Prilagodba i integracija programske podrške za CAN sabirnicu u ERIKA okruženje na Aurix platformi

Dekanić, Stjepan

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:310281

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-15**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTOTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PRILAGODBA I INTEGRACIJA DRIVERA ZA CAN SABIRNICU U LINUX OKRUŽENJU NA AURIX PLATFORMI

Diplomski rad

Stjepan Dekanić

Osijek, 2017.

SADRŽAJ

1. UVOD 1
2. CAN PROTOKOL 2
   2.1. Osnovne karakteristike 2
   2.2. Okvir 5
   2.3. Pristup sabirnici 6
   2.4. Bit popune 7
   2.5. CAN FD 7
   2.6. ISO standardi 9
      2.6.1. ISO 15765-2 9
      2.6.2. ISO 14229 11
3. AURIX PLATFORMA 15
   3.1. Razvojna ploča 15
   3.2. Razvojno okruženje 17
   3.3. ERIKA Enterprise 18
4. CAN PROTOKOL NA AURIX PLATFORMI 19
   4.1. CAN upravljački program 19
   4.2. Implementacija CAN protokola 21
   4.3. Implementacija CAN upravljačkog programa 23
   4.4. Implementacija programske podrške za standarde 24
      4.4.1. Programska podrška za standard ISO 15765-2 24
      4.4.2. Programska podrška za standard ISO 14229 27
5. EKSPERIMENTALNA MJERENJA 35
   5.1. PCAN – USB Pro 35
   5.2. Testiranje funkcionalnosti kontrolne jedinice 36
      5.2.1. Servis 1 - ponovno pokretanje kontrolne jedinice 37
      5.2.2. Servis 2 - pisanje podataka na memorijsku adresu 37
      5.2.3. Servis 3 - čitanje podataka s memorijske adrese 38
      5.2.4. Servis 4 - upravljanje ulazno/izlaznim stanjima prema identifikatoru 39
      5.2.5. Poruka o kontroli toka 41
6. ZAKLJUČAK 45

1. UVOD

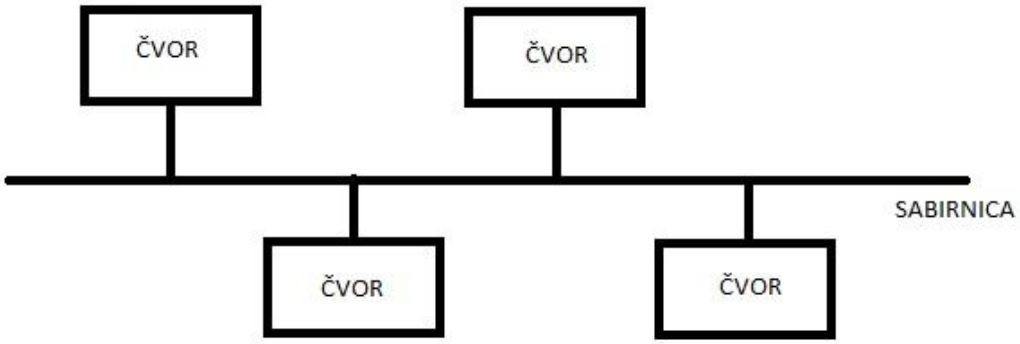
Novi i suvremeni trendovi u autoindustriji zahtijevaju sve veću upotrebu elektroničkih upravljačkih uređaja. Razvoj elektroničkih uređaja namijenjenih autoindustriji uzrokovani su potrebom za ekonomičnom vožnjom i željama kupca za sigurnijom i ugodnijom vožnjom. Zbog povećane količine elektroničkih uređaja povećava se i količina mjernih i upravljačkih podataka. Slanje podataka nekim od standardnih načina je praktički neizvedivo zbog složenosti i visoke cijene, a i pri tome je nesigurno sa stajališta automobilske industrije. Cilj novih protokola u autoindustriji je težio smanjenju količine komunikacijskih linija za povezivanje upravljačkih uređaja sa senzorima i aktuatorima i standardizaciji pravila komunikacije konstruirajući sigurne i pouzdane protokole. Danas se najčešće zbog svoje pouzdanosti i sigurnosti koriste CAN (engl. *Controller Area Network*) i *FlexRay* protokoli. Svaki od navedenih protokola koristi se istovremeno u automobilu za posebne funkcionalnosti. Omogućeno je ispitivanje i dijagnosticiranje sustava mjerenjem osciloskopom ili spajanjem dijagnostičkih uređaja.

Cilj diplomskom rada je opisati CAN protokol. Izvršiti prilagodbu postojećeg CAN upravljačkog programa za AURIX platformu namijenjen spajanju na sabirnicu. Omogućiti programsku podršku za standarde ISO15765-2 i ISO14229, namijenjeno za dijagnosticiranje sustava.

Diplomski rad je strukturiran na sljedeći način. U drugom poglavlju objašnjen je CAN protokol. Navedene su osnovne karakteristike, vrste okvira i objašnjeni standardi. U trećem poglavlju je objašnjen mikroupravljač TC297A i predviđeno razvojno okruženje. U četvrtom dijelu provedena je analiza postojećeg Infineon CAN upravljačkog programa za komunikacijske primopredajnike. Opisana je prilagodba CAN upravljačkog programa za spajanje na CAN sabirnicu i programsku podršku za CAN komunikaciju po standardima ISO15765-2 i ISO14229. U petom dijelu izvršeno je testiranje CAN upravljačkog programa i programske podrške u okviru CAN mreže na Aurix platformi.

2. CAN PROTOKOL

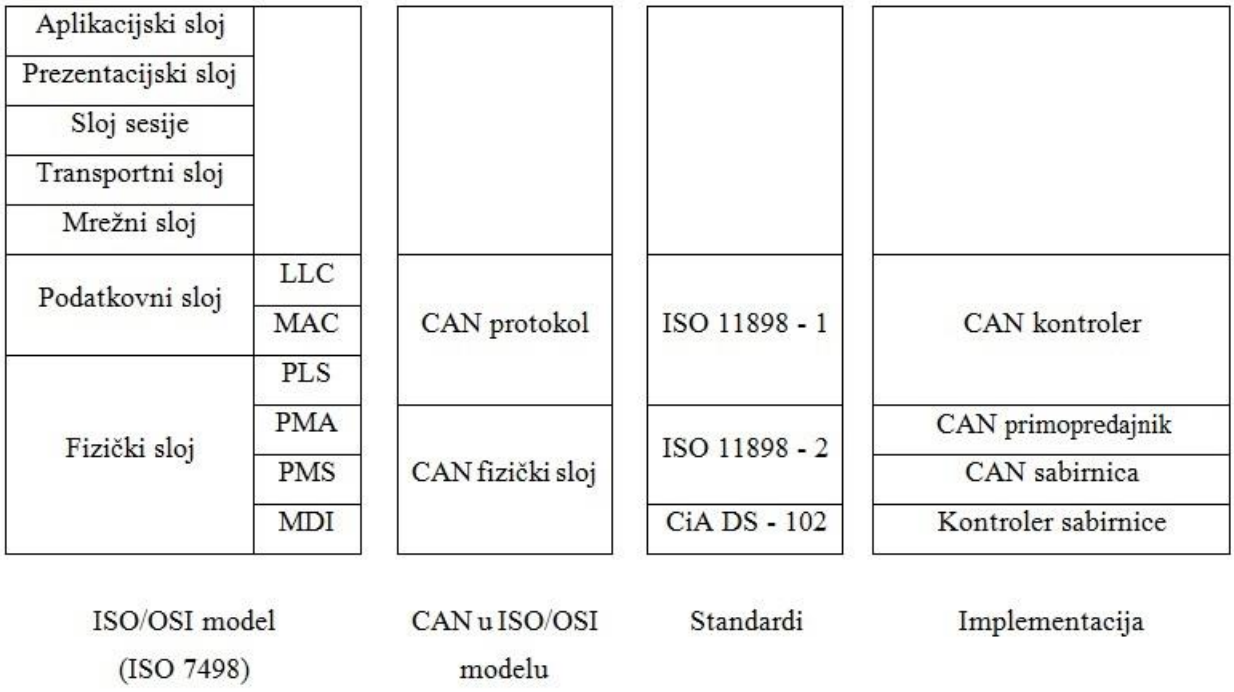
CAN (engl. *Controller Area Network*) je industrijski serijski protokol više razine. Zbog niske cijene, jednostavne i sigurne izvedbe pogodan je za dizajniranje komunikacijskih sustava. Razvio ga je BOSCH, 1986. godine. Konstruiran je prema ISO (engl. *International Standardization Organization*) 11898-1 i ISO 11898-2 standardima. Razvijen je za komunikaciju u stvarnom vremenu, visoke sigurnosne zaštite. Osnovni dio protokola je format poruke ili podatkovnog okvira (engl. *frame*) (u daljnjem tekstu okvir), identifikator okvira i natjecanje za pristup sabirnici. Čvorovi se spajaju na sabirnicu u obliku topologije linije, kao što je prikazano sa slici 2.1. U nastavku poglavlja opisane su osnovne karakteristike protokola, oblik okvira i otvorene implementacije CAN stoga [1].



Sl. 2.1. CAN mreža.

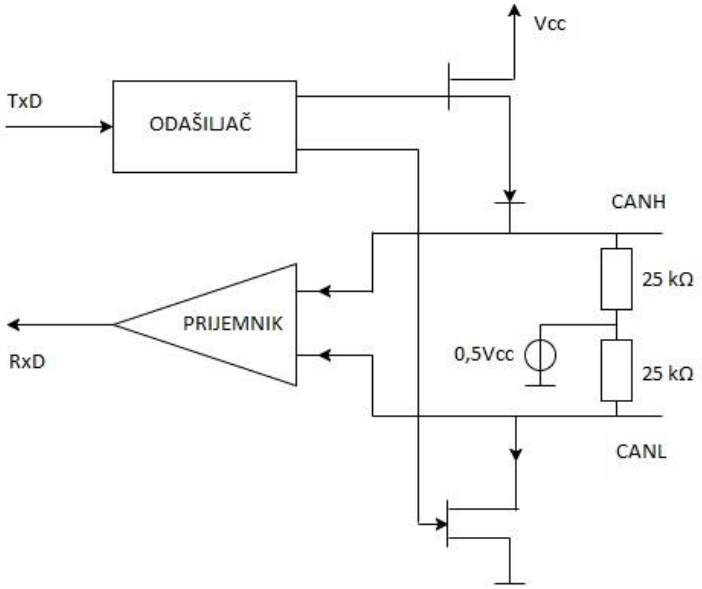
2.1. Osnovne karakteristike

Prema gore navedenim ISO standardima, sabirnica je konstruirana pomoću vodljive upletene parice. Standard zahtjeva visoku otpornost na električne smetnje, samostalno otkrivanje pogrešaka i popravak pogrešaka u komunikaciji. Maksimalna brzina slanja poruka je 1 Mbit/s. Maksimalna duljina sabirnice iznosi 40 metara. Prema OSI (engl. *Open Systems Interconnection*) modelu, protokol pokriva podatkovni sloj i fizički sloj. Shema modela prikazana je na slici 2.2. Standard dozvoljava maksimalno 32 čvora [2].



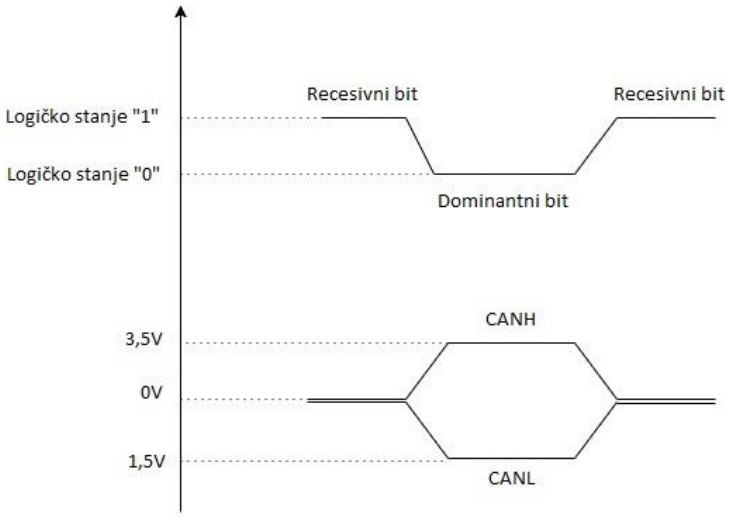
Sl. 2.2. CAN standard i implementacija [2].

Čvor se sastoji od CAN kontrolera i CAN primopredajnika. Kontroler upravlja komunikacijom kako bi se izvodila prema definiranom protokolu. Primopredajnik povezuje kontroler sa sabirnicom. Primopredajnik sadržava dva priključka i pomoću njih se spaja na sabirnicu. Jedan kontakt je za liniju s oznakom visoka (engl. *High line*), a drugi kontakt za liniju s oznakom niska (engl. *Low line*). Shema primopredajnika je prikazana na slici 2.3. Standardno se primopredajnici dijeli u dvije skupine prema brzini slanja podataka. Prva skupina je velike brzine (engl. *High-speed*), gdje maksimalna brzina slanja iznosi do 1 Mbit/s. Druga skupina je niske brzine (engl. *Low-speed*), brzina slanja iznosi do 125 kbit/s [2].



Sl. 2.3. Shema CAN primopredajnika [2].

CAN sabirnicu čine vodljive upletene parice. Prema ISO 11898-2 standardu logička stanja „0“ i „1“ prezentiraju se razlikom napona između linija. Standardne vrijednosti za logičko stanje „0“ (u daljnjem tekstu dominantni bit) je razlika napona iznosa dva volta, a za logičko stanje „1“ (u daljnjem tekstu recesivni bit) je razlika napona iznosa nula volti. Kod CAN-a visoke brzine, razlika napona za dominantni bit su one vrijednost koje prelaze 0,9 volti. Standardne vrijednosti napona dominantnog i recesivnog bita su između -12V i 12V. Na slici 2.4. su prikazane naponske razine za CAN visoke brzine [2].



Sl. 2.4. Naponske razine za CAN visoke brzine [2].

2.2. Okvir

CAN protokol sadrži tri okvira: podatkovni okvir (engl. *data frame*), upravljački okvir (engl. *remote frame*) i okvir za slanje podataka o pogrešci (engl. *error frame*).

Podatkovni okvir služi za slanje podataka i maksimalno može prenijeti podatak veličine osam bajtova. Podatkovni okvir prikazan je na slici 2.5. Slanje podataka počinje SOF (engl. *Start Of Frame*) bitom. Kada se podaci ne šalju, sabirnica ima vrijednost recesivnog bita, zbog toga se za početak slanja okvira SOF bit postavlja na vrijednost dominantnog bita. Zatim slijedi polje od jedanaest bita za identifikaciju (engl. *identifier*, skraćeno ID). Ovo polje služi za postavljanje prioriteta okvirima, a paralelno tome služi kao filter prijemniku, odnosno propuštat će okvire s odgovarajućim identifikatorima, koji su unaprijed zadani. Sljedeći bit je RTR (engl. *Remote Transmission Request*) bit. Kada je u dominantnom stanju, koristi se kako bi pošiljatelj obavijestio primatelja kako slijedi podatkovni okvir. IDE (engl. *Identifier Extension*) bit pokazuje da li je format polja za identifikaciju od jedanaest bita ili dvadeset devet bita. Slika 2.6. prikazuje prošireni podatkovni okvir, odnosno okvir s poljem od dvadeset devet bitova za identifikaciju. DLC (engl. *Data Length Code*) polje od četiri bita pokazuje koliko sljedećih bajtova sadržava podatke. Vrijednost je između nula i osam. Poslije polja s podacima slijedi polje od petnaest bitova zaštite od pogreške. Algoritam za zaštitu je CRC (engl. *Cyclic Redundancy Check*). Završava s DEL (engl. *Delimiter*) bitom. Zatim slijedi bit za pozitivnu potvrdu, ACK (engl. *Acknowledgement*) s pripadajućim DEL bitom. Na kraj okvira ide EOF (engl. *End Of Frame*) polje od sedam recesivnih bitova [1].

SOF	ID – 11 bitova	RTR	IDE	r1	DLC	Podatkovno polje	CRC	DEL	ACK	DEL	EOF

Sl. 2.5. Podatkovni okvir.

SOF	ID – 11 bitova	SRR	IDE	ID – 18 bitova	RTR	r1	r2	DLC	Podatkovno polje	CRC	DEL	ACK	DEL	EOF

Sl. 2.6. Prošireni podatkovni okvir.

Upravljački okvir je istog oblika kao i podatkovni okvir, odnosno upravljački okvir samo nema podatkovno polje. Dijelove upravljačkog okvira može vidjeti na slici 2.7. Kada je polje RTR

u recesivnom stanju, pokazuje kako slijedi upravljači okvir. Svrha upravljačkih okvira je slanje zahtjeva drugom čvoru za slanje unaprijed definiranog podatkovnog okvira.

S O F	ID – 11 bitova	R T R	I D E	r	D L C	C R C	D E L	A C K	D E L	E O F

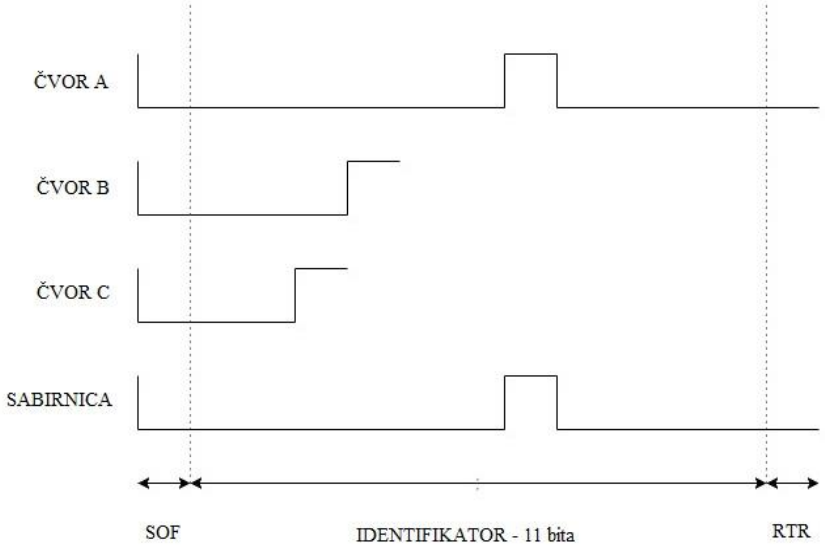
Sl. 2.7. Upravljački okvir.

Okvirom za slanje podataka o pogreški šalju se podaci o vrsti pogreške koja se dogodila tokom komunikacije između čvorova. Standardni čvorovi automatski ponovno šalju podatkovni ili upravljački okvir u kojem se dogodila pogreška što se naziva retransmisija [1].

2.3. Pristup sabirnici

Svaki se čvor može povezati na sabirnicu bez dodatnih zahtjeva i dozvola. Problem nastaje kada više čvorova želi u isto vrijeme slati podatke sabirnicom. Tada dolazi do natjecanja između čvorova. Prilikom slanja podataka, okvir sadrži identifikator, koji ujedno nosi vrijednost prioriteta podatka. Onaj čvor koji ima veći prioritet (manja vrijednost identifikatora), prvi šalje okvir. Na primjer, istovremeno dva čvora šalju okvir. Okvir A ima vrijednost identifikatora 0x00000000010, okvir B ima vrijednost 0x00000000100. Jedan i drugi čvor šalju bite s lijeve na desnu stranu. Do osmog bita su im vrijednosti jednake, odnosno vrijednosti dominantnog bita. Na mjestu devetog bita, okvir čvora B ima vrijednost recesivnog bita, dok okvir čvora A ima vrijednost dominantnog bita. Zato što sabirnica prepisuje recesivni bit dominantnim bitom, prilikom provjere čvor B na tome mjestu prima dominantni bit. Pošto je poslao recesivni bit, a primio je dominantni bit, zna da nema prioritet pri slanju okvira i odustaje od natjecanja i čeka da se sabirnica oslobodi [2].

Na slici 2.8. prikazan je primjer natjecanja između tri čvora. Istodobno kreću slati okvire. Na trećem bitu identifikatora čvor C poprima vrijednost recesivnog bita, zbog toga odustaje od natjecanja i čvor C postaje prijemnik, dok čvor A i B nastavljaju natjecanje. Na četvrtom bitu čvor B poprima vrijednost recesivnog bita. Isto kao i čvor C izlazi iz natjecanja i postaje prijemnik. Pošto je čvor A ostao sam, normalno nastavlja sa slanjem okvira. Nakon što čvor A završi sa slanjem okvira, sljedeći za slanje podataka je čvor B jer ima veći prioritet u odnosu na čvor C. Ako neki drugi čvor želi poslati poruku dok čvor A ili neki preostali čvor šalju, uključuje se u natjecanje s preostalim čvorovima.

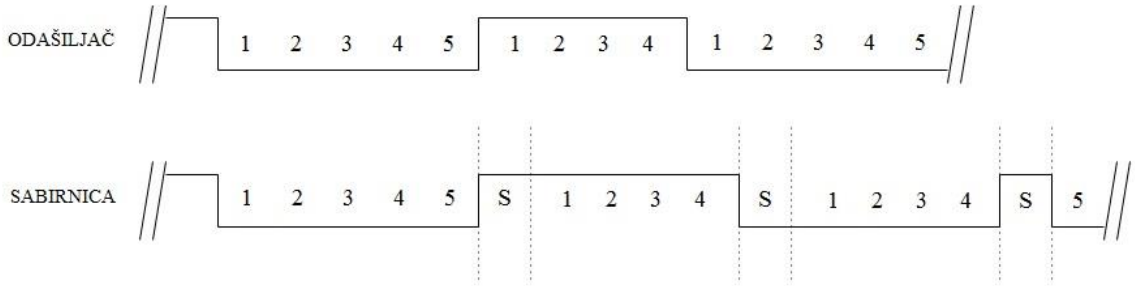


ČVOR	IDENTIFIKATOR
A	0x008 - 00000001000
B	0x080 - 00010000000
C	0x100 - 00100000000

Sl. 2.8. Primjer natjecanja za slanje okvira sabirnicom između tri čvora.

2.4. Bit popune

Bit popune (engl. *Bit Stuffing*, skraćeno S) se dodaje prilikom slanja okvira kada zaredom ide pet isto vrijednosnih bita i suprotne je vrijednosti. Na primjer, sljedeći niz od deset bitova ima vrijednost 0000000110, s dodanim bitom popune poprima sljedeći oblik 00000**1**00110. Kako se EOF polje sastoji od sedam recesivnih bitova potrebno je zamaskirati takav niz bitova u ostalim poljima okvira kako bi se izbjeglo pogrešno čitanje poruke. Odašiljač ih automatski ubacuje, dok prijemnik prilikom čitanja poruke preskače taj bit. Slika 2.9. prikazuje poruku prije i poslije dodavanja bita popune [2].



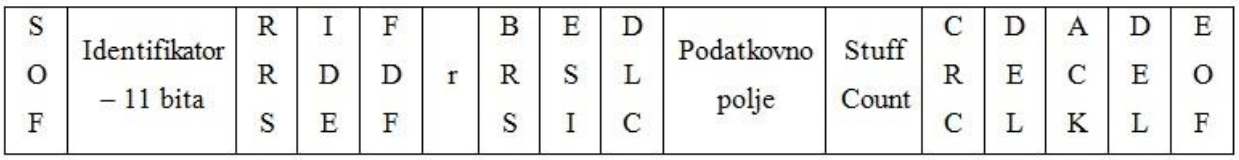
Sl. 2.9. Prikaz dijela poruke prije i poslije dodavanja bita popune.

2.5. CAN FD

Modernizacijom autoindustrije, povećali su se zahtjevi na sabirnicu u vidu velike količine signala koje treba poslati preko sabirnice, s početnih nekoliko stotina do danas preko nekoliko

desetaka tisuća signala. Zbog toga je potrebno povećati brzinu prijenosa podataka. Paralelno tome primopredajnici su se konstruirali od elemenata koje omogućuju bržu obradu podataka, što je povećalo maksimalnu brzinu prijenosa vrijednosti do 150 Mbit/s. Kako bi se smanjio trošak u autoindustriji, zbog prebacivanja sustava na neki drugi pogodniji protokol, Bosch je konstruirano poboljšanu verziju CAN protokola. Uz minimalne izmjene sustava, dobio je CAN FD protokol veće brzine. Novi primopredajnici mogu obrađivati i poruke starije verzije CAN protokola.

Okvir CAN FD protokola je sličan klasičnom CAN protokolu, samo što umjesto osam bajtova podatkovnog polja CAN FD sadrži podatkovno polje do 64 bajta. Slika 2.10. prikazuje CAN FD okvir. Okvir se šalje s dvije različite brzine. Sva ostala polja okvira, osim podatkovnog polja i CRC zaštite, se šalju standardnom brzinom od 1 Mbit/s, a podatkovno polje i CRC zaštita se šalju brzinom do 5 Mbit/s. U natjecateljskom polju okvira umjesto RTR bita dodan je RRS (engl. *Remote Request Substitution*) bit i uvijek je u dominantnom stanju. U kontrolnom polju okvira nakon IDE bita, umjesto rezerviranog bita dodan je FDF (engl. *Flexible Data Rate Format*) bit. Kada je u dominantnom stanju označava klasični CAN okvir, a kada je postavljen u recesivno stanje označava novi CAN FD okvir. Nakon FDF bita slijedi BRS (engl. *Bit Rate Switch*) bit. BRS bit pokazuje brzinu prijenosa podatkovnog polja, ako je u dominantnom stanju brzina prijenosa podatkovnog polja jednake je brzine kao i prijenos ostalih polja okvira, odnosno ako je u recesivnom stanju brzina prijenosa je veće vrijednosti. Slijedi ESI (engl. *Error State Indicator*) bit. Kada je u dominantnom stanju označava kako je kontroler u stanju kontroliranja pogrešaka, odnosno ne prati greške kada je u recesivnom stanju. DLC polje pokazuje koliko je sljedećih bajtova podatkovno polje. Iznosi DLC polja i ekvivalentno tomu količina bajtova podatkovnog polja za klasični CAN i CAN FD prikazane su na slici 2.11. Bit popune u CAN FD ne ide preko cijelog okvira, nego se dodaje samo do kraja podatkovnog polja. Nakon podatkovnog polja ubačeno je polje od četiri bita u kojem se nalazi broj bita popune i to na principu bitnog brojača po modulu 7 (engl. *Stuff Count*). U CRC polje se isto dodaje bit popune, ali u drugačijem obliku. Nakon svaka četiri bita, bez obzira na vrijednost ovih bitova, dodaje se bit suprotne vrijednosti od prethodnog bita. Do 16 bajtova podatkovnog polja CRC polje je veličine 17 bita, a iznad 16 bajtova podatkovnog polja, veličina CRC polja je 21 bit [2].



Sl. 2.10. Okvir CAN FD protokola.

DLC	CAN [bajt]	CAN FD [bajt]
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	8	12
10	8	16
11	8	20
12	8	24
13	8	32
14	8	48
15	8	64

Sl. 2.11. Vrijednost DLC polja i ekvivalentne veličine za CAN i CAN FD.

2.6. ISO standardi

Međunarodna organizacija za standardizaciju (engl. *International Organization for Standardization*, skraćeno ISO) je donijela standarde u autoindustriji vezane uz CAN protokolu. Standardi su doneseni kako bi različiti proizvođači u autoindustriji imali slična pravila u komunikaciji unutar prijevoznog sredstva, odnosno kako bi se pojednostavila i ubrzala proizvodnja različitih dijelova, kao i komunikacija unutar vozila. U diplomskom radu obrađena su dva standarda: ISO 15765-2 i ISO 14229.

2.6.1. ISO 15765-2

Standard je namijenjen slanju podatkovnih okvira putem CAN mreže, a zadaća mu je identificirati vrstu podatkovnog okvira i prema identifikaciji, ako je potrebno, dodijeliti brojčanu vrijednost. CAN protokol u svojem okviru može prenijeti maksimalno osam bajtova podatka. Ako je podatak veći od osam bajtova mora se dijeliti u više okvira, zbog toga je uveden standard. Standard propisuje četiri vrste oznaka podatkovnog polja, i ovisno o vrsti zauzima jedan ili dva bajta podatkovnog polja u okviru. Svako podatkovno polje počinje s oznakom vrste (četiri bita prvog bajta), te ovisno o vrsti oznake slijedi brojčana vrijednost (četiri preostala bita prvog bajta ili dvanaest bitova – četiri bita prvog bajta i drugi bajt). Preostalih šest ili sedam bajtova podatkovnog polja su podaci. Tablica 2.1. prikazuje vrste podatkovnih polja s oznakom [3].

Tab. 2.1. Tablica prikazuje naziv vrste podatkovnog polja s pripadajućom oznakom.

Vrsta	Oznaka
Single frame	0
First frame	1
Consecutive frame	2
Flow control frame	3

Podatkovno polje s oznakom *Single frame* koristi se kada podatak nije veći od sedam bajtova. Za identifikaciju podatkovnog polja dovoljan je jedan bajt, zato može poslati do sedam bajtova podataka. Četiri bita prvog bajta imaju oznaku „0“, zato što je *Single frame*, a preostala četiri bita sadrže broj bajtova podatka. Na primjer, podatak ima vrijednost 0x010203040506 i nakon oblikovanja prema standardu poprima oblik 0x**06**010203040506. U ovom slučaju na početak podatkovnog polja dodana je vrijednost 06 (0 - *Single frame,* 6 – broj bajtova podatka).

Preostale tri vrste podatkovnog polja koriste se zajedno i to za podatke veće od sedam bajtova. Podaci se dijele i prenose se u više okvira. Prilikom slanja više okvira, prvo se šalje okvir s vrstom podatkovnog polja *First frame*. Kako i sam naziv pokazuje, to je prvi okvir i označava primatelju da je to prvi dio podatka i da slijede okviri s ostatkom podatka. Prva četiri bita prvog bajta podatkovnog polja imaju vrijednost „1“. Zatim slijedi dvanaest bitova s vrijednošću ukupnog broja bajtova podatka, kako bi primatelj znao koliko bajtova sadrži podatak, odnosno kako bi mogao rekonstruirati podatak iz više primljenih okvira. Preostalih šest bajtova su podaci. Kada primatelj primi okvir s oznakom podatkovnog polja *First frame,* odgovara s okvirom vrste podatkovnog polja *Flow control frame*. Oblik podatkovnog polja vrste *Flow control frame* prikazan je u tablici 2.2. Polje ima tri vrste statusa slanja, odnosno primatelj odgovara pošiljatelju može li primiti preostale okvire. „0“ označava kako primatelj može primati preostale okvire, „1“ označava kako je primatelj zauzet i da pošiljatelj pričeka novu poruku od primatelja s novim informacijama, odnosno da pošalje novi okvir s podatkovnim poljem vrste *Flow control frame*, „2“ označava kako primatelj ne može primiti preostale okvire. Polje broj okvira informira pošiljatelja koliko može poslati preostalih okvira, te da pričeka novu poruku primatelja s informacijama o kontroli toka za preostale okvire. Vrijednost nula označava kako može poslati sve preostale okvire, a ako je vrijednost veća od nula, označava kako pošiljatelj može poslati taj broj preostalih okvira, te da pričeka odgovor primatelja s novim informacijama o kontroli toka za preostale okvire. Zatim slijedi polje u kojem se nalazi minimalno vrijeme između slanja dva okvira. Vrijednost nula označava kako pošiljatelj može preostale okvire poslati što prije. Vrijednosti od jedan do 127 su rezervirane za milisekunde. Vrijednosti za mikrosekunde se povećavaju za 100 i kreću od

vrijednosti 100 do 900. Za mikrosekunde se šalju kodirane vrijednosti i to u rasponu od 0xF1 za 100 do 0xF9 za 900 [3].

Tab. 2.2. Oblik podatkovnog polja vrste *Flow control frame* [3].

bitovi	7-4 (bajt 0)	3-0 (bajt 0)	15-8 (bajt 1)	23-16(bajt 2)
naziv polja	oznaka	status slanja	broj okvira	Vrijeme između dva okvira
oblik	0	0, 1 i 2	0 - ∞	milisekunde: 0 - 127 mikrosekunde: 100 - 900

Okviri s podatkovnim poljem oznake *Consecutive frame* označavaju kako slijedi preostali dijelovi podatka. Prva četiri bita imaju oznaku „2“, a preostala četiri bita prvog bajta označavaju koji je po redu okvir. Vrijednosti rednog broja okvira kreću se od 1 do 15, a nakon 15 vrijednost se postavlja na 0 i ide do 15. Preostalih sedam bajtova su podaci.

Primjer:
Podatak: 0x 11 22 33 44 55 66 77 88 99 AA BB
Pošiljatelj: 0x 10 0B 11 22 33 44 55 66
Primatelj: 0x 30 00 64
Pošiljatelj: 0x 21 77 88 99 AA BB
Podatak sadržava 11 bajtova. Potrebno ga je podijeliti u dva okvira. Pošiljatelj šalje „1“ za *First frame* i 0x00B=11 za ukupan broj bajtova podatka. Zatim slijedi preostalih šest bajtova podatka. Primatelj odgovara s *Flow control frameom*. 0x3 – oznaka podatkovnog polja. 0x0 – može slati preostale okvire. 0x00 – može slati preostale okvire jer primatelj nema ograničenja u broju okvira koje može primiti. 0x64 – minimalno vrijeme između slanja dva okvira je 100 milisekundi. Pošiljatelj šalje preostali okvir oznake „2“ i vrijednosti 0x1 jer je prvi preostali okvir. Preostali bajtovi podatkovnog polja su podaci.

2.6.2. ISO 14229

Standard definira komunikacijski protokol između elektroničkih komponenata u vozilima i korisnika izvan vozila. Sustav se naziva UDS (engl. *Unified Diagnostic Services*) i nalazi se u ECU (engl. *Electronic Control Unit*) okruženju. ECU se predstavlja kao server koji upravlja čvorom, kao što su aktuatori, senzori i slično, dok UDS predstavlja skup točno definiranih pravila prilikom komunikacije između servera i korisnika koji se povezuje na sustav u vozilu, odnosno gdje korisnik želi doznati podatke od pojedinih servera ili upravljati pojedinim komponentama.

Sve moguće vrste upravljanja i testiranja servera od strane korisnika podijeljene su u točno definirane servise, te svaki servis ima svoj identifikator (engl. *Service Identifier*, skraćeno SID). U tablici 2.3. su navedeni standardni servisi s identifikatorom i opisom [4].

Tab. 2.3. Standardni servisi s pripadajućim identifikatorima i opisom [4].

SID	Servis	Opis
0x10	Diagnostic Session Control	Odabire se način rada, ovisno koji su servisi potrebni
0x11	ECU reset	Ponovno pokretanje servera – kontrolne jedinice
0x14	Clear Diagnostic Information	Briše sve pohranjene informacije o pogreškama u sustavu
0x19	Read DTC Information	Čita pohranjene podatke o pogreškama u sustavu
0x22	Read Data by Identifier	Čita podatak prema identifikatoru
0x23	Read Memory by Address	Čita podatak na memorijskoj adresi
0x27	Security Access	Otključavanje sigurne veze između servera i korisnika
0x28	Communication Control	Upravljanje komunikacijom
0x2A	Read Data by Identifier Periodic	Periodično čitanje podataka prema identifikatoru
0x2E	Write Data by Identifier	Upis podatka prema identifikatoru
0x2F	Input Output Control by Identifier	Upravljanje ulazno/izlaznim stanjima prema identifikatoru
0x34	Request Download	Pohranjivanje programa ili podataka na server
0x35	Request Upload	Pohranjivanje programa ili podataka sa servera na korisnika
0x36	Transfer Data	Dodatak Request Download/Upload, dodatni podaci
0x37	Transfer Exit	Prijenos podataka je dovršen
0x3D	Write Memory by Address	Upis podataka na memorijsku adresu
0x3E	Tester Present	Obavještava server da je korisnik još uvijek prisutan
0x85	Control DCT Setting	Uključuje ili isključuje otkrivanje pogrešaka

Na svaki zahtjev korisnika prema serveru, server odgovara sa povratnom porukom. Ako je zahtjev ispravno napisan i server ga može obraditi, server odgovara sa pozitivnim identifikatorom, odnosno vrijednost pozitivnog identifikatora je primljeni identifikator servisa uvećan za 0x40,

zajedno s pripadajućim podacima. Na primjer, šalje se zahtjev za upis podatka 0x0123 na adresu 0x1111 što se predstavlja zahtjevom 0x3D 1111 0123. Server odgovara da je zahtjev ispravan i odgovara s vrijednošću 0x7D 1111. Ako je zahtjev neispravan server odgovara s negativnom potvrdom. Negativna poruka uvijek počinje s 0x7F, zatim slijedi identifikator servisa te definirani kodovi zašto server nije mogao obraditi zahtjev. Tablica 2.3. prikazuje standardne odgovore s opisom. Na primjer korisnik šalje zahtjev za upisom podatka na memorijsku adresu, 0x3D 1111 0123. Server ne može obraditi zahtjev jer server ne može upisati podatak na tu adresu, jer je zaštićena. Server odgovara sa negativnom potvrdom vrijednosti 0x7F 3D 31 33, pri tome 0x31 označava kako je vrijednost zahtjeva izvan dopuštenog opsega, a 33 znači da je sigurnosno odbijen [5].

Tab. 2.4. Standardne vrijednosti negativnih potvrda s pripadajućim opisom [4].

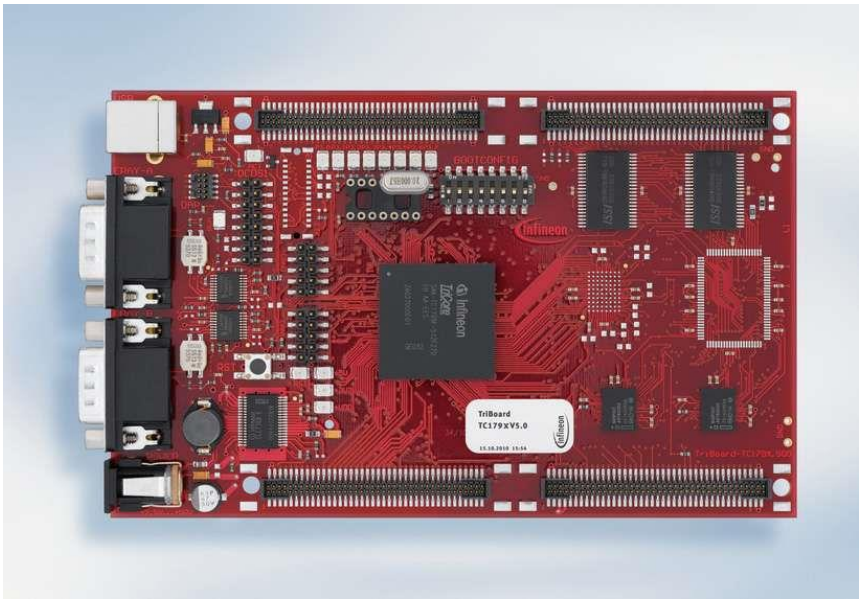
Vrijednost	Opis
0x10	Odbijeno
0x11	Servis nije podržan
0x12	Podfunkcije servisa nisu podržane
0x13	Netočna veličina poruke ili neispravan format
0x14	Odgovor je prevelik
0x21	Server je zauzet za primanje novih zahtjeva
0x22	Uvjeti ili podaci nisu točni
0x24	Pogrešan oblik zahtjeva
0x31	Zahtjev je izvan definiranih veličina
0x33	Odbijen iz sigurnosnih zahtjeva
0x35	Neispravan ključ
0x36	Previše zahtjeva u odnosu na definiranu vrijednost za sigurnosni pristup
0x37	Isteklo sigurnosno vrijeme u kojem je server čekao ključ korisnika
0x70	Nije dozvoljen *download/upload* zbog pogrešnih uvjeta
0x71	Prekinut je prijenos podataka zbog pogreške
0x72	Pogreška prilikom brisanja ili pohrane podataka na memoriju uređaja
0x73	Pogreška u brojaču prilikom slanja podataka
0x78	Server trenutno ne može obraditi zahtjev ali će ga obraditi što prije, šalje periodično dok ne obradi zahtjev
0x7E	Podfunkcija nije podržana u trenutnom načinu rada
0x7F	Funkcija nije podržana u trenutnom načinu rada

3. AURIX PLATFORMA

3.1. Razvojna ploča

TriBoard TC2x7 v1.0 je razvojna ploča tvrtke Infineon, na kojoj se nalazi mikroupravljač TC297A iz grupe Aurix mikroupravljača. Slika 3.1. prikazuje razvojnu ploču TriBoard TC2x7 v1.0. Osnovne karakteristike razvojne ploče:

- 100mm * 160mm EURO tiskana pločica,
- napajanje: 5V,
- oscilator: kristal – 20MHz,
- mikro USB priključak: UART komunikacija (FTDI),
- pristup svim priključcima mikroupravljača,
- 320 muških/ženskih priključaka,
- priključak za primopredajnike: CAN, FlexRay i LIN,
- RJ45 priključak za Internet,
- osam LED,
- tipkalo za ponovno pokretanje [6].



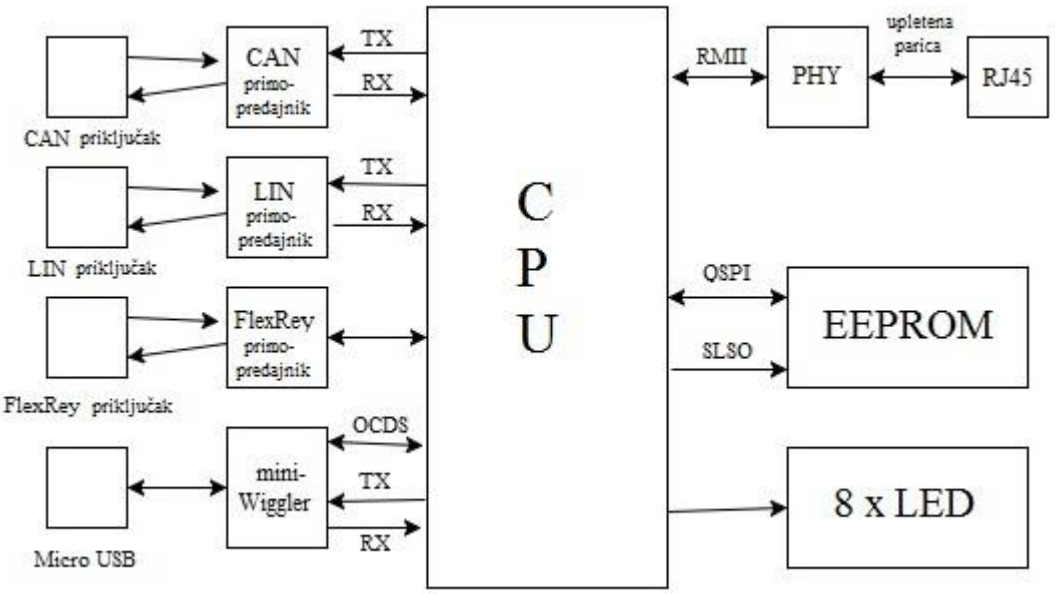
Sl. 3.1. Razvojna ploča TriBord TC297A [6].

Prilikom izrade mikroupravljača TC297A (Sl. 3.2.) korištene su tri tehnologije s kojima se postigla snaga, brzina i ekonomičnost programa koji se izvode na ploči. Smanjen je set instrukcija (engl. *Reduced Instruction Set Computing, skraćeno RISC*), digitalna obrada signala (engl. *Digital*

Signal Processing, skraćeno DSP), memorija i periferija se nalazi na istom integriranom sklopu. Blok shema mikroupravljača prikazana je na slici 3.3. Osnovne karakteristike mikroupravljača:

- 5V napajanje (maksimalno 50V), 500mA,
- Izlazni napon: 5V, 3.3V,
- tri 32 bitne jezgre,
- do 300MHz radni takt u normalnim uvjetima,
- 8MB Flash,
- 384KB EEProm ,
- 728KB RAM,
- GTM (engl. *Generic Timer Module*),
- Internet 100Mbit,
- FlexRey, CAN, CAN FD, LIN, SPI,
- DMA (engl. *Direct Memory Access*) kanali,
- 12 bitni analogno digitalni pretvornik,
- Delta-Sigma ADC pretvarač [6].

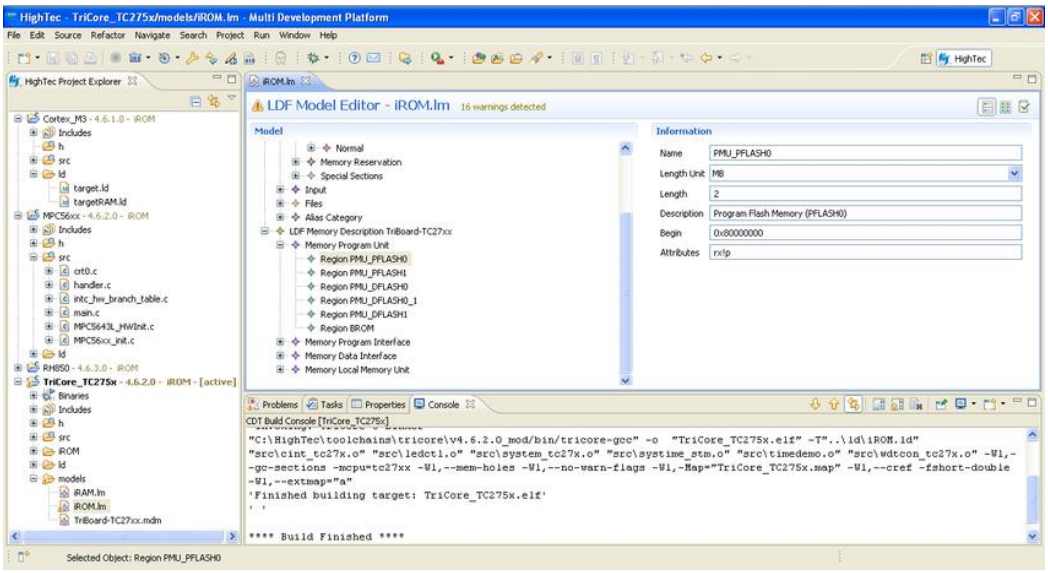
Sl. 3.2. Mikroupravljač TC297A [6].



Sl. 3.3. Blok shema mikroupravljača TC297A [6].

3.2. Razvojno okruženje

HighTec EDV je tvrtka u privatnom vlasništvu. Razvila je HighTec razvojno okruženje. HighTec je alat za razvoj programske podrške ugrađenih sustava. Slika 3.4. prikazuje sučelje HighTec razvojnog okruženja. Zbog svoje pouzdanosti i sigurnosti koristi se u autoindustriji. Programski prevoditelj (engl. *compiler*) je dostupan za sve nove verzije mikroupravljača. Pruža usluge razvoja programske podrške i konzultacije u području optimizacije performansi, kao i prijenos jednojezgrenih sustava na višejezgrene sustave. Baziran je na programskom jeziku C i integriran je u Eclipse razvojno okruženje. Podržava najnovije platforme Aurix-a, ARM-a, RH850, ... HighTec pruža kompletno upravljanje postavkama projekta, kao i procesom izgradnje prevoditelja, *asemblera* i *linkera*. Prilikom pokretanja novoga projekta, omogućuje odabir željenog mikroupravljača. Time generira *startup* program, početne inicijalizacijske postavke za odabrani mikroupravljač, pregled memorijskih mjesta, te posebne funkcije registara [7].



Sl. 3.4. Sučelje HighTec razvojnog okruženja [7].

3.3. ERIKA Enterprise

Erika je prvi otvoreni RTOS (engl. *Real Time Operating System*) koje je certificiran za OSEK/VDX (engl. *Open Systems and their Interfaces for the Electronics in Motor Vehicles / Vehicle Distributed eXecutive*) jezgru (engl. *Kernel*). Programsko okruženje dodaje se kao dodatak (engl. *Plugin*) Eclipse razvojnom okruženju i dolazi kao dodatak s RT-Druidom koji implementira OIL OSEK (engl. *OSEK Implementation Language*). Pruža rad jezgre u stvarnom vremenu otiska (engl. *Footprint*) veličine do 4 kB. Podržava od 8 bitnih do 32 bitnih mikroupravljača, kao i višejezgrene mikroupravljače. Erika API (engl. *Application Programming Interface*) je prenosiv i jednak za sve mikroupravljače, zato što je pisan po OSEK/VDX standardu. Razvojno okruženje pruža gotove funkcije kojima pokreće niti (engl. *Task*), te upravlja njima pomoću funkcija za upravljanje semaforima, alarmima, prekidima, sl. Organizira i raspodjeljuje memoriju [8].

4. CAN PROTOKOL NA AURIX PLATFORMI

Implementacija CAN protokola izvršava se na AURIX platformi. Proizvođač mikroupravljača omogućio je biblioteke upravljačkih programa za sve module koje podržava. U daljnjem dijelu poglavlja opisan je samo CAN upravljački program i njegova prilagodba za spajanje na CAN sabirnicu. Opisana je implementacija programske podrške za standarde ISO 15765-2 i ISO 14229.

4.1. CAN upravljački program

Upravljački program (engl. *Low Level Driver*) je skup funkcija nekog modula koje su potrebne kako bi funkcionalnosti modula bile dostupne. Drugi naziv je i platforma modula, odnosno programiranje niske razine. Najčešće se sastoji od inicijalizacijskih funkcija u kojima se podešavaju postavke modula, zatim ulaznih i izlaznih funkcija u kojima se predaju vrijednosti i iz kojih se dobivaju određene vrijednosti.

Za Aurix mikroupravljače, Infineon je izdao knjižnicu iLLD (engl. *IFX Low Level Driver*) u kojoj se nalaze biblioteke s upravljačkim programima svih modula. S njima su osigurane pristupne i konfiguracijske funkcije svih integriranih perifernih modula. Sve biblioteke su stilski jednako napisani. U daljnjem poglavlju je opisan samo upravljački program za CAN modul.

Mikroupravljač može upravljati s više CAN čvorova na razvojnoj ploči, kao i mijenjati brzinu slanja poruka preko CAN sabirnice. Zbog toga prije same uporabe funkcija modula za slanje i primanje poruka potrebno je inicijalizirati CAN modul, odnosno podesiti postavke kako bi mikroupravljač znao koji se CAN čvor koristi, na kojoj brzini, kao i sami identifikator poruke i druge postavke. Zaglavlje biblioteke poziva se kodom: #include <Multican/Can/IfxMultican_Can.h>. Tablica 4.1. prikazuje osnovne naredbe CAN upravljačkog programa za postavljanje čvora, postavljanje poruke i slanja poruke.

Nakon što se pozove biblioteka deklariraju se potrebne strukture kao globalne strukture u kojima se pohranjuju željene postavke. Upravljački program sadržava posebne funkcije za inicijalizaciju čvorova i oblika okvira. Slijedi podešavanje CAN modula s radnim taktom modula i inicijalizacija. Kada je izvršena inicijalizacija, slijedi kreiranje i podešavanje instance strukture CAN čvorova. Podešava se brzina slanja okvira, koji se čvor koristi za slanje i primanje okvira i kada će očitavati vrijednosti s bita. Kod prikazuje kreiranje instance strukture i postavljanje brzine slanja okvira, zatim slijedi odabir koji će CAN čvor biti za slanje okvira i na kraju inicijalizacija podešenih vrijednosti.

Tab. 4.1. Osnovne naredbe CAN upravljačkog programa.

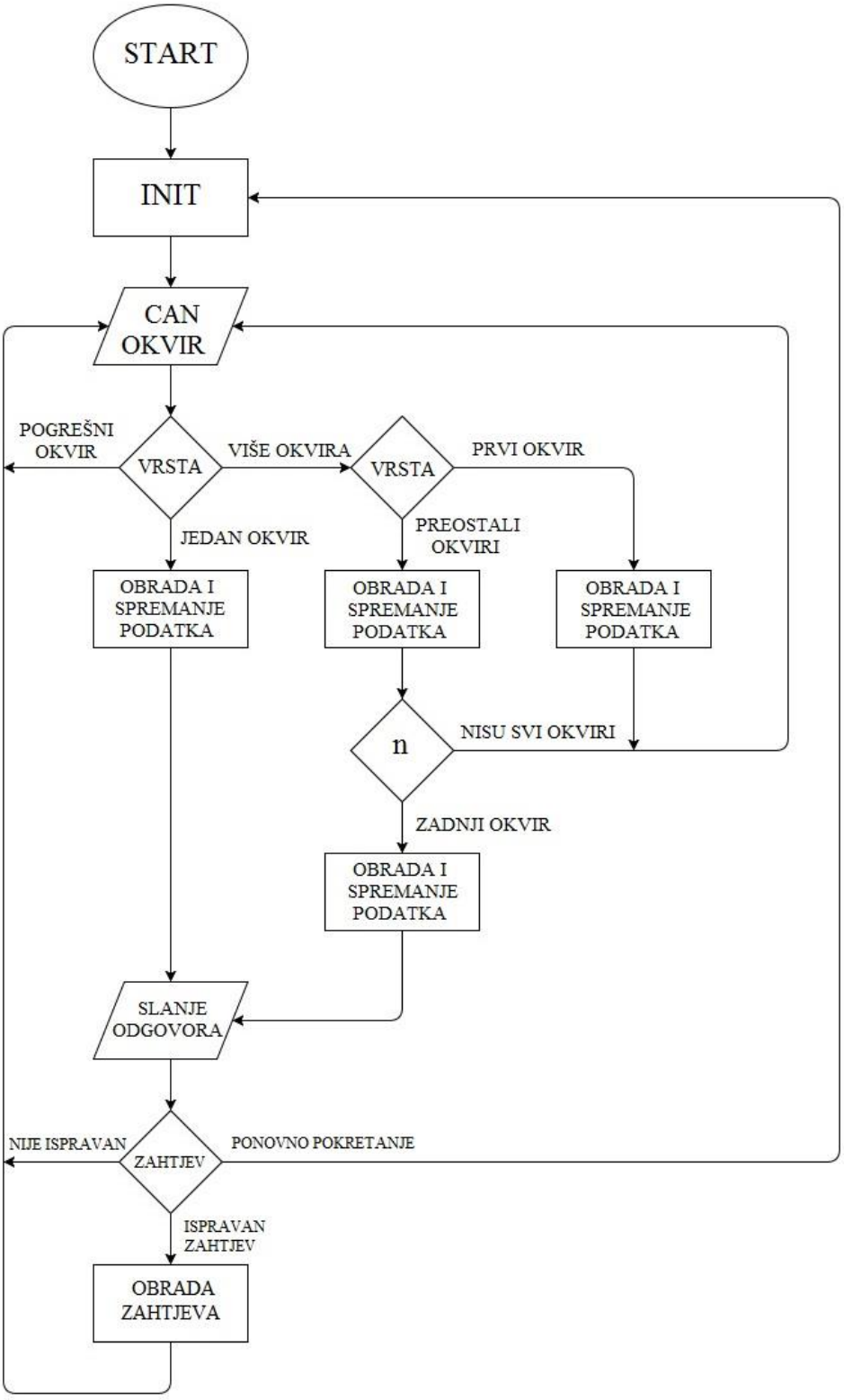
Funkcija	Kod	Opis
Globalne strukture	IfxMultican_Can can	Globalne postavke čvora
	IfxMultican_Can_Node canSrcNode	Postavke odabranog čvora
	IfxMultican_Can_MsgObj canSrcMsgObj	Postavke instance poruke
Podešavanje čvora	IfxMultican_Can_NodeConfig canNodeConfig	Kreiranje instance čvora
	IfxMultican_Can_Node_initConfig(&canNodeConfig, &can)	Inicijalizacija sa definiranim postavkama
	canNodeConfig.baudrate = 1000000	Podešavanje brzine slanja podataka
	canNodeConfig.nodeId = IfxMultican_NodeId_0	Odabir CAN čvora
	canNodeConfig.rxPin = &IIfxMultican_RXD0B_P20_7_IN	Primjer definiranja kontakta za slanje podataka
	IfxMultican_Can_Node_init(&canSrcNode, &canNodeConfig)	Inicijalizacija čvora s podešenim vrijednostima
Podešavanje poruke	IfxMultican_Can_MsgObjConfig canMsgObjConfig	Kreiranje instance poruke
	IfxMultican_Can_MsgObj_initConfig(&canMsgObjConfig, &canSrcNode)	Inicijalizacija sa definiranim postavkama
	canMsgObjConfig.messageId = 0x100	Identifikator
	canMsgObjConfig.frame = IfxMultican_Frame_transmit	Način rada – slanje
	canMsgObjConfig.control.messageLen = IfxMultican_DataLengthCode_8	Veličina podatkovnog polja je osam bajtova
	IfxMultican_Can_MsgObj_init(&canSrcMsgObj, &canMsgObjConfig)	Inicijalizacija poruke s podešenim vrijednostima
Slanje poruke	const unsigned dataLow = 0x11111111 const unsigned dataHigh = 0x22222222	Vrijednost podatka za slanje
	IfxMultican_Message txMsg	Kreiranje instance poruke za slanje
	IfxMultican_Message_init(&txMsg, id, dataLow, dataHigh, IfxMultican_DataLengthCode_8)	Inicijalizacija poruke za slanje s vrijednošću podatka
	IfxMultican_Can_MsgObj_sendMessage(&canSrcMsgObj, &txMsg)	Slanje poruke

Nakon inicijalizacije instance čvora slijedi kreiranje instance Poruka, podešavanje postavki i inicijalizacija. Podijeljeno je u dvije instance, za slanje i za primanje poruka. Dodjeljuje mu se identifikator, način rada (slanje/primanje), duljina podatkovnog polja, vrsta identifikatora i sl.

Gore navedenim postavkama završeno je konfiguriranje CAN čvora i postavljanje oblika poruke za slanje i primanje, te je CAN modul spreman za uporabu. Slanje poruka izvršava se pozivanjem funkcije IfxMultican_Can_MsgObj_sendMessage() i predaje joj se dva argumenta. Prvi argument je odabrani čvor za slanje poruka, a drugi argument je instanca poruke s podacima koja se sastoji od: podatka za slanje, identifikatora i standardne veličine podatkovnog polja.
Primanje poruke izvršava se pozivom funkcije IfxMultican_Can_MsgObj_readMessage() i predaje joj se dva argumenta. Kao i kod primanja poruke prvi argument je čvor s kojim prima poruku, a drugi argument je instanca poruke u koji će spremiti vrijednost koju pročita.

4.2. Implementacija CAN protokola

Postojeći CAN upravljački program za Aurix platformu prilagođen je komunikaciji na CAN sabirnici. Napravljena je programska podrška za CAN komunikaciju prema standardima ISO 15765-2 i ISO 14229. Algoritam programa izvršava se tako što prima poruke definirane standardima preko CAN sabirnice, obrađuje poruke, izvršava zahtjeve i odgovara porukom preko CAN sabirnice. Algoritam programa prikazan je na slici 4.1. Prilikom prvog pokretanja programa ili prilikom ponovnog pokretanja inicijalizira se CAN čvor prema definiranim vrijednostima. Inicijaliziraju se objekti Poruka za način rada slanja i primanja poruka preko definiranog CAN čvora. Nakon uspješne inicijalizacije program poziva funkciju za primanje poruka. Ulazi u beskonačnu petlju i čeka poruku na definiranom CAN čvoru. Kada čvor primi poruku dolazi do provjere vrste podatkovnog polja i prema vrsti podatkovnog polja se dalje obrađuje na odgovarajući način. Ako podatkovno polje ne sadrži pravilnu konstrukciju, program ponovno poziva funkciju za primanje podataka. Ako je vrsta podatkovnog polja *Single frame*, što znači da se podatak nalazi samo u jednoj poruci, podatkovno polje se obrađuje. Iz podatkovnog polja se izbacuju podaci namijenjeni za standard ISO 15765-2, a preostali podaci se pohranjuju. Zatim se po standardu ISO 14229 obrađuje spremljeni podatak i na osnovu tih podataka i podataka definiranih u programu konstruira odgovor. Odgovor se šalje pozivanjem odgovarajuče funkcije. Ako je poruka uspješno poslana, slijedi izvršavanje zahtjeva. Ako zahtjev nema standardom definirani oblik ili neki od parametara imaju netočne vrijednosti, poziva se funkcija



Sl. 4.1. Blok shema algoritma CAN upravljačkog programa s programskom podrškom za standarde ISO 15765-2 i ISO 14229.

za primanje sljedeće poruke. Ako je zahtjev ispravno napisan i svi parametri su unutar definiranih vrijednosti, obrađuje se zahtjev. Nakon obrade zahtjeva poziva se funkcija za primanje sljedeće poruke. U drugom slučaju kada se primi podatkovno polje one vrste gdje treba više poruka kako bi se poslao podatak, slijedi nova provjera. Ako je prva poruka, obrađuje se na isti način kao i kada se šalje podatak samo jednom porukom na način da se izbace podaci iz podatkovnog polja za standard, a preostali se spremaju na definirano mjesto. Zatim se ponovno poziva funkcija za primanje poruka. Pretpostavljamo kako slijedi druga ili viša po redu poruka kod slanja podatka pomoću više poruka. Poruka se obrađuje, i podaci se spremaju na definirano mjesto. Zatim se provjerava dali je primljena poruka zadnja po redu kod slanja više poruka. Ako nije zadnja ponovno se poziva funkcija za primanje poruka, u suprotnom se obrađuje podatak koji je nastao spajanjem više primljenih poruka. Daljnji nastavak programa je isti kao u slučaju s jednom porukom. U daljnjem dijelu potpoglavlja bit će objašnjena prilagodba CAN upravljačkog programa i implementacija standarda.

4.3. Implementacija CAN upravljačkog programa

Postupak inicijalizacije i podešavanja CAN čvora izvršen je prema funkcijama opisanim u poglavlju 4.1. Na razvojnoj ploči odabran je CAN čvor identifikacije nula. Podešen je kako bi čvor primao i slao poruke. Polje za identifikator je standardne veličine, 11 bita. Brzina slanja je postavljena na 1 Mbit/s, ali se po potrebi može mijenjati.

Primanje i slanje poruka podijeljeno je u dvije funkcije: receive() i send(). Funkcija receive() čita poruku s CAN sabirnice. Kako bi se poruka pročitala unutar funkcije se poziva funkcija CAN upravljačkog programa za primanje poruka. Na slici 4.2. je prikazano kako prije poziva funkcije za primanje poruke program ulazi u beskonačnu *while* petlju, odnosno izlazi iz nje tek kada stigne poruka na čvor. Pozivom funkcije za primanje poruke predaju se dva argumenta, čvor identifikacije nula sa statusom rada primanja poruke i struktura u koju će biti pohranjeni podaci o poruci.

```
// wait until Multican received a new message
while( !IfxMultican_Can_MsgObj_isRxPending(&canDstMsgObj) );
// read message
IfxMultican_Status readStatus = IfxMultican_Can_MsgObj_readMessage(&canDstMsgObj, &rxMsg);
```

Sl. 4.2. Funkcija za provjeru stigla li je poruka i funkcija za čitanje poruke.

Ako je poruka uspješno pročitana, vrijednost podatkovnog polja se dijeli na dva dijela po 32 bita. Prvi dio ili bitovi niske vrijednosti (engl. *Low*) spremaju se u definiranu strukturu na prvo mjesto

polja. Drugi dio poruke ili bitovi visoke vrijednosti (engl. *High*) spremaju se na drugo mjesto polja. Radi lakšeg rukovanja s pohranjenim podacima (nalaze se u strukturi s ostalim podacima o poruci), pohranjuju se u strukturu kao što je prikazana na slici 4.3. Nakon podjele i pohrane podatka funkcija završava povratnom vrijednošću oblika te strukture s pohranjenim podacima.

```
typedef struct dataCAN{
    unsigned dataLow;
    unsigned dataHigh;
}dataCAN;
```

Sl. 4.3. Struktura podataka za pohranu podatkovnog polja CAN poruke.

Funkcija send() kao argument prima dvije vrijednosti: nisku i visoku vrijednost poruke koja se šalje. Unutar funkcije instancira se struktura u koju se pohranjuju postavke poruke kao što je identifikator i dva primljena argumenta. Zatim se poziva funkcija CAN upravljačkog programa i predaju se dva argumenta. Prvi argument je struktura s podacima odabranog čvora za slanje poruke, drugi argument je struktura s podacima o poruci. Slika 4.4. prikazuje primjer koda za instancu i pohranu podataka o poruci i funkciju za slanje poruke.

```
// Initialise the message strcture
IfxMultican_Message txMsg;
IfxMultican_Message_init(&txMsg, id, dataLow, dataHigh, IfxMultican_DataLengthCode_8);
// Transmit Data
IfxMultican_Can_MsgObj_sendMessage(&canSrcMsgObj, &txMsg);
```

Sl. 4.4. Inicijalizacija poruke s podacima i funkcija za slanje poruke.

4.4. Implementacija programske podrške za standarde

4.4.1. Programska podrška za standard ISO 15765-2

Implementacija standarda izvršena je prema pravilima opisanim u poglavlju 2.6.1. Cilj standarda je omogućiti slanje podatka veličine veće od osam bajtova, zato što se jednom CAN porukom može prenijeti podatak veličine do osam bajtova. U daljnjem dijelu potpoglavlja opisana je implementacija programske podrške za standard ISO 15765-2.

Kada se poruka primi, poziva se funkcija parseReceive() i u argumentu se predaje visoka (u daljnjem tekstu B podatak) i niska (u daljnjem tekstu A podatak) vrijednost primljenog podatka. Cijela komunikacija u slučaju ovog standarda odvija se slanjem podataka, odnosno naredbi u heksadekatskom obliku, zbog toga se A i B podaci rastavljaju na grupe bitova koje predstavljaju

neku definiranu vrijednost. Četiri najznačajnija bita A podatka predstavljaju vrstu podatkovnog polja. Podatak se pročita i pohrani u varijablu *check*, te služi kao uvjet prilikom provjere vrste podatkovnog polja kada se obrađuje primljena poruka. Podatak je u rasponu vrijednosti od nula do tri. Slijedi provjera podatka. Ako je podatak vrijednosti nula tada sljedeća četiri bita (od 27. bita do 24. bita) trebaju imati vrijednost koja predstavlja broj bajtova podatka i pohranjuje se u varijablu *sizeOfMessageR*. Vrijednost je u rasponu od nula do sedam. Zatim slijedi grananje, odnosno na osnovu vrijednosti *check*, program unutar funkcije ima drugačiju funkcionalnost. Ako je vrijednost nula, izvršava se *while* petlja s brojem koraka iznosa *sizeOfMessageR* (Sl. 4.5.). Petlja se najviše ponavlja do sedam puta, jer se najviše sedam bajtova može nalaziti u podatkovnom polju. Podaci koji nisu vezani za standard se pohranjuju u globalno polje *dataReceive* veličine 50 bajtova. U prvom koraku izvođenja petlje sljedećih osam bitova (od 23. bita do 16. bita) ili drugi bajt po redu počevši od najznačajnijeg bajta u A podatku, se sprema na prvo mjesto polja dataReceive. U sljedećim koracima se redom pohranjuju ostala dva bajta podatka A i četiri bajta podatka B, ovisno o količini bajtova podataka koji se šalju.

```
while(i<sizeOfMessageR)
{
    switch (i)
    {
        case 0:
        {
            dataReceive[i] = (dataLow & 0x00ff0000) >> 16;
            break;
        }
        case 1:
        {
            dataReceive[i] = (dataLow & 0x0000ff00) >> 8;
            break;
        }

         .

         .

         .

        case 6:
        {
            dataReceive[i] = (dataHigh & 0x000000ff);
            break;
        }
    }
    i++;
}
```

Sl. 4.5. Pohrana primljenog podataka u polje.

Ako je vrijednost varijable *check* jednaka jedan, umjesto sljedećih četiri bita za broj bajtova podatka uzima se sljedećih 12 bita (od 27. bita do 16. bita) A podatka i spremaju se u varijablu *sizeOfMessageR*. Sljedeća dva bajta A podatka i četiri bajta B podatka spremaju na prvih šest mjesta polja *dataReceive*. Nakon pohrane podataka slijedi slanje poruke o kontroli toka. Algoritam

sadrži nekoliko scenarija za kontrolu toka i ovisno o odabranom scenariju nastavlja se izvršavanje algoritma. Mogući scenariji su:

- čvor ne može primiti preostale okvire,
- čvor može primiti preostale okvire bez ograničenja, minimalno vrijeme između okvira 100 ms,
- čvor može primiti sljedeća dva okvira i pričekat sljedeću poruku o kontroli toka, minimalno vrijeme između okvira 100ms,
- čvor čeka sljedeću poruku o kontroli toka.

Kada se poruka pošalje algoritam se vraća u stanje ponovnog primanja poruke, jer slijede preostali okviri s podacima. Određeni broj sljedećih okvira imaju vrstu podatkovnog polja s oznakom dva. Odnosno vrijednost varijable *check* je dva. Sljedeća četiri bita u podatkovnom polju (od 27. bita do 24. bita) imaju vrijednost rednog broja okvira koji je poslan/primljen. Vrijednost polja može biti u rasponu od 1, za prvi poslani okvir odmah nakon okvira s oznakom polja jedan, do 15. Nakon 15 kreće brojati od nula, te u pomoćnu varijablu (brojač) poveća za jedan, kako bi se znao točan redni broj primljenog okvira. Ostalih sedam bajtova podatkovnog polja su podaci. Na osnovu rednog broja okvira i pomoćne varijable pozicionira se mjesto za spremanje podataka u polje *dataReceive*.

Ako vrijednost podatka koji označava vrstu podatkovnog polja nije vrijednosti od nula do dva, vrijednost 0xFF se pohranjuje na prvo mjesto u polje *dataReceive*. Ova vrijednost je proizvoljna, a potrebna je zbog daljnjeg izvođenja algoritma, odnosno ako je vrijednost na prvom mjestu 0xFF, primljeni okvir se odbacuje, ne obrađuje se i ne šalje odgovor.

Ako je podatak koji se šalje kao odgovor veći od sedam bajtova, dijeli se na više okvira. Prvo se šalje okvir s podatkovnim poljem oznake jedan s prvih šest bajtova podatka. Zatim algoritam čeka poruku, odnosno poruku o kontroli toka, a oznaka podatkovnog polja je tri. Na osnovu primljene poruke, pretpostavlja se da je primljena poruka o kontroli toka, te se šalju preostali okviri. Minimalno vrijeme između slanja dva okvira dobiva se iz poruke o kontroli toka i poziva se funkcija koja pauzira algoritam na dato vrijeme. Funkcija broji taktove procesora pa se kao argument funkcije predaje broj taktova preračunat iz primljenih milisekundi ili mikrosekundi. Slika 4.6. prikazuje dio koda za slanje prvog okvira i nalazi se unutar *for* petlje (petlja se ponavlja onoliko puta koliko je potrebno poslati okvira). Brojač *k* predstavlja redni broj okvira (nula je prvi okvir). Brojač *j* predstavlja broj bajtova podatka koji se šalje. Prvo se poziva funkcija za dijeljenje podataka u okvire i predaju se argumenti *k* i *j* kako bi se prema standardu mogla složiti podatkovna

polja okvira. Ako se šalje samo jedan okvir, petlja će se izvršiti samo jednom. Kada podatak sadrži više od sedam bajtova (uvjet $j > 7$) algoritam čeka poruku o kontroli toka.

```
if(k == 0)
{
    dataA = parse(k,j);//split message into parts
    send(dataA.dataLow, dataA.dataHigh);
    if(j>7)
    {
        dataARF = receive();
    }
}
```

Sl. 4.6. Kod prikazuje obradu i slanje prvog okvira poruke odgovora i čekanje poruke o kontroli toka ako je podatak veći od sedam bajtova.

Nakon što je primljena poruka o kontroli toka, slijedi slanje preostalih okvira (Sl. 4.7.). Pripremaju se podaci za podatkovno polje (oznake dva s rednim brojem okvira i sedam bajtova podatka) koji se zatim šalju. Nakon toga se iz prethodno primljene poruke o kontroli toka iščita minimalno vrijeme između slanja dva okvira. Poziva se funkcija za pretvaranje milisekundi (mikrosekundi) u broj taktova procesora. Zatim se poziva funkcija za pauziranje izvođenja algoritma i nakon isteka pauze ponavlja se ovaj proces dok se ne pošalju svi okviri.

```
dataA = parse(k,j);//split message into parts
send(dataA.dataLow, dataA.dataHigh);
waitTime1 =  (dataARF.dataLow & 0x0000ff00) >> 8;//minimum time between frames (ms,us)
waitTime = parseTime(waitTime1);//convert ms(us) to ticks
IfxStm_waitTicks(&MODULE_STM0,waitTime);
```

Sl. 4.7. Obrada i slanje preostalih okvira.

4.4.2. Programska podrška za standard ISO 14229

Implementacija standarda izvršena je prema pravilima opisanim u poglavlju 2.6.2. Cilj standarda je definirati komunikaciju prilikom dijagnosticiranja sustava u automobilu, što čini vezu klijent – server. U daljnjem dijelu potpoglavlja opisana je implementacija programske podrške za standard ISO 14229.

Zbog količine i složenosti servisa u standardu, odabrana i obrađena su samo sljedeća četiri servisa: ponovno pokretanje kontrolne jedinice, upis podataka na memorijsku adresu, čitanje podataka s memorijske adrese i upravljanje ulazno/izlaznim stanjima prema identifikatoru. Servisi su odabrani iz razloga što se njihova funkcionalnost može izvršiti na dostupnoj razvojnoj ploči.

Unutar servisa su obuhvaćena sva predviđena stanja, kao i negativni odgovori u slučaju netočnih podataka. Korišteni SID-ovi, vrijednosti podfunkcija i vrijednosti za negativnu potvrdu u algoritmu prikazani su na slici 4.8. Nakon što je podatak u cijelosti primljen i pohranjen slijedi njegova obrada ovisno o odabranom servisu. SID odabranog servisa nalazi se na prvom mjestu polja *dataReceive* i ovisno o odabranom servisu obrađuju se preostala mjesta. Obrada svakog servisa podijeljena je u dva dijela. Prvi dio je slaganje poruke odgovora na zahtjev, a drugi dio je obrada primljenog zahtjeva (npr. ponovno pokretanje kontrolne jedinice, upis vrijednosti na memorijsku adresu i sl.). Ako se vrijednost odabranog servisa ne podudara sa vrijednostima četiri navedena servisa, šalje se negativna potvrda poruka vrijednosti 0x7F XX 11. Poruka se sastoji od sljedećih dijelova: 0x7F što predstavlja da u trenutnom načinu rada nema odabranog servisa, 0x11 predstavlja kako odabrani servis nije podržan i 0xXX predstavlja vrijednost odabranog servisa. U daljnjem dijelu poglavlja objašnjeno je slaganje poruke odgovora i obrada zahtjeva odabranih servisa.

```
#define ER 0x11 //ECU reset
#define WMBA 0x3D //write memory by address
#define RMBA 0x23 //read memory by address
#define IOCBI 0x2F //input-output control by identifier

#define RCTECU 0x00 //return control to ECU
#define IOCP_RTD 0x01 //input-output control parameter - reset to default
#define IOCP_STA 0x03 //input-output control parameter - short term adjustment

#define GR 0x10 //general reject
#define SNS 0x11 //service not supported
#define ROOR 0x31 //request out of range
#define IMLOIF 0x13 //incorrect message length or invalid format
#define CNC 0x22 //condition not correct
#define SAD 0x33 //security access denied
#define SNSIAS 0x7F //service not supported in active session
```

Sl. 4.8. Korišteni SID-ovi s podfunkcijama i vrijednosti negativnih potvrda.

Ponovno pokretanje kontrolne jedinice ima SID vrijednosti 0x11. Nakon bajta koji sadržava vrijednost SID-a slijedi bajt koji označava vrstu ponovnog pokretanja kontrolne jedinice. Tablica 4.2. prikazuje format servisa. Vrijednosti mogu biti od jedan pa do broja koliko ima različitih vrsta podservisa.

Tab. 4.2. Format servisa [9].

Bajt	Opis	Vrijednost
1.	SID	0x11
2.	Vrsta ponovnog pokretanja (podfunkcija)	0x00 – 0xFF

Odabrana je samo jedna vrsta ponovnog pokretanja koja simulira isključivanje napajanja kontrolne jedinice i ponovno uključivanje.

Kada je primljen zahtjev s ispravnim vrijednostima i zahtjev je moguće obraditi, tada slijedi poruka potvrde. Pozitivna potvrda sadržava dvije vrijednosti: pozitivni SID (SID servisa uvećan za 0x40) i vrijednost podfunkcije. U ovom slučaju bi potvrda imala vrijednost 0x51 00 XX, pri čemu 0xXX predstavlja vrijeme potrebno za ponovno pokretanje kontrolne jedinice. Ako je vrijednosti podfunkcije veća od 0x00, zahtjev se odbacuje i šalje se negativna potvrda vrijednosti 0x12 11 10, pri čemu 0x12 označava kako podfunkcija nije podržana, 0x11 označava SID, a 0x10 da je zahtjev odbijen.

Kada je poslana pozitivna potvrda, poziva se funkcija za ponovno pokretanje kontrolne jedinice. Ponovno pokretanje mikroupravljača izvršava se tako što upišemo vrijednost jedan u dva registra (Sl. 4.9.). Prije toga mora se dozvoliti upis vrijednosti u registar, a potom postaviti upisane vrijednosti.

```
uint16 pass =  IfxScuWdt_getSafetyWatchdogPassword();
IfxScuWdt_clearSafetyEndinitInline(pass); //unlock
SCU_RSTCON.B.SW = 1u; // System reset performed when SW reset is triggered
SCU_SWRSTCON.B.SWRSTREQ = 1u;
IfxScuWdt_setSafetyEndinit(pass);
```

Sl. 4.9. Aktiviranje ponovnog pokretanja mikroupravljača.

Servis za upisivanje podatka na memorijsku adresu sastoji se od SID-a vrijednosti 0x3D. Slijedi bajt s vrijednostima veličine polja u kojem se nalazi broj podataka i memorijske adrese (po četiri bita za svaki podatak). Zatim slijedi memorijska adresa, broj podataka koji se šalju i podaci. Format servisa prikazan je u tablici 4.3.

Na primjer, ako se želi spremiti podatak vrijednosti 0x12 34 56 78 na memorijsku adresu 0xAA BB CC DD, zahtjev će imati sljedeću vrijednost: 0x3D **14** AA BB CC DD **04** 12 34 56 78. Drugi bajt po redu ima vrijednost 0x14. Vrijednost „1“ pokazuje da će u polju veličine jedan bajt biti upisana količina bajtova u kojima se nalaze podaci. Vrijednost „4“ pokazuje da je pomoću četiri bajta zapisana memorijska adresa. Bajt vrijednosti 0x04 pokazuje kako se u četiri bajta nalazi podatak.

Tablica 4.4. prikazuje format pozitivne potvrde. Prema gore navedenom primjeru, odnosno zahtjevu, pozitivna potvrda imat će vrijednost: 0x7D **14** AA BB CC DD **04**.

Tab. 4.3. Format servisa [9].

Bajt	Opis	Vrijednost
1.	SID	0x3D
2.	bit 7. – 4. Broj bajtova za polje u koje se sprema veličina podatka u bajtovima (k) bit 3. – 0. Broj bajtova za memorijsku adresu (m)	0x00 – 0xFF
3. ... (m - 2).	Memorijska adresa [1. bajt – m * bajt]	0x00 – 0xFF ... 0x00 – 0xFF
(n - r - 2 - (k - 1)). ... (n - r - 2).	Polje u koje se sprema veličina podatka u bajtovima (r) [1. bajt – k * bajt]	0x00 – 0xFF ... 0x00 – 0xFF
(n - r - 1) ... n.	Podaci [1. bajt – r * bajt]	0x00 – 0xFF ... 0x00 – 0xFF

Tab. 4.4. Format pozitivne potvrde [9].

Bajt	Opis	Vrijednost
1.	SID pozitivne potvrde	0x7D
2.	bit 7. – 4. Broj bajtova za polje u koje se sprema veličina podatka u bajtovima (k) bit 3. – 0. Broj bajtova za memorijsku adresu (m)	0x00 – 0xFF
3. ... ((m - 1) + 3).	Memorijska adresa [1. bajt – m * bajt]	0x00 – 0xFF ... 0x00 – 0xFF
(n – k - 1). ... n.	Polje u koje se sprema veličina podatka u bajtovima (r) [1. bajt – k * bajt]	0x00 – 0xFF ... 0x00 – 0xFF

Algoritam se izvodi u stvarnom vremenu i zbog sigurnosnih razloga nije poželjno mijenjati vrijednosti na određenim memorijskim adresama. Iz navedenog razloga dozvoljava se samo pristup određenim memorijskim adresama. Pomoću *linker script* sintakse u Lcf_Gnuc.lsl datoteci kreirano je polje adresa kojima se može pristupiti, odnosno mijenjati vrijednosti (Sl. 4.10.). Postavlja se početna adresa, a polje je veličine ovisno o broju deklariranih varijabli unutar njega. Memorija mikroupravljača je podijeljena u polja, te se lokalne i globalne varijable pozicioniraju od memorijske adrese 0x60 00 00 00 do 0x60 03 BF FF. Kao početna adresa definiranog polja odabrana je 0x60 00 10 00 i veličine je 28 bajtova (unutar polja se nalazi sedam varijabli veličine četiri bajta. Varijable se deklariraju sljedećim kodom:

- `unsigned int __attribute__((section (".carSection"))) temperature;,`
- `unsigned int __attribute__((section (".carSection"))) speed;.`

Prilikom obrade zahtjeva, ako je poslana adresa izvan vrijednosti definiranog polja adresa, zahtjev se odbija i slijedi poruka negativne potvrde.

```
SECTIONS
{
    .carSection 0x60001000 :
    {
        __MY_SECTION_START = .;
        KEEP (*(.carSection))
        __MY_SECTION_END = .;
    }
}
```

Sl. 4.10. Kreiranje polja u memoriji.

Poruka negativne potvrde šalje se u sljedećim uvjetima:

- memorijska adresa nije veličine četiri bajta i podatak nije zapisan pomoću četiri bajta,
- memorijska adresa je izvan definiranog polja adresa.

Za prvi uvjet memorijska adresa mora biti veličine četiri bajta, odnosno 32 bita, zato što mikroupravljač ima 32 – bitnu adresnu sabirnicu. Zapis podatka pomoću četiri bajta je postavljen samo zbog lakše obrade podatka u daljnjim procesima. Negativna potvrda ima vrijednost 0x13 3D 22, pri čemu 0x13 znači da format poruke nije ispravan, a 0x22 da uvjet nije zadovoljen. Kada se šalje zahtjev s adresom izvan dozvoljenog polja adresa, kao negativnu potvrdu ima vrijednost 0x22 3D 33, pri čemu 0x33 kako je zahtjev sigurnosno odbijen, zato što je odabrana adresa izvan dozvoljenog polja adresa.

Servis za čitanje podataka sa memorijske adrese ima vrijednost SID-a 0x23. Format servisa identičan je kao format pozitivne potvrde kod servisa za pisanje podataka na memorijsku adresu. Na primjer, želi se pročitati podatak veličine četiri bajta sa adrese 0xAA BB CC DD. Zahtjev ima vrijednost 0x23 **14** AA BB CC DD **04**. Format pozitivne potvrde prikazan je u tablici 4.5. Prema gore navedenom primjeru, odnosno zahtjevu, pozitivna potvrda ima vrijednost 0x63 **12 34 56 78**, pri čemu 0x63 označava SID za pozitivnu potvrdu, preostala četiri bajta su podaci.

Tab. 4.5. Format pozitivne potvrde [9].

Bajt	Opis	Vrijednost
1.	SID pozitivne potvrde	0x63
2. ... n.	Podaci [1. bajt ... n * bajt]	0x00 – 0xFF ... 0x00 – 0xFF

Kao i u servisu za pisanje podataka na memorijsku adresu vrijede isti uvjeti za negativne potvrde. Ako adresa nije veličine četiri bajta, negativna potvrda ima vrijednost 0x13 23 22. Ako je adresa izvan definiranog polja adresa, negativna potvrda ima vrijednost 0x22 23 33.

Servis za upravljanje ulazno/izlaznim stanjima prema identifikatoru možemo podijeliti u dvije grupe, odnosno jednim zahtjevom možemo upravljati s jednim ili više ulazno/izlaznih stanja. SID servisa ima vrijednost 0x2F. Identifikator predstavlja jednu ili grupu varijabli i u njih se upisuju podaci (stanja) ili se iz njih stanja čitaju. Servis sadrži sljedeće četiri podfunkcije: 0x00 predstavlja vračanje kontrole nad ulazno/izlaznim stanjem, 0x01 postavlja stanje na početnu vrijednost, 0x02 zadržava trenutno stanje i 0x03 postavlja novo stanje. Svaka od navedenih podfunkcija kao potvrdnu poruku, ovisno od odabranoj podfunkciji, vrača vrijednost stanja koja je bila prije primanja trenutnog zahtjeva. Format servisa prikazan je u tablici 4.6. Prvi bajt predstavlja SID servisa, zatim slijede dva bajta identifikatora ulazno/izlaznog stanja. Četvrti bajt predstavlja odabranu podfunkciju. Ako se odabere podfunkcija za postavljanje novog stanja, peti bajt predstavlja vrijednost novog stanja. U slučaju kada identifikator predstavlja više ulazno/izlaznih stanja, peti pa sljedećih nekoliko bajtova predstavljaju vrijednosti za svaki od stanja (unaprijed je definiran redoslijed). Nakon posljednjeg bajta podatka slijedi bajt koji nosi vrijednost maske. Maska definira koji će se podatak spremiti, npr. od pet stanja koje sadrži identifikator spremit će se samo treći i peti po redu. Ostali podaci se prilikom slanja zahtjeva popunjavaju s vrijednosti 0xXX, kako bi se zadržao redoslijed podataka.

Na primjer ako želimo na identifikatoru vrijednosti 0x11 11 (upravlja s jednim stanjem) promijeniti vrijednost i postaviti na 0x22, zahtjev će imati vrijednost 0x2F 11 11 **03** 22.

Tab. 4.6. Format servisa [9].

Bajt	Opis	Vrijednost
1.	SID	0x2F
2. i 3.	Identifikator [najznačajniji bajt, bajt manjeg značaja]	0x00 – 0xFF ... 0x00 – 0xFF
4. ... (4 + (m - 1)).	Podfunkcija i podaci za svako stanje [podfunkcija, ... m * podatak]	0x00 – 0xFF ... 0x00 – 0xFF
(4 + m). ... (4 + m + (r - 1)).	Maska [1. bajt ... r * bajt]	0x00 – 0xFF ... 0x00 – 0xFF

Ako identifikator upravlja s pet stanja i želimo promijeniti vrijednost na drugom i petom stanju, zahtjev će imati vrijednost 0x2F 11 11 **03** XX **22** XX XX **22 48**. Maska ima vrijednost 0x48, a računa se tako što na najznačajniji bit dolazi prvi podatak i tako redom prema najmanje značajnom bitu. Ako se želi poslati podatak, bit na tome mjestu poprima vrijednost jedan i pretvori se u heksadekatski oblik. U ovome primjeru binarni zapis maske ima vrijednost 0b0**1**00**1**000 (drugi i peti podatak).

Pozitivna potvrda ima format poruke prikazan u tablici 4.7. Za bilo koju odabranu podfunkciju u povratnoj poruci se nalazi trenutna vrijednost stanja, odnosno vrijednost stanja prije promjene. Prema navedenom primjeru (zahtjevu), prije promjene, stanja imaju vrijednosti: 0x33, 0x44, 0x55, 0x66 i 0x77. Pozitivna potvrda ima vrijednost 0x6F 11 11 **03** 33 44 55 66 77.

Tab. 4.7. Format pozitivne potvrde [9].

Bajt	Opis	Vrijednost
1.	SID pozitivne potvrde	0x6F
2. i 3.	Identifikator [najznačajniji bajt, bajt manjeg značaja]	0x00 – 0xFF ... 0x00 – 0xFF
4. ... (4 + (m - 1)).	Podfunkcija i podaci za svako stanje [podfunkcija, ... m * podatak]	0x00 – 0xFF ... 0x00 – 0xFF

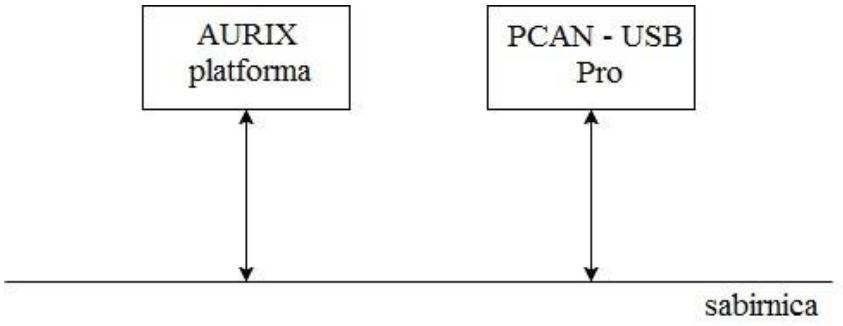
Negativna potvrda događa se u sljedećim uvjetima:

- podfunkcija nije podržana,
- odabran je identifikator sa zabranom pristupa.

Ako podfunkcija nije podržana, odnosno vrijednost bajta nije unutar vrijednosti od 0x00 do 0x03, slijedi negativna potvrda vrijednosti: 0x22 2F 31, pri čemu 0x22 znači da uvjet nije zadovoljen i 0x31 da je zahtjev izvan definiranog okvira. Negativna potvrda za odabrani identifikator sa zabranom pristupa ima vrijednost 0x33 2F 31, pri čemu 0x33 znači da je zahtjev odbijen iz sigurnosnih razloga, nije dozvoljen pristup.

5. EKSPERIMENTALNA MJERENJA

Testiranje prilagođenog CAN upravljačkog programa za Aurix platformu i programske podršku za standarde ISO 15765-2 i ISO 14229 izvršeno je spajanjem razvojne ploče TriBoard TC297A i uređaja PCAN-USB Pro na sabirnicu pomoću CAN priključka. Slika 5.1. prikazuje blok shemu spajanja uređaja na sabirnicu.



Sl. 5.1. Blok shema spajanja Aurix platforme i uređaja.

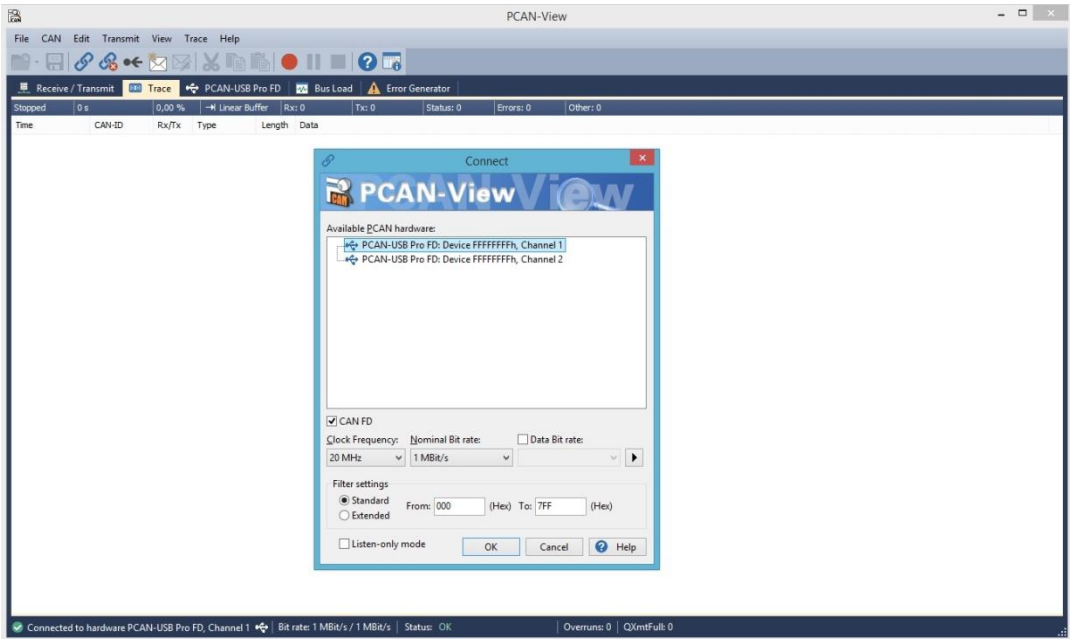
PCAN – USB Pro uređaj služi za testiranje funkcionalnosti CAN upravljačkog programa i programske podrške za standarde, na način da simulira čvor koji predstavlja klijenta. Pomoću uređaja se šalju definirani zahtjevi i primaju se odgovori.

5.1. PCAN – USB Pro

PCAN – USB Pro je uređaj namijenjen spajanju na CAN sabirnicu (Sl. 5.2.). Na računalo se spaja pomoću USB kabela. Uređaj prima i šalje CAN okvire. Brzina slanja okvira je do 1 Mbit/s. Podržava standardni i prošireni identifikator. Istovremeno može raditi kao dva odvojena CAN čvora. Ovaj uređaj prati i broji pogreške na sabirnici i u svrhu testiranja može proizvoditi pogreške prilikom slanja i primanja okvira [10].

Sl. 5.2. PCAN – USB Pro uređaj [10].

Kontrola prometa na sabirnici i slanje okvira izvršavaju se na računalu pomoću softvera PCAN-View (Sl. 5.3.). Softver nema mogućnosti automatskog izvođenja, pa je svaku poruku koja se šalje potrebno ručno odabrati i poslati. Za svaki primljeni i poslani okvir bilježi vrijeme u odnosu na referentnu vrijednost, vrstu identifikatora, veličinu podatkovnog polja, itd. U slučaju pogreške ispiše vrstu pogreške. Vremenski period za obradu podataka je oko 100 mikrosekundi. Prilikom prikaza vrijednosti, podatak je podijeljen u dva dijela po četiri bajta i najznačajniji bajt se prikazuje na desnoj strani i redom ide prema lijevoj strani. Na primjer, ako se sabirnicom šalje podatak vrijednosti 1234 5678, u softveru će biti prikazan oblika 4321 8765. Prilikom kontrole prometa na sabirnici, poruke koje je uređaj poslao imat će oznaku Tx, a poruke koje je pročitao sa sabirnice imat će oznaku Rx [11].



Sl. 5.3. Sučelje softvera PCAN-View.

5.2. Testiranje funkcionalnosti kontrolne jedinice

Testiranje funkcionalnosti kontrolne jedinice, odnosno CAN upravljačkog programa i programske podrške za standarde moguće je provesti slanjem oblikovanih zahtjeva objašnjenih u poglavlju 4.4. i proučavanjem primljenih potvrdnih odgovora. Pomoću uređaja PCAN – USB Pro i softvera PCAN View prati se promet na sabirnici, odnosno odgovore na poslane zahtjeve. Mjerenja se izvršavaju na tri različite brzine slanja podataka: 250 kBit/s, 500 kBit/s i 1 Mbit/s. Ako podatkovno polje od osam bajtova sadrži podatak koji je manji od osam bajtova, dodaju se

bajtovi popune i imaju vrijednost 0xAA. U standardu ISO 15765-2 se najčešće koriste vrijednosti popune: 0x00, 0xAA i 0xFF. Neki bajtovi podataka sadrže vrijednosti 0x00 i 0xFF, pa zbog lakšeg pregleda podataka odabrana je vrijednost 0xAA. U daljnjem dijelu poglavlja prikazati će se rezultati testiranja za odabrane servise i rezultati mjerenja graničnih uvjeta u kojima sustav funkcionira.

5.2.1. Servis 1 - ponovno pokretanje kontrolne jedinice

Poslani zahtjevi:

a) 0x02 11 01 AA AA AA AA AA – ispravan i može se izvršiti,

b) 0x02 11 **04** AA AA AA AA AA – nije ispravan, podfunkcija nije podržana.

Slika 5.4. prikazuje pročitane podatke sa sabirnice prilikom slanja zahtjeva. Na prvi zahtjev odgovoreno je s pozitivnom potvrdom jer je SID zahtjeva uvećan za 0x40 i podatak ima vrijednost 0x02 **51** 01 FF AA AA AA AA. Na drugi zahtjev je odgovoreno negativnom potvrdom i ima vrijednost 0x04 12 11 10 AA AA AA AA. Prema slici se vidi kako je identifikator CAN okvira vrijednosti 0x100, vrsta okvira je podatkovni okvir veličine osam bajtova. Prema vrijednostima vremena u jednom i u drugom slučaju vrijeme potrebno za obradu zahtjeva i slanje odgovora je unutar 100 ms. Softver mjeri vrijeme u sekundama s jednim decimalnim mjestom.

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294976,2	100h	Tx	Data	8	AA 01 11 02 AA AA AA AA
4294976,2	100h	Rx	Data	8	FF 01 51 03 AA AA AA AA
4294987,6	100h	Tx	Data	8	AA 04 11 02 AA AA AA AA
4294987,6	100h	Rx	Data	8	10 11 12 03 AA AA AA AA

Sl. 5.4. Poslani zahtjevi i primljeni odgovori.

5.2.2. Servis 2 - pisanje podataka na memorijsku adresu

Poslani zahtjevi: a) 1. okvir: 0x10 0B 3D 14 60 00 10 00

2. okvir: 0x21 04 00 00 00 22 AA AA,

b) 1. okvir: 0x10 0B 3D **13** 60 00 10 00

2. okvir: 0x21 04 00 00 00 22 AA AA,

c) 1. okvir: 0x10 0B 3D 14 **60 00 01 00**

2. okvir: 0x21 04 00 00 00 22 AA AA.

Na slici 5.5. se vide pročitane poruke sa sabirnice prilikom slanja zahtjeva redom a), b) i c) i iza svakog zahtjeva pripadajući odgovor. Kako se zahtjevi šalju sa po dva okvira, između poslanog prvog okvira, kontrolna jedinica šalje okvir za kontrolu toka. Vrijednost kontrole toka je identične vrijednosti za sva testiranja. Kontrola toka ima vrijednost 0x30 00 64. Vrijednost podatka pokazuje kako se preostali okviri mogu poslati bez uvjeta, s minimalnim vremenom između okvira 100 ms (0x64). Zahtjev a) je ispravno zapisan i može se izvršiti. Podatak vrijednosti 0x00 00 00 22 se zapisuje na memorijsko mjesto 0x60 00 10 00. Zahtjev se sastoji od 11 bajtova i zbog toga se dijeli na dva okvira. Nakon poslanog prvog okvira, kontrolna jedinica odgovara s kontrolom toka, zatim se šalje drugi okvir. Nakon obrade podatka kontrolna jedinica odgovara potvrdnom porukom vrijednosti 0x07 **7D** 14 60 00 10 00 04 (4. poruka). Zahtjev b) nije ispravan. Vrijednost broja bajtova za adresu pokazuje kako je adresa zapisana s tri bajta i kontrolna jedinica odgovara s negativnom potvrdom porukom vrijednosti 0x03 13 3D 22 AA AA AA AA (8. poruka). Zahtjev c) nije ispravan. Vrijednost adrese je izvan dozvoljenog polja adresa. Stoga, kontrolna jedinica pravilno odgovara negativnom potvrdom vrijednosti 0x03 22 3D 33 AA AA AA AA (12. poruka).

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294978,4	100h	Tx	Data	8	14 3D 0B 10 00 10 00 60
4294978,4	100h	Rx	Data	8	AA 64 00 30 AA AA AA AA
4294979,3	100h	Tx	Data	8	00 00 04 21 AA AA 22 00
4294979,3	100h	Rx	Data	8	60 14 7D 07 04 00 10 00
4294986,1	100h	Tx	Data	8	13 3D 0B 10 00 10 00 60
4294986,1	100h	Rx	Data	8	AA 64 00 30 AA AA AA AA
4294987,0	100h	Tx	Data	8	00 00 04 21 AA AA 22 00
4294987,0	100h	Rx	Data	8	22 3D 13 03 AA AA AA AA
4294995,3	100h	Tx	Data	8	14 3D 0B 10 00 01 00 60
4294995,3	100h	Rx	Data	8	AA 64 00 30 AA AA AA AA
4294996,5	100h	Tx	Data	8	00 00 04 21 AA AA 22 00
4294996,5	100h	Rx	Data	8	33 3D 22 03 AA AA AA AA

Sl. 5.5. Poslani zahtjevi i primljeni odgovori.

5.2.3. Servis 3 - čitanje podataka s memorijske adrese

Poslani zahtjevi: a) 0x07 23 14 60 00 10 00 04,

b) 0x07 23 **13** 60 00 10 00 04,

c) 0x07 23 14 **60 00 01 00** 04.

Funkcionalnost servisa slična je servisu za pisanje podataka na memorijsku adresu. Kao primjer šalju se tri zahtjeva. Poslani zahtjevi i primljeni odgovori prikazani su na slici 5.6. Prilikom odgovora potvrdnom porukom, poruka sadrži podatak pohranjen na toj adresi i ukupna vrijednost

poruke zapisana je pomoću pet bajtova, SID pozitivne potvrde i četiri bajta podatka. Zahtjev a) je ispravno zapisan i kontrolna jedinica može izvršiti zahtjev. Šalje se zahtjev za čitanjem podatka na memorijskoj adresi 0x60 00 10 00. Kontrolna jedinica odgovara pozitivnom potvrdom vrijednosti 0x05 63 **00 00 00 22** AA AA. Zahtjev b) nije ispravan. Vrijednost broja bajtova za adresu pokazuje kako je adresa zapisana s tri bajta i kontrolna jedinica šalje negativnu potvrdu vrijednosti 0x03 13 23 22 AA AA AA AA (4. poruka). Zahtjev c) nije ispravan. Vrijednost adrese je izvan dozvoljenog polja adresa i kao odgovor kontrolna jedinica odgovara negativnom potvrdom vrijednosti 0x03 22 23 33 AA AA AA AA (6. poruka).

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294971,1	100h	Tx	Data	8	60 14 23 07 04 00 10 00
4294971,1	100h	Rx	Data	8	00 00 63 05 AA AA 22 00
4294984,6	100h	Tx	Data	8	60 13 23 07 04 00 10 00
4294984,6	100h	Rx	Data	8	22 23 13 03 AA AA AA AA
4295002,3	100h	Tx	Data	8	60 14 23 07 04 00 01 00
4295002,3	100h	Rx	Data	8	33 23 22 03 AA AA AA AA

Sl. 5.6. Poslani zahtjevi i primljeni odgovori.

5.2.4. Servis 4 - upravljanje ulazno/izlaznim stanjima prema identifikatoru

Poslani zahtjevi za identifikator s jednim stanjem:

a) 0x04 2F 11 11 **00** AA AA AA,

b) 0x05 2F 11 11 **03** 50 AA AA,

c) 0x04 2F 11 11 **01** AA AA AA,

d) 0x04 2F **11 00** 00 AA AA AA,

e) 0x04 2F 11 11 **04** AA AA AA.

Identifikator ima vrijednost 0x11 11 i predstavlja varijablu u koju će se pohranjivati podaci i po potrebi čitati. Ulazno/izlazno stanje izravno je povezano s vrijednošću varijable, odnosno na ovaj način se upravlja stanjem. Prva tri zahtjeva su ispravno zapisana i mogu se izvršiti. Slika 5.7. prikazuje poruke na sabirnici prilikom slanja navedenih zahtjeva i primljenih odgovora za svaki poslani zahtjev. Zahtjev a) ima vrijednost podfunkcije 0x00, što predstavlja vraćanje kontrole nad stanjem kontrolnoj jedinici. Pozitivna potvrda ima vrijednost 0x05 6F 11 11 00 **00** AA AA (2. poruka). Vrijednost 0x00 na 6. bajtu nosi vrijednost podatka pohranjenog u varijabli. Zahtjev b) ima vrijednost podfunkcije 0x03, što predstavlja slanje nove vrijednosti kojom se upravlja. Pozitivna potvrda ima vrijednost 0x05 6F 11 11 03 **00** AA AA (4. poruka). Vrijednost 0x00 na 6. bajtu nosi vrijednost podatka pohranjenog u varijabli prije pohrane novog podatka. Zahtjev c) ima

vrijednost podfunkcije 0x01, što predstavlja postavljanje stanja na vrijednost početnih postavki. Pozitivna potvrda ima vrijednost 0x05 6F 11 11 01 **50** AA AA (6. poruka). Vrijednost 0x50 na 6. bajtu nosi vrijednost podatka pohranjenog u varijabli prije postavljanja na vrijednost početnih postavki. Zahtjev d) nije ispravan. Kontrolna jedinica ne sadrži stanje s vrijednošću poslanog identifikatora, te kao negativnu potvrdu šalje vrijednost 0x03 33 2F 31 (8. poruka). Zahtjev e) nije ispravan. Kontrolna jedinica ne sadrži podfunkciju s vrijednošću 0x04, te kao negativnu potvrdu šalje vrijednost 0x03 22 2F 31 (10. poruka).

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294970,6	100h	Tx	Data	8	11 11 2F 04 AA AA AA 00
4294970,6	100h	Rx	Data	8	11 11 6F 05 AA AA 64 00
4294985,7	100h	Tx	Data	8	11 11 2F 05 AA AA 50 03
4294985,7	100h	Rx	Data	8	11 11 6F 05 AA AA 64 03
4294988,8	100h	Tx	Data	8	11 11 2F 04 AA AA AA 01
4294988,8	100h	Rx	Data	8	11 11 6F 05 AA AA 50 01
4294996,2	100h	Tx	Data	8	11 00 2F 04 AA AA AA 00
4294996,2	100h	Rx	Data	8	31 2F 33 03 AA AA AA AA
4295003,3	100h	Tx	Data	8	11 11 2F 04 AA AA AA 04
4295003,3	100h	Rx	Data	8	31 2F 22 03 AA AA AA AA

Sl. 5.7. Poslani zahtjevi i primljeni odgovori.

Poslani zahtjevi za identifikator s više stanja:

a) 0x05 2F 01 55 **00 FF** AA AA,

b) 1. okvir: 0x10 0A 2F 01 55 **03** 07 05,

2. okvir: 0x21 DC 22 96 **78** AA AA AA.

c) 1. okvir: 0x10 0A 2F 01 55 03 XX **04**,

2. okvir: 0x21 **B0** XX XX **40** AA AA AA.

Identifikator ima vrijednost 0x10 55 i definiran je standardnom. Upravlja s četiri stanja. Slika 5.8. prikazuje poruke na sabirnici prilikom slanja navedenih zahtjeva i odgovore na zahtjeve. Zahtjev a) ima vrijednost podfunkcije 0x00. Vrača kontrolu nad svim stanjima (vrijednost 0xFF – vrijednost maske za sva stanja) kontrolnoj jedinici. Pozitivna potvrda sadrži podatke o svakom stanju i veličine je devet bajtova, zbog toga se dijeli na dva okvira. Prvi okvir ima vrijednost 0x10 0A 6F 10 55 00 **07 05** (5. poruka). Zatim kontrolna jedinica čeka poruku o kontroli toka i na osnovu podataka šalje preostali okvir. Drugi okvir ima vrijednost 0x21 **DC 22 96** AA AA AA AA (7. poruka). Vrijednosti unutar ovog okvira koje su pisane masnim slovima su podaci pohranjeni u četiri varijable koje predstavljaju četiri stanja. Zahtjev b) ima vrijednost podfunkcije 0x03. Šalju

se nove vrijednosti za svako stanje. Zahtjev je veličine deset bajtova i dijeli se na dva okvira. Vrijednost maske je 0x78 i označava kako se šalju novi podaci za četiri stanja. Zahtjev c) šalje podatak vrijednosti 0x04 B0 samo za stanje drugo po redu, te maska poprima vrijednost 0x40 (12. i 14. poruka). Za svaki od navedenih zahtjeva u potvrdnoj poruci se nalaze pohranjeni podaci za svako stanje. Poruka je veličine devet bajtova i dijeli se na dva okvira. Negativne potvrde su istog oblika kao i u slučaju s identifikatorom za jedno stanje. Vrijednost 0xXX predstavlja bilo koju vrijednost, zato što se mora postaviti zbog redoslijeda podataka i ne uzima se u obzir prilikom obrade (korišteno 0x00).

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294988,8	100h	Tx	Data	8	55 01 2F 05 AA AA FF 00
4294988,8	100h	Rx	Data	8	01 6F 09 10 03 07 00 55
4294991,3	100h	Tx	Data	8	AA 64 00 30 AA AA AA AA
4294991,3	100h	Rx	Data	8	64 AB E8 21 AA AA AA AA
4295011,1	100h	Tx	Data	8	01 2F 0A 10 05 07 03 55
4295011,1	100h	Rx	Data	8	AA 64 00 30 AA AA AA AA
4295013,7	100h	Tx	Data	8	96 22 DC 21 AA AA AA 78
4295013,7	100h	Rx	Data	8	01 6F 09 10 03 07 03 55
4295015,9	100h	Tx	Data	8	AA 64 00 30 AA AA AA AA
4295015,9	100h	Rx	Data	8	64 AB E8 21 AA AA AA AA
4295049,5	100h	Tx	Data	8	01 2F 0A 10 04 00 03 55
4295049,5	100h	Rx	Data	8	AA 64 00 30 AA AA AA AA
4295050,9	100h	Tx	Data	8	00 00 B0 21 AA AA AA 40
4295050,9	100h	Rx	Data	8	01 6F 09 10 05 07 03 55
4295053,1	100h	Tx	Data	8	AA 64 00 30 AA AA AA AA
4295053,1	100h	Rx	Data	8	96 22 DC 21 AA AA AA AA

Sl. 5.8. Poslani zahtjevi i primljeni odgovori.

5.2.5. Poruka o kontroli toka

Kontrolna jedinica koristi jednu vrstu kontrole toka. Pomoću pomoćnog čvora prikazat ćemo ostale vrste kontrole toka. Za primjer su korištene sljedeće vrste kontrole toka:

a) preostali okviri se mogu poslati, nema ograničenja, minimalno vrijeme između okvira 50ms ili 100ms (vrijednost: 0x30 00 32 ili 0x30 00 64),

b) pričekaj slijedeću poruku o kontroli toka (vrijednost: 0x31 00 64),

c) ne šalji preostale poruke (vrijednost: 0x32 00 64),

d) pošalji sljedeća dva okvira i pričekaj novu poruku o kontroli toka (vrijednost: 0x30 02 64).

Kako bi se mogao pokazati posljednji primjer, potrebno je minimalno poslati četiri okvira podataka, pa je zbog toga promijenit oblik pozitivne potvrde za servis čitanje podataka s memorijske adrese, te se sada šalje unutar šest okvira 35 bajtova podataka

Slika 5.9. prikazuje kontrolu toka za vrstu a). Nakon primljenog zahtjeva, kontrolna jedinica je poslala prvi okvir i čekala je poruku od drugog čvora za kontrolu toka. Nakon primljene poruke poslani su preostali okviri. Prema očitanim vremenima za svaku poruku vidi se, za slučaj minimalnog vremena između dva okvira 50 ms, kako su dvije poruke primljene unutar 100 ms, a u slučaju kada je minimalno vrijeme 100 ms, kako je samo jedna poruka primljena unutar 100 ms.

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294969,9	100h	Tx	Data	8	60 14 23 07 04 00 10 00
4294969,9	100h	Rx	Data	8	00 63 23 10 00 64 00 00
4294971,2	100h	Tx	Data	8	AA 32 00 30 AA AA AA AA
4294971,2	100h	Rx	Data	8	56 34 12 21 34 12 90 78
4294971,3	100h	Rx	Data	8	90 78 56 22 78 56 34 12
4294971,3	100h	Rx	Data	8	34 12 90 23 12 90 78 56
4294971,4	100h	Rx	Data	8	78 56 34 24 56 34 12 90
4294971,4	100h	Rx	Data	8	AA AA 78 25 AA AA AA AA
4294978,8	100h	Tx	Data	8	60 14 23 07 04 00 10 00
4294978,8	100h	Rx	Data	8	00 63 23 10 00 64 00 00
4294979,7	100h	Tx	Data	8	AA 64 00 30 AA AA AA AA
4294979,7	100h	Rx	Data	8	56 34 12 21 34 12 90 78
4294979,8	100h	Rx	Data	8	90 78 56 22 78 56 34 12
4294979,9	100h	Rx	Data	8	34 12 90 23 12 90 78 56
4294980,0	100h	Rx	Data	8	78 56 34 24 56 34 12 90
4294980,1	100h	Rx	Data	8	AA AA 78 25 AA AA AA AA

Sl. 5.9. Minimalno vrijeme između dva okvira: 50 ms i 100 ms.

Slika 5.10. prikazuje slučaj pod b). Kontrolna jedinica nakon primljenog zahtjeva šalje prvi okvir potvrdne poruke i čeka poruku o kontroli toka. Drugi čvor šalje poruku o kontroli toka za slučaj pod b) (3. poruka), što znači da kontrolna jedinica ponovno čeka novu poruku o kontroli toka. Zatim drugi čvor šalje poruku o kontroli toka za slučaj pod a) (4. poruka), što znači da može poslati preostale poruke bez ograničenja, s minimalnim vremenom između okvira 100 ms.

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294972,1	100h	Tx	Data	8	60 14 23 07 04 00 10 00
4294972,1	100h	Rx	Data	8	00 63 23 10 00 64 00 00
4294974,6	100h	Tx	Data	8	AA 64 00 31 AA AA AA AA
4294982,0	100h	Tx	Data	8	AA 64 00 30 AA AA AA AA
4294982,0	100h	Rx	Data	8	56 34 12 21 34 12 90 78
4294982,1	100h	Rx	Data	8	90 78 56 22 78 56 34 12
4294982,2	100h	Rx	Data	8	34 12 90 23 12 90 78 56
4294982,3	100h	Rx	Data	8	78 56 34 24 56 34 12 90
4294982,4	100h	Rx	Data	8	AA AA 78 25 AA AA AA AA

Sl. 5.10. Čekanje sljedeće poruke o kontroli toka.

Slučaj pod c) i d) prikazan je na slici 5.11. Slučaj c) prekida slanje preostalih okvira i kontrolna jedinica čeka novi zahtjev (2. poruka). U drugom slučaju, nakon poslanog prvog okvira, kontrolna jedinica prima poruku o kontroli toka vrijednosti pod d) (6. poruka). Zatim šalje sljedeća dva okvira (7. i 8. poruka) i ponovno čeka novu poruku o kontroli toka. Slijedi poruka vrijednosti pod a) (9. poruka), te kontrolna jedinica pošalje preostale okvire.

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294973,6	100h	Tx	Data	8	60 14 23 07 04 00 10 00
4294973,6	100h	Rx	Data	8	00 63 23 10 00 64 00 00
4294975,7	100h	Tx	Data	8	AA 64 00 32 AA AA AA AA
4294983,4	100h	Tx	Data	8	60 14 23 07 04 00 10 00
4294983,4	100h	Rx	Data	8	00 63 23 10 00 64 00 00
4295002,7	100h	Tx	Data	8	AA 64 02 30 AA AA AA AA
4295002,7	100h	Rx	Data	8	56 34 12 21 34 12 90 78
4295002,8	100h	Rx	Data	8	90 78 56 22 78 56 34 12
4295008,4	100h	Tx	Data	8	AA 64 00 30 AA AA AA AA
4295008,4	100h	Rx	Data	8	34 12 90 23 12 90 78 56
4295008,5	100h	Rx	Data	8	78 56 34 24 56 34 12 90
4295008,6	100h	Rx	Data	8	AA AA 78 25 AA AA AA AA

Sl. 5.11. Kontrola toka za slučaj pod c) i d).

Prilikom brzine slanja od 250 kbit/s, i ako je odabrano minimalno vrijeme između okvira manje od 500 us, dolazi do odbacivanja okvira. Softverska implementacija upravljačkog programa u ovom slučaju ne stigne obraditi prethodni primljeni paket jer mu je potrebno oko 400 us za obradu primljenog paketa pri navedenoj brzini. Slika 5.12. prikazuje pročitane podatke sa sabirnice pri minimalnom vremenu između okvira: 100 us i 500 us. Pri vremenu od 100 us uspješno je primljen samo jedan okvir (4. poruka) od pet. Pri minimalnom vremenu od 500 us uspješno su primljeni svi paketi.

Time	CAN-ID	Rx/Tx	Type	Length	Data
4294983,3	100h	Tx	Data	8	60 14 23 07 04 00 10 00
4294983,3	100h	Rx	Data	8	00 63 23 10 00 64 00 00
4294983,9	100h	Tx	Data	8	AA F1 00 30 AA AA AA AA
4294983,9	100h	Rx	Data	8	56 34 12 21 34 12 90 78
4294998,4	100h	Tx	Data	8	60 14 23 07 04 00 10 00
4294998,4	100h	Rx	Data	8	00 63 23 10 00 64 00 00
4294999,4	100h	Tx	Data	8	AA F5 00 30 AA AA AA AA
4294999,4	100h	Rx	Data	8	56 34 12 21 34 12 90 78
4294999,4	100h	Rx	Data	8	90 78 56 22 78 56 34 12
4294999,4	100h	Rx	Data	8	34 12 90 23 12 90 78 56
4294999,4	100h	Rx	Data	8	78 56 34 24 56 34 12 90
4294999,4	100h	Rx	Data	8	AA AA 78 25 AA AA AA AA

Sl. 5.12. Odbacivanje okvira zbog tromosti softvera.

Ako je odabrana brzina slanja iznosa 500 kbit/s, odbacivanje paketa bit će uslijed minimalnog vremena između okvira vrijednosti manje od 300 us. Pri brzini slanja od 1 Mbit/s ne dolazi do odbacivanja okvira.

Do odbacivanja dolazi zato što je softveru PCAN-View pri brzini slanja od 1 Mbit/s potrebno oko 100 us za obradu primljenog podatka. Ako smanjivo brzinu upola, odnosno na 500 kbit/s, vrijeme obrade se poveća dva puta i iznosi oko 200 us. Do promjene u vremenu obrade podataka dolazi zato što se mijenja radni takt mikroupravljača na PCAN-USB uređaju, ovisno o odabranoj brzini. Funkciju kontrolne jedinice pri manjim brzinama u graničnim uvjetima nije moguće testirati zbog ograničenosti funkcionalnosti opreme.

6. ZAKLJUČAK

CAN komunikacijski protokol se primjenjuje u autoindustriji za komunikaciju kontrolne jedinice sa drugim sporednim jedinicama, kao što su senzori i aktuatori. Pogreške koje se događaju tijekom komunikacije ujedno se i pohranjuju i po potrebi se mogu obraditi. U automobilima je omogućeno povezivanje uređaja koji mogu provesti dijagnozu kontrolne jedinice, odnosno cijelog sustava. Po potrebi se mogu provoditi testiranja sustava i mijenjati postavke. Povezivanje dijagnostičkog uređaja i kontrolne jedinice, radi na principu klijent – server. Dijagnostički uređaj predstavlja klijenta i šalje zahtjeve serveru, odnosno kontrolnoj jedinici. Veza klijent – server izvodi se prema standardu ISO14229 u kojem su definirana pravila komunikacije. Kako je većina podataka veličine veća od osam bajtova, podatak se dijeli na više CAN poruka. Dijeljenje podatka i pravilno označavanje poruka i stanja čvorova tijekom komunikacije odvijaju se prema standardu ISO15765-2.

U radu je postojeći CAN upravljački program prilagođen za spajanje na CAN sabirnicu pomoću Aurix platforme. Izrađena je programska podrška za standarde ISO15765-2 i ISO14229. Zbog složenosti standarda ISO14229 obrađena su samo četiri servisa (zahtjeva) koji se mogu izvršiti na odabranoj Aurix platformi. Testiranje funkcionalnosti kontrolne jedinice izvršeno je povezivanjem Aurix platforme (server) i uređaja PCAN – USB Pro (klijent) pomoću CAN sabirnice. Praćenjem poruka na CAN sabirnici, zabilježene su poruke zahtjeva klijenta i odgovore servera za svaki pojedini servis u svim mogućim uvjetima, te je potvrđena valjanost implementirane programske podrške. Zbog ograničenosti uređaja PCAN – USB Pro nije bilo moguće testiranje funkcionalnosti kontrolne jedinice u uvjetima minimalnog vremena između dva okvira što predstavlja 500 us za brzinu slanja od 250 kbit/s, odnosno 300 us za brzinu slanja od 500 kbit/s. U ostalim uvjetima funkcionalnost kontrolne jedinice se izvršavala prema pretpostavljenim vrijednosti.

Izrađenu programsku podršku moguće je nadograditi tako da podržava i ostale servise standarda ISO14229. Izvršiti testiranje funkcionalnosti kontrolne jedinice uređajem koji može raditi u uvjetima definiranim standardom.

LITERATURA

[1] S. Corrigan, Introducion to the Controller Area Network (CAN), Texas Instruments, 2002.

[2] Vector Informatik GmbH (2016.), E-learning: CAN [online]. Dostupno na: https://elearning.vector.com/index.php?wbt_ls_kapitel_id=1329975&root=378422&seit=vl_can_introduction_en, 15.08.2017.

[3] Wikipwdia (2017.), ISO 15765-2 [online]. Dostupno na: https://en.wikipedia.org/wiki/ISO_15765-2, 18.08.2017.

[4] Softing Automotive Electronics (2016.) , UDS: Unified Diagnostic Services [online]. Dosutopno na: https://automotive.softing.com/fileadmin/soffiles/pdf/de/ae/poster/UDS_Faltposter_softing2016.pdf, 18.08.2017.

[5] Wikipwdia (2017.), Unified Diagnostic Services [online]. Dostupno na: https://en.wikipedia.org/wiki/Unified_Diagnostic_Services, 18.08.2017.

[6] Infineon (2017), Aurix Family: TC297TA [online]. Dostupno na: https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-tm-microcontroller/aurix-tm-family/aurix-tm-family-%E2%80%93-tc297ta/channel.html?channel=db3a304342c787030142dbf31c2b151c, 20.08.2017.

[7] HighTec (2017), Development platform [online]. Dostupno na: https://www.hightec-rt.com/en/products/development-platform.html, 23.08.2017.

[8] Erika Enterprise (2012), Erika Enterprise [online]. Dostupno na: http://erika.tuxfamily.org/drupal/, 23.08.2017.

[9] Internacional Standard, Road vehicles – Unified diagnostic services (UDS) – Specification and requirements, ISO 2006, Switzerland, 2006.

[10] Peak System (2017), PCAN-USB Pro [online]. Dostupno na: http://www.peak-system.com/PCAN-USB-Pro.200.0.html?&L=1, 27.08.2017.

[11] Peak System (2017), PCAN View [online]. Dostupno na: http://www.peak-system.com/PCAN-View.242.0.html?&L=1, 27.08.2017.

POPIS OZNAKA I KRATICA

CiA	(engl. *CAN in Automa*tion)
LLC	(engl. *Logical Link Control*)
MAC	(engl. *Medium Access Control*)
PLS	(engl. *Physical Layer Signaling*)
MDI	(engl. *Medium Dependent Interface*)
PMS	(engl. *Physical Medium Specification*)
PMA	(engl. *Physical Medium Attachment*)

SAŽETAK

CAN je komunikacijski protokol koji se zbog svoje sigurnosti i pouzdanosti najčešće koristi se u autoindustriji. U radu je opisan CAN upravljački program predviđen za AURIX platformu. Postojeći CAN upravljački program prilagođen je za spajanje na CAN sabirnicu. Izrađena je programska podrška za standarde ISO15765-2 i ISO14229. Programski kod pisan je u programskom okruženju ERIKA. Algoritam se pokreće na Infineon Triboard TC297A v1.0 razvojnoj ploči. Testiranje funkcionalnosti CAN upravljačkog programa i programske podrške izvršava se slanjem definiranih zahtjeva uređajem PCAN -USB Pro preko CAN sabirnice prema razvojnoj ploči i proučavanjem odgovora.

Ključne riječi: CAN upravljački program, AURIX platforma, ISO15765-2, ISO14229, ERIKA Enterprise.

ABSTRACT

CAN is communication protocol which is used in automotive industry because of its security and reliability. This master thesis describes CAN driver made for AURIX platform. The CAN driver is adjusted for CAN bus connections and program solutions for ISO15765-2 and ISO14229 were made. Source code has been written using ERIKA Enterprise operating system. Algorithm is running on Infineon Triboard TC297A v1.0 experimental board. CAN driver and program solution is being tested by sending certain kind of requests by PCAN-USB Pro device which also uses CAN bus to communicate with experimental board and by interpreting the received response.

Key words: CAN low level driver, AURIX platform, ISO15765-2, ISO14229, ERIKA Enterprise.

ŽIVOTOPIS

Stjepan Dekanić je rođen 28.05.1992. godine u Županji. Osnovnu školu završio je 2007. godine u Andrijaševcima. Tehničku školu Ruđera Boškovića Vinkovci, smjer elektrotehničar, upisuje 2007. godine, te je završava 2011. godine, s odličnim uspjehom. 2007. godine je osvojio treće mjesto na Državnom natjecanju mladih tehničara. Tijekom srednjoškolskog obrazovanja sudjeluje na raznim državnim natjecanjima. 2012. godine upisuje preddiplomski sveučilišni studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Komunikacije i informatika, te završava 2015. godine. 2015. godine upisuje diplomski sveučilišni studij elektrotehnike, smjer Komunikacije i informatika na istom fakultetu. Od mjeseca studenog, 2016. godine prima stipendiju tvrtke Institut RT-RK iz Osijeka.

Dobro poznavanje engleskog jezika i odlično poznavanje informatike.

Posjeduje vozačku dozvolu B kategorije.