УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Милан Ковачевић

Реализација подршке за управљање системом датотека на вишепроцесорској платформи

МАСТЕР РАД

Нови Сад, Октобар 2013.

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**:		Милан Ковачевић	
Ментор, **МН**:		Проф. Др Јелена Ковачевић	
Наслов рада, **НР**:		Реализација подршке за управљање системом датотека на вишепроцесорској платформи	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2013	
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)		7/45/11/3/10/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		Наменски систем, интегрисано развојно окружење, систем датотека, вишепроцесорска платформа	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У овом раду реализована је подршка за управљање системом датотека на вишепроцесорској платформи. Приказан је начин подешавања система датотека у оквиру развојног окружења и начин слања и пријема података из окружења на платформу. Дата је реализација модула од значаја за овај рад и начин верификације истих.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	Проф. др Растислав Струхарик	
	Члан:	Проф. др Илија Башичевић	Потпис ментора
	Члан, ментор:	Проф. др Јелена Ковачевић	

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**:		Milan Kovačević	
Mentor, **MN**:		PhD Jelena Kovačević	
Title, **TI**:		Implementation of file system management support on an multiprocessor platform	
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**:		2013	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/45/11/3/10/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Integrated development environment, embedded system, file system, multiprocessor platform	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper contains implementation of file system management support on an multiprocessor platform. File System configuration inside integrated development environment and data transfer on target platform are the main parts of this support. Realization of the modules relevant to this paper is presented, as well as verification of those modules.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	PhD Rastislav Struharik	
	Member:	PhD Ilija Bašičević	Menthor's sign
	Member, Mentor:	PhD Jelena Kovačević	

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Никола Јорговановић

Студент:	Милан Ковачевић	Број индекса:	Е2 4/2012
Област:	Наменски системи		
Ментор:	Проф. др Јелена Ковачевић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ МАСТЕР РАДА:

Реализација подршке за управљање системом датотека на вишепроцесорској платформи

ТЕКСТ ЗАДАТКА:

Реализовати програмску подршку за управљање системом датотека на вишепроцесорској платформи. Подршка треба да омогући кориснику подешавање система датотека у оквиру интегрисаног развојног окружења, као и да обезбеди спрегу са циљном платформом. Спрега са платформом подразумева упис датотека на платформу, као и читање садржаја из система датотека.

Руководилац студијског програма:	Ментор рада:

Примерак за: o - Студента; o - Ментора

Zahvalnost

Želim da se zahvalim mojoj porodici na podršci, brizi i strpljenju tokom mog školovanja.

Srdačno se zahvaljujem svom mentoru Prof. Dr Jeleni Kovačević i Prof. Dr Vladimiru Kovačeviću na savetima i usmerenjima pruženim tokom studija.

Na kraju se zahvaljujem svim kolegama na pruženoj pomoći u izvođenju praktičnog dela istraživanja, kao i svima onima koji su na bilo koji način doprineli izradi ovog master rada.

SADRŽAJ

1. Uvod....1
2. Teorijske osnove....4
2.1 Eclipse platforma....4
2.1.1 RCP aplikacije....5
2.1.2 Osnovni podsistemi Eclipse platforme....6
2.1.3 Radno okruženje....7
2.1.4 Alati za spregu sa korisnikom....8
2.1.5 Radni sto....10
2.2 Namenski sistemi....11
2.3 Digitalni signal procesori....11
2.4 SIDE razvojno okruženje....12
2.5 Sonnet platforma....13
3. Koncept rešenja....15
3.1 Podešavanje datoteka u okviru razvojnog okruženja....16
3.2 NVM uređivač....16
3.3 Sistem datoteka....18
3.3.1 Datoteka za podešavanje parametara platforme....19
3.3.2 Datoteka za učitavanje adresnog prostora sistema....20
3.3.3 Datoteka pokretačkog programa....22
3.3.4 Datoteka ključeva za šifrovanje podataka....22
3.3.5 Datoteke programske podrške....23
3.4 Validator datoteka....25
3.5 Raspoređivač sistema datoteka....25

3.6 Stvaranje binarnih datoteka 27
3.7 Sprega okruženja i platforme 28
3.7.1 Sonnet Proxy Server modul 28
3.7.2 Upis sistema datoteka na platformu 28
3.7.3 Čitanje sistema datoteka sa platforme 29
4. Programsko rešenje 30
4.1 Priključak grafičke korisničke sprege 30
4.1.1 Klasa EepromConfigurationEditor 30
4.1.2 Klasa CommonEepromPage 31
4.1.3 Klasa GeneralSettingsPage 31
4.1.4 Klasa FileSystemPage 32
4.1.5 Klasa FileDetailsPage 33
4.1.6 Klasa BlockHeaderDetailsPage 34
4.1.7 Klasa StatisticsPage 34
4.2 Priključak modela programske podrške 35
4.2.1 Klasa EepromProject 35
4.2.2 Klasa FirmwareFile 36
4.2.3 Klasa ReadElfProcessForFirmwareFiles 36
4.2.4 Klasa NVMBootLoaderFile 37
4.2.5 Klasa ReadElfProcessForBootLoader 37
4.2.6 Klasa NVMScheduler 38
4.2.7 Klasa LFSREncryptionAlgorithm 38
4.2.8 Klasa DownloadNVMImage 39
4.2.9 Klasa UploadNVMImage 39
5. Ispitivanje i verifikacija 40
5.1 Ispitivanje rezervisanih datoteka 40
5.2 Ispitivanje datoteka programske podrške 41
5.3 Ispitivanje upisa i čitanja sistema datoteka 42
6. Zaključak 44
7. Literatura 45

SPISAK SLIKA

Slika 1. Struktura Eclipse razvojnih alata 6
Slika 2. SIDE razvojno okruženje 13
Slika 3. Sonnet platforma 14
Slika 4. Dijagram upravljanja sistemom datoteka iz radnog prostora Sonnet razvojnog okruženja 15
Slika 5. Višestranični NVM uređivač 17
Slika 6. Struktura sistema datoteka 19
Slika 7. Datoteka za učitavanje adresnog prostora sistema.....21
Slika 8. Izvršna ELF datoteka podeljena na zaglavlja i segmente 24
Slika 9. Raspoređen sistem datoteka 27
Slika 10. Dijagram razmene poruka prilikom upisa sistema datoteka 29

SPISAK TABELA

Tabela 1. Rezultati ispitnih slučajeva rezervisanih datoteka 41
Tabela 2. Rezultati ispitnih slučajeva datoteka programske podrške 42
Tabela 3. Rezultati ispitnih slučajeva prilikom upisa i čitanja sistema datoteka 43

SKRAĆENICE

IDE - *Integrated Development Environment*, Integrisano razvojno okruženje

RCP - *Rich Client Platform,* Alat za integraciju komponenata programske podrške

API - *Application Programming Interface,* Aplikativna programska sprega

DSP - *Digital Signal Proccesor,* Digitalni signal procesor

CRC - *Cyclic Redundancy Check,* Ciklična provera

RAM - *Random Access Memory,* Memorija sa slučajnim pristupom

ROM - *Read-Only Memory,* Memorija za čitanje

EEPROM -*Electrically Erasable Programmable Read-Only Memory,* Obrisiva programabilna memorija za čitanje

NVM - *Non-Volatile Memory,* Postojana memorija

ELF - *Executable Linking Format,* Izvršna datoteka

XML - *Extensible Markup Language,* Proširiv jezik za opisivanje

TCP - *Transfer Client Protocol,* Protokol za razmenu podataka

1. Uvod

U ovom radu je prikazana realizacija programske podrške za upravljanje sistemom datoteka na Sonnet višeprocesorskoj platformi u okviru Sonnet integrisanog razvojnog okruženja (eng. SIDE – Sonnet Integrated Development Environment) zasnovanog na Eclipse platformi. Upravljanje sistemom datoteka podrazumeva dodavanje, brisanje, ispitivanje, podešavanje parametara datoteka u okviru razvojnog okruženja, kao i upis i čitanje datoteka sa Sonnet platforme.

Integrisano razvojno okruženje je integrisan skup sistemskih programa (programskih alata) koji se koristi za konforan i pouzdan razvoj programske podrške [1]. Pre pojave integrisanog razvojnog okruženja programeri su za razvoj programske podrške imali na raspologanju komandni procesor (na UNIX sistemima on nosi naziv „školjka“, eng. shell), niz sistemskih komandi (kao što su komanda za promenu radnog kataloga, prikaz sadržaja datoteke, kompajler, povezivač i slično) i niz uslužnih programa (kao što je program za uređivanje teksta, program kontrolisanog izvršavanja, eng. debugger, i dr). Sistemske komande i uslužni programi se tradicionalno pozivaju pomoću tzv. komandne linije, unosom naziva komande i njenih argumenata.

Za razvoj programske podrške je karakteristično da je to iterativan proces. Na početku programer poziva program za uređivanje teksta (eng. Editor, u nastavku uređivač), unosi kod programa, zatim napušta uređivač, prevodi svoj kod kompajlerom, pregleda izveštaj o prevođenju i u slučaju sintaksnih grešaka, vrši ispravke, ponovo poziva kompajler, sve dok ne otkloni sve sintaksne greške. Kada se prevođenje uspešno završi, prelazi u fazu povezivanja, a nakon toga i u faze otkrivanja i otklanjanja grešaka, odakle se opet može vratiti na početak. Za svo ovo vreme programer uvek nanovo mora da kuca komande komandne linije, što dovodi do gubitka vremena, zamora, gubitka koncentracije i pojave grešaka u programima.

Suštinski kvalitet koji donosi IDE je da su svi uslužni programi i komande potrebne za razvoj programske podrške, integrisani sa stanovišta korisnika. Nema potrebe za napuštanjem jednog uslužnog programa radi izdavanja komande ili prelaska na drugi ulužni program. Umesto toga, prostim pritiskom funkcionalnog tastera ili klikom miša se iz tekuće faze prelazi na novu, željenu fazu, fazu razvoja programske podrške. Dakle, sve komande i sve usluge razvoja programske podrške su stalno na raspolaganju programeru i on ih sa lakoćom bira prema svom nahođenju. Ovim se dramatično povećava produktivnost razvoja programske podrške i smanjuje broj grešak. Drugim rečima, povećava se produktivnost razvoja programske podrške i smanjuje broj grešaka, odnosno povećava se pouzdanost programske podrške. Ovo je posebno važno sa sisteme u realnom vremenu, kod kojih je pouzdanost programske podrške od vitalnog značaja.

Eclipse platforma za razvoj aplikacija nudi generičku podršku za pisanje razvojnih okruženja i omogućava brz razvoj funkcionalno bogatih, stabilnih i lako upotrebljivih razvojnih okruženja [2]. Korišćenjem ove podrške, na Katedri za Računarsku Tehniku i Računarske Komunikacije Fakulteta Tehničkih Nauka, razvijeno je SIDE integrisano razvojno okruženje za razvoj namenske programske podrške za Sonnet platformu. Sonnet je sistem za digitalnu obradu signala na čipu (eng. SoC – System on a chip), dizajniran za aplikacije kompletno opremljenog slušnog aparata male snage. Takvi sistemi obično podrazumevaju integrisano kolo koje se sastoji od računarskih komponenti (memorije sa slučajnim pristupom RAM - Random Access Memory ili memorija za čitanje ROM - Read Only Memory, mikroprocesora, periferne sprege, ulazno-izlaznog podsistema, kao i ostalih komponenti koje obuhvataju jedan računarski sistem) integrisanih u jedan procesor. Sonnet je višeprocesorski sistem sastojan od pet digitalnih signal procesora, gde jedan od njih ima ulogu mikrokontrolera. Za obradu signala velik značaj predstavlja upravljanje sistemom datoteka. Realizovana podrška za upravljanje sistemom datoteka podrazumeva rad sa datotekama u okviru posebnog uređivača u razvojnom okruženju, kao i spregu između okruženja i platforme.

Rad je sačinjen od sedam poglavlja.

Prvo poglavlje sadrži osnovne podatke o radu i opis rada.

Drugo poglavlje sadrži sadrži opis Eclipse platforme, opis Sonnet platforme i SIDE razvojnog okruženja, kao i kratak istorijat, razvoj i opis namenskih sistema i digitalnih signal procesora.

U trećem poglavlju je dat koncept rešenja, odnosno način korišćenja razvojnog okruženja za potrebe podešavanja sistema dadoteka, i opis sprege sa ciljnom platformom.

Četvrto poglavlje sadrži opis programskog rešenja, odnosno realizacije funkcija korišćenih za rad sa sistemom datoteka u okviru razvojnog okruženja, što podrazumeva korišćenje

uređivača, čarobnjaka, pogleda, kao i komunikacije sa platformom prilikom operacija čitanja i upisa sistema datoteka.

Peto poglavlje daje opis ispitivanja i verifikacije rešenja.

U šestom poglavlju je u okviru zaključka dat pregled svega urađenog u radu kao i ideje za dalji razvoj.

Lista korišćene literature data je u sedmom poglavlju.

2. Teorijske osnove

U ovom poglavlju opisan je način korišćenja programskih alata koje pruža *Eclipse* razvojno okruženje za razvoj novih, integrisanih razvojnih okruženja kao što je SIDE. Takođe, date su teorijske osnove o namenskim sistemima, njihovom razvoju i primeni. Na kraju ovog poglavlja biće objašnjena arhitektura i način rada digitalnih signal procesora (eng. Digital Signal Processor).

2.1 Eclipse platforma

Eclipse je integrisano razvojno okruženje slobodno dostupnog koda (eng. Open source) napisano Java programskim jezikom. Kompanija IBM počela je sa projektom Eclipse 1998. godine sa ciljem da napravi zajedničku platformu za IBM alate i time privuče programere da koriste posrednika (eng. Middleware) zasnovanog na Javi. Kako bi privukli korisnike i proizvođače, odlučeno je da se napravi proširiva platforma za integraciju alata, odnosno da se napravi proširivo Java razvojno okruženje, u čemu su i uspeli. U 2001. godini IBM je napravio Eclipse proizvod slobodno dostupnog koda i uspostavio Eclipse konzorcijum koji je brojao ukupno devet kompanija. Eclipse verziju 2.0 i 2.1 mnoge kompanije su usvojile kao pogodno Java integrisano razvojno okruženje i platformu za integrisanje velikog broja različitih alata [3]. Svojom robusnošću i proširujućom arhitekturom, bogatom funkcionalnošću, Eclipse je podstakao korisnike i proizvođače da se pridruže njegovom daljem razvijanju i poboljšanju. Ono što Eclipse čini različitim od ostalih proizvoda slobodno dostupnog koda, odnosno ono šta je rezultovalo ogromnim prihvatanjem Eclipse-a od strane brojnih komercijalnih kompanija je Eclipse-ova javna licenca (eng. Eclipse Public Licence). Ova licenca omogućava prava na sav izvorni kod i besplatnu redistribuciju njihove tehnologije. To je rezultovalo velikom broju razvijenih alata, gde su mnoge kompanije učestvovale u proširenju okvira (eng. Framework) i standardizaciji alata. Trenutno Eclipse ima najveći uticaj u razvijanju namenskih sistema,

internet alata, u ispitivanju i profilisanju programske podrške, modelovanju itd. Mnoge kompanije iz ovih oblasti su pristupile Eclipse zajednici i međusobno se udružuju u rešavanju specifičnih zahteva njihovog domena. U užem smislu, pod Eclipse-om podrazumevamo Eclipse alate za razvoj programske podrške. Eclipse alati za razvoj programske podrške (*eng. Software Development Kit - SDK*) je kombinacija nekoliko Eclipse projekata: Eclipse platforme, Java razvojnih alata (*eng. Java Development Tools*) i okruženja za razvoj priključaka (eng. Plug-in Development Environment PDE). PDE okruženje obezbeđuje alate za stvaranje, razvoj, ispitivanje, otklanjanje grešaka Eclipse priključaka (eng. Plug-in), funkcionalnosti (eng. Feature, grupa međusobno povezanih priključaka) i proizvoda razvijenih pomoću alata za integrisanje komponenata programske podrške - RCP (eng. Rich Client Platform).

Koristeći ove alate, kompanije su u mogućnosti da razvijaju sopstvene alate koje prodaju kao komercijalni proizvod.

2.1.1 RCP aplikacije

Eclipse platforma je skup komponenti koje se mogu upotrebiti za izgradnju samostalnih RCP aplikacija [4]. RCP je platforma koja sadrži svu funkcionalnost potrebnu za razvoj novih integrisanih razvojnih okruženja i alata [5], i mnogi ovo smatraju njenom najvažnijom primenom. RCP aplikacija je skup priključaka i okvira (eng. Framework) na osnovu kojih se ta aplikacija izvršava.

Priključak predstavlja najmanju funkcionalnu celinu Eclipse platforme. Celokupna platforma je sastavljena od priključaka. Razvijajući RCP aplikaciju programer koristi priključke koji se nalaze u Eclipse bazi priključaka (njihova funkcionalnost je već definisana), kao i priključke koje je sam razvio. Priključci se najčešće sastoje od Java koda arhiviranog u Java arhiviranoj biblioteci, kao i dodatnih resursa u vidu slika, biblioteka sa funkcijama izvornog programskog jezika, internet stranica, dokumenata i sličnog. Svaki priključak ima svoj manifest koji definiše veze priključka sa ostalim priključcima u sistemu. Pri pokretanju platforme jezgro platforme pretražuje dostupne priključke, čita njihove manifest datoteke i popunjava registar priključaka (*eng. plugin register*) ovim informacijama. Pri tome se vrši provera kompatabilnosti priključaka i tačaka na koje se povezuju. Manifest priključka se sastoji od dve datoteke, MANIFEST.MF i plugin.xml. U MANIFEST.MF datoteci definisani su opšti podaci vezani za priključak koji podrazumevaju unikatnu identifikaciju priključka, njegovu verziju, ime i slično. Datoteka plugin.xml opisuje kako priključak proširuje platformu, koja proširenja objavljuje drugim priključcima, i kako sprovodi svoju funkcionalnost. Datoteka je napisana u proširivom jeziku za označavanje (eng. Extensible Markup Language – XML). Njen sadržaj se učitava kada platforma aktivira priključak. Priključak se aktivira tek kad to bude potrebno. Priključak će ostati

aktiviran sve dok ne bude eksplicitno deaktiviran ili dok platforma ne bude ugašena Ovakav koncept aktivacije priključka se naziva “lenjo učitavanje” (eng. Lazy loading). Proširivanje funkcionalnosti priključka omogućavaju tačke proširenja (eng. Extension points).

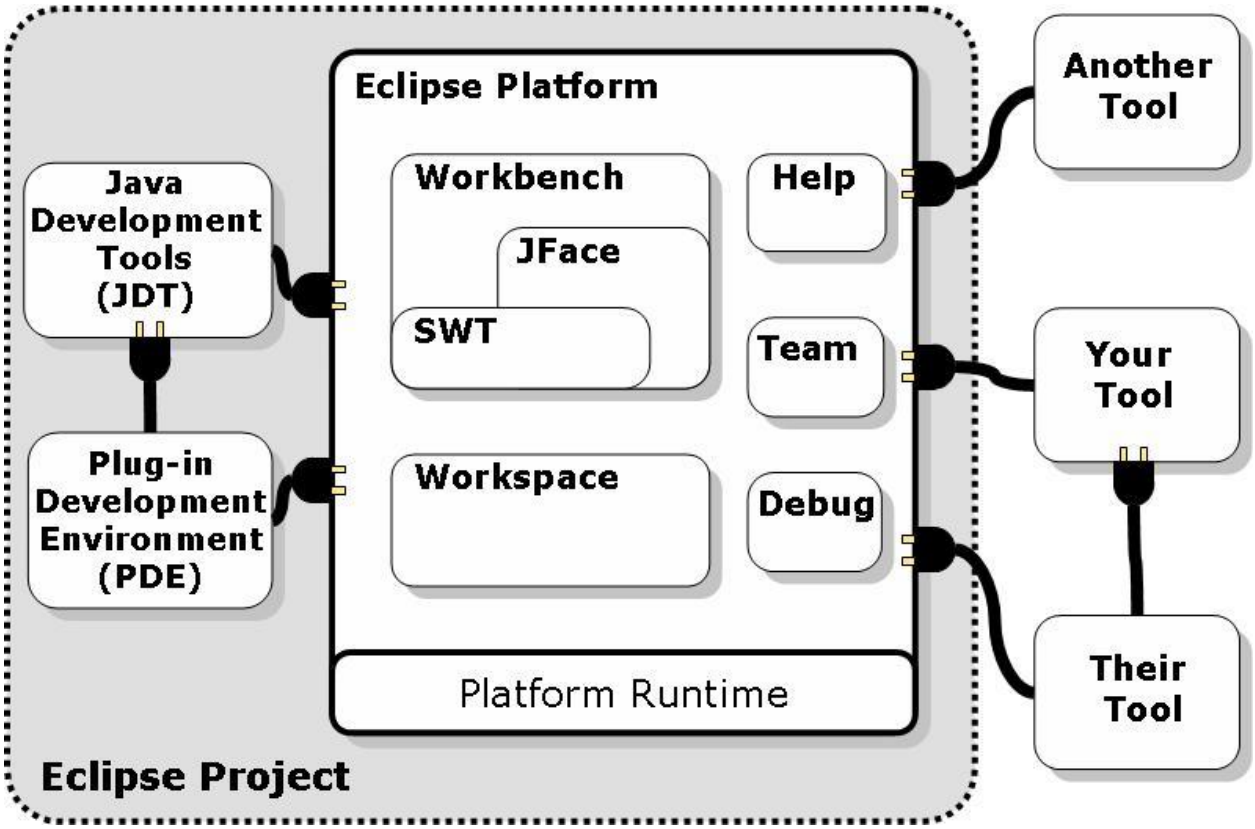
Tačke proširenja predstavljaju mehanizam komunikacije između priključaka Eclipse-a koji se definiše u XML datotekama i ove veze nije moguće menjati u toku izvršenja programa. Dva priključka se povezuju uparivanjem tačke proširenja jednog priključka sa tačkom proširenja drugog priključka ukoliko realizuju isto proširenje.

2.1.2 Osnovni podsistemi Eclipse platforme

Jezgro Eclipse platforme se sastoji iz nekoliko osnovnih podsistema:

- Radno okruženje - (*eng. Workspace*)
- Radni sto (*eng. Workbench*) – podsistem zadužen za spregu sa korisnikom
- Podsistem za pomoć korisniku (*eng. Help*)
- Podsistem za timski rad

Struktura Eclipse alata za razvoj programske podrške se može videti na Slici 1. Osnovni podsistemi [6] će u daljem tekstu biti detaljnije opisani.



Slika 1. Struktura Eclipse razvojnih alata

2.1.3 Radno okruženje

Za mnoge aplikacije je neophodno da omoguće korisniku da otvara, menja i rukuje datotekama u radnom okruženju. Eclipse platforma podršku ovakvim aplikacijama nudi u vidu priključaka radnog okruženja (*eng. Workspace*). Radno okruženje se sastoji od jednog ili više projekata koji zapravo predstavljaju odgovarajući direktorijum u direktorijumskoj strukturi. Projekti se mogu nalaziti u bilo kom direktorijumu mada se po pravilu nalaze u odgovarajućim poddirektorijumima direktorijuma radnog okruženja. Za opis tipa projekta koristi se mehanizam svojstva (eng. nature) projekta. Priključci mogu definisati nove tipove svojstava projekata i definisati način na koji se oni konfigurišu. Takođe jedan projekat može imati i više svojstava tako da može biti deljen između većeg broja priključaka.

Projekti sadrže datoteke koje stvara i kojima upravlja korisnik. Platforma definiše programsku spregu za rad sa resursima radnog okruženja preko koje im priključci mogu pristupiti. Sami resursi su predstavljeni prilagodljivim objektima preko kojih im se može definisati funkcionalnost. Kao zaštitu od slučajnog gubitka datoteka radno okruženje nudi i jednostavan mehanizam koji prati kratku istoriju izmena na datotekama tako da one mogu biti vraćene u neko od prethodnih stanja.

Radno okruženje nudi mehanizam markera za obeležavanje resursa. Oni služe za obeležavanje delova u datotekama kao što su: greške prevodioca, delovi koda koje treba doraditi (TODO lista), rezultati pretraga (*eng. Search hits*), obeleživači pozicije (*eng. Bookmarks*) i prekidne tačke (*eng. Breakpoints*). Ovaj mehanizam je takođe dostupan tako da priključci mogu definisati nove tipove markera i načine njihovog korišćenja.

Platforma poseduje i mehanizam koji alatima omogućava da reaguju na promene u resursima. Priključci se mogu registrovati kao slušaoci izmena (*eng. Resource change listener*) i na taj način biti informisani o događajima kao što su pravljenja i brisanja datoteka ili izmene u njihovom sadržaju. Ukoliko se radi o složenijoj izmeni, platforma odlaže prijavljivanje promene dok se ceo skup i operacija ne izvrši. Izveštaji o izmenama imaju oblik stabla izmena nad resursom, uključujući i promene markera. Princip stabla izmena je naročito koristan za alate kao što su prevodioci i povezivači, koji obavljaju analizu velikog broja datoteka. Platforma sadrži i osnovu za razvoj inkrementalnih prevodilaca, ulaz za ovakve prevodioce je stablo izmena u resursima koje opisuje razlike u datotekama u odnosu na poslednji poziv prevodioca. Postoji mogućnost automatskog pozivanja prevodioca nakon svake izmene resursa.

Proces snimanja sadržaja radnog okruženja je takođe otvoren za povezivanje priključaka koji žele da ostanu sinhronizovani sa okruženjem u različitim sesijama. Potrebni elementi priključaka koji se prijave biće sačuvani u direktorijumskoj strukturi tvrdog diska do sledeće sesije. Po sledećem pokretanju konkretnog priključka, on prima izveštaj o izmenama u mreži

resursa u odnosu na svoje poslednje snimljeno stanje. Na ovaj način priključak može preneti svoje stanje u sledeću sesiju, a ipak odreagovati na promene koje su se desile dok je bio neaktivan.

2.1.4 Alati za spregu sa korisnikom

Aplikativna programska sprega (eng. Application Programming Interface – API) radnog stola (eng. Workbench) i njegova implementacija zasnivaju se na dve grupe alata, SWT (eng. Standard widget toolkit) i JFace biblioteke alata. Ove dve biblioteke se koriste za razvijanje grafičke sprege sa korisnikom u Eclipse okruženju, kao i prilikom razvijanja samostalnih aplikacija sa grafičkom korisničkom spregom.

- SWT (Standard Widget Toolkit) - biblioteka dodataka i grafičkih elemenata integrisana sa grafičkim sistemom operativnog sistema i sa programskim spregama nezavisnim od sistema.
- JFace – biblioteka alata za pravljenje korisničke sprege implementiranih upotrebom SWT-a koji pojednostavljuje programiranje osnovnih ulazno izlaznih zadataka.

SWT definiše jedinstvenu (nezavisnu od operativnog sistema) programsku spregu za grafičke elemente i dodatke, implementiranu na način koji omogućava čvrstu integraciju sa grafičkim podsistemom operativnog sistema na kome se koristi. Celokupna Eclipse platforma i njeni alati koriste SWT za predstavljanje informacija korisniku.

Problemi Java AWT (*eng. Abstract Window Toolkit*) grafičke biblioteke su njena prenosivost i integracija sa operativnim sistemom. AWT biblioteka se direktno oslanjala na grafičke elemente operativnog sistema na kome se izvršava Java virtuelna mašina, zato programiranje korišćenjem AWT-a znači da se mogu koristiti samo elementi zajednički za sve operativne sisteme. Tako AWT podržava elemente nižeg nivoa (liste, polja sa tekstom, dugmad...), ali ne i višeg nivoa (stabla, formatirani tekst, ...). Osim toga veliki deo logičke sprege sa korisnikom se nalazi u kodu operativnog sistema što otežava otkrivanje i uklanjanje grešaka. SWING biblioteka ovo rešava tako što emulira sve grafičke elemente. Ponašanje i izgled ovakvih grafičkih elemenata se ponekad razlikuje od elemenata operativnog sistema. Starije verzije SWING-a su imale i problema sa performansama i prenosivošću na Java Micro Edition virtuelnu mašinu (J2ME) zbog velikih memorijskih zahteva. Iako su u novijim verzijama ovi problemi delom prevaziđeni, IBM je odlučio da za svoju Eclipse platformu izgradi novu biblioteku – SWT. SWT unapređuje AWT tako što koristi elemente grafičkog podsistema operativnog sistema kad god je to moguće, a za elemente koji nisu podržani od operativnog sistema vrši se njihova emulacija niskog nivoa. Takođe logika grafičkog sprežnog sistema se nalazi u Javi, što olakšava programiranje. Na ovaj način SWT održava ujednačen model programiranja u raznim okruženjima, a da pri tom u velikoj meri očuva funkcionalnost i izgled

operativnog sistema na kome se izvršava. Praktično, postoji posebna SWT implementacija za svaki podržani operativni sistem.

Za operativne sisteme koji pružaju veliku količinu novih elemenata koje ostali operativni sistemi nemaju, SWT pruža način pristupa programskoj sprezi tih elemenata operativnog sistema. Jedan primer za ovakav pristup su ActiveX objekti operativnog sistema Windows. Ovakva programska sprega je odvojena u posebne biblioteke da bi se naglasila njena neprenosivost. Prednost bliske veze sa operativnim sistemom ne omogućava samo verniji izgled i ponašanje, već omogućava i interakciju sa kontrolnim elementima operativnog sistema kao što su ActiveX kontrole i podrška za prevuci-i-pusti (*eng. drag and drop*) funkciju.

JFace je biblioteka sa klasama koje služe za programiranje ulazno-izlaznih zadataka korisničke sprege. Ona je potpuno nezavisna od platforme (i u programskoj sprezi koju nudi i u implementaciji) i projektovana je za rad sa SWT bibliotekom. Ona sadrži registre slika, fontova, dijaloge, prozore sa opcijama, podršku za izradu "čarobnjaka" (eng. Wizard) i praćenje napretka izvršenja dugotrajnih operacija. Najvažniji elementi ove biblioteke su akcije *(eng. Actions*) i pogledi (*eng. Viewers*).

Akcije omogućavaju da korisničke komande mogu biti definisane nezavisno od toga gde se nalaze u korisničkoj sprezi. Akcija je komanda koja može biti pokrenuta pritiskom na dugme, stavku u meniju ili polje u paleti sa alatkama. Svaka akcija definiše svoje osobine (natpis, ikonu, pomoćni tekst) koji se koriste za konstruisanje odgovarajućeg grafičkog elementa. Razdvajanjem funkcionalnosti akcije od konteksta u kome se prikazuje omogućena je laka izmena mesta prikaza u korisničkoj sprezi bez promene koda same akcije.

Pogledi su adaptacije grupe SWT dodataka. Predstavljaju prozore koji prikazuju standardne sadržaje kao što su liste, stabla i tabele, podržavaju upravljanje ovim sadržajima i održavanju sinhronizacije između njih i sistema. Ovi prozori se konfigurišu rukovaocem sadržaja i rukovaocem natpisa. Rukovalac sadržaja zna kako da prevede dostavljeni sadržaj u očekivani sadržaj pogleda (na primer: iz liste tačaka prekida, izvući će samo nazive datoteka u kojima se nalaze i brojeve linija u kojima se nalaze). Rukovalac natpisima zna koji natpis i ikona su potrebni za prikaz elemenata u pogledu. Pogledima se mogu dodeliti i klase za filtriranje i slaganje sadržaja koji prikazuju. Standardni pogled podržava uobičajene operacije: reakciju na dvostruki pritisak tastera na mišu, poništavanje efekata poslednje izmene (*eng. Undo*), obeležavanje bojom, navigaciju po broju linije i slično. Pogledi za prikaz teksta su zaduženi i za definisanje odgovarajućeg rukovaoca dokumenata koji vrši pretvaranje sadržaja dokumenta u informacije neophodne za ispravan prikaz u SWT tekst prozoru.

2.1.5 Radni sto

Dok su SWT i Jface biblioteke opšte namene, karakter korisničke sprege Eclipse platforme definišu priključenja radnog stola koja sadrže strukture preko kojih alati komuniciraju sa korisnikom. Zbog ove svoje centralne uloge radni sto je ujedno i sinonim za korisničku spregu Eclipse platforme. Programska sprega radnog stola se oslanja na programsku spregu SWT-a i u manjoj meri JFace-a. Osnovni elementi korisničke sprege Eclipse platforme su uređivači teksta (*eng. Editor*), pogledi i perspektive. Sa stanovišta korisnika perspektive se ne vide već se manifestuju izborom i rasporedom pojedinačnih elemenata korisničke sprege na ekranu.

Uređivači teksta omogućavaju korisniku da otvori, izmeni i sačuva datoteke. U tom smislu su slični priključcima koji rade sa resursima u direktorijumskoj strukturi, ali su mnogo bliže povezani sa radnim stolom. Kad su aktivni, uređivači teksta mogu dodati svoje akcije u menije ili paletu sa alatkama. Platforma definiše i standardni uređivač teksta za tekstualne datoteke. Uređivači za posebne tipove datoteka moraju biti prilagođeni od strane priključaka.

Pogledi prikazuju podatke o nekom objektu sa kojim korisnik radi. Na primer, dok korisnik unosi programski kod u nekom uređivaču teksta, pogled može biti upotrebljen da prikaže informacije o tom delu koda (funkcijama i promenljivama) definisanim u tekućem programskom bloku i slično). Pogled može biti izvor sadržaja koji će biti prikazan u drugom pogledu (na primer: jedan pogled može prikazivati detalje objekta izabranog u drugom pogledu). Pogledi imaju jednostavniji životni ciklus od uređivača teksta – najčešće su izmene u njima momentalno sačuvane i njihove posledice odmah prikazane u ostalim delovima UI platforme.

UI platforma može imati veći broj perspektiva, ali u datom momentu samo jedna može biti prikazana. Perspektiva može sadržati više različitih uređivača teksta i pogleda. Njihov izbor i raspored zavise od svrhe konkretne perspektive. Platforma ima unapred definisane perspektive za navigaciju kroz resurse, otklanjanje grešaka, podršku za rad u timu i sl.

Priključci se integrišu sa platformom korisničke sprege na jasno definisane načine. Glavne tačke proširenja omogućavaju proširenje radnog stola na sledeće načine:

- dodavanjem novih uređivača teksta
- dodavanjem novih tipova pogleda
- dodavanjem novih perspektiva (koje raspoređuju stare i nove uređivače i poglede)

Platforma vodi računa o elementima perspektive, učitava ih po potrebi i zatvara kad više ne budu neophodni. Ikone i nazivi njihovih akcija se nalaze u manifestu priključaka tako da stavke u meniju mogu biti iscrtane bez učitavanja samih elemenata.

Alati napisani u Javi korišćenjem programske sprege platforme imaju najveći stepen integracije sa platformom. Spoljašnji alati pokrenuti iz platforme moraju se izvršavati u

posebnim prozorima da bi komunicirali sa korisnikom i koristiti direktorijumsku strukturu da bi pristupili korisničkim podacima.

2.2 Namenski sistemi

Namenski sistemi (eng. Embedded systems) su računarski sistemi sa jako izraženom integracijom fizičke arhitekture i programske podrške, pre svega namenjeni da obavljaju specifične funkcije [7]. Reč *embedded* ukazuje na činjenicu da su ovi sistemi obično integralni deo nekog većeg sistema, poznat kao *embedding-namenski* sistem. To su sistemi specijalne namene kod kojih je računar u potpunosti enkapsuliran od strane uređaja koga on kontroliše. Nasuprot računaru opšte namene, kakav je personalni računar (eng. Personal Computer) koji treba da zadovolji širok spektar funkcionalnosti, namenski sistem obavlja jedan ili veći broj unapred definisanih zadataka, obično sa veoma specifičnim zahtevima. Široko su rasprostranjeni u potrošačkoj, komercijalnoj i vojnoj industriji. Namenski sistemi namenjeni digitalnoj obradi signala zasnivaju se na digitalnim signal procesorima.

2.3 Digitalni signal procesori

U smislu radne definicije, sistem za digitalnu obradu signala je elektronski sistem koji obavlja digitalnu obradu signala. U neformalnom smisli radi se o primeni matematičkih radnji nad signalima predstavljenih u obliku cifara (digitalno).

Digitalno se signali predstavljaju kao odmerci. Oni se dobijaju iz fizičkog signala (npr. Audio signala) preko pretvarača fizičkog signala u električni oblik (mikrofon) i pretvaranjem analognog u digitalni signal, u analogno-digitalnom pretvaraču (konvertoru). Po matematičkoj obradi digitalni signal se može vratiti u polazni fizički oblik, postupkom digitalno analognog pretvaranja u digitalno analognim pretvaračima. U postupcima digitalne obrade signala veoma važnu ulogu igraju namenski procesori, Digitalni Signal Procesori.

Digitalna obrada signala poseduje znatne prednosti nad analognom obradom. Najvažnija od prednosti odnosi se na mogućnost da DSP-i mogu da obave posao obrade za mnogo manju cenu, što je veoma teško ili skoro nemoguće za analognu elektroniku. Primeri za to su sinteza govora, prepoznavanje govora, modemi velike brzine sa kodiranjem i sa korekcijom i detekcijom greške.

Ove obrade se baziraju na kombinaciji obrade signala i donošenja odluke na bazi primljenih bita ili govora, što je izuzetno teško realizovati komponentama analogne elektronike. Značajne prednosti digitalne obrade signala su i neosetljivost na okruženje (npr. Temperatura), neosetljivost na tolerancije komponenti, što je slučaj kod analognih komponenti koje se proizvode sa određenim tolerancijama.

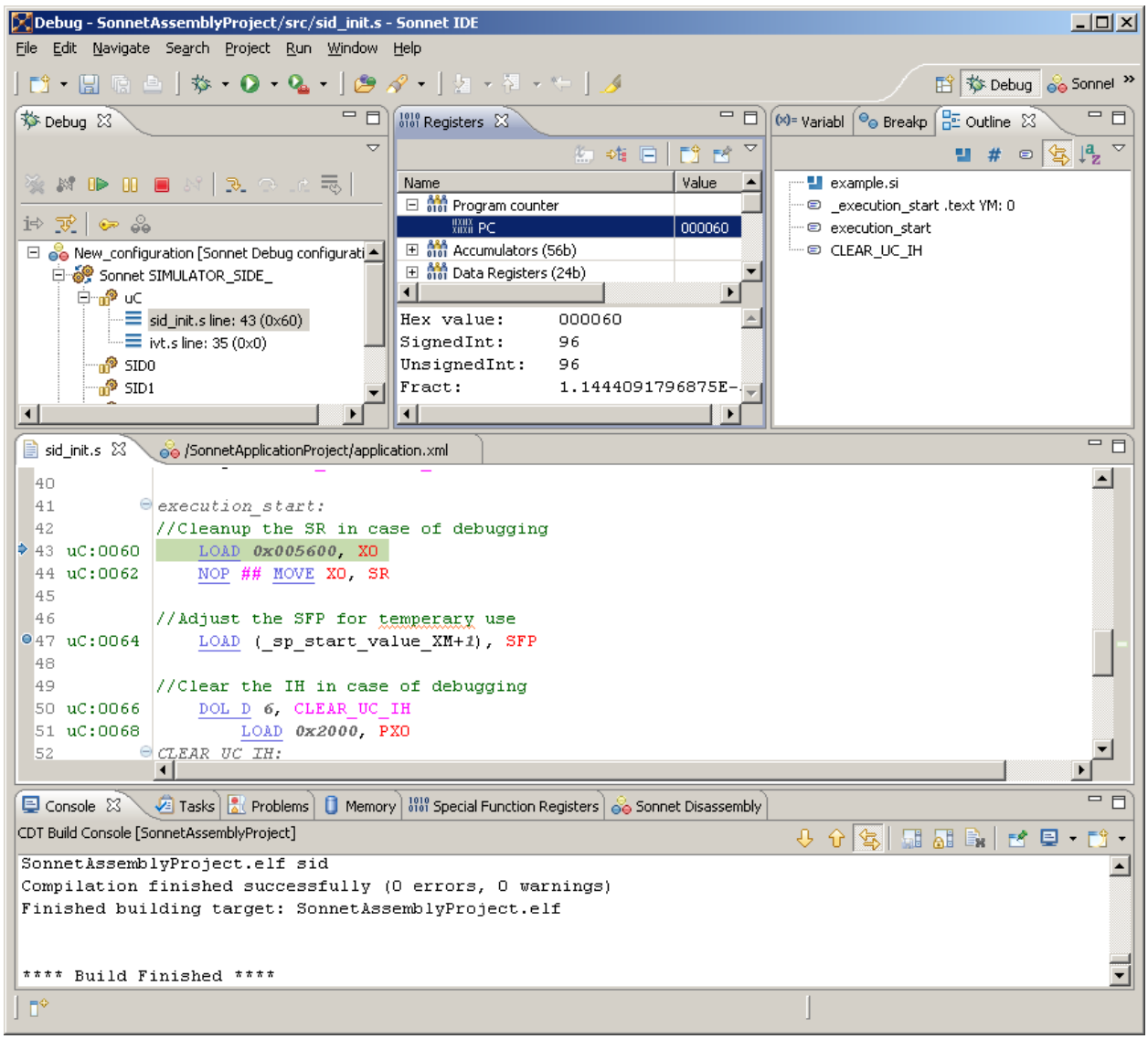
Digitalni sistemi su i reprogramabilni, jer najčešće koriste programabilan DSP. On se može programirati da obavi neku drugu funkciju bez promene strukture, što je za analogni sistem neizvodljivo.

Broj i raznovrsnost proizvoda koji obuhvataju neki oblik digitalne obrade signala raste iz godine u godinu [8]. DSP-i su tokom godina postali ključna komponenta u mnogim proizvodima iz oblasti komunikacija, medicine, potrošačke, vojne industrije itd. Ovi proizvodi koriste različite pristupe fizičke arhitekture za implementaciju DSP-a, od upotrebe mikroprocesora do korišćenja integrisanih kola (FPGA - *Field-Programmable Gate Array* , IC – *Integrated Circuit*). Zbog mogućnosti programabilnosti, klasa mikroprocesora optimizovanih za DSP je postala optimalno rešenje iz nekoliko razloga. U poređenju sa procesorima sa fiksnim funkcionalnostima (eng. Fixed-function solutions), programabilni procesori mogu nadograditi postojeću funkcionalnost, ili je promeniti ili popraviti. DSP-i su često isplativija i manje rizična rešenja od specijalizovane fizičke arhitekture, posebno za aplikacije koje zahtevaju ograničenu količinu resursa.

Sa stanovišta masovnosti, najrasprostranjenije primene za digitalne signal procesore su jeftini namenski sistemi (celularni telefoni, modemi, disk kontroleri) kod kojih je dominantna cena i integracija.

2.4 SIDE razvojno okruženje

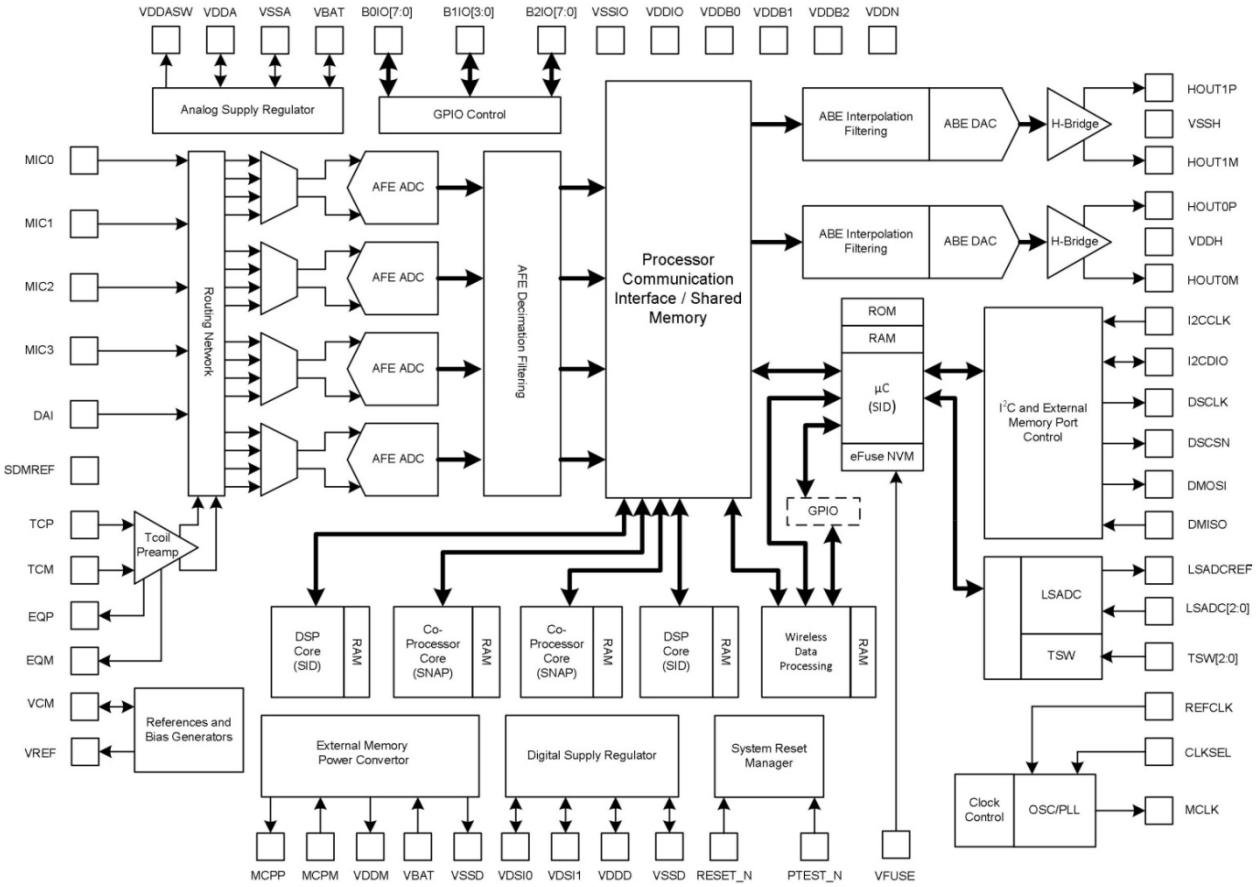
Na osnovu Eclipse platforme razvijeno je razvojno okruženje za programsku podršku Sonnet familije DSP procesora. Ovo razvojno okruženje se naziva SIDE (slika 2). U razvoju SIDE-a korišćena je verzija Eclipse okruženja za razvoj samostalnih RCP aplikacija i priključaka Eclipse okruženja (Eclipse for RCP - Plugin Developers). SIDE je realizovan kao RCP aplikacija koja pomoću gotovih priključaka Eclipse razvojnog okruženja gradi novo razvojno okruženje i dodavanjem novih priključaka ga prilagođava novoj svrsi. SIDE je razvojno okruženje namenjeno za razvijanje namenske programske podrške (eng. Firmware) i obezbeđuje alate potrebne za pisanje, ispitivanje, prevođenje, otklanjanje grešaka namenske programske podrške koja se izvršava na ciljnoj platformi. Pored toga, obezbeđuje alate za optimizaciju, sinhronizaciju i verifikaciju programske podrške za Sonnet sistem. Praćenje istorije izvornog koda, podrška za timski rad, pomoć u radu, su samo neke od podržanih mogućnosti za integraciju u SIDE razvojno okruženje.



Slika 2. SIDE razvojno okruženje

2.5 Sonnet platforma

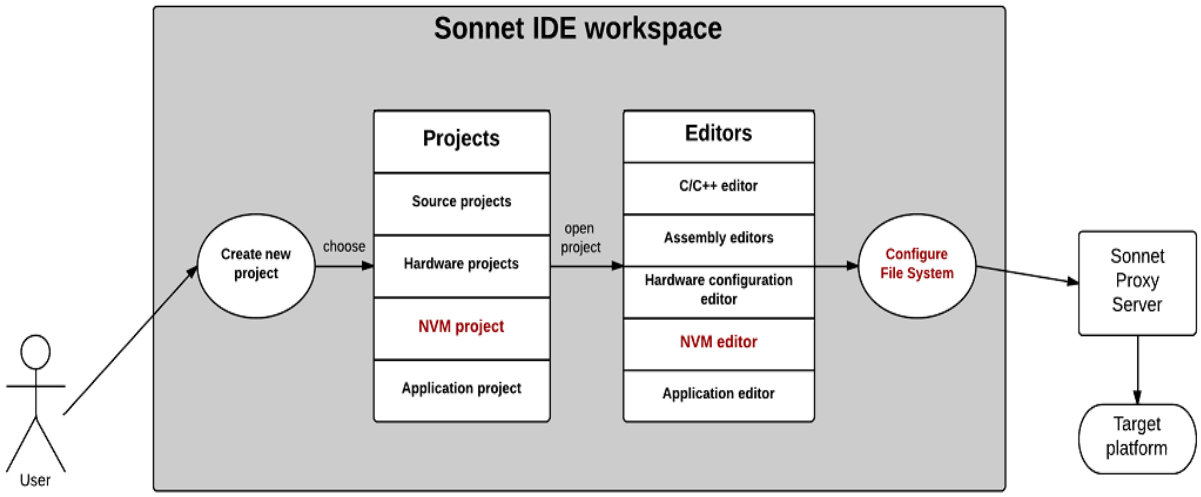
Sonnet familija procesora (slika 3) sadrži pet procesora koji se razlikuju u namenskoj programskoj podršci kao i količini raspoložive memorije. Ovi DSP procesori poseduju odvojene memorijske zone za podatke (X i Y), kao i posebnu programsku memorijsku zonu P. Tri procesorska jezgra, od kojih jedno ima ulogu mikrokontrolera, učestvuju u obradi i kontroli audio signala unutar slušnog aparata. Preostala dva procesora su numerički procesori posebno usmereni na obradu numeričkih algoritama. Podržavaju širok spektar algoritama, uključujući Furijeove transformacije, vektorske operacije, logičke operacije, algoritme filtriranja itd.



Slika 3. Sonnet platforma

3. Koncept rešenja

Upravljanje sistemom datoteka na ciljnoj platformi podrazumeva spregu SIDE razvojnog okruženja i Sonnet platforme. Realizovana podrška za upravljanje sistemom datoteka unutar razvojnog okruženja obuhvata odgovarajuću grafičku spregu (pravljenje projekata, čarobnjaka, uređivača, pogleda) unutar Sonnet radnog prostora (Slika 4), dok modul za komunikaciju sa platformom podrazumeva podršku za upis i čitanje sistema datoteka. Ovo poglavlje sadrži opis datoteka sistema, način njihovog raspoređivanja u adresni prostor, upis i čitanje sa Sonnet platforme.



Slika 4. Dijagram upravljanja sistemom datoteka iz radnog prostora Sonnet razvojnog okruženja

3.1 Podešavanje datoteka u okviru razvojnog okruženja

Izvorni programski kod napisan za familiju Sonnet procesora se nakon asembliranja i povezivanja dobija u vidu izršnih datoteka ELF (eng. Executable Linking Format) formata. Skup takvih datoteka čine sistem datoteka koji se upisuje u memoriju fizičke arhitekture. Proces podešavanja sistema datoteka počinje iz SIDE razvojnog okruženja. Takav proces zahteva realizaciju programske sprege koja bi korisniku ponudila jednostavan način upravljanja memorijom u koju se datoteke smeštaju. Ta memorija je „postojana“ memorija (eng. Non-volatile memory - NVM) i zauzima vrlo mali deo na platformi. NVM je trajna memorija koja čuva uskladištene informacije i kada nije pod napajanjem. Primeri takve memorije su *RAM*, *FRAM*, *flash*, tvrdi disk, diskete... Konkretno, na Sonnet platformi se koristi *EEPROM* ili FLASH tip memorije za potrebe skladištenja sistema datoteka. U SIDE razvojnom okruženju razvijen je poseban uređivač za podešavanje memorije tog tipa, pod nazivom NVM Editor.

3.2 NVM uređivač

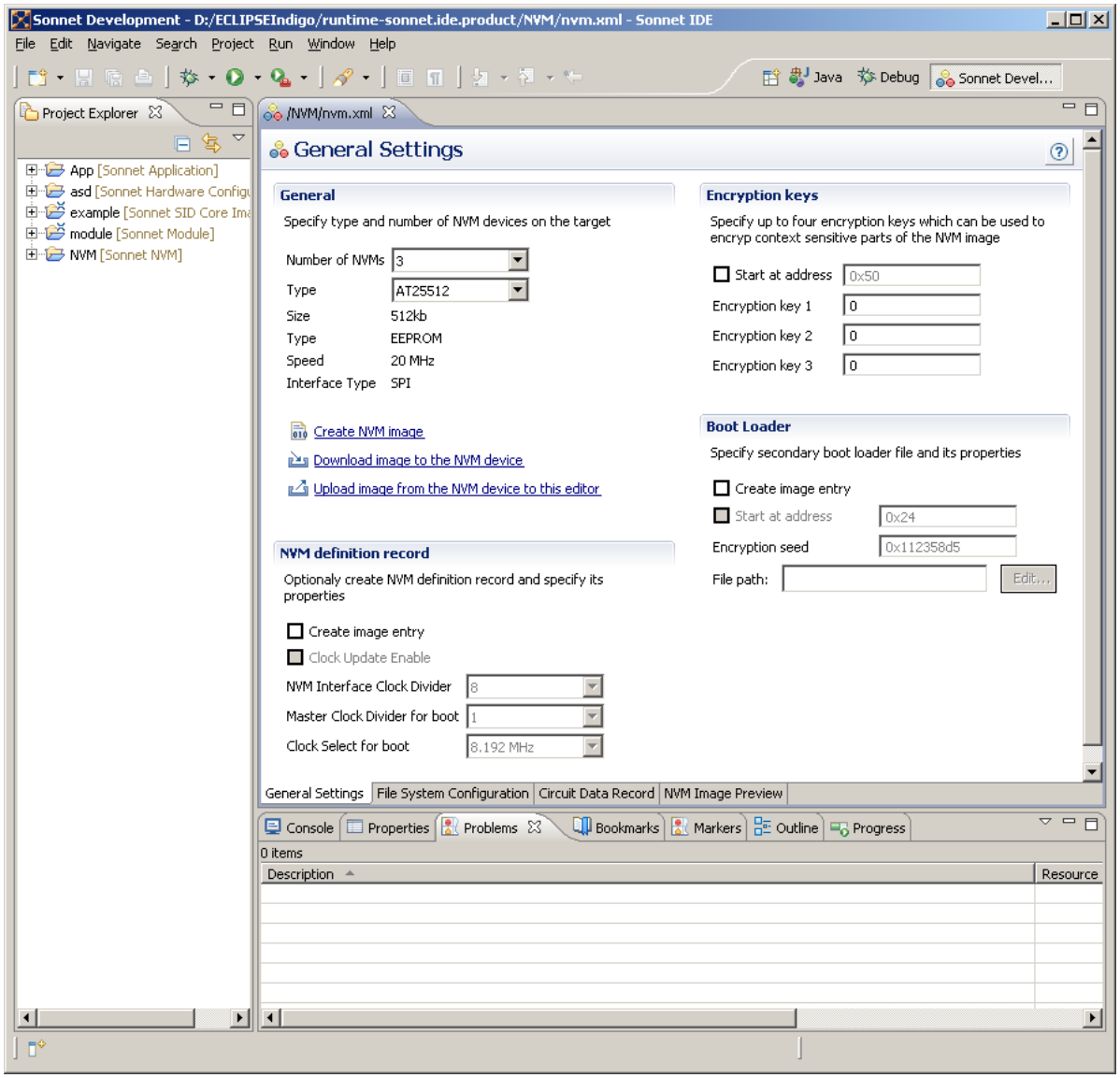
Da bi bio omogućen rad u uređivaču, potrebno napraviti NVM tip projekta koji se nalazi u Sonnet kategoriji projekata. Pravljenje projekta podrazumeva podešavanje i unos informacija o veličini i tipu NVM memorije, tipu magistrale, brzini takta itd. U plugin.xml datoteci je registrovana nova priroda projekta, *Eeprom Configuration* projekat. Za pravljenje projekta napravljen je čarobnjak za koji je korišćena podrška platforme, i on je povezan na `org.eclipse.ui.newWizards` tačku proširenja čarobnjaka za nove objekte u radnom okruženju. Čarobnjak stvara novi projekat koji se smešta u radno okruženje i sadrži datoteku `nvm.xml` koja se otvara u uređivaču.

NVM uređivač (slika 5) je realizovan koristeći `org.eclipse.ui.editors` proširenje. Uređivač je u osnovi *FormEditor*. Ovakav tip uređivača se najčešće koristi kada je potrebno da uređivač sadrži više stranica. *FormEditor* je povezan na `org.eclipse.ui.forms.editor` proširenje koje je projektovano sa ciljem da unese određene novine prilikom interakcije sa korisnikom, kakve se mogu videti u HTML alatima, kao i da pruži podršku u stvaranju korisničke sprege za sve Eclipse tipove projekata.

NVM uređivač se sastoji iz sledećih stranica:

- General Settings – stranica za prikaz opštih informacija o platformi. U okviru nje se vrši podešavanje rezervisanih datoteka.

- File System Configuration – stranica u kojoj se vrši odabir i podešavanje datoteka programske podrške.
- Circuit Data Record – stranica koja prikazuje rezervisanu datoteku, i neće biti detaljnije razmatrana.
- NVM Image Preview – stranica u kojoj se u svakom trenutku (ukoliko ne postoji greška u podešavanju datoteka) može videti raspored sistema datoteka po početnim adresama i veličini svake datoteke. Predstavlja opšti prikaz sistema datoteka.



Slika 5. Višestranični NVM uređivač

3.3 Sistem datoteka

Datoteka se definiše kao skup slogova podataka, grupisanih tako da se omogući rukovanje pristupom, očitavanje i modifikacija individualnog sloga. U tom smislu slog je osnovna jedinica obrađivane informacije i predstavlja listu objekata informacionog karaktera. Skup slogova u memoriji definiše blok, a broj slogova u bloku definiše koeficijent bloka. Datoteka se može definisati i skupom blokova, interpretirajući blok kao jedinicu razmene informacija između između procesa i datoteke, koja je smeštena na ulazno-izlaznom podsistemu.

U opštem slučaju nad datotekama je moguće vršiti sledeće operacije:

- otvaranje datoteke predstavlja pripremu datoteke da bi joj se moglo pristupiti,
- zatvaranje datoteke je operacija sprečavanja daljeg pozivanja datoteke,
- formiranje nove datoteke,
- odstranjivanje datoteke,
- promena naziva datoteke, štampanje ili prikazivanje sadržaja datoteke,

Nad slogom, kao osnovnom jedinicom obrađivane informacije, moguće su operacije:

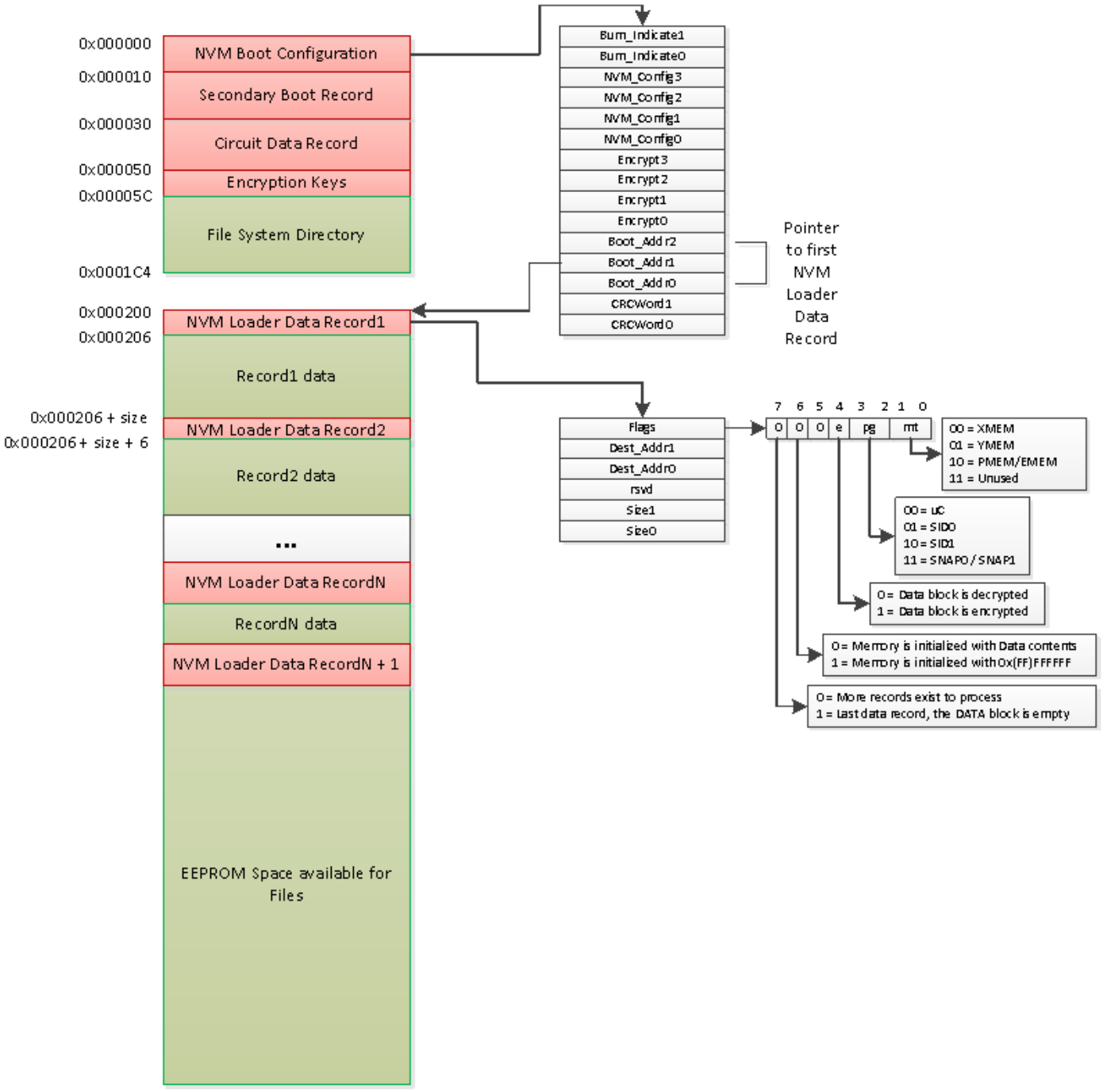
- čitanja, odnosno, operacija prenosa sloga od datoteke do procesa,
- upis ili prenos sloga od procesa u datoteku,
- obnavljanje ili ažuriranje je operacija promene sadržaja postojećeg sloga,
- umetanje je operacija dodavanja novog sloga,
- brisanje je operacija odstranjivanja sloga iz datoteke.

Skup programskih komponenti koje realizuju gore pobrojane operacije nad datotekama, naziva se sistemom datoteka (eng File System). Bez sistema datoteka paralelno programiranje bi bilo neefikasno jer sistem datoteka podržava paralelan rad ulazno-izlaznog podsistema i procesora.

NVM sistem datoteka sastoji se od nekoliko različitih tipova datoteka. Kao što već napomenuto, datoteke se dodaju u okviru *General Settings* i *File System Configuration* stranica NVM uređivača. Sistem čine sledeće datoteke:

- Datoteka za podešavanje parametara platforme
- Datoteka za učitavanje adresnog prostora
- Datoteka pokretačkog programa
- Datoteka ključeva za šifrovanje
- Datoteke programske podrške

Struktura ovih datoteka pojašnjena je u nastavku, dok je na slici 6 prikazan primer raspoređenog sistema datoteka u NVM memorijskom prostoru.



Slika 6. Struktura sistema datoteka

3.3.1 Datoteka za podešavanje parametara platforme

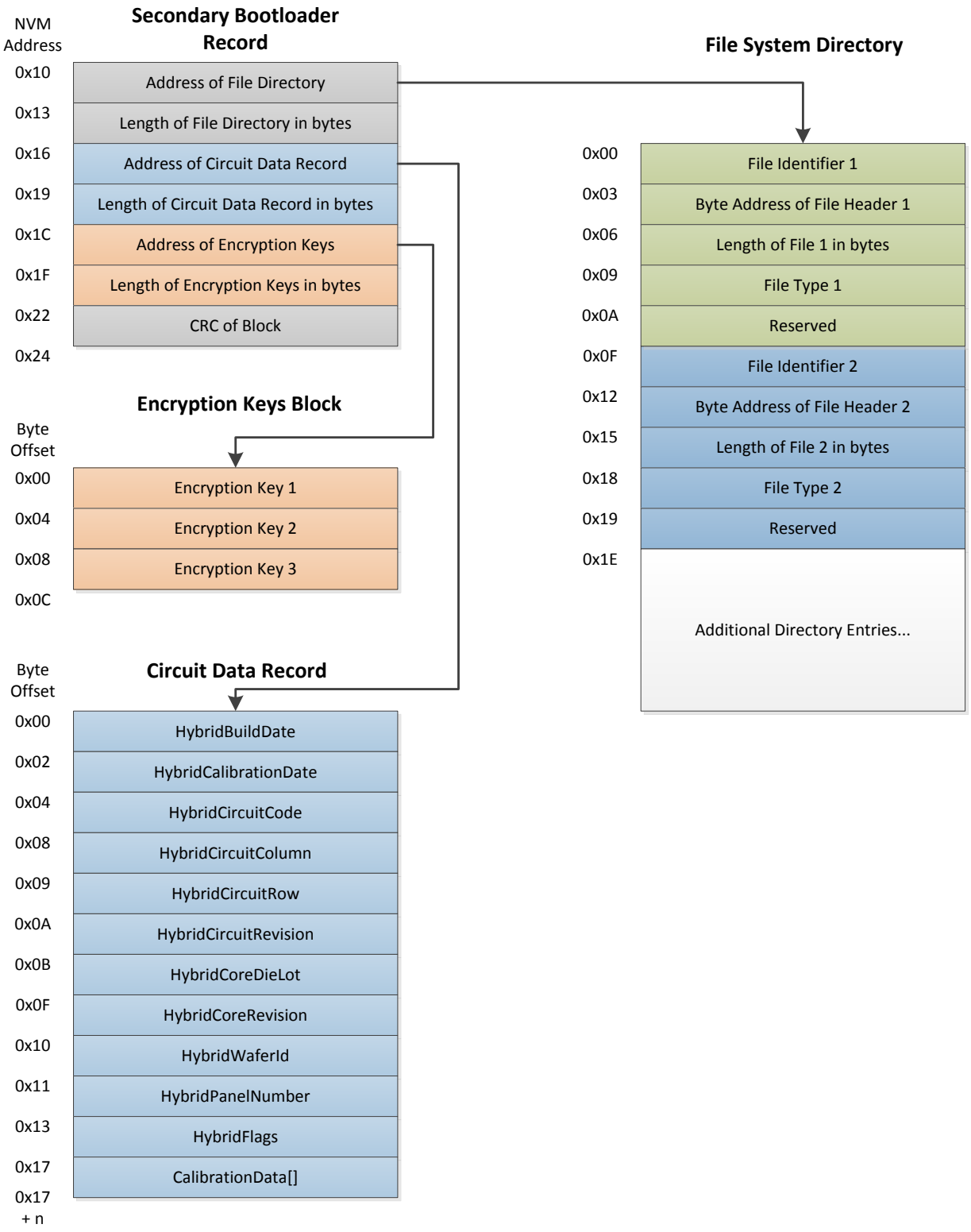
Nakon što korisnik uspešno napravi NVM projekat, parametri ciljne platforme uneti u čarobnjaku su prikazani na General Settings stranici. Parametri memorije mogu da se razlikuju, počevši od veličine, brzine takta do tipa magistrale (SPI ili I2C). Ukoliko korisnik

iz nekog razloga želi da promeni parametre, on ne mora da pravi novi projekat sa novim podešavanjima, već je dovoljno da izmeni postojeća podešavanja na General Settings stranici.

Datoteka za podešavanje parametara platforme je jedna od dve datoteke u sistemu datoteka koja ima rezervisanu početnu adresu u memoriji i ona je 0x0. Veličina ove datoteke iznosi 16 bajta. Pored već navedenih informacija o tipu memorije, brzini, tipu magistrale, u ovu datoteku se još upisuju i informacije o zaštitnom kodu (eng. Cyclic Redundancy Check kod) i identifikatorskom kodu koji ima jedinstvenu vrednost (0xCA53) i označava da li je sistem datoteka inicijalizovan, odnosno spreman za čitanje i upis.

3.3.2 Datoteka za učitavanje adresnog prostora sistema

Pored datoteke za podešavanje ciljne platforme, druga datoteka sistema sa rezervisanom adresom je datoteka za učitavanje adresnog prostora sistema. Ova datoteka sadrži informacije o početnim adresama svih datoteka koje su dodate u sistem i njihovim veličinama. Pomoću ove datoteke moguće je izvući podatke o svakoj datoteci sistema i zbog toga predstavlja važan deo sistema. Rezervisana adresa na kojoj se nalazi je 0x10 i njena veličina iznosi 20 bajta. Ova datoteka se automatski generiše, što znači da je korisnik u okviru razvojnog okruženja ne može podesiti, izmeniti, obrisati, već se ona prilikom svakog upisa sistema na platformu formira na osnovu podataka drugih datoteka. Na slici 7 prikazan je sadržaj ove datoteke (Secondary Bootloader Record).



Slika 7. Datoteka za učitavanje adresnog prostora sistema

3.3.3 Datoteka pokretačkog programa

Još jedna datoteka koja se podešava u okviru stranice opštih informacija NVM uređivača je datoteka pokretačkog programa (eng. Boot loader). Uloga pokretačkog programa je da obezbedi inicijalni set operacija koje će sistem izvršiti. Ovaj proces se izvršava svaki put kada je sistem ponovo pokrenut nakon resetovanja ili nakon uključivanja. Na Sonnet platformi, iz datoteke pokretačkog programa procesor sa ulogom mikrokontrolera će učitati programsku podršku namenjenu za svaki od procesora i njom inicijalizovati RAM memoriju svakog procesora. Time će svakom od procesora namenjenih za obradu signala pružiti potrebne informacije za dalji rad. Ova datoteka nema rezervisanu adresu u NVM memoriji, ali se njena adresa upisuje na određeno polje u datoteci za podešavanje parametara platforme. Na taj način mikrokontroler prilikom čitanja konfiguracione datoteke uvek zna na koju adresu da skoči da bi izvukao informacije pokretačkog programa.

Podešavanje datoteke vrši se unutar stranice za opšte informacije. U okviru posebne sekcije ove stranice, korisnik je u mogućnosti da navede putanju datoteke programske podrške, sa sistema datoteka računara na kom radi, sa koje će se učitati pokretački program. U toj datoteci mora biti napisan izvorni kod pokretačkog programa ili će uređivač obavestiti korisnika o nastaloj grešci. Ukoliko je korisnik odabrao ispravnu datoteku, u mogućnosti je da definiše početnu adresu na kojoj će biti smeštena ova datoteka u okviru NVM adresnog prostora.

3.3.4 Datoteka ključeva za šifrovanje podataka

Za potrebe datoteka koje u sebi sadrže izvorni kod namenske programske podrške, ili bilo koje podatke od značaja, razvijena je podrška za šifrovanje podataka. Podaci se šifruju koristeći algoritam pomeračkog registra sa povratnom spregom (eng. *Linear Feedback Shift Register* – LFSR [9]).

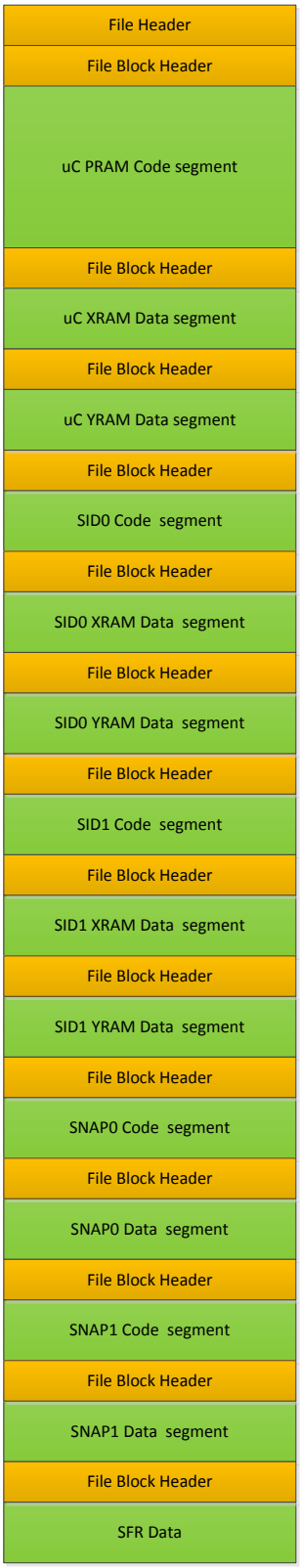
Stranica za opšte informacije NVM uređivača sadrži i sekciju ključeva za šifrovanje podataka. U ovoj sekciji se definišu vrednosti koje predstavljaju ključeve za šifrovanje na osnovu kojih funkcioniše LFSR algoritam. Moguće je definisati tri vrednosti, odnosno tri ključa, čija vrednost ne prelazi najveću vrednost četvorobajtnog broja (0xFFFFFFFF). Ove vrednosti čine datoteku ključeva za šifrovanje podataka. Kao i kod ostalih datoteka, moguće je definisati početnu adresu koju će ova datoteka zauzimati u NVM adresnom prostoru.

3.3.5 Datoteke programske podrške

Ovu grupu čine više različitih tipova datoteka. Ove datoteke se dodaju u stranici za podešavanje datoteka programske podrške NVM uređivača (File System Configuration). Stranica je realizovana tako da se sastoji iz dva dela, glavnog sa leve i pomoćnog sa desne strane (eng. Master-Details page). Glavni deo se sastoji iz stabla u kom su prikazane podešene datoteke programske podrške koje je korisnik dodao. Odabirom neke datoteke iz stabla, na strani pomoćnog dela prikazani su parametri te datoteke. Datoteke se međusobno razlikuju po veličini, početnoj adresi, identifikatorskom broju itd.

Za svaki tip datoteke napravljen je poseban čarobnjak za njeno dodavanje unutar NVM sistema datoteka. Kako svaka datoteka ima različite atribute, proces podešavanja svake je različit. Ono što je zajedničko za svaku datoteku programske podrške je definisanje početne adrese, definisanje jedinstvenog identifikatorskog broja i imena datoteke koje će biti prikazano u stablu.

Jedan tip datoteka programske podrške je izvršna datoteka dobijena nakon asembliranja i povezivanja datoteke izvornog programskog koda. Takva datoteka sadrži instrukcije i podatke [10] koji su spremni za izvršavanje na jednom od procesora Sonnet platforme. Ti podaci se izvlače iz datoteke pomoću alata komandne linije, i upisuju u Java strukture podataka kako bi nakon što se podešavanje u čarobnjaku završi, u delu za detalje bili prikazani segmenti i zaglavlja korisniku na uvid (slika 8). To podrazumeva da su korisniku prikazani detalji o veličini svakog segmenta, informacija da li su podaci šifrovani ili ne i sa kojim ključem (od već pomenuta tri), na kom procesoru se podrška izvršava i sl. Pored datoteka programske podrške sa izvornim kodom, u ovu grupu spadaju još i datoteke za inicijalizaciju memorije određenom konstantom, datoteke za rezervisanje adresa, audio datoteke i datoteke za podešavanje komponenti fizičke arhitekture.



Slika 8. Izvršna ELF datoteka podeljena na zaglavlja i segmente

Korisnik u svakom trenutku u delu za detalje može da menja parametre već dodate datoteke programske podrške, bez da prolazi kroz proces podešavanja unutar čarobnjaka.

Takođe, brisanje jedne ili više datoteka iz stabla, odnosno sistema datoteka, je moguće u svakom trenutku birajući opciju *„Remove“*.

3.4 Validator datoteka

Za svaki tip datoteke postoji validator njenih parametara. Pomoću njega, okruženje obaveštava korisnika ukoliko došlo do greške prilikom podešavanja parametara datoteke. Greške mogu biti različite, od pogrešno navedene putanje datoteke programske podrške, pogrešno definisane početne adrese, pogrešne inicijalizacije, ukoliko postoje datoteke sa istim identifikatorskim brojem itd. Ukoliko postoji greška, na vrhu uređivača prikazana je poruka o grešci, odnosno koji je parameter u pitanju i na koji način se može ispraviti. Poruka o grešci je prikazana i u Eclipse-ovom *„Problems“* pogledu. Pogled sa greškama se odnosi na ceo radni sto, odnosno na sve projekte koji ga čine. To znači da iako je NVM projekat koji sadrži grešku zatvoren i korisnik ga ne koristi, u pogledu sa greškama će i dalje biti ispisana poruka koja se odnosi na NVM projekat. Validator datoteka obezbeđuje da sistem datoteka bude uspešno podešen i sprečava probleme koji bi mogli nastati prilikom upisivanja sistema datoteka sa greškom.

3.5 Raspoređivač sistema datoteka

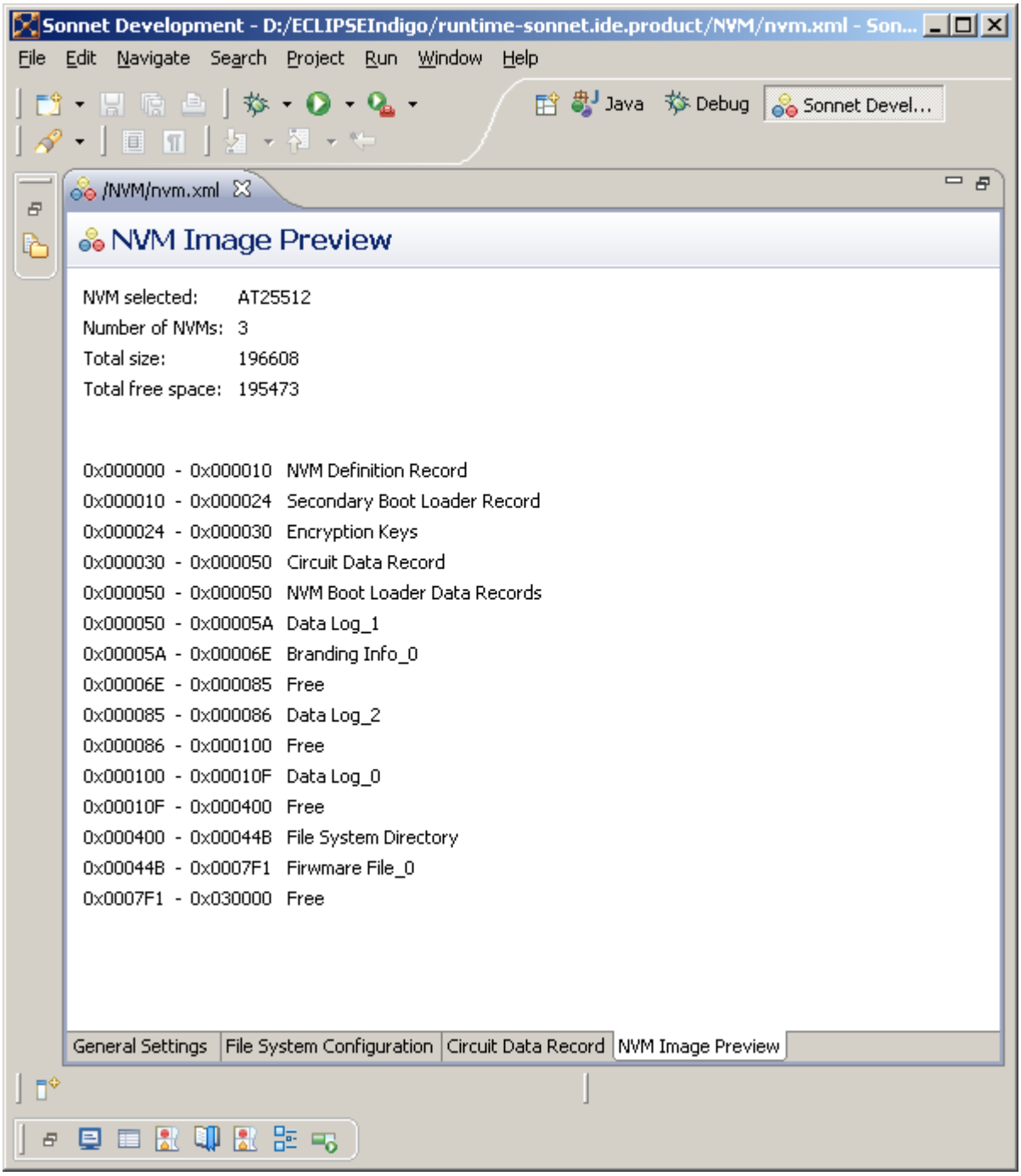
Kao što je već spomenuto, korisnik prilikom dodavanja datoteka može da definiše početnu adresu na kojoj će se ona naći u NVM sistemu datoteka. Ukoliko ne želi da joj definiše početnu adresu, algoritam raspoređivanja adresa će to učiniti za njega. Na poslednjoj stranici NVM uređivača korisnik je u mogućnosti da u svakom trenutku, ukoliko nema grešaka u podešavanju, pogleda predloženi raspored sistema datoteka. Kako Sonnet platforma podržava više tipova NVM memorija koje se razlikuju po veličini, na stranici se nalaze informacije sa kojim adresnim opsegom korisnik raspolaže.

Ukoliko u podešenom sistemu datoteka postoje datoteke kojima je korisnik zadao početnu adresu, a postoje i one kojima nije, algoritam raspoređivanja radi na sledeći način: Najveći prioritet prilikom raspoređivanja imaju datoteke sa rezervisanom početnom adresom, datoteka opisa za podešavanje parametara platforme i datoteka za učitavanje adresnog prostora. Te dve datoteke zauzimaju adresni prostor u opsegu od 0x0 do 0x24, odnosno 36 bajta. Nakon te dve datoteke, raspoređuju se datoteke kojima je korisnik zadao početnu adresu. Tek kada se upišu datoteke sa zadatom adresom, algoritam raspoređivanja dodeljuje adrese datotekama kojima početna adresa nije definisana. Kada prilikom pretraživanja adresnog prostora algoritam naiđe na slobodan prostor, procenjuje se koja od preostalih

datoteka je pogodna da se rasporedi na to mesto (slika 9). Na prvi slobodan prostor na koji datoteka može da se rasporedi (u zavisnosti od njene veličine), algoritam joj dodeljuje početnu adresu, preračunava preostali slobodan prostor ukoliko ga ima, i nastavlja sa raspoređivanjem ostalih datoteka sve dok se ne definišu početne adrese za sve datoteke sistema.

Greške prilikom raspoređivanja datoteka mogu da se dese ukoliko se datotekama preklapaju adrese. Takve greške se najčešće dešavaju kada korisnik definiše početnu adresu neke datoteke, a ona se već nalazi u opsegu druge (početna adresa plus veličina datoteke). Kako se algoritam raspoređivanja aktivira kada korisnik otvori stranicu za pregled NVM sistema u uređivaču, uređivač će izbaciti poruku u kom ga obaveštava da je nastala greška i da se adresni opsezi datoteka preklapaju i ne mogu rasporediti po korisnikovoj želji.

Algoritam za rapoređivanje nije moguće aktivirati samo u slučaju da validator datoteka prijavljuje određenu grešku. Tada je korisnik dužan da prvo pravilno unese parametre svih datoteka, a tek nakon toga će biti omogućen ovaj algoritam.



Slika 9. Raspoređen sistem datoteka

3.6 Stvaranje binarnih datoteka

U svakom trenutku moguće je napraviti datoteke koje sadrže binarnu predstavu trenutno dodatih datoteka u sistemu. Dugme na koje se stvaraju ove datoteke nalazi se na stranici sa opštim informacijama. Ukoliko validator datoteka nije prijavio greške, pritiskom na dugme *"Create NVM image"* stvara se direktorijum u okviru NVM projekta koji sadrži stvorene datoteke. Za svaku podešenu datoteku u uređivaču stvara se binarna datoteka u .xml formatu.

Datoteka nvmXML.xml sadrži listu svih novonastalih datoteka sa binarnom predstavom podataka. Uz pomoć nvmXML.xml datoteke korisnici bez korišćenja razvojnog okruženja mogu da imaju uvid u sistem datoteka i koje datoteke ga čine. Korisnici mogu da vrše razmenu ovih datoteka u cilju brzog podešavanja sistema, pronalaska grešaka, ispitivanja itd.

3.7 Sprega okruženja i platforme

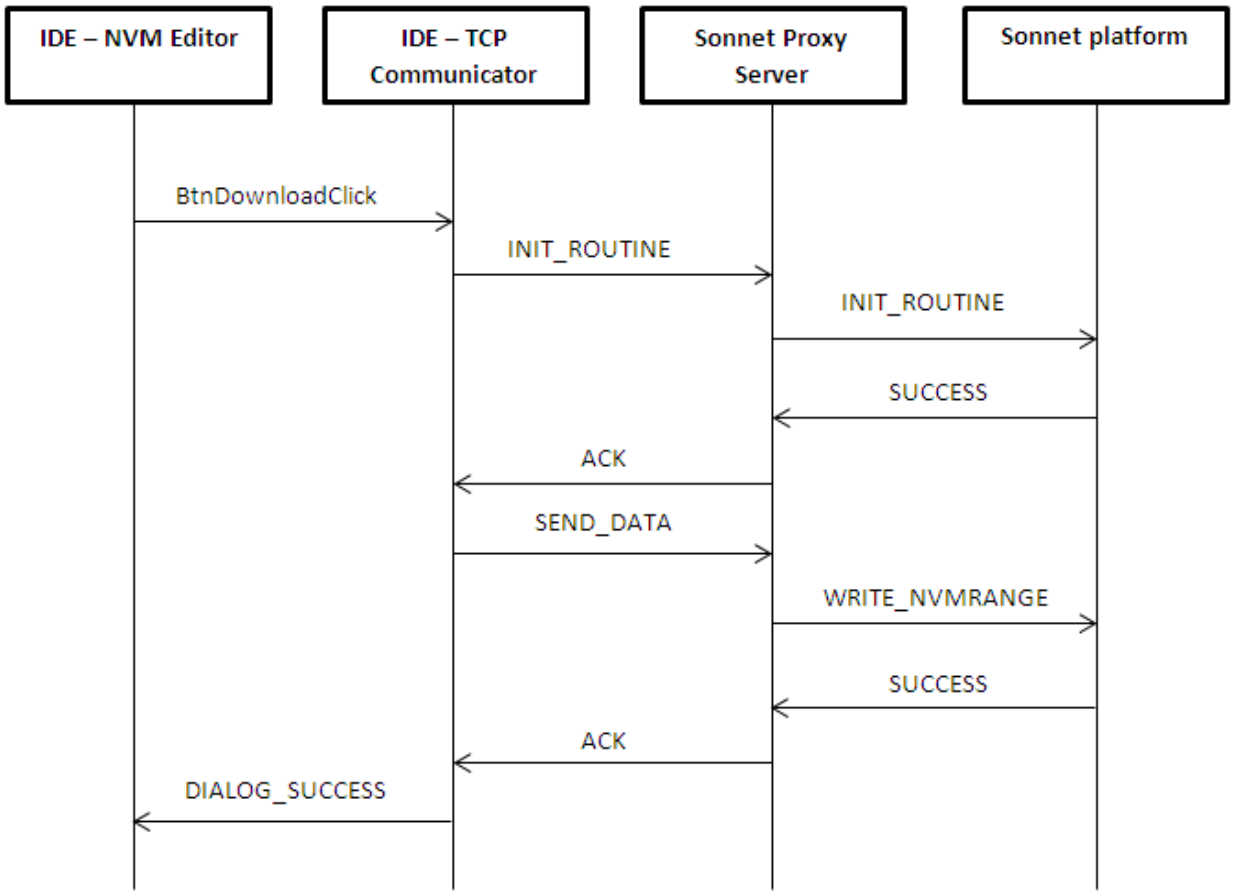
U prethodnom delu objašnjen je način dodavanja, podešavanja i stvaranja datoteka unutar NVM uređivača. Na stranici sa opštim informacijama NVM uređivača korisniku su na raspolaganju akcije koje omogućavaju spregu sa platformom. Pomoću tih akcija moguće je preuzeti sistem datoteka koji je trenutno na platformi ili upisati novi, podešen od strane korisnika. Komunikacija između okruženja i platforme odvija se preko uslužnog servera za kontrolisano izvršavanje programa - Sonnet Proxy Server modula, skraćeno SPS.

3.7.1 Sonnet Proxy Server modul

SPS je sprežna komponenta prema ciljnoj platformi koja izvršava korisničke naredbe vezane za podešavanje fizičke arhitekture i odgovarajuće programske podrške, operacije čitanja i upisa podataka u registre, memorijske lokacije, sistem datoteka, kontrolisanog izvršavanja programa, postavljanja prekidnih tačaka kao i operacije pristupa perifernim jedinicama na razvojnoj ploči. Poruke koje se razmenjuju protokolom za razmenu podataka (eng. Transfer Control Protocol – TCP) sastoje se od nekoliko reči. Prva reč koja koja prenosi zahtev prema serveru je ime zahteva. Na osnovu imena zahteva, SPS identifikuje metodu koja je zahtevana i ostatak poruke prevodi u parametre te metode. Modul je podržava komunikaciju sa više različitih klijenata, bilo da je to komunikacija sa razvojnim okruženjem, skript jezicima ili alatima komandne linije.

3.7.2 Upis sistema datoteka na platformu

Priskom na dugme *„Download image to the NVM device“* stvorene binarne datoteke se šalju na platformu. Datoteka nvmXML.xml sadrži listu binarnih datoteka sortiranih po rastućoj adresi koja im je podešena ili dodeljena u uređivaču. Podaci se izvlače iz datoteka i upisuju u *StringBuffer* strukturu podataka. Ova struktura je u mogućnosti da skladišti velik broj karaktera. Skladište se podaci veličine četiri bajta međusobno odvojeni zarez (',') karakterom i šalju protokolom za prenos podataka do SPS modula. Korisnik u vidu dijaloga biva obavešten porukom da li su podaci uspešno poslati ili je došlo do greške. Ova komunikacija prikazana je preko dijagrama niza poruka (slika 10).



Slika 10. Dijagram razmene poruka prilikom upisa sistema datoteka

3.7.3 Čitanje sistema datoteka sa platforme

Pritiskom na *"Upload image from the NVM device to this editor"* korisnik učitava informacije NVM sistemu datoteka direktno sa platforme. Ovaj vid komunikacije okruženja i platforme se takođe odvija preko SPS modula. Na isti način na koji su podaci upisivani na platformu, pomoću niza karaktera, vrši se i čitanje. Čitajući prvih 36 bajta podataka, koji predstavljaju prve dve datoteke sistema (sa rezervisanom adresom), moguće je otkriti informacije o svim datotekama u sistemu. Na osnovu primljenih podataka, reprodukuju se parametri svih datoteka i prikazuju u NVM uređivaču. Ukoliko je korisnik imao podešene datoteke u uređivaču, one neće biti prepisane, već će učitane datoteke biti dodate postojećem sistemu datoteka u uređivaču. Parametri koji ne mogu biti pročitani, kao što je putanja do određene datoteke na sistemu datoteka računara sa koje je dodata u uređivač, će biti onemogućeni. Datoteke koje su pročitane sa platforme imaju tekstualnu napomenu u nazivu, kako bi korisnik razlikovao datoteke koje je on podesio i koje su pročitane.

4. Programsko rešenje

Programska realizacija ovog rada napisana je u Java programskom jeziku i primenjen je objektno orijentisan pristup programiranja. Korišćeno je *Eclipse Indigo* razvojno okruženje sa *GCC* prevodiocem. Za razvoj Java koda korišćen je paket alata JDK 1.7 (*eng. Java Development Kit*).

Rešenje se sastoji iz dva realizovana projekat-priključka (*eng. Plug In project*). Jedan priključak sadrži model, odnosno osnovu programske podrške na koju se naslanja programska podrška drugog priključka koja se odnosi na spregu sa korisnikom.

4.1 Priključak grafičke korisničke sprege

4.1.1 Klasa EepromConfigurationEditor

Opis:

- Klasa u kojoj se stvara višestranični NVM uređivač

Konstruktori:

- Nasleđen podrazumevani konstruktor

Atributi:

- generalSettingsPage – instanca stranice uređivača o opštim informacijama o platformi
- masterDetailsPage – instanca stranice uređivača koja se sastoji iz dva dela, glavnog dela za prikaz dodatih datoteka i dela za prikaz detalja datoteka
- statisticsPage – instanca stranice uređivača za prikaz rasporeda sistema datoteka po početnim adresama, raspoložive memorije itd...
- circuitDataPage – instanca stranice uređivača za prikaz rezervisane datoteke

Metode:

- addPages – metoda za dodavanje stranica u uređivač
- doSave – metoda za snimanje sadržaja stranica uređivača

- init – metoda za inicijalizaciju atributa klase
- setInput – metoda za učitavanje sadržaja stranica uređivača sa diska
- resetInput – metoda za poništavanje sadržaja stranica uređivača i postavljanja vrednosti na početne
- setIsDirty – metoda za obeležavanje stranice uređivača da je došlo do promene parametara datoteke u odnosu na poslednju sesiju
- isDirty – metoda za proveru parametara datoteke (da li su promenjeni) stranice uređivača
- dispose – metoda za „uništavanje“ grafičkih komponenti stranica uređivača
- refreshAllPages – metoda za osvežavanje stranica uređivača

4.1.2 Klasa CommonEepromPage

Opis:

- Klasa u kojoj se stvara grafička korisnička sprega koju koriste neke od stranica NVM uređivača

Konstruktori:

- Parametri konstruktora su instanca FormEditor-a, naslov stranice i identifikator stranice

Atributi:

- scrolledForm, managedForm, shell – atributi koji se odnose na grafički format stranice
- dirty – boolean atribut, pokazuje da li je promenjen parametar datoteke u stranici koja nasleđuje CommonEepromPage klasu
- refreshing – boolean atribut, pokazuje da li se stranica osvežava

Metode:

- createFormContent – metoda u kojoj se prave grafičke komponente stranice
- readModelFromFile – metoda u kojoj se učitava sadržaj stranice
- refreshMarkers – metoda u kojoj se osvežava sadržaj “Problems” pogleda

4.1.3 Klasa GeneralSettingsPage

Opis:

- Klasa u kojoj se definiše kompletan izgled i sadržaj stranice NVM uređivača koja prikazuje podatke o ciljnoj platformi, pokretačkom programu i ključevima za šifrovanje

Konstruktori:

- Parametar konstruktora je instanca EepromConfigurationEditor klase

Atributi:

- bootLoaderFileTextField, bootLoaderEditBtn, nvmInterfaceLabel, bootLoaderRecordLabelsComposite, client, initializeBtns, encryptBtns, initListeners, encryptListeners – atributi koji se koriste za stvaranje komponenti grafičke sprege

Metode:

- createFormContent - metoda u kojoj se prave grafičke komponente stranice
- createGeneralSection, createBootLoaderSection, createEncryptionKeysSection, createEepromDefinitionSection– metode u kojima se prave sekcije sa specifičnim grafičkim komponentama
- addListeners – metoda koja dodaje slušaoce (eng. listeners) za komponente grafičke sprege
- validateFields – metoda u kojoj se proveravaju parametri datoteka ove stranice
- setBootLoaderErrorMessage, setEncryptionKeyErrorMessage – metode u kojima se šalju poruke greške, ukoliko takve postoje, uređivaču i "Problems" pogledu
- runElfReader – metoda u kojoj se poziva alat komandne linije potreban za obradu ELF datoteka

4.1.4 Klasa FileSystemPage

Opis:

- Klasa u kojoj se definiše kompletan izgled i sadržaj stranice u kojoj se dodaju datoteke programske podrške. Sadrži glavni deo u kom je prikazana lista dodatih datoteka, i deo za detalje u kom si prikazani detalji o parametrima odabrane datoteke iz liste.

Konstruktori:

- Podrazumevani konstruktor

Atributi:

- btnAddOverlay, btnAddParameterSet, btnAddHardwareConfiguration, btnAddDataLog, btnAddBrandingInfo, btnAddAudioSample, btnRemove – dugmad na glavnom delu stranice koja otvaraju čarobnjake za dodavanje datoteka programske podrške

Metode:

- createMasterPart – metoda u kojoj se prave komponente grafičke korisničke sprege glavnog dela stranice
- addListeners – metoda koja dodaje slušaoce (eng. listeners) za komponente grafičke sprege
- validate – metoda u kojoj se vrši provera podešenih parametara datoteka
- refreshTreeViewer – metoda koja osvežava sadržaj stabla koje sadrži listu dodatih datoteka

4.1.5 Klasa FileDetailsPage

Opis:

- Klasa koja definiše izgled pomoćnog dela stranice za dodavanje datoteka programske podrške.

Konstruktori:

- Parametar konstruktora je instanca FileSystemPage stranice.

Atributi:

- mform, detailsSection, fsp, witeToNvm, client, editButton, filePathTextField, input, startingAddress, checkAddress – atributi koji se koriste za stvaranje grafičkih komponenti koje čine pomoćnu sekciju

Metode:

- selectionChanged – metoda koja se poziva kada datoteka iz stabla bude selektovana (prelaz sa jedne datoteke na drugu)
- createContents – metoda u kojoj se prave komponente grafičke korisničke sprege dela za prikaz detalja datoteka
- addListeners – metoda koja dodaje slušaoce (eng. listeners) za komponente grafičke sprege
- validate – metoda u kojoj se vrši provera podešenih parametara datoteke

4.1.6 Klasa BlockHeaderDetailsPage

Opis:

- Klasa koja definiše izgled pomoćnog dela prilikom prikaza detalja segmenata i zaglavlja datoteka programske podrške

Konstruktori:

- Podrazumevani konstruktor

Atributi:

- mform, detailsSection, targetBankID, input, targetAddress, fileOffset, blockLength, nextHeaderOffset – atributi koji se koriste za stvaranje komponenti grafičke korisničke sprege

Metode:

- initialize – metoda koja inicijalizuje parametre datoteke
- refresh – metoda za osvežavanje sekcije
- createContents - metoda u kojoj se prave komponente grafičke korisničke sprege za prikaz detalja segmenata i zaglavlja
- addListeners – metoda koja dodaje slušaoce (eng. listeners) za komponente grafičke sprege

4.1.7 Klasa StatisticsPage

Opis:

- Klasa koja definiše izgled stranice za prikaz rasporeda sistema datoteka

Konstruktori:

- Parametar konstruktora je instanca EepromConfigurationEditor klase

Atributi:

- infoClient, client, spaceGridData, toolkit, labels, editor – atributi koji se koriste za stvaranje komponenti grafičke sprege

Metode:

- createFormContent – metoda u kojoj se prave komponente grafičke sprege
- refresh – metoda koja osvežava stranicu uređivača
- refreshLabels – metoda koja osvežava ispis sistema datoteka po početnim adresama

- getTotalFreeSpace – metoda koja računa ukupan adresni prostor sistema datoteka
- pageChanged – metoda koja se proziva prilikom promene stranice uređivača

4.2 Priključak modela programske podrške

4.2.1 Klasa EepromProject

Opis:

- Klasa koja sadrži modele svih datoteka sistema

Konstruktori:

- Podrazumevani konstruktor

Atributi:

- eepromDefinitionRecord – objekat klase EepromDefinitionRecord koja definiše datoteku za opis ciljne platforme
- encryptionKeyData – objekat klase EncryptionKeyData koja definiše datoteku ključeva za šifrovanje podataka
- nvmBootLoaderFile – objekat klase NvmBootLoaderFile koja definiše datoteku pokretačkog programa
- secondaryBootRecord – objekat klase SecondaryBootRecord koja definiše izgled datoteke koja sadrži opšte informacije o datotekama programske podrške
- scheduler – objekat klase NvmScheduler koja vrši raspored sistema datoteka
- imageGenerator – objekat klase EepromImageGenerator koja stvara binarne datoteke sistema

Metode:

- createNvmXMLOutput – metoda koja stvara datoteku koja sadrži spisak svih napravljenih binarnih datoteka koje čine sistem
- getDOMTree – metode koja definiše format .xml datoteke
- createSectionImages – metoda koja stvara binarne datoteke koje čine sistem
- checkInterfaceType – metoda za proveru sprege (I2C ili SPI)
- canAdd – metoda za raspoređivanje datoteka; provera da li za datoteku postoji dovoljno veliki slobodan prostor
- schedule – metoda koja raspoređuje datoteke

4.2.2 Klasa FirmwareFile

Opis:

- Klasa koja sadrži model datoteke programske podrške

Konstruktori:

- Parametar konstruktora je tip datoteke programske podrške

Atributi:

- firmwareFileHeader - objekat klase FirmwareFileHeader koja sadrži podatke o zaglavljima i segmentima datoteke programske podrške
- coreType – tip procesorskog jezgra

Metode:

- getDOMTree – metoda koja definiše format .xml datoteke
- calculateFirmwareFileLength – metoda koja računa veličinu datoteke
- getErrorMessages – metoda koja vraća poruku u grešci ukoliko je neki od parametara pogrešno podešen

4.2.3 Klasa ReadElfProcessForFirmwareFiles

Opis:

- Klasa u kojoj se poziva alat komandne linije koji raščlanjuje datoteku programske podrške na zaglavlja i segmente

Konstruktori:

- Podrazumevani konstruktor

Atributi:

- fileHeader – objekat klase FirmwareFileHeader u koji se upisuju iščitani podaci zaglavlja
- blockHeaders – lista objekata klase FirmwareBlockHeaders u koje se upisuju iščitani podaci segmenata
- errorMessage – poruka o grešci
- outputFile – privremena datoteka u koju alat upisuje raščlanjene podatke

Metode:

- run – metoda u kojoj se pokreće alat komandne linije
- isOk – provera da li je alat uspešno raščlanio datoteku
- parseElfReaderOutput – metoda u kojoj se popunjavaju strukture podataka sa podacima pročitanim u privremenoj datoteci

4.2.4 Klasa NVMBootLoaderFile

Opis:

- Klasa koja sadrži model datoteke pokretačkog programa

Konstruktori:

- Podrazumevani konstruktor

Atributi:

- startingAddress – vrednost početne adrese
- enableStartAddress – indikator da li je korisnik uneo početnu adresu ili je ona dodeljena
- filePath – putanja do datoteke na računaru
- coreType – tip procesorskog jezgra
- uploaded – indikator da li je datoteka pročitana sa platforme ili ne
- bootLoaderHeaders – zaglavlja datoteke pokretačkog programa

Metode:

- getAddress – metoda koja vraća vrednost početne adrese
- getSize – metoda koja vraća veličinu datoteke
- getErrorMessages – metoda koja vraća poruku o grešci ukoliko je neki od parametara pogrešno podešen
- getDOMTree – metoda koja definiše format .xml datoteke

4.2.5 Klasa ReadElfProcessForBootLoader

Opis:

- Klasa u kojoj se poziva alat komandne linije koji raščlanjuje datoteku pokretačkog programa na zaglavlja i segmente

Konstruktori:

- Podrazumevani konstruktor

Atributi:

- exitValue – indikator da li je alat komandne linije uspešno raščlanio datoteku
- outputFile – privremena datoteka u koju alat komandne linije upisuje raščlanjene podatke datoteke
- errorMessage – poruka o grešci
- nvmBootLoaderFile – objekat klase NVMBootLoaderFile u koji se upisuju pročitani podaci iz privremene datoteke

Metode:

- run – metoda u kojoj se pokreće alat komandne linije
- parseElfReaderOutput – metoda u kojoj se popunjavaju strukture podataka sa podacima pročitanim u privremenoj datoteci

4.2.6 Klasa NVMScheduler

Opis:

- Klasa koja definiše algoritam za raspoređivanje sistema datoteka

Konstruktori:

- Parametar konstruktora je objekat klase EepromProject, odnosno model svih datoteka sistema

Atributi:

- freeSlots – lista koja sadrži vrednosti slobodnih opsega adresnog prostora

Metode:

- getFullSize – metoda koja vraća vrednost ukupne veličine adresnog prostora
- getFreeSpace – metoda koja vraća vrednost ukupne veličine slobodnog adresnog prostora
- canAddFileAtFixedAddress – metoda koja proverava da li je moguće dodati datoteku na adresu koju je korisnik uneo za početnu
- canAddFileAtCalculatedAddress – metoda koja proverava da li je moguće dodati datoteku u neki od slobodnih opsega adresnog prostora
- addFile – metoda koja dodaje datoteke u sistem

4.2.7 Klasa LFSREncryptionAlgorithm

Opis:

- Klasa koja definiše algoritam šifrovanje podataka

Konstruktori:

- Parametar konstruktora je ključ za šifrovanje

Atributi:

- TYPE_NVMMASK, TYPE_I2C, TYPE_SIZEMASK, TYPE_24BIT, TYPE_SPI16, TYPE_SPI24, TYPE_32BIT – atributi pomoću kojih se definiše tip šifrovanja, veličina podataka za šifrovanje itd...

Metode:

- LFSR_encrypt – metoda u kojoj se vrši šifrovanje podataka
- LFSR_get – metoda koja generiše ključeve za šifrovanje

4.2.8 Klasa DownloadNVMImage

Opis:

- Klasa koja definiše algoritam za slanje sistema datoteka na paltformu

Konstruktori:

- Podrazumevani konstruktor

Atributi:

- binaryData – objekat klase StringBuffer u koji se smeštaju podaci binarnih datoteka
- entry – binarna datoteka iz koje se podaci upisuju u StringBuffer
- length – dužina podataka koji se odvajaju zarez karakterom. Mogu biti tri ili četiri bajta dugi
- valuesArray – nizovi podataka binarnih datoteka

Metode:

- appendData – metoda u kojoj se formira niz podataka koji se smešta u StringBuffer
- writeNVM – metoda u kojoj se šalje binaryData ka SPS modulu
- getErrorMessage – metoda koja prima poruku o grešci od strane SPS modula

4.2.9 Klasa UploadNVMImage

Opis:

- Klasa u kojoj se vrši čitanje sistema datoteka

Konstruktori:

- Podrazumevani konstruktor

Atributi:

- Nema

Metode:

- readData – metode u kojoj se vrši čitanje podataka primljenih sa platforme od strane SPS modula
- uploadEncryptionKeys, uploadCircuitDataRecord, uploadFileSystemDirectories, uploadDefinitionRecord, uploadSecondaryBootLoaderRecord, uploadNVMBootLoaderRecords, readFirmwareFiles, readContstants – metode koje se koriste za učitavanje svih datoteka koje čine sistem datoteka

5. Ispitivanje i verifikacija

U okviru ovog rada obavljena su ispitivanja rada sa sistemom datoteka koristeći nezavisne JUnit ispitne slučajeve [11]. JUnit je okvir predviđen za jedinično ispitivanje pojedinačnih Java klasa ili grupe klasa. Filozofija koja se promoviše uz JUnit je pisanje ispitnih slučajeva pre razvijanja same programske podrške (eng. *test-first*).

U radu su verifikovani algoritmi stvaranja, upravljanja, čitanja, pisanja i raspoređivanja datoteka u sistem. Cilj ovih ispitnih slučajeva je bio da se potvrdi efikasno i precizno korišćenje raspoloživog adresnog prostora (memorije) na platformi prilikom rada sa datotekama. Prilikom upisivanja ili čitanja sistema datoteka svaki bit je moguće ispitati, što ovaj sistem čini pogodnim za verifikaciju, ali ujedno i komplikovanim zbog toga što se može napisati velik broj ispitnih slučajeva. U ovom poglavlju, biće pojašnjeni neki od njih.

5.1 Ispitivanje rezervisanih datoteka

Kao što je u prethodnom delu rada spomenuto, dve datoteke u sistemu datoteka imaju rezervisanu početnu adresu. Čitajući podatke tih datoteka, mikrokontroler prosleđuje informacije ostalim procesorima koji ih dalje obrađuju.

„Magična reč“ veličine dva bajta je podatak koji se prvi upisuje u datoteku za opis ciljne platforme. Vrednost te reči je konstanta koja iznosi 0xCA53. Ispitni slučaj provere ove konstante izvršava se prilikom čitanja i pisanja datoteka. Ukoliko prva dva bajta u sistemu datoteka nisu jednaka ovoj konstanti, obrada podataka na platformi neće biti moguća.

Veličina dve rezervisane datoteke mora biti 36 bajta podataka velika. To su fiksne veličine datoteka, 16 bajta za datoteku opisa ciljne platforme, 20 bajta za datoteku koja učitava adresni prostor sistema datoteka. Provera veličina rezervisanih datoteka prilikom upisa i čitanja sistema predstavlja nezavisan ispitni slučaj.

Svaki bit od 36 bajta podataka ima svoju važnost. U zavisnosti od podešenih parametara koji se odnose na tip platforme, veličinu, početnu adresu, broj dodatih datoteka, polja u kojima se upisuju ti podaci su ispitivana i proveravana da li odgovaraju onim podacima koji su unapred definisani da predstavljaju taj odabir.

U tabeli 1 prikazani su rezultati ispitnih slučajeva koji se odnose na verifikaciju rezervisanih datoteka. Ispitni slučajevi odnose se na proveru tipa NVM memorije, provere veličine i adrese datoteke pokretačkog programa, provere „magične“ reči, ukupne veličine adresnog prostora i odabira takta.

Naziv ispitnog slučaja	Uspešan	Neuspešan
testMasterClockDivider	☑	
testNVMInterfaceClockDivider	☑	
testNVMType	☑	
testClockSelect	☑	
testClockUpdateEnabled	☑	
testMagicWordFirstBytePadded	☑	
testAddressTypeLastBytePadded	☑	
testAddressTypeFirstBytePadded	☑	
testBootLoaderAddress	☑	
testBootLoaderSize	☑	

Tabela 1. Rezultati ispitnih slučajeva rezervisanih datoteka

5.2 Ispitivanje datoteka programske podrške

Važan deo sistema datoteka predstavljaju datoteke programske podrške koje sadrže izvorni kod, odnosno instrukcije koje će se izvršavati na samoj platformi. Ispitivanje ovih datoteka je složeno zato što sadrže veliki broj informacija. Kao što je već spomenuto dodavanjem izvršnih ELF datoteka, alat komandne linije će raščlaniti datoteku na segmente i zaglavlja koji će se prilikom stvaranja nove datoteke upisati kao binarne informacije. Kako ove datoteke nisu fiksne veličine, već variraju od jedne do druge datoteke, samo izračunavanje veličine na osnovu podataka zaglavlja je zahtevan proces. Precizno izračunavanje veličine datoteke je važan ispitni slučaj jer se ta informacija koristi prilikom čitanja i upisa sistema.

Datoteka može imati više zaglavlja, gde svako od zaglavlja može imati više segmenata. Kada korisnik preko čarobnjaka u okruženju doda željenu datoteku, podaci o njihovim veličinama se čuvaju u Javinim strukturama podataka sve dok korisnik ne stvori binarnu datoteku. Veličina koja je upisana u binarnoj datoteci mora biti jednaka vrednosti koja je pročitana iz sistema (podatak se čita na osnovu početne adrese datoteke koju je korisnik zadao, ili koja je raspoređena).

Kako podaci zaglavlja i segmenata mogu biti šifrovani određenim ključem, napisani su i ispitni slučajevi za verifikaciju LFSR algoritma. Implementacija LFSR algoritma za potrebe šifrovanja podataka u razvojnom okruženju, rađena je u Java programskom jeziku na osnovu referentnog C/C++ koda. Ispitni slučajevi su realizovani tako da podaci šifrovani određenim ključem prvo propuste kroz referentni algoritam, a zatim kroz algoritam korišćen za potrebe SIDE razvojnog okruženja. Ti podaci su zatim upisivani u datoteke, čiji podaci su se poredili i verifikacija bi uspela ukoliko su podaci šifrovani podaci jednaki. Algoritam je verifikovan korišćenjem različitih ključeva, kao i korišćenjem niza podataka različite veličine (32 ili 24 bita).

U tabeli 2 prikazani su rezultati ispitnih slučajeva koji se odnose na verifikaciju datoteka programske podrške. Ispitni slučajevi odnose se na proveru veličina datoteka, njihovih zaglavlja i šifrovanja podataka.

Naziv ispitnog slučaja	Uspešan	Neuspešan
testFileName	☑	
testFirmwareBlockHeader	☑	
testFirmwareFileHeader	☑	
testHardwareConfigurationSize	☑	
testLFSRWithType1	☑	
testLFSRWithDifferentSeed	☑	
testBootLoaderValidation	☑	
testCRCBytes	☑	

Tabela 2. Rezultati ispitnih slučajeva datoteka programske podrške

5.3 Ispitivanje upisa i čitanja sistema datoteka

Operacije upisa i čitanja sistema datoteka predstavlja važan deo razvijene podrške. Ispitni slučaj za proveru pravilnog upravljanja datotekama sastoji se iz podešavanja sistema u

uređivaču, upisa na platformu, a zatim čitanja. Rezultat toga trebalo bi da bude da su upisani podaci jednaki onima koji su iščitani sa platforme. Prilikom pisanja ovih ispitnih slučajeva, korišćen je referentni sistem datoteka obezbeđen od strane proizvođača platforme. Isti takav sistem datoteka podešen je i u NVM uređivaču, a zatim upisan na platformu. Rezultat toga bila je podjednaka obrada podataka na platformi, bilo da je korišćen referentni sistem, bilo onaj podešen u uređivaču.

U tabeli 3 prikazani su rezultati ispitnih slučajeva koji se odnose na verifikaciju operacija upisa i čitanja sistema datoteka.

Naziv ispitnog slučaja	**Uspešan**	**Neuspešan**
testImageExists	☑	
testFileLengths	☑	
testFlagsBits	☑	
testDataWordSize	☑	
testStartingAddresses	☑	
testReservedSections	☑	

Tabela 3. Rezultati ispitnih slučajeva prilikom upisa i čitanja sistema datoteka

6. Zaključak

U radu je realizovana podrška za upravljanje sistemom datoteka na Sonnet višeprocesorskoj platformi u okviru Sonnet integrisanog razvojnog okruženja zasnovanog na Eclipse platformi. Realizovano je dodavanje, brisanje, ispitivanje, podešavanje parametara datoteka u okviru razvojnog okruženja , kao i upis i čitanje datoteka sa Sonnet platforme.

Da bi se omogućilo upravljanje sistemom datoteka na fizičkoj platformi, razvijena je odgovarajuća podrška u okviru SIDE razvojnog okruženja. Ova podrška se sastoji iz grafičke sprege sa korisnikom kao i sprege okruženja i platforme. Grafička korisnička sprega obuhvata poseban projekat za podešavanje sistema datoteka, korišćenje posebnog uređivača, čarobnjaka, pogleda, prečica, uputstva za korišćenje, koji korisniku umnogome olakšavaju upravljanje datotekama. Sprega koja obezbeđuje komunikaciju okruženja i platforme, posredstvom Sonnet Proxy Servera, realizovana je putem TCP veze. Ova sprega omogućava slanje i primanje podataka, odnosno čitanje i upis sistema datoteka na fizičku platformu.

Rešenje je ispitano skupom JUnit ispitnih slučajeva, čime je potvrđena ispravnost predloženog rešenja. Potvrđena je pouzdanost upravljanja sistemom datoteka i u razvojnom okruženju i na platformi.

Dalji razvoj podrške za upravljanje sistemom datoteka usmeren je ka poboljšanju algoritma dodavanja i raspoređivanja datoteka u sistem. Poboljšanje bi podrazumevalo prepoznavanje nepotrebnih podataka (eng. Zero blocks) u izvršnim datotekama, čime bi došlo do uštede adresnog prostora (manje podataka za upis), a samim tim i povećanja raspoložive memorije. Takođe, dodavanje datoteka metodom prevuci i otpusti (eng. drag-and-drop) iz sistema datoteka računara u uređivač razvojnog okruženja, samo su neke od ideja za dalji razvoj ove podrške.

7. Literatura

[1] Prof. Dr Vladimir Kovačević, Prof. Dr Miroslav Popović: *Sistemska programska podrška u realnom vremenu I*, Novi Sad, 2011.

[2] Lars Vogel: *Eclipse IDE: Eclipse IDE based on Eclipse 4.2 and 4.3*, 2013.

[3] Zhihui Yang, Michael Jiang: *Using Eclipse as a Tool-Integration Platform for Software Development*, 2007.

[4] Jeff McAffer, Jean-Michel Lemieux, Chris Aniszczyuk: *Eclipse Rich Client Platform second edition*, 2010.

[5] Bruno Dinis Ormonde Medeiros: *Creation of an Eclipse-based IDE for the D programming language*, 2007.

[6] *Eclipse Platform Technical Overview*, International Business Machines Corp., 2006.

[7] Vladimir Kovačević, Miroslav Popović, Miodrag Temerinac, Nikola Teslić: *Arhitekture i algoritmi digitalnih signal procesora I*, Novi Sad, 2005.

[8] Jennifer Eyre, Jeff Bier: *The Evolution of DSP Processors*, 2000.

[9] Liang W.: *A cryptographic algorithm based on Linear Feedback Shift Register*, 2010.

[10] Prof. Dr Vladimir Kovačević, Prof. Dr Miroslav Popović: *Sistemska programska podrška u realnom vremenu II*, Novi Sad, 2011.

[11] P.Tahchiev, F.Leme, V.Massol, G.Gregory: *JUnit in Action*, Second Edition, 2010.