SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5252

RAZVOJ APLIKACIJA DIGITALNE OBRADBE SIGNALA POD OPERACIJSKIM SUSTAVOM ZA RAD U STVARNOM VREMENU

Mateo Lasić

Zagreb, lipanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 9. ožujka 2017.

ZAVRŠNI ZADATAK br. 5252

Pristupnik: **Mateo Lasić (0036487549)**
Studij: Elektrotehnika i informacijska tehnologija
Studij: Elektrotehnika i informacijska tehnologija

Zadatak: **Razvoj aplikacija digitalne obradbe signala pod operacijskim sustavom za rad u stvarnom vremenu**

Opis zadatka:

U okviru završnog rada potrebno je istražiti postupak razvoja aplikacija za DSP procesor prilagođenih za izvođenje u sklopu operacijskog sustava za rad u stvarnom vremenu. Kao konkretnu arhitekturu procesora koristiti cjelobrojni DSP procesor firme Texas Instruments iz porodice TSM320C55X, a kao jezgru operacijskog sustava potrebno je koristiti DSP/Bios jezgru prilagođenu za ovu arhitekturu. Posebnu pažnju posvetiti postupku inicijalizacije sklopovlja cijelog sustava, a kao razvojnu platformu koristiti sustav TMDX5505eZdsp. Istražiti postupke inicijalizacije A/D i D/A pretvornika koji su sastavni dio vanjske Codec komponente TLV320AIC3204 povezane s DSP procesorom serijskim sabirnicama I2C i I2S . Istražiti pogodne načine dohvata ulaznih i izlaznih podataka iz ovog pretvornika uz korištenje DMA sklopa i automatizirani prijenos podataka u memoriju procesora podržanog od strane operacijskog sustava. Analizirati vremenska svojstva aplikacije koja radi pod operacijskim sustavom, s obzirom na kašnjenja koja uvodi operacijski sustav kao i s obzirom na podatkovnu propusnost koja se može ostvariti cijelim sustavom za obradbu signala.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 9. lipnja 2017.

Mentor:
Prof. dr. sc. Davor Petrinović

Djelovođa:
Prof. dr. sc. Dražen Jurišić

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Mladen Vučić

Sadržaj

1. Uvod

Ugradbeni računalni sustavi (embedded systems) u prošlosti su zahtjevali izvođenje jednostavnih upravljačkih programa, dok se danas njihova primjena uveliko proširila i zakomplicirala. Takvi sustavi zahtjevaju obavljanje više aplikacija istovremeno, što je izuzetno kompliciranо ili uopće nije moguće izvesti u jednostavnim sustavima za određene vremenske zahtjeve sustava , te se samim time postavio zahtjev za operacijskim sustavom. Operacijski sustav za rad u stvarnom vremenu (eng. Real Time Operating System, RTOS) predstavlja mikroračunalni operacijski sustav koji upravlja i nadgleda fizičke procese. Rad u stvarnom vremenu za DSP sustav predstavlja vremenski zahtjev, koji kaže da sama obrada signala unutar DSP procesora mora biti manja od perioda uzorkovanja signala. Takav sustav danas predstavlja jedan od glavnih dijelova složenih sustava, ali se ne koristi u jednostavnim aplikacijama, jer bih bez potrebe zakomplicirao razvoj takvog sustava. RTOS sustav nam omogučava relativno jednostavniji razvoj sustava, iskorištavanje maksimalne procesorske snage, te nam je omogučilo vrlo jednostavan prijenos programske aplikacije na drugi sustav, što u običnim sustavima nije moguće. Jedan od takvih sustava razvila je firma Texas instruments za svoje DSP (Digital Signal Processor) porodice (c4000, c5000, c6000) i naziva se DSP/BIOS.

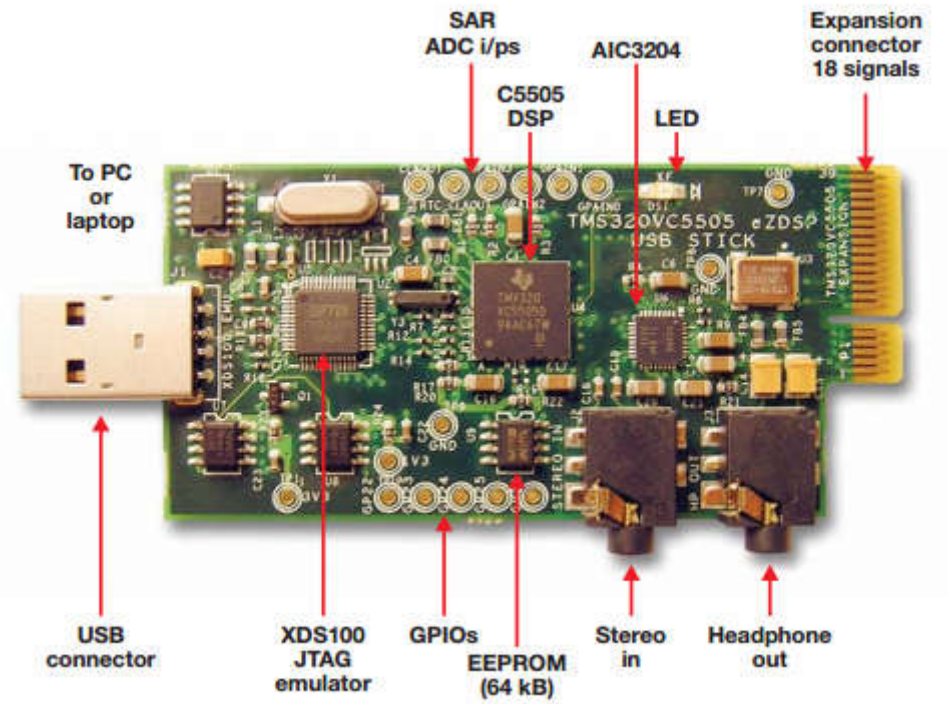
2. DSP/BIOS

DSP/BIOS je operacijski sustav za rad u stvarnom vremenu, razvijen od firme Texas Instruments, te se koristi u razvoju aplikacija za porodice DSP procesora, također razvijenim u firmi Texas Instruments. Dizajniran je za korištenje u aplikacijama koje zahtjevaju zakazivanja i sinkronizaciju u stvarnom vremenu, komunikaciju između sustava i osobnog računala, te za analizu aplikacija pri izvršavanju na konkretnom sustavu u stvarnom vremenu. Neke od karakteristika DSP/BIOS-a su:

- Objekti se konfiguriraju statički i uključuju se pomoću header datoteka
- Smanjuje veličinu koda i optimizira unutarnju strukturu
- API funkcije su modulirane, tako da se unutar programa uključuju samo one koje se koriste
- Biblioteke su optimizirane tako da koriste minimalan broj instrukcija procesora
- Komunikacija između osobnog računala i DSP alata obavlja se izvan glavnog programa, što onemogućava smetnje u glavnom programu

Datoteka koja opisuje DSP/BIOS odnosno sadrži instrukcije za kreiranje datoteka potrebnih za korištenje DSP/BIOS-a, naziva se konfiguracijska datoteka (.tconf) i sadrži konfiguracijske instrukcije potrebne za rad operacijskog sustava. Aplikacije koriste DSP/BIOS pozivanjem API funkcija koje su definirane samim operacijskim sustavom . API funkcije koristimo preko DSP/BIOS modula koji sadrže sve API funkcije potrebne za korištenje određenog modula. Svi DSP/BIOS moduli se mogu pozivati iz glavnog .C programa. Također se većina API funkcija mogu pozivati sa asemblerskim instrukcijama. DSP/BIOS je podržan u integriranom razvojnom okruženju Code Composer Studio.

3. Razvojni DSP alat TMDX5505eZdsp

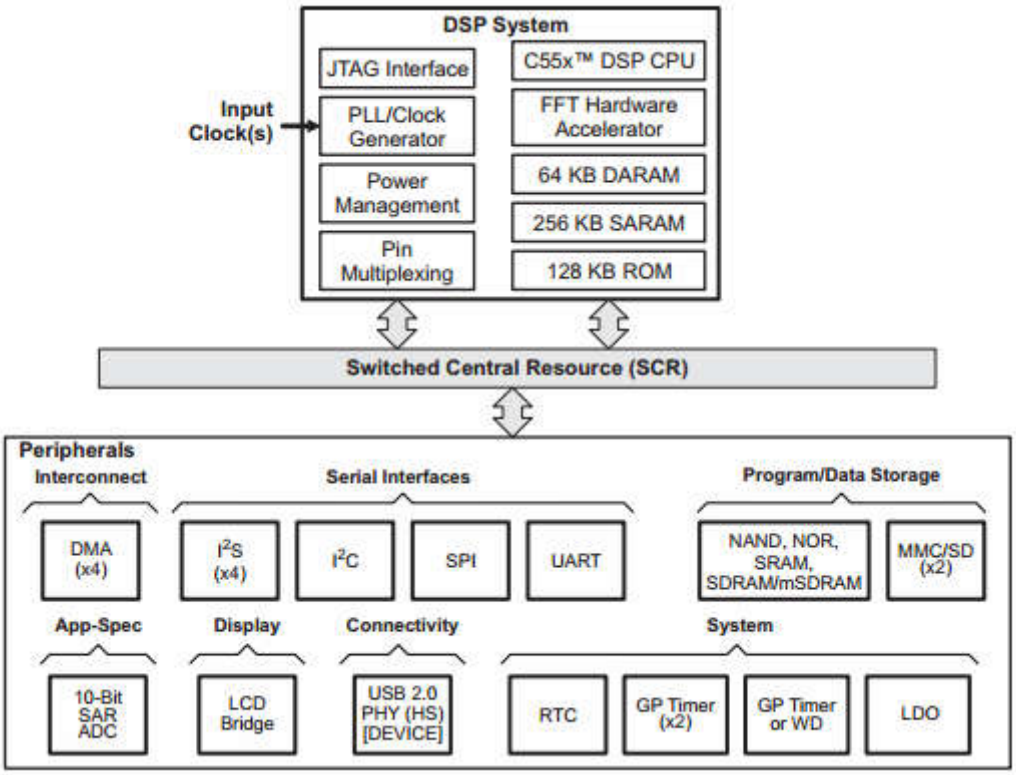


Slika 1. TMDX5505eZDSP

Razvojni alat koji se koristi za analizu operacijskog sustava DSP/BIOS je TMDX5505eZdsp. Sastoji se od: DSP procesora C5505, kodeka AIC3204, dva 3.5mm konektora (Stereo in, Headphone out), napajanja (sa 5V na 1.3V), konektora za proširenje i ostale komponente potrebne za rad razvojnog alata. Komunikacija sa računalom i napajanje razvojnog alata ostvareno je pomoću USB sučelja. DSP kontroler je spojen sa osobnim računalom preko JTAG emulatora XDS100 (USB-JTAG). JTAG je jedan od standarda serijske komunikacije hardvera sa osobnim računalom i omogućava nam jednostavno programiranje DSP-a, te jednostavnu analizu izvođenja programa u realnom vremenu na osobnom računalu.

3.1 TMS320C5505

DSP kontroler TMS320C5505 je dio porodice TMS320C5000 DSP kontrolera sa fiksnim zarezom, razvijen je u firmi Texas Instruments. Njegova arhitektura postiže velike perfomanse uz nisku potrošnju kroz visoku paraleliziranost i fokusom na potrošnju. Procesor sadrži unutarnju sabirnicu sadržanu od programske sabirnice, jedne 32-bitne i dvije 16-bitne podatkovne sabirnice za čitanje, dvije 16-bitne sabirnice za pisanje i dodatne sabirnice za periferiju i DMA. Također sadržava 4 DMA kontrolera, svaki sa 4 kanala. CPU sadrži dvije MAC (multiply-accumulate) jedinice, sposobne za 17x17 bitno množenje, 32-bitno zbrajanje u jednom ciklusu. Centralna 40-bitna ALU jedinica je podržana sa dodatnom 16-bitnom ALU jednicom, te omogućava visoku brzinu i nisku potrošnju. Na DSP možemo spojiti različitu periferiju pomoću GPIO sučelja i 10-bitnog SAR ADC-a. Serijska komunikacija omogućena je pomoću MMC/SD, četiri I2S, SPI, I2C i UART sučelja.

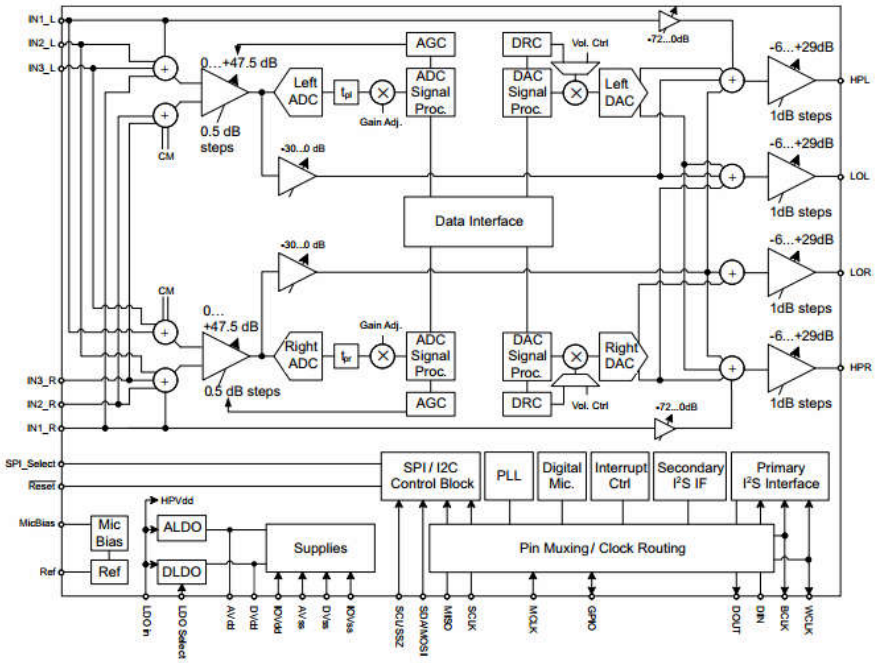


Slika 2. Blokovska shema TMS320C5505

3.2 Kodek - TLV320AIC3204

AIC3204 je fleksibilni, niskopotrošni stereo audio kodek sa programabilnim I/O, PowerTune sposobnostima, fiksiranim parametrima blokova za obradu signala, integrirani PLL (phase-locked loop oscillator), integrirane ragulatore napona LDO i fleksibilno digitalno sučelje.

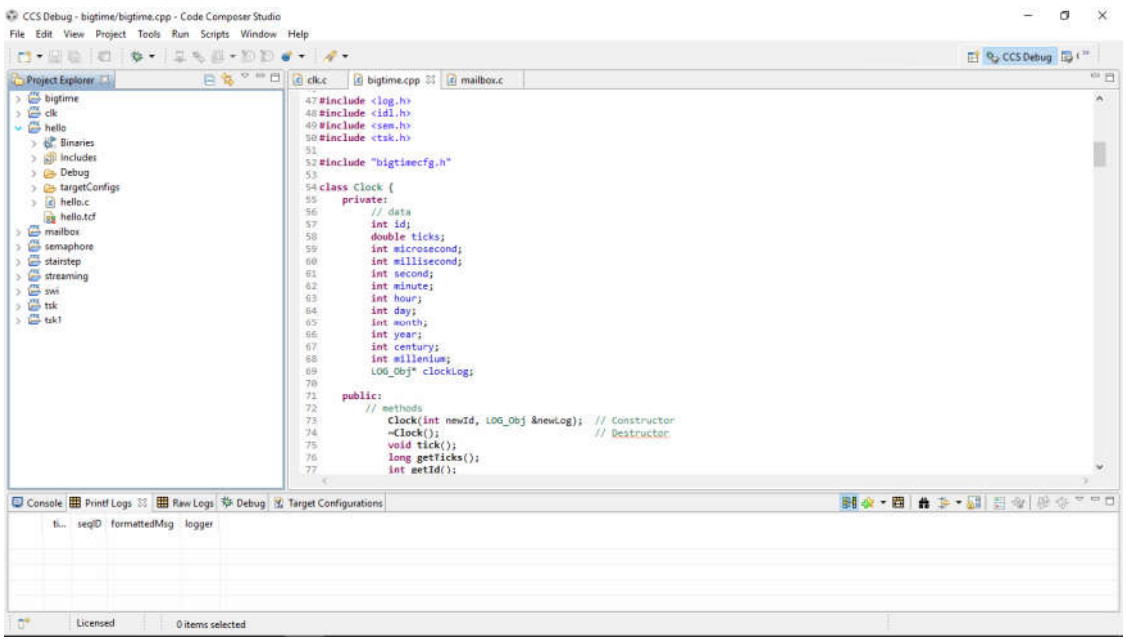
Karakteristike kodeka: digitalno analogni konverter (DAC) sa 100dB SNR (eng. Signal To Noise Ratio), 4.1mW stereo 48ksps DAC reprodukcija, stereo audio analogno-digitalni konverter (ADC) sa 93dB SNR, 6.1mW stereo 48kbps ADC snimanje, opsežne mogućnosti obrade signala. Kodek ima mogućnost snimanja od 8kHz mono do 192kHz stereo i sadrži programibilne konfiguracije ulaznog kanala koje pokrivaju asimetrične i diferencijalne postavke, kao i mješanje ulaznih signala. Uređaj nudi blokove za obradu signala (filtriranje i efekti), a podržava i mješanje DAC-a i anlognih ulaznih signala, kao i programibilno upravljanje glasnoće. Kodek se konfigurira uz pomoć DSP kontrolera preko I2S serijske komunikacije



Slika 3. Blokovska shema TLV320AIC3204

4. Programski alat - Code Composer Studio

Code Composer Studio (CCS) je integrirano razvojno okruženje (IDE) koje podržava ugradbene procesore (mikrokontrolere i DSP kontrolere). Code Composer Studio je razvijen u firmi Texas Intruments, kao i podržani kontroleri, također su podržani neki kontroleri drugih tvrtki.Code Composer Studio sadrži skup alata koji se koriste za razvoj i ispravljanje grešaka u aplikaciji (eng. debugging) . To uključuje optimirajući C/C++ kompajler, editor izvornog koda, okruženje razvojnog projekta, profiler itd. IDE je vrlo intuitivan i dobro opisan programskom dokumentacijom. Code Composer Studio kombinira prednosti Eclipse aplikacije s ugrađenim naprednim alatima za debuggiranje programa. Da bih se program mogao povezati sa odgovarajućom razvojnom pločicom, mora sadržavati datoteke potrebne za informaciju o razvojnom sustavu i za njihovu komunikaciju. Program također sadrži instalirane alate potrebne za rad DSP/BIOS



Slika 4. Code Composer Studio

4.1 Instalacija Code Composer Studio aplikacije

DSP/BIOS sa vremenom gubi podršku i potrebno je instalirati odgovarajuću verziju CCS-a. Starije verzije programa imaju problema sa razvojnim alatima, dok novije verzije ne podržavaju korišteni razvojni alat. Dugotrajnim istraživanjem i instalacijama, može se pronaći prava verzija programa. Verzija programa korištena na ovom radu je Code Composer Studio 5.5.0, uz koju dolazi instalirani DSP/BIOS 5.42

Upute za instalaciju:

1. Preuzeti Code Composer Studio 5.5.0 sa službene stranice (postoji besplatna verzija za korištenje sa ovim alatom)
2. Pokrenuti instalaciju i odabrati instalacijsku mapu (preporučljivo C:\ti)
3. Potrebno je odabrati potrebnu programsku podršku za razvojni alat, a to je C5000 Ultra Low Power DSP, te TI C5500 Compiler Tools i DSP/BIOS v5
4. Odabrati XDS100 JTAG Emulator Support
5. Nakon instalacije potrebno je promijeniti naziv .gel datoteke potrebne za povezivanje razvojnog alata sa CCS-om, koja se nalazi u mapi C:\ti\ccsv5\ccs_base\emulation\boards\usbstk5505\gel , naziva se usbstk5505.gel a potreban naziv je usbstk5505_VC5505.gel
6. Program je sada spreman za pokretanje

4.2 Pokretanje prvog DSP/BIOS projekta u CCS-u

Prije otvaranja projekta potrebno je namjestiti povezanost osobnog računala i razvojnog alata. To ćemo obaviti u slijedećim koracima:

1. Kreirati novi Target Configuration File tako da se otvori **File > New > Target Configuration File**
2. Unutar njega je potrebno izabrati **Connections > XDS100v1 i Board > USBSTK5505_5505** i kliknuti **Save Configuration**
3. Nakon toga potrebno je pokrenuti konfiguracijsku datoteku tako da se u prozoru Target Configurations (kojeg možemo otvoriti u **View > Target Configurations**)na datoteku NewConfiguration pritisne desnim klikom i izabere **Launch Selected Configuration**
4. Zadnji korak je pokretanje konfiguracije pritiskom na **Run > Connect Target**

Kada smo povezali razvojni alat sa CCS-om, možemo pokrenuti prvi DSP/BIOS projekt slijedećim koracima:

1. Prvo moramo otvoriti new project manager tako da kliknemo na **File > New > Project**
2. Otvoriti će nam se prozor gdje izabiremo CCS Project
3. Nakon toga nam se otvara prozor CCS Project gdje moramo navesti ime projekta, porodicu uređaja **C5500 > USBSTK5505_VC5505**, te pod Project templates and examples otvaramo postojeći projekt **Hello Example** koji se nalazi u kartici **DSP/BIOS v5.xx Examples > ezdsp5505 Examples**
4. Kada pritisnemo Finish u Project Explorer-u će nam se prikazati naš projekt

Nakon kreiranja projekta potrebno ga je pokrenuti, a to radimo slijedećim koracima:

1. Desnim klikom pritisnuti na naš projekt u Project Explorer-u i odabrati **Build Project**
2. Kada *buildanje* završi možemo prenesti program na razvojni alat, a to radimo tako da pritisnemo **Run > Load** i odabremo destinaciju .out datoteke koja je kreirana pri *buildanju*
3. Nakon toga možemo pokrenuti program pritiskom **Run > Run Project**

5.1 Konfiguracijski alat i konfiguracijska datoteka

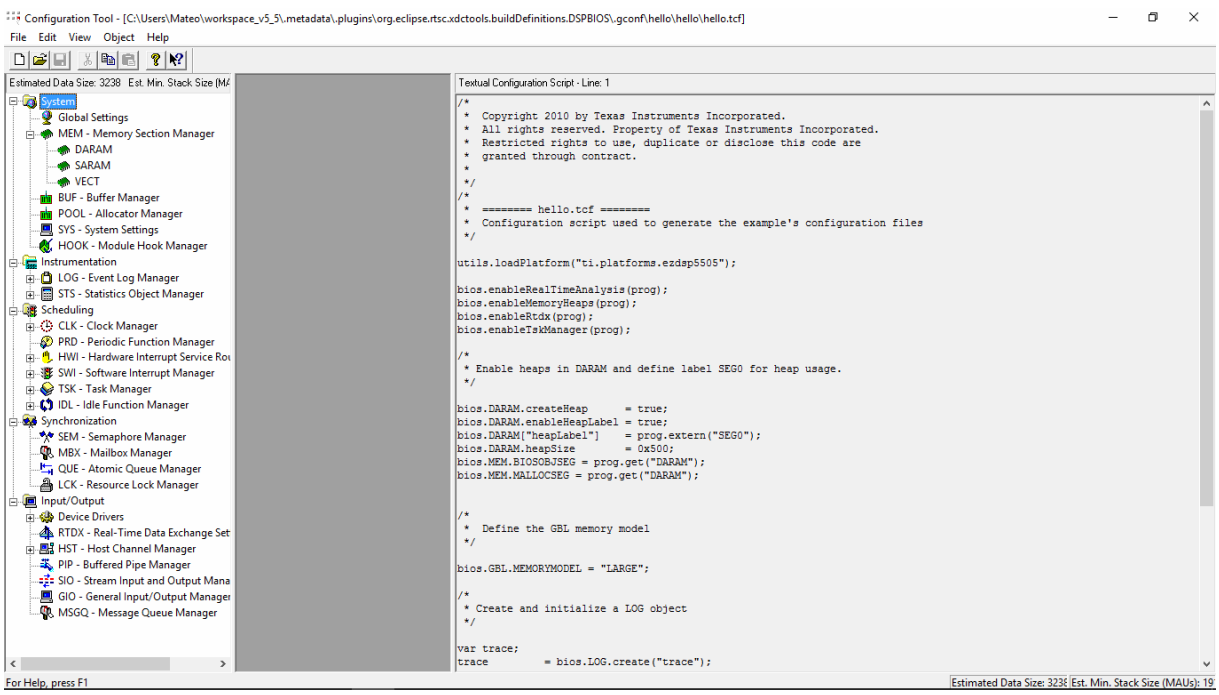
Konfiguracijska datoteka za DSP/BIOS se može kreirati slijedećim koracima **File > New > DSP/BIOS v5.x Configuration File**. Pri kreiranju konfiguracijske datoteke potrebno je odabrati ime i određene alate potrebne za korišteni razvojni alat. Nakon stvaranja datoteke ona se može izmjenjivati otvaranjem i mijenjanjem .Tconf datoteke pomoću tekstualnog editora (što se ne proporuča, jedino ako je nužno) i pomoću konfiguracijskog alata (eng. Configuration Tool) Slika 6, koji se otvara duplim klikom na .Tconf datoteku. Kada namjestimo postavke DSP/BIOS - a unutar konfiguracijskog alata, instrukcije potrebne za rad se spremaju u konfiguracijsku datoteku. Primjer standardnih instrukcija unutar konfiguracijske datoteke možemo vidjeti na slici 5, gdje su konfigurirane osnovne postavke, kao što je npr. uključivanje platforme na kojoj radimo, odnosno razvojnog alata. Pri buildanju projekta, iz konfiguracijske datoteke se generiraju sve potrebno datoteke, koje nam omogućavaju korištenje DSP/BIOS-a unutar glavnog programa.

```
utils.loadPlatform("ti.platforms.ezdsp5505");

bios.enableRealTimeAnalysis(prog);
bios.enableMemoryHeaps(prog);
bios.enableRtdx(prog);
bios.enableTskManager(prog);

/*
 * Enable heaps in DARAM and define label SEG0 for heap usage.
 */

bios.DARAM.createHeap      = true;
bios.DARAM.enableHeapLabel = true;
bios.DARAM["heapLabel"]    = prog.extern("SEG0");
bios.DARAM.heapSize        = 0x500;
bios.MEM.BIOSOBJSEG = prog.get("DARAM");
bios.MEM.MALLOCSEG = prog.get("DARAM");


/*
 *  Define the GBL memory model
 */

bios.GBL.MEMORYMODEL = "LARGE";
```

Slika 5. Konfiguracijska datoteka

5.2 DSP/BIOS moduli

DSP/BIOS se sastoji od modula, pomoću kojih možemo konfigurirati operativni sustav tako da odgovara našoj aplikaciji. Konfiguracijski moduli su: GBL, MEM, BUF, POOL, SYS, HOOK, LOG, STS, CLK, PRD, HWI, SWI, TSK, IDL, SEM, MEX, QUE, LCK, RTDX, HST, PIP, SIO, GIO, MSGQ, CSL. U konfiguracijskom alatu možemo dodati svaki modul u naš projekt tako da pritisnemo desnim klikom na određeni modul i odaberemo Insert <ime_modula> , te pritiskom na novostvoreni modul otvaramo opciju Properties gdje zadajemo postavke modula. Svaki objekt koji dodajemo unutar konfiguracijskog alata biti će dodan u konfiguracijsku datoteku, te će se pri buildanju projekta stvoriti header datoteka za taj modul. Unutar glavnog C programa potrebno je uključiti stvorenu header datoteku (#include <ime_modula>), kako bih mogli koristiti modul. U nastavku će biti opisani neki od glavnih modula i njihova konfiguracija.

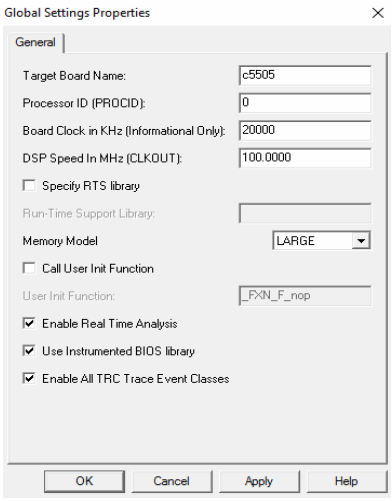


Slika 6. Konfiguracijski alat

5.2.1 Opis modula

GBL (Global Settings)

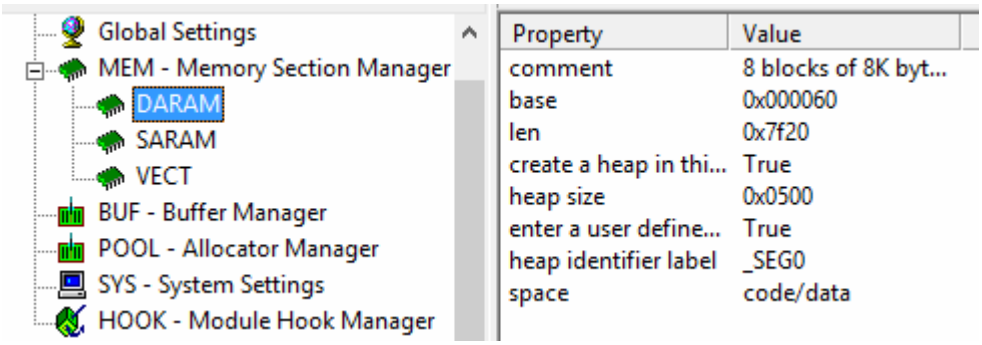
GBL je opći inicijalizacijski modul u kojem se definiraju osnovne postavke kao što su: verzija DSP-a, frekvencija procesora, memorijski model, omogućavanje osnovnih alata itd. Uključivanjem header datoteke GBL modula, omogućava nam korištenje funkcija kao što su: GBL_getClkin, GBL_getFrequency, GBL_getProcid, GBL_setFrequency, GBL_getVersion. GBL funkcije se praktički ne koriste, jer su informativnog tipa.



Slika 7. Global Settings modul

MEM (Memory Management)

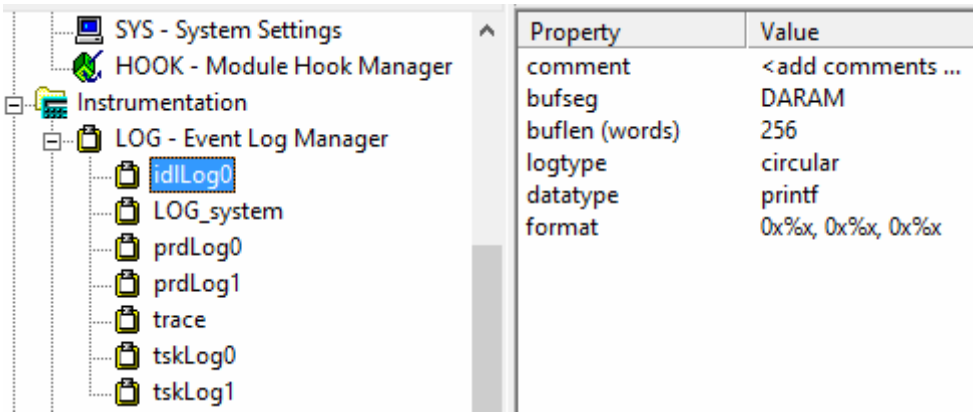
Memorijski modul služi za definiranje memorije unutar DSP-a i stvara se incijalno, odnosno kada stvorimo konfiguracijsku datoteku u kojem smo odredili koji DSP koristimo, pa se njegov raspored memorije prepisuje u modul. Napravljeno je kao modul da se može pronaći raspored memorije, te da korisnik može kontrolirati segmente memorije. Služi i za dinamičku memorijsku alokaciju, također se memorijska alokacija može obavljati pomoću BUF modula. Neke od funkcija MEM modula su: MEM_alloc, MEM_calloc, MEM_free, MEM_getBaseAddress, MEM_valloc. Korištenjem ovih funkcija možemo vremenski optimizirati program, što je glavna svrha korištenja DSP/BIOSA.



Slika 8. MEM modul

LOG (Event LOG)

LOG modul se koristi za hvatanje (analizu) događaja u stvarnom vremenu, dok se program izvršava. Mogu se stvoriti sistemski logovi, također i korisnički definirani logovi. Može se koristiti za ispisivanje grešaka, te za ispisivanje stanja programa koja je sam korisnik zadao. Neke od funkcija su: LOG_disable, LOG_enable, LOG_error, LOG_event, LOG_message, LOG_printf, LOG_reset. Pomoću ovih funkcijama omogućujemo i definiramo LOG modul.

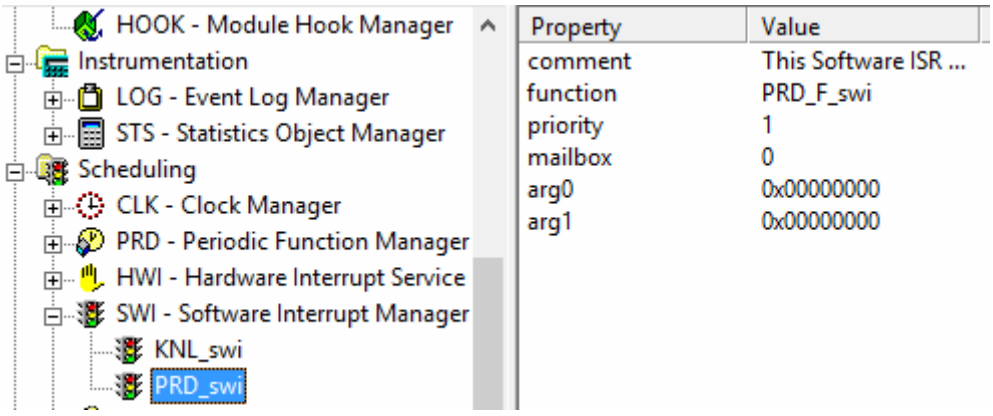


Slika 9. LOG modul

SWI (Software Interrupts)

Modul za programske prekide definira i kontrolira programske prekide. Svaki prekid ima svoj prioritet i funkciju koju definiramo u konfiguracijskom alatu.

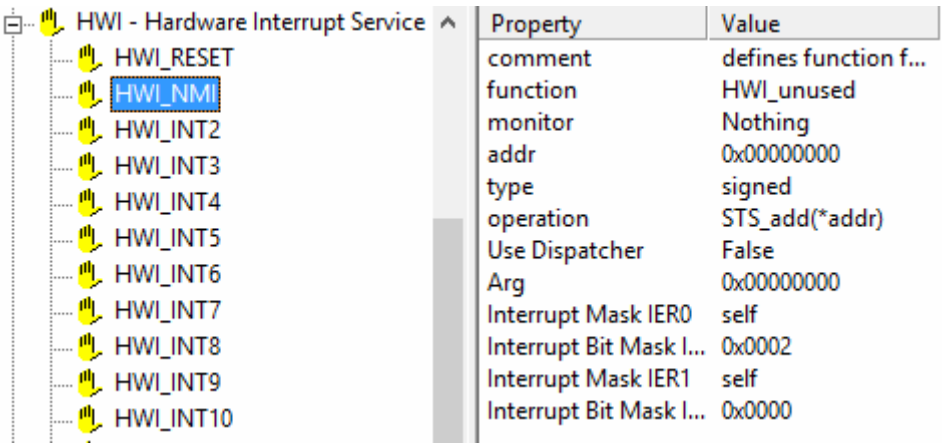
Također možemo stvoriti, definirati i upravljati programskim prekidom uz pomoć funkcija. Neke od funkcija su: SWI_create, SWI_enable, SWI_disable, SWI_post, SWI_andn, SWI_raisepri itd.



Slika 10. SWI modul

HWI (Hardware Interrupts)

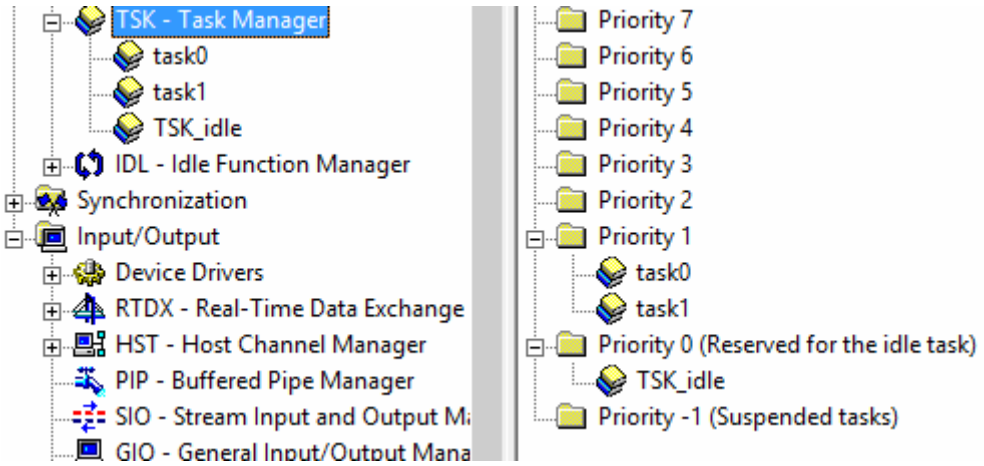
Modul za hardverske prekide preuzima kontrolu hardverskih prekida koje prouzročuju vanjski asinkroni događaji. U tipičnom DSP sustavu prekide prouzročuju unutarnji periferni uređaji ili uređaji van DSP-a. Funkcije HWI modula su većinski napisane u asemblerskom kodu. Funkcije HWI modula su: HWI_disable, HWI_dispatchPlug, HWI_enable, HWI_enter, HWI_exit, HWI_isHWI, HWI_restore. One se koriste za omogućavanje, upravljanje i definiranje hardverskih prekida. Tipičan DSP/BIOS sadrži sistemske hardverske prekide, koji omogućavaju rad sustava. Postoji modul C55 koji se koristi za hardverske prekide specifične za C55xx porodicu DSP-a.



Slika 11. HWI modul

TSK (Task Manager)

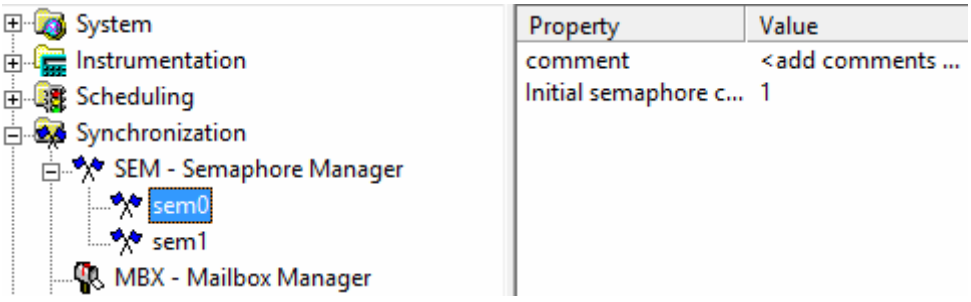
TSK modul služi za stvaranje, definiranje i kontrolu posebnih programskih zadaća. Zadaće imaju veći prioritet nego glavna petlja, ali manji od hardverskih i softverskih prekida. Zadaće se također mogu stvarati i definirati statički pomoću konfiguracijskog alata ili dinamički pomoću funkcija. Funkcije potrebne za kontrolu i stvaranje zadaća su: TSK_create, TSK_delete, TSK_disable, TSK_enable, TSK_stat, TSK_time, TSK_setenv.



Slika 12. TSK modul

SEM (Semaphore)

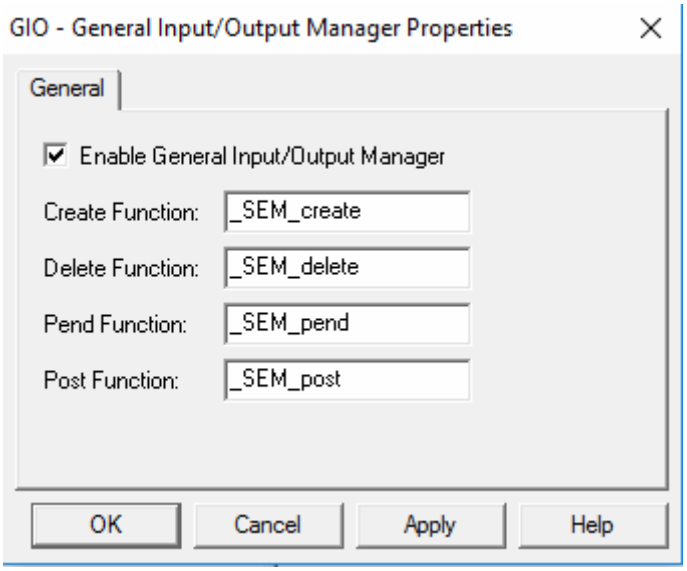
Semaphore modul se koristi za definiranje i stvaranje samaphore-a. Semaphore je tip zastavice koja prati više događaja, ali kao rezultat daje dvije vrijednosti, isto kao i standardne zastavice. Koristi se za sinkronizaciju programa. Može se konfigurirati statički i dinamički, a neke od funkcija su: SEM_count, SEM_create, SEM_delete, SEM_new, SEM_pend, SEM_postBinary.



Slika 13. SEM modul

GIO (General Input Output)

GIO modul pruža standardno sučelje za male upravljačke programe (eng. drivere) za uređaje kao što su UART uređaji, kodek i video prihvati/prikaži (eng. capture/display). Omogućava nam povezivanje DSP-a sa ostalim uređajima.



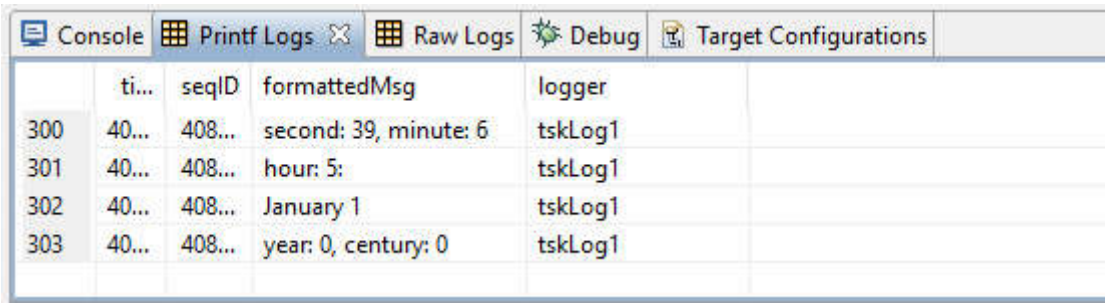
Slika 14. GIO modul Manager

5.3 Programska instrumentacija DSP/BIOS

Jedna od glavnih prednosti koje nudi DSP/BIOS je analiza sustava u stvarnom vremenu. Analiza u stvarnom vremenu (eng. Real-Time Analysis, RTA) nam omogućava analizu podataka dobivenim tokom rada sustava, te nam govori da li naš sustav radi u zadovoljavajućim vremenskim zahtjevima, potrošnji snage, te perfomansama potrebnim za optimalan rad sustava. Neki od RTA alata su: Printf Logs, CPU Load, Load Data, Statistics Data, RTOS Object Viewer

Printf Logs

Printf Log je pogodan način prikazivanja korisnički napisanih poruka, koje se ispisuju tokom izvođenja programa. Alat se otvara putem **Tools > RTOS Analyzer > RTA(Legacy) > Printf Log**, kao i ostali alati koje ćemo spomenuti.

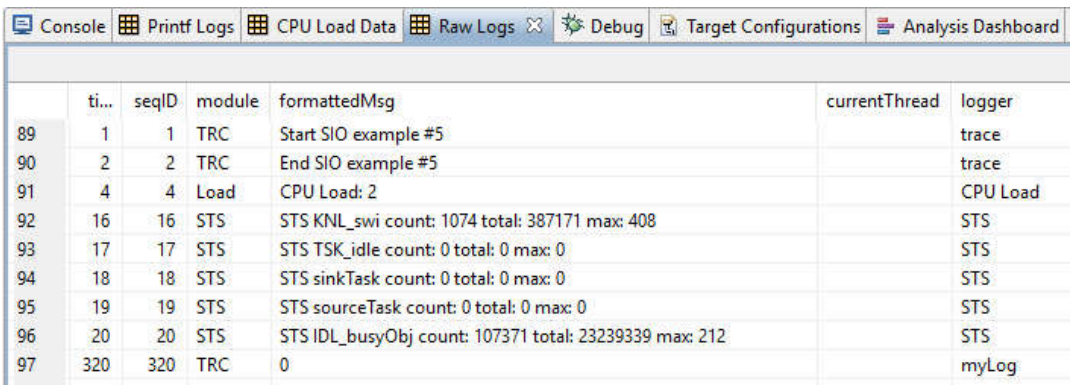


	ti...	seqID	formattedMsg	logger
300	40...	408...	second: 39, minute: 6	tskLog1
301	40...	408...	hour: 5:	tskLog1
302	40...	408...	January 1	tskLog1
303	40...	408...	year: 0, century: 0	tskLog1

Slika 15. Printf Log

Raw Logs

Raw Logs je alat koji nam prikazuje neformatirane podatke dobivene od LOG modula, odnosno prikazuje izvršavanje trenutnih događaja u stvarnom vremenu, koji su definirani unutar samog LOG modula.

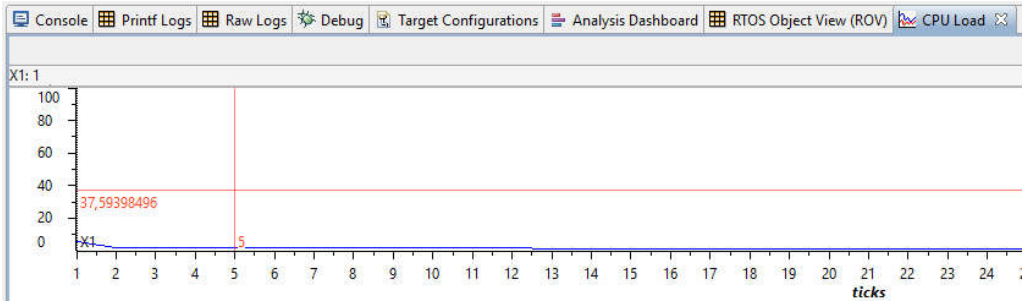


	ti...	seqID	module	formattedMsg	currentThread	logger
89	1	1	TRC	Start SIO example #5		trace
90	2	2	TRC	End SIO example #5		trace
91	4	4	Load	CPU Load: 2		CPU Load
92	16	16	STS	STS KNL_swi count: 1074 total: 387171 max: 408		STS
93	17	17	STS	STS TSK_idle count: 0 total: 0 max: 0		STS
94	18	18	STS	STS sinkTask count: 0 total: 0 max: 0		STS
95	19	19	STS	STS sourceTask count: 0 total: 0 max: 0		STS
96	20	20	STS	STS IDL_busyObj count: 107371 total: 23239339 max: 212		STS
97	320	320	TRC	0		myLog

Slika 16. Raw Log

CPU Load

CPU Load alat nam govori koliko vremena aplikacija nije u idle funkciji, izraženo u postotcima na grafu.



Slika 17. CPU Load

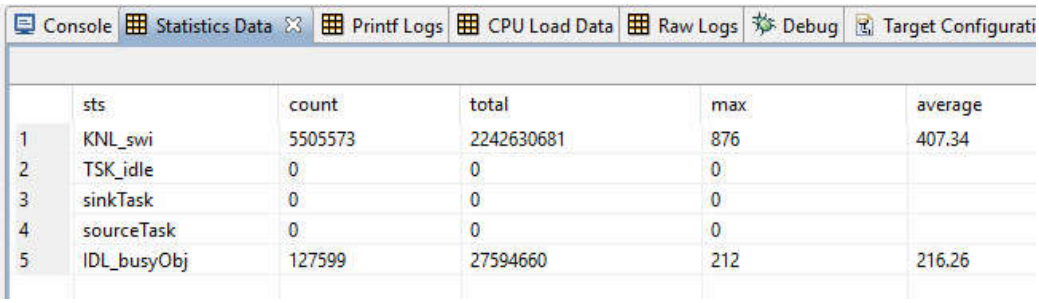
Postotak se računa prema formuli Slika 18., gdje je M*T (broj instrukcija*vrijeme izvršavanja jedne instrukcije), c_w je vrijeme izvršavanja ″korisnih″ ciklusa.

$$\text{CPUload} = \frac{c_w}{MT} \times 100 = \frac{MT - NI_1}{MT} \times 100 = \left(1 - \frac{NI_1}{MT}\right) \times 100$$

Slika 18. Formula za CPU Load

Statistics Data

Statistics Data alat nam ispisuje stastiku vezanu za izvršavanje pojedinih dretvi. Statistički podatci se izarčunavaju i dobivaju preko STS modula, koji je povezan Statistics Data alatom. Prikazuje sve STS objekte koji su definirani u konfguracijskom alatu, te određene podatke vezane za njih, kao što su: Count (broj dobivenih vrijednosti u podatkovnoj seriji), Total (aritmetička sredina vrijednosti), MAX (najveća vrijednost podatka unutar definiranog objekta).



	sts	count	total	max	average
1	KNL_swi	5505573	2242630681	876	407.34
2	TSK_idle	0	0	0	
3	sinkTask	0	0	0	
4	sourceTask	0	0	0	
5	IDL_busyObj	127599	27594660	212	216.26

Slika 19. Statistics Data

6.1 CSL – Chip Support Library

CSL je biblioteka, također i modul (API funkcija) podržan CCS-om i DSP/BIOS jezgrom, te je specifična za svaki procesor. Namjena CSL-a je olakšani pristup periferiji DSP procesora, odnosno njihovoj inicijalizaciji i korištenju unutar glavnog programa. CSL se može koristiti na dva načina: pomoću konfiguracijskog alata i standardnim pozivanjem napisanih funkcija. Unutar konfiguracijskog alata nalazi se skup modula Chip Support Library, unutar kojih možemo statički inicijalizirati i definirati periferiju samog DSP procesora. Neki od modula za sam C5505 DSP su : DMA, I2S, I2C, MMC/SD, SPI, GPIO, UART, PWR, RTC itd. Ovakav način predstavlja jednostavnu inicijalizaciju na grafičkom sučelju, ali nije više podržana na novijim DSP/BIOS konfiguracijskim alatima, zbog mnogih greški koje su se događale pri takvom načinu rada, te portabilnosti aplikacije na neki drugi sustav. Drugi način je direktno pozivanje API funkcija tijekom izvođenja programa. Postoje dvije vrste inicijalizacije periferije: registarski bazirana konfiguracija i konfiguracija sa funkcijskim parametrima. Registarskom konfiguracijom dobivamo manji broj instrukcija za izvršavanje, a sa funkcijski baziranom konfiguracijom dobivamo na portabilnosti same aplikacije.

6.2 Inicijalizacija periferije pomoću CSL biblioteke

Prije same incijalizacije potrebno je uključiti CSL header datoteke za odgovarajuće module periferije koje ćemo koristiti, kao npr. #include "csl_dma.h", te moramo incijalizirati CSL funkcijom CSL_init(); .

DMA (eng. Direct Memory Acces) kontroler

DMA kontroler nam omogućava direktan prijenos podataka između memorije i periferije, odnosno oslobađa procesor od zadaće kontrole, upravljanja komunikacijom i razmjenom podataka s periferijama. Sastoji se od 11 kontrolnih registara, koji su definirani konfiguracijskom strukturom Slika 20.

DMA_Config *DMA configuration structure used to set up DMA interface*

Structure	DMA_Config	
Members	Uint16 dmacsdp	DMA Channel Control Register
	Uint16 dmaccr	DMA Channel Interrupt Register
	Uint16 dmacicr	DMA Channel Status Register
	(DMA_AdrPtr) dmacssal	DMA Channel Source Start Address (Lower Bits)
	Uint16 dmacssau	DMA Channel Source Start Address (Upper Bits)
	(DMA_AdrPtr) dmacdsal	DMA Channel Source Destination Address (Lower Bits)
	Uint16 dmacdsau	DMA Channel Source Destination Address (Upper Bits)
	Uint16 dmacen	DMA Channel Element Number Register
	Uint16 dmacfn	DMA Channel Frame Number Register

Slika 20. DMA konfiguracijska struktura

Prije same incijalizacije DMA kontrolera potrebno je deklarirati *handle* objekt, a to radimo slijedećom linijom *DMA_Handle myDma;* . Inicijalizacija DMA kontrolera je obavljena pomoću registarske konfiguracije, tako da pozovemo funkciju *DMA_config(DMA_Handle hDma, DMA_Config *Config);* , čiji su argumenti handle objekt i pokazivač na strukturu registara koju smo kreirali za našu konfiguraciju.

I2C kontroler sabirnice (I2C bus controller)

I2C je vrsta serijske komunikacije koja se obavlja između procesora i vanjske periferije, te se koristi kada je potrebno spojiti više vanjskih sklopova na procesor. TMS320C5505 sadrži četiri I2C kontrolera, a svaki od njih sadrži 10 konfiguracijskih registara. Incijalizacija će biti obavljena sa funkcijskim parametrima. Funkcijski parametri Slika 21. se definiraju unutar strukture Slika 22. i onda adresu strukture zapisujemo kao argument funkcije PER_SETUP(&I2C_Setup), koja vrši inizijalizaciju I2C kontrolera.

Uint16 addrmode	Address Mode: 0 = 7 bit 1 = 10 bit
Uint16 ownaddr	Own Address (I2COAR)
Uint16 sysinclock	System Clock Value (MHz)
Uint16 rate	Desired Transfer rate (10–400 kbps)
Uint16 bitbyte	Number of bits per byte to be received or transmitted: Value Bits/byte transmitted/received 0 8 1 1 2 2 3 3 4 4 5 5 6 6 7 7
Uint16 dlb	Data Loopback mode 0 = off, 1 = on
Uint16 free	emulator FREE mode 0 = off, 1 = on

Slika 21. Funkcijski parametri

```
I2C_Setup Setup = {
0,          /* 7 or 10 bit address mode              */
0x0000,     /* own address - don't care if master    */
144,        /* clkout value (Mhz)                    */
400,        /* a number between 10 and 400           */
0,          /* number of bits/byte to be received or */
            /* transmitted (8 bits)                  */
0,          /* DLB mode                              */
1           /* FREE mode of operation                */
}
```

Slika 22. Struktura parametara

7. Zaključak

DSP/BIOS jezgra operacijskog sustava nam omogućava relativno jednostavan razvoj kompliciranih sustava, koji zahtjeva obavljanje više procesa na optimalan način. Također nam olakšava duboku analizu aplikacije pomoću odgovarajućih alata, što nam omogućava rješavanje grešaka (eng. debugging), te optimiziranje aplikacije tako da se iskorištava potpuna procesorska snaga DSP procesora. Velika prednost korištenja operacijskog sustava je portabilnost aplikacija između različitih DSP sustava. DSP/BIOS se izbacio iz podrške u novijim razvojnim alatima i DSP porodicama, ali predstavlja osnovu novijih RTOS sustava.

8. Sažetak

U ovom radu se proučava DSP/BIOS jezgra operacijskog sustava za rad u stvarnom vremenu. Pokazuje se način povezivanja razvojnog alata TMDX5505eZdsp sa integriranim razvojnim okruženjem IDE Code Composer Studio, te njegova instalacija i pokretanje projekta na osobnom računalu. Opisan je TMDX5505eZDSP razvojni sustav i njegove karakteristike. Proučavaju se dijelovi DSP/BIOS jezgre, te je opisana njihova funkcija unutar operacijskog sustava. Također su opisani alati za analizu aplikacije u stvarnom vremenu, sadržani u IDE razvojnom okruženju. Obrađene su i vrste inicijalizacije perifernih sklopova, unutar DSP procesora.

Ključne riječi: DSP/BIOS, TMDX5505eZdsp, RTOS, Code Composer Studio, konfiguracijski alat

Summary

This paper examines the DSP/BIOS kernel of the operating system for real-time operation. It shows how to connect the TMDX5505eZdsp development tool with the integrated development environment Code Composer Studio, and installation of tools. The TMDX5505eZDSP development system features and functionality are described in the paper. Parts of the DSP/BIOS core are being studied, and their function within the operating system is described. Real-time application analysis tools, contained in the IDE development environment are described and analyzed. There are also types of initialization of peripheral circuits, within the DSP processor.

Keywords: DSP/BIOS, TMDX5505eZdsp, RTOS, configuration tool, Code Composer Studio

9. Literatura

[1] Sen M. Kuo, Real-Time Digital Signal Processing: Implementations and applications

[2] DSP/BIOS User Guide, http://www.ti.com/lit/ug/spru423i/spru423i.pdf 25.5.2017

[3] TMS320VC5505 eZdsp USB Stick, *Technical Reference,* http://support.spectrumdigital.com/boards/usbstk5505/revb/files/usbstk5505_TechRef_revb.pdf 1.6.2017

[4] TMS320C55x Optimizing C/C++ Compiler v 4.4, *User's Guide,* ,http://www.ti.com/lit/ug/spru281g/spru281g.pdf 25.5.2017

[5] TMS320C55x, *Technical Overview,* http://www.ti.com/lit/ug/spru393/spru393.pdf 24.5.2017

[6] TMS320C55x DSP, *Programmer's Guide*, http://www.ti.com/lit/ug/spru376a/spru376a.pdf 20.5.2017

[7] TMS320C55x DSP/BIOS 5.x API Reference Guide, http://www.ti.com/lit/ug/spru433j/spru433j.pdf 26.5.2017