УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Урош Јовић

Једно решење програмске подршке за тестирање FlexRay комуникације помоћу алата специјализованих за тестирање у аутомобилској индустрији

МАСТЕР РАД

Нови Сад, 2017.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Дипломски – мастер рад	
Аутор, **АУ**:		**Урош Јовић**	
Ментор, **МН**:		**Доц. Др Богдан Павковић**	
Наслов рада, **НР**:		**Једно решење програмске подршке за тестирање FlexRay комуникације помоћу алата специјализованих за тестирање у аутомобилској индустрији**	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		**2017**	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		**(7/40/0/1/19/1/0)**	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кључне речи, **ПО**:		**Тестирање, FlexRay, AUTOSAR, наменски системи**	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		**У овом раду приказано је једно решење за тестирање комуникације преко FlexRay магистрале помоћу које се одвија размена података на мрежи међусобно повезаних електронских контролних јединица. Презентована имплементација је специјално подешена за тестирање наменских система. Фокус је на детекцији грешака које настају током преноса FlexRay порука.** **У оквиру рада такође је представљен предлог имплементације дијагностичког модула који детектује, прикупља и приказује јединствен код грешке у циљу што прецизнијег откривања самог порекла те грешке.**	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	**Др Миодраг Ђукић**	
	Члан:	**Др Јелена Радић**	Потпис ментора
	Члан, ментор:	**Др Богдан Павковић**	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		**Uros Jovic**	
Mentor, **MN**:		**PhD Bogdan Pavkovic**	
Title, **TI**:		**One solution for testing FlexRay communication using dedicated tools in automotive industry**	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		**2017**	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**(7/40/0/1/19/1/0)**	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		**Testing, FlexRay, AUTOSAR, embedded systems**	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		**This paper shows one solution for testing the communication through FlexRay bus, which connects multiple electronic control units in the system. Presented implementation is specially adjusted for testing of embedded systems. The focus is on detecting the errors that appear during FlexRay message transfer. Within the paper there is also a proposed solution of a diagnostic module that collects and displays the unique error code in order to precisely determine the origin of the error that occurred.**	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	**PhD Miodrag Djukic**	
	Member:	**PhD Jelena Radic**	Menthor's sign
	Member, Mentor:	**PhD Bogdan Pavkovic**	

Zahvalnost

Zahvaljujem se Danijelu Spasojeviću i celom TTTech timu koji su mi pomogli pri izradi rada.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

ECU - *Electronic Control Unit*

CAPL - *Communication Access Programming Language*

HTML - *Hypertext Markup Language*

CAN - *Controller Area Network*

TDMA - *Time Division Multiple Access*

FTDMA - *Flexible Time Division Multiple Access*

FIBEX - *FIeld Bus Exchange Format*

AUTOSAR - *AUTmotive Open System ARchichecture*

CRC - *Cyclic Redundancy Check*

RTE - *Runtime environment*

MW - *Middleware*

DTC - *Diagnostic Trouble Code*

E2E - *End To End*

API - *Application Programming Interface*

CSV - *Comma Separated Values*

PDU - *Protocol Data Unit*

1. Uvod

Elektronika je danas jedna od glavnih stvari koje pokreću razvoj novih, originalnih rešenja u automobilskoj industriji (eng. *automotive*). Još početkom sedamdesetih godina bio je primetan rast elektronskih sistema u kolima koji su polako zamenjivali mehaničke i hidrauličnе sisteme. Tada je svaka nova funkcionalnost bila implementirana kao zasebna elektronska kontrolna jedinca (eng. *ECU*), koja prestavlja podsistem sačinjen od mikro-kontrolera i skupa senzora i aktuatora. Ubrzo je postalo jasno da ovakav pristup nije dovoljno dobar usled potrebe da funkcionalnost bude distribuirana na više elektronskih jedinica i takođe, usled potrebe da implementirane funkcionalnosti razmenjuju podatke među sobom. Na primer, brzina vozila koja se estimira na osnovu informacija dobijenih sa senzora obrtaja točkova se koristi da se prilagodi napor upravljanja volanom ili da bi se podesila brzina kojom rade brisači. Sve do početka devedesetih godina, podaci su se prenosili preko zasebnih veza koje spajaju dve elektronske jedinice. Ova strategija nije uspela da se nosi sa povećanjem broja elektronskih jedinica u automobilu (moderni automobili današnjice upošljavaju preko 70 ECU-ova) jer su se javili problemi sa težinom, cenom, kompleksnošću i pouzdanošću ovako umreženog sistema. To je dovelo do pojave mreža gde se komunikacija odvija pre zajedničkog, deljenog medija, što je za posledicu imalo potrebu da se definišu pravila - protokoli za upravljanje komunikacijom. Da bi se olakšalo upravljanje ovim i budućim namenskim sistemima, industrijski konzorcijum sačinjen od vodećih proizvođača i dobavljača elektronskih komponenti je predstavio *FlexRay* protokol, koji će biti detaljnije objašnjen u narednom poglavlju. [1]

Porast kompleksnosti namenskih sistema uslovio je i porast vremena i napora potrebnog za testiranje ovih sistema. Na primeru autopilota koji kombinuje sakupljene podatke sa radara, senzora, kamera i na osnovu njih generiše upravljačke instrukcije, vidi se da samo jedna pogrešna vrednost signala može dovesti do situacija koje mogu ugroziti život putnika. Jasno je

da je testiranje presudno kada treba proceniti da li sistem ispravno implementiran i podešen i da li će ispravno funkcionisati, čak i u slučaju pojave grešaka.

1.1 Zadatak rada

Cilj rada je bio da se razviju, implementiraju i potpuno automatizuju testovi koji će temeljno ispitati sve vrednosti *FlexRay* signala u kompleksnom namenskom sistemu. Ideja je bila da se ovi testovi mogu koristiti u različitim fazama životnog ciklusa sistema.

Rešenje je realizovano pomoću *VECTOR*-ovih alata uz prateću programsku podršku a za izradu je korišćen *CAPL* (*Communication Access Programming Language*) jezik. Izveštaj se generiše u obliku .html (*Hypertext Markup Language*) fajla koji se može otvoriti u bilo kom internet pretraživaču i koji omogućava detaljan uvid u dobijene rezultate i njihovu analizu.

2. Teorijske osnove

Prilikom dizajniranja namenskog sistema za rad u automobilu, jedan od glavnih ciljeva jeste obezbediti pravilno i pravovremeno izvršavanje funkcija vozila u skladu sa unapred definisanim nivoima bezbednosti. Kako su funkcionalnosti od velike važnosti podeljene između više elektronskih jedinica i imaju potrebu da stalno komuniciraju, mreže u vozilu su te koje se staraju da sve funkcioniše ispravno.

U automobilskoj industriji postoje dva tipa komunikacije:

1. Komunikacija pokrenuta na osnovu nekog događaja (eng. *event-triggered*).
2. Komunikacija pokrenuta u određenom periodu vremena (eng. *time-triggered*).

2.1 *Event-triggered* komunikacija

Event-triggered komunikacija je komunikacija u kojoj se slanje poruka odvija nakon prijema nekog od značajnih događaja (dinamički), npr. otvaranja/zatvaranja vrata. Sistemi bazirani na ovom tipu komunikacije se oslanjaju na komunikacione protokole koji definišu prava pristupa magistrali za razmenu podataka kako ne bi došlo do kolizije. CAN protokol (eng. *Controller Area Network*) je ovo rešio dodelom prioriteta svakoj poruci, tako da pristup magistrali ima poruka najvišeg prioriteta. Ovo je jedna od prednosti *event-triggered* komunikacije jer ne koristi celokupan propusni opseg (eng. *bandwidth*), nego se samo šalju neophodne poruke. Takođe, sistemi su u stanju da munjevito reaguju na bilo kakvu pojavu asinhronih događaja koji nisu unapred poznati. Mane su nepredvidljivost, problemi sa detekcijom grešaka kod mrežnih čvorova i može doći do problema kad postoje određeni vremenski okviri koji moraju da se ispoštuju.[13]

Tipični predstavnici *event-triggered* komunikacije, pored *CAN*-a, su i *Byteflight*, *LonWorks* i *Profibus*. [1] [2]

2.2 *Time-triggered* komunikacija

Kod *time-triggered* komunikacije slanje poruka se odvija u predodređenim periodima vremena (statički). Ovi vremenski periodi se najčešće nazivaju vremenski otvori (eng. *slot*) i njihov raspored se kontinualno ponavlja. Strategija kojoj se pristupa magistrali se naziva TDMA (eng. *Time Division Multiple Access*). Pošto je raspored frejmova statički definisan, lako je proveriti da li su svi vremenski okviri u kojima se odvija razmena poruka ispoštovani. Pozitivna osobina koja proizilazi iz toga je da sve poruke koje nisu poslate odmah detektuju, što indirektno može da bude od koristi prilikom identifikacije mrežnih čvorova koji su prestali sa radom. Najveća mana ovog tipa komunikacije je manjak fleksibilnosti jer se uspešna implementacija ove komunikacije zasniva na unapred poznatim vremenskim periodima izvršavanja procesa i slanja poruka. [13]

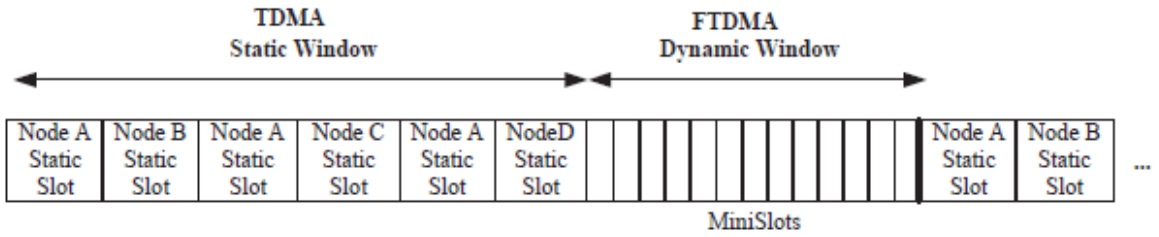
Neki od predstavnika *time-triggered* komunikacije su *SAFEbus*, *SPIDER* i *Time-Triggered Protocol* (TTP).

2.3 *FlexRay* protokol

FlexRay protokol je predstavljen 2004. godine od stane konzorcijuma čiji najbitniji članovi su bili *BMW, Bosch, Daimler, General Motors, NXP Semiconductors, Freescale, Semiconductor* i *Volkswagen*. Ima dva kanala (A i B) koji podržavaju razmenu podataka u oba smera brzinama i do 20 Mb/s (10Mb/s po kanalu). Zbog toga postoji mogućnost razmene podataka preko samo jednog od kanala, čime se dobija na uštedi propusnog opsega. Poruke takođe mogu simultano da se prenose na oba kanala što doprinosi otpornosti na greške, jer ukoliko se jedan kanal ošteti, razmena podataka se može nesmetano odvijati preko drugog kanala.[11]

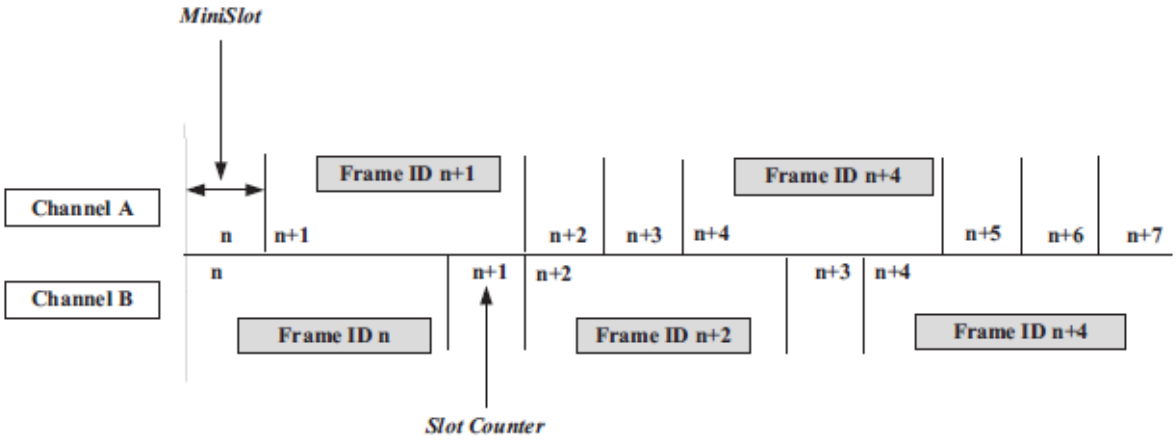
FlexRay protokol je hibridni protokol jer se podaci mogu prenositi na način na koji se prenose u *event-triggered* i *time-triggered* komunikaciji. Svaka poruka u jednom od komunikacionih ciklusa sadrži statički i dinamički deo. Statički deo koristi TDMA protokol, dok dinamički deo koristi FTDMA (*Flexible Time Division Multiple Access*) protokol.[10]

Oba dela su dodatno podeljena na vremenske otvore (eng. *mini-slots*) koji su kod statičkog dela identične veličine, dok kod dinamičkog dela veličina varira.



Slika 2.1 Izgled *FlexRay* komunikacionog ciklusa

Na slici 2.2 se može videti da se u okviru dinamičkog dela frejmovi šalju u *mini-slot*-ovima n, n+2 i n+4 na kanalu B. Frejm iz *mini-slot*-a n+4 koji se šalje na oba kanala, nije primljen simultano na kanalima A i B pošto se prenos podataka u dinamičkom delu se odvija nezavisno na oba kanala.[1]



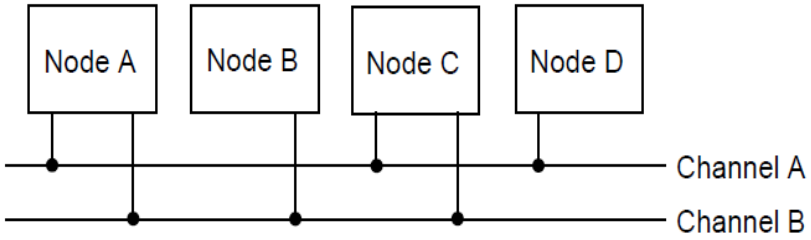
Slika 2.2 Raspored poruka u dinamičkom segmentu *FlexRay* komunikacionog ciklusa

2.3.1 Topologija *FlexRay* mreže

Postoji nekoliko opcija za konfigurisanje *FlexRay* mreže. Može biti konfigurisana kao magistrala, zvezda ili kombinacija ove dve topologije. Svaki mrežni čvor (eng. *node*) može biti povezan na samo jedan ili na oba kanala. Ova fleksibilnost se može iskoristiti da se poveća propusni opseg ili da se dodano poveća otpornost sistema na greške.[11]

2.3.1.1 Topologija magistrale

Svaki čvor može ima mogućnost povezivanja samo sa jednim ili sa oba kanala *FlexRay* mreže. Razdaljina između dva čvora ne sme da bude veća od 24 metra i maksimalan broj čvorova na jednom kanalu ne sme da pređe 22.



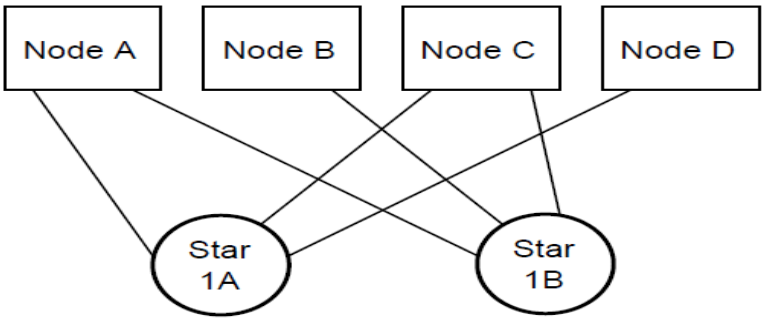
Slika 2.3 Topologija magistrale *FlexRay* mreže

2.3.1.2 Topologija zvezde

Čvorovi su povezani direktno (eng. *point-to-point*) na spojnice (eng. *coupler*) koje u sebi sadrže *transmitter* i *receiver* za svaku granu zvezde. Čim se podaci prime na jednoj, automatski se šalju na sve ostale grane.

Jedna od prednosti ove topologije je to što su grane nezavisne, što znači da ukoliko dođe do greške u radu na jednoj do grana, ostale mogu neometano da nastave sa radom.

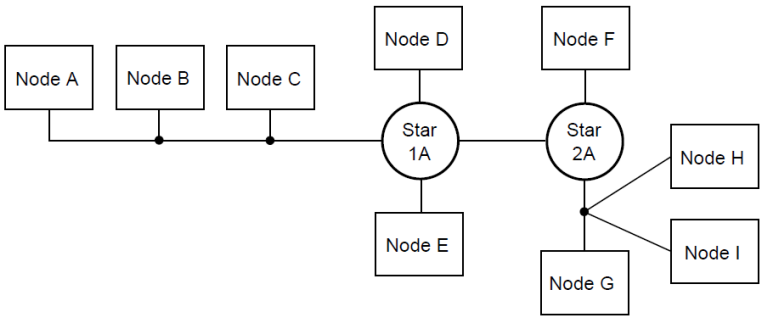
Maksimalan broj grana po jednoj zvezdi je 16, a maksimalna dužina grane je kao i kod magistrale 24 metra.



Slika 2.4 Topologija zvezde *FlexRay* mreže

2.3.1.3 Hibridna topologija

Hibridni tip *FlexRay* mreže predstavlja kombinaciju magistrale i zvezde. Postoji veliki broj rešenja za konfigurisanje ovog tipa mreže, jedino o čemu se mora voditi računa je da se ne prekorače limiti koji se odnose na svaku individualnu topologiju od koje je hibridna sačinjena.

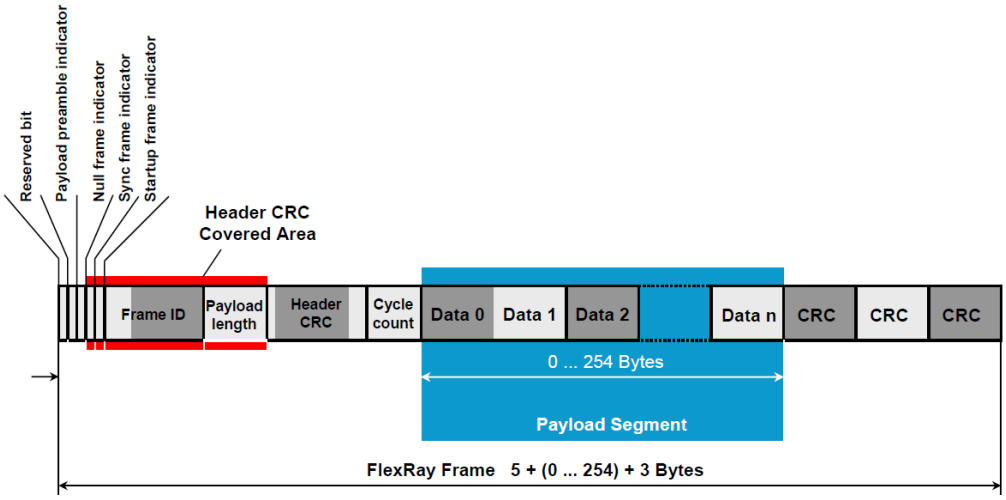


Slika 2.5 Hibridna topologija *FlexRay* mreže

2.4 Izgled *FlexRay* frejma

FlexRay frejm se sastoji iz tri dela:

- zaglavlja (eng. *header*)
- podataka (eng. *data*)
- pratećeg segmenta



Slika 2.6 Izgled *FlexRay* frejma

Zaglavlje *FlexRay* frejma čine 5 bajtova:

- rezervisani bit za buduću upotrebu (šalje se nula najčešće)
- indikator podataka:
  - za statički frejm označava prisustvo vektora za upravljanje *FlexRay* mrežom
  - za dinamički frejm označava prisustvo ID-a poruke u podacima
- indikator nultog frejma:
  - ukoliko je nula znači da nema validnih podataka
  - ukoliko je jedinica znači da su podaci validni
- indikator frejma za sinhronizaciju koji ukoliko je jedinica označava da je ovo frejm za sinhronizaciju
- indikator početnog frejma koji ukoliko je jedinica označava da se radi o početnom frejmu

ID frejma (11 bita) definiše okvir (eng. *slot*) u kom bi frejm trebalo da se pošalje. Može imati vrednosti od 1 do 2047. (ID = 0 znači da je u pitanju nevalidan frejm ID)

Veličina podataka (7 bita) služi da pokaže kolika je veličina segmenta za podatke. Broj ovog polja u frejmu se množi sa 2 i tako se dobija prava veličina segmenta sa podacima u bajtovima.

Kod za kontrolu tačnosti prenetih podataka zaglavlja (eng. *Header CRC*) je veličine 11 bita. Računa se nad indikatorom frejma za sinhronizaciju, indikatorom početnog frejma, ID-jem frejma i nad poljem za veličinu podataka.

Broj ciklusa (6 bita) predstavlja trenutnu vrednost brojača ciklusa u momentu slanja frejma. Segment podataka može da sadrži od 0 do 254 bajta.

Prateći segment se sastoji od 24-bitnog koda za proveru tačnosti prenetih podataka. koja se računa za zaglavlje i segment podataka. U proračun su uključena sva polja od kojih se sastoje ovi segmenti.[3]

2.5 FIBEX

FIBEX (eng. *FIeld Bus Exchange Format*) je XML format koji se koristi za opisivanje mreža u automobilskoj industriji. Dizajniran je tako da podržava jednostavne i kompleksne mreže, uključujući i podršku za vezne čvorove (eng. *gateway*). Kompatibilan je sa mnogim protokolima koji komunikaciju baziraju na razmeni poruka : CAN, *FlexRay*, LIN i MOST. [4][5]

Sadrži informacije o raznim aspektima mreže:

- Rasporede slanja i primanja poruka
- Definicije frejmova
- Definicije signala
- Topologiju mreže
- Konfiguraciju mreže zajedno sa brzinom prenosa podataka i vremenskim okvirima
- Informacije o ECU-ovima i njihovim konekcijama

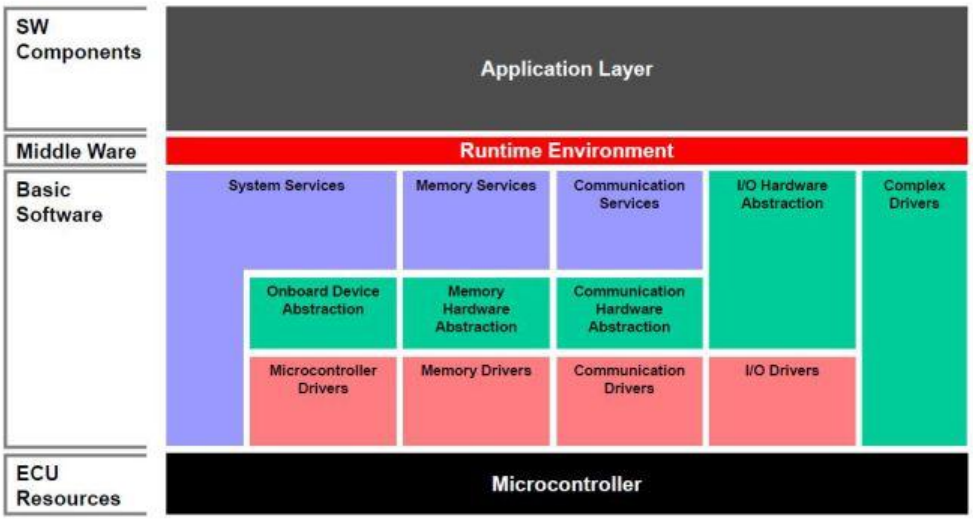
```
- <FRAME>
    <SHORT-NAME>C_W1</SHORT-NAME>
    <FRAME-LENGTH>16</FRAME-LENGTH>
  - <PDU-TO-FRAME-MAPPINGS>
    - <PDU-TO-FRAME-MAPPING>
        <PDU-REF DEST="SIGNAL-I-PDU">/AUTOSAR/C_W1</PDU-REF>
        <START-POSITION>0</START-POSITION>
      </PDU-TO-FRAME-MAPPING>
    </PDU-TO-FRAME-MAPPINGS>
  </FRAME>
- <FRAME>
    <SHORT-NAME>W1_M1</SHORT-NAME>
    <FRAME-LENGTH>16</FRAME-LENGTH>
  - <PDU-TO-FRAME-MAPPINGS>
    - <PDU-TO-FRAME-MAPPING>
        <PDU-REF DEST="SIGNAL-I-PDU">/AUTOSAR/W1_M1</PDU-REF>
        <START-POSITION>0</START-POSITION>
      </PDU-TO-FRAME-MAPPING>
    </PDU-TO-FRAME-MAPPINGS>
  </FRAME>
- <FRAME>
    <SHORT-NAME>C_W2</SHORT-NAME>
    <FRAME-LENGTH>16</FRAME-LENGTH>
  - <PDU-TO-FRAME-MAPPINGS>
    - <PDU-TO-FRAME-MAPPING>
        <PDU-REF DEST="SIGNAL-I-PDU">/AUTOSAR/C_W2</PDU-REF>
        <START-POSITION>0</START-POSITION>
      </PDU-TO-FRAME-MAPPING>
    </PDU-TO-FRAME-MAPPINGS>
  </FRAME>
```

Slika 2.7 Deo FIBEX fajla

2.6 AUTOSAR model

AUTOSAR model pruža detaljan opis arhitekture programske podrške koja se nalazi u elektronskim kontrolnim jedinicama.

Model AUTOSAR-a je sačinjen od nekoliko slojeva: aplikativni sloj, sloj osnovne programske podrške i okruženje koje nastaje tokom izvršavanja (eng. *Runtime environment - RTE*).[1]



Slika 2.8 Izgled AUTOSAR modela

2.6.1 Aplikativni sloj

Aplikativni sloj se sastoji od nekoliko softverskih komponenti. Softverska komponenta predstavlja najmanji deo aplikativne programske podrške koja ima određenu funkcionalnost.

Kada je reč o AUTOSAR-u, komponente se generalno sastoje od nekoliko zasebnih entiteta koji se nazivaju *runnables.* Oni se ciklično izvršavaju u skladu sa definisanim rasporedom. Svaka komponenta ima strogo definisane portove preko kojih komunicira sa ostalim komponentama. Svaki port se sadrži tačno jedan interfejs. [6][7]

Aplikativne softverske komponente su povezane i komuniciraju preko magistrale virtualnih funkcija (eng. *Vritual Function Bus*). Zahvaljujući njoj, komponente ne moraju znati sa kojim komponentama komuniciraju. One samo proslede svoje izlazne informacije magistrali koja se postara da te informacije stignu do portova komponenti kojima su potrebne. Logički, magistrala virtualnih funkcija je implementirana preko okruženja nastalog tokom izvršavanja i sloja osnovne programske podrške, dok je njena fizička reprezentacija takođe komponenta koja se naziva *middleware.*

2.6.2 Okruženje nastalo tokom izvršavanja

Kroz okruženje nastalo tokom izvršavanja komponente mogu da komuniciraju i ako nisu mapirane na određenu fizičku arhitekturu ili elektronsku kontrolnu jedinicu. Praktično RTE "oslobađa" komponente od fizičke arhitekture i komponente jedne od drugih. Ključ je u tome da svaka ECU mora imati implementiran RTE u sistemu zasnovanom na AUTOSAR-u.

Preko RTE se odvijaju inter-ECU i intra-ECU komunikacije. Inter-ECU komunikacija je tip komunikacije u kojoj se softverske komponente ne nalaze na istoj ECU, dok intra-ECU je tip komunikacije u kojoj se komponente nalaze na istoj ECU.[6][7]

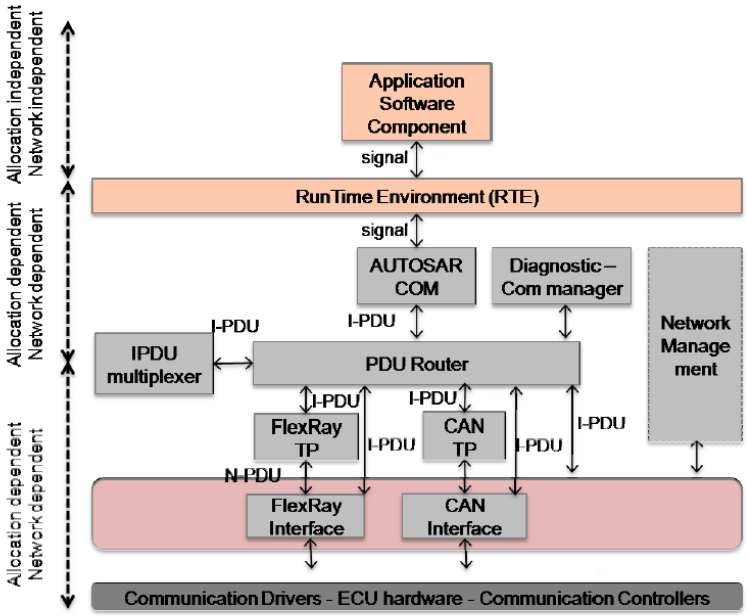
2.6.3 Sloj osnovne programske podrške

Sloj osnovne programske podrške u AUTOSAR-ovom modelu predstavlja sloj standardizovane programske podrške koja pruža usluge softverskim komponentama. Sačinjen je od servisa koji uključuju dijagnostičke protokole, servise za upravljanje memorijom i komunikacijom kao i komponentama specifičnim za ECU-ove poput sloja za pristup mikrokontrolerma i složenih drajvera za uređaje.

Softverske komponente iz aplikativnog sloja nemaju direktan pristup sloju osnovne programske podrške, već je pristup omogućen zahvaljujući RTE-u koji generiše odgovarajuće aplikativne sprege za pristup ovom sloju. [6]

2.6.4 AUTOSAR COM komponenta

Složeni zadatak upravljanja objektima, kojim na raznim nivoima upravljaju različiti servisi, poveren je AUTOSAR COM komponenti. COM komponenta je bazirana i izgrađena na osnovu prethodnog poznavanja PDU-ova, signala unutar njih i rasporeda ostalih softverskih komponenti na ECU-ovima. Jedan od glavnih zadataka COM komponente je pakovanje signala u PDU i njihovo raspakivanje iz istih. Ona obezbeđuje prenos informacija ukoliko se komponente nalaze na istoj namenskoj jedinici ili kreira odgovarajuće objekte za slanje kroz niže slojeve ka udaljenim softverskim komponentama.[1]



Slika 2.9 Izgled AUTOSAR COM komponente

2.6.5 E2E mehanizmi

Da bi se obezbedio siguran prenos kritičnih podataka, AUTOSAR standard je definisao "sa kraja na kraj" (eng. *End to End*) mehanizme koji garantuju zaštitu razmenjenih podataka. Nadogradnjom signala unutar PDU-a prilikom slanja sa CRC i BZ-om (brojač sekvenci) i proverom ovih polja nakon prijema informacija, obezbeđena je sigurna komunikacija među softverskim komponentama. CRC štiti sadržaj podataka dok je ispravan redosled prijema podataka zaštićen brojačem sekvenci podataka.[8]

2.7 Testiranje namenskih sistema

Namenski sistem predstavlja deo mnogo veće celine sačinjene od mehaničkih i električnih komponenti. Veza između fizičkog dela celine i namenskog sistema je ostvarena pomoću senzora i aktuatora. U stanju je da obradi velike količine informacija, istovremeno poštujući određene vremenske okvire. Kontrolisani procesi određuju vreme reakcije namenskog sistema što znači da sistem mora da odgovori na pobudu za vreme koje je definisano od strane tog procesa.

Ciklus dizajniranja namenskog sistema sastoji se od nekoliko faza: pisanje specifikacija, validacija, implementacija, testiranje i analiza rezultata testiranja.

Testiranje predstavlja bilo kakvu aktivnost namenjenu evaluaciji da li se program ili sistem ponaša u skladu sa očekivanjima. Pre nego što samo testiranje počne, neophodno je znati da li sistem koji se testira (eng. *System Under Test*) uopšte može biti testiran, koje testne sekvence se mogu primeniti na njega i kako.[12]

Postoji nekoliko testnih metoda koje daju odgovor na pitanje da li sistem radi u skladu sa očekivanjima:

1. Verifikacija - uobičajena procedura tokom razvoja sistema koja uključuje proveru da li se sistem ponaša u skladu sa osobinama koje su implementirane. Glavni cilj ove metode je upravo otkrivanje grešaka u implementaciji.
2. Testiranje usklađenosti - predstavlja proces evaluacije da li su parametri implementiranog sistema usklađeni sa propisanim standardom ili specifikacijom. U ovom slučaju sa specifikacijom *FlexRay* protokola.
3. Testiranje robusnosti - definisano je kao procena ponašanja sistema u prisustvu grešaka i/ili po sistem stresnih ulaznih vrednosti. Idealno, sve moguće ulazne vrednosti bi trebalo testirati.
4. Testiranje interoperabilnosti - svrha ovog testiranje je dokazivanje da dva komunikaciona sistema funkcionišu bez problema , zajedno u namenskom sistemu, poštujući standarde na kojima su ti komunikacioni sistemi zasnovani.
5. Testiranje u cilju održavanja - koristi se da otkrije i lokalizuje greške koje se pojave tokom upotrebe sistema. Testiranje ovog tipa je najčešće orijentisano na greške fizičke prirode koje se mogu desiti kao posledice starenja.

Neke metode testiranja zahtevaju umetanje grešaka da bi se proverio rad implementiranih mehanizama za detekciju i obradu grešaka. One se obično koriste zato što tokom normalnog rada sistema nema grešaka.[9]

Dok se testiranje uglavnom koristi za detekciju problema, debagovanje, koje se koristi za nalaženje grešaka koje su uzrok nastalog problema, je podjednako važno. To je razlog zašto se mora obezbediti mehanizam za beleženje pobuda i događaja zajedno sa njihovim vremenom izvršavanja. Takođe, vrlo je korisno ukoliko postoji mehanizam za reprodukovanje zabeleženog sadržaja tako da je moguće ponovo izazvati isti problem i utvrditi njegov uzrok.

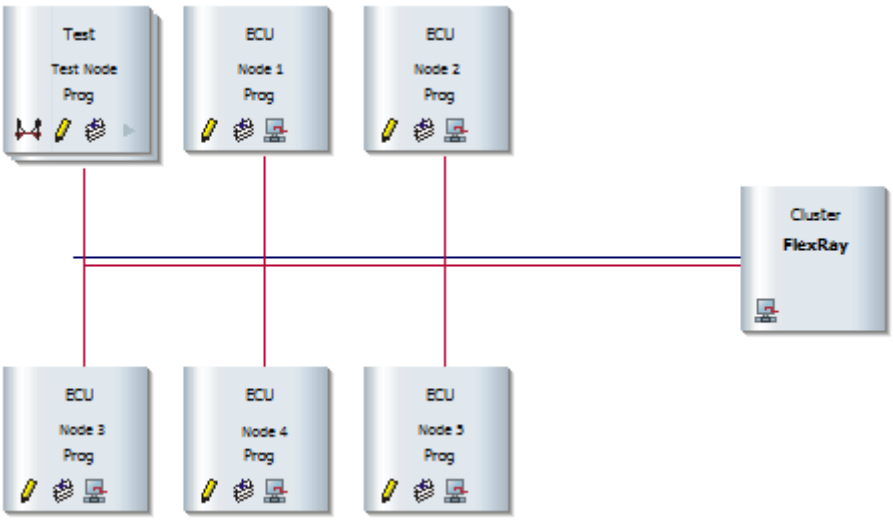
3. Koncept rešenja

U radu je predstavljeno nekoliko grupa testova specijalno razvijenih za kompleksnu ciljnu platformu. Testovi su korišteni za proveru implementacije, kao i za simuliranje punog opterećenja kojem će sistem biti izložen prilikom rada.

Platforma koja se testira ima nekoliko namenskih jedinica (eng. *host*). Namenske jedinice imaju jedinstvene IP (eng. *Internet Protocol*) adrese i komunikacija između njih se odvija putem *Ethernet*-a. Više od deset softverskih komponenti je prisutno na svakoj od namenskih jedinica. One komuniciraju preko RTE-a i MW-a tokom čega vrše proces prikupljanja i procesiranja podataka. Jedna od pomenutih komponenti direktno povezanih sa *FlexRay* magistralom je COM komponenta. Vremenska ograničenja u izvršavanju procesa i zadataka na namenskim jedinicama i sama kompleksnost sistema, uslovili su korišćenje moćnih alata koji su u stanju da odgovore potrebama testiranja ovog sistema.

Testiranje je izvršeno pomoći *VECTOR*-ovih alata, a sam testni računar je opremljen CANoe programskom podrškom takođe razvijenom od strane *VECTOR*-a. Svi ECU-ovi na mreži koji komuniciraju preko *FlexRay*-a sa ciljnom platformom su napisani u *CAPL* jeziku, a saobraćaj između njih je simuliran koristeći CANoe. Na slici 3.1 se može videti da su ECU-ovi predstavljeni kao mrežni čvorovi. Kompletna komunikacija (razmena poruka i signala) između njih se zasniva na informacijama koje se nalaze unutar FIBEX datoteke.

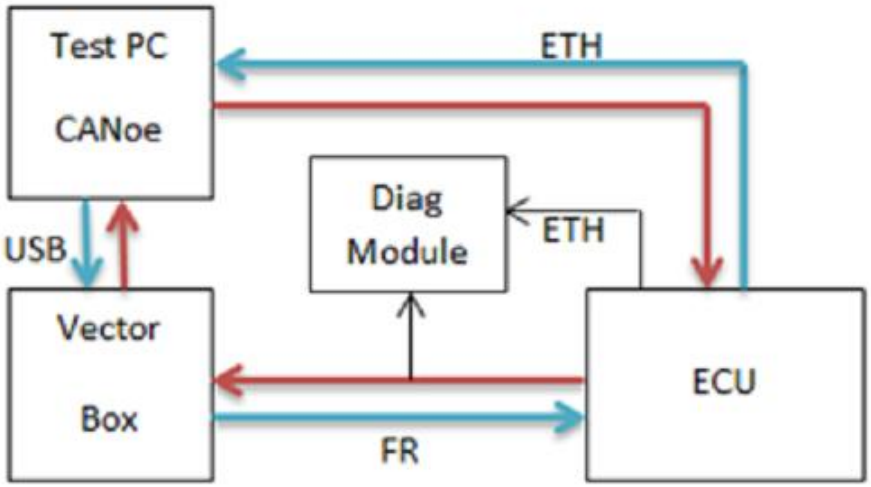
Testni čvor je takođe napisan u *CAPL*-u i povezan je na *FlexRay* magistralu zajedno sa ostalim čvorovima. U stanju je da simulira ili potpuno zameni saobraćaj koji šalju ostali čvorovi na mreži. Ovom postavkom postignuta je puna kontrola saobraćaja koji ide ka ciljnoj platformi. U skladu sa tim, postoji mogućnost umetanja širokog spektra grešaka.



Slika 3.1 Mrežni čvorovi simulirani pomoću CANoe-a

Uz sve ovo, do izražaja dolazi i jedna od prednosti CANoe alata, a to je sposobnost praćenja reakcija sistema, snimanje celokupnog saobraćaja i njegovo ponovno puštanje da bi bile ponovljene tačno one okolnosti koje su se dešavale tokom izvršavanja testova.

Slika 3.2 prikazuje izgled testnog okruženja, uređaje koji su prisutni u okruženju i konekcije između njih. Tok informacija kreće od testnog računara, preko *VN BOX*-a, odakle se *FlexRay* frejmovi šalju ka ciljnoj platformi. Krug se završava odgovorom ciljne platforme koji stiže na testni računar preko Ethernet-a. Protok informacija može da se takođe odvija i u suprotnom smeru.



Slika 3.2 Izgled testnog okruženja

Na slici se još može videti i dijagnostički modul koji je zadužen za detekciju i prijavljivanje grešaka. Modul je povezan na *FlexRay* magistralu sa jedne, a sa druge strane preko Ethernet-a sa komponentama koje se nalaze na namenskim jedinicama ciljne platforme. Konfigurisan je na osnovu ulaznih vrednosti koje se nalaze u dijagnostičkoj tabeli. Ova tabela sadrži spisak svih grešaka koje mogu da se pojave u sistemu. Svaka greška je mapirana na specifičan DTC (eng. *Diagnostic Trouble Code*). Pored ovoga, tabela sadrži i vremena za registrovanje greške (eng. *debouncing*) i prioritete za prijavljivanje DTC-ova. Prioriteti se koriste za međusobno isključivanje DTC-ova. To znači da ukoliko je prijavljen DTC višeg prioriteta, sprečiće se prijavljivanje DTC-a koji ima niži prioritet. Na primer: ako je prijavljen kratak spoj *FlexRay* primopredajnika ili globalni *FlexRay timeout* (nema *FlexRay* komunikacije), to će sprečiti prijavljivanje ostalih DTC-ova nižeg prioriteta koji se odnose na *FlexRay* greške (CRC greške, prestanak rada čvora na mreži itd...).

Korišćenjem dijagnostičkog alata prisutnog u CANoe, moguće je poslati zahtev iz testnog čvora ka dijagnostičkom modulu sa ciljem provere trenutnog stanja testiranog sistema. Alat ima opciju automatskog osvežavanja svake sekunde i u stanju je da svaku iteraciju zajedno sa trenutkom provere snimi u zaseban izveštaj u obliku tekstualne datoteke.

3.1 Tipovi testova

Jedan testni slučaj sadrži instrukcije za izvršavanje i proveru određene sprege u CANoe-u. Pod ovim se podrazumeva okidanje *FlexRay* sprege da bi se proverila reakcija željene softverske komponente. Aktivacija i provera veza između testnog računara i komponenti na namenskim jedinicama ostvarene su korišćenjem AUTOSAR-ovih API (eng. *Application programming interface*) sprega:

- Rte_Write
- Rte_IWrite
- Rte_WriteRef
- Rte_Send
- Rte_Read
- Rte_IRead
- Rte_Receive
- Rte_Call

Komplet testova (eng. *Test suite*) predstavlja logičko raspoređivanje testnih slučajeva koji, ovako grupisani, sačinjavaju listu testova. CANoe koristi ovu listu za kontrolu izvršavanja testova. Testna lista predstavlja jedan CSV (eng. *Comma Separated Values*) fajl i ona se zajedno sa testnim čvorom koristi za definisanje koraka izvršavanja testnog slučaja i očekivanih rezultata. Svaki testni slučaj u listi predstavlja validnu konekciju između simuliranih čvorova na *FlexRay* mreži i softverskih komponenti na namenskim jedinicama.

Razlika je napravljena između dolazećih i odlazećih *FlexRay* signala. Smer signala se gleda u odnosu na ciljnu platformu.

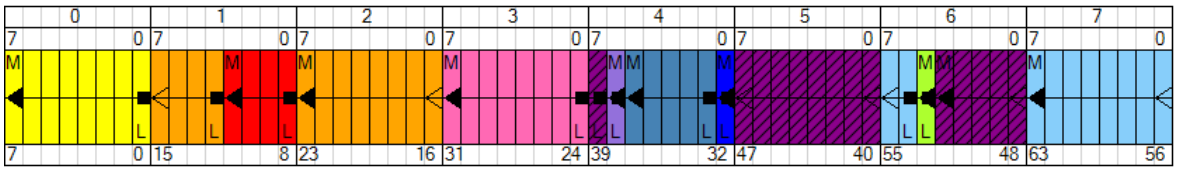
1) Odlazeći *FlexRay* testovi (crvena linija na slici 3.2):
   - testni računar šalje kontrolnu poruku preko Ethernet-a, kroz RTE do softverske komponente, zahtevajući test.
   - softverska komponenta modifikuje sadržaj PDU-a (promeni vrednost signala ili dela signala) i šalje ga preko RTE-a do COM komponente. Beleži se trenutak slanja i on se u kontrolnoj poruci šalje natrag na testni računar.
   - COM komponenta šalje PDU preko FR ka ostatku mreže simulirane od stane CANoe-a. Vrednost signala poslatog u PDU-u se prvobitno detektuje na *FlexRay* magistrali. Zabeleži se trenutak detekcije i pošalje se na testni računar. Nakon toga, ostatak mreže takođe detektuje promenjeni signal i obavesti testni računar o prijemu podataka zajedno sa trenutkom detekcije.
   - bez obzira na rezultat, testni računar će nastaviti sa izvršavanjem sledećeg koraka koji je definisan u testnoj sekvenci.
2) Dolazeći *FlexRay* testovi (plava linija na slici 3.2):
   - testni računar šalje modifikovani PDU sa izmenjenim signalom ili delom signala, preko *FlexRay* magistrale do COM komponente
   - poslati PDU se prvo detektuje na *FlexRay* magistrali. Zabeleženi trenutak detekcije se šalje na testni računar. Nakon toga, modifikovani PDU je detektovan od strane COM komponente koja šalje kontrolnu poruku sa trenutkom detekcije na testni računar.
   - COM komponenta prosleđuje modifikovani PDU preko RTE-a do komponente kojoj je namenjen. Testni računar se obaveštava o trenutku slanja PDU-a od strane COM komponente.
   - po prijemu PDU-a, softverska komponenta šalje vreme prijema na testni računar i jedinstveni identifikator, u slučaju da ima više od jedne komponente koje primaju modifikovani PDU.

- testni računar proverava da li su tačne vrednosti signala primljene od strane komponenti kojima su namenjene .
- bez obzira na rezultat, testni računar će nastaviti sa izvršavanjem sledećeg koraka koji je definisan u testnoj sekvenci.

Modifikacija sadržaja PDU-a u ovim testovima se svodi na menjanje polaznog signala ili njegovih delova i zatim se vrši konverzija iz vrednosti signala na magistrali (eng. *raw value*) u vrednost signala koja je primljena u softverskoj komponenti (eng. *physical value*) i obratno, shodno prema sledećoj formuli:

$$physical\ value = raw\ value * 2 + offset$$

Izgled PDU-a koji se modifikuje prikazan je na slici 3.3. Prikazani PDU je veličine osam bajtova. Signali unutar njega su veličine od jednog do četrnaest bita (jedna boja predstavlja jedan signal, biti u signalu ne moraju stajati jedan do drugog). Kao što je već ranije pomenuto, čak i signal koji zauzima jedan bit može biti sačinjen od više delova i svi oni podležu modifikaciji i proveri. Testni čvor sadrži listu očekivanih vrednosti signala nakon modifikacije koja se koristi za proveru primljenih vrednosti na magistrali odnosno u softverskoj komponenti. Na osnovu ove provere donosi se odluka da li je testni segment bio uspešan ili ne.



Slika 3.3 Izgled *FlexRay* PDU-a

Specifičan slučaj odlazećih *FlexRay* testova kada se vrednost signala modifikuje tako da nakon konverzije vrednost signala bude izvan dozvoljenog opsega. U tom slučaju, biće prijavljen DTC koji signalizira grešku u konverziji. Ukoliko je modifikovani signal deo E2E zaštićenog PDU-a, ovaj PDU neće biti prosleđen od strane COM komponente. Ako je pak signal deo običnog PDU-a, COM komponenta će poslati PDU sa originalnim, nepromenjenim podacima.

Pojavi li se bilo kakva greška (neispravan sadržaj *FlexRay* poruke, pogrešan CRC, poruka nije primljena u očekivanom vremenskom okviru) tokom izvršavanja testova, softverske komponente će istog trenutka obavestiti dijagnostički modul i određeni DTC će biti prijavljen i

smešten u dijagnostički izveštaj. PDU-ovi neispravnog sadržaja se neće prosleđivati od strane COM komponente ka ostalim komponentama u cilju sprečavanja širenja grešaka kroz sistem.

Testiranje E2E PDU-ova se uglavnom bazira na umetanju grešaka (CRC, brojač, TimeOut) i iščekivanju reakcije od strane softverske komponente na njih. Za umetanje ovih grešaka koriste se sledeće funkcije napisane u *CAPL*-u:

- FCTManipulateE2E - koja ubacuje grešku u CRC ili grešku u brojač u zavisnosti od parametra koji joj je prosleđen
- FCTTimeoutControl - koja unosi TimeOut grešku u E2E PDU

Testni slučaj je uspešno izvršen ukoliko softverska komponenta koja prima E2E PDU detektuje grešku u istom i prijavi odgovarajući DTC dijagnostičkom modulu. Po završetku izvršavanja testova, CANoe generiše izveštaj u obliku *.html* fajla. Izveštaj sadrži detaljan tok izvršavanja testova iz koga može da se vidi, za svaki pojedinačni korak, da li je bio uspešan ili nije.

4. Kontrola izvršavanja testova

Prilikom automatizacije testova u CANoe/*CAPL* okruženju, koriste se dva kontrolna fajla. Prvi kontrolni fajl je *FR_Test.ini*. Sadrži inicijalne vrednosti koje je neophodno učitati u CANoe pre pokretanja samih testova. Direktno je povezan sa CSV listom. Promena njegovih parametara utiče na odabir i tok izvršavanja testova.

poglavlje	parametar	vrednost	komentar
Platforma	Broj namenskih jedinica	A	A=1
		B	B=2
		C	C=3
		D	D=4
Testni Modul	Skraćenica	FR	
	Podlista	FROut	
		FRinc	
		E2E	
Podešavanja	Vreme izvršavanja	Minimum	Minimalno vreme izvršavanja *runnable*-a
		70%	70% vremena izvršavanja *runnable*-a
		80%	80% vremena izvršavanja *runnable*-a
		90%	90% vremena izvršavanja *runnable*-a
		100%	100% vremena izvršavanja *runnable*-a
Izgled izveštaja	Dodati datum i vreme	Da	Dodaje datum i vreme u naziv izveštaja
		Ne	Naziv izveštaja bez datuma i vremena

Tabela 4.1 Izgled *FR_Test.ini* fajla

4.1 CSV lista

Drugi fajl je pomenuta CSV lista. Ova lista se sastoji od jedne ili više komandi koje su razdvojene tačka-zarezom. Svaka linija počinje sa testnom komandom za korišćenje sprege ili nekom naredbom (eng. *keyword*) sa dodatnim parametrima u zavisnosti od upotrebljene naredbe.

```
__Label__;TestGroup_FlexRayOutgoing;
FR.FOut-001;ABCD;10;1;0.Channel_A-15.AFR_PDU;1;0.Host_1-7.SWCeABCpreRE-1.RApeABCpreREMain-6.iwrite-2.Rte_IWrite_RApeABCpreREMain_PpFReABCpreview
FR.FOut-002;A;960;1;0.Channel_A-24.ARA2_PDU;1;0.Host_1-22.SWCParkControl-130.RParkControlLogic20ms-6.iwrite-5.Rte_IWrite_RParkControlLogic20ms_P
FR.FOut-002;B;960;1;0.Channel_A-24.ARA2_PDU;1;0.Host_1-22.SWCParkControl-130.RParkControlLogic20ms-6.iwrite-5.Rte_IWrite_RParkControlLogic20ms_P
FR.FOut-002;CD;960;1;0.Channel_A-24.ARA2_PDU;1;0.Host_1-22.SWCParkControl-130.RParkControlLogic20ms-6.iwrite-5.Rte_IWrite_RParkControlLogic20ms_
FR.FOut-003;CD;100;1;0.Channel_A-25.ARA2_PDU;1;2.Host_3-32.SWCTrailerManeuverAssistIP-1.RTrailerManeuverAssistIPMain-4.send-1.Rte_Send_PpFRTVTAI
FR.FOut-004;A;100;1;0.Channel_A-26.ARA2_PDU;1;0.Host_1-22.SWCParkControl-130.RParkControlLogic20ms-6.iwrite-5.Rte_IWrite_RParkControlLogic20ms_P
FR.FOut-004;B;100;1;0.Channel_A-26.ARA2_PDU;1;0.Host_1-22.SWCParkControl-130.RParkControlLogic20ms-6.iwrite-5.Rte_IWrite_RParkControlLogic20ms_P
FR.FOut-004;CD;100;1;0.Channel_A-26.ARA2_PDU;1;0.Host_1-22.SWCParkControl-130.RParkControlLogic20ms-6.iwrite-5.Rte_IWrite_RParkControlLogic20ms_
FR.FOut-005;A;40;1;0.Channel_A-71.BV2_PDU;1;0.Host_1-36.SWCEyeQCom-2.REyeQCom40ms-6.iwrite-2.Rte_IWrite_REyeQCom40ms_PpFREyeQComOut_DeLanesMV;2;
FR.FOut-005;BCD;40;1;0.Channel_A-71.BV2_PDU;1;0.Host_1-36.SWCEyeQCom-2.REyeQCom40ms-6.iwrite-2.Rte_IWrite_REyeQCom40ms_PpFREyeQComOut_DeLanesMV;
#;
__Label__;TestGroup_FlexRayIncoming;
FR.FInc-001;ABCD;80;1;0.Channel_A-20.APS_PDU;1;0.Host_1-221.SWCComSafeB-1.RComSafeBRX0-6.iwrite-1.Rte_IWrite_RComSafeBRX0_PpFRACFIncn_DeFRACFInc
FR.FInc-002;ABCD;80;1;0.Channel_A-21.APS_PDU;1;0.Host_1-221.SWCComSafeB-1.RComSafeBRX0-6.iwrite-1.Rte_IWrite_RComSafeBRX0_PpFRACFIncn_DeFRACFInc
FR.FInc-003;ABCD;80;1;0.Channel_A-22.APS_PDU;1;0.Host_1-221.SWCComSafeB-1.RComSafeBRX0-6.iwrite-1.Rte_IWrite_RComSafeBRX0_PpFRACFIncn_DeFRACFInc
FR.FInc-004;ABCD;80;1;0.Channel_A-23.APS_PDU;1;0.Host_1-221.SWCComSafeB-1.RComSafeBRX0-6.iwrite-1.Rte_IWrite_RComSafeBRX0_PpFRACFIncn_DeFRACFInc
FR.FInc-005;ABCD;1000;1;0.Channel_A-27.ARA_PDU;1;0.Host_1-221.SWCComSafeB-1.RComSafeBRX0-6.iwrite-1.Rte_IWrite_RComSafeBRX0_PpFRACFIncn_DeFRACFI
FR.FInc-006;ABCD;1000;1;0.Channel_A-28.ARA_PDU;1;0.Host_1-204.SWCCom-1.RComRX0-6.iwrite-13.Rte_IWrite_RComRX0_PpFRParkControlin_DeFRParkControli
FR.FInc-007;A;100;1;0.Channel_A-29.ARA_PDU;7;0.Host_1-204.SWCCom-1.RComRX0-4.send-4.Rte_Send_PpFRTVMasterIn_DeFRTVMasterIn;0.Host_1-204.SWCCom-1
FR.FInc-007;B;100;1;0.Channel_A-29.ARA_PDU;7;0.Host_1-204.SWCCom-1.RComRX0-4.send-4.Rte_Send_PpFRTVMasterIn_DeFRTVMasterIn;0.Host_1-204.SWCCom-1
FR.FInc-007;CD;100;1;0.Channel_A-29.ARA_PDU;7;0.Host_1-204.SWCCom-1.RComRX0-4.send-4.Rte_Send_PpFRTVMasterIn_DeFRTVMasterIn;0.Host_1-204.SWCCom-
```

Slika 4.1 Izgled CSV liste

Testni slučajevi iz testne grupe *FlexRayOutgoing* su sastavljeni po sledećem principu:

- *FR.FOut - <broj_testa>;<broj_namenskih_jedinica>;<trajanje>;<FR_PDU>; <sprega_koja_šalje_PDU>;<sprega_koja_prima_PDU>; <broj_modifikovanih_signala>;*

Sprega koja šalje PDU u ovom slučaju je deo od softverske komponente prisutne na jednoj od namenskih jedinica. Sprega koja prima PDU je deo COM komponente.

Testni slučajevi iz testne grupe *FlexRayIncoming* su sastavljeni po sledećem principu:

- *FR.FInc - <broj_testa>;<broj_namenskih_jedinica>;<trajanje>;<FR_PDU>; <sprega_koja_šalje_PDU>;<sprega_koja_prima_PDU>; <broj_modifikovanih_signala>;*

Sprega koja šalje PDU u ovom slučaju je deo COM komponente. Ostale sprege koje primaju PDU su deo softverskih komponenti na namenskim jedinicama.

4.2 Najčešće korišćene naredbe

Naredbe predstavljaju dodatke testnim komandama i koriste se za kontrolu toka izvršavanja testova.

4.2.1 *Repeat*

Sintaksa:

__RepeatTC__;<broj_ponavljanja_navedenih_testnih_slučajeva>;

Upotreba:

Svaki navedeni tesni slučaj će se izvršiti onoliko puta kolko je zadato u ovoj naredbi.

4.2.2 Definisanje labele

Sintaksa:

__Label__;<ime_labele>;

Upotreba:

Definisanje imena labele koja se može koristiti za grupisanje testnih slučajeva kao i za naredbe skoka i petlje ponavljanja.

4.2.3 *GoTo* naredba

Sintaksa:

__GoTo__;<ime_labele>;

Upotreba:

Koristi se za skakanje na navedenu labelu.

4.2.4 *Loop* naredba

Sintaksa:

__LoopBegin__;<ime_labele>;<broj_ponavljanja_testne _sekvence>;

__LoopEnd__;<ime_labele>;

Upotreba:

Koristi se da se sekvenca odabranih testova izvrši zadati broj puta.

4.2.5 Naredba *Reset Data*

Sintaksa:

__ResetData__;<broj_sprega>;[<sprege>;]

Upotreba:

Ponovno pokretanje željenih sprega u cilju vraćanja u početno stanje nakon testova pod punim opterećenjem.

4.3 Funkcije za unošenje grešaka

Funkcije koje se koriste za unošenje grešaka su implementirane u okviru testnog čvora. One se pozivaju nakon promene vrednosti sistemske promenljive upotrebom ugrađene CANoe funkcije za obradu događaja - *on sysvar_update*.

Pozivaju se sa sledećim parametrima:

- *FCTManipulate(char PDUName[], char NodeName[], enum E2EDist DistType, word NofDist)* gde su:
  - *PDUName* -> ime PDU-a u koji unosimo grešku
  - *NodeName* -> ime čvora koji šalje PDU
  - *DistType* -> vrsta greške koja se unosi(CRC ili brojač)
  - NofDist -> broj uzastopnih ciklusa u kojima je ubačena greška

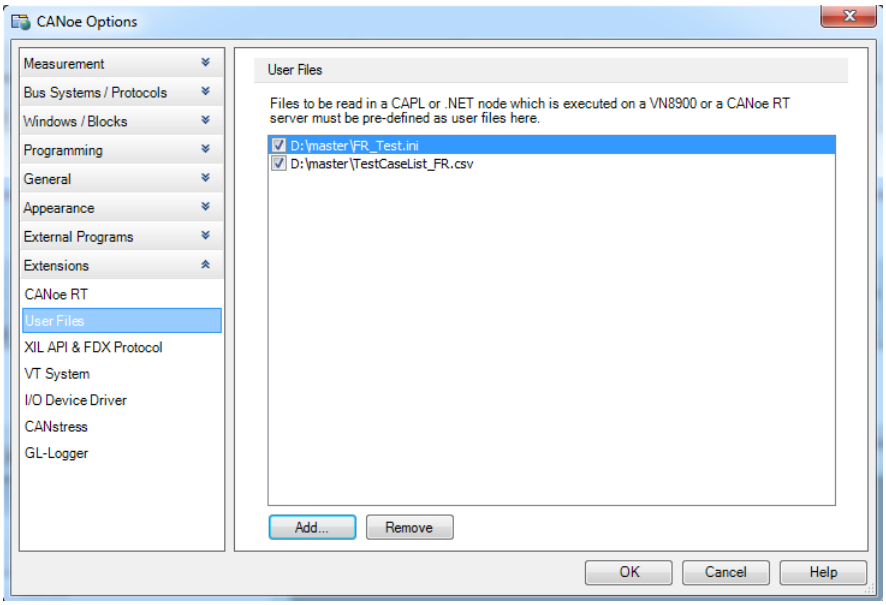
- *fctTimeOutControl(char PDUName[], char NodeName[], byte ActiveTO, word TOtime)* gde su:
  - *PDUName* -> ime PDU-a u koji unosimo grešku
  - *NodeName* -> ime čvora koji šalje PDU
  - *ActiveTO* -> aktivirano/deaktivirano unošenje *timeout*-a
  - *TOtime* -> vreme timeout-a u mili sekundama

Nakon pozivanja pomenutih funkcija, sledi provera prijavljenih DTC-ova slanjem dijagnostičkog zahteva prema dijagnostičkom modulu.

5. Ispitivanje i rezultati

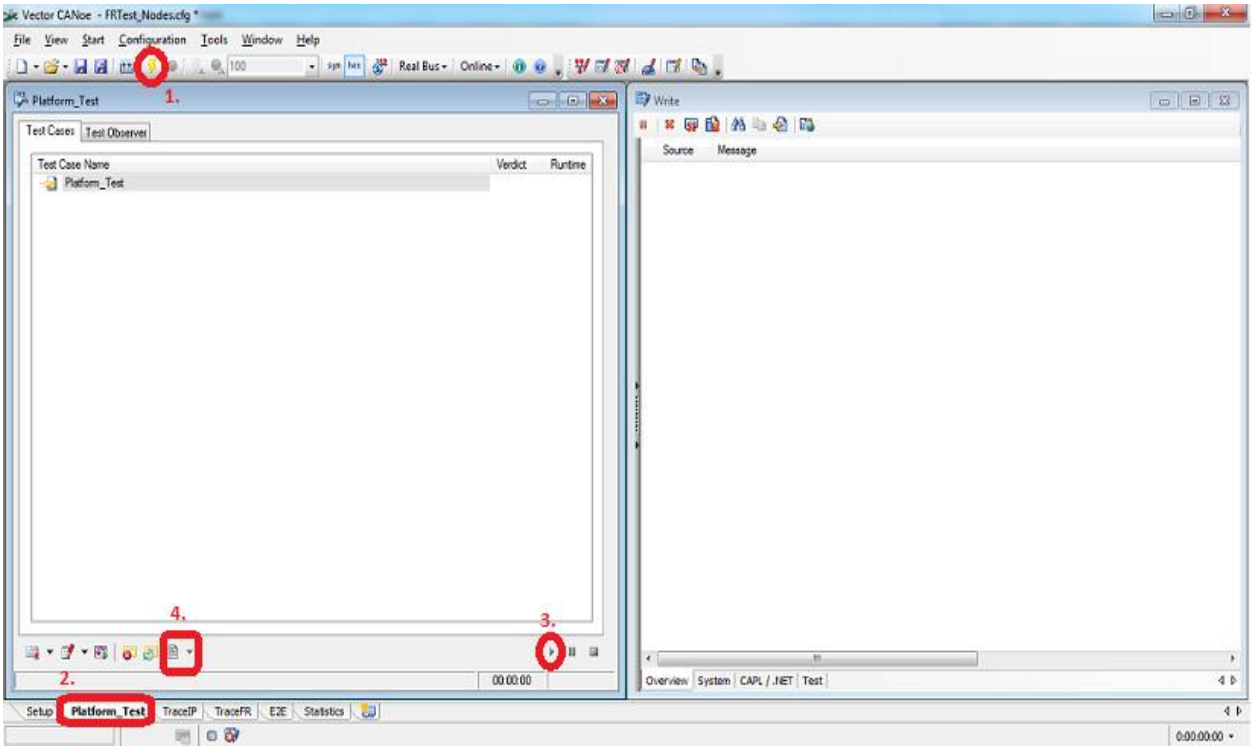
Postupak ispitivanja izvršen je upotrebom *VECTOR*-ovog okruženja koje čine CANoe programska podrška instalirana na testnom računaru i VN BOX 7600 sa pratećim licencama za *FlexRay* i *Ethernet*.

Pri pokretanju same testne CANoe konfiguracije, prvenstveno se moraju učitati odgovarajuće datoteke. Vrši se odabir željenog FIBEX fajla i dijagnostičke tabele neophodne za ispravno funkcionisanje testnog čvora i dijagnostičkog modula. Potrebno je takođe učitati i konfiguracione fajlove za kontrolu i odabir testova koji su opisani u prethodnom poglavlju. Učitavanje konfiguracionih fajlova u CANoe se vrši klikom na *Configuration* -> *Options* -> *Extensions* i odabrati opciju *User Files*. Klikom na dugme *Add...* dodajemo FR_Test.ini i CSV listu.



Slika 5.1 Pravilno učitani konfiguracioni fajlovi

Prevođenje (eng. *compile*) *CAPL* koda i pokretanje CANoe konfiguracije vrši se klikom na ikonicu munje (označeno sa 1. na slici 5.2). Ukoliko je prevođenje prošlo bez grešaka, može se pristupiti pokretanju testova. Klikom na tab Platform_Test (2. na slici 5.2) i zatim na *play* (3. na slici 5.2) startujemo izvršavanje testova. U prikazanim prozorima konfiguracije moguće je pratiti trenutno izvršavanje testova.



Slika 5.2 CANoe konfiguracija

Po završetku testova, klikom na dugme označeno sa 4. na slici iznad, otvara se generisani HTML izveštaj u predefinisanom internet pretraživaču. Pored uobičajenih informacija o testiranoj platformi i testnom okruženju, može se videti uopšten prikaz rezultata izvršenih testova.

Test Case Results

1	FR.Init-001 ~ Host_1(28) /	pass
2	FR.Init-002 ~ Host_2(21) /	pass
3	FR.Init-003 ~ Host_3(14) /	pass
4	Setup ~ StartMeasure(65)	pass
5	FR.FOut-001 ~ Host_1 / iwrite --> AFR_PDU	pass
6	FR.FOut-002 ~ Host_1 / iwrite --> ARA2_PDU	pass
7	FR.FOut-003 ~ Host_3 / send --> ARA2_PDU	fail
8	FR.FOut-004 ~ Host_1 / iwrite --> ARA2_PDU	pass

Slika 5.3 Uopšten prikaz rezultata

Klikom na bilo koji od izvršenih testova dobijamo detaljan prikaz testnog slučaja od interesa. Označeno je ukupno vreme trajanja testnog slučaja, svi njegovi koraci kao i presude o uspešnosti svakog od njih.

[−] 5 Test Case : FR.FOut-001 ~ Host_1 / iwrite --> AFR_PDU: Passed

Snd ITF (10ms) : 0.Host_1 -- 7.SWCeABCpreRE -- 1.RApeABCpreREMain -- 6.iwrite -- 1.Rte_IWrite_RApeABCpreREMain_PpFReABCpreviewRoadout_DeFReABCpreviewRoadout
Rcv ITF (80ms) : 0.Host_1 -- 204.SWCCom -- 17.RComTX0 -- 3.iread -- 34.Rte_IRead_RComTX0_PpFReABCpreviewRoadout_DeFReABCpreviewRoadout
FR pdu (10ms) : 0.Channel_A -- 15.AFR_PDU

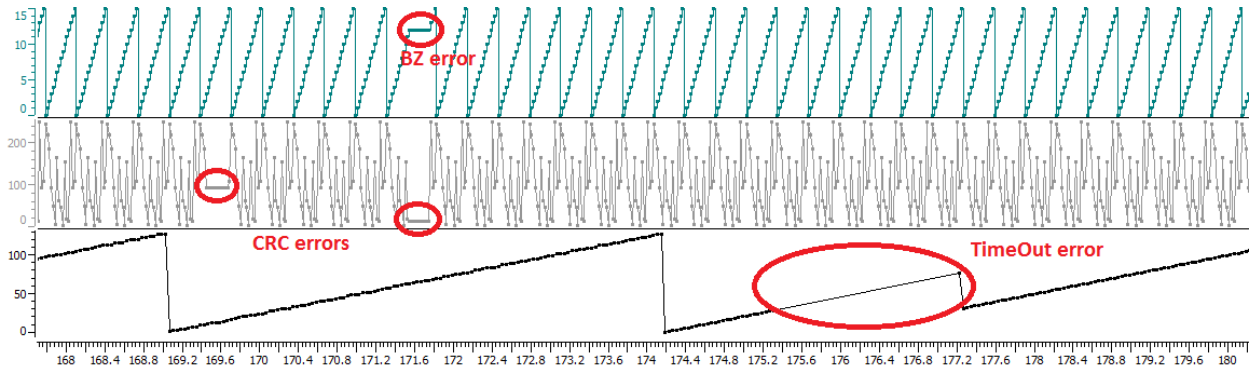
Test case begin: 2016-06-23 01:48:36 (logging timestamp 156.232429)
Test case end: 2016-06-23 01:48:40 (logging timestamp 160.100721)

Main Part of Test Case

Timestamp	Test Step	Description	Result
156.232429		- ----- ----- ----- AFR_PDU_Str_Hoehe_HL ----- ----- number of tuples: 4 ----- ----- ----- -	-
156.232429	PC	Control: message sent to ITF 0.Host_1-7.SWCeABCpreRE-1.-6.iwrite-1., value (0xFFFFFF01 [-255]), conversion <i>	pass
156.240467	ITF	Response: expected status (0x00 [OK]) from ITF 0.Host_1-7.SWCeABCpreRE-1.-6.iwrite-1.	pass
156.312429	Resume reason	Elapsed time=80ms (max=80ms)	-
156.320779	Resume reason	Resumed on FlexRay PDU: Slot='23', Cycle='30', Channel Mask='A', Channel='1' Elapsed time=8.3495ms (max=15ms)	-
156.320779	FR	Response: value (0x00000000 [0]) received in SIG 0.Channel_A-15.AFR_PDU-0.AFR_PDU_Str_Hoehe_HL	pass
156.320779	Snd ITF	Acknowledge: correct number of messages (1) from ITF 0.Host_1-7.SWCeABCpreRE-1.-6.iwrite-1.	pass
156.320779		----- ----- ----- ----- ----- -----	-
156.320779	PC	Control: message sent to ITF 0.Host_1-7.SWCeABCpreRE-1.-6.iwrite-1., value (0x00000000 [0]), conversion <i>	pass
156.333262	ITF	Response: expected status (0x00 [OK]) from ITF 0.Host_1-7.SWCeABCpreRE-1.-6.iwrite-1.	pass
156.400779	Resume reason	Elapsed time=80ms (max=80ms)	-
156.410778	Resume reason	Resumed on FlexRay PDU: Slot='23', Cycle='48', Channel Mask='A', Channel='1' Elapsed time=9.99869ms (max=15ms)	-
156.410778	FR	Response: value (0x000000FF [255]) received in SIG 0.Channel_A-15.AFR_PDU-0.AFR_PDU_Str_Hoehe_HL	pass
156.410778	Snd ITF	Acknowledge: correct number of messages (1) from ITF 0.Host_1-7.SWCeABCpreRE-1.-6.iwrite-1.	pass
156.410778		----- ----- ----- ----- ----- -----	-
156.410778	PC	Control: message sent to ITF 0.Host_1-7.SWCeABCpreRE-1.-6.iwrite-1., value (0x000000FF [255]), conversion <i>	pass
156.420203	ITF	Response: expected status (0x00 [OK]) from ITF 0.Host_1-7.SWCeABCpreRE-1.-6.iwrite-1.	pass

Slika 5.4 Detaljan prikaz testnog slučaja

Testiranje E2E zaštićenih PDU-ova može se ispratiti u grafičkom prozoru CANoe-a i lako je uočiti da su pozvane funkcije uspešno unele greške u testirani PDU.



Slika 5.5 Unos grešaka u E2E zaštićeni PDU

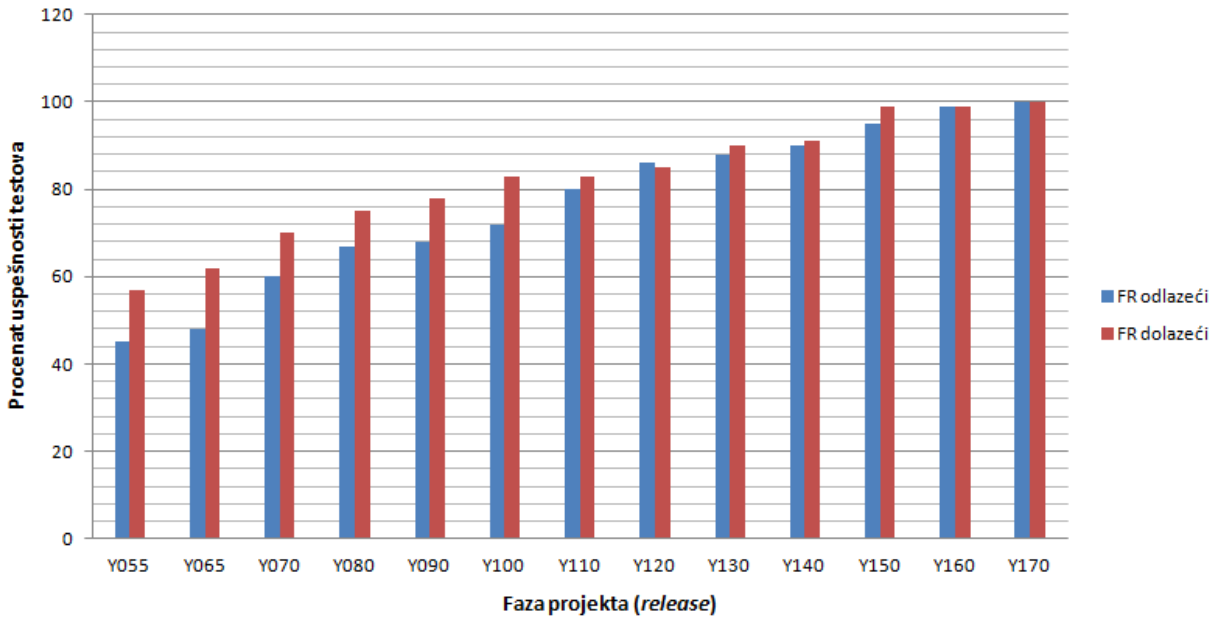
Izazvane greške su uspešno registrovane od strane softverskih komponenata i prijavljene su dijagnostičkom modulu.

DTC	Description	Status
90051a	BRS_BU_02_BZ	Warning Indicator Off : Test Completed Since Last Clear : Confirmed DTC : ...
90051c	BRS_BU_02_CRC	Warning Indicator Off : Test Completed Since Last Clear : Confirmed DTC : ...
900422	TIMEOUT_Airbag_05	Warning Indicator Off : Test Completed Since Last Clear : Confirmed DTC : ...

Slika 5.6 Dijagnostički izveštaj sa potvrđenim DTC-ovima

Testovi su vršeni tokom čitave faze razvoja ciljne platforme. Svaka faza razvoja savremenih platformi, pre nego što dođe do finalne verzije, sastoji iz više iteracija (eng. *release*). Pokretanje testova je vršeno nakon svake iteracije (često i više puta po iteraciji) da bi se izvršila verifikacija rada svih softverskih komponenti u sistemu. Kroz proces formalne recenzije testova, utvrđeno je da rezultati testova odgovaraju očekivanom ponašanju sistema traženog u specifikacijama. Nakon formalne recenzije, svaka promena u implementaciji funkcionalnosti na ciljnoj platformi kroz iteracije uticala je na prolaznost testova, što je još jedan od pokazatelja da su testovi propisno implementirani.

Procenat uspešnosti testova po fazama razvoja projekta prikazan je na slici 5.7.



Slika 5.7 Procenat uspešnosti po fazama razvoja projekta

6. Zaključak

U radu su predstavljeni precizni i opširni testovi koji ispituju sve vrednosti *FlexRay* signala u namenskom sistemu. Sistematskim razvojem testova uspešno su izbegnuti problemi slabe pokrivenosti i dužine trajanja samih testova koji su karakteristični za ovako složene sisteme.

Njihovim izvršavanjem povećali smo stabilnost i pouzdanost testiranog sistema. Generisani izveštaji obezbedili su detaljne rezultate i doveli do zaključka da je svaki deo komunikacije propisno testiran i da radi kako treba. Korišćenjem dijagnostičkog izveštaja o greškama i snimljenog saobraćaja, može se utvrditi tačan momenat i okolnosti pod kojim se prijavljena greška dogodila, što značajno skraćuje vreme otkrivanja uzroka greške.

Prikazano rešenje ima sve elemente efikasnog i kvalitetnog testnog okruženja jer poseduje potpuno automatsko izvršavanje testnih slučajeva i generisanje izveštaja, kontrolu nad izvršavanjem i tokom testova kao i neograničenu mogućnost njihovog ponavljanja. Takođe se lako može proširiti ukoliko testirana platforma ima veći broj namenskih jedinica ili softverskih komponenti.

Jedan od pravaca za proširenje prikazanog rešenja je dodavanje funkcionalnosti dijagnostičkom modulu koji bi na zahtev, bio u stanju da pruži još neke vrednosti karakteristične za trenutno stanje sistema osim aktivnih grešaka. U bliskoj budućnosti trebalo bi se fokusirati na optimizaciju rada i na merenje vremena oporavka sistema nakon prisustva umetnutih grešaka, kao i na prilagođavanje testova ostalim komunikacionim protokolima (*CAN*).

7. Literatura

[1] Navet N, Simonot-Lion F, "Automotive Embedded Systems Handbook", *Industrial Information Technology Series*, CRC press, December 2008.

[2] Albert A, "Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems" , *Embedded World*, Nurnberg, February 2004.

[3] FlexRay Consortium. FlexRay Communications System – Protocol Specification V3.0.1, October 2010.

[4] dSPACE. Tools for controller development, http://www.dspace.com

[5] Stroop J, Stolpe R, "Prototyping of Automotive Control Systems in a Time-Triggered Environment Using FlexRay", February 2006.

[6] Embedded Systems Course ,"Lesson 19 - An introduction to AUTOSAR", Real-Time Systems Laboratory, https://retis.sssup.it/

[7] Schreiner D, "Component Based Communication Middleware for AUTOSAR", *Institute of Computer Languages, Compilers and Language Group at Vienna University of Technology*, Austria, October 2009.

[8] MICROSAR Product Information, VECTOR, v2.5, May 2017

[9] Armengaud E, Steininger A, Horauer M, „Efficient stimulus generation for testing embedded distributed systems – The FlexRay example", Conference Paper, October 2005.

[10] Lari S, Dehbashi M, Miremadi S. G, Amiri M, "Evaluation of Babbling Idiot Failures in FlexRay-Based Networks", *Sharif University of technology*, Teheran, Iran, 2007.

[11] Somers B, "Investigation of a FlexRay - CAN Gateway in the Implementation of Vehicle Speed Control", *Department of Engineering Technology of Waterford Institute of Technology*, June 2009.

[12] Armengaud E, Steininger A, Horauer M, "Concepts and Tools for the Test of the Communication Sub-System of Time-Triggered Distributed Embedded Systems", Article, January 2007.

[13] Pop, T, P. Pop, P. Eles, Z. Peng, and Timing A, “Analysis of the FlexRay Communication Protocol”, *Proc. of the 18th Euromicro Conference on Real-Time System*, pp. 203-216, 2006.