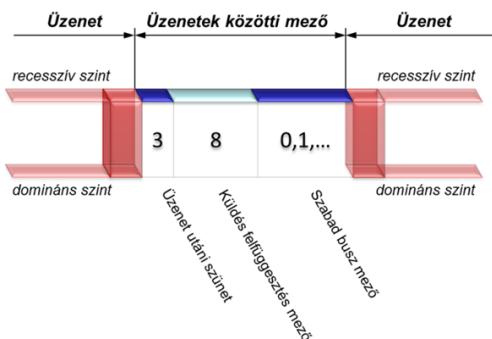
Értelemszerűen a buszáért való versengésbe a Szabad busz mező bármelyik bitjén be lehet szállni az első, SOF domináns bittel, melyet követően a Szabad busz mező lezárul.



3.32. ábra - Üzenetek közötti mező „hiba aktív” csomópontknál

3.3.1.5.2. Hiba passzív csomópont esetén

Hiba passzív állapotú csomópontok esetén az Üzenetek közötti mező közepébe egy 8 bites Küldés felfüggesztés (Suspend transmission) mező ékelődik ([3.33. ábra]). E mező biztosítja, hogy a hibásan működő csomópontok ne hátráltassák túlságosan a többi csomópontot. Egy Hiba passzív állapotban lévő csomópont köteles kiváni a Küldés felfüggesztés mezőt, mely alatt a Hiba aktív csomópontok elkezdhették a forgalmazást. Ha ez bekövetkezne, akkor a Hiba passzív állapotban lévő csomópont automatikusan fogadó-csomóponttá válna.



3.33. ábra - Üzenetek közötti mező „hiba passzív” csomópontknál

3.3.2. Üzenetek késleltetése

Ahogy a korábbiakban megvillágítást nyert, a CAN-en kétféle üzenetformátum létezik, a standard és a kiterjesztett. Mindkettőben az adatbájtok száma 0 és 8 között lehet. Az adatátviteli sebesség és a késleltetések az üzenet hosszától – ennek megfelelően az üzenet típusától és az adatbájtok számától – függnek. Egy üzenet maximális késleltetése csak a legnagyobb prioritású üzenetre nézve határozható meg, a többi üzenetre ez a paraméter – a CAN arbitrációs mechanizmusa következtében – statisztikai módszerekkel becsülhető. Standard üzenetformátumot tekintve a leghosszabb üzenet 130 bit hosszú, kiterjesztett formátum esetén 154 bit hosszú:

3.2. TÁBLÁZAT - STANDARD ÉS KITERJESZTETT FORMÁTUMÚ ADATHORDOZÓ ÜZENETEK MEZŐINEK HOSSZA BITBEN

Mezők:	Standard formátum:	Kiterjesztett formátum:
Üzenet kezdete bit	1	1
Alapazonosító mező	11	11
Üzenetküldés kérés bit hely.	-	1
Azonosító kiterjesztés bit	1	1
Kiterjesztett azonosító mező	-	18
Üzenetküldés kérés bit	1	1
Foglalt bit	1	2
Adathossz kód	4	4
Adatmező	64	64
CRC mező	15	15
CRC határoló bit	1	1
Nyugtázó bit	1	1
Nyugtázó bitet határoló bit	1	1
Üzenet vége mező	7	7
Üzenet utáni szünet mező	3	3
Lehetséges beszűrt bitek száma ^[a]	0-19	0-23
Összesen	111-130	131-154
[a] Az Üzenet kezdete bit és a CRC határoló között érvényes a bitbeszúrás, 5 egymást követő azonos bit esetén.		

Tehát standard formátumú üzenetek esetén a legnagyobb prioritású üzenetnek maximum 130 bitidőnyit kell várnia, amíg megkapja a busz használatának jogát (ez 130 µs-nyi várakozást jelent a maximális, 1Mbit/s-s átviteli sebesség mellett). Kiterjesztett formátumú üzenetek esetén ez az idő 154 bitidő hosszú (azaz a CAN legnagyobb átviteli sebessége mellett 154 µs).

Ugyancsak az üzenetek hosszától és a bitrátától függ az is, hogy mennyi idő alatt kell a mikrokontrollernek feldolgozna az érkező üzeneteket. A legrosszabb esetet tekintve (100%-os buszhasználat mellett 0 bajt adat minden üzenetben, így beszűrt bitekre sincs szükség) standard üzenetek esetében 47µs-ként érkezik új üzenet, kiterjesztett formátumú üzenetkeretek esetén ez az idő 67 µs. Ennek különösen BasicCAN architektúra használata esetén van jelentősége, hiszen ekkor minden egyes beérkezett üzenet szűrése a felhasználói alkalmazásra hárul, ami több mint 21000 (illetve kiterjesztett üzenetek esetében majdnem 15000) megszakítást jelent másodpercenként legrosszabb esetben.

3.4. Hibakezelés a CAN hálózaton

[3.4.1. Üzenet jóváhagyás](#)

[3.4.2. Hibatípusok, és Hibafelismerés](#)

[3.4.2.1. Bithiba – Bitellenőrzés](#)

[3.4.2.2. Kitöltési hiba – Bitbeszúrás ellenőrzés](#)

[3.4.2.3. CRC hiba – CRC ellenőrzés](#)

[3.4.2.4. Formai hiba – Üzenetkeret ellenőrzés](#)

[3.4.2.5. Nyugtázási hiba – Nyugtázás ellenőrzés](#)

[3.4.3. Hibafelismerési képesség](#)

[3.4.3.1. Számítási példa felderítetlen hibára](#)

[3.4.4. Hibaforrás megszüntetése, a CAN csomópont állapotgépe](#)

E fejezet egyes elemei a CAN protokoll Fizikai-, míg mások az Adatkapcsolati rétegébe tartoznak, azonban az előző fejezetek anyagát megértve az olvasó e pontnál már birtokában van a Hibakezelés fejezetben leírtak értelmezéséhez szükséges fogalmaknak, ismereteknek.

3.4.1. Üzenet jóváhagyás

A CAN protokollban a küldő és a fogadó csomópontok különböző időpontban tekintik érvényesnek az aktuális üzenetkeretet. A küldő csomópont szempontjából érvényes az üzenet, ha nem történik hibajelzés az Üzenet vége jelzés (EOF Flag) utolsó bitjéig.

A fogadó csomópont akkor hagyja jóvá az üzenetet, ha nem észleli más csomópontok hibajelzését az Üzenet vége jelzés hatodik bitjéig. Ha a hatodik bitet egy lokális hiba miatt mégis dominánsnak észleli, akkor elveti az üzenetet. Az ekkor fellépő adat inkonziszencia feloldása végett (más csomópontok elfogadtatták már az üzenetet) az üzenet vége mező 7 bit hosszú, így az utolsó biten a lokális hibával rendelkező csomópont Hibaüzenetet generálhat, ezzel újraküldésre kényszerítve az üzenet forrását. E mechanizmus magában foglalja annak a lehetőséget, hogy egyes csomópontok újra megkapják az általuk előzőleg már elfogadott üzenetet. Bizonyos alkalmazásokban kitüntetett figyelmet kell fordítani a duplikált üzenetek felismerésére, melyel magasabb rendű CAN protokollok foglalkoznak.

Tehát, ha a fogadó nem észlel hibát az Üzenet vége mező utolsó előtti bitjéig, akkor jóváhagyja az üzenetet. Így ebből a szempontból az utolsó bit már nem számít „don't-care” bit. Ha az utolsó biten domináns szintet észlel a fogadó, akkor ezt nem tekinti formai hibának ([[3.4.2.4. szakasz](#)] fejezet), és csak egy Túlcordulás üzenetet ([[3.3.1.4. szakasz](#)] fejezet) generál, ami nem kényszeríti újraküldésre a küldő csomópontot, de segíthet újraszinkronizálni egy esetlegesen rosszul szinkronizált csomópontot.

Ha mind a küldő, mind a fogadó domináns bitet észlel az Üzenet vége mező utolsó bitjén, akkor az üzenet érvényes a fogadó, de nem érvényes a küldő szempontjából. Ebben az esetben az újraküldés miatt a fogadó kétszer kapja meg ezt az üzenetet.

3.4.2. Hibatípusok, és Hibafelismerés

[3.4.2.1. Bithiba – Bitellenőrzés](#)

[3.4.2.2. Kitöltési hiba – Bitbeszúrás ellenőrzés](#)

[3.4.2.3. CRC hiba – CRC ellenőrzés](#)

[3.4.2.4. Formai hiba – Üzenetkeret ellenőrzés](#)

[3.4.2.5. Nyugtázási hiba – Nyugtázás ellenőrzés](#)

A CAN protokollnál 5 féle típusú hiba különböztethető meg, melyekről az elkövetkező fejezetekben olvashatunk.

3.4.2.1. Bithiba – Bitellenőrzés

A csomópontok miközben kiküldik a buszra a biteket, monitorozzák is azokat. Bithiba (Bit error) akkor történik, ha az elküldött bit és a monitorozott bit eltér egymástól. Kivétel, ha a csomópont recesszív bitet küld az Arbitrációs mezőben vagy a Nyugtázó mezőben. Enzen kívül, ha a küldő csomópont egy Passzív hibaüzenetet küld (6+8 recesszív bit), és a monitorozott bitek közül valamelyik is domináns, akkor ezt nem értelmezi bithibának.

3.4.2.2. Kitöltési hiba – Bitbeszúrás ellenőrzés

Ha az Üzenet kezdete bit és a CRC határoló között (itt ugyanis a bitbeszúrás szabályai érvényesek) 6 egymást követő bit értéke azonos, akkor kitöltési hiba (Stuff error) történt. A hatodik egyforma bitet kitöltési bitnek nevezzük.

Bármely csomópont Hibaüzenet generálásával jelezheti, ha bitbeszúrási hibát észlel.

3.4.2.3. CRC hiba – CRC ellenőrzés

A CRC szekvencia tartalmazza a küldő csomópont által kiszámolt CRC eredményét, amelyet a fogadó csomópontok hasonló módon generálnak. Ha a két eredmény nem egyezik meg, akkor CRC hiba történt.

A fogadó csomópontok a CRC ellenőrzést használják a fogadott adat integritásának ellenőrzésére. Ez egy ún. polinom-kódon alapuló módszer, nevét a ciklikus redundancia kód (Cyclic Redundancy Code) angol rövidítéséből kapta. Széles körben alkalmazott hibajelző kód, elsősorban magas hatásfoka, és alacsony maradék hiba aránya miatt.

Az elv szerint az üzenetek meghatározott részeit egy polinommal reprezentálják, és ezt egy előre definiált ún. generátor polinommal osztják. Ennek a modulo 2 osztásnak a maradéka alkotja a CRC szekvenciát.

Ezt a keret részeként továbbítja a küldő csomópont a buszon. A fogadók ugyanúgy kiszámítják a CRC szekvenciát a kapott üzeneten, és hibátlan kommunikáció esetén a kettő megegyezik.

Az üzenetet reprezentáló polinom együtthatóit a még bitbeszúrás előtti Üzenet kezdete bit, Arbitrációs, Vezérlési, és Adatmező bitértékei alkotják, kiegészítve 15 darab nullával a 15 legkisebb helyiértéken. A generátor polinomban a felek előre megegyeznek, ez a CAN esetében:

(27)

Az ellenőrző összeg kiszámítása szoftveresen bonyolult, azonban Peterson és Brown (1961) megmutatta, hogy shift regiszterekből egyszerű áramkörrel megvalósítható az alábbi algoritmus szerint, a gyakorlatban szinte minden ilyet használnak.

CRC_RG = 0; // a shift regiszter inicializálása

repeat

CRCNXT = NXTBIT exor CRC_RG(14);

CRC_RG(14:1) = CRC_RG(13:0); // 1 bitnyi balra tolás

```
CRC_RG(0) = 0;  
if CRCNXT then  
    CRC_RG(14:0) = CRC_RG(14:0) exor (0x4599);  
endif
```

until (CRC mező első bitje, vagy hiba történt)

Az utolsó adatbit feldolgozása után a CRC_RG shift regiszter tartalmazza a CRC ellenőrző összeget.

3.4.2.4. Formai hiba – Üzenetkeret ellenőrzés

A CAN üzeneteknek vannak rögzített domináns (pl.: foglalt bit 1) és recesszív értékű bitmezői (pl. a recesszív határoló bitek), amelyek helyességét minden csomópont ellenőrzi. Ha ezek között eltérés van, akkor formai hiba történt.

Nem tekinthető formai hibának egy fogadó csomópont által az Üzenet vége mező utolsó bitjén fogadott domináns bit, hiszen ez lehet egy másik csomópont Üzenet kezdete bitje.

Ugyanígy nem okoz formai hibát, ha bármely csomópont domináns bitet fogad a Túlcordulás üzenet utolsó bitjén.

3.4.2.5. Nyugtázási hiba – Nyugtázás ellenőrzés

A nyugtázás a Nyugtázó bittel történik úgy, hogy a küldő csomópont az üzenetben a Nyugtázó bitet recesszíven küldi el, és ha a fogadó csomópont a buszról az üzenetet hibátlanul olvassa be a Nyugtázó bitig, akkor a küldő csomópont által az üzenetben küldött recesszív Nyugtázó bitet a fogadó csomópont felülírja egy domináns bittel. Ezzel jelez a küldő csomópontnak, hogy megkapta az üzenetet. Ennek elmaradása esetén a küldő csomópont nyugtázási hibát észlel.

3.4.3. Hibafelismerési képesség

3.4.3.1. Számítási példa felderítetlen hibára

Az adatátviteli rendszerek adatintegritását erősen befolyásolják a rendszer működési körülményei (elektromágneses zavarások), és a rendszer hibafelismerő képessége. A protokollok hibafelismerő képessége elég változatos, függ az alkalmazott módszerektől. A hibafelismerő módszerekre azért van szükség, hogy a fogadó csomópontok ellenőrizni tudják az érkező adatok helyességét. Az adatintegritás statisztikai mérőszáma az ún. maradó-hiba valószínűség, ami a felderítetlen hibás üzenetek valószínűségét adja meg.

Ahhoz, hogy mérhető legyen egy rendszer hibára „hajlamosságának” mértéke, szükség van a bithiba valószínűség és az üzenetkeret-hiba arány fogalmainak tisztázására. A keretek hibaaránya megadja a hibás üzenetek arányát az összes elküldött üzenethez képest. Ezzel egy megfelelően hosszú megfigyelési periódus jellemezhető. A valószínűségét annak, hogy egy átvitt üzenet egy bitje hibás a bithiba valószínűség adja meg.

Természetesen a hibák (zavarások) nem feltétlenül érintik a hálózat összes csomópontját. A CAN hálózaton lokálisan detektált hibákat globálisan jelzik a csomópontok a [[3.3.1.3. szakasz](#)] fejezet szerint leírt Hibaüzenetek elküldésével.

A megismert hibaellenőrzési módszerek kombinálásával a CAN protokoll meglehetősen kifinomult megoldást nyújt a hibák felismerésére. minden a vezetékezéssel kapcsolatos, tehát globális hiba felismerhető a küldő csomópont bitellenőrző módszerével. A lokális, vevőkben előforduló hibákat a CRC ellenőrzés szűri ki. Akár öt véletlenszerű hibát, vagy egy maximum 15 bites löketszerű hibát képes jelezni egy kereten belül. Ezek alapján a CAN kódolásának 6 a Hamming távolsága^[21], míg más terebuszoknál ez általában 4, vagy kevesebb.

Kimerítő elemzések után megállapítható, hogy a CAN maradó-hiba valószínűsége a következő szabállyal közelíthető:

(28)

Ezen összefüggés alapján kiszámolható egy CAN-es rendszer élettartama alatt előforduló fel nem derített hibák száma.

3.4.3.1. Számítási példa felderítetlen hibára

A feladat legyen a felderítetlen hibák előfordulási gyakoriságának kiszámítása, ha az alábbi adatok ismertek:

Bit ráta = 100kBit/s

Átlagos busz forgalom = 30%

Átlagos üzenethossz = 85bit

Éves működési idő = 2200óra

Átlagos hiba arány = 10^{-3}

$3,1 * 10^9$ db üzenet/év

Maradó hiba valószínűség $4,7 * 10^{-11} * 10^{-3} = 4,7 * 10^{-14}$

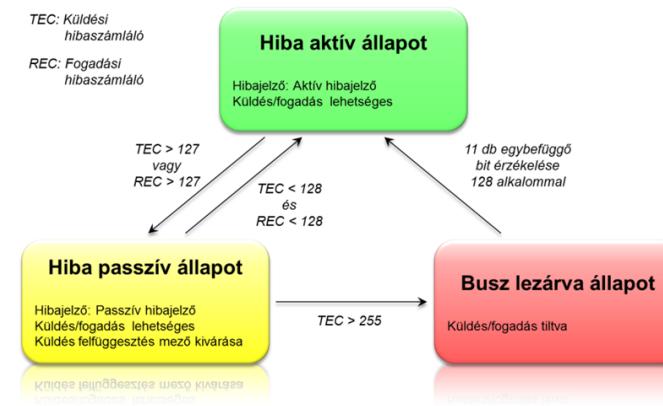
A példában szereplő hálózatra vonatkozóan ez azt jelenti, hogy megközelítőleg 6800 évente marad 1 felderítetlen hiba.

3.4.4. Hibaforrás megszüntetése, a CAN csomópont állapotgépe

A hagyományos pont-pont vezetékezés helyetti buszstruktúra alkalmazása új problémákat vet fel. Az egyik, hogy egy hibás működésű csomópont akár az egész rendszert blokkolhatja Aktív hibaüzenetek ([3.29. ábra]) küldésével. Ennek elkerülésére vezettek be a CAN protokollban egy Hiba-elszigetelési algoritmust. Ez alapján a rendszer automatikusan szabályozza a csomópontok „jogait”, akár le is kell kapcsolóniuk a hálózatról. Az algoritmus alapja két számláló, a Küldési hibaszámláló (TEC – Transmit Error Counter) és a Fogadási hibaszámláló (REC – Receive Error Counter). Ezek értéke minden sikeres küldés/fogadás után meghatározott értékkal csökken, és minden sikertelen művelet után növekszik. A Küldési és Fogadási hibaszámlálók aktuális értékei szerint a csomópontoknak 3 féle állapotuk lehetséges (a hibaállapotokról a [3.3.1.3. szakasz] fejezet a(z) részletesen ír):

- A Hiba aktív állapotban lévő csomópont részt vehet a kommunikációban, és Aktív hibaüzenetet küldhet, ha hibát észlel.
- A Hiba passzív állapotban lévő csomópontnak tilos Aktív hibaüzenetet küldenie. Részt vesz a kommunikációban, de ha hibát észlel, akkor csak Passzív hibaüzenetet küldhet, és csak a Küldés felfüggesztés mező után kezdheti meg egy másik üzenet küldését.

- A Bus off (buszkiesés) állapotban lévő csomópontnak semmiféle hatása sincs a buszon történő kommunikációra, a csomópont kimeneti meghajtója pedig ki van kapcsolva. Az, hogy fogadhat-e üzeneteket, az implementációtól függ.



3.34. ábra - CAN csomópont hibaállapotai

Tehát a Küldési és a Fogadási hibaszámláló dönti el, hogy melyik csomópont milyen állapotban van. A hibaszámlálók szabályainak megalkotásakor szempont volt, hogy egy hibát elsőként detektáló csomópont súlyozottan nagyobb „büntést” kapjon, mint a többiek. A másik fontos tényező volt, hogy az algoritmus képes legyen csökkenteni a számlálókat is, hogy az ideiglenes magasabb hiba előfordulási arányt túlélő csomópontok visszatérhessenek Hiba aktív állapotukba. Ezek alapján a hibaszámlálók a következő szabályok szerint változhatnak ([3.34. ábra]):

- Ha a fogadó csomópont hibát érzékel, akkor a Fogadási hibaszámláló értéke eggyel nő kivéve, ha egy Aktív hibajelző vagy egy Túlcordulás jelző küldése alatt következett be a bithiba.
- Ha a fogadó csomópont a saját maga által elküldött Hibaüzenet utáni első bitet domináns bitként érzékeli, akkor a Fogadási hibaszámláló értéke 8-cal nő.
- Ha a küldő csomópont küld egy Hibaüzenetet, akkor a Küldési hibaszámláló értéke 8-cal nő. A következő kivételek esetén a Küldési hibaszámláló értéke nem változik:

1-es kivétel: Ha Hiba passzív állapotban lévő küldő csomópont nyugtázási hibát érzékel, és nem detektál domináns bitet a hibához tartozó Passzív hibaüzenetet elküldése alatt.

2-es kivétel: Ha az Arbitrációs mezőben történt bitbeszúrási (azaz kitöltési) hiba miatt a fogadó csomópont Hibaüzenetet küld.

- Ha a küldő csomópont bithibát érzékel, mialatt Aktív hibaüzenetet vagy Túlcordulás üzenetet küld, akkor a Küldési hibaszámláló értéke 8-cal nő.
- Ha a fogadó csomópont bithibát érzékel, mialatt Aktív hibaüzenetet vagy Túlcordulás üzenetet küld, akkor a Fogadási hibaszámláló értéke 8-cal nő.

- minden csomópont 7 egymást követő domináns bitet tolerál Aktív hibaüzenet, Passzív hibaüzenet és Túlcordulás üzenet küldése után. minden küldő csomópontnál a Küldési hibaszámlálójának értéke és minden fogadó csomópontnál a Fogadási hibaszámlálójának értéke 8-cal nő a következő esetekben: Aktív hibaüzenet vagy Túlcordulás üzenet után 14 egymást követő domináns bit következik; Passzív hibaüzenet után 8 egymást követő domináns bit következik; valamint minden egyes 8 egymást követő domináns bit szekvencia után.
- minden helyesen elküldött, nyugtázott üzenetnél a Küldési hibaszámláló értéke eggyel csökken, ha eredetileg nem nulla volt.
- minden helyesen fogadott, nyugtázott üzenetnél a Fogadási hibaszámláló értéke eggyel csökken, ha a Hibaszámláló értéke 1 és 127 között volt. Ha nulla, akkor nem változik az értéke. Ha nagyobb mint 127, akkor a Hibaszámláló értéke 119 és 127 közé lesz beállítva.
- Ha a Fogadási hibaszámláló vagy a Küldési hibaszámláló nagyobb vagy egyenlő 128-al, akkor a csomópont Hiba passzív állapotban van.
- A csomópont akkor kerül Bus off (buszkiesés) állapotba, ha a Küldési hibaszámláló értéke nagyobb mint 255.
- A csomópont akkor van/kerül Hiba passzív állapotból Hiba aktív állapotba, ha a Küldési hibaszámláló és a Fogadási hibaszámláló értéke kisebb vagy egyenlő 127-el.
- A csomópont akkor kerül Bus off állapotból Hiba aktív állapotba, ha minden Fogadási hibaszámláló minden Küldési hibaszámláló értéke nullázódik. Ez akkor következik be, ha a Bus off állapotban lévő csomópont 128-szor érzékel 11 egymást követő recesszív bitet. Ekkor a csomópont központi egysége megkezdi az újrainicializálás a hibaszámlálók nullázásával.

3.5. CAN üzenet válaszideje

[3.5.1. Adott m üzenet legrosszabb esetben vett válaszidejének analízise](#)

[3.5.2. Válaszidőt befolyásoló tényezők](#)

[3.5.3. CAN válaszidő jitterének minimalizálása](#)

[3.5.3.1. Bitbeszűrás minimalizálása az Arbitrációs mezőben](#)

[3.5.3.2. Bitbeszűrás minimalizálása az adatmezőben](#)

Az előző évtizedekben többen is próbáltak a valós idejű rendszerek analízisével, majd 1995-ben megjelent K. Tindell, A. Burns és A. Wellings (University of York) cikke [29], melyben bemutatásra került az általuk kifejlesztett módszer. Ezen analízist olyan rendszerekre fejlesztették ki, melyekben a különböző aktivitások és a küldési egyeztetések prioritásokon alapulnak.

Az analízis bemutatása előtt célszerű bizonyos fogalmakat definiálni:

- üzenet: egyedülálló azonosítóval rendelkező, 1 és 8 bájt közötti adatot tartalmazó CAN üzenet. Feltételezett, hogy az adott üzenet ciklikusan érkezik, ugyanazzal a mérettel és azonosítóval.
- sorban állási ablak (queuing window): az adott üzenet, melyet küldeni kíván egy csomópont, egy sorban állási ablakba kerül, és egészen addig tartózkodik ott, amíg az üzenet elküldése meg nem kezdődik.
- T_m : az m üzenetre vonatkozó üzenet küldési periódus.

- J_m : az m üzenet számára a sorban állási ablak szélessége, azaz az üzenet sorban állási késesi ingadozása (jitter).
- b_m : az üzenet adatbájtjainak száma.
- C_m : a legrosszabb esete az üzenet fizikai terjedési idejének. A buszáért való versengés miatt a C_m nem tartalmazza az esetleges késéseket, csupán azt az időt, ami az Azonosítómező és egyéb üzenetrészek (pl. CRC ellenőrző) illetve az Adatmező átküldéséhez szükséges. A C_m függvénye lesz b_m -nek.
- B: a CAN hálózaton a blokkolási idő. Az a leghosszabb idő, amíg az üzenet fizikailag a buszon tartózkodik. Ez 8 bájtos üzenetnél egyenlő C-vel, és 1Mbit/sec átviteli sebességnél 130 μ s.
- R_m : az m üzenetnek a legrosszabb esetben vett válasz ideje (worst-case response time), a leghosszabb idő az üzenet sorban állása és a célállomáshoz való megérkezése között.
- D_m : az m üzenetre vonatkozó határidő (deadline).
- τ_{bit} : a buszon egy bit átviteléhez szükséges idő.

Egy üzenetet csak akkor ütemezhető, ha

A legrosszabb válaszidőre tett korlát szerint: a sorban álló üzenetet az ismételt/újra sorba állítása előtt el kell küldeni (ezzel megakadályozva az üzenet felülírását). Tehát:

(29)

Ebből látható, hogy az üzeneteknél a sorban állási ablaknak kisebbnek kell lennie, mint az üzenet küldési periódusának.

3.5.1. Adott m üzenet legrosszabb esetben vett válaszidejének analízise

A legrosszabb esetben vett válaszidő két késésből áll:

- sorban állási késés (t_m): a t_m az a leghosszabb idő, amíg egy üzenet sorban áll, ezáltal késik, mert magasabb és alacsonyabb prioritású üzenetek továbbítására vár.
- átviteli késés (C_m): az üzenetnek a buszon való tartózkodási ideje.

Tehát a legrosszabb esetben vett válaszidő:

(30)

Egy korábbi ütemező elmélet [1] alapján meghatározható egy t intervallumra vett magasabb prioritású üzenetek által okozott késés:

(31)

A $hp(m)$ egy olyan halmaz, amely (prioritási sorrendben) tartalmazza az összes olyan üzenetet, melynek prioritása magasabb m -nél. A CAN buszon egy bit átviteli idejét a τ_{bit} változó fejezi ki. A prioritások kiosztása DMA (deadline monotonic algorithm) [19] elv szerint történik, mely alapján minden prioritásnak megfelelően (deadline) rendelkező üzenet prioritása lesz a legnagyobb. A CAN esetben a jitter megjelenésével a prioritás optimális rendezését a határidő és a jitter különbsége adja:

Az eddig leírtak alapján a sorban állási késleltetés:

(32)

Ezen egyenletnek eleget tesz a t_m legkisebb értéke. Más t_m esetén a fent említett egyenlet nem rendezhető át, azonban egy rekurzív összefüggéssel felírható:

(33)

Mivel a rekurzív összefüggés t_m értékét tekintve monoton nő, az iterációt $t_m=0$ -val kell kezdeni. Ez kisebb, mint t_m legkisebb értéke, így minden esetben kielégíti az egyenletet. Habár a $t_m=0$ kezdeti érték a számításhoz megfelelő, mégis előnyösebb egy olyan i üzenet t_i értékét használni, ahol az i üzenetnek (éppen) magasabb a prioritása m -nél, hiszen ez csökkenti az iteráció futási idejét.

3.5.2. Válaszidő befolyásoló tényezők

A CAN protokoll lehetővé teszi az adatátviteli sebesség, azaz a bitsebesség (Baud rate) módosítását. Maximális értéke $1MBaud=10^6$ bit/sec.

Ha az üzenet arbitrációs száma a hálózat tervezésénél alacsonyra lett megválasztva, akkor megnyeri a buszáért való versengést, gyorsan és pontosan eljut a csomópontokhoz. Egy CAN hálózaton a legnagyobb prioritású üzenet az 1-es arbitrációs számmal rendelkező üzenet. Ezen üzenetnek minden esetben csupán a busz fizikai foglaltságát kell kivárnia.

A busz telítettsége azt jelenti, hogy magas buszforgalom esetén hosszabb a sorban állás. Több üzenet verseng a busz használatáért, mellyel felértekelődik az arbitrációs szám szerepe. Egy CAN üzenet válaszideje fordítottan arányos az arbitrációs számmal, a bitsebességgel, és egyenesen arányos a busz telítettségével.

Az üzenetek hossza a bitbeszűrás ([3.3.2. szakasz]. fejezet) következtében megnő, így egy üzenet akár 24 „nem hasznos” bitet is tartalmazhat a 111 hasznos bit mellett (standard üzenet esetén).

3.5.3. CAN válaszidő jitterének minimalizálása

3.5.3.1. Bitbeszűrás minimalizálása az Arbitrációs mezőben

3.5.3.2. Bitbeszűrás minimalizálása az adatmezőben

Az üzenetek késleltetésének (jitter) változása valós idejű alkalmazásokban kellemetlenségeket okozhat. A jitter pontos értékét több tényező (például: a buszterheltség, a számítási idő, az üzenet hosszának változása, a végrehajtási idő változása, stb.) együttes hatása alakítja ki.

Az üzenetekben a beszúrt bitek hatásának vizsgálata az egyik, és a hatásuk csökkentésére alkalmazható módszerek keresése a másik célja e fejezetnek [22]. Az alábbiakban a CAN üzenet standard formátumú üzenet jelöl, de a megfogalmazásra kerülő állításokat ki lehet bővíteni

kiterjesztett formátumú üzenetekre is.

Egy üzenet bitjeinek száma bitbeszúrás előtt:

Ahol s az Adatmező bájtjainak száma ($s = [0,8]$).

Egy üzenetben 47 vezérlő bit található, viszont csak 34 bitre érvényes a bitbeszúrás mechanizmusa. Ezért a bitek maximális száma a bitbeszúrás után:

(34)

A fenti formula teljesüléséhez a [[3.35. ábra](#)] által bemutatott látható bitmintázatra lenne szükség.

before stuffing → 111110000111100001111....
stuffed bits ↓ ↓ ↓ ↓ ↓
after stuffing → 1111100000111110000011110....

[3.35. ábra - A legrosszabb esete a bitbeszúrásnak](#)

Legyen τ_{bit} a bitidő. Így a legrosszabb esetben egy α keret átvitele a buszon:

(35)

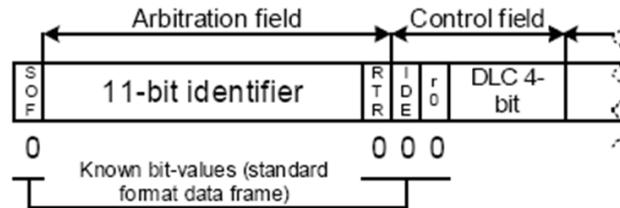
Az $\alpha = 8$ értéket választva és 1Mbit/sec buszsebességet ($\tau_{\text{bit}} = 1\mu\text{s}$) feltételezve a $C_\alpha = 135\mu\text{s}$ érték adódik.

A beszúrt bitek számának csökkentésében az Arbitrációs-, és az Adatmező játszhat szerepet.

3.5.3.1. Bitbeszúrás minimalizálása az Arbitrációs mezőben

A használható arbitrációs számok mennyiségének kis csökkentésével csökkenhető a üzenetekben előforduló beszúrt bitek maximális száma.

Egy CAN üzenet Arbitrációs mezeje ([[3.36. ábra](#)]), mely egyben az üzenet prioritását is meghatározza, 11 bitból áll és érvényes rá a bitbeszúrás.



3.36. ábra - Arbitrációs mező

Megfelelően megválasztott Azonosítók használatával minimalizálható a beszúrt bitek hatása az üzenet fejlécében. A hátránya ennek a módszernek, hogy nem használható a 11 bit által lehetővé tett 2048-féle különböző Azonosítót.

A megfelelő azonosítók kizárása után a CAN üzenet fejlécére két eset lehetséges:

- nem lesz benne beszúrt bit,
- a beszúrt bitek száma lecsökken 1-re.

Az alábbi táblázatban ([[Táblázat 3.3](#)]) megfigyelhető, hogy a 2048 db prioritásból mennyi kerül felhasználásra, ha az üzenet fejlécében adott számú beszúrt bit elérése a cél. Érdemes megfigyelni, hogy a beszúrt bitek száma függ az üzenet DLC (Data Length Code) mezőjétől is.

3.3. TÁBLÁZAT - AZ ADATMEZŐ HOSSZÁTÓL ÉS A BESZÚRT BITEK SZÁMÁTÓL FÜGGŐEN A VÁLASZTHATÓ AZONOSÍTÓK SZÁMA

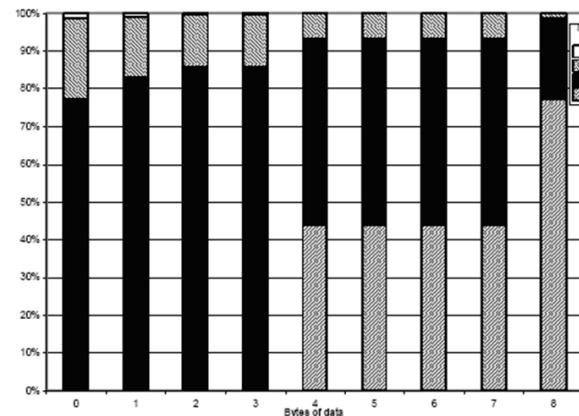
Beszúrt bitek száma	Adatbájtok száma (Adatmező hossza) a CAN üzenetben								
	0	1	2	3	4	5	6	7	8
0	0	0	0	0	897	897	897	897	1585
1	1585	1703	1763	1763	1020	1020	1020	1020	436
2	436	332	278	278	130	130	130	130	27
3	27	13	7	7	1	1	1	1	0

A [[Táblázat 3.3](#)] adatainak értelmezése:

- Az első esetnél 0-3 bájt az Adatmező hossza (2-5 oszlopok). Ekkor lehetetlen úgy megválasztani az azonosítót, hogy ne legyen az üzenet fejlécében beszúrt bit, viszont garantált, hogy maximálisan 1 beszúrt bit lesz. Ennek elérésére 0 bájtos Adatmezőnél 1585 db, 1 bájtos Adatmezőnél 1703 db, 2 és 3 bájtos Adatmezőnél 1763 db különböző prioritás kerülhet kiosztásra.
- A második esetben 4-7 bájt az Adatmező hossza. Ekkor elérhető, hogy a keret fejlécében ne legyen beszúrt bit, ahol 897 db Azonosítót lehet felhasználni.

- A harmadik esetben 8 bajt van az Adatmezőben, és szintén elkerülhetők a beszúrt bitek úgy, hogy a felhasználható prioritások száma 1585.

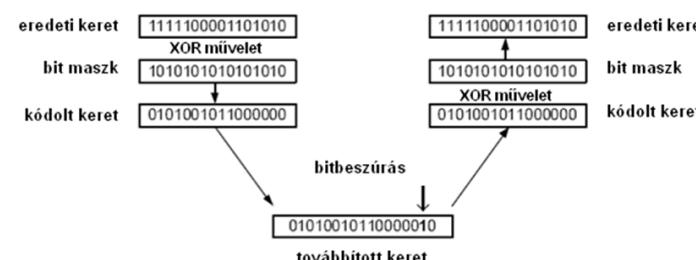
A [[3.37. ábra](#)] beszúrt bitek függvényében megmutatja, hogy adott Adatmező hossz esetén hány százaléka használható az Azonosítóknak.



[3.37. ábra](#) - CAN üzenet fejlécében előforduló prioritások valószínűsége (adott számú beszúrt bittel) az üzenetben lévő adatbájtok függvényében

3.5.3.2. Bitbeszúrás minimalizálása az adatmezőben

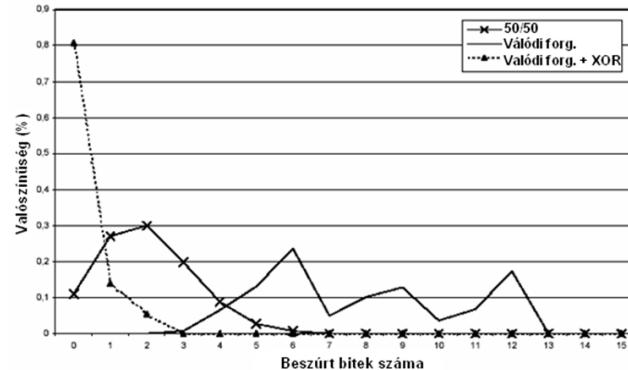
A CAN üzenet Adatmezejében is – az Arbitrációs mezőhöz hasonlóan – megjelennek a beszúrt bitek, melyek száma csak a rendszer adatforgalmának alapos vizsgálatával csökkenthető. Egy érdekes megfigyelés szerint a valós kommunikáció során az 1-es és 0-s bitek valószínűsége nem azonos. Az üzenetekbe beszúrt bitek száma átlagosan elég nagy, hozzávetőlegesen 2 és 13 közötti.



[3.38. ábra](#) - Kódolás és dekódolás

A beszúrt bitek számát csökkentő egyik módszer szerint, az összefüggő bitsorozatból váltakozó sorozat előállítása a cél. Ez megtehető a küldés előtti egész üzenet, és a 101010...10 bitmintázat közötti XOR művelet elvégzésével. A módosított üzenet fogadását követően ugyanezzel a bitmintázattal egy ismételt XOR műveletet után, az eredeti üzenet visszakapható ([[3.38. ábra](#)]). Ezzel a módszerrel a beszúrt

bitek valószínűségi eloszlása a [3.39. ábra] szerint változik. Tehát a kész üzenet XOR művelettel való kódolásával elérhető, hogy a beszúrt bitek száma az üzenetek 80%-ánál 0 legyen, amely a busz telítettségét tekintve akár 13%-os javulást is eredményezhet.



3.39. ábra - A beszúrt bitek valószínűségi eloszlásfüggvénye: 1. ha az 1-es és 0-s bitek aránya 50/50; 2. valódi adatforgalomnál; 3. manipulált valódi CAN forgalom esetén.



[10] 1986-ban a detroiti SAE kongresszuson Automotive Serial Controller Area Network néven mutatták be először.

[11] Vivőjel érzékeléses többszörös hozzáférés ütközésérzékeléssel, (Carrier Sense, Multiple Access/Collision Detection + Contention-Resolution).

[12] Programozható logikai vezérlő (Programmable Logic Controllers)

[13] 1987: Intel 82526, nem sokkal később: Philips 82C200

[14] OSEK: Open Systems and their Interfaces for the Electronics in Motor Vehicles. A rövidítés valójában német szavakból származik, így a K a Kraftfahrzeugen (Motor Vehicles) szóból. VDX: Vehicle Distributed eXecutive.

[15] A kiolvasás a beírás sorrendjében történik, az előbb beírt adatok előbb kerülnek feldolgozásra.

[16] Az Arbitrációs mező továbbítása.

[17] Egyező CRC kód a fogadó és a küldő oldalán

[18] Ebben az esetben csak a nyugtázás mező után kezdi, hogy ne zavarja a nyugtázást.

[19] És a CRC sorozat vége történetesen csupa recesszív bitból áll.

[20] Az oszcillátor tolerancia miatt lehetséges.

[21] bizonyos nagyon ritka esetekben, bitbeszúrási hibáknál ez csak 2

Vissza

2. fejezet - LIN: Local Interconnect Network

Főoldal

Előre

4. fejezet - FlexRay kommunikációs rendszer
protokoll leírás