Laboratorija za mikroprocesorsko upravljanje elektromotornih pogona
Elektrotehnički fakultet, Univerzitet u Beogradu

Implementacija CAN protokola na pogonskom kontroleru baziranom na TMS320LF2407 digitalnom signal procesoru

seminarski rad iz predmeta
Upravljački računarski sistemi

Željko Pantić, binella@etf.bg.ac.yu
Igor Stamenković, igor@etf.bg.ac.yu

Beograd, novembar 2004.

Sadržaj

I. Abstrakt – *Ovaj rad se bavi savremenim trendovima u primeni digitalne komunikacije u oblasti distribuiranih merno-upravljačkih računarskih mreža. U prvom delu se kritički predstavljaju tržišno najznačajniji industrijski komunikacioni protokoli Modbus, industrijski Eternet, Profibus i CAN. Detaljnije se analiziraju karakteristike najpre osnovnog, a zatim i specifičnosti i primena viših CAN protokola: DeviceNet, CANopen i TTCAN. Drugi deo donosi prikaz eksperimenata kojima su testirane funkcionalnosti osnovnog CAN protokola uspostavljanjem CAN komunikacije između pogonskog kontrolera baziranog na signalnom procesoru TMS320LF2407 i PC-a sa odgovarajućim adapterom koji podržava CAN protokol. Na istom eksperimentalnom setapu praktično je realizovano U/f upravljanje asinhronim motorom. Digitalna referenca za ovo upravljanje je distriburana od PC-a do pogonskog konrolera CAN baziranom komunikacijonom linijom. Dobijeni praktični rezultati predstavljaju dobru osnovu za implementaciju opisanog sistema u složenijim merno-upravljačkim sistemima.*

II. Uvod

Trendovi u savremeno dizajniranim industrijskim, vojnim, medicinskim, pa i kućnim okruženjima obuhvataju veliki broj senzora i aktuatora koji poseduju lokalnu inteligenciju i mogućnost komunikacije kako između sebe, tako i sa centralnim računarom koji koordinira njihov rad. Funkcije koje oni obavljaju na lokalnom nivou se uglavnom ograničavaju na prikupljanje i obradu podataka za senzore, odnosno upravljanju procesima električno-električne i elektromehaničke konverzije za slučaj električnih aktuatora.

Zbog potrebe preciznog upravljanja fluksom i momentom električnog motora u realnom vremenu, digitalni pogonski kontroleri sadrže moćne procesore sa velikim brojem periferijskih modula. Među njima su najznačajniji specijalizovani brojački sistemi potrebni za generisanje signala za upravljanje stanjem poluprovodničkih prekidača snage i, eventualno, prihvat impulsa sa inkrementalnog enkodera i proračun brzine i pozicije, te A/D konvertori koji digitalizuju podatke o vrednostima struja u faznim namotajima motora. Značajan broj numeričkih operacija koje se postavlja pred procesor dovodi do trenda korišćenja posebnog kontrolera čija je uloga samo komunikacija sa ostalim delovima sistema.

Komunikacija podrazumeva razmenu rezultata merenja, statusnih i upravljačkih informacija između distribuiranih merno-upravljačkih sistema, i može se ostvariti na različite načine u zavistnosti od same potrebe i svrhe sistema, ali i od upotrebljene opreme. Za ostvarivanje veze između uređaja koji potiču od različitih proizvođača je stoga potrebno da ti uređaji imaju iste komunikacione interfejse i procedure za razmenu podataka. Ovakva kompatibilnost je ostvarena definisanjem standardnog modela za komunikaciju (***O****pen* ***S****ystem* ***I****nerconnection*) uspostavljenog od strane Međunarodne organizacije za standarde (***I****nternational* ***S****tandard* ***O****rganization*). Po ovom modelu sistem povezivanja se ostvaruje preko sedam nivoa koji definišu formu komunikacionog protokola:

- fizički nivo (*physical level*) definiše električne, funkcionalne i proceduralne karakteristike koje omogućavaju transparentan prenos serije bitova; na ovom nivou sistem prepoznaje samo individualne bitove, a ne i karaktere ili poruke;

- nivo veze (*data link level)* obezbeđuje funkionalne i proceduralne mehanizme za prenos elementarnih poruka i korektnu kontrolu grešaka u prenosu;
- nivo mreže (*network level*) definiše nivo povezivanja mreže, i obezbeđuje funkcije koje ostvaruju, održavaju ili prekidaju puteve komunikacija i prenos blokova podataka između korisnika;
- nivo prenosa (*transport level*) predstavlja prvi nivo koji u potpunosti standardizuje vezu od jednog čvora do drugog; na ovom nivou se garantuje sigurna i efikasna razmena podataka koju potražuje aplikacioni program;
- nivo sesije (*session level)* obezbeđuje tehnike za organizaciju i struktuiranje blokova podataka i utvrđuje tačke sinhronizacije;
- nivo prezentacije (*presentation level*) omogućava nezavisnost aplikacije od različite prezentacije podataka tako što dozvoljava korisniku da izabere odgovarajuću sintaksu, to jest odgovarajuću konverziju prenetih podataka;
- nivo primene (*application level*) odgovara potrebama sistema; on obuhvata bibliotečke rutine koje omogućavaju komunikaciju između različitih procesa, te eventualni pristup i razmenu podataka sa serverima sistema;

Kao što se vidi iz navedenog, OSI referentni model je prilično detaljan i uopšten, pa se često za primenu u industrijksim distribuiranim merno-upravljačkim mrežama koristi uprošćen model koji obuhvata samo prvi, drugi i sedmi nivo.

Povezivanje prostorno distribuiranih inteligentnih mernih i upravljačkih modula se najčešće ostvaruje magistralom. Budući da je priroda signala kojom moduli komuniciraju digitalna, postoji mogućnost multipleksiranja prenosnog puta. Razdvajanje podataka se u tom slučaju vrši u vremenskom ili frekvencijskom domenu. Vremenski multipleks zahteva arbitriranje između uređaja korišćenjem protokola, dok je za upotrebu frekvencijskog multipleksa potreban simultani prenos u širokom frekvencijskom opsegu, što znatno poskupljuje hardver.

Aktivnost na magistrali se reguliše na dva načina: pomoću klasičnog centralnog kontrolera koji arbitrira u prenosu (poznata i kao *master-slave* komunikacija), ili bez njega kada nezavisni moduli međusobno arbitriraju. Ovaj drugi način se kod savremenih protokola najčešće ostvaruje upotrebom CSMA/CD protokola: CS označava *Carrier Sense* i znači da magistrala mora da je određeno vreme neaktivna pre nego što bilo koji čvor pošalje poruku; MA predstavlja *Multiple Access* i ukazuje da kada period neaktivnosti magistrale prođe, svaki čvor ima jednaku mogućnost da postavi poruku; CD, skraćenica od *Collision Detection*, označava da su čvorovi na magistrali u stanju da detektuju pojavu kolizije i adekvatno reaguju na nju. Kolizija je pojava koja nastaupa kada više od jenog čvora pokuša da jednovremeno pristupi magistrali u cilju slanja poruke.

Digitalni signal se može poslati serijski ili paraleleno. Paralelni prenos 8-bitnih podataka se ostvaruje IEEE 488 instrumentacionom magistralom koja obezbeđuje brzu razmenu podataka. Nedostatak ovakvom načina za prenos digitalnih signala je veliki broj linija za vezu, te se u praksi uglavnom koristi serijski prenos podataka.

Prenos se ostvaruje definisanjem različitih naponskih nivoa koji predstavljaju logičku jedinicu i nulu. On može biti realizovan na način da nivo signala zadržava istu vrednost tokom trajanja bita (*Non Return to Zero (*NRZ*) encoding*) ili kao Mančester kod. Mančester kod podrazumeva promenu naponskog nivoa na polovini trajanje bita, te je

njegova prednost česta promena stanja magistrale što omogućava resinhronizaciju na strani prijema u odnosu na svaki primljeni bit. Na ovaj način je moguće ostvariti brže prenose nego u slučaju NRZ kodiranja.

Prepoznavanje prijemnih bitova se ostvaruje očitavanjem prijemnih registara. Uslov za validnu interpretaciju poruke je svakako sposobnost prijemnika da prepozna trenutak kada je odgovarajući bit u pomenutim registrima. Takozvani *bit tajming* se ostvaruje na dva načina: asinhrono i sinhrono. Asinhrona komunikacija ne razmenjuje takt između učesnika komunikacije, tako da su start i stop bitovi neophodni. Ovakva komunikacija je jednostavnija za realizaciju. Kao proveru validnosti primljenih podataka najčešće se koristi parnost, a maksimalne brzine prenosa su ograničene do 19,2 kb/s. Sa druge strane, sinhrona komunikacija je složenija za izvođenje, ali podržava brzine prenosa do 1 Gb/s. Signal takta se šalje ili u okviru same poruke, čime prijemna strana može da ostvari resinhronizaciju, ili preko posebne taktne linije. Kao proveru validnosti primljenih podataka sinhrona komunikacija najčešće koristi CRC (*Cyclic Redundancy Check*), koji će biti objašnjen kasnije u okviru *CAN* protokola.

Kao prenosni medijum najčešće se koristi bakarni provodnik. Zahvaljujući pouzdanosti u odnosu na smetnje u prenosu, u prednosti su upredeni provodnici i koaksijalni kabl. U poslednje vreme se sve više koriste optički kablovi, koji pokazuju slabije karakteristike samo u agresivnim sredinama, poput naftnih polja ili rudnika. Za komunikaciju između veoma udaljenih sistema ili između onih koji se nalaze na nepristupačnom terenu optimalna je upotreba radio veze. Za ovakvu vezu na magistrali koristi se ista frekvencija, pri čemu se rad odvija u vremenskom multipleksu.

III. Komunikacioni protokoli

Komunikacioni protokol predstavlja precizno definisane procedure i sekvence bita, karaktera i upravljačkih kodova korišćene za prenos podataka preko komunikacione linije. Zahtevno tržište komunikacija i veliki broj proizvođača je glavni uzrok velikog broja različitih industrijskih komunikacionih mreža. U poglavljima koja slede biće analizirani najvažniji industrijski komunikacioni protokoli. Učinjen je pokušaj da se ta poglavlja koncipiraju nezavisno, tako da čitalac može što efikasnije da stekne željene informacije.

Prvi standardni interfejs za serijski prenos podataka je RS232. Električne, fizičke i funkcionalne karakteristike ovog interfejsa su standardizovane od strane Asocijacije elektro industrije EIA [2]. Njegovi osnovni nedostaci su malo rastojanje na kome se podaci mogu razmenjivati (do 15 m), relativno mala brzina prenosa (do 20 kb/s) i mogućnost povezivanja samo jednog predajnika i prijemnika. Zato je za prenos podataka na većim rastojanjima (do 1200m) i za veće brzine (do 10 Mb/s) ustanovljen EIA RS422. Ova poboljšanja su ostvarena zahvaljujući upotrebi po dve linije za prijem i predaju podataka diferencijalnim prenosom. Diferencijalni prenos je omogućio i veću imunost na smetnje, budući da se eventualni šum reflektuje na oba kabla tako da ne utiče na razliku njihovih naponskih nivoa. Novost koju je RS422 uveo je i mogućnost komunikacije jednog predajnika i deset prijemnika. Zahvaljujući dodatnom EIA standardu RS449, kojim su specificirane funkcije 37 pinova glavnog konektora i 9 pinova opcionog konektora, omogućeno je povezivanje uređaja sa RS232 interfejsom sa uređajima sa RS422 interfejsom.

Ipak, najznačajniji interfejs za industrijske primene u ovoj grupi je svakako RS485, čija je najvažnija razlika u odnosu na RS422 interfejs mogućnost komuniciranja

32 predajnika i 32 prijemnika (tzv. *Multi–drop*). Ova karakteristika je ostvarena zahvaljujući korišćenju trostatičkih bafera od strane drajvera predajnika, koji na ovaj način omogućava postavljanje izlaza predajnika u stanje visoke impedanse kada postoji aktivnost na magistrali. Karakteristike RS485 po pitanju maksimalne daljine i brzina prenosa, te osetljivosti na smetnje su ostale iste.

Razlozi zbog kojih ove serijske veze ne odgovaraju u potpunosti za rad u industrijskom okruženju su: nedostatak galvanske izolovanosti i mala robusnost u odnosu na PWM šum, problemi u povezivanju više komunikacionih čvorova paralelno na istu liniju, neusaglašenost protokola za razmenu informacija, nedovoljne brzine prenosa i odsustvo hardverskih automata za korekciju greške [1].

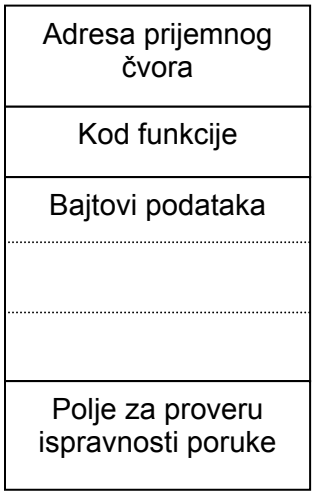
III.a. Modbus

Modbus [16] je serijski protokol koji predstavlja nadgradnju RS232, sa cilju njegove primene kao industrijskog komunikacionog protokola. Osnovu ovog protokola (nivo 1 i delimično nivo 2 OSI referentnog modela) predstavlja RS-232 kompatibilni serijski interfejs koji definiše parametre poput izlaznih pinova konektora, naponskih nivoa signala, prenosni medijum za prenos poruka, brzinu prenosa (*baud rate*), proveru parnosti (*parity checking*) i slično.

Modbus isključivo tretira *master-slave* komunikaciju u kojoj jedan od uređaja (*maste*r) inicira komunikaciju slanjem tzv. upita (*query*), na šta drugi uređaj (*slave*) odgovara dostavljajući tražene podatke *master*-u ili pak preduzimajući akciju koja je zahtevana u upitu.

Master može adresirati pojedinačni *slave*, ili pak proslediti tzv. *broadcast* poruku ka svim ostalim uređajima. U prvom slučaju prozvani *slave* odgovara na prosleđeni upit, dok u slučaju *broadcast* poruke master ne očekuje i ne dobija nikakav odgovor.

Format upita *master*-a, kao i odgovora *slave*-a je dat na slici 1:



Slika 1: *Format poruke Modbus-a*

Ako se radi o upitu, poruka započinje adresom prijemnog čvora ili broadcast adresom. Kod funkcije definiše akciju koju će adresirani uređaj preduzeti. Polje sa podacima sadrži dodatne informacije koje su neophodne *slave*-u da ostvari zahtevanu funkcionalnost. Polje za proveru ispravnosti primljene poruke (*Error Checking Field*) predstavlja metod koji omogućava *slave*-u da verifikuje ispravnost primljene poruke.

U slučaju "normalnog" odgovora *slave*-a, kod funkcije u poruci odgovora je samo eho koda funkcije iz upita. U tom slučaju, polje sa podacima sadrži informacije koje je *master* potraživao, dok *error check* polje omogućava sada *master*-u da proveri ispravnost primljene poruke. U slučaju greške pri prijemu upita, *slave* u svom odgovoru modifikuje kod funkcije na način koji nedvosmisleno ukazuje da se radi o poruci o grešci. Pri tome, bajtovi podataka sadrže kod koji detaljnije opisuje kod greške.

Modbus protokol implementira dva komunikaciona moda: ASCII i RTU (*Remote Terminal Unit*). Komunikacioni mod ne menja informacioni kvalitet polja u poruci već samo definiše način na koji će informacije biti pakovane na predajnoj strani i raspakivane na prijemnoj. Ispravan rad svih čvorova povezanih na *Modbus* mrežu zahteva da svi oni moraju biti postavljeni u identičan mod rada.

Kada je komunikacioni čvor na *Modbus* mreži setovan u ASCII modu, tada se svaki bajt poruke šalje u formi dva ASCII karaktera. Glavna prednost ovog moda u toku prijema je vremenski interval između dva uzastopna karaktera bez signalizacije do jedne sekunde u slučaju kada je nastupila greška u prenosu.

Primena RTU moda rada omogućava da se svaki bajt poruke prenosi kao dva četvorobitna heksadecimalna karaktera. Glavna prednost ovog tipa prenosa je veća gustina pakovanja karakatera, što omogućava bolje iskorišćenje komunikacione putanje pri istoj brzini prenosa nego u slučaju ASCII moda.

Bez obzira koji je komunikacioni mod korišćen, predajnik pakuje *Modbus* poruku u okvir kome je nedvosmisleno poznata početna i krajnja tačka. Time je omogućeno da prijemnik prepozna početak poruke i kraj poruke, te pročita adresni deo i tako odredi kome je poruka upućena.

Oblik poruke u ASCII modu je prikazan tabelom 1:

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	n CHARS	2 CHARS	2 CHARS CRLF

Tabela 1: *poruka u ASCII modu*

U ovom modu poruka započinje ASCII kodom znaka dvotačka (3A hex), a završava se *carriage return - line feed* parom (0D0A hex).

Oblik poruke u RTU modu je prikazan tabelom 2:

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1-T2-T3-T4	8 BITS	8 BITS	n X 8 BITS	16 BITS	T1-T2-T3-T4

Tabela 2: *poruka u RTU modu*

U RTU modu poruka započinje nakon "tihog" intervala u trajanju od najmanje 3,5 karaktera. Kako se u praksi taj interval daleko lakše implementira kao ceo broj karaktera, to je on u šemi prikazan kao T_1 - T_2 - T_3 - T_4. Otuda je prvo polje koje se prenosi kroz fizički medijum adresa prijemnog čvora. Po završetku poslednjeg primenjenog karaktera, vremenski interval zatišja na liniji za prenos od najmanje 3,5 karaktera označava kraj poruke. Nakon tog intervala može započeti nova poruka. U RTU modu se cela poruka mora preneti u kontinualnom toku. Naime, ako se u toku poruke javi "tihi" interval trajanja duži od 1,5 karaktera, prijemnik briše celu primljenu poruku i prvi naredni bajt tretira kao početak nove poruke, tj. adresu prijemnika unutar nje.

Kao što se iz prethodno izloženog već moglo naslutiti, standardni *Modbus* protokol implementira dve metode za proveru ispravnosti prenetog sadržaja.

Prva metoda je opciona provera parnosti. Ova opcija je "nasleđena" od RS232 protokola i može se primeniti na svaki pojedinačni karakter poruke. Kako je ova metoda opšte poznata, ona neće biti detaljnije diskutovana.

Specifičnost standardnog *Modbus* protokola je postojanje polja za proveru greške na nivou cele poruke. U okviru prethodnog izlaganja, u tabelama 1 i 2 je naznačeno da se sadržaj ovog polja različito formira, zavisno od toga da li je implementiran ASCII ili RTU mod rada.

U ASCII modu rada, ovo polje sadrži dva ASCII karaktera. Ovi karakteri su nastali kao rezultat proračuna takozvane longitudinalne redundanse (*Longitudinal Redundance Check* - LRC) koja je izvršena nad sadržajem poruke, ne uzimajući u obzir početnu dvotačku i završne CRLF karaktere. LRC predstavlja drugi komplement rezultata sabiranja svih preostalih bajtova poruke uz zanemarenje prenosa koji se pri sabiranju eventualno javi. Naime, njega najpre proračunava predajnik i umeće u poruku, na za to predviđeno mesto. Prijemni čvor takođe proračunava LRC tokom prijema poruke i poredi sa sadržajem primljenog LRC polja. Ako ove dve vrednosti nisu identične, konstatuje se postojanje greške u primljenoj poruci.

U RTU modu rada, polje za proveru greške sadrži 16-bitnu binarnu vrednost. Ova vrednost nastaje kao rezultat proračuna tzv. ciklične redundanse (*Cyclical Redundancy Check* - CRC). Primena CRC polja pri detektovanju greške u primljenoj poruci je identična opisanom postupku u slučaju LRC polja za ASCII mod rada. Postupak izračunavanja vrednosti CRC polja je optimalno izabran za ovaj tip poruke. Kako je sam postupak daleko složeniji nego u slučaju proračuna LRC polja, on neće biti izlagan u ovom kratkom pregledu *Modbus* protokola.

Važno je na kraju napomenuti da je provera parnosti opciona mogućnost (tj. korisnik može zabraniti da se ova provera vrši), dok je provera validnosti poruke preko LRC i CRC polja inherentna osobina *Modbus* protokola i ne može se onemogućiti u toku rada.

III.b. Industrijski Eternet (Industrial Ethernet)

Da bi se objasnio pojam industrijskog Eterneta [17], potrebno je poći od "tradicionalnog" Eterneta ili možda čak i dalje, od pojma lokalne mreže LAN (*Local Area Network*).

Lokalna mreža je skup računara i pridruženih uređaja lociranih na relativno malom prostoru (na primer, unutar jedne poslovne zgrade) koji dele zajedničku komunikacionu liniju ili bežičnu vezu (*wireless link*), kao i zajedničke resurse jednog procesora ili servera. Najčešće tehnike realizacije LAN su:

- Eternet
- *Token Ring*
- FDDI (*Fiber Distributed Data Interface*)

FDDI se isključivo koristi kao spona između LAN baziranih na Eternet i *Token Ring* tehnikama.

Token Ring mreža je LAN u kojoj se za povezivanje računara koristi topologija zvezde ili prstena, i u kojoj se kao mehanizam za sprečavanje "sudara" podataka pri pokušaju istovremenog slanja od strane dva ili više čvorova koristi specijalni bit koji se naziva token. Ovaj tip mreže u osnovi podrazumeva da neprekidno jedna "prazna" poruka cirkuliše između čvorova. Čvor koji želi da pošalje podatke umeće *token* u praznu poruku (to se najčešće postiže prostom promenom odgovarajućeg bita u poruci), dok u ostatak poruke unosi bajtove podataka i identifikator prijemnog čvora. Čvorovi koji nakon toga detektuju *token* na taj način saznaju da je neko već aktivan na liniji za prenos. Složenost protokola (što se inače manifestuje i kroz relativno malu brzinu prenosa - tipično ispod 10Mb/s), kao i smanjena pouzdanost cele LAN bazirane na *Token Ring* tehnici su glavni razlozi njene sve ređe primene.

Eternet je daleko najčešće korišćena LAN tehnika. Ovaj sistem je još 1972. godine za svoje potrebe razvila firma Xerox PARC. U neznatno izmenjenoj verziji u odnosu na prvobitnu, Eternet je i svetski ozvaničen 1985. godine od strane IEEE organizacije kroz standard IEEE 802.3, i do danas je postigao toliku popularnost da pokriva oko 85% LAN struktura u svetu.

Kao fizički medijum za prenos poruka (a što odgovara prvom nivou OSI referentnog modela) Eternet primenjuje optička vlakna, kablove sa upredenim provodnicima ili koaksijalne kablove.

Za formiranje paketa podataka za slanje (*Frame of Data*) Eternet koristi MAC (*Medium Access Control*) protokol, koji u potpunosti definiše drugi nivo OSI referentnog modela. MAC protokol enkapsulira korisne informacije sa 14-bajtnim zaglavljem (tzv. *Protocol Control Information* - PCI) i 4-bajtnim CRC (*Cycle Redundancy Check*) poljem, kojim se poruka i završava. Poruke su međusobno razdvojene kratkim periodom neaktivnosti na liniji (*idle* period) u trajanju od 9.6 μs, i uvodnom porukom dužine 8 bajtova. Period neaktivnosti na liniji je neophodan da bi se elektronskim sklopovima

prijemnog čvora omogućio dodatni vremenski interval za kompletiranje obrade prethodno primljene poruke. Uvodna poruka se sastoji od 62 naizmenične logičke jedinice i nule, uz dve logičke jedinice na samom kraju. Zadatak ove poruke je da omogući sinhonizaciju digitalne fazno kontrolisane petlje (*Digital Phase Lock Loop*) prijemnika na takt predajnika. Zaglavlje poruke ima tri funkcije:

- da definiše adresu prijemnika; pri tome se može specificirati jedinstven prijemni čvor (*unicast mode*), grupa prijemnih čvorova (*multicast mode*) ili pak specificirati da su mogući prijemnici svi dostupni čvorovi (*broadcast mode*);
- da definiše adresu predajnog čvora;
- da specificira tip protokola (najčešće korišćen je IP mrežni protokol);

Na kraju poruke se dodaje 32-bitni CRC, koji treba da obezbedi detekciju eventualne greške u primljenoj poruci. Interesantno je da MAC ne obezbeđuje predajniku nikakvu povratnu informaciju ukoliko je analizom CRC polja detektovana greška u primljenoj poruci.

Eternet mreža može biti korišćena da obezbedi deljeni pristup grupi čvorova povezanih na isti fizički medijum. Za te čvorove se često kaže sa formiraju oblast sudara (*Collision Domain*). Koji od posmatranih čvorova treba da prihvati poruku sa fizičkog medijuma se rešava preko adresa prijemnika u zaglavlju poruke. Problemi prosleđivanja informacija u suprotonom smeru i mogući "sudari" na liniji za prenos se rešavaju pomoću ranije pomenutog CSMA/CD protokola. Naime, kako Eternet koristi Mančester kod za kodovanje pojedinačnih bitova na mreži, to se prenos bilo logičke jedinice, bilo logičke nule može detektovati "slušanjem" linije za prenos. Osnovu ovog protokola predstavlja činjenica da čvor pre slanja paketa podataka najpre preko svog predajnika "sluša" fizički medijum za prenos ne bi li otkrio da li je na njemu prisutan signal nosioca. To se najčešće vrši merenjem struje koja postoji u kablu (tipična njena vrednost je između 18 i 20mA). Ako se signal nosioca ne detektuje, započinje se sa slanjem poruke.

Godinama, Eternet se razvijao i dobijao nove karakteristike mrežne inteligencije. Danas, Eternet omogućava četiri brzine prenosa podataka:

- 10BASE-T Eternet omogućava performanse od 10Mb/s kroz kabl sa upredenim parom bakarnih provodnika;
- FAST Ethernet omogućava deset puta brzi prenos (oko 100Mb/s) uz zadržavanje kompatibilnosti sa 10BASE-T realizacijom;
- Gigabit Ethernet, kao što mu samo ime kaže, omogućava brzinu protoka podataka i do 1 Gb/s i tipično se implementira kao veza između dve skretnice (*Ethernet Switch*);
- 10 Gigabite Ethernet predstavlja danas najbržu standardizovanu verziju Ethernet tehnologije. On uključuje mnoge napredne mrežne usluge i zbog svog veoma širokog propusanog opsega predstavlja odličnu osnovu za razvoj računarskih mreža na širem geografskom području, kao što su MAN (*Metropoliten Area Network*) koje se tipično primenjuju na nivou jednog grada ili jedne manje oblasti i WAN (*Wide Area Network*) koji se primenjuje za mnogo veća geografska rastojanja.

Uočavajući da Ethernet postaje dominantno rešenje u realizaciji LAN, mnoge industrijske kompanije su poželele da ga primene u cilju zamene tradicionalnih fieldbus arhitektura kao što su Modebus, ProfiBus i sl. Tako nastaje takozvani industrijski Ethernet koji primenjuje modifikovanu verziju klasičnog Ethernet-a, a predstavlja osnovu za realizaciju fabričkih upravljačkih mreža. Mada je industrijski Ethernet baziran na istom standardu kao i tradicionalna verzija, implementacija ova dva rešenja nije identična. Svaki komunikacioni protokol koji pretenduje na širu primenu u industriji mora da zadovolji tri osnovna zahteva: bezbednost poruka, pouzdanost rada i maksimalni nivo determinističkog ponašanja.

Osnovna razlika izme|u industrijskog i tradicionalnog Etherneta je u tipu hardvera koji koriste. Oprema za industrijski Ethernet je dizajnirana tako da radi i u grubim industrijskim uslovima. To uključuje primenu industrijskih komponenti, specifične metode hlađenja, izlaznu relejnu signalizaciju i sl. Napajanje je prilagođeno industrijskom okruženju, a to podrazumeva primenu 24-voltnog DC napajanja uz obaveznu primenu i redundantnog izvora za napajanje u cilju povećane pouzdanosti.

Analizirajući strukturu koju nudi OSI referentni model, može se konstatovati da je tradicionalni Ethernet dominantno lociran na njegovom drugom nivou (*data link layer*). Specijalizacija koja je učinjena kod industrijskog Ethernet-a omogućila je da se za ovaj detaljnije definišu i treći nivo (*network layer*) i četvrti nivo (*transport layer*) OSI referentnog modela. Na nivou 3 OSI referentnog modela se vodi računa o logičkom adresiranju i rutiranju poruka. Na tom nivou industrijski Ethernet dominantno koristi Internet Protocol (IP) koji je jezgro *Word Wide Web* adresiranja i rutiranja. Na nivou 4 se vodi račina o tačnosti sekvence podataka i imunosti na greške u prenosu. Industrijski Ethernet na tom nivou uobičajeno koristi dva protokola: poznatiji *Transmission Control Protocol* (TCP), te manje poznat i ređe korišćen *User Datagram Protocol* (UDP).

Industrijski Ethernet se odlikuje i u specifičnoj realizaciji *multicast* tipa prenosa. Naime, kod njega se često primenjuje tzv "proizvođač - korisnik" (*producer - consumer*) komunikacija, u kojoj jedan uređaj predstavlja izvor informacija koje prosleđuje tako da ih više uređaja može koristiti. Iako je osnovna ideja identična, zahtevi koji se postavljaju pred takve realizacije u tradicionalnom i industrijskom Ethernetu su različite. Dok se tradicionalni Ethernet dominantno fokusira na što bolje iskorišćenje raspoloživog propusnog opsega, industrijski Ethernet prioritet stavlja na specifičan zahtev za sinhronizovanim pristupom podacima od strane svih prijemnh čvorova, tj. garantovanju da će svi prijemnici podatke dobiti u isto vreme. To je u skladu sa zahtevom za što većim determinizmom u prenosu upravljačkih signala u industrijskom okruženju. Ovaj specifičan zahtev je ispunjen uključivanjem u industrijski Ethernet nekih naprednih funkcija za organizaciju i prioritiranje *multicast* prenosa.

Industrijski Ethernet se često susreće i pod nazivom *Switched* Ethernet. Ime potiče od činjenice da ovakva Ethernet arhitektura isključivo koristi skretnice (*switches*), a ne i hub-ove u implementaciji LAN. Time se postiže da svaki čvor poseduje sopstveni segment mreže i komunicira isključivo sa skretnicom, a nikada direktno sa drugim čvorom. Zbog toga od nekog čvora najpre prihvata skretnica, a zatim prosleđuje do ciljnog čvora kroz samo jedan segment mreže. Tako se postiže da više korisnika može istovremeno slati podatke bez usporavanja prenosa.

Specifičnost industrijskog Etherneta je skoro isključiva primena full-duplex veza, tj. dva odvojena provodna kanala za prijem i predaju podatak između čvora i skretnice.

Implementacijom specifične mrežne inteligencije na industrijski Ethernet moguće je realizovati fabričku komunikacionu infrastrukturu koja poseduje fleksibilnost i zaštitu poruka karakterističnih za tradicionalne fieldbus protokole, uz istovremeno daleko širi propusni opseg, otvorenu arhitekturu (*open connectivity*) i standardizaciju koju nudi Ethernet platforma.

III.c. PROFIBUS

Industrijski komunikacioni sistemi koji će biti predmet našeg daljeg razmatranja se u anglosaksonskog literaturi sreću pod terminom *fieldbus*. To su komunikacioni sistemi koji koriste različite medijume, kao što je bakarni provodnik, fiber optička vlakna ili pak bežični prenos za ostvarivanje serijskog prenosa bitova između distribuiranih uređaja (senzora, aktuatora, pogona itd.) i centralnog kontrolnog ili upravljačkog sistema. *Fieldbus* tehnologija je razvijena 80-tih godina prošlog veka sa namerom da se tradicionalno korišćeni prenos analognim signalom (4-20mA ili +/-10V interfejs) sa centralnim paralelnim ožičavanjem zameni digitalnom komunikacionom tehnologijom. Specifični zahtevi proizvođača ali i korisnika su uslovili da se u prvo vreme na tržištu pojavi više komunikacionih sistema. Da bi se ta delimična konfuzija prevazišla, izvršeno je integrisanje primarnih karakteristika glavnih komunikacionih tehnika u dva standarda, IEC 61158 i IEC 61784. PROFIBUS [18] koji će na dalje biti razmatran je integralni deo ovih standarda i komunikacioni protokol koji se u poslednje vreme izdvaja kao dominantna solucija na tržištu profesionalne opreme u oblasti industrijske i procesne automatizacije.

PROFIBUS [19] je otvoreni digitalni komunikacioni protokol sa primenom u skoro svim oblastima automatizacije (industrijska i procesna automatizacija, saobraćajno inženjerstvo, generisanje i distribucija električne energije i sl.). Stvaranje ovog protokola je vezano za zajednički projekat 21 nemačke kompanije i instituta koji je započeo 1987. godine. Cilj je bio razvoj novog serijskog protokola koji će omogućiti standardizaciju komunikacionog interfejsa različitih proizvođača. Prvi korak je predstavljao razvoj protokola PROFIBUS FMS (FMS – *Fielbus Message Specification*) koji je bio namenjen za realizaciju složenih komunikacionih zadataka. Dalji napredak je učinjen 1993. godine kada je promovisan jednostavniji i brži PROFIBUS DP (DP - *Decentralized Peripherals*) protokol koji je danas dostupan u tri verzije DP–V0, DP–V1 i DP–V2. Danas je PROFIBUS vodeći protokol sa oko 20% tržišta industrijskih komunikacionih protokla, implementiran u preko 2000 različitih proizvoda sa preko 5 miliona čvorova. U cilju koordinacije razvoja i distribucije PROFIBUS protokola oformljene su 1993. organizacija korisnika PROFIBUS–a (PNO – PROFIBUS *User Organisation*) i 1995. PROFIBUS *International* (PI).

Standardizacija PROFIBUS protokola je bazirana na razradi uprošćenog OSI modela. U tekstu koji sledi će biti detalnije razrađeni svi pomenuti elementi PROFIBUS protokola.

Tehnologije prenosa

PROFIBUS definiše 5 različitih tehnika prenosa podataka: RS485, RS485-IS, FISCO model, MBP i primenu fiberoptičke tehnologije.

RS485:

RS485 tehnologija prenosa se odlikuje jednostavnošću i niskom cenom, a primenjuje se u sistemima gde se zahteva velika brzina prenosa. Kao prenosni medijum se koristi oklopljeni upredeni par bakarnih provodnika. Ova tehnika je jednostavna za upotrebu i omogućava dodavanje, inicijalizaciju i startovanje ili pak uklanjanje čvora sa komunikacione linije bez narušavanja rada ostalih priključenih čvorova.

Moguće brzine prenosa diferencijalnim NRZ signalom su između 9.6 Kbit/s i 12 Mbit/s. Kao što je i za očekivati, u fazi podešavanja parametara sistema, svi čvorovi sistema moraju imati setovane iste brzine prenosa. U takvom slučaju, do 32 čvora (*master*-a ili *slave*-ova) je moguće povezati na jedan segment linije. Ako je potrebno povezati više od 32 čvora, onda se mora koristiti repetitor između pojedinačnih segmenata mreže u cilju "osvežavanja" signala koji se propagiraju iz jednog segmenta u drugi. Maksimalna dozvoljena dužina linije unutar jednog segmenta zavisi od brzine prenosa i manja je što je brzina prenosa veća. Svaki segment komunikacione linije mora posedovati aktivni linijski završetak, realizovan otpornicima koji se napajaju od strane samog čvora.

RS485-IS:

U cilju primene RS485 u sredinama koje zahtevaju povišen stepen bezbednosti, projektovana je nova RS485-IS (*Intrinsically Safe*) tehnologija prenosa. U okviru nje se, između ostalog, specificira nivo struje i napona svakog konkretnog čvora prema mreži koji ovaj ne sme da prevaziđe u cilju bezbednog rada. Takođe, kada se poveže više aktivnih čvorova na mrežu, suma njihovih struja ne sme da prevaziđe propisanu vrednost. Prednost RS485-IS tehnologije u odnosu na daleko rasprostranjeniji FISCO model je što omogućava da se na liniji u jednom trenutku nalazi povezano čak 32 energetski aktivna čvora.

FISCO model:

FISCO (*Fieldbus Intrinsically Safe Concept*) značajno pojednostavljuje planiranje, instalaciju i proširenje PROFIBUS mreže koja radi u potencijalno eksplozivnim sredinama. Ovaj model je razvijen od strane Nemačkog državnog tehnološkog Instituta, ali je u međuvremenu daleko prevazišao prvobitnu namenu i postao praktično standard za implementaciju PROFIBUS mreža u takvim sredinama. Model je baziran na nizu normi u pogledu napona, struje, izlazne induktivnosti i kapacitivnosti koje moraju da zadovolje uređaji koji se priključuju na mrežu, kablovi, mrežni završeci i uređaji za povezivanje više segmenata mreže. Ako su te norme zadovoljene, ne samo da se garantuje bezbednost rada, već je dozvoljena zamena uređaja na mreži čak i uređajem drugog proizvođača kao i proširivanje mreže i sve to u radnom režimu mreže. Dakle moguć je prosti *plug & play* čak i u potencijalno eksplozivnim sredinama. Na taj način se sertifikacija prenosi sa sistema na pojedinačne komponente što znatno uprošćava i pojeftinjuje implementaciju PROFIBUS mreže. Ovako lagodni uslovi za korisnika mreže su "plaćeni" veoma stogim kriterijumima po pitanju funkcionisanja uređaja namenjenog povezivanja na nju. Neki od osnovnih ograničenja koje propisuje FISCO model su:

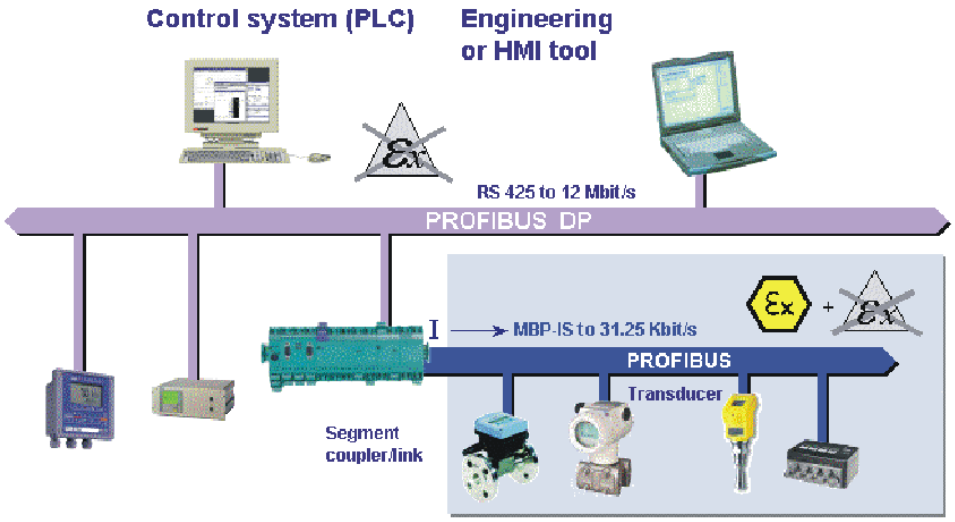
- Svaki segment mreže ima samo jedan izvor za napajanje, tzv. napojnu jedinicu;
- Linija se ne napaja kada neki od čvorova šalje podatke;
- Svaki uređaj na mreži mora da je u stanju da prihvati tačno 10mA struje u stacionarnom stanju koja treba da omogući njegovo napajanje, tj. uređaji na mreži prestavljaju pasivne potrošače struje;
- Realizuju se pasivni linijski završeci na oba kraja komunikacione linije;

- Kominikacioni signal generiše uređaj koji šalje podatke modulišući osnovnu struju sa +/-9mA;

MBP:

Skraćenica **MBP** (*Manchester Coding Bus Powered*) označava glavne atribute ove tehnike prenosa. MBP je tehnika sinhronog prenosa sa definisanom brzinom prenosa od 31.25 Kbit/s i Mančester kodovanjem pri prenosu. Ona se dominantno koristi u procesnoj automatizaciji kao i u hemijskoj i petrohemijskoj industriji jer zadovoljava bezbedonosne zahteve, a pored toga omogućava napajanje uređaja energijom sa signalinh linija. Kao fizički medijum za prenos se koristi oklopljeni dvožični kabl. Komunikaciona linija mora posedovati pasivni linijski završetak u obliku rednog RC elemetna (R=100Ω i C=2μF).

Veoma često projektanti mreže koriste inherentnu bezbednost koju poseduje ova tehnika prenosa i brzinu RS485 protokola kombinujući ove dve tehnike u okviru jedne mreže, tako da se MBP primenjuje u potencijalno opasnim sredinama, a RS485 za povezivanje kontrolnih sistema i inženjerskih uređaja u kontrolnoj sobi. Jedan takav sistema je ilustrovan na slici 2.



Slika 2*: Princip kombinovanja RS485 – MBP tehnika prenosa*

Dva segmenta mreže (RS485 i MBP) se povezuju pomoću segmentnih spojnica. Jedna vrsta tih spojnica (*segment coupler*) ne poseduje lokalnu inteligenciju i vrši samo prostu modulaciju RS485 signala u MBP signale i obrnuto. Stoga je ona transparentna za sam protokol. Druga vrsta spojnice (*sement link*) poseduje interni kontroler koji mapira sve uređaje na MBP segmentu mreže u jedan *slave* uređaj za RS485 segment. Ovo hardversko usložnjavanje sistema je veoma promišljeno učinjeno: na ovaj način se na segmentu linije od centralnog računara do spojnice može koristiti brzi RS485 prenos, dok je lokalna inteligencija spojnice zadužena za dalju distribuciju poruka.

Broj uređaja povezanih na jedan segment je ograničen na 32, ali taj broj može biti i manji ako su stroži bezbedonosni zahtevi i/ili limitirano napajanje linije. Takođe je moguć i zajednički rad uređaja koji se napajaju sa mreže i onih koji poseduju sopstveno napajanje. MBP podržava i linijske mrežne topologije i topologije mreže u obliku stabla.

Kod linijskih struktura, svi uređaji su povezani paralelno na komunikacionu liniju. Time je višežični kabl od *master*-a zamenjen sa dva komunikaciona provodnika. Međutim, ovde treba voditi računa o potkovičastim linijskim strukturama koje formiraju linije prema bilo koja dva uređaja, čija dužina može znatno da umanji maksimalnu dozvoljenu dužinu komunikacione linije.

Fiber-optička tehnologija:

Fiber-optički prenosni medijum se dominantno koristi kao prenosni medijum u dva slučaja: kada se prenos ostvaruje u sredinama sa izraženim elektromagnetnim smetnjama i/ili kada je prenos potrebno ostvariti na veoma velike distance. Glavnu prepreku češćoj primeni ove tehnologije predstavlja potreba da se interfejs tradicionalnih uređaja prilagodi tako da omogućava integraciju u optičku mrežu bez izmene protokola na fizičkom nivou. Kada to nije moguće postići, najčešće se kombinuju RS485 i optički prenosni medijum. Kao interfejs među njima se koristi neka vrsta eletrično–optičkog transformatora.

Optički prenos podržava zvezdastu, linijsku i prstenastu topologiju mreže.

DP (Decentralized Peripherals) komunikacioni protokol

DP komunikacioni protokol je dominantno razvijen za aplikacije u kojima centralni programabilni uređaj (na primer PLC, PC ili neki drugi uređaj za kontrolu procesa) ima potrebu za brzom razmenom podataka sa distribuiranim periferijskim uređajima, kao što su I/O uređaji, električni pogoni, ventili, razni davači i sl.

DP protokol je danas dostupan u tri verzije: DP-V0, DP-V1 i DP-V2, koje su i hronološki nastajale tim redom.

DP-V0:

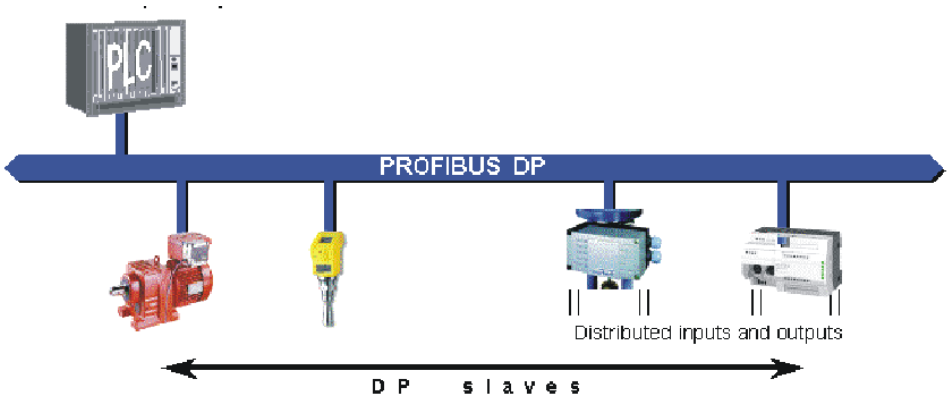
DP-V0 realizuje osnovne funkcionalnosti DP protokola, pod kojima je uglavno podrazumeva ciklična razmena podataka kao i dijagnostika uređaja periferijskog modula, kao i celog komunikacionog kanala. Centralni kontroler (*master*) je u stanju da periodično (ciklično) čita ulazne informacije od *slave*-ova i šalje izlazne informacije istima. Da bi se to postigle u realnom vremenu nije dovoljno da komunikacioni kanal ima veliku propusnu moć. Bitna je i struktura samog protokola koji treba da obezbedi jednostavan mehanizam za rukovanje porukama, mogućnost brze dijagnostike stanja i otpornost na smetnje usled preslušavanja signala.

Dijagnostičke funkcije koje implementira DP-V0 omogućavaju proveru opšte spremnosti uređaja na mreži za normalan rad (tj. detekciju stanja pregrejanosti, podnapona, prenapona i sl.), proveru I/O podsistema uređaja, kao i ispravnost funkcionisanja samog komunikacionog kanala (npr. detekciju kratkog spoja između provodnika prenosnog medijuma).

DP protokol definiše tri tipa uređaja: DPM1 (DP *master Class* 1), DPM2 (DP *master Class* 2) i *Slave* uređaje. DPM1 predstavlja formalan opis centralnog kontrolera koji periodično razmenjuje podatke sa distribuiranim *slave* uređajima. Tipični predstavnici ove grupe su PLC ili PC. DPM1 ostaruje aktivan periodičan pristup magistrali i u okviru njega vrši čitanje i upis podataka prema *slave*-ovima. DPM2 takođe ostvaruje aktivan pristup na komunikacionu liniju ali to koristi samo u periodima konfigurisanja novog komunikacionog čvora. Van tih perioda DPM2 najćešće nije ni aktivan na liniji. DPM2 su specijalni uređaji koji pomažu konfigurisanje novog čvora i učestvuju u procesu dijagnosticiranja uzroka otkaza i oporavka uređaja nakon njega. *Slave* je periferijski uređaj koji se prema komunikacionoj liniji ponaša pasivno, tj. nikad

sam ne generiše već samo reaguje na direktne upite. Takvo ponašanje *slave*-a je krajnje jednostavno za implementaciju.

DP-V0 mreža se može realizovati sa jednim (*mono-master*) i više (*multi-master*) *master*-a. U slučaju *mono-master* konfiguracije samo jedan *master* je aktivan na mreži u toku rada. Primer takve sistema je prikazan na slici 3. Na njoj je PLC centralna upravljačka komponenta, dok su *slave*-ovi povezani na njega preko prikazanog prenosnog medijuma. Ovakva konfiguracija obezbeđuje najkraće trajanje jednog komunikacionog ciklusa, tj. omogućava najefikasniji prenos.



Slika 3: *PROFIBUS DP mono-master sistem*

Multi-master konfiguracija podrazumeva da je više *master*-a povezano na komunikacionu liniju. Međutim, na nivou DP-V0 i dalje postoje određena ograničenja u toliko što je svaki *slave* na mreži pridružen samo jednom *master*-u. To znači da samo taj *master* može da upisuje sadržaj u ulazne registre tog *slave*-a, dok svi *master*-i na liniji mogu da prihvataju njegove izlazne podatke i čitaju njegove statusne registre. Projektant mreže je odgovoran za pridruživanje *slave*-ova odgovarajućem *master*-u

Da bi se obezbedio visok nivo zamenljivosti uređaja istog tipa, a različitih proizvođača, izvršena je standardizacija ponašanja sistema. Kako je ponašanje sistema dominantno određeno akcijom DPM1 uređaja, to su njegova stanja grupisane u tri kategorije: pasivno stanje (*stop*), režim brisanja (*clear*) i operativno stanje (*operate*). U pasivnom stanju nema komunikacije između DPM1 i *slave*-a. U režimu brisanja DPM1 može da čita statusne registre *slave*-a, ali van tog intervala izlaz *slave*-a su u neaktivnom stanju . Operativno stanje posrazumeva periodični radni režim u kom DPM1 prima informacije od *slave* i piše u njegove interne registre. Da bi *master* izvestio *slave*-ove o planiranoj aktivnosti, on šalje *multicast* komandu svim pridruženim *slave*-ovima sa sadržajem koji opisuje planirane aktivnosti.

Način reagovanja sistema na grešku u procesu razmene informacija je definisan u vreme konfigurisanja DPM1 uređaja. Prema tome razlikujemo dve mogućnosti: ako se detektuje greška u procesu komunikacije, DPM1 može da prevede problematični *slave* uređaj u pasivno stanje u kom on nije više u stanju da vrši razmenu informacija po komunikacionoj liniji, ili da nastavi sa normalnim radom u kom slučaju korisnik preuzima odgovornost za dalje reagovanje celog sistema.

DP-V0 definiše tri tipa komunikacije između DPM1 uređaja i *slave*-a: parametrizaciju, konfiguraciju i razmenu podataka. U procesu parametrizacije i

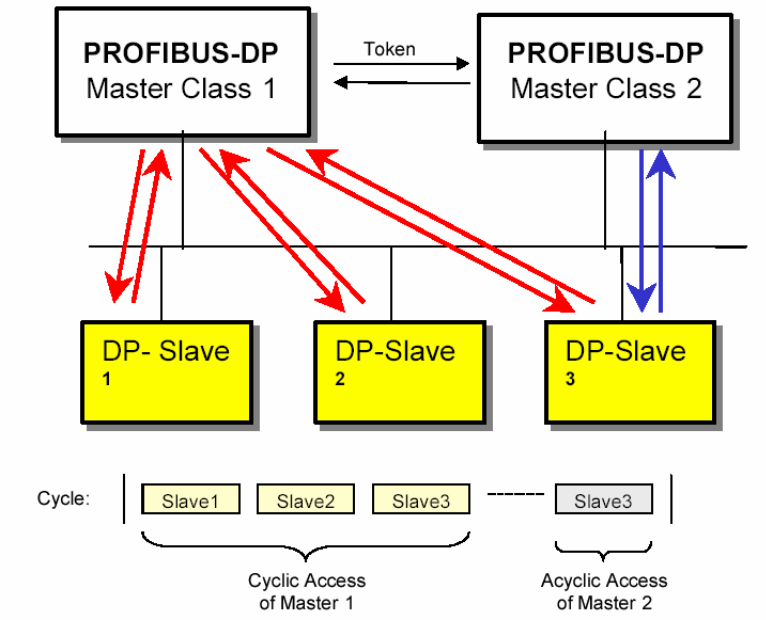
konfigurisanja DPM1 ili DPM2 uređaji prosleđuju *slave*-u informacije o njegovoj adresi (rednom broju), njegovom tipu, formatu i dužini informacija koje se razmenjuju i slično. Sistem može biti konfigurisan tako da *master* može u toku normalnog rada porukama za razmenu informacija po potrebi rekonfigurisati bilo koji *slave* na mreži.

Pored razmene podatak DMP1 može svim, ili grupi *slave*-ova slati i izvestan broj kontrolnih komandi čiji je zadatak da omoguće sinhronizaciju rada *master*-a na neki događaj na *slave* uređaju. U toj grupi komandi razlikujemo *sync* i *freeze* komande. Obe komande uzrokuju da *slave* obustavi slanje podatka *master*-u do prijema naredne iste komande. Razlika među njima je u tome što u slučaju *sync* komande *slave* neprekidno *update*-uje svoj sadržaj pri promeni vrednosti svojih eksternih ulaza, dok u slučaju *freeze* komande sadržaj se čuva, a informacije sa eksternih ulaza se u tom periodu nepovratno gube.

U toku konfigurisanja sistema, dizajner mreže definiše i jedan bitan bezbedonosni parametar. To je period monitoringa komunikacione linije. Naime i *master* i *slave* prate komunikaciju na mreži i reaguju u slučaju da se ne ostvari ni jedna transmisija u predviđenom vremenskom intervalu. DPM1 to realizuje preko više tajmera pri čemu je za svaki *slave* rezervisan po jedan tajmer. *Slave* monitoring komunikacije ostvaruje korišćenjem dobro poznate *watchdog* funkcionalnosti. Pored toga, u *multi-master* konfiguraciji, *slave* je zadužen i za kontrolu pristupa, tj. dužan je da spreči direktan pristup (pristup koji podrazumeva na samo čitanje stanja već i upis podataka) onom DPM1 uređaju kojem konkretni *slave* nije pridružen u procesu konfigurisanja sistema.

DP - V1:

Ključna novina koja je implemetnirana u ovoj verziji DP protokola je mogućnost aperiodične razmene podataka između *master*-a i *slave*-a. Ova funkcionalnost se dominantno koristi za izmenu parametara i kalibraciju *slave* uređaja u toku rada, kao i za unapređenje dijagnostike *slave* uređaja kroz alarmne i status poruke. Kombinovanje periodičnog i aperiodičnog prenosa podataka je ilustrovana na slici 4. *Master* označen sa 1 na početku ciklusa poseduje *token* i započinje periodični ciklus u kom se obraća svakom od njemu pridruženih *slave*-ova šaljući im ili preuzimajući podatke od njih. Kada se i završi obraćanje i poslednjem *slave*-u tipično do početka narednog ciklusa preostaje izvesni vremenski interval koji se naziva vremenski procep. U tom periodu *master* 1 predaje *token master*-u 2 koji onda može da pristupi željenom broju *slave*-ova, očita njihova stanja ili podatke itd.



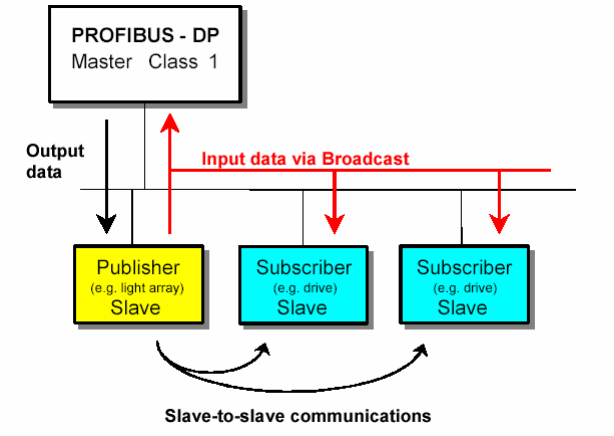
Slika 4 – *Periodična i aperiodična komunikacija na DP-V1*

Inspekcija stanja *slave* uređaja je uz pomoć protokola DP–V1 zanatno poboljšana kroz specijalne statusne i alarmne poruke. Oba tipa poruka imaju smer od *slave*-a ka *master*-u. Statusna poruka služi da izvesti *master* o trenutnom statusu *slave* uređaja u nehavarijskim uslovima rada i na nju *master* ne mora da odgovori. Alarmnu poruku *slave* šalje u slučaju nastupanja nekih havarijskih uslova i na nju se očekuje eksplicitni odgovor *master*-a. *Slave* ne šalje novu alarm poruku pre nego do njega ne stigne potvrda od *master*-a o prijemu prethodne.

DP – V2:

Ova verzija protokola unosi mnoštvo novih funkcija od kojih su najznačajnije: mogućnost direktne *slave*-*slave* komunikacije, sinhronizaciju rada *master*-a i *slave*-ova, očitavanje iz i učitavanje u *slave* željenog programskog koda i drugo.

DP–V2 po prvi put unutar DP komunikacionog protokola uvodi mogućnost direktne komunikacije *slave* čvorova na mreži. To se postiže *broadcast* porukama koje odašilje *slave* (u takvoj ulozi on se u PROFIBUS dokumentaciji naziva "*publisher*"). *Master*, ali i ostali *slave*-ovi su u stanju da registruju poruke. Ako ti *slave*-ovi imaju potrebu za informacijama koje šalje *publisher*, onda je njima u procesu inicijalizacije i konfiguracije omogućeno da prihvate i uskladište ovu poruku. *Slave*-ovi koji primaju ovu poruku se nazivaju i PROFIBUS literaturi "*subscribers*". Slika 5 ilustruje opisanu *slave*-*slave* razmenu podataka.



Slika 5: S*lave – slave razmena podataka*

Primenom ove nove funkcionalnosti se otvaraju mogućnosti za razvoj sasvim novih aplikacija. Kako *master* nije više obavezno uključen u svaku razmenu podataka na komunikacionoj liniji, to se značajno povećava brzina odziva, u nekim slučajevima čak i do 4 puta.

Sinhronizovan rad više uređaja na mreži je problem koji je bilo veoma teško rešiti primenom DP-V1 protokola. DP-V2 to rešava na dva načina. Prvi način podrazumeva primenu globalne upravljačke *broadcast* poruke koja omogućava početnu sinhronizaciju *slave* uređaja ili održavanje već postignute sinhronizacije sa devijacijom manjom od jedne μs. Visoki proiritet ove poruke omogućava očuvanje ove sinhronizacije i u slučajevima izuzetno opterećene komunikacione linije. Drugi način podrazumeva primenu tzv. vremenskih markera (*"time stamp"*). Ove poruke šalje *master* svim *slave*-ovima i u njih upisuje podatak o trenutnoj vrednosti sistemskog sata. Tako se postiže sinhronizacija događaja na *slave*-ovima unutar intervala od jedne ms. Ova funkcionalnost je posebno povoljna za sinhronizaciju različitih procesa na mreži sa više *master*-a.

DP-V1, a samim tim i naprednija varijanta protokola DP-V2 podržavaju upis ili čitanje blokova podataka od strane *master*-a. U DP-V2 je to unapređeno na taj način što je omogućeno *master*u da po potrebi očita (upload) ili upiše (download) kod u posebne memorijske oblasti na *slave* uređaju. Time se automatizuje do tada jedan manuelni posao. Čak što više, *master* ima nove mogućnosti da kontroliše funkcionisanje *slave*-a (stop, start, restart i sl.) ili da aktivira nove funkcije već startovanog pogona (kao što bi bila akvizicija izmerenih vrednosti i sl.).

Usložnjavanje komunikacionog algoritma je iziskivalo sofisticiraniji način adresiranja uređaja na mreži. U tom cilju se zahteva podela *slave* uređaja na jedan ili više modula. Ovi moduli se koriste za realizaciju periodična razmena podataka. Svakom modulu se pridružuje jedinstveni identifikator. Kada *master* šalje poruku, on na tačno određena mesta u njoj pozicionira idnetifikator. *Slave* analizira poruku, uoči identifikator i ako se on poklopi sa njegovim identifikatorom prihvata dalji sadržaj poruke. Slična je situacija i kada *slave* šalje podatke *master*-u. Tada *slave* u poruku umeće identifikator da bi *master* znao od kog uređaja je prosleđena ta poruka.

Blokovsko aperiodično čitanje i upis podataka koje uvode poslednje dve verzije protokola, DP-V1 i DP-V2 je zahtevalo dalje usložnjavanje mehanizma adresiranja. Stoga se sada određene memorijske oblasti unutar modula dele na segment kojim se pristupa preko tzv. indeksa. Svaki blok podataka je maksimalne dužine do 244 bajta i

nemu se pristupa tako što se unutar adresnog segmenta aperiodične poruke sa podacima navedu i identifikator i indeks.

Opšti aplikacijski profili (General application Profiles)

Opšti aplikacijski profili definišu funkcionalnosti i karakteristike koje su zajedničke za više različitih primena PROFIBUS komunikacionog sistema. U daljem tekstu će biti analizirana dva najznačajnija opšta aplikacijska profila: PROFIsafe i Redundancija *slave* uređaja (*Slave redundancy*). PROFIBUS organizacije PI i NPO za njih, kao iza ostale opšte aplikacijske profile, nude detaljnu dokumentaciju.

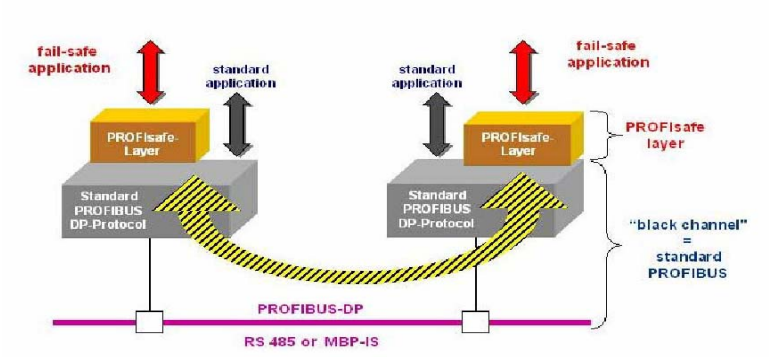
PROFIsafe:

U trenutku kada je postalo očigledno da će PROFIBUS postati jedan od dominantih *fielbus* protokola, javila se potreba da se omogući njegova primena i u sistemima fabričke i procesne automatizacije koji zahtevaju povišen stepen pouzdanosti i bezbednosti. Pored unapređenja fizičkog nivoa ovog protokola kroz razvoj RS485-IS, FISCO ili MBP tehnika, bilo je potrebno i definisati kako bezbedonosni uređaji (npr. dugme za automatsko zaustavljanje procesa, svetlosna signalizacija, alarmi, bezbedonosni ventili, kontroleri zaduženi za praćenje stanja bezbednosti i sl.) treba da komuniciraju na mreži da bi se zadovoljili najstroži nacinalni i međunarodni propisi u toj oblasti. Tako je, kao rezultat zajedničkog rada proizvođača opreme, korisnika i nacionalnih komiteta za standardizaciju, nastao PROFIsafe. PROFIsafe predstavlja otvoren protokol koji definiše specijalni format korisničkih poruka i specijalni protokol za njihovu razmenu.

PROFIsafe uzima u obzir sve moguće tipove grešaka koje se u pocesu serijske komunikacije mogu desiti: vremensko kašnjenje, gubitak podataka, ponavljanje podatka, nekorektnu sekvencu bitova u poruci, degradiran kvalitet poruke, pogrešna adresa i sl. U tom kontekstu je u okviru ovog protokola obezbeđen niz mehanizama koje treba da spreče da se ove pojave dešavaju ili bar da onemoguće njihove negativne posledice po bezbednost celog sistema:

- Slanje uzastopno numerisanih bezbedonosnih poruka;
- Vremenski *timeout* između pristigle poruke i odgovora na nju;
- Specijalni identifikacioni mehanizam (lozinka) između pošiljaoca i primaoca;
- Umetanje u poruku dodatnih bitova za proveru ispravnosti prenosa: CRC (Cyclic Redundancy Check);

PROFIsafe koristi aperiodičnu komunikaciju i može biti korišćen sa RS485, fiber-optičkim ili MBP tehnikama prenosa. Na taj način se istovremno postiže kratko vreme odziva (što je važno za primenu u industrijskoj automatizaciji) i inherentna bezbednost rada (što je bitno u procesnoj automatizaciji). PROFIsafe je jednokanalni protokol koji se implementira kao nadgradnja na nivo 7 OSI referentnog modela, tako da svi ostali standardni PROFIBUS elementi mogu biti nepromenjene. To je izuzetno povoljno jer PROFIsafe može u funkcionisati u redundantnom modu rada ili u koegzistenciji sa drugim uređajima koji poseduju samo osnovnu verziju PROFIBUS protokola. Ta izuzetno korisna osobina je ilustrovana na slici 6:



Slika 6: *PROFIsafe*

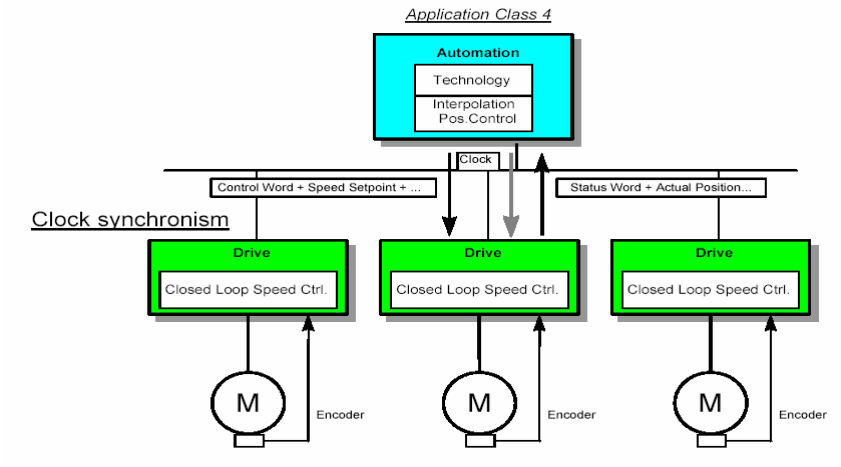
Redundancija *slave* uređaja (*Slave* redundancy):
Anlizirajući zahteve tržišta, PROFIBUS organizacije su uočile potrebu za razvojem profila koji će opisivati porocedure za razvoj i implementaciju *slave* uređaja koji poseduje redundanto ponašanje u smislu komunikacije preko merže. Kao rezultat tih napora nastala je specifikacija u kojoj jsu opisane osnovne karakteristike takvog uređaja. Takav *slave* uređaj mora da poseduje dva interface-a prema komunikacionoj liniji, koji se nazivaju primarni i backup-ovani *slave* interface. Oni mogu da budu implementirani unutar jednog ili čak u dva odvojena uređaja koji se prema mreži manifestuje kao celina. Dalje, takav uređaj (ili uređaji) dalje mora da poseduje dva nezavisna stack-a za opsluživanje protokola. Među njima treba da bude implementirana tzv. redundantna komunikacija (RedCom). Ta komunikacija je nezavisna od samog PROFIBUS protokola, a karakteristike njene realizacije u najvećoj meri određuju brzinu zamene primarnog i backup-ovanog interface-a.

U normalnom režimu rada komunikacija se isključivo obavla preko primarnog *slave*-a. Primarni *slave*, pored svojih osnovnih zadataka, poeriodično, na upit *master*a, dostavlja i informacije o stanju backup *slave*-a. U slučaju otkaza primarnog *slave*-a, sekundarni *slave* je zadužen da preuzme njegovu funkciju. Otkaz primarnog *slave*-a može detektovati i sam backup *slave*, ali može biti o tome izvešten i od strane *master*. Interesantno je konstatovati da redundatni *slave* može funkcionisati na jednoj PROFIBUs komunikacionoj liniji, ali u slučaju postojanja dodatne redundantne linije, čak i sa razdvojenim konunikacionim linijama za primarni i backup *slave* interface.

Specifični aplikacijski profili (Specific Application Profiles):

PROFIBUS je prvi protokol koji je uveo koncept specifičnih aplikacijskih profila i to je jedan od ključnih faktora koji mu je omogućio da se izdvoji među ostalim industrijskom komunikacionim protokolima. Specifični aplikacijski profil predstavlja niz pravila kako formirati interfejs specifične grupe uređaja prema mreži, kako bi se postigla zamenljivost istih ili sličnih uređaja različitih proizvođača. Pored toga ovi profili ponekad definišu i neke specifičnosti koje mora i sama komunikaciona linija da zadovolji. Sve te specifikacije su organizovane u niz dokumenata koji se mogu kupiti od PI ili NPO organizacija. Neki od najznačajnijih specifičnih profilia su namenjeni za električne pogone, procesnu automatizaciju, rukovanje robotskim sistemima, HMI uređaje, enkodere, identifikacionime sisteme, daljinsko upravljanje I/O uređajima itd.

Za nas je svakako najinteresantniji PROFIdrive profil koji definiše ponašanje električnog pogona povezanog na PROFIBUS, počev od jednostavnih frekvencijskih pretvarača do servo pogona strogih dinamičkih zahteva. Kako integracija električnog pogona u mrežno okruženje jako zavisi od namene samog pogona, to PROFIdrive definiše šest aplikacijskih klasa, kojima pokriva najveći broj mogućih primena.



Slika 7: *Regulacija pozicije preimenom PROFIdrive*

Standardni uređaji (klasa 1) zahtevaju dostavljanje samo referentne vrednosti (recimo, željene ugaone brzine motora) preko PROFIBUS mreže, dok pogonski kontroler, implementiran na samom uređaju realizuje regulacioni ili upravljački algoritam. Klasa 2 opisuje standardne pogone koji poseduju ograničenu funkcionalnost u automatizacionom procesu. U tom slučaju je automatizacioni proces podeljen u nekoliko podsistema i svaki od podsistema je preuzeo deo funkcionalnosti od centralnog programabilnog kontrolera. U tom slučaju se može reći da PROFIBUS služi kao interfejs između distribuiranih delova pogona. Pozicioni servo-pogoni (klasa 3) integrišu dodatni pozicioni kontroler. Takvo značajno povećanje procesorske moći proširuje spektra mogućih primena pogona. U realizaciji pogona klase 3 PROFIBUS omogućava da se pozicionom kontroleru dostave parametri za regulaciju pozicije i parametri trajektorije. Klase 4 i 5 opisuju pogone za najsloženije procese centralizovanog upravljanja višeosnim kretanjem. Kretanje se primarno kontroliše od strane centralnog procesnog računara (CNC), dok PROFIBUS služi da zatvori povratnu spregu i izvrši sinhronizovanje takta, kao što je i ilustrovano na slici 7. U ovu kategoriju su uključeni i savremeni koncepti upravljanjem linearnim motorima.

Distribuirana automatizacija koja pre svega uključuje taktovane procese i električnu osovinu (*electronic shaft*) je pokrivena klasom 6. Ovde se podrazumeva primena DP-V2 protokola zbog uobičajenih zahteva za *slave – slave* komunikacijom i aperiodičnim prenosom informacija.

Opšta funkcionalnost pogona je definisana skupom njegovih parametara. Za razliku od drugih profila, PROFIdrive samo definiše mehanizme za pristup parametrima i skup od tipično 30 tzv. parametara profila. Parametri profila daju osnovne informacije o radu pogonskog kontrolera, identifikaciji uređaja na mreži, trenutnom statusu pogona i slično. Svi ostali parametri (a može ih biti i preko 1000 u složenim pogonima) su pod kontrolom projektanta pogona što obezbeđuje izuzetan nivo fleksibilnosti pri implementaciji

upravljačkih funkcija. Svim tim parametrma se pristupa aperiodično pomoću DP-V1 ili DP-V2 komunikacionog protokola.

Implementacija PROFIBUS protokola

Kao pomoć pri razvoju uređaja i implementaciji PROFIBUS protokola na njima, danas je na tržištu dostupno mnoštvo standardnih komponetni i razvojnih alata. PNO i PI na svojim site-ovima daju kataloški pregled komponenti i razvijenih uređaja različitih proizvođača.

Izbor komponenti i topologije procesorskog sistema je bitan koraka u razvoju uređaja sa implementiranim PROFIBUS protokolom, pa će njemu biti posvećena posebna pažnja.

Upotreba gotovih PROFIBUS interfejsnih modula se preporučuje za uređaje za koje se ne planira serijska proizvodnja. Ovi moduli, ne veći od obične kreditne kartice su svojim interfejsom u potpunosti prilagođeni standardnim konektorima glavne ploče uređaja i mogu se uvek montirati kao dodatni modul.

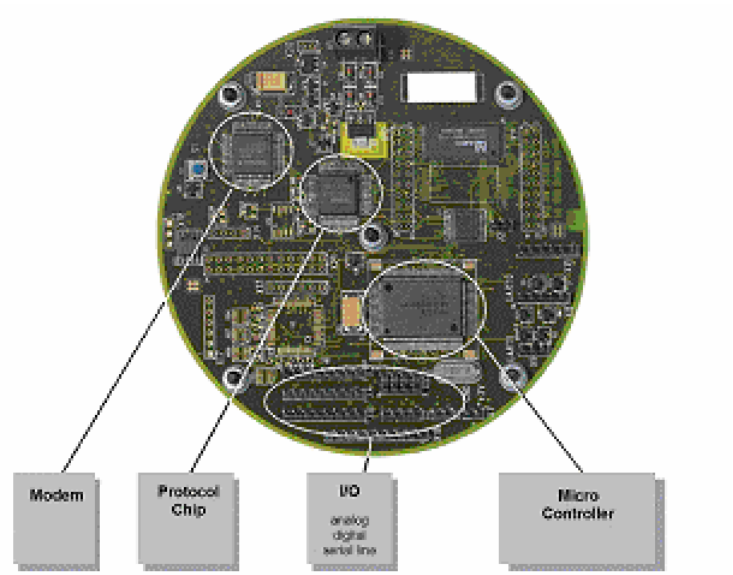
U slučaju većeg broja uređaja, preporučuje se namenski razvoj PROFIBUS interfejsa, jer se samo na taj način mogu maksimalno iskoristiti sve prednosti koje nudi PROFIBUS protokol. U tom slučaju treba koristiti standardne komponente, preporučene od strane PNO. Prvi korak je donošenje odluke da li na ploči koja se razvija primeniti samo jedan čip, primeniti komunikacioni čip sa implementiranim manjim ili većim delom protokola uz postojanje dodatnog kontrolera ili primeniti tzv protokol čip sa integrisanim mikrokontrolerom. Odluka najčešće zavisi od kompleksnosti uređaja, zahtevanih performansi i funkcionalnosti.

Realizacija jednostavnih *slave* uređaja se može uspešno ostvariti primenom jednog ASIC, pri čemu su sve funkcije protokola već integrisane u njemu. U najvećem broju sličajeva, jedino se još zahteva realizacija elektronike za napajanje kao i primena *driver*-a komunikacione linije i kristala kvarca radi definisanja željene učestanosti rada sistema.

Pri realizaciji inteligentnog *slave* uređaja od ključne važnosti je kvalitetna implementacija 2. nivoa PROFIBUS protokola. Tom nivou protokola se najčešće posveti jedan protokol čip dok se ostali deo protokola implementira u mikrokontroleru. Na taj način se vremenski kritičan deo protokola izdvaja u poseban čip i uređaj globalno čini bržim i pouzdanijim. Savremena ASIC dostupna na tržištu imaju u sebi implementirane sve periodične funkcije protokola i jednastavno se mogu primeniti u ulozi protokol čipa. Mikrokontroler može biti poseban čip ili integrisan sa protokol čipom.

Ista struktura se preporučuje i za realizaciju uređaja koji obavljaju kompleksne *master*-ske funkcije.

Primena MBP tehnike prenosa zahteva da se posebna pažnja posveti niskoj potrošnji energije uređaja koji se napajaju sa komunikacione linije. Iskustveno pravilo kaže da struja napajanja interfejsa i merne elektronike svih uređaja koji se napajaju sa *bus*-a ne sme da premaši 15mA, što je veoma restriktivno ograničenje. Da bi se taj zahtev ispunio, tipično se primenjuje dodatni modemski čip. On sa jedne strane preuzima energiju sa signalnih linija i prilagođava je potrebama napajanja celog uređaja. Sa druge strane on prihvata digitalni signal od protokol čipa i njima moduliše signal na istoj liniji. Tipičan izgled jednog komercijalno dostupnog interfejsa sa opisanim modemskim čipom je dat na slici 8.



Slika 8: *Primer realizacije PROFIBUS slave-a*

Za uređaje koje se ne mogu napajati sa komunikacione linije, jeftini, robusni i široko rasprostranjeni RS485 interfejs je najčešće dovoljno dobro rešenje.

PROFInet

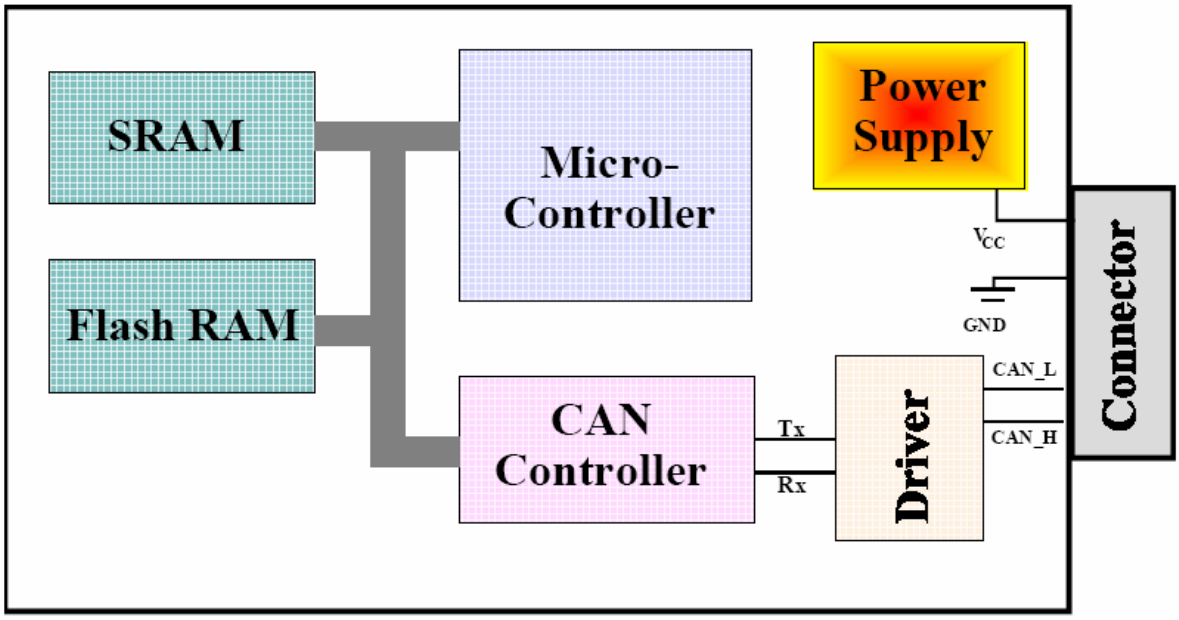
Na nivou većih sistema PLC kontroleri i PCs komuniciraju međusobno a i sa IT sistemima unutar kontrolnih soba razmenjujući velike količine podataka, koji se najčešće prenose kroz neeksplozinu, bezbednu sredinu, zaštićenu od jakih elektromagnetnih smetnji. Uočavajući da u tavim uslovima i sa takvim komunikacionim zahtevima standardi kao što su TCP/IP, Eternet, Intranet i Internet imaju značajne prednosti nad PROFIBUS-om, NPO je razvila i standardizovala PROFInet, komunikacioni protokol baziran na Eternetu koji koristi standardne mehanizme IT komunikacije u realnom vremenu. Pod tim se podrazumeva upotreba TCP/IP i COM/DCOM (*Microsoft Component Object Model* i njegovu ekstenziju za distribuirane sistem – *Distributed COM*) standarda, trenutno najrasprostanjenijih PC protokola. Prednosti primene PROFInet-a u odnosu na, recimo, Eternet se manifestuju tek ako je komunikacija na nižem nivou, nivou distribuiranog sistema ostvarena primenom PROFIBUS-a sa kojim je PROFInet maksimalno kompatibilan.

IV. CAN bazirane mreže

CAN protokol ostvaruje komunikacionu putanju koja povezuje sve čvorove (*nodes*) povezane na magistralu. Čvor predstavlja deo celog sistema ili mreže koji ostvaruje jednu ili više funkcija u okviru njega. Tipična struktura čvora je predstavljena na slici 9. Čvor podrazumeva postojanje procesora (mikrokontroler), koji zavisno od svog moda rada koristi internu ili eksternu memoriju za skladištenje podataka i programa, i koji ostvaruje potrebnu digitalnu obradu podataka. Na prikazanoj slici CAN protokol je implementiran primenom specijalizovanog CAN kontrolera, koji sa jedne strane predstavlja konekciju čvora na mrežu, a sa druge strane razmenjuje primopredajne

informacije sa procesorom. Ovakav CAN kontroler je u stručnoj literaturi poznat kao *stand alone.* Savremene realizacije mikrokontrolera visokog stepena integracije, pak, omogućava postojanje integrisanog modula koji ostvaruje sve funkcionalnosti CAN protokola. Takva realizacija je pouzdanija, jeftinija i lakša za implementaciju u aplikacijama koje zahtevaju digitalnu obradu signala na nivou čvora. Nasuprot tome, kada se ne zahteva digitalna obrada signala na ovom nivou, tipična struktura čvora podrazumeva upotrebu *stand alone* CAN kontrolera.

Napajanje čvora može biti realizovano interno, u okviru hardverskog sklopa čvora, kao što je prikazano na slici 9. Za čvorove manjih energetskih zahteva (poput inteligentnih senzora sa manjim brojem funkcija), u cilju smanjenja gabarita čvora napajanje je ostvareno preko komunikacione linije.



Slika 9: *primer CAN čvora*

IV.a. Osnovni CAN

IV.a.i. Format CAN poruke

CAN protokol je zasnovan na poruci, što praktično znači da do svakog čvora u sistemu stiže poruka sa magistrale (za razliku od sistema zasnovanom na adresi u okviru koga poruka stiže samo do delova sistema sa naznačenom adresom). Zahvaljujući tome do svakog novopostavljenog čvora poruke stiže odmah po njegovom postavljanju na magistralu bez potrebe rekonfiguracije sistema. CAN hardver u okviru svakog čvora ostvaruje lokalno filtriranje koje omogućuje da samo određeni čvorovi mogu da reaguju na pristiglu poruku.

CAN protokol definiše četiri tipa poruke:

- poruku sa podacima (*data frame*);

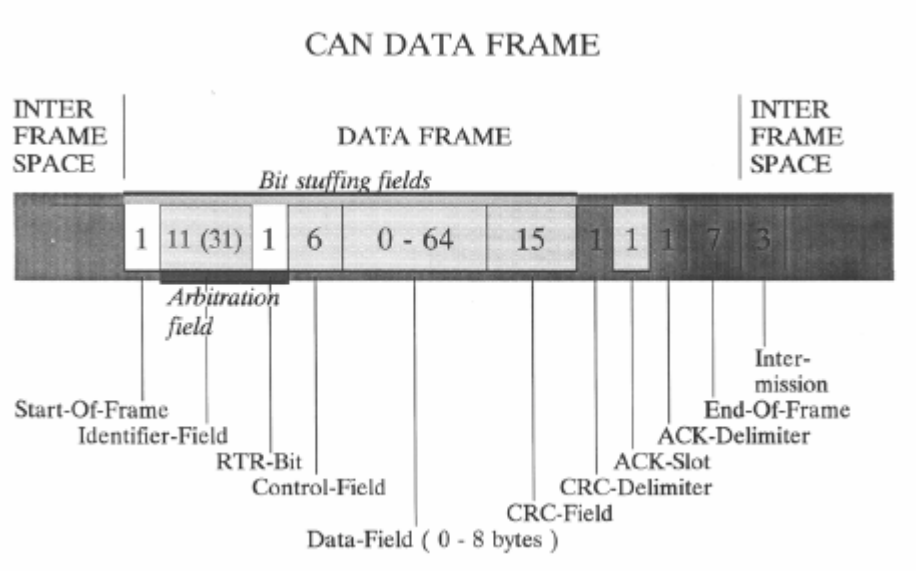
- poruku sa zahtevom za podacima (*remote frame)*;
- poruku greške (*error frame*);
- poruku o zauzetosti (*overload frame*);

Poruka sa podacima

Ovo je najčešći tip poruke i koristi se kada čvor šalje korisne podatke ostalim čvorovima u sistemu. Format poruke sa podacima, prikazan na slici 10, sadrži sledeća polja:

- start bit (s*tart of frame*), koji predstavlja jedan dominantan bit (pojam dominantnog i recesivnog bita biće naknadno objašnjen);
- polje arbitracije (*arbritation field)*, koje određuje prioritet poruke kada dva ili više čvorova zajedno postaju aktivni na magistrali; ono sadrži:
    - u slučaju CAN 2.0A protokola (poznatijeg kao ‘standardni CAN’ protokol) 11-bitni identifikator poruke;
    - u slučaju CAN 2.0B protokola (poznatijeg kao ‘prošireni CAN’ protokol) 29-bitni identifikator i dva dodatna bita (SRR – *Substitute Remote Requst*, IDE – *Identifier Extension Bit*) koja su recesivna;

  Oba protokola podrazumevaju da se ovo polje završava jednim dominantnim RTR (*Remote Transmission Request*) bitom;
- kontrolno polje (*control field*); ovo polje počinje sa dva rezervisana bita, i nastavlja se 4-bitno DLC polje (*data lenght code*), koje koduje dužinu polja podataka;
- polje podataka (*data field*), koje sadrži do osam bajtova korisnih podataka;
- CRC polje, koje predstavlja 15-bitnu vrednost proračunatu na osnovu svih prethodnih delova poruke, a koristi se za proveru tačnosti poslatih podataka; proračun ove vrednosti biće prikazan u okviru poglavlja koji opisuje upravljanje greškama;
- jednobitni, recesivni CRC *Delimiter*;
- polje potvrde (*acknowledgement slot*), koje predstavlja recesivan bit;
- jednobitni, recesivni ACK *Delimiter*;
- kraj poruke (*end of frame*), iza koga sledi IFS (*Inter Frame Space*) koji predstavlja minimalan broj bitova koji odvajaju susedne poruke;



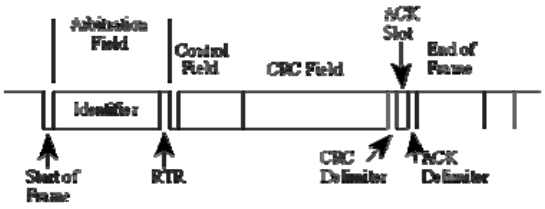
Slika 10: *format poruke sa podacima*

Poruka sa zahtevom za podacima

Ovaj tip poruke se koristi kada jedan čvor traži informaciju od drugog čvora. On se od poruke sa podacima razlikuje u sledećem:

- RTR bit u polju arbritracije je recesivan, čime se direktno ukazuje da je u pitanju poruka sa zahtevom za podacima;
- nema polja podataka;

U DLC polju poruke sa zahtevom za podacima se nalazi upisana vrednost koja eksplicitno određuje dužinu polja podataka odgovora. CAN protokol ostavlja izvesnu slobodu korisniku da definiše način na koji će prozvani čvor reagovati na poruku ovog tipa: on može biti podešen ili da CAN kontroler sam automatski odgovori slanjem odgovarajućeg podatka, ili da prekidnim zahtevom obavesti lokalni procesor o pristiglom zahtevu.

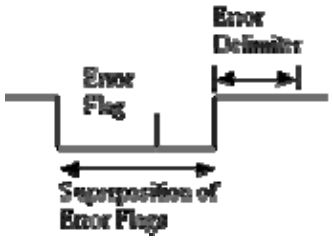


Slika 11: *format poruke sa zahtevom za podacima*

Poruka greške

Poruka greške predstavlja specijalnu poruku koja namerno ruši standardni format CAN poruke. Nju najpre šalje čvor koji je prvi uočio grešku sa namerom da upozori ostale čvorove na grešku, posle čega svaki od njih šalje poruku ovog tipa.

Poruka je sastavljena od flega greške (*error flag*), koji se sastoji od 6 bitova istog naponskog nivoa koji će sigurno prouzrokovati *staff error*, o kome će kasnije biti reči. Iza njega sledi *error delimiter* koji se sastoji od 8 recesivnih bitova, čiji je cilj da omogući ostalim čvorovima da pošalju svoju poruku greške kada su primetili prvi fleg greške.



Slika 12: *format poruke greške*

Poruka o zauzetosti

Ovaj tip poruke se šalje kada je čvoru potrebno još vremena da procesira podatke koje je već primio. Savremeni brzi procesori podataka retko susreću ovakve poteškoće, tako da se u praksi ovakva poruka može očekivati samo kod starih CAN kontrolera, poput Intelovog 82526.

Standardni i prošireni format CAN-a

CAN protokol razlikuje sledeća dva formata prema dužini identifikatora:

- standardni CAN, koji definiše 11-bitni identifikator, i koji se u tehničkoj literatruri najčešće označava 2.0A;
- prošireni CAN, koji definiše 29-bitni identifikator, sa oznakom 2.0B;

Podatak o dužini identifikatora sa nalazi u okviru IDE bita u okviru polja arbitracije. Kompatibilnost između dva različita formata je očuvana. To praktično znači da kontroleri koji podržavaju različite formate mogu biti prisutni u istoj mreži, dokle god kontroleri sa implementiranim 2.0B formatom šalju samo standardni format.

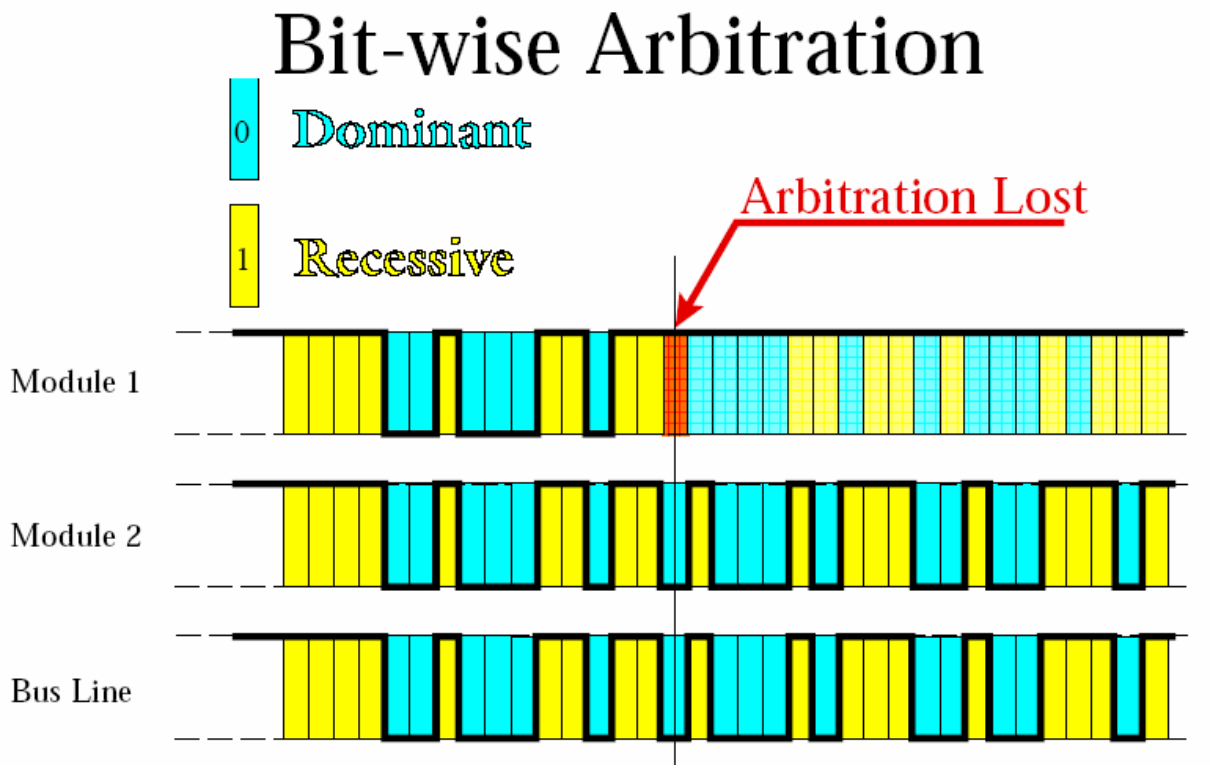
Savremni CAN kontroleri najčešće podržavaju 2.0B format. Ipak, i u okviru njega postoje dva podformata: aktivni i pasivni. Prvi od njih ima mogućnost i primanja i slanja poruka sa proširenim formatom, dok 2.0B pasivni format toleriše prisustvo proširenog formata na magistrali, odgovora bitom potvrde na takvu poruku, ali ne reaguje na nju.

Arbitracija na magistrali

Prioritet poruke koja se pojavljuje na magistrali se određuje arbitracijom. Prema CAN protokolu sadržaj arbitracionog polja određuje prioritet. Ranije je pomenuto da je logička nula dominantan bit, što znači da poruka sa numerički najmanjom vrednošću u arbitracionom polju ima najviši prioritet.

Ukupan broj različitih identifikatora koje se može upotrebiti zavisi od formata kako samog kontolera, tako i ostalih učesnika na magistrali. Ukoliko se na magistrali nalaze samo kontroleri koji podržavaju 2.0B format, mogući broj različitih identifikatora je 536870912, za slučaj postojanja 2.0A formata na magistrali taj broj iznosi 2048. Ukoliko se na magistrali, pak, nalazi stariji CAN kontroleri, zbog očuvanja kompatibilnosti nije dozvoljeno da najvažnijih sedam bitova identifikatora budu sve logičke jedinice; otuda je broj mogućih različitih vrednosti 11-bitnih identifikatora 2032, a 29-bitnih 532676608.

Aktivnost na magistrali dozvoljena je za svaki čvor po isteku predefinisanog minimalnog vremenkog intervala, to jest ranije pomenutog IFS intervala. Sam proces arbitracije se dešava kada dva ili više čvorova istovremeno počnu slanje poruke. Tokom slanja oni stalno proveravaju stanje magistrale. Kada jedan čvor primeti dominantan bit u trenutku kada je sam poslao recesivan, on se povlači. Čvor koji je slao poruku sa većim prioritetom nastavlja slanje bez znanja da je “pobedio” u arbitraciji. Na ovaj način nema utroška dodatnog vremena za procesa arbitracije. Čvor koji se “povukao” pokušava sa slanjem svoje poruke po završetku slanja tekuće poruke i predefinisanog vremena mirovanja magistrale.



Slika 13: *proces arbitracije*

Ovakva arbitracija stoga predstavlja nedestruktivnu *bit-wise* arbitraciju, i ona traje tokom celog arbitracionog polja. Jasno je da je za njenu realizaciju neophodno da svaka dva čvora imaju različite sadržaje arbitracionog polja. Izuzetak predstavljaju poruke koje ne sadrže podatke.

IV.a.ii. Fizički nivo CAN protokola

Format kodovanja koji definiše CAN protokol je NRZ kod sa umetanjem bitova *(bit-stuffing)*. Dva različita stanja električnog signala su definisana kao dominantan (za logičku nulu) i recesivan (za logičku jedinicu). Naponski nivoi ovih električnih signala zavise od fizičke veze kojom je CAN implementiran. Postoje više različitih načina za ovu realizaciju, među kojima su najznačajniji:

- veza izvedena sa dve žice, poznatija kao *high-speed* CAN, koja je definisana CAN standardom ISO 11898;
- modifikovana veza sa dve žice koja omogućava realizaciju uspešnog prenosa i u slučaju kada je jedna žica prekinuta ili uzemljena; ona podržava manje brzine prenosa, te je poznata kao *low-speed* CAN, a definisana je CAN standardom ISO 11519;
- veza izvedena sa jednom žicom prenosa signala i masom, koja se uglavnom koristi u automobilskom okruženju; standard je definisan kao SAE J2411;

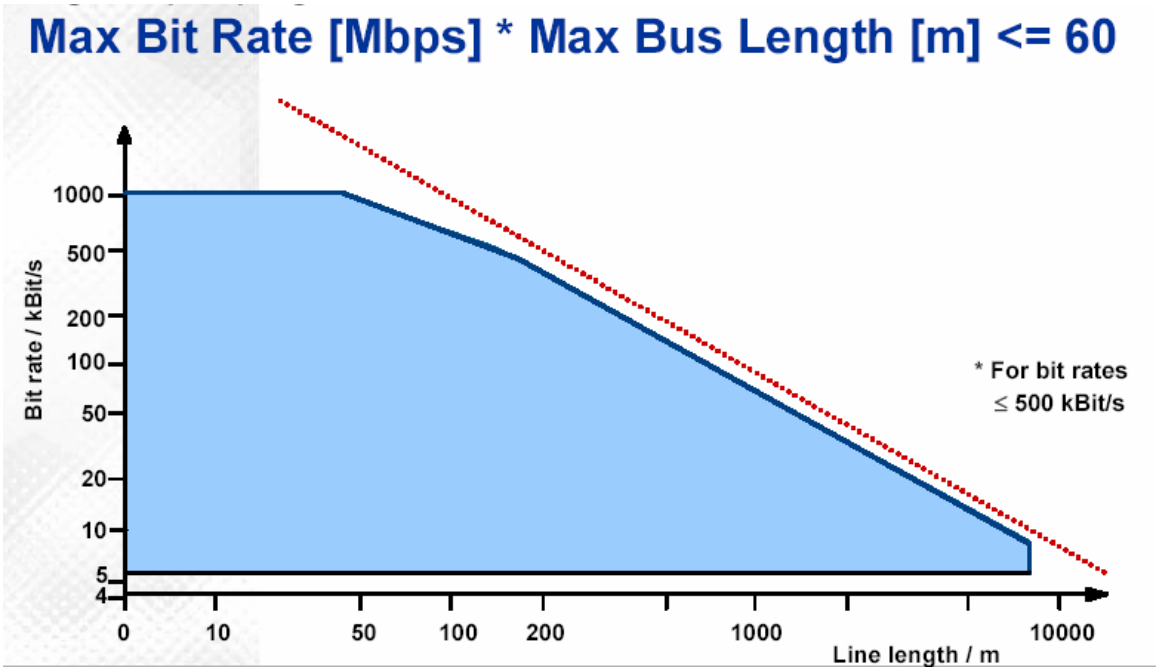
Prve dve realizacije prenose signal diferencijalno, kao razlika potencijala dva kabla koja su označena kao CANH (CAN *High*) i CANL (CAN *Low*). Budući da su oba kabla pod uticajem istih elektromagnetnih smetnji, to se neće odraziti na njihovu razliku što omogućava značajnu imunost sistema na spoljne izvore. Sama veza je najčešće ostvarena oklopljenim ili neoklopljenim kablom sa upredenim provodnicima (*Shielded Twisted Pair* ili *Un-shielded Twisted Pair*).

Najčešče korišćena veza u praksi je ona koja je opisana standardom ISO 11898, te će ona na dalje biti podrobnije proučavana. Pouzdani i poznati transiveri izvedeni po ovom standardu su 82C250 i 82C251, proizvodeni od strane Philipsa, najvećeg proizođača CAN transivera na tržištu.

Karakteristike magistrale

Maksimalna brzina prenosa i maksimalno rastojanje između najudaljenijih čvorova su međusobno uslovljeni arbitracijom na magistrali. Ovo ograničenje je usled potrebe da signal koji putuje brzinom svetlosti pređe rastojanje između najdaljenijih čvorova pre postavljanja narednog bita na magistralu. U praksi se pokazalo da proizvod maksimalne brzine prenosa izražene u Mbit/s i maksimalnog rastojanja između najudaljenijih čvorova izraženog u metrima manji od 60, što je pokazano na slici 7. Primer neregularne situacije koja može da nastupi ukoliko ovaj iskustveni uslov nije ispunjen je slučaj arbitracije između dva najudaljenija čvora, kada dominantan bit mora stići do čvora koji je poslao recesivan bit kako bi se isti povukao.

Maksimalna brzina prenosa koju je moguće postići specificiana od strane CAN standarde iznosi 1Mbit/s, što je prikazano na slici 14.



Slika 14: *zavisnost maksimalnog rastojanja dva najudaljenija čvora od maksimalne brzine prenosa*

Navedeni standard takođe nalaže da karakteristična imedansa voda bude 120Ω gledano sa obe strane, mada u praktičnim realizacijama dozvoljeni opseg je u granicama između 108Ω i 132Ω. Potreba za ovim je obezbeđivanje odgovarajućih DC naponskih nivoa signala i uklanjanje refleksije signala na krajevima voda.

ISO 11898 definiše sledeće kvarove na magistrali:

- CAN_H prekinut;
- CAN_L prekinut;
- CAN_H na naponskom nivou napajanja;
- CAN_L na naponskom nivou uzemljenja;
- CAN_H na naponskom nivou uzemljenja;
- CAN_L na naponskom nivou napajanja;
- CAN_L kratkospojeno na CAN_H;
- CAN_H i CAN_L prekinuto na istom mestu;
- prekinuta veza na krajevima mreže;

Konektrori

CAN protokol ne definiše standard za konektore, tako da najčešće svaki viši CAN protokol ima svog favorita. U praksi se često sreću sledeća tri konektora:

- 9-pinsiki D konektor, predložen od strane grupacije CiA [4];
- 5-pinski Mini-C ili Mikro-C, koji koriste DeviceNet;
- 6-pinski *Deutch* konektor, predložen od strane CANHUG za mobilnu hidrauliku;

U nastavku slede slike i *pin-out* pomenutih konektora.

1	-	Reserved
2	CAN_L	CAN_L bus line (dominant low)
3	CAN_GND	CAN Ground
4	-	Reserved
5	(CAN_SHLD)	**Optional** CAN shield
6	(GND)	**Optional** CAN ground
7	CAN_H	CAN_H bus line (dominant high)
8	-	Reserved (error line)
9	CAN_V+	**Optional** power

Tabela 3: *pin-out D konektora*

Pin	Function	DeviceNet Color
1	Drain	Bare
2	V+	Red
3	V-	Black
4	CAN_H	White
5	CAN_L	Blue

Tabela 4: *pin-out mini-C konektora*

Pin	Function	Recommended cable colour
1	Power negative	Black
2	CAN_H	White
3	Optional Signal GND	Yellow
4	Optional Initiate	Gray
5	Power positive	Red
6	CAN_L	Blue

Tabela 5: *pin-out Deutch konektora*

Slika 15: *D, mini-C i Deutch konektori, respektivno*

Bit tajming

Osnovni vremenski interval po specifikacijama CAN protokola je minimalni vremenski kvant *(minimal time quantum)* koji je jednak periodu internog oscilatora. Međutim, u praksi se češće koristi tzv. vremenski kvant *(time quantum)* čija je vrednost proizvod minimalnog vremenskog kvanta i programabilnog preskalera.

Trajanje jednog bita poruke je podeljeno u četiri vremenska segmenta, pri čemu njihovo trajanje predstavlja ceo broj vremenskih kvantova:

- segment za sinhronizaciju *(the Synchronization Segment)*, koji je fiksnog trajanja od jednog vremenskog kvanta; ovaj segment omogućava sinhronizaciju različitih čvorova na magistralu; sinhronizovan rad čvora garantuje da će uzlazna ili silazna ivica novog bita ležati unutar ovog segmenta;
- segment za propagaciju *(the Propagation Segment)* je programabilnog trajanja između jednog i osam vremenskih kvantova, i koristi se za kompenzaciju kašnjenja na mreži, što uključuje kašnjenje izlaznog drajvera, vreme propogacije signala na mreži i kašnjenje ulaznog komparatora;
- fazni segment 1 *(the Phase Segment 1)* je programabilnog trajanja između jednog i osam vremenskih kvantova;
- fazni segment 2 *(the Phase Segment 2)*;

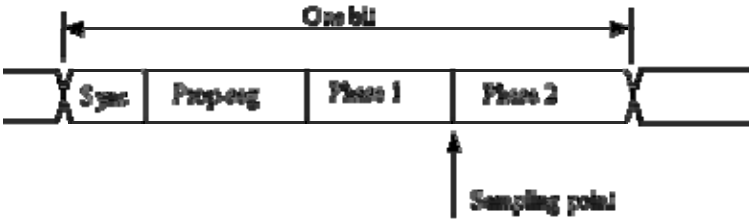
U toku prenosa jedne poruke svaki čvor ostvaruje glavnu sinhronizaciju (*Hard Synchronization*) i resinhronizaciju (*Resynchronization*). Glavna sinhronizacija se ostvaruje na početku svake poruke pomoću tranzicije recesivnog bita IFS polja u dominantni SOF bit poruke. Taj trenutak postaje početak prvog bita poruke. Na osnovu poznate vrednosti brzine prenosa, svaki čvor je u stanju da proračuna trenutke kada očekuje početke narednih bitova poruke koji treba da se nalaze unutar segmenta za sinhronizaciju.

Resinhronizacija se dešava kada uzlazna ili silazna ivica novog bita nije detektovana unutar segmenta za sinhronizaciju. Ako se to desi, svaki čvor meri faznu grešku (*Phase Error*) koja predstavlja vremenski period od kraja segmenta za sinhronizaciju do pojave novog bita. Vrednost fazne greška izražena u jedinicama vremenskog kvanta definiše za koliko će biti produženi ili skraćeni fazni segmenti 1 ili 2, a time i produženo trajanje tog bita, kako bi se početak narednog bita locirao unutar narednog segmenta za sinhronizaciju.

Za vrednost bita merodavan je naponski nivo magistrale u trenutku odabiranja, koji je uvek lociran između faznih segmenata 1 i 2, i prikazan na slici 16. U procesu

resinhronizacije trenutak odabiranja se automatski pomera za vremenski interval koji odgovara faznoj grešci. Dizajner CAN mreže ima na raspolaganju parametar SJW (*Synchronization Jump Width*), kojim je moguće definisati maksimalnu korekciju trajanja bita.

CAN protokol specificira vreme za procesiranje informacije o novom bitu (*Information Processing Time*), koje je locirano na početku faznog segmenta 2, to jest neposredno nakon trenutka odabiranja. On traje najviše dva vremenska kvanta. Vreme trajanja faznog segmenta 2 predstavlja maksimum između vremena trajanja faznog segmenta 1 i vremena za procesiranje informacije o novom bitu.



Slika 16: *bit tajming*

Neki CAN kontoleri, poput Intelovog 82527, daju mogućnost tri odabiranja tokom jednog bita. U tom slučaju dodatna dva odabiranja se ostvaruju na početku vremenskog kvanta koji prethodi centralnom trenutku odabiranja i na kraju vremenskog kvanta koji sledi nakon njega. Rezultat odabiranja je naponski nivo koji je više puta prepoznat.

CAN kontroleri takođe ostavljaju mogućnost podešavanja trajanja bita, to jest brzine prenosa. To se ostvaruje pomoću četiri parametra: već pomenutog preskalera, trajanja faznih segmenata 1 i 2 i parametra SJW izraženih u jedinicama vremenskih kvantova. Mogućnost podešavanja ovih parametra je ostvarena preko dva specijalizovana registra.

IV.a.iii. Upravljanje greškama

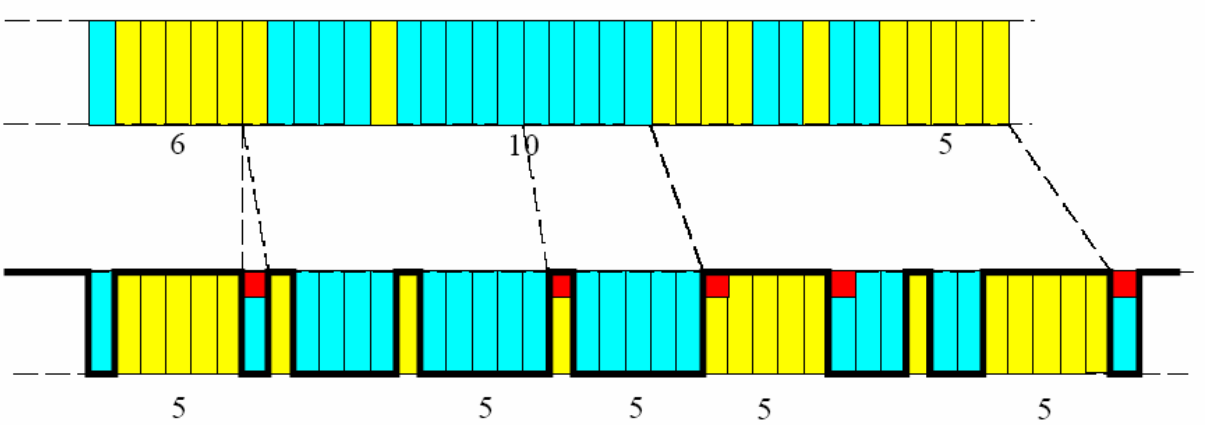
Jedna od značajnih karakteristika CAN protokola je veoma efikasno upravljanje greškama koje ima za cilj da primeti grešku u poruci koja se trenutno nalazi na magistrali, kako bi čvor koji šalje poruku ponovio slanje. Svaki CAN kontroler na magistrali ima mogućnost da detektuje grešku. Kada se ona pojavi, prvi čvor koji je detektovao odmah šalje fleg greške remeteći trenutnu aktivnost na magistrali. Ako ne ranije, ostali čvorovi uočavajući ovaj fleg, konstatuju grešku u poruci i reaguju tako što najpre pošalju svoj fleg greške, a potom i poništе poruku koju su do tada primali.

CAN je implementirao mogućnost za detektovanje pet tipova grešaka, od kojih prve tri na nivou poruke, a preostale dve na nivou bita:

- CRC greška *(Cyclic Redundancy Check Error)* nastaje kada se ne poklapaju CRC vrednost poslata od strane čvora koji šalje poruku i bilo kog od čvorova koji je primio istu;
- ACK grešku *(Acknowledgement Check)* proverava čvor koji je poslao poruku; naime, predajnik šalje recesivni bit u ACK polje, dok svi čvorovi koji su primili

poruku, bez obzira da li su zainteresovani za njen sadržaj ili ne, istovremeno šalju dominantan bit; ukoliko predajnik ne primeti dominantni bit na magistrali alarmira ACK grešku;

- greška u formi poruke *(Frame Check)* se javlja kada se uoče nepravilnosti u okviru delova poruke koja su tačno definisana kada se javljaju i kako izgledaju, poput IFC -a, polja potvrde, CRC *Delimitera* i kraja poruke;
- greška u poslatom bitu *(Bit Monitoring)* se javlja kada čvor koji šalje poruku detektuje recesivni bit kada je poslao dominantni, i obratno; predajni čvor neće alarmirati grešku ovog tipa u dva slučaja: kada uoči dominantan bit kada je poslao recesivan u okviru polja potvrde, i u slučaju gubitka arbitracije na magistrali u toku slanja svog identifikatora;
- stuff error *(Bit Stuffing)* se javlja pri detekciji šest uzastopnih bitova istog naponskog nivoa; naime nakon pet bitova istog naponskog nivoa CAN kontroler koji šalje poruku automatski ubacuje bit suprotnog nivoa koji ostali čvorovi ne koriste kao podatak (videti sliku 17);



Slika 17: *bit stuffing (bitovi sa crvenim krajem su umetnuti)*

Svaki čvor, pored mogućnosti detekcije grešaka, ima i dva brojača grešaka za dva moguća režima rada: slanje i primanje poruka. Brojač grešaka koji radi u režimu slanja poruka inkrementira brže iz razloga što je najverovatnije izvor greške upravo čvor koji šalje poruku. Po pravilima za promenu stanja brojača svaka greška u slanju rezultuje povećanjem brojača za osam, svaka greška u prijemu za jedan, dok pravilno poslate i primljene poruke ga dekrementiraju. U zavistnosti od vrednosti brojača, čvor može biti:

- aktivan po pitanju detekcije greške *(Error Active)*, kada šalje dominantne bitove u okviru flega greške svaki put kada primeti grešku;
- pasivan po pitanju detekcije greške *(Error Passive)*, kada jedan od brojača greške dostigne vrednost od 127; od tog trenutka čvor šalje recesivne bitove u okviru flega greške svaki put kada primeti grešku;
- onemogućen da šalje poruke *(Bus Off)*, kada je vrednost brojača greške za režim slanja poruke veći od 255;

Nakon konfigurisanja, svaki čvor je u aktivnom stanju po pitanju detekcije greške. Ukoliko jedan čvor pokuša da pošalje poruku i nastupi greška, odgovarajući brojač se poveća za osam. Ukoliko šestanest puta zaredom on pogreši u slanju, stanje tog brojača je 127 i čvor postaje pasivan po pitanju detekcije greške. Od tog trenuka, svaki put kada detektuje grešku, ovaj čvor šalje recesivne bitove u okviru flega greške, što znači da neće remetiti aktivnosti na magistrali. Ako isti čvor nastavi sa greškama u slanju i stanje brojača postane 255, čvor biva onemogućen da dalje šalje poruke. Ostali čvorovi su svaki put po lošem slanju pomenutog čvora povećavali za jedan stanje brojača greške za režim primanja poruke. Kako je taj broj prilično manji od 127, nema bojazni da će oni otići u pasivno stanje po pitanju detekcije greške. Svaki naredni tačan prijem će dovesti do dekrementiranja vrednosti ovog brojača.

Zanimljiv je slučaj magistrale na kojoj se nalazi samo jedan čvor koji pokušava da pošalje poruku. On će, naravno, uspeti da je pošalje, ali budući da nema drugih čvorova u sistemu, neće dobiti bit potvrde, što će rezultovati ACK greškom. Tada šalje poruku greške, povećava svoj brojač grešaka pri slanju za osam, i pokušava da ponovo pošalje poruku. Nakon 16 uzastopnih uvećavanja brojača, čvor postaje pasivan po pitanju detekcije greške. Po specijalnom pravilu algoritma za upravljanje grešakama *(error confinement algorithm)*, brojač grešaka se više ne inkrementira, a čvor nastavlja sa daljim pokušajima slanja poruke dokle god ne dobije bit potvrde.

Mnogi CAN kontroleri imaju statusne bitove kojima je moguće očitati stanje upozorenja *(Error Warning)*, koje nastupa kada su vrednosti bilo kog od dva brojača iznad 96, i stanje kada je čvor onemogućen da šalje poruke. Neki kontroleri imaju statusne bitove koji ukazuju i na pasivno stanje čvora po pitanju detekcije greške.

IV.a.iv. Primena CAN baziranih sistema

CAN serijski interfejs, orginalno razvijan za potrebe automobilske industrije, nalazi svoju sve širu primenu i u drugim oblastima, kao što su: industrijska automatizacija, automatizacija stambenih i poslovnih zgrada, medicinska oprema, pomorska elektronika i sl. U svim navedenim slučajevima glavni zahtevi su:

- niska cena
- sposobnost funkcionisanja u različitim okruženjima
- podrška radu u realnom vremenu
- izuzetno visoka pouzdanost
- jednostavna upotreba

Primena CAN u vozilima

Savremeni trendovi u razvoju vozila svih vrsta podrazumevaju primenu sve većeg broja elektronskih kontrolnih sistema. Razvoj automobilske elektronike je delom uzrokovan željom kupaca za što sigurnijom i udobnijom vožnjom, delom potrebom za što ekonomičnijom vožnjom, a delom i sve rigoroznijim propisima po pitanju emisije štetnih gasova u atmosferu. Dakle kontrolni uređaji (inteligentni senzori i aktuatori) koji treba da ispune ove zahteve moraju biti svuda prisutni: u motoru, prenosnom mehanizmu, moraju kontrolisati protok goriva kroz karburator, implementirati ABS (A*nti Blocking System*) i ASC (*Acceleration Skid Control*) sisteme za kontrolu proklizavanja i slično. To

podrazumeva razmenu relativno velike količine podataka, što bi primenom konvencionalnih signalnih linija centralizovanog upravljačkog sistema postalo skoro neizvodljivo zbog složenosti takvog komunikacionog sistema, visoke cene i smanjene pouzdanosti. Pored toga, mnoge od ovih funkcija podrazumevaju sinhronizovanu akciju više kontrolnih uređaja. Na primer, ASC zahteva sadejstvo kontrolnih uređaja motora i karburatora da bi se smanjio moment koji se stvara na pogonskim točkovima.

Primena CAN baziranih mreža je omogućila da se svi kontroleri, senzori i aktuatori povežu na jednu serijsku komunikacionu liniju. Sam protokol poseduje moćne mehanizme za korekciju grešaka pri prenosu. Dijagnostika stanja sistema je znatno pojednostavljena, kao i mogućnost *online* konfiguracije pojedinih kontrolnih uređaja na liniji.

Prema zahtevima i ciljevima koji se postavljaju pred serijsku komunikaciju u vozilima, CAN primene se tipično dele u 3 grupe: mreže za kontrolu motora, prenosnog mehanizma i šasije. Prve dve primene CAN protokola zahtevaju brzinu prenosa podataka koja je tipična za sisteme koji rade u realnom vremenu, a to je od 200 kb/s – 1 Mb/s. Treća primena podrazumeva umrežavanje tzv. šasijske elektronike, koja je zadužena za komfor vozila. Pod tim se podrazumeva kontrola svetla, *air-condition* sistema, centralna brava, podešavanje sedišta i retrovizora i sl. Zahtevana brzina prenosa podataka za ove primene je relativno mala, do 50 kb/s.

U bliskoj budućnosti CAN protokol će sigurno naći još jedno polje primene. Radi se o primeni ove komunikacije u cilju povezivanja uređaja u vozilu, kao što su: radio uređaj, telefon, navigaciona pomoćna sredstva i slično za centralni konrolni panel u vozilu. Napredak u trenutno aktivnom projektu Prometej koji se bavi razvojem komunikacije vozilo – vozilo i vozlo – okruženje u mnogome zavisi od razvoja serijske komunikacione opreme.

Primena CAN u industriji

Pokretna i stacionarna poljoprivredna mehanizacija, nautička oprema i mašine, medicinska oprema, tekstilna industrija, liftovski sistemi i oprema specijane namene su samo neke od oblasti gde se danas može sresti serijska komunikacija bazirana na CAN protokolu. Tekstilna industrija je jedna od prvih primena ovog protokola [7]. Proizvođači medicinske opreme su se odlučili za ovaj protokol jer ispunjava njihove veoma stroge bezbedonosne zahteve po pitanju pouzdanosti prenosa podataka. Slični problemi su rešeni i u transportnim i robotskim sistemima. Takođe, mnoge industrijske grane prihvataju CAN pre svega zbog njegove niske cene po instaliranom čvoru.

IV.b. Viši CAN protokoli

Prvi pokušaji praktične primene osnovnog CAN protokola u implementaciji čak i za realizaciji jednostavnih distribuiranih sistema pokazali su se dosta mukotrpni. Neki od problema na koje se u praksi nailazi će na dalje biti opisani:

- Osnovni CAN protokol standardizovan preko ISO11898 standarda obezbeđuje servis za prenos maksimalno 8 bajtova; to predstavlja prepreku u aplikacijama koje zahtevaju prenos mnogo većih blokova podataka;

- Industrijska automatizacija i *embedded* sistemi podrazumevaju postojanje dva osnovna komunikaciona moda: ranije opisani *broadcasting* ili "*producer – consumer*" mod komunikacije i drugi, "*client – server*" mod komunikacje za prenos konfiguracionih podataka i upravljanje mrežom; osnovni CAN protokol pruža samo podlogu ali ne i potpunu implementaciju ova dva moda rada;
- CAN ne specificira poceduru pri startovanju mreže, niti obezbeđuje jednostavan mehanizam za nadgledanje čvorova u toku rada programa;
- Bilo je neophodno posedovati određene rutine za servis i dijagnostiku čvora kod koga nastupi otkaz;
- Određene primene zahtevaju metode sa sihronizaciju aktivnosti na mreži, kao i mnogo veći nivo determinističkog ponašanja mreže na kojoj je implementiran osnovni CAN protokol.
- Javila se potreba standardizacije opisa uređaja (čvorova) na mreži u pogledu njihove funkcionalnosti, parametara i podataka;
- Bilo je potrebno standardizovati način kodovanja/dekodovanja podataka koji se dobijaju ili prosleđuju nadređenoj aplikaciji.

Sve su ovo razlozi zbog kojih se javila potreba za razvojem tzv. viših CAN protokola koji bi uklonili ako ne sve, a ono bar unapred određenu grupu prethodno opisanih nedostataka. Prve više CAN protokole su razvile određene firme za svoje potrebe. Ne dugo zatim, uviđajući prednosti standardizovanih protokola, prešlo se na njhov razvoj.

Najpre se javlja SAE J1939 protokol prevashodno namenjen primeni u kamionima i autobusima. On predstavlja niz preporuka za realizaciju serijske upravljačke i komunikacione mreže u ovim vozilima. Baziran je na proširenom CAN protokolu, i specificira format podataka, transportni protokol, mrežni menadžment i aplikacijski sloj OSI referentnog modela.

Nakon njega sledi mnogo drugih protokola od kojih samo izvestan broj opstaje i nalazi širu primenu u praksi. Neki od tih protokola su namenjeni specijalizovanoj primeni u vozilima, dok su neki od njih dominantno orijentisani na primenu u otvorenim distribuiranim industrijskim sistemima. Pomenuti industrijski sistemi, a posebno industrijska automatizacija, postavljaju zahtev za razvojem takvog protokola koji će omogućiti međusobni rad i zamenljvost uređaja različitih proizvođača.

Pored ove podele viših CAN protokola, određeni autori klasifikuju protokole u dve grupe po tome kako uspostavljaju vezu osnovnog CAN protokola prema aplikacijskom programu.

U prvu grupu tzv. *service-oriented* protokola svrstavaju one protokole koji dozvoljavaju aplikacijskom programu direktni pristup funkcijama CAN kontrolera. Takvi protokoli na veoma jednostavan način implementiaju osnovne funkcionalnosti CAN kontrolera, što za posledicu ima da razvoj distribuirane aplikacije veće složenosti nije znatnije olakšan u odnosu na primenu osnovnog CAN protokola. Kako ovde postoji izražena zavisnost između aplikacionog i komunikacionog softvera, ovi protokoli uglavnom imaju usko specijalizovanu oblast primene.

Druga grupa su tzv. *message – oriented* protkoli, kod kojih je funkcionalnost CAN kontrolera u velikoj meri enkapsulirana, tako da aplikacijski softver dominantno barata apstraktnim objektima koji sadrže podatke namenjene za prenos komunikacionom

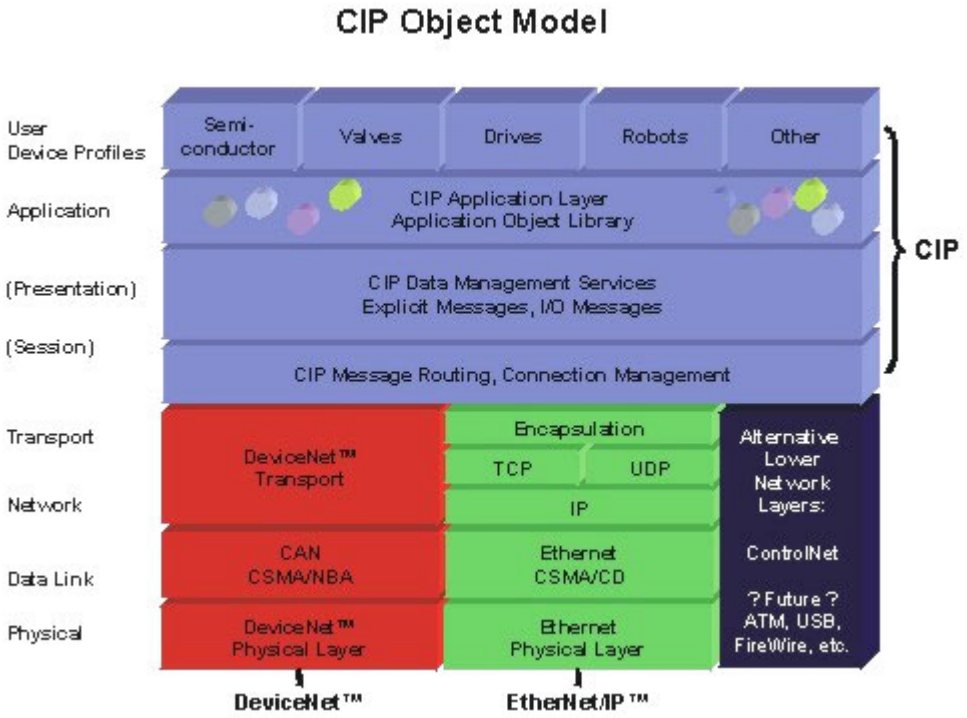
mrežom. Time je znatno olakšan razvoj aplikacijskog programa, jer je on u velikoj meri postao nezavistan od komunikacionog softvera.

Tri najznačjnija viša CAN protokola DeviceNet, *CANopen* i TTCAN će biti detaljno opisani u narednom poglavlju. Prva dva protokola su namenjeni primeni u industrijskim postrojenjima, dok je treći dominantno orijentisan na primenu u vozilima.

iV.b.i. DeviceNet

DeviceNet protokol [20], [4] je razvila firma *Rockwell Automation* za sopstvene potrebe. Kako se ovaj protokol pokazao veoma dobrim u praksi, posebno u oblasti industrijske automatizacije, ubrzo je formirana organizacija njegovih korisnika pod imenom *Open DeviceNet Vendor Association* (ODVA). Ona postaje odgovrna za specificiranje i održavanje *DeviceNet* standarda, kao i za njegovu svetsku promociju. Trenutno aktuelni *DeviceNet* standard nosi oznaku 2.0. To je otvoren protokol, i svaki član ODVA može učestvovati u njegovom daljem razvoju u nekoj od interesnih grupa.

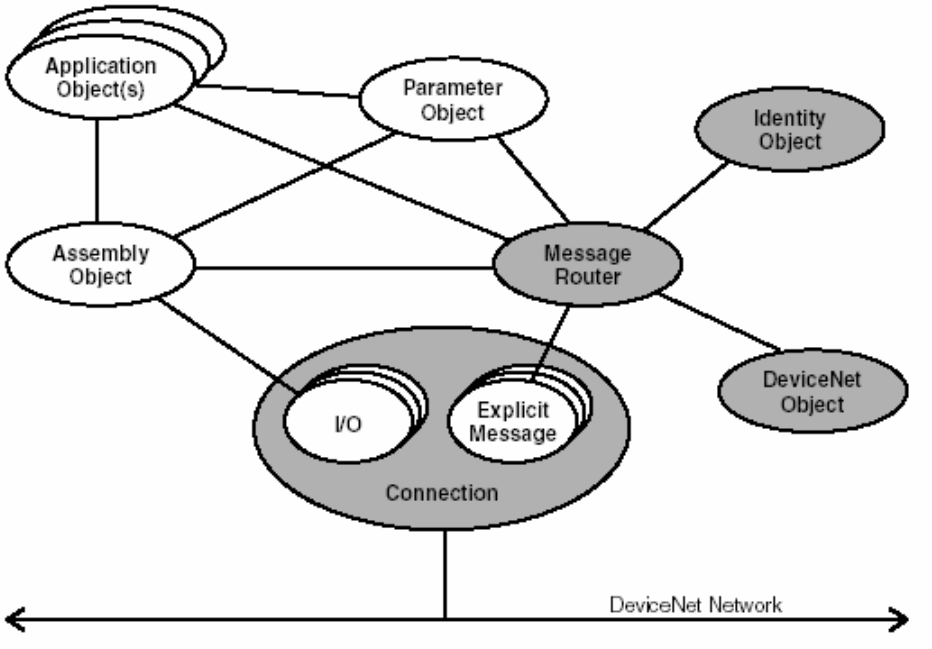
DeviceNet je jedan od tri mrežna standarda (*DeviceNet, ControlNet* i Eternet/IP) koji koristi identičan aplikacijski sloj, tzv. *Common Industrial Protocol* (CIP), što je prikazano na slici 18.



Slika 18: *CIP model*

CIP je strogo objektno orijentisan protokol. U skladu sa CIP protokolom, model jednog *DeviceNet* čvora je prikazan na slici 19. Taj model uključuje nekoliko objekata. Neki od njih su zahtevani od strane *DeviceNet* - a, a neki od njih su definisani od strane

proizvođaža *DeviceNet* opreme u cilju postizanja dodatnih funkcionalnosti. Svaki objekat ima svoje atribute (podatke), servise (komande) i ponašanje prema okruženju (reagovanje na događaje). Glavne karakteristike i namene objekata prikazanih na slici 19 su date u tabeli 6.



Slika 19: *Model DeviceNet objekta*

Objekat	**Funkcija**
Connection	Opis trenutnih konekcija
DeviceNet	Održava konfigurciju i status fizičkg priključka na *DeviceNet*
Message Router	Proseleđuje primljene eksplicitne poruke do ciljnog objekta
Assembly	Grupiše atrubute više objekata u jedan blok podataka tako da se oni mogu preneti preko mreže kao blok podataka
Parameter	Obezbeđuje standardizovane načine za konfigurisanje uređaja i pristup atributima objekata
Identity	Obezbeđuje opšte informacije o identifikaciji uređaja na mreži
Application	Podržava aplikacijski definisane podatke i ponašanje uređaja

Tabela 6: *Opis funkcionalnosti objekata DeviceNet čvor*

Da bi se postigao međusobni rad i zamenljivost uređaja različitih proizvođača povezanih na mrežu sa implementirani CIP protokolom, morao se definisati tzv. profil uređaja (*Device profile*), koji predstavlja minimalni set objekata, konfiguracionih opcija i tipova I/O poruka za konkretni tip uređaja. Treba uočiti da je pojam profila jednog *DeviceNet* uređaja širi od pojma modela jednog *DeviceNet* čvora, tj. uvek je jedan model *DeviceNet* čvora uključen u opis konkretnog *DeviceNet* profila.

Na taj način se postiže da svi uređaji koji slede isti profil imaju iste tipove I/O poruka, iste konfiguracione opcije i odgovaraju na iste komande na identičan način. Neki od mnogobrojnih definisanih profila su: AC pogon, DC pogon, komunikacioni adapter, anlaser, ventil, fotoelektrični senzor, diskretni I/O uređaj opšte namene i sl. Na taj način se za svaki uređaj implementira minimalni set zajedničkih osobina, čime se postiže zajednički rad na istoj mreži uređaja različitih tipova i uređaja proizvedenih od strane različitih proizvođača.

Kako pomenute tri mreže (*DeviceNet, ControlNet* i Eternet/IP) koriste CIP kao zajednički aplikacijski sloj, to dodatno olakšava pisanje aplikativnog softvera, jer programer koji to radi ne mora više čak ni da zna kojom od pomenutih mreža su njegovi uređaji povezani.

Adresiranje objekata unutar *DeviceNet* čvora je bazirano na hijerarhijskoj šemi koja se sastoji od 4 identifikatora: MAC-ID (*Medium Access Control Identifier*), *ClassID*, *InstanceID* i *AttributeID*. Zadatak MAC-ID je da identifikuje konkretni čvor među ostalim čvorovima na liniji. Zadatak *ClassID* je da u datom čvoru identifikuje klasu objekata da bi u njoj *InstanceID* mogao da identifikuje odgovarajući primerak te klase. Konačno, *AttributID* omogućava pristup odgovarajućem atributu identifikovanog primerka.

Fizički sloj DeviceNet protokola

DeviceNet poseduje jedinstvenu karakteristiku da mu je napajanje prisutno na mreži. To omogućava da uređaj adekvatne snage bude napajan direktno sa mreže, čime se smanjuje broj njegovih priključaka, kao i fizičke dimenzije uređaja. Zavisno od primenjenog kabla nominalna struja na linijama za napajanje može da dostigne vrednost i do 8A.

Jedinstvena karakteristika *DeviceNet*-a je i mogućnost da se napajanje može uvesti na proizvoljnom mestu u mreži. To omogućava da se u aplikacijama gde je pouzdanost veoma bitna ostvari redundantni sistem napajanja.

Ovaj protokol podržava brzine prenosa od 125 kb/s, 250 kb/s i 500 kb/s. Brzina od 1Mb/s, koju podržavaju neki drugi protokoli (npr. *CANopen*) nije podržana od strane *DeviceNet* protokola. Razlog leži u činjenici da maksimalna dužina komunikacione linije veoma brzo opada sa porastom brzine prenosa (npr. za brzine prenosa od 125 kb/s ona iznosi 500 m, dok za brzine od 500 kb/s ona iznosi samo 100 m), tako da bi za brzine prenosa od 1 Mb/s ta dužina bila neprihvatljivo mala.

DeviceNet dozvoljava najviše 64 aktivna čvora na mreži, pa je i po tom kriterijumu jedan od najrestriktivnih viših CAN protokola. Takođe, *DeviceNet* postavlja i strože zahteve po pitanju karakteristika linijskih primopredajnika (*bus transceiver*-a), tako

da ako neki primopredajnik zadovoljava specifikacije iz ISO 11898 standarda, ne znači da će uspešno funkcionisati primenjen na *DeviceNet* mreži.

Ovaj protokol podržava i izolovani i neizolovani dizajn komunikacione linije. Izbor konketnog rešenja dominantno zavisi od tipa uređaja. Ako uređaj nije električno povezan sa okruženjem i napaja se preko *DeviceNet* mreže (što je recimo slučaj sa većinom senzora), onda se on tipično ne izoluje od mreže. Sa druge strane, uređaji koji poseduju eksternu električnu vezu skoro isključivo pristupaju mreži preko jednog izolacionog segmenta. Ako se za izolaciju koristi neki optički sistem, mora se voditi računa i o njegovoj brzini, jer se njegovo kašnjenje superponira na kašnjenje linijskog primopredajnika.

Raspodela identifikatora poruka unutar DeviceNet mreže

Metode za raspodelu identifikatora poruka predstavljaju jedan od najznačajnijih segmenata sistema baziranih na CAN protokolu. Od raspodele identifikatora u velikoj meri zavisi efikasnost sistema, sposobnost filtriranja poruka i opterećenost mreže. Posmatrajući ovaj segment viših CAN protokola, može se konstatovati da *DeviceNet* i *CANopen* po ovom pitanju imaju sasvim različit pristup. Naime, izuzimajući određeni broj identifikatora rezervisanih u svrhu upravljanja mrežom, *CANopen* protokol ne koristi nikakvu predefinisanu šemu za dodelu identifikatora. Nasuprot tome, *DeviceNet* protokol koristi predefinisane identifikatore.

DeviceNet mreža postavlja specifične zahteve po pitanju dužine identifikatora poruka. Naime, upotreba 29–bitnog identifikatora ne samo da se ne zahteva, već nije ni dozvoljena. Ovaj protokol definiše podelu 11-bitnog identifikatora u 3 različite grupe. Svakom od maksimalno 64 čvora pripada set identifikatora iz svake od grupa, pri čemu su ovi raspodeljeni ravnomerno na svaki od čvorova. Rezervacija maksimalnog broja identifikatora za maksimalni broj čvorova na mreži implicira da za mrežu sa manje od 64 čvora identifikatora nepostojećih čvorova neće biti dostupni za ostatak sistema. Identifikatori prve grupe se koriste za poruke visokog prioriteta. Svakom čvoru pripada po 16 identifikatora iz ove grupe. Identifikatori treće grupe su namenjeni za poruke niskog prioriteta i u ovom slučaju svakom čvoru pripada po pet. Druga grupa identifikatora je namenjena kao podrška za starije uređaje sa smanjenim mogućnostima filtracije identifikatora. To su tzv. osnovni CAN kontroleri (*Basic* CAN *Type Controllers*) koji imaju mogućnost filtriranja samo osam najznačajnijih bitova identifikatora.

Konfigurisanje uređaja na DeviceNet mreži

Nakon što se novi uređaj poveže na mrežu neophodno je izvršiti njegovu konfiguraciju, pod čime se podrazumeva definisanje njegovih klasa, objekata, objekatskih atributa i slično. Da bi se to postiglo neophodno je da uređaj ima implementiran set funkcija u okviru jednog malog internog programa. U praksi se sreću dva seta tih funkcija: noviji *Unconnected Message Manager* (UCMM) i stariji, *Group 2 Unconnected Port*. Oba imaju zadatak da preko *DeviceNet* mreže uspostave tzv. *Explicit Message Connection* koji ne predstavlja ništa drugo do sposobnost novopriključenog, nekonfigurisanog porta da prihvata poruke sa mreže sa unapred predifinisanim identifikatorima. Preko tog “kanala” od strane nekog drugog čvora na mreži šalju se takozvane eksplicitne poruke. Eksplicitna poruka predstavlja 8-bitnu informaciju neophodnu za konfiguraciju novopriključenog čvora i locirane su u polju podataka

standardne poruke sa podacima CAN protokola. Kao identifikator u CAN poruci koriste se dva specijalno rezervisana identifikatora, tako da ostali čvorovi na mreži neće reagovati na ove poruke.

Po završetku konfigurisanja zatvara se komunikacioni "kanal", nakon čega se konfigurisani čvor ponaša kao standardni *DeviceNet* čvor sa pridruženim setom identifikatora. U toku procesa konfiguracije čvor koji se konfiguriše pošalje 27 i primi 1701 eksplicitnu poruku.

Implementacija master-slave komunikacije na DeviceNet mreži

Iako je osnovna namena *DeviceNet* protokola tzv. *Peer-to-peer* ili *Multi-Master* komunikacija, on implementira i uprošćenu komunikacionu šemu baziranu na *master-slave* relacijama. Ova predefinisana šema se naziva set predefinisanih *master-slave* veza (*Predefined Master-Slave Connection Set*). Ovaj uprošćeni model povezivanja pojednostavljuje prenos I/O poruka koje se i najčešće koriste u upravljačkim aplikacijama. Naime, mnogi senzori i aktuatori su dizajnirani tako da izvršavaju neke predefinisane funkcije u kojima je tip i količina podataka koje uređaj proizvodi i/ili koristi poznat odmah po uključenju. Takvi uređaji poseduju objekte za konekciju na mrežu koji su, zbog jednostavnosti postavljenog zadatka, skoro u potpunosti konfigurisani samim ukuljučenjem uređaja. Po priključenju na mrežu takvog uređaja *master*-u preostaje da jedino interno označi "vlasništvo" nad novouvedeni *slave*-om, nakon čega može odmah da se započne sa prenosom podataka.

Slave može generisati podatke koristeći jedan od više unapred definisanih tipova, kao i koristiti specifične, aplikacijski definisane, tipove podataka. On može primati poruke metodom prozivanja (*polling*). *Slave*-ovi konfigurisani za ovaj mod komunikacije primaju poruke od *master*-a na sekvencijalan način, po redosledu definisanom u internoj listi *master*-a. Učestanost sa kojom se određeni *slave* proziva zavisi od broja čvorova u internoj listi *master*-a, realizovane brzine prenosa *DeviceNet* mreže, veličine poruka koja se predaje pojedinačnom čvoru, ali i od internog vremenskog rasporeda *master*-a. Podaci koje generiše *master* mogu biti *Unicast* ili *Multicast* tipa. Metod *master-slave* komunikacije se odlikuje najvećim stepenom determinističkog ponašanja *DeviceNet* mreže.

Slave može biti konfigurisan da ciklično, na precizno definisane vremenske intervale generiše poruke *master*-u. Ovaj tip komunikacije omogućava da se učestanost prenosa podataka prilagodi zahtevima aplikacije. To opet, zavisno od same aplikacije, može značajno redukovati protok informacije po mreži, i omogućiti bolje iskorišćenje raspoloživog propusnog opsega. Detaljnija analiza ovog metoda razmene poruka je data u poglavlju koje se bavi TTCAN protokolom.

Konačno, *slave* uređaj može biti konfigurisan tako da poruke generiše na promenu nekog od svojih stanja (*Change of State* (COS) *Message*). Za ovaj mod rad vezana su dva dodatna konfiguraciona parametra. Prvi od njih definiše vremenski interval nakon koga *slave* mora da pošalje unapred definisanu poruku (tzv. *Heartbeat Message)* u slučaju da se u toku tog intervala ne desi promena nijednog od analiziranih stanja. Ovim se obezbeđuje da se *maste*r obavesti da je *slave* "živ" i aktivan bez obzira što ne šalje očekivane I/O poruke. Drugi, korisnički podesivi parametar, je tzv. *Production Inhibit Time parametar*, čiji je zadatak da ograniči frekvenciju pojavljivanja COS poruke i tako spreči da posmatrani čvor zaguši komunikacionu liniju. Ova dva

parametra pružaju dizajneru mreže moćno sredstvo da optimalno prilagodi karakteristike aplikacije propusnom opsegu mreže.

IV.b.ii. CANopen

CANopen standard [21], [22] , [4] je rezultat razvojnog projekta finansiranog od strane zemalja EU, i podržan je od strane organizacije CiA (CAN-*in Automation*). Iz tog razloga je ovaj standard daleko rasprostranjeniji u Evropi nego u Severnoj Americi i Aziji, gde se dominantno koristi ranije opisani *DeviceNet* protokol.

CANopen je originalno razvijan za primenu u aplikacijama koje upravljaju kretanjem, tj za najkompleksniju oblast industrijske automatizacije. Tu su zahtevi veoma rigorozni: poruke moraju biti što kraće, sa visokim stepenom iskorišćenosti sadržaja poruke za prenos korisne informacije. Takođe, pouzdanost prenosa uz ispunjenje strogih vremenskih zahteva po pitanju odziva na zahtev za prenosom poruke, jasno su ukazivali da CAN protokol mora biti osnova na kojoj će biti razvijan ovaj viši protokol. Neke od ovih zahteva zadovoljavo je i bazični CAN protokol. Međutim, on je imao i mnogo ograničenja (npr. korisna informacija u komunikaciji između dva uređaja je limitirana na 8 bajtova), što je neizostavno zahtevalo razvoj nadređenog protokola koji koji bi u sebe uključio bazični CAN protokol.

Ciljna grupa za primenu *CANopen* protokola su distribuirani sistemi sa lokalnom inteligencijom, dakle, ona ista oblast primene u kojoj su za sada dominantni *fieldbus* sistemi kao što je recimo *Profibus*. Međutim, prednosti *CANopen* protokola su jednostavnost realizacije, visoka pouzdanost i izuzetno kratko vreme reagovanja. Pored toga, *CANopen* ima veoma kratko vreme oporavka nakon detektovane greške u prenosu. Često se zbog toga kaže će da *CANopen* bazirana mreža pre detektovati grešku i ponovo poslati ispravnu poruku, nego što će TCP/IP bazirana mreža i detektovati da je nastupila greška u prenosu. *CANopen* protokol vrši segmentaciju velikih poruka u osmobajtne pakete i šalje ih, paket po paket, u okviru poruke dužine 111 bitova. To omogućava da poruka većeg prioriteta uvek prekine prenos velike poruke nakon što se prenos jednog paketa završi. Prenos preostalih paketa će biti nastavljen čim mreža ponovo postane slobodna. Nasuprot tome, TCP/IP protokol zahteva integritet i neprekidnost čak 1500 bajtova, što je krajnje nepovoljno za primenu u *real time* aplikacijama.

Još jednu vrlo važnu osobinu *CANopen* protokola (a koju poseduje i *DeviceNet*) je zameljivost i međusobni rad proizvoda različitih proizvođača koji podržavaju *CANopen* protokol. O toj karakteristici ovog protokola će biti više reči kasnije

CANopen je otvoreni sistem, ne samo u smislu da nije vlasništvo ni jednog većeg proizvođača ili grupacije, već i u pogledu njegove fleksibilnosti. Naime, *CANopen* se može konfigurisati da podržava najrazličitije mrežne strukture i topologije. On čak omogućava međusobnu komunikaciju *slave* uređaja. Sve to je moguće zbog njegove jednostavnosti što prouzrokuje i male memorijske zahteve. Na primer, veličina koda za osmobitni mikrokontroler koji piše i čita osmobitne podatke preko *CANopen* mreže je manja od 5k bajtova.

CANopen podržava dva puta više čvorova na mreži od *DevicNet*-a, čak 127, dok je izbor mogućih brzina prenosa podataka daleko veći: 1 Mb/s, 800 kb/s, 500 kb/s, 250 kb/s, 125 kb/s, 50 kb/s, 20 kb/s i 10 kb/s.

Osnovna struktura CANopen protokola

Dva osnovna elementa u strukturi *CANopen* protokola su: komunikacioni profil (*Communication profile*) i profil uređaja (*Device profile*).

Profil uređaja

Koncept profila uređaja je uveden da bi bilo moguće umrežavanje uređaja različitih proizvođača bez potrebe za pisanjem specijalizovanog softvera za svaki pojedinačni uređaj. U tom cilju CiA je prezentovala precizno uputstva koja omogućavaju proizvođačima uređaja da realizuju svoje uređaje u skladu sa predefinisanim profilima. Tek za uređaje koji su u skladu sa predefinisanim profilima, *CANopen* može da garantuje osnovi set mrežnih funkcija. Da bi se omogućilo postizanje dodatnih funkcionalnosti, u okviru profila je određen i jedan opcioni deo u kome proizvođači mogu definisati dodatne karakteristike uređaja, opet na kontrolisan način. Kao pomoć proizvođačima, dostupan je veliki broj standardnih profila za različite tipove uređaja: I/O module, kontrolere, merne uređaje i sl. Ovi profili su implementirani u obliku standardizovane baze podataka koja se naziva rečnik objekata (*object dictionary*). Taj rečnik objekata je u obliku ASCII karaktera zapisan u fajl koji se najčešće susreće pod nazivom EDS (*Electronic Data Shit*). Uz pomoć specijalnog softverskog alata, ova baza se može konfigurisati prema željenom uređaju, nakon čega se dobija tzv. DCF (*Device Configuration File*). Važno je razumeti da ovaj fajl sa rečnikom objekata ne mora biti prisutan na uređaju, ali mora u procesu instalacije tog uređaja na mrežu biti dostupan *master*-u (aplikaciji) koji tu inicijalizaciju vrši, bilo preko *CANopen* mreže, bilo putem Interneta, ili na bilo koji drugi razumljiv način.

CANopen rečnik objekata je tipično podeljen u 4 segmenta:

- segment tipova podatak; u ovom segmentu je definisano koje sve tipove podataka podržava posmatrani uređaj; među njima mogu biti i tipovi podataka definisani od strane proizvođača.
- segment komunikacionog profila; u ovom segmenu se, pored podataka o tome koji se konkretni uređaj koristi, nalaze i neki osnovni podaci o samom uređaju; takođe, ovde je mapirano i koji je tip svake vrste poruke koju konkretni uređaj razmenjuje preko mreže.
- segment profila uređaja; ovaj segment predstavlja opis onih bazičnih osobina uređaja koji obezđuje da svi uređaji istog tipa, a od različitih proizvođača povezani na *CANopen* mrežu pokazuju isto bazično ponašanje.
- segment definisan od strane proizvođača; ovo je opcioni segmen u kome proizvođač može da definiše neke dodatne funkcionalnosti specifične za njegov uređaj, ali na unapred propisan način.

Komunikacioni profil

Komunikacioni profil zahteva da na mreži mora postojati bar dva čvora od kojih u svakom trenutku bar jedan mora biti *master* a drugi *slave*. Za takvu mrežu, *CANopen* protokol definiše nekoliko metoda za prenos i prijem poruka preko CAN linija. Te poruke

za nas predstavljaju tzv. komunikacione objekte. *CANopen* razlikuje sinhroni i asinhroni mehanizam prenosa podataka.

Sinhroni prenos je podržan preko dva tipa poruka: predefinisanih poruka (*Sync Object, Time Stamp Object* i *Emergency Object*), kao i PDO tipa poruka. Ovaj mod omogućava uređaju na mreži da bude strogo sinhronizovan sa taktom *master*-a. To je izuzetno bitno u nekim primenama vezanim za kontrolu kretanja. U takvim aplikacijmama je upravljačka petlja najčešće zatvorena preko *CANopen* linije, što zahteva sinhronizovano uzorkovanje podataka. U ovom modu rada, na raspolaganju je nekoliko parametara koji ga mogu prilagoditi konkretnoj aplikaciji. Na primer, moguće je definisati da se odbirci nekog ulaznog signala uzimaju i proseđuju preko mreže svaki *n*-ti ciklus i sl.

Asinhroni prenos je podržan preko tri tipa poruka: poruka za administriranje na mreži (*Network administration message*), SDO i PDO tipova poruka. Asinhroni prenos omogućava uređaju da automatski obavesti druge uređaje da se neki događaj ostvario, smanjujući na taj način vreme reagovanja. Ovaj mod rada omogućava redukovanje opterećenja mreže, kao i visoke performanse u komunikaciji uz relativno malu brzinu prenosa.

Tipovi komunikacionih objekata (tipovi poruka)

CANopen protokol pravi razliku između četiri osnovna tipa podataka (komunikacionih objekata):

- poruke za administriranje na mreži (*Layer Management* (LM) i *Network Management* (NMT)); ove poruke se koriste u svrhu kontrole uređaja na mreži; pod tim se podrazumeva dinamička distribucija identifikatora pojedinačnim uređajima, kontrolu stanja i komunikacionog moda jednog ili više čvorova na mreži, periodačan *polling* čvorova na mreži ne bi li se detektovao eventualni otkaz nekog od njih i sl.
- poruke sa servisnim podacima (*Service Data Object* (SDO)); ove poruke se šalju asinhrono, i tipično se primenjuju u dva slučaja; prva primena je u procesu konfigurisanja uređaja na *CANopen* mrežu; pod tim se podrazumeva inicijalizacija parametara čvora, kao i učitavanje određenog programa u uređaj; druga primena SDO je za prenos poruka koje su veće od 8 bajtova u formi više CAN telegrama; zajednička osobina za obe primene SDO je da se radi o porukama relativno niskog prioriteta.
- poruke sa podacima procesa (*Process Data Object* (PDO)); PDO se tipično koristi za poruke velike brzine i visokog prioriteta; podaci u PDO poruci su ograničeni na dužinu od 8 bajtova; format ovih poruka, kao i dužina i tip podataka koji se njima prenose se mogu konfigurisati korišćenjem SDO poruka; ovaj tip poruke se može odaslati od bilo kog čvora mreže ka jednom ili više drugih čvorova na mreži; PDO poruke se mogu prenositi i asinhronim i sinhronim mehanizmom; u slučaju asinhronog mehanizma, taj proces može biti iniciran bilo zahtevom od nekog drugog čvora (*remote request*), bilo određenim događajem na konkretnom uređaju.
- predefinisane poruke, kao što su *Sync Object, Time Stamp Object* i *Emergency Object*; *emergency message* je poruka najvišeg prioriteta i rezervisana je da

izvesti o havarijskom stanju na uređaju koji je odašilje; *Sync Object* predstavlja podršku za razvoj aplikacija u realnom vremenu.

CANopen upravljanje mrežom – proces inicijalizacije

Za razliku od *DeviceNet* protokola, *CANopen* protokol ne koristi predefinisanu šemu za raspodelu identifikatora. Po priključenju na mrešu svaki uređaj dobija tzv. inicijalizacioni identifikator poruke koji je najčešće određen stanjem nekih DIP prekidača na samom uređaju. Nakon podizanja mreže, *master* najpre uspostavlja dijalog sa svakim pojedinačnim *slave*-om preko NMT servisa. Kada je veza uspostavljna, *master* najpre svakom od *slave*-ova prosleđuje identifikatore za njihove SDO-PDO poruke. Nakon toga *master* upisuje potrebni kod i inicijalizuje preostale parametre svakom od *slave*-ova. Ove operacije *master* realizuje preko svojih 5 različitih NMT poruka (*Start_Remote_Node, Stop_Remote_Node, Enter_Pre-Operational _State, Reset_Node* i *Reset_Communication*), a sa druge strane *slave* podržava preko svoje mašine stanja (*state machine*) sa 4 predefinisana stanja: Initialisation*, Pre_Operational, Operational* i *Prepared*.

Komparacija DeviceNet i CANopen protokola

Glavna oblast primene oba protokola je industrijska automatizacija. U odnosu na *CANopen* protokol, *DeviceNet* nudi u osnovi slične funkcije, ali u okviru njih postavlja drugačije prioritete. Na primer, upravljanje mrežom kod *DeviceNet* protokola je uočljivo decentralizovano, tako da svaki čvor ima mogućnost praćenja stanja svakog čvora. Nasuprot tome, *CANopen* poseduje centralizovani autoritet, već pominjani NMT servis *master*-a. Komunikacioni mehanizam *CANopen* mreže je prostiji, te su i profili uređaja jednostavniji. *DeviceNet*, međutim, nudi veći nivo zaštite pri korišćenju kroz dodati set funkcija, što zahteva značajno veće resurse čvorova.

IV.b.iii. TTCAN

Prenos svake pojedinačne poruke na komunikacionoj mreži realizovanoj uz pomoć bazičnog CAN protokola je uvek uzrokovan nekim događajem na predajnom ili prijemnom čvoru (*event triggered* - ET). U praktičnim primenama je realno očekivati i situaciju kada više čvorova zahteva prenos svojih poruka u istom vremenskom trenutku, što prouzrokuje znatno opterećenje mreže. Nedestruktivni arbitracioni mehanizam ovog protokola obezbeđuje da se poruke prenose sekvencijalno u skladu sa prioritetom identifikatora pripadajućeg čvora. Ovakve situacije mogu biti kritične za sisteme koji rade u realnom vremenu, jer tipično za svaku od poruka postoji maksimalni vremenski interval unutar kog ona mora biti prosleđena do odgovarajućeg prijemnika. Neispunjavanje tog uslova može da naruši ispravan rad sistema.

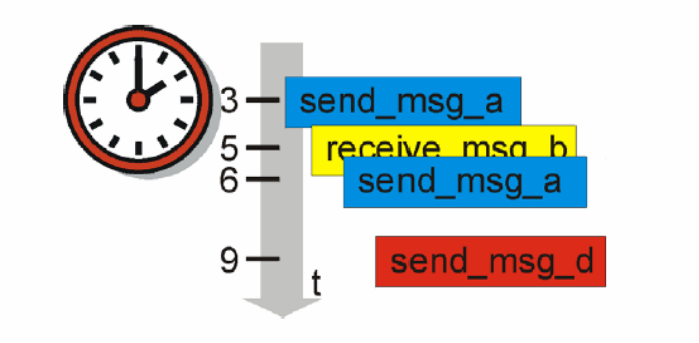
U savremenim vozilima, broj mikrokontrolera, senzora i aktuatora rapidno raste, neprekidno povećavajući protok informacija po mreži. Sve je veći i broj bezbedonosnih sistema u vozilu, čije informacije ni u kom slučaju ne smeju osetiti posledice preopterećenja mreže. Takođe, jedan od najkritičnijih segmenata u budućnosti će biti tzv. xbw (*x – by – wire*) sistemi. To su sistemi koji će u budućnosti samostalno kontrolisati vozilo i njegovu dinamiku, jer se očekuje hidraulično i pneumatsko rasprezanje komande vozača i točkova (danas dostupna prva generacija xbw sistema

još uvek poseduje pneumatski i/ili hidraulični *backup*). Svi pomenuti sistemi zahtevaju veći determinizam u ponašanju CAN komunikacione mreže. TTCAN protokol predstavlja jedno od najboljih do sada prezentovanih i standardizovanih rešenja koje je usmereno ka ispunjenu pomenutih zahteva.

TTCAN (*time – triggered communication on* CAN) [23], [24], [4] je viši CAN protokol koji obezbeđuje sinhronizaciju svih čvorova na mreži, kao i planiranje i realizaciju vremenski determinisanog rasporeda prenosa poruka (*time triggered* - TT). Pored toga, TTCAN podržava i slanje ET poruka. Tako u vozilima tipično imamo TT poruke od sistema za kočenje i ET poruke od sistema za regulaciju temperature. Uvođenje TT poruka je omogućilo implementaciju zatvorenih upravljačkih petlji na sisteme sa CAN baziranim mrežama. Pored automobilske industrije, TTCAN nalazi primenu i u domenu industrijske automatizacije i medicinske opreme, naravno, u daleko manjem obimu.

Osnove TT funkcionalnosti

TT funkcionalnost komunikacionog sistema podrazumeva da se sve aktivnosti nekog čvora koje predstavljaju njegov *interface* prema mreži (slanje i prijem poruka, i sl.) odvijaju po predefinisanom vremenskom rasporedu i sinhronizovano sa zajedničkim (globalnim) vremenom. To se najbolje da ilustrovati slikom 20. Sa slike se vidi da se poruka *a* šalje kada apstraktni časovnik dostigne 3 i 6, dok se, recimo, poruka *c* šalje kada se postigne vremenski trenutak označen sa 5.



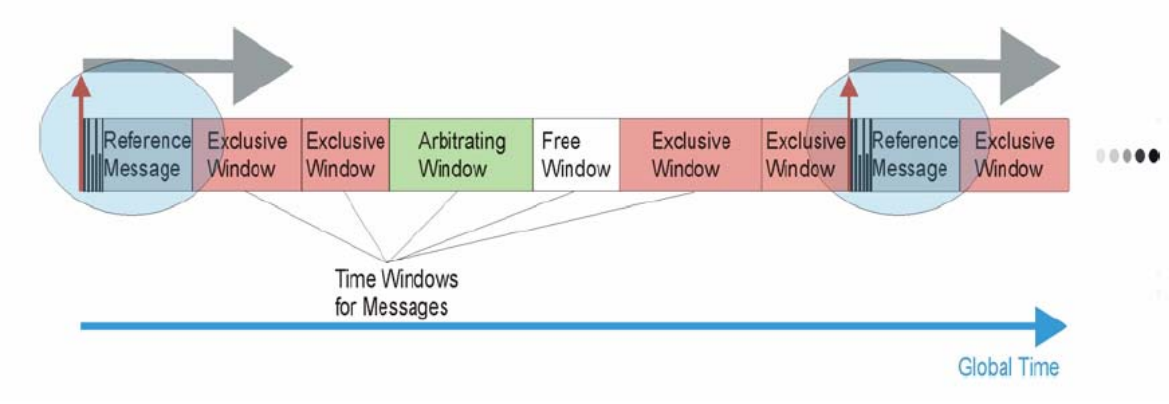
Slika 20: *TT funkcinalnost komunikacionog sistema*

Pojam referentne poruke (reference message) i osnovnog ciklusa (basic cycle)

TTCAN je baziran na vremenskoj aktivaciji i periodičnoj komunikaciji koja je taktovana preko tzv. referentne poruke koju šalje vremenski *master*. Vremenski *master* je čvor na mreži koji ne mora biti ni po čemu drugačiji od drugih, već kao dodatak ima implementiranu rutinu za kreiranje i slanje pomenute poruke. Referentna poruka se prepoznaje po specijalno rezervisanom identifikatoru. Sadržaj polja podataka te poruke je različit u zavisnosti od toga da li se radi o osnovnoj verziji TTCAN protokola (TTCAN – *level 1*) ili o novijoj proširenoj veziji (TTCAN – *level 2*). U osnovnoj verziji samo prvi bajt polja podataka sadrži izvesne kontrolne bitove, dok se preostalih 7 bajtova mogu koristiti za prenos aplikacijski orijentisanih podataka vremenskog *master*-a kao čvora. U

proširenoj verziji TTCAN protokola kontrolni bitovi u polju za podatke zauzimaju donja 4 bajta, jer sada sadrže i informaciju o trenutnoj vrednosti globalnog vremena vremenskog *master*-a. Opet su gornja 4 bajta polja za podatke raspoloživa za prenos korisnih podataka.

Vremenski period između dve uzastopne referentne poruke se naziva osnovni cilkus (jedan osnovni cilkus je prikazan na slici 21). On se sastoji od nekoliko vremenskih prozora različitog trajanja i predstavlja omogućeni vremenski interval za jedan ciklus komunikacija na mreži. Sve poruke koje se prenose imaju strukturu poruke podataka standardnog CAN protokola. Po svojoj nameni vremenski prozori se dele u tri grupe: isključive, arbitracione i slobodne vremenske prozore. Početni trenutak isključivog vremenskog prozora predstavlja trenutak za start poruke konkretno određenog čvora. Arbitracioni vremenski prozor je vremenski interval u kom svi čvorovi mogu slati svoje ET poruke, uz ponovno uspostavljanje arbitracionog mehanizma na komunikacionoj liniji. U tom intervalu se može poslati proizvoljan broj poruka, ali tako da se vremenski interval arbitracionog prozora ne prekorači. Slobodni vremenski prozori su namenjeni za buduća proširenja mreže. Naime, kada se instalira novi čvor na mrežu, ovaj prozor se može preimenovati u bilo isključivi, namenjen tom čvoru, bilo u arbitracioni, čime se proširuje propusni opseg komunikacione linije.



Slika 21: *Isključivi i arbitracioni vremenski prozori*

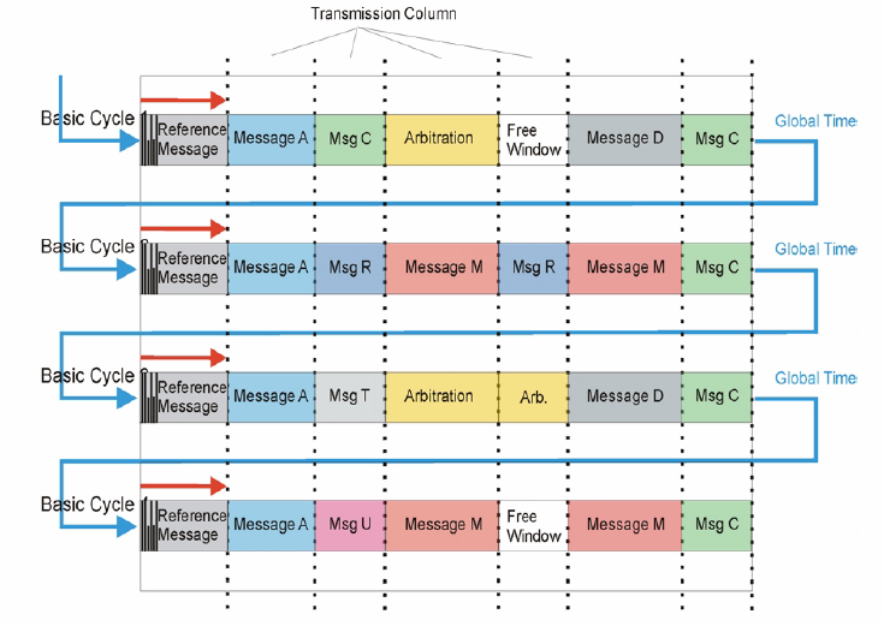
Potrebno je posebno istaći jednu činjenicu koja se često pogrešno interpretira: u vremenskom intervalu trajanja isključivog prozora ni na koji način nije suspendovano dejstvo nedestruktivne arbitraže na komunikacionoj liniji. Naime, najčešće se jedan isključivi vremenski prozor od strane dizajnera mreže dodeljuje jednom čvoru. Međutim, moguće je da se isti isključivi vremenski interval dodeli za dva ili više čvorova. U takvom slučaju, arbitracioni mehanizam određuje koja poruka je višeg prioriteta, i samo će ona da se prenese. Kako će se ista situacija ponavljati u svakom od osnovnih ciklusa, ovakvo dizajnersko rešenje nema previše smisla i retko se sreće u praksi.

Interesantno je napomenuti da automatska retransmisija poruke nije omogućena ni u insključivom ni u arbitracionom vremenskom prozoru. Delimično ublaženje ove restriktivne osobine TTCAN protokola predstavlja činjenica da je u okviru jednog osnovnog ciklusa moguće različite isključive intervale dodeliti istom čvoru.

Mada se na prvi pogled čini da mrežni kontroler u svakom čvoru mora pamtiti mnoštvo informacija o rasporedu komunikacije na mreži, to nije slučaj. Naime, kontroler mora samo da zna kada on sam šalje i prihvata TT poruke, kao i kada započinje, i koliko traje arbitracioni vremenski prozor za slanje simultanih poruka. Čak šta više, poredeći sa konkurentskim TT sistemima, znanje koje TTCAN čvor mora posedovati o rasporedu komunikacije na mreži je minimalno. Tabela sa rasporedom informacija mora biti konfigurisana u svakom čvoru pre startovanja celog sistema.

Sistemske matrice

Praksa je pokazala da konkretne primene zahtevaju mnoštvo različitih upravljačkih petlji zadatog različitog perioda trajanja. Svi oni zahtevaju informacije po različitoj vremenskoj šemi, što se svakako ne može postići uz pomoć jednog TTCAN osnovnog cilkusa. Zbog toga se osnovni ciklusi povezuju tako da formiraju tzv. sistemsku matricu. Ovakve vemenske šeme sada pružaju daleko veću fleksibilnost u definisanju aktivnosti pojedinačnih čvorova. Na primer, neki čvor može slati svoju poruku unutar svakog osnovnog cilkusa, unutar svakog drugog ili recimo, samo jednom u toku cele sistemske matrice. Primer sistemske matrice od 4 osnovna cilkusa je dat na slici 22.

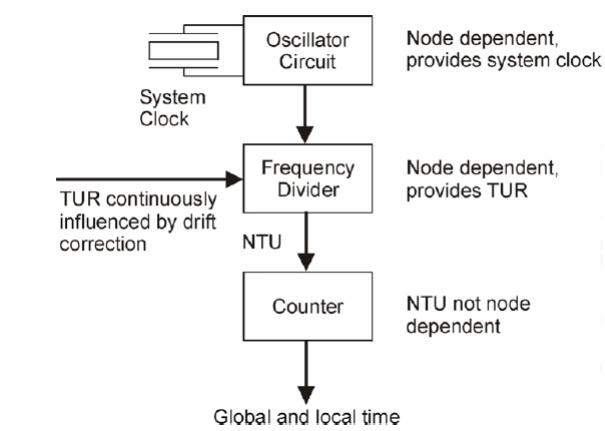


Slika 22: *Primer TTCAN matrice sistema*

Formiranje osnovne jedinice za merenje vremena na TTCAN mreži

Izvršavanje osnovnog ciklusa na mreži je u svakom čvoru kontrolisano preko tzv. ciklusnog vremena (*cycle time*). Ciklusno vreme garantuje TT rad ovog protokola. Brojač koji meri ovo vreme se restartuje nakon svakog završetka osnovnog ciklusa. Rezolucija sa kojom se može iskazati ciklusno vreme je tzv. mrežna vremenska jedinica (*network time unit* – NTU). U osnovnoj verziji TTCAN protokola, NTU je jednak nominalnom vremenskom intervalu trajanja jednog bita i fiksnog je trajanja. U proširenoj verziji

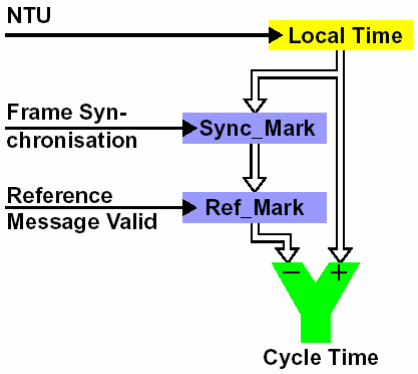
TTCAN potokola, NTU se dobija deljenjem vremenskog intervala trajanja sistemskog takta definisanog učestanošću oscilatornog kola konkertnog čvora sa odgovarajućim koeficijentom (*time unit ratio* – TUR). U tom slučaju se NTU izražava u sekundama. Koeficijent TUR nije fiksan već se neprekidno koriguje u toku rada, a sve u cilju postizanja što bolje usklađenosti između globalnog i ciklusnog vremena. Mehanizam za korekciju vrednosti TUR je relativno složen i njegovo izlaganje bi prevazišlo okvire ovog kratkog pregleda TTCAN protokola. Dovoljno je samo reći da se u tom procesu koristi informacija o vrednosti globalnog vremena koja je u proširenom TTCAN protokolu dostupna kroz polje podataka referentne poruke. Opisani postupak izračunavanja mrežne vremenske jedinice i ciklusnog vremena za slučaj proširenog TTCAN protokola je prikazan na slici 23.



Slika 23: *Generisanje NTU*

Sinhronizacija rada čvorova na mreži

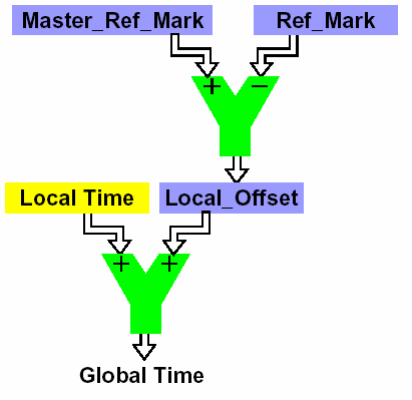
Sinhronizacija svih čvorova, uključujući i vremenski *master*, se obavlja na isti način. Međutim, sam način kako se to izvodi zavisi od toga da li je na konkretnoj meži implementirana osnovna ili proširena verzija TTCAN protokola. Način sinhronizacije u slučaju osnovne verzije TTCAN protokola ilustrovan je na slici 24.



Slika 24: *Sinhronizacija cilkusnog vremena (osnovna verzija TTCAN protokola)*

Kao što se sa slike može videti, SOF bit svake poruke uzrokuje da se podatak o lokalnom vremenu upiše u lokalni registar svakog čvora označen na slici sa *Sync_Mark*. Nakon toga, svaki čvor ispitje da li se radi o referentnoj poruci. Ako je to slučaj, sadržaj *Sync_Mark* registra se prepiše u *Ref_Mark* registar, i automatski oduzme od vrednosti lokalnog vremena formirajući na taj način podatak o ciklusnom vremenu. Dakle u tom trenutku, ciklusno vreme će biti jednako vremenskom intervalu koji je protekao od trenutka detektovanje prvog bita referente poruke, i nastaviće da se menja na dalje u skladu sa lokalnim vremenom sihnronišući tako sve aktivnosti na čvoru. Tačnost sinhronizacije u ovom slučaju je zavisna od brzine prenosa na mreži i tipično je na nivou trajanja jednog bita CAN poruke.

Sinhrovizacija rada čvorova na mreži sa proširenim TTCAN prokolom je bazirna na tzv. globalnom vremenu (*global time*). Postupak proračuna globalnog vremena je ilustrovan na slici 25, i biće u najkrćim crtama opisan u daljem tekstu.



Slika 25: *Računanje globalnog vremena (proširena vrezija TTCAN protokola)*

Vremenski *master* u okviru polja za podatke referentne poruke šalje svoju (po definiciji ispravnu) vrednost globalnog vremena. To je najčešće vrednost vremenskog trenutka kada vremenski *master* odašilje prvi SOF bit svoje poruke. Svaki čvor, kada detektuje da se radi o referentnoj poruci prihata taj podatak i na osnovu njega formira tzv. lokalni *offset*, kao razliku globalnog vremena vremenskog mastera i vrednosti lokalnog vremena kada je prihvaćen prvi bit referentne poruke. Tokom narednog osnovnog ciklusa svaki čvor koriguje vrednost lokalnog vremena lokalnim ofsetom i tako formira svoje globalno vreme. Na ovaj način se nezavisno od brzine prenosa poruka na mreži, može postići tačnost sihronizacije od 1μs što tipično zadovoljava i najstrože zahteve proizvođača vozila. Treba konstatovati da vremeski *master* podatak o globalnom vremenu može dobijati i od nekog eksternog izvora, pa je u tom smislu veoma često koristi GPS sistem.

Napredne karakteristike TTCAN

Ispravno funkcionisanje vremenskog *master*-a je od presudnog značaja za pravilan rad kompletne mreže. Zbog toga TTCAN protokol posebnu pažnju posvećuje

pouzdanosti baš ovog segmenta mreže. Tako je omogućeno dizajneru sistema da postavi čak osmostruku redundantnost vremenskog *master*-a, tj. da proglasi i do 8 čvorova za rezervne vremenske *master*-e na mreži. Svaki vremenski *master* je u stanju da formira i šalje referentnu poruku. Svaka od tih referentnih poruka poseduje različiti prioritet, tj. različiti identifikacioni kod unutar poruke. Nakon aktiviranja kompletne mreže, dovoljna su samo dva osnovna segmenta, da se metodom nedestruktivne arbitracije na mreži definiše koji je vremenski *master* najvišeg prioriteta. Nakon toga on ostaje jedini aktivan, dok vremenski *master*-i nižeg prioriteta nastavljaju sa radom kao i svi drugi čvorovi, tj. sinhronišući se na globalno vreme aktivnog vremenskog *master*-a. TTCAN obezbeđuje i mehanizme za detektovanje otkaza aktivnog vremenskog *master*-a, postupak kojim će rezervni vremenski *master* najvišeg prioriteta preuzeti njegovu ulogu, kao i postupak ponovnog uspostavljanja dominacije vremenskog *master*-a najvećeg prioriteta nakon njegove, eventualne, naknadne aktivacije.

Uočavajući određeni broj aplikacija u kojima se periodična komunikacija na CAN mreži javlja sporadično, projektanti TTCAN protokola su implemenirali jednu dodatnu funkcionalnost. Naime, radi se o mogućnosti da dizajner mreže definiše da se TT rad na mreži definisan sistemskim matricama odvija samo određeni period, da bi nakon toga TTCAN mreža uspostavila rad definisan osnovnim CAN protokolom. Takođe, u unapred definisanom trenutku, mreža bi mogla da ponovo uspostavi komunikaciju preko TTCAN protokola. Ova migracija TTCAN mreže na CAN mrežu i nazad, je moguća iz razloga što TTCAN korisit isti fizički nivo kao osnovni CAN protokol, maksimalne brzine do 1Mb/s, a podržan je od istih liniskih primopredajnika (*bus transceivers*).

Naravno, treba uočiti da neki CAN čvor na TTCAN mreži može da radi samo kao prijemnik, tj. da bi njegova aktivacija u ulozi predajnika narušila komunikacioni raspored i ugrozila funkcionisanje mreže. Zbog toga prelaz sa CAN mreže na TTCAN mrežu treba izvoditi postepeno, najpre zamenjujući jedan po jedan CAN čvor TTCAN čvorom koji bi nastavio da radi pod CAN protokolom, a kada se kompletna zamena završi, moguć je prelaz na TTCAN rad svih čvorova.

Implementacija TTCAN protokola

Nakon što je podnet predlog za standardizaciju TTCAN protokola, nekoliko proizvođača je već realizovalo čipove koji implementiraju ovaj protokol. Paralelno sa tim razvijaju se i softverski alati, koji se delom preuzimaju od osnovnog CAN sistema, a delom razvijaju specijalno za TTCAN primenu.

Firma Bosch je implementirala TTCAN protokol u FPGA kolu, da bi ubrzo proizvela i *stand-alone* kontroler koji podržava i osnovni i prošireni TTCAN protokol, i pin kompatibilan je sa poznatijim CAN kontrolerima: Intelovim 82527 i Simensovim CC170.

Ubrzo je i firma Atmel razvila svoj CAN kontoler koji u potpunosti podržava i TTCAN protokol. Razvojem i proizvodnjom sličnih čipova bave se i Motorola, NEC i drugi.

Između ostalog, prisutni su i pokušaji da se izvrši uključivanje karakteristika TTCAN protokola u trenutno znatno rasprostranjeniji *CANopen* protokol.

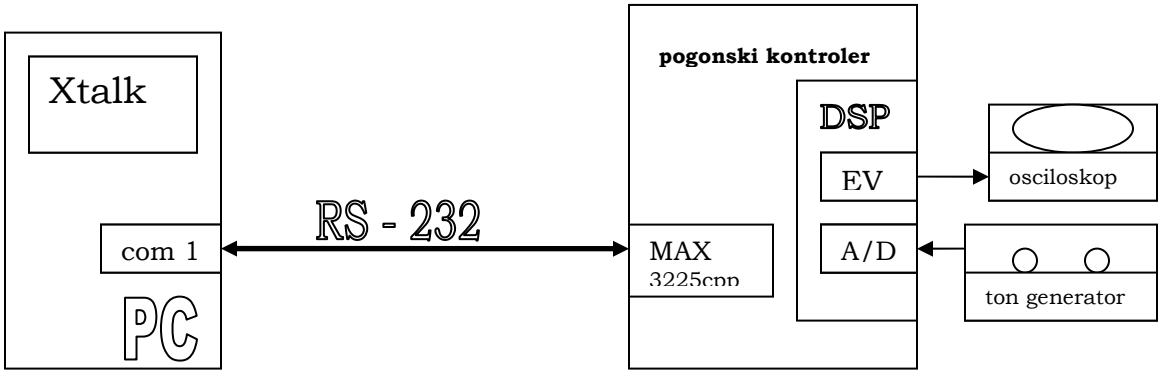
IV.c. Zaključak o CAN-u

Na kraju priče o CAN protokolu možemo sumirati njegove karakteristike na osnovu kojih ga mnogi analitičari svrstavaju među najznačajnije komunikacione protokole za upravljenje sistemima u realnom vremenu novije generacije [7]. CAN je protokol koji je zasnovan na poruci, što znači da svaka poruka stiže do svakog čvora. Svaka poruka ima odgovarajući prioritet. Arbitraža na magistrali je bit-wise, nedestrukivna i bez gubitka vremena. Veoma velika brzina prenosa (do 1Mb/s za dužine do 40m) je i veoma efikasna zahvaljujući moćnom sistemu detetkcije i upravljanja greškama koji ostvaruje svaki čvor ponaosob. Protokol je podržan ISO standardom, a tržišno jeftini CAN kontroleri omogućavaju brzu i ekonomičnu realizaciju komunikacije. CAN je samo osnovi protokol koji ima mogućnost nadogradnje i prilagođavanja određenom tipu problema i okruženja.

V. Praktični deo

Praktična ispitivanja se grubo mogu podeliti u dva dela.

U prvom delu zadatak je realizacija upravljanja asinhronim motorom na principu U/f upravljanja [8]. Eksperimentalni *setup* korišćen u prvom delu je prikazan na slici 26.



Slika 26: *Blok šema veza prvog eksperimenta*

Upravljačka veličina je frekvencija osnovnog harmonika izlaznog napona digitalnog pogonskog kontrolera, koja se zadaje digitalnim i/ili analognim putem u opsegu 0 - 100[Hz]. Digitalno zadata referentna frekvencija se distribuira RS232 komunikacijom od PC-a do DSP-a. Sa strane PC-a se koristi softverski paket XTALK , dok je na strani pogonskog kontrolera serijska komunikacija realizovana pomoću transivera MAX3225cpp i odgovarajuće periferije DSP-a. Karakteristike implementirane RS232 komunikacije su: brzina prenosa 9600b/s, jedan stop bit i parna parnost. Referentna vrednost frekvencija se zadaje slanjem štampajućih karaktera čiji je ASCII kod u željenom opsegu referenetne frekvencije. Upisom bilo kog karaktera čiji je ASCII kod veći od 100, omogućava se i analogno zadavanje reference. Analogno zadavanje reference se ostvaruje ton generatorom koji je priključen na 10-bitni A/D konvertor pogonskog kontrolera. Naponski nivo na ulazu A/D konvertora, koji je iz opsega 0 – 3,3[V], se programski konvertuje u opseg frekvencije od 0 - 100[Hz].

Kako se ovaj rad bavi problematikom digitalne komunikacije, teorijsko objašnjenje U/f upravljanja neće biti detaljnije izlagano [8]. Specifičnosti praktične realizacije ostvarene u okviru ovog eksperimenta su:

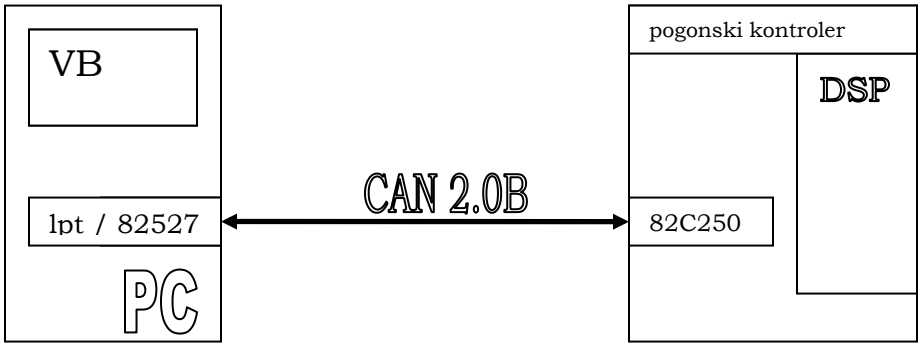
- implementirana je trosegmenta U/f karakteristika;
- upravljački signali za prekidače snage invertorskog mosta se generišu *space-vector* impulsno širinskom modulacijom (*Space Vector Pulse Width Modulation* - SVPWM);
- SVPWM je realizovana korišćenjem tri *compare* registra *Event Menager* periferije DSP-a i odgovarajuće sinusne *look-up* tabele;
- implementirano je mrtvo vreme;
- generisanje PWM signala, serijska komunikacija i A/D konverzija su realizovane u okviru tri prekidne rutine;

Ispravan rad programa je verifikovan:

- posmatranjem na osciloskopu upravljačkih signala za prekidače snage;
- preko eho signala serijske komunikacije;

Usled nedostataka serijskog interfejsa RS232 za upotrebu u industrijskim uslovima nazančenih u prethodnom delu ovog rada, drugi deo ispitivanja je imao za cilj implementaciju CAN serijske veze između PC-a i pogonskog kontrolera.

Za potrebe ovog dela ispitivanja se koristi *setup* prikazan na slici 27.



Slika 27: *Blok šema veza drugog eksperimenta*

Na slici se jasno vidi da postoje dva čvora: jedan čine PC, CAN kontroler 82527 i transiver 82C250, a drugi pogonski kontroler u okviru koga se nalaze isti transiver i signalni procesor TMS320LF2407 sa svojim CAN modulom.

V.a. Zadatak

Kao zadatak implementacije CAN komunikacije između pomenuta dva čvora planirana je realizacija slanja poruke sa podacima i poruke sa zahtevom za podacima od strane PC-a, i odgovarajući odgovor pogonskog kontrolera.

U prvom delu ovog zadatka PC šalje poruku sa podacima. Ovi podaci mogu da se upotrebe kao digitalna referenca frekvencije korišćena u prvom eksperimentu. Kao

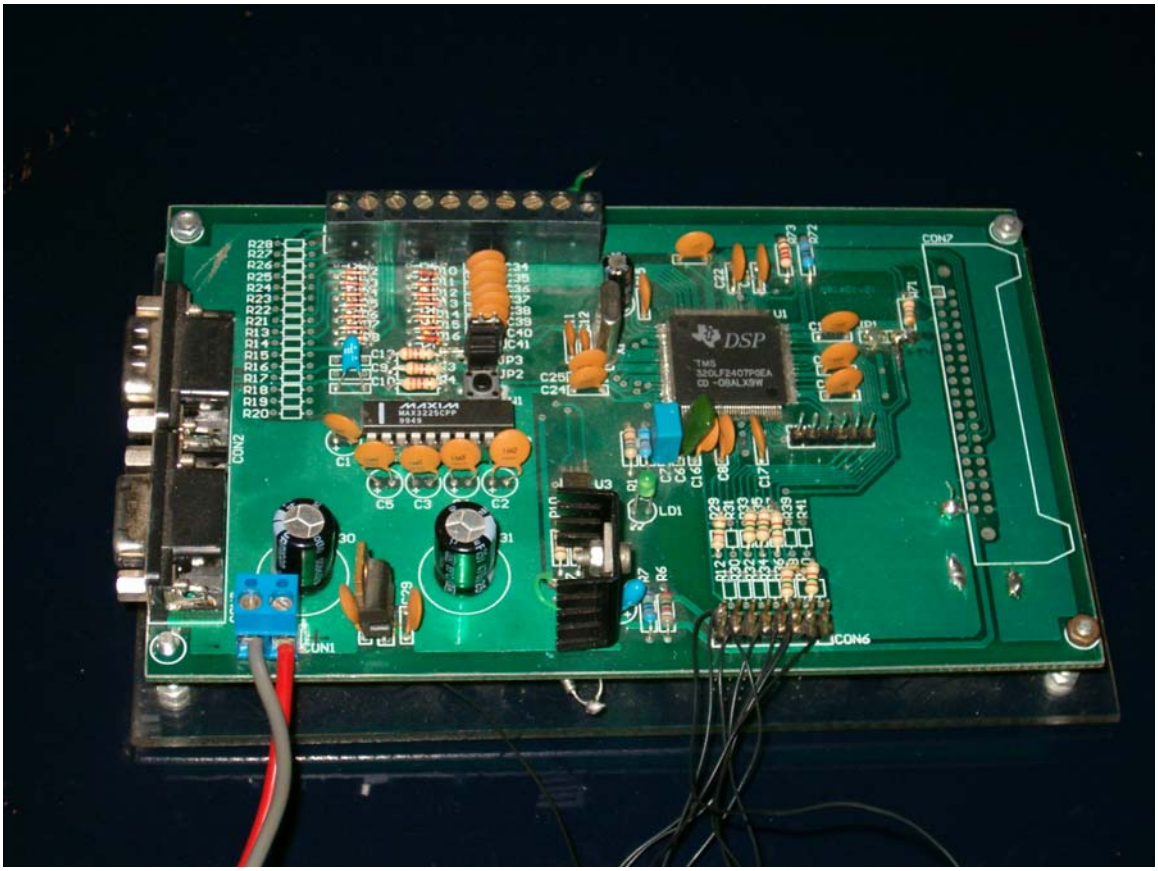
verifikacija pravilnog rada novog programa korišćena je LED montirana na ploči pogonskog kontrolera i aktivirana sa jednog od digitalnih izlaza DSP-a. U slučaju ispravno ostvarene komunikacije ova dioda će se uključiti i isključiti tačno onoliki broj puta kolika je vrednost polja podataka u okviru CAN poruke. Kao odgovor pogonski kontroler šalje CAN poruku sa podatkom o broju uspešno primljenih poruka. U okviru ovog dela je predviđeno korišćenje identifikatora poruke i lokalne maske.

U drugom delu ovog zadatka PC šalje poruku sa zahtevom za podacima. Pogonski kontroler po prijemu ove poruke jednom uključi i isključi LED, čime signalizira da je poruka ispravno primljena, a potom kao odgovor pošalje adrese i sadržaje dve susedne memorijske lokacije u polju podataka jedne CAN poruke.

V.b. Opis aparature

V.b.i. Digitalni pogonski kontroler

Pogonski kontroler je zasnovan na digitalnom signalnom procesoru TMS320LF2407PGE, koji je zadužen za implementaciju algoritama, i komponenetama za komunikaciju Maxim MAX3225cpp i Philips PCA82C250. Razvojna kartica koja povezuje pomenute komponente je urađena u okviru diplomskog rada u Laboratoriji za mikroprocesorsko upravljanje elektromotornim pogonima [9], i prikazana je na slici 28.



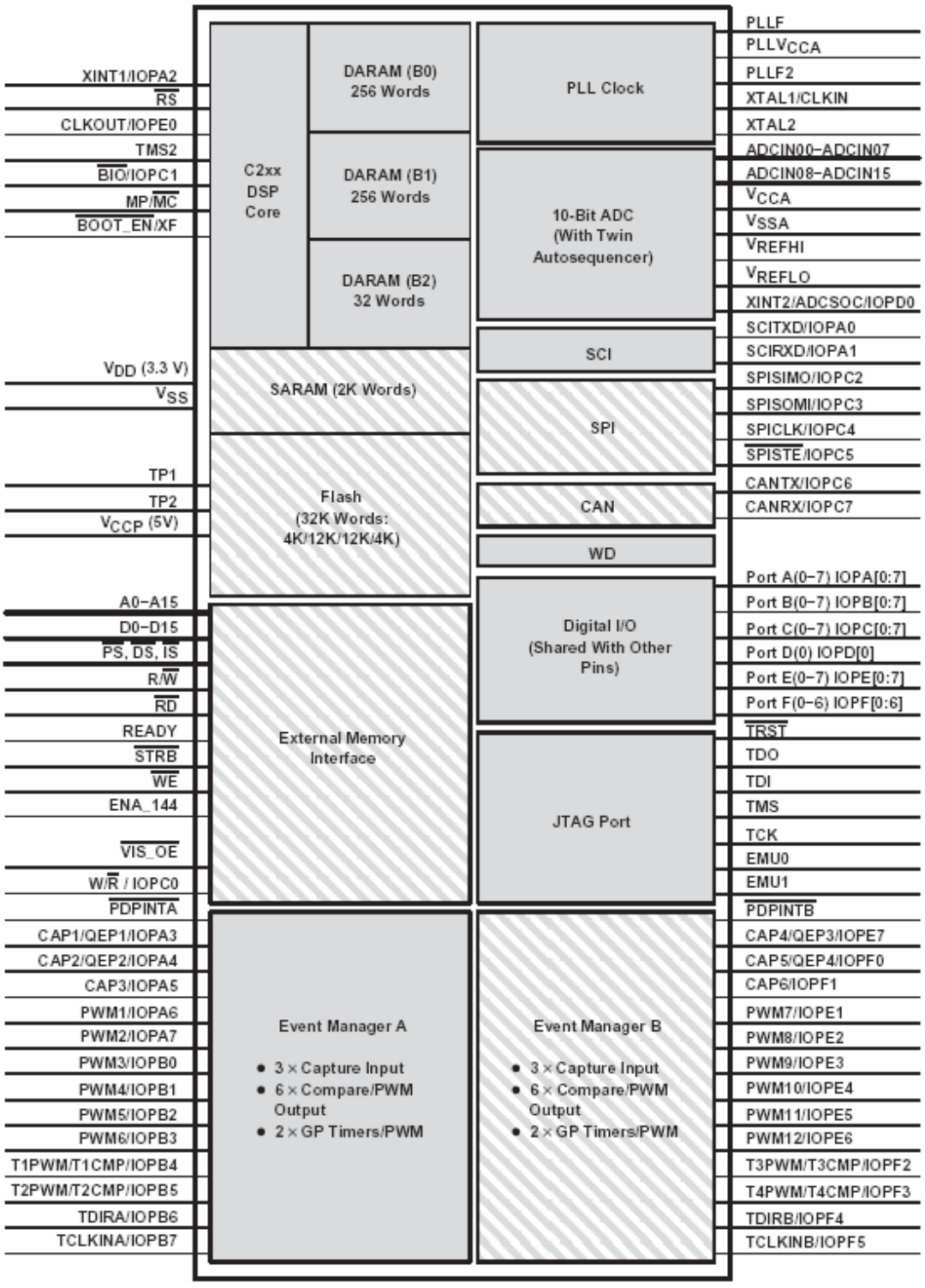
Slika 28: *Digitalni pogonski kontroler*

Digitalni signalni procesor TMS320LF2407PGE

Ovaj DSP pripada familiji procesora 240x namenjen digitalnom upravljanju motorom. Familija je zasnovana na 16-bitnoj centralnoj procesorskoj jedinici (*CPU*) koja radi sa fiksnim zarezom. Nju odlikuje harvardska arhitektura koja omogućuje istovremeno čitanje instrukcije i podataka zahvaljujući odvojenom programskom i magistralom podataka, što u kombinaciji sa četvorostrukim *pipeline*-om ostvaruje izvršavanje većine instrukcija u jednom mašinskom ciklusu. Moguća su tri načina adresiranja: neposredno, direktno i indirektno adresiranje (korišćenjem odgovarajuće aritmetičko-logičke jedinice *ARAU* i pomoćnih registara *AR0-AR7*). CPU sadrži i šesnaestobitni pomerač (*ISCALE*), hardverski množač (*Multiplier*), 32-bitnu centralno aritmetičko-logičku jedinicu (*CALU*), 32-bitni akumulator koji može biti podeljen u dva 16-bitna segmenta (*ACCH* i *ACCL*), te pomerače na izlazu iz akumulatora (*OSCALE*) i množača (*PSCALE*).

Od *on-chip* memorije ovaj signalni procesor sadrži RAM sa dvostrukim (*DARAM*) i konvencionalnim pristupom (*SARAM*), te Boot ROM i *flash*. DARAM je podeljena u tri bloka, od kojih se *B1* i *B2* nalaze u memoriji za podatke, dok je položaj u memoriji memorijskog bloka *B0* konfigurabilan. Ovakva realizacija omogućava tri istovremena pristupa memoriji u bilo kom mašinskom ciklusu što je naročito pogodno za brzo množenje velikih matrica [10]. Postoji 2K 16-bitnih reči SARAM-a koje se mogu mapirati u programsku memoriju, memoriju podataka, ili u obe. Boot ROM predstavlja 256 16-bitnih reči koje su mapirane na samom početku programske memorije. U njemu se nalazi generički punilac koji omogućava inicijalizacione radnje neophodne za brisanje i programiranje *flash* memorije, kao i sam proces njenog brisanja i programiranja. *Flash* je podeljen na četiri sektora veličina 4K/16K/16K/4K koji zasebno mogu biti programirani. Programiranje *flash*-a zahteva naponski nivo od 5V. Prenos podataka od PC-a do DSP-a se može ostvariti sinhronom ili asinhronom serijskom vezom. Programiranje počinje pokretanjem Boot programa koji inicijalizuje komunikaciju. Po usaglašavanju brzine najpre se prenosi programska rutina pod nazivom *Kernel*. Ako je ova aktivnost uspešno obavljena *Kernel* preuzima dalju kontrolu rada aktivirajući sukcesivno tri programske rutine koje obavljaju sledeće radnje: pripremu za brisanje (algoritam *clear*), brisanje (algoritam *erase*) i programiranje *flash*-a (algoritam *program*) [11]. U toku rada *flash* (kao i celokupan DSP) se napaja sa 3,3V, a odgovarajući mod se dobija promenom stanja pina BOOT_EN/XF pomoću odgovarajuđeg džampera [9].

Veliki broj korisnih integrisanih periferija značajno olakšavaju eksterni hardverski dizajn neophodan za ulogu pogonskog kontrolera [12]. Funkcionalna blok šema signalnog procesora je prikazan na slici 29.



Slika 29: *Funkcionalna blok šema DSP-a*

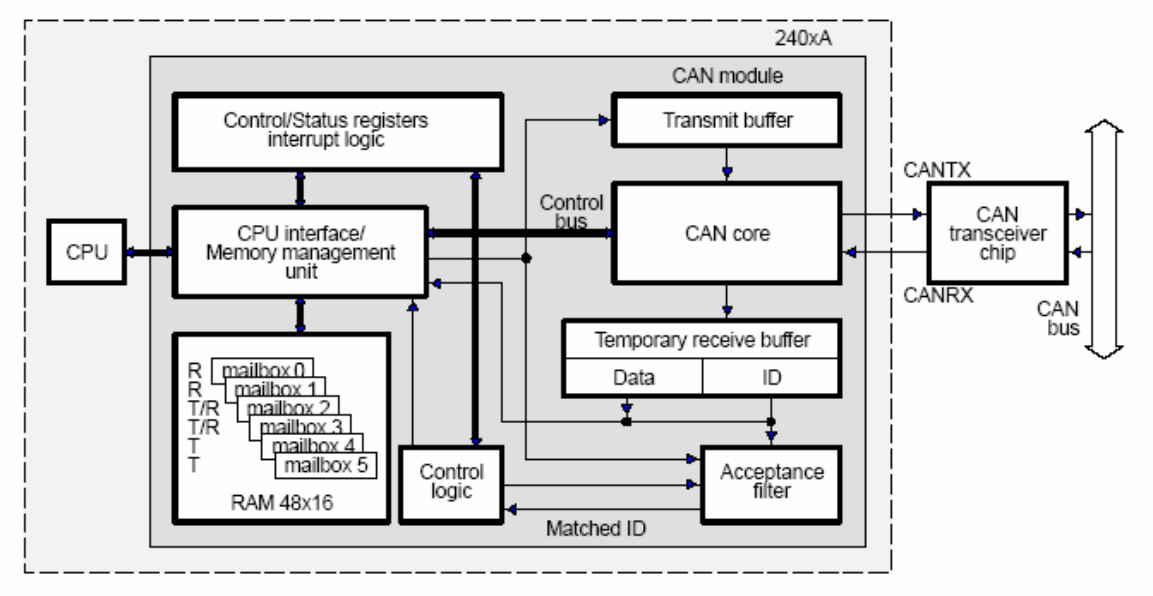
- Moduli menadžera događaja (*EVA* i *EVB*) u sebi sadrže:
  - dva 16-bitna brojača opšte namene koji generišu period odabiranja u upravljačkim sistemima i obezbeđuju vremensku bazu za rad komparatorskih kola i QEP-a;
  - *full compare output* logiku sa šest 16-bitnih kanala za generisanje sinhrono, asinhrono ili *space vector* impulsno-širinski modulisanih signala (*PWM1-PWM6*), sa programabilnim izlaznim polaritetom i mrtvim vremenom neopohodnim za rad dva tranzistora snage energetskog pretvarača;
  - tri *capture input* registra (*CAP1-CAP3*) za registrovanje spoljnih događaja;
  - kola za detekciju impulsa u kvadraturi (*QEP1* i *QEP2*) koje može da se koristi kao interfejs sa enkoderom;
  - mogućnost generisanja SOC signala A/D konvertora nezavisno od CPU-a;
- Interfejs ka spoljnoj memoriji kojim je moguće adresirati do 64K 16-bitnih reči programske, U/I i memorije podataka;
- Modul Watchdog brojača koji nezavisno od CPU-a generiše softverski prekid ako ne dolazi do periodičnog upisa odgovarajućeg ključa;
- A/D konvertor 10-bitne rezolucije sa ugrađenim S/H kolom i minimalnim vremenom konverzije od 500ns, te autosekvencer koji upravlja sa 16 multipleksiranih analognih ulaza;
- CAN modul koji podržava 2.0B standard;
- Asinhrona serijska veza (*SCI*) sa duplo baferisanim *receiver*-om i *transmiter*-om;
- Sinhrona serijska veza (*SPI*) koja je predviđena za komunikaciju sa eksternom periferijom ili drugim procesorom;
- PLL kolo sa mogućnošću množenja takta spoljašnjeg oscilatora do četiri puta;
- 41 bidirekcioni digitalni U/I pin opšte namene;

Određeni broj pinova je multipleksiran kako bi veliki broj podržanih periferija stao na 144-pinskom PGE kućištu.

Periferije imaju mogućnost generisanja velikog broja prekida u zavisnosti od stanja odgovarajućih registara. Svi ovi prekidi su maskirajući i podeljeni na šest globalnih prioritetnih nivoa. Višeg prioriteta od ovih su samo reset i nemaskirajući prekid. Upravljanje periferijskim prekidima ostvaruje odgovarajući interni kontroler (*Peripheral Interrupt Expansion* (PIE) *Controller*). CPU i kontroler prekida ostvaruju vektorski prekidni sistem: na zahtev za prekid od strane PIE, CPU vraća *interupt acknowledge* (*IACK*) signal ukoliko je prekid prihvaćen.

CAN modul digitalnog signalnog procesora

CAN modul, koji je sastavni deo novijih 24x/240x familija, predstavlja CAN kontroler projektovan kao 16-bitna periferija koja podržava 2.0B standard. Blok dijagram koji pokazuje njegovu osnovnu arhitekturu se nalazi na narednoj slici.



Slika 30: *Blok dijagram TMS320x240x CAN modula*

CAN modul ostvaruje dvožičnu komunikaciju sa CAN transceiver - om preko pinova *CANTX i CANRX* sa jedne strane. Sa druge strane CPU ostvaruje pristup kontrolnim i statusnim registrima, kao i specifičnom memorijskom prostoru (tzv. *Mailbox* RAM) CAN modula.

Mailbox-ovi su locirani u delu RAM-a veličine 48 memorijskih reči. U Mailbox RAM-u se nalaze poruke koje su upravo primljene, odnosno upisuju se poruke namenjene za slanje. Mailbox-ovi 0 i 1 su prijemni, dok su 4 i 5 predajni; mailbox-ovi 2 i 3 su konfigurabilni i mogu poslužiti za slanje ili za prijem.

Svaki od šest mailbox-ova sadrži po četiri 16-bitna registra u kojima može da se smesti maksimalno 8 bajtova podataka, dva 16-bitna registra za identifikator i nekoliko kontrolnih registara. U okviru dva registra za identifikator, pored samog 29 - bitnog identifikatora, se nalaze i tri bita kojima se definiše dužina identifikatora, upotreba lokalne maske i *auto-answer* mod (AAM bit) potreban za automatski odgovor na zahtev za podacima.

CAN modul šalje ili prima podatke koristeći tip poruke sa podacima čiji je format prikazan na slici 3. Po prijemu nove poruke se najpre identifikator same poruke upoređuje sa identifikatorima prijemnih *mailbox*-ova, i ukoliko se ovi poklope poruka se prihvata. Postavljanje lokalne maske omogućava da se određeni bitovi identifikatora *mailbox*-a maskiraju i ne učestvuju u upoređivanju sa odgovarajućim bitovima identifikatora pristigle poruke. Naravno, lokalnu masku je moguće postaviti samo za prijemne *mailbox*-ove.

Prijem poruka sa zahtevom za podacima se može ostvariti samo preko *mailbox*-ova 0, 1, 2 i 3. Ukoliko stigne poruka u kojoj je setovan RTR bit, CAN modul upoređuje

identifikatore ovih *mailbox*-ova sa identifikatorom pristigle poruke. Identifikatori mailbox-ova se porede počev od mailbox-a broj 3 naniže. Za slučaj kada se identifikatori poklope dalja pretraga se završava. Zavisno od toga da li je identifikaovani mailbox prijemni ili predajni, kao i da li je setovan AAM bit, moguće su sledeće situacije:

Ukoliko je prozvani *mailbox* konfigurisan za slanje (uočiti da to mogu biti samo konfigurabilni mailbox - ovi 2 i 3), a AAM fleg je setovan, dolazi do automatskog slanja trenutnog sadržaja tog mailbox-a.

Ukoliko je prozvani mailbox konfigurisan kao predajni, a AAM bit nije setovan, prihvaćena poruka sa zahtevom za podacima će biti ignorisana, tj neće biti nikakvog odziva na poruku, kao ni bilo kakve signalizacije ka CPU da je ovakva potuka primljena.

Ukoliko je prozvani *mailbox* konfigurisan kao prijemni, on poruku prihvata i signilizira CPU preko bita RCR iz prijemnog kontrolnog registra. Odgovor na ovaj zahtev za podacima sada u potpunosti zavisi od odluke CPU.

Kada CPU želi da pošalje zahtev za podacima, to se ostvaruje sa konfigurabilnim mailbox-ovima 2 i 3. Naime, jedan od njih se najpre konfiguriše kao prijemni mailbox. Tako konfigurisan mailbox je u stanju da pošalje poruku sa zahtevom za podacima. Prijem očekivanih podataka se ostvaruje u istom mailbox-u.

Dva statusna registra daju informacije o funkcionisanju cele periferije (Global Status Register (GSR)), odnosno o tipu greške koja je nastupila (Error Status Register (ESR)). ESR prikazuje samo prvu grešku koja je nastupila, to jest naredne greške ne menjaju njegov sadržaj. CAN modul ima dva brojača grešaka, po jedan za režim slanja i režim prijema čiji sadržaji su dostupni CPU.

Kontrolni registri CAN modula omogućavaju konfigurisanje mailbox-ova pomoću kojih se oni uključuju ili isključuju, kontrolišu predajne ili prijemne funkcije, određuju brzinu prenosa i upravljaju prekidima. Postoje dva tipa prekidih zahteva između CAN modula i PIE kontrolera: jedan je iniciran promenom stanja nekog mailbox-a, a drugi usled uočene greške. Oba tipa mogu koristiti visoki i niski nivo prioriteta. Sledeći aktivnosti iniciraju prekid:

- poruka je uspešno primljena ili poslata;
- slanje poruke je prekinuto;
- CPU nije uspeo da upiše poruku za slanje;
- wake-up stanje;
- stara poruka je prebrisana od strane nove poruke;
- CAN modul je onemogućen da šalje poruke (*Bus-off* stanje);
- CAN modul je pasivan *(Error Passive)*;
- jedan ili oba brojača grešaka ima vrednost koja je jednaka ili veća od 96;

Maxim MAX3225cpp

Maxim MAX3225cpp predstavlja RS-232 drajver [12] zadužen za asinhronu serijsku vezu sistema sa periferijama. Maksimalna brzina komunikacije je 1 Mb/s. Podizanje signala na RS-232 nivo se ostvaruje takozvanom naponskom pumpom koja je realizovana u vidu četiri kondenzatora reda veličine 0,1μF. Kolo automatski prelazi u

stanje niske potrošnje kada je RS-232 kabl otkačen ili u slučaju da su transmisiona kola zakačene periferije neaktivna, odnosno ukoliko je UART koji pogoni transmitere neaktivan više od 30 sekundi koristeći *AutoShutdown Plus* aplikaciju. Kolo se ponovo aktivira pri novoj validnoj tranziciji na bilo kom *transmiter*-skom ili *receiver*-skom ulazu.

Kolo je smešteno u standardnom DIP 20 kućištu.

Philips PCA82C250

PCA82C250 je interfejs između CAN kontrolera i fizičke magistrale [13]. Primarna upotreba je u industrijskim aplikacijama koje koriste brzine od 40 kb/s do 1 Mb/s. Poseduje zaštite od kratkog spoja ka pozitivnom i negativnom naponu, zaštitu od termalnog preopterećenja koja kontroliše da temperatura spoja ne pređe 165ºC. Kolo zahteva 5V napajanje, i njegovo ulazno kolo za prijem podataka od CAN kontrolera (pin TXD) očekuje za logičku jedinicu napon od 5V, dok njegovo izlazno kolo za slanje podatka ka CAN kontroleru daje napon od 5V za logičku jedinicu. U oba slučaja, logičkoj nuli odgovara naponski nivo od 0V.

PCA82C250 je spakovan u veoma malo osmopinsko SMD kućište sa oznakom SO-8 (*Small Outline package*). Dimenzije su mu 4mm x 5mm.

V.b.ii. PC u ulozi CAN čvora

Kao što je dobro poznato, PC ne poseduje integrisan port koji ima mogućnost direktne CAN komunikacije. To je uzrokovalo da se na tržištu pojave adapterske kartice koje prilagođavaju USB ili paralelni port formirajući CAN čvor. Ove adapterske kartice u osnovi funkcionišu tako što prihvataju signale sa USB ili paralelnog port u formatu koji je definisan karaktristikama tih portova, izdvajaju korisne informacije iz tih poruka, formiraju poruke u skladu sa CAN protokolom i prosleđuju ih na svoje izlaze fizički i logički kompatibilne sa CAN standardom.

Adapterska kartica *FULL-CAN Adapter* [14] koja je korišćena tokom praktičnog rada je proizvedena od strane švajcarske firma ProControl i koristi paralelni port PC-a. Spoljašnji izgled ove kartice je prikazan na slici 31.

Slika 31: CAN adapter

FULL-CAN Adapter koristi Intelov CAN kontroler 82527. Adapter na fizičkom nivou podržava ISO 11898 standard i brzine prenosa do 1Mb/s, budući da je CAN transiver već više puta pomenuti PCA82C250T. Adapter se priključuje na CAN magistralu preko 9-pinskog D konektora čiji je *pin-out* dat u tabeli 3.

Kartica zahteva eksterno napajanje u opsegu 5–24 [Vdc]. Ono se na karticu adaptera može dostaviti dvojako: preko PS/2 konektora koji se nalazi na kraju crnog kabla na prethodnoj slici ili posebnim linijama sa CAN magistrale u skladu sa CAN standardom preko DB-9 konektora.

Uz ovaj adapter stiže i 32-bitni drajver za MS Windows 95/98/NT/2000/XP i aplikativni softver. Drajver je enkapsuliran i tako da je njegov kod nedostupan korisniku. Korisniku je dostupan samo opis (deklaracija) velikog broja struktura i funkcija niskog nivoa. Dat je i opis parametara koje ove funkcije koriste kao argumente i kodovi grešaka koji se potencijalno mogu javiti pri njihovom pozivu. Time je u velikoj meri zaštićen sam proizvod, ali je i korisniku ostavljena mogućnost razvoja sospstvenog softvrskog proizvoda.

Aplikativni softver sadrži definisane C-funkcije visokog nivoa koje omogućavaju otvaranje i zatvaranje čvorova i objekata podataka, očitavanje njihovih statusnih registara, manipulacije prekidima i lokalnim maskama, kao i određene *CANopen* funkcionalnosti [15]. Zajedno sa ovim funkcijama postoji demonstracioni *Visual Basic* kod pod nazivom *HalloCan*. U okviru njega su realizovane sledeće aktivnosti:

- konfigurisan je *LPT*-a i omogućen je njegov interapt prema centralnom procesoru PC-a;
- kreiran novi čvor koji je zatim konfigurisan prema mreži;
- definisano je trajanje ranije opisanog vremenskog kvanta, *SJW*, trajanje segmenata faza 1 i 2, te broj odabiranja u toku jednog bita;
- kreiran je po jedan prijemni i predajni objekat na adapterskoj kartici u okviru kojih su inicijalizovani prošireni identifikatori i definisan broj bajtova podataka koji se prenose u polju za podatke jedne poruke. Pored toga, u memoriji PC-a je rezervisan određeni memorijski prostor koju korisnik programa vidi kao fifo memoriju. Uvođenjem ove memorijske particije se formalizuje razmena podataka između korisnika i adapterske kartice. Naime korisnik u okviru *HelloCan* programa unosi podatke za prenos preko CAN komunikacione linije i oni se smeštaju u tu fifo memoriju. U toku interapta paralelnog porta se podaci iz fifo memorije prepisuju u RAM 82527 kontrolera. Slično funkcioniše i mehanizam za prijem podataka preko CAN magistrale.

Aktiviranjem programa se automatski kreira čvor na CAN magistrali, zatim i prijemni i predajni objekat, i opet automatski započinje slanje i prijem poruka preko CAN magistrale. Slanje poruka podrazumeva ciklično slanje paketa koji u polju podatak prenose vrednosti uzastopnih celih brojeva iz opsega od 0 – 32767. Program omogućava korisniku da očita statusne registre prijemnog i predajnog objekta, kao i celog čvora. Kada adapterska kartica nije priključena na magistralu, program je moguće startovati, ali CAN čvor veoma brzo ulazi u *Error pasive* mod rada što se može detektovati očitavanjem statusa predajnog i prijemnog objekta.

V.c. Realizacija zadataka

U prvom zadatku PC šalje poruku sa podatkom. Pogonski kontroler reaguje na poruku blinkanjem LED-a onoliko puta koliko iznosi celobrojna vrednost prenetog podatka, i vraćanjem podatka čija numerička vrednost predstavlja broj do tada uspešno izvršenih prenosa poruke.

Program napisan u asembleru je realizovan kao beskonačna petlja u kojoj se *polling*-om ispituje *flag* koji signalizira da li postoji poruka u prijemnom *mailbox*-u koja još uvek nije procesirana. Kada poruka stigne ovaj *flag* se softverski (programski) setuje, a resetuje ga program kada započne obradu pristigle poruke. U okviru obrade pristigle poruke, najpre se varijabla koja ″pamti″ broj uspešno izvršenih prenosa inkrementira, a potom se u posebnu memorijsku lokaciju smešta pristigli podatak. U predajni *mailbox* se smešta broj uspešnih prenosa i startuje slanja odgovora.

Nakon prosleđenog odgovora, program naizmenično pali i gasi LED potrebni broj puta čime signalizira da je kompetan ciklus završen. Blinkanje je realizovano pomoću dve ugnežđene petlje, kako bi LED bila dovoljano vremena uključena i isključena da to ljudsko oko može registrovati. Program se zatim vraća na ispitivanje prijemnog *mailbox*-a. Identifikacija ispravnog rada programa je tačan broj blinkanja *LED*-a i korektan podatak vraćen PC-u.

U okviru programa se konfiguriše *mailbox* 0 kao prijemni, a *mailbox* 5 kao predajni. Oba *mailbox*-a koriste prošireni identifikator, dok se prijemnom *mailbox*-u setuje lokalna prihvatna maska. *LED* je priključena na pin 2 porta A. Eksterni takt pogonskog kontrolera iznosi 6MHz, što uz PLL koji je podešen na vrednost četiri omogućava da signalni procesor radi sa taktom 24MHz. Brzina *CAN* prenosa je inicijalizovna na 500kb/s.

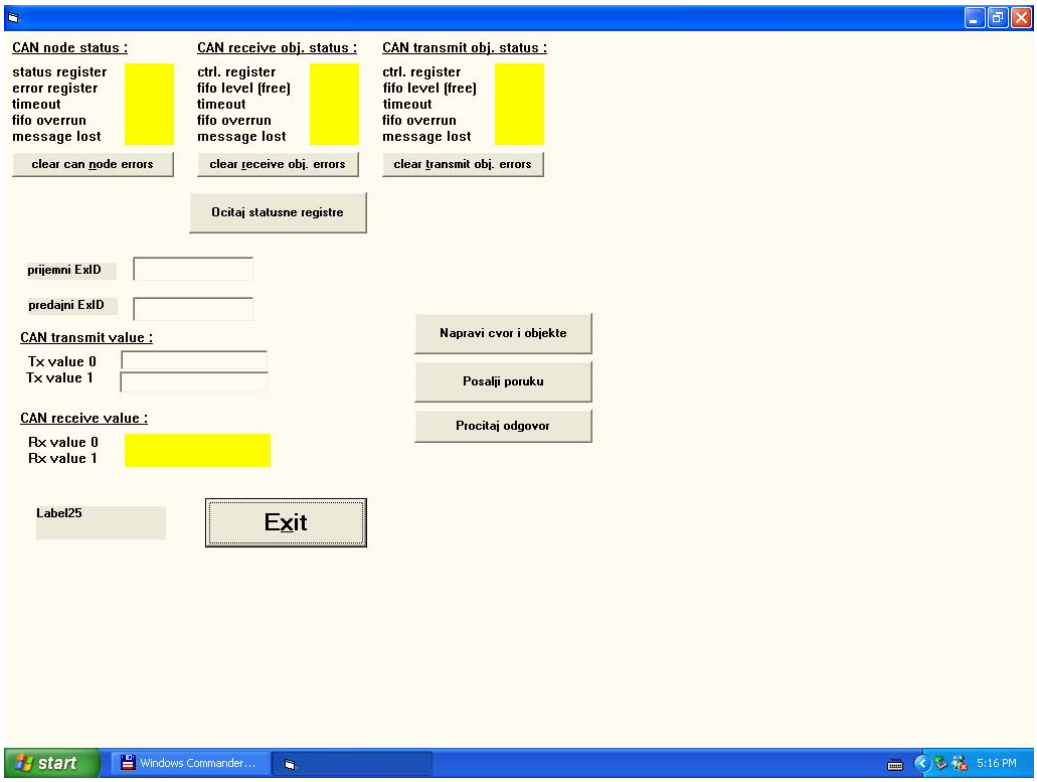
U realizaciji softverske podrške sa strane PC-a se krenulo od već postojećeg demonstracionog VB koda. U okviru procedure koja pravi čvor dodata je aktivnost koja omogućava unos identifikatora objekta koji prima, odnosno objekta koji šalje poruke. U slučaju da identifikator nije unet, čvor se neće napraviti. Pri kreiranju objekta za slanje poruke definisana je 8–bajtna dužina podatka u poruci.

Podaci koji se šalju se upisuju u za to predviđena polja kao 16 heksadecimalnih cifara. U polje TX_value_0 treba uneti donjih 8 heksadecimalnih cifara, a u polje TX_value_1 gornjih 8 heksadecimalnih cifara poruke. Dvostrukim klikom na komandu ″Posalji poruku″ se aktivira programska rutina koja spakuje uneti sadržaj u 8-bajtni niz, koji pozivom C funkcija nižeg nivoa upiše u predajni objekat na adapterskoj kartici i time automatski aktivira slanje CAN poruke.

Sadržaj objekta za prijem poruka je moguće pročitati dvostrukim klikom na komandu ″Pročitaj odgovor″. Rutina koja se tada aktivira opet poziva odgovarajuće C funkcije nižeg nivoa i od njih preuzima sadržaj iz prijemnog objekta koji zatim prikaže na ekranu u za to predviđenim poljima RX_value_0 i RX_value_1. U slučaju da prilikom očitavanja pristiglog sadržaja nastupi neka od grešaka, u pomenutim poljima će biti ispisan kod greške.

Izgled grafičkog interfejsa programa koji na strani PC-a stvaruje opisane funkcionalnosti je dat na slici 32. Opisani hardverski elementi, asemblerski kod upisan u *flash* DSP-a i modifikovan VB softver aktiviran na PC računaru omogućili su da se u ovom ogledu testira osnovni vid komunikacije u kome dva čvora na mreži međusobno razmenjuju podatke. Izmenom identifikatora predajnog objekta čvora koji formiraju PC i adapterska kartica, testiran je mehanizam masikranja određenih bita identifikatora i

dobijen očekivan rezultat: čvor na CAN komunikacionoj liniji će prihvatati samo one poruke koje su u procesu konfiguracije čvora omogućene prihvatnom maskom.



Slika 32: *komunikacioni prozor za prvi zadatak*

U drugom delu zadatka PC šalje poruku sa zahtevom za podatkom, na šta pogonski kontroler odgovara jednim blinkanjem LED-a i slanjem adrese memorijske lokacije X, njenog sadržaja, te adrese memorijske lokacije X+1 i njenog sadržaja. Adresa X se u svakom obraćanju DSP-u sukcesivno inkrementira za 2. U slučaju da je pogonski kontoroler uočio grešku u komunikaciji, on to signalizira tako što opet blinlinkne LED, samo ovaj put osam puta dužim periodom uključenosti.

Sam program je realizovan u asembleru kao beskonačna petlja sa mogućnošću dva prekida na različitim globalnim nivoima prioriteta. Prekid na globalnom nivou jedan nastupa kada pogonski kontroler uoči grešku u prenosu, dok se prekid na globalnom nivou pet dešava tek kada su po prijemu zahteva podaci automatski poslati.

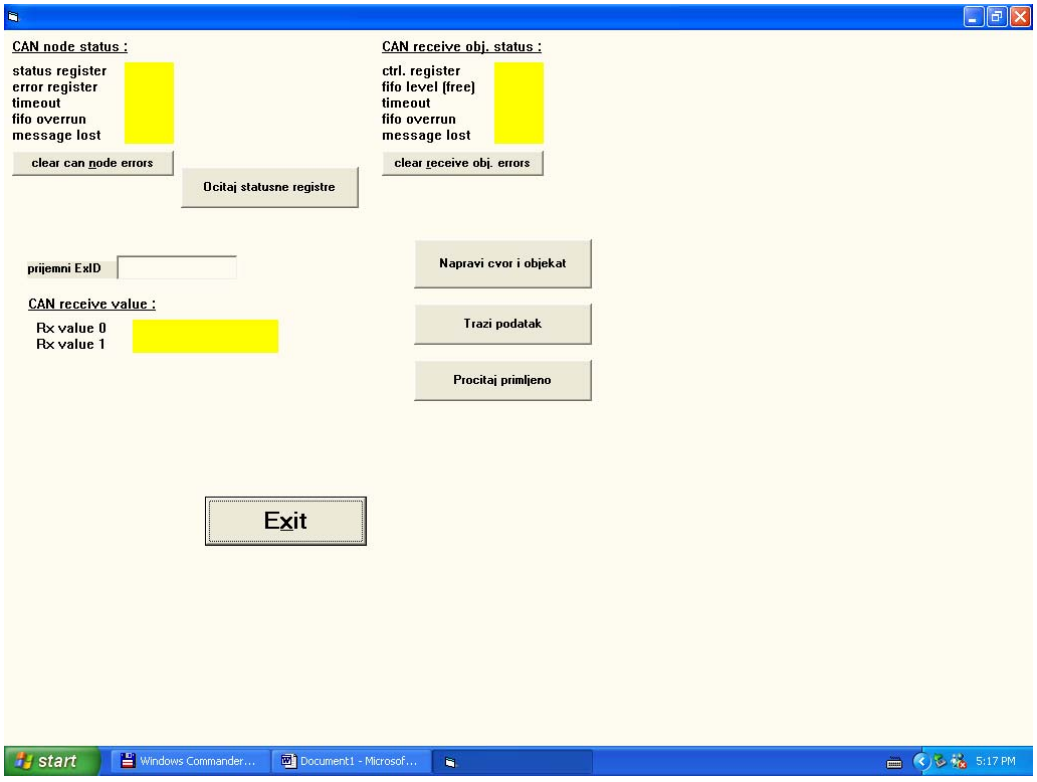
Prijem i prenos podataka je ostvaren preko *mailbox*-a 2 koji je konfigurisan da radi u ranije opisanom a*uto-answer* modu u kome se po prijemu poruke sa zahtevom za podacima sadržaj prozvanog *mailbox*-a automatski pošalje kao odgovor. Priprema novog sadržaja za slanje se ostvaruje u okviru prekidne rutine koja nastupa po prosleđivanju poslednjeg bita odgovora. Kako je funkcionisanje mehanizma maskiranja već testirano u prvom delu zadatka, otuda ovde nije stavljana lokalna prihvatna maska, tj. ona je inicijalizovana na vrednost koja omogućava prihvatanje svih poruka sa mreže.

Ponovo se blinkanjem LED signalizira ispravno izvršena akcija na strani DSP. Način aktivacije LED i pin na koji je ona povezana sa digitalnim portom DSP-a je ostao

isti kao i u prvom delu zadatka. Takođe je nepromenjen i takt signalnog procesora i konfiguracioni parametri CAN prenosa (brzina prenosa, vremenski kvant, broj uzorkovanja po bitu poruke itd.).

Softverska podrška sa strane PC-a je slična onoj opisanoj za prvi zadatak. I u ovom slučaju je primenjena ista procedura koja kreira čvor kao i u prethodnom zadatku. Kao što je već objašnjeno, za opsluživanje poruke sa zahtevom za podacima je dovoljan samo prijemni objekat, pa zato predajni objekat nije ni kreiran. Dvostrukim klikom na komandu ″Trazi podatak″ se šalje poruka sa zahtevom za podacima, dok se primljeni odgovor može prikazati na ekranu dvostrukim klikom na komandu ″Procitaj primljeno″. Primljeni sadržaj se prikazuje na za to predviđenim poljima.

Izgled grafičkog interfejsa programa koji na strani PC-a stvaruje opisane funkcionalnosti je dat na slici 33.



Slika 33: *komunikacioni prozor za drugi zadatak*

VI. Zaključak

VIII. L I T E R A T U R A

[1] Slobodan Vukosavić, Digitalno upravljanje elektromotornim pogonima, Akademska misao, Beograd 2003.

[2] Vujo Drndarević, Personalni računari u sistemima merenja i upravljanja, Akademska misao, Beograd 2003.

[3] www.embedded.com

[4] www.can-cia.de

[5] www.interfacebus.com

[6] www.kvaser.se

[7] Lars-Berno Frederiksson, Controller Area Networks and the protocol CAN for machine control systems, www.kvaser.se

[8] Slobodan Vukosavić, Mikroprocesorsko upravljanje elektromotornim pogonima, skripta sa predavanja

[9] Aleksandar Berić, Digitalni pogonski kontroler zasnovan na TMS320LF2407 signalnom procesoru, diplomski rad, oktobar 2001.

[10] Texas Instruments, TMS320C2x User's Guide, decembar 1990.

[11] Texas Instruments, TMS320LF240x Flash Programming

[12] Texas Instruments, TMS320LF/LC240x DSP Controllers Systems and Peripherals Reference Guide, spru357a.pdf

[13] Philips Semiconductors, “PCA82C250, CAN controller interface”

[14] www.procontrol.ch

[15] ProControl, Can_func32.doc, www.procontrol.ch

[16] http://www.modicon.com/techpubs/

[17] www.lothlorien.net/collections/computer/ethernet.html

[18] www.profibus.com/technology/documentation/index.html

[19] Profibus – technology and application, www.profibus.com/imperia/md/content/pisc/technicaldescription/

[20] www.ixxat.de/english/knowhow/artikel/devicenet

[21] K. Etschberger, C. Schlegel, CANopen-based Distributed Intelligent Automation, IXXAT Automation

[22] Lars-Berno Frederiksson, Controller area networks and the protocol CAN for machine control system, KVASER AB

[23] Florian Hartwich, Bernd Müller, CANNetwork with Time Triggered Communication, Robert Bosch GmbH

[24] Florian Hartwich, Bernd Müller, Time triggered communication on CAN, Robert Bosch GmbH

VIII. Abstract – *na engleskom jeziku.*