

VEŽBA 1 – Upoznavanje sa alatima i razvojnim okruženjem ciljne DSP platforme

1.1 Uvod

Razvoj softvera za digitalne signal procesore (DSP u daljem tekstu) podrazumeva odgovarajuće softverske alate: assembler, povezič, kompajler, alat za kontrolisano izvršavanje softvera, simulator, uređivače izvornog koda, alate za profilisanje koda i sl. Ovi softverski alati se najčešće distribuiraju kao jedan zajednički softverski paket pod nazivom integrisano razvojno okruženje (eng. *Integrated Development Environment*).

Pored simulatora, za izvršavanje i ispitivanje softvera (aplikacija) se koriste razvojne ploče. Pojednostavljeno, razvojna ploča predstavlja hardver na kome centralni deo čini DSP, dok ostatak čine periferne jedinice koje se često sreću u realnim sistemima ili hardverske komponente koje omogućavaju ispitivanje same aplikacije. Primer tipičnih hardverskih komponenti na razvojnoj ploči su tasteri, prekidači, ekran, konektori za različite sprege (analogne i digitalne), dodatni mikrokontroleri, A/D ili D/A pretvarači.

Prvi korak prilikom početka rada sa bilo kojom računarskom arhitekturom je upoznavanje sa razvojnim alatima i hardverskim uređajima namenjenim razvoju softvera za tu arhitekturu.

U okviru ove vežbe prikazano je razvojno okruženje procesora CS48x i kako se:

- vrše operacije nad projektima (otvaranje i formiranje različitih tipova projekata, osvrt na bitne opcije),
- dodaju postojeće i prave nove datoteke sa izvornim kodom,
- tumače prijavljene greške prilikom prevođenja i uvezivanja,
- pokreću izvršavanje programa na simulatoru,
- pokreću izvršavanje programa na razvojnoj ploči,
- vrši kontrolisano izvršavanje programa – upotreba alata za pomoć prilikom otklanjanja grešaka tokom izvršenja (*Prikaz memorije, Prikaz izraza, Tačke prekida*).

1.2 Razvojni alati platforme CS48x

1.2.1 Integrisano razvojno okruženje CLIDE

CLIDE (eng. *Cirrus Logic's Integrated Development Environment*) predstavlja integrisano razvojno okruženje za razvoj programske podrške za DSP procesore proizvođača *Cirrus Logic* (CS47x, CS48x, CS49x familije procesora). Ovo razvojno okruženje obezbeđuje potpuno funkcionalne uređivače teksta za C ili asemblerski kod, mogućnost prevođenja aplikacije, podršku za smeštanje i spuštanje DSP aplikacije na razvojnu ploču, kontrolisano izvršavanje programa i profilisanje koda koji se izvršava. Pomenuti skup alata uključuje C programski prevodilac, asembler, povezič, simulator, prozore za uređivanje izvornog koda, alat za kontrolisano izvršenje koda (eng. *debugger*), alat za profilisanje koda (eng. *profiler*), i mnoge druge. Pored navedenog, CLIDE omogućava i grafičko uređivanje aplikacija. Ovaj režim podrazumeva da se određenim celinama programskog koda dodele predstave u vidu grafičkih blokova. Koristeći grafički uređivač moguće je slagati pomenute grafičke blokove, postavljati veze između njih i na osnovu napravljenog modela generisati izvorni kod aplikacije. CLIDE je razvijen na Odseku za računarsku tehniku i računarske komunikacije u Novom Sadu, uz saradnju sa stručnjacima iz RT-RK Instituta.

CLIDE razvojno okruženje zasnovano je na Eclipse platformi. Eclipse se često koristi kao osnov mnogih razvojnih okruženja. Otvorenog je koda i nudi mnoštvo korisnih alata za razvoj programske podrške. CLIDE proširenja u Eclipse-u podržavaju upotrebu specifičnih proširenja DSP procesora proizvedenih od strane Cirrus Logic kompanije.

U nastavku je dat opis osnovnih aspekata korišćenja CLIDE-a, potrebnih za dalje savladavanje i praćenje laboratorijskih vežbi. U toku čitanja ove vežbe podrazumeva se korišćenje CLIDE razvojnog okruženja. Oznakom „**Zadatak**“ označena je akcija koju je potrebno izvršiti pre nastavka čitanja teksta.

1.2.2 Pokretanje CLIDE razvojnog okruženja

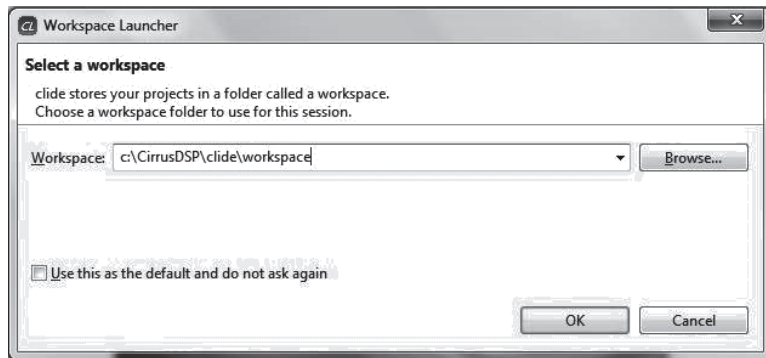
Zadatak:

- o Pokrenuti CLIDE razvojno okruženje.

Nakon pokretanja CLIDE razvojnog okruženja, od korisnika se zahteva da izabere putanju do radnog prostora (*workspace*). *Workspace* je direktorijum na disku koji će biti korišćen za čuvanje informacija tokom rada sa CLIDE okruženjem (aktivni projekti, podešavanja izgleda okruženja, log...). Putanja do radnog prostora ujedno predstavlja i podrazumevanu putanju prilikom pravljenja novih projekata.

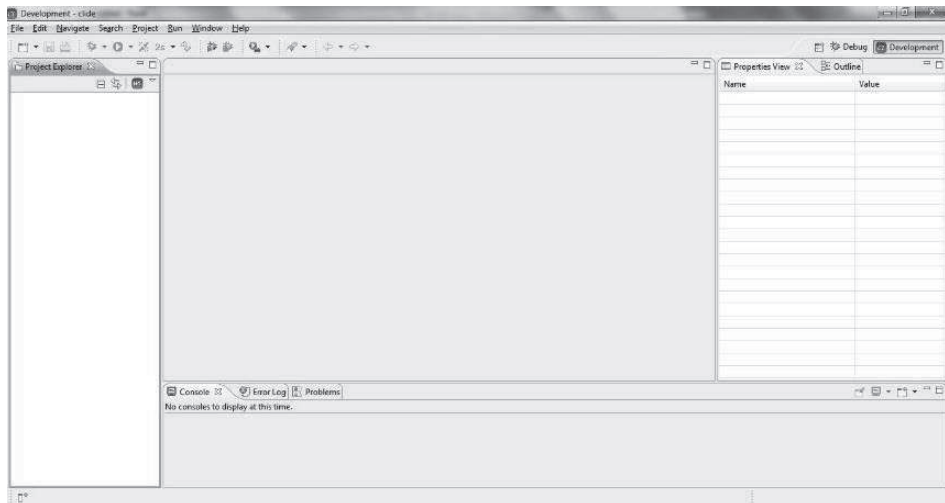
Zadatak:

- o Odabrati željenu putanju do radnog prostora.
- o Ukoliko je obeležena opcija *“Use this as the default...”*, izabrani direktorijum biće smatran za podrazumevani radni prostor i ovaj prozor se više neće pojavljivati prilikom pokretanja CLIDE.



Slika 1.1 - Odabir putanje do radnog prostora

Radni direktorijum je moguće naknadno promeniti iz *File -> Switch Workspace* menija. Nakon pokretanja CLIDE razvojnog okruženja i izbora radnog prostora, biće prikazan glavni prozor CLIDE programa, kao na slici broj 1.2.

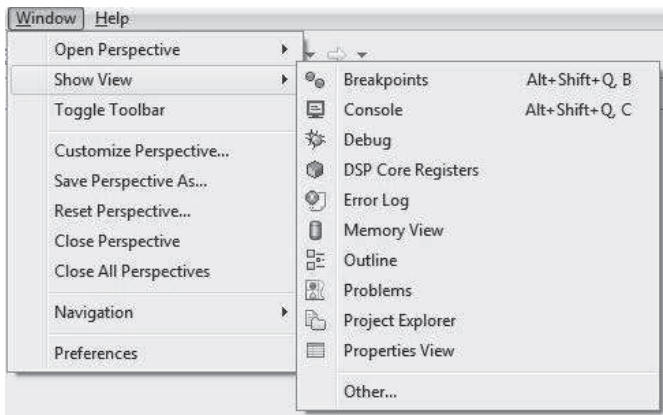


Slika 1.2 - CLIDE, glavni prozor

1.2.3 Prikazi i perspektive

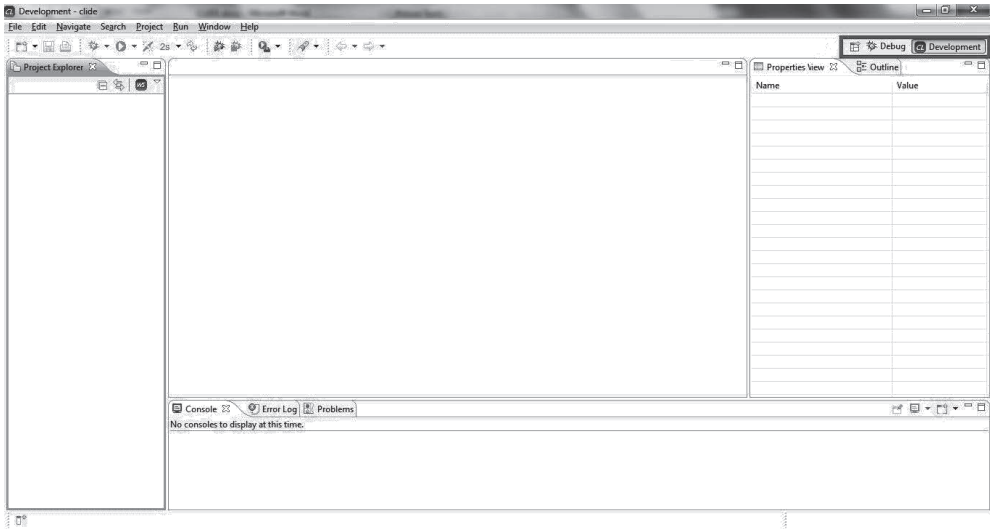
Prikaz (*eng. view*) predstavlja vizuelnu komponentu (oblast) koja prikazuje određenu vrstu informacije. Prikazi se obično koriste za navigaciju kroz neku listu ili hijerarhiju pojedinih informacija. Na slici 1.2 je prikazan glavni prozor CLIDE razvojnog okruženja i on se sastoji od nekoliko različitih prikaza: *Project Explorer*, *Console*, *Properties*. Pojedini tipovi prikaza mogu biti aktivni ili prikazani tokom nekih aktivnosti. Sadržaj pojedinih prikaza može da zavisi od trenutno aktivnog uređivača. Otvaranje nekog prikaza je moguće uraditi kroz *Window* meni, kao što je prikazano na slici 1.3.

Perspektiva definiše skup i raspored aktivnih prikaza u radnom prozoru. CLIDE omogućava postojanje više različitih perspektiva, međutim, u određenom trenutku samo jedna od njih može biti aktivna. Perspektive su obično definisane tako da sadrže skup prikaza i alata potrebnih za ostvarenje nekog tipa zadatka. Na primer, u okviru *Development* perspektive, aktivni su najčešće korišćeni prikazi i alati koji se koriste tokom pisanja izvornog koda, kao što su uređivač koda, pretraživač projekata, prikaz problema nastalih u toku prevođenja itd. Kada je pokrenut proces kontrolisanog izvršavanja programa, CLIDE automatski prelazi na perspektivu za kontrolisano izvršavanje programa (*eng. Debug perspective*). Meni, trake sa alatima, raspored aktivnih prikaza i alata u okviru ove perspektive prilagođeni su procesu kontrolisanog izvršavanja programa. Korisnik može ručno da prelazi iz jedne perspektive u drugu. Sve izmene nad perspektivom ostaju sačuvane, i vezane su za radni prostor. Vraćanje neke od ugrađenih perspektiva na podrazumevani izgled vrši se naredbom *Window->Reset Perspective*. Nove perspektive se prave tako što se trenutna perspektiva sačuva pod drugim imenom (*Window->Save Perspective As...*).



Slika 1.3 - Otvaranje prikaza

Promena perspektive može biti urađena korišćenjem dugmadi koja su prikazana crvenom bojom na sledećoj slici (Slika 1.4). Perspektiva može biti zamenjena i kroz *Windows* meni, korišćenjem sledećih naredbi *Windows -> Open perspective*.



Slika 1.4 - Prikaz perspektiva

1.2.4 Projekti

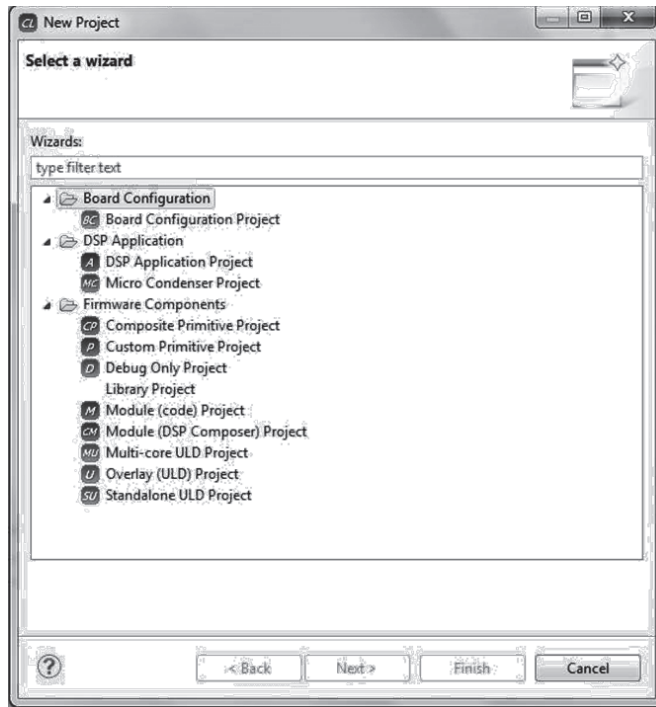
Datoteke unutar CLIDE radnog prostora organizovane su po projektima. Svaki projekat predstavlja skup direktorijuma i datoteka. Za pregled projekata definisanih u radnom prostoru koristi se *Project Explorer* prikaz. Datoteke unutar projekta mapirane su na direktorijume i datoteke na disku. Svaka izmena, dodavanje ili brisanje datoteka unutar prikaza projekta u okviru *Project Explorer View* prozora, rezultiraće istom izmenom nad datotekama na disku.

1.2.4.1 Tipovi projekata

Prilikom pravljenja novog projekta neophodno je odabrati odgovarajući tip projekta. Tipovi projekata odgovaraju pojedinim delovima *Cirrus Logic* programskog okruženja. Detaljan opis programskog okruženja dat je u poglavlju 7.

1.2.4.2 Pravljenje novog projekta

Pravljenje novog projekta vrši se odabirom opcije *File -> New -> Project...* ili aktiviranjem padajućeg menija desnim tasterom miša unutar *Project Explorer View*-a, pa na *New -> Project...*



Slika 1.5 - Čarobnjak za pravljenje novog projekta

Nakon ovog koraka otvara se čarobnjak za pravljenje novog projekta. Prikaz čarobnjaka za pravljenje novog projekta dat je na slici 1.5. Nakon pravljenja projekta, projekat sa datim imenom će biti dodat u *Project Explorer View* i biće otvoren odgovarajući uređivač za konkretan tip projekta.

U CLIDE razvojnom okruženju moguće je napraviti sledeće tipove projekata:

- **Board Configuration Project** - Ovaj tip projekta omogućava konfigurisanje razvojnih ploča kompanije *Cirrus Logic*. Projekat sadrži odgovarajuće skripte koje koristi CLIDE razvojno okruženje. Ovaj tip projekta može biti proširen tako da se omogući konfigurisanje neke druge razvojne ploče koju korisnik koristi. Samo jedna instanca ovog projekta može postojati u radnom prostoru.
- **Library Project** - Ovaj tip projekta se koristi za grupisanje pojedinih delova izvornog koda koji treba da bude na raspolaganje drugim projektima. U okviru ovog projekta se grupiše izvorni kod pisan u asemblerskom jeziku ili programskom jeziku C, ali se drugim projektima ili drugim korisnicima

prosleđuju, tj. daju na korišćenje objektna datoteke koje se dobijaju nakon prevođenja.

- **Debug Only Project** - Ovaj tip projekta omogućava kontrolisano izvršavanje programa nakon pravljenja datoteke sa mašinskim instrukcijama (.uld) i odgovarajućih informacija o kontrolisanom izvršavanju programa (.dw2).
- **Custom Primitive Project** - Ovaj tip projekta se koristi za pravljenje jednostavnih blokova obrade audio signala. Ovako napravljeni blokovi obrade se mogu koristiti za grafičko programiranje DSP aplikacija.
- **Composite Primitive Project** - Ovaj projekat omogućava pravljenje blokova obrade audio signala koji se sastoje od nekoliko blokova obrade audio signala.
- **Module (code) Project** - Ovaj projekat omogućava pravljenje osnovnih modula koji služe za obradu signala. Projekti ovog tipa sadrže datoteke sa izvornim kodom pisanim u asemblerskom jeziku ili programskom jeziku C.
- **Module (DSP Composer) Project** - Ovaj projekat omogućava module za obradu audio signala korišćenjem grafičkog programiranja. Spajanjem jednostavnijih blokova obrade (*primitiva*) moguće je definisati transformacije nad ulaznim odbircima.
- **Overlay (ULD) Project** - Ovaj tip projekta omogućava formiranje određenog nivoa programske podrške. Rezultat prevođenja i povezivanja ovog projekta je datoteka sa kodovima mašinskih instrukcija (.uld) koja se može izvršavati od strane operativnog sistema. Projekat se sastoji od jednog ili više modula, modula zasnovanih na kodu (*Module (code) Project*) i modula zasnovanih na primitivama (*Module (DSPComposer) Project*).
- **Standalone (ULD) Project** - Ovaj tip projekata omogućava formiranje .uld datoteka koje će se izvršavati samostalno, bez *Cirrus OS-a*.
- **Multi-core ULD Project** - Ovaj tip se koristi za generisanje .uld datoteka koje će biti smeštene na više jezgara. To praktično znači kombinovanje više .uld datoteka generisanih u okviru *Overlay* projekta.
- **DSP Application** - Ovaj projekat omogućava pravljenje izvršnog okruženja koje se sastoje od jednog ili više nivoa programske podrške (jednog ili više *Overlay (ULD)* projekata).
- **Micro-condenser Project** - Ovaj tip projekta omogućava specifikaciju i pravljenje datoteka koje će biti smeštene u fleš memoriju. Prilikom prevođenja projekta dobija se *flash image* datoteka koja sadrži odgovarajuću programsku podršku kao i datoteke za konfiguraciju programske podrške.

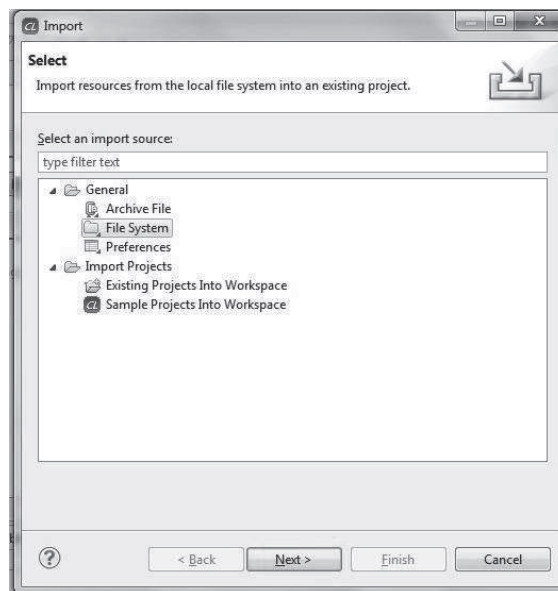
Zadatak:

- Napraviti novi *Standalone (ULD)* projekat sa sledećim podešavanjima:
 - Naziv projekta: *standalone*
 - Ciljna platforma: *Cirrus DSP Core*

Nakon uspešnog pravljenja projekta otvoriće se kartica *Standalone Uld editor* koja može da se ostavi otvorena, jer će biti potrebna u nastavku rada. Ukoliko se zatvori, može se ponovo pokrenuti iz *Project Explorer* prikaza otvaranjem datoteke pod nazivom *.suldproj*.

1.2.4.3 Dodavanje datoteke u projekat

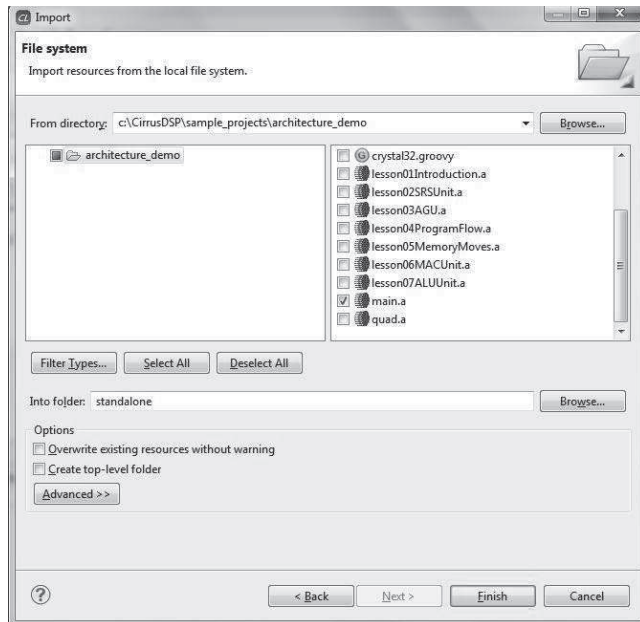
Dodavanje nove datoteke u projekat vrši se odabirom opcije *File ->New ->Other ->General ->File* ili aktiviranjem padajućeg menija desnim tasterom miša na ikonicu sa nazivom projekta u *Project Explorer View*-a, pa *New -> File*. Za razliku od *Visual Studio* razvojnog okruženja, u *Project Explorer View*-u su datoteke u okviru nekog projekta dodate na odabrano mesto. Ne postoji ugrađen mehanizam filtriranja datoteke u odnosu na ekstenziju datoteke.



Slika 1.6 – Dodavanje nove datoteke u projekat

Dodavanje postojećih datoteka u projekat je moguće uraditi aktiviranjem padajućeg menija desnim tasterom miša na ikonicu sa nazivom projekta u *Project Explorer View*-u, pa na *Import -> File system*, kao što je prikazano na slici 1.6.

Po izboru opcije *File system* neophodno je navesti putanju do direktorijuma sa datotekama koje želite da dodate u projekat. Prikaz menija za dodavanje datoteka dat je na slici 1.7.



Slika 1.7 - Izbor datoteka za dodavanje u projekat

Zadatak:

- U prethodno napravljen *Standalone* (ULD) projekat potrebno je dodati novu datoteku sa izvornim kodom unutar *src* foldera. Datoteka treba da ima naziv *main.c*. U okviru *main.c* datoteke potrebno je napisati kratak C program koji ispisuje pozdravnu poruku "Hello World!" u konzolu.

```
#include <stdio.h>

void main() {

    printf("Hello World\n");

    return;

}
```

1.2.5 Prevođenje

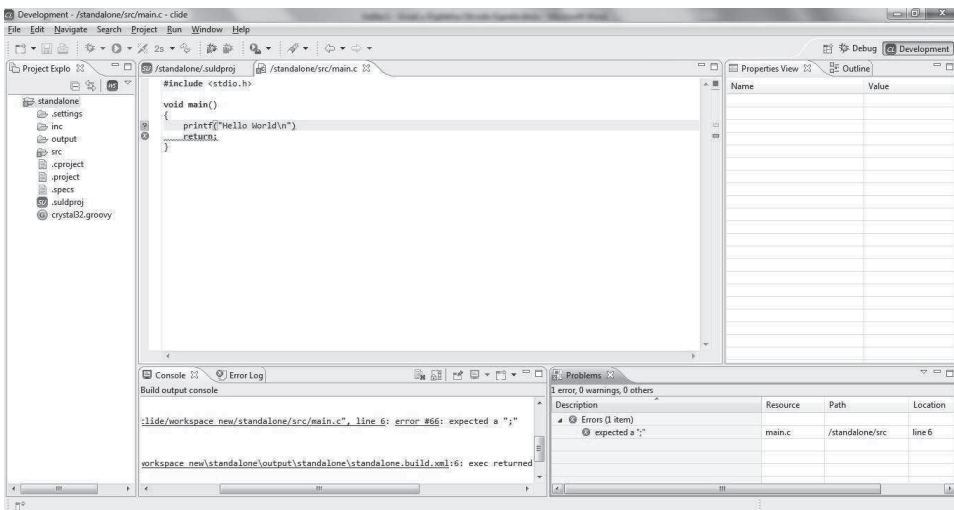
Prevođenje programa vrši se odabirom opcije *Project->Build Project* ukoliko se želi da se prevede jedan projekat, ili *Project->Build All* ukoliko se želi da se pokrene prevođenje svih projekata unutar radnog prostora. CLIDE vrši inkrementalno prevođenje, što znači da se ponovo prevode samo one datoteke koje su menjane nakon prethodnog prevođenja uz one koje zavise od tih datoteka. Ponekad je neophodno prevesti čitav projekat ispočetka, da bi bile uračunate sve izmene u poslednjim prevedenim objektnim datotekama. Za to se koristi opcija *Project->Clean*, koja briše sve datoteke nastale u toku prevođenja jednog projekta.

Nakon pokretanja prevođenja, tok prevođenja (izvršene komande i ispis njihovih rezultata) se može videti u konzoli (*Console View*). Sve greške ili upozorenja, prijavljena u toku prevođenja od strane nekog od alata, biće izlistana u okviru *Problems* prikaza.

Dvostrukim klikom na grešku ili upozorenje otvara se odgovarajuća datoteka sa kodom na mestu na koje se greška ili upozorenje odnosi.

Zadatak:

- Pokrenuti prevođenje projekta.
- Dodati sintaksnu grešku u vaš izvorni kod. Pokrenuti prevođenje. Uočiti prijavljenu grešku.
- Ispraviti prijavljenu grešku, pokrenuti prevođenje.



Slika 1.8– Prijavljivanje grešaka i problema tokom procesa prevođenja

1.2.6 Kontrolisano izvršavanje programa

Kontrolisano izvršavanje programa (eng. *Debuging*) predstavlja tehniku koja omogućava korisniku detaljnu analizu izvršenja programa sa ciljem otklanjanja grešaka ili poboljšanja performansi aplikacije. U toku kontrolisanog izvršavanja korisnik je u mogućnosti da zaustavi procesor, da izvršava kod instrukciju po instrukciju, da se zaustavlja u željenim trenucima ili pregleda trenutni sadržaj memorije ili procesorskih registara. U daljem tekstu opisani su alati za kontrolisano izvršavanje programa koje nudi razvojno okruženje CLIDE.

1.2.6.1 Tačke prekida

Tačka prekida (eng. *breakpoint*) služi za zaustavljanje izvršenja programa na liniji na kojoj su postavljene, odnosno pre izvršavanja prve instrukcije definisane tom linijom. Tačke prekida se mogu, kako pre pokretanja, tako i u toku izvršavanja programa, dodavati, brisati ili privremeno deaktivirati. Dodavanje nove tačke prekida je moguće izvršiti na više načina: dvostrukim levim klikom u prikazu izvornog koda, sa leve strane linije na koju želimo da dodamo tačku prekida, desnim klikom na isto mesto i odabirom opcije *Toggle Breakpoint* iz padajućeg menija ili postavljanjem kursora na željenu liniju i pritiskom na kombinacije tastera *Ctrl+Shift+B*. Da je tačka prekida dodata potvrđuje crveni kružić u vertikalnoj liniji, koji se nalazi na svakoj liniji za koju je postavljena tačka prekida.

Nakon dodavanja tačke prekida, program pokrenut sa ciljem kontrolisanog izvršavanja zaustavlja se na željenoj liniji, odnosno, pre izvršenja instrukcija koje ona definiše.



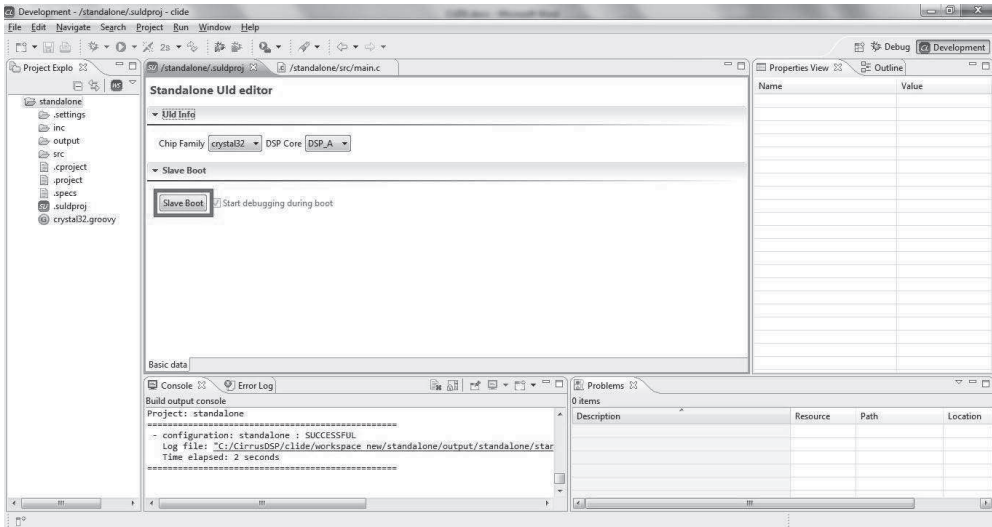
Slika 1.9 - Postavljanje tačke prekida

1.2.6.2 Pokretanje programa u simulatoru

Pokretanje kontrolisanog izvršavanja programa se vrši klikom na *Slave Boot* dugme u okviru uređivača za *Standalone ULD* projekat.

Arhitekture i algoritmi digitalnih signal procesora

Zbirka zadataka i laboratorijski priručnik



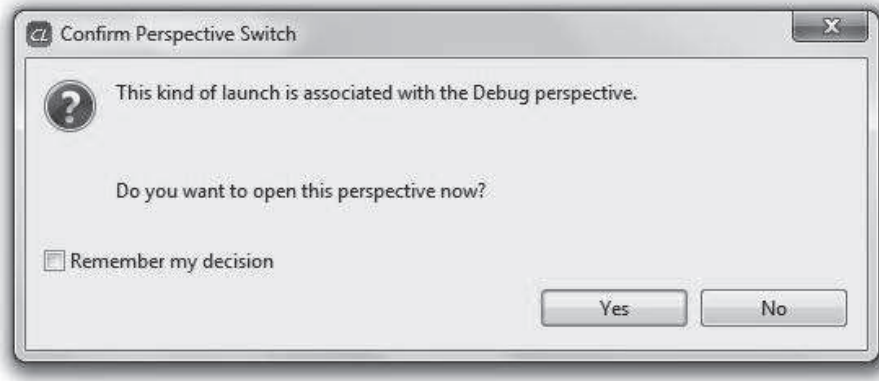
Slika 1.10 - Pokretanje kontrolisanog izvršavanja programa

Klikom na *Slave Boot* dugme vrši se pokretanje i inicijalizacija DSP-a i konfigurisanje razvojne ploče. U slučaju projekata sa cilnom platformom *Cirrus DSP*, vrši se konfigurisanje simulatora i pokretanje simulacije. Za konfiguraciju se koriste *.groovy* skripte. Prilikom *Slave boot* procesa dolazi do izvršavanja *slave_boot* funkcije iz odgovarajuće *.groovy* datoteke. Prilikom formiranja programa, generišu se konfiguracione skripte sa podrazumevanim podešavanjima koje su vezana za konkretan tip projekta i odabrani tip DSP-a. Deo konfiguracione skripte za Standalone (ULD) projekat dat je na sledećoj slici.




Slika 1.11- Konfiguraciona datoteka

Prilikom pokretanja programa pojaviće se dijalog koji omogućava promenu perspektive. Prilikom kontrolisanog izvršavanja programa, podrazumevana perspektiva je *Debug*:



Slika 1.12 - Promena perspektive prilikom pokretanje kontrolisanog izvršavanja programa

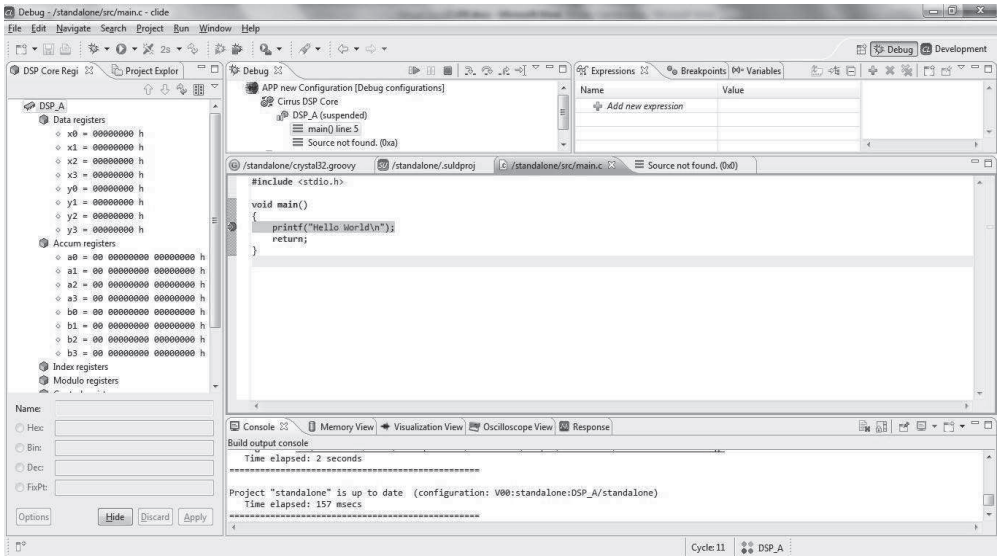
Po pokretanju kontrolisanog izvršavanje programa, dobija se prozor sa tekstem da datoteka sa izvornim kodom nije pronađena. Razlog za to je sam postupak pokretanja C programa. Pre samog poziva *main* funkcije neophodno je uraditi inicijalizaciju pokazivača na *stek* (registar *i7*). Inicijalizaciju obavlja veoma jednostavna biblioteka funkcija koja pored inicijalizacije podataka ima zadatak da pozove i *main* funkciju. Za nastavak izvršenja programa neophodno je kliknuti na dugme *Resume* . Nakon ove akcije, kontrolisano izvršavanje programa bi trebalo da se zaustavi na nekoj od postavljenih tačaka prekida.

Zadatak:

- Staviti tačku prekida pored linije koja vrši ispis pozdravne poruke.
- Pokrenuti kontrolisano izvršavanje programa.
- Zaustaviti kontrolisano izvršavanje programa na prethodno postavljenoj tački prekida.

1.2.6.3 Osnovni alati za kontrolisano izvršavanje programa

Debug perspektiva u okviru *CLIDE* okruženja poseduje podrazumevani skup otvorenih prikaza i njihov raspored. Svaki od prikaza moguće je ukloniti ili promeniti njegovu poziciju. Konfiguracija izgleda perspektive vezana je za radni prostor. Ponovno otvaranje uklonjenih prikaza ili otvaranje novih vrši se iz menija *Window->Show View*.



Slika 1.13 - Debug perspektiva

1.2.6.3.1 Prikaz informacija o kontrolisanom izvršenju

Prikaz informacija o kontrolisanom izvršenju (*Debug View*) sadrži informacije o ciljnoj platformi na kojoj se program izvršava, prikazuje trenutno stanje procesorskih jezgara ciljne platforme, kao i *stek* poziva.

Odabirom funkcije iz *steka* poziva, otvara se izvorni kod te funkcije na datoj lokaciji unutar prozora za prikaz izvornog koda.

1.2.6.3.2 Prikaz izvornog koda

Prikaz izvornog koda (*Source code View*) sadrži prikaz izvornog koda programa koji se trenutno izvršava. Ukoliko je izvršavanje koda na ciljnoj platformi zaustavljeno, plavom strelicom sa leve strane označena je naredna instrukcija koja će biti izvršena.

1.2.6.3.3 Alati za kontrolu toka izvršavanja programa

Traka sa alatima za kontrolu toka izvršavanja programa nalazi se iznad prikaza informacija o kontrolisanom izvršenju.



Slika 1.14 - - Alati za kontrolu kontrolisanog izvršavanja programa

Opcije koje nudi su redom:

- Pokreni/nastavi izvršavanje, zaustavi izvršavanje.
- Prekini sesiju kontrolisanog izvršavanja programa.
- Ulazak u funkciju iz trenutno izvršavane linije (*Step Into*).
- Izvršavanje trenutne linije bez ulaska u funkciju (*Step Over*).
- Izlazak iz funkcije i povratak na mesto poziva funkcije.

Zadatak:

- Dodati novu funkciju *foo()* koja treba da ispiše ime i prezime studenta. Funkcija treba da se pozove iz *main* funkcije nakon poziva funkcije *printf*.
- Postaviti tačku prekida pre poziva funkcije *printf*.
- Koračati kroz program do linije poziva funkcije *foo()*.
- Ući unutar funkcije *foo()*.
- Pokušati ući unutar funkcije *printf*. Šta se dešava? Zbog čega?

1.2.6.3.4 Prikaz promenljivih

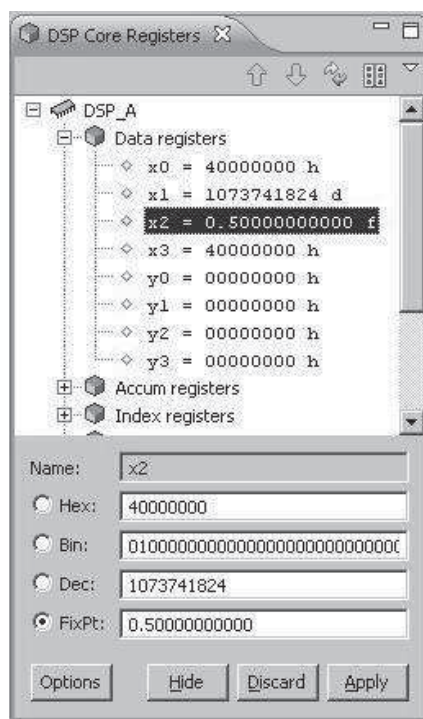
Prikaz promenljivih (*Variables*) sadrži informacije o promenljivim koje su aktivne u toku izvršenja trenutne instrukcije. Prikaz sadrži naziv, tip i trenutnu vrednost promenljive.

Zadatak:

- Ukoliko nije otvoren, otvoriti prikaz promenljivih.

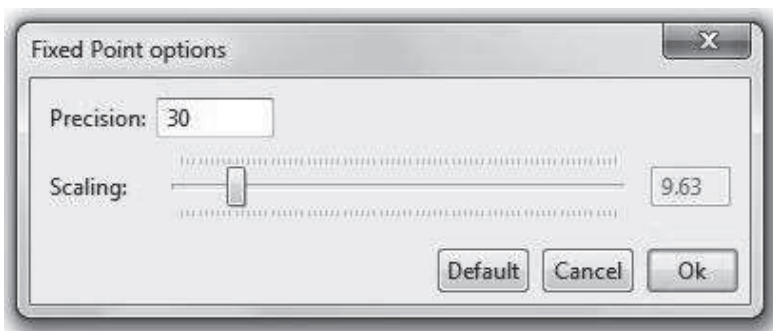
1.2.6.3.5 Prikaz registara

Prikaz registara (*Registers View*) omogućava pregled sadržaja registara procesorskog jezgra i perifernih registara uređaja u toku izvršenja programa (Slika 1.15). Prikaz je organizovan po grupama registara. Vrednosti registara je moguće prikazati u četiri formata: heksadecimalni, binarni, decimalni i format nepokretnog zareza. Format prikaza registra se bira iz kontekstnog menija ili iz donjeg dela prozora koji služi za promenu vrednosti registra. Vrednost se može menjati u sva četiri formata.



Slika 1.15- Prikaz registara

Format nepokretnog zareza se može podešavati. Dijalog za podešavanje prikaza i unosa vrednosti u nepokretnom zarezu se može otvoriti iz kontekstnog menija ili klikom na dugme *Options* iz *Register View*. Dijalog za izbor formata nepokretnog zareza je prikazan na sledećoj slici:



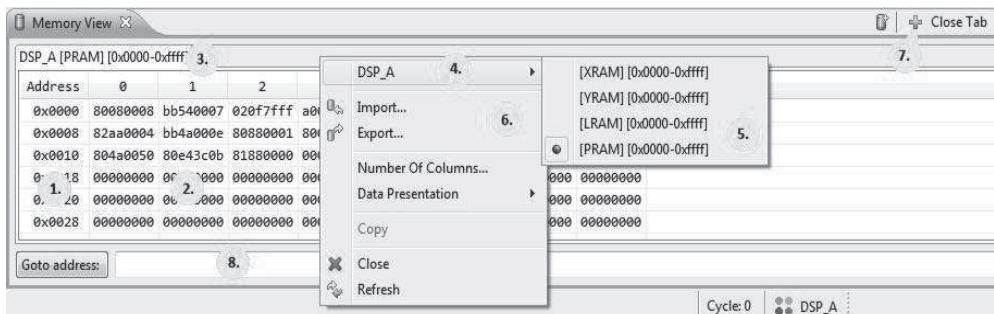
Slika 1.16 -- Dijalog za podešavanje formata prikaza i unosa vrednosti u nepokretnom zarezu

Zadatak:

- Pokrenuti kontrolisano izvršavanje prethodno napravljenog *Standalone (ULD)* projekta.
- Dodati tačku prekida.
- Nakon zaustavljanja komponente za kontrolisano izvršavanje programa na dodatoj tački prekida otvoriti *Register View* i ručno izmeniti vrednost registra b0 na 0.5 u polju *FixPt*.
- Promeniti format prikaza brojeva sa nepokretnim zarezom.
- Proveriti da li prikaz vrednosti u registru b0 odgovara prethodno promenjenom formatu prikaza brojeva sa nepokretnim zarezom.

1.2.6.3.6 Prikaz memorije

Prikaz memorije (*Memory View*) nije otvoren u podrazumevanim podešavanjima *Debug* perspektive. Otvaranje ovog prikaza vrši se odabirom opcije *View --> Memory View*. Prikaz memorije dat je na sledećoj slici.



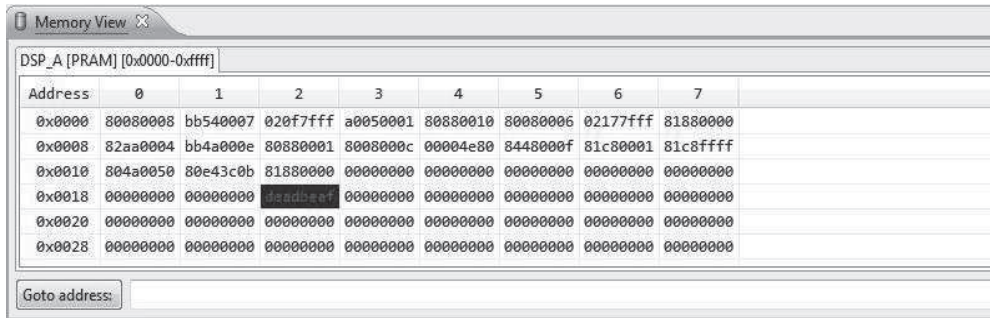
Slika 1.17 - Prikaz memorije

Memory view prikazuje sadržaj memorije u obliku tabele. Polja u prvoj koloni (1.) sadrže adresu prve memorijske lokacije u redu. Polja u ostalim kolonama (2.) pokazuju vrednosti memorijskih lokacija.

Vrednost se može menjati dvostrukim klikom na polje u tabeli. *Memory view* može sadržati više tabela. One se biraju pomoću kartica (3.), a dodaju se dugmetom plus (7.). Svaka kartica ima kontekstni meni. Iz ovog menija se bira koju memorijsku zonu (5.) i sa kojeg jezgra (4.) prikazuje tabela. Pomoću opcija *Import* i *Export* (6) se može zapisivati sadržaj memorije u datoteku i obrnuto. Tekstualno polje na dnu ovog prozora omogućava izbor adrese ili labela na koju se pozicionira tabela (8).

Neposredna izmena memorijskih lokacije je moguća kroz *Memory View*. Očekivani format unosa je heksadecimalan. Po završetku unosa nove vrednosti u neku

memorijsku lokaciju, izmenjena lokacija će biti označena u okviru *Memory View* do sledećeg osvežavanja prikaza memorije. Prikaz izmenjene memorije je dat na slici 1.18.



Slika 1.18- Prikaz izmenjenog sadržaja memorijske lokacije

Zadatak:

- Pokrenuti kontrolisano izvršavanje prethodno napravljenog *Standalone (ULD)* projekta.
- Dodati tačku prekida.
- Otvoriti *Memory View* i ručno promenite vrednost neke memorijske lokacije.
- Neposredno uneti adresu memorijske lokacije koja se želi pogledati (npr. memorijska zona X, adresa 0x120).
- Odabrati neku labelu koja je automatski dodata u *Memory View* prilikom pokretanja kontrolisanog izvršavanja programa.

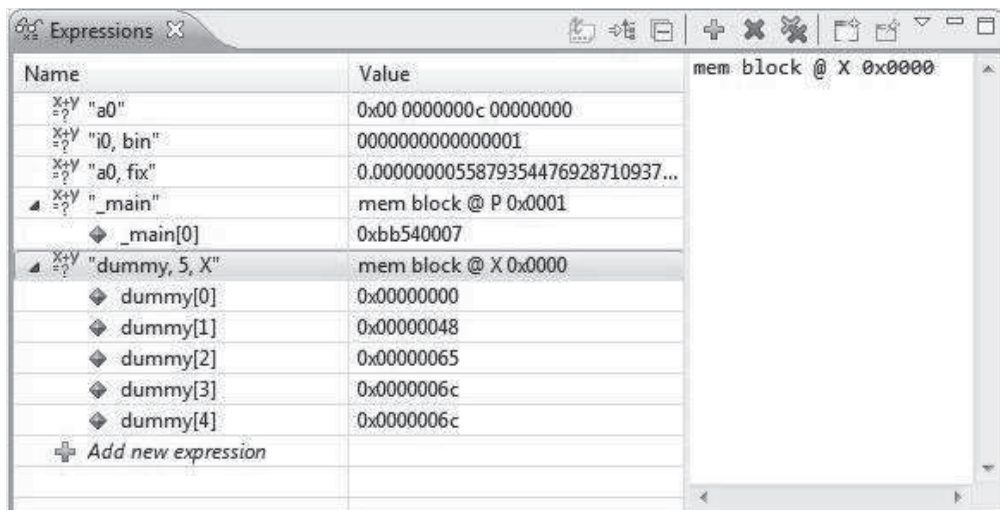
1.2.6.3.7 Prikaz vrednosti izraza

Pomoću prozora *Expressions* je moguće prikazati vrednosti izraza kako u asemblerskom tako i u C programskom jeziku. Izrazi mogu sadržati C promenljive, asemblerske labela i adrese. Ukoliko se koriste adrese, one se moraju kombinovati sa oznakom memorijske zone pomoću operatora zareza.

U izrazima su podržani sledeći operatori:

- Operator indeksiranja niza `[]` može se primeniti nad C nizovima, asemblerskim labelama i adresama.
- Operatori pristupa poljima struktura `.` i `->` mogu se primeniti samo nad C strukturama.
- Operator pokazivača `*` može se primeniti samo nad pokazivačima.

- Adresni operator & može se primeniti na izrazima čija se vrednost nalazi u memoriji.
- Specijalni operator , se može koristiti u nekoliko svrha.



Slika 1.19 – Prikaz vrednosti izraza

Pomoću zareza se može promeniti memorijska zona, promeniti format prikaza brojeva i odrediti koliko elemenata treba prikazati. Memorijsku zonu je moguće promeniti bilo kom asemblerskom izrazu, ili adresi. Ovo se radi tako što sa leve strane zareza stoji izraz, a sa desne oznaka memorijske zone. Oznake memorijske zone mogu biti:

- X – X zona podataka
- Y – Y zona podataka
- L – L ili XY Zona podataka
- P - zona programskog koda

Format brojeva se može menjati svim izrazima tako što se sa leve strane zareza navede izraz, a sa desne jedna od sledećih oznaka formata broja:

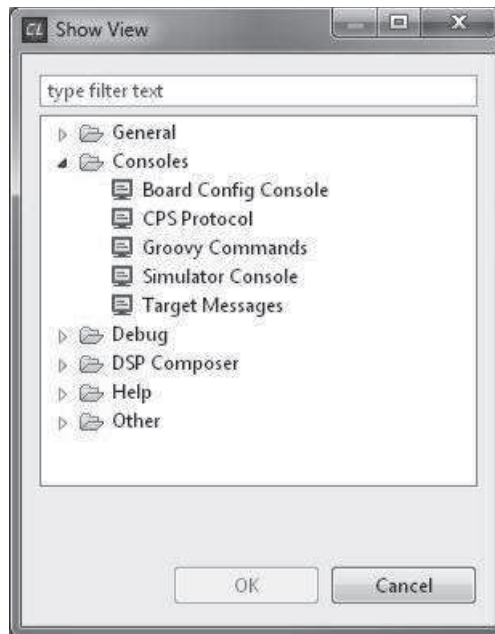
- DEC – decimalni
- BIN – binarni
- HEX – heksadecimalni
- FIX – Format broja u nepokretnom zarezu

Broj elemenata se može primeniti na nizove, asemblerske labela i adrese. Broj elemenata (lokacija) se navodi sa desne strane zareza.

Primer korišćenja operatora zareza je dat na slici 1.19.

1.2.6.3.8 Prikaz konzola

U razvojnom okruženju *CLIDE* postoji nekoliko tipova konzola koje omogućavaju prikaz različitih informacije vezanih za razvojno okruženje ili za razvojnu ploču. Otvaranje konzola može biti urađeno kroz *Window->Show Views->Other ... ->Consoles*. Prikaz dodatnih *CLIDE* konzola je dat na slici 1.20. Poruke poslate na standardni izlaz (stdout) ili standardan izlaz za greške (stderr) će biti prikazane u okviru Simulator Console prozora.



Slika 1.20 - Odabir konzole

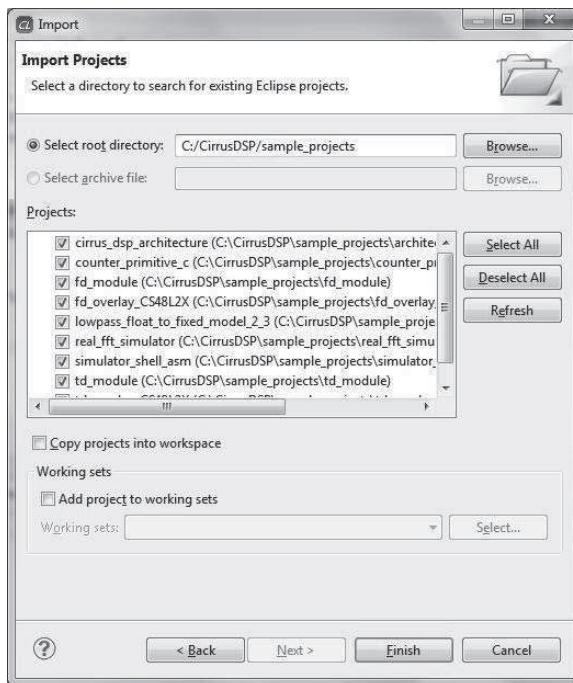
1.2.7 Uvlačenje postojećih projekata u radni prostor

Ukoliko na disku postoji već napravljen projekat, da bi se omogućio rad nad njim potrebno je da pomenuti projekat postane deo aktivnog radnog prostora. Uvlačenje već postojećih projekata u radni prostor vrši se komandom *File -> Import -> Import Projects -> Existing Projects into Workspace*.

U otvorenom prozoru potrebno je izabrati putanju do direktorijuma ili arhive koja sadrži željeni projekat. Putanja se može uneti ručno, ili pretražiti pritiskom na

dugme *Browse*. Nakon odabira putanje, u okviru prozora su izlistani svi CLIDE projekti na datoj putanji. Potrebno je obeležiti jedan ili više projekata i pritisnuti dugme *Finish*.

Dodatne opcije koju čarobnjak pruža jeste kopiranje projekta u radni prostor. Ova opcija se odnosi na uključivanje projekata koji se ne nalaze u direktorijumu koji predstavlja aktivni radni prostor. Ukoliko se ova opcija izabere, direktorijum koji sadrži projekat, sa svim datotekama biće prekopiran u direktorijum radnog prostora. Sve izmene nad projektom odnosiće se na prekopirane datoteke. U suprotnom slučaju, datoteke će biti uvučene u radni prostor, ali će se na disku nalaziti na drugoj lokaciji.



Slika 1.21 - Uključivanje projekata u radni prostor

Zadatak:

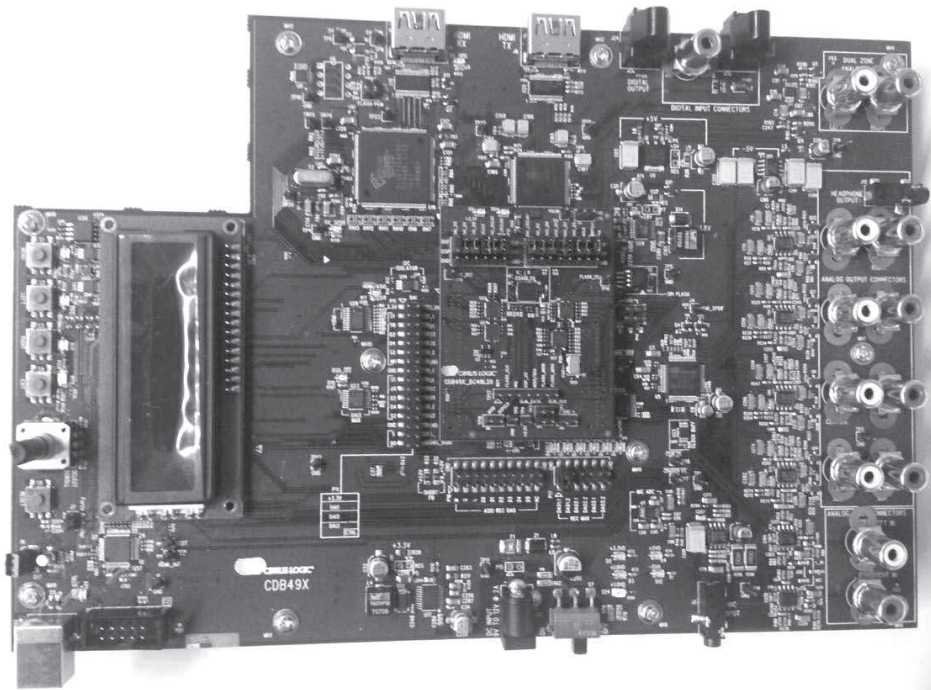
- Uključiti projekat *cirrus_dsp_architecture* u aktivni radni prostor. Projekat se nalazi na putanji *C:\CirrusDSP\sample_projects*.
- Uključiti opciju kopiranja projekta u direktorijum radnog prostora.
- Pokrenuti prevođenje uključenog projekta.

1.2.8 Pokretanje programa na razvojnoj ploči

CLIDE razvojno okruženje omogućava učitavanje prevedenog programa u memoriju DSP uređaja i njegovo izvršavanje. U toku izvršavanja datog programa *CLIDE* omogućava kontrolisanje toka izvršavanja, kao i dobavljanje informacija o stanju uređaja u toku izvršavanja.

1.2.8.1 Povezivanje razvojne ploče sa računarom

Prikaz razvojne ploče dat je na slici 1.22.



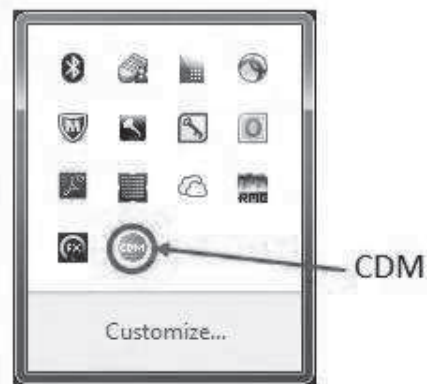
Slika 1.22 - CDB49x-DC48L20 razvojna ploča

Detaljan opis pojedinih komponenti na samoj razvojnoj ploči dat je u dokumentu *CDB49x_DC48L20 Board Manual* koji je smešten na sledećoj lokaciji *C:\CirrusDSP\CS48L2X\doc\CDB49x_DC48L20 Board Manual.pdf*.

Zadatak:

- Otvoriti prethodno navedeni dokument.
- Pronaći u dokumentu gde se nalazi ulaz za napajanje, ulaz za mikrofona, USB ulaz, izlaz za slušalice.
- Pronaći odgovarajuće komponente na samoj razvojnoj ploči.

Sprega između razvojnog okruženja ili drugih alata i ciljne platforme realizovana je komponentom koja se naziva *Cirrus Proxy Server (CPS)*. Detekcija razvojnih ploča urađena je kroz klijentsku aplikaciju koja se naziva *Cirrus Device Manager (CDM)*. Pre samog povezivanja razvojne ploče i računara neophodno je proveriti da li je *CDM* već pokrenut. Indikacija da li je *CDM* aplikacija pokrenuta, prikazana je na slici 1.23.



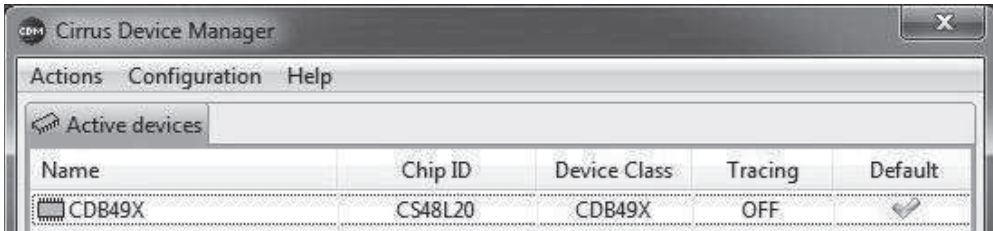
Slika 1.23 –CDM, bez detektovanih uređaja

Ukoliko *CDM* nije pokrenut, pokretanje se može izvršiti na sledeći način: *Start->All Programs->Cirrus Logic DSP->Cirrus Device Manager*. Nakon priključenja razvojne ploče potrebno je proveriti da li je razvojna ploča upaljena. Detektovanje razvojne ploče je automatsko.



Slika 1.24 - CDM, sa detektovanim uređajem

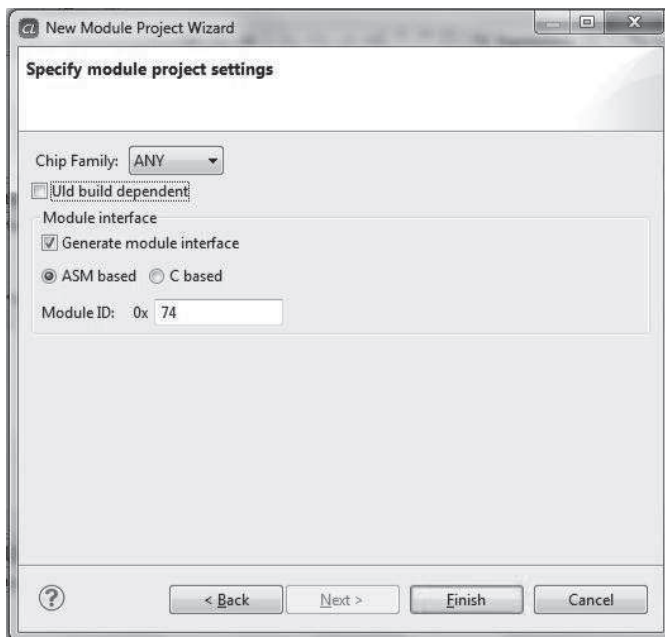
Otvaranje CDM aplikacije je moguće uraditi duplim klikom na ikonicu CDM ili desnim tasterom na ikonicu i izborom opcije *Open CDM*. Prikaz CDM aplikacije dat je na slici 1.25.



Slika 1.25 - CDM

1.2.8.2 Pravljenje passthrough projekta

Pravljenje projekata u razvojnom okruženju CLIDE je zasnovano na korišćenju odgovarajućih čarobnjaka. Sam postupak pravljenja i opis pojedinih tipova projekata dat je u prethodnom tekstu.



Slika 1.26 - Čarobnjak za pravljenje module (code) projekta

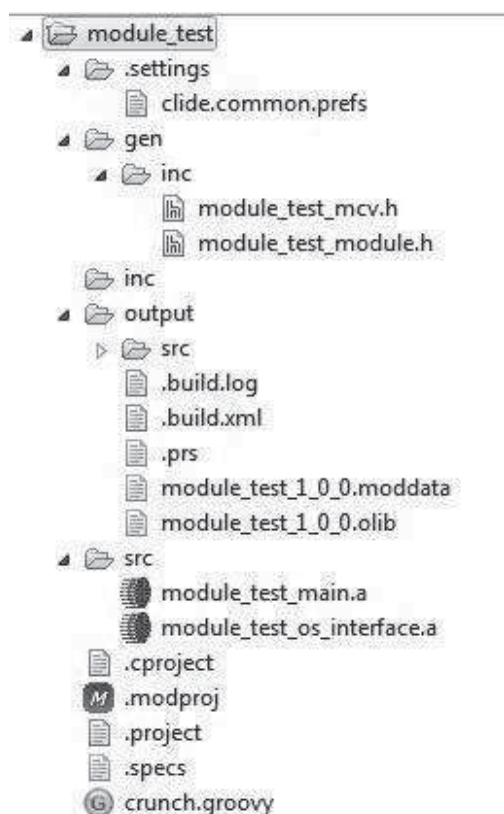
1.2.8.2.1 Pravljenje Module (code) projekta

Prilikom formiranja ovog tipa projekta neophodno je navesti ID modula kao i vrstu familije za koju je taj modul namenjen. Ukoliko konkretan modul ne koristi

specifičnosti određene familije DSP procesora, onda treba odabrati ANY prilikom izbora familija čipa. Deo čarobnjaka za *Module(Code)* projekta je prikazan na slici 1.26.

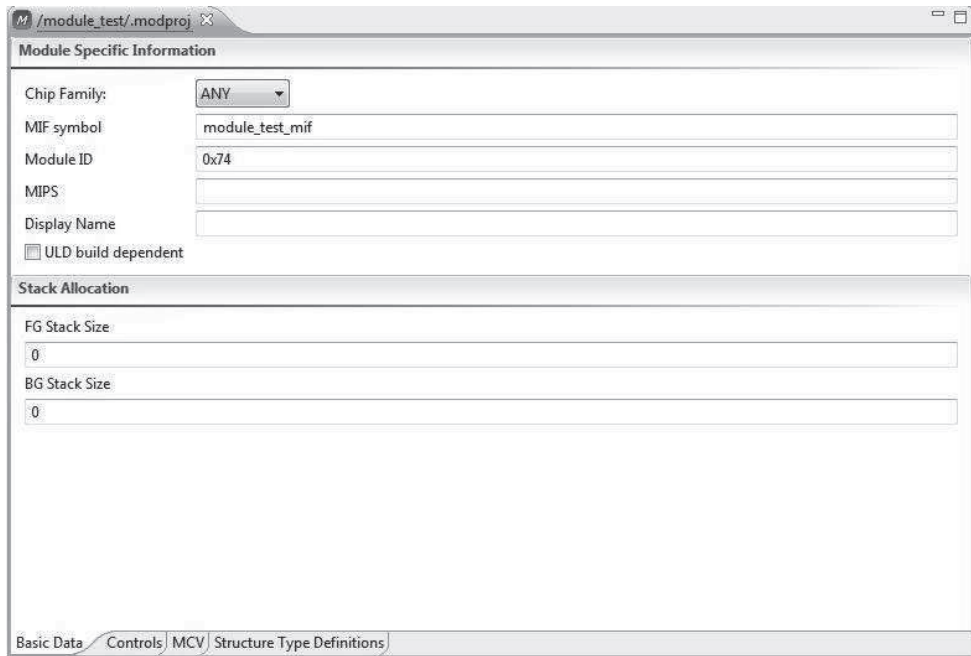
Prilikom pravljenja novog modula moguće je generisanje odgovarajućih funkcija koje povezuju modul sa ostatkom programskog okruženja (*framework-a*). Šabloni sa mestima za korisničke funkcije mogu biti generisani u C-u ili u asemblerskom jeziku.

Klikom na dugme *Finish* vrši se pravljenje novog projekta. Struktura generisanih direktorijuma prilikom pravljenja novog *Module (code)* projekta data je na slici 1.27.



Slika 1.27 - *Module (Code) Project, Project Explorer View*

Nakon pravljenja novog projekta automatski se pokreće proces prevođenja napravljenog projekta i automatsko otvaranje odgovarajućeg uređivača projektne datoteke.



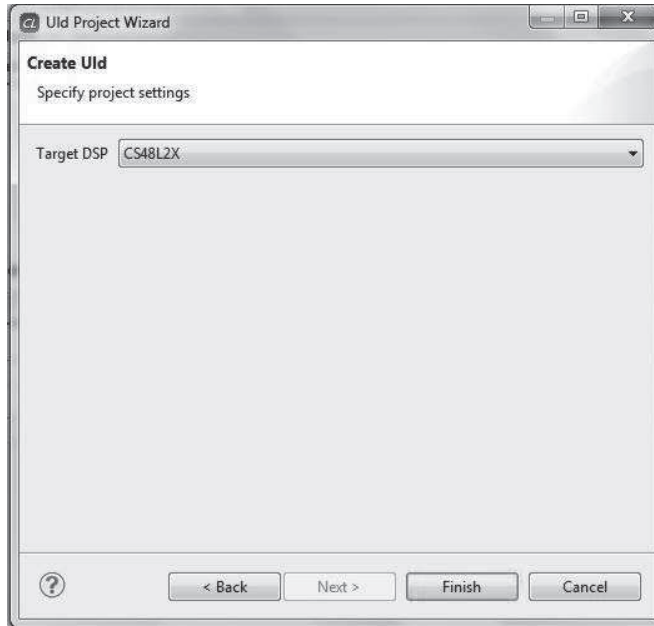
Slika 1.28 – Uređivač Module (code) projekta

1.2.8.3 Pravljenje Overlay (ULD) projekta

Kod namenskih računarskih sistema, celokupan memorijski prostor često je izdijeljen na nekoliko nivoa, odnosno slojeva (eng. *Overlay*). Vrlo često postoje zahtevi za dinamičkom izmenom memorije u zavisnosti od spoljnih uticaja ili aktivnosti korisnika. *Overlay* predstavlja jedan nivo programske obrade. Rezultat prevođenja i povezivanja ovog projekta je datoteka sa kodovima mašinskih instrukcija (.uld) koja se može izvršavati od strane operativnog sistema. Projekat se sastoji od jednog ili više modula. Postoje moduli zasnovani na programskom kodu (*Module (code) Project*) i moduli zasnovani na grafičkim blokovima (*Module (DSPComposer) Project*). Više detalja o organizaciji različitih nivoa obrada dato je u Vežbi 7.

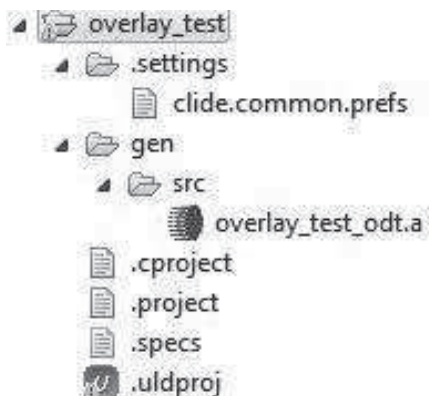
Prilikom pravljenja novog *Overlay (ULD)* projekta neophodno je odabrati familiju čipova. Raspored i imena nivoa obrade mogu biti različiti od jedne do druge familije čipova. Dijalog za izbor familije čipova prilikom pravljenja *Overlay (ULD)* projekta dat je na slici 1.29.

Po zatvaranju čarobnjaka za pravljenje novog projekta, izvršiće se postupak prevođenja i povezivanja projekta u cilju dobijanja datoteke sa mašinskim instrukcijama (.uld datoteka).



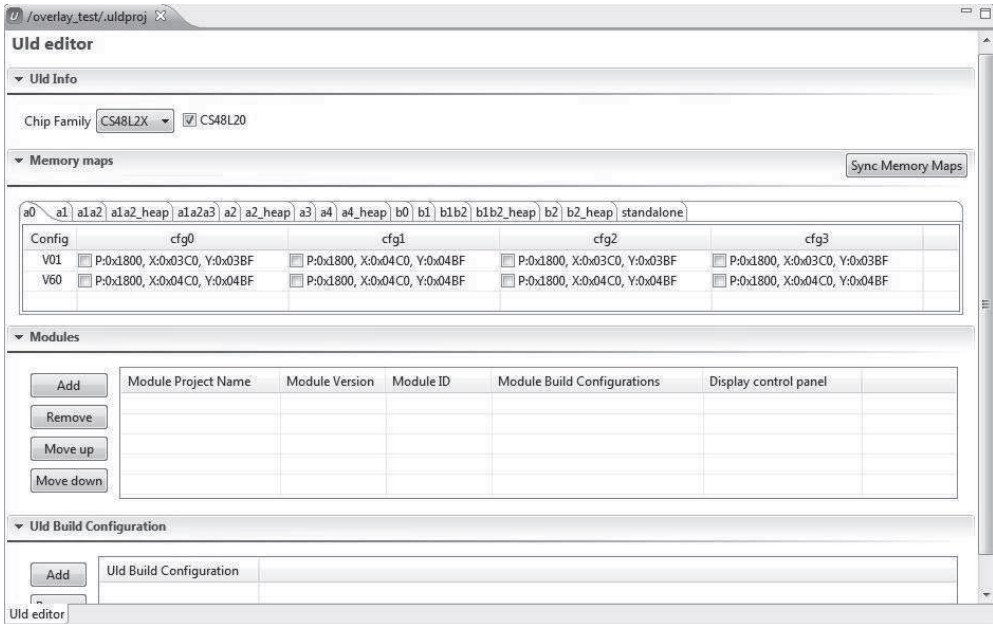
Slika 1.29 – Izbor familije čipova prilikom pravljenja Overlay (ULD) projekta

Struktura napravljenih datoteka u procesu pravljenja projekta data je na slici 1.30.



Slika 1.30 –Struktura Overlay (ULD) projekta

Pored prevođenja i povezivanja projekta automatski se vrši i otvaranje odgovarajućeg uređivača projektnog fajla. Prikaz uređivača za *Overlay (ULD)* projekat dat je na slici 1.31.



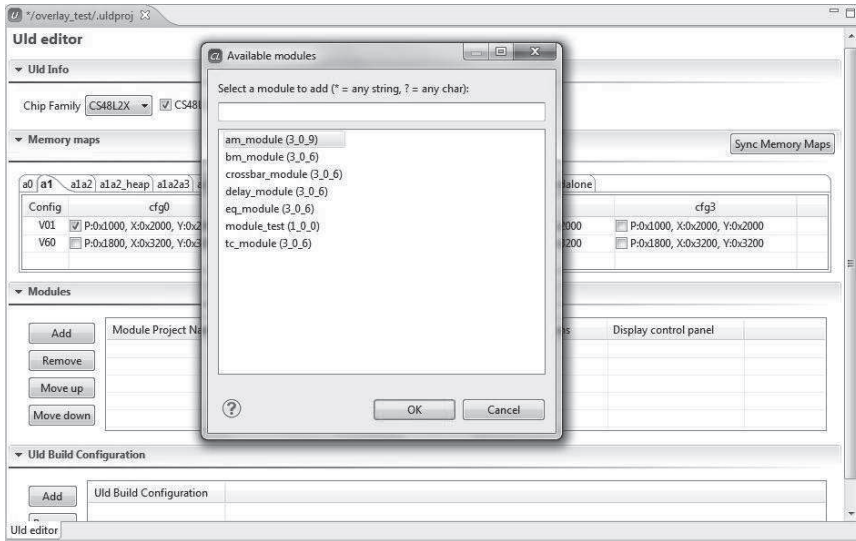
Slika 1.31 – Uređivač Overlay (ULD) projekta

Uređivač je organizovan u nekoliko različitih sekcija:

- *ULD info* – sekcija koja omogućava promenu familije procesora, kao i opciono uključivanje ili isključivanje mogućnosti korišćenja konkretnog *Overlay (ULD)* projekta za pojedine tipove procesa iz iste familije.
- *Memory Maps* – sekcije u kojoj se nalaze informacije o samoj podeli memorije za konkretnu familije procesora. Izbor sloja (nivoa) obrade se vrši u okviru ove sekcije.
- *Modules* – sekcija u kojoj se vrši izbor i pregled korišćenih modula obrade.
- *Uld Build Configuration* – sekcija koja omogućava navođenje specifičnih konfiguracija projekata. Konfiguracija bi se u ovom slučaju odnosila na izbor pojedinih kombinacija modula koji su uključeni u sam *Overlay (ULD)* projekat.

Izbor željenog nivoa programske podrške vrši se klikom na polja u okviru *Memory Maps* sekcije u uređivaču. Dodavanje novih modula vrši se klikom na dugme *Add* u okviru *Modules* sekcije u uređivaču. Prikaz dijaloga za dodavanje novog modula dat je na slici 1.32.

Prilikom dodavanja novog modula u okviru *Overlay (ULD)* projekta, korisnik ima mogućnost da doda neki modul koji se nalazi u korisničkom radnom prostoru, ili da doda već definisan (sistemski) modul.



Slika 1.32 - Dodavanje novog modula u okviru *Overlay (ULD)* projekta

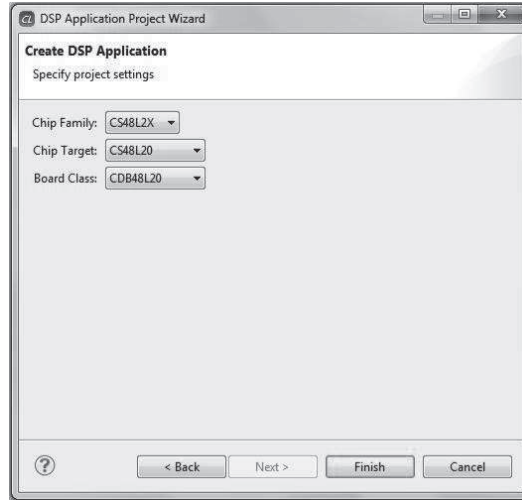
Zadatak:

- Napraviti novi *Overlay (ULD)* projekat i nazvati ga *overlay_test*.
- Dodati prethodno napravljeni modul.
- Odabrati sledeći *overlay* (A1, v01, cfg0).
- Pokrenuti prevođenje projekta.

1.2.8.4 Pravljenje DSP Application projekta

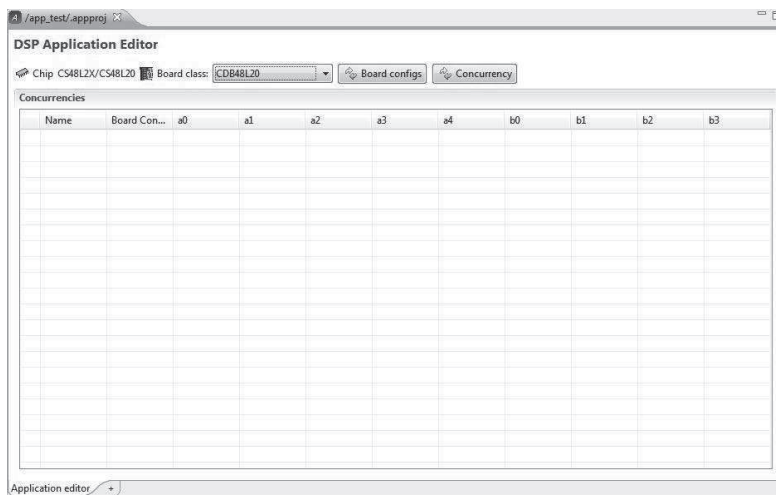
DSP Application projekat omogućava definisanje jednog ili više izvršnih okruženja koja se mogu samostalno pokretati. Izvršno okruženje predstavlja skup slojeva koji će biti u DSP memoriji u isto vreme. Nazivi i veličine memorije potrebne za pojedine slojeve obrade se razlikuju od jedne do druge familije procesora. Iz tog razloga je neophodno napraviti različite *DSP Application* projekte za različite familije procesora, odnosno, konkretne čipove.

Prikaz čarobnjaka za izbor familije DSP procesora, modela DSP procesora i izbor modela razvojne ploče naveden je na slici 1.33.



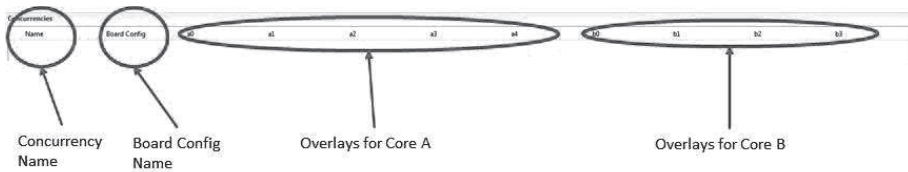
Slika 1.33 Čarobnjak za pravljenje novog DSP Application projekta

Po završetku čarobnjaka za pravljenje novog *DSP Application* projekta otvoriće se odgovarajući uređivač projektne datoteke. Prikaz uređivača za *DSP Application* projekta dat je na slici 1.34.



Slika 1.34 – Uređivač za DSP Application projekat

Najveći deo prostora zauzima tabela u kojoj se definišu izvršna okruženja. Svako izvršno okruženje je određeno imenom. Nije moguće imati više izvršnih okruženja sa identičnim imenom.



Slika 1.35- Struktura izvršnog okruženja za DSP Application projekat

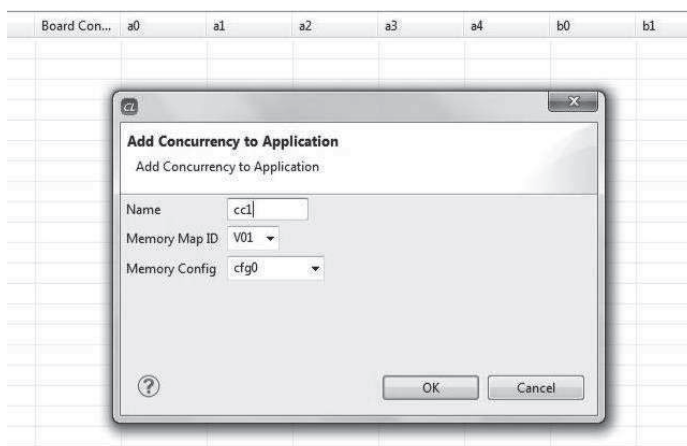
Kolone u okviru ove tabele se mogu podeliti u 4 grupe:

- Ime izvršnog okruženja – ime okruženja i ime odabrane memorijske mape/konfiguracije.
- Naziv konfiguracije ploče – naziv odabrane skriptne datoteke kojom se vrši podešavanje periferija na razvojnoj ploči.
- Slojevi obrade na A jezgru – skup izabranih slojeva obrade.
- Slojevi obrade na B jezgru – skup izabranih slojeva obrade.

Otvaranje dijaloga za dodavanje novog izvršnog okruženja je moguće otvoriti klikom na znak “+” u donjem levom uglu uređivača ili desnim tasterom na neko slobodno polje u okviru tabele.

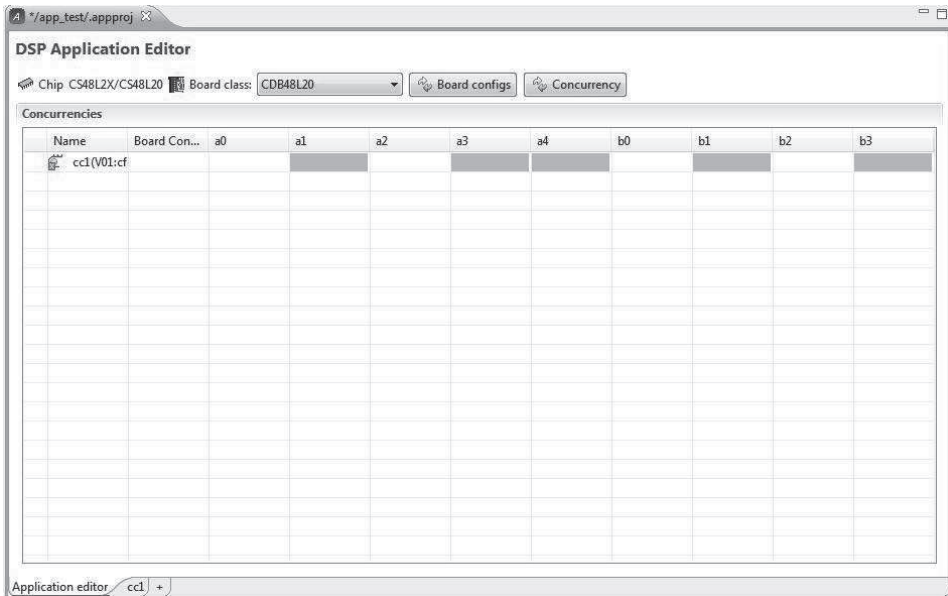
Prilikom definisanja novog izvršnog okruženja neophodno je definisati naziv okruženja. Pored naziva izvršnog okruženja neophodno je odabrati odgovarajuću memorijsku mapu i memorijsku konfiguraciju.

Prikaz dijaloga za dodavanje novog izvršnog okruženja dat je na sledećoj slici:



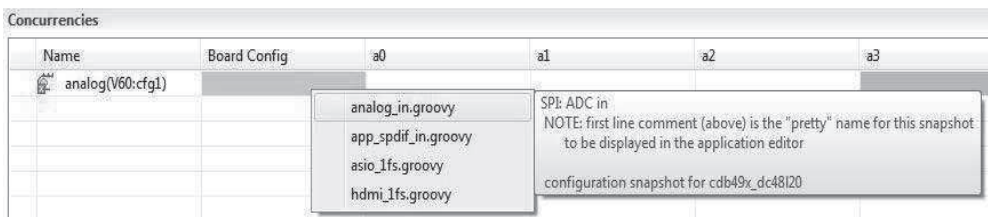
Slika 1.36 - Dodavanje novog izvršnog okruženja

Svako izvršno okruženje je predstavljano jednom vrstom u okviru *Concurrencies* tabele. Siva polja za određeno izvršno okruženje označavaju da ne postoje odgovarajuće .uld fajlovi za pojedine slojeve obrade, kao što je i prikazano na slici broj 1.37.



Slika 1.37 - DSP Application, novo izvršno okruženje

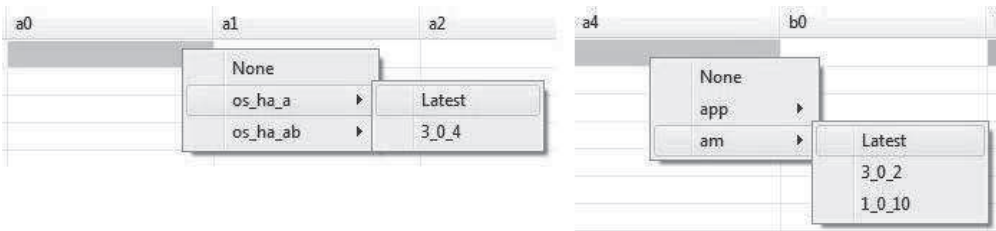
Izbor konfiguracione skript datoteke kojom se vrši podešavanje periferija na razvojnoj ploči, za konkretno izvršno okruženje, vrši se iz odgovarajućeg menija. Meni se aktivira desnim klikom na odgovarajuće polje iz kolone sa nazivom *Board Config*. Prikaz menija za izbor konfiguracione skripte dat je na slici 1.38.



Slika 1.38 - Izbor konfiguracione skripte

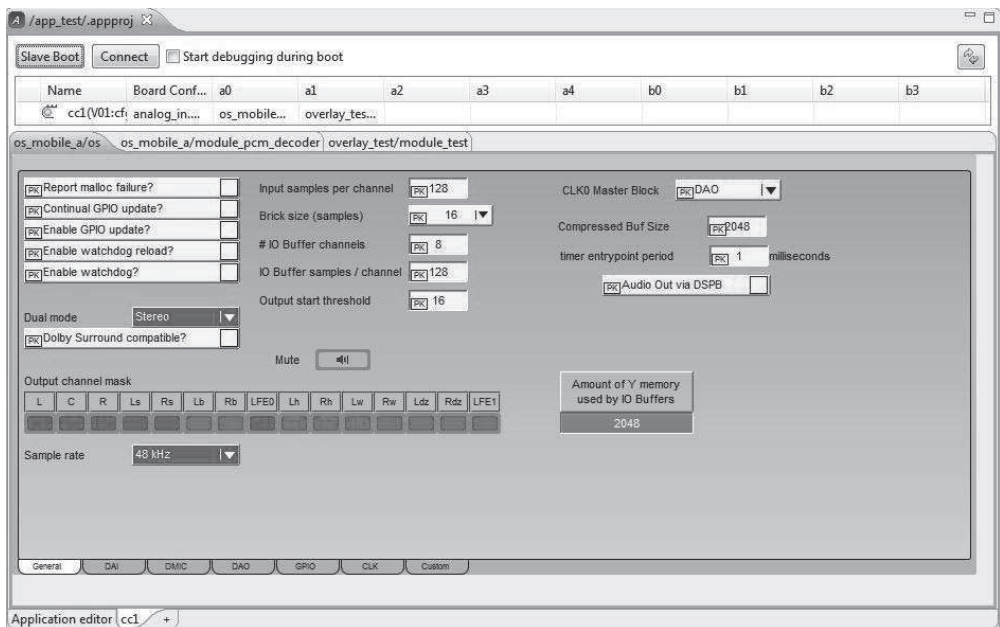
Izbor određenog sloja obrade vrši se kroz meni koji se aktivira desnim klikom na odgovarajuće polje u tabeli. Prilikom izbora sloja obrade moguće je odabrati konkretnu verziju određenog sloja programske podrške. Pored izbora konkretne

verzije programske podrške, postoji i mogućnost izbora najnovije (eng. *latest*) verzije. Prikaz menija i podmenija za izbor sloja obrade dat je na slici 1.39.



Slika 1.39 - Izbor sloja obrade

Za svako napravljeno izvršno okruženje u okviru uređivača se pravi posebna kartica na kojoj se vide kontrole vezane za pojedine module iz tog okruženja. Promenu kontrole je moguće vršiti tokom samog rada aplikacije, tj. izvršavanja na ciljnoj platformi. Pored kontrola, na karticama za izvršna okruženja se nalaze i upravljačka dugmad koja omogućavaju spajanje sa razvojnom pločom ili pokretanje procesa slanja programske podrške na DSP i pokretanje DSP-a. Prikaz kartice za odabrano izvršno okruženje dat je na slici 1.40.



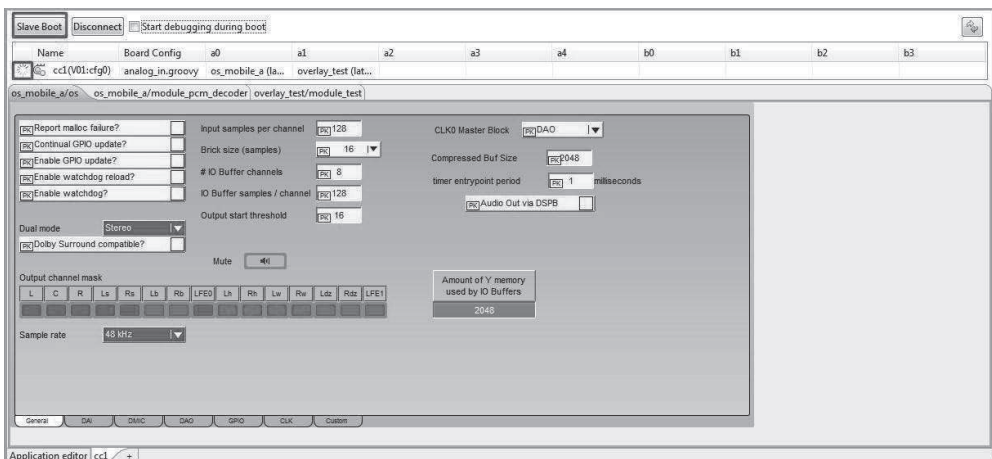
Slika 1.40 - Prikaz kartice za izvršno okruženje cc1

Zadatak:

- Napraviti novi *DSP Application* projekat i dodati novo izvršno okruženje (ime: cc1, memorijska mapa: v01, memorijska konfiguracija: cfg0).
- Odabrati analog.groovy za konfiguracionu skriptu.
- Dodati OS_A *overlay* u A0 polje.
- Dodati prethodno napravljeni *Overlay* (ULD) projekat u A1 polje.

1.2.8.5 Pokretanje DSP aplikacije na razvojnoj ploči

Pokretanje DSP aplikacije se vrši klikom na dugme *Slave Boot*. Navedeno dugme se nalazi na kartici za odabrano izvršno okruženje u okviru uređivača (*DSP Application* projekat). Nakon uspešnog pokretanja programa, indikator koji ukazuje na pokrenutu aplikaciju će biti prikazan pored naziva okruženja. Prikaz kartice sa označenim dugmetom za pokretanje programa i sa indikatorom trenutno pokrenutog izvršnog okruženja dat je na slici 1.41.



Slika 1.41 - Pokretanje projekta *DSP Application*

Zadatak:

- Otvoriti prethodno napravljeni *DSP Application* projekat i pokrenuti izvršno okruženje cc1.