SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1040

Ugradbeni računalni sustav

Davor Cihlar

Zagreb, prosinac 2009

Sadržaj

1 Uvod

Ugradbena računala su najraširenija u današnjem svijetu. Nalaze se u različitim oblicima od minijaturnih čipova za identifikaciju (RFID), preko mobitela, digitalnih fotoaparata, e-knjiga i igračih konzola, pa sve do pravih malih prijenosnih računala poput Internet *tableta*. U ovom radu prikazana je izgradnja jednog višefunkcionalnog ugradbenog računalnog sustava koji je u potpunosti osmišljen i ostvaren.

Mali "kompjuterčić" pogonjen i386 procesorom napravljen je iz dva jednostavna razloga:

- Prvi razlog je moja stara želja da napravim neki prenosivi uređaj s ekranom osjetljivim na dodir u stilu "Tablet-PC-a" koji bi bio pogonjen najmoćnijim IA32 kompatibilnim procesorom kojeg je moguće spojiti u kućnoj radinosti. Zbog toga mi je trebao neki manji projekt samo da proučim Intelovu sabirnicu.
- Drugi razlog je što sam htio napraviti platformu za operacijski sustav CleanOS kojeg sam već napravio prije. Taj sustav zahtijeva IA32 kompatibilni procesor zbog njegove vrlo moćne memorijske jedinice, te privilegijskih zaštita koje onemogućavaju da stabilnost jednog procesa naruši stabilnost drugog.

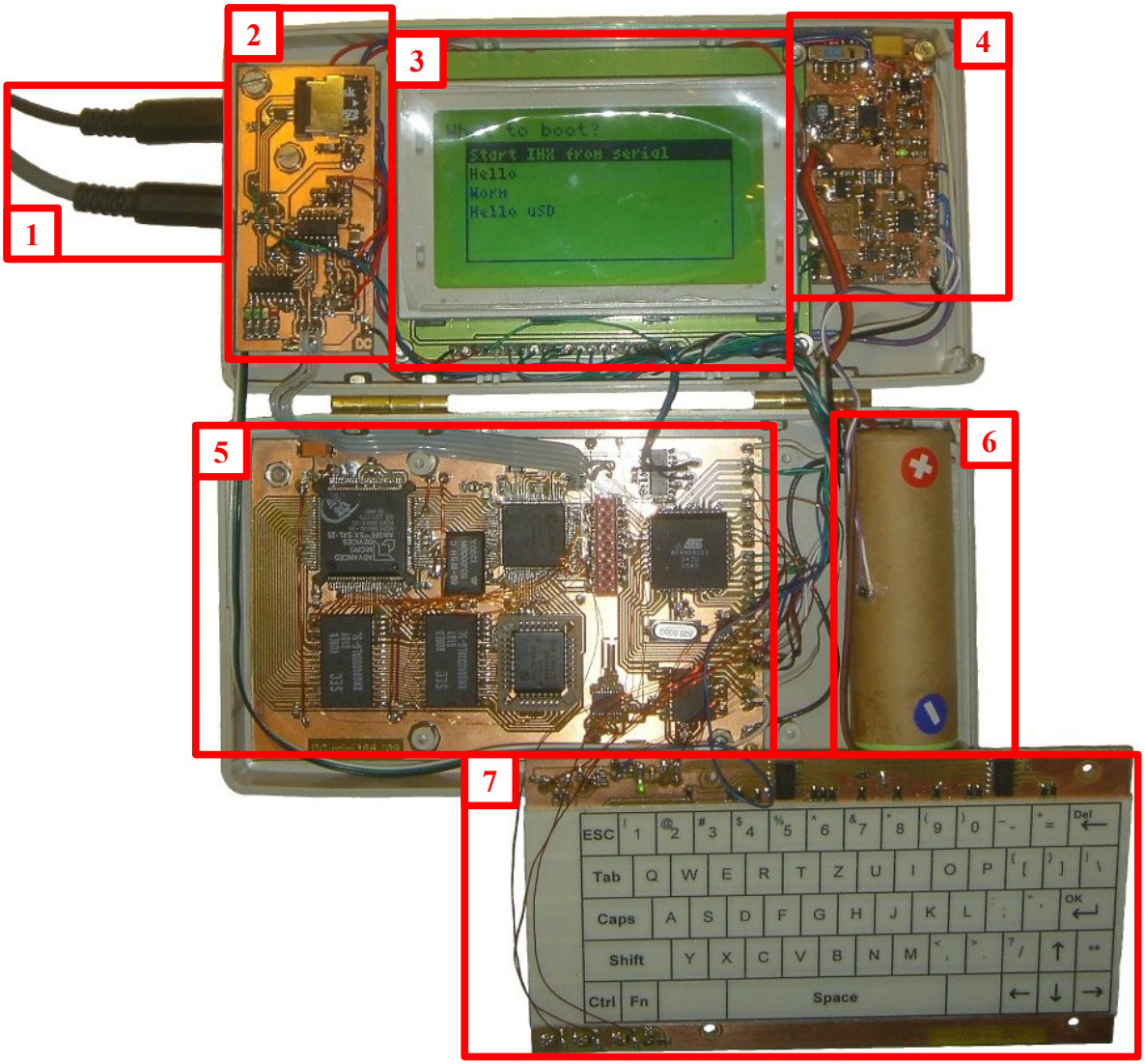
Kako nije potrebna veća količina memorije da se to ostvari, odabrao sam *flash* memoriju od 512kB kao programsku memoriju (ROM[1]), te 1MB za radnu memoriju (RAM[2]).

Grafički ekran, tipkovnica, svjetleće diode i baterija dodani su kako bi se omogućila interakcija s korisnikom i kako bi u potpunosti to bio prenosivi uređaj, a microSD memorijska kartica je ovdje kao neki izmjenjivi disk na kojeg možemo spremati podatke i dodatne programe.

Zbog programske prilagodljivosti, BIOS[3] ima ugrađenu podršku za odabir operacijskog sustava ili neke jednostavne programske aplikacije.

1 engl. read only memory
2 engl. random access memory
3 engl. basic input output system – skup osnovnih funkcija namijenjenih komunikaciji sa sklopovljem

2 Sklopovski elementi



Slika 1: Pregled svih komponenata

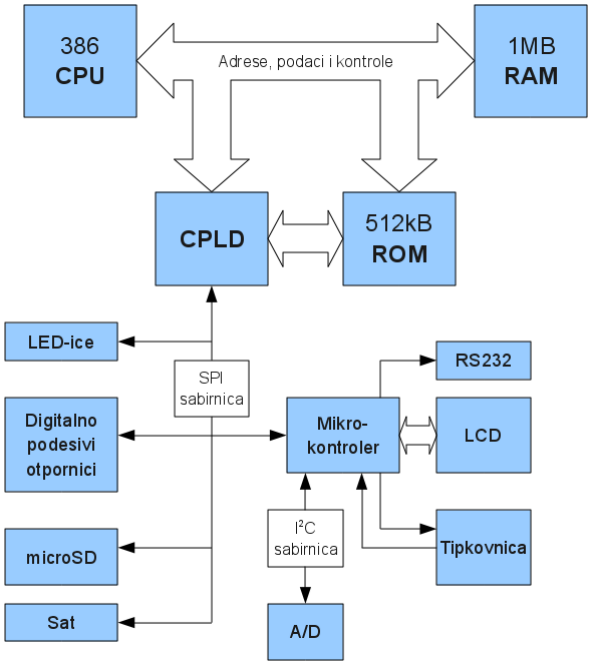
Sklopovski dio sustava mini386 sastoji se od nekoliko osnovnih dijelova, kao što je prikazano na *slici 1*:

1. priključci za vanjski izvor napajanja (gornji priključak), te serijsko sučelje (donji priključak),
2. pločica sa status LED-icama (svjetlećim diodama), regulacijom kontrasta i pozadinskog osvjetljenja, te microSD memorijskom karticom,
3. modul s crno-bijelim grafičkim LCD-om,
4. sustav za napajanje i punjenje baterije,
5. matična pločica, tj. "mozak" cijelog uređaja u kojem je 386 procesor, memorija i sve ostale bitne komponente,

6. NiMH baterije na koje je postavljen NTC otpornik za temperaturnu provjeru,
7. tipkovnica.

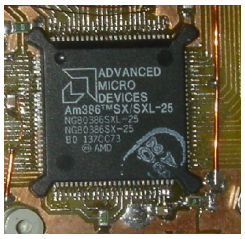
Najsloženiji dio cijelog mini386 sustava je matična pločica. Ona je ujedno i glavni dio cijelog sustava pošto se na njoj nalazi procesor.

Slika 2 prikazuje blokovski prikaz strukture sustava na kojem je moguće uočiti da postoji hijerarhijska struktura. Na vrhu je procesor sa svojom memorijom i CPLD-om. CPLD zatim ima ugrađenu kontrolu SPI sabirnice na kojoj su manje prioritetne vanjske jedinice, a mikrokontroler upravlja s vanjskim jedinicama najmanjeg prioriteta.



Slika 2: Struktura sustava

2.1 Procesor



Slika 3: Procesor

Procesor je AMD-ov Am386SX koji radi do 25MHz skinut iz jednog starog prijenosnog računala. Zbog toga što radi isključivo na 5V, cijeli sustav je morao biti osmišljen tako da radi na tom naponu. i386 je 32-bitan procesor, ali ova izvedba ima 16-bitnu podatkovnu sabirnicu, te 24-bitnu adresnu sabirnicu. Stoga su za 32-bitne podatke potrebna dva pristupa memoriji te se može obuhvatiti do 24MB. No za ove potrebe je to i više nego dovoljno. Bitno je napomenuti da je zbog 16-bitne podatkovne sabirnice poželjno koristiti 16-bitne cijele brojeve u programskoj podršci gdje god je to moguće. Shema koja prikazuje procesor nalazi se na strani 55.

Zbog svoje specifične sabirnice[1], potreban je dodatni sklop koji određuje što procesor zapravo želi i na temelju toga upravlja ostalim sklopovljem. Stoga je uz i386 kompatibilan procesor praktički nužan neki "chipset", a njegovu ulogu ovdje igra CPLD[4] čiju je logičku funkciju moguće jednostavno mijenjati pomoću programskog jezika VHDL[5].

i386 procesor razlikuje dvije vrste vanjskih jedinica: ulazno-izlazne vanjske jedinice i memorijske vanjske jedinice. Prilikom pristupa nekoj vanjskoj jedinici, signal $M/\overline{IO}$ označava kojem tipu vanjske jedinice se pristupa. Ulazno-izlazne vanjske jedinice moguće je adresirati s maksimalno 16 bita, dok je za memorijske vanjske jedinice moguće koristiti čitav adresni prostor.

Procesor podržava i protočan pristup sabirnici, tj. može u jednom ciklusu zatražiti adresu kojoj želi pristupiti i u istom ciklusu prihvatiti/zapisati podatak na adresi koju je prethodno zatražio. Taj pristup se u ovom rješenju zbog jednostavnosti ne koristi, te je signal $\overline{NA}$ koji omogućuje protočan pristup fiksno spojen na logičku jedinicu.

Direktan pristup 8-bitnim podacima i386 sabirnica omogućuje pomoću signala $\overline{BHE}$ i $\overline{BLE}$ koji označavaju kojih 8 bita sabirnice treba uzeti u obzir. Zahvaljujući tome može se na sabirnicu spajati i 8-bitne vanjske jedinice kojima je kao signal A_0 spojen signal $\overline{BHE}$. Jedini problem u tom slučaju je što tada nije moguće čitati 16-bitne podatke u jednom komadu, a to je veoma bitno kako bi procesor čitao izvršni kod iz ROM memorije. Rješenje tog problema je razrađeno u poglavlju *2.2.3 Romdemux*.

Ostali bitni signali prikazani su u *tablici 1*. Stanja signala koja jednoznačno određuju što procesor želi raditi su prikazana u *tablici 2*.

Slika 4 iz Intelove dokumentacije prikazuje primjer čitanja i pisanja pomoću kojeg je jednostavno shvatiti cijeli princip. Taj primjer vrijedi za svaki tip ciklusa iz *tablice 2*. Ciklus 2 i ciklus 3 prikazuju pristup sporoj vanjskoj jedinici. Kad procesor na kraju ciklusa ustvrdi da je vanjska jedinica zauzeta jer je signal $\overline{READY}$=0, ciklus se produžuje za jedan dodatni takt nakon kojeg ponovo provjerava stanje signala $\overline{READY}$ sve dok ne prijeđe u neaktivno stanje.

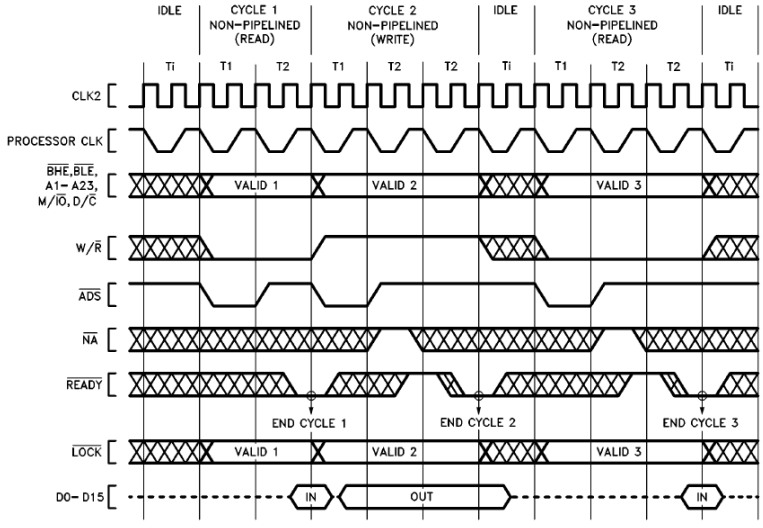
4 engl. complex programmable logic device – programirljiva logika
5 programski jezik za opis sklopovlja

Tablica 1: Osnovni signali i386 procesora

Naziv signala	Opis
CLK2	Ulaz za dvostruki takt (50MHz)
A_1 - A_{23}	Adresna sabirnica
D_0 - D_{16}	Podatkovna sabirnica
$\overline{BHE}$, $\overline{BLE}$	Označava da li pristupa visokih ($\overline{BHE}$) i/ili niskih ($\overline{BLE}$) 8-bita podataka
W/$\overline{R}$	1 ako piše, inače čita
M/$\overline{IO}$	1 ako se pristupa memoriji, inače se pristupa vanjskoj jedinici
D/$\overline{C}$	1 ako se pristupa podacima, inače se radi o kontrolnom ciklusu
$\overline{ADS}$	Označava početak ciklusa
$\overline{READY}$	Pomoću tog signala procesor zna da li je vanjska jedinica spremna

Tablica 2: Određivanje tipa ciklusa procesora

M/$\overline{IO}$	D/$\overline{C}$	W/$\overline{R}$	Tip ciklusa
0	0	0	Potvrda prekida
0	1	0	Čitanje s vanjske ulazno/izlazne jedinice
0	1	1	Pisanje u vanjsku ulazno/izlaznu jedinicu
1	0	0	Čitanje iz programske memorije
1	1	0	Čitanje iz podatkovne memorije
1	1	1	Pisanje u podatkovnu memoriju



Slika 4: Primjer čitanja i pisanja vanjske jedinice

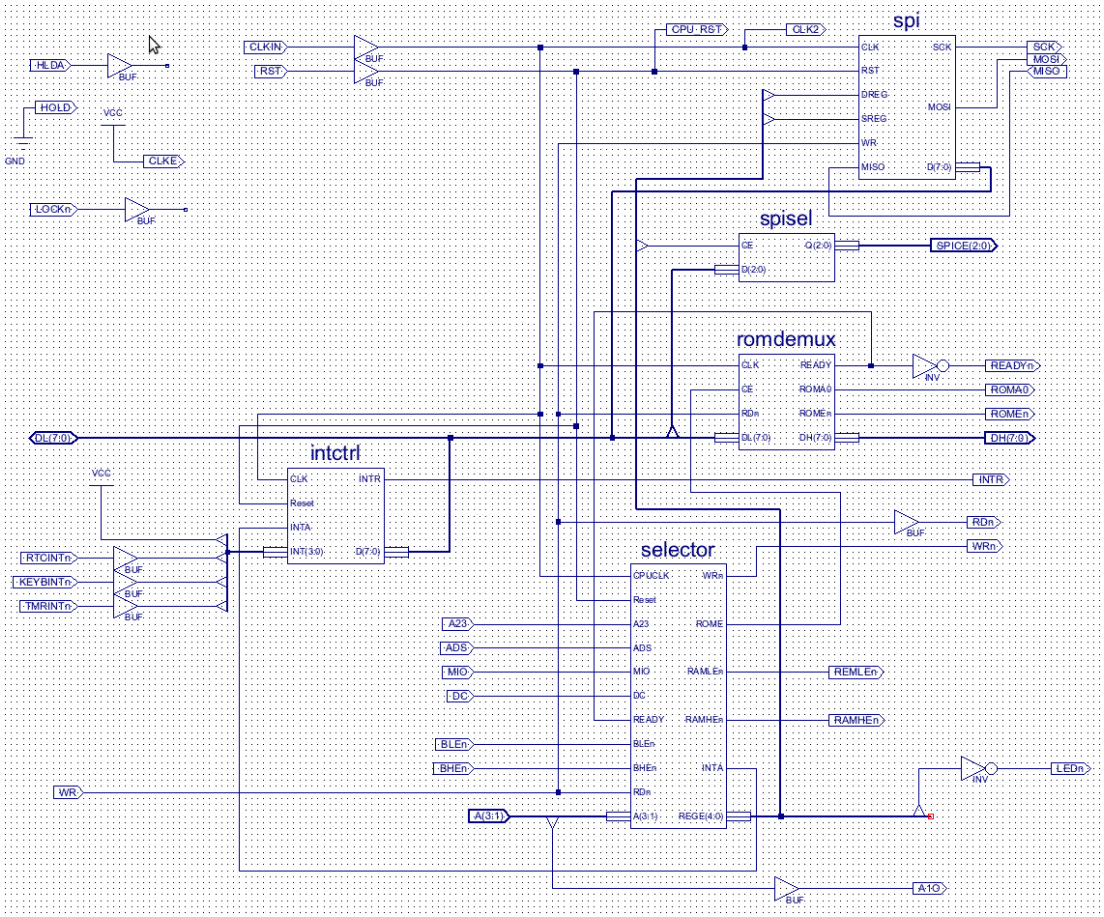
2.2 CPLD

Slika 5: CPLD

CPLD je integrirani krug kojem je moguće programirati njegovu logičku funkciju. Složeniji je od PAL[6] sklopova, ali jednostavniji od FPGA[7] sklopova. U grubo se može reći da se CPLD sastoji od stotinjak PAL sklopova, a FPGA od stotinjak CPLD sklopova. Shema na kojoj se nalazi CPLD nalazi se na strani 54.

mini386 koristi CPLD kao logiku koja povezuje cijeli sustav (popularno nazvan u engleskom jeziku: *glue logic*). Bez programabilne logike, sustav bi jedino bilo moguće povezati pomoću puno diskretnih logičkih integriranih krugova što bi bilo veoma složeno za izvesti, posebno zbog gotovo nikakve mogućnosti za ispravljanje pogrešaka, a kamo li izmjene ideje. Osim zbog tehničkih problema, izvedba pomoću diskretnih integriranih krugova bi bila i znatno veća i skuplja.

Najbolji CPLD koji sam imao pri ruci je Xilinx-ov XC95144 sa 144 makroćelija i 81 ulazno-izlaznim izvodom. Od toga je iskorišteno 93 (65%) makroćelija, te 51 izvod. To znači da je XC95144 bio i više nego dovoljan, te čak još ima slobodnih makroćelija za nadogradnju.

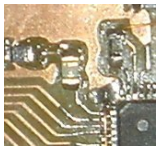


Slika 6: Shema sklopa u CPLD-u

6 engl. programmable array logic
7 engl. field programmable gate array logic

Slika 6 prikazuje shemu sa svim modulima koji su napravljeni u programskom jeziku VHDL za potrebe mini386. Srce svega je "selector" koji prati stanje na sabirnici procesora, te generira upravljačke signale za sve vanjske jedinice. Ništa manje bitan je "romdemux" koji za svaki pristup procesora ROM-u dohvaća dva bytea kako bi procesor mogao dohvatiti cijeli 16-bitni podatak u komadu. "intctrl" je prioritetni upravljač prekidima, a "spi" je upravljač SPI[8] sabirnicom. "Spisel" je 3-bitni registar za odabir vanjske jedinice s kojom se komunicira preko SPI sabirnice. U gornjem lijevom kutu sheme su prikazani neiskorišteni signali.

2.2.1 Svjetleća dioda za uhodavanje

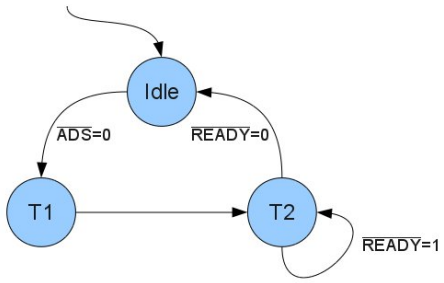


Slika 7: LED-ica za uhodavanje

"LEDn" izlaz na pločici spojen je na LED-icu koju je moguće koristiti kao signalizaciju prilikom razvoja kada ništa nije sigurno da radi. Zasvijetli samo na kratko kada se pristupa točno određenom registru. Da bi se vidjelo da svijetli, potrebno je neprestano pristupati njenom registru. Traženje grešaka pomoću takvog načina je izuzetno nepraktično, pa je to isključivo bila prva pomoć pri samom uhodavanju sklopa. Nakon što se ustvrdi da SPI sabirnica radi, moguće je koristiti 4 LED-ice koje su inače zamišljene kao status LED-ice, a nakon što prekidi i *firmware*[9] prorade, moguće je ispisivati i na LCD ili slati podatke na serijsko sučelje.

2.2.2 Selector

Modul "selector" pomoću Moorovog automata prati stanje na sabirnici procesora, te na temelju trenutnog stanja određuje što koja vanjska jedinica treba činiti.



Slika 8: Mooreov automat za određivanje stanja sabirnice

Slika 8 prikazuje pojednostavljenu verziju implementiranog sinkronog Mooreovog automata za određivanje stanja na sabirnici. U konačnici se stanja "T1" i "T2" sastoje od dva podstanja zbog toga što je CPLD-ov izvor takta dvostruko veći od procesorovog.

8 engl. serial peripheral interface – sinkrona serijska sabirnica
9 Programska podrška koja upravlja mikrokontrolerom

Početno stanje je "Idle" te čim se detektira da je $\overline{ADS}$=0, prelazi se u stanje "T1" što je prva polovica procesorovog ciklusa. Bez uvjeta iz stanja "T1" prelazi se u stanje "T2" u kojem automat ostaje sve dok vanjska jedinica nije spremna. Nakon što vanjska jedinica javi da je spremna, automat se vraća u početno stanje "Idle" nakon čega cijeli proces počinje iz početka.

Osim praćenja stanja sabirnice procesora, "selector" ima još jednu bitnu zadaću po kojoj je i dobio ime. Radi se o odabiru vanjske jedinice s kojom procesor namjerava komunicirati. Na temelju *tablice 2* određuje se da li se pristupa memoriji, ulazno-izlaznoj jedinici ili se čita broj prekida. Pristup vanjskoj jedinici vrši se samo kada je sabirnica u aktivnom stanju, tj. nalazi se u stanju "T1" ili "T2".

Sve ulazno-izlazne vanjske jedinice su 8-bitne i one se nalaze unutar CPLD-a. Zbog jednostavnosti izvedbe, signali $\overline{BHE}$ i $\overline{BLE}$ se ignoriraju prilikom adresiranja ulazno-izlaznih vanjskih jedinica, te su iz tog razloga korištene samo parne adrese. Osim ta dva signala, ignoriraju se i svi adresni signali osim A_3-A_1. *Tablica 3* prikazuje adresu svakog registra.

Tablica 3: Adrese ulazno-izlaznih registara

Adresa	Opis registra
`000X` (0_{16})	Registar za omogućenje RAM-a
`001X` (2_{16})	Registar "spisel" (odabir vanjske jedinice na SPI sabirnici)
`010X` (4_{16})	Podatkovni registar modula "spi"
`011X` (6_{16})	Registar stanja modula "spi"
`100X` (8_{16})	Rezerviran za budući razvoj
`101X` (A_{16})	*Debug* LED-ica

Područje za podatkovnu i programsku memoriju jednostavno je podijeljeno na dva dijela: RAM i ROM. Kao što je prikazano u *tablici 4*, A_{23} određuje da li se pristupa ROM-u (1) ili RAM-u (0).

Tablica 4: Memorijska područja

Područje$_2$	Područje$_{16}$	Memorija
`0XXX 0000 0000 0000 0000 0000` – `0XXX 1111 1111 1111 1111 1111`	`000000` – `0FFFFF`	RAM
`1XXX X000 0000 0000 0000 0000` – `1XXX X111 1111 1111 1111 1111`	`800000` – `87FFFF`	ROM

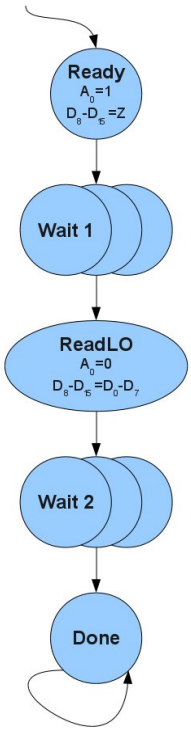
Registar 0 je specijalni registar, odnosno zastavica za omogućenje RAM-a. Ta zastavica prilikom pokretanja sustava se postavlja, što onemogućuje pristup RAM-u, ali omogućuje preslikavanje ROM-a na niske adrese. Procesor zbog kompatibilnosti počinje izvođenje koda od lokacije $FFFFF0_{16}$ u 16-bitnom načinu rada ("real mode"). To nam daje jako malo prostora za programsku memoriju (16 bytea), te nije moguće skočiti unutar ROM područja

zbog toga što "real mode" maksimalno omogućuje samo 20-bitno adresiranje (maksimalna adresa je $0FFFFF_{16}$). Stoga je pri pokretanju sustava na mjesto RAM-a doveden ROM samo kako bi se procesor prebacio u "protected mode". Nakon inicijalizacije je potreban pristup registru 0 kako bi se omogućio RAM i nastavilo izvršavanje koda iz ROM-a.

2.2.3 Romdemux

Modul "romdemux" omogućuje procesoru da pristupa 8-bitnom ROM-u kao da pristupa 16-bitnom. Kako bi to bilo moguće, potrebno je pročitati dva bytea iz ROM-a unutar jednog ciklusa sabirnice procesora. ROM-ova podatkovna sabirnica je spojena na nižih 8-bita podatkovne sabirnice procesora, a ROM-ove adrese su sve osim A_0 spojene na adresnu sabirnicu procesora. Stoga "romdemux" prilikom čitanja iz ROM-a treba samo prvo pročitati byte iz ROM-a s postavljenim A_0=1 (ostale adrese određuje procesor), postaviti pročitani na visokih 8 bita adresne sabirnice, te postaviti A_0=0 što će novi byte iz ROM-a postaviti na niskih 8 bita podatkovne sabirnice procesora. Na složenost ovog procesa čitanja dodatno utječe sporost ROM-a zbog čega je potrebno dodati posebna stanja za čekanje.

Slika 9 prikazuje implementiran Mooreov automat za demultipleksiranje, tj. 16-bitan pristup 8-bitnom ROM-u. Automat se cijelo vrijeme nalazi u stanju "Ready" sve dok se ne zatraži dohvaćanje podatka iz ROM-a pomoću signala ROME=1. Stanje "Ready" je stanje čekanja, tj. pripravnosti i ono postavi A_0=1, a D_8-D_{15} u stanje visoke impedancije kako bi se oslobodilo sabirnicu. Nakon što dođe do zahtjeva za podatkom iz ROM-a, čeka se neko vrijeme kako bi nam ROM dohvatio podatak pomoću stanja koja ne rade ništa. Zatim se u stanju "ReadLO" podatak spremi i postavi na visokih 8 bita podatkovne sabirnice, te A_0 prebaci u 0. Nakon toga opet čekamo neko vrijeme da ROM dohvati novi podatak, te prijeđemo u stanje "Done". U trenutku prelaska u stanje "Done", na niskih 8 bita podatkovne sabirnice nalazi se novi podatak iz ROM-a, te sad procesor može dohvatiti svih 16 bita: zapamćenih visokih 8 bita sa adrese A_0=1 i niskih 8 bita koje nam daje ROM dok je A_0=0.



Slika 9: Mooreov automat za romdemux

Sva stanja osim "Ready" i "Done" označavaju da je vanjska jedinica (tj. ROM) zaposlena, te će zbog toga prilikom čitanja ROM memorije procesor čekati sve dok automat ne prijeđe u stanje "Done". Nakon što procesor pročita podatak, ROME prelazi u 0, a time i automat u stanje "Ready".

Automat je također pogonjen osnovnim taktom od 50MHz, pa se na temelju toga može znati koliko stanja treba ubaciti za čekanje kako bi ROM dohvatio podatak. Kao ROM je korišten integrirani krug AM29F040-70 i prema njegovim uputama sufiks "-70" kaže da je za dohvaćanje podatka potrebno 70ns, a jedan period 50MHz takta je 20ns. Uvrštavanjem tih podataka u formulu [1] dobije se da je potrebno čekati 4 stanja kako bi ROM dohvatio podatak.

$$N = \left\lceil \frac{T_{kašnjenja}}{T_{takta}} \right\rceil \qquad [1]$$

Kako se funkcija pojedinog stanja izvodi tek na prelasku u novo stanje, zapravo su dovoljna samo N-1=3 prazna stanja i jedno funkcionalno stanje ("ReadLO").

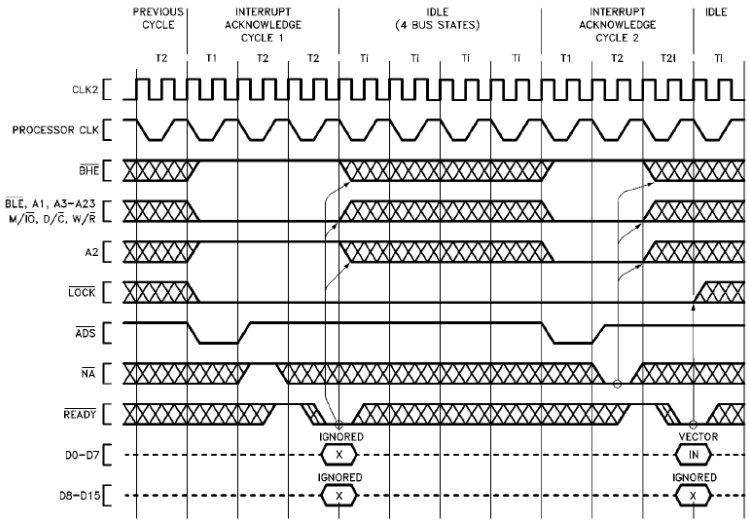
Pri bilo kojem tipu čitanja podatka iz ROM-a (8 ili 16 bitni pristup) uvijek se radi isti proces, stoga je čitanje samo jednog bytea iz ROM-a je jednako sporo kao i 16-bitno. Jednostavnosti radi, tim problemom se nema smisla baviti jer se u konačnici nakon učitavanja odabranog programa, program ionako izvodi iz RAM-a koji je dovoljno brz da mu se može pristupiti unutar samo jednog ciklusa procesora.

Jedna još ne implementirana podrška (u ovom slučaju vrlo korisna) je mogućnost izmjene podataka u ROM-u. Naime, ovdje korišten integrirani krug s funkcijom ROM-a nije stvarno memorija koju je moguće samo čitati, već je *flash* tipa, tj. sam sustav ju može mijenjati bez ikakvih potrebnih dodataka tipa ultraljubičasto svjetlo ili visok napon. Bez te podrške je za svaku promjenu podataka u ROM-u potrebno izvaditi memorijski integrirani krug, a svako vađenje znači moguće oštećenje pločice.

2.2.4 Upravljač prekidima - Intctrl

"Intctrl" je skraćenica od engl. *interrupt controller*, tj. upravljač prekidima. Njegova zadaća je prihvaćanje prekida, određivanje najvišeg prioriteta, te obavještavanje procesora do kojeg je prekida došlo.

Upravljač prekidima posjeduje 4 ulaza za prekid od kojih je jedan rezerviran za nadogradnju. Na padajući brid signala na ulazu, postavi se pripadajuća zastavica kao oznaka na kojem ulazu je došlo do prekida. Ako je bilo koja od tih zastavica postavljena, procesoru se javlja da je došlo do prekida, nakon čega dolazi do potvrde prekida prikazane na *slici 10*. Potvrda prekida se sastoji od 2 ciklusa procesora koja možemo prepoznati iz *tablice 2*, a razlikovati pomoću signala A_2. Prvi ciklus možemo ignorirati, ali na kraju drugog ciklusa, na niskih 8 bitova podatkovne sabirnice potrebno je staviti broj koji označava tip prekida. Za potrebe određivanja rednog broja prekida predviđeno je 3 bita, što je i više nego dovoljno za trenutna 4 ulaza za prekid. Preostalih 5 bita se konstantno postavlja na "00101" kako bi prekidi bili u području od 28_{16} na dalje. Bitno je da prekidi dolaze s brojem većim ili jednakim 20_{16} jer je prvih 20_{16} Intel rezervirao za sistemske, tj. prekide koji se javljaju unutar samog procesora (npr. dijeljenje s nulom, pokušaj pozivanja nedopuštene instrukcije, ...). 28_{16} je odabran kao broj prvog prekida kako bi operacijski sustav CleanOS imao slobodan prekid 20_{16} za sistemske pozive budući da je inicijalno bio napisan za PC, te bi bilo moguće jednostavno kopirati program s PC-a na mini386.



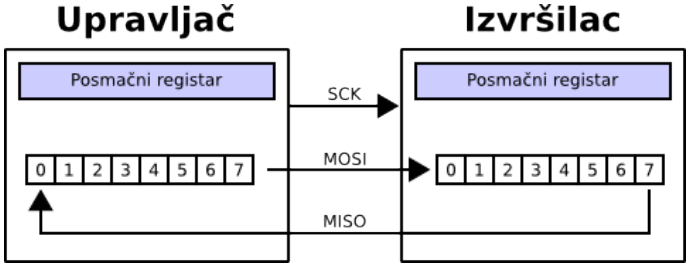
Slika 10: Primjer potvrde prekida

Slika 10 namjerno ima malo više detalja kako bi pokrila više slučajeva. No kako je već rečeno, $\overline{NA}$ je konstantno spojen na logičku jedinicu, te to nije od interesa. Osim toga, u oba ciklusa prekida ubačen je slučaj kada je $\overline{READY}$=1, tj. označeno je da procesor treba čekati da jedinica za obradu prekida bude spremna, ali to nije ni bitno jer je implementirana jedinica za obradu prekida dovoljno brza.

Nakon što procesor potvrdi prekid i zatraži njegov redni broj, upravljač prekidima treba poništiti zastavicu tog prekida. Bilo bi idealno kada bi odabrani CPLD podržavao bistabile koji se postave na rastući brid, a poništе na padajući, ali kako to nije moguće, bilo je potrebno improvizirati. Problem je riješen tako da se prilikom potvrde prekida iz rednog broja prekida odredi o kojoj se zastavici radi, te joj se postavi reset signal. Rješenje možda zvuči jednostavno, ali zahtijeva dodatni registar koji pamti redni broj prekida o kojem se obavještava procesor. Kada bi direktno izračunavali prekid iz zastavica i taj rezultat javljali procesoru i koristili za brisanje zastavice, sve zastavice bi se obrisale prije nego bi procesor pročitao o kojem se prekidu radi.

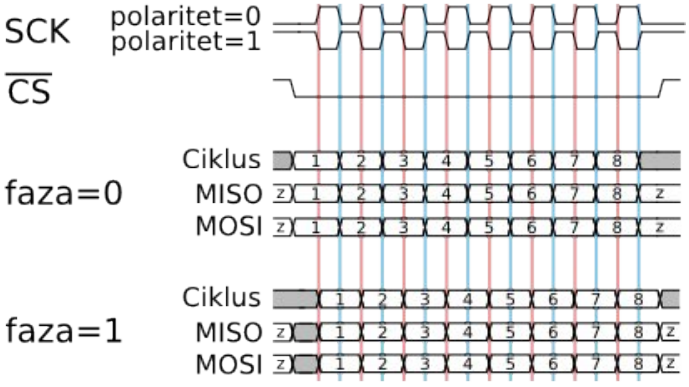
2.2.5 SPI i spisel

Moduli SPI i spisel služe za komunikaciju s vanjskim jedinicama spojenim na SPI[2] sabirnicu. SPI sabirnica je sinkrona serijska sabirnica za dvosmjernu komunikaciju kroz 3 vodiča. *Slika 11* prikazuje osnovni princip rada te sabirnice. Prijemnik i predajnik temeljeni su na 8-bitnom posmačnom registru sa zajedničkim signalom za takt (SCK). Nakon 8 perioda, svi bitovi iz upravljača će biti preneseni putem vodiča MOSI (engl. *master output; slave input*) do izvršilaca, a svi bitovi iz izvršilaca će biti preneseni do upravljača putem vodiča MISO (engl. *master input; slave output*).



Slika 11: Osnovni princip rada SPI sabirnice

Postoji više izvedbi SPI sabirnice, ali odlučio sam da zbog jednostavnosti ne bi trebalo omogućiti odabir tipa. U protivnom bi Mooreov automat za SPI sabirnicu bio značajno složeniji i bila bi potrebna 2 dodatna registra (time i dodatne dvije makroćelije), a na samom početku razvoja sklopovlja u CPLD-u nisam još bio siguran da li će biti dovoljno 144 makroćelija. *Slika 12* prikazuje sve 4 izvedbe SPI sabirnice.



Slika 12: Tipovi SPI sabirnice

"Polaritet" označava u kojem stanju je SCK dok se ne šalju podaci, a "faza" označava na koji brid podaci izlaze iz posmačnog registra, odnosno na koji brid se uzorkuju. Kako bi bili zadovoljeni zahtjevi microSD kartice, faza i polaritet moraju biti 1. Sve ostale vanjske jedinice odabrane su tako da budu kompatibilne s odabranim tipom SPI sabirnice.

Korištenje sabirnice sa samo nekoliko vodiča značajno olakšava povezivanje komponenata na tiskanoj pločici, ali i sami integrirani krugovi su manji jer trebaju manje izvoda. Jedina mana ovakvog rješenja je brzina. Takt sabirnice zbog izvedbe Mooreovog automata može maksimalno biti 25MHz, ali je ovdje dodatno dvostruko usporen kako bi se mogao vidjeti oblik signala na osciloskopu koji sam imao na raspolaganju, ali i kako bi se smanjili mogući problemi s izobličenjem signala. Zbog nekih sporijih vanjskih jedinica na SPI sabirnici, dodana je opcija za dodatno dvostruko usporavanje sabirnice, pa bi se ta opcija u budućnosti mogla unaprijediti kako bi se omogućila i puna brzina sabirnice od 25MHz.

Gore spomenuto izobličenje signala na SPI sabirnici nije bio mali problem pri povezivanju matične pločice s onom za korisničko sučelje. Kako bi se moglo utjecati na nju, potrebno je znati više o samoj refleksiji i utjecaju jednog signala na drugi. Bilo je potrebno staviti dobro uzemljenje, veliki kondenzator paralelan s napajanjem, te dobro napraviti transmisijsku liniju. Ovo zadnje sam "pokupio" s PC-jeve IDE sabirnice. Dizajneri IDE

sabirnice su na strani predajnika ubacili otpor od 22Ω, a sa strane prijemnika 82Ω. Osim toga, vodiči takta su ograđeni vodičima za uzemljenje. Takva organizacija sprečava utjecaj signala takta na ostale, a samo taj problem je bio iznenađujuće velik uzrok izobličenja.

Na SPI sabirnicu spojeno je više vanjskih jedinica, te je zbog toga bilo potrebno uvesti prozivanje. Sve vanjske jedinice koje su predviđene za spajanje na SPI sabirnicu imaju još jedan ulaz za prozivanje: $\overline{CE}$[10]. U CPLD-u je implementiran samo 3-bitni registar za odabir vanjske jedinice, ali samo prozivanje vrši demultipleksor 3-8: 74HCT138. *Tablica 5* prikazuje adresu svake vanjske jedinice spojene na SPI sabirnicu.

Tablica 5: Adrese i brzine vanjskih jedinica

Adresa	**Brza**	**Vanjska jedinica**
1	da	Shift registar s LED-icama
2	da	microSD
3	ne	Digitalno podesivi otpornici za kontrast i pozadinsko osvjetljenje
5	ne	Alarm (budilica)
6	da	Mikrokontroler
7	-	Koristi se kad je potrebno da niti jedna vanjska jedinica nije prozvana
0, 4	-	Slobodno za nadogradnju

Kad se vanjski signal RESET postavi u aktivno stanje, svi signali SPI sabirnice postavljaju se u stanje visoke impedancije kako bi se omogućilo programiranje mikrokontrolera.

Podaci se šalju na SPI sabirnicu tako da se prvo odabere vanjska jedinica zapisom broja u spisel registar, te odabere brzina prijenosa postavljanjem drugog bita status registra u potrebno stanje. Nakon toga je dovoljno samo zapisati u podatkovni registar podatak kojeg je potrebno poslati na SPI sabirnicu. Zapisom u taj registar započinje slanje.

Slanje traje vrlo kratko, pa je bolje radno čekati nego koristiti prekide. Nakon što prvi bit status registra postane 0, podatak je poslan, te je moguće ponovno pisati u podatkovni registar kako bi se započelo slanje novog podatka.

Tablica 6: Status registar modula SPI

SPI SREG	0	1	0	0	0	0	fastclk	busy
bit	7	6	5	4	3	2	1	0

U *tablici 6* prikazan je format status registra modula SPI.

- Najviši bitovi su konstantni i u početku pri uhodavanju sustava namijenjeni provjeri da li radi pristup ulazno-izlaznim vanjskim jedinicama
- fastclk – ako je ta zastavica postavljena podaci će se slati brzinom od 12.5Mbit/s, inače 6.25Mbit/s
- busy – označava da li se trenutno šalje podatak na SPI sabirnicu

10 engl. chip enable – ulaz za aktiviranje vanjske jedinice

2.3 Memorija

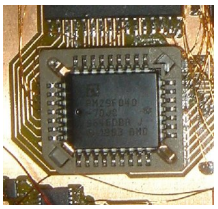
U *tablici 7* je popis svih pet tipova memorije koje su ugrađene u mini386. BIOS trenutno podržava prva četiri tipa, dok zadnji tip nema programsku podršku niti unutar samog mikrokontrolera. Takav tip memorije nije planiran za mini386, stoga podrška za taj tip memorije neće biti implementirana sve dok se ne pokaže potreba.

Tablica 7: Tipovi memorije u mini386

Tip	Veličina	Opis
Flash ROM	512kB	Memorija koju je moguće samo čitati (trenutno romdemux podržava samo čitanje). Sadrži BIOS-ov izvršni kod, te FAT32 virtualni disk.
SRAM[11]	1MB	Radna memorija.
microSD	0 – 4GB	Memorijska kartica koju je moguće izmjenjivati. (Nema podrške SDHC protokol.)
Sat	96B	Memorija se čuva sve dok to napon baterije dopušta.
Flash u mikrokontroleru	2kB	Mikrokontroler ima ugrađenu *flash* memoriju, ali podrška za pristup toj memoriji nije implementirana.

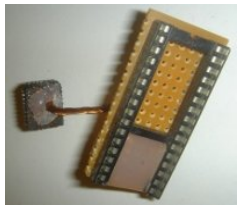
2.3.1 ROM

ROM je namijenjen za izvršni kod BIOS-a, te virtualni disk sa CleanOS-om i još ponekim osnovnim aplikacijama za BIOS i CleanOS. ROM prvenstveno mora biti na neki način promjenljiva memorija kako bi se sustav mogao nadograđivati. Najbolje rješenje bi bila 16-bitna *flash* memorija i sklop u CPLD koji bi podržavao programiranje takve memorije, ali to rješenje je u početku izgledalo veoma riskantno jer CPLD možda ne bi bio dovoljan za takav sklop. Osim toga, kod takvog rješenja je otežan dizajn tiskane pločice i prilikom razvoja bilo bi potrebno dugo čekati da se izvrši svako ponovno programiranje.


Slika 13: ROM

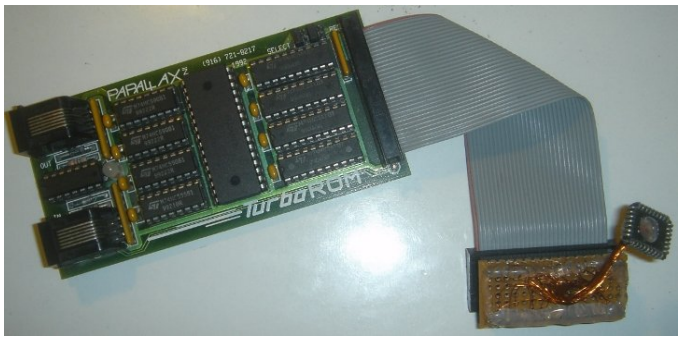
Najpraktičniji odabir je bio integrirani krug AM29F040 u PLCC kućištu koji se uobičajeno koristi kao 8-bitna *flash* memorija. 512kB je najveća moguća memorija koja dolazi u tom pakiranju, što je i više nego dovoljno. Na matičnu pločicu je postavljeno PLCC podnožje kako bi se taj integrirani krug mogao fizički izmjenjivati, ali je i ostalo mjesta za sklop u CPLD-u koji bi omogućio izmjenu podataka.

Prednost ovog rješenja je mogućnost priključivanja ROM emulatora umjesto AM29F040. ROM emulator značajno ubrzava razvoj jer nije potrebno prilikom svake izmjene programske podrške vaditi, programirati, te vraćati AM29F040. Za ROM emulator koji je predviđen samo za DIP podnožje napravljen je adapter na *slici 14* pomoću integriranog kruga istog kućišta kojem je probušena sredina sa


Slika 14: Adapter PLCC u DIP

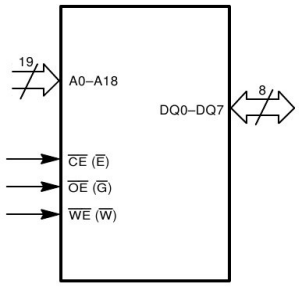
11 Statički RAM; memorija koja pamti samo dok ima napajanje i nije ju potrebno osvježavati

silicijskom pločicom. Tako su oslobođeni svi izvodi, a na vrh svakog od njih je spojena po jedna kratka žica čiji je drugi kraj spojen na pripadajući utor DIP konektora za ROM emulator.



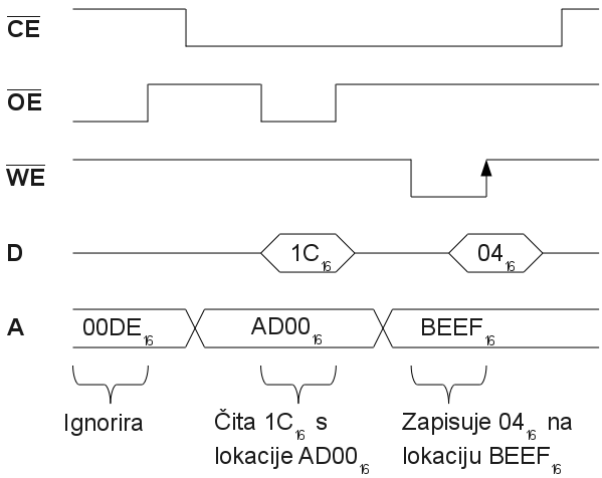
Slika 15: Korišteni TurboROM emulator s priključenim adapterom

Slika 16 prikazuje shematski prikaz integriranog kruga AM29F040. Jedini kontrolni signali su $\overline{CE}$, $\overline{OE}$ i $\overline{WE}$. $\overline{OE}$ i $\overline{WE}$ određuju da li je potrebno čitanje ili pisanje, a $\overline{CE}$ služi kao prozivanje. AM29F040 će se odazvati ako i samo ako je $\overline{CE}$=0.



Slika 16: AM29F040 blok

Ciklus pisanja i čitanja prikazan je na *slici 17*. Prelazak signala $\overline{WE}$ u neaktivno stanje ili $\overline{CE}$ u neaktivno stanje dok je $\overline{WE}$ u aktivnom označava trenutak kad će AM29F040 pročitati podatak s podatkovne sabirnice, a za vrijeme dok je signal $\overline{OE}$ aktivan, AM29F040 će postaviti podatak sa zatražene adrese na podatkovnu sabirnicu.



Slika 17: Ciklus čitanja i pisanja AM29F040

Pisanje podataka u *flash* memoriju nije moguće samo tako da se napravi ciklus pisanja sa željenim podatkom i adresom, već je prvo potrebno nekoliko ciklusa pisanja s točno određenim adresama i podacima kako bi AM29F040 prešao u način rada za zapis 1 bytea. Takav pristup za pisanje je nužan, kako bi se osiguralo da neće doći do pisanja po ROM memoriji zbog neke greške, što bi moglo "uništiti" cijeli sustav, te je omogućeno slanje komandi za brisanje sektora i provjera da li je gotov zapis podatka.

Jedini problem je potreba za pristupom 16-bitnih podataka iz odabrane 8-bitne memorije. Rješenje tog problema je poseban sklop u CPLD-u, detaljno opisan u poglavlju *2.2.3 Romdemux*.

Odabran tip matične pločice ima samo jednu ravninu za povezivanje komponenata, te je puno truda potrebno za dodati žicu koja preskače vodiče na pločici. Da bi se ROM što jednostavnije povezao na tiskanoj pločici nisu sve adrese spojene kako bi to trebalo biti, već kako je bilo najjednostavnije za spojiti. *Tablica 8* prikazuje kako su spojene adrese procesora s adresama ROM-a. Tablicu je moguće ispuniti pomoću sheme s ROM-om na strani 56.

Tablica 8: Spoj adresa ROM-a s procesorom

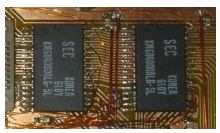
Procesor	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ROM	0	1	2	3	14	12	10	8	11	13	16	15	4	9	7	5	6	17	18

Prvi redak prikazuje broj adresne linije s procesora, a drugi redak na koju adresnu liniju ROM-a je ona spojena. Primjerice, kad procesor zatraži podatak iz ROM-a s adrese 10_{16}, dohvatiti će zapravo onaj na adresi 8000_{15}. Da bi procesor čitao prave podatke, potrebno je "promiješati" podatke unutar ROM-a. Stoga je napravljen poseban program `rommix` koji nakon prevođenja i povezivanja BIOS-ovog izvršnog koda "promiješa" dobivene binarne podatke kako bi ih procesor čitao kao da su adrese dobro spojene.

Pri dodavanju podrške za izmjenu virtualnog diska u ROM-u unutar samog sustava doći će do još većeg problema. AM29F040 podržava samo brisanje sektora veličine 64kB, što znači da bi bilo potrebno brisati i programsku memoriju, te ju vraćati nazad. Moguće je zaobići taj problem ako se virtualni disk postavi na fiksnu lokaciju na kraju ROM-a gdje se adrese A_{17} i A_{18} preklapaju (područje 10000_{16} – $3FFFF_{16}$). No i dalje ostaje potreba za "miješanjem" podataka zbog nižih adresnih linija.

2.3.2 RAM

Kao RAM se koriste dva 8-bitna SRAM integrirana kruga od 512kB, što ukupno daje 1MB. CleanOS nakon učitavanja u memoriju zauzme nešto manje od 40kB, što još uvijek ostavlja i više nego dovoljno memorije za svoje aplikacije i BIOS.



Slika 18: SRAM

Oba integrirana kruga su u SOIC kućištu što značajno pojednostavljuje izradu tiskane pločice jer im je većina izvoda (adrese i neki kontrolni signali) spojena paralelno kao što je moguće vidjeti na *slici 18*.

Sučelje, tj. shematski prikaz SRAM-a je identičan onom od AM29F040 na *slici 16*. Jedina razlika je što SRAM nema ugrađenu nikakvu "pamet", te zapis na zadanu memorijsku lokaciju doslovno znači da taj podatak treba zapisati na tu lokaciju. Primjerice, prema *slici 17* AM29F040 bi ignorirao zapis podatka 04_{16} jer se zadana adresa i podatak ne poklapaju s protokolom za zapis podatka, dok bi SRAM jednostavno sačuvao podatak 04_{16} u svojoj memoriji na lokaciji $BEEF_{16}$.

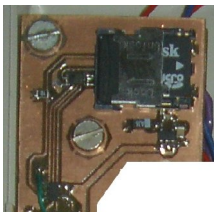
Takvu konfiguraciju dva 8-bitna SRAM integrirana kruga veoma lako je spojiti na 16-bitnu podatkovnu sabirnicu. Prilikom pristupa memoriji (vidi *tablicu 4*), signali $\overline{\text{BHE}}$ i $\overline{\text{BLE}}$ određuju kojem će se integriranom krugu pristupati.

Kako su točno SRAM-ovi spojeni moguće je vidjeti na shemi na strani 56. Na toj shemi je moguće i uočiti sličnost s *flash* ROM-om.

2.3.3 microSD

microSD je nasljednica memorijske kartice SD[12] i danas je jedna od najpopularnijih medija za mobilne uređaje.

Prema dokumentaciji, microSD ne mora podržavati SPI protokol kojeg koristi mini386, ali zbog činjenice da postoje adapteri koji omogućuju da se microSD kartica spoji umjesto veće SD kartice, proizvođači moraju proizvoditi microSD kartice koje su u potpunosti kompatibilne sa starim SD karticama. Stoga je jako mala vjerojatnost da neka microSD kartica neće moći komunicirati s mini386 jer ne podržava SPI protokol. Pogotovo ako se prodaje u paketu s adapterom za SD karticu.

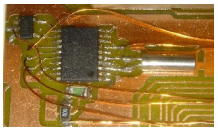


Slika 19: microSD i elektronika

SD i microSD kartice rade isključivo na 3.3V ili manje, stoga je bilo potrebno ugraditi dodatno napajanje (linearni konverter s malim padom napona: TPS73033DBVT), te konverter za ulazne (NC7NZ34K8X) i izlazne (74VHC1GT125) signale. NC7NZ34K8X je vrlo sitan integrirani krug u MAB08A kućištu s tri ugrađena *buffera*[13] koji niskonaponske (3.3V) signale prevode u visokonaponske (5V). 74VHC1GT125 je samo jedan *buffer* u SOT-23 kućištu koji radi upravo suprotno. Jedina razlika je što 74VHC1GT125 ima dodatan ulaz kojim je moguće odrediti kada je na izlazu potrebno stanje visoke impedancije. Bez toga ostale vanjske jedinice na SPI sabirnici ne bi mogle slati podatke CPLD-u. Shema sklopa s microSD karticom nalazi se na strani 60, a napajanje za microSD karticu na strani 61.

BIOS podržava primarne particije, te FAT32 datotečni sustav, ali može samo čitati. Zbog toga što je u konačnici zamišljeno da BIOS pokreće CleanOS koji bi imao svoje funkcije za pisanje i čitanje kartice, nije bilo potrebe trošiti vrijeme za razvoj podrške za pisanje podataka po microSD kartici. Dovoljno je čitanje podataka kako bi se mogla odabrati i pokrenuti BIOS aplikacija.

2.3.4 Sat



Slika 20: Sat i kristal

Sve što je potrebno kako bi integrirani krug za sat DS1305 mogao funkcionirati je kristal od 32.768kHz. Binarnim brojilom tu frekvenciju dovodi do vrlo precizne frekvencije od 1Hz, tj. 1s. Na *slici 20* je moguće vidjeti nekoliko bakrenih žica, te u gornjem lijevom kutu još jedan

12 engl. secure digital
13 sklop koji na izlazu daje isti podatak koji je na njegovom ulazu

integrirani krug. Žice su spojene na SPI sabirnicu, a integrirani krug služi za inverziju $\overline{CS}$ signala kako bi postao CS jer DS1305 zahtijeva pozitivnu logiku za taj signal, a korišteni demultipleksor 74HCT138 daje negativnu logiku na svim izlazima.

Shema sklopa sa satom nalazi se na strani 59.

DS1305 osim svoje namijenjene funkcije praćenja vremena ima dvije dodatne funkcije:

1. dvije budilice koje po isteku vremena šalju procesoru prekid,
2. 96 bytea memorije koja se čuva sve dok to baterija omogućuje.

Memorija u integriranom krugu za sat, kao ni *flash* memorija unutar mikrokontrolera nije planirana da se koristi, ali je poslužila kao neka vrsta provjere da li su podaci u satu točni.

Prvi byte je kontrolni broj za ostalih 95 bytea, a drugi byte je konstanta $4A_{16}$ koja označava da je podatke postavio BIOS. Kontrolni broj se računa kao XOR svih ostalih i ako on nije točan, vjerojatno se uređaj pokreće prvi puta ili je sat zaboravio sve podatke zbog prazne baterije. Kad BIOS ustvrdi da kontrolni broj nije točan, ili drugi byte nije $4A_{16}$, vrijeme se postavlja na 18.12.2009, 4:00:00pm, svi podaci na 0, te se postavlja drugi byte na $4A_{16}$ i izračunava kontrolni broj.

Zbog toga što je BIOS zauzeo prva dva bytea, aplikacijama omogućuje pristup preostalim 94 bytea, te pri svakom zapisu automatski izračunava kontrolni broj.

Osim tih podataka, aplikacije mogu postavljati i budilice. Ali za to prvo moraju dodijeliti svoju prekidnu funkciju jer ju BIOS nema i ako dođe do prekida, sustav više ne bi bio stabilan.

2.4 Veza s vanjskim svijetom

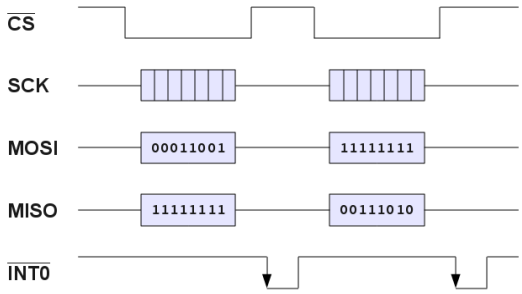
Kako bi se mini386 mogla na neki način koristiti, potrebna je veza s vanjskim svijetom. Kao korisničko sučelje dodana je tipkovnica, grafički LCD, te LED-ice koje prikazuju status. Za komunikaciju s drugim uređajima dodano je serijsko sučelje RS232.

Sve nabrojane vanjske jedinice, osim status LED-ica, kontrolira mikrokontroler AT89S8253. Taj mikrokontroler je novija inačica mikrokontrolera 8051 pogonjena kristalom za takt od 20MHz. Ima ugrađenu programsku *flash* memoriju od 12kB koju je moguće mijenjati nakon ugradnje preko SPI sabirnice dok je signal RESET aktivan.

Mikrokontroler je spojen prema shemi na strani 57.

Sve komande od procesora dobiva preko SPI sabirnice. Sinkronizacija 386 procesora i mikrokontrolera se provodi tako da mikrokontroler šalje procesoru prekid nakon što procesira svaki primljeni byte. Ovisno o tome od koliko byteova se sastoji komanda, taj prekid znači da je potrebno slati daljnje podatke ili da je sve procesirano, te se može slati nova komanda.

Protokol je osmišljen tako da procesor šalje 1 byte s komandom, te nakon toga koliko je potrebno byteova s podacima. Byte za komandu je tako strukturiran da prvih pet bitova označava komandu, a zadnja tri bita označavaju koliko byteova podataka slijedi iza komande. Na taj način je osigurano da će protokol biti sinkroniziran, tj. da se neće pomiješati podaci i komande zbog neke greške. *Slika 21* prikazuje primjer čitanja 1 bytea iz međuspremnika za tipkovnicu. Procesor prvo pošalje komandu 19_{16} koja znači "čitaj tipkovnicu" i primi/šalji jedan podatkovni byte. Nakon prekida procesor šalje "prazni" podatak FF_{16}, dok istovremeno primi podatak s tipkovnice. Signal $\overline{CS}$ ne mora nužno prelaziti u neaktivno stanje, ali mora biti aktivan dok se šalju podaci mikrokontroleru.



Slika 21: Primjer komunikacije procesora s mikrokontrolerom

Drugi prekid kojeg mikrokontroler može slati procesoru označava da se neki izlazni međuspremnik ispraznio ili da je neki ulazni međuspremnik dobio novi podatak. Osim toga označava i istek zadanog vremena kojeg je moguće koristiti kao *timer* za OS. Kako bi se doznalo o kojoj vrsti prekida se radi, potrebno je poslati komandu sa zahtjevom status bytea.

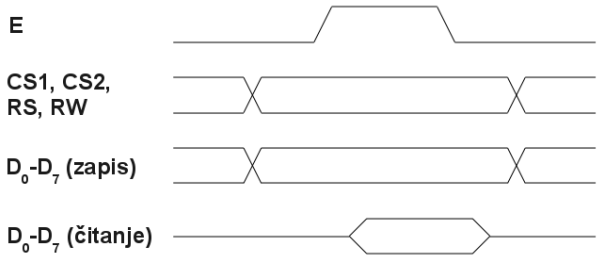
2.4.1 LCD

Za LCD se koristi modul GDM12864A rezolucije 128×64 piksela. Modul ima ugrađeni DC/DC konverter za generiranje negativnog napona potrebnog za podešavanje kontrasta, pa ga nije bilo potrebno dodatno ugrađivati.

Modul je smješten točno na sredinu poklopca kućišta između tiskane pločice sa status LED-icama i tiskane pločice za napajanje. Kako do modula s matične pločice ide 16 tankih žica, prilikom spajanja, zajedno su pletene kako bi stajale uredno kao jedna deblja žica.

Slika 22: LCD modul prikazuje izbornik

Modul se interno sastoji od dva djelomično odvojena sustava. Svaki za svoju polovicu prikaznika. Signali CS1 i CS2 prilikom komunikacije određuju s kojom polovicom se komunicira, signal RS označava da li se komunicira s kontrolnim ili podatkovnim registrom, a signal WR da li se čita ili piše. Nakon što se postave svi signali, potrebno je napraviti impuls na signalu E. Samo u slučaju čitanja podataka s LCD-a je potrebno za vrijeme dok je signal E=1 pročitati podatke s LCD-a. Univerzalni grafički prikaz komunikacije s LCD-om prikazan je na *slici 23*.



Slika 23: Komunikacija s LCD-om

LCD modul je vrlo spora vanjska jedinica, bilo je potrebno ubaciti 8 NOP[14] instrukcija prije postavljanja signala E i isto toliko prije gašenja signala E.

14 engl. no operation – instrukcija koja ne radi ništa

Jedine funkcije koje modul nudi su mijenjanje bilo kojeg dijela memorije, te određivanje grafičkog retka koji će biti na vrhu. Druga funkcija je veoma korisna prilikom pomicanja teksta prema gore prilikom prelaska u novi red (engl. *scroll*). Bez te funkcije bi bilo potrebno pristupiti svakoj memorijskoj lokaciji otprilike dva puta kako bi se pomaknula slika prema gore, a to bi veoma dugo trajalo.

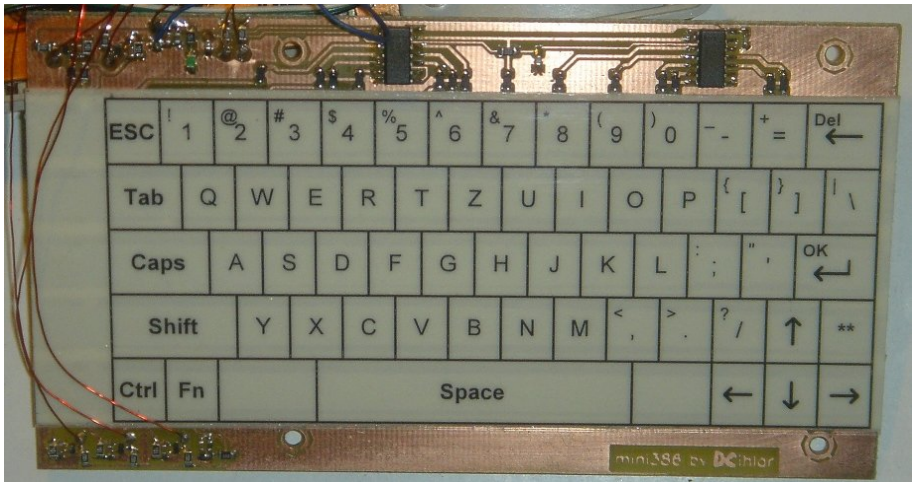
2.4.2 Tipkovnica

Posebna tipkovnica je dizajnirana kako bi pristajala u kućište mini386. Tipkovnica je trebala biti tanka, te imati sve osnovne tipke koje ima uobičajena tipkovnica za PC računalo. Kao inspiracija je iskorišten jedan stariji prijenosnik. Uz uobičajene tipke, dodana je jedna posebna tipka "**" koja bi pod CleanOS-om imala funkciju Alt+Tab[15] kombinacije tipaka za odabir aplikacije koju korisnik želi koristiti. Tipka "Ctrl" je ostavljena kako bi se mogle raditi funkcijske kombinacije tipaka (primjerice: Ctrl+C = spremanje odabranog teksta u privremenu memoriju), a tipka "Fn" bi mogla poslužiti u slučaju da neka tipka nedostaje (primjerice: Fn+→ bi moglo poslužiti umjesto tipke "End" za prelazak na kraj trenutnog retka).

Osim dizajna, bilo je potrebno osmisliti samu izradu tipkovnice. Korišteni su materijali koji se lako mogu naći. Osnova tipkovnice je matrica vodoravnih i okomitih vodiča. Okomiti vodiči su na samoj tiskanoj pločici, a vodoravni su trake od aluminijske folije zalijepljene tankom bijelom (neprozirnom) dvostruko ljepljivom trakom za foliju na kojoj je slika s tipkama. Kako se vodoravni i okomiti vodiči ne bi dodirivali dok nisu pritisnuti, folija s trakama je zalijepljena za pločicu pomoću dvostruko ljepljive trake debele oko 1mm tako da je narezana na uske trakice koje su zalijepljene između redaka s tipkama.

S lijeve strane tipkovnice su posebni kontakti koji omogućuju da se vodoravni aluminijski vodiči spoje s pločicom kako bi se mogli dalje spajati. Veza između folije i pločice na tim kontaktima ostvarena je pomoću debljih smotaka aluminijske folije koji su samo utaknuti između kontakta na pločici i aluminijskih traka na foliji.

15 grafičko sučelje popularnih operacijskih sustava za PC tu kombinaciju koristi za odabir aktivne aplikacije



Slika 24: Tipkovnica

Skeniranje tipkovnice radi na principu prozivanja stupca i ispitivanja retka. Stupac je prozvan kad je na njemu napon od 5V, a nije prozvan kad je u stanju visoke impedancije. Prozivanje stupaca vrše dva 8-bitna posmačna registra spojena u seriju. Mikrokontroler ih inicijalno sve postavi u 0 (tj. stanje visoke impedancije) tako da na podatkovni ulaz prvog posmačnog registra postavi logičku 0, te na ulaz za takt pošalje 16 perioda. Kada je potrebno skenirati tipkovnicu, pošalje se jedna logička 1, te nakon toga 15 logičkih 0. Pri svakom ubacivanju jedne 0, prethodno ubačena jedinica se pomiče za jedno mjesto u desno. Na taj način je dobiveno prozivanje jednog po jednog stupca.

Ispitivanje redaka je provjera da li je neki redak spojen na izvor 5V ili nije. Tu funkciju vrši N-MOS tranzistor s RC filtrom koji sprečava utjecaj visokofrekventnih smetnji iz matične pločice na skeniranje tipkovnice.

Prilikom skeniranja, kad se ustvrdi da nisu svi vodoravni vodiči 0V, poznato je koji je stupac prozvan i koji vodoravni vodiči su 5V. Iz toga su poznate i lokacije pritisnutih tipaka.

Shema tipkovnice nalazi se na strani 65.

2.4.3 Status LED-ice i digitalno podesivi otpornici

Lijevo od LCD modula nalazi se tiskana pločica sa status LED-icama, te sustavom za podešavanje kontrasta i pozadinskog osvjetljenja.

Status LED-ice su četiri svjetleće diode na dnu tiskane pločice:

1. zelena – upaljen je "Shift Lock",
2. zelena – upaljen je "Caps Lock",
3. žuta – označava pristup microSD kartici,
4. crvena – označava da je došlo do neke greške.

LED-icama upravlja posmačni registar 74HC595. Zahvaljujući tome što je SPI sabirnica bazirana na posmačnom registru, 74HC595 je praktički direktno spojen na nju. Jedino je bilo potrebno izlazni bit odvojiti od sabirnice pomoću *buffera* koji omogućuje ostalim vanjskim jedinicama da šalju svoje podatke CPLD-u. Identičan princip se koristi u poglavlju *2.3.3 microSD*.

Osim za kontrolu LED-ica, 74HC595 služi za paljenje i gašenje pozadinskog osvjetljenja LCD-a. *Tablica 9* prikazuje namjenu svakog pojedinog bita posmačnog registra 74HC595.

Tablica 9: 74HC595 registar

LEDs reg	BL	-	-	LED4	LED3	LED2	LED1	-
bit	7	6	5	4	3	2	1	0

Funkcije pojedinih bitova su:

- BL – aktivacija pozadinskog osvjetljenja,
- LED1 - LED4 – LED-ice redom kojim su funkcionalno opisane na početku ovog poglavlja.

Integrirani krug MCP4231 ima ugrađena dva digitalno podesiva otpornika od 10kΩ, izvedena pomoću 128 otpora spojenih u seriju, te analognog multipleksora sa 129 ulaza koji omogućuje odabir srednjeg kontakta digitalno podesivog otpornika.

Prvi otpornik koristi se za regulaciju kontrasta. Budući da je LCD-u za kontrast potreban napon u rasponu 0 do -10V, nije moguće direktno koristiti digitalno podesivi otpornik jer on radi isključivo unutar područja napona napajanja, tj. 0 do +5V. Iz tih naponskih raspona je očito da je moguće drugi raspon dobiti množenjem prvog s -2. Pomoću digitalno podesivog otpora moguće je određivati napon između 0 i +5V, te operacijskim pojačalom TS461CLT koje je spojeno tako da pojačava -2 puta, možemo dobiti napon potreban za kontrast LCD-a.

Shema tog sklopa, kao i sklopa s 74HC595 registrom nalazi se na strani 60.

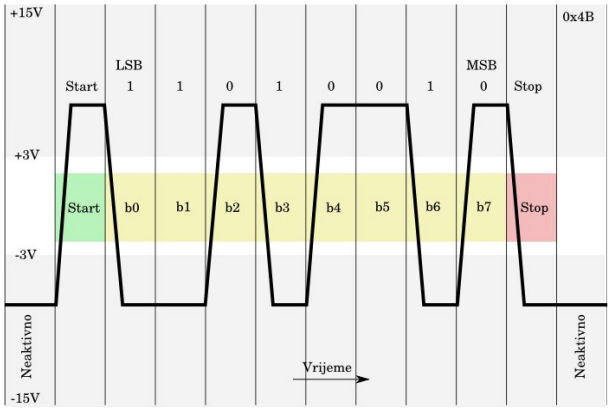
Drugi otpornik koristi se za regulaciju intenziteta pozadinskog osvjetljenja. BL bit u 74HC595 registru određuje da li će DC/DC konverter za pozadinsko osvjetljenje raditi ili ne. Svojstva elementa za pozadinsko osvjetljenje su takva da može raditi u rasponu 3-4.2V, dok pri 4.2V troši oko 200mA. Integrirani krug TPS62200 odabran je kao DC/DC konverter zbog toga što ima sve što je potrebno: signal za gašenje, ulaz za regulaciju izlaza, te na izlazu može dati preko 200mA. Shema na strani 61 prikazuje kako je TPS62200 spojen. Ulaz za povratnu vezu spojen je djelitelj napona R1 i R2, s time da je u seriju s otpornikom R1 spojen digitalno podesivi otpornik kako bi bilo moguće podešavanje izlaznog napona. Pri izlazu željenog napona, na ulazu za povratnu vezu potreban je napon od 0.5V. Stoga, prema formuli [2] odabrane su vrijednosti za otpore R1 i R2.

$$U_{BL} = 0.5\text{V} \cdot \frac{R1 + R_{BL} + R2}{R2} \qquad [2]$$

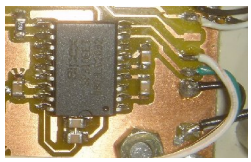
U_{BL} je izlazni napon, a R_{BL} je digitalno podesivi otpornik. Kako nije uvijek pametno nabavljati otpornike nestandardnih vrijednosti, odabrani su lako dostupni otpornici. Stoga nije dobiven precizan naponski raspon na izlazu već raspon 3.5-4.5V. Bitno je da se napon maksimalnog osvjetljenja 4.2V nalazi unutar raspona, te da je moguće smanjiti intenzitet osvjetljenja do vrijednosti koja je prihvatljiva u mraku. Programski je moguće zabraniti prijelaz preko 4.2V kako ne bi došlo do pregrijavanja DC/DC konvertera i zavojnice koja je zbog svojih malih dimenzija na granici specifikacija za potrebnu struju.

2.4.4 Serijsko sučelje

Serijsko sučelje je standardno RS232[3] sučelje. Radi se o asinkronom dvosmjernom serijskom prijenosu podataka. Logička 1 određena je naponom između -3 i -15V, a logička 0 naponom između 3 i 15V. Podaci se šalju tako da se prvo pošalje start bit (0), 8 bita podataka od najnižeg prema najvišem bitu, te stop bit (1). Primjer slanja bytea $4B_{16}$ prikazan je na *slici 25*. Prije stop bita moguće je slati bit s paritetnom zaštitom, ali zbog jednostavnosti i velike pouzdanosti ovog sučelja taj bit nije podržan. Ako je iz bilo kojeg razloga potrebna zaštita, moguće ju je dodati na razini slanja podataka, odnosno, u protokolu.



Slika 25: Slanje podatka $4B_{16}$ preko RS232



Slika 26: SP310E

Mikrokontroler ima sklopovsku podršku za takav serijski prijenos podataka, ali nema ugrađenu potrebnu konverziju signala. Stoga je dodan dodatni integrirani krug SP310E koji ima dva ugrađena DC/DC konvertera za +10 i -10V koji mu omogućuju slanje signala s potrebnim naponskim razinama. SP310E za razliku od ostalih sklopova koji obavljaju istu tu funkciju (primjerice, vrlo popularni MAX232) nudi i signal za gašenje kako bi bilo moguće uštedjeti na energiji dok nije potrebno serijsko sučelje.

Priključnica za serijsku vezu je ista kao i za slušalice. 3.5mm konektor s tri kontakta. Jedan za uzemljenje i dva za podatkovne signale. Kako bi bilo moguće koristiti takav konektor, a i kako bi općenito serijsko sučelje bilo jednostavnije, nisu podržani signali za kontrolu toka podataka.

Atfir (*firmware*, detaljnije objašnjen u poglavlju *3.1 Atfir*) prilikom pokretanja brzinu prijenosa postavi na 9600 bit/s, ali podržava izmjenu brzine prijenosa. Najveća standardna brzina je 19200 bit/s. Pri većim brzinama dolazi do prevelikog odstupanja od nazivne brzine, te bi bilo potrebno zamijeniti mikrokontrolerov kristal. Prilikom izmjene brzine prijenosa, potrebno je izračunati vrijednost registra kojeg mikrokontroler koristi za dijeljenje frekvencije kako bi dobio takt za slanje podataka pomoću formule [3].

$$REG = 256 - \frac{f_{mikrokontrolera}}{96 \cdot brzina} = 256 - \frac{20 \cdot 10^6}{96 \cdot brzina} \qquad [3]$$

Inicijalno Atfir gasi SP310E, stoga je prije upotrebe serijskog sučelja potrebno poslati komandu za paljenje SP310E.

Shema sa sklopom SP310E nalazi se na strani 58.

2.4.5 ISP priključak

ISP[16] priključak omogućuje programiranje CPLD-a i mikrokontrolera. Sa sheme na strani 58 moguće je vidjeti da ovaj priključak omogućuje pristup JTAG i SPI sabirnici. JTAG je standard koji osim za provjeru vodova na tiskanoj pločici može koristiti i za programiranje integriranih krugova. JTAG signali su TCK, TMS, TDI i TDO. CPLD koristi JTAG za programiranje svoje logike.

Slika 27: ISP priključak

SPI sabirnica koristi se za programiranje mikrokontrolera, a kako bi programiranje mikrokontrolera moglo započeti, na ISP priključak doveden je i signal RESET.

Signal $\overline{\text{SPI\_DBG\_CS}}$ je dodan za slučaj da će tokom uhodavanja biti potreban $\overline{\text{CS}}$ signal sa integriranog kruga 74HC138, ali kako ipak nije bilo potrebe za tim signalom, nije u potpunosti niti spojen.

16 engl. in circuit programmer – programator sadržan u sklopu

2.5 Napajanje

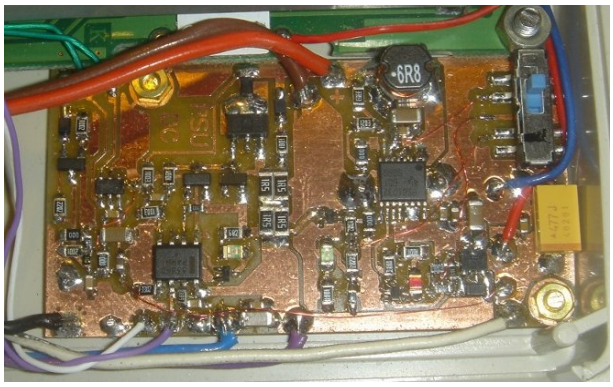
mini386 je prijenosno računalo, te mu je potrebna baterija kao izvor energije. Baterija bi trebala omogućiti da uređaj radi barem jedan sat dok je upaljen. To je dovoljno vrijeme za potrebe prosječnog korisnika. Uz bateriju je potreban i sustav za punjenje baterije. Kako punjenje baterije nije moguće dok je na nju spojeno trošilo, potreban je i sustav koji određuje da li će se uređaj napajati iz baterije ili iz vanjskog izvora.

Cijeli sustav radi na 5V, osim microSD kartice. Za microSD karticu je posebno ugrađeno napajanje, kao što je opisano u poglavlju *2.3.3 microSD*. Stoga je potreban samo naponski izvor od 5V. Još je samo bilo potrebno odrediti drugi parametar: maksimalnu struju koju će mini386 trošiti. Eksperimentalno je ustvrđeno da pri normalnom radu mini386 troši oko 400mA bez pozadinskog osvjetljenja. Uz pozadinsko osvjetljenje bi potrošnja bila oko 600mA. Po formuli [4] za snagu dobije se da se radi o ukupnoj snazi od 3W. To je vrlo velika potrošnja za neki mobilni uređaj i predstavlja veliki problem prilikom izrade napajanja i odabira baterije.

$$P = U \cdot I \qquad [4]$$

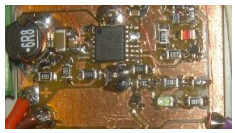
Najveći potrošač je CPLD. Dok ništa ne radi troši konstantno 300mA, što znači da samo CPLD doprinosi potrošnju od 1.5W. Stoga je pri izradi svih shema bilo moguće zanemariti svaku potrošnju manju od 1mA, a to nije uobičajeno za mobilne uređaje. Osim toga, bilo je potrebno odustati od predviđenog *standby* načina rada jer bi CPLD bio potreban za isključivanje takta procesora, te zato ne bi imalo smisla isključiti CPLD, pa bi potrošnja u *standby* načinu rada bila jako mala ušteda spram normalnog načina rada.

Slika 28 prikazuje tiskanu pločicu sa cijelim sustavom za napajanje. Kad se mini386 zatvori, ispod te pločice se nalazi baterija. Zbog nje je bilo slobodno jedva 2mm za komponente na površini pločice. Kako je baterija okrugla, više komponente su mogle doći na lijevi rub, a kako nije dugačka koliko i kućište, bilo je mjesta na vrhu pločice za prekidač i elektrolitski kondenzator.



Slika 28: Tiskana pločica za napajanje

2.5.1 DC/DC konverter



Slika 29: DC/DC konverter

DC/DC konverter vrši pretvorbu jednog napona u drugi. U ovom slučaju je potreban konverter s točno određenim naponom od 5V na izlazu, ali s varijabilnim ulaznim naponom (zbog pražnjenja baterije) do maksimalno 4V. Stoga će biti potreban "step up" DC/DC konverter, tj. konverter s nižeg istosmjernog napona na viši istosmjerni napon.

Kao što je već prije rečeno, konverter mora u najgorem slučaju dati 600mA. Najbolji konverter kojeg je bilo moguće nabaviti za takve potrebe je TPS61032. Taj integrirani krug može dati do 1A pri 5V na izlazu iz samo 1.8V na ulazu. Ključna komponenta za njegovu funkcionalnost je zavojnica od 6.8µH. Ostale komponente su uglavnom kondenzatori za stabilizaciju napona, te kao blokada visokofrekventnih smetnji.

TPS61032 je u posebno prilagođenom kućištu sa metalnom pločicom na dnu. Pločica je u direktnom kontaktu sa samom silicijskom pločicom tog integriranog kruga, te tako omogućuje vrlo efikasno hlađenje. Budući da tu metalnu pločicu nije moguće zalemiti u kućnoj radinosti, premazana je s termalnom pastom kako bi došla u direktan kontakt s bakrenom površinom na tiskanoj pločici.

Pri izradi tiskane pločice je bilo potrebno uzeti u obzir velike struje koje idu kroz ulazne vodiče, te je bilo potrebno napraviti što veću površinu sa uzemljenjem kako bi se TPS61032 mogao hladiti. Prilikom uobičajenog rada mini386 nije primijećeno značajnije grijanje tog integriranog sklopa.

Shema opisanog sklopa s DC/DC konverterom nalazi se na strani 62.

2.5.2 Baterija

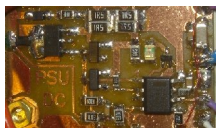
Posebna pažnja je posvećena prilikom odabira baterije. Baterija je trebala biti dovoljno mala da stane u predviđeno mjesto u kućištu od mini386, ali i dovoljno jaka kako bi mogla davati 3W otprilike 1h.

Pronađena je NiMH baterija namijenjena za električne bušilice. Dvije valjkaste ćelije spojene u seriju stanu u kućište.

Baterija je nominalnog kapaciteta 2200mA/h. Ako se zanemari promjena napona prilikom pražnjenja i pretpostavi da je konstantni napon obje ćelije 2.4V, uvrštavanjem u formulu [4] možemo dobiti da takva baterija može davati 5.28W sat vremena. Stoga bi pri maksimalnom opterećenju baterija mogla izdržati 1:45h rada. Ali u praksi se zapravo radi o nešto manjem vremenu zbog toga što se baterija ne smije istrošiti do kraja, te nije konstantan napon od 2.4V, već može pasti do 1.8 kad DC/DC konverter prestaje raditi.

Nije obavljeno eksperimentalno mjerenje vremenske izdržljivosti baterije.

2.5.3 Punjač baterije



Slika 30: Punjač baterije

Punjač baterije temeljen je na integriranom krugu MC33340, te strujnom izvoru.

Nažalost, nije bilo moguće naći integrirani krug u kojem je sve potrebno za punjenje baterije već ugrađeno. Vjerojatno iz vrlo raznolikih zahtjeva prilikom izrade punjača proizvođači rade univerzalne integrirane krugove. Doduše, postoje integrirani krugovi s ugrađenim sklopom za strujni izvor, ali takvog tipa da je potrebno puno dodatnih komponenata od kojih je jedna zavojnica. Zavojnice su uglavnom po dimenzijama relativno visoke komponente, te bi tiskana pločica bila previsoka i baterija koja je točno ispod tiskane pločice za napajanje više ne bi stala.

Stoga je izabran MC33340 kao najjednostavniji integrirani krug za kontrolu punjenja baterije, a punjač u pravom smislu je zapravo strujni izvor kojeg pali i gasi MC33340.

Shema na strani 63 prikazuje kompletan sklop za punjenje baterije, te A/D konverter za mjerenje stanja baterije. Tranzistori Q3 i T1 su u spoju strujno-ograničenog izvora. Otpori R7-R10 su efektivno jedan otpor od 1.5Ω, ali veće snage od samo jednog otpornika. Nazovimo taj otpor R_k. Q3 vodi struju dok je napon na R_k manji od 0.6V. Nakon što taj napon počne rasti preko 0.6V, tranzistor T1 će sve više smanjivati struju I_{BE} tranzistora Q3. Na taj način se održava pad napona na Rk otprilike 0.6V, a time struja kolektora Q3, tj. punjenja baterije oko 400mA. Do toga je moguće doći uvrštavanjem napona 0.6V i otpora R_k=1.5Ω u formulu [5].

$$I = \frac{U}{R} \qquad [5]$$

P-MOS Q4 kontroliran MC33340 u potpunosti zaustavlja struju I_{BE} tranzistora Q3 kako bi zaustavio punjenje.

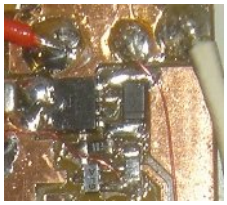
Dioda D2 služi za zaštitu baterije od pražnjenja kroz tranzistor Q3 dok je punjenje onemogućeno.

Osim kontroliranog punjenja, dodana je podrška i za punjenje "na kapaljku" (engl. *trickle charge*). Radi se o konstantnoj struji punjenja koja je toliko mala da ne može oštetiti bateriju, a kompenzira parazitsko pražnjenje baterije. Za tu funkciju su zaduženi otpornik R15, te dioda D3.

A/D konverter je naknadno prespojen na $0.5 \cdot VCC$ jer bi u protivnom uzemljio V_{sen} ulaz integriranog kruga MC33340 dok je mini386 ugašen. Kako bi bilo moguće mjerenje stanja baterije pomoću tog A/D konvertera, potrebno bi bilo napraviti još jedno naponsko dijelilo kao ono na shemi s otporima R12, R13 i C8. Shema sklopa s A/D konverterom nalazi se na strani 63.

2.5.4 Sustav za odabir izvora

Sustav za odabir izvora određuje koji izvor će se koristiti za napajanje. Vanjski izvor mora imati prioritet kako bi se smanjilo opterećenje baterije, te omogućilo njeno punjenje, a kad vanjski izvor nije prisutan, potrebno je omogućiti DC/DC konverter.



Slika 31: Sustav za odabir izvora

Shema na strani 64 prikazuje glavni dio sheme tog sustava. N-MOS Q1:A i Q1:B su unutar istog kućišta i oni su ključ odabira izvora. Oba dva mogu podnijeti struju od 4A sa jako malim padom napona. Q1:B je

u spoju diode s ekstremno malim padom napona. On sprečava protjecanje struje iz vanjskog izvora u DC/DC konverter. Q1:A omogućuje gašenje sustava dok je priključen vanjski izvor.

Shottkyjeva dioda D1 sprečava nepoželjno "curenje" struje iz DC/DC konvertera u ulaz za vanjski izvor. Primjerice, kad je uređaj upaljen i na punjenju, a nestane struje, upalio bi se DC/DC konverter, te slao nepotrebno struju u sklop unutar vanjskog izvora. Ta dioda je bila loše rješenje jer sustav s vanjskim izvorom sad radi na 4.5V umjesto 5V i trebalo bi ju zamijeniti MOSFET-om. Pad napona napajanja ne utječe na logiku mini386, ali primjetno smanji kontrast na LCD-u. Drugo alternativno rješenje bi bilo da se ne odabire direktno izvor napajanja za cijeli sustav, već da se bira izvor za DC/DC konverter. Tako bi uvijek na izlazu bio reguliran napon od 5V, te ne bi bilo potrebe za složenijim sklopom za odabir izvora. Dovoljan bi bio samo jedan MOSFET i jedna dioda. Paljenje i gašenje bi bio samo problem DC/DC konvertera (trenutna realizacija se brine za gašenje DC/DC konvertera i za gašenje vanjskog izvora). Druga velika prednost alternativnog rješenja odabira izvora bi bila i ta što se sustav ne bi ugasio prilikom isključivanja vanjskog izvora. Naime, zbog trenutne realizacije nakon što se isključi vanjski izvor, DC/DC konverter je neaktivan, te mu je potrebno neko vrijeme kako bi se pokrenuo.

Jedini dodatak toj shemi je N-MOS Q7 na strani 62. Q7 gasi DC/DC konverter kad je prisutan vanjski izvor napajanja.

Shema s glavnim dijelom sustava za odabir izvora dodatno ima sklop za generiranje RESET signala koji aktivira cijeli sustav tek nakon što se napajanje stabilizira.

2.6 Izrada tiskanih pločica

Sve pločice su izrađene u kućnoj radinosti. Zbog vrlo složene procedure za izradu pločica s dva sloja vodiča, sve pločice su prilikom dizajniranja prilagođene za jedan sloj vodiča. Za izradu shema i nacrta korišten je alat P-CAD 2006.

Kako je nemoguće napraviti složene pločice sa samo jednim slojem vodiča, korištene su dvije metode za premošćivanje.

1. Otpornici od 0Ω – SMD otpornike u 0805 kućištu od 0Ω moguće je koristiti za premošćivanje jednog vodiča.

   Na shemama "JUMPER_0805" označava takav premosnik, te je njegovo prisustvo moguće ignorirati prilikom shvaćanja koncepta sklopa. Te komponente su na shemu dodane prilikom dizajniranja tiskanih pločica.

2. Naknadno dodane tanke žice – Tanke lakirane bakrene žice korištene su za premošćivanje više vodiča, te za premošćivanje velikih udaljenosti.

Jedina iznimka je pločica za napajanje. Ona ima dva sloja, ali je drugi sloj zbog jednostavnosti samo jedan jedini vodič. Tj. cijela površina je bakrena i koristi se kao vodič za uzemljenje. Osim toga, korisna je i kao velika površina za hlađenje DC/DC konvertera.

Zbog toga što su sve komponente u SMD kućištu, od kojih neke imaju razmak između izvoda oko 0.3mm, bio je potreban foto-postupak. Svi predlošci su tiskani zrcaljeno na termo foliju, pritisnuti pomoću stakalca na foto-pločicu te je takav "sendvič" osvijetljen UV-svjetlom. Nakon toga su pločice razvijene u natrijevom hidroksidu, te jetkane u mješavini solne kiseline i vodikovog peroksida.

Pločice su direktno vezane žicama jedna za drugu kako bi izrada bila jednostavnija i jeftinija. Osim toga, nije potreban dodatan prostor za priključke.

Na nekim shemama je moguće primijetiti da nedostaje oznaka vrijednosti komponente, te je umjesto vrijednosti postavljena oznaka "{Value}". Radi se najčešće o elektrolitskim kondenzatorima za stabilizaciju napajanja za koje vrijedi pravilo "što veći to bolji", tj. o komponentama za koje nije bitna vrlo precizna vrijednost, te je moguće iskoristiti bilo što se već nalazi na zalihama. Također, ta ista oznaka se koristi i za komponente kod kojih je vrijednost bilo potrebno naknadno odrediti, tj. bilo je potrebno eksperimentalno potvrditi potrebnu vrijednost.

3 Programska podrška

Programska podrška podijeljena je na tri dijela:

1. **Atfir** – *firmware* u mikrokontroleru

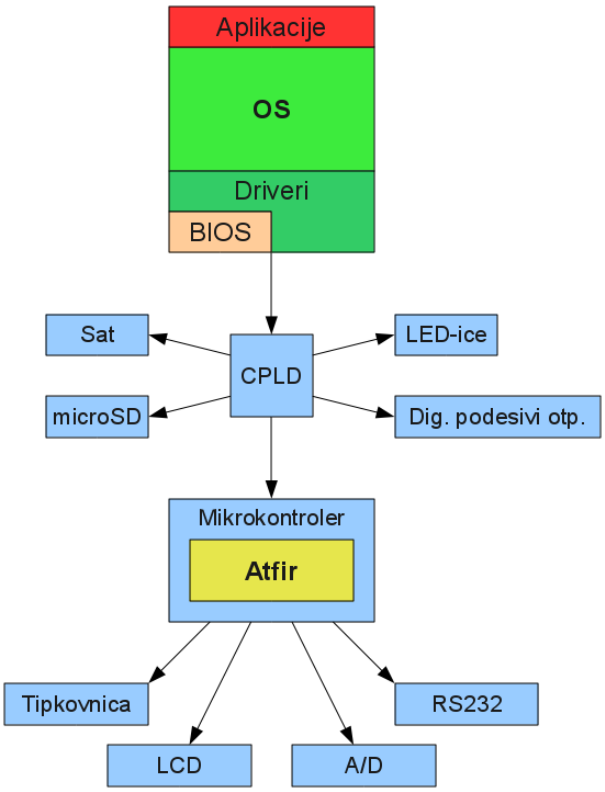
   Atfir je posebna vrsta programske podrške koja je na najnižoj razini, tj. nalazi se u veoma uskom dodiru sa sklopovljem.

2. **BIOS** – osnovna programska podrška za mini386

   BIOS je također u dodiru sa sklopovljem jer omogućuje primjerice izmjenu vremena u integriranom krugu za sat. No upravljanje sklopovljem je samo dio BIOS-ove funkcije. Osim upravljanja sklopovljem, BIOS mora poznavati FAT32 datotečni sustav, kao i ELF zapis kako bi mogao pokretati zadane aplikacije.

3. **BIOS-aplikacije** – programi koje BIOS može pokretati

   Aplikacije su korisnički programi raznih namjena. U konačnici, CleanOS je zamišljen da bude aplikacija za mini386 BIOS.



Slika 32: Blok dijagram programske podrške

Slika 32 prikazuje hijerarhijski prikaz zamišljene programske podrške. Plavom bojom označene su sklopovske komponente, a ostalim bojama komponente programske podrške.

Na samom vrhu nalaze se korisničke aplikacije. One komuniciraju isključivo s operacijskim sustavom (OS) koji im omogućuje pristup driverima[17]. Osim toga, operacijski sustav zadužen je za upravljanje memorijom, te omogućuje višedretvenost. Driveri su dio operacijskog sustava. Sklopovljem mogu upravljati direktno, ali i preko BIOS-a.

Operacijski sustav je zapravo BIOS-aplikacija. Kako CleanOS nije u potpunosti prilagođen za mini386 arhitekturu, trenutno su gornja tri sloja (aplikacije, OS i driveri) zamijenjena samo s BIOS-aplikacijom.

BIOS-aplikacije nemaju nikakvih ograničenja, te mogu sklopovlju pristupati direktno i preko BIOS-a. Osim u slučaju operacijskog sustava, BIOS-aplikacije su jednostavne aplikacije poput korisničkih koje bi pokretao operacijski sustav. Stoga najčešće nema potrebe za direktnim pristupom sklopovlju.

17 engl. drivers – programska podrška za upravljanje sklopovljem

3.1 Atfir

Učestali naziv za programski kod mikrokontrolera je *firmware*, ali u ovom slučaju još koristim naziv "Atfir" što je skraćenica od "Atmel firmware".

Atfir podržava slijedeće:

- funkcije za crtanje po ekranu: postavljanje piksela, crtanje linija, pravokutnika, krugova, te ispis teksta s dva različita fonta,
- skeniranje tipkovnice (ne vrši se dekodiranje u ASCII),
- slanje podataka na serijsko sučelje,
- očitavanje A/D pretvornika koji mjeri napon napajanja,
- slanje prekida procesoru u zadanom taktu.

3.1.1 Komunikacija s procesorom

Procesor šalje komande mikrokontroleru preko SPI sabirnice kao što je to opisano u poglavlju *2.4 Veza s vanjskim svijetom*. Prihvaćanje podataka sa SPI sabirnice izvodi se u prekidnoj funkciji. Podatke koje je moguće brzo obraditi, obrađuje se u prekidnoj funkciji, ali svi ostali podaci obrađuju se u beskonačnoj petlji u glavnom programu. Stoga na obradu takvih podataka, potrebno je duže čekati.

Tablica 10 prikazuje sve podržane komande koje procesor može slati mikrokontroleru. Jedina iznimka su komande odabir ispisa prozirnog ili neprozirnog teksta. Trenutno je moguće samo ispisivati proziran tekst.

Tablica 10: Podržane komande

Komanda$_{16}$	**Opis**
Bez dodatnih podataka	
00	Ne radi ništa (NOP komanda)
08	Briše ekran
10	Odabir crtanja s bijelom bojom
18	Odabir crtanja s crnom bojom
20	Onemogući RS232 konverter signala
28	Omogući RS232 konverter signala
30	Isprazni međuspremnik za serijsku komunikaciju
38	Isprazni međuspremnik za tipkovnicu
40	Onemogući kursor
48	Omogući kursor
50	Zamijeni boju (ako je bila crna, postavi bijelu i suprotno)

Komanda$_{16}$	**Opis**
58	Zahtjev za visinom trenutnog fonta
60	Ispis teksta s brisanjem pozadine (nije implementirano)
68	Ispis prozirnog teksta (nije implementirano)
1 dodatan podatak	
01	Zahtjev za status-zastavicama
09	Ispis znaka na LCD
11	Zahtjev za zauzećem međuspremnika tipkovnice
19	Čitanje podatka s tipkovnice
21	Slanje podatka na serijsko sučelje
29	Čitanje podatka sa serijskog sučelja
31	Zahtjev za zauzećem izlaznog međuspremnika serijskog sučelja
39	Zahtjev za zauzećem ulaznog međuspremnika serijskog sučelja
41	Odabir fonta
49	Odabir brzine serijskog prijenosa
51	Odabir znaka za mjerenje širine i vraćanje rezultata prethodnog mjerenja (SPI omogućuje nezavisan prijem i predaju)
59	Zahtjev za izmjerenim naponom napajanja
61	Brisanje zadanog znaka na prikazniku (primjerice, brisanje znaka 'c' znači da će biti izbrisano područje ispred kursora širine znaka 'c')
2 dodatna podatka	
42	Postavljanje piksela na zadane koordinate
4A	Pozicioniranje kursora na zadanu koordinatu
52	Zahtjev za trenutnom pozicijom kursora
3 dodatna podatka	
03	Postavljanje timera (parametri: visoki i niski byte za generiranje frekvencije, byte za omogućavanje i onemogućavanje timera)
0B	Crtanje kruga na LCD
4 dodatna podatka	
44	Crtanje linije na LCD
4C	Crtanje pravokutnika na LCD
54	Crtanje ispunjenog pravokutnika na LCD
7 dodatnih podataka	
07	Crtanje N (0-4) stupaca od 8 piksela (parametri: x, y, N, d[4])

3.1.2 Podrška za LCD

Korišteni LCD modul podržava samo funkcije za mijenjanje grafičke memorije i *scrollanje*. Stoga je bilo potrebno programski implementirati sve grafičke funkcije, kao i sam ispis teksta, te fontove i blinkanje kursora.

Isječci iz knjige na web stranicama [4] i [5] sadrže algoritme koji su korišteni prilikom izrade funkcija za crtanje linija i krugova. Crtanje pravokutnika i ispunjenog pravokutnika ne koristi isključivo funkciju za crtanje linije. Umjesto funkcije za crtanje linije, koristi se posebno optimirana metoda ispune i crtanja okomitih linija kako bi broj pristupa LCD-u bio što manji. Zahvaljujući web stranici [6] gdje je korišten identični LCD modul, preuzeo sam kompletno rješenje za ispis teksta, te 2 osnovna fonta širine 4 i 5 piksela. Na taj način je riješen velik dio podrške za LCD. Rješenje je dodatno prepravljeno kako bi Atfir omogućio automatski prelazak u novi red. Ako se jednom ukaže prilika, moguće je dodati još fontova s te stranice. Jedino je potrebno u font ubaciti na početak byte koji označava koliko je font visok u pikselima.

3.1.3 Podrška za tipkovnicu

Skeniranje tipkovnice vrši se kao što je objašnjeno u poglavlju *2.4.2 Tipkovnica*.

Kodiranje tipaka je ostvareno na najjednostavniji mogući način. Visoka 4 bita označavaju redak, a preostala niska 4 bita označavaju stupac pritisnute tipke. Takav byte s kodom tipke dodaje se u izlazni međuspremnik tipkovnice, ali samo ako ta tipka već prije nije bila pritisnuta.

3.2 BIOS

BIOS-ova glavna funkcija je učitavanje i pokretanje operacijskog sustava. Osim toga, BIOS mora omogućiti jednostavan programski pristup sklopovlju. Stoga BIOS omogućuje aplikaciji (ili operacijskom sustavu) pristup nekim njegovim funkcijama za upravljanje sklopovljem.

Prilikom pokretanja aplikacije, BIOS omogućuje i prosljeđivanje parametara aplikaciji kako bi se povećala fleksibilnost, te olakšalo traženje grešaka.

3.2.1 Izbornik

BIOS prilikom pokretanja ispisuje razne podatke o verziji, napajanju, memorijama, pronađenim particijama, itd. Nakon što se pokrene, prikaže izbornik za odabir aplikacije.

Pomoću tog izbornika je moguće odabrati aplikaciju (ili operacijski sustav) kojeg je potrebno pokrenuti. Tipkama gore i dolje je moguće "šetanje" po izborniku, a tipka "OK" služi za konačni odabir.

Podržano je još i uređivanje parametara za odabranu aplikaciju pomoću tipke 'P', te prelazak u minimalističku konzolu pomoću tipke 'Q'. Konzola je objašnjena u poglavlju ispod.

Aplikacije se automatski pretražuju prilikom pokretanja. U svakoj postojećoj particiji traži se datoteka /boot/menu.txt. Ona sadrži popis aplikacija s particije koje je potrebno prikazati u izborniku.

menu.txt, kao što joj sama ekstenzija kaže, je tekstualna datoteka. Zapis za jednu aplikaciju zauzima tri retka i svi zapisi su posloženi jedan iza drugog. Prvi redak označava natpis koji će biti prikazan u izborniku. Drugi redak su parametri koji se prosljeđuju aplikaciji. Treći, tj. zadnji redak sadrži putanju do izvršne datoteke.

Primjer jedne takve datoteke bi mogao izgledati ovako:

```
Start IHX from serial

/boot/serun.elf
Hello
neki argumenti
/boot/hello.elf
```

Cijeli koncept izbornika, uključujući i parametriranje aplikacija temeljen je na programu GRUB[18]. GRUB je program za pokretanje operacijskih sustava, najčešće na PC računalima. Koristi se u slučaju kad se na računalu nalazi više operacijskih sustava.

Za razliku od PC računala kod kojih BIOS pokreće automatski samo ono što se nalazi na specijalnoj lokaciji na točno određenom disku, mini386 BIOS ima ugrađen sustav za odabir aplikacija, te automatsko pronalaženje mogućih odabira.

Trenutno nije podržan automatski odabir (odbrojavanje do pokretanja aplikacije), niti pamćenje zadnjeg odabira. Za pamćenje zadnjeg odabira moguće bi bilo koristiti memoriju iz sata pošto zapis po FAT32 datotečnom sustavu nije podržan.

18 GNU GRand Unified Bootloader

3.2.2 Konzola

Konzola omogućuje razne funkcije koje nisu često potrebne, a niti predviđene za običnog korisnika.

Iz početnog grafičkog izbornika, moguće je preći u konzolu pritiskom na tipku 'Q'.

Podržane su slijedeće komande:

- **mem** – prikazuje koliko memorije BIOS zauzima (po toj memoriji aplikacije ne smiju ništa pisati)
- **disks** – prikazuje pronađene diskove, te njihove particije
- **vcc** – prikazuje napon napajanja
- **clock** – prikazuje trenutno vrijeme, te omogućuje izmjenu istog
- **x** – pokretanje aplikacije komandnolinijski (s parametrima zadanim komandama "a" i "f")
- **a *<tekst>*** – postavi <*tekst*> kao argument
- **f *<disk> <particija> <putanja>*** – postavi datoteku s diska <*disk*>, particije <*particija*> i lokacije <*putanja*> kao izvršnu datoteku (parametri <*disk*> i <*particija*> su redni broj koji počinje od 0)
- **quit** – izlazak iz konzole, tj. povratak na grafički izbornik

3.2.3 Komunikacija s BIOS-aplikacijama

BIOS pojednostavljuje izradu aplikacija tako što im omogućuje pristup svojim funkcijama za upravljanje sklopovljem. Komunikacija aplikacija s BIOS-om izvedena je pomoću Intelovih[7] vrata za pozive (engl. *call gate*). Vrata za pozive omogućuju korisničkim aplikacijama poziv jezgrinih funkcija operacijskog sustava pomoću posebne instrukcije (`lcall` <*broj*>`, 0`). Rade na istom principu kao i programski prekidi (`int` <*broj*>), samo što ne onemogućuju prekide prilikom ulaska u jezgrinu funkciju.

BIOS-ove funkcije koje komuniciraju s mikrokontrolerom pretpostavljaju da su prekidi omogućeni zbog toga što se prekidi koriste prilikom čekanja mikrokontrolerove potvrde.

Zahvaljujući vratima za pozive, aplikacija prilikom poziva BIOS-ovih funkcija ne mora znati na kojoj se memorijskoj lokaciji nalazi svaka pojedina funkcija. Adrese tih funkcija se mijenjaju prilikom izmjene BIOS-a, te bi bez vrata za pozive bilo potrebno prevesti svaku aplikaciju.

Podrška za poziv jezgrine funkcije iz manje privilegiranog načina rada procesora se ne koristi, ali ju je moguće koristiti u slučaju da aplikacija iz nekog razloga odbaci svoje privilegije.

Parametri se prosljeđuju kroz registre `ebx`, `ecx` i `edx`. Registar `eax` koristi se kao identifikacijski broj funkcije koju se poziva jer poziv kroz vrata može pozvati samo jednu funkciju. Definicija više vrata za pozive nije praktična. Registar `eax` se također koristi i za prosljeđivanje povratne vrijednosti aplikaciji. Identični princip koristi CleanOS prilikom poziva jezgrinih funkcija.

Kako se koriste samo tri registra za parametre, kod nekih funkcija je bilo potrebno izmijeniti način slanja parametara. Primjerice, funkcija za crtanje linije ima četiri parametra – po dva za svaku koordinatu. Kako bi ju bilo moguće pozvati kroz vrata za poziv, po dvije koordinate su smještene unutar jednog parametra. Visokih 16 bita je jedna koordinata, a niskih 16 druga. Ograničenje od 16 bita po koordinati nije problem zbog toga što za dimenzije prikaznika ne treba više od 7 bita.

Za većinu BIOS-ovih funkcija napisane su funkcije u C-u koje je zatim moguće koristiti tokom programiranja aplikacija. Funkcije sadrže asemblerski kod za poziv kroz vrata. "`static inline`" su tipa što omogućuje da se smjeste u datoteke za zaglavlje, a prevoditelj njihov izvršni kôd smjesti tamo gdje su pozvane i na taj način izbaci potrebu za instrukcijama za poziv funkcije.

U nastavku je primjer funkcije za poziv BIOS-ove funkcije `at_read_reg`:

```
static inline unsigned char at_read_reg(atcmdid_t cmdid) {
    long r;
    __asm__ __volatile__("lcall $0x18, $0\n\t" : "=a"(r) : "0"(52), "b"(cmdid));
    return (unsigned char)r;
}
```

Kako se nalazi mnogo takvih kratkih funkcija s gotovo identičnim sadržajem, napravljene su pretprocesorske funkcije kojima je samo potrebno zadati parametar s povratnom vrijednosti, te sa svim ostalim parametrima funkcije.

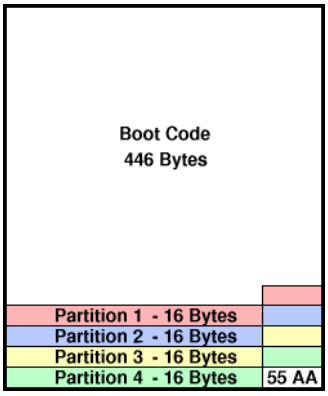
3.2.4 FAT32 i datotečni podsustav

Trenutno BIOS podržava samo FAT32[8] datotečni sustav. FAT32 je odabran zato što se najčešće koristi na prijenosnim *flash* memorijama poput USB *stickova*, MP3 *playera* i memorijskih kartica poput podržane microSD kartice. Popularnosti FAT32 datotečnog sustava uvelike pridonosi njegova jednostavnost. Isti princip koristio se na računalima prije 1980-e (FAT12). Ime je dobio po tablici koja sadrži informacije o zauzetosti svakog pojedinog segmenta datotečnog sustava (engl. *file allocation table – FAT*).

Osim podrške za sam datotečni sustav, dodana je podrška i za podjelu diska na particije. Podržane su samo primarne particije kojih može biti samo četiri. Format zapisa particija na disk također datira iz 1980-e, te je čak i popularniji od samog FAT32 datotečnog sustava.

Disk je virtualno podijeljen na sektore, tj. blokove veličine 512 bytea. Takva podjela je bila pogodna za tvrde diskove gdje je cilindar (kružni zapis na ploči diska) podijeljen na sektore iste veličine. Današnje *flash* memorije su podijeljene na puno veće sektore, te je prilikom izmjene samo jednog sektora potrebno ponovno zapisivati sve ostale.

Slika 33 prikazuje prvi sektor diska koji sadrži tablicu particija. Prvih 446 bytea je rezervirano za izvršni kod kojeg BIOS na PC-u pokreće nakon učitavanja. Kod prijenosnih *flash* memorija taj dio se rijetko kad koristi. Na samom kraju se nalaze 2 bytea oznake koja kaže da je zapis particija u FAT formatu. Između se nalaze četiri bloka od 16 bytea. Svaki blok opisuje pojedinu particiju diska i uređen je kao na *slici 34*.



Slika 33: Zapis particija

Slika 34: Format zapisa jedne particije

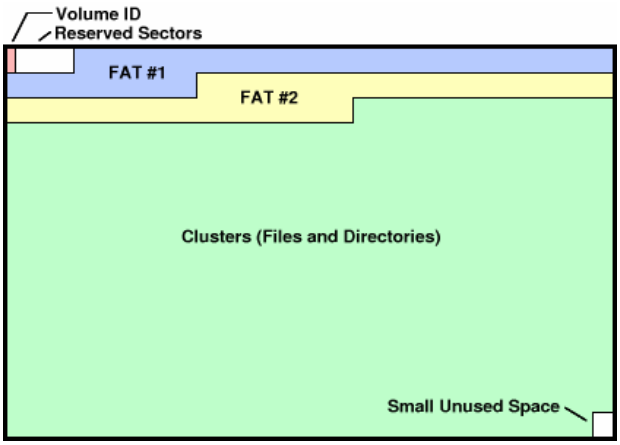
Za pronalaženje tražene particije potrebno je samo pročitati "LBA Begin" što označava odmak od početka diska gdje počinje particija, te "Number of Sectors" što označava od koliko sektora se sastoji particija, tj. koliko je velika particija. Opcionalno je korisno pročitati "Type Code" koji označava koji datotečni sustav se nalazi na toj particiji.

Za pristup particijama napravljene su zasebne funkcije koje uzimaju u obzir odmak od početka, te provjeravaju da li pristup particiji ne prelazi njen kraj. Čitanje podataka s diska se zatim prepušta funkciji za pristup mediju (različite su funkcije za pristup virtualnom disku i za pristup microSD kartici).

Tokom pretrage za postojećim particijama po oba diska (ugrađeni virtualni disk i microSD kartica) potrebno je i provjeriti da li se na njoj nalazi podržani datotečni sustav. Već prije je spomenuto da je podržan samo FAT32 datotečni sustav, ali moguće je vrlo jednostavno dodati podršku i za druge datotečne sustave. Svaki je opisan pomoću strukture `fs_t` koja sadrži naziv datotečnog sustava, te pokazivače na funkcije za inicijalizaciju, čitanje mapa, te čitanje datoteka.

Prvi sektor FAT32 datotečnog sustava sadrži zaglavlje s raznim informacijama o datotečnom sustavu. Neke od tih informacija omogućuju potvrdu da se stvarno radi o FAT32 datotečnom sustavu.

FAT32 je podijeljen na grupe podataka (engl. *clusters*). Clusteri se sastoje od više sektora, ovisno o tome koliko je zapisano u zaglavlju. Kod tvrdih diskova, veličina clustera odgovara veličini cilindra, te je na taj način efikasniji pristup. Također, prilikom ugradnje FAT32 datotečnog sustava u *flash* memoriju, poželjno je odrediti veličinu clustera tako da se poklapa s veličinom segmenta *flash* memorije.



Slika 35: Struktura FAT32 datotečnog sustava

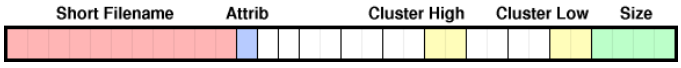
Slika 35 prikazuje organizaciju glavnih dijelova FAT32 datotečnog sustava unutar particije. Na početku se nalazi zaglavlje iza kojeg slijedi prazan prostor, FA-tablice, te prostor za podate (datoteke i mape). Lokacije svih glavnih dijelova zapisane su u zaglavlju.

FA-tablice služe kao opis povezanosti clustera. Sve tablice su iste kako bi prilikom oštećenja jedne bilo i dalje moguće pristupiti podacima. Moguće je koristiti i više od dvije FA-tablice. mini386 BIOS trenutno podržava isključivo pristup prvoj tablici, te nije definirano što će se dogoditi u slučaju oštećenja prve FA-tablice. Podrška za FAT32 datotečni sustav napravljena je da provjerava smislenost ključnih podataka, tako da se sustav u konačnici ne bi "zamrznuo". Primjerice, može se dogoditi da je moguće pristupiti samo početku datoteke kao da je manja no što bi trebala biti, ali neće doći do obavijesti o tome da nije pročitana cijela datoteka.

FA-tablica sadrži onoliko elemenata koliko ima clustera na particiji. Svaki element sadrži redni broj slijedećeg clustera. Ukoliko slijedeći cluster ne postoji (npr. kraj datoteke), element sadrži broj veći ili jednak $FFFFFFF0_{16}$, a ukoliko je segment slobodan sadrži broj 00000000_{16}.

Sadržaj datoteka i mapa na početku datotečnog sustava nalazi se u početnoj mapi. Prvi cluster početne mape zapisan je u zaglavlju. Format clustera s početnom mapom identičan je formatu svih clustera za opis mape.

Cluster za opis mape sastoji se od niza segmenata dugačkih 32 bytea. Format takvog segmenta opisan je na *slici 36*:



Slika 36: Zapis elementa mape

- "Short Filename" sadrži naziv datoteke ili mape. Dopušteni nazivi za datoteke su u formatu 8.3, tj. 8 znaka za naziv, te 3 znaka za ekstenziju. Nije podržano proširenje za duge nazive datoteka i mapa (LFN proširenje).

- "Attrib" označava o kojem tipu zapisa se radi (datoteka ili mapa) te koje su postavljene zastavice (moguće je samo čitanje, sakriven podatak, sistemski podatak, slobodan zapis...).

  Ako je označeno da se radi o slobodnom zapisu, znači da je mapa ili datoteka bila obrisana. Prilikom dodavanja nove datoteke ili mape, moguće je iskoristiti ovaj zapis.
- "Cluster High" i "Cluster Low" kad se zajedno kombiniraju daju prvi cluster datoteke ili mape.
- "Size" određuje veličinu datoteke.

Funkcije za pristup datotečnom sustavu napravljene su tako da nije bitno o kojem datotečnom sustavu se radi, te nije potrebno dinamički rezervirati memoriju. Svi podaci nalaze se na stogu. Zahvaljujući tome nije potrebno zatvarati datoteke nakon pristupa.

Svi podsustavi su odvojeni. Pristup datotekama je univerzalan, te ga je moguće vrlo jednostavno koristiti. No da bi funkcionirao, koristi pristup datotečnom sustavu koji omogućuje pristup datotekama i mapama specifičan za svoj format. Pristup datotečnom sustavu koristi funkciju za pristup particiji, a ta funkcija u konačnici koristi funkciju za pristup mediju. Na taj način moguće je podržati više datotečnih sustava, te više tipova medija.

U nastavku je primjer čitanja datoteke "/boot/menu.txt" znak po znak. DID i PID predstavljaju konstante za odabir diska i particije na odabranom disku:

```
void ispisi_menu() {
    file_t dat;
    char znak;

    if (fs_open(&dat, &disk[DID].partition[PID], "/boot/menu.txt")) {
        puts("Greska prilikom otvaranja datoteke!\n");
        return;
    }

    while (fs_read(&znak, 1, &dat) == 1)
        putc(znak);
}
```

3.2.5 BIOS-aplikacije

BIOS-aplikacije su aplikacije koje odabire korisnik nakon što se BIOS učita. Sklopovlju mogu pristupati direktno, ali i preko BIOS-ovih funkcija. BIOS-ove funkcije uvelike olakšavaju primjerice pristup prikazniku zbog toga što imaju ugrađenu podršku za komunikaciju s mikrokontrolerom. Više o komunikaciji opisano je u poglavlju *3.2.3 Komunikacija s BIOS-aplikacijama*.

Zamišljeno je da postoje dvije vrste BIOS-aplikacija:

1. male aplikacije poput aplikacije za prijem aplikacija preko serijskog sučelja, te poput jednostavnih igrica,
2. operacijski sustav.

Prilikom pokretanja aplikacije, BIOS prosljeđuje argumente aplikaciji. Argumenti mogu biti maksimalne duljine od 31-og znaka, te su smješteni unutar BIOS-ove globalne memorije. Kako bi aplikacije mogle biti operacijski sustav, BIOS ostavlja sve privilegije aplikaciji. Stoga aplikacije ne smiju pisati po početku memorije. Također, nemoguće je učitati aplikaciju koja zahtjeva da se neki njen segment nalazi preko BIOS-ove globalne memorije.

Trenutno je podržan samo ELF zapis izvršne datoteke. Moguće je u nastavku razvoja podržati još `ihx`, te PE[19] formate kako bi se proširio izbor alata za izradnju BIOS-aplikacija.

Primjer jednostavne BIOS-aplikacije izgleda ovako:

```
#include <bios_term.h>

int _start(const char *args) {
    puts("Zdravo svijete! :)\n");
    puts("Args: ");
    puts(args);
    puts("\n");
    return 0;
}
```

19 Portable Executable – Microsoftov format za izvršne datoteke

3.3 Razvojni alati

Prilikom razvoja mini386, korišteni su mnogi razvojni alati. Neke je čak bilo potrebno izraditi zbog svoje specifične potrebe.

3.3.1 SDCC

Atfir je preveden i povezan pomoću programskog paketa SDCC (engl. *small device C compiler*). SDCC je besplatni prevodioc za procesore poput 8051, Z80, PIC, ... Sintaksno je kompatibilan s Keilovim programskim alatom koji se koristi za profesionalan razvoj 8051 sustava.

3.3.2 GCC

BIOS je preveden pomoću programskog paketa GCC (engl. *GNU compiler collection*). GCC je također besplatan, ali ujedno i jedan od najpopularnijih prevodioca. U početku je bio zamišljen kao prevodioc za GNU operacijski sustav, ali danas se koristi kao prevodioc za mnoge operacijske sustave i mnoge arhitekture. GCC omogućuje vrlo detaljnu konfiguraciju, te ima specifične dodatke sintaksi programskog jezika C.

GCC bez parametara generira izvršnu datoteku za operacijski sustav na kojem se nalazi. Zbog toga je potrebno prilikom prevođenja GCC-u zadati parametre da ne ubacuje standardne biblioteke u izvršnu datoteku. Također, nije poželjan niti podrazumijevani izlazni format datoteke ELF. BIOS mora biti čisti izvršni kod, te se neki dijelovi moraju točno pozicionirati unutar ROM-a. Skripta za povezivanje određuje gdje treba svaki segment izvršne datoteke pozicionirati u ROM-u, te određuje da je izlaz binarni, bez formatiranja.

Dodatno je uključena optimizacija za 386 procesor, kao i uobičajena optimizacija prilikom prevođenja. Isključena je zaštita stoga jer nije nužna, te zahtjeva dodatnu biblioteku ili implementaciju funkcija za zaštitu stoga. Također, zaštita stoga dodatno usporava izvođenje.

Za automatizirano prevođenje koriste se `Makefile` skripte.

3.3.3 objcopy

Alat `objcopy` omogućuje izmjenu međukoda (engl. *object code*). Pomoću njega je u međukod datoteke sa funkcijama za virtualni disk naknadno ugrađena datoteka koja sadrži virtualni disk.

`objcopy` je moguće koristiti i za pretvorbu izvršnih datoteka iz ELF formata u `ihx`.

3.3.4 rommix

Nakon povezivanja i generiranja binarnog zapisa, pokreće se program `rommix` koji prilagodi generirani binarni zapis za zapis u ROM. `rommix` je detaljnije opisan u poglavlju *2.3.1 ROM*.

3.3.5 trom

Prilikom testiranja potrebno je poslati izvršni kod u ROM emulator. Korišten je veoma star emulator čiji originalni program za slanje podataka radi isključivo pod DOS okruženjem. Kako bi se zaobišla uporaba emulatora, te omogućilo jednostavno i brzo automatizirano slanje izvršnog koda u emulator nakon prevođenja, napisana je suvremena verzija programa za slanje podataka u ROM emulator.

Osim toga, osuvremenjeno je i sučelje ROM emulatora kako bi ga bilo moguće spojiti na USB priključak.

3.3.6 mkfs.vfat

Pomoću alata `mkfs.vfat` kreira se prazan virtualni disk s FAT32 datotečnim sustavom. Kreiranje virtualnog diska veoma je optimirano kako bi što više mjesta bilo za korisničke podatke. Primjerice, pretpostavljena vrijednost za broj tablica koje opisuju zauzeće je 2 što je nepotrebno kad se datotečni sustav nalazi u ROM-u gdje se ne može dogoditi da se neka od tablica ošteti.

`mkfs.vfat` kreira disk unutar postojeće datoteke, te je stoga prije kreiranja virtualnog diska potrebno kreirati praznu datoteku željene veličine pomoću alata `dd`. Minimalna veličina za FAT32 datotečni sustav je oko 64kB.

3.3.7 mountromdisk.sh

`mountromdisk.sh` je ljuskina skripta koja omogućuje jednostavno prijavljivanje virtualnog diska kao prave mape datotečnom sustavu. U slučaju da ne postoji datoteka s virtualnim diskom, automatski generira novu koja sadrži prazni virtualni disk.

Nakon što automatizirano prevođenje pozove tu skriptu, kopira sve što se treba nalaziti na virtualnom disku u mapu. Kopiranjem datoteka u mapu automatski se mijenja datoteka s virtualnim diskom. Nakon što je sve kopirano, odjavi se virtualni disk, te obnovi datum datoteke s virtualnim diskom. Obnavljanje datuma je potrebno kako bi automatizirano prevođenje znalo da je generiran novi virtualni disk, te da ga nije potrebno ponovno generirati ako se ništa drugo nije promijenilo, a izmjena mape s virtualnim diskom ne mijenja datum datoteke s virtualnim diskom.

3.3.8 sendihx.sh

`sendihx.sh` je ljuskina skripta za slanje `ihx`[9] (Intel hex) datoteka preko serijskog sučelja. Posebna BIOS-aplikacija služi za prijem te datoteke, te njeno izvršavanje nakon prijema.

`ihx` je vrlo jednostavan tekstualan format namijenjen baš za ovakve slučajeve gdje je na neki način potrebno slati izvršni kod u uređaj. Podaci su organizirani po redovima. Svaki redak počinje sa znakom ':' iza kojeg slijedi niz parova heksadecimalnih znamenaka (jedan par = 1 byte).

Tablica 11: Format retka `ihx` *datoteke:*

Početak retka	Količina podataka (*N*)	Adresa	Tip retka	Podaci	Zaštita
: (0B)	1B	4B	1B	*N*B	1B

Tablica 11 prikazuje format retka `ihx` datoteke, te veličine koje pojedini podatak zauzima. Tip retka može biti:

- 00_{16} – redak sadrži podatke
- 01_{16} – oznaka kraja datoteke (zadnji redak)
- 02_{16} – proširenje adrese pomoću segmenta (svaki redak definira 16-bitnu adresu na koju se zbraja proširenje pomnoženo s 16)
- 04_{16} – proširenje adrese pomoću visokih 16 bita (svaki redak definira niskih 16-bita adrese, a proširenje visokih 16)
- 05_{16} – definira početnu adresu izvršnog koda

Na početku skripta postavi parametre serijskog sučelja. Nakon toga čeka obavijest da je moguće slati podatke, te počinje slanje podataka red po red. Nakon što se pošalje jedan redak, očekuje se odgovor "OK".

3.3.9 fs_test

`fs_test` je program za PC pomoću kojeg je testirana podrška za FAT32 datotečni sustav.

Zahvaljujući tome nije bilo potrebno ograničeno testiranje na samom uređaju.

3.3.10 ihxsender

`ihxsender` je naprednija inačica skripte `sendihx.sh` napisana u programskom jeziku C++. Umjesto slanja originalne ASCII `ihx` datoteke, šalje se sažeta binarna verzija iste. Potvrda je također znatno skraćena i prorijeđena.

Sažeto slanje podataka kroz serijsko sučelje je iznimno bitno zbog toga što slanje izvorne `ihx` datoteke operacijskog sustava CleanOS traje preko jedne minute! To je predugo vrijeme koje uvelike produljuje testiranje. Brzinu prijenosa podataka nije moguće povećati preko 19200 bit/s zbog loše vrijednosti kristala koja onemogućuje dovoljno precizno određivanje viših brzina.

Pretvorba `ihx` datoteke u binarni format daje najveću uštedu od 160%. Zatim se vrši LZ77 metoda sažimanja sa specijalnim načinom zapisa koja uštedi oko 44%.

LZ77[10] kompresija kao rezultat daje odmak i duljinu koji određuju niz byteova u prethodno poslanim podacima koji su identični podacima koje je potrebno poslati, te jedan dodatan byte koji slijedi pronađeni niz. Odmak se broji od trenutne pozicije do koje su poslani podaci. Kao maksimalni odmak i maksimalna duljina niza određen je broj 254 kako bi bio potreban točno jedan byte za svaki podatak.

Pretraga prethodno poslanog niza podataka vrši se pomoću Knuth-Morris-Prattovog[11] algoritma što omogućuje veoma brzu kompresiju.

LZ77 format zapisa izgleda ovako: (odmak, duljina, novi podatak). Pri čemu su sva tri podatka jedan byte. Moguće je primijetiti da zapis nije efikasan ukoliko je pronađeno manje od dva poslana bytea. Stoga se ne šalje uvijek po tri bytea. U slučaju kad nije pronađen niti jedan poslani podatak, šalje se samo jedan byte s novim podatkom. U slučaju kad je pronađen samo jedan poslani podatak, šalju se dva bytea: pronađeni podatak i novi podatak. U svim ostalim slučajevima šalje se cijeli LZ77 zapis.

Da li se šalje jedan ili tri bytea, određuje maska koja se šalje prva. Maska je dugačka 31 bit, pri čemu 1 označava da se šalje jedan byte, a 0 označava da se šalju tri bytea. 32-i bit se koristi kao paritetna zaštita koja se provjerava potvrdom.

Zahvaljujući svim iznad opisanim optimizacijama, slanje izvršne datoteke s CleanOS-om traje nešto više od 10s, što je prihvatljivo.

4 Zaključak

Napravljen je mobilni uređaj temeljen na i386 procesoru. Ugrađen je 1MB RAM, te 512kB *flash* ROM u kojoj se nalazi BIOS i virtualni disk. Za povezivanje logike koristi se CPLD. Sve periferne vanjske jedinice spojene su na SPI sabirnicu kojom upravlja CPLD. Na SPI sabirnicu spojen je i mikrokontroler koji upravlja tipkovnicom, LCD-om, serijskim sučeljem, te A/D konverterom. Atfir (*firmware*) je posebno razvijen kako bi sve to podržavao, te mogao primati komande s procesora.

Dok uređaj nije priključen na vanjski izvor napajanja, koristi se baterija. Nakon što se uređaj priključi na vanjski izvor napajanja, izvor se prebacuje na vanjski, te se započne punjenje baterije. Napajanje i baterija moraju biti "jaki" zbog toga što mini386 u najgorem slučaju može trošiti i do 3W.

BIOS prilikom učitavanja pretražuje aplikacije na virtualnom disku, te na microSD memorijskoj kartici. Za oba diska podržan je samo FAT32 datotečni sustav. Pronađene aplikacije zatim ponudi u grafičkom izborniku. Aplikacije pristupaju sklopovlju pomoću BIOS-ovih funkcija.

Pomoću posebnog seta aplikacija moguće je slati izvršne programe na mini386 preko serijskog sučelja, te ih pokretati. Zahvaljujući tome, moguće je nastaviti prilagođavanje operacijskog sustava CleanOS bez potrebe za ROM emulatorom ili učestalim kopiranjem podataka na microSD karticu. Prilikom slanja, podaci se kompresiraju LZ77 algoritmom kako bi što kraće trajalo.

5 Sažetak

Napravljen je mobilni uređaj upravljan i386 procesorom s 1MB RAM-a, te 512kB *flash* ROM-a. Sustav za napajanje omogućuje rad bez vanjskog izvora napajanja, te punjenje baterije. Za interakciju s korisnikom ugrađen je grafički LCD rezolucije 128×64 piksela, te posebno dizajnirana tipkovnica. Komunikacija s drugim uređajima je moguća preko serijskog sučelja.

BIOS podržava pokretanje aplikacija s virtualnog diska ugrađenog u ROM-u, te s microSD memorijske kartice. Podržan je FAT32 datotečni sustav, a aplikacije su u ELF formatu.

Ključne riječi: mobilni uređaj, i386, 8051, CPLD, microSD, FAT32

6 Summary

A mobile device was made, run by a i386 processor with 1MB of RAM and a flash ROM of 512kB in size. The power system allows the device to work independently of an external power source, using a rechargable battery. User interface consists of a graphical LCD with a resolution of 128×64 pixels, and a specially designed keyboard. A serial interface link enables the communication with oter devices.

The BIOS supports running applications from a virtual drive implemented in the ROM, and from a microSD memory card. The supported filesystem is FAT32, and the executable applications are in the ELF format.

Keywords: mobile device, i386, 8051, CPLD, microSD, FAT32

7 Literatura

[1] Skupina autora. 376™ High Performance 32-bit Embedded Processor: Functional data. Internet: Intel, 1990

[2] Skupina autora, Serial Peripheral Interface Bus, 26.12.2009, *Wikipedia*, http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus, 26.12.2009

[3] Skupina autora, RS-232, 29.12.2009, *Wikipedia*, http://en.wikipedia.org/wiki/Rs232, 29.12.2009

[4] Hearn & Baker, Line-Drawing Algorithms, 16.9.1996, *Raster Graphics*, http://www.cs.unc.edu/~mcmillan/comp136/Lecture6/Lines.html, 3.1.2010

[5] Hearn & Baker, Circle-Drawing Algorithms, 17.9.1996, *Raster Graphics*, http://www.cs.unc.edu/~mcmillan/comp136/Lecture7/circle.html, 3.1.2010

[6] Paul Stoffregen, 128x64 Graphical LCD, 5.3.2008, *PJRC*, http://www.pjrc.com/tech/8051/board5/lcd_128x64.html, 3.1.2010

[7] Skupina autora. Intel® 64 and IA-32 Architectures Software Developer's Manual: Volume 3A - Call Gates. Internet: Intel, 2007

[8] Paul Stoffregen, Understanding FAT32 Filesystems, 24.2.2005, *PJRC*, http://www.pjrc.com/tech/8051/ide/fat32.html, 9.1.2010

[9] Skupina autora, Intel HEX, 8.1.2010, *Wikipedia*, http://en.wikipedia.org/wiki/Intel_hex, 8.1.2010

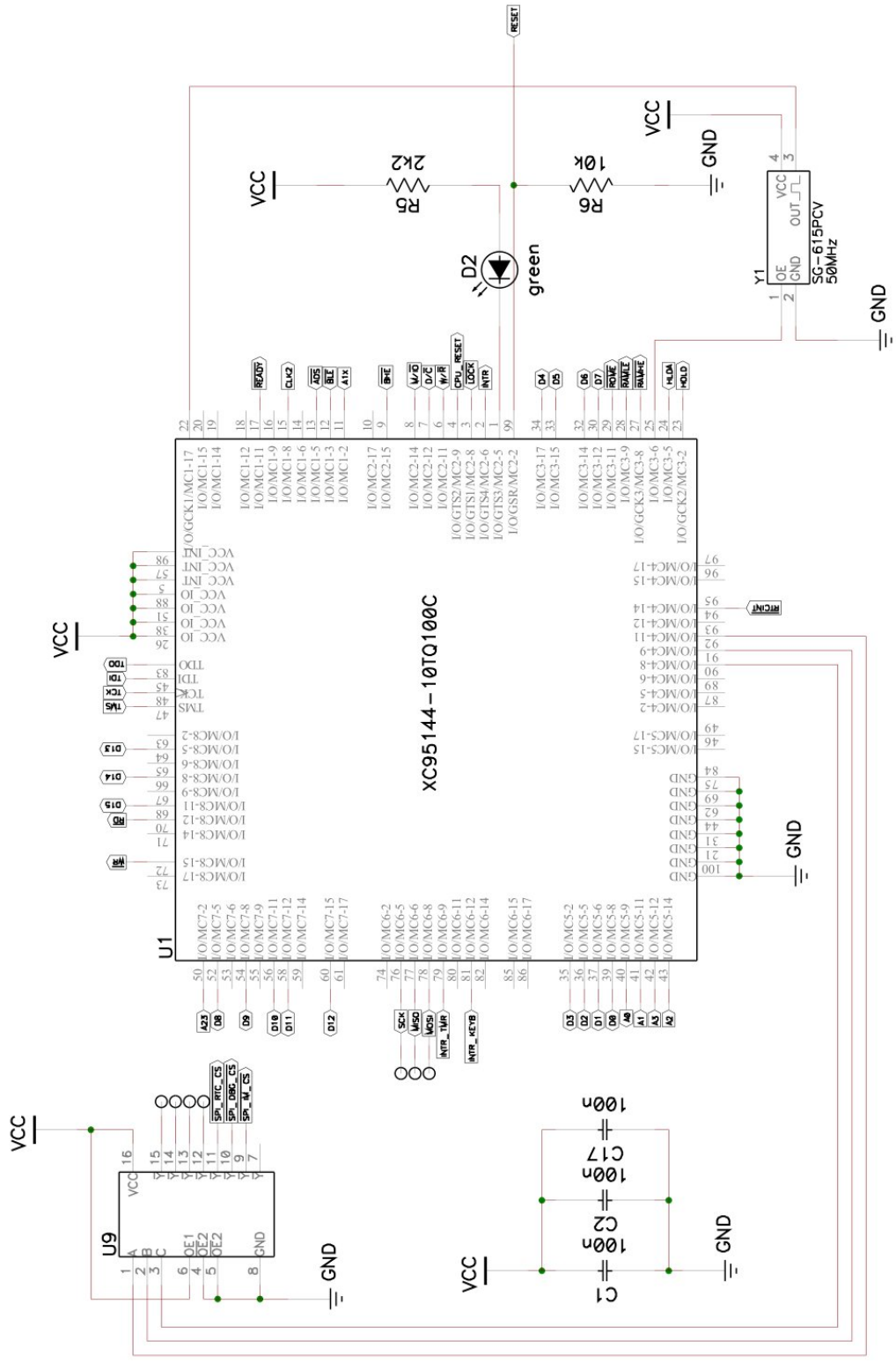
[10] Igor S. Pandžić. Uvod u teoriju informacije i kodiranje: Algoritam LZ77. Zagreb: Element, 2009

[11] Ryan Reich, Knuth–Morris–Pratt algorithm, 10.8.2006, *Wikipedia*, http://en.wikipedia.org/w/index.php?title=Knuth–Morris–Pratt_algorithm&oldid=68814731, 1.9.2010

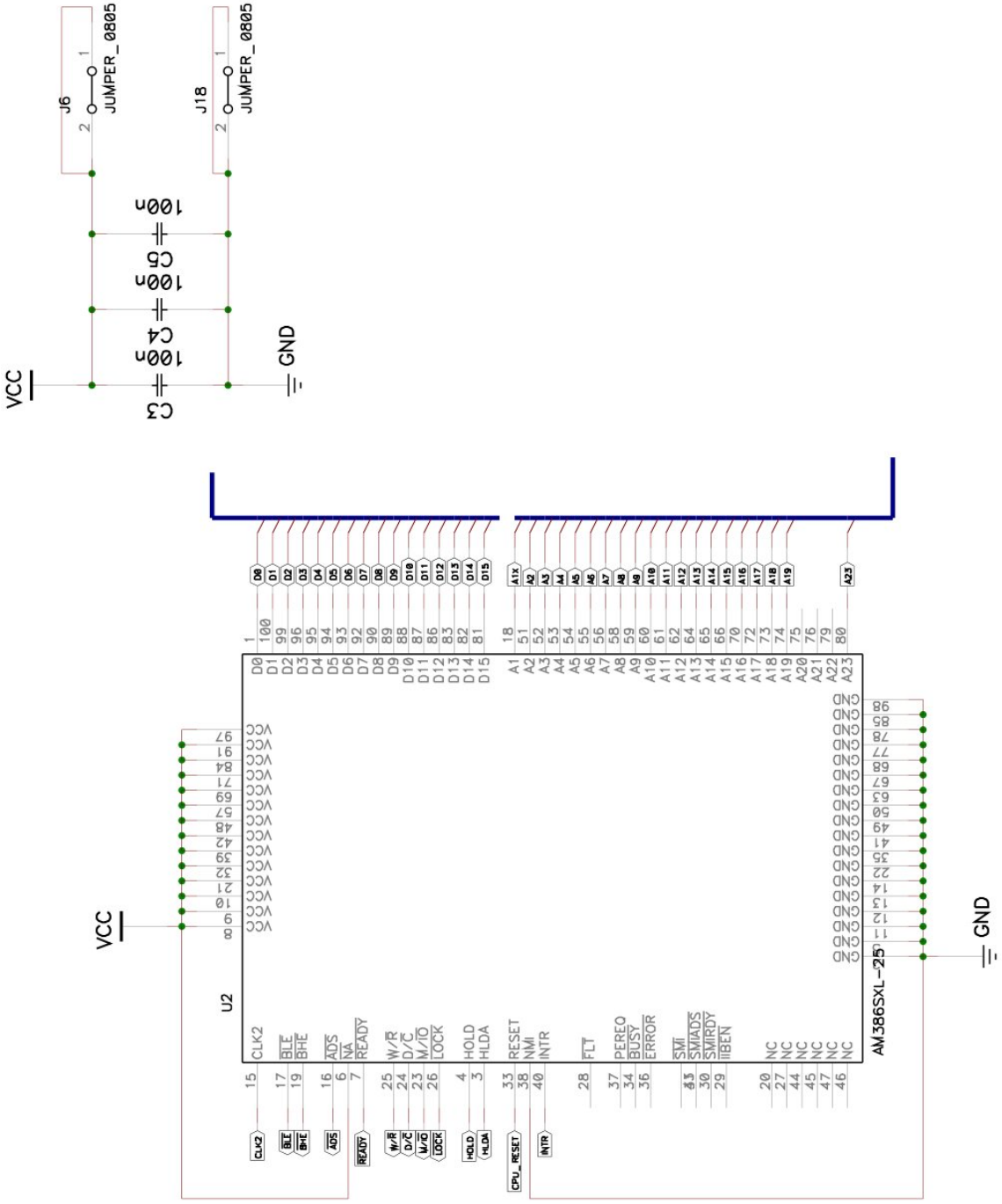
8 Privitak

8.1 Sheme matične pločice

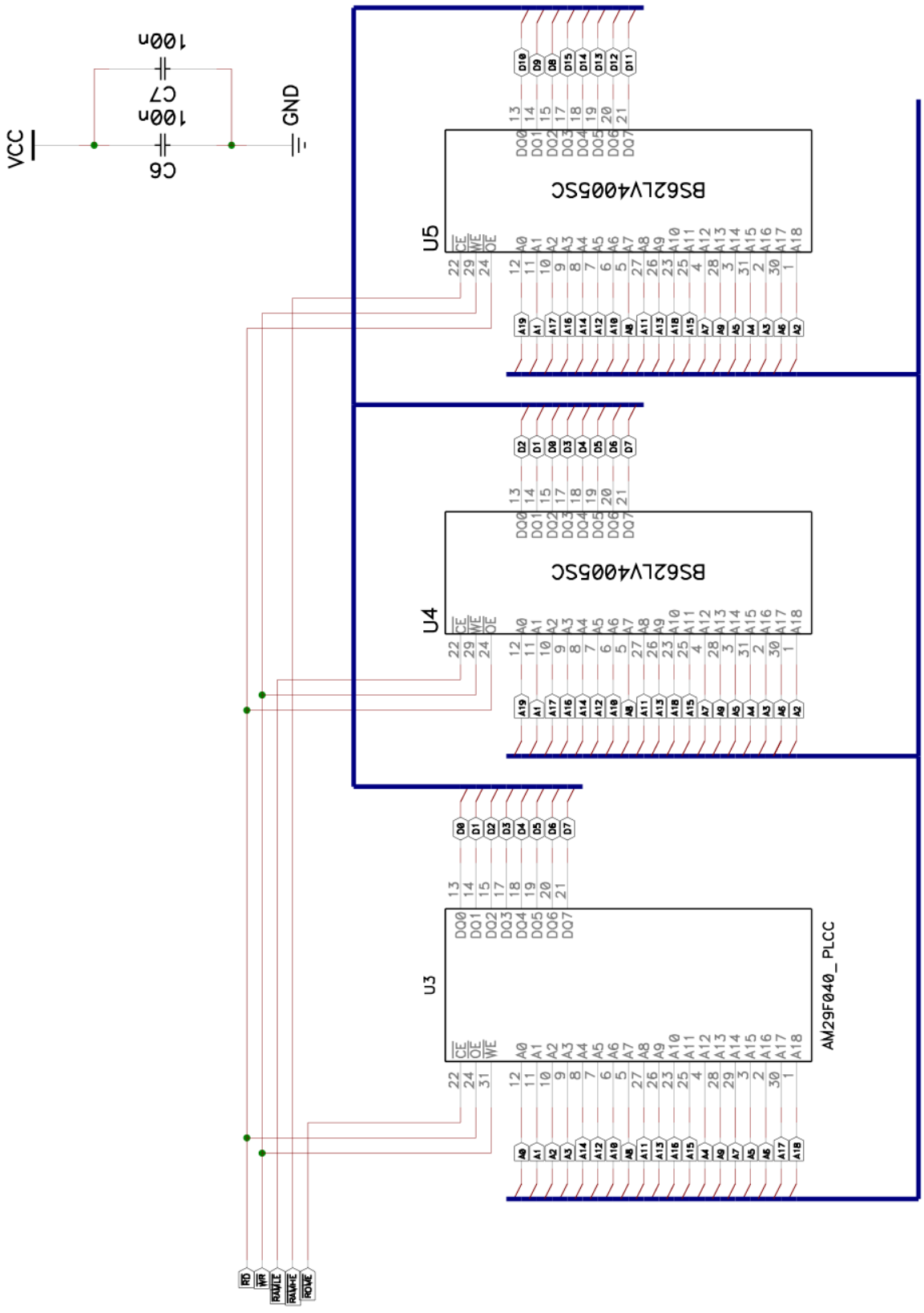
8.1.1 CPLD



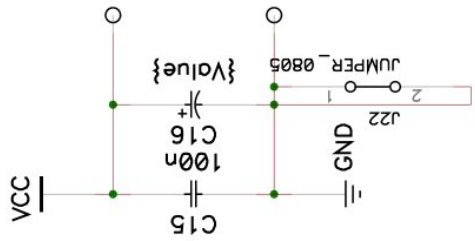
8.1.2 Procesor

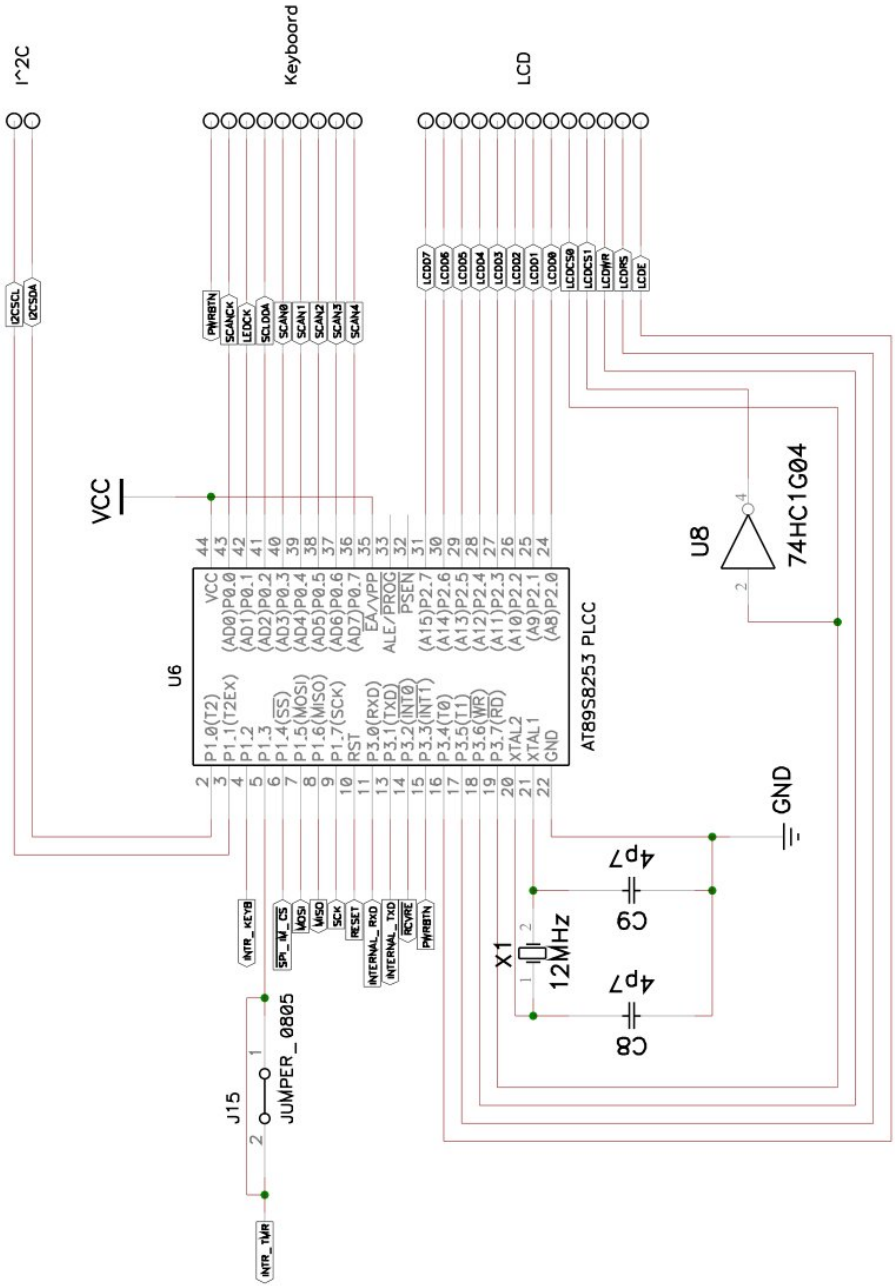


8.1.3 Memorija

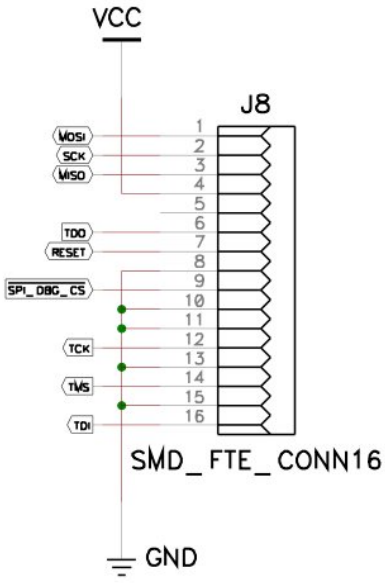


8.1.4 Mikrokontroler

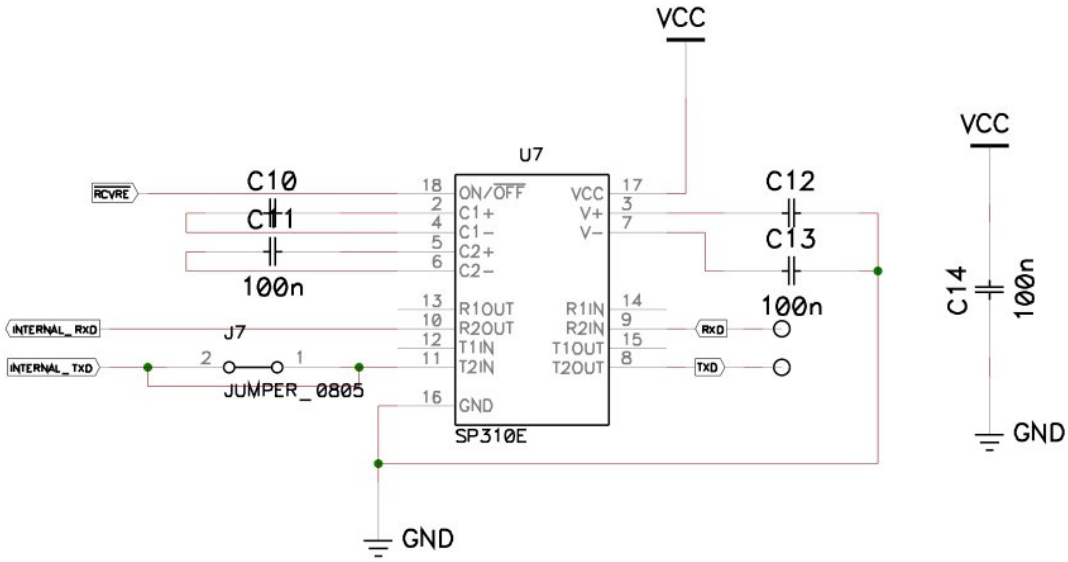




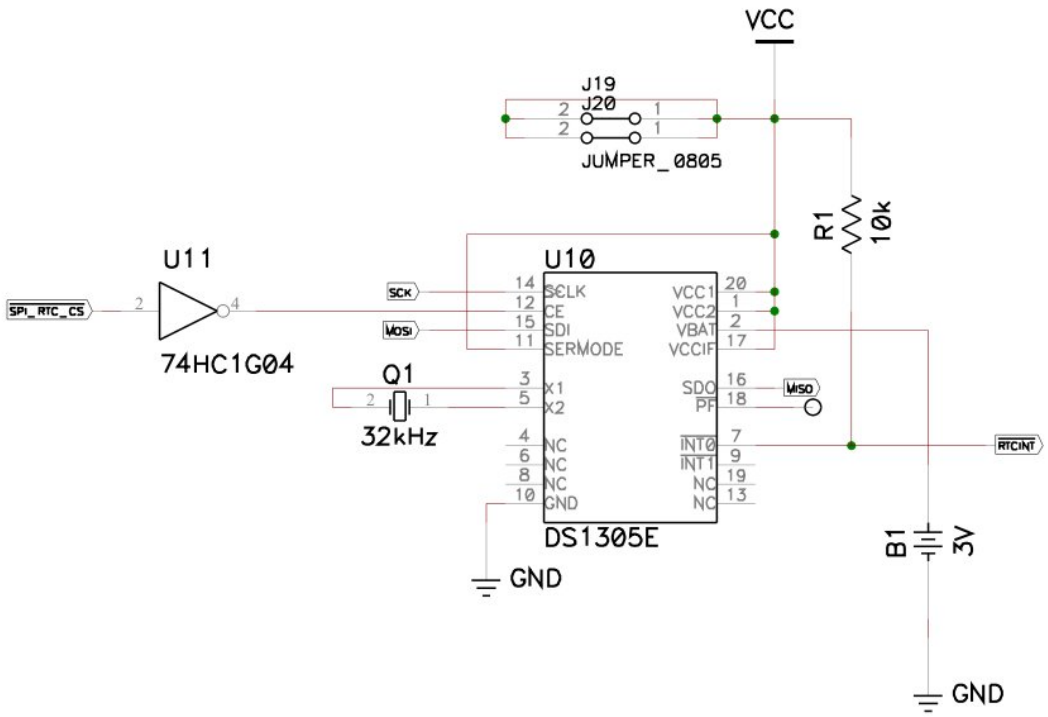
8.1.5 ISP priključak



8.1.6 RS232

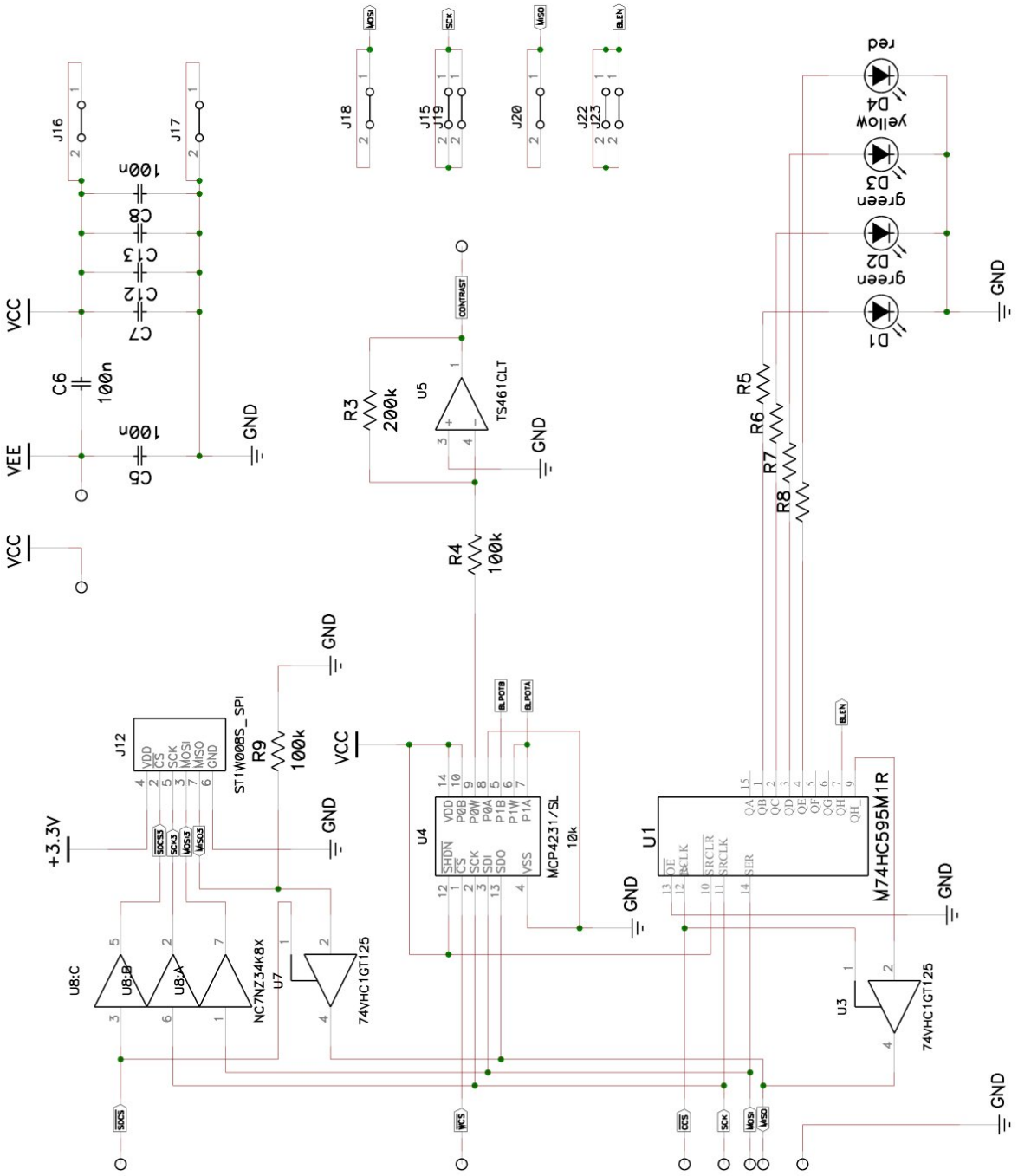


8.1.7 Sat

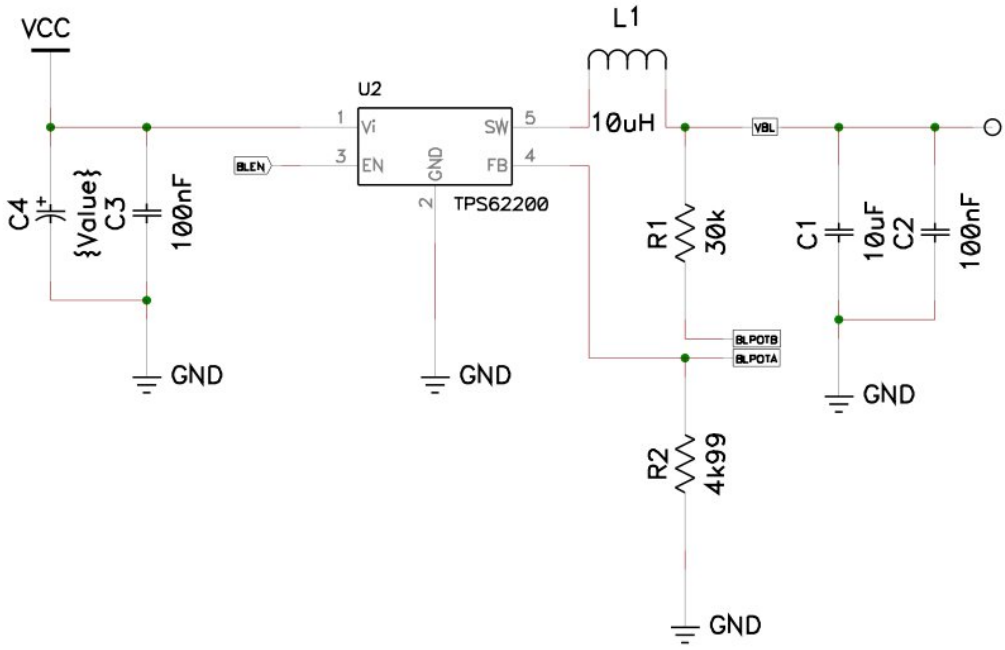

VCC
J19
J20
JUMPER_0805
R1
10k
U11
74HC1G04
SPI_RTC_CS
U10
SCK
MOSI
MISO
SCLK
CE
SDI
SERMODE
VCC1
VCC2
VBAT
VCCIF
SDO
PF
INT0
INT1
NC
GND
DS1305E
Q1
32kHz
RTCINT
B1
3V
GND
GND

8.2 Sheme pločice sa status LED-icama

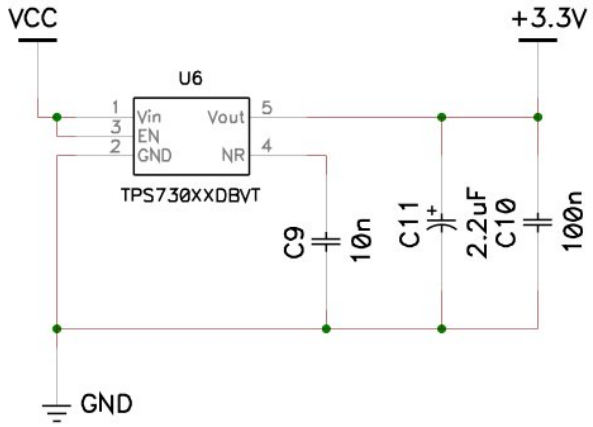
8.2.1 microSD, kontrast, LED-ice



8.2.2 Pozadinsko osvjetljenje

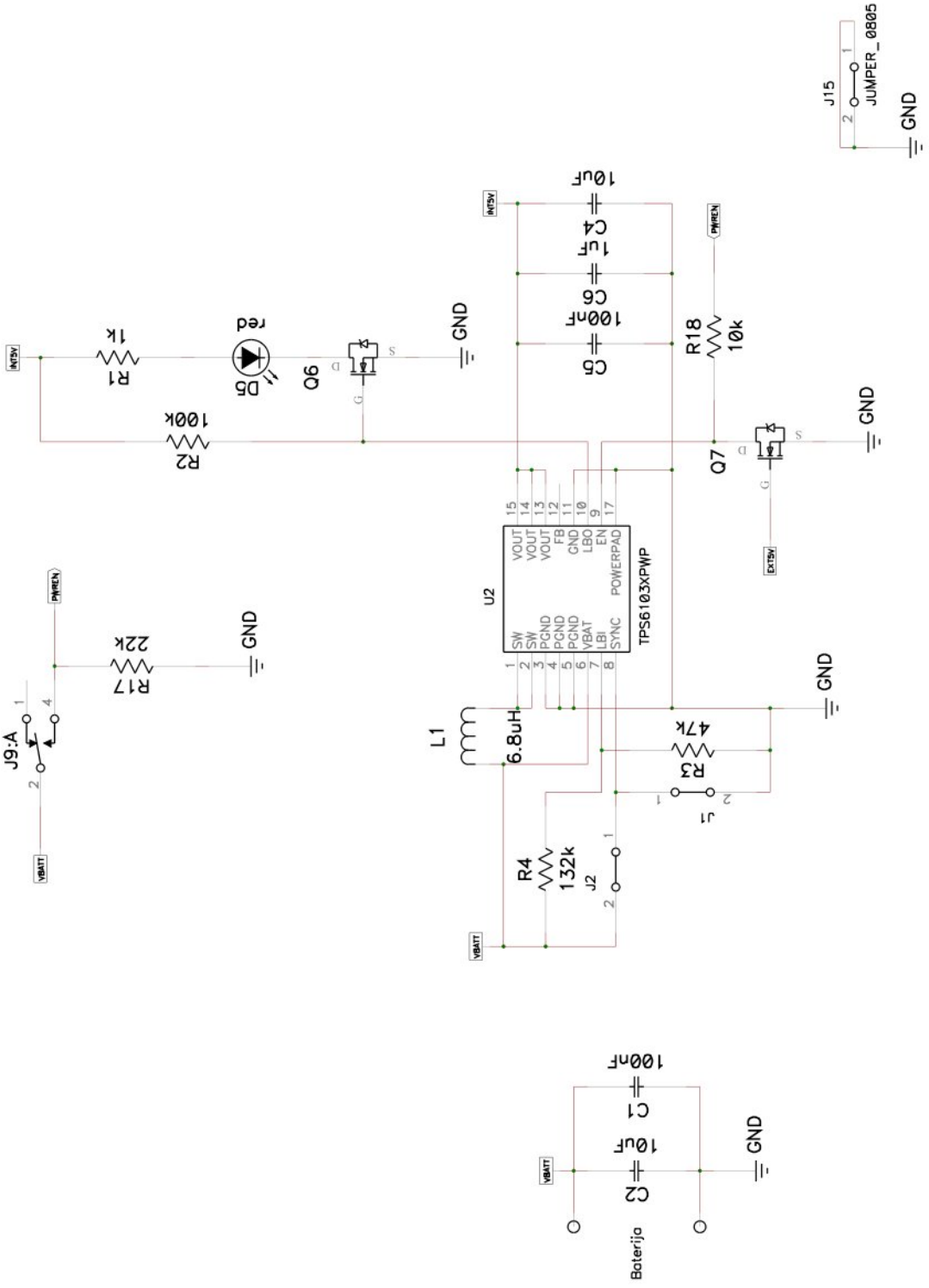


8.2.3 Napajanje za microSD

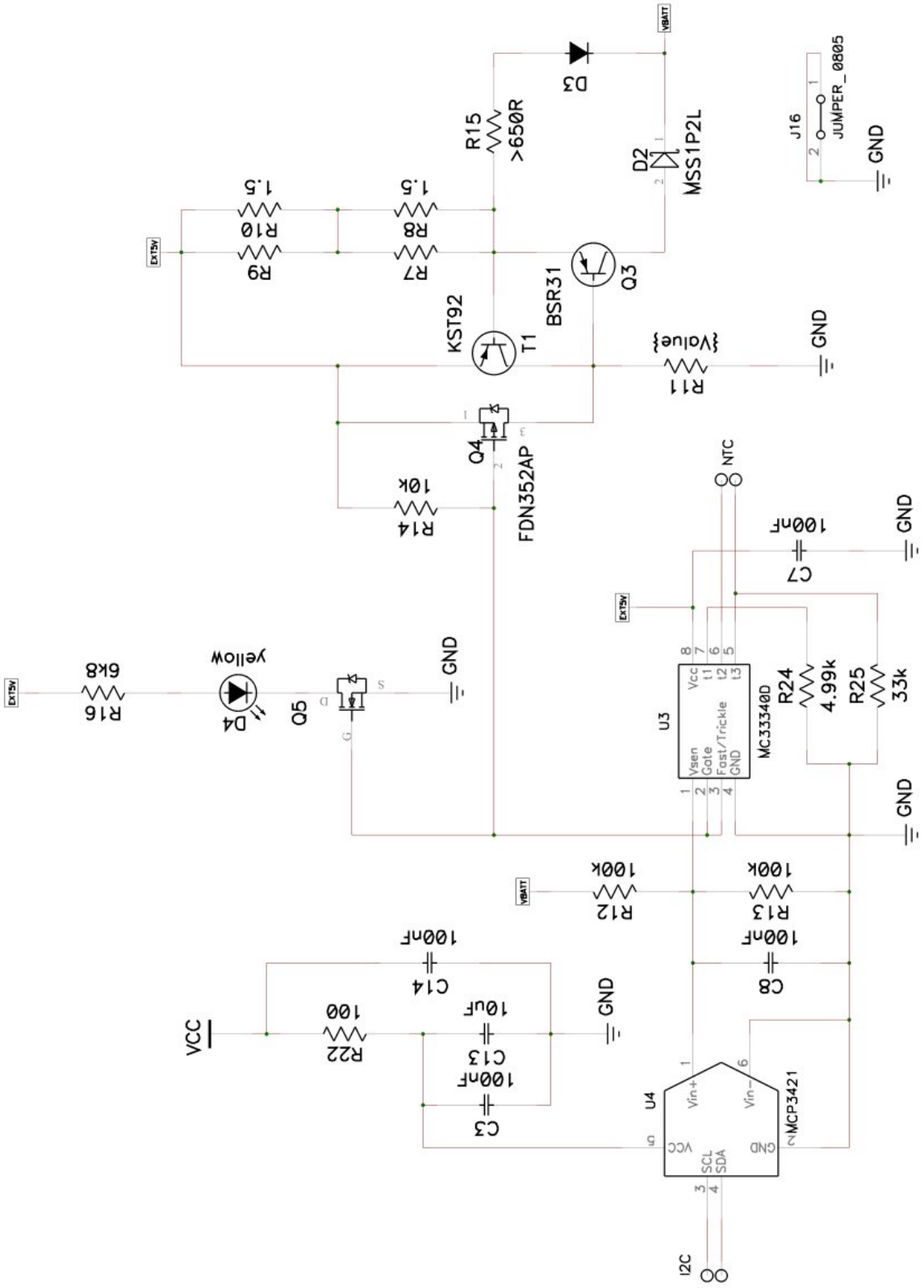


8.3 Sheme pločice za napajanje

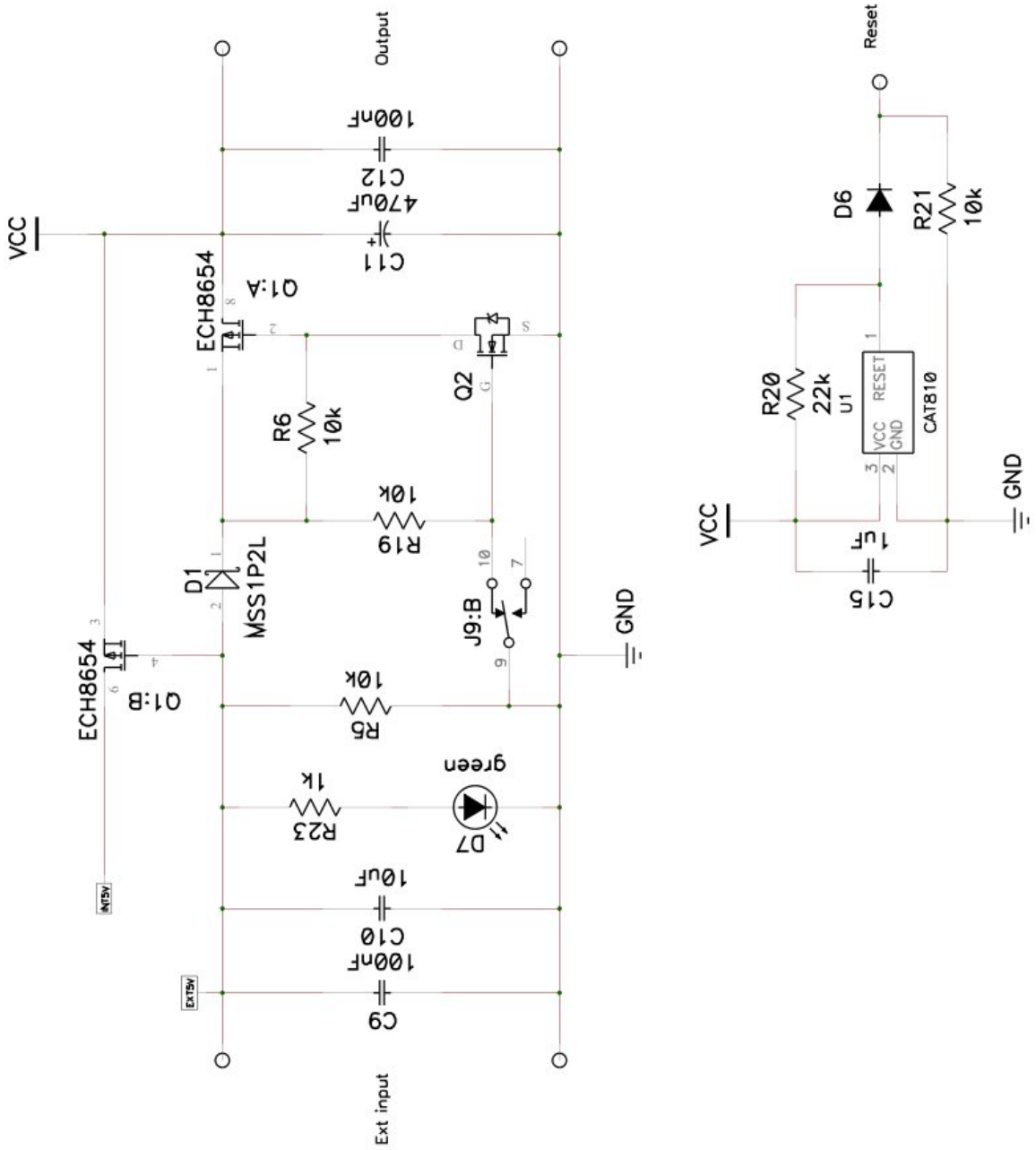
8.3.1 DC/DC konverter



8.3.2 Punjač baterije



8.3.3 Sustav za odabir izvora napajanja



8.4 Shema tipkovnice