SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 79

RAČUNALNI SUSTAV TEMELJEN NA KONTROLERU LH79525

Matija Glavinić Pecotić

Zagreb, lipanj 2010.

Sadržaj

1. Razvojni sustavi

1.1. Razvoj proizvoda

Što je potrebno za kvalitetan razvoj proizvoda? Sposoban inženjer, nešto novca, tržište i vrijeme. Bez detaljnog ulaženja u proizvodne procese, konačan cilj razvoja je razviti proizvod u što kraćem vremenu, sa što manjim troškovima koje će tržište prihvatiti. Kako bi se ovo postiglo, potrebno je optimizirati svaku od navedenih stavki. Pitanje koje se nameće je: „Dobro, gdje su tu uklapaju razvojni sustavi?“. Uklapaju se, i to u tri od ovdje navedene četiri stavke.

1.2. Razvojni sustavi

Razvojni sustav je povijesno uređaj koji se sastoji od procesora, ili mikrokontrolera te minimalne količine dodatne logike koja je potrebna kako bi centralna procesna jedinica mogla izvršavati neki program i demonstrirati dio svojih mogućnosti. Temeljna namjena im je bila ponuditi okruženje za učenje rada s novim procesorom, ciljana skupina su bili razvojni inženjeri. Upravo su inženjeri jedna od bitnih stavki razvoja. Naime, da bi inženjer projektirao sustav temeljen na nekom procesoru, on mora biti upoznat s njim. Razvojni sustavi omogućuju upravo jedno takvo upoznavanje, kako s razvojem programske podrške, tako i s razvojem sklopovlja.

S vremenom, razvojni sustavi su prerasli osnovnu logiku potrebnu za rad i počeli su se razvijati u smjeru koji pokazuje ne samo da procesor radi, nego što sve procesor može. Tako se mogu naći sustavi koji implementiraju gotovo sto posto mogućnosti procesora u smislu podržanih perifernih sklopova. Ovakvi razvojni sustavi u velikoj mjeri pozitivno utječu na razvoj, pogotovo u današnje vrijeme kada je brzina razvoja apsolutni imperativ. Neka primjer bude situacija u kojoj kupac treba što prije uređaj koji mora podržavati komunikaciju putem etherneta te implementirati znatnu količinu radne memorije. U ovom slučaju ako je inženjer već radio s traženim procesorom, razvoj će biti znatno olakšan. Samo informacije o načinu dovođenja napajanja, takta i reseta su već dragocjene. Uz to, ako je dostupan razvojni sustav kojemu su navedene komponente samo podskup funkcija, prve verzije uređaja, kroz razvojne sustave, mogu

biti gotove u mnogo kraćem vremenu od prototipa. U situaciji koja je za današnje prilike sasvim realna, a ta je da je kupac uključen u razvoj, moguće je u ovako ranoj fazi kupcu demonstrirat rad uređaja i još uvijek bezbolno unijeti izmjene ako ih kupac zahtjeva. Dodatno, ako je kupac mogao sudjelovati u izradi uređaja, veća je mogućnost da će konačan proizvod više odgovarati njegovim željama.

Konačno, pri izradi prototipa se može upotrijebiti cijela baza znanja koja leži u već razvijenom i uhodanom razvojnom sustavu.

Uz pomoć razvojnih sustava inženjeri mogu prethodno naučiti raditi s novim im procesorima, mikrontrolerima ili nečim trećim, i tek potom razvijati prototip. Prilikom izrade prototipa cijeli podskupovi razvojnih sustava se mogu uzeti kao referentni čime je znatno smanjeno vrijeme razvoja kao i prostor za grešku. Ovakav razvoj se pogotovo ohrabruje u današnje vrijeme kada je ideja što manje razvijati, a što više iskoristiti već razvijeno. Konačno, razvojni sustavu su idealni i u edukaciji budućih inženjera.

U idućim poglavljima će biti dan povijesni pregled razvojnih sustava i neki od trenutno dostupnih razvojnih sustava na tržištu.

1.3. Razvojni sustavi kroz povijest

Prije povijesnog pregleda razvojnih sustava, potrebno je steći dojam o tehnologijama u vrijeme razvoja promatranih sustava. Iako razvojni sustav može označavati jako širok pojam uređaja, u ovom kontekstu se razmatraju oni koji su razvijeni za određeni mikrokontroler ili mikroprocesor.

Prva tehnologija koja je imperativ za razvoj mikroprocesora i mikrokontrolera su integrirani krugovi.

1.3.1. Integrirani krugovi

Ideja integracije kakvu poznajemo danas javila se 1952. godine u radu britanskog znanstvenika Geoffreya Dummera. Dummer je na kraju svog rada napisao:

> „*S napretkom tranzistora i poluvodiča općenito, trenutno se čini moguće elektroničke komponente objediniti u cjelovit blok bez međusobnog povezivanja vodičima. Blokovi se mogu sastojati od različitih slojeva kao što su izolacijski,*

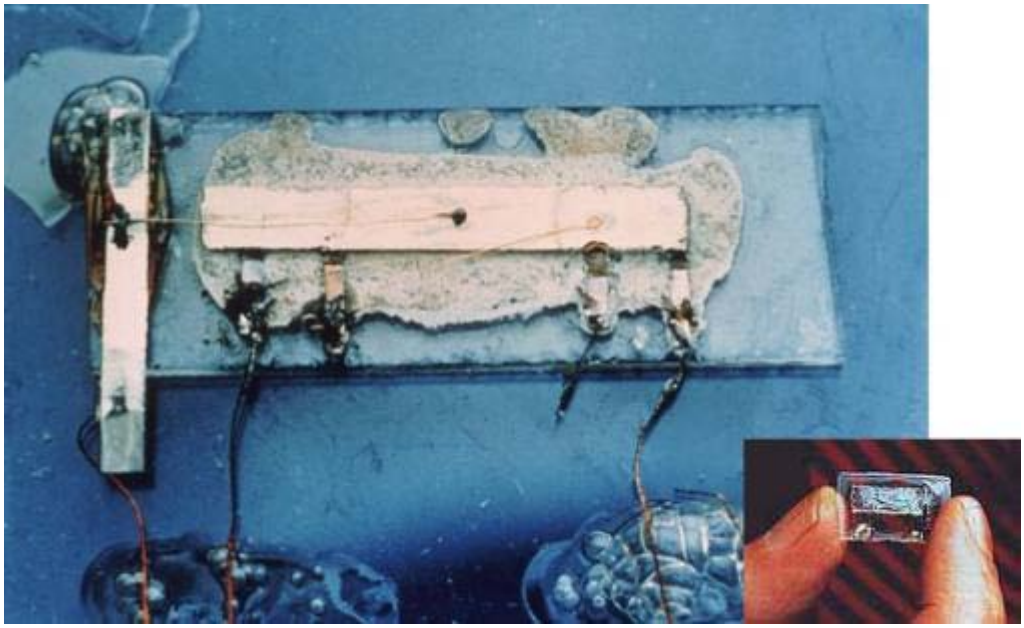
vodljivi, ispravljački te slojeva koji imaju mogućnosti pojačavanja. Funkcija bloka se ostvari na način da se slojevi, podijeljeni na željena područja, povežu međusobno u željenu funkciju“.

Nakon pet godina istraživanja i rada, Dummer je sa svojim timom napravio prototip integriranog kruga i predstavio ga na Međunarodnom sajmu komponenti u Malvernu, u Velikoj Britaniji. Dummer je ponudio integrirani krug koji se sastojao od jednog SR bistabila (flip-flop). Komponenta se sastojala od komada silicija koji je bio oblikovan i dopiran tako da čini četiri tranzistora. Četiri otpornika su bila izvedena kao mostovi u siliciju, a ostale komponente (otpornici i kondenzatori) su bili u obliku filma koji je bio nanešen ravno na silicij.

Tijelo koje je u to vrijeme u većoj mjeri određivalo što će se dalje razvijati, a što neće, britansko ministarstvo obrane, nije prepoznalo ogromni značaj projekta kojeg su imali pred sobom te su konačno Dummerovom projektu ukinuli financiranje, a razvoj integriranih krugova se preselio preko oceana u SAD. Bilo je mnogo ponuđenih razloga, a jedan od njih je bio da nisu imali odgovarajuću aplikaciju u kojoj bi se takva tehnologija mogla iskoristiti. Srećom po modernu elektroniku, amerikanci će kroz par godina imati odgovarajuće aplikacije, no o tome malo kasnije.

Gornja izjava da se razvoj preselio u SAD ipak nije bila posve točna. Naime, bar kako tvrde mnogi izvori, u vrijeme kada je Dummerova ideja odbačena u Britaniji, dvoje ljudi u SAD-u su, neovisno o Dummerovim idejama, razmišljali o istoj stvari. 1958. godine u Texas Instruments je došao Jack Kilby. Nedugo nakon njegova dolaska, većina zaposlenika je poslana na kolektivni godišnji odmor, a kako je bio novi zaposlenik, Kilby je morao ostati u uredu. Mir oko njega i vrijeme za razmišljanje su urodili plodom, Kilby se u to vrijeme bavio problemom elektroničkih uređaja koji se nazivao „tiranija brojeva“. Problem se sastojao u tome da su složeniji dizajni imali prevelik broj komponenti (u to vrijeme je još uvijek svaki tranzistor bio u svom kućištu). Ovo je vodilo mnogim problemima kao što su kompleksnost, složenost sastavljanja i integracije, kratko vrijeme do kvara, itd. Kilby je primjetio da se više tranzistora može staviti na istu pločicu germanija. Osim toga, otkrio je da se na istoj pločici istom tehnologijom mogu realizirati i druge poluvodičke komponente te otpornici i kondenzatori. Ubrzo je sastavio ono što se

danas često naziva prvim integriranim krugom (godina je 1958. Dummer je svoj bistabil predstavio 1957.). Kilbyev integrirani krug (prikazan slikom (Slika 1.1)) se sastojao od jednog tranzistora, par otpornika i kondenzatora.



Slika 1.1 Kilbyev integrirani krug iz 1958. godine

Za razliku od britanskog ministarstva obrane, Kilbyevi su šefovi bili oduševljeni te je TI nastavio istraživanje na novom izumu, a Kilby je podnio zahtjev za izdavanje patenta američkom uredu za patente.

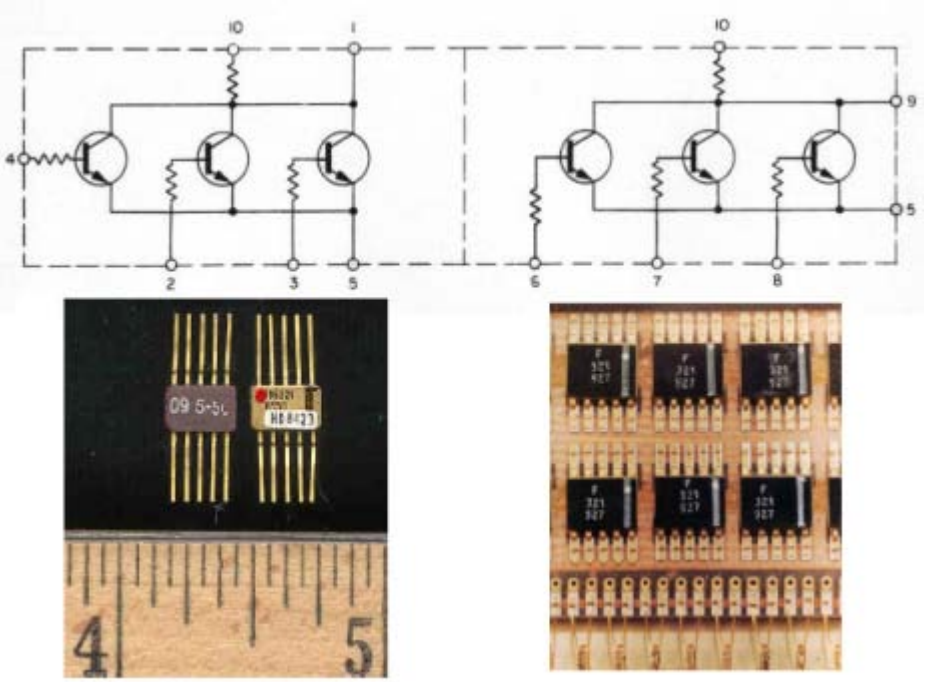
Kako je ranije rečeno, u SAD-u je u to vrijeme dvoje ljudi razmišljalo o ideju integracije. Osim Kilbya, drugi je bio Robert Noyce iz tek 18 mjeseci stare tvrtke, Fairchild Semiconductor. Noyce je također došao na jako sličnu ideju kao i Kilby, no u vrijeme kada je imao razrađenu ideju, u Fairchildu su znali da TI radi nešto jako slično i da su već podnijeli zahtjev za patentom. Noyce povlači genijalan potez te također predaje zahtjeva za patentom, ali je prethodno osigurao dvije stvari: u prvoj se pobrinuo da ne dođe do sukoba s zahtjevom iz TI, odnosno, da jedan patent ne isključuje drugi, a druga stvar je da je napravio jako dobru i detaljnu prijavnicu.

Ovaj se pokušaj isplatio, 24. travnja 1961. ured za patente dodijelio je Robertu Noyceu prvi patent za integrirani krug. Za to vrijeme se Kilbyev zahtjev još uvijek razmatrao, i konačno mu je dodijeljen u lipnju, 1964. Potom su uslijedile tužbe između

TI i Fairchilda koje su okončane 1966. godine tako da su obje tvrtke jedna drugoj dale licencu za proizvodnju integriranih krugova a bilo koja druga kompanija koja je htjela proizvoditi integrirane krugove je morala platiti spomenutima. Geoffrey Dummer nije tužio nikoga, niti je ikada tražio pravo prvenstva na izum. Kilby je za izum integriranog kruga 2000. godine dobio Nobelovu nagradu za fiziku, u to vrijeme je Robert Noyce već preminuo. Dummer je preminuo dvije godine kasnije. Zanimljivo je da je prvi čip koji je 1960. izdao Fairchild bio upravo SR bistabil, odnosno, flip flop. Nakon što su izdani prvi prototipovi, uskoro su uslijedili projekti, kako komercijalni, tako i vladini koji su integrirane krugove uveli široko u industriju i počelo je novo doba za razvoj elektronike, dizajneri više nisu bili ograničeni brojem komponenti, integracija je samo rasla i postalo je moguće sve kompleksnije ideje pretočiti u dizajn.

1.3.2. Stupnjevi integracije

Stupnjevi integracije su se s vremenom samo povećavali i to točno po Mooreovom zakonu koji glasi: „Stupanj integracije raste dva puta približno svake dvije godine“. Prvi čipovi su imali tek par tranzistora u sebi (primjerice, Fairchildov prvi komercijalni čip – flip flop). Ovakav stupanj integracije je dobio ime Niski stupanj integracije (engl. SSI – *Small Scale Integration*). Iako su imali tek par tranzistora, SSI je integrirane krugove odveo u povijest. Projekti američke vlade kao što su AGC – *Apollo Guidance Computer* (ugradbeni računalni sustav za kontrolu navođenja, navigaciju i kontrolu letjelice u misiji Apollo) i Minuteman projektil su početkom 1960. osigurali proboj integriranih krugova na tržište i znatan pad cijene, sa početnih 1000$, na 25$ po čipu. Jedan od prvih široko korištenih integriranih krugova su bila troulazna NILI vrata na kojima je bazirano računalo AGC. Slika (Slika 1.2) prikazuje NILI vrata.



Slika 1.2 Troulazna NILI vrata iz kojih je izvedena cijela logika računala AGC

Kasnih 60. godina pojavljuje se nova tehnologija – Srednja stopa integracije (engl. MSI – *Medium Scale Integration*). MSI tehnologija je bila zanimljiva jer je cjenovno bila u rangu SSI a putem nje je bilo moguće ostvariti još kompleksniji dizajn. Konkretno, podržavala je stotine tranzistora na jednom čipu, za razliku od prethodne SSI sa desetak tranzistora po čipu.

Ranih 1970. javlja se sljedeća tehnologija – Velika stopa integracije (engl. LSI – *Large Scale Integration*). Tehnologija je donijela još više tranzistora, njih tisuće po čipu, te je bila jedna od prekretnica – upravo će ova tehnologija omogućiti znatno kompleksniji i zahtjevniji dizajn kao što je dizajn mikroprocesora ili memorije.

Početkom 80. godina javlja se Vrlo velika stopa integracije (engl. VLSI – *Very Large Scale Integration*) i počela je sa stotinama tisuća tranzistora po čipu i nastavila sve do danas kada govorimo o nekoliko milijardi tranzistora po čipu. Ovime je završen osvrt na integrirane krugove i njihove početke, sljedeća stavka bitna za temu su mikroprocesori i njihov razvoj.

1.3.3. Rani razvoj mikroprocesora

S rastom integracije, 70. godina se javljaju mnogi članci o ideji računala na jednom čipu. Ipak, gotovo svi su imali jednak zaključak, a taj je da trenutna tehnologija nije spremna za takav dizajn.

Kada se govori o prvom mikroprocesoru, teško je reći tko je bio prvi. Povijest se ponovila u smislu da je ponovo bilo nekoliko ljudi i kompanija u igri, a Robert Noyce je i ovog puta imao sreće. Za prve procesore se uzimaju tri čipa: CADC, Intel4004 i TMS1000.

Od tri navedena, prvi koji je proizveden je CADC – *Central Air Data Computer* (MP944) 1970. godine. CADC je razvijen za potrebe američkog borbenog zrakoplova F14 - Tomcat. CADC je bio odgovoran za mnogo stvari na zrakoplovu, od kontrole same letjelice, kontrole unutarnjih sustava, kontrole raketa i još mnogo toga – u konačnici mnogo više od navigacijsko-upravljačkog računala. Ipak, CADC je izgubio svoje mjesto u bitci za prvi mikroprocesor iz više razloga: prvi je što se u vrijeme kada je završen nije niti znalo za njega, naime američka mornarica je smatrala da je dizajn toliko napredan te nije dopustila njegovo objavljivanje. Drugi razlog je taj što se CADC može više poistovjetiti sa *čipsetom*, a ne sa mikroprocesorom. Ovo ima poprilično smisla, CADC se sastojao od 6 različitih čipova: PMU – *Parallel Multiplier Unit* (1), PDU – *Parallel Divider Unit* (1), RAS – *Random Access Storage* (3), ROM – *Read Only Memory* (19), SLF – *Special Logic Function* (1), SLU – *Steering Data Unit* (3). Posljednja jedinica je bila u stvari podatkovni ulazno-izlazni multipleksor koja je podržavala 16 ulaznih i 3 izlazna kanala. Treći razlog koji ne ide u prilog CADC-u kao prvom mikroprocesoru je što je previše aplikacijski ovisan – upravo suprotno ideji mikroprocesora. Ipak, CADC-u se mora priznat nekoliko stvari, jedna su prvi matematički koprocesori (PMU, PDU). Drugo, CADC je mogao mnogo više od i4004 i TMS1000 zajedno, održavati jedan tako složen sustav kao što je F14 u zraku nije nimalo jednostavan zadatak.

Drugi kandidat, koji je konačno odnio pobjedu je Intel4004. Intel je bio relativno mlada kompanija, jedan od osnivača je bio ranije spomenuti Robert Noyce te su se u vrijeme proizvodnje svog mikroprocesora bavili primarno izradom statičke radne memorije.

Prilikom jednog ranijeg pokušaja da Intel proizvede prethodno dizajnirani mikroprocesor, Noyce je izjavio:

> „*Zašto bi proizvodili mikroprocesore ako već imamo memorije. Memorija se uz jedno računalo može prodati stotine, a mikroprocesora samo jedan*“.

No srećom po Intel, 1969. japanska tvrtka Busicom je zatražila od Intela da im proizvedu *čipset* koji se sastojao od 12 čipova kojima je bio cilj ugradnja u kalkulatore. Svaki od čipova je imao svoju namjenu: prikaz, računanje, itd. Intel je zadužio Federica Faggina, Marciana (Ted) Hoffa i Stana Mazora za rad na čipsetu. Nakon proučavanja dizajna, Hoff je došao do zaključka da nema smisla proizvoditi toliko čipova već da se problem može riješiti sa jednim općenamjenskim čipom. Japanci su teškom mukom pristali na promjene u dizajnu i 1971. je proizvedena konačna verzija Intel4004 mikroprocesora. Napon napajanja je bio 15V, početni takt na kojem je radio je bio 108 kHz, maksimalni 740 kHz. Memorijska arhitektura je bila harvardska. I4004 je bio 4-bitni mikroprocesor, adresa je bila 12-bitna, mogao je adresirati 4096x8-bit ROM, 1280x4bit RAM. Instrukcije su 8-bitne te ih je bilo ukupno 46. Imao je 16 registara opće namjene i trorazinski stog. Preciznije rečeno, i4004 je bio dio MCS-4 čipseta, postojali su još 4001 – ROM, 4002 – RAM i 4003 – posmačni registar koji se koristio kao proširenje podatkovnih linija.

Konačno, treći kandidat je bio TMS1000 Proizvođača Texas Instruments. TMS1000 se može slobodno nazvati prvim mikrokontrolerom. Imao je integriran RAM (64x4), ROM (1024x8), ulazno-izlazne sklopove (4-bitni ulaz, 8-bitni izlaz) i centralnu procesnu jedinicu na istom čipu. Radio je na taktu 400 kHz i zahtijevao je -15V napon napajanja. Imao je dva 4-bitna registra i 43 instrukcije. Razlog zašto nije proglašen prvim je što, iako je proizveden 1971., tek je 1974. izašao kao samostalna komponenta na tržište, do tada ga je TI koristio u kalkulatorima.

U godinama koje slijede mikroprocesori počinju osvajati tržište te se pojavljuje sve više proizvođača koji nude svoje mikroprocesore. Iduća značajna godina je bila 1974. Intel je izdao i8080, 8-bitni mikroprocesor koji je radio na taktu od 2 MHz. Poznat je i po tome što se smatrao prvim stvarno korisnim mikroprocesorom. 1975. godine Motorola izlazi na tržište s čipom koji će značajno utjecati na tržište, 6800. 6800 je 8-bitni

mikroprocesor, adresna sabirnica je 16-bitna i u stanju je adresirati 64KB memorije. Radni takt mu je 1 MHz, a što je bilo zanimljivo i privlačno u to vrijeme, zahtijevao je samo jedno +5V napajanje. Motorola 6800 je poznata po uvođenju indeksnog registra i time indeksnog adresiranja. Indeksno adresiranje je uvelike olakšalo rad s poljima. Konkretno, instrukcije tipa su sada bile moguće:

load *odredisni_registar, bazni_registar, indeksni_registar*

U odredišni registar će biti pohranjena vrijednost koja se nalazi na adresi bazni_registar + indeksni_registar. Kasniji mikrokontroleri i mikroprocesori uglavnom podržavaju ovakve instrukcije s bilo kojim registrima opće namjene.

Godine 1976. Izlazi još jedan mikroprocesor vrijedan spomena, RCA(CDP)1802, poznat i pod nazivima RCA COSMAC ili COSMAC 1802. COSMAC je engleska skraćenica od *COmplementary Silicon MetAl-oxide Conductor* što znači da je ovo prvi mikroprocesor temeljen na CMOS tehnologiji. Osim što je bio značajan po tehnologiji izvedbe, imao je i neke arhitekturne značajke koje su bile posebne. RCA1802 ima 8-bitnu podatkovnu sabirnicu i 16-bitnu, multipleksiranu, adresnu sabirnicu. Sadrži 16 registara opće namjene od kojih svaki može preuzeti ulogu programskog brojila. Postoji jedan 4-bitni ulazni port, te jedan 1-bitni izlazni. Minimalna frekvencija je 0 Hz, a maksimalna 3.2 MHz. Potrebno napajanje iznosi +5 V. Ono po čemu je bio još zanimljiv je činjenica da je izrađen u tehnologiji koja mu je osigurala pojačanu otpornost na radijaciju i statički izboj. Ovo ga je učinilo idealnim za aplikacije u svemiru gdje se poprilično vremena i koristio.

Postoji još jedan mikroprocesor koji je potrebno spomenuti kada se govori o ranom razvoju, a taj je Zilog Z80. Z80 je izašao 1976. godine. Glavni dizajner je bio Frederico Faggin, čovjek koji je radio u Intelu. Z80 je bio programski kompatibilan sa i8080 što znači da je mogao izvršavati njegove instrukcije plus vlastite dodatne. Z80 je brzo stekao veliku popularnost. Osim cijene, naprednog skupa instrukcija nudio je i jednostavniju integraciju u sustav. Jedan je od prvih mikroprocesora koji je generirao signale za osvježavanje DRAM-a. također je zahtijevao samo jedan izvor napajanja, i što je bilo novo u to vrijeme, samo jedan izvor takta, odnosno, samo jednu fazu. Z80 se

proslavio u kućnim računalima i u ugradbenim u kojima ga se i danas može naći. Z80 je bio 8-bitni, imao je 16-bitnu adresnu sabirnicu te je radio na taktu do 2.5 MHz.

1.3.4. Memorije

Za potpuno dobivanje osjećaja o stanju u elektronici i računalstvu u vrijeme sedamdesetih godina prošlog stoljeća potrebno je reći još par riječi o memorijama i njihovom razvoju.

U prošlosti, a i danas, pogotovo u ugradbenim sustavima, memorije su bile popriličan problem u smislu da se redovito bile jako ograničen resurs. Opisivanje memorija od početaka razvoja bi bilo previše za uvod, biti će dovoljno reći da su se prije pojave MSI i LSI tehnologija integracije mahom koristile memorije bazirane na magnetskim svojstvima materijala, od čega ni danas nismo previše odmakli. Razlog tome je vjerojatno izvrsna tehnologija koja se krije iza takvih memorija i nedostatak alternative, no srećom po računalstvo, početkom sedamdesetih godina nije postojao dovoljno jak kartel proizvođača magnetski temeljenih memorija i stvari su se donekle pomakle najviše u pogledu RAM i kapacitetom manjih ROM memorija.

Podsjetimo se, prvi integrirani krug je proizveden 1960. Ipak, pojava memorija u obliku integriranih krugova će pričekati još deset godina. Razlog je krajnje jednostavan, broj tranzistora koji se u to vrijeme mogao implementirati po čipu je bio dostatan tek za par bitova, alternative su pružale mnogo više.

Već 1968. godine IBM-ov inženjer Robert Dennard dobiva patent za jedno-tranzistorsku DRAM ćeliju. 1969. novoosnovana tvrtka Intel proizvodi 1 KB RAM-ove, a već iduće godine dostupna je prva DRAM memorija – 1103 (1 Kb). 1971. godine Intel proizvodi 256-bitnu programibilnu memoriju (1101) i 256 B izbrisivu ROM memoriju (EPROM, 1701). U godinama koje slijede kapaciteti memorija prate Mooreov zakon što znači da im se kapacitet udvostručuje dva puta približno svake dvije godine.

1.3.5. Rani razvojni sustavi

Kako uopće definirati razvojni sustav? Ranije je spomenuta definicija o uređaju koji nudi dovoljnu količinu logike kako bi centralna procesna jedinica mogla pokazati kako i što radi te koji je otvoren na način da se takav sustav može dodatno proširiti s vlastitom

sklopovskom i programskom podrškom. Iako na prvi pogled opis i nije tako loš, nakon što ga se još jednom pročita dolazi se do zaključka da se PC slobodno može proglasiti razvojnim sustavom, što i jest u neku ruku istina, ali u ovom kontekstu PC ne spada u tu kategoriju. U ovom će radu gornja definicija biti malo proširena na način da je razvojni sustav onaj uređaj za koji primarno vrijedi gornja definicija odnosno koji je proizveden za tu namjenu. PC više ne spada u tu skupinu.

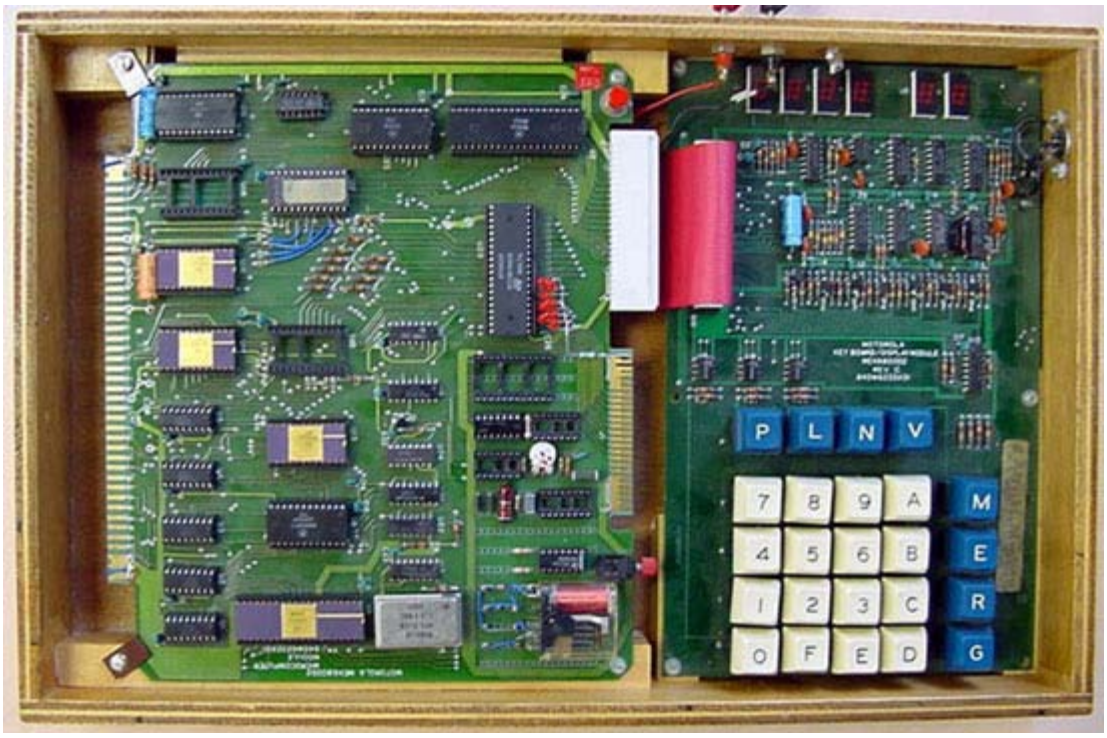
Jedan od prvih poznatijih i široko korištenih razvojnih sustava je bio KIM-1 temeljen na mikroporocesoru 6502 proizvođača MOS Technologies. Nakon što je Motorola izdala 6800, razvojni tim je napustio tvrtku i osnovao svoju, spomenutu MOS Technologies. Proizveli su mikroprocesor, 6501, koji je defakto bio poboljšana verzija 6800 te je uz to i bio pin kompatibilan s 6800. Cijena mu je bila samo 25$ za razliku od Intelovog 8080 i Motorolinog 6800 koji su se prodavali po cijeni od 179$. Motorola je potom tužila MOS Technologies, te su morali povući 6501 te uvest 6502 koji je bio identičan 6501, osim po rasporedu priključaka. Ovim potezom je MOS Technologies izgubio velik dio tržišta, naime, 6501 su korisnici mogli koristiti u sustavima u kojima je bio korišten 6800, a sada su ostali bez platformi.

MOS Technologies 1975. dizajnira i izdaje na tržište KIM-1 – *Keyboard Input Monitor*. Iako je razvojni sustav bio namijenjen inženjerima, vrlo se brzo našao u rukama mnogih drugih. KIM-1 je za to vrijeme imao nisku cijenu, bilo ga je lako nadograditi, a s rastom korisnika, rasla je i zajednica te su se na tržištu ubrzo pojavile i razne kartice za proširivanje funkcionalnosti. KIM-1 se sastojao od 1152 B RAM-a te 2048 B ROM-a. Imao je 16 ulazno-izlaznih linija, tipkovnicu, šest sedam-segmentnih pokazivača, 2 serijska priključka te programibilnim vremenskim sklopom (engl. *timer*). Mikroprocesor je radio na taktu od 1 MHz. Sustav je prikazan slikom (Slika 1.3).

Slika 1.3 KIM-1 razvojni sustav

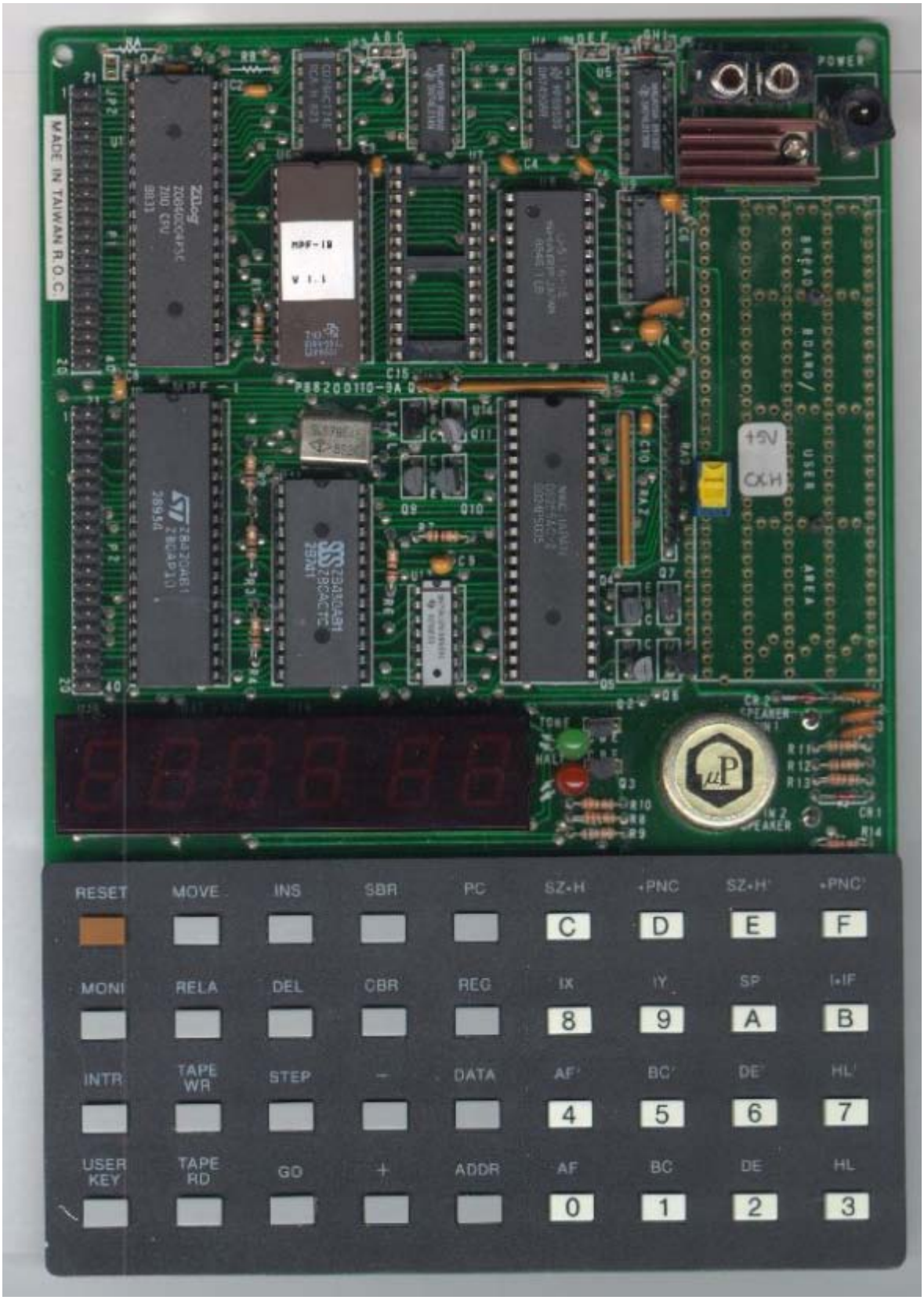
KIM-1 je jako lijep primjer jednog od prvih razvojnih sustava koji je u prvom redu niskom cijenom i otvorenošću privukao kupce i stvorio zajednicu. KIM-1 je također i poznat kao jedno od prvih računala na jednoj ploči, a njegov daljnji razvoj je odveo k Commodoreu 64.

Sljedeći sustav koji vrijedi spomenuti nakon KIM-1 je MEK6800D2. Radi se o službenom, Motorolinom sustavu za 6800. Sustav je dolazio u nekoliko raznih konfiguracija, 256-512 B RAM-a, 1-4 KB ROM-a, u obliku PROM-a ili EPROM-a. Mikroprocesor je radio na 1 MHz. Od zanimljivih komponenti, dolazio je s monitor programom koji je omogućavao praćenje sustava putem serijske veze. Sustav je prikazan slikom Slika 1.4.



Slika 1.4 Razvojni sustav MEK6800D2

Sljedeći zanimljiv razvojni sustav je bio Micro-Professor MPF-I iz 1981. MPF je postigao toliku popularnost da se još uvijek i danas proizvodi. Primarna mu je namjena bila učenje rada s Zilogovim Z80. Sustav se sastojao od spomenutog procesora koji je radio na 1.79 MHz. Imao je 2 KB statičkog RAM-a, 8 KB ROM-a (EPROM) te je dolazio sa monitor programom. Memorija je bila proširiva kroz konektore. Od periferije je implementirao LED pokaznik (6 sedam-segmentnih pokazivača), tipkovnicu i zvučnik. Sustav je prikazan slikom Slika 1.5.



Slika 1.5 Razvojni sustav Micro-Professor MPF-I

Posljednji sustav koji će biti spomenut je Intelov SDK-86 iz 1979. Intel je u toku godina izdao nekoliko razvojnih sustava za svoje mikroprocesore, od kojih će ovdje biti predstavljen spomenuti.

Ono što je najznanimljivije kod SDK-86 je da je imao cijenu manju od samog čipa, prodavao po cijeni od 780$. Intel je uvidio da prodaja mikroprocesora ovisi o broju ljudi koji dođu s njim u kontakt. Sustav se sastojao od 2 KB RAM-a, proširivo do 4 KB, 8 KB ROM-a, tipkovnice, prikaznika te priključaka za proširivanje po želji korisnika. Procesor je radio taktu od 2.5 MHz ili 5 MHz. Sustav je prikazan slikom Slika 1.6.

Slika 1.6 Razvojni sustav Intel SDK-86

1.4. Razvojni sustavi danas

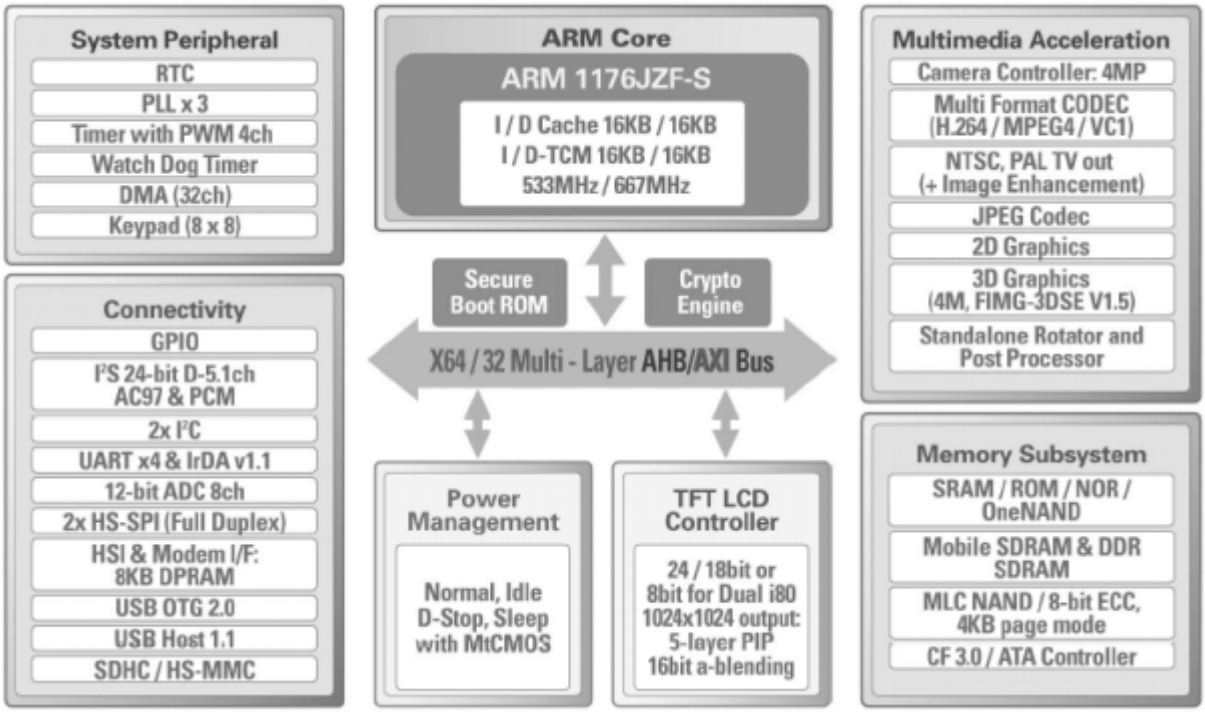
Razvojne sustave danas je teško nabrojati, postoji ih bezbroj na tržištu. Cilj ovog poglavlja je pokazati kakvi se uređaji danas nude.

1.4.1. ARM6410SBC

ARM6410SBC spada u tip ploča kojima je cilj ponuditi što veći broj aplikacija korisniku i koje znatno premašuju definiciju „*Dovoljan skup elemenata koji pokazuju da centralna procesna jedinica radi*“. Sustav je prikazan slikom Slika 1.7.

Slika 1.7 Razvojni sustav ARM6410SBC

Mikrokontroler u sustavu implementira ARM11 jezgru koji radi na 800 MHz. Ostatak sustava čini: 128 MB DDR SDRAM-a, 256 MB NAND flasha, 2 MB NOR flasha, audio konektor (ulazni, izlazni), VGA konektor, TVOUT konektor, SD/MMC sučelje, SPI, ethernet konektor s pripadajućom logikom i još mnoge druge periferije. Sustav dolazi s prilagođenim verzijama WinCE 5.0/6.0, Linux 2.6 te Android operacijskih sustava. Cijena ovakvog sustava je 400$. O snazi samog mikrokontrolera dovoljno govori slika Slika 1.8.



Slika 1.8 Blok shema mikrokontrolera S3C6410

1.4.2. LPC-P2106

Druga krajnost na tržištu je LPC-P2106 razvojni sustav. Sustav je prilično siromašan opcijama i periferijama, te je puno bliži definiciji o nužnoj količini logike dovoljnoj za rad. Sustav se sastoji od LPC2106 mikrokontrolera te pripadajućeg sklopovlja koje je nužno za programiranje i rad. Sustav je prikazan slikom Slika 1.9.

Slika 1.9 Razvojni sustav LPC-P2106

Sustav se prodaje po cijeni od 75$.

1.4.3. Drugi razvojni sustavi

Na tržištu se mogu naći razvojni sustavi svih boja i oblika, po opcijama negdje između dva prethodno dva prikazana s time da postoje naravno i jači, kao i slabiji. Cilj prethodnog razmatranja je bio pokazati stanje tržišta i trendove koji su prilično šaroliki. Još jedna stvar postaje popularna u posljednje vrijeme a to su razvojni sustavi temeljeni na karticama – *Card engine.* To su razvojni sustavi koji na sebi implementiraju mnoštvo periferija i konektora, ali ne i centralnu jedinicu. Centralna jedinica se umeće u sustav putem konektora, i ovime je omogućeno da se razvije jedan takav razvojni sustav te mnoštvo kartica sa željenim centralnim jedinicama. Ovo je pogodno i za proizvođače (većina sustava je razvijena, želi li se podržati nova centralna jedinica dovoljno je projektirati karticu za nju) te za korisnike (jednom kupljen razvojni sustav se može proširiti dodatnim centralnim jedinicama).

2. Razvojni sustav temeljen na mikrokontroleru LH79525

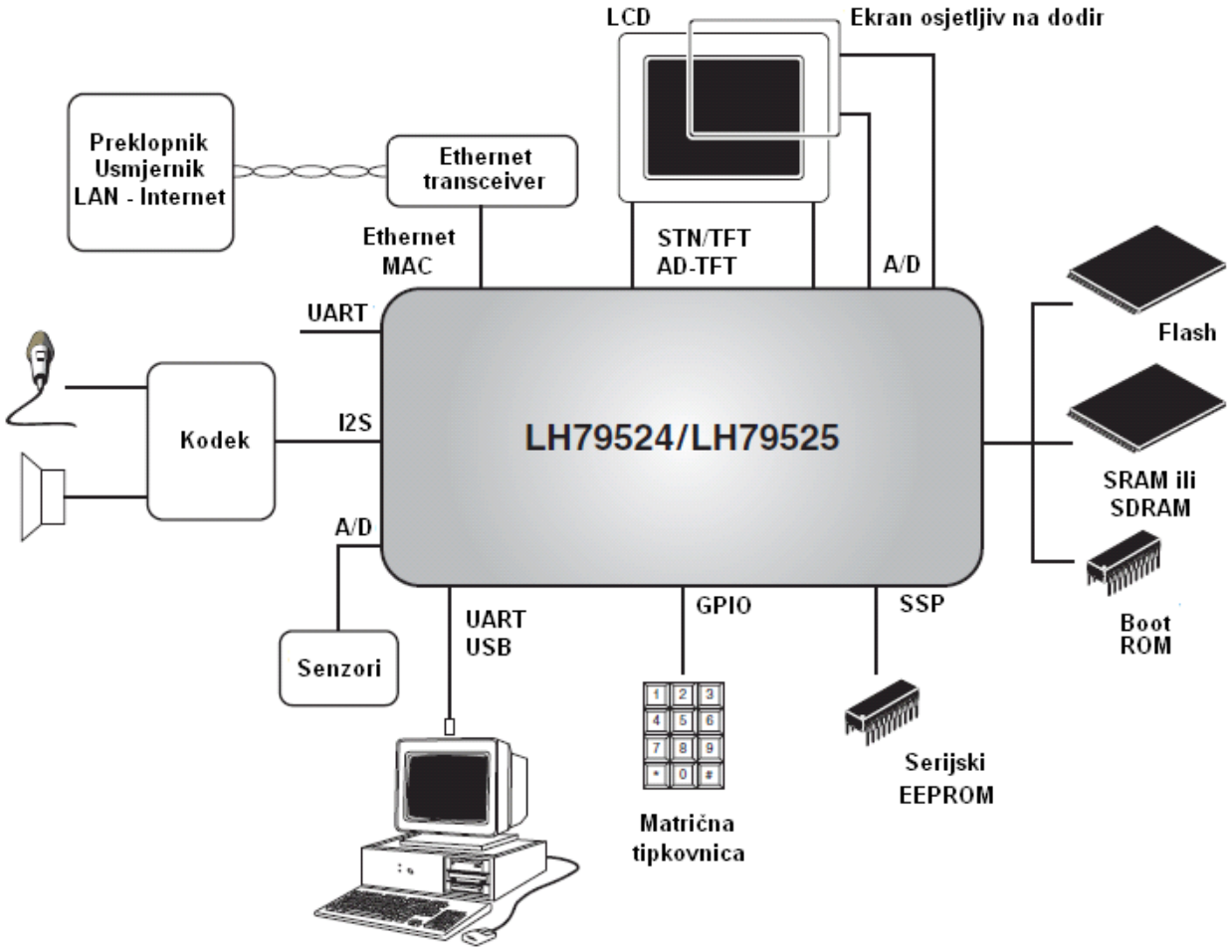
U radu je razvijen i uhodan prototip razvojnog sustava temeljenog na mikrokontroleru LH79525 proizvođača NXP. Kratki opis mikrokontrolera dan je u poglavlju 2.1, a kratak opis sustava je dan u poglavlju 2.2.

2.1. Mikrokontroler LH79525

Mikrokontroler LH79525 implementira ARM720T jezgru te širok izbor dodatne periferije. Glavne značajke sustava su:

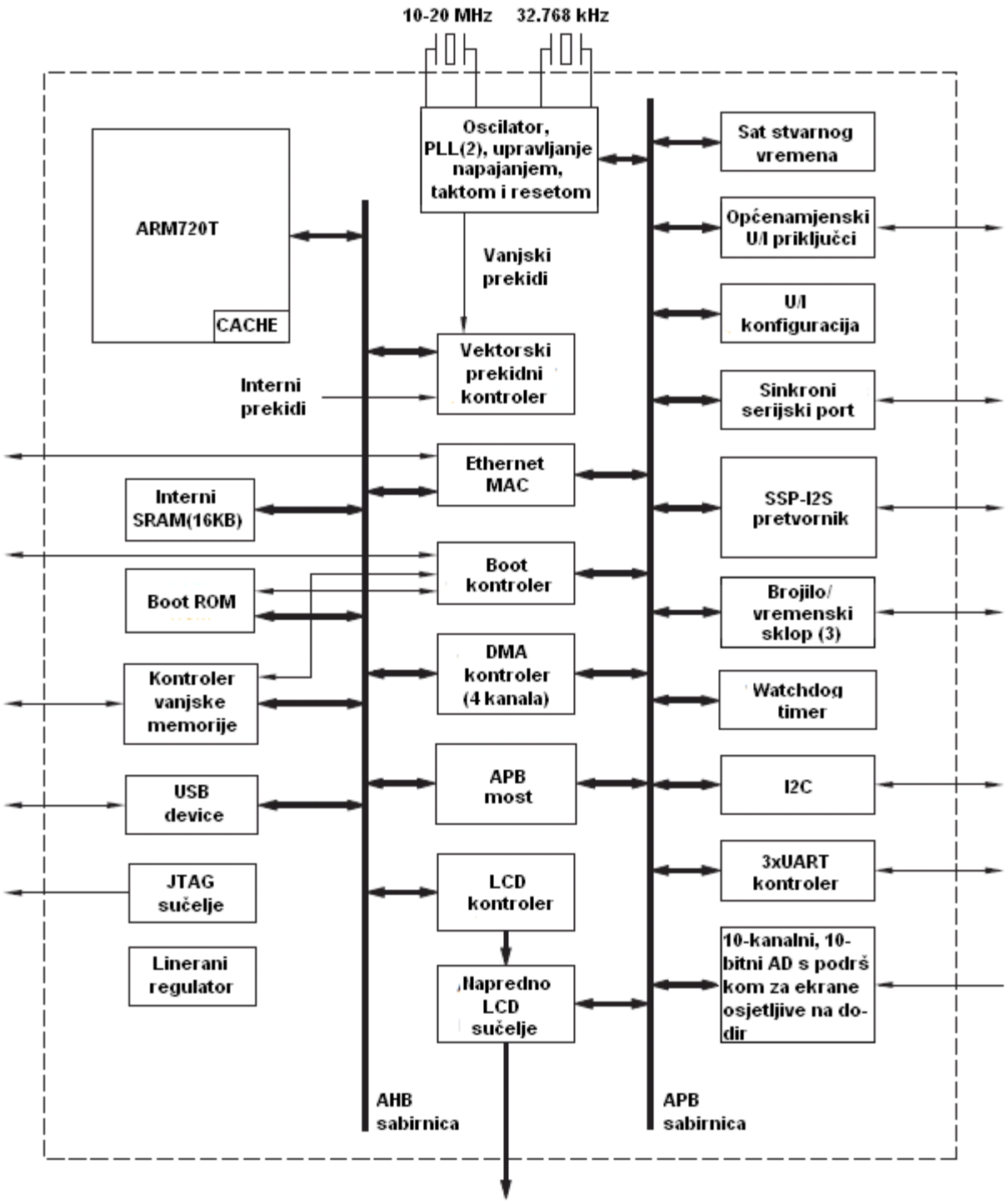
- Fleksibilno pakiranje (208 LFBGA LH79524, 176 LQFP LH79525)
- Maksimalni takt jezgre 76.205 MHz, maksimalni takt sustava (sabirnice) 50.803 MHz
- 8 KB priručne memorije
- MMU (moguće izvršavanje operacijskog sustava Windows CE)
- 16 KB internog SRAM-a
- Vanjska memorijska sabirnica:
    - SDRAM
    - Flash
    - SRAM
    - NAND Flash
- Punjenje (*boot*) iz NAND flasha, statičke memorije, UART-a i I2C-a
- USB *device*
- Ethernet kontroler
- 10-kanalni, 10-bitni AD konverter s ugrađenim kontrolerom za ekrane osjetljive na dodir
- LCD kontroler (STN, TFT do 800x600 rezolucije)
- I2C modul
- UART kontroler (3 kanala) s podrškom za IrDA SIR
- Sat stvarnog vremena
- Tri vremenska sklopa s podrškom za PWM
- Sinkroni serijski kontroler (SPI, TWI, *Microwire*)
- JTAG sučelje
- …

Kako je vidljivo iz prethodnih značajki, radi se o poprilično moćnom mikrokontroleru. Veliki plus je LQFP pakiranje koje je rijetko na ovako periferijama bogat mikrokontroler. LH7952x je jedan od rijetkih mikrokontrolera koji implementira vanjsku memorijsku sabirnicu, LCD kontroler, ethernet kontroler i MMU. Jedna od većih zamjerki je nedostatak interne flash memorije što čini sustav mnogo složenijim za punjenje. Neke tipične primjene mikrokontrolera su dane slikom Slika 2.1.



Slika 2.1 Neke od mogućih primjena mikrokontrolera LH79525

Blok shema mikrokontrolera je dana slikom Slika 2.2.



Slika 2.2 Blok shema mikrokontrolera LH79524/LH79525

Na gornjoj slici je vidljivo bogatstvo perifernim jedinicama uređaja. Dodatan plus cijelom sustavu je što je za jezgru odabran ARM720T koji i nije toliko uobičajen među mikrokontrolerima. Sabirnička arhitektura se temelji na AMBA sabirnici, brze periferne

jedinice su spojene na AHB sabirnicu, dok su one manje zahtjevne spojene na APB sabirnicu.

2.2. Razvojni sustav temeljen na LH79525 mikrokontroleru

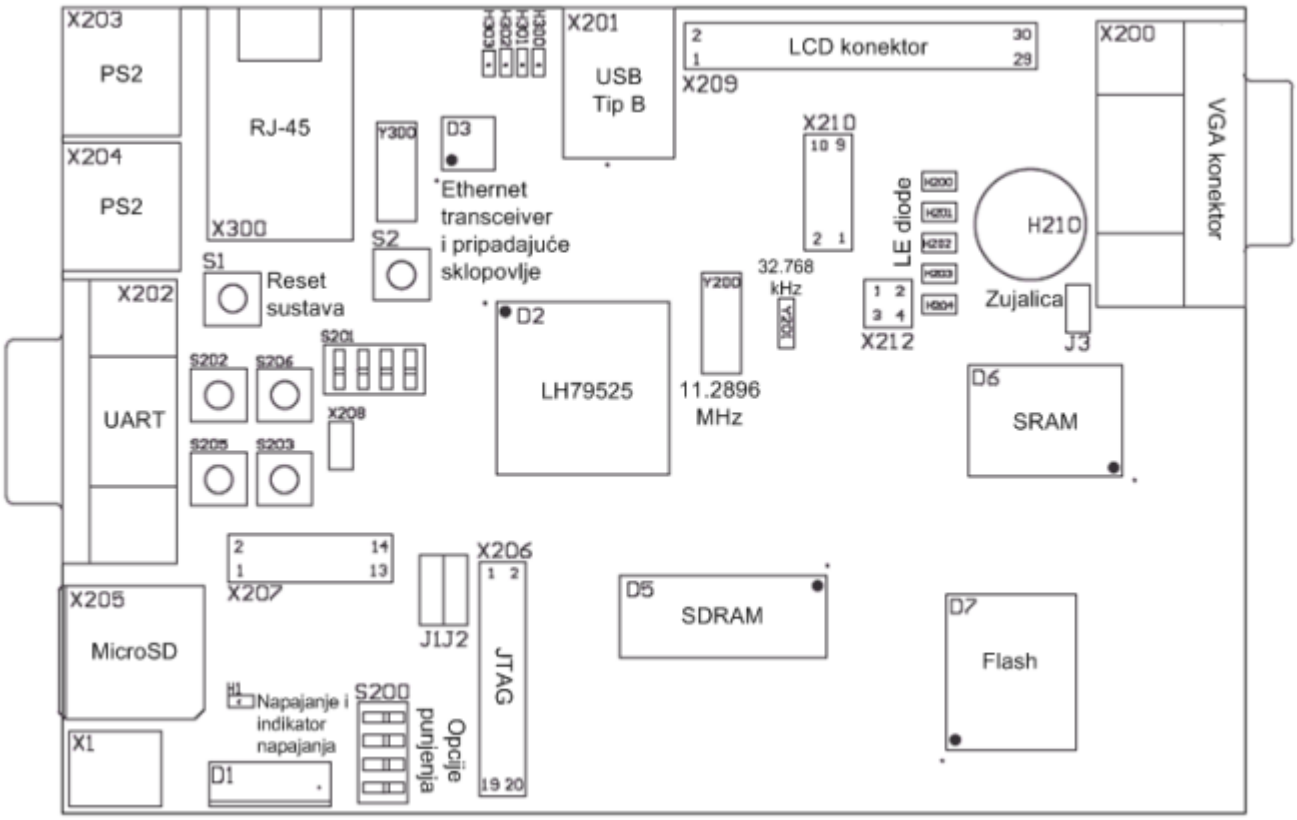
U radu je ostvarena sklopovska i programska podrška koja čini razvojni sustav temeljen na spomenutom mikrokontroleru. Sustav je prikazan slikom Slika 2.3.

Slika 2.3 Razvojni sustav temeljen na LH79525 mikrokontroleru

Glavne značajke sustava su:

- Mikrokontroler: LH79525, jezgra ARM720T
- Memorija:
  - 16 KB internog SRAM-a (LH79525)
  - 8 MB SDRAM-a, proširivo do 256 MB
  - 16 MB Flasha, proširivo do 32 MB
  - 2 MB SRAM-a
- 2 PS/2 konektora
- UART konektor
- RJ-45 konektor zajedno s pripadajućim sklopovljem
- µSD Card konektor
- VGA konektor
- LCD konektor
- USB 2.0 konektor (Device način rada, tip B)
- Tipkala i DIP prekidači
- Svjetleće diode
- Zujalica
- 2 AD kanala
- JTAG konektor
- Generator stvarnog vremena (RTC – *Real Time Clock*)
- Maksimalno 29 općenamjenskih ulaznih pinova, 32 izlazna

Slika 2.4 podrobnije prikazuje funkcije pojedinih djelova razvojnog sustava.



Slika 2.4 Blok shema sustava

Sljedeća poglavlja će detaljno razraditi sklopovsku i programsku podršku sustavu te sve specifičnosti vezane uz iste. Prije korištenja sustava obavezno je potrebno proučiti navedena poglavlja. U suprotnom može doći do nepredviđenog rada ili uništenja sklopovlja.

3. Sklopovska podrška sustavu

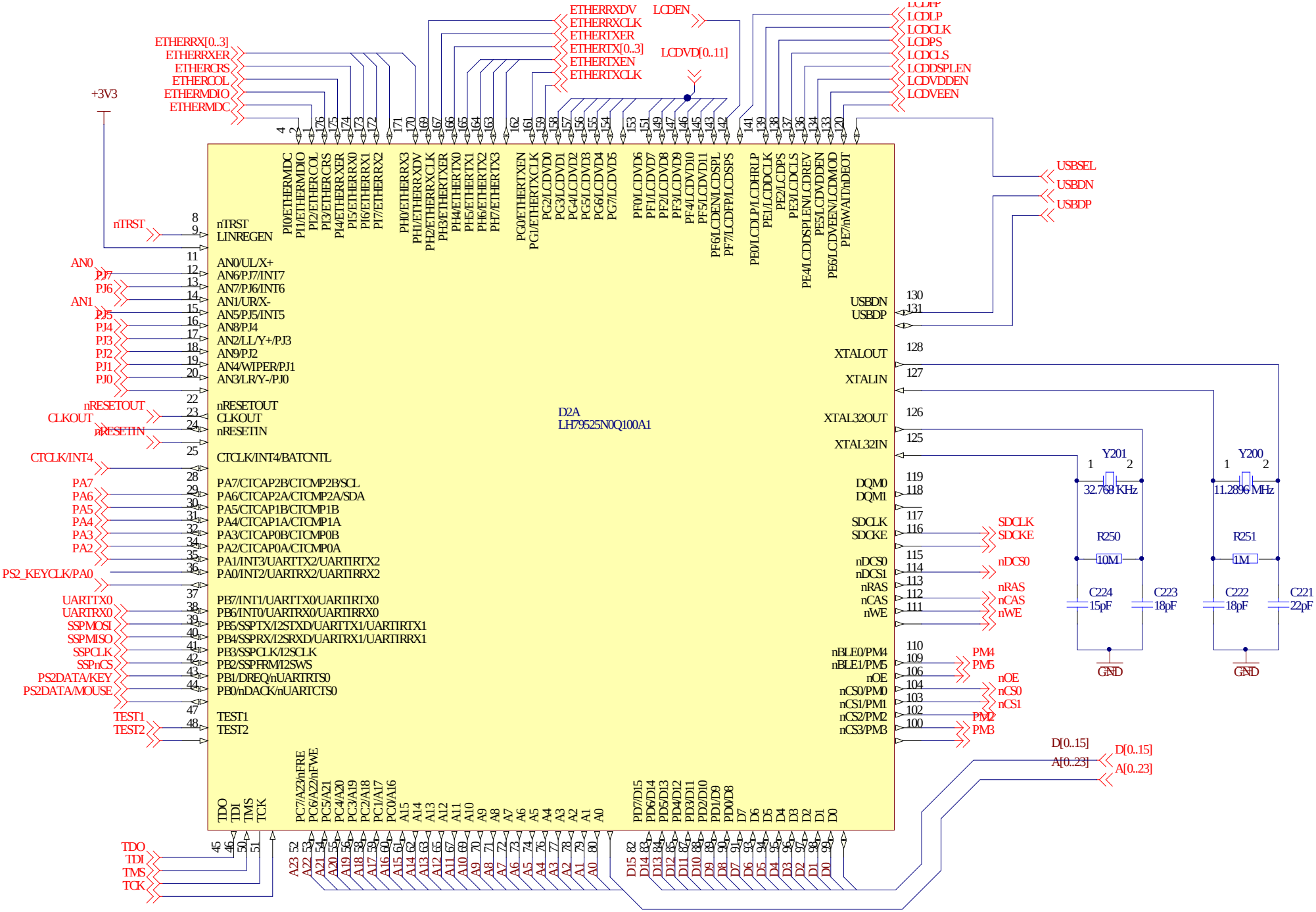
Poglavlje daje detaljan uvid u sklopovsku podršku sustavu. Organizirano je na način da su dijelovi sustava podijeljeni te je svakom dijelu posvećeno posebno poglavlje. Poglavlje završava važnim napomenama vezanima uz daljnji razvoj.

Ovdje neka kao podjela sustava

Mozda i malo terminologije(lemna strana, itd.)

3.1. Spoj mikrokontrolera s ostatkom sustava

Slika 3.1 prikazuje električnu shemu spoja mikrokontrolera s ostatkom sustava. Na shemi se može vidjeti spoj dva oscilatora, jedan je glavni takt za mikrokontroler i iznosi 11.2896 MHz (Y200), dok je drugi namijenjen satu stvarnog vremena – sklop *Real Time Clock* unutar mikrokontrolera i iznosi 32.768 kHz (Y201). Za glavni takt je ne samo poželjno, već i obavezno staviti kristal spomenute frekvencije, u suprotnome neće biti moguće provesti početnu inicijalizaciju mikrokontrolera (za detalje pogledati poglavlje 3.3.2.



Slika 3.1 Električna shema spoja mikrokontrolera s ostatkom sustava

3.2. Napajanje sustava

Sustav zahtjeva samo jedan izvor napajanja, +5V, ulazni krug napajanja spušta ulazni napon na +3.3V, stabilizira ga te dalje razvodi po sustavu. U sustavu je na nekima od konektora dostupno +3.3V i +5V napajanje. Što se tiče napajanja +5V, ono se nalazi iza osigurača, ali nije stabilizirano.

NAPOMENA: Sustav, iako posjeduje predviđeno mjesto za osigurač (komponenta pod oznakom F1 u donjem lijevom kutu na lemnoj strani), taj osigurač nikada nije ugrađen već je premošten žicom, zbog toga je OBAVEZNO koristiti kabel s ugrađenim osiguračem, i to na + žici. Razlog ovome je taj što za standardni osigurač (20x5) nije bilo mjesta u blizini napajanja pa je kao osigurač uzeta komponenta dana u [ref na BOM]. Kako komponenta nikada nije naručena, tako osigurač ne postoji u sustavu već ga je potrebno dovesti izvana.

3.2.1. Napajanje mikrokontrolera

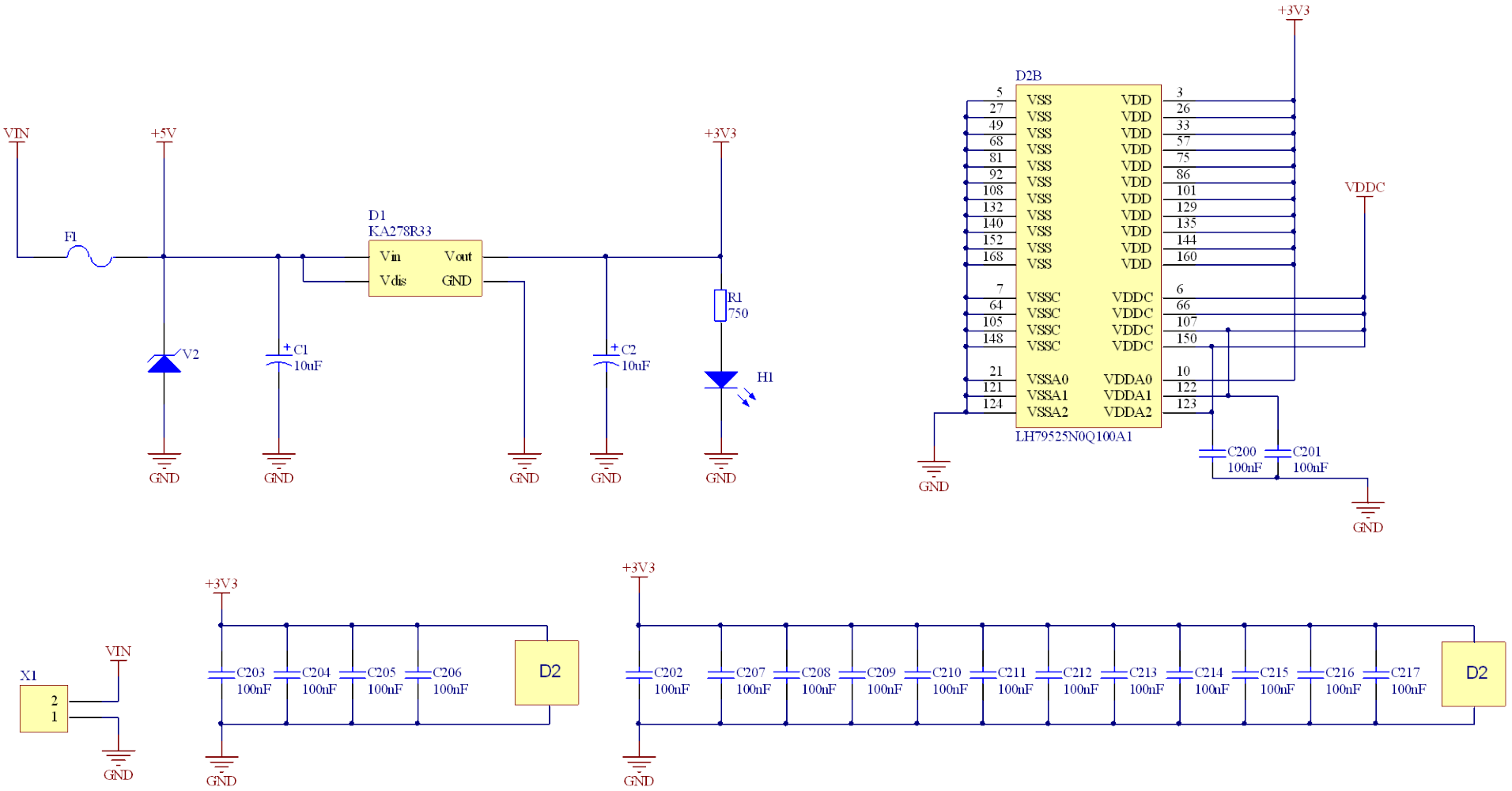
Mikrokontroler zahtijeva dva izvora napajanja, +3.3V i +1.8V. Kako bi se izbjegla potreba za dovođenjem i razvođenjem dodatnog napajanja, iskorišten je linearni regulator samog mikrokontrolera koji se omogućuje dovođenjem +3.3V na priključak LINREGEN. Mikrokontroler posjeduje ukupno 5 različitih priključaka napajanja:

- VDD – 11 priključaka, napajanje ulazno-izlaznih priključaka, +3.3V
- VDDC – 4 priključka, napajanje jezgre, +1.8V
- VDDA0 – 1 priključak, napajanje analogno-digitalnog pretvornika, +3.3V
- VDDA1 – 1 priključak, napajanje za USB-ov PLL, +1.8V
- VDDA2 – 1 priključak, napajanje za sustavski PLL, +1.8V

Kako je vidljivo iz prethodnog opisa, priključci VDD i VDDA0 zahtijevaju napajanje u iznosu 3.3V. Mikrokontroler detektira propad napajanja (engl. *Brownout*) upravo na priključku VDDA0.

Priključci VDDC, VDDA1 i VDDA1 zahtijevaju 1.8V. Ako se koristi interni linearni regulator, priključci VDDC su u tom slučaju izlazi internog linearnog regulatora. Potrebno je napomenuti da su u slučaju korištenja internog linearnog regulatora priključci VDDC napojeni iznutra, međutim to ne vrijedi za priključke VDDA1 i VDDA2.

Ovi priključci se mogu napojiti s VDDC priključaka, što znači da ih je potrebno izvana spojit. Također je potrebno napomenuti da se priključci VDDC ne smiju koristiti za napajanje bilo kojih drugih djelova sustava. Za detalje oko napajanja pogledati poglavlja 1.2, 2.1.4 i 13.1.4 u [ref user manual] te poglavlje *Electrical Specifications* u [ref onaj mali]. Slika 3.2 prikazuje električnu shemu ulaznog kruga napajanja te napajanja mikrokontrolera

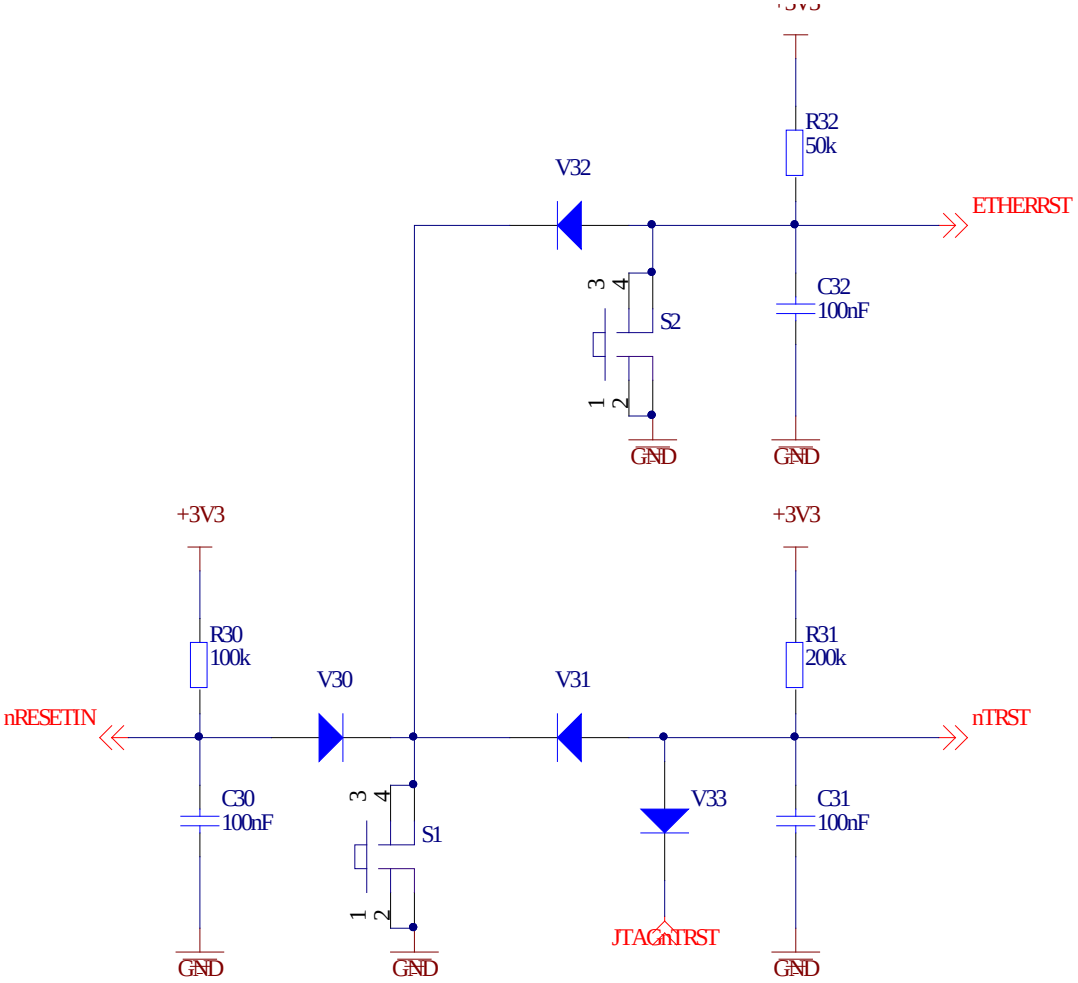


Slika 3.2 Električna shema napajanja mikrokontrolera i ulazni krug napajanja

3.3. Sklopovlje za reset i konfiguraciju sustava

3.3.1. Sklopovlje za reset

Pod sklopovlje za reset spada sklopovlje koje resetira djelove sustava. Sklopovlje je prikazano slikom Slika 3.3.



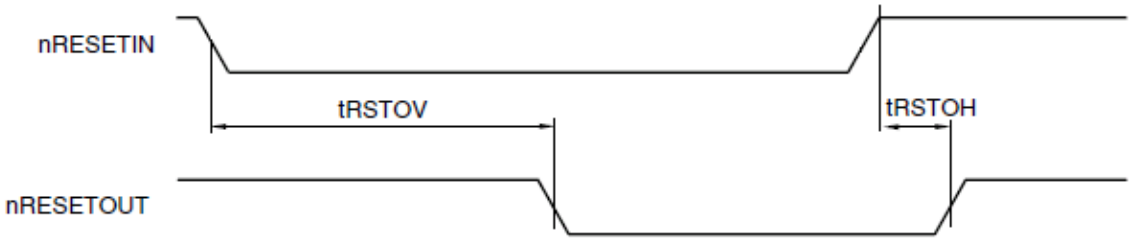
Slika 3.3 Sklopovlje za reset sustava

Mikrokontroler posjeduje dva priključka za resetiranje, to su nTRST i nRESETIN. nRESETIN resetira mikrokontroler i sve registre u njemu dovodi u početno stanje, osim registara JTAG sklopovlja. Priključak nTRST dovodi JTAG sklopovlje u početno stanje. Oba priključka su aktivna nisko. Svaki puta kada se resetira mikrokontroler, potrebno je resetirati i JTAG sklopovlje. JTAG sklopovlje se može resetirati i samo za sebe, no uvijek se mora resetirati zajedno s mikrokontrolerom. Kako je vidljivo na slici Slika 3.3,

reset mikrokontrolera dolazi s tipke S1. Pritiskom na tipku S1, resetirat će se mikrokontroler, JTAG sklopovlje te ethernet kontroler o čemu će bit riječi kasnije. Reset JTAG sklopovlja dolazi iz JTAG kabela, priključka JTAGnTRST. Aktiviranjem signala JTAGnTRST (aktivan nisko) resetirat će se samo JTAG sklopovlje, dok će se aktiviranjem signala S2, odnosno, pritiskom tipke S2 resetirati samo ethernet kontroler.

Mikrokontroler zahtijeva duljina reseta po stabilizaciji u trajanju od minimalno 200 μs, te širinu pulsa u normalnom načinu rada od 2 perioda HCLK (izvori i vrste takta su opisani u poglavlju 4.2.2). Sklopovljem prikazanim na slici Slika 3.3 duljina trajanja reseta za mikrokontroler je oko 10 ms, za JTAG sklopovlje oko 20 ms, dok je za ethernet kontroler duljina trajanja reseta oko 5 ms. Iz reseta najprije izlazi ethernet kontroler što je i logično u primjenama kada mikrokontroler radi s njim odmah po resetu, da je situacija drukčija, mikrokontroler bi morao čekati ethernet kontroler da izađe iz reseta što bi dodatno kompliciralo sustav. Mikrokontroler nema zahtjeve na prvenstvo izlaza iz stanja reseta za JTAG sklopovlje i ostatak sustava, međutim trenutno je ostvareno da prije izađe mikrokontroler i potom JTAG sklopovlje.

Aktiviranje signala nRESETIN aktivira signal *System Reset* unutar mikrokontrolera koji je zadužen za distribuciju reset signala u mikrokontroleru. Priključak nRESETOUT (konektor X208, opisan u poglavlju 3.6.6) također ima izvor u *System reset* signalu. Ovisnost nRESETIN priključka o nRESETOUT priključku je prikazana slikom Slika 3.4.



Slika 3.4 Ovisnost signala nRESETOUT o signalu nRESETIN

Vrijeme je vrijeme nakon kojeg je signal nRESETOUT valjan po aktiviranju signala nRESETIN i iznosi 3.5 perioda HCLK. Vrijeme predstavlja vrijeme nakon deaktivacije

signala nRESETIN za koje je signal System reset još aktivan i ono iznosi minimalno 8 perioda HCLK.

3.3.2. Konfiguracija sustava

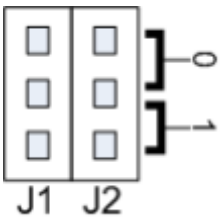
Po resetu, sklopovlje mikrokontrolera uzorkuje stanja pojedinih pinova te u ovisnosti o njima započinje s radom. Pinovi koji se uzorkuju su: TEST1, TEST2 te PC[7:4].

TEST1 i TEST2 su spojeni na premosnike J1 i J2. Tablica 3.1 pokazuje ulogu priključaka.

Tablica 3.1 Stanje priključaka po resetu i pripadajući način rada

TEST1	TEST2	Način rada
1	1	Normalni
0	1	Embedded ICE
0	0	PLL bypass

Slika 3.5 pokazuje moguća stanja premosnika.



Slika 3.5 Moguća stanja premosnika J1 i J2 te njihov utjecaj na priključke TEST1 i TEST2

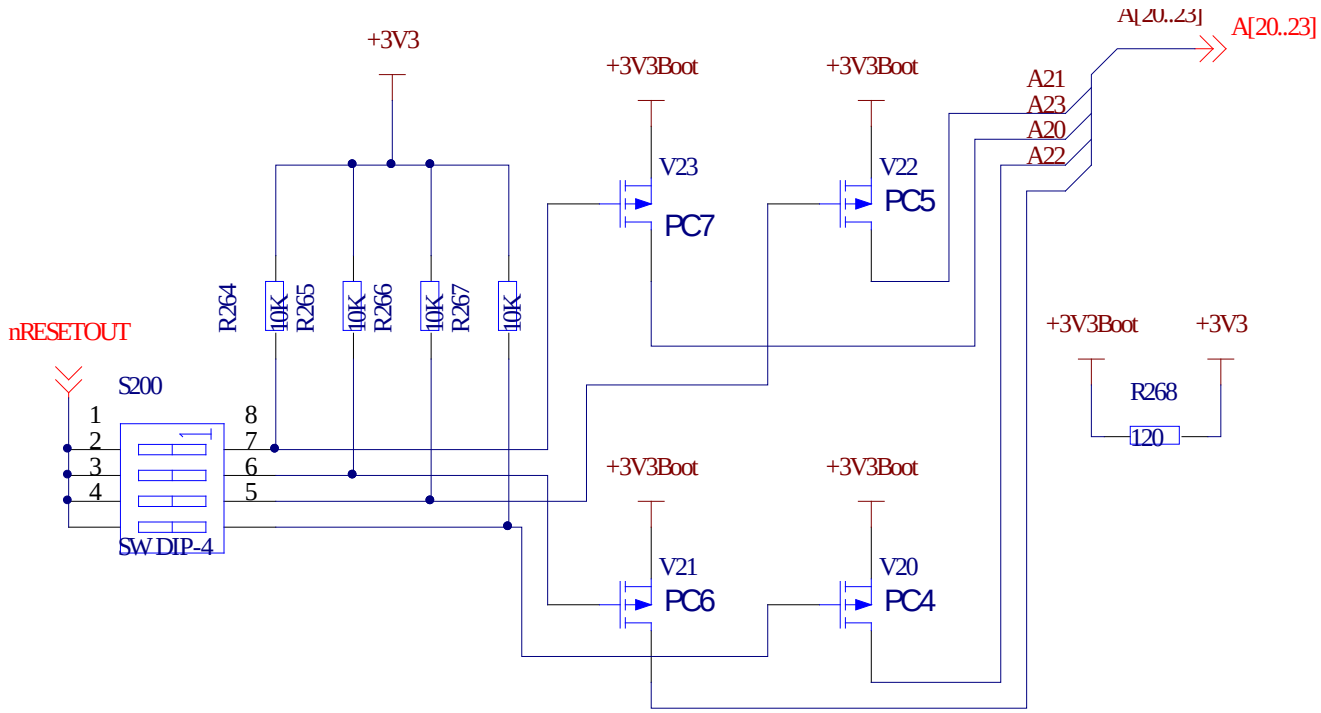
Kako TEST1 i TEST2 imaju unutarnje pritezne otpornike prema napajanju, za normalni način rada stanje se ne mora definirat, odnosno, premosnici i ne moraju biti spojeni.

Druga grupa priključaka koja se uzorkuje su PC[7:4]. Stanje priključaka se određuje putem DIP prekidača S200. Slika 3.6 prikazuje njegovu poziciju u sustavu te odgovarajuće stanje.

Slika 3.6 Prekidač S200 i utjecaj na stanje priključaka PC[7:4]

Priključci PC[7:4] određuju odakle će mikrokontroler učitati program. Punjenje mikrokontrolera kodom je daljnje razrađeno u programskoj podršci sustavu, poglavlje 4.2.4.

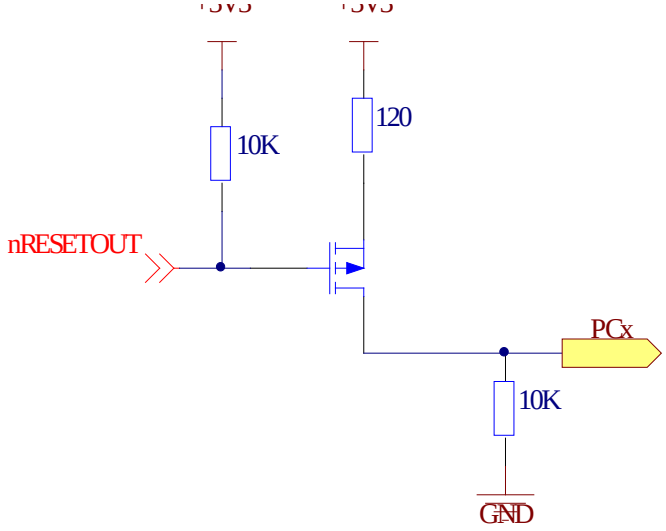
Problem koji se ovdje nameće je sljedeći: priključci PC[7:4] se po resetu koriste za odabir izvora programa, međutim već po završetku reseta priključci se koriste kao adresna sabirnica, naime, sekundarna funkcija priključaka PC[7:4] su A23, A22, A21 i A20, respektivno. Kako se stanje ovih priključaka učitava na rastući brid signala nRESETOUT, tako je upravo ovaj signal iskorišten kako bi se priključke za vrijeme reseta koristilo kao konfiguracijske, a kasnije po potrebi. Slika 3.7 prikazuje shemu sklopovlja koje određuje stanje priključaka PC[7:4] za vrijeme reseta.



Slika 3.7 Sklopovlje za konfiguraciju priključaka PC[7:4] za vrijeme trajanja reseta

Za vrijeme reseta, nRESETOUT je u nuli. Sklopka S200 dovodi signal nRESETOUT do P-kanalnih FET-ova. U slučaju da je sklopka za pojedini FET zatvorena, taj FET će provesti te će priključak spojen na njega biti u visokom stanju. Ako je sklopka otvorena, gate će biti u visokom stanju zbog R264:R267 priteznih otpornika te FET u tom slučaju ne vodi te ne utječe na stanje priključka. U ovom stanju će na priključku biti nisko stanje iz razloga što su za vrijeme reseta na PC[7:4] uključeni pritezni otpornici prema masi.

Također je potrebno primjetiti da je korišten zajednički pritezni otpornik prema napajanju za sve FET-ove (otpornik R268). Po kraju reseta, stanje je jednako situaciji u kojoj je sklopa otvorena. Slika 3.8 prikazuje pojednostavljenu situaciju promatranu za samo jedan priključak.



Slika 3.8 Pojednostavljena situacija za samo jedan priključak

U ovakvom spoju će priključak PCx biti u logičkoj nuli za vrijeme trajanja reseta, dok po isteku reseta sklopovlje nema utjecaj. Također je potrebno primjetiti da je na ovoj shemi 10K pritezni otpornik prema napajanju nepotreban, da postoji sklopka između nRESETOUT i gatea od FET-a koja bi gate u jednom stanju ostavila u zraku, u tom slučaju bi pritezni otpornik bio potreban kako bi se definiralo stanje na gateu kada je sklopka isključena. Otpornik 10K prema masi se nalazi unutar mikrokontrolera. Napomena: Kako bi se mikrokontroler ispravno napunio putem UART ili I2C sučelja, potrebno je spojiti oscilator 11.2896 MHz.

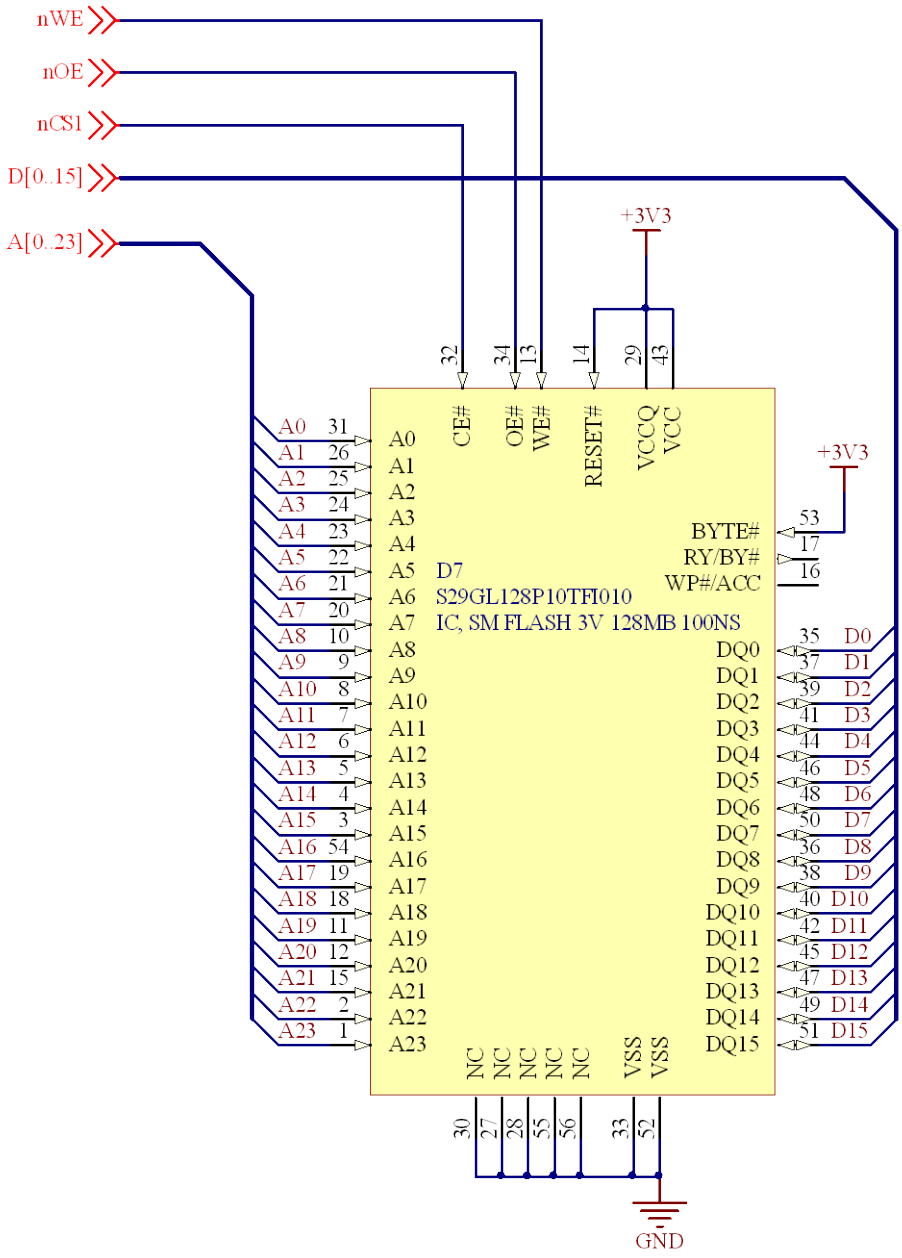
3.4. Memorije

U sustavu se nalaze tri memorije, SDRAM, 8 MB; Flash, 16 MB i 2 MB SRAM-a.

3.4.1. Flash memorija

Flash memorija koja se nalazi u sustavu je čip oznake S29GL128P10TFI010 proizvođača Spansion i pod oznakom je D7. Veličine je 16 MB, konfiguracije 8Mx16 (8M

16-bitnih riječi) ili 16Mx8 (16M 8-bitnih podataka). Deklarirano vrijeme čitanja memorije je 100 ns, vrijeme pisanja jedne riječi 60 μs a vrijeme brisanja sektora 0.5 s. Napaja se s +3.3V. Za detalje oko memorije pogledati [datasheet memorije], a za detalje oko korištenja pogledati poglavlje 4.7.3. Slika 3.9 prikazuje spoj memorije u sustav.



Slika 3.9 Spajanje flash memorije na mikrokontroler

Signal RESET# je aktivan nisko te provodi sklopovski reset memorije. Kako u sustavu ovo nije potrebno, signal RESET# je spojen na napajanje.

Priključak WP#/ACC ima dvije funkcije. Prva je sklopovska zaštita od upisa i brisanja sektora (Write Protect) kada je priključak u niskom stanju. Druga je ubrzano programiranje memorije kada je na priključak spojen napon 11.5V – 12.5V. Priključak ima interni pritezni otpornik prema napajanju, pa je ostavljen odspojen.

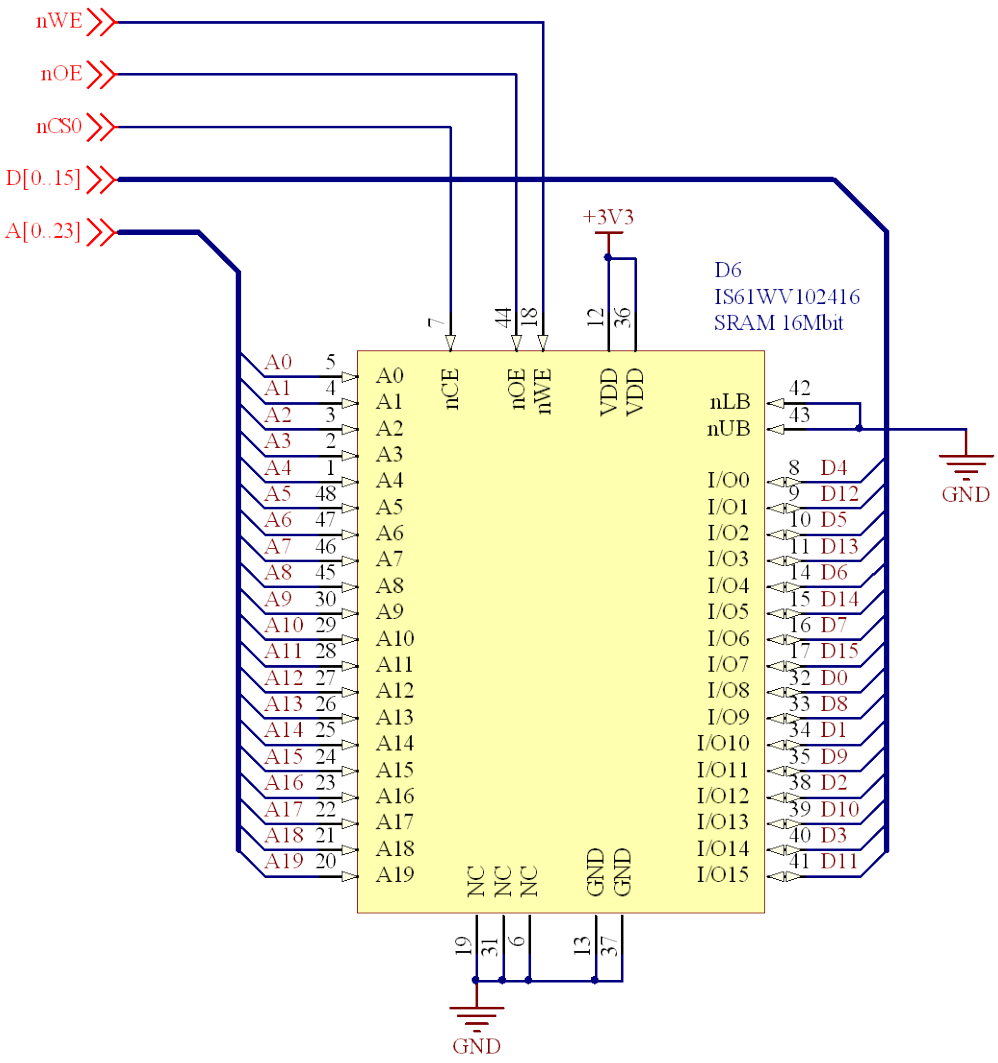
Priključak RY/BY# pokazuje da li je memorija slobodna za prihvat nove komande ili je u stanju upisa/brisanja/nečeg trećeg. Ovaj signal se ne koristi u dizajnu.

Priključak BYTE# određuje širinu podatkovne sabirnice memorije. Za BYTE# u niskom stanju, podatkovna sabirnica je 8-bitna (DQ0-DQ7), a priključak DQ15 poprima ulogu najmanje značajnog adresnog bita. Za BYTE# u visokom stanju, podatkovna sabirnica je 16-bitna što se i koristi u sustavu.

Memorije je spojena na nCS1 mikrokontrolera. Razlog ovome je što kada se mikrokontroler puni iz vanjske statičke memorije, to radi upravo u domeni nCS1. Detalji su dani u poglavlju 4.2.4.

3.4.2. SRAM memorija

SRAM memorija korištena u sustavu je oznake IS61WV102416BLL-10TLI, oznaka u sustavu je D6. Radi se o 2 MB čipu, konfiguracije 2Mx16, brzine pristupa 10ns. Memorija zahtjeva +3.3V napajanje te je u potpunosti asinkrona. Za detalje oko same memorije pogledati [referenca datasheet memorije] a za korištenje pogledati 4.7.4. Slika 3.10 prikazuje način spajanja memorije u sustavu.



Slika 3.10 Spajanje SRAM memorije u sustav

Memorija je spojena na nCS0 mikrokontrolera. Signali koje je potrebno prokomentirati su nLB i nUB te podatkovna sabirnica. Signali nLB i nUB omogućuju pojedine poluriječi na sabirnici te su aktivni nisko. Kako u sustavu nema potrebe za maskiranjem pojedinih bajtova, ovi signali su spojeni tako da su gornji i donji bajt cijelo vrijeme aktivni.

Ranije je spomenuto da se u sustavu nalaze tri memorije te da je sustav izveden na dva sloja. Kako bi bilo uopće moguće projektirati tiskane veze, odnosno, spojiti sve signalne linije, neke su stvari morale biti izvedene mimo specifikacija. Jedna od tih je podatkovna sabirnica SRAM-a. Ako se na gornjoj shemi malo bolje pogleda podatkovna sabirnica, vidljivo je da se priključci i linije na sabirnici ne poklapaju. Ovo nije greška na

shemi, već je memorija zaista tako spojena. Ipak, ovo ni najmanje ne utječe na sustav niti na neke aspekte njegova ponašanja, podaci se uvijek jednako zapisuju i čitaju iz memorije te je iz programskog pogleda memorija najnormalnije spojena. Tablica 3.2 prikazuje priključke na memoriji i na njih spojene priključke mikrokontrolera.

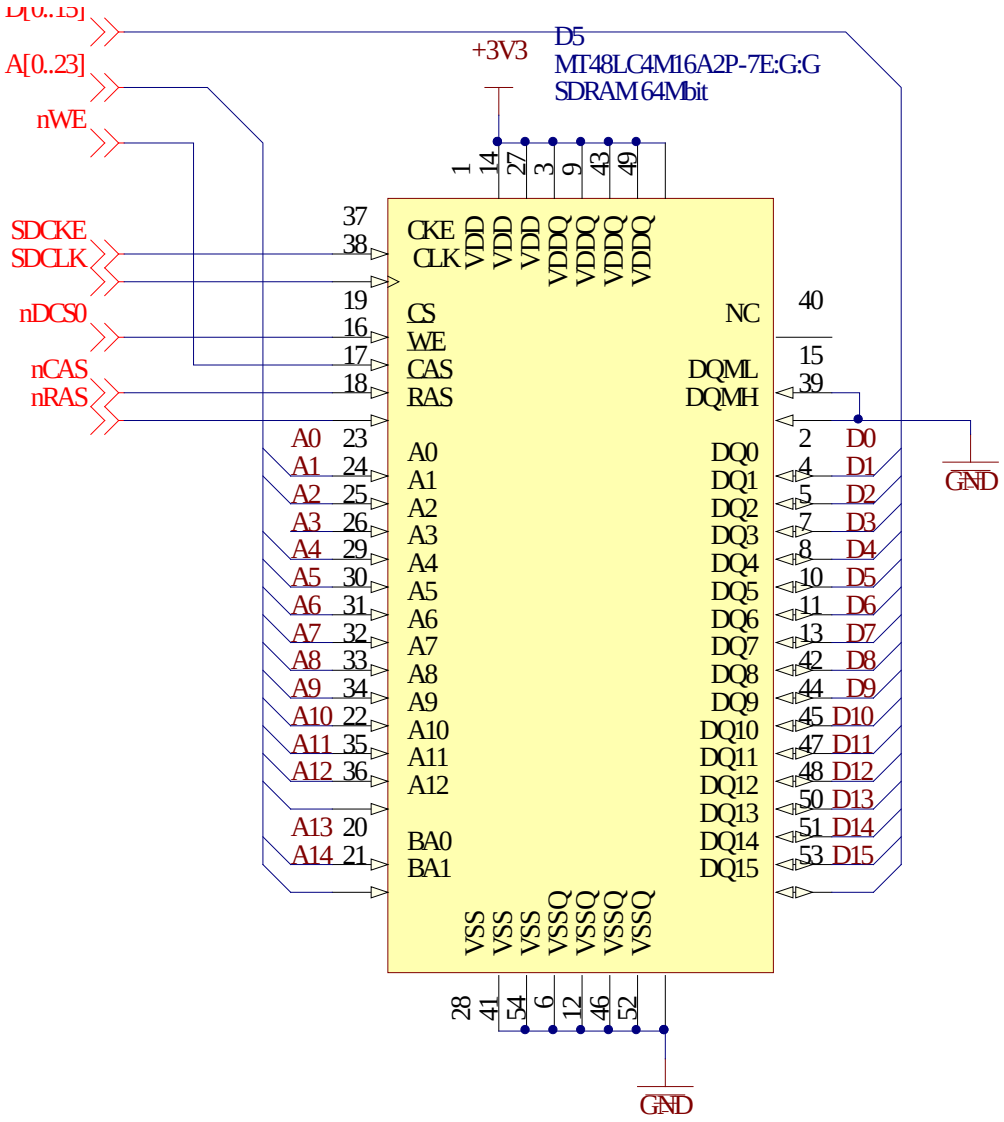
Tablica 3.2 Permutacija priključaka podatkovne sabirnice SRAM memorije

SRAM	LH79525
IO0	D4
IO1	D12
IO2	D5
IO3	D13
IO4	D6
IO5	D14
IO6	D7
IO7	D15

SRAM	LH79525
IO8	D0
IO9	D8
IO10	D1
IO11	D9
IO12	D2
IO13	D10
IO14	D3
IO15	D11

3.4.3. SDRAM memorija

Korištena memorija je čip oznake MT48LC4M16A2P-7E:G:G proizvođača Micron. Oznaka u sustavu je D5. Vrijeme pristupa memoriji je 5.4 ns a konfiguracija je 1Mx16x4 (1M 16 bitnih riječi u svakoj od 4 banke). Napon napajanja je +3.3V. Spoj memorije u sustav je prikazan slikom Slika 3.11



Slika 3.11 Spoj SDRAM memorije u sustav

SDRAM memorija je spojena preko tipičnog sučelja kako se spajaju SDRAM memorije, a to je preko signala RAS, CAS, CLK, CKE i WE. Prilikom spajanja i projektiranja tiskanih vodova posebno se vodilo računa o količini mase ispod čipa te o tome da tiskani vodovi budu što kraći. Ipak, SDRAM memorija posjeduje određene probleme u radu koji su diskutirani u poglavlju 3.4.4. Za detalje oko samog čipa pogledati [ref datashhet], a za upute za rad pogledati 4.7.1.

3.4.4. Problemi sa SDRAM memorijom

U poglavlju 4.7.1.1 je dano objašnjenje rada s SDRAM memorijom. Mikrokontroler uvijek pristupa memoriji putem međuspremnika veličine 8x16 bita. Ovo nema utjecaja na sustav dok god se SDRAM čita ili dok se upisuje na kontinuirane adrese. Međutim, ako se upiše podatak na adresu n i potom se forsira ispražnjavanje spremnika (primjerice drugim upisom na adresu n + 1000), na adresu n će se upisati željeni podatak, ali u memoriju se upisuje sadržaj cijelog međuspremnika što znači da će okolne memorijske lokacije biti pregažene sadržajem međuspremnika. Razlog ovome je taj što signali DQM0 i DQM1 nisu dovedeni s mikrokontrolera na memoriju. Zadaća ovih signala je maskiranje podatkovnih linija. Kako su u radu ovi signali na memoriji spojeni na masu, podatkovne linije su aktivne cijelo vrijeme te prilikom upisa stanja međuspremnika upisuje se i stanje neželjenih memorijskih lokacija koje bi inače bilo maskirano da su signali DQMx spojeni na memoriju.

U možebitnoj sljedećoj reviziji uređaja, ove signale je potrebno obavezno dovesti do memorije dok je u ovoj problem rješen dovođenjem signala DQMx žicama do memorije.

Još jedan problem sa SDRAM-om koji se tiče cijele memorijske sabirnice je i preslušavanje o čemu ima više riječi u poglavlju 3.8.

3.5. Spoj ethernet trancievera

Mikrokontroler posjeduje ethernet kontroler, međutim, kako bi se komunikacija mogla stvarno implementirati izvana je potrebno spojiti čip koji vrši sučeljavanje sa stvarnim fizičkim medijem. Analogija je jednaka onoj i za UART, da bi se UART mogao koristiti potrebno je izvana dodati MAX232 koji će TTL/LVTTL signale pretvoriti u one definirane EIA-232 standardom.

U radu je korišten čip KS8001L proizvođača Micrel koji podržava fizičke slojeve 10BASE-T/100BASE-TX/FX. Za detalje oko čipa pogledati [ref datasheet od KS800L], a za detalje oko korištenja etherneta pogledati poglavlje 4.11.

Slika [] prikazuje položaj trancievera u sustavu zajedno s podržavajućim sklopovljem i konektorom, električna shema je dana u prilogu.

Slika 3.12 Ethernet tranciever zajedno s podržavajućim sklopovljem

3.6. Konektori i periferija

U ovom poglavlju će biti dani i objašnjeni spojevi konektora i perifernih jedinica. Prije toga, potrebno je napomenuti da su svi ulazno-izlazno priključci TTL kompatibilni. Također, svi ulazno-izlazni priključci mogu davat struju do 8 mA, osim priključka SDCLK koji može dati do 12 mA.

3.6.1. LCD konektori

Pod LCD konektore spadaju konektori X209 i X210. Njihova pozicija kao i numerijacija priključaka dana je slikom.

Slika 3.13 Pozicija priključaka X209 i X210 i numeracija pojedinih priključaka

Kada se ne koriste kao LCD konektori, konektori X209 i X210 se mogu koristiti kao općenamjenski ulazno-izlazni priključci. Tablica 3.3 prikazuje odgovarajuće ulazno-izlazne priključke mikrokontrolera i njihovu poziciju na konektoru X209. Tablica 3.4 daje isti prikaz za konektor X210.

Tablica 3.3 Funkcije priključaka na konektoru X209

X209	Funkcija
1	GND
2	LCDCLK/PE1
3	LCDLP/PE0
4	LCDFP/PF7
5	GND
6	NC
7	LCDVD0/PG2
8	LCDVD1/PG3
9	LCDVD2/PG4
10	LCDVD3/PG5

X209	Funkcija
11	NC
12	GND
13	NC
14	LCDVD4/PG6
15	LCDVD5/PG7
16	LCDVD6/PF0
17	LCDVD7/PF1
18	NC
19	GND
20	NC

X209	Funkcija
21	LCDVD8/PF2
22	LCDVD9/PF3
23	LCDVD10/PF4
24	LCDVD11/PF5
25	NC
26	GND
27	LCDEN/PF6
28	+3V3
29	+3V3
30	NC

Tablica 3.4 Funkcija priključaka na konektoru X210

X210	Funkcija		X210	Funkcija

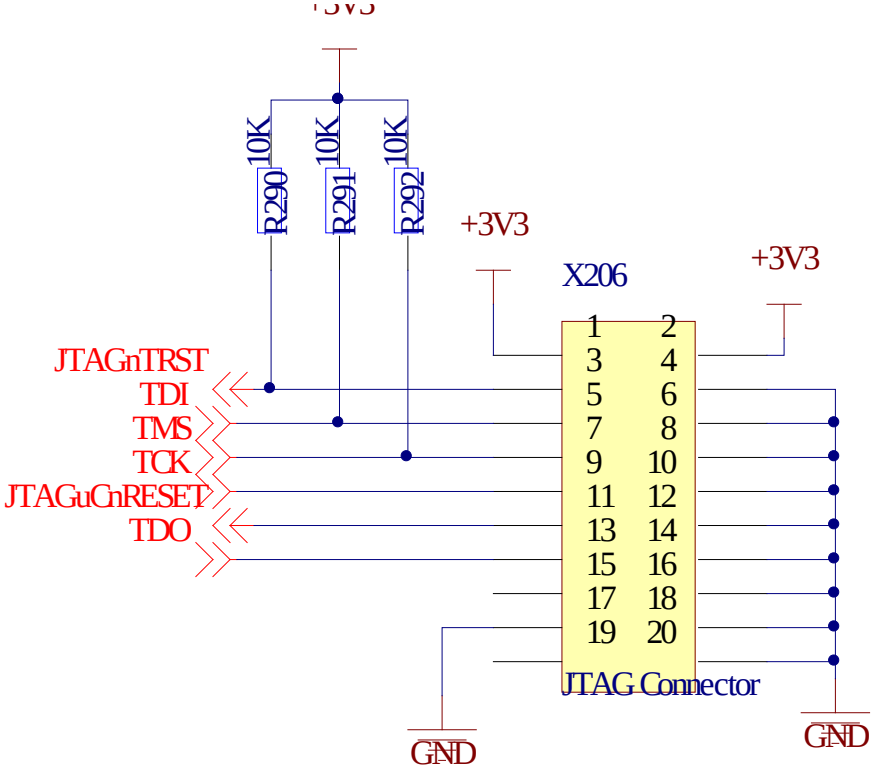
1	LCDVEEN/PE6		6	GND
2	LCDVDDEN/PE5		7	+3V3
3	LCDDSPLEN/PE4		8	+3V3
4	LCDCLS/PE3		9	+5V
5	LCDPS/PE2		10	+5V

NAPOMENA: priključci konektora X210 su multipleksirani s LED-icama H200:H204, a priključci konektora X209 s VGA konektorom X200. Za detalje i druge multipleksirane priključke pogledati poglavlje 3.7.

3.6.2. JTAG konektor

JTAG konektor je standardni 20-pinski JTAG konektor za ARM procesorske jezgre. Oznaka u sustavu mu je X206. Slika i funkcija priključaka su dani slikom Slika 3.14. Shema konektora je dana slikom Slika 3.15.

Slika 3.14 JTAG konektor zajedno s funkcijom pojedinih priključaka



Slika 3.15 Električna shema spoja JTAG konektora

Signali TDI, TMS, TCK, TDO i JTAGnTRST su spojeni na odgovarajuće priključke mikrokontrolera. Signal JTAGuCnRESET, koji je namjenjen resetiranju sustava (s pogleda JTAG kabela) je ostavljen odspojen. NAPOMENA: na liniju TCK nije spojen pritezni otpornik prema napajanju.

3.6.3. Konektor X207

Konektor X207 sadrži ulazno-izlazni priključak PA, dva kanala AD pretvornika, vanjski takt za vremenske sklopove te napone +3.3V i +5V. Slika 3.16 prikazuje položaj u sustavu i numeraciju priključaka. Tablica 3.5 prikazuje funkcije pojedinih priključaka.

Slika 3.16 Konektor X207 i numeracija priključaka

Tablica 3.5 Funkcije pojedinih priključaka konektora X207

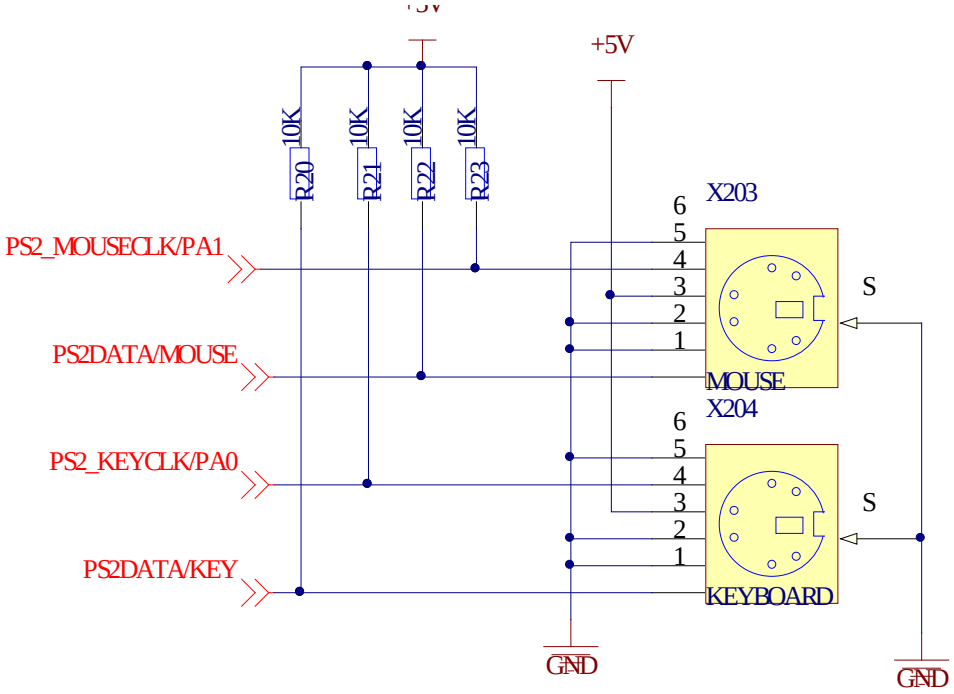
X209	Funkcija		X209	Funkcija		X209	Funkcija
1	+5V		6	AN0		11	PA2
2	+3V3		7	PA6		12	PA3
3	GND		8	PA7		13	PA0/PS2_KEYCLK
4	AN1		9	PA4		14	PA1/PS2_MOUSECLK
5	CTCLK/INT4		10	PA5			

Napomena: Priključci PA0/PS2_KEYCLK i PA1/PS2_MOUSECLK su multipleksirani sa signalima takta za PS/2 konektora. Pregled svih multipleksiranih priključaka je dan u poglavlju 3.7.

3.6.4. PS/2 konektori

Konektori X203 i X204 su PS/2 konektori. Slika 3.17 prikazuje položaj u sustavu, a Slika 3.18 shemu spajanja.

Slika 3.17 Položaj PS/2 konektora u sustavu



Slika 3.18 Shema spajanja PS/2 konektora

Priključci PS2_KEYCLK/PA0 i PS2_MOUSECLK/PA1 su multipleksirani s priključcima porta PA. Spoj podatkovnih linija dat je u nastavku:

- PS2_DATA/MOUSE – PB0
- PS2_DATA/KEY – PB1

Ovdje je još jednom potrebno napomenuti da su ulazno-izlazni priključci mikrokontrolera TTL kompatibilni.

3.6.5. Konektor X212

Konektor X212 sadrži tri izlazna priključka porta PM (port PM posjeduje samo izlazne priključke). Slika 3.19 prikazuje položaj u sustavu i numeraciju. Tablica 3.6 prikazuje funkcije pojedinih priključaka.

Slika 3.19 Položaj X212 konektora i numeracija priključaka

Tablica 3.6 Funkcije priključaka konektora X212

X212	Funkcija		X212	Funkcija
1	GND		3	PM4

2	PM3		4	PM5

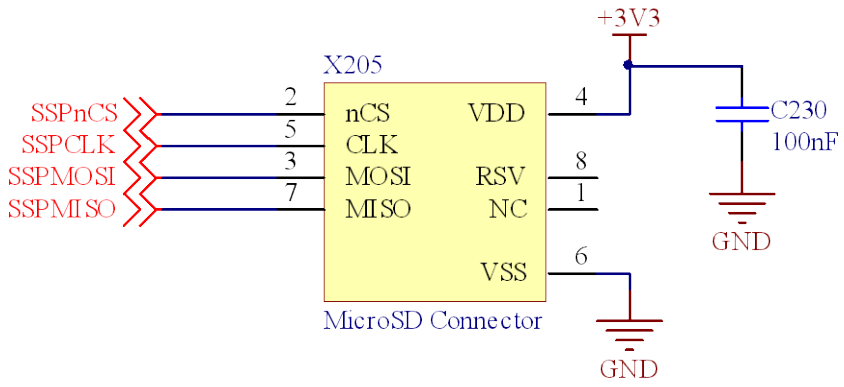
3.6.6. Konektor X208

Konektor X208 sadrži dva priključka, nRESETOUT i CLKOUT. Oba priključka mogu davati struju do 8 mA. Izlazni takt na priključku CLKOUT može biti sustavski takt, HCLK i FCLK. Za detalje oko taktova, pogledati poglavlje 4.2.2 te poglavlje 13.1.3 u [ref user manual]. Slika 3.20 prikazuje položaj konektora i funkciju priključaka.

Slika 3.20 Konektor X208 i pripadajuće funkcije priključaka

3.6.7. Konektor X205 - µSD konektor

µSD konektor je spojen u SPI načinu na priključke mikrokontrolera kojima je jedna od sekundarnih funkcija SPI kontroler. Konektor se nalazi odmah poviše ulaza napajanja. Slika 3.21 prikazuje shemu spajanja konektora.



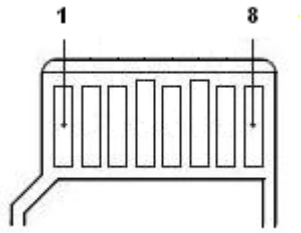
Slika 3.21 Shema spoja µSD konektora

Tablica 3.7 prikazuje funkcije pojedinih priključaka konektora.

Tablica 3.7 Funkcije pojedinih priključaka µSD konektora

X205	Funkcija		X205	Funkcija		X205	Funkcija
1	NC		4	+3V3		7	Data out – PB4
2	nCS – PB2		5	CLK – PB3		8	NC
3	Data in – PB5		6	GND			

Slika 3.22 prikazuje priključke i njihov redoslijed na µSD kartici. Tablica 3.8 prikazuje funkcije pojedinih priključaka.



Slika 3.22 Priključci µSD kartice

Tablica 3.8 Funkcije priključaka µSD kartice u SD i SPI načinu

Način	**SD način**			**SPI način**		
Priključak	Ime	Tip	Opis	Ime	Tip	Opis
1	DAT2	U/I/PP	Podatkovna linija	Rez	-	-
2	CD/DAT3	U/I/PP	Detektiranje kartice/podatkovna linija	CS	U	Chip select
3	CMD	PP	Odgovor na komandu	DI	U	Ulazna podatkovna linija
4	VDD	N	Napajanje	VDD	N	Napajanje
5	CLK	U/I	Takt	SCLK	U	Takt
6	VSS	N	Napajanje	VSS	N	Napajanje
7	DAT0	U/I/PP	Podatkovna linija	DO	I	Izlazna podatkovna linija
8	DAT1	U/I/PP	Podatkovna linija	Rez	-	-

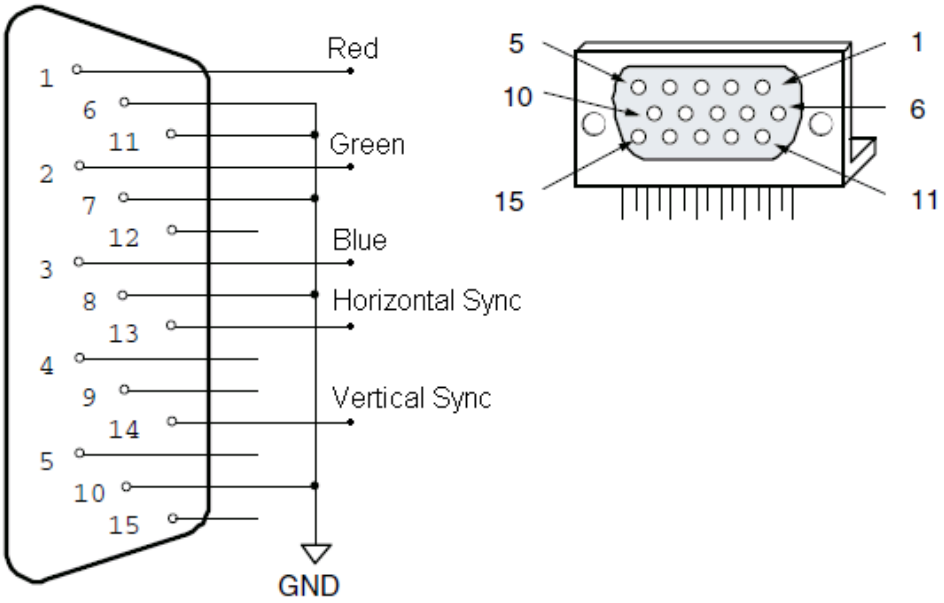
U – Ulazni priključak
I – Izlazni priključak
PP – Push-Pull priključak
N – Priključak napajanja
Rez – Rezervirano

3.6.8. Konektor X200 – VGA konektor

Iako mikrokontroler nema poseban VGA izlaz ono je ostvareno preko LCD sučelja odnosno, LCD kontrolera. VGA protokol je razmjerno jednostavan, sastoji se od 2 kontrolna signala (HSYNC, VSYNC) i tri analogna signala koja određuju intenzitet boje (R, G i B). Iscrtavanje na ekranu počinje u gornjem lijevom kutu (piksel (0, 0)) te se iscrtavanje ekrana naziva jedan *frame.* Početak iscrtavanja *framea* je označen bridom na signalu VSYNC. Početak iscrtavanja novog reda je označen bridom na signalu HSYNC. Standardna frekvencija iscrtavanja cijelog ekrana je 60 Hz što uz 640*480 piksela, koliko je definirano VGA standardom, čini približno 25 MHz frekvenciju takta promjene piksela. Vrijednost piksela je u svakom trenu određena stanjem na R, G i B

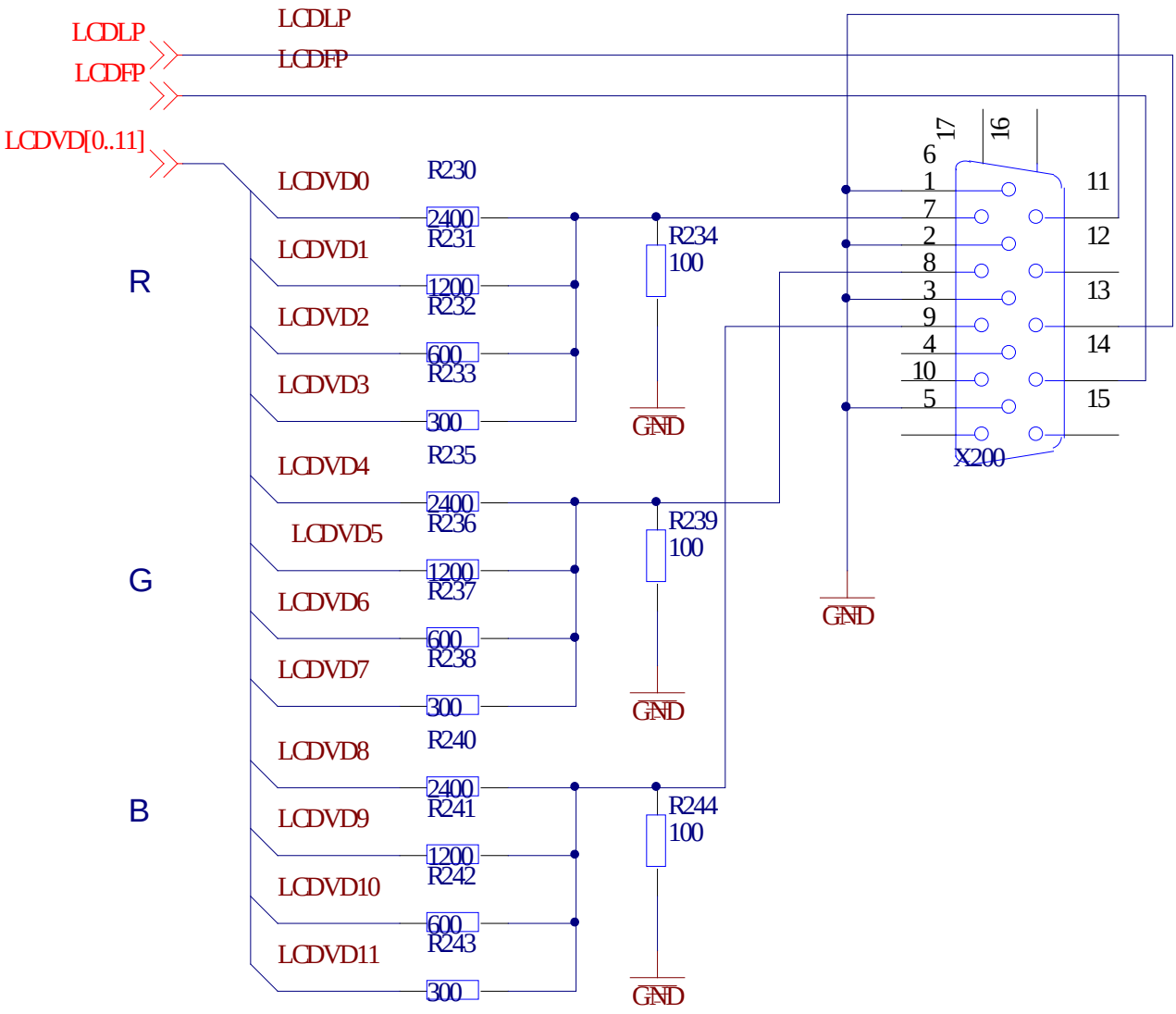
liniji na kojima se nalaze naponske razine između 0 i 0.7 V. Točna razina napona određuje intenzitet komponente boje. Osim spomenutih značajki, VGA standard propisuje i zaključenje linija u iznosu 75Ω.

Slika 3.23 prikazuje raspored priključaka u VGA konektoru.



Slika 3.23 Funkcija, raspored priključaka i način spajanja VGA konektora

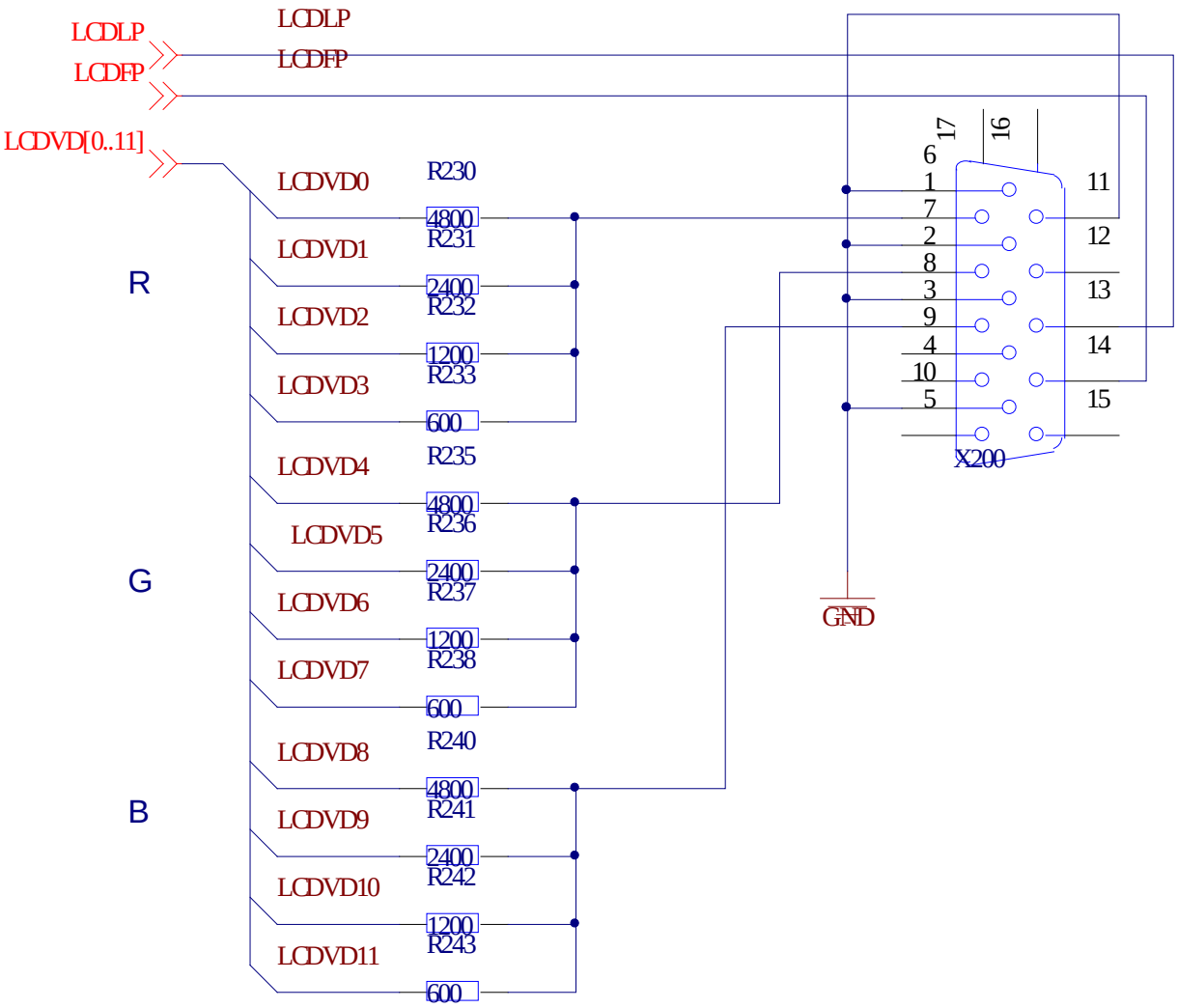
Kod LCD ekrana TFT tipa, situacija varira od ekrana do ekrana no sam protol je vrlo sličan. Vrijede ista pravila za HSYNC i VSYNC a razlika je, osim u brzinama, u tome što se boje prenose digitalnim paralelnim linijama. Primjerice, kod korištenog mikrokontrolera svaka boja ima 4 bita odnosno 4 linije. LCD-ovi također zahtijevaju i složenije sučelje u vidu dodatnih signala kao što su ENABLE, PIXEL-CLOCK signala i još nekih drugih, no to trenutno nije bitno. Ono što je važno uočiti jest to da postoji digitalno sučelje koje je moguće prilagoditi VGA sučelju. Ideja je sljedeća, paralelne linije putem kojih se prenosi informacija o boji putem DA pretvornika pretvoriti u odgovarajuće razine napona, a signale HSYNC i VSYNC samo proslijediti. Slika 3.24 prikazuje sklopovlje kojim je ovo ostvareno.



Slika 3.24 Pogonsko sklopovlje VGA konektora

Sučelje je ostvareno putem jednostavnog R-2R DA pretvornika. Otpornici R234, R239 i R244 osiguravaju zaključenje linije u iznosu 75Ω.

Za vrijeme projektiranja ovog sklopovlja počinjena je greška, naime ovo sklopovlje ispravno radi te ima ispravno zaključenje no problem je da vuče previše struje iz priključaka mikrokontrolera. U skladu s maksimalnim mogućim opterećenjem, projektirano je novo sučelje prikazano slikom Slika 3.25. Potrebno je napomenuti da je greška primjećena tek nakon izrade tiskane pločice.



Slika 3.25 Druga verzija pogonskog sklopovlja VGA konektora

U ovoj verziji su zadržane naponske razine, no zaključenje više nije 75Ω. Teoretski, ako je zaključenje sa strane monitora dobro, loše zaključenje sa strane VGA konektora neće utjecat na rad sustava.

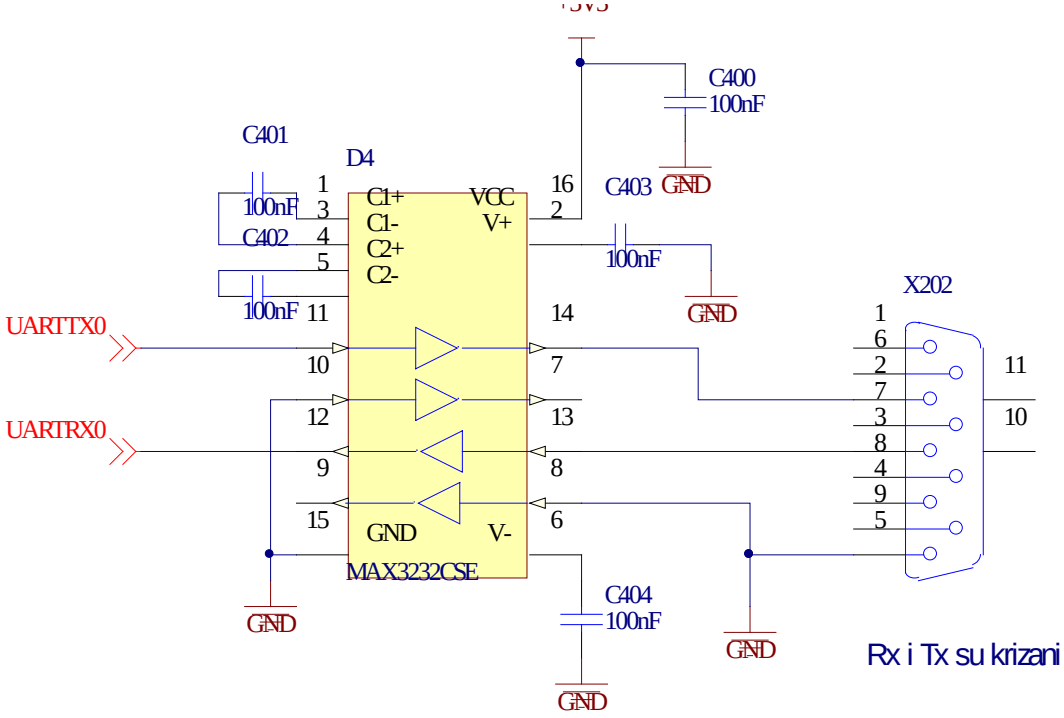
Napomena: Priključci konektora X200 su multipleksirani s priključcima konektora X209. Za detalje i pregled svih multipleksiranih priključaka pogledati poglavlje 3.7.

U slučaju revizije dizajna, savjetuje se ugraditi pojačalo, na tržištu se mogu naći pojačala predviđena upravo za ovu namjenu s minimalnim brojem priključaka i veličine kućišta kao što je primjerice sklop TSH344ID, proizvođača STMicroelectronics.

3.6.9. Konektor X202 i pogonsko sklopovlje

Na konektoru X202 se nalazi DB-9 muški konektor, odnosno, UART sučelje. Za pogonsko sklopovlje sučelja je iskorišten pogonski sklop MAX3232 proizvođača Texas Instruments [ref datasheet]. Ono što je zanimljivo u vezi sklopa je da radi na naponu napajanja od 3.3V. Također, u dizajnu nisu korišteni elektrolitski već keramički kondenzatori. Slika [] prikazuje položaj konektora, a slika Slika 3.27 predstavlja shemu spajanja konektora i pogonskog sklopa. Pogonski sklop se nalazi na lemnoj strani.

Slika 3.26 položaj DB9 konektora u sustavu



Slika 3.27 Električka shema spajanja UART sučelja i pripadnog pogonskog sklopa

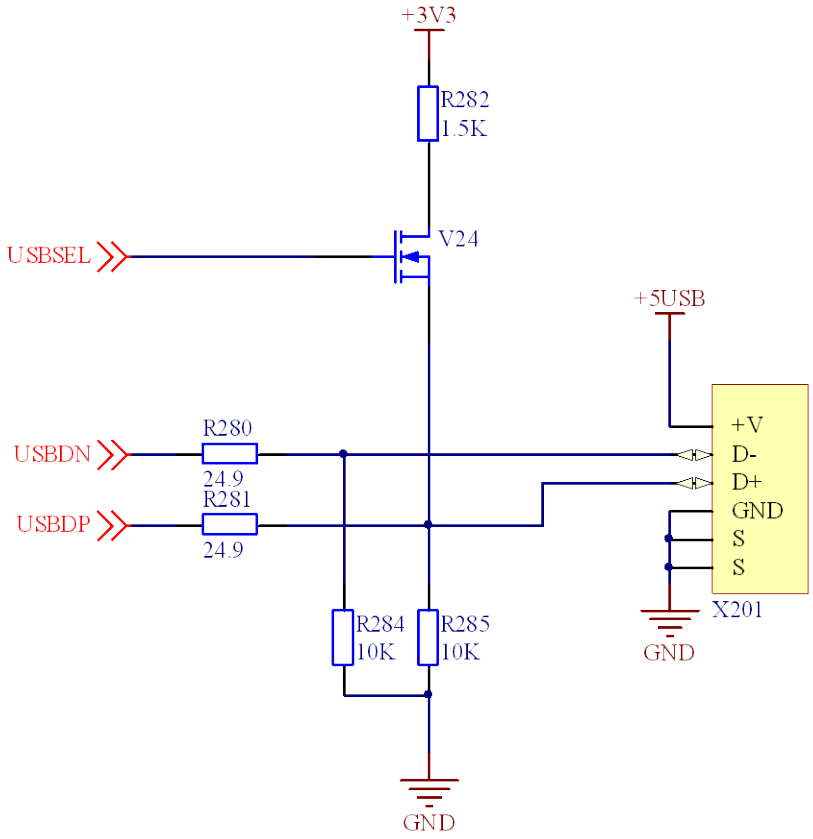
Za slučaj da sustav komunicira s DTE uređajem (PC), koristi se ne-križani kabel. Priključci UARTTX0 i UARTRX0 su spojeni na PB7 i PB6, respektivno.

3.6.10. Konektor X201 – USB konektor

Slika 3.28 prikazuje položaj USB konektora.

Slika 3.28 USB (tip B) konektor u sustavu

Slika 3.29 prikazuje shemu spajanja, a Tablica 3.9 prikazuje spoj pojedinih priključaka na mikrokontroler.



Slika 3.29 Električka shema spoja USB konektora

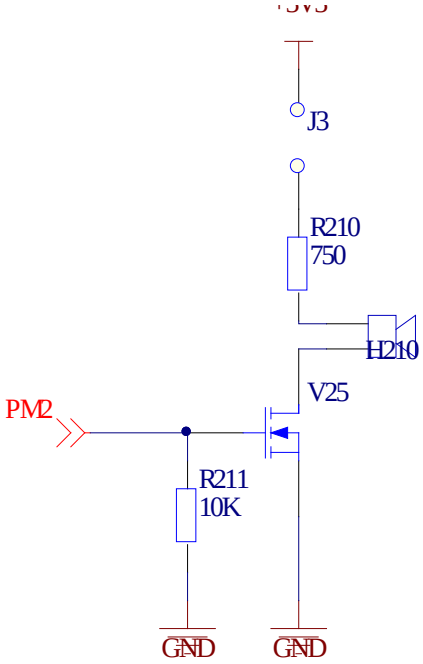
Tablica 3.9 Spoj priključaka USB konektora i pripadnog pogonskog sklopovlja na mikrokontroler

Priključak	LH79525
USBSEL	PE7
USBDN	USBDN
USBDP	USBDP

3.6.11. Zujalica

Slika 3.30 prikazuje položaj zujalice u sustavu, a Slika 3.31 električku shemu spoja.

Slika 3.30 Položaj zujalice



Slika 3.31 Električka shema spoja zujalice

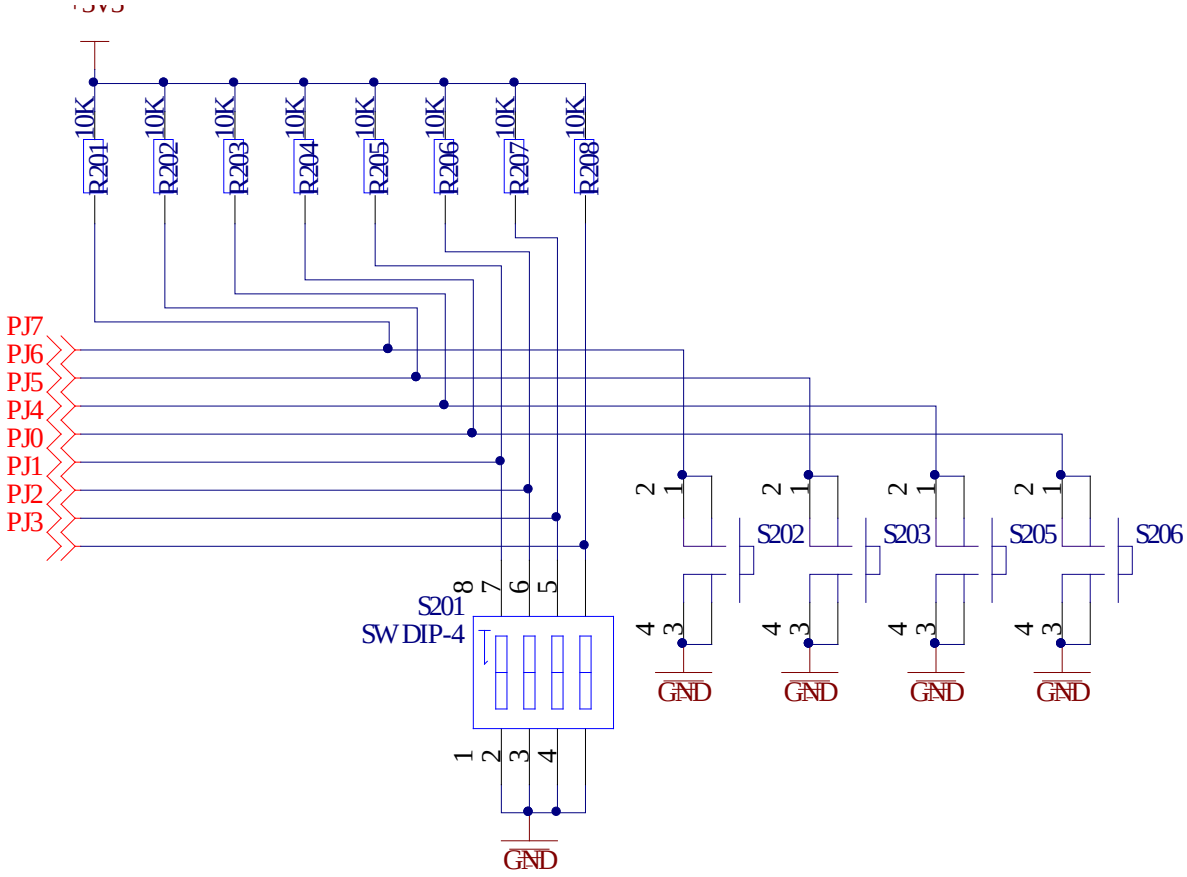
Pritezni otpornik R211 služi za definiranje početnog stanja priključka PM2 kada mikrokontroler još uvijek ne pogoni spomenuti priključak. Premosnik J3 omogućuje mehaničko gašenje zujalice u slučaju potrebe, ako je spojen, zujalica je omogućena, inače nije. Iako je stanje priključka po resetu uzeto u obzir, te osiguravanje inicijalnog stanja kroz otpornik R211, zujalica pišti po resetu mikrokontrolera.

Rad sa zujalicom je jednostavan, kada je priključak PM2 u visokom stanju, ona će pištat, inače neće.

3.6.12. Sklopke i prekidači

Slika 3.32 prikazuje sklopke i prekidače, a Slika 3.33 prikazuje električnu shemu spajanja.

Slika 3.32 Sklopke i prekidači u sustavu



Slika 3.33 Električka shema spoja tipki i prekidača

Sve sklopke i prekidači su aktivni u nuli. Tablica 3.10 prikazuje spoj prekidača i tipki na mikrokontroler.

Tablica 3.10 Spoj prekidača i tipki na mikrokontroler

Sklopka/Prekidač	LH79525		Sklopka/Prekidač	LH79525
S201 – 1	PJ0		S202	PJ7
S201 – 2	PJ1		S203	PJ6
S201 – 3	PJ2		S204	PJ5
S201 – 4	PJ3		S205	PJ4

3.6.13. LE diode

Slika 3.34 prikazuje položaj LE dioda u sustavu i njihovu numeraciju, a Slika 3.34 položaj i numeracija LE dioda u sustavu

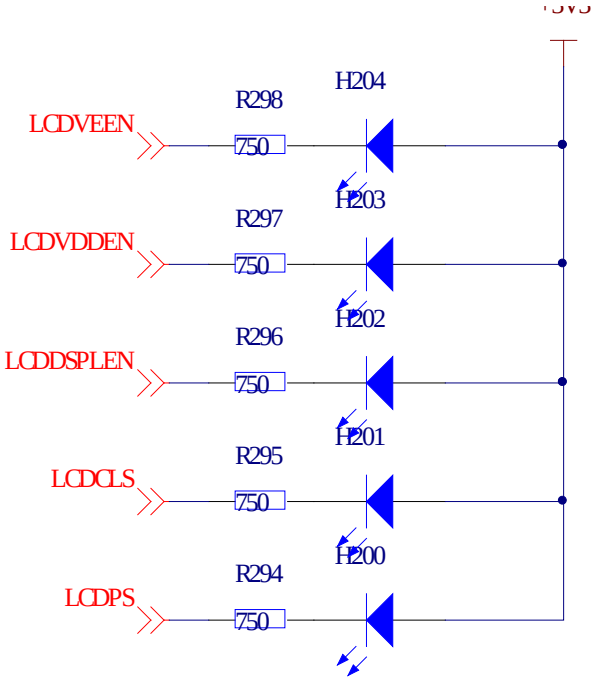
Tablica 3.11 spoj dioda na mikrokontroler.

Slika 3.34 položaj i numeracija LE dioda u sustavu

Tablica 3.11 Spoj LE dioda na mikrokontroler

LE dioda	LH79525
H200	LCDPS/PE2
H201	LCDCLS/PE3
H202	LCDSPLEN/PE4
H203	LCDVDDEN/PE5
H204	LCDVEEN/PE6

Slika 3.35 prikazuje električki spoj dioda u sustavu.



Slika 3.35 Električki spoj LE dioda

LE diode su aktivne nisko. Napomena: LE diode su multipleksirane s priključcima konektora X210. Za detalje i pregled svih multipleksiranih priključaka pogledati poglavlje 3.7.

3.7. Multipleksirani priključci

Tablica 3.12 sadrži popis multipleksiranih priključaka.

Tablica 3.12 Popis multipleksiranih priključaka u sustavu

Priključak		Multipleksiran sa	
Priključak	Konektor	Priključak	Konektor
LCDLP/PE0	X209	LCDLP/PE0	X200
LCDFP/PF7	X209	LCDFP/PF7	X200
LCDVD0/PG2	X209	LCDVD0/PG2	X200
LCDVD1/PG3	X209	LCDVD1/PG3	X200
LCDVD2/PG4	X209	LCDVD2/PG4	X200
LCDVD3/PG5	X209	LCDVD3/PG5	X200
LCDVD4/PG6	X209	LCDVD4/PG6	X200
LCDVD5/PG7	X209	LCDVD5/PG7	X200
LCDVD6/PF0	X209	LCDVD6/PF0	X200
LCDVD7/PF1	X209	LCDVD7/PF1	X200
LCDVD8/PF2	X209	LCDVD8/PF2	X200
LCDVD9/PF3	X209	LCDVD9/PF3	X200
LCDVD10/PF4	X209	LCDVD10/PF4	X200
LCDVD11/PF5	X209	LCDVD11/PF5	X200
LCDVEEN/PE6	X210	LCDVEEN/PE6	H204
LCDVDDEN/PE5	X210	LCDVDDEN/PE5	H203
LCDDSPLEN/PE4	X210	LCDDSPLEN/PE4	H203
LCDCLS/PE3	X210	LCDCLS/PE3	H202
LCDPS/PE2	X210	LCDPS/PE2	H201
PS2_MOUSECLK/PA1	X203	PS2_MOUSECLK/PA1	X207
PS2_KEYCLK/PA0	X204	PS2_KEYCLK/PA0	X207

3.8. Greške počinjene u dizajnu i ostali problemi sa sustavom

Otkrivene greške u dizajnu su:

- Na priključak TCK JTAG konektora nije spojen pritezni otpornik na napajanje, niti je predviđen za vrijeme projektiranja.
- Prilikom dizajniranja VGA sučelja nije uzeta u obzir maksimalna struja koju priključak mikrokontrolera može dati, detalji u **Error! Reference source not found.**.
- Svaki MOSFET u sustavu ima pogrešan *footprint* zbog čega je svaki prilikom lemljenja zarotiran.
- R300 odnosno je pogrešnog iznosa, umjesto 6K64 spojeno je 6K8.

- DQM linije na SDRAM memoriji su morale biti spojene, detaljnije u poglavlju 3.4.4.

U dizajnu je počinjena još jedna greška, a ta je da nije bilo potrebe za konfiguracijskim sklopovljem koje određuje stanja PC[7:4] po resetu. Kako će biti pokazano u poglavlju 4.2.4 opcije potrebne sustavu su 0b0000 ili 0b1111. Jasno je da se ovako nešto moglo riješiti samo jednim prekidačem ili premosnikom.

3.8.1. Problemi s preslušavanjem na memorijskoj sabirnici

Iako je puno truda uloženo u projektiranje tiskanih veza memorijske sabirnice, sabirnica je neispravno radila. Glavni cilj prilikom projektiranja je bio što kraće signalne linije i što više mase ispod komponenti i tiskanih vodova. Ipak, u nekim područjima na tiskanoj pločici nije bilo moguće slijediti ove ciljeve. Slika 3.36 prikazuje lemnu stranu tiskane pločice problematično područje.

Slika 3.36 Lemna strana tiskane pločice

Kako bi se poboljšao integritet signala, preko problematičnog područja je dodana velika ploha spojena na masu. Slika 3.37 prikazuje lemnu stranu tiskane pločice s dodanom plohom spojenom na masu.

Slika 3.37 Lemna strana tiskane pločice – sa plohom mase

3.9. Smjernice za daljnji razvoj

U slučaju daljnjih revizija, za početak je potrebno popraviti greške dane u poglavlju 3.8. Ono čemu je potrebno posvetiti najviše pozornosti je isplativost projektiranja ovakvog sustava na dvoslojne tiskane pločice. U radu je pokazano da projektiranjana dva sloja vodi k problemima prilikom uhodavanja sustava te da su potrebni dodatni zahvati u sklopovlje kako bi komponente proradile. Preslušavanja, zrcalne impedancije i ostale pojave su realnost kojih treba biti itekako svjestan prilikom projektiranja sustava.

U slučaju revizije na dvoslojnoj tiskanoj pločici, potrebno je pokušati prilagoditi vodove tako da uz svaki vod ide linija mase i/ili da na suprotnoj strani vodu bude također masa. Pošto je mala vjerojatnost da se na ovakav način uspiju komponente ožičit, uvijek se mogu razmotriti metode pokazane u poglavlju 3.8.1.

Za slučaj revizije na višeslojnoj tiskanoj pločici, smatra se da se uvjeti na projektiranje tiskanih vodova iz prethodnog paragrafa mogu zadovoljiti. U tom slučaju je potrebno dodatno razmotriti zaključenja linija. Problem s mikrokontrolerom je taj što generira brze bridove (SDCLK izmjereno 2 ns).

U bilo kojem slučaju, potrebno je razmotriti i analognu stranu sustava. Linije moraju biti što kraće, što bliže masi te linije iste funkcije bi trebalo projektirati na način da su jedna pored druge i jednakih duljina.

4. Programska podrška sustavu

U ovom poglavlju će biti dane upute za izradu programske podrške sustavu kao i programski pogleda na sustav. Prvo slijedi programski pogled na sam mikrokontroler, dakle, što je potrebno za početak rada, omogućavanje takta te cijeli proces inicijalizacije. Potom slijedi rad s periferijom i na kraju pregled i upute za razvijenu programsku podršku.

4.1. Struktura direktorija

Uz rad dolazi i direktorij sa primjerima i razvijenim funkcijama. [].

4.2. Mikrokontroler LH79525 – programski pogled

U ovom poglavlju će biti dan programski pogled na mikrokontroler. Prije čitanja ovog poglavlje preporuča se pročitati poglavlja 2 i 3. Prije početka rada s bilo kojim mikrokontrolerom i sustavom u kome se on nalazi, potrebno se upoznati s arhitekturom mikrokontrolera, napajanjem, taktom (inicijalizacija, vrste, brzina) te memorijskom mapom.

4.2.1. Napajanje mikrokontrolera

S programskog pogleda ne postoje zahtjevi na napajanje, ovdje će tek biti napomenuto da se prilikom korištenja sustava OBAVEZNO mora koristiti kabel s ugrađenim osiguračem u + liniji. Naprednije opcije napajanja te načina distribucije napajanja se mogu naći u poglavlju 13, [ref velki].

4.2.2. Takt i inicijalizacija

LH79525 posjeduje više opcija razvođenja takta po sustavu, kao i veći broj različitih taktova. Postoje dva vanjska izvora takta: jedan je 11.2896 MHz (Y200), dok je drugi 32.768 KHz (Y201). Prvi, Y200, se koristi kao izvor sustavskog takt, dok se drugi koristi kao izvor takta za generator stvarnog vremena (RTC – *Real Time Clock*). Tablica 4.1 prikazuje sve postojeće taktove u sustavu.

Tablica 4.1 Postojeći taktovi u sustavu i njihove maksimalne vrijednosti

Takt	Frekvencija (MAX)	Opis
Takt oscilatora – *Oscillator clock* (CLK OSC)	20.0 MHz	Ulaz vanjskog, 11.2896 MHz oscilatora
Takt na izlazu iz sustavskog PLL-a – *PLL System Clock* (CLK PLL)	304.819 MHz	Izlaz iz sustavskog PLL-a te izvor takta sklopovlju za kontrolu i distribuciju takta
Takt USB-ovog PLL-a – *PLL USB Clock*	48.0 MHz	Izlaz iz USB PLL-a, ovaj je takt moguće kontrolirati unutar sklopovlja za kontrolu i distribuciju takta (RCPC kontroler). Za ispravno funkcioniranje sklopovlja, mora biti 48 MHz
Takt 32.768 kHz oscilatora – *32.768 kHz Oscillator Clock*	32.768 kHz	Vanjski, 32.768 KHz oscilator
AHB takt – *AHB Clock* (HCLK)	50.803 MHz	Jedan od bitnijih taktova u sustavu, ovaj takt izlazi iz RCPC te je izvor takta za AHB sabirnicu i mnogim jedinicama spojenima na nju. Također ima ulogu sustavskog takta
AHB brzi takt jezgre – *AHB Fast CPU Clock* (FCLK CPU)	76.205 MHz	Takt jezgre (ARM720T).
Ethernet takt – *Ethernet Clock*	50.803 MHz	Takt ethernet kontrolera, sinkron je s HCLK.
DMA takt – *DMA Clock*	50.803 MHz	Takt DMA kontrolera, sinkron s HCLK
Takt kontrolera vanjske memorije – *External Memory Controller Clock*	50.803 MHz	Takt vanjske memorije, sinkron s HCLK
Takt SSP kontrolera – *SSP Clock* (SSPCLK)	50.803 MHz	Takt SSP kontrolera. Asinkron svim drugim taktovima.
Takt LCD kontrolera - *CLCD Clock* (LCDDCLK)	50.803 MHz	Takt LCD kontrolera. Asinkron svim drugim taktovima.
Takt UART[2:0] kontrolera - *UART[2:0] Clock*	20.0 MHz	Takt za UART blok, ovaj takt je jednak taktu CLK OSC, asinkron je svim drugim taktovima
Takt generatora stvarnog vremena – *RTC Clock*	1 Hz	Takt generatora stvarnog vremena, nastaje nakon djeljenja takta *32.768 kHz Oscillator Clock* s 32768
Izlazni takt iz mikrokontrolera - CLKOUT	50.803 MHz	Takt na priključku CLKOUT. Ovaj takt može proizaći iz HCLK, FCLK i CLK OSC.

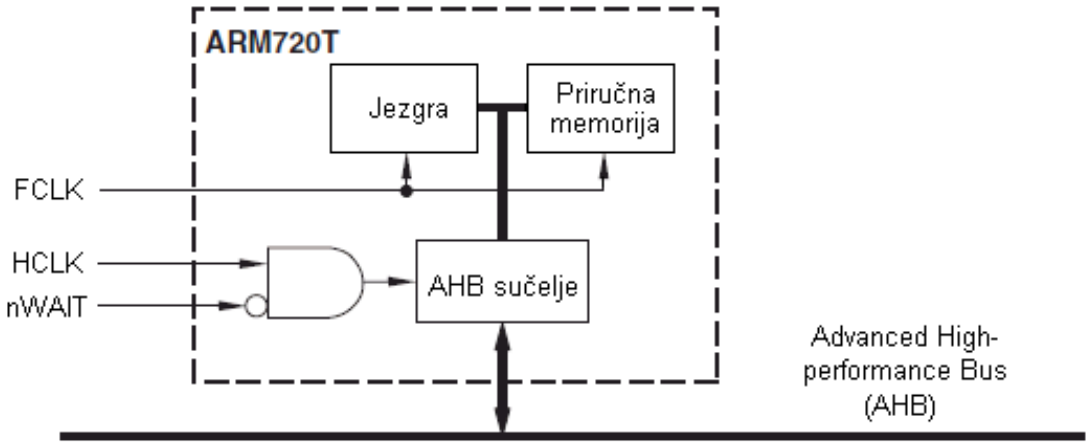
Bitni taktovi koje je u ovom trenutku potrebno primjetiti su HCLK i FCLK. Uloga ova dva takta nije jednoznačno određena, da bi se razumjela njihova uloga i približio način distribucije takta potrebno je pogledati načine distribucije takta na sabirnici.

4.2.2.1. Odnos sabirničkog i takta jezgre

Ranije je rečeno da jezgra i sabirnica mogu raditi na dva različita takta, HCLK i FCLK. Maksimalne vrijednosti su dane u tablici Tablica 4.1. Razlikuju se tri načina rada: Standardni koji se dijeli na sinkroni i asinkroni te *fastbus* način rada. Načini distribucije takta između sabirnice i jezgre u većoj mjeri utječu na propusnost sustava, kao i potrošnju energije.

Ranije je spomenuto da jezgra može raditi na većem taktu od sabirnice. Ovo je idealno za aplikacije koje ne zahtijevaju česte pristupe memoriji te se mogu izvršavati iz priručne memorije. Kod ovakvog načina rada, brzina izvršavanja je maksimalna dok god se program izvršava iz priručne memorije, međutim, čim se pojavi potreba za čitanjem vanjske memorije, jezgra čeka dok ne primi podatak, i tek potom nastavi s radom. Osim što je potrebno čekanje, prilikom komunikacije jezgre s sabirnicom, potrebno je obaviti sinkronizaciju između jezgre i sabirnice što unosi dodatne troškove u vremenu. Pažljivim odabirom načina distribucije takta se ovakvi troškovi mogu optimizirat s obzirom na aplikaciju.

Standardni način rada je koristan pri projektiranju sustavu u kome se ne zahtijeva velika propusnost sabirnice AHB ali je poželjno da jezgra radi na većem taktu. Primjer je situacija u kojoj je na AHB sabirnicu spojena memorija brzine 1MHz. Zadaća mikrokontrolera je pročitati podatke iz memorije te potom nad njima provesti složenu operaciju. U ovakvom sustavu se AHB sabirnica može slobodno postaviti da radi na 1 MHz, veća brzina nije potrebna, a jezgra po potrebi i do 76.205 MHz. Ovime je znatno smanjena potrošnja energije. Standardni način omogućuje dva izvora takta (HCLK, FCLK) te njihovu sinkronizaciju putem nWAIT signala. Jezgra i priručna memorija rade na FCLK, dok sabirnica radi na taktu HCLK. Slika 4.1 prikazuje konceptualnu izvedbu standardnog razvođenja takta.

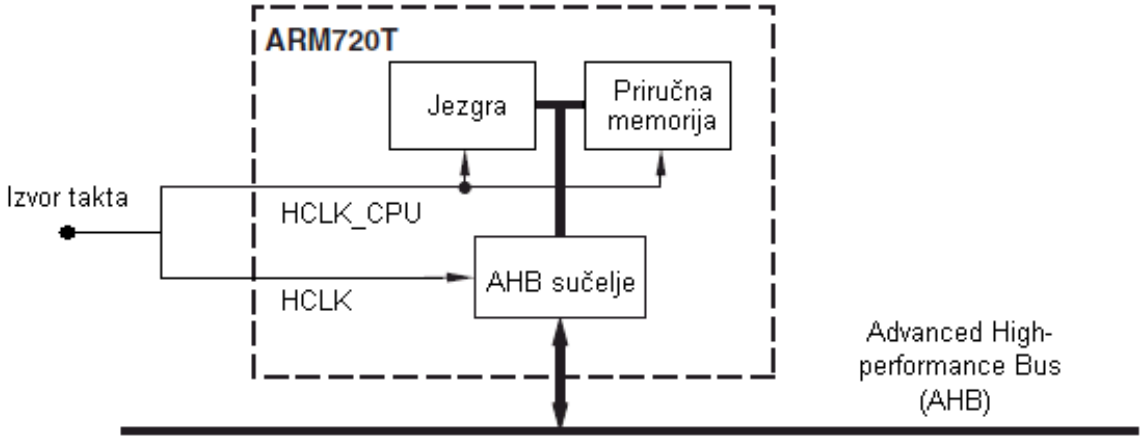


Slika 4.1 Konceptualni prikaz razvođenja takta između AHB sabirnice i jezgre mikrokontrolera u standardnom načinu rada

U standardnom, sinkronom načinu rada, FCLK mora biti veća ili jednaka od HCLK. Također, FCLK mora biti djeljiva s HCLK. Sinkronizacija uzrokuje kašnjenje od najmanje jedne periode takta. U asinkronom načinu rada nije potrebno čuvati odnose između dva takta te oni mogu biti asinkroni.

Za sustave koji često pristupaju brzim memorijama na AHB sabirnici, sinkronizacija i blokiranje jezgre za vrijeme pristupa nije nikako poželjno. U ovakvom slučaju, koristi se *fastbus* način rada.

U fastbus načinu rada jezgra, priručna memorija i AHB sabirnica su pogonjene istim taktom. Slika 4.2 prikazuje konceptualnu izvedbu ovakvog načina distribucije.



Slika 4.2 Konceptualni prikaz razvođenja takta između AHB sabirnice i jezgre mikrokontrolera u fastbus načinu rada

Po resetu, postavljen je standardni, asinkroni način rada. Način rada se može promijeniti kroz CORECONFIG registar opisan u poglavlju 13.2.2.21 [ref user gudie] koji je dostupan i u konfiguracijskom *wizardu*.

4.2.2.2. Inicijalizacija i rad s taktom

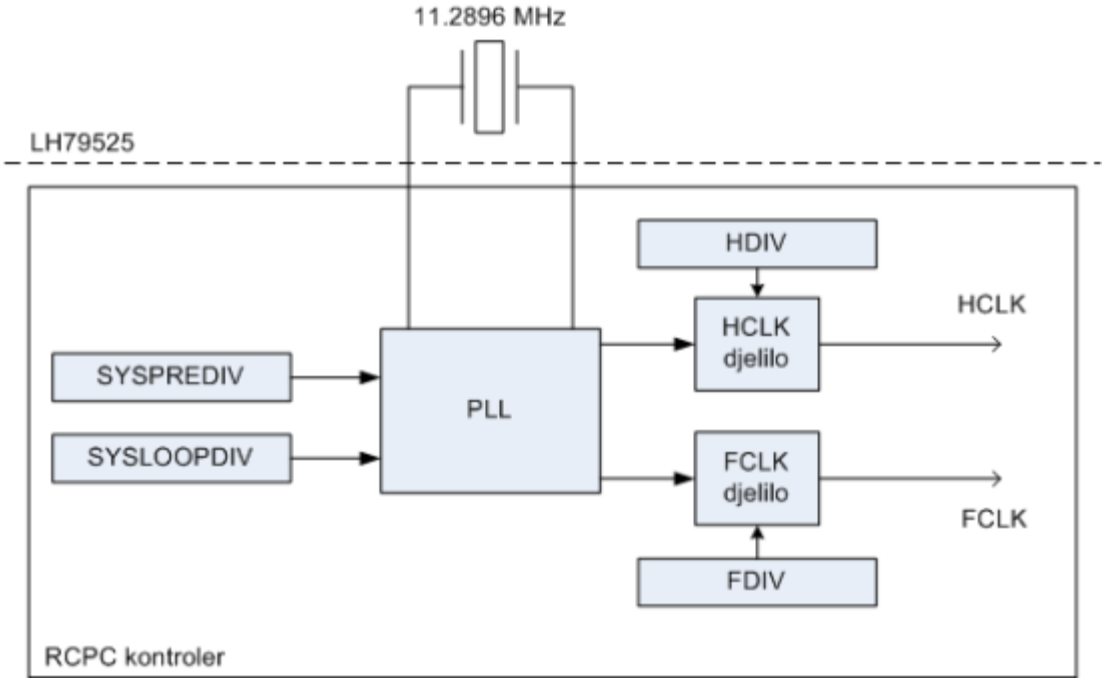
Kontrola svih taktova u sustavu se vrši kroz RCPC – *Reset, Clock and Power Managment* kontroler opisanim u poglavlju 13 u [ref velki guide]. U ovom poglavlju će biti obrađen samo podskup funkcija RCPC kontrolera potrebnih za osnovnu inicijalizaciju kontrolera.

Da bi mikrokontroler normalno počeo s radom potrebno mu je inicijalizirati sustavski takt, takt jezgre te način distribucije takta. Tablica 4.2 predstavlja popis registara koje je potrebno modificirati u ovu svrhu te njihov kratak opis. Kolona *konfiguracijski wizard* označava da li se registrom može upravljati putem konfiguracijskog wizarda unutar alata Keil.

Tablica 4.2 Popis potrebnih registara za rukovođenje i inicijalizaciju nužnih taktova sustavu

Registar	**Opis**	**Poglavlje u [r]**	**Konfiguracijski wizard**
CTRL	Postavke CLKOUT priključka	13.2.2.1	✓
SYSCLKPRE	Postavke djelila takta na izlazu iz sustavskog PLL-a, odnosno, djelilo za HCLK	13.2.2.7	✓
CPUCLKPRE	Postavke djelila za takt jezgre	13.2.2.8	✓
CORECONFIG	Odabir *standard* (*synchronous*, a*synchronous*) ili *fastbus* načina distribucije takta	13.2.2.1	✓
SYSPLLCTL	Postavke sustavskog PLL-a	13.2.2.23	✓

Prilikom rada sa RCPC kontrolerom, RCPC kontroler garantira čiste prijelaze s jedne frekvencije na drugu za HCLK, FCLK i izlaz sustavskog PLL-a. Ovo pak ne vrijedi za sve druge taktove koje RCPC generira. Slika 4.3 prikazuje konceptualnu shemu distribucije takta.



Slika 4.3 Konceptualna shema distribucije takta u mikrokontroleru LH79525

Prvo je potrebno postaviti SYSPLLCTL registar. Tablica 4.3 prikazuje građu registra, a Tablica 4.4 funkciju pojedinih polja.

Tablica 4.3 Građa registra SYSPLLCTL

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Polje	//		//	SYSFRFNGE	SYSPREDIV						SYSLOOPDIV					
Reset	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFE2000 + 0xC0															

Tablica 4.4 Funkcije pojedinih polja registra SYSPLLCTL

Bitovi	**Ime**	**Opis**
31:14	//	Rezervirano, čitanje vraća 0, potrebno upisati reset vrijednost
13	//	Rezervirano, potrebno upisati 1
12	SYSFRANGE	Raspon frekvencijskog izlaza sustavskog PLL-a:

		1 = 100 do 304.819 MHz (za najmanje podrhtavanje takta koristiti ovu vrijednost) 0 = 20 do 100 MHz
11:6	SYSPREDIV	Uloga vrijednosti definirana izrazom 1
5:0	SYSLOOPDIV	Uloga vrijednosti definirana izrazom 1

Izlaz sustavskog PLL-a dan je izrazom 1.

$$SystemPLLfrequency = \frac{SystemClockOscillatorFrequency \quad *SYSLOOPDIV}{SYSPREDIV} \tag{1}$$

Preporučene vrijednosti su za *fastbus* način rada i željeni HCLK = 50.803 MHz:

- SYSFRANGE – 1
- SYSPREDIV – 1
- SYSLOOPDIV – 9

Ovim se vrijednostima na izlazu iz PLL-a dobiva takt u iznosu 101.6064 MHz. Sljedeće je potrebno podesiti registre SYSCLKPRE i CPUCLKPRE. Njihova građa dana je tablicama Tablica 4.5 i Tablica 4.7, dok je opis dan tablicama Tablica 4.6 i Tablica 4.8.

Tablica 4.5 Registar SYSCLKPRE

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Polje	//															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	//												HDIV			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFE2000 + 0x18															

Tablica 4.6 Funkcije polja registra SYSCLKPRE

Bitovi	**Ime**	**Opis**
31:4	//	Rezervirano, upisati reset vrijednost

3:0	HDIV	Funkcija određena izrazom 2

Registar SYSCLKPRE određuje vrijednost djelila frekvencije HCLK, rezultantna frekvencija je dana izrazom 2:

$$f_{HCLK} = \frac{f_{SystemPLL}}{2*HDIV} \quad (2)$$

Preporučena vrijednost za HCLK = 50.803 MHz je HDIV = 1.

Tablica 4.7 Registar CPUCLKPRE

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Polje	//															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	//												FDIV			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFE2000 + 0x1C															

Tablica 4.8 Funkcije polja registra CPUCLKPRE

Bitovi	**Ime**	**Opis**
31:4	//	Rezervirano, upisati reset vrijednost
3:0	FDIV	Funkcija određena izrazom 3

$$f_{FCLK} = \frac{f_{SystemPLL}}{2*FDIV} \quad (3)$$

Preporučena vrijednost za FDIV je 1. Konačno je još ostalo podesiti registar CORECONFIG. Njegova građa dana je tablicom Tablica 4.9, a sadržaj tablicomTablica 4.10.

Tablica 4.9 Registar CORECONFIG

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**

Polje	//															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	//														CCLK	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFE2000 + 0x88															

Tablica 4.10 Funkcije polja registra CORECONFIG

Bitovi	Ime	Opis
31:2	//	Rezervirano, upisati reset vrijednost
1:0	CCLK	Konfiguracija distribucije takta: 00 = Standardni, asinkron način 01 = Fastbus način 10 = Standardni, sinkron način 11 = Fastbus način

Preporučuje se odabrati Fastbus način rada. Ovime je konfiguracija takta završena.

4.2.2.3. Predefinirana distribucija takta

Tablica 4.11 prikazuje postavke takta u slučaju da inicijalizacija nije provedena:

Tablica 4.11 Vrijednosti takta u sustavu po resetu mikrokontrolera

Ime	Vrijednost	Zadovoljava
Takt na izlazu is sustavskog PLL-a	56.4480 MHz	✓
Frekvencijski opseg takta na izlazu iz sustavskog PLL-a	0 – 100 MHz	✓
HCLK	3.7632 MHz	✓
FCLK	3.7632 MHz	✓
Način distribucije takta	Standardni asinkroni	✓

Kako je vidljivo iz tablice svi taktovi u sustavu su unutar dozvoljenih vrijednosti po resetu. Ipak, takav pristup dizajnu se nipošto ne preporuča, ali ako je potrebno napisati

par linija koda u asembleru koje će upaliti diodu, što se tiče takta, sustav će raditi u dopuštenim granicama.

4.2.3. Reset mikrokontrolera

Reset sustava i mikrokontrolera je od iznimne važnosti za pravilan rad i razumijevanje sustava. U poglavlju 3.3.1 je u većoj mjeri pojašnjen reset uređaja, te vanjski priključci koji u trenutku reseta definiraju daljnje ponašanje mikrokontrolera. U ovom poglavlju će biti spomenute tek neke dodatne opcije kao i mogućnosti praćenja i kontrole reseta dok će sve zajedno biti zaokruženo u poglavlju 4.2.4 i 4.2.5. Prije nastavka, svakako se preporuča još jednom pročitati poglavlje 3.3.1.

Reset osim iz vanjskog izvora, odnosno priključka nRESETIN može također doći i iz *Watchdog timera* te se također može uzrokovati i programski. Tablica 4.12 prikazuje registre vezane uz reset mikrokontrolera, te njihov opis. Svi registri se nalaze u okviru RCPC kontrolera. Za više detalja o registrima i njihovoj uporabi, pogledati poglavlje 13 u [ref vlki].

Tablica 4.12 Registri vezani uz reset mikrokontrolera i njihov kratak opis

Registar	Adresa	Opis
SOFTRESET	0xFFFE2000 + 0x0C	Registar omogućuje programsko resetiranje upisom 0xDEAD u isti
RSTSTATUS	0xFFFE2000 + 0x10	Omogućuje uvid u izvor reseta.
RSTSTATUSCLR	0xFFFE2000 + 0x14	Briše zastavice registra RSTSTATUS

4.2.4. Opcije punjenja (*bootanja*) mikrokontrolera

U ovom poglavlju će biti dani tek postupci punjenja mikrokontrolera. Svi detalji koji ostanu u ovom poglavlju nejasni će biti detaljnije razjašnjeni u poglavlju 4.2.5. Pod pojmom punjenja mikrokontrolera podrazumijeva se izvor programa za mikrokontroler. Kako je već ranije spomenuto, mikrokontroler ne posjeduje nikakvu internu programsku memoriju pa je punjenje ovim znatno otežano. Jedina memorija koju mikrokontroler posjeduje je interni SRAM veličine 16 KB. Prilikom reseta mikrokontroler uzorkuje stanja priključaka PC[7:4] te na osnovu njih odabire odakle će početi izvršavat program. Tablica 4.13 prikazuje opcije punjenja.

Tablica 4.13 Opcije punjenja mikrokontrolera u ovisnosti i stanju priključaka PC[7:4]

PC[7:4]	Tip izvora	Širina podatkovne sabirnice	Kontrola
0x0	NOR Flash ili SRAM	16-bit	nBLEx aktivni nisko
0x1	NOR Flash ili SRAM	16-bit	nBLEx aktivni visoko
0x2	NOR Flash ili SRAM	8-bit	nBLEx aktivni nisko
0x3	NOR Flash ili SRAM	8-bit	nBLEx aktivni visoko
0x4	NAND Flash (mali blok)	8-bit	Adresa široka 3 B
0x5	NAND Flash (mali blok)	8-bit	Adresa široka 4 B
0x6	NAND Flash (veliki blok)	8-bit	Adresa široka 4/5 B
0x7	NAND Flash (mali blok)	16-bit	Adresa široka 3 B
0x8	NOR Flash ili SRAM	32-bit	nBLEx aktivni nisko
0x9	NOR Flash ili SRAM	32-bit	nBLEx aktivni visoko
0xA	Rezervirano	//	//
0xB	Rezervirano	//	//
0xC	NAND Flash (mali blok)	16-bit	Adresa široka 4 B
0xD	NAND Flash (veliki blok)	16-bit	Adresa široka 4/5 B
0xE		–	–
0xF	UART	–	–

Konfiguracija priključaka po resetu koju je mikrokontroler pročitao se može očitati iz PBC registra u *Boot* kontroleru.

Prilikom reseta se događa sljedeće: u ovisnosti o konfiguraciji (punjenje iz vanjske statičke memorije) na adresu 0x0 se mapira vanjska statička memorija, konkretno, CS1. U slučaju punjenja putem UART-a, I2C-a ili iz NAND flasha na adresu 0x0 se mapira program iz *boot* ROM memorije čija je zadaća pravilna inicijalizacija odgovarajućih kontrolera i kopiranje koda u internu SRAM memoriju.

Prilikom punjenja putem i UART-a, mikrokontroler čita ne više od 4 KB programskog koda, a prilikom punjenja iz NAND flasha ne više od 1 KB. Kako se punjenje putem UART-a koristi u većoj mjeri u sustavu, ovo je potrebno posebno imati na umu, za sav kod koji je veći od 4 KB je potrebno koristiti neki oblik programa punioca (*bootloader*). Tablica Tablica 4.14 prikazuje postavke punjenja putem UART sučelja

Tablica 4.14 Postavke punjenja putem UART sučelja i pripadni protokol

Parametar	Vrijednost
Protokol	XMODEM checksum

Brzina prijenosa	115 Kbps
Broj podatkovnih bitova	8
Paritetni bit	Ne
Broj STOP bitova	1
Veličina paketa	128 B

Postupak punjenja je sljedeći, putem prekidača S200 se odabere željeni način punjenja. Ako se radi o punjenju iz neke od memorija, nakon resetiranja mikrokontrolera (S1) počet će se izvršavati program. Ako se radi u punjenju putem UART sučelja, nakon što je na prekidačima odabran način, mikrokontroler je potrebno resetirati. Nakon što je mikrokontroler resetiran, očekuje da je prijenos podataka već pokrenut. Potrebno je napomenuti da neće doći do gubitka podataka, naime XMODEM protokol se brine za ovo, nakon što je pokrenuto slanje sa strane PC-a, protokol čeka na slanje posebnog znaka prije nego pošalje cijelu datoteku, u ovom slučaju program. Dakle, ispravni koraci za punjenje putem UART-a su: odabrati konfiguraciju na prekidačima, pokrenuti slanje te potom resetirati mikrokontroler. Također je potrebno napomenuti da će mikrokontroler svakih 10 sekundi pokušat primit podatke.

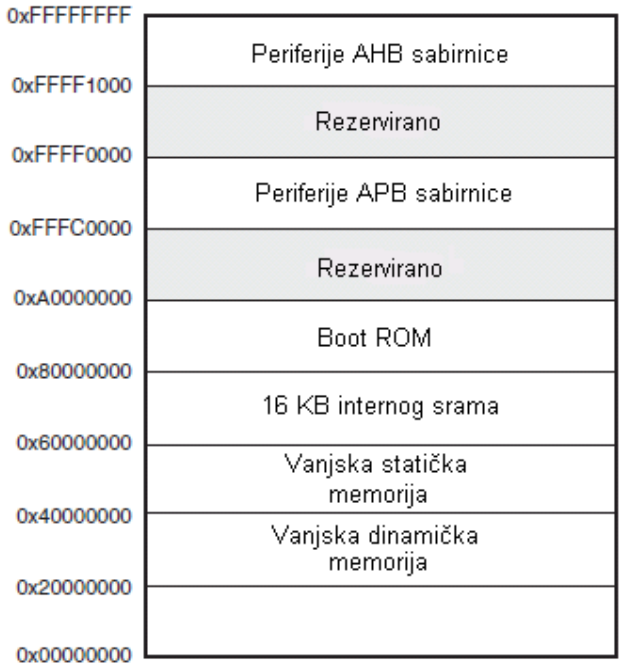
Napomena: verzija XMODEM protokola ima nekoliko, ovdje se koristi XMODEM checksum. Prije slanja programskog koda potrebno je provjeriti da li terminalski program podržava protokol XMODEM checksum, ili možda neki drugi, primjerice samo XMODEM CRC (što primjerice podržava HyperTerminal). Terminalska aplikacija koji se preporučuje koristiti je Tera Term, verzija 3.1.3. U nastavku rada se podrazumijeva korištenje upravo ove aplikacije. Primjer punjenja se nalazi u poglavlju 4.3.

4.2.5. Memorijska mapa mikrokontrolera

Sve komponente u sustavu, od vanjskih i unutarnjih memorija do registara perifernih jedinica su memorijski mapirane. Jezgra svim komponentama pristupa putem AHB sabirnice. Gruba podjela komponenti u sustavu se može izvršiti na:

- Vanjsku memoriju
- Unutarnju memoriju
- Periferne jedinice APB sabirnice
- Periferne jedinice AHB sabirnice

Slika 4.4 prikazuje jednu takvu podjelu te adrese na kojima se pojedine komponente nalaze.



Slika 4.4 Memorijska mapa mikrokontrolera LH79525

Na slici se mogu primjetiti vanjska statička memorija koja je mapirana na adresi 0x40000000 te zauzima na 512 MB, potom vanjska dinamička memorija koja je mapirana na adresu 0x20000000, interni SRAM i boot ROM. Tu su također upravljački i statusni registri periferija spojenih na neku od sabirnica.

Prvih 512 MB je namjerno ostavljeno prazno. Naime, što će biti mapirano u ovom djelu memorije ovisi o nekoliko faktora a ti su konfiguracija priključaka PC[7:4] po resetu, potom stanje REMAP registra (RCPC kontroler) te stanju registra CS1OV (Boot kontroler). Tablica 4.15 prikazuje memorijsko mapiranje u ovisnosti o stanju REMAP registra.

Tablica 4.15 Memorijsko mapiranje u ovisnosti o stanju REMAP registra

Adresa	REMAP = 0b00	REMAP = 0b01	REMAP = 0b10	REMAP = 0b11
0x00000000 – 0x1FFFFFFF	nCS1	nDCS0	Interni SRAM	nCS0
0x20000000 – 0x3FFFFFFF	SDRAM	SDRAM	SDRAM	SDRAM

0x40000000 – 0x5FFFFFFF	Statička memorija	Statička memorija	Statička memorija	Statička memorija
0x60000000 – 0x7FFFFFFF	Interni SRAM	Interni SRAM	Interni SRAM	Interni SRAM
0x80000000 – 0x80001FFF	Boot ROM	Boot ROM	Boot ROM	Boot ROM
0x80002000 – 0xFFFBFFFF	Nedopušten pristup – uzrokuje *Memory Abort*			

U tablici valja primjetiti da se mijenja samo mapiranje prvih 512 MB adresnog prostora. Prije pojašnjenja kako utjecati na mapiranje, potrebno je pogledati mapiranje statičke i dinamičke memorije. Tablica 4.16 prikazuje memorijsku mapu vanjskih memorija i pripadnih CS priključaka.

Tablica 4.16 Memorijska mapa vanjskih memorija i pripadni *chip selectovi*

Adresa	Komponenta spojena na priključak	Tip komponente
0x20000000 - 0x2FFFFFFF	nDCS0	Dinamička memorija
0x30000000 - 0x3FFFFFFF	nDCS1	Dinamička memorija
0x40000000 - 0x43FFFFFF	nCS0	Statička memorija
0x44000000 - 0x47FFFFFF	nCS1	Statička memorija
0x48000000 - 0x4BFFFFFF	nCS2	Statička memorija
0x4C000000 - 0x4FFFFFFF	nCS3	Statička memorija
0x50000000 - 0x5FFFFFFF	//	Nedopušten pristup

Prije nastavka potrebno je utvrditi što znači *chip select* te što znači da se određeni *chip select* nalazi mapiran na određenoj adresi. Chip select priključak ima za zadatak aktivirati odgovarajuću memoriju, memorijsko mapiranje CSx priključka označava da se pristupom na adresu gdje je memorija mapirana aktivira upravo taj CS te se pristupom vanjskoj sabirnici pristupa memoriji koja je spojena na odabrani CS. Primjerice, u tablici Tablica 4.15 za slučaj da je REMAP registar jednak 0, mikrokontroler će početi izvršavati program iz memorije spojene na CS1.

Sada je potrebno do kraja razjasniti što utječe na mapiranje memorije. Prvo REMAP registar. Njegov sadržaj je dan tablicom

Tablica 4.17 REMAP registar

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Polje	//															

Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	//															REMAP
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFE2000 + 0x08															

REMAP registar je po resetu uvijek jednak 0. Ako se pogleda tablica Tablica 4.13 može se primjetiti da nijedna opcija ne vodi punjenju iz SDRAM-a, interne SRAM memorije ili nCS0 priključka. Dakle, stanje REMAP registra određuje korisnik, ne dira ga čak niti program u *boot* ROM-u. Ako postoji potreba da se primjerice SDRAM mapira na početnu adresu, potrebno je skladno tome programirati REMAP registar.

Posljednji registar koji određuje mapiranje po resetu, je CS1OV registar u *boot* kontroleru. Registar je prikazan tablicom Tablica 4.18

Tablica 4.18 CS1OV registar

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Polje	//															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	//															CS1OV
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFE6000 + 0x04															

Ime registra dolazi od CS1 *OVerride*. Bit CS1OV se postavlja u ovisnosti o bitu PC:6. Mikrokontroler uvijek izvršava program s CS1 priključka. Ako se pogleda Tablica 4.14 može se primjetiti da je CS1 uvijek izvor programa osim kada je PC:6 jednak 1. PC:6 ima utjecaj na CS1OV, a konačno, zadatak CS1OV je mapirati *boot* ROM memoriju na adresu 0x0 umjesto vanjskog statičkog SRAM-a. I sada se konačno može proširiti

objašnjenje punjenja dano u 4.2.4. Mikrokontroler po resetu uzorkuje stanje priključaka PC[7:4] te u ovisnosti o njima konfigurira memorijski kontroler (širina podatkovne sabirnice, aktivan signal na linijama nBLEx) te počinje izvršavati program s CS1 priključka. Ovo pravilo ima iznimku kada je PC:6 jednak 1. U tom slučaju, mikrokontroler će na CS1 mapirati *boot* ROM. *Boot* ROM će u skladu sa stanjem priključaka PC:7, PC:5 i PC:4 (PC:6 je jednak 1) konfigurirati određeno sučelje (NAND flash, UART ili), učitati programski kod putem sučelja (podsjetnik, 4 KB za UART ili , 1 KB za NAND flash), kopirati kod u interni SRAM na adresu 0x60000000, te potom očistiti stanje CS1OV registra i konačno skočiti na adresu 0x60000000, odnosno na mjesto gdje je upravo kopirao programski kod. Dakle, ono što je potrebno primjetiti je da mikrokontroler uvijek mapira CS1 priključak na 0x0 a kada je PC:6 po resetu u stanju 1, tada će mapirati *boot* ROM na CS1. Sva druga mapiranja korisnik mora osigurati sam putem REMAP registra. Tablica 4.19 prikazuje memorijsku mapu kada je PC:6 jednak 1.

Tablica 4.19 Memorijska mapa uz PC[7:4] jednak 0bx1xx

Adresa	**REMAP = 0b00**
0x00000000 - 0x1FFFFFFF	Boot ROM
0x20000000 - 0x2FFFFFFF	SDRAM nDCS0
0x30000000 - 0x3FFFFFFF	SDRAM nDCS1
0x40000000 - 0x43FFFFF	SRAM nCS0
0x44000000 - 0x47FFFFF	Boot ROM
0x48000000 - 0x4BFFFFFF	SRAM nCS2
0x4C000000 - 0x4FFFFFFF	SRAM nCS3
0x50000000 - 0x5FFFFFFF	Nedopušten pristup – *Memory Abort*
0x60000000 - 0x7FFFFFFF	Interni SRAM
0x80000000 - 0x80000FFF	Boot ROM
0x80001000 - 0xFFFBFFFF	Nedopušten pristup – *Memory Abort*

Još je ostalo mapiranja registara perifernih jedinica spojenih na APB i AHB sabirnicu, međutim oni ne zahtijevaju dodatno pojašnjavanje te se mogu naći u [ref veliki], poglavlju 1.6.

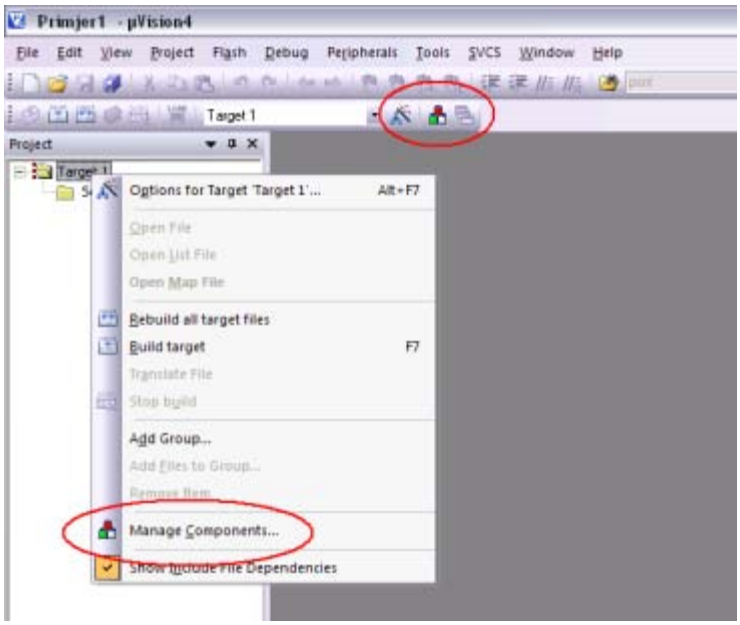
4.3. Hello World

Cilj ovog poglavlja je pokazati osnove korištenja programskog alata Keil uVision 4 te osnove korištenja sustava. Prvo će biti pokazane osnovne stavke potrebne za rad u Keil uVisionu, potom podešavanje terminalskog programa i konačno izvršavanje programa na sustavu.

Iz direktorija `diplomskiRad` kopirati direktorij `Primjer1` u proizvoljni direktorij te usput proučiti strukturu direktorija. Direktorij `project` će sadržavati projektnu datoteku te ostale datoteke koje koristi programski alat. Također sadrži i aplikaciju Hexbin. Direktorij `output` sadrži datoteke kreirane u postupku prevođenja dok direktorij `source` sadrži programski kod korišten u projektu. U nastavku se za svaki projekt podrazumijeva ovakva struktura.

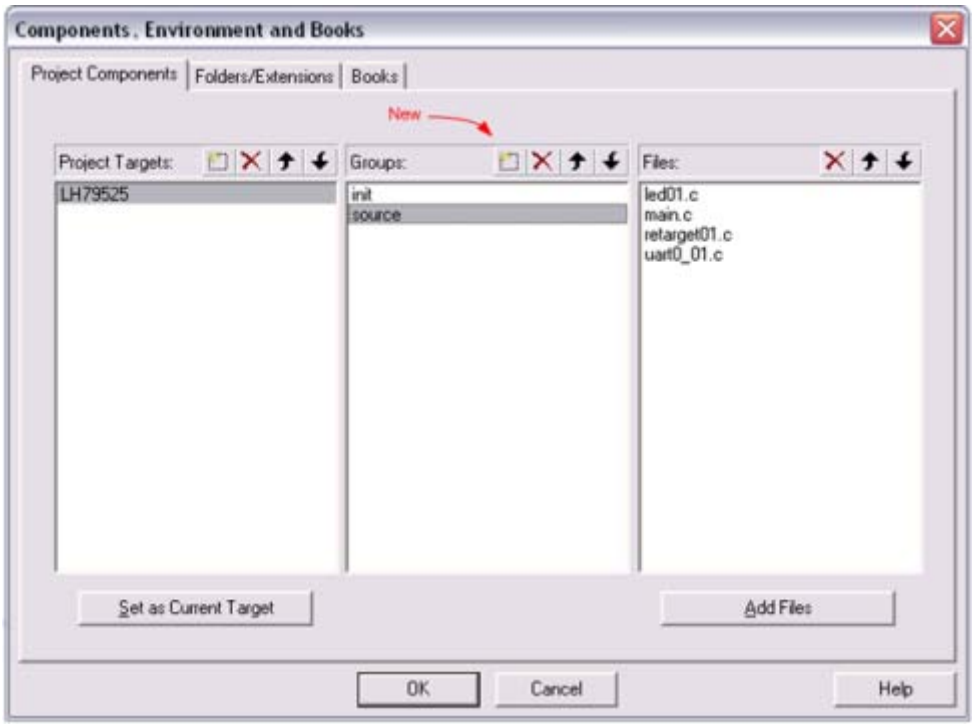
Pokrenuti programski alat Keil uVision te odabrati novi projekt. Projekt je potrebno imenovati „Primjer1“ i smjestiti ga u `[odabrani direktorij]/Primjer1/project`. Za mikrokontroler odabrati NXP-ov LH79525, a automatsko uključivanje *Startup* datoteke odbiti.

Sada je potrebno uključiti programski kod. Odabrati opciju *Manage Components* (kroz alatnu traku ili desnim klikom na Target1 u *Project Exploreru*) kako je prikazano na slici Slika 4.5.



Slika 4.5 Odabir opcije *Manage Components* u programskom alatu Keil uVision 4

U prozoru *Project Targets* dvostrukim klikom na *Target1* preimenovati *Target1* u *LH79525*. Na isti način preimenovati i *Source Group 1* u prozoru *Groups* u *init.* Potom, u prozoru *Groups* dodati novu grupu pritiskom na tipku *New* ili označavanjem prozora i pritiskom tipke Insert na tipkovnici. Novu grupu imenovati *source.* Sada je potrebno u grupe dodati kodove programa. Označiti grupu *source* te u prozoru *Files* kliknuti na *Add Files*, pozicionirati se u direktorij `[odabrani direktorij]/Primjer1/source` te zaokružiti sve `*.c` datoteke i dodati ih u grupu. Potom odabrati grupu *init,* pozicionirati se u prethodni direktorij, u opcijama *Files of type* odabrati *ASM Source File* te dodati `LH79524.s`. Sada bi dijalog trebao izgledati kao na slici Slika 4.6.

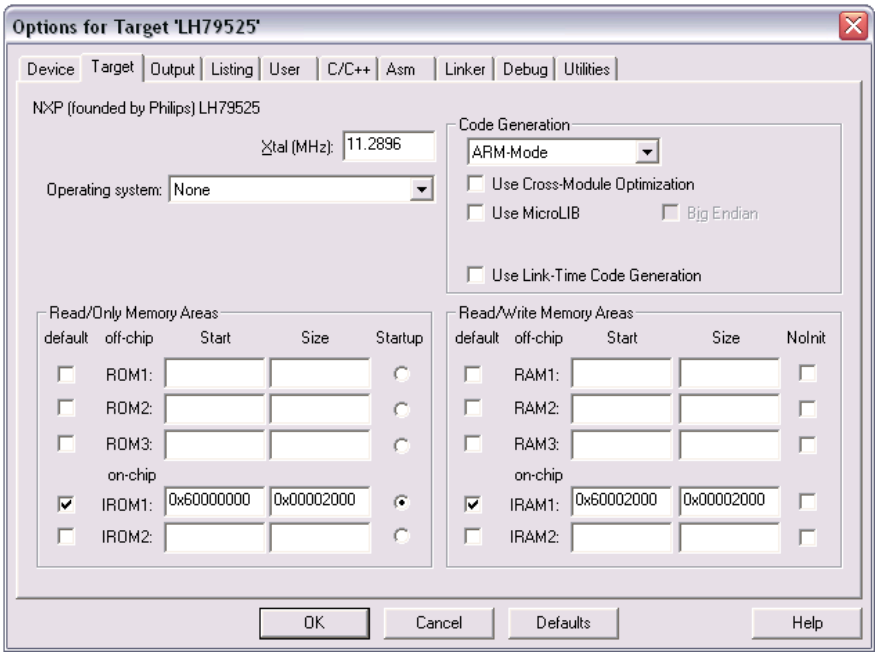


Slika 4.6 Prikaz dijaloga *Manage Components* nakon dodavanja svih potrebnih programskih kodova

Sada je potrebno podesiti opcije linkera, te se pobrinuti za kreiranje `*.bin` datoteke koja se šalje mikrokontroleru. Uključiti dijalog *Options for target „LH79525“* desnim klikom na LH79525 u *Project Exploreru.* U tabu *Target* je potrebno podesiti veličinu i poziciju ROM i RAM memorije. Na temelju ovih adresa linker generira adrese varijabli,

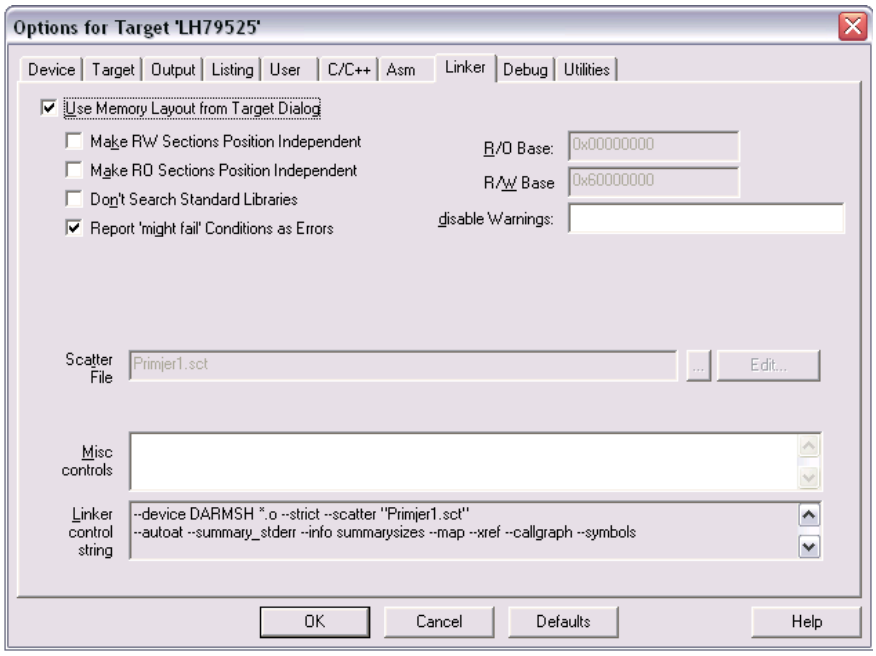
adrese skokova i slično, ako se program prevede sa različitim adresama od onih na kojima će se izvršavati doći će do greške već pri prvom skoku ili učitavanju varijable.

Pošto će se ovaj program izvršavati iz internog SRAM-a koji je veličine 16 KB, SRAM će se logički podijeliti na dva sektora po 8 KB. Pod *Read/Only Memory Areas* u polje *IROM1* upisati adresu 0x60000000 (adresa internog SRAM-a), a pod size upisati 0x2000. Označiti opcije *Default* i *Startup*. Pod *Read/Write Memory Areas* u polje *IRAM1* upisati 0x60002000, a pod *Size* ponovo upisati 0x2000. Označiti opciju *default*. Izgled taba nakon ovih koraka je prikazan slikom Slika 4.7.



Slika 4.7 Izgled taba *Target* nakon potrebnih izmjena

Sada je potrebno reći linkeru da koristi upravo postavljene vrijednosti. U tabu *Linker* je potrebno označiti opciju *Use Memory Layout From Target Dialog,* potom ju odznačiti, te ju na kraju ponovo označiti. Rezultat bi trebao biti zasivljeno ime datoteke *Primjer1.sct* u polju *Scatter File.* Izgled taba je dan slikom Slika 4.8.

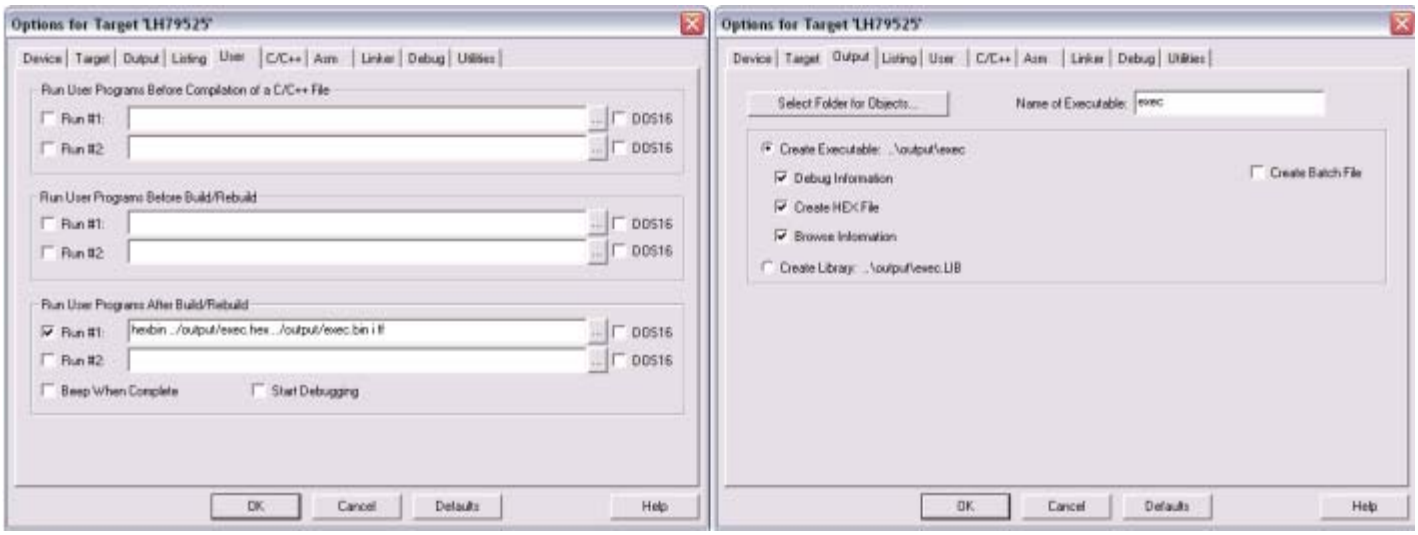


Slika 4.8 Izgled taba *Linker* nakon potrebnih izmjena

Sljedeće je potrebno omogućiti generiranje `*.HEX` datoteke. U tabu *Output* omogućiti *Create Hex File.* Također je potrebno postaviti direktorij u koji će se spremati rezultati prevođenja. Odabrati *Select Folder for Objects,* pozicionirati se u `[odabrani dikretorij]/Primjer1/output` te odabrati *OK*. Potom polje *Name of Executable* izmjeniti u *exec*. Još je ostalo omogućiti generiranje `*.bin` datoteke koja sadrži strojne kod programa. Prvo se pobrinuti da se u direktoriju `[odabrani dikretorij]/Primjer1/project` nalazi aplikacija *hexbin.exe.* Potom u tabu *User* utipkati sljedeću komandu u polje *Run #1* opcije *Run User Programs After Build/Rebuild*:

```
hexbin ../output/exec.hex ../output/exec.bin i ff
```

Naredba ima zadaću nakon uspješnog prevođenja programa iz *hex* datoteke stvoriti binarnu datoteku koja sadrži program za izvođenje. Napomena: hexbin ima ograničenje na ime ulazne i izlane datoteke (hex, bin) na 8 znakova pa je potrebno u skladu s time odgovarajuće imenovati datoteke. Slika 4.9 prikazuje stanje tabova nakon potrebnih izmjena.



Slika 4.9 Izgled taba output i user nakon potrebnih izmjena

Konačno, potrebno je kliknuti *OK* i time zatvoriti *Options for Target* dijalog. U *Project* ili u alatnoj traci odabrati *Rebuild all target files* te se nakon prevođenja uvjeriti da se u direktoriju `[odabrani dikretorij]/Primjer1/output` nalazi datoteka `exec.bin`.

Sljedeće je potrebno pripremiti sustav za primanje programa. Postaviti kratkospojnike J1 i J2 u položaj 1 te prekidač S200 u položaj za punjenje putem UART-a (0b1111) (poglavlje 3.3.2) te spojiti sustav putem serijskog kabela (koristi se kabel sa izravno spojenim Rx i Tx linijama). Pokrenuti Tera Term te odabrati odgovarajući COM priključak. Potom u *setup* → *serial port* podesiti parametre na način dan u tablici Tablica 4.20.

Tablica 4.20 Parametri terminala za punjenje mikrokontrolera putem UART-a

Parametar	Vrijednost
Port	COM priključak na koji je spojen sustav
Baud rate	115 200 b/s
Data	8 bitova
Parity	None
Stop	1 b
Flow control	None

Sada je sve spremno za slanje. Provjeriti još jednom da je sustav pod napajanjem, ispravno spojen s računalom, te da je odabrana odgovarajuća konfiguracija. U terminalu

odabrati *File → Transfer → XMODEM → Send.* Odabrati `Primjer1/output/exec.bin`, provjeriti da je odabran XMODEM checksum u donjem dijelu dijaloga te odabrati *Send* i potom resetirati mikrokontroler. Nakon slanja u terminalu bi se trebalo ispisati Hello World!, a svjetleće diode bi se trebale paliti i gasiti.

4.4. Općenito o radu unutar alata Keil uVision

Prošlo je poglavlje dalo kratak uvod u rad sa sustavom i programskim alatom uVision. Pri radu s alatom treba imati na umu nekoliko stvari. Prva je da je prilikom kreiranja novog projekta uvijek potrebno inicijalizirati okolinu za rad i dati upute prevodiocu što i kako treba raditi. U ovom slučaju je potrebno reći prevodiocu na kojim će se adresama nalaziti ROM i RAM. Druga stvar koju je potrebno zapamtiti je da alat ne kreira izvršnu, binarnu, datoteku. U radu se za ovo koristi aplikacija hex2bin. Aplikaciji se može pristupati putem komandne linije (DOS) što znači da je nakon svakog prevođenja potrebno pokrenuti stvaranje binarne datoteke, ili se kao u gornjem primjeru postupak može automatizirati. Svakako se preporuča automatizirati stvaranje, vrlo česta greška uključuje izmjene u programu i potom programiranje mikrokontrolera bez da je stvorena nova binarna datoteka, odnosno, mikrokontroler se programira sa starom verzijom što vodi nepredviđenom ponašanju.

Konačno, uVision mnoge stvari u radu s mikrokontrolerom pojednostavljuje. U projektu Primjer1 potrebno je unutar *Project explorera* odabrati datoteku `init/LH79524.s` te u donjem lijevom kutu odabrati tab *Configuration wizard*. Sljedeće je poglavlje posvećeno upravo konfiguracijskom wizardu.

4.4.1. Konfiguracijski čarobnjak (wizard)

Inicijalizacija mikrokontrolera zahtijeva poprilično količinu koda. Istina je da se takva inicijalizacija može napisati samo jednom te po potrebi izmjeniti i uključiti u projekt. Iako najbolje, ovo je ipak pomalo nezgrapno rješenje. Problem otežava i inicijalizacija koja se mora izvršiti prije ulaska u *main*. Upravo za ovakve situacije služi konfiguracijski wizard. Kroz wizard korisnik može jednostavnim odabirom odgovarajućih registara podesiti početno stanje sustava. Kod koji nastaje iz wizarda se izvršava prvi, prije inicijalizacije *librarya* (ako su uopće uključeni) i maina što garantira čisti početak izvršavanja koda.

Wizard omogućuje sljedeće:

- Konfiguraciju stoga
- Konfiguraciju gomile (*heap*)
- Konfiguracija takta
    - HCLK, FCLK, konfiguracija takta perifernih jedinica, itd.
- Konfiguracija ulazno-izlaznih priključaka
    - Odabir funkcije priključka, uključivanje priteznih otpornika
- Konfiguracija memorijskog kontrolera
- Konfiguracija priručne memorije

Wizard štedi vrijeme i smanjuje prostor za grešku (osim gdje sam stvara grešku, više o ovome na kraju poglavlja). Također se brine i za kompaktnost inicijalizacijskog koda. U nastavku rada će se izbjegavati ručno pisanje inicijalizacijskog koda te će wizard biti iskorišten gdje god to bude moguće. Također je potrebno biti svjestan činjenice da ako se određeni registar/dio konfiguracijskog wizarda uopće ne uključi, pripadni kod uopće neće biti stvoren te će postavke ovisiti o stanju registara po resetu.

Napomena: podešavanje putem wizarda rezultira neispravnim ponašanjem u slučaju:

- UART kontrolera
- SDRAM memorije

Više o greškama se može naći u pripadnim poglavljima.

4.5. Rad s ulazno-izlaznim priključcima

Elementi sustava kojima se upravlja putem općenamjenskih ulazno-izlaznih priključaka su:

- Konektor X207
- Konektor X209
- Konektor X210
- Konektor X212
- Konektori X203 i X204 (PS/2)
- Prekidač S201, tipke S202, S203, S205 i S206

- LE diode: H200:H204
- Zujalica H210

Za rad s ulazno-izlaznim priključcima mjerodavni su registri dani tablicomTablica 4.21.

Tablica 4.21 Mjerodavni registri prilikom korištenja ulazno-izlaznih priključaka

Registar	Opis	Adresa
MUXCTLx	Kontrola funkcije priključka	Ulazno-izlazni kontroler (*I/O controller*)
RESCTLx	Kontrola priteznih otpornika priključka	Ulazno-izlazni kontroler (*I/O controller*)
P1DRx	Podatkovni registri portova A/C/E/G/I/K/M	Općenamjenski ulazno-izlazni kontroler (*GPIO*)
P2DRx	Podatkovni registri portova B/D/F/H/J/L/N	Općenamjenski ulazno-izlazni kontroler (*GPIO*)
P1DDRx	Registri smjera podataka portova A/C/E/G/I/K/M	Općenamjenski ulazno-izlazni kontroler (*GPIO*)
P2DDRx	Registri smjera podataka portova B/D/F/H/J/L/N	Općenamjenski ulazno-izlazni kontroler (*GPIO*)

Koraci u radu s ulazno-izlaznim priključcima se sastoje u sljedećem:

1. Odrediti željeni priključak s kojim se želi raditi, ovo se može napraviti uz pomoć poglavlja 3.63.6 koje sadrži popis svih konektora i pripadnih priključaka
2. Uz pomoć tablice Tablica 3.12 provjeriti da li je priključku dodijeljena još koja funkcija u sustavu te da li ga se smije koristiti
3. Odrediti odgovarajuće MUXCTLx i RESCTLx registre te ih podesiti kroz konfiguracijski wizard. Ovo se može uz pomoć Tablica 4.24
4. Programirati željeni smjer priključka kroz PyDDRx registar
5. Priključku se sada normalno pristupa putem registra PyDRx

Registar MUXCTLx određuje funkciju priključka, odnosno, ako priključak ima više dodijeljenih funkcija, kroz MUXCTLx se odredi funkcija priključka (u ovom slučaju priključak će imati funkciju željenog porta). Kroz registar RESCTLx se podešavaju pritezni otpornici. Registri MUXCTLx i RESCTLx su dostupni putem konfiguracijskog wizarda.

Općenamjenski priključci mikrokontrolera nisi bidirekcionalni, potrebno im je programirati smjer prije korištenja. Ovo se ostvaruje putem PyDDRx (*Port Data Direction Register*) registara. Jednom kada je smjer programiran, priključci se koriste

putem PyDRx (*Port Data register*) registara. Tablica 4.22 prikazuje bazne adrese PyDDRx i PyDRx registara.

Tablica 4.22 Bazne adrese registara PyDRx i PyDDRx

Port	Adresa
A, B	0xFFFDF000
C, D	0xFFFDE000
E, F	0xFFFDD000
G, H	0xFFFDC000
I, J	0xFFFDB000
K, L	0xFFFDA000
M, N	0xFFFD9000

Tablica 4.23 prikazuje odmak od bazne adrese za pojedine registre.

Tablica 4.23 Adrese registara PyDRx i PyDDRx i pripadni im portovi

Adresni odmak	Registar	Opis
0x00	P1DRx	Podatkovni registri portova A/C/E/G/I/K/M
0x04	P2DRx	Podatkovni registri portova B/D/F/H/J/L/N
0x08	P1DDRx	Registri smjera podataka portova A/C/E/G/I/K/M
0x0C	P2DDRx	Registri smjera podataka portova B/D/F/H/J/L/N

Priključak PJ je samo ulazni, tako da on ne posjeduje pripadni mu registar smjera. Priključak PM je samo izlazni, ali on posjeduje pripadni registar smjera kojeg je potrebno programirati prije korištenja priključka.

Priključci su po resetu podešeni kao ulazni. Priključak se programira kao izlazni postavljanjem u 1 odgovarajući bit registra smjera. Iako možda na prvi pogled djeluje zbunjujuće, korištenje registara je jednostavno. Prvo se odredi željeni port koji se želi koristiti, primjerice, port H. Potom se iz tablice Tablica 4.22 odredi njegova bazna adresa, u ovom slučaju 0xFFFDC000. Nakon toga se iz tablice Tablica 4.23 odredi adresni pomak od bazne adrese. Za port H to je 0x04 za pripadni mu PDR registar, te 0x0C za PDDR registar. Konačno, adresa registra smjera je 0xFFFDC00C, a adresa podatkovnog registra 0xFFFDC004. Port I bi primjerice imao adresu PDR registra 0xFFFDB000 a adresu PDDR registra 0xFFFDB008.

Tablica 4.24 daje pripadne MUXCTLx i RESCTLx registre pripadnih općenamjenskih ulazno-izlaznih registara.

Tablica 4.24 Pripadni MUXCTLx i RESCTLx registri općenamjenskih ulazno-izlaznih priključaka

Priključak	**MUXCTLx registar**	**RESCTLx registar**
PS2_MOUSECLK/PA1, PS2_KEYCLK/PA0	MUXCTL5	RESCTL5
PA2, PA3, PA4, PA5, PA6, PA7	MUXCTL4	RESCTL4
LCDVD0/PG2, LCDVD1/PG3, LCDVD2/PG4, LCDVD3/PG5, LCDVD4/PG6, LCDVD5/PG7, LCDVD6/PF0, LCDVD7/PF1	MUXCTL22	RESCTL22
LCDVD8/PF2, LCDVD9/PF3, LCDVD10/PF4, LCDVD11/PF5	MUXCTL21	RESCTL21
LCDEN/PF6, LCDFP/PF7, LCDLP/PE0, LCDCLK/PE1, LCDPS/PE2	MUXCTL20	RESCTL20
LCDCLS/PE3, LCDDSPLEN/PE4, LCDVDDEN/PE5, LCDVEEN/PE6	MUXCTL19	RESCTL19
PM3, PM5, PM2	MUXCTL14	//
PM4	MUXCTL15	RESCTL15
PS2DATA/MOUSE (PB0), PS2DATA/KEY (PB1)	MUXCTL6	RESCTL6
PJ0, PJ1, PJ2, PJ3, PJ4, PJ5, PJ6, PJ7	MUXCTL25	//

Napomena: prilikom promjene određenog registra u konfiguracijskom wizardu, potrebno je obratiti pozornost i na druge priključke u registru iako se ne koriste. Odličan primjer gdje na ovako nešto treba obratiti pozornost je konfiguracija registra MUXCTL5. Registar, osim stanja priključaka PA0 i PA1 također konfigurira priključke UARTTX0 i UARTRX0 koji su nužni za komunikaciju putem UART-a. Ako se koristi punjenje putem UART-a, program iz *boot* ROM-a je inicijalizirao ove priključke na funkcije pridjeljene UART kontroleru (upravo je ovo razlog zašto UART0 radi iako nije nigdje inicijaliziran po punjenju putem UART-a). Međutim, ako se uključi konfiguracija registra MUXCTL5 u svrhu rada s priključcima PA0 i PA1, ona će po *defaultu* postaviti priključke UARTTX0 i UARTRX0 na funkcije općenamjenskog priključka porta PB. Naravno, ako se na ovo ne pazi, UART će „*odjednom*“ prestati s radom.

4.5.1. Primjer rada s ulazno-izlaznim priključcima

U poglavlju će biti dan jednostavan primjer rada s ulazno-izlaznim priključcima. Cilj primjera je omogućiti blinkanje LE dioda.

U odabranom direktoriju kreirati strukturu projekta:

```
[odabrani direktorij]/Primjer2/project
                                    /output
                                    /source
```

U direktorij `project` kopirati hexbin.exe aplikaciju. Pokrenuti Keil uVision te kreirati novi projekt imena *Primjer2,* smjestiti ga u direktorij `project`, odabrati LH79525 mikrokontroler te uključiti *startup* datoteku.

Prvo je potrebno podesiti opcije projekta. Na isti način kako je podešeno u 4.3 podesiti opcije u *Options for target*. Ovo uključuje podešavanje tabova *Target*, *Output*, *User* i *Linker*. Način kako je potrebno podesiti opcije projekta je dan u 4.3.

Sljedeće je potrebno identificirati koji se priključak koristi. Uvidom u poglavlje 3.6.13 može se vidjeti da su diode spojene na priključke porta PE i to redom: PE2, PE3. PE4, PE5 i PE6. Uvidom u poglavlje 4.5, tablice Tablica 4.22 i Tablica 4.23 može se ustanoviti da je adresa podatkovnog registra porta PE 0xFFFDD000, dok je adresa smjera priključaka porta PE jednaka 0xFFFDD008. Konačno, uvidom u tablicu Tablica 4.24 vidljivo je da se pripadne funkcije priključaka postavljaju kroz registre MUXCTL19 i MUXCTL20.

U project exploreru pokrenuti konfiguracijski wizard (dvostrukim klikom na LH79524.s te u donjem lijevom kutu odabrati *Configuration Wizard*). Uključiti I/O Configuration te se pobrinuti da su u registrima MUXCTL19 i MUXCTL20 funkcije priključaka PE2:PE6 dodijeljene portu PE. Također na spomenutim priključcima u RESCTL19 i RESCTL20 uključiti pritezne otpornike prema napajanju.

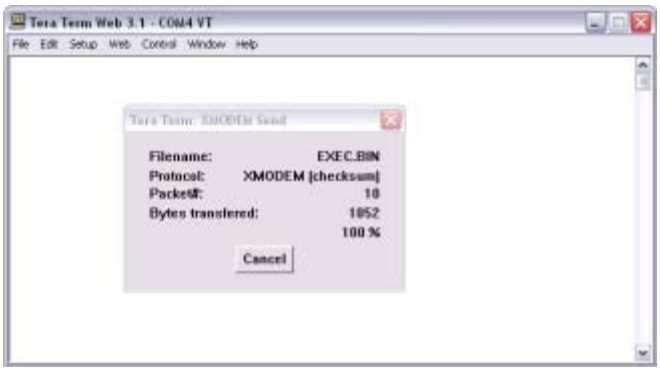
Još je ostalo napisati main program koji će pogoniti LE diode. Odabrati *New* iz alatne trake ili izbornika *File* → *New* te unijeti sljedeći kod:

```
volatile unsigned int *PEdataRegister = (volatile unsigned int*)
                                        0xFFFDD000;
volatile unsigned int *PEdirectionRegister = (volatile unsigned
                                        int*) 0xFFFDD008;

int main(void)
{
      int i;

      // Set direction of port PE, PE2:PE6 are output
      *PEdirectionRegister = 0x7C;

      // Turn all LEDs on and off
```

```
	while(1)
	{
		// LEDs on
		*PEdataRegister = 0x00;
		for(i = 0; i < 500000; i++);
		// Leds off
		*PEdataRegister = 0x7C;
		for(i = 0; i < 500000; i++);
	}
}
```

Odabrati opciju *Save* te spremiti kod u `source` pod imenom `main.c`. Dvostrukim klikom na *Source Group 1* odabrati upravo spremljeni program te ga dodati u grupu. Odabrati *Rebuild all* te se uvjeriti da je prevođenje prošlo bez problema i da se u direktoriju `output` nalazi binarna datoteka `exec.bin`.

Pokrenuti i podesiti Tera Term kako je pokazano u poglavlju 4.3, odabrati *File* → *Transfer* → *XMODEM* → *Send,* odabrati datoteku `Primjer2/output/exec.bin` te resetirati mikrokontroler. LE diode bi se trebale paliti i gasiti. Ovdje se može primjetit jedna neželjena situacija. Nakon slanja, moguće je da se dijalog Tera Term terminala nikada ne ugasi što se može protumačit na način da slanje nikada nije završeno. Slika 4.10 prikazuje takvu situaciju.



Slika 4.10 Stanje Tera Term terminala nakon slanja binarne datoteke

Slanje je zaista završeno, međutim razlog ovakvom ponašanju je neinicijaliziran UART0 kontroler. Svi su podaci poslani i program će se normalno izvoditi, te se oko ovoga ne treba zabrinjavati.

4.5.2. Rad s LE diodama

U prethodnom je poglavlju pokazan rad s LE diodama. Kako bi se rad njima olakšao, razvijene su funkcije za rad s njima. u ovom će biti dan primjer rada s LE diodama. Kopirati projekt LED1 u odabrani direktorij te ga otvoriti u Keil uVision alatu. Program radi jako sličnu stvar osim što koristi funkcije za rad s LE diodama. Za korištenje ovih funkcija potrebno je iz direktorija [dir za libove] kopirati `led01`.c i `led01`.h u projektni direktorij (`source`) te ih dodati u projekt. Detaljniji opis funkcija danih u okviru `led01.c` se nalazi u 4.14.1.

4.5.3. Rad s tipkama i prekidačima

Za potrebe rada su također razvijene i funkcije za rad s tipkama i prekidačima. Za korištenje je potrebno datoteke `buttons01.c` i `buttons01.h` dodati u projekt. Opis funkcija je dan u poglavlju 4.14.2.

4.5.4. Rad sa zujalicom

Zujalici se pristupa putem priključka PM2. U direktoriju se nalazi primjer rada s istom. Primjer je pod imenom Buzzer01 te demonstrira rad sa zujalicom. Iako je stanje priključka po resetu uzeto u obzir, te osiguravanje inicijalnog stanja kroz otpornik R211, zujalica pišti po resetu mikrokontrolera. Kako bi se ovo izbjeglo, premosnik J3 je potrebno držati u otvorenom stanju. Ako se pak zujalica koristi, potrebno je odmah na početku programa inicijalizirati stanje priključka kako je prikazano u nastavku:

```
volatile unsigned int *PM_DATA = (unsigned int*) 0xFFFD9000;
volatile unsigned int *PM_DIR = (unsigned int*) 0xFFFD9008;

void main(void)
{
      //(…)

      // INIT PM2 direction
      *PM_DIR |= 0x04;
      // Turn buzzer off
      *PM_DATA &= 0xfffffffb;

      //( Rest of the code)
}
```

Rad sa zujalicom je jednostavan, kada je priključak PM2 u visokom stanju, ona će pištat, inače neće.

4.6. UART sučelje

Dosada je već par puta korišten UART ali bez ikakvog objašnjenja. UART kontroler posjeduje tri kanala, UART0, UART1 i UART2. Od navedenih, samo je UART0 dostupan u sustavu za korištenje. Tablica 4.25 sadrži popis registara koje je potrebno programirati za korištenje UART sučelja.

Tablica 4.25 Popis registara koji utječu na rad UART kontrolera i korištenje UART0 sučelja

Registar	Adresa	Opis
PCLKCTRL0	0xFFFE2000 + 0x24	Omogućavanje takta za UART0 kanal, registar dostupan putem konfiguracijskog wizarda
PCLKSEL0	0xFFFE2000 + 0x30	Definiranje izvora takta ua UART0 kanal, registar dostupan putem konfiguracijskog wizarda*
MUXCTL5	0xFFFE5000 + 0x20	Pridjeljivanje priključaka PB6 i PB7 UART kontroleru, registar dostupan putem konfiguracijskog wizarda
UARTDR	0xFFFC0000 + 0x000	Primljeni/poslani podaci
UARTFR	0xFFFC0000 + 0x018	Status UART kontrolera i FIFO-a
UARTIBRD	0xFFFC0000 + 0x024	Zajedno s UARTFBRD određuje brzinu prijenosa
UARTFBRD	0xFFFC0000 + 0x028	Zajedno s UARTIBRD određuje brzinu prijenosa
UARTLCR_H	0xFFFC0000 + 0x02C	Konfiguracijski registar UART kontrolera
UARTCR	0xFFFC0000 + 0x030	Kontrolni registar UART kontrolera

**Podešavanje putem konfiguracijskog wizarda rezultira greškom, obavezno pročitati sljedeće poglavlje*

4.6.1. Inicijalizacija UART kontrolera i UART sučelja

Prije podešavanja, prvo je potrebno dovesti takt, a potom i pridjeliti odgovarajuće priključke mikrokontrolera UART kontroleru. Iako može raditi i s HCLK taktom, primarni je zamišljeno da UART radi s taktom oscilatora (11.2896 MHz). Registri kroz koje se omogućuje takt UART-u te odabire izvor su PCLKCTRL0 i PCLKSEL0. Tablica 4.26 prikazuje sadržaj registra PCLKSEL0, a Tablica 4.27 funkcije pojedinih polja.

Tablica 4.26 Registar PCLKSEL0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Polje	//															

Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	//													UART2	UART1	UART0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFE2000 + 0x30															

Tablica 4.27 Opis pojedinih bitova registra PCLKSEL0

Bitovi	Ime	Opis
2, 1, 0	UARTx	Odabir izvora takta UART kanala 0 – Izvor takta za UARTx kanal je takt oscilatora 1 – Izvor takta UARTx kanala je HCLK

S registrom PCLKSEL0 postoji problem kada se njegova vrijednost namješta u konfiguracijskom wizardu. Naime, kada se kao izvor takta namjesti takt oscilatora, registar PCLKSEL0 se programira s vrijednošću koja odgovara odabiru takta HCLK (1). Također, vrijedi i obrat. Najbolje je ako se već registar programira putem konfiguracijskog wizarda provjeriti vrijednost koja će biti upisana u registar (potražiti konstantu PCLKSEL0_Val u teksutalnom prikazu inicijalizacijske datoteke). Sljedeće je potrebno omogućiti takt UART0 kanalu te dodijeliti priključke PB6 i PB7 UART0 kontroleru.

Kada je ovo sve provedeno, može se pristupiti programiranju UART kontrolera. U radu je kontroler korišten na način da ponudi široko korištenu 8-bitnu serijsku komunikaciju sa po jednim STOP i START bitom. Kontroler također posjeduje i FIFO međuspremnik za odlazni i dolazni dio. Ako je spremnik omogućen, upis u UARTDR registar će rezultirati upisom u spremnik te će slanje početi čim kontroler bude u mogućnosti, slično vrijedi i za čitanje, dolazni podaci se spremaju u spremnik te čitanjem UARTDR registra podaci se uklanjaju onim redoslijedom kojim su došli. Detalji oko korištenja UART-a su dani u poglavlju 4.14.3.

4.7. Rad sa memorijama

Rad s memorijama bitno olakšava memorijski kontroler. Konkretno, u mikrokontroleru je implementiran ARM-ov memorijski kontroler *PrimeCell MultiPort Memory Controller (PL175)* [ref ovo]. Implementacija memorijskog kontrolera posjeduje neke specifičnosti kojih je potrebno biti svjestan prilikom rada s njim. Jedna od bitnijih je ona da memorijski kontroler uvijek izvršava 32-bitne pristupe u memoriju bez obzira da li se radi o naredbama (na razini asemblera) LDR, LDRH ili LDRB odnosno da li se traži podatak tipa char, short ili int. Ostale specifičnosti će biti navedene kroz poglavlja o dinamičkim i statičkim memorijama.

U sustavu se nalaze tri vanjske memorije, te su:

- SDRAM, 8MB
- SRAM, 2MB
- Flash, 16 MB

Tablica 4.28 daje memorijsko mapiranje memorija.

Tablica 4.28 memorijsko mapiranje vanjskih memorija u sustavu

Memorija	Početna adresa	Veličina
SDRAM	0x2000 0000	8 MB
SRAM	0x4000 0000	2 MB
Flash	0x4400 0000	16 MB

Svaka od memorija posjeduje neke specifičnosti kako u radu, tako i u inicijalizaciji. U sljedećim poglavljima će biti dane upute za inicijalizaciju i rad.

4.7.1. SDRAM memorija

SDRAM memorija zahtijeva preciznu inicijalizaciju kroz koju se serijom naredbi upućenih SDRAM memoriji inicijalizira komponenta. Memorijski kontroler podržava slanje inicijalizacijskih naredbi međutim važno je napomenuti da ako se inicijalizacije ne provodi uobičajenim redoslijedom memorijski kontroler će se početi nepredviđeno ponašati.

4.7.1.1. Inicijalizacija SDRAM memorije

Inicijalizacija memorija tipa SDRAM ovisi od proizvođača do proizvođača, a generalno izgleda nekako ovako:

1. Dovesti napajanje komponenti
2. Pričekati minimalno 100 ms, za to vrijeme SDRAM memoriji uputiti NOP ili COMMAND INHIBIT naredbe
3. Poslati PRECHARGE ALL naredbu
4. Poslati AUTO REFRESH naredbu
5. Poslati LOAD MODE REGISTER naredbu

Točni detalji vezani uz navedene komande u ovom trenu nisu bitni. Dovoljno je reći da su naredbe kodirane odgovarajućim stanjem CS, RAS, CAS i WE signala. Ipak, jedna naredba je zanimljiva i nju je potrebno dodatno prokomentirati, a ta je LOAD MODE REGISTER naredba. Spomenuta naredba puni interni MODE registar SDRAM memorije odgovarajućom konfiguracijom u kojoj će memorija raditi. Registar se puni na način da se za vrijeme naredbe LOAD MODE REGISTAR učitava stanje adresne sabirnice te se puni u registar. Tablica 4.29 prikazuje registar a Tablica 4.30 prikazuje funkcije pojedinih bitova.

Tablica 4.29 MODE registar SDRAM memorije

Bit	11	10	9	8	7	6	5	4	3	2	1	0
Polje	Rezervirano		Način *bursta* pri upisu	Način rada		Latencija CAS-a			Tip *bursta*	Duljina *bursta*		

Tablica 4.30 Funkcije pojedinih polja MODE registra

Bitovi	**Ime**	**Opis**
11:10	Rezervirano	Upisati 0b00
9	Način *bursta* pri upisu	Omogućavanje *burst* načina upisa podataka pri upisu podataka u memoriju 0 – Duljina *bursta* ovisi o poljima 2:0 1 – Upis samo jedne lokacije
8:7	Način rada	Način rada memorije 0b00 – standardni Sve drugo – rezervirano
6:4	Latencija CAS-a	0b010 – 2 0b011 – 3 Sve drugo – rezervirano

3	Tip *bursta*	0 – sekvencijalni 1 – križani		
2:0	Duljina *bursta*	M2:0	M3 = 0	M3 = 1
		000	1	1
		001	2	2
		010	4	4
		011	8	8
		100	Rezervirano	Rezervirano
		101	Rezervirano	Rezervirano
		110	Rezervirano	Rezervirano
		111	Cijela stranica	Rezervirano

Ovaj registar je potrebno imati na umu prilikom daljnjeg objašnjavanja inicijalizacije memorijskog kontrolera.

U pogledu rada s SDRAM memorijama, memorijski kontroler ima jedan pomalo nezgodan uvjet, a taj je da se SDRAM memoriji uvijek pristupa u *burst* načinu rada duljine 8 (u slučaju 16-bitne memorije, za slučaj 32-bitne memorije duljina *bursta* je 4). Također, memorijskom kontroleru se mora omogućiti rad s ulazno/izlaznim međuspremnicima, inače neće raditi.

Inicijalizaciju SDRAM memorije je najbolje obaviti kroz konfiguracijski wizard kako bi se pobrinulo da se memorija inicijalizira po resetu mikrokontrolera. Za konfiguraciju i inicijalizaciju SDRAM memorije su odgovorni registri dani Tablica 4.31. U tablici su također i dane potrebne vrijednosti na koje je registre potrebno postavit. Pripadne vrijednosti su dane u formatu pogodnom za inicijalizaciju putem konfiguracijskog wizarda.

Tablica 4.31 Parametri za inicijalizaciju SDRAM memorije

Registar	**Vrijednost**
AHBCLKCTRL	Omogućiti takt za SDRAM memoriju
EMC – CONTROL	Omogućiti memorijski kontroler Onemogućiti rad u režimu niske potrošnje
EMC – CONFIG	Ne dirati sadržaj ovog registra, registar nema utjecaja na rad memorijskog kontrolera, podešavanje vrijednosti može dovesti do nepredviđenog rada. Registar se ovdje nalazi greškom.
DYNMREF	0x0032
DYNMRCON	*Command delayed strategy*
PRECHARGE	1

DYNM2PRE	2
REFEXIT	4
DOACTIVE	2
DIACTIVE	4
DWRT	2
DYNACTCMD	3
DYNAUTO	4
DYNREFEXIT	4
DYNACTIVEAB	1
DYNAMICMRD	2
DYNCFG0	• Zaštita od upisa – onemogućiti • Ulazno-izlazni međuspremnici – omogućiti • Širina vanjske podatkovne sabirnice – 16b • Tip vanjske sabirnice – *High performance* • Veličina SDRAM-a – 64 Mbit • Organizacija SDRAM-a – x16 • Tip memorijskog uređaja – SDRAM
DYNRASCAS0	Postaviti RAS i CAS na dva perioda CCLK takta

Podešavanjem parametara na ovaj način *start-up* kod generiran dijelom iz konfiguracijskog wizarda će ispravno podesiti memorijski kontroler i SDRAM memoriju osim jednog bitnog djela. Ako se detaljnije pogleda Tablica 4.31,može se primjetiti da ne postoji nigdje opcija kojom se bira duljina *bursta*. Memorijski kontroler uvijek radi u *burstu* duljine 8, međutim konfiguracijski registar ovo ne zna te će on podesiti SDRAM memoriju, kroz LOAD REGISTER naredbu, da radi s duljinom *bursta* 1, odnosno da uopće ne radi u *burst* načinu rada. Ovaj se problem može zaobići pisanjem vlastite inicijalizacije ili ručnim mijenjanjem koda kojeg generira konfiguracijski wizard. Za sada je odabran drugi način.

Potrebno je otvoriti datoteku LH79525.s (donji lijevi kut, *Text editor*) te pronaći liniju:

```
SDRAM0_MODE_REG     EQU 0x20011000      ; SDRAM0 Mode Register Address
```

I zamijeniti je s:

```
SDRAM0_MODE_REG     EQU 0x20011800      ; SDRAM0 Mode Register Address
```

Sada je osigurana pravilna inicijalizacija SDRAM memorije.

4.7.1.2. Rad s SDRAM memorijom

Za razliku od inicijalizacije, rad s SDRAM memorijom je daleko jednostavniji. Memorija nema posebne zahtjeve na rad, jednom kada je inicijalizirana koristi se kao najobičnija memorija.

4.7.2. Rad sa statičkom memorijom

Prije pojašnjavanja flash i SRAM memorija, potrebno je reći par riječi općenito o statičkoj memoriji. Prva stvar koje je potrebno biti svjestan prilikom rada s memorijom je automatsko posmicanje adrese.

Poznato je da je memorija mikrokontrolera 8-bitna. Ovo znači da je najmanja jedinica koju je moguće adresirati jedan bajt. Ovo ne vrijedi samo za mikrokontroler, ovakva praksa je uobičajena u mnogim sustavima, od najjednostavnijih mikrokontrolera, pa do PC računala. Slika 4.11 prikazuje jednu takvu tipičnu organizaciju memorije.

Adresa	Sadržaj
0x00000000	0x00
0x00000001	0x0A
0x00000002	0x0B
0x00000003	0x0C
0x00000004	0x12
0x00000005	0x13
0x00000006	0x14
0x00000007	0x15

Slika 4.11 Primjer 8-bitne memorije

Iako je prikazana memorija 8-bitna, za tipične 32-bitne procesore nije uobičajeno da rade s podacima širine 1 bajt već tipično 4 bajta. Primjerice, zamišljena instrukcija LDR R0, #0x00 koja učitava 32-bitni podatak u registar R0 će rezultirati dohvatom podataka s adresa 0x00, 0x01, 0x02 i 0x03 te njihovim spremanjem u 32-bitni R0.

Sada zamislimo da je na vanjsku adresnu sabirnicu spojena memorija koja nije 8-bitna, već primjerice 16-bitna. Primjer takve memorije je prikazan Slika 4.12.

Adresa	Sadržaj
0x00000000	0x0000
0x00000001	0x0A0B

0x00000002	0x0B0C
0x00000003	0x0C0D
0x00000004	0x1234
0x00000005	0x1312
0x00000006	0x1498
0x00000007	0x1556

Slika 4.12 Primjer 16-bitne memorije

Prva stvar kojoj se memorijski kontroler sada mora prilagoditi je ta da za 32-bitni podatak ne čita 4 memorijske lokacije već dvije. Drugi problem koji postoji je adresna sabirnica. Neka je vanjska adresna sabirnica spojena 1:1 s unutarnjom, odnosno, memorijski kontroler tretira vanjski adresni prostor jednako kao i unutarnji s tom razlikom da ipak zna da je potrebno učitati samo dvije memorijske lokacije. Ako mikrokontroler želi podatak s adrese 0x00, dobit će upravo njega. Međutim, sljedeći 32-bitni podatak mikrokontroler će tražiti na adresi 0x04 s obzirom da se na adresama 0x00:0x03 nalazi prvi podatak (ovo ponašanje nije samo rezultat arhitekture mikrokontrolera već i kompajlera, no radi jednostavnije objašnjavanja neki detalji su izostavljeni). Ako mikrokontroler vanjskoj memoriji da adresu 0x04 za sljedeći podatak, jasno je da je izgubljen podatak na adresi 0x02. Ono što bi bilo potrebno u ovakvom slučaju je posmaknuti adresu za jedan u desno, znači da se za adresu 0x04 adresira podatak na adresi 0x02. Prije se ovo obično rješavalo na način da bi se adresna linija A1 spojila na adresni priključak memorije A0, A2 bi se spojila na A1 itd. Signal adresne sabirnice A0 bi ostao u zraku. Danas je normalno da su memorijski kontroleri inteligentniji po ovom pitanju. Primjerice, za prikazani sustav, odnosno adresiranje 16-bitne memorije memorijski kontroler će automatski posmaknuti adresu za jedan u desno, odnosno, zanemarit će adresni signal A0, a na njegovo mjesto će proslijediti A1. Da je ciljana memorija 32-bitna, posmak bi bio za 2 mjesta u desno pa bi bitovi A0 i A1 bili zanemareni. Ovakav način rada će biti od posebnog značaja prilikom rada sa flash memorijom.

4.7.3. Flash memorija

U sustavu se nalazi flash memorija konfiguracije 8Mx16 (8M lokacija po 16 bita). Vremenske karakteristike memorije su dane tablicom Tablica 4.32.

Tablica 4.32 Vremenske karakteristike flash memorije

Parametar	Trajanje
Vrijeme čitanja	100 ns
Vrijeme programiranja jedne lokacije	60 µs
Vrijeme programiranja jedne lokacije uz korištenje međuspremnika i algoritma za programiranje više lokacija	15 µs
Vrijeme brisanja sektora (64K riječi)	0.5 s

Memorija je podijeljena u 128 sektora, svaki veličine 128 Kb. Tablica 4.33 prikazuje podjelu po sektorima i početne adrese:

Tablica 4.33 Podjela po sektorima i početne adrese sektora

Veličina sektora	Broj sektora	Sektori	Adresni opseg
64 K riječi/128 Kb	128	SA00	0x0000000 – 0x000FFFF
		SA01	0x0010000 – 0x001FFFF
		….	
		SA127	0x07F0000 –0x7FFFFF

Inicijalizacija flash memorije nije ni približno kompleksna kao inicijalizacija primjerice SDRAM memorije. Preciznije, inicijalizacija flash memorije se svodi na inicijalizaciju memorijskog kontrolera. Registri i pripadne vrijednosti za podešavanje kroz konfiguracijski wizard su dani u Tablica 4.34.

Tablica 4.34 Registri i pripadne vrijednosti potrebni za inicijalizaciju memorijskog kontrolera za rad s flash memorijom

Registar	Vrijednost
EMC – CONTROL	Omogućiti memorijski kontroler Onemogućiti rad u režimu niske potrošnje
EMC – CONFIG	Ne dirati sadržaj ovog registra, registar nema utjecaja na rad memorijskog kontrolera, podešavanje vrijednosti može dovesti do nepredviđenog rada. Registar se ovdje nalazi greškom.
WAIT	0
SCONFIG1	Zaštita od upisa – onemogućeno Međuspremnik za čitanje i pisanje – vidi poglavlje [ref rad s flash] Produženo vrijeme čekanja – onemogućeno *Byte lane* – aktivni nisko Polaritet CS – aktivan nisko Stranični način rada – onemogućeno Širina memorije – vidi poglavlje [isto ko i ovo gore]

SWAITWEN1	
SWAITOEN1	
SWAITRD1	
SWAITPAGE1	
SWAITWR1	
STURN1	

4.7.3.1. Rad s flash memorijom

Rad s flash memorijom je nešto složeniji od rada s drugim memorijama u sustavu. Ako će se flash memorija samo čitati, opcija *Međuspremnik za čitanje i pisanje* se može postaviti po volji, kao što i ime govori, ovo će omogućiti međuspremnik, u ovom slučaju samo za čitanje. Širina memorije se u ovom slučaju postavlja na 16 bita, tj. onoliko koliko stvarno i je.

U slučaju da se u memoriju želi pisati, situacija je drukčija. Međuspremnik se obavezno mora isključiti, a širina memorije se mora postaviti na 32 bita. Razlog ovome je sljedeći: na početku ovog poglavlja je rečeno da memorijski kontroler uvijek obavlja 32-bitne transakcije prema memoriji. U slučaju memorije koja je 16 bitna, to će rezultirati s dva pristupa. U slučaju čitanja flash memorije, ovo je sasvim regularno, prvo će se pročitat jedna lokacija, potom sljedeća, dva 16-bitna podatka će se spojiti u 32-bitni koji će se proslijediti jezgri i sve je u redu. Međutim, kod pisanja je posve drukčija stvar. Flash memorija za upis, brisanje ili bilo koju drugu operaciju osim čitanja zahtjeva slanje predefiniranih podataka na određene adrese, ovo se nazivaju komande. Primjerice, kako bi se upisao podatak u memoriju prvo je potrebno na adresu 0x555 poslati podatak 0xAA. Ako je širina podatkovne sabirnice postavljena na 16 bita, memorijski kontroler će na adresu 0x555 poslati 0x00AA, a potom će na adresu (0x555 + 1) poslati 0x0000, odnosno, gornju riječ 32-bitnog podatka. U ovom slučaju, sekvenca koju flash memorija zahtijeva za programiranje je prekinuta. Uz postavke 16-bitne sabirnice je nemoguće provoditi bilo koje operacije nad memorijom osim čitanja.

No, ako se širina podatkovne sabirnice postavi na 32-bita, memorijski kontroler će na adresu 0x555 poslati podatak 0x0000 00AA. Iako gornjih 16 bitova na sabirnici uopće

ne postoji, memorijski kontroler to ne zna, te u ovom slučaju neće imati potrebu za dvjema transakcijama jer je upravo poslao jednu 32-bitnu.

Čitanje 32 bitnog upisa!

Za olakšan rad s flash memorijom su razvijene funkcije koje se velikim djelom oslanjaju na biblioteku funkcija koju je razvio proizvođač. Rad s funkcijama je objašnjen u poglavlju 4.14.5.

4.7.3.2. Korištenje FlashWriter01 programa

Za olakšan rad s flash memorijom, osim razvijenih funkcija je razvijen i program za olakšano upisivanje i čitanje flash memorije. Kroz jednostavan primjer programiranja flash memorije s novim programom puniocem će biti dan primjer rada s programom.

Pokrenuti Tera Term i podesiti postavke veze kako je danu u tablici Tablica 4.20. Pripremiti sustav za učitavanje programa putem UART-a kako je objašnjeno u poglavlju 4.3 i potom isprogramirati sustav s datotekom [datotek, dir] kako je objašnjeno u prethodno navedenom poglavlju. Ispis terminala je dan u nastavku:

```
************************************************************
*                                                          *
*                     FLASHWRITER v01                      *
*                                                          *
************************************************************

>Opcije:
        'w' - pisanje
        'r' - citanje
```

Odabrati opciju upisa u flash memoriju. Prvo se postavlja početna adresa od koje kreće upis. Adresa se može unijeti dekadski ili heksadekadski, u tom je slučaju broju potrebno staviti prefiks 'x' (npr. x1000). Ako dođe do greške prilikom upisa adrese, ili bilo koje druge vrijednosti, nema veze, ionako sve unešene vrijednosti treba potvrditi. Ispis unosa adrese je dan u nastavku:

```
>w

>Pisanje
>Unesite pocetnu adresu
>0
>Adresa: 0x00000000
>Unesite 'd' ako je adresa ispravna, bilo sto drugo inace
>d
```

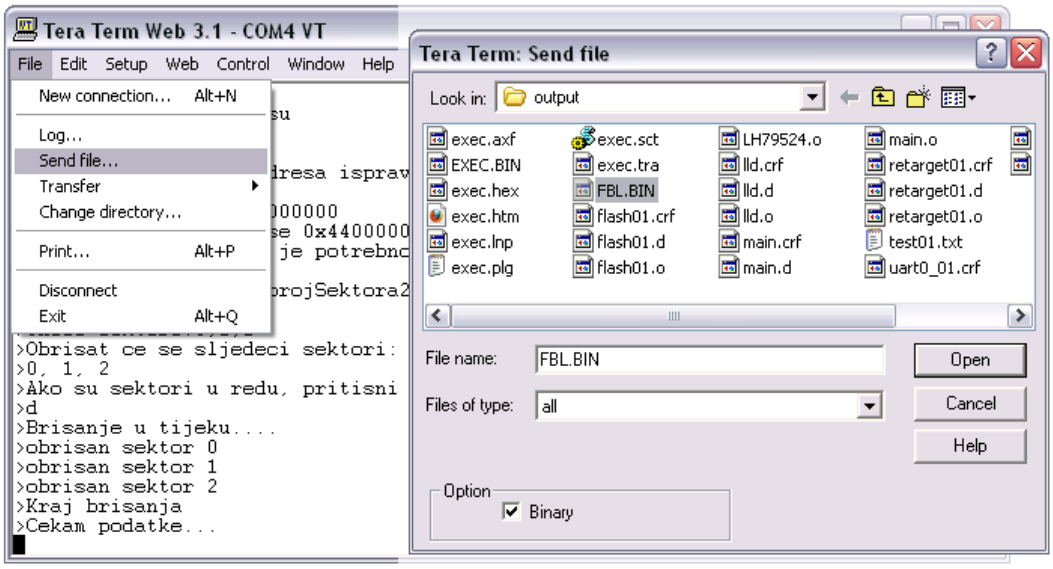
```
>Pocetna adresa: 0x00000000
>Upisat ce se od adrese 0x44000000
```

Pocetna adresa se prikazuje u relativnom obliku u flash memoriju, linija „Upisat ce se od adrese“ označava adresu u sustavu od koje kreće upis. U ovom primjeru je cilj snimiti novi program punilac u sustav, pa je odabrana adresa 0.

Sljedeće što slijedi je unos sektora u flash memoriji koje je potrebno obrisati prije upisa. Iako je dovoljan samo prvi sektor, primjera radi će se obrisati njih 5. Maksimalan broj sektora koji se mogu upisati na ovaj način je 30, za veći broj sektora je potrebno izmjeniti programski kod FlashWriter01 i ponoviti prevođenje, ili se snaći sa postojećim, primjerice opetovanim pokretanjem. Unose se indeksi sektora, a ne njihove adrese.

```
>Unesite sektore koje je potrebno izbrisati prije upisa u formatu:

        brojSektora1,brojSektora2,(...),brojSektoraN[enter]

>Unesi sektore:0,1,2,3,4
>Obrisat ce se sljedeci sektori:
>0, 1, 2, 3, 4
>Ako su sektori u redu, pritisni 'd', bilo sto inace
>d
>Brisanje u tijeku....
>obrisan sektor 0
>obrisan sektor 1
>obrisan sektor 2
>obrisan sektor 3
>obrisan sektor 4
>Kraj brisanja
>Cekam podatke...
```

Nakon što su sektori obrisani, program čeka da počnu stizat podaci. Podaci se prenose na jednostavan način, a to je slanjem podataka (bajtova) putem serijske veze. Podaci se mogu čak i slati tipkovnicom, ali je zato potrebno povećati TIME_TO_LEAVE konstantu u kodu za red veličine 10, ili i više, kako program ne bi prebrzo zaključio da više nema podataka. Kako se programira novi program punilac, a srećom tako što ne treba programirat putem tipkovnice/prekidača/trake, odabrati opciju u izborniku *File → Send File…* te u direktoriju [] odabrati datoteku []. Slika 4.13 prikazuje izbornik i rezultantni dijalog.



Slika 4.13 Glavni izbornik programa

Odabrati datoteku i obavezno označiti opciju *Binary* koja se može naći na dnu. Nakon što se datoteka odabere, slijedi slanje:

```
1
2
3
4
5
>Kraj ucitavanja podataka, ucitano je 2408 rijeci

>Opcije:
        'w' - pisanje
        'r' - citanje
>
```

Odbrojavanje označava koliko je puta petlja unutar programa došla do TIME_TO_LEAVEa. Nakon kraja učitavanja ponovo se može pristupiti programiranju ili čitanju memorije.

Program ne provjerava podatke koje je poslao, niti omogućuje zaštitu ili detekciju greške. Razlog ovome je i taj što je već program na granici 4K, a mora ostati kompaktan kako bi ga se moglo podići putem UART-a s obzirom da služi između ostalog i programiranju novog programa punioca.

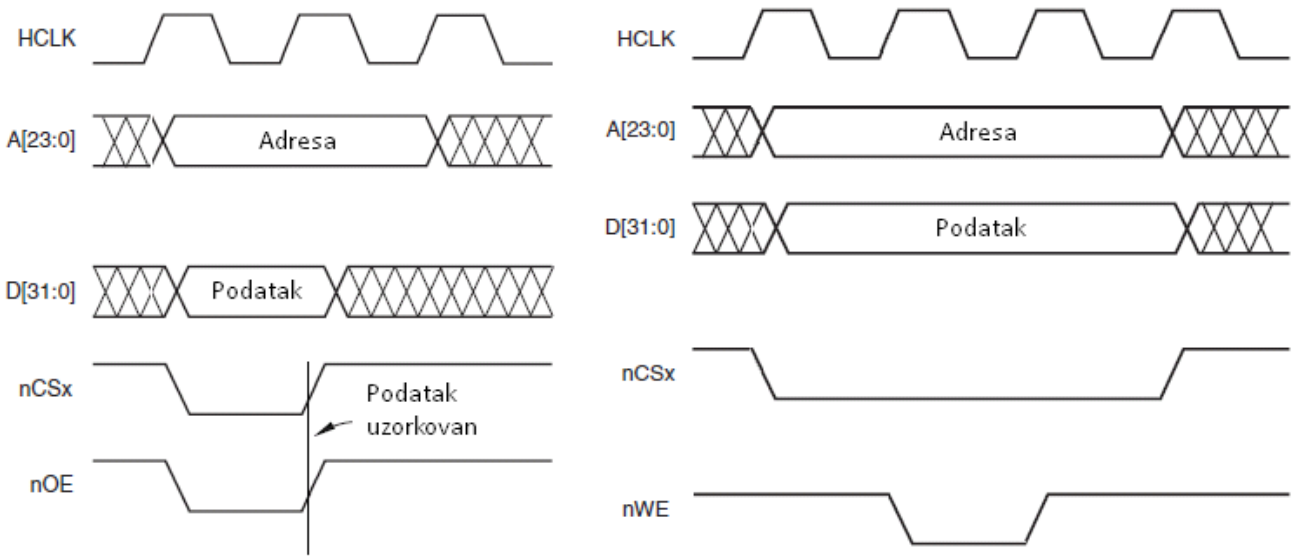
4.7.4. Rad s SRAM memorijom

SRAM memorija je među bržim memorijama u sustavu, deklarirano vrijeme pristupa (čitanje/pisanje) joj je 10 ns te je ograničena samo brzinom memorijskog kontrolera. SRAM memorija također ne zahtijeva nikakvu inicijalizaciju (osim inicijalizacije pripadnog CS u memorijskom kontroleru) te ne pati od specifičnih ponašanja za vrijeme rada. Tablica 4.35 prikazuje inicijalizaciju memorijskog kontrolera za dani CS.

Tablica 4.35 Inicijalizacija memorijskog kontrolera za SRAM (CS0)

Registar	**Vrijednost**
EMC – CONTROL	Omogućiti memorijski kontroler Onemogućiti rad u režimu niske potrošnje
EMC – CONFIG	Ne dirati sadržaj ovog registra, registar nema utjecaja na rad memorijskog kontrolera, podešavanje vrijednosti može dovesti do nepredviđenog rada. Registar se ovdje nalazi greškom.
WAIT	0
SCONFIG0	Zaštita od upisa – onemogućeno Međuspremnik za čitanje i pisanje – onemogućiti Produženo vrijeme čekanja – onemogućeno *Byte lane* – aktivni nisko Polaritet CS – aktivan nisko Stranični način rada – onemogućeno Širina memorije – 16 bita
SWAITWEN0	0
SWAITOEN0	0
SWAITRD0	1
SWAITPAGE0	Ne koristi se
SWAITWR0	0
STURN0	1

Rezultanti ciklusi čitanja i pisanja prikazani su slikom Slika 4.14.



Slika 4.14 Ciklusi čitanja i pisanja u SRAM memoriju

Lijeva strana slike prikazuje čitanje SRAM-a, a desna pisanje u SRAM.

4.8. Rad s LCD kontrolerom (VGA)

U radu je LCD kontroler korišten za upravljanje VGA konektorom odnosno VGA sučeljem. LCD kontroler omogućuje:

- Upravljanje sljedećim tipovima LCD ekrana:
  - TFT, HR-TFT, AD-TFT
  - STN:
    - Jednobojni
    - U boji
    - Obični
    - Dvostrani
- Teoretska do 1024x1024, realno 640x480 maksimalno
- LH79525 – 12 bitno podatkovno sučelje (4 bita po boji), LH79524 – 16 bitno
- Bogate opcije programiranja vremenskih odnosa upravljačkih signala

S LCD kontrolerom se radi na način da se željeni LCD spoji na odgovarajuće priključke mikrokontrolera te se potom kroz registre LCD kontrolera odaberu odgovarajući vremenski odnosi, odnosno, brzina takta LCD-a, rezolucija, širina

podatkovnog sučelja… Izvor podataka za LCD kontroler se nalazi na proizvoljnom mjestu u memoriji, prilikom inicijalizacije potrebno je LCD kontroleru zadati pokazivač na mjesto u memoriji odakle počinje sadržaj ekrana. Količina potrebne memorije ovisi o rezoluciji i broju bitova po pikselu.

Za čitanje memorije i slanje podataka na izlazne priključke, LCD kontroler koristi DMA. Prilikom rada s LCD kontrolerom potrebno je imati na umu da DMA bez obzira na sve uzima podatke iz memorije onim tempom kojeg određuje brzina LCD-a odnosno njegova rezolucija i frekvencija osvježavanja. Podaci se potom pohranjuju u međuspremnik dubine 32 riječi po 32 bita.

Prilikom rada s LCD kontrolerom također valja imati na umu i opterećenje sustava. Primjerice, VGA sučelje znatno opterećuje mikrokontroler. Standardna rezolucija koju VGA standard propisuje je 640x480. Uz standardnu frekvenciju osvježavanja ekrana od 60 Hz, piksel takt je u tom slučaju 25 MHz što znači da bi memorija morala upravo na ovom taktu davat podatke. Jasno je da ovako nešto gotovo u potpunosti zagušuje AHB sabirnicu.

Ipak, sustav se može olakšat na način da se smanji broj bita boje po pikselu a time i potreban broj čitanja memorije. Tablice Tablica 4.36 i Tablica 4.37 prikazuje moguće načine spremanja piksela u memoriju:

Tablica 4.36 Format piksela u memoriji – gornja riječ

BPP	Format podataka u memoriji															
1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16
2	p15		p14		p13		p12		p11		p10		p9		p8	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p7				p6				p5				p4			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
12	p1															
		11	10	9	8		7	6	5	4		3	2	1	0	
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tablica 4.37 Format piksela u memoriji – donja riječ

BPP	Format podataka u memoriji															
1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
2	p7		p6		p5		p4		p3		p2		p1		p0	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p3				p2				p1				p0			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
12	p0															
		11	10	9	8		7	6	5	4		3	2	1	0	
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Pikseli se u memoriji pohranju slijedno.

LCD kontroler, osim LCD kontrolera, nudi i ALI – *Advanced LCD Interface.* ALI se koristi u slučaju da je potrebnije generirati složenije vremenske odnose od onih koji su mogući putem LCD kontrolera. ALI također nudi i niz kontrolnih signala koji se koriste za sekvenciranje napajanja LCD modulu i slično.

U narednom poglavlju će biti dane kratke upute za rad s LCD kontrolerom.

4.8.1. Rad s LCD kontrolerom

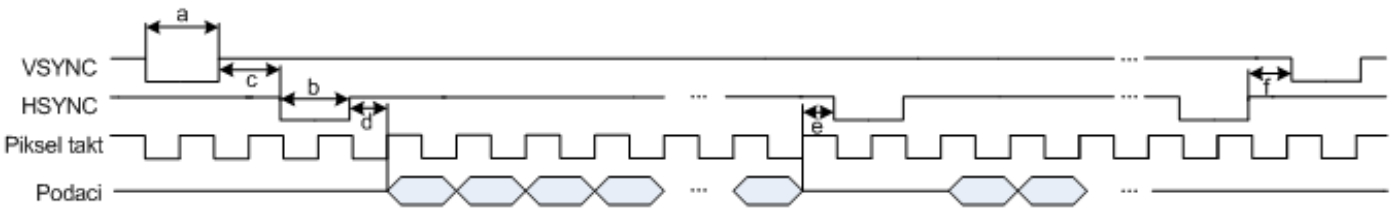
LCD kontroler zahtijeva samo inicijalizaciju, jednom kada je pravilno inicijaliziran nema potrebe mu više pristupati. Registri koje je potrebno programirati za korištenje LCD kontrolera su dani tablicom Tablica 4.38.

Tablica 4.38 Registri povezani s radom LCD kontrolera

Registar	Adresa	Opis
MUXCTL19, 20, 21, 22	I/O kontroler	Dodjeljivanje priključaka LCD kontroleru
PCLKCTRL1	RCPC kontroler	Omogućavanje takta LCD kontroleru
AHBCLKCTRL	RCPC kontroler	Omogućavanje takta LCD kontroleru

LCDPRE	RCPC kontroler	Djelilo takta LCD kontroleru
TIMING0	LCD kontroler	Kontrola horizontalnih vremenskih odnosa
TIMING1	LCD kontroler	Kontrola vertikalnih vremenskih odnosa
TIMING2	LCD kontroler	Kontrola takta i polariteta signala
UPBASE	LCD kontroler	Pokazivač na video memoriju
CTRL	LCD kontroler	Opće kontrolni registar

Prvo što je potrebno je dodijeliti priključke LCD kontroleru te mu dodijeliti takt, kao i postavit vrijednost takta. Takt koji se dovodi LCD kontroleru prije preddjelila je jednak taktu HCLK. Nakon što je ovo obavljeno, može se pristupiti programiranju TIMING0 registra. No prije toga potrebno se upoznati s formatom upisa u LCD ekran te kontrolnim signalima. Slika 4.15 prikazuje generički dijagram vremenskih odnosa kontrolnih signala LCD ekrana tipa TFT.



Slika 4.15 Dijagram vremenskih odnosa LCD ekrana tipa TFT

Značenje vremenskih odnosa je dano tablicom Tablica 4.39.

Tablica 4.39 Opis vremenskih odnosa LCD ekrana tipa TFT

Parametar	**Opis**
a	Duljina vertikalnog sinkronizacijskog pulsa
b	Duljina horizontalnog sinkronizacijskog pulsa
c	*Vertical front porch*- vrijeme od deaktivacije VSYNC-a do aktiviranja signala HSYNC
d	*Horizontal front porch* – vrijeme od deaktivacije signala HSYNC do početka toka podataka
e	*Horizontal back porch* – vrijeme od kraja toka podataka jedne linije do ponovne aktivacije signala HSYNC
f	*Vertical back porch* – vrijeme od kraja toka podataka (linija) do ponovne aktivacije signala VSYNC

Sučelje je razmjerno jednostavno, padajući brid signala VSYNC (također se naziva i *Frame Pulse*) označava početak iscrtavanja novog ekrana. Nakon njega, padajući brid signala HSYNC (*Line Pulse*) označava iscrtavanje nove linije. Kada je dan signal

iscrtavanja nove linije, sljede podaci o boji za svaki pojedini piksel linije u onom redoslijedu kako se nalaze na ekranu. Nakon što je završeno iscrtavanje linije, ponovo slijedi aktiviranje signala HSYNC koji označava iscrtavanje nove linije, pa sve tako dok ima linija. Konačno, kada je iscrtan cijeli ekran, ponovno se aktivira signal VSYNC te se sve ponavlja. Između aktiviranja pojedinih signala i akcija koje oni uzrokuju umeću se vremena čekanja pod nazivima *Vertical front porch, Horizontal front porch, Horizontal back porch* i *Vertical back porch.* Sada se može pristupiti objašnjenju pojedinih registara LCD kontrolera.

Kroz TIMING0 registar se namještaju horizontalni vremenski odnosi. Registar je prikazan tablicom Tablica 4.40 a opis pojedinih polja tablicom Tablica 4.41.

Tablica 4.40 Registar TIMING0 LCD kontrolera

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Polje	HBP								HFP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	HSW								PPL						//	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFF4000 + 0x00															

Tablica 4.41 Opis polja registra TIMING0 LCD kontrolera

Bitovi	**Ime**	**Opis**
31:24	HBP	*Horizontal Back Porch* – Duljina HBP-a u periodama LCDDCLK. Označava broj perioda LCDDCLK nakon kraja LCDLP signala do početka toka podataka o vrijednostima piksela. HBP = LCDDCLK – 1 Za HBP == 48 potrebno je upisati 49
23:16	HFP	*Horizontal Front Porch* – Duljina HFP-a u periodama LCDDCLK. Označava broj perioda LCDDCLK nakon kraja toka podataka o vrijednostima piksela do aktivacije signala LCDLP. HFP = LCDDCLK – 1 Za HFP == 16 potrebno je upisati 17
15:8	HSW	*Horizontal Synchronization Pulse Width* – Duljina LCDPS u periodama LCDDCLK. HSW = LCDDCLK – 1

7:2	PPL	*Pixels Per Line* – Broj piksela po liniji. U ovu vrijednost ne ulaze HFP i HBP. Uvjet za PPL je da bude djeljiv sa 16. PPL = (Broj vidljivih piksela po liniji/16) – 1 Za Broj vidljivih piksela po liniji == 640, PPL = 39

Sljedeći registar koji je potrebno isprogramirati je TIMING1 i pripadajuće vertikalne vremenske odnose. Tablicama Tablica 4.42 iTablica 4.43 je dana građa registra.

Tablica 4.42 Registar TIMING1 LCD kontrolera

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Polje	VBP								VFP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	VSW						LPP									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFF4000 + 0x04															

Tablica 4.43 Opis pojedinih polja registra TIMING1 LCD kontrolera

Bitovi	**Ime**	**Opis**
31:24	VBP	*Vertical Back Porch* – Broj neaktivnih linija prije između kraja aktivacije signala LCDFP do aktivacije signala LCDLP. VBP definira broj LCDLP pulseva koji se ubacuju prije toka podataka.
23:16	VFP	*Vertical Front Porch* – Broj neaktivnih linija na kraju toka podataka, a prije aktiviranja signala LCDFP.
15:10	VSW	*Vertical Synchronization Pulse Width* – Duljina LCDFP u periodama LCDPS. VSW = LCDPS – 1 Za VSW == 2, VSW = 3
8:0	LPP	*Lines Per Panel* – Broj vidljivih linija po ekranu. LPP = Broj vidljivih linija – 1 Za broj vidljivih linija == 480, LPP = 479

Sljedeći na redu je registar TIMING2. TIMING2 je prikazan tablicama Tablica 4.44 i Tablica 4.45.

Tablica 4.44 Registar TIMING2 LCD kontrolera

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Polje	PCD_HI					BCD	CPL									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	//	IOE	IPC	IHS	IVS	ACB					//	PCD_LO				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFF4000 + 0x08															

Tablica 4.45 Opis pojedinih polja registra TIMING2 LCD kontrolera

Bitovi	Ime	Opis
31:27	PCD_HI	Zajedno s PCD_LO čini PCD – *Panel Clock Divisor.* Daje vrijednost CLCP takta odnosno takta za LCD ekran. CLCP = CLCDCLK/(PCD+2)
26	BCD	Zaobilaženje logike za dijeljenje CLCDCLK takta 1 – zaobiđi logiku za djeljenje CLCDCLK takta 0 – omoguću logiku za djeljenje CLCDCLK takta
25:16	CPL	*Clocks Per Line* – Broj perioda takta proslijeđen LCD ekranu putem priključka LCDCLK. BVPPL (Broj Vidljivih Piksela Po Liniji) = 16 x (TIMING0:PPL + 1) CPL = (BVPPL – 1) Za BVPPL = 640, CPL = 639
14	IOE	Polaritet signala LCDEN 1 – LCDEN – aktivan nisko 0 – LCDEN – aktivan visoko
13	IPC	Polaritet takta LCD ekrana – LCDCLK 1 – Podaci se šalju na padajući brid signala LCDCLK 0 – Podaci se šalju na rastući brid signala LCDCLK
12	IHS	Polaritet LCDLP signala 1 – LCDLP je aktivan nisko 0 – LCDLP je aktivan visoko
11	IVS	Polaritet LCDFP signala 1 – LCDFP je aktivan nisko 0 – LCDFP je aktivan visoko
10:6	ACB	AC *Bias Signal Frequency* – signal važeći samo za STN ekrane, za detalje pogledati [ref veliki]
4:0	PCD_LO	Pogledati PCD_HI

Sada je potrebno programirati registar UPBASE (0xFFFF4000 + 0x14) te CTRL registar. U registar UPBASE je potrebno upisati adresu na kojoj počinju podaci o slici. Građa CTRL registra je dana tablicama Tablica 4.46 i Tablica 4.47.

Tablica 4.46 Registar TIMING2 LCD kontrolera

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Polje	//															WATERMARK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Polje	//		VCI		PWR	BEPO	BEBO	BGR	DUAL	MONO8L	TFT	BW	BPP			LCDEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Adresa	0xFFFF4000 + 0x1C															

Tablica 4.47 Opis pojedinih polja registra CTRL LCD kontrolera

Bitovi	**Ime**	**Opis**
16	WATERMARK	Razina popunjenosti međuspremnika pri kojoj DMA traži nove podatke 1 – Traži podatke kada međuspremnik ima 8 ili više praznih lokacija 0 – Traži podatke kada međuspremnik ima 4 ili više prazne lokacije
13:12	VCI	Generiranje prekida pri: 00 – početak vertikalne sinkronizacije 01 – početak *back porcha* 10 – početak aktivne slike 11 – početak *front porcha*
11	PWR	Omogućavanje napajanja LCD ekranu 1 – LCD napajanje omogućeno 0 – LCD napajanje onemogućeno Napomena: Ovaj bit mora biti postavljen kada god se koristi LCD kontroler
10	BEPO	*Big Endian Pixel Ordering* – odabire *big* ili *little* endian format zapisa za 1, 2 i 4 bita po pikselu. Nema utjecaja na 8, 12 i 16 bita po pikselu. 1 – *Big endian* poredak bitova u bajtu 0 – *Little endian* poredak bitova u bajtu

9	BEBO	*Big Endian Byte Order* – poredak poslanih podataka prema LCD ekranu u bajtovima 1 – *Big endian* poredak 0 – *Little endian* poredak
8	BGR	RGB ili BGR odabir formata 1 – bitovi 14:10 i 4:0 zamijenjeni (crvena i plava) 0 – normalan poredak boja
7	DUAL	Utjecaj samo na STN ekrane, za detalje pogledati [ref veliki]
6	MONO8L	Utjecaj samo na STN ekrane, za detalje pogledati [ref veliki]
5	TFT	TFT LCD: 1 – LCD ekran je tipa TFT 0 – LCD ekran je tipa STN
4	BW	Utjecaj samo na STN ekrane, za detalje pogledati [ref veliki]
3:1	BPP	Bitova po pikselu. LH79525, 12 bpp se bira postavljanjem u 0b100. 000 = 1 BPP 001 = 2 BPP 010 = 4 BPP 011 = 8 BPP 100 = 16 BPP 101 =nedopušteno 110 = nedopušteno 111 = nedopušteno
0	LCDEN	Omogućavanje LCD kontrolera 1 – LCD kontroler je omoguće 0 – LCD kontroler je onemogućen

4.8.2. LCD kontroler i VGA

U poglavlju 3.6.8 je dan kratak osvrt na korištenje LCD kontrolera za pogon VGA sučelja. Uz prethodno opisane sklopovske preinake, kako bi se postigao pogon VGA sučelja potrebno je podesiti LCD kontroler s parametrima danima u tablici Tablica 4.48.

Tablica 4.48 Vremenski parametri VGA standarda

Općeniti vremenski odnosi	
Frekvencija osvježavanja ekrana	60 Hz
Vertikalna frekvencija	31.46875 kHz
Frekvencija promjene piksela	25.175 MHz
Horizontalni vremenski odnosi	Piksela
Broj vidljivih piksela	640
Front porch	16
Duljina sinkronizacijskog pulsa	96
Back porch	48
Broj piksela u cijeloj liniji	800
Vertikalni vremenski odnosi	Linija

Vidljivih linija	480
Front porch	10
Duljina sinkronizacijskog pulsa	2
Back porch	33
Broj linija u cijelom okviru	525

U poglavlju 4.14.4 su dani opisi funkcija za olakšan rad s LCD kontrolerom.

4.9. Punjenje mikrokontrolera

Dosada je uglavnom bilo razmatrano punjenje mikrokontrolera putem UART sučelja. Ipak, ovaj način punjenja ima svojih mana, a najveća je ograničenost koda na veličinu do 4 KB. Osim punjenja putem UART sučelja, sustav se može puniti (odnosno, mikrokontroler u sustavu) i iz dostupne flash memorije. Flash memoriju je, prije punjenja iz nje, potrebno isprogramirati s nekim korisnim programom.

Sustav je zamišljen tako da se u flash memoriji nalazi program punilac – *bootloader.* Prilikom punjenja iz flash memorije, učitava se upravo program punilac koji je zadužen za učitavanje programa u neku od dostupnih memorija te početak izvršavanja programa. Sljedeća poglavlja se bave upravo *bootloaderom.* Punjenje iz flash memorije se postiže konfiguracijskom riječi 0b0000 na PC[7:4].

4.9.1. Rad s programom puniocem

Rad s programom puniocem je jednostavan, dovoljno je spojiti razvojni sustav s računalom putem serijskog kabela, u terminalu (preporuča se Tera Term) podesiti parametre komunikacije zadane u tablici [] te potom resetirati mikrokontroler. Potom se pojavljuje jednostavan izbornik prikazan slikom [] putem kojeg je potrebno odabrati željeni način punjenja te potom poslati binarnu datoteku koja sadrži program. Sljedeći primjer pokazuje rad s programom puniocem.

Ovdje primjer

4.9.2. Specifičnosti programske podrške koja će biti učitana u sustav putem programa punioca

Prilikom pisanja programske podrške koja će biti učitana putem programa punioca, važno je obratiti pozornost na nekoliko stvari. Prva je da se obavezno mora obratiti pozornost na to da se u projektu odaberu odgovarajuće adresne regije za programski i podatkovnu memoriju. Sljedeće, što je jako bitno je da programska podrška u svojoj inicijalizaciji apsolutno ne smije inicijalizirati memorijski kontroler. Naprimjer, neka je program učitan u SDRAM i neka se počeo izvršavati. Ako on u svojoj inicijalizaciji krene inicijalizirati između ostalog i dio memorijskog kontrolera odgovornog za SDRAM doći će do nepredviđenog i neispravnog rada. Ukratko, program nikada ne bi trebao inicijalizirati svoj izvor, bilo da se radi o SDRAM-u, vanjskom SRAM-u ili flashu.

Također je potrebno imati na umu da program punilac inicijalizira SDRAM i SRAM memoriju na načine opisane u poglavlju 4.7.

4.9.3. Programiranje flash memorije s novim programom puniocem

Vrlo česta stvar u razvoju je brisanje programa punioca. Srećom, u sustavu to nije niti najmanji problem te se rješava jako brzo. Dovoljno je slijediti upute dane u poglavlju 4.7.3.2.

4.10. AD konverter

[opisat način rada i uputi na direktorij s primjerom]

4.11. Ethernet kontroler – http server

[uputi na direktorij s rješenjem i opiši korištenje]

4.12. µSD kartica

[uputi na direktorij s rješenjem i opiši korištenje]

4.13. Vremenski sklopovi i prekidi

Direktorij [dir] sadrži primjer rada sa prekidnim kontrolerom i vremenskim sklopovima. Projekt korištenjem vremenskog sklopa i prekida kojeg generira pali i gasi svjetleće diode. Više i prekidnom kontroleru se može pročitati u poglavlju 18, a o vremenskim sklopovima u poglavlju 15 [ref velki datsahhet]

4.14. Razvijene funkcije

U radu su razvijene funkcije kako za olakšan rad, tako i za primjer korištenja nekih od perifernih jedinica u sustavu. U ovom poglavlju će biti dane upute za rad s njima. Sve su funkcije pisane za LH79525 razvojni sustav što znači da niti jednu datoteku nije potrebno prilagođavati sustavu.

4.14.1. Funkcije za rad s LE diodama

4.14.1.1. Datoteke

Datoteke potrebne za rad su `led01.c` i `led01.h`

4.14.1.2. Korisnički definirani tipovi podataka

- LED_TYPE:

```
typedef enum
{
      LED0  = 0x04,
      LED1  = 0x08,
      LED2  = 0x10,
      LED3  = 0x20,
      LED4  = 0x40,
      LEDALL      = 0x7C
} LED_TYPE;
```

LED_TYPE označava svaku od dioda, koristi se prilikom paljenja i gašenja dioda. Tablica 4.49 prikazuje vezu između LED_TYPE i LE dioda:

Tablica 4.49 Veza između LED_TYPE i LE dioda u sustavu

LED_TYPE	LE dioda
LED0	H200
LED1	H201
LED2	H202

LED3	H203
LED4	H204

4.14.1.3. Funkcije

`void LEDInit(void)`

Opis:

Priključke PE2:PE6 dodjeljuje ulazno-izlaznom kontroleru te ih postavlja u izlazne.

Primjer korištenja:

```
LEDinit();
```

`void LEDon(unsigned int LEDdata)`

Opis:

Pali diode zadane parametrom

Parametri:

- LEDdata – diode koje se želi upaliti, LEDdata se direktno preslikava na priključke PE2:PE6, primjerice, za paljenje diode na PE2 potrebno je proslijediti 0x04.

Primjer korištenja:

```
LEDon(LED0 | LED1 | LED4); // Pali H200, H201 i H204
LEDon(LEDALL); // Pali sve LE diode
```

`void LEDoff(unsigned int LEDdata)`

Opis:

Gasi diode zadane parametrom

Parametri:

Jednak kao i za LEDon

Primjer korištenja:

LEDoff(LED1 | LED4); // Gasi H201 i H204

LEDoff(LEDALL); // Gasi sve LE diode

4.14.2. Funkcije za rad s tipkama i prekidačima

4.14.2.1. Datoteke

Datoteke potrebne za rad su `button01.c` i `button01.h`

4.14.2.2. Korisnički definirani tipovi podataka

BUTTON:

```
typedef enum
{
      PJ7   = 0x70,
      PJ4   = 0xB0,
      PJ5   = 0xD0,
      PJ6   = 0xE0
} BUTTON;
```

DIPSW

```
typedef enum
{
      PJ0   = 0x0E,
      PJ1   = 0x0D,
      PJ2   = 0x0B,
      PJ3   = 0x07
} DIPSW;
```

BUTTON i DIPSW označavaju svaki od prekidača. Tablica 4.50 daje vezu između BUTTON i DIPSW.

Tablica 4.50 Veza između BUTTON, DIPSW i tipki i prekidača u sustavu u sustavu

Varijabla	Prekidač
PJ7	S202
PJ6	S206
PJ5	S205
PJ4	S203
PJ3	S201-1

PJ2	S201-2
PJ1	S201-3
PJ0	S201-4

4.14.2.3. Funkcije

```
int buttonsActive(void)
```

Opis:

Provjerava da li je neka od tipki stisnuta u trenu. Napomena1: ne provjerava prekidače. Napomena2: nije implementirano istitravanje.

Povratna vrijednost:

1 – ako je neka od tipki pritisnuta

0 – ako nije

Primjer korištenja:

```
if(buttonsActive()) printf(„Stisnuta je tipka“);
```

```
int buttons(void)
```

Opis:

Vraća stanje priključka PJ maskiranog tako da se vraćaju samo priključci na kojima su tipke.

Povratna vrijednost:

Stanje priključka PJ – samo bitova koji su povezani s tipkama

Primjer korištenja:

```
if(buttons() == PJ7) printf(„Stisnuta je tipka S202“);
```

```
int DIPSWActive(void)
```

Opis:

Provjerava da li je neki od prekidača u aktivnom položaju. Napomena1: ne provjerava tipke. Napomena2: nije implementirano istitravanje.

Povratna vrijednost:

1 – ako je neki od prekidača u aktivnom položaju

0 – ako nije

Primjer korištenja:

```
if(DIPSWActive()) printf(„Prekidač je u aktivnom položaju“);
```

```
int DIPSwitch(void)
```

Opis:

Vraća stanje priključka PJ maskiranog tako da se vraćaju samo priključci na kojima su prekidači.

Povratna vrijednost:

Stanje priključka PJ – samo bitova koji su povezani s prekidačima

Primjer korištenja:

```
if(DIPSwitch() == PJ0) printf(„Aktivan je prekidač S201-4“);
```

4.14.3. Funkcije za rad s UART-om

4.14.3.1. Datoteke

Potrebne datoteke su UART0_01.c i UART0_01.h

4.14.3.2. Funkcije

Void UART0init(void)

Opis:

Inicijalizira UART kontroler, kanal 0 s vrijednostima definiranima u UART0_01.h. Pretpostavljene postavke inicijalizacije:

- Omogućeno sklopovlje za slanje i primanje
- Omogućeni dolazni i odlazni međuspremnik
- Podatak: 8-bitni, 1 START, 1 STOP bit

Ovo su standardne postavke serijskog sučelja, za naprednije opcije pogledati [ref veliki]. Prije inicijalizacije je potrebno postaviti i željenu brzinu komunikacije što se može obavit u UART0_01.h u dijelu brzina. Ako su potrebne vrijednosti koje nisu među navedenima, pogledati [ref veliki].

Int UART0putchar(int ch)

Opis:

Čeka dok međuspremnik za slanje nema barem jedno slobodno mjesto te potom ubacuje podatak ch (donjih 8 bita) u spremnik. Čim podatak dođe na red, on se šalje.

Parametri:

Int ch – podatak koji se šalje putem serije (samo donjih 8 bita)

Povratna vrijednost:

Funkcija vraća poslani podatak

Primjer korištenja:

UART0putchar('a');

Int UART0getchar

Opis:

Čita prvi neobrađeni podatak iz dolaznog spremnika, ako je spremnik prazan,čeka dok podatak ne dođe, a ako se spremnik ne koristi čita posljednji primljeni neobrađeni podatak ili čeka dok ne stigne novi. Ako je definiran makro UART_ECHO_RECEIVED_CHAR, primljeni podatak se šalje natrag.

Povratna vrijednost:

Primljeni podatak

Primjer korištenja:

Char c;

C = (char)UART0getchar();

Int UART0dataReceived(void)

Opis:

Provjerava da li ima novih podataka u spremniku, poziv nije blokirajući

Povratna vrijednost:

1 – ako ima novih podataka u spremniku

0 – inače

Primjer korištenja:

If(UART0dataReceived()) c = UART0getchar();

int UART0receiveFIFOfull(void)

Opis:

Provjerava da li se dolazni spremnik potpuno popunio

Povratna vrijednost:

1 – ako je dolazni spremnik pun

0 – inače

Primjer korištenja:

If(UART0receiveFIFOfull()) printf(„Dolazni spremnik je pun“);

4.14.4. Funkcije za inicijalizaciju LCD kontrolera

4.14.4.1. Datoteke

Potrebne datoteke su vga01.c i vga01.h

4.14.4.2. Funkcije

Void VGAInit(void)

Opis:

Inicijalizira LCD kontroler s postavkama zadanima u vga01.h. Postavke su podešene za sučeljavanje LCD kontrolera s VGA sučeljem. Prije poziva funkcije može se izmjeniti parametar LCDUPPBASE u vga01.h koji predstavlja pokazivač na video memoriju. Funkcija ne pali LCD kontroler.

Void VGADumpRegs(void)

Opis:

Ispisuje stanje TIMING0, TIMING1, TIMING2, CTRL i UPBASE registara

Void VGAEnable(void)

Opis:

Omogućuje (pali) LCD kontroler

Void VGADisable

Opis:

Funkcija gasi LCD kontroler

Void VGASetVideoBuffer(unsigned int *buffer)

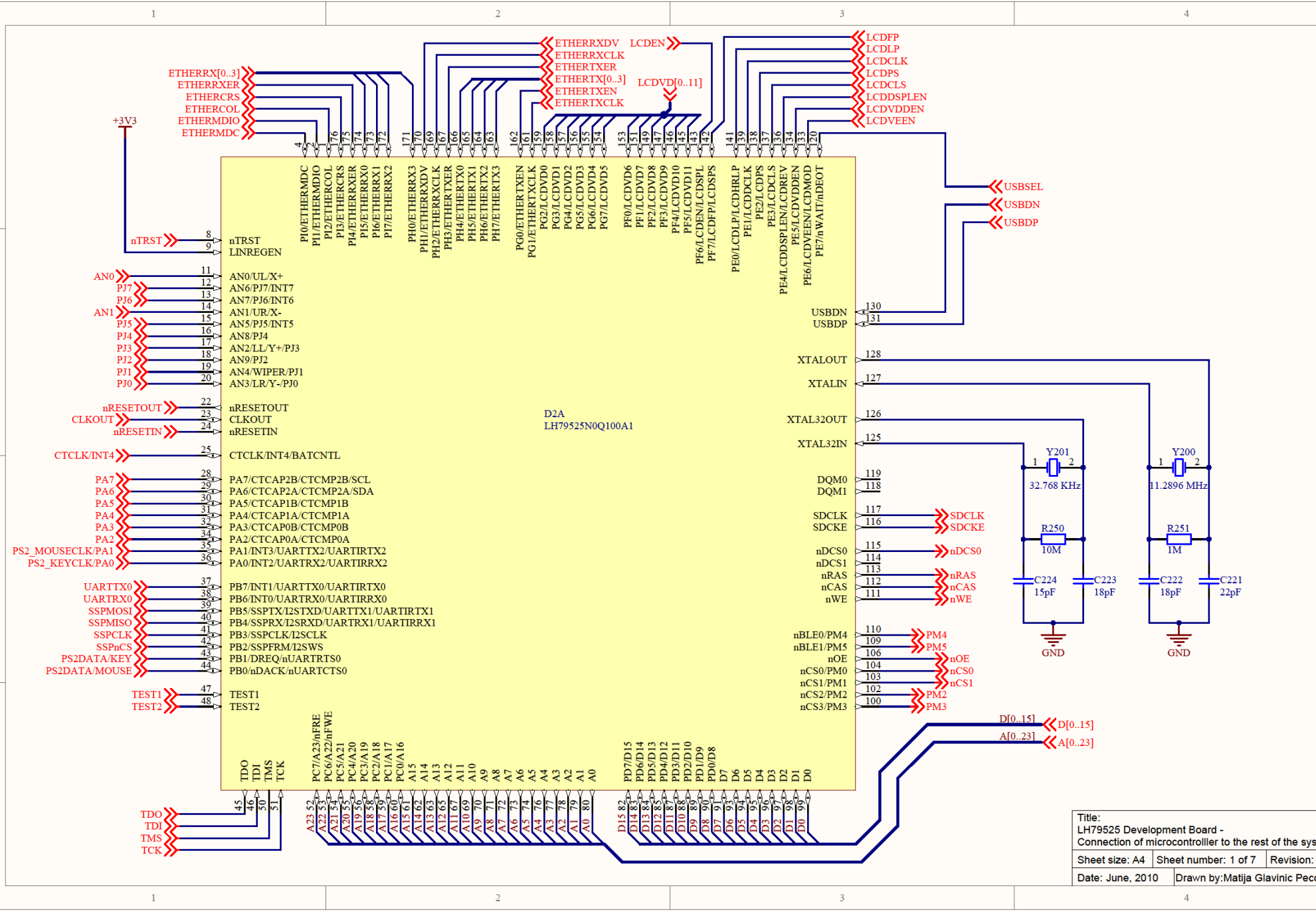
Opis:

Postavlja pokazivač na video memoriju.

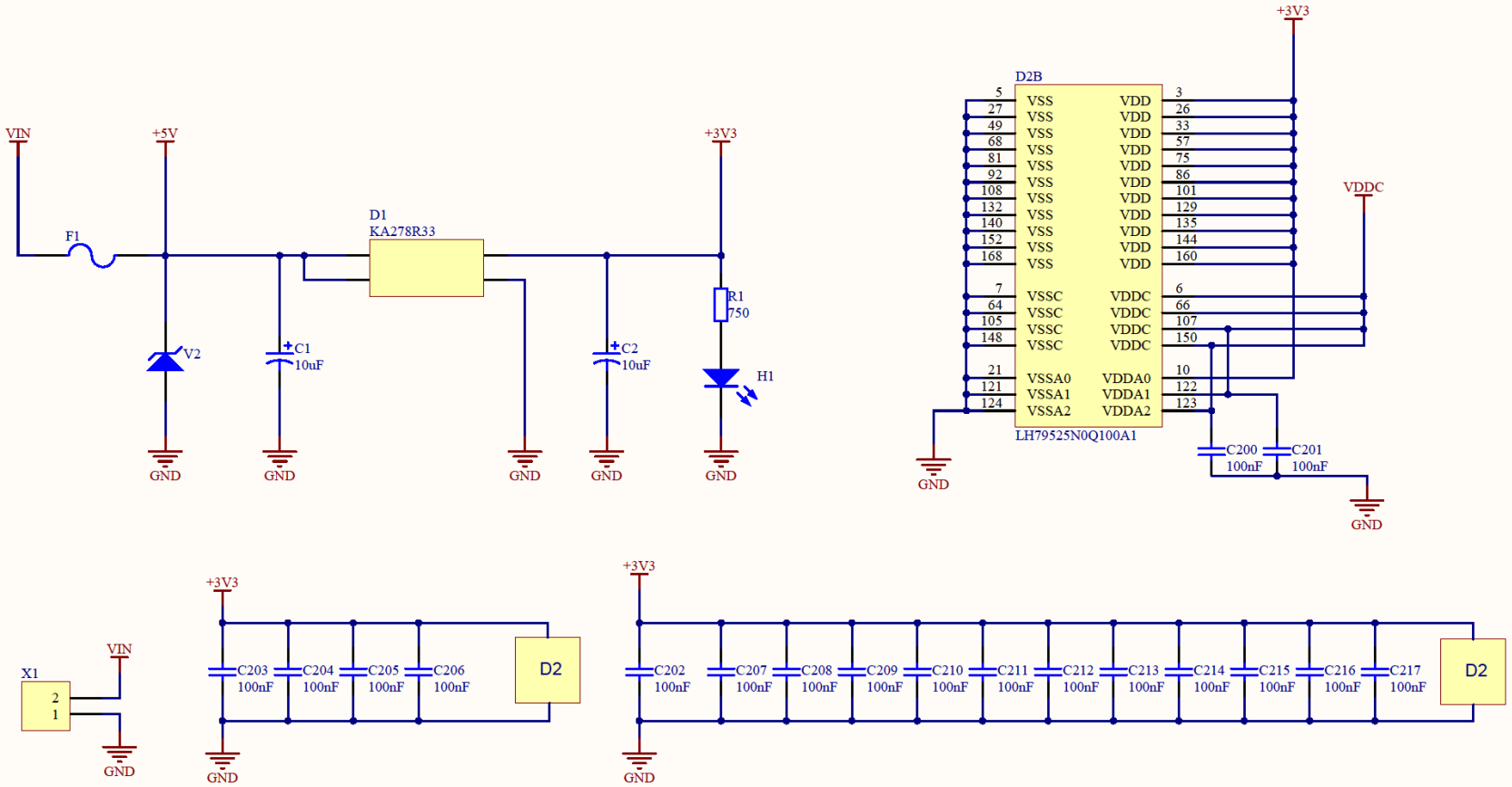
Parametri:

Unsigned int *buffer – pokazivač na video memoriju

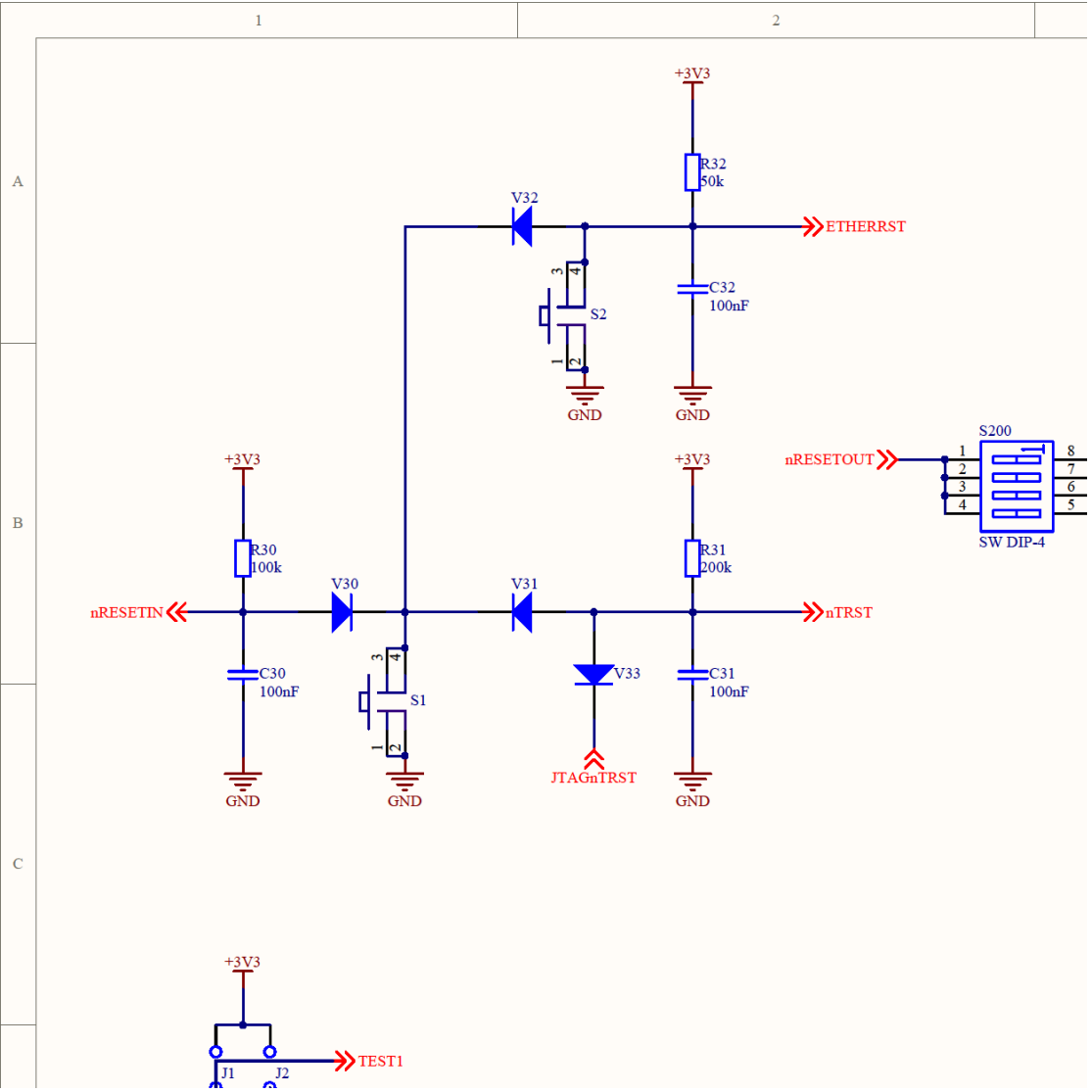
4.14.5. Funkcije za rad s flash memorijom

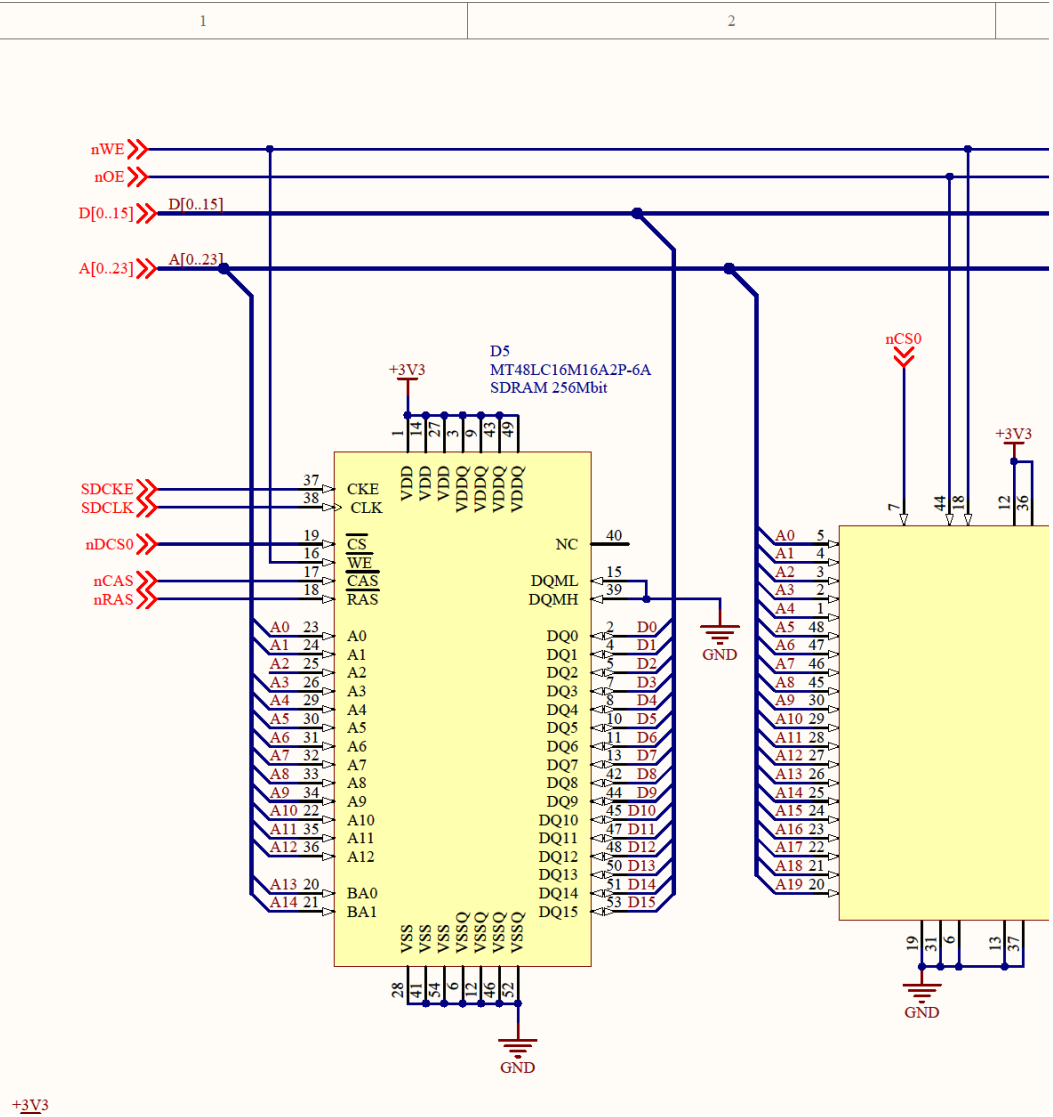


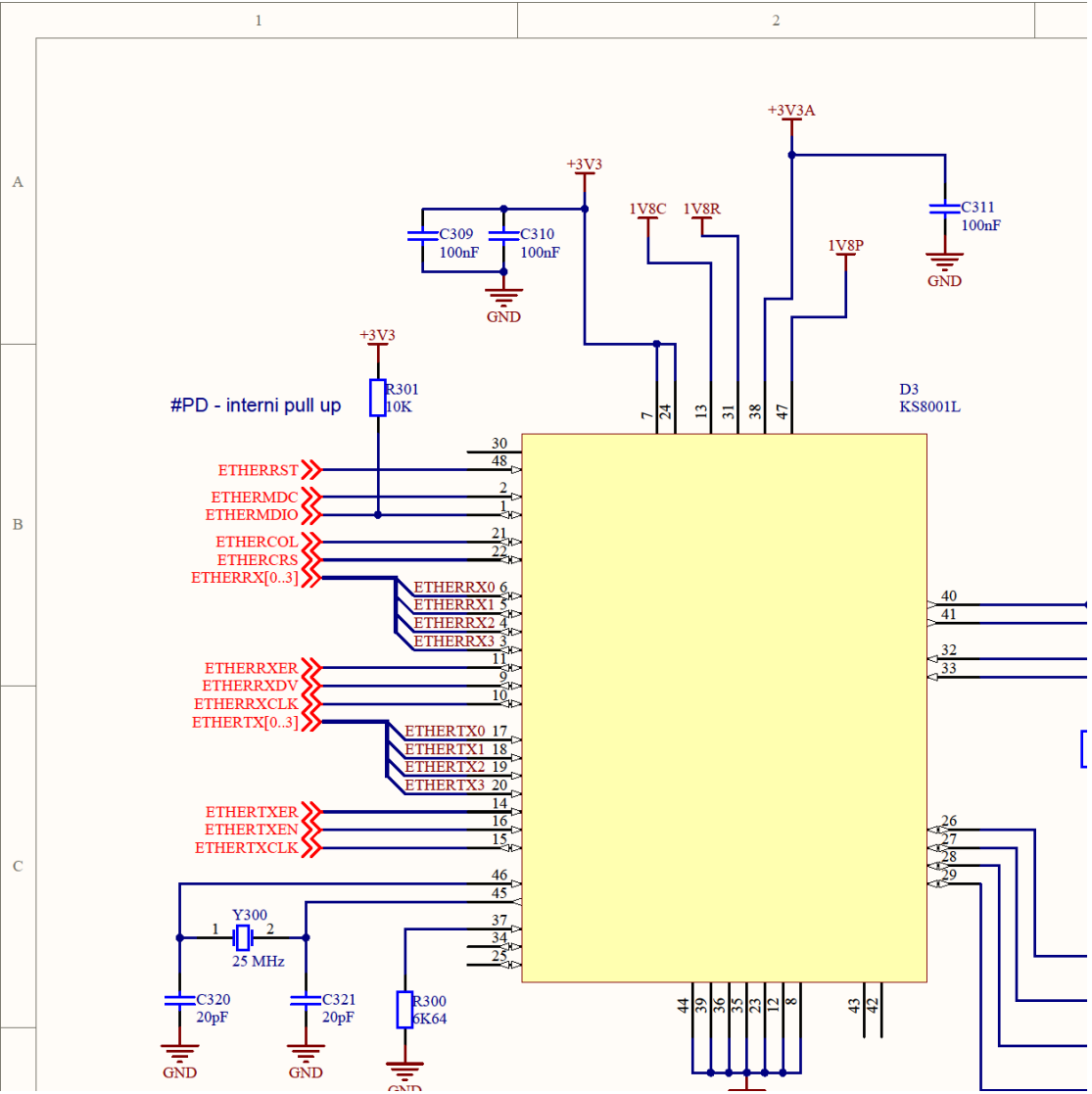
Title: LH79525 Development Board - Connection of microcontroller to the rest of the system		
Sheet size: A4	Sheet number: 1 of 7	Revision: 0
Date: June, 2010	Drawn by:Matija Glavinic Pecotic	

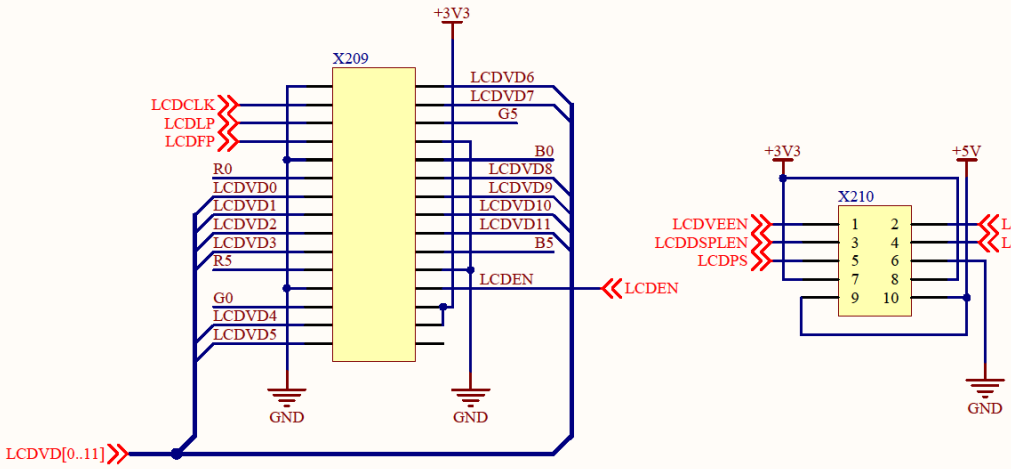


Title: LH79525 Development Board - Power supply and microcontroller's decoupling capacitors		
Sheet size: A4	Sheet number: 2 of 7	Revision: 0
Date: June, 2010	Drawn by:Matija Glavinic Pecotic	

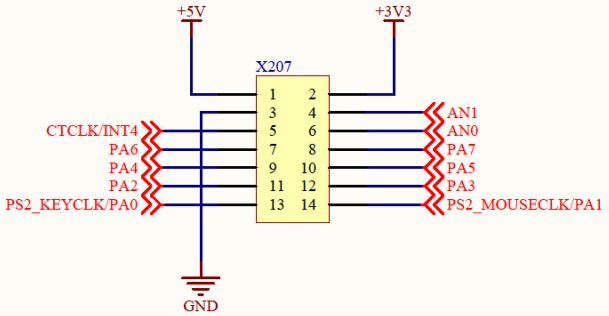

1
2
A
B
C
+3V3
R32
50k
V32
ETHERRST
C32
100nF
S2
GND
GND
S200
nRESETOUT
SW DIP-4
+3V3
+3V3
R30
100k
R31
200k
V30
V31
nRESETIN
nTRST
C30
100nF
S1
V33
C31
100nF
JTAGnTRST
GND
GND
GND
+3V3
TEST1
J1
J2

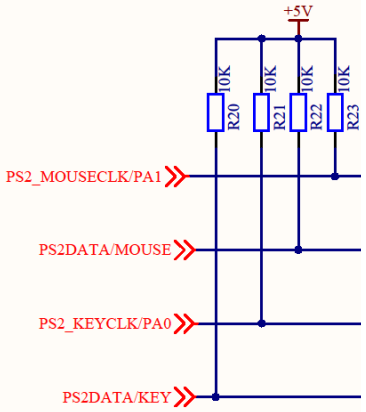

nWE
nOE
D[0..15]
A[0..23]
D5
MT48LC16M16A2P-6A
SDRAM 256Mbit
+3V3
nCS0
SDCKE
SDCLK
nDCS0
nCAS
nRAS
CKE
CLK
CS
WE
CAS
RAS
NC
DQML
DQMH
BA0
BA1
VDD
VDDQ
VSS
VSSQ
GND

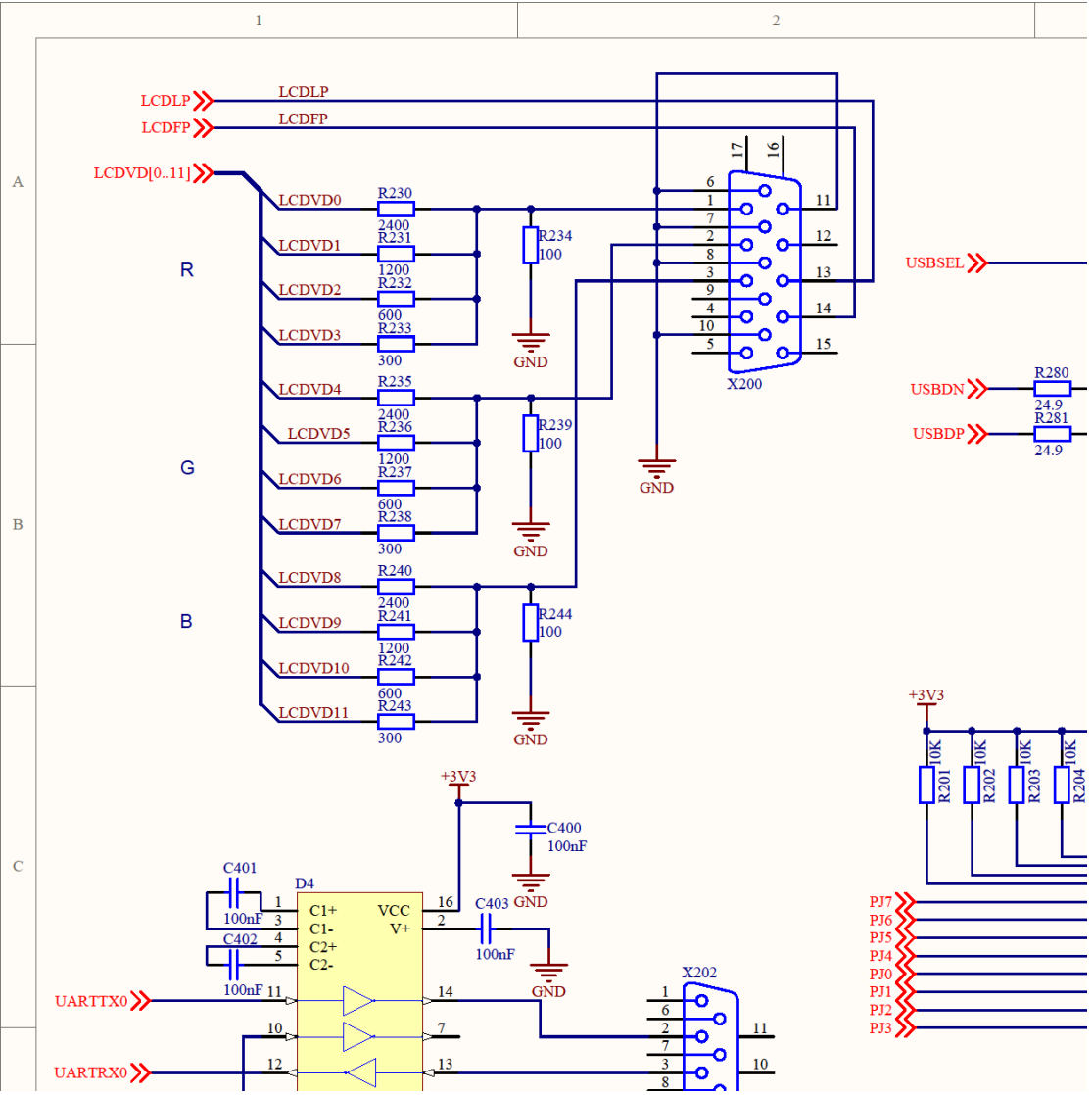

+3V3A
+3V3
1V8C
1V8R
1V8P
C309
100nF
C310
100nF
C311
100nF
GND
#PD - interni pull up
R301
10K
D3
KS8001L
ETHERRST
ETHERMDC
ETHERMDIO
ETHERCOL
ETHERCRS
ETHERRX[0..3]
ETHERRX0
ETHERRX1
ETHERRX2
ETHERRX3
ETHERRXER
ETHERRXDV
ETHERRXCLK
ETHERTX[0..3]
ETHERTX0
ETHERTX1
ETHERTX2
ETHERTX3
ETHERTXER
ETHERTXEN
ETHERTXCLK
Y300
25 MHz
C320
20pF
C321
20pF
R300
6K64


+3V3
X209
LCDCLK
LCDLP
LCDFP
R0
LCDVD0
LCDVD1
LCDVD2
LCDVD3
R5
G0
LCDVD4
LCDVD5
LCDVD6
LCDVD7
G5
B0
LCDVD8
LCDVD9
LCDVD10
LCDVD11
B5
LCDEN
LCDEN
GND
GND
LCDVD[0..11]

+3V3
+5V
X210
LCDVEEN
LCDDSPLEN
LCDPS
1 2
3 4
5 6
7 8
9 10
GND


+5V
+3V3
X207
1 2
3 4
5 6
7 8
9 10
11 12
13 14
CTCLK/INT4
PA6
PA4
PA2
PS2_KEYCLK/PA0
AN1
AN0
PA7
PA5
PA3
PS2_MOUSECLK/PA1
GND


+5V
10K
R20
10K
R21
10K
R22
10K
R23
PS2_MOUSECLK/PA1
PS2DATA/MOUSE
PS2_KEYCLK/PA0
PS2DATA/KEY


LCDLP
LCDFP
LCDVD[0..11]
R
G
B
LCDVD0
LCDVD1
LCDVD2
LCDVD3
LCDVD4
LCDVD5
LCDVD6
LCDVD7
LCDVD8
LCDVD9
LCDVD10
LCDVD11
R230
2400
R231
1200
R232
600
R233
300
R234
100
R235
2400
R236
1200
R237
600
R238
300
R239
100
R240
2400
R241
1200
R242
600
R243
300
R244
100
GND
X200
USBSEL
USBDN
USBDP
R280
24.9
R281
24.9
+3V3
R201
R202
R203
R204
10K
PJ7
PJ6
PJ5
PJ4
PJ0
PJ1
PJ2
PJ3
C400
100nF
C401
C402
C403
D4
C1+
C1-
C2+
C2-
VCC
V+
UARTTX0
UARTRX0
X202