УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Момир Милановић

Једно решење софтверког модула за контролу приступа садржајима дигиталне телевизије на CAM модулу

МАСТЕР РАД

Нови Сад, 2012

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**:		Момир Милановић
Ментор, **МН**:		Проф др Вељко Малбаша
Наслов рада, **НР**:		Једно решење софтверког модула за контролу приступа садржајима дигиталне телевизије преко CAM модула.
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2012
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)		7/47/0/8/11/0
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кqучне речи, **ПО**:		контрола приступа, CAM, модул, , дигитална телевизија
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У раду је представљено решење софтверског модула који контролише пристуо заштићеним садржајима, када се контрола приступа врши преко CAM модула. Детаљно је представљено и анализирано постојеће решење. Ново решење је дато до детаља, са образложењима. На крају су дати су резултати тестирања
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	**<име председника комисије>**
	Члан:	**<име члана комисије>** — Потпис ментора
	Члан, ментор:	**<име ментора>**

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**:		Momir Milanovic
Mentor, **MN**:		PhD Veljko Malbasa
Title, **TI**:		One solution of sofrware module for control of access to content of dogotal television over CAM module.
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**:		<godina odbrane>
Publisher, **PB**:		Author's reprint
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		<upisati statistiku>
Scientific field, **SF**:		Electrical Engineering
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, **S/KW**:		Conditiona access, CAM, module, digital television
UC		
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, **N**:		
Abstract, **AB**:		This paper presents one solution of software module which controls conditional access of digital television over CAM module. Existing solution is presented and analized with . New solution is presented in details with explanation. Results are shown at the end.
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		
Defended Board, **DB**:	President:	**<ime predsednika komisije>**
	Member:	**<ime člana komisije>**
	Member, Mentor:	**<ime mentora>**

Menthor's sign

Zahvalnost

Zahvaljujem se kompaniji RT-RK doo, Novi Sad, gde je ovaj rad realizovan, posebno kolegama koje su pomogle pri realizaciji.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

API - *Application Protocol Interface,* aplikacioni programski intefejs;

CAM - *Conditiobal Access Modul*, modul za kontrolu uslovnog pristupa;

DVB - *Digital Video Broadcasting*, digitalno emitovanje video signala;

CAPMT - Conditional Access Program Menagment Table, tabela za upravljanje uslovnim pristupom programu;

CI - Common Interface;

ECM - *Entitlement Control Message*, poruka za kontrolu prava;

EMM -*Entitlement Menagment Message*, poruka za upravljanje pravima.

1. Uvod

Modul za kontrolu pristupa sadržajima digitalne televizije je jedan od najvažnijih modula u svakom sistemu digitalne televizije na korisničkoj strani. U ovom diplomskom-master radu je predstavljeno jedno rešenje tog modula kada se kontrola pristupa vrši pomoću CAM-a (od engl. *Conditional Access Modul*). Postojeće (komercijalno) nije optimalno iz sledećih razloga:

- brzina izvršavanja i
- kompleksnost primenjenog rešenja.

U drugom poglavlju će biti reči o digitalnoj televiziji i uslovnom pristupu sadržajima. Objašneni su glavni pojmovi iz digitalne televizije, sa akcentom na one koji su bitni za ovaj rad.

U trećem je dat pregled softvera za CAM, i preciznije je objašnjena uloga modula i veze sa drugim modulima

Zatim je dat opis trenutnog rešenja ovog modula. Detaljno su opisani podmoduli i funkcionisanje taskova. Takođe, nabrojane su sve bitnije funkcije, strukture i nabrajanja kao i API prema drugim modulima.

Novo rešenje je predstavljeno u petom poglavlju. Analizirane su dobre i loše strane postojećeg rešenja. Predstavljena su i objašnjena unapređenja: taskovi, funckcije, nabrajanja, stil kodiranja.

U šestom poglavlju su predstavljneni postignuti rezultati.

Na kraju je dat zaključak.

2. Digitalna televizija i uslovni pristup

Od pojave televizije u boji pa sve do devedesetih godina prošlog veka nije bilo značajnijeg napretka u konceptu distribucije i prenosa televiziskog signala. U poslednjoj deceniji mogu se uočiti četiri glavna pravca razvoja televizije:

- povećanje dimenzija i prelazak na upotrebu ravnih ekrana:
- povećanje rezolucije slike;
- primena multimedijalnih i internet tehnologija;
- pojava digitalne televizije.

Pojavom digitalnih tehnologija njihova primena otpočinje i u domenu obrade video i audio informacija. Pojavljuju se novi mediji i formati za zapis i prenos audio/video podataka: CD audio, DVD video, internet. Digitalne tehnologije počinju da se primenjuju i u kontroli TV prijemnika, a obradu TV slike takođe počinju da kontrolišu digitalni procesori. Takođe, pojavljuje se prenos digitalnog televiziskog signala tj. digitalna televizija.

Digitalna televizija predstavlja prenos digitalnog video i audio signala kao i dodatnih informacija, što donosi i mogućnost upotrebe novih servisa. Digitalni prenos obezbeđuje bolji kvalitet slike i zvuka, koji više ne mogu biti u prenosu ometani interferencijom sa drugim signalima, bez obzira na rastojanje na koje se slika i zvuk prenose. Takođe, slika i zvuk koju digitalni signal nosi su isti kao i na izvoru emitovanja, sve dok signal ne postane toliko slab da prijem više nije moguć. Omogućen je i prenos slike i zvuka visokog kvaliteta što je bilo nezamislivo u analognoj televiziji. Digitalna televizija uvodi mogućnost uvođenja novih usluga i izvršavanje aplikacija pisanih na Java, html ili ActionScript jeziku.

Trenutno se u svetu u digitalnoj televiziji dominantno koristi MPEG2 standard koji definiše algoritme za kodovanje i dekodovanje audio/video podataka. Tokovi MPEG2 audio i video podataka koji se emituju nazivaju se elementarni tokovi (engl. *elementary streams* ES). Koduju se i prenose kao zasebni tokovi podataka. Kodovani podaci se prenose korišćenjem paketski orijentisanih protokola u kombinaciji sa odgovarajućim mehanizmima za detekciju i obradu grešaka. Da bi se na prijemnoj strani elementarni tokovi mogli povezati u logičke celine kao i da bi se obezbedila odgovarajuća sinhronizacija audio i video podataka, moraju se prenositi i dodatni podaci. Ovako kompleksna struktura prenosa podataka zahteva postojanje posebnog transportnog sloja/protokola koji se takođe definiše MPEG2 standardom ali i odgovarajućim standardima za digitalnu televiziju. Najvažniji standardi su DVB, ATSC, ISDB, DMB, DMB-T/H ili DTMB. U ovom projektu se radi o DVB standardu, koji se inače koristi u većem delu sveta i on će biti detaljnije opisan.

2.1. DVB standard

DVB standard koristi MPEG-2 / MPEG-4 AVC standarde za video kompresiju i MPEG-2 , AC3 i AAC za audio kompresiju. Osim digitalizacije signala, u DVB digitalnoj televiziji uvedene su i nove mogućnosti: elektronski programski vodič, uslovni pristup sadržaju (engl. *Conditional Access* - CA), opcioni povratni informacioni kanal za interaktivne usluge, uvođenje novih tipova paketa u transportni protokol. Omogućen je visok stepen fleksibilnosti koji ne nameće nikakva ograničenja po pitanju tipa podataka koji se mogu prenositi. Napr. jedan satelitski DVB transponder propusnog opsega 38Mbps se može koristiti za emitovanje različitih skupova usluga:

- 4-8 standardnih TV kanala (zavisno od vrste i kvaliteta programa)
- 2 HDTV kanala
- 150 radio programa
- 550 ISDN kanala podataka na 64 kbps

Prenos digitalnih TV podataka se može obavljati upotrebom različitih prenosnih tehnologija i medija. Zato su definisani posebni podstandardi za:

- satelitski prenos: DVB-S – od *DVB Satellite*;
- prenos putem digitalne kablovske mreže: DVB-C – *DVB Cable*;
- zemaljski prenos putem UHF/VHF: DVB-T – *DVB Terrestrial*;
- za prenosne uređaje kao što su mobilni telefoni: DVB-H – *DVB Handheld*.

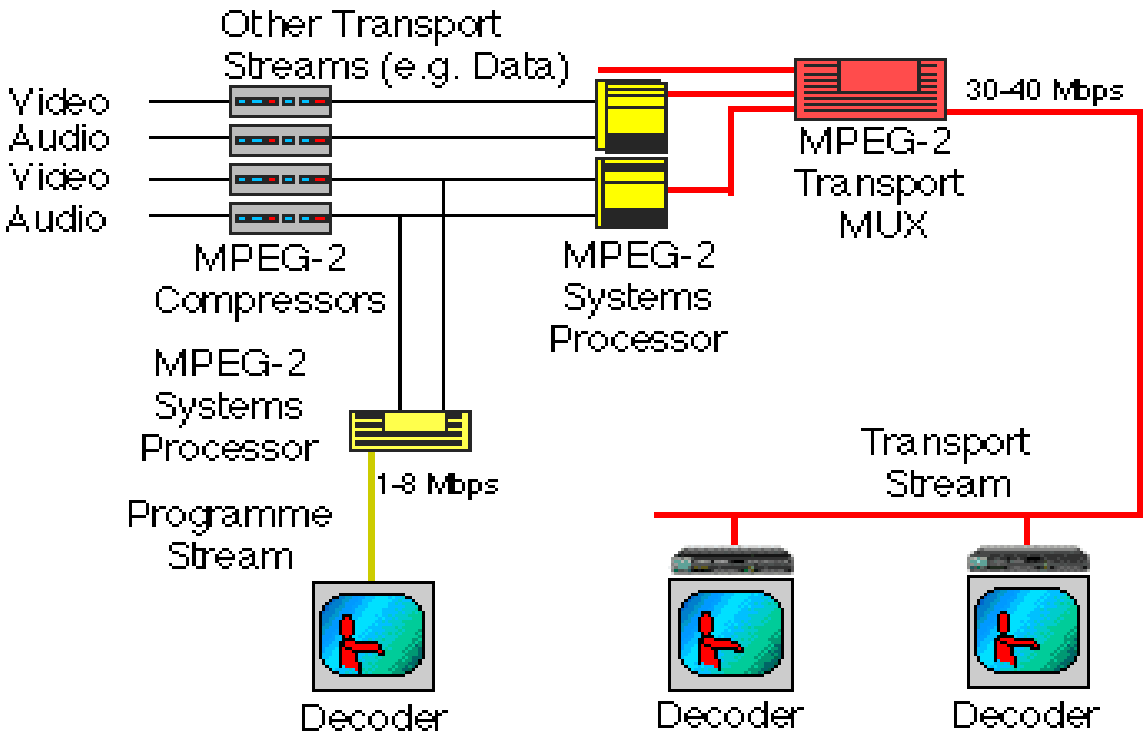
U toku je definisanje druge generacije DVB standarda, od kojih se neki već primenjuju:

- DVB-S2, DVB-T2, DVB-C2 – uvode se nove metode digitalne modulacije u cilju postizanja bolje iskorišćenosti frekventnog propusnog opsega;
- DVB-SH – definiše DTV prenos putem satelitskih veza za prenosne uređaje.

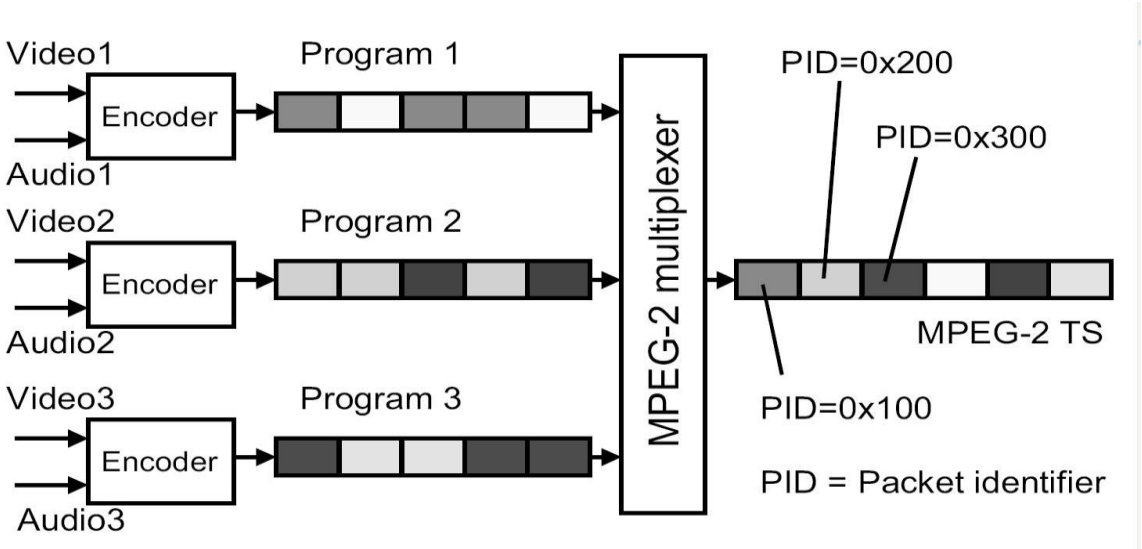
U zavisnosti od prenosnog medijuma MPEG2 standard definiše različite formate za kombinovanje MPEG2 elementarnih tokova u jedinstven informacioni tok podataka. Ukoliko prenosni medijum nije podložan pojavi grešaka u toku prenosa, MPEG2 elementarni tokovi se kombinuju u tzv. programski tok (engl *Program Stream (PS)*). Ovakav format omogućava jednostavnije programsko rukovanje podacima i koristi se za audio/video reprodukciju (npr. reprodukcija sa CD-a ili DVD-a) kao i u nekim mrežnim aplikacijama. Ukoliko je međutim prenosni put podložan pojavi grešaka (npr.radiodifuzija), elementarni tokovi se kombinuju i prenose kao transportni tok (*Transport Stream (TS)*). Zbog toga se ovaj format koristi u digitalnoj televiziji. Sem toga TS format je pogodan za kombinovanje više TV programa u jedinstven informacioni tok. Ovaj proces kombinovanja MPEG2 elementarnih tokova u jedinstveni informacioni tok u skladu sa definisanim formatom se naziva se multipleksiranje MPEG2 toka.

Kao što je prikazano na Slici 1.1. jedan transportni tok se satoji iz više elementarnih tokova, kao i dodatnih podataka (kontrolni podaci, titlovi itd.). Prvo se audio i video podaci prevedu u signale po MPEG-2 standardu. Zatim MPEG-2 sistem procesor formatira dobijene ES u pakete i tako formira paketske elementarne tokove (PES, engl. *Packetised Elementary Stream*). PES paketi su blokovi podataka fiksne ili promenljive dužine, do 65536 bajta po bloku uključujući i obavezno 6-bajtno zaglavlje paketa. Svi paketi imaju polje PID (od engl. *Packet Identifier*). Ovako paketizovani podaci se šalju na multiplekser koji ih multipleksira u transportni tok. Ovakav signal ne može da se gleda sa klasičnim (analognim) TV prijemnikom. Zato se mora koristiti integrisani digitalni televizor ili uređaj koji će da DVB signal da konvertuje u klasični i prosledi ga televizoru.

Jednim transportnim tokom se može prenositi više servisa (Slika 2). Takođe, paketi nisu pravilno raspoređeni u vremenu. MPEG-2 TS nije vremenski multipleks tj. paketi sa bilo kojim PID-om mogu biti od strane TS multipleksera umetnuti u TS u bilo kom trenutku. Ukoliko TS multiplekser nema raspoloživih podataka za formiranje paketa, u TS će se umetati *null* paketi (PID=0x1FFF) da bi se održao zahtevani protok podataka.



Slika 2.1: Multipleksiranje elementarnih tokova u transportni tok



Slika 2.2: Multipleksiranje više servisa u jedan TS

Treba napomenuti da ono što je jedan program (kanal) u analognoj televiziji u digitalnoj televiziji zovemo jedan servis.

2.2. Signalne tabele

Da bi se prijemnoj strani omogućilo da poveže PID vrednosti sa odgovarajućim servisima sadržanim u TS, zajedno sa PES paketima u istom TS se prenose i specijalni kontrolni tokovi koji sadrže tzv. *signalne tabele* (engl. *Signalling Tables*). Ove tabele nose podatke o svakom od DTV servisa koji se prenosi unutar transportnog toka. Prenose se kao zasebni tokovi podataka unutar toka, multipleksirani zajedno sa ostalim paketima. Tabele (nazvane *Program Specific Information (PSI)* u MPEG-2 standardu) se sastoje od opisa elementarnih tokova koje treba kombinovati da bi se dobio određeni DTV servis, kao i od opisa samih DTV servisa. Signalne tabele predajnik šalje periodično, pakovanjem i multipleksiranjem odgovarajućih sekcija u tok.. Tabele koje definiše MPEG-2 standard, a koriste se u DVB standardu su:

- ***NIT*** (engl. *Network Information Table*) – sadrži informacije o mreži koja emituje TS multipleks kojem pripada i dati DTV servis (npr. Sky, PREMIERE, Canal+, itd.);
- ***PMT*** (engl. *Program Map Table*) – definiše listu PID vrednosti TS paketa koji sadrže elementarne strimove (npr. video, audio, teletekst) pridružene određenom programu;
- ***PAT*** (engl. *Program Association Table*) – sadrži listu svih programa u transportnom toku, koji se identigikuju preko polja *program_number*;
- ***CAT*** (engl. *Conditional Access Table*) – koriste se kod zaštićenih DTV servisa. Definišu tip kriptovanja i PID vrednosti TS paketa koji sadrže informacije neophodne za dekriptovanje sadržaja;
- ***DCM-CC*** (engl. *Digital Storage Media Command and Control*) – ove tabele u sebi sadrže komande koje se šalju prijemniku. U pitanju je protokol koji koristi *client-server* model komunikacije i koji se koristi za kontrolu prijema.

DVB standard proširuje skup signalnih tabela uvodeći tzv. tabele sa servisnim informacijama (engl. *Service Information Tables – SI tables*). Ove tabele nose informacije o DTV servisima sadržanim u transportnim tokovima koji nose više servisa. Neke od važnijih SI tabela definisanih DVB standardom:

- **SDT** (engl. *Service Description Table*) daje nazive i druge detalje o servisima (PID = 0x0011) ;

- **BAT** (engl. *Bouquet Association Table*) grupiše servise u logičke celine (npr. sport, muzika, filmovi, itd.) ;

- **EIT** (engl. *Event Information Table*) sadrži detalje o rasporedu emitovanja programa;

- **TDT** (engl. *Time and Date Table*) sadrži informaciju o vremenu emitovanja.

Osim sekcija definisanih MPEG-2 i DVB standardom, dozvoljeno je i emitovanje tzv. privatnih sekcija čiji je format poznat samo emiteru servisa i prijemnoj strani kojoj su sekcije namenjene.

2.2. Uslovni pristup

Jedna od novina koje donosi dogitalna televizija je mogućnost uslovnog pristupa (engl. *Conditional Access - CA*) određenim sadržajima. Ovo je urađeno korišćenjem tehnika kriptovanja i šifrovanja. Naime, tok je šifrovan 48-bitnim ključem, koji se naziva kontrolna reč (engl. *control word*). Kontrolna reč se menja nekoliko puta u minutu i to automatski, i njena vrednost se ne može predvideti. Da bi prijemnik mogao da dešifruje program, mora da ima aktuelnu i tačnu kontrolnu reč. Kriptovanje služi za sakrivanje kontrolne reči u toku, tačnije kontrolna reč je kriptovana u okviru ECM poruke (skraćenica od engl. *Entitlement Control Message*). Prijemnik će dekriptovati ECM poruku samo ako je autorizovan za to, a autorizacija se šalje u vidu EMM poruka (skraćeno od engl. *Entitlement Menagment Message*). EMM poruke su specifične za svakog korisnika ili grupu korisnika. One se menjaju mnogo ređe ECM poruke, najčešće na mesečnom nivou.

2.3. CAM modul i CI+ standard

CAM (Slika 2.3, od engl. *Conditional Access Module*) je elektronski uređaj koji u sebi obično sadrži slot za smart karticu. CAM se ubacuje u prijemnike kada se korisniku želi pružiti mogućnost da pristupa zaštićenim sadržajima (tačnije sadržajima koje je platio). Uloga CAM-a je da iz toka izvlači ključne podatke i na osnovu njih izvrši dekriptovanje i dešifrovanje ako korisnik ima pravo da gleda određeni sadržaj. CAM od prijemnika prima DTV transportni tok i ako tok nije šifrovan (tj. čist je), prosleđuje ga prijemniku. Ako je šifrovan, CAM iz toka izvlači podatke bitne za uslovni pristup i proverava da li korisnik ima

pravo da gleda program koji je izabrao. Ako ima, iz toka se izvlače podaci na osnovu kojih se vrši dekriptovanje i dešifrovanje i prijemniku se prosleđuje signal koji korisnik može da gleda.

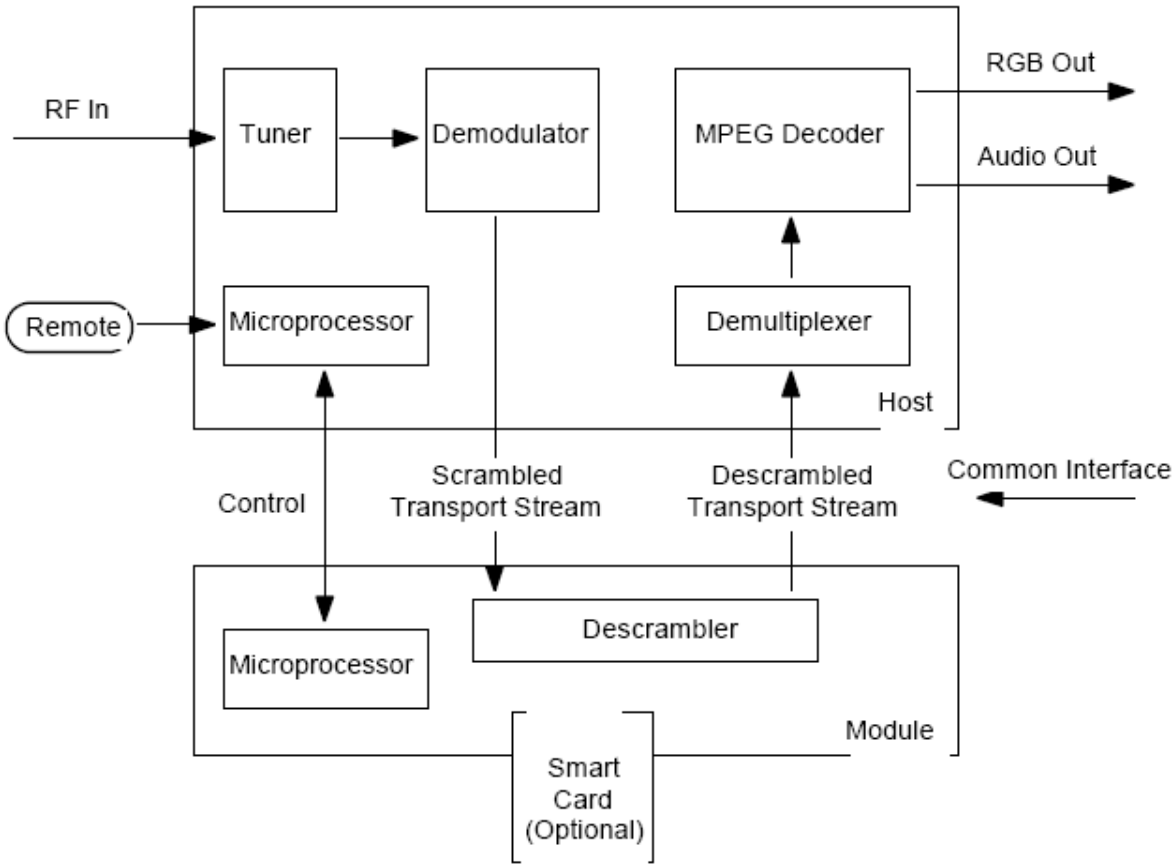
Slika 2.3: CAM modul

Komunikacija između prijemnika i CAM-a je definisana CI+ standardom (od engl. *Common Interface*, aktuelna verzija je 1.3).

Slika 2.4: Zvanični logo CI+ standarda

CI+ interfejs se može podeliti na sva dela: komandni i transportni. Oba dela imaju po više nivoa. Transportni deo rukuje MPEG-2 transportnim tokom u oba smera. Komandni

služi za komunikaciju između aplikacije na CAM-u i prijemnika. Zadatak komandog dela interfejsa je da omogući složenu komunikaciju između prijemnika i CAM-a.



Slika 2.5: Komunikacija između prijemnika i CAM-a

Na Slici 2.5. je prikazana pojednostavljena komunikacija i osnovne funkcije prijemnika i CAM-a. Na ulazu prijemnika se odabere servis koji korisnik gleda, a zatim se signal demoduliše (isti princip kao u analognoj televiziji). Sada se ovaj signal prosleđuje CAM-u, koji ga eventualno dešifruje, i vraća prijemniku. Mikroprocesori kontrolišu i prijemnik i CAM.

3. Softver za CAM i uloga CAPMT modula

Programska podrška za CAM je pisana u programskom jeziku C.

Arhitektura programske podrške za CAM je podeljena u tri nivoa. To su: najniži nivo HAL (od engl. *Hardware Abstraction Layer*), srednji novo (u engl. literaturi se zove *Midleware*) i najviši aplikativni nivo. Na Slici 3.1. su prikazani ti nivoi i veze koje CAPMT modul ima sa drugim modulima.

HAL nivo je nivo najbliži hardveru. U njemu su impelementirani operativni sistem MQX i funkcije koji ostalim modulima omogućavaju jednostavno korišćenje hardverskih resursa.

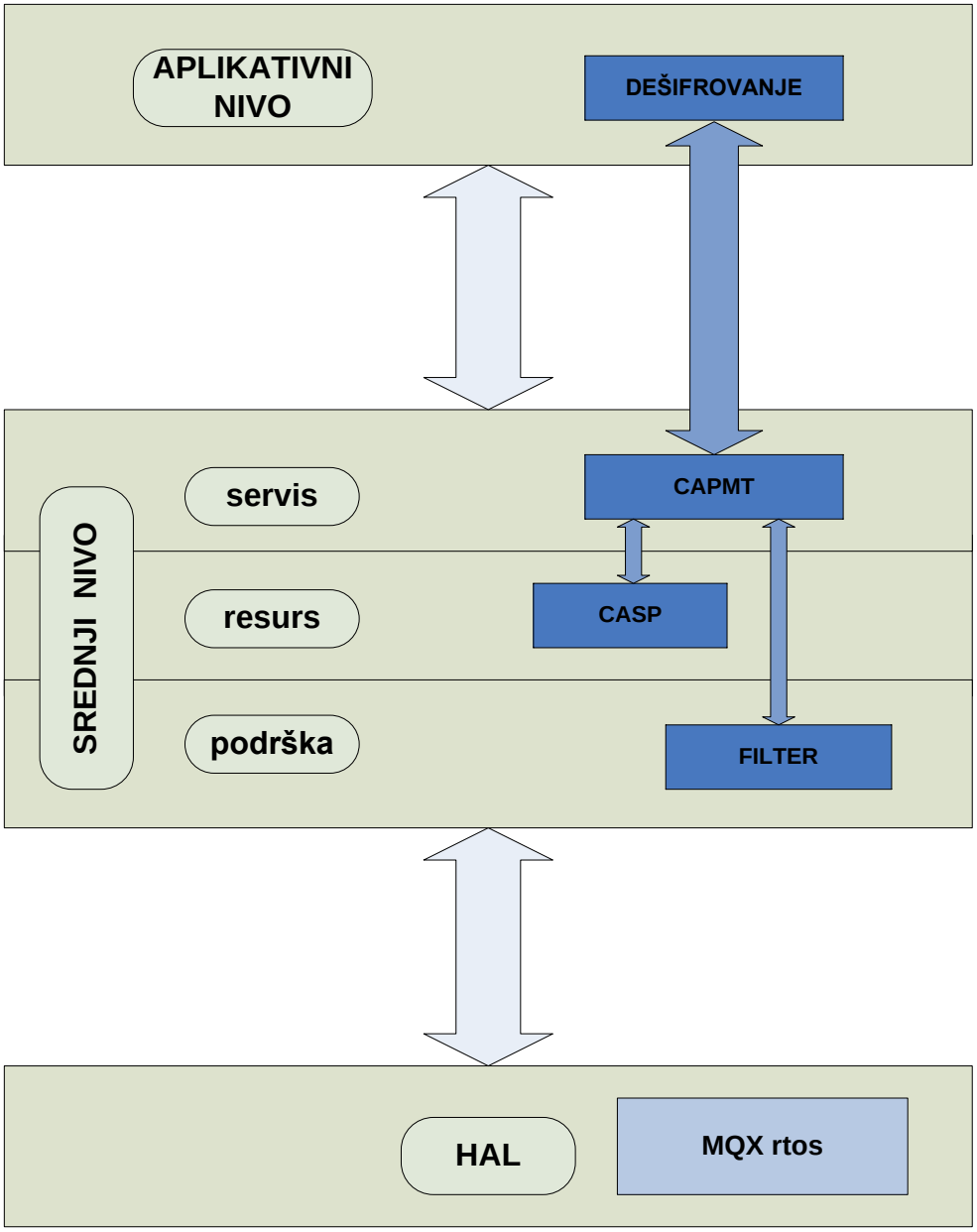
MQX[5] je operativni sistem za rad u realnom vremenu. Primenjuje se u mnogim samostalnim sistemima. Njegova veličina je zavisna od funkcionalnosti koje će biti korišćene, tako da najmanje bude 6kB, uključujući kernel, rukovanje programskim prekidima, semaforima, redovima i upravljanje memorijom. Generalno, MQX ima moćne performanse koje su omogućene multitasking jezgro, brz odgovor na programske prekide, brzu komunikaciju između taskova i fajl sistem.

HAL nivo pruža apstrakciju hardvera višim nivoima. Tako isti viši nivoi rade sa različitim hardverskim konfiguracijama. U njemu su realizovane brojne funkcije koje rešavaju zadatke koji se često javljaju u drugim modulima. To su napr. rukovanje memorijom, kreiranje determinističkih konačnih automata itd.

Aplikativni nivo (u daljem tekstu aplikacija) ima dve osnovne funkcije: komunikaciju sa korisnikom i komunikacija sa prijemnikom preko transportnog intefejsa. Komunikacija sa korisnikom se obavlja preko grafičkog interfejsa kada korisnik u meniju prijemnika izabere opciju *Common Interface* (u zavisnosti od prijemnika može biti nazvana drugačije). Na osnovu naredbi koje korisnik izdaje, aplikacija će menjati izgled ekrana i pokretati razne operacije na nižim nivoima. Aplikacija vrši i kontrolu dešifrovanja i slanje transportnog toka prijemniku, kao i skidanje nove verzije softvera.

Srednji nivo je praktično uslužni servis aplikacije. Služi da aplikativnom nivou pruži jednistven API tj. apstrakciju CI+ modula kao i potrebnih funkcionalnosti hardvera. Ovaj nivo pruža funkcije koje obezbeđuju potrebne podatke aplikativnom nivou. Napr. moduli u ovog nivoa filtriraju tabele iz transportnog toka, čuvaju podatke o aktivnim programima itd. Podeljen je na tri podnivoa: servis, resurs i podrška.

Svaki od ovih nivoa ima veći broj svojih modula. Svaki modul ima strukturu (koja se u kodu naziva objekat) u kome se nalaze osnovni podaci o modulu. To su napr. muteks, red za poruke i podaci karakteristični za sam modul. Svaki modul ima funkcije za inicijalizaciju i deinicijalizaciju. Funkcija za inicijalizaciju stavlja u početno stanje gorepomenuti objekat i obavlja još neke stvari u zavisnosti od samog modula (napr. kreiranje taska).



Slika 3.1: Struktura softvera i mesto CAPMT modula

Imena funkcija, nabrajanja i nekih promenljivih se daju tako da se na osnovu imena može zaključiti kom modulu pripadaju. Za svaki nivo i modul je usvojena skraćenica. Napr. imena funkcija iz CAPMT modula počinju sa *mwser_capmt*, a iz aplikacije za dešifrovanje *app_descr*.

CAPMT modul se nalazi u srednjem nivou. Od modula iz srednjeg nivoa komunicira sa CASP modulom koji predstavlja CI+ resurs, i sa modulom Filter koji naleže na samu

platformu.. Od modula iz aplikacije komunicira sa modulom za dešifrovanje (vrši kontrolu dešifrovanja toka).

Kada korisnik izabere novi program, prijemnik pošalje CAM-u CAPMT tabelu koju prima CASP (skraćenica od *Conditional Access SuPport*). CASP je prosleđuje CAPMT modulu koji proverava sadržaj tabele. CAPMT tabela sadrži sve podatke o programu. To su:

- redni broj programa (servisa) u toku;
- mesto u nizu aktivnih programa koje treba da zauzme (prvi, jedini, poslednji, sledeći);
-komanda za dešifrovanje (dozvoljeno, nedozvoljeno, dozvoljeno uslovno, vrati prijemniku informaciju o programu);
- CA deskriptori na nivou programa ili CA deskriptori na nivou elementarnih tokova;
- bit koji pokazuje da li je program šifrovan na nivou programa;
- broj elementarnih tokova.
- PID (*Packet IDentifier*);

CAPMT modul proverava da li je servis šifrovan. Ako jeste, proveriće da li korisnik ima pravo da ga gleda, i ako ima, poslaće poruku aplikaciji. Zatim aplikacija traži od CAPMT modula da pošalje poruku modulu Filter za filtriranje ECM poruka iz toka za taj program. Kada Filter završi filtriranje, vraća CAPMT-u isfiltrirane poruke, koje se prosleđuju aplikaciji. CAPMT modul komunicira i sa HAL-om.

4. Trenutno rešenje CAPMT modula

CAPMT modul spada među servise, tj. nalazi se u višem podnivou srednjeg (*midleware*) nivoa u hijerahiji softvera. Njegova uloga je da vrši operacije bitne za uslovni pristup programu koji korisnik želi da gleda. Pod ovim se ugrubo smatra:

- čuvanje podataka o programima i njegovim elementima;
- upravljanje procesima filtriranja ECM podataka o programu;
- proveravanje da li korisnik ima pravo da gleda izabrani program (ako je program šifrovan, ako nije nema potrebe za proverom)
- obaveštavanje modula za dešifrovanje iz aplikativnog nivoa o promenama u listi aktivnih programa;
- čuvanje podataka o svim aplikacijama koje su prijavljene na CAPMT modul.

Modul komunicira sa aplikacijom preko API-a koja je u hijerarhiji iznad njega. API čine funkcije koje počinju imenom **mw _capmt,** i njih poziva aplikacija kada želi da CAPMT pokrene neki niz operacija (šta to može biti biće objašnjeno kasnije). Inače, ove funkcije se mogu pozivati i iz samog CAPMT modula.

Trenutno, CA PMT modul se sastoji iz krovnog modula i četiri podmodula: modul za čuvanje podataka (DB, od engl. *Data Base*), ECME (od engl. *Entiltement Control Message Engine*), kontrolni (CTR, od engl. *ConTRol*) i IPP podmodula.

DB podmodul implementira čuvanje podataka o programima, elementarnim tokovima i ECM porukama (sve vezano za čuvanje ovih podataka se nalazi u fajlovima sa **_db** u imenu). Takođe, ovaj modul omogućava rukovanje s njima: dodavanje novih, brisanje

postojećih, i izmjena postojećih. To je realizovano javnim i API funkcijama koje se pozivaju iz samog CAPMT modula i drugih modula.

ECME podmodul implementira funkcionalnosti vezane za filtriranje ECM poruka o programu. To su: startovanje i zaustavljanje filtriranja određenog programa, očitavanje rezultata filtriranja i slanje istih aplikaciji koja vrši dešifrovanje. Ako se neki program briše iz spiska aktivnih, ECME podmodulu se šalje poruka da zaustavi filtriranje istog i izbriše iz spiska filtriranih programa. Ako korisnik izabere da gleda neki šifrovani program i to mu je omogućeno, ECME podmodulu se pošalje odgovarajuća poruka, i on startuje filtriranje programa.

CTR podmodul je modul u kome je implementiran konačni deterministički automat (u budućem tekstu konačni automat). Pod tim se smatraju funkcije koje pozivaju preko mehanizma konačnog automata, i funkcije koje se pozivaju iz njih. Na osnovu događaja koji se desio (mehanizam će preciznije biti objašnjen kasnije) poziva se određena funkcija.

IPP sadrži samo funkcije koje se pozivaju iz APP_IPP nivoa koje daju neke informacije koje sadržiCAPMT modul.

4.1. Glavni podmodul

Može se smatrati da je glavni-centralni de modula smešten u fajlovima **mwser_capmt.c** i **mwser_capmt.h**. U fajlu **mwser_capmt.c** se nalazi task, zatim su u njemu definisane API funkcije čije ime počinje sa **mw_capmt**. Takođe, tu su i definisane i javne funkcije za inicijalizaciju i uništavanje modula, kao i funkcija za slanje poruka u red koje će izvršavati task. Takođe, tu su i nabrajanje i strukture koje su bitne za modul u celini. U fajlu **mwser_capmt.h** su date deklaracije funkcija, nabrajanja i promjenljivih koje se koriste u **mwser_capmt.c** fajlu. Na početku **mwser_capmt.c** fajla je definisan primerak strukture **MWSER_CAPMT_Object** (definicija je data na Slici 4.1.) pod imenom **gst_MWSER_CAPMT_Object**, koji definiše CAPMT modul. Definisan je muteks, parametri taska, red za poruke itd.

```
typedef struct
{
    hal_mq_Handle                hCaPmtMessageQueue;          /**< MWSER_CAPMT
message queue handle             */
    hal_krn_TaskParams           TaskParams;                  /**< MWSER_CAPMT
message task parameters          */
    hal_stm_Handle               CtrStmHandle;               /**< MWSER_CAPMT_CTR
state machine handle         */
    uint32                       u32CaSpPayLoadSize;         /**< Payload size of
Ca Support resource        */
    hal_krn_MutexHandle          MutexHandle;
    hal_krn_SemHandle            SemHandle;
} MWSER_CAPMT_Object;
```

Slika 4.1:Definicija strukture **MWSER_CAPMT_Object**

U globalu, ovaj glavni dio CAPMT modula funkcioniše tako što drugi moduli šalju poruke u red za poruke. Moduli koji pripadaju aplikativno nivou to rade pozivajući API funkcije, čiji argument je u stvari poruka CAPMT modulu. Ovo se tumači kao da je sam modul pošiljalac poruke. API funkcije stavljaju tu poruku u red za poruke funkcijom **SendCapmtStmEvent**. Poruke u task može slati i CASP modul kada primi CA PMT tabelu od prijemnika. Poruka u task se šalje i pri prijavi aplikacija na modul. Kada operativni sistem pozove task, pročita se poruka iz reda i analizira se.U tasku, se prvo provjeri koji modul šalje poruku. Ako to nije CASP ili sam CAPMT, neće se ništa dešavati, tj. modul da reaguje na poruke samo ova dva modula.

CASP modul šalje CA PMT tabelu ekstrahovanu iz toka. Tabela je u stvari struktura **Rsc_Casp_CaPmtData** koja je prikazana na Slici 4.2. Kao što se vidi, ona sadrži i pokazivače na strukturu **Rsc_Casp_EsInfoData**, koja sadrži sve bitne podatke o elementarnim tokovima (Slika 4.3).

```
typedef struct
{
    uint8                    u8ListManagement; /**used to indicate whether the
user has selected one or several programmes*/
    uint32                   u32CaAppHandle;   /**< App handle  */
    uint16                   u16ProgramNb;     /**programme number*/
    uint8                    u8Version;        /**version number*/
    bool                     bCurrent;
    bool                bCaProgramLevel;  /** if set to TRUE ==> CA_descriptor
at programme level*/
    uint8                    u8CmdId;          /**indicates what response is
required from the application*/
    uint8           u8CaDescriptorNb; /** nb of element in pCaDescriptor*/
    Rsc_Casp_CaDescriptor*          ppCaDescriptor[MAX_NB_PROG_CA_DESCRIPTOR];
/**Rsc_Casp_CaDescriptor handles table used if bCaProgramLevel == TRUE*/
    uint8                    u8EsNb;           /** nb of element in pEsInfo*/
    Rsc_Casp_EsInfoData*   ppEsInfo[MAX_ES_NUMBER];   /**   Rsc_Casp_EsInfoData
handles table */
  } Rsc_Casp_CaPmtData;
```

Slika 4.2: Definicija **Rsc_Casp_CaPmtData** strukture

```
typedef struct
{
    uint8             u8StreamType;     /** stream type*/
    uint16            u16Pid;           /** PID*/
    bool               bCaEsLevel;       /** if set to TRUE ==> CA_descriptor
at elementary stream level*/
    uint8            u8CmdId;          /**indicates what response is required
from the application at ES level*/
    uint8              u8CaDescriptorNb; /** pCaDescriptor length*/
    Rsc_Casp_CaDescriptor*    ppCaDescriptor[MAX_NB_ES_CA_DESCRIPTOR];      /**
Ca_Descriptor handles table*/
} Rsc_Casp_EsInfoData;
```

Slika 4.3: Definicija **Rsc_Casp_EsInfoData** strukture

Kada CASP modul pošalje poruku, program uđe u deo koda gde se u **switch** strukturi proverava vrednost polja **ca_pmt_list_management** u CA PMT tabeli, i u zavisnosti od toga će se putem funkcije **hal_stm_SendEvent** poslati poruku-događaj mašini stanja, koja će na

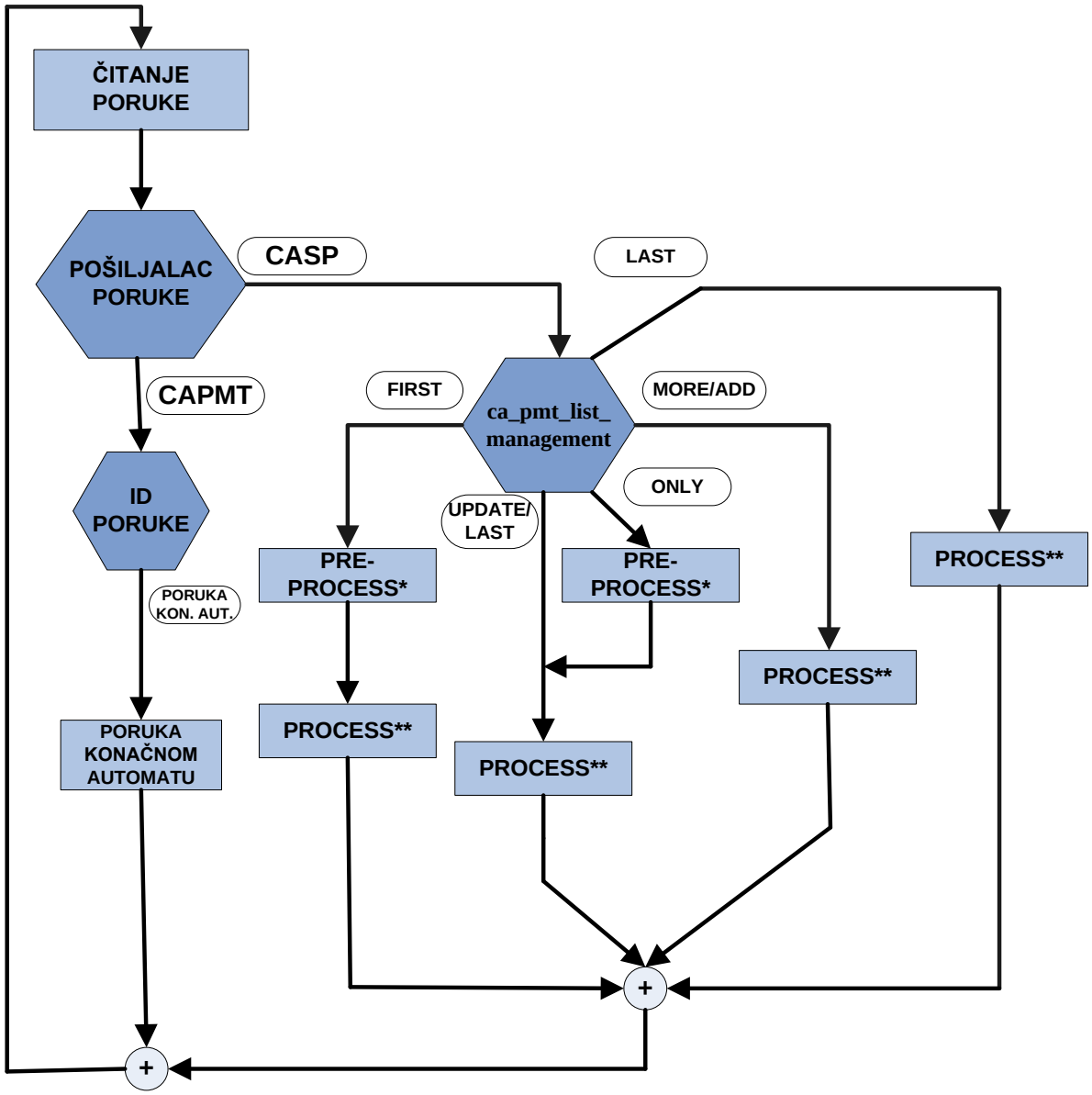
osnovu parametara funkcije pozvati određenu funkciju iz CTR podmodula. Poruke koje šalje CASP modul su vezane za program koji je korisnik izabrao i preko njih se traži promjena podataka o nekom programu, brisanje ili dodavanje novog bazu. Moguće vrednosti polja **ca_pmt_list_management** su predstavljene u Tabeli 4.1.

ca_pmt_list_management	*značenje*
more	program nije ni prvi ni poljednji u listi
first	prvi u novoj listi
last	posljenji u listi
only	jedini u listi
add	dodavanje programa
update	apdejtovanje programa
ostalo	Nema efekta

Tabela 4.1: Moguće vrednosti polja **ca_pmt_list_management**

U task mogu stizati i poruke o trenutnom stanju sesije (otvorena, zatvorena, već otvorena) pri prijavi aplikacije. Ovo ne izaziva neke bitne radnje, tako da će se zanemariti.

Ako je pošiljalac poruke sam modul, šalje se poruka mašini stanja sa pogodnim argumentom, a ona na osnovu toga poziva odgovarajuću funkciju iz CTR podmodula. Može se reći da se u tasku poruka samo "prepakuje" i šalje kao argument funkcije **hal_stm_SendEvent** konačnom automatu. Ovo je grafički prikazano na Slici 4.5. Ovim se pokreću operacije vezane za filtriranje, vraćanje statusa dešifrovanja, prijavljuje ili odjavljuje aplikacija. Bitno je napomenuti, da prijava prve aplikacije modulu predstavlja otvaranje sesije i da se o tome šalje poruka u task, a pri prijavljivanju svake sljedeće aplikacije samo se šalje poruka da je sesija već otvorena.



* *slanje događaja mašini stanja EV_CAPMT_CTR_FSM_PREPARE*

** *slanje događaja mašini stanja EV_CAPMT_CTR_FSM_PROCESS*

Slika 4.5: Dijagram glavnog taska

Funkcija za inicijalizaciju CAPMT modula je **mwser_capmt_Init**. Na početku se provjerava da li je modul već inicijalizovan, a zatim se kreira red za poruke pozivom funkcije. Inicijalizuje se **gst_MWSER_CAPMT_Object**, gdje se pre svega definišu parametri taska a zatim se kreira task. Na kraju, inicijalizuju se podmoduli CTR, ECME i DB pozivanjem odgovarajućih funkcija za inicijalizaciju

Funkcija za uništavanje objekta **mwser_capmt_Deinitialize** funkcioniše tako što se na početku se proveri da li je objekat uopšte inicijalizovan, a zatim se unište red za poruke i task. Na kraju se unište podobjekti pozivanjem odgovorajućih funkcija. Sve operacije sa taskom i redom za poruke se vrše funkcijama iz HAL nivoa.

Funkcija za stavljanje poruka u red za poruke CAPMT modula je **SendMsg2CaPmt** Ona od ulaznih paramtera funkcije (najvažniji su ID poruke koja se šalje modulu, njen sadržaj i veličina) pravi poruku (instanca strukture **hal_mq_Message**) koja se šalje modulu pozivom funkcije **hal_mq_Write.** Ova funkcija ustvari stavlja poruku u red za poruke CA PMT modula.

API funkcije (koje su pobrojane i opisane u tabeli) funkcionišu na sličan način. Na početku se proveri da li je CAPMT modul inicilalizovan, i alocira se memorija potrebna da se u nju smjesti struktura koja je ulazni parametar funkcije. Zatim se na osnovu ulaznih parametara (pošiljalac, ID poruke, sadržaj) sastavljaju poruku koju šalju tasku pozivom prethodno opisane funkcije **SendMsg2CaPmt.** U Tabeli 4.2. dat je spisak tih funkcija sa kraćim opisom.

mw _capmt_Register	Funkcija za prijavu aplikacije. Ulazni parametar je primjerak strukture koja sadrži sve podatke o aplikaciji koja se registruje.
mw_capmt_Unregister	Funkcija za odjavu aplikacija. Ulazni parametar je ID funkcije koja se odjavljuje.
mw_capmt_PrgEcmFilter	Funkcija za startovanje ili stopiranje filtriranja. Ulazni parametri funkcije su indeks programa, ID aplikacije koja traži filtriranje i akcija koja se traži u vezi filtriranja (početak ili kraj filtriranja)
mw_capmt_Dump	Funkcija za ispis svih informacija o modulu (služi samo u svrhu debagovanja).
mw_capmt_DescramblingStatus	Funkcija za setovanje statusa dešifrovanja. Ulazni parametri funkcije su indeks programa i ID aplikacije koja želi da setuje status i sam status, koji se dodjeljuju primerku strukture koja sadrži sve bitne parametre za setovanje statusa dešifrovanja

Tabela 4.2: API funkcije

4.2. CTR podmodul

U CTR modulu je definisan konačni automat kompletnog CAPMT modula. Ovo znači da se u task funkciji šalje događaj konačnom automatu, koji je definisan u CTR podmodulu. U zavisnosti od toga u kom stanju je konačni automat, i koji se događaj desio, biće pozvana određena funkcija. Ovaj podmodul je smješten u sljedećim fajlovima: **mwser_capmt_ctr.c, mwser_capmt_ctr_private.h** i **mwser_capmt_ctr.h**.

U fajlu **mwser_capmt_ctr_private.h** je definisan konačni automat (Slika 4.6.)

```
    Static                                                  hal_stm_EventCallback*
CAPMT_CTR_FSM_FN[EV_CAPMT_CTR_FSM_MAX][CAPMT_CTR_FSM_STATE_MAX] = {
/*****************************************************************/

 * EVENTS          \      STATES :        IDLE      READY        PROCESSING
 *****************************************************************/
/*EV_CAPMT_CTR_FSM_CA_REGISTER   */ {    A1,        A2,            A3,          },
/*EV_CAPMT_CTR_FSM_CA_UNREGISTER */ {     X ,        B2,           B3,          },
/*EV_CAPMT_CTR_FSM_PREPARE       */ {     X ,        C2,           X ,          },
/*EV_CAPMT_CTR_FSM_PROCESS       */ {     Y ,        D3,           D3,          },
/*EV_CAPMT_CTR_FSM_END           */ {     X ,        X ,           E3,          },
/*EV_CAPMT_CTR_FSM_ECM_FILTER    */ {     X ,        F2,           F3,          },
/*CTR_FSM_DESCRAMBLING_STATUS    */ {     X ,         Z,            Z,          },
};
```

Slika 4.6: Konačni automat CA PMT modula, definisana u CTR podmodulu

Kao što se vidi, događaji koji utiču na mašinu stanja su:

EV_CAPMT_CTR_FSM_CA_REGISTER - registracija nove aplikacije (u daljem tektsu CA_REGISTER);
EV_CAPMT_CTR_FSM_CA_UNREGISTER – odjava aplikacije;
EV_CAPMT_CTR_FSM_CA_PREPARE – stavlja sve programe u stanje DELETE;
EV_CAPMT_CTR_FSM_CA_PROCESS – procesiranje programa
EV_CAPMT_CTR_FSM_CA_END – kraj procesiranja programa
EV_CAPMT_CTR_FSM_CA_FILTER – pokretanje filtriranja programa
CTR_FSM_DESCRAMBLING_STATUS – davanja informacije o programu (u smislu da li je dozovljeno dešifrovanje).

Moguća stanja konačnog automata su: **IDLE**, **READY** i **PROCESSING**. U **IDLE** stanju automat se nalazi na samom početku i pri prijavi prve aplikacije (dešava se odmah pri ubacivanju CAM-a u host) se prebacuje u stanje **READY**. U stanju **READY** modul može samo da registruje ili odjavljuje aplikaciju, i čim automat dobija poruku da treba da se izvrši operacijam sa programima modul prelazi u stanje **PROCESSING**. U ovom stanju modul ostaje trajno, i u njemu se postiže puna funkcionalnost modula.

Na Slici 4.7. su date definicije skraćenica koje se koriste u gornjoj matrici. To su funkcije koje su takođe deklarisane u ovom fajlu.

```
#define A1  CAPMT_CTR_IdleCaRegister
#define A2  CAPMT_CTR_ReadyCaRegister
#define A3  CAPMT_CTR_ProcessingCaRegister

#define B2  CAPMT_CTR_ReadyCaUnregister
#define B3  CAPMT_CTR_ProcessingCaUnregister

#define C2  CAPMT_CTR_ReadyPrepare

#define D3  CAPMT_CTR_ProcessingProcess

#define E3  CAPMT_CTR_ProcessingEnd

#define F2  CAPMT_CTR_EcmFilter
#define F3  CAPMT_CTR_EcmFilter

#define X   CAPMT_CTR_Ignore
#define Y   CAPMT_CTR_IgnoreAndFree
#define Z   CAPMT_CTR_DescramblingStatus
```

Slika 4.7: Skraćenice za funkcije konačnog automata

Sve funkcije konačnog determinističkog automata imaju iste argumente: void pokazivač na podatak koji je ulazni argument i pokazivač na globalni podatak **retValue**, koji je korisnički definisanog tipa podataka ***hal_stm_CBReturnValue***, kojim se predstavlja stanje mašine stanja. Primjer za deklaraciju jedne takve funkcije je :

```
static ERROR CAPMT_CTR_ProcessingCaUnregister (void *data,
hal_stm_CBReturnValue *retValue);
```

Void pokazivač se koristi jer ulazni argument može biti jedna od nekoliko različitih struktura koje se koriste u kodu, u zavisnosti od potreba i uloge funkcije. Zato se podatak na koji pokazuje void pokazivač "kastuje" u tip podatka koji se koristi u datoj funkciji (to je u CTR podmodulu neka od struktura). Sve ove funkcije su definisane kao *static*, a povratna vijednost im je član nabrajanja **ERROR**. Kada se neka od ovih funkcija pozove iz konačnog automata, one "odrade" neki posao i po potrebi promene stanje konačnog automata. One vrlo često pozovu neku od privatnih funkcija ovog modula ili neku od funkcija drugih podmodula (ECME ili DB).

U fajlu **mwser_capmt_ctr_private.h** je takođe definisana i struktura **MWSER_CAPMT_CTRObject**, koja sadrži sve podatke o CTR modulu. Ona sadrži podatke o programima koji se filtriraju i registrovanim aplikacijama, što je vrlo bitno. Na Slici 4.8. je data definicija ove strukure. Definisan je pokazivač na deo memorije koji zauzima ova struktura, muteks, red za poruke, kao i niz sa registrovanim aplikacijama i niz sa programima.

```
typedef struct
{

   hal_krn_MutexHandle        MutexHandle;
/**< MWSER_CAPMT_CTR mutex handle                              */
   hal_stm_Handle             StmHandle;
/**< MWSER_CAPMT_CTR State Machine handle                      */
   hal_stm_StateInfo
pStateInfoMatrix[EV_CAPMT_CTR_FSM_MAX][CAPMT_CTR_FSM_STATE_MAX];  /**< Closures and
possible next states for each state/event */
   CtrRegisteredCaApp         stCtrRegisteredApps[CAPMT_CTR_MAX_APP_NB];
/**< Registred Ca applications array                           */
   CtrPrgFilter               stPrg[CAPMT_DB_MAX_PRG_NB];
   BOOL                       isRscCaspSessionOpened;
   uint8                      u8AppNumber;
/**< App Number                                                */
} MWSER_CAPMT_CTRObject;
```

Slika 4.8: Definicija strukture **MWSER_CAPMT_CTRObject**

U Tabeli 4.3. je spisak funkcija konačnog automata. Rečeno je šta one rade, na koji događaj reaguju, u kojim se stanjima mogu pozivati i kako mjenjaju stanje mašine stanja.

FUNKCIJA	ULOGA
CAPMT_CTR_IdleCaRegister	Prijavljuje aplikaciju kada je modul u IDLE stanju na CA_REGISTER događaj stanje modula menja u READY.
CAPMT_CTR_ReadyCaRegister	Prijavljuje aplikaciju kada je modul u READY stanju na CA_REGISTER događaj i ne menja stanje modula.
CAPMT_CTR_Processing CaRegister	Prijavljuje aplikaciju kada je modul u PROCESSING stanju na CA_REGISTER ne menja stanje modula.
CAPMT_CTR_ReadyCaUnregister	Odjavljuje aplikaciju kada je modul u READY stanju na CA_UNREGISTER događaj i ne menja stanje modula.
CAPMT_CTR_ ProcesingCaUnregister	Odjavljuje aplikaciju kada je modul u PROCESSING stanju na CA_UNREGISTER događaj i ne menja stanje modula.
CAPMT_CTR_ReadyPrepare	Postavlja sve programe u DB-u u stanje IN_DELETION kada je u stanju READY i postavlja stanje PROCESSING na PREPARE događaj.
CAPMT_CTR_ProcessingProcess	Provjerava da li je program ili neki njegov ES šifrovanje i dodaje program u DB ili apdejtuje postojeći. Poziva se u READY ili PROCESSING stanju, a mašinu stanja ostavlja u PROCESSING stanju na PROCESS događaj.
CAPMT_CTR_EcmFilter	Pokreće akciju u vezi filtriranja: startovanje ili stopiranje filtriranja programa. Poziva se u READY ili PROCESSING stanju na ECM_FILTER i ne menjaju stanje mašine stanja.
CAPMT_CTR_Ignore	Ne radi ništa, može se pozvati u svim stanjima, i ne menja stanje.
CAPMT_CTR_IgnoreAndFree	Ne radi ništa, može se pozvati u stanju IDLE, i ne menja stanje.
CAPMT_CTR_ DescramblingStatus	Kreira odgovor na osnovu koga će aplikacija da pošalje odgovor hostu o određenom programu. Poziva se u READY ili PROCESSING na DESCRAMBLING_STATUS i ne menja stanje mašine stanja.

Tabela 4.3: Funkcije mašine stanja

Funkcije mašine stanja su definisane u fajlu **mwser_capmt_ctr.c**. U ovom fajlu su definisane i privatne i javne funkcije CTR podmodula. S tim što su javne deklarisane u fajlu **mwser_capmt_ctr.h** a privatne **mwser_capmt_ctr_private.h**.

Od ostalih funkcija nabitnije su **mwser_capmt_ctr_Init** i **mwser_capmt_Deinitialize**. Prva inicijalizuje CTR podmodul: kreira muteks, inicijalizuje **MWSER_CAPMT_CTRObject**, kreira mašinu stanja. Druga služi za uništavanje podmodula, tako što briše muteks i mašinu stanja i odjavljuje sva aplikacije.

Od privatnih funkcija najvažnije su:

- **CAPMT_CheckCaSysIdAndNotifyApp** – upoređuje CaSysId-jeve servisa i prijavljenih aplikacija. Time se proverava da li aplikacija treba da dobije informacije o programu. U slučaju aplikacije za dešifrovanje to praktično znači proveru da li korisnik ima pravo da gleda program koji jeizabrao. Poziva se pri dodavanju novog ili ažururanju postojećeg programa.
- **CAPMT_NotifyAppToClearCapmt** – poziva se iz prethodne funkcije kada treba poslati poruku aplikaciji o nešifrovanom programu.
- **CAPMT_Register** – služi za registraciju aplikacija. Poziva se u funkcijama za prijavu aplikacije iz konačnog automata, i slična je funkciji **CAPMT_CTR_IdleCaRegister**. U ove dve funkcije se osim poziva ove funkcije nalazi samo kastovanje ulazne vrednosti (koja je argument ove funkcije) i definisanje novog stanja konačnog automata.
- **CAPMT_Unregister** – isto kao u prethodnoj funkciji, samo je reč o funkciji za odjavu aplikacija. Poziva se u funkcijama za odjavu aplikacije.
- **CAPMT_BuildCaspAppRegister** – dodaje CaSysId-jeve (ako ih aplikacija ima) nove aplikacije na listu postojećih koja se šalje CASP modulu. Poziva se pri registraciji aplikacija.

4.3. DB podmodul

Ovaj podmodul služi za čuvanje svih podataka o programima, elementarnim tokovima i ECM porukama, kao i za rukovanje istima. To znači dodavanje, brisanje i izmjene podataka.

Sve vezano za ovaj modul nalazi se u fajlovima sa **mwser_capmt_db** u imenu. To su fajlovi: **mwser_capmt_db.c**, **mwser_capmt_db_tbx.c**, **mwser_capmt_db_private.h**, **mwser_capmt_db.h** i **mwser_capmt_db_tbx.h**. Ovaj modul ima i nekoliko API funkcija i struktura i nabrajanja koja se prosleđuju drugim modulima koji su deklarisani u fajlu **mw_capmt_db.h**. Ove funkcije su definisane u uslovno rečeno centralnom fajlu modula

mwser_capmt_db.c. U ovom fajlu su pored API funkcija, definisane i javne funkcije koje "odrađuju" glavne poslove ovog modula: dodavanje, brisanje promena i vraćanje podataka o programima, elementarnim tokovima i ECM porukama. One su deklarisane u fajlu **mwser_capmt_db.h.** Pored ovih funkcija, u ovom fajlu su definisani sva nabrajanja i strukture koje se koriste u modulu. U fajlovima **mwser_capmt_db_tbx.c** i **mwser_capmt_db_tbx.h** su definisane, odnosno deklarisane neke privatne funkcije modula, koje se pozivaju iz API i glavnih funkcija.

Posebno mjesto od svih struktura zauzima **MWSER_CAPMT_DB_Object** (Slika 4.8.).

```
typedef struct
{
   hal_krn_MutexHandle        MutexHandle;                          /**<
MWSER_CAPMT_DB mutex handle   */
   mw_capmt_Prg*          ppPrgHandles[CAPMT_DB_MAX_PRG_NB];  /**< Program
handles table       */
   mw_capmt_Es*           ppEsHandles[CAPMT_DB_MAX_ES_NB];    /**< ES
handles table            */
   mw_capmt_Ecm*          ppEcmHandles[CAPMT_DB_MAX_ECM_NB];  /**< ECM
handles table          */
} MWSER_CAPMT_DB_Object;
```

Slika 4.9: Deklaracija **MWSER_CAPMT_DB_Object**

Ova struktura sadrži sve podatke o svim programima, elementarnim tokovima i ECM porukama. Tačnije, njeni članovi su nizovi pokazivača na strukture koje imaju sve relevantne podatke o programima (**mw_capmt_Prg),** elementarnim tokovima (**mw_capmt_Es**) i ECM porukama (**mw_capmt_Ecm**). Ovi podaci se prosleđuju aplikaciji koja vrši dešifrovanje i drugim podmodulima CAPMT modula. Instanca ove strukture **gst_MWSER_CAPMT_DB_Object** je *de facto* baza podataka. Njen sadržaj se može menjati i očitavati funkcijama koje su implementirane u DB podmodulu. U strukturi je definisano onoliko pokazivača, koliko DB najviše može da čuva struktura sa podacima o programima, elementarnim tokovima i ECM porukama. Kada treba da se sačuva novi program, elementarni tok ili ECM poruka, prvo se alocira memorija koja je potrebna za čuvanje tih podataka. Zatim se odgovarajući pokazivač postavi da pokazuje na tu memoriju, i na kraju ta memorija "napuni" potrebnim podacima. Važno je napomenuti da kada se dodaje program, elementarni tok ili ECM poruka u DB, na novu strukturu u kojoj su podaci dodeliće se prvi sljedeći slobodan pokazivač iz niza pokazivača koji je za tu vrstu strukture namjenjen (napr. na novi program će pokazivati prvi sljedeći slobodan pokazivač iz niza pokazivača na programe).

U fajlu **mwser_capmt_db.c** se nalaze sve javne funkcije čije ime počinje sa sa **mwser_capmt_db.** One se koriste za rad sa **MWSER_CAPMT_DB_Object**-om (u budućem teksu DB, od engl. *Data base*).

Inicijalizacija ovog modula vrši se funkcijom **mwser_capmt_db_Init**. U njoj se prvo kreira muteks koji "štiti" ovaj podmodul, a zatim se taj muteks zaključa. To se radi da bi se svi pokazivači na sve strukture koje sadrže sve podatke o svim programima, elementarnim tokovima i ECM porukama postavili na nulu. Na kraju se podmodul otključa i bit koji pokazuje da li je modul inicijalizovan se setuje.

Funkcija za uništavanje DB podmodula je **mwser_capmt_db_Deinitialize.** U njoj se pozivom funkcije **mwser_capmt_clean** oslobodi sva memorija koju je zauzimao objekat, a svi pokazivači DB-a se postave na nulu.

U Tabeli 4.4. su date funkcije za operacije sa programima u DB-u.

FUNKCIJA	ULOGA
`mwser_capmt_db_AddPrg`	Dodaje program u DB. Ulazni podatak je pokazivač na strukuru **Rsc_Casp_CaPmtData,** a funkcija vraća poziciju programa u DB-u.
`mwser_capmt_db_RemovePrg`	Uklanja program iz DB-a. Ulazni podatak je pozicija programa koji se uklanja iz DB-a.
`mwser_capmt_db_UpdatePrg`	Menja sadržaj programa. Ulazni podatak je pokazivač na strukuru **Rsc_Casp_CaPmtData** i pozicija programa u DB-u koji se menja.
`mwser_capmt_db_GetListPrg`	Vraća listu programa u DB-u i broj programa u DB-u. Ulazni parametar je veličina te liste.
`mwser_capmt_db_GetPmtLenght`	Vraća veličinu pmt tabele programa u DB-u, čiji je broj ulazni parametar funkcije.

Tabela 4.4: Funkcije za opracije sa programima u DB-u

Podaci o pojedinim ES-ovima se čuvaju u instancama strukture **Rsc_Casp_EsInfoData.** To su podaci ekstrahovani iz ca_pmt tabele a vezani su za ca_deskriptore na nivou elementarnih tokova. U Tabeli 4.5. su date funkcije za operacije sa elementarnim tokovima u DB-u.

mwser_capmt_db_AddEs	Dodaje ES u DB. Ulazni paramtar je pokazivač na strukturu **Rsc_Casp_EsInfoData,** funkcija vraća poziciju ES-a u DB-u.
mwser_capmt_db_RemoveEs	Uklanja ES iz DB-a. Ulazni parametar je pozicija ES-a u DB-u.
mwser_capmt_db_UpdateEs	Mjenja podatke o određenom ES-u. Ulazni podatak je pokazivač na strukturu **Rsc_Casp_EsInfoData**. Izlazni je pozicija tog ES-a u DB-u.
mwser_capmt_db_GetListEs	Vraća listu ES-ova u DB-u i njihov broj u DB-u. Ulazni parametar je veličina te liste.
mwser_capmt_db_GetPrgParentEs	Vraća broj programa kojima tok pripada, i njihovu listu. Ulazni parametri su PID toka, i veličina liste.

Tabela 4.5: Funkcije za opracije sa ES-ovima u DB-u

Svi podaci o ECM poruci smeštaju se u primjerke strukture **Rsc_Casp_CaDescriptor.** To su ID deskriptora, veličina korisnog sadržaja (engl. *Payload*) i pokazivač na sam sadržaj. U Tabeli 4.6. su date funkcije za operacije sa ECM porukama u DB-u.

mwser_capmt_db_AddEcm	Dodaje ECM u DB. Ulazni paramtar je pokazivač na strukturu **Rsc_Casp_CaDescriptor,** funkcija vraća poziciju ECM-a u DB-u.
mwser_capmt_db_RemoveEcm	Uklanja ECM iz DB-a. Ulazni parametar je pozicija ECM-a u DB-u.
mwser_capmt_db_UpdateEcm	Menja podatke o određenom ECM-u. Ulazni podatak je pokazivač na strukturu **Rsc_Casp_CaDescriptor**. Izlazni je pozicija tog ES-a u DB-u.
mwser_capmt_db_GetListEcm	Vraća listu ECM-ova u DB-u i njihov broj u DB-u. Ulazni parametar je veličina te liste.

Tabela 4.6: Funkcije za operacije sa ES-ovima u DB-u

Kao što se vidi iz prethodne tri tabele, vrlo slične operacije se obavljaju nad programima, elementarnim tokovima i ECM porukama. Postoje funkcije za dodavanje, uklanjanje, promjenu i listanje programa, elementarnih tokova i ECM poruka, koje imaju interfejse na istom principu. Na primer, interfejs funkcija za dodavanje programa i ES-ova i

ECM poruka u DB su slični: funkcija dodaje u DB program, odnosno ES ili ECM poruku, tako da je ulazni parametar funkcije pokazivač na pogodnu strukturu (**Rsc_Casp_CaPmtData**, **Rsc_Casp_EsInfoData** ili **Rsc_Casp_CaDescriptor**), a izlazni podatak je pozicija programa, odnosno ES-a ili ECM poruke u DB-u.

Takođe, u većina gorepomenutih funkcija, pre nego što se krene sa promenom podataka u DB-u, se zaključa sa muteksom koji "čuva" objekat, a kad se završi rad sa ovim objektom, muteks se otključa.

Treba napomenuti da je ulazni i izlazni parametar funkcije u cjelom ovom projektu vrlo relativan pojam. Naime, sve funkcije u ovom projektu vraćaju jednu od vrednosti enumeracije **ERROR**. Ono što hoćemo da bude ulaz funkcije je klasičan ulaz. Međutim, rezultat rada funkcije je nova vrednost neke promenljive, koja se prosleđuje preko adrese, tako da se koristi tzv. bočni efekti funkcije. Često se u funkciji menja stanje neke globalne npr. u funkcijami stanja objekta **MWSER_CAPMT_DB_Object**), tako da ona ne daje rezultat u klasičnom smislu.

U tabeli 4.7 su date javne funkcije podmodula. Jasno je da one vrše operacije nad kompletnim **MWSER_CAPMT_DB_Object**-om.

`mwser_capmt_db_SetAllInDeletion`	Postavlja, stanja svih programa u stanje **IN DELETION**.
`mwser_capmt_db_ SetAllNewUpdatedToProcessed`	Postavlja stanje svih novih i apdejtovanih programa u stanje **PROCESSED**.
`mwser_capmt_db_Clean`	Briše kompletnu bazu podataka. Oslobađa memoriju koju dinamički drži objekat **MWSER_CAPMT_DB_Object**, a sve njegove pokazivače postavlja na NULL.

Tabela 4.7: Ostale bitne funkcije u DB podmodulu

Kao što je već navedeno, u fajlu **mw_capmt_db.h** su deklarisane funkcije i nabrajanja koja predstavljaju API. Ovde definisane funkcije su deklarisane u fajlu **mwser_capmt_db.c**. U tabeli 5 je dat prikaz ovih funkcija. Ove funkcije služe pre svega da daju informacije o podacima u DB-u. One se ne pozivaju samo iz aplikacija, već i iz srednjeg nivoa. One vraćaju

pokazivače na strukture koje sadrže podatke o programima, elementarnim tokovima i ECM porukama. To su strukture: **mw_capmt_Prg**, **mw_capmt_Es, mw_capmt_Ecm**, respektivno.

mw_capmt_db_GetInfoPrg	Uzima informacije o programu u DB-u. Ulazni parametar je pozicija programa, a izlazni je pokazivač na strukturu **mw_capmt_Prg**.
mw_capmt_db_GetPmtData	Kopira podatke o programu čija je pozicija ulazni parametar u memoriju na koju pokazuje pokazivač koji je izlazni parametar.
mw_capmt_db_GetInfoEs	Uzima informacije o ES-u u DB-u. Ulazni parametar je pozicija programa, a izlazni je pokazivač na strukturu **mw_capmt_Es**.
mw_capmt_db_GetInfoEcm	Uzima informacije o ES-u u DB-u. Ulazni parametar je pozicija programa, a izlazni je pokazivač na strukturu **mw_capmt_Ecm**.
mw_capmt_db_GetPrgParentEcm	Izlazni parametri su lista u kojoj su programi kojima pripada tok i njihov broj. Ulazni su ID ECM poruke i veličina liste.
mw_capmt_db_GetEsParentEcm	Izlazni parametri su lista u kojoj su ES-ovi kojima pripada ECM i njihov broj. Ulazni su ID ECM poruke i veličina liste.
mw_capmt_db_Dump	Ispisuje sve programe, ES-ove i ECM poruke u DB-u, u svrhu debagovanja.

Tabela 4.8: API funkcije DB podmodula

Osim do sada pomenutih, postoji još veliki broj funkcija, koje se mogu smatrati za privatne funkcije DB podmodula (funkcije koje se pozivaju samo u DB podmodulu). One su definisane u fajlovima **mwser_capmt_db.c** i **mwser_capmt_db_tbx.c**, i prema tome deklarisane i objašnjene u **mwser_capmt_db.h** i **mwser_capmt_db_tbx.h**. One se mogu smatrati "pomoćnim" funkcijama modula, i definisane su da "odrade" nešto za čim se često javlja potreba u drugim funkcijama podmodula. To su, čisto za ilustraciju, **FindEcm**, **FreePrg** itd. Prva funkcija nalazi poziciju ECM poruke u DB modulu na osnovi njenog PID-a, a druga oslobađa memoriju na kojoj su bili upisani podaci o nekom programu.

4.4. ECME podmodul

ECME podmodul služi za upravljanje operacijama filtriranja ECM poruka. On je implementiran na način sličan glavnom podmodulu. Naime, ECME podmodul ima svoj task, deterministički konačni automat, red za poruke. Sve vezano za ovaj podmodul nalazi se u fajlovima koji počinju sa **mwser_capmt_ecme** u imenu: **mwser_capmt_ecme.c, mwser_capmt_ecme_private.h** i **mwser_capmt_ecme.h**.

Svi podaci o modulu se nalaze u strukturi **CAPMT_ECME_Object** (Slika 4.8.), tačnije u instanci ove strukture **gCaPmtEcmeObject**. Tu su, između ostalog, definisani task, muteks i niz gde se nalaze spisak svih filtera koji rade. Pod pojmom filtar smatra se proces filtriranja, kojih može da bude više.

```
typedef struct
{
    hal_krn_MutexHandle         hProtectDataAccess; /* Mutex to protect
ECME data accessors */
    hal_mq_Handle               hMsgQueue;          /* Msg Q to process
commands */
    hal_krn_TaskHandle          hEcmeTask;          /* Task to read message
from above message Q */
    hal_stm_StateInfo
stStateInfoMatrix[MWSER_CAPMT_ECME_EVENT_MAX][MWSER_CAPMT_ECME_STATE_MAX];
    FilterIdentifier
hRunningFilters[CAPMT_ECME_MAX_SUPPORTED_FILTERS];  /* Global data of SCTR
module */
     } CAPMT_ECME_Object;
```

Slika 4.10: Definicija **gCaPmtEcmeObject**-a

U fajlu **mwser_capmt_ecme_private.h** su, pored prethodno date definicije definisana nabrajanja koja se koriste u ovom podmodulu (moguća stanja determinističkog konačnog automata i događaji koji to stanje mjenjaju). Tu su definisane i strukture koje se koriste u ECME podmodulu. Najbitnije su: **FilterIdentifier**, koja sadrži sve bitne podatke o pojedinačnom procesu filtriranja, i **CaPmtEcmeFsmData**, u koju se smeštaju podaci o mašini stanja. Takođe, u ovom fajlu je definisana konačni deterministički automat (Slika 4.9.) i deklarisana je task funkcija (naravno, task je definisan u **mwser_capmt_ecme.c)**. Ovdje su deklarisane i funkcije koje se pozivaju iz determinističkog konačnog automata u zavisnosti od njenog stanja. Konačni automat ima dva moguća stanja: **IDLE** i **RUNNING**.

U **IDLE** stanju modul je kada nija pokrento filtriranje nijedgnog programa. Čim se filtriranje pokrene, prelazi se u stanje **RUNNING**.

Mogući događaji koji utiču na stanja mašine stanja su:

MWSER_CAPMT_ECME_EV_START_FILTERING – startovanje filtriranja;
MWSER_CAPMT_ECME_EV_STOP_FILTERING – stopiranje filtriranja;
MWSER_CAPMT_ECME_EV_TABLE_RECEIVED – primljena tabela sa isfiltriranim podacima.

```
#define A1  MWSER_CAPMT_ECME_StartFilter
#define B2  MWSER_CAPMT_ECME_StopFiltering
#define C2  MWSER_CAPMT_ECME_TableReceived
#define XX  MWSER_CAPMT_ECME_Ignore

hal_stm_EventCallback*
CAPMT_ECME_FSM_FN[MWSER_CAPMT_ECME_EVENT_MAX][MWSER_CAPMT_ECME_STATE_MAX] = {
/*************************************************************************
  * EVENT          \        STATE  :     IDLE            RUNNING
*************************************************************************/
/*MWSER_CAPMT_ECME_EV_START_FILTERING*/ {   A1,               XX  },
/*MWSER_CAPMT_ECME_EV_STOP_FILTERING */ {   XX,               B2  },
/*MWSER_CAPMT_ECME_EV_TABLE_RECEIVED */ {   XX,               C2  }
};
```

Slika 4.11: Realizacija konačnog automata

Stanja konačnog automata se mjenjaju pozivanjem odgovarajućih funkcija, koje su takođe deklarisane u ovom fajlu (biće objašnjene kasnije), a definisane u **mwser_capmt_ecme.c**. Takođe, u ovom fajlu su deklarisane i privatne funkcije ECME podmodula, koje su takođe definisane u **mwser_capmt_ecme.c** fajlu i biće objašnjene kasnije.

Kada konačni automat primi da se desio neki od tri moguća događaja, on će u zavisnosti od toga u kom je trenutnom stanju pozvati neku od funkcija.

Na Slici 4.10. je dat grafički prikazan način funkcionisanja taska, a u Tabeli 4.10. je dato objašnjenje tih funkcija.

U fajlu **mwser_capmt_ecme.h** je definisana struktura **FilterEcmData**, u koju se smještaju filtrirani podaci. U tom fajlu su deklarisane i funkcije koje se pozivaju iz drugih modula, kada ti moduli nešto traže od ECME podmodula. One se definisane u

mwser_capmt_ecme.c fajlu, kao i sve druge funkcije u modulu, i počinju sa **mwser_capmt_ecme** u imenu.

mwser_capmt_ecme_Init	Inicijalizuje ECME podmodul. Kreira muteks, task i sve filtere stavlja u početni stanje da ne filtriraju.
mwser_capmt_ecme_Deinitialize	Uništava modul. Zaustavlja sve procese filtriranja, briše task i red za poruke ECME podmodulu.
mwser_capmt_ecme_StartFilter	Kreira filtar. Definiše moguća stanja filtra, i kreira mašinu stanja za njega, kao i transportni tok i PID za koji se definiše filtar.
mwser_capmt_ecme_StopFilter	Zaustavlja filtar.
mwser_capmt_ecme_DumpECME	Ispisuje podatke o svim filtrima koji rade.

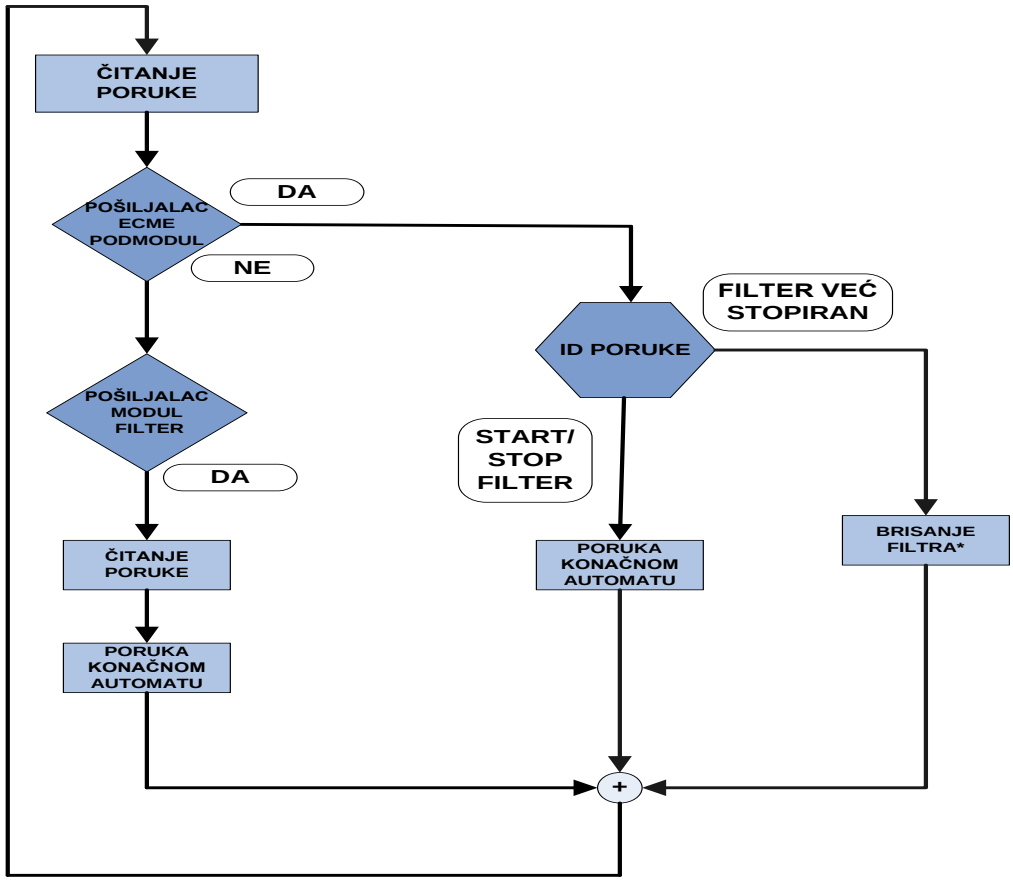
Tabela 4.9: Osnovne funkcije modula

U globalu, ECME podmodul funkcioniše kao svaki drugi modul sa taskom. Dakle, kada neki drugi modul traži da ECME pokrene neku akciju, on pošalje poruku u red za poruke. Poruka se šalje funkcijom **ecme_SendMessage.** U funkciji je definisan pokazivač na strukturu **hal_mq_Message**, čija se polja popunjavaju, i koja se funkcijom **hal_mq_Write** šalje kao poruka ECME podmodulu. Struktura sadrži ime pošiljaoca, ID poruke i sadržaj.

U tasku se (ime task funkcije je **pCapmtEcmeTaskFunc**), posle deklaracije lokalnih promenljivih pročita poruka iz reda za poruke. Zatim se uzima sadržaj iz poruke. Zatim se proverava koji modul je pošiljalac poruke. Ako pošiljalac nije sam ECME podmodul ili **Filter** modul, neće se ništa desiti.

Ako je pošiljalac poruke sam ECME podmodul, proveri se ID poruke pomoću ***switch*** strukture. Ako je u poruci dato (u smislu naredbe) da se filtar pokrene ili zaustavi, šalje se odgovarajući događaj mašini stanja pozivom funkcije **hal_stm_SendEvent** sa pogodnim argumentima. Ako je u poruci rečeno (u smislu informacije) da je filtar stopiran filtar, isti će biti izbrisan iz spiska aktivnih.

Poruku u task može poslati i modul **Filter**, kada završi filtriranje programa. Tada se poziva funkcija **readFilterData** koja čita sadržaj poruke iz niza u koji **Filter** stavlja dobijene podatke. Zatim se ti podaci (pozivom funkcije **ecme_FilterNotification**) šalju aplikaciji koja ih koristi za dešifrovanje programa. Funkcionisanje taska je prikazano na Slici 4.10.



** brisanje filtra iz spiska aktivnih*

Slika 4.12: Funkcionisanje taska ECME podmodula

Bitno je napomenuti da se proces filtriranja startuje i stopira istom funkcijom **ecme_ChangeFiltar**. Na osnovu ulazne BOOL varijable se određuje da li se traži zaustavljanje i pokretanje filtriranja filtera iz spiska filtera. Koji će filtrar biti pokrenut određuje se drugim ulaznim argumentom **u8FilterId**. Dakle, ovom funkcijom se pokreće tj. zaustavlja filtar čiji je ID dat u argumentu.

Sve funkcije koje se pozivaju iz mašine stanja su definisane sa istim argumentima: pokazivač na promjenljivu koja predstavlja stanje mašine stanja, i void pokazivač na podatak koji je ulazni, i u zavisnosti od funkcije do funkcije ima različitu ulogu.

ECME podmodul, naravno, ima i određeni broj privatnih funkcija, od kojih su neke već pomenute.

MWSER_CAPMT_ECME_StartFilter	Startuje proces filtriranja filtra čiji je ID dat kao argumernt funkcije. Time se menja stanje mašine stanja iz **IDLE** u **RUNNING**.
MWSER_CAPMT_ECME_StopFilter	Stopira proces filtriranja filtra čiji je ID dat kao argumernt funkcije. Time se menja stanje mašine stanja iz **RUNNING** u **IDLE**.
MWSER_CAPMT_ECME_TableReceived	Poziva se kada je primljen filtriran podatak (ulazni argument funkcije je pokazivač na taj podatak). Taj podatak se, zajedno sa odgovarajućim informacijama šalje u red sa filtriranim podacima za dati filtar. Ne menja stanje mašine stanja.
MWSER_CAPMT_ECME_Ignore	Ne dešava se ništa, stanje mašine stanja se neće promeniti.

Tabela 4.10: Funkcije koje se pozivaju iz mašine stanja

5. Novo rešenje CAPMT modula

Analizom postojećeg rešenja CAPMT modula nađeni su mnogi nedostaci. Pre svega, opšti utisak je da modul može mnogo jednostavnije da se uradi. Rešenja pojedinih zadataka koje modul treba da obavi su nepotrebno zakomplikovana. Isti problemi su rešavani na različite načine. Nije poštovan jedinstveno pravilo za davanje imena funkcija i promenljivih (tzv. *coding style*).

Dobro su rešeni DB i IPP podmoduli. Oba podmodula je realizovana efikasno i jednostavno. Zato ih ne treba menjati.

Sve strukture i nabrajanja su dobro osmišljena i u velikoj većini će ostati isti. Dobro modeluju stanja i podatke sa kojima modul rukuje. Menjaće se samo onoliko koliko veće promene drugih elemenata programa to budu zahtevale.

Bez obzira na promene u CAPMT modulu, sprega sa drugim modulima ne sme da se promeni. Tačnije, drugi moduli ne smeju da primete nikakvu promenu u modulu .

5.1. Glavni i CTR podmodul

Analizom zadataka koje CAPMT modul treba da obavi, došlo se do zaključka da je konačni automat u suštini nepotreban. Da bi modul obavljao sve potrebne funkcije, treba stalno da bude u stanju PROCESSING, pa je postojanje ostalih stanja praktično suvišno. Zato je prva promena izbacivanje konačnog determinističkog automata

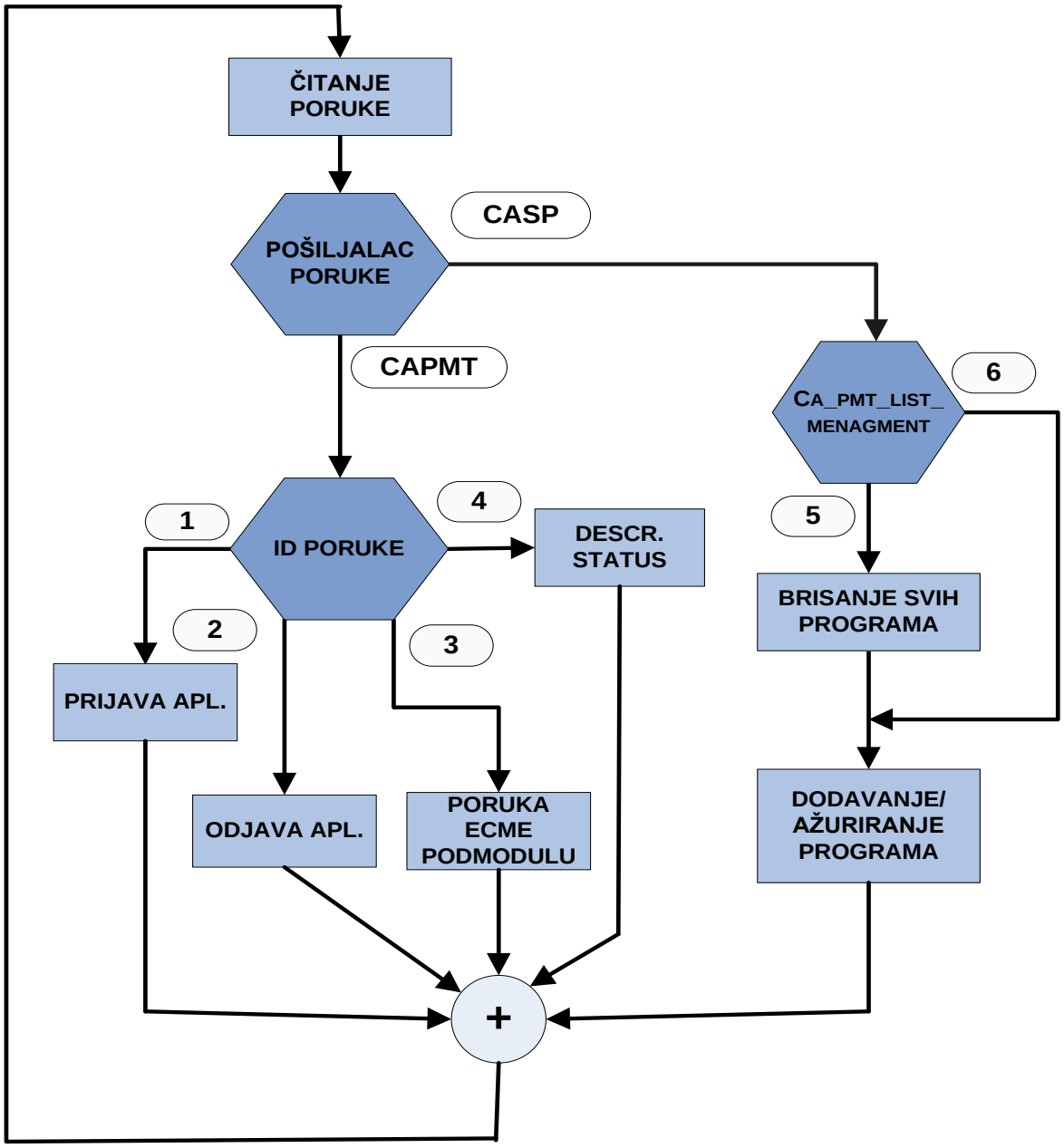
Analizom zahteva koji taskovi treba da ispune, došlo se do zaključka da njihova struktura treba da bude slična. Zato je definisana nova arhitektura na osnovu kojih će se uraditi novi taskovi. Ta arhitekutra je definisana na sledećim pravilima:

- u tasku, posle čitanja poruke koja je poslata se vršu njena analiza, i na osnovi nje se poziva adekvatna funkcija. Dakle, nema nikakve dalje obrade same poruke u task funkciji;
- prvo se u jednoj *switch* strukturi proveri ko je pošiljalac poruke. Za svaki *case* se napravi još jedna *switch* struktura u kojoj se proverava se polje koje pokazuje šta se konkretno traži od modula. Na osnovu tog polja pozivaće se određena funkcija. Ako program uđe u *default* u *switch* strukturi, znači da je došlo do neke greške, i o tome se pri debagovanju šalje poruka na računar (ako je slanje poruka uključeno kao opcija). Ovo znači izbacivanje konačnog deterministički automata, što znači da iz svih funkcija treba izbaciti sve što ima veze sa konačnim automatom.

Kako je ovo realizovano na slučaju glavnog taska, prikazano je na Slici 5.1.

Operacije sa programima (dodavanje, brisanje, ažuriranje) su izvedene loše. Za ove operacije se koriste funkcije **CAPMT_CTR_ReadyPrepare, CAPMT_CTR_ProcessingProcess** i **CAPMT_CTR_ProcessingEnd**. One se pozivaju preko konačnog automata, što predstavlja dodatnu kompleksnost. Kod dodavanje/ažuriranja, prvo se na kompleksan način proveri da li je program šifrovan, a zatim se odradi samo dodavanje (ažuriranje) pozivom funkcije iz DB modula. Zatim se šalju poruke aplikaciji za dešifrovanje. Kada treba da se doda program koji je jedni u listi, ili prvi u novoj listi (isti niz koraka), prvo se pozove funkcija koja sve programe u bazi stavi u stanje DELETE, a zatim se prethodno pomenuta procedura ponovi. Program koji je i stanju DELETE će biti izbrisan u funkciji **CAPMT_CTR_ProcessingEnd**. Uz to biće zaustavljeno filtriranje programa i poslata poruka u aplikaciji. Ovo čini da se program koji treba da bude prvi ili jedini u listi nađe na drugom mestu. Uz to, kod ove tri funkcije je nepregledan. Zato je odlučeno da se operacije sa programima reše na potpuno drugačijim osnovama. Promene obuhvataju task i dodavanje novih funkcija.

Posle čitanja poruke, u *switch* strukturi se proveri koji moduo šalje poruku. Ako je to CASP modul, to znači da je CAM primio novu ca_pmt tabelu koju CASP prosleđuje CAPMT modulu. Za obradu svih mogućih vrednosti polja **list_menagment** je dovoljno imati dve funkcije. Za vrednosti **first** i **only**, znači da treba obrisati sve dosadašnje programe, i dodati novi na prvo mesto. U slučaju vrednosti **first** sledeći programi će posle prijema tabela biti dodavani na sledeća slobodna mesta. Ostale vrednosti se mogu svesti na to da je potrebno dodati novi program na sledeće slobodno mesto u bazi, ili ažurirati postojeći.



1 – prijava aplikaciju
2 – odjava aplikacije
3 – poruka ECME podmodulu
4 – slanje poruke o statusu dešifrovanja programa
5 – brisanje svih programa
6 – dodavanje novog programa

Slika 5.1: Blok-šema novog taska

Zato su napravljene dve nove funkcije: Jedna za brisanje svih programa (**mwser_capmt_DeletingAllPrograms**) i za dodavanje i ažururanje programa (**mwser_capmt_AddingProgram**).

Funkcija **mwser_capmt_DeletingAllPrograms** se poziva za **first** i **only** vrednosti polja **list_menagment**. U funkciji se prvo zaustavi filtriranje programa pozivanjem funkcije iz ECME podmodula. Zatim se izbriše celi niz aktivnih programa u DB podmodulu i za svaki se šalje poruka aplikaciji. Poruka aplikaciji se šalje novom funkcijom **mwser_capmt_NotifyApp4DeletePrg**.Ova funkcija prolazi kroz sve prijavljene aplikacije i proverava da li je program bio u listi aplikacije. Ako jeste, šalje joj poruku da je program izbrisan i briše ga iz liste. Posle ove funkcije se poziva **mwser_capmt_AddingProgram.** Ovim je postignuto da novi program bude jedini i na prvom mestu.

Za ostale vrednosti polja **list_menagment** pozvaće se samo funkcija **mwser_capmt_AddingProgram**. Ona dodaje novi na prvo slobodno mesto ili ažurira postojeći program u nizu aktivnih programa. Ovo se vrši pozivanjem pogodnih funkcija iz DB podmodula. Ako se radi o ažuriranju programa, zaustaviće se njegovo filtriranje. Zatim se šalje se poruka svim prijavljenim aplikacijama sa svim bitnim podacima o novom/ažuriranom programu.

U slučaju da je pošiljalac poruke sam CAPMT modul, proveriće se ID poruke. Na osnovu toga će se pozivati određena funkcija (Tabela 5.1).

ID PORUKE	FUNKCIJA KOJA SE POZIVA
EV_CAPMT_CTR_FSM_CA_REGISTER	**mwser_capmt_RegisterCaApp**
EV_CAPMT_CTR_FSM_CA_UNREGISTER	**mwser_capmt_UnregisterCaApp**
EV_CAPMT_CTR_FSM_ECM_FILTER	**mwser_capmt_EcmFiltering**
EV_CAPMT_CTR_FSM_DESCRAMBLING_STATUS	**mwser_capmt_DescramblinStatus**

Tabela 5.1: Pozivanje funkcija na osnovu ID-ja poruke

Dodavanjem ove dve funkcije operacije sa programima su bitno pojednostavljene. Zbog ovog su izbačene funkcije **CAPMT_CTR_ReadyPrepare**, **CAPMT_CTR_ProcessingProcess** i **CAPMT_CTR_ProcessingEnd**.

Zbog brze provere da li je i na kom nivou program šifrovan, uvedeni su novo nabrajanje **CAPMT_SCRAMBLING_TYPE** (Slika 5.2) i nova funkcija **mwser_capmt_ScramblingTypePrg**. Ovo nabrajanje ima tri vrednosti: za slučaj kada program nije šifrovan, kada je šifrovan na nivou programa i kada je šifrovan na nivou elementarnih tokova. Funkcija vraća jednu od vrednosti nabrajanja u zavisnosti da li je i na kom nivou program šifrovan. Provera se vrši proverom polja ca_pmt tabele koju modul dobija kada korisnik izabere novi program. Početno stanje promenljive koja se vraća kao rezultat funkcije je da program nije šifrovan. Prvo se proveri polje koje označava da li je program šifrovan na nivou programa. Ako nije, proveravaju se ta polja za elementarne tokove. Dakle, ako je nađeno da je program šifrovan stanje promenljive će se promeniti u zavisnosti na kom nivou je šifrovan.

```
typedef enum
{
   CAPMT_PROGRAM_NOT_SCRAMBLED = 0,     /**  Program not scrambled  */
   CAPMT_PROGRAM_LEVEL_SCRAMBLED,       /**  Program scrambled on prg.lev.*/
   CAPMT_ES_LEVEL_SCRAMBLED,            /**  Program scrambled on ES lev. */
} CAPMT_SCRAMBLING_TYPE;
```

Slika 5.2: Definicija nabrajanja **CAPMT_SCRAMBLING_TYPE**

Realizovana je nova funkcija **mwser_capmt_NotifyApp** koja po potrebi šalje poruku aplikaciji. Kao što je već rečeno, neke aplikacije se prijavljuju na CAPMT modul i one nose određene CaSysId polja. Programi takođe nose CaSysId polja. Ova funkcija proverava podudaranje CaSysId polja programa i aplikacija, i ako se desi podudaranje šalje poruku aplikaciji o novom programu. Prvo se pozivom gorepomenute funkcije proveri da li je i na kom nivou program šifrovan. Ako je program šifrovan na nivou programa vrši se provera CaSysId polja sa programskog nivoa i CaSysId polja koje nosi aplikacija. Ako je program šifrovan na nivou elementarnih tokova, vrši se ista provera samo sa CaSysId poljima koje su na nivou elementarnih tokova. Za nađeno poklapanje CaSysId poljima, promenljivoj u kodu se dodeli adekvatna vrednost. Ako te promenljiva imaju jednu od traženih vrednosti šalje se poruka aplikaciji u vidu strukture koja nosi sve bitne podatke o programu. Upoređivanjem sa EMM porukama koje nosi program, praktično se proverava da li korisnik ima pravo da gleda izabrani program.

Takođe, realizovana je funkcija **mwser_capmt_NotifyApp4DeletedPrg** koja se poziva se u funkciji **mwser_capmt_DeletingAllPrograms.** Ona zaustavlja filtriranje programa koji treba da bude obrisan i šalje poruke svim aplikacijama o tome.

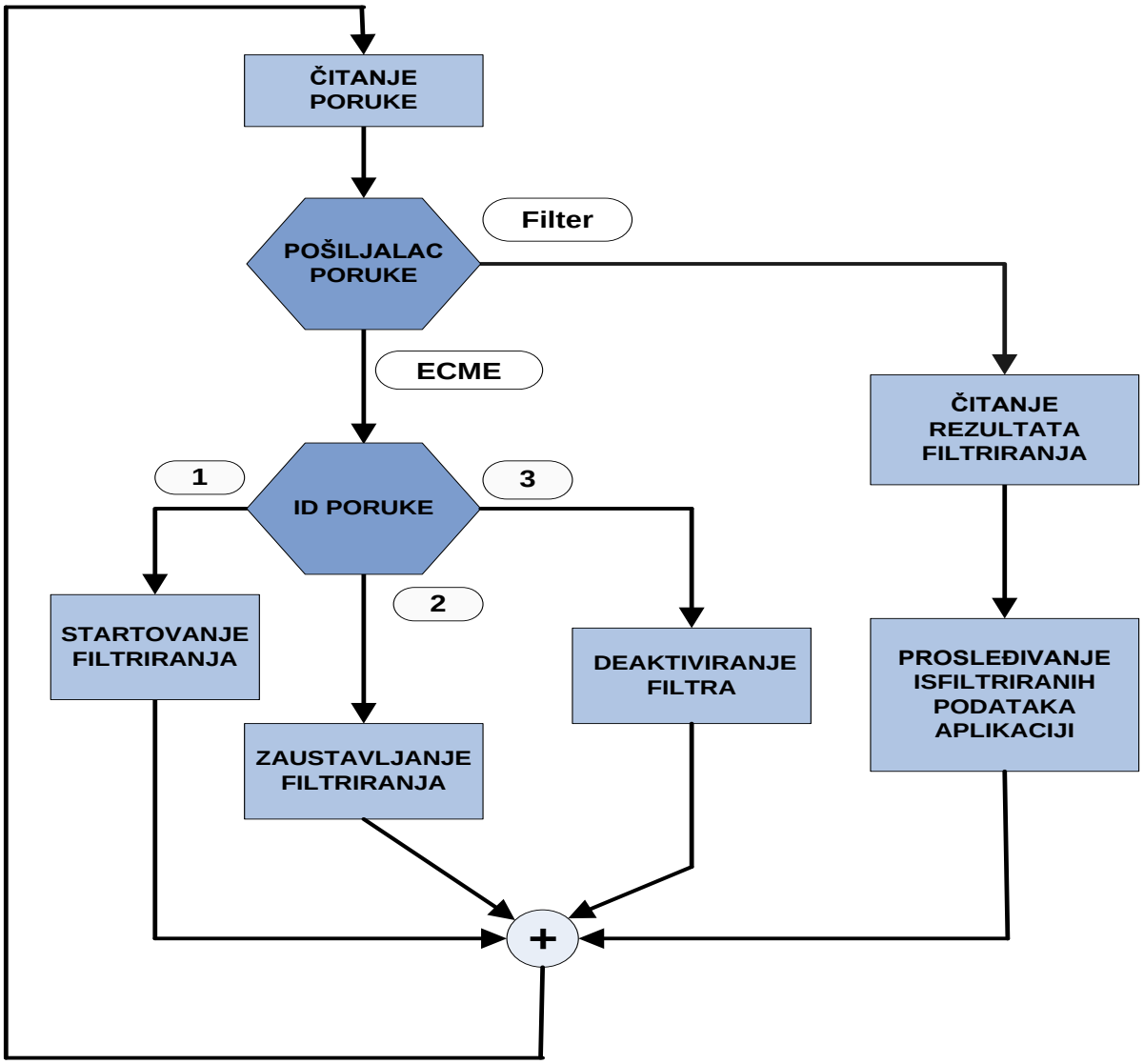
Analizom je utrvđeno da su postojeće funkcije za prijavu i odjavu aplikacija neadekvatne. Za svako od tri stanja je postoje posebne funkcija, koje su iste ili vrlo slične. S obzirom da više nema konačnog determinističkog automata, treba uvesti po jednu funkciju za prijavu i odjavu aplikacija. Pošto su postojeće neadekvatne, realizovanje su nove.

Funkcija za prijavu aplikacija je **mwser_capmt_RegisterCaApp.** Svi podaci o aplikaciji se nalaze u jednoj strukturi, koja se na početku ove funkcije doda u niz svih prijavljenih aplikacija. Zatim se proveri da li aplikacija ima CaSysId polja. Ako ima CaSysId polja se dodaju u listu svih CaSysId polja svih aplikacija koje su prijavljene na modul i ova struktura se šalje CASP modulu. Ovo je slučaj sa aplikacijom za dešifrovanje. Ako aplikacija nema CaSysId polja, ona se samo doda u listu, i pošalju joj se poruke o svim aktivnim programiima. Ako se aplikacija prva prijavljuje na modul, šalje se poruka glavnom tasku da je sesija otvorena.

Funkcija za odjavu aplikacija **mwser_capmt_UnregisterCaApp** u suštini radi suprotno od funkcije za prijavu aplikacija. Ona briše aplikaciju iz spiska prijavljenih, i o tome šalje poruku CASP modulu. Takođe, oslobađa memoriju na kojoj su bili upisane CaSysId polja te aplikacije.

5.2. ECME podmodul

U ECME modulu su napravljene velike promene, tako da on sada izgleda bitno drugačije. Realizivan je novi task, prema pravilima koja su uvedena za taskove. Dijagram novog taska je prikazan na Slici 5.3.



1 - *startuj filtriranje programa*

2 – *zausavi filtriranje programa*

3 – *filter stopiran*

Slika 5.3: Dijagram ECME taska

Sa Slike 5.3. se vidi da je struktura taska slična glavnom tasku, samo što je ovaj jednostavniji. Na početku se pročita poruka i pogleda se koji modul je pošiljalac poruke. Za slučaj da je to ECME podmodul, to znači da je ili pozvana jedna od njegovih API funkcija (slučaj startovanja i zaustavljanja filtriranja) ili je iz funkcije za zaustavljanje fitriranja upućena poruka u task da je filtriranje zaista zaustavljeno, i da treba deaktivirati filtar.

Kada korisnik izabere šifrovan program, CAPMT modul pozivom API funkcije ECME podmodula **mwser_capmt_StartEcmeFiltering** traži početak filtriranja programa. U API funkciji se kreira novi filtar u nizu aktivnih i pošalje se poruka tasku. Kreiranje novog filtra obuhvata ubacivanje u niz strukture koja sadrži sve bitne podatke o programu i procesu filtriranja. Iz taska se poziva funkcija **mwser_capmt_ecme_StartFiltering** koja šalje poruku modulu za filtriranje da započne filtriranje novog programa. Argument ove funkcije je pozicija filtra u prethodno pomenutom nizu. U slučaju kad prestane potreba za filtriranjem programa, dešava se sličan proces. Naravno, pozivaju se druge funkcije (**mwser_capmt_StopEcmeFiltering** i **mwser_capmt_ecme_StopFiltering**). Posle slanja poruke modulu Filter da zaustavi filtriranje, šalje se i poruka tasku da je filtriranje zaustavljeno i da obriše filtar iz niza aktivnih.

Modul Filter periodčno šalje poruke koje je isfiltrirao iz toka programa koji je korisnik izabrao. Te poruke šalje ECME podmodulu. One se prvo očitaju (funkcijom **mwser_capmt_ecme_ReadFilterData**), a zatim se upakuju u strukturu pogodnu za slanje i pošalju aplikaciji na dalju obradu (**mwser_capmt_ecme_TableReceived**).

Ove funkcije su slične funkcijama u staroj realizaciji. U odnosu na prethodnu realizaciju, moralo je da bude izbačeno sve što ima veze sa konačnim automatima. Takođe, izvšeno je više sitnih modifikacija radi preglednosti i jednostavnosti koda.

6. Analiza novog rešenja

Analiza novog rešenja CAPMT modula obuhvata poređenje sa postojećim rešenjem sledećih parametara:

- provera da li CAM defifruje sadržaje koje korisnik ima pravo da gleda;
- praćenje ispisa sa CAM-a na računaru radi traženja grešaka;
- merenje brzine rada modula;
- upoređivanje veličine koda (broja linija koda);
-vizuelni pregled koda radi utvđivanja preglednosti i jednostavnosti koda;

Prva i druga provera mogu se raditi zajedno. U CAM se stavi kartica koja omogućava RS-232 komunikaciju sa računarom. CAM se stavi u prijemnik i otvori se program koji prati ispise koje CAM šalje. Ispisi su definisani u kodu, i obično se u njima šalju vrednosti promenljivih bitnih za deo koda koji se trenutno izvršava. Može se definisati da se iz svih modula ispisuju samo greške. Na prijemnik se pusti tok sa više šifrovanih i nešifrovanih kanala. Testiranje je pokazalo da CAM dešifruje kanale koje treba da dešifruje, i da u ispisima koje šalje nema grešaka. To znači da da provera prava pristupa servisu radi kako treba i da novo rešenje ne unosi grešku u sistem.

Vrlo bitan parametar kvaliteta je brzina rada modula. Konkretno, za ovaj modul je bitna brzina rada pri prebacivanju na šifrovani servis. Pošto su poznata mesta u kodu gde program ulazi i izlazi iz modula, na tim mestima se poziva funkcija za ispis trenutnog vremena.

Oduzimanjem ova dva vremena se dobije vreme rada modula. Testiranje je pokazalo da između ulaska i izlaska iz modula prođe 0.9 sekundi. Prethodno rešenje je radilo 1.3 sekunde.

Veličine koda je određena jednostavnim softverski alatom *CodeAnalyzer*. Postojeće rešenje ima 4865, a novo 4032 linje čistog koda (ne računaju se komentari i prazne linije).

Vizulenim pregledom je nedvosmisleno ustvređeno da je novi kod mnogo pregledniji i jednostavniji za razumenvanje. Takođe, stavljeno je više komentara, koji objašnjavaju šta se želelo uraditi. Imena novih funkcija i promenljivih su data tako da se na osnovu imena može zaključiti njihova uloga. Sve ovo je značajno pre svega zbog nekog sledećeg programera koji će iz nekog razloga pregledati kod ovog modula.

7. Zaključak

U ovom radu je predstavljno jedno rešenje modula za kontrolu pristupa sadržajima digitalne televizije, kada se kontrola vrši preko CAM modula.

Predstavljeni su svi bitni aspekti projekta. Bilo je reči o digitalnoj televiziji, sa naglaskom na pojmove bitne za ovaj rad (standardi, tabele itd.). Zatim je dat grub pregled softvera na CAM-uu, uloga CAPMT modula i njegove veze sa drugim modulima.

Najviše reči je naravno bilo o postojećem i novom rešenju. Oba rešenja su predstavljena detaljno. Objašnjene su svi bitne funkcije, nabrajanja i strukture. Opis novog rešenja obuhvata analizu postojećeg, i sve promene koje su unesene.

Na kraju je opisano ocenjivanje novog rešenja u svim bitnim kategorijama za ovu vrstu projekta.

Može se reći da je projekat ispunio svoju svrhu. Dato je novo rešenje koje u svim bitnim kategorijama nadmašuje potojeće. Ovo rešenje će biti predloženo za komercijalnu upotrebu, što je poseban kvalitet ovog rada.

8. Literatura

[1] http://en.wikipedia.org/wiki/Digital_television;

[2] http://en.wikipedia.org/wiki/Conditional_access;

[3] Dokumentacija kompanije RT-RK d.o.o, kurs iz digitalne televizije;

[4] http://en.wikipedia.org/wiki/Conditional-access_module;

[5] http://en.wikipedia.org/wiki/MQX