SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1696

Ugradbeni računalni sustav za mjerenje kvalitete električne energije

Davor Štefok

Voditelj: *prof. dr. sc. Davor Petrinović*

Zagreb, lipanj, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 9. ožujka 2018.

DIPLOMSKI ZADATAK br. 1696

Pristupnik: **Davor Štefok (0036466160)**
Studij: Elektrotehnika i informacijska tehnologija
Profil: Elektroničko i računalno inženjerstvo

Zadatak: **Ugradbeni računalni sustav za mjerenje kvalitete električne energije**

Opis zadatka:

U okviru diplomskog rada potrebno je razviti programsko i sklopovsko rješenje za mjerenje kvalitete energije trofazne niskonaponske mreže. Sustav mora mjeriti ove veličine po svakoj fazi: struju, napon, radnu i jalovu snagu, radnu i jalovu energiju, faktor snage i faktor totalnog harmoničkog izobličenja. Pri izvedbi potrebno je sagledati sve izvore pogrešaka u mjernom lancu, te istražiti mogućnosti za kalibraciju svih mjerenja. Ostvariti mogućnost prikaza izmjerenih veličina na LCD prikazniku, pohranu u ugrađenu Flash memoriju, te eventualnu mogućnost preuzimanja izmjerenih podataka putem mrežnog sučelja. Posebnu pažnju posvetiti mogućnosti ispravnog rada sustava u stvarnom vremenu.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 29. lipnja 2018.

Mentor:
Prof. dr. sc. Davor Petrinović

Djelovođa:
Prof. dr. sc. Dražen Jurišić

Predsjednik odbora za diplomski rad profila:
Prof. dr. sc. Mladen Vučić

Sadržaj

1 Uvod

U sklopu diplomskog rada potrebno je obraditi osnovne mogućnosti mjerenja električnih parametara trofazne niskonaponske mreže 230/400V, 50 Hz kao što su struja (I), napon (U), radna snaga (P), jalova snaga (Q), prividna snaga (S), faktora snage (cosφ), radne energije (E_P) i jalove energije (E_Q) pomoću mikrokontrolera NXP LPC1768.

Mjerenje električnih parametara mreže potrebno je temeljiti na mjerenju prave efektivne vrijednosti tri struje (IL1, IL2, IL3) i tri napona (UL1, UL2, UL3), te računanju preostalih parametara numeričkom obradom signala.

Iznosi pravih efektivnih vrijednosti struja i napona računaju se nad prikupljenim uzorcima pojedinih faza unutar intervala od jedne sekunde. Unutar intervala prikuplja se 3200 uzoraka nad kojima se vrši numerička obrada.

$$U_{MJ} = \sqrt{\frac{1}{N}\sum_{0}^{3199} u_{MJ}^2(n)} \tag{1.1}$$

$$I_{MJ} = \sqrt{\frac{1}{N}\sum_{0}^{3199} i_{MJ}^2(n)} \tag{1.2}$$

Izrazi 1.1 i 1.2 služe za izračun efektivnih vrijednosti napona i struja, gdje N predstavlja broj uzoraka nad kojima se vrši račun, dok $u_{MJ}^2(n)$ i $i_{MJ}^2(n)$ predstavljaju kvadrirane uzorke pojedinih faza.

Račun snaga nadovezuje se na račun struja i napona.

$$S_{MJ} = U_{MJ} I_{MJ} \tag{1.3}$$

Prividna snaga u intervalu od jedne sekunde je produkt pravih efektivnih vrijednosti struje i napona unutar tog intervala, što pokazuje izraz 1.3.

$$P_{MJ} = \frac{1}{N}\sum_{0}^{3199} u_{MJ}(n) i_{MJ}(n)$$

(1.4)

Radna je snaga, za razliku od prividne, suma produkata uzoraka struje i napona unutar intervala od jedne sekunde, što pokazuje izraz (1.4)

$$Q_{MJ} = \sqrt{S_{MJ}^2 - P_{MJ}^2}$$

(1.5)

Jalovu snagu moguće je nakon izračuna radne i prividne snage izračunati pomoću trokuta snage. Izraz za takav postupak dan je izrazom 1.5.

Faktor snage računa se kao omjer radne i prividne snage, te je za njegovu točnost potrebno posebnu pažnju posvetiti kompenzaciji faznih pomaka struja i napona uslijed slijedne AD pretvorbe, što je jedan od ciljeva rada.

Radne i jalove energije su u suštini akumulacije radnih i jalovih snaga. Cilj je zabilježiti i prikazati svaku promjenu energije veću od Wh (u slučaju radne snage) i VArh (u slučaju jalove energije).

Zbog sve veće pojave viših harmonika u mreži potrebno je analizirati i mogućnosti mjerenja harmonijskog izobličenja (THD) kao i viših harmonika struja i napona (do 31. harmonika).

$$THD = \sqrt{\frac{U_{2rms}^2 + U_{3rms}^2 + \cdots + U_{Nrms}^2}{U_{1rms}^2}}\, x100[\%]$$

(1.6)

Izraz 1.6 predstavlja formulu za izračun THD-a (*Total Harmonic Distortiona*). Iznosi pojedinih harmonika u slučaju konkretne realizacije računaju se pomoću FFT-a (brze Fourierove transformacije).

U radu je potrebno praktično prikazati mjerni sustav koji se sastoji od: ulaznog sklopovlja, analogno-digitalne pretvorbe (uzorkovanje signala), numeričke obrade signala (mjerenja prave efektivne vrijednosti - TRMS, računa snaga i energije, brze Fourierove transformacije - FFT) i prikaza dobivenih veličina pomoću *web* poslužitelja i LCD prikaznika.

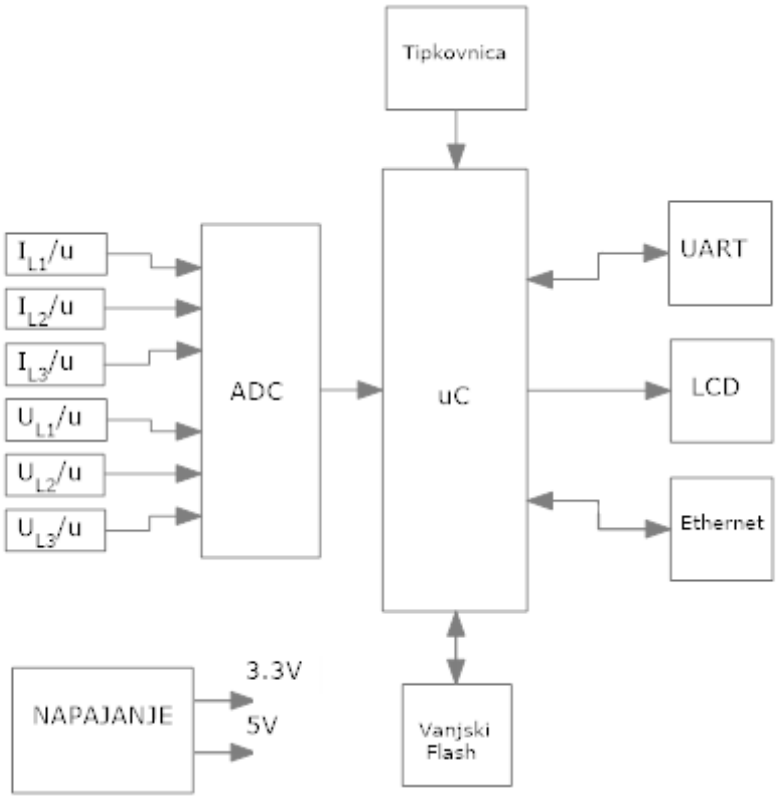
Ovako zamišljen mjerni sustav najčešće se koristi za mjerenje parametara trofazne niskonaponske mreže. Primjena može sezati od ugradnje unutar niskonaponskih razvodnih ormara u trafostanicama do nadzora potrošnje pojedinih uređaja unutar niskonaponske mreže (primjerice automata za kavu).

Uz mjerenje struja, napona, snaga i energija, moguće je mjeriti i THD, koji spada u parametre kvalitete električne energije. Veća posvećenost parametrima kvalitete energije kao rezultat može imati prelazak iz klase monitora osnovnih parametra niskonaponske mreže u klasu monitora kvalitete električne energije.

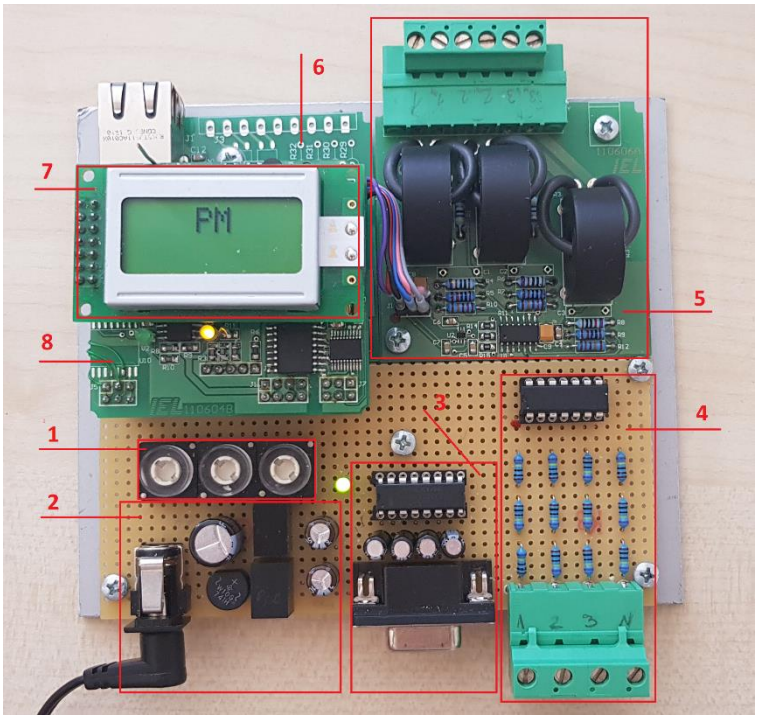
Neki od izraženijih proizvođača koji se bave mjerenjima parametara niskonaponske su Janitza, Landis+Gyr, Iskra, Siemens, PCE-Instruments. Većina proizvođača nudi širok raspon proizvoda kako bi se prilagodili tehničkim zahtjevima i financijskim okvirima projekata. Proizvodi se razlikuju prema broju ulaza, mogućnosti mjerenja pojedinih veličina, preciznosti mjerenja pojedinih veličina, načinu komunikacije s okolinom itd.

Cilj je ostvariti sustav koji zadovoljava osnovne potrebe mjerenja parametara trofazne niskonaponske mreže uz pogreške do 1% na mjerenu vrijednost. Dinamička promjena frekvencije uzorkovanja uzrokovana promjenom frekvencije ulaznih signala nije podržana, već je pretpostavljeno da je frekvencija niskonaponske mreže 50 Hz. Ograničene mogućnosti sustava omogućavaju jeftinije projektiranje istog, što za posljedicu ima potencijalnu konkurentnost među proizvođačima sličnih proizvoda.

2 Sklopovska osnova za realizaciju projekta

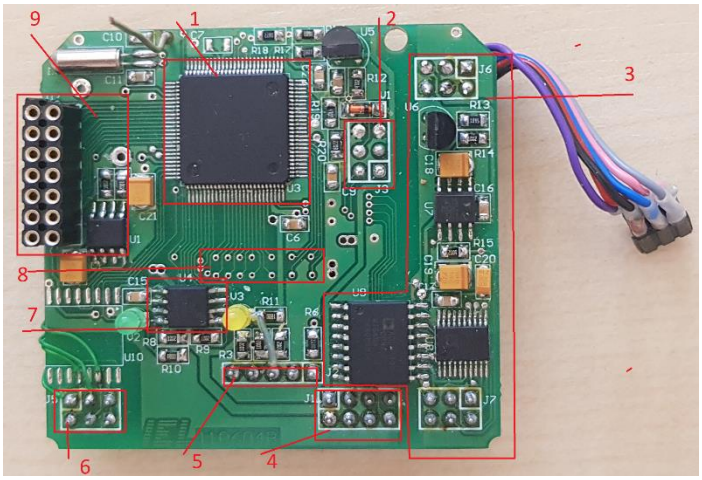


Slika 2.1. Blok shema mjernog sustava



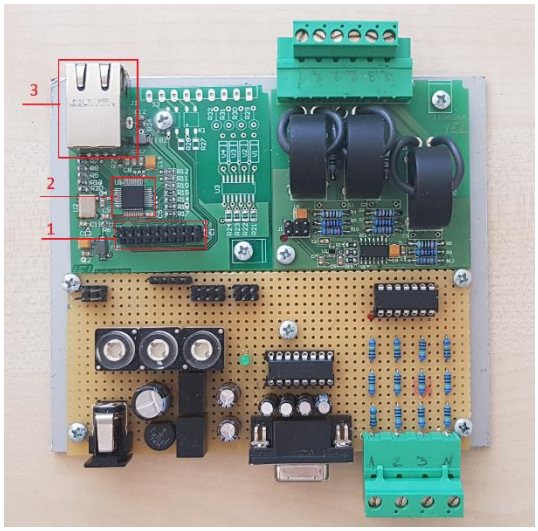
Slika 2.2. Prikaz cjelokupnog sustava

Slika 2.2.: 1) tipkovnica, 2) napajanje, 3) UART, 4) naponski prilagodni sklop, 5) strujni prilagodni sklop, 6) Ethernet pločica, 7) LCD prikaznik, 8) mikrokontrolerska pločica



Slika 2.3. Mikrokontrolerska pločica

Slika 2.3.: 1) mikrokontroler LPC1768, 2) UART priključci, 3) AD pretvornik i periferije, 4) priključci napajanja, 5) priključci za tipkovnicu, 6) priključci za potencijalno proširenje slanja uzoraka, 7) vanjska Flash memorija, 8) priključci za Ethernet pločicu, 9) priključci za LCD prikaznik i izvor negativnog napajanja za pozadinsko svjetlo



Slika 2.4. Prikaz modela bez mikrokontrolerske pločice (u cilju prikaza Ethernet pločice)

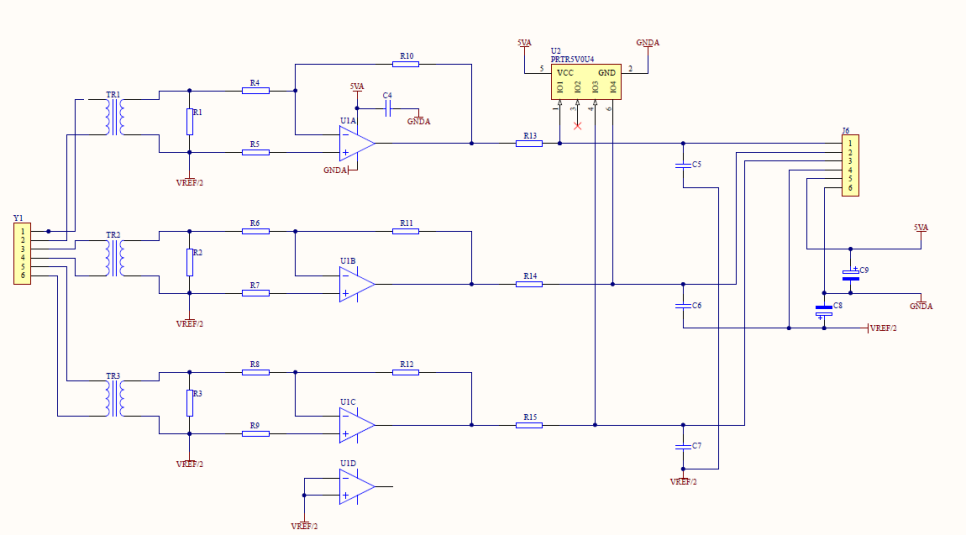
Slika 2.4.: 1) priključci Ethernet pločice za mikrokontrolersku pločicu, 2) PHY čip, 3) RJ-45 konektor

2.1 Ulazno sklopovlje

2.1.1 Strujni prilagodni sklop

Strujni prilagodni sklop radi na principu pretvorbe struje do 6A na napone unutar opsega A/D pretvornika. Korišteni su transformatori s N=2500 zavoja sekundara i jednim zavojem primara. Otpori koji zaključuju sekundar transformatora su iznosa 47Ω, a otpornici na ulaznim stezaljkama iznosa su 10kΩ, dok je otpornik u povratnoj vezi iznosa 121kΩ. Izraz (2.1) prikazuje odnos ulazne struje primara (I_{L1}) i izlaznog napona (U_{out}). Različiti indeksi otpornika istog iznosa otpora posljedica su označavanja otpornika na tiskanoj pločici.

$$U_{out} = -\frac{R_{10,11,12}}{R_{4,6,8}} R_{1,2,3} \frac{I_{L1}}{N} \qquad (2.1)$$



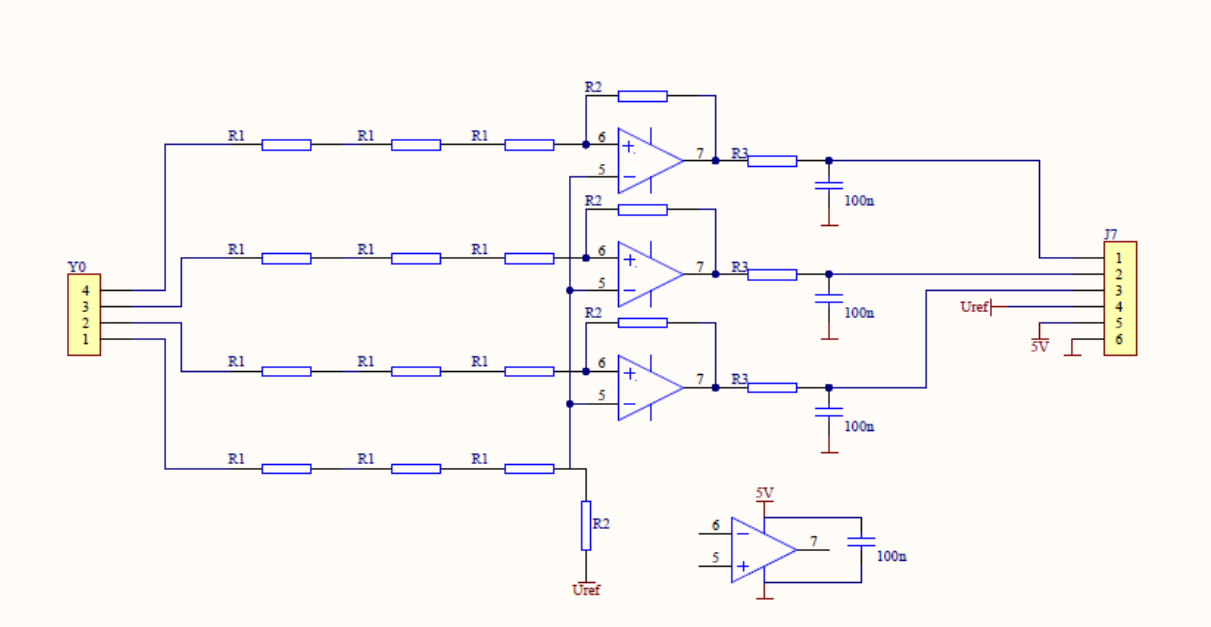
Slika 2.5. Električna shema strujnog prilagodnog sklopa

Sklop je realiziran sa četverostrukim operacionim pojačalom AD8544 koja zahtjeva samo jedan izvor napajanja. Komponenta PRTR5V0U4 je korištena kako bi se daljnji dio modela zaštitio od mogućih prenapona. Sklop štiti od elektrostatskih pražnjenja do -/+8kV, te time zadovoljava standard IEC 61000-4-2 (level 4) koji govori o kontaktnom pražnjenju.

2.1.2 Naponski prilagodni sklop

Naponski prilagodni sklop služi za prilagodbu napona do 300V, na napone unutar opsega A/D pretvornika. Realiziran je diferencijalnim pojačalom s vrlo visokim ulaznim otporom. U konkretnoj izvedbi otpornici R1 su iznosa 1MΩ, a otpornici R2 6.8kΩ. Ulazi $Y0_{4,3,2}$ predstavljaju redom fazne napone U1, U2 i U3, dok ulaz $Y0_1$ predstavlja nultu fazu. Izraz (2.2) prikazuje prijenosnu funkciju naponskog prilagodnog sklopa.

$$U_{out} = \frac{R_2}{3\ x\ R_1} U_{in} \qquad (2.2)$$

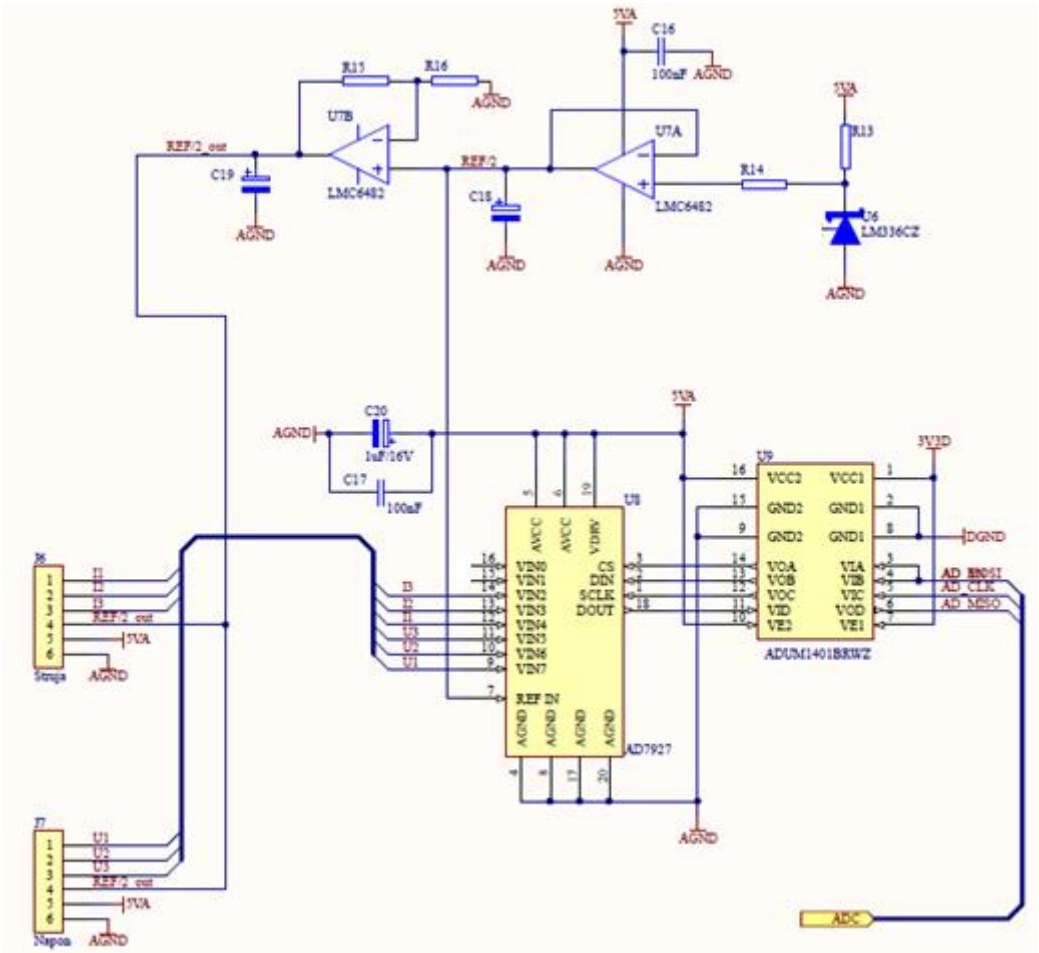


Slika 2.6. Električna shema naponskog prilagodnog sklopa

U konkretnom slučaju pojačala se nalaze unutar komponente LC6484.

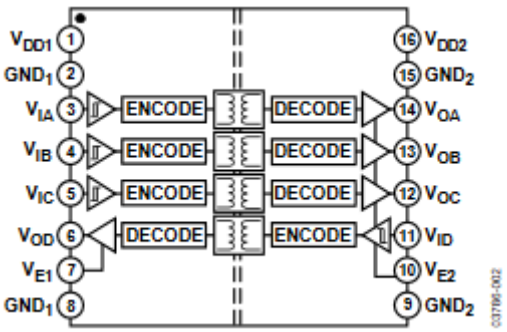
2.2 AD pretvornik

U modelu je korišten osmo-kanalni AD pretvornik AD7927. Spomenuti AD pretvornik primjenjuje dvanaest bitnu sukcesivnu aproksimaciju. Prva dva kanala nisu korištena, dok se na sljedećih šest kanala vrši uzorkovanje naponskih signala koji predstavljaju ulazne napone i struje trofazne niskonaponske mreže. Napon napajanja iznosi 5V, dok referentni napon iznosi 2.5V. Minimalno vrijeme između pojedine pretvorbe prema naputcima proizvođača mora iznositi 5µs, iako je tokom razvoja modela primijećeno kako je moguće ostvariti i manja vremena između AD pretvorbi (~4µs) uz jednaku efikasnost.



Slika 2.7. Električna shema AD pretvornika i pomoćnog sklopovlja

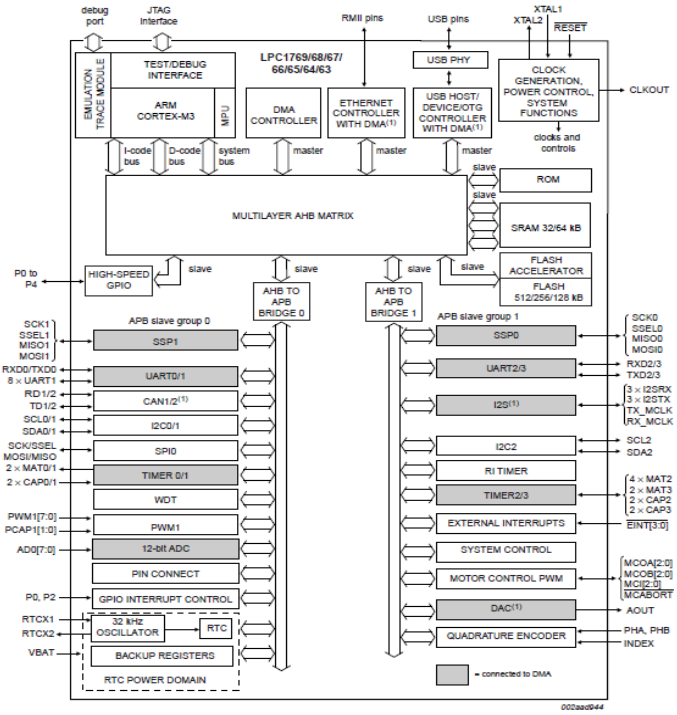
Kako bi bilo moguće ostvariti komunikaciju preko SPI sučelja s mikrokontrolerom potrebno je naponske razine AD pretvornika (5V) prilagoditi naponskim razinama mikrokontrolera (3.3V). Za tu namjenu korišten je četvero-kanalni digitalni izolator ADUM1401. SPI komunikacijom sa strane mikrokontrolera upravlja SSP0 kontroler.



Slika 2.8. Blok shema četvero-kanalnog digitalnog izolatora ADUM1401, [3] str. 1.

2.3 Mikrokontroler LPC1768

Mikrokontroleri LPC1769/68/67/66/65/64/63 spadaju u skupinu ARM Cortex-M3 mikrokontrolera. Korišteni mikrokontroler LPC1768 prodan je u više od milijardu primjeraka do 2017. godine. Od navedenih, LPC1769 podržava frekvenciju CPU-a do 120MHz, dok ostali podržavaju frekvencije do 100 MHz. ARM Cortex-M3 skupina mikrokontrolera izvedena je Harvard arhitekturom, s odvojenim instrukcijskim i podatkovnim sabirnicima, te sabirnicama za periferije. Lakše upravljanje prekidima omogućeno je preko ugrađenog Nested Vectored Interrupt Controller-a (NVIC-a). Korišteni mikrokontroler nalazi se u LQFP100 kućištu.

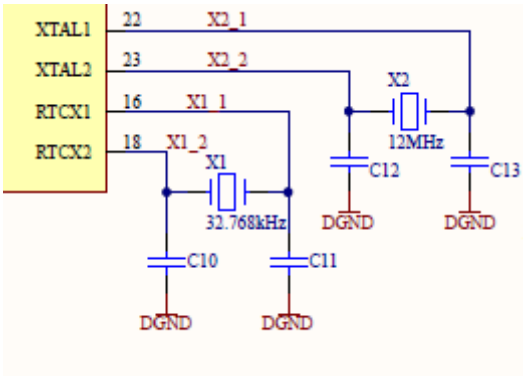


Slika 2.9. Blok dijagram skupine mikrokontrolera LPC1769/68/67/66/65/64/63, [1] str.6.

2.3.1 Izvori takta

Na priključke 16 i 18 spaja oscilator kojem je svrha generiranje RTC-a (*Real Time Clock-a*), odnosno generiranje signala frekvencije 1Hz. Na priključke 22 i 23 spaja se oscilator koji služi za generiranje glavnog takta mikrokontrolera. Prilikom odabira oscilatora i kondenzatora potrebno je proučiti *datasheet* mikrokontrolera i pronaći odgovarajuću izvedbu.

Daljnje upravljanje izvorom i iznosom glavne frekvencije mikrokontrolera programske je prirode te je objašnjeno u poglavlju vezanim uz programsku podršku.



Slika 2.10. Električna shema spoja kristalnih oscilatora

2.3.2 Memorije

Cortex-M3 procesori sadrže 4GB adresnog prostora. Programski kod nalazi se unutar 512kB *on-chip flash* memorije koja počinje od adrese 0x00000000 i završava na adresi 0x0007FFFF.

Boot ROM veličine 8kB nalazi se na memorijskoj adresi 0x1FFF0000 i završava na memorijskoj adresi 0x1FFF1FFF. Sadrži *boot loader* s *In-System Programming* (ISP) te aplikacijska programska sučelja (API) kao što su *In-Application Programming* (IAP) *flash* memorije, USART i I2C *driver*-e i profile snage za optimizaciju potrošnje snage.

SRAM memorija podijeljena je u dva dijela po 32kB. Prva banka počinje od memorijske adrese 0x10000000 i završava na memorijskoj adresi 0x10007FFFF, dok drugi banka počinje na memorijskoj adresi 0x2007C000 i završava memorijskoj adresi 0x20080000. Prva banka spojena je na CPU-ove instrukcijske i podatkovne sabirnice. Druga banka dodatno je podijeljena u dva dijela po 16kB radi lakšeg pristupa. Primarna zadaća druge banke su podaci s periferija i to je najčešća njena primjena, no također se može iskoristiti i za instrukcije i podatke s CPU-a.

2.3.3 Serijska sučelja

Komunikaciju s okolinom i periferijama omogućavaju, među ostalim, serijska sučelja. Za komunikaciju s vanjskim A/D pretvornikom i vanjskom Flash memorijom korištena su dva SSP kontrolera. Za komunikaciju s računalom i programiranje

korišten je UART, dok je za potrebe web servera korišten Ethernet MAC sa RMII sučeljem i pripadajućim DMA kontrolerom. Detaljniji opis izvedbi komunikacija naveden je u kasnijim poglavljima.

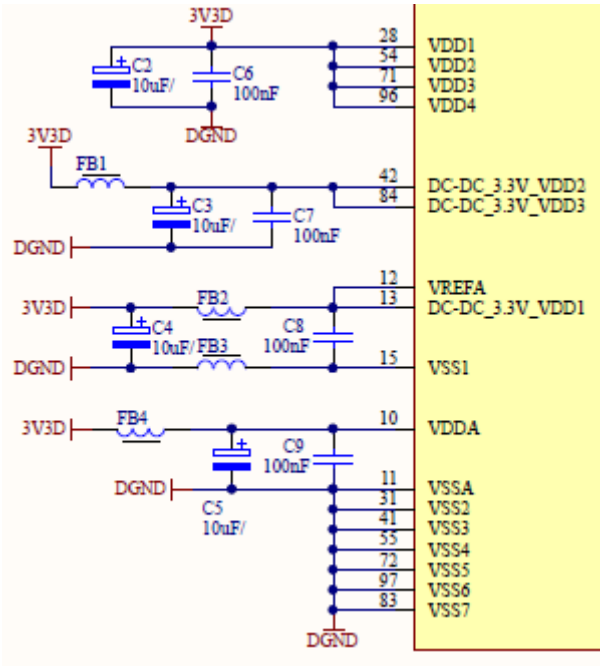
Od ostalih serijskih sučelja spomenuta skupina mikrokontrolera podržava CAN 2.0B kontroler, USB 2.0, SPI kontroler te I^2C i I^2S sučelje.

2.3.4 Napajanje mikrokontrolera

U konkretnoj izvedbi mikrokontrolera (LQFP100) na pinove 28, 54, 71, 96 priključuje se plus pol izvora napajanja za I/O skupine pinova. Pinovi 42 i 84 služe kao izvor napajanja za *on-chip* regulator napona. Pin 10 služi kako priključak za napajanje analognog dijela mikrokontrolera, dok pin 11 služi kao analogno uzemljenje. Za oba pina nužno je posvetiti pažnju minimiziranju šuma i greške. Minus polove napajanja također predstavljaju pinovi 31, 41, 55, 72, 83, 97.

U izvedbi u kojoj se koristi unutarnji ADC i DAC, pinovi 12 i 15 predstavljaju ulaz referentnog napona za rad ADC-a i DAC-a.

Feritna zavojnica služi za gušenje visokofrekventnog šuma i primjenjuje se u priključku svakog navedenog pina u slučajevima kad je potrebno minimizirati šum i grešku.



Slika 2.11. Električna shema priključaka za napajanje mikrokontrolera

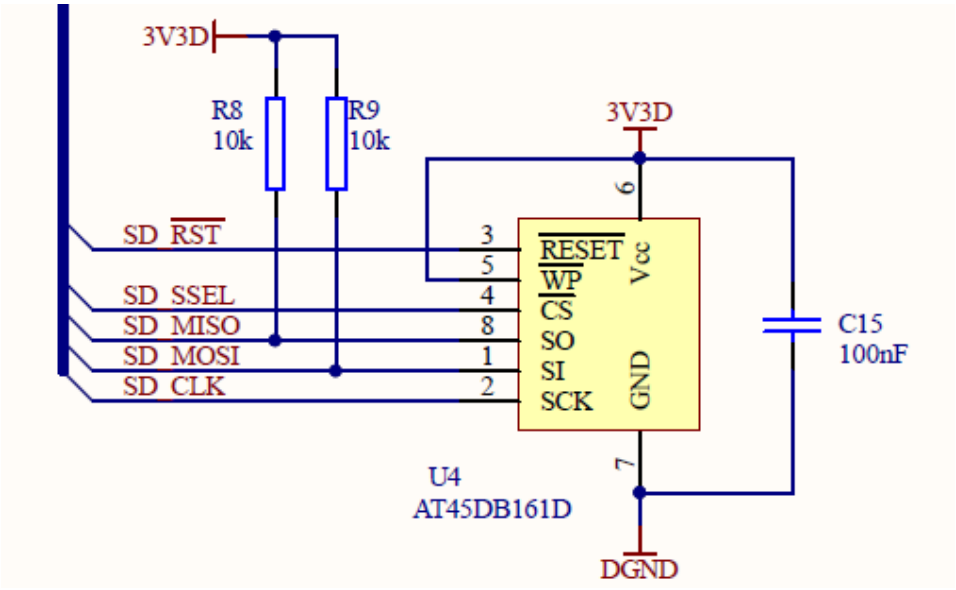
2.4 Vanjska Flash memorija

Odabrana je Flash memorija AT45DB161D, proizvođača Adesto. Konkretna memorija nalazi se u SOIC kućištu. Ukupni broj bitova memorije iznosi 17301504, te su raspodijeljeni na 4096 stranica po 512 ili 528 *byte*-ova. Potonja izvedba omogućava maksimalnu iskorištenost memorije. Osim po stranicama moguće je definirati memoriju preko blokova (jedan blok sadrži osam stranica) ili preko sektora (prvi sektor veličine osam stranica, drugi sektor veličine 248 stranica, petnaest sektora veličine 256 stranica). Uz glavnu memoriju postoje i dva SRAM *buffera* s također 512/528 *byte*-ova po *bufferu.*

Za ostvarivanje veze između memorije i mikrokontrolera korištena je SPI komunikacija. Za razliku od AD pretvornika, naponske razine memorije i mikrokontrolera su kompatibilne, stoga nema potrebe za digitalnim izolatorom kao kod veze između AD pretvornika i mikrokontrolera.

Chip Select služi za odabir memorije (periferije). Kada je neaktivan memorija će biti postavljena u *stand by mode* te ulazni pin neće primati podatke iz mikrokontrolera. *Serial Clock* daje takt za prijenos podataka između memorije i mikrokontrolera, te je podešen na 10MHz. Podaci koji dolaze na *Serial Input* okidaju se na rastući brid SCK-a, dok se okidanje podataka, koji se šalju prema mirkokontroleru sa *Serial Outputa*, odvija na padajući brid SCK-a. Oba pina SO i SI su priteznim otpornicima priključeni prema naponu napajanja. Za dobivanje SPI funkcionalnosti sa strane mikrokontrolera korišten je SSP1 kontroler. Programska izvedba objašnjena je u poglavlju programske podrške.

Write protection pin je onemogućen spojem na napajanje koje je iznosa 3,3V. *Reset* pin služi za prekid operacije u tijeku i resetiranje unutarnjeg automata stanja na početno stanje. S ponovnim radom memorija kreće tek kada *Reset* pin postane neaktivan.



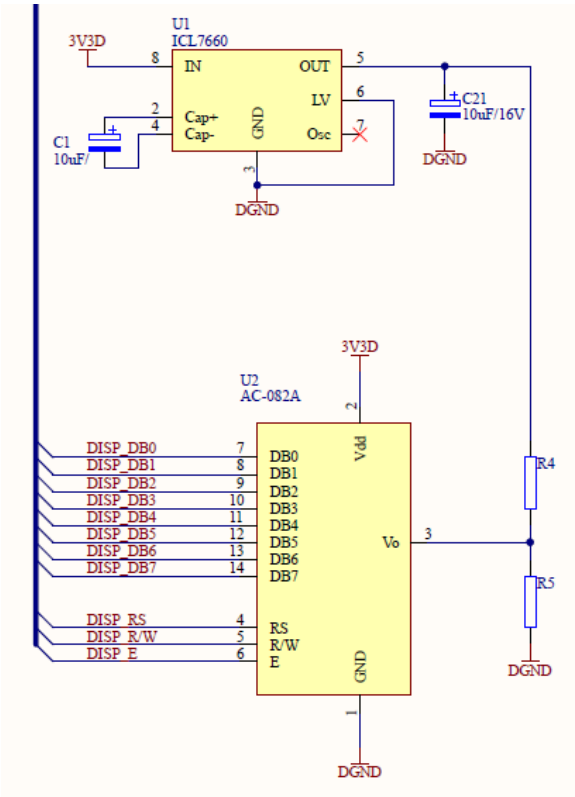
Slika 2.12. Flash memorija AT45DB161D

2.5 LCD prikaznik i tipkovnica

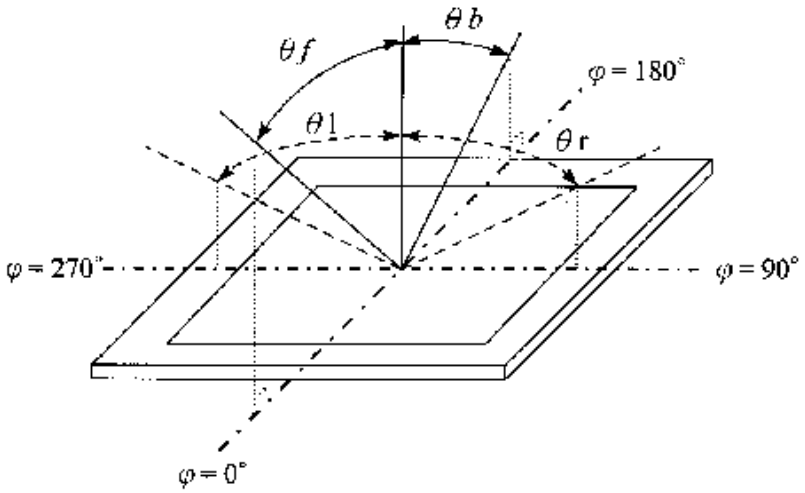
2.5.1 LCD prikaznik

Prikaz pojedinih veličina odvija se na LCD prikazniku AC-082A, proizvođača AMPIRE CO. LTD., uz upravljanje prikazom pomoću tipkovnice koja je opisana unutar sljedeće cjeline. Mogućnost prikaza na LCD prikazniku je ograničena na 2x8 znakova, te je zato pogodno da se istovremeno prikazuje samo jedna veličina. Na slici 2.11. među ostalim su vidljive podatkovne i upravljačke linije prikaznika. DB0-DB7 služe kao podatkovne linije, RS služi za odabir radi li se o poslanoj instrukciji ili podatku, R/W omogućuje čitanje/pisanje, dok E služi za omogućavanje slanja podataka. DB0-DB7 spojeni su na pinove mikrokontrolera P0.15-P0.22, dok su upravljački signali spojeni na pinove P2.6-P2.8.

Prikaznik koristi napajanje iznosa 3.3V, dok se za omogućavanje pozadinskog osvjetljenja koristi naponski pretvarač ICL7660, koji osigurava negativno napajanje iznosa -3,3V.



Slika 2.13. Električna shema LCD prikaznika i naponskog pretvarača ICL7660



Slika 2.14. Maksimalni rasponi kuteva gledanja (f=40°, b=35°, l=35°, r=35°) kod LCD prikaznika, [4] str. 6.

2.5.2 Tipkovnica

Tri tipke čine tipkovnicu koja ima dvije funkcije. Prva funkcija je upravljanje prikazima na LCD prikazniku. Početno stanje prikaza je „PM“ (*power measurement*).

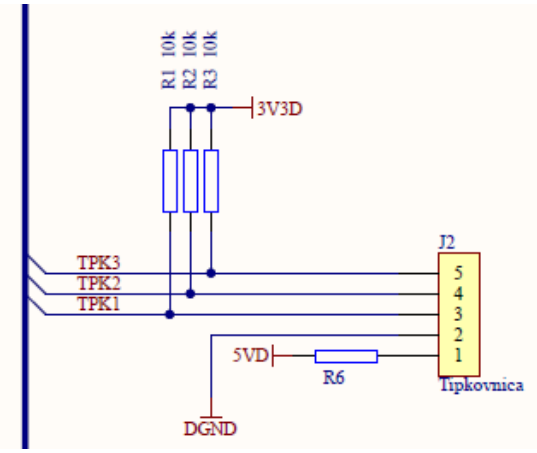
Pritiskom na TPK1 (tipka broj jedan) odabire se ulaz u izbornik koji prikazuje iznose THD-a faznih struja i napona te RMS vrijednosti faznih napona, faznih struja i

linijskih napona. Daljnjim pritiskanjem TPK1 prikazuju se redom sve navedene veličine. U trenutku kada je na prikazniku RMS iznos struje I_3 slijedeći stisak TPK1 pokrenuo bi umjeravanje modela na 230V/5A, što predstavlja drugu funkciju tipkovnice.

Pritiskom na TPK2 (tipka broj dva) odabire se ulaz u izbornik koji prikazuje iznose prividnih, jalovih i efektivnih snaga i energija.

TPK 3 (tipka broj tri) služi za pomicanje unazad unutar pojedinog izbornika.

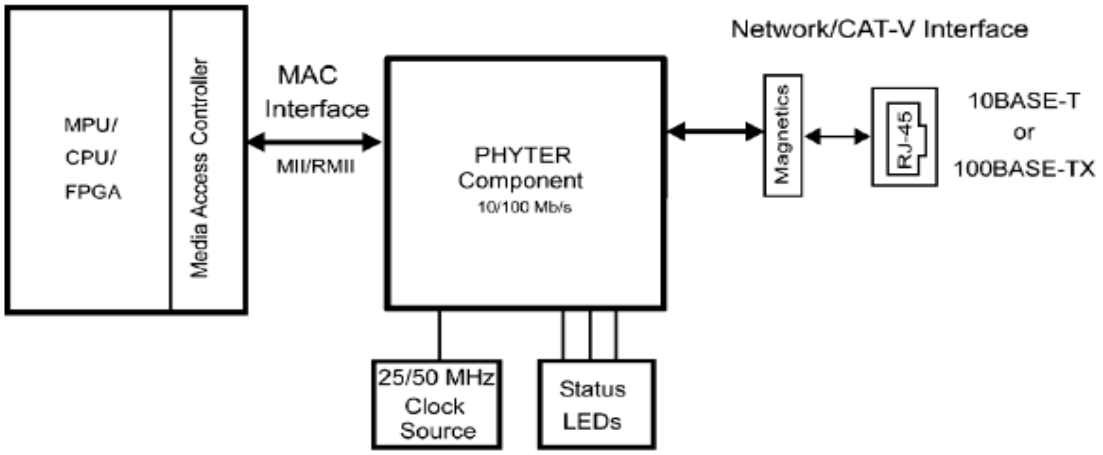
Tipke su spojene redom na mikrokontrolerske pinove P2.3-P2.5.



Slika 2.15. Električna shema tipkovnice

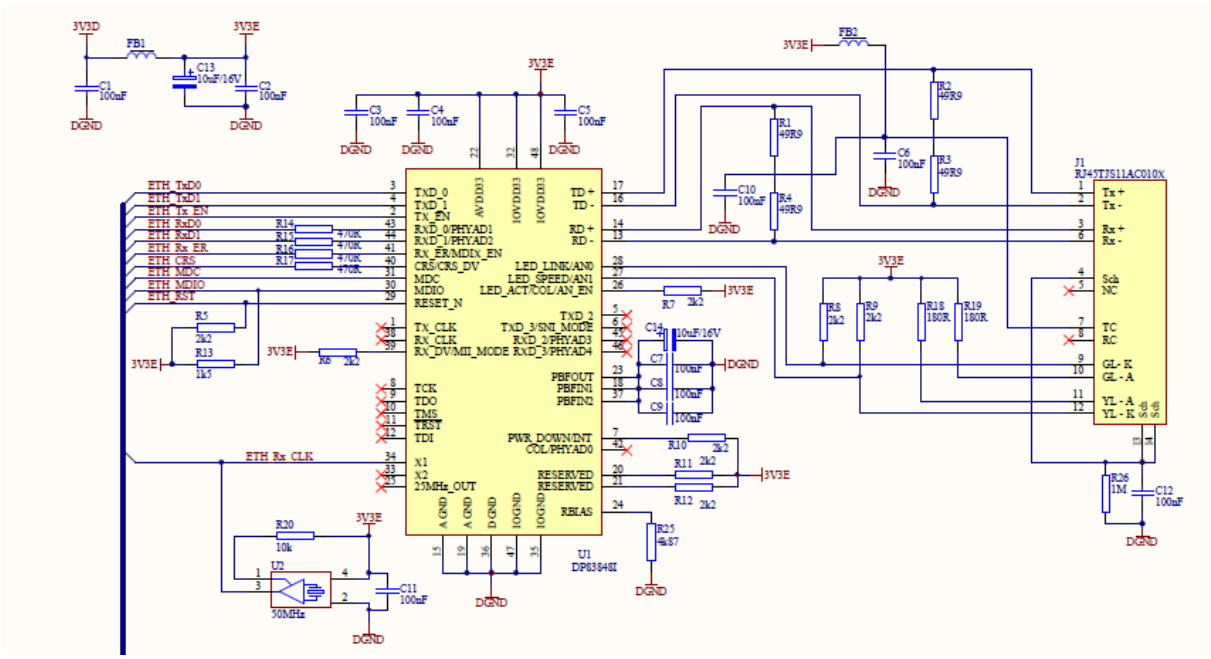
2.6 Ethernet

Model pruža mogućnost povezivanja preko *Ethernet*-a, odnosno slanja mjerenih veličina preko *Ethernet*-a i prikazivanje istih na *web* poslužitelju.



Slika 2.16. Blok shema tipičnog Ethernet sučelja, [5] str. 68.

Slika 2.14. na jednostavan način opisuje *Ethernet* sučelje. Konkretna realizacija sučelja sastoji se od MAC-a *(Media Access Controllera*), koji je integriran unutar korištenog mikrokontrolera, PHY čipa DP83848i, porodice PHYTER, te magnetskog priključka AT45TJS11AC010X. PHY čip pokriva fizički sloj (prvi sloj) OSI (*Open System Interconnection*) referentnog modela. MAC pokriva podatkovni sloj OSI modela, dok se za više slojeve koristi HTTP CGI server kojega kao djelomično rješenje pruža KEIL razvojni alat. Unutar konkretnog poglavlja pažnja je posvećena sklopovskog izvedbi sučelja. Sklopovska izvedba, u kojoj je PHY čip i njegovi ulazi i izlazi čine najvažnije dijelove, nepobitno utječe na konačno rješenje.



Slika 2.17. Električna shema *Ethernet* sučelja

PHY čip nalazi se u 48-pinskom HLQFP kućištu. Pinovi MDC i MDIO dio su upravljanja serijskim prijenosom. MDC je sinkroni izvor takta čija je maksimalna frekvencija 25MHz, dok je MDIO dvosmjerni upravljačko/podatkovni signal čiji izvor u našem slučaju može biti mikrokontroler ili sam PHY čip.

Prilikom korištenja MII (*Media Independent Interface*) pinova moguće je ostvariti različite načine rada. MII način rada za prijenos koristi četiri pina podataka i vanjski izvor takta frekvencije 25 MHz (±50 ppm).

RMII (*Reduced Media Independent Interface*) implementiran je na konkretnom modelu. RMII koristi dva pina podataka (TXD_0 i TXD_1) na kojima je slanje podataka na MAC sinkrono s vanjskim izvorom takta frekvencije 50 MHz (±50 ppm)

koji je spojen na pin X1. Prihvat RMII podataka s MAC-a također se odvija preko dva pina podataka RXD_0 i RXD_1, koji su također sinkroni sa spomenutim izvorom takta. Pin RX_DV omogućava indikaciju ispravnosti primljenih podataka koja je nezavisna obzirom na *Carrier Sense.* Prilikom detekcije neispravnog simbola pin RX_ER postavlja se u visoko stanje (sinkrono s taktom na X1), te je CRS_DV također postavljen ako je riječ o 100 Mb/s načinu rada. Pin RX_ER nije neophodan prilikom implementacije, neovisno o načinu rada (RMII/MII). Pin CRS (*Carrier Sense*) se prilikom *Half Duplex* operacije postavlja u visoko stanje bilo da je riječ o slanju ili primanju paketa, dok se prilikom *Full Duplex* operacije CRS postavlja u visoko stanje samo kod aktivnosti primanja paketa. CRS se postavlja u nisko stanje nakon što se pojavi kraj paketa.

PHY čip podržava opciju *Auto-Negotiation*. Takva opcija pruža dinamičku mogućnost mijenjanja konfiguracije slanja i primanja podataka s obzirom na najbolje zajedničke performanse povezanih uređaja. Pin AN_EN omogućava *Auto-Negotiation*, dok pinovi AN_1 i AN_0 sugeriraju PHY čipu koja je najbolja moguća opcija za trenutni prijenos podataka. Opcija najvišeg prioriteta je *100BASE-TX Full Duplex*, dok je najnižeg prioriteta *10BASE-T Half Duplex*.

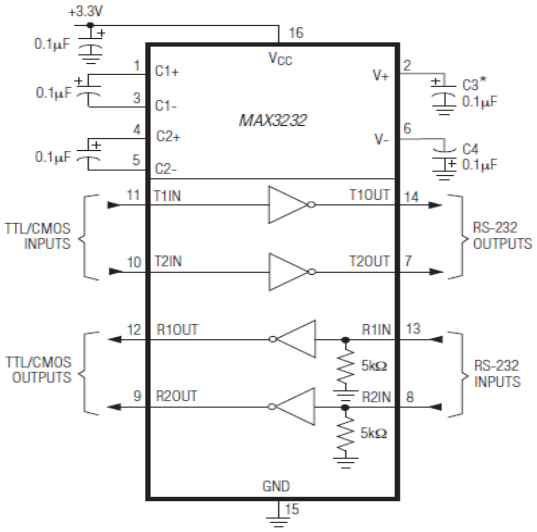
PMD (*Physical Medium Depedent*) sučelje sastoji se od dva para diferencijalnih signala za slanje i primanje podataka. Pinovi TD- i TD+ služe za slanje podatka, dok pinovi RD- i RD+ služe za primanje podatka. Unutar oba diferencijalna para mora postojati razlika u naponskim nivoima u iznosu od 3.3V. Preporučljivo je da prilikom izrade PCB-a parovi budu paralelni, jednake duljine i što kraći.

Kako bi se osigurao ispravan rad PHY čipa na pin PFBOUT (*Power Feedback Output*) je spojen elektrolitski kondenzator (po mogućnosti od tantala) iznosa 10 µF u paraleli s keramičkim kondenzatorom iznosa 0.1 µF. Pinovi PFBIN1 i PFBIN2 (*Power Feedback Input*) spojeni su na pin PFBOUT, uz korištenje kondenzatora iznosa 0.1 µF.

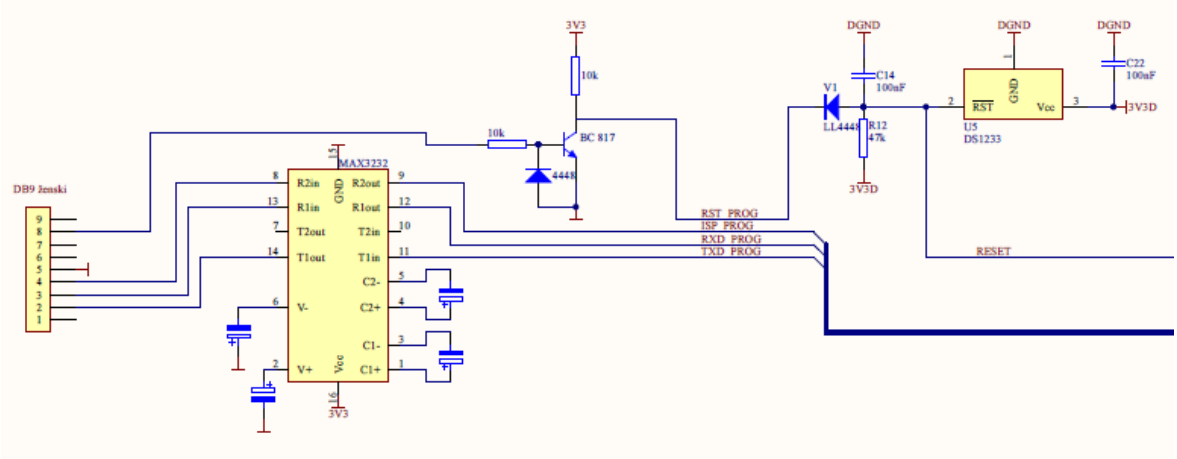
2.7 Serijsko sučelje za programiranje i ispis rezultata

Za komunikaciju osobnog računala s mikrokontrolerom i programiranje samog mikrokontrolera koristi se UART sučelje. Ukoliko računalo nema serijskog sučelja RS232 potreban je dodatni USB/RS232 konvertor (konektor DB9 ženski) kako bi se fizički povezalo računalo i mikrokontroler. Unutar modela UART komunikacija je

ostvarena pomoću komponente Maxim MAX3232 koja interno stvara potrebna napajanja (+/- 7V-12V RS232 naponske razine) iz standardnih 3.3V u konkretnoj izvedbi.



Slika 2.18. Blok shema komponente MAX3232, [2] str.12.



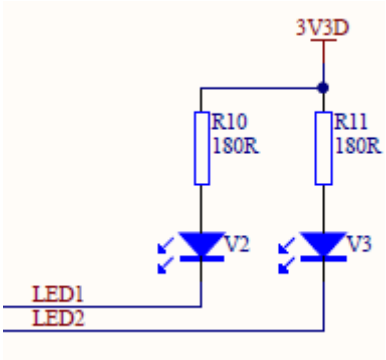
Slika 2.19. Električna shema sučelja za programiranje

Pomoću linija RXD_PROG i TXD_PROG vrši se prijenos podataka koji zadovoljava protokol RS232. Sa strane mikrokontrolera korišten je kontroler UART0. Linija ISP_PROG (*In-System Programming*) spojena je na priključak P2.10 mikrokontrolera kako bi bilo omogućeno reprogramiranje *on-chip flash* memorije, koristeći *boot loader* i UART0 serijski port. Linija RST_PROG koristi se kako bi se mikrokontroler resetirao prije početka svakog transfera na spomenutom sučelju, te također prije svakog reprogramiranja (kako bi *boot loader* uopće bio pokrenut).

Zbog zauzeća obje linije MAX3232 koje prihvaćaju signale RS232 naponskih razina, za liniju RST_PROG potrebno je napraviti pretvorbu naponskih razina izvan MAX3232 (ili koristiti još jedan MAX3232 isključivo radi jedne linije). RST_PROG linija i RST izlaz iz komponente DS1233 jedine mogu resetirati mikrokontroler. Komponenta DS1233 nadzire napajanje, te daje upozorenje (odnosno reset) ako dođe do pada razine napajanja.

2.8 LED diode

LED diode služe kao pomoć u određivanju vremenskih intervala i dijagnostici kvarova prilikom izrade modela. Upravljanje LED diodama vrši se preko mikrokontrolerskih pinova P0.10 i P0.11.



Slika 2.20. Električna shema LED diodi

2.9 Napajanje

Model mjernog uređaja moguće je napajati sa istosmjernim (DC) ili izmjeničnim (AC) naponom iznosa 10 do 15V. Maksimalna potrošnja modela je otprilike 3,5 VA. Za napajanje je korišten vanjski AC-DC pretvornik koji se napaja mrežnim naponom 230VAC i daje na izlazu 12VDC i priključuje se na utični konektor modela.

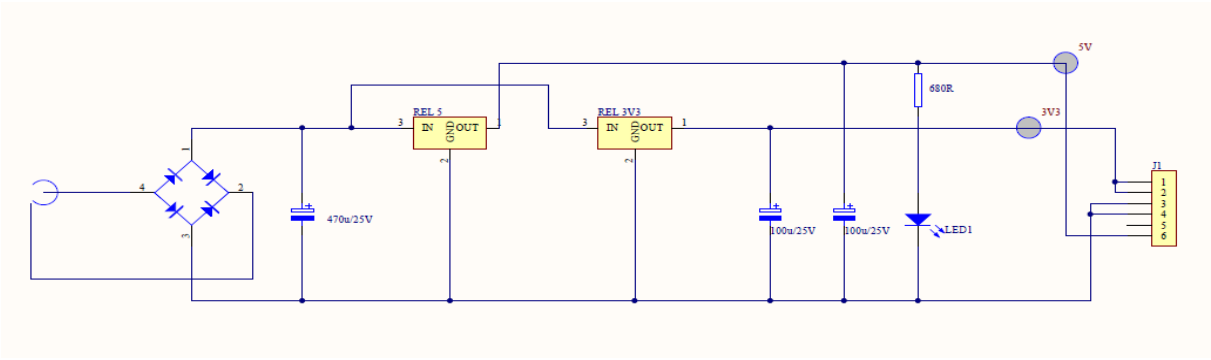
Sklop napajanja u modelu uređaja ima diodni most, koji djeluje kao punovalni ispravljač u slučaju napajanja sa AC naponom, i kao zaštita kod priključka reverznog polariteta u slučaju napajanja sa DC naponom. Smanjenje valovitosti napona ostvaruje se elektrolitskim kondenzatorom.

Dimenzioniranje kapaciteta ovisno je o potrošnji uređaja i može se izvesti prema Schadeovu dijagramu. Za potrošnju Pmax=3,5W i minimalni napon napajanja 10V, struja tereta je I_L=0,35A odnosno ekvivalentni otpor tereta je R_L=29 ohm. Ako dozvoljavamo valovitost izlaza 20% iz Shadeovog dijagrama očitavamo $\omega R_L C > 3,5$ odnosno C > 384 uF. Izabrana je standardna vrijednost 470 uF. Pri odabiru elektrolitskih kondenzatora pažnju je potrebno obratiti na njihove napone. Kako bi se produžio životni vijek elektrolitskih kondenzatora za kvadrat, potrebno je odabrati napon koji je dvostruko veći od napona na kojeg je kondenzator priključen, što je u konkretnoj primjeni i učinjeno.

Model uređaja zahtjeva za rad dva naponska nivoa: 3,3V i 5V.

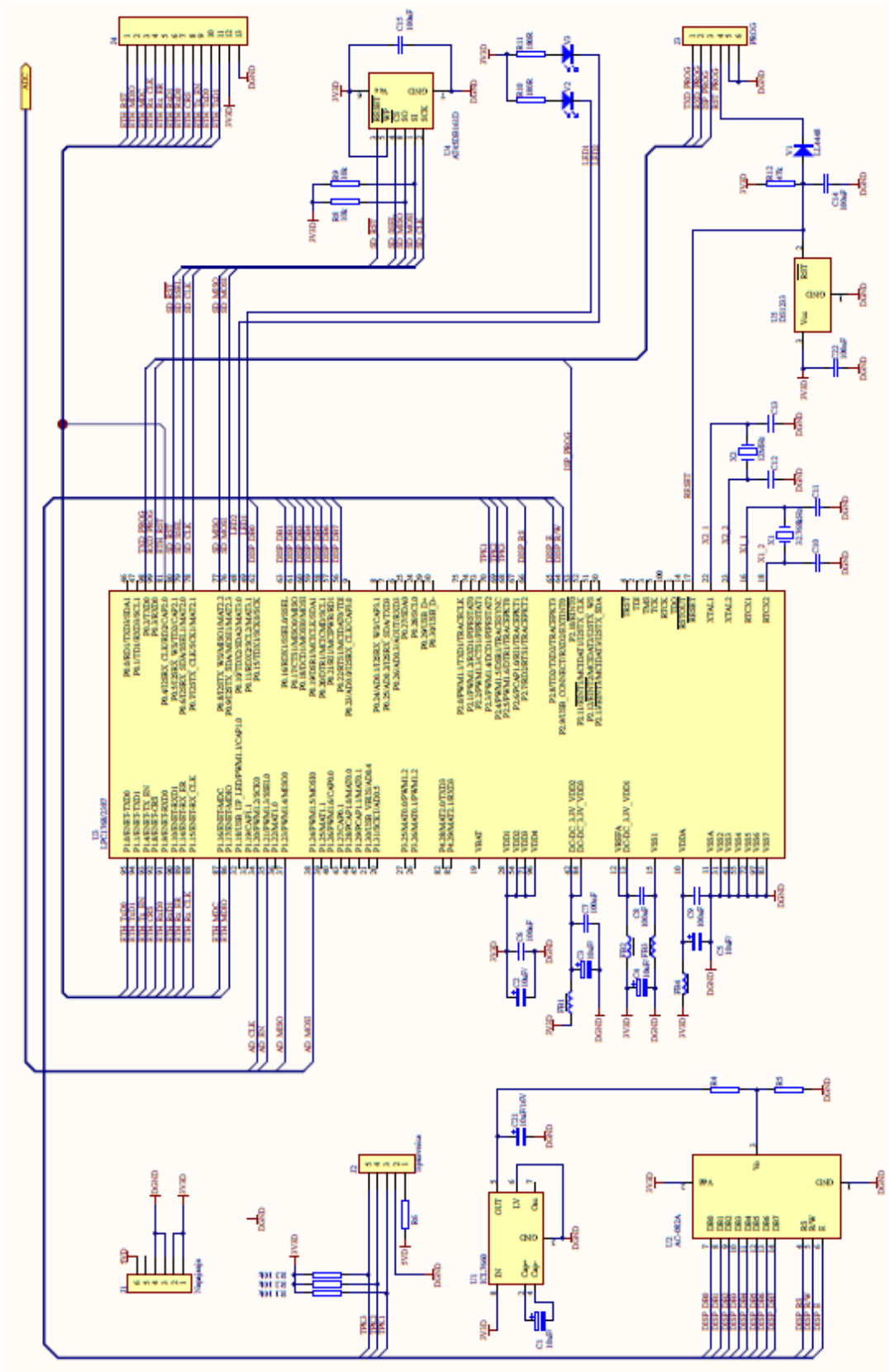
Za prilagodbu razine napona korištena su dva DC-DC pretvornika, jedan za dobivanje razine od 5V, drugi za razinu od 3.3V. Na izlazima regulatora je paralelni spoj elektrolitskih i keramičkih kondenzatora koji kompenziraju brze promjene tereta karakteristične za mikroprocesorske sklopove.

LED dioda na slici 2.1. predstavlja diodu koja je upaljena kada je priključeno napajanje.



Slika 2.21. Električna shema napajanja modela

2.10 Prikaz mikrokontrolera sa svim vanjskim jedinicama

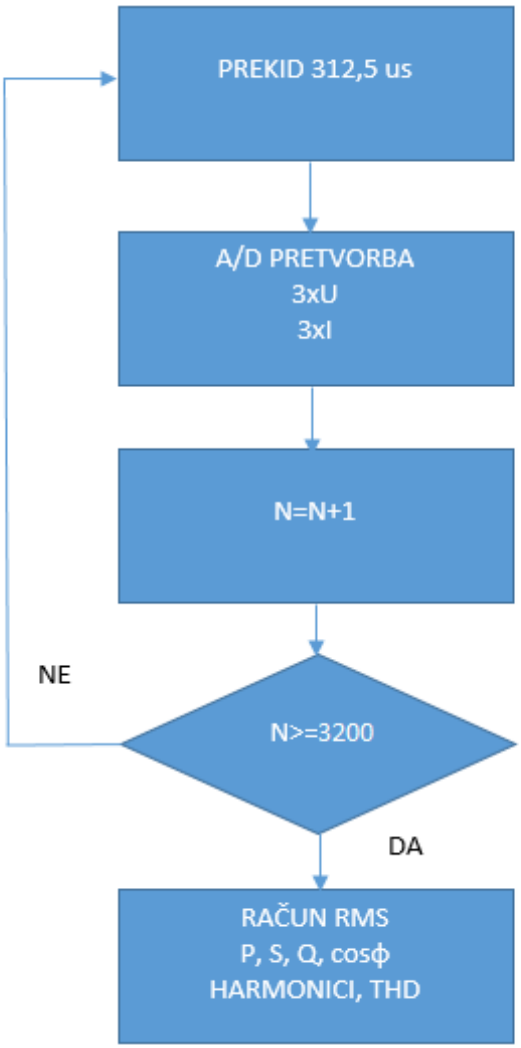


Slika 2.22. Prikaz mikrokontrolera i vanjskih jedinica

3 Programska realizacija

Programska realizacija pisana je velikom većinom u C programskom jeziku, dok je dio potreban za implementaciju *web* poslužitelja pisan pomoću HTML-a, Javascript-a i CGI-a. Razvojno okruženje korišteno prilikom implementacije programske realizacije je Keil MDK-ARM Version 5.

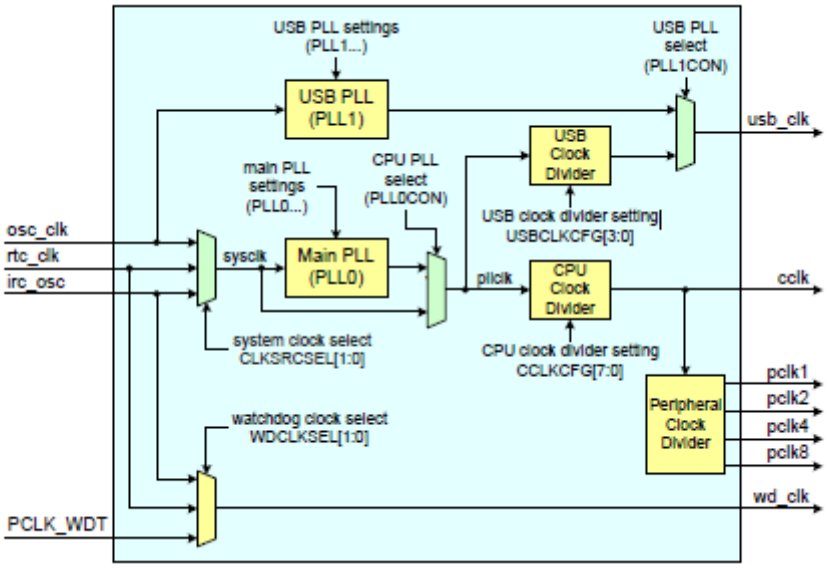
3.1 Blok shema osnovne programske potpore



Slika 3.1. Blok shema osnovne programske potpore

3.2 Inicijalizacija frekvencije takta

Radi lakšeg prilagođavanja modela potrebi, frekvenciju oscilatora koji su priključeni na mikrokontroler (poglavlje 2.2.2.) moguće je modificirati, te također izabrati izvor frekvencije čija će frekvencija predstavljati glavnu frekvenciju mikrokontrolera.



Slika 3.2. Generiranje takta kod mikrokontrolera LPC 176x/175x, [6] str. 31.

Pomoću registra CLKSRCSEL odabire se izvor sistemskog takta. Oscilator označen s X1, frekvencije 32kHz, na PRIKAZU XX korišten je kao isključivo izvor RTC-a (*Real-Time Clock*). Moguće ga je odabrati i kao izvor sistemskog takta, ali nije pogodno za potrebe modela. Kao izvor sistemskog takta odabire se oscilator X2, frekvencije 12 MHz. Osim dva vanjska izvora takta postoji i unutarnji RC oscilator (IRC).

Maksimalna frekvencija moguća na ulazu predviđenom za oscilator iznosi 25 MHz. Kako takva vrijednost ne može zadovoljiti sve potrebe modernih mikrokontrolera, unutar samog mikrokontrolera postoji PLL (*Phase Locked Loop*) i *CPU Clock Divider*-i što omogućava daljnje operacije nad radnom frekvencijom mikrokontrolera. Unutar PLL-a moguće je skalirati frekvenciju množenjem i dijeljenjem. Pomoću registra PLL0CFG određujemo argumente M (množenje) i N (dijeljenje). Na izlazu iz PLL-a raspolažemo sa frekvencijom prema izrazu 3.1:

$$F_{PLL} = (2 * M * sysclk)/N \qquad (3.1)$$

U konkretnom slučaju odabrano je M=25, N=1. Ti iznosi trebaju biti upisani u registar PLL0CFG kao M=24 i N=0 jer se povećava vrijednost faktora za jedan. Nakon izlaska iz PLL-a slijedi *CPU Clock Divider* koji konfiguriramo pomoću registra CCLKCFG upisom odgovarajuće vrijednosti djelitelja. U konkretnom slučaju u registru se nalazi vrijednost pet, ali je ona kao i kod PLL0CFG uvećana za jedan, te je vrijednost djelitelja nakon PLL-a šest. Nakon obavljenih inicijalizacija na izlazu iz

CPU Clock Dividera nalazi se frekvencija iznosa 100 MHz, što čini maksimalni dozvoljeni radni takt mikrokontrolera primijenjen u konkretnoj realizaciji.

Daljnje modifikacije frekvencije takta periferija moguće je izvršiti preko registra PCLKSEL0, te pomoću *prescale* registara kod pojedinih kontrolera periferija. Osim kod definiranja frekvencije takta periferija preko registara kontrolera periferija, frekvencije takta definiraju se u registrima u sistemskoj .c datoteci mikrokontrolera.

3.3 Povezivanje mikrokontrolera s vanjskim jedinicama

3.3.1 Serijsko sučelje za programiranje i ispis rezultata

Za ostvarenje komunikacije između računala i mikrokontrolera, te programiranje samog mikrokontrolera korištena je periferija UART0 na mikrokontroleru. Inicijalizacija UART0 periferije i odgovarajućih pinova na mikrokontroleru opisana je u prikazu XX. *Baud* rate iznosi 115200.

```
void uart0_init(void)
{
  LPC_PINCON->PINSEL0 &= ~0x000000F0;
  LPC_PINCON->PINSEL0 |= 0x00000050;        // RxD- P0.3, TxD0 P0.2
  LPC_UART0->LCR = 0x83;                    // 8 bita, bez pariteta, 1 stop bit

  LPC_UART0->DLM = 0;                       //
  LPC_UART0->DLL = 31;                      // brzina uart0
  LPC_UART0->FDR = 0x43;                    //

  LPC_UART0->LCR = 0x03;
  LPC_UART0->FCR = 0x07;                    // omogući i resetiraj TX i  RX FIFO
}
```

Isječak koda 3.1. Inicijalizacija UART0 periferije

Kako bi bilo moguće prikazivati podatke na računalu preko UART-a, potrebno je redefinirati funkcije koje se izvršavaju u *semihost* okruženju na način da se kod izvršava na stvarnom sklopovlju. Cilj je zamijeniti funkcije iz biblioteke novima. Taj postupak se naziva *retargeting.* U konkretnom primjeru potrebno je redefinirati funkcije koje poziva printf familija (fputc(), ferror(), _stdout), funkciju za obradu grešaka (_ttywrch) i funkciju za izlaz (_sys_exit). Također je u kod potrebno dodati simbol __use_no_semihosting_swi. Nakon provedenog redefiniranja funkcija printf služit će ispisivanju vrijednosti preko terminala na osobno računalo.

```
#include <stdio.h>
#include <rt_misc.h>

#pragma import(__use_no_semihosting_swi)

extern int  sendchar(int ch);

struct __FILE { int handle};
FILE __stdout;

int fputc(int ch, FILE *f) {
  return (sendchar(ch));
}

int ferror(FILE *f) {
  /* Your implementation of ferror */
  return EOF;
}

void _ttywrch(int ch) {
  sendchar(ch);
}

void _sys_exit(int return_code)
{
          label:  goto label;  /* beskonacna petlja */
}
```

Isječak koda 3.2. Retarget.c

3.3.2 Povezivanje mikrokontrolera s AD pretvornikom

Registar PINSEL3 predstavlja gornjih petnaest pinova registra *porta* 1. Pin 20 predstavlja takt serijske veze, pin 21 služi kao *serial select,* dok pinovi 23 i 24 predstavljaju MISO (*master in, slave out*) i MOSI (*master out, slave in*). Registar CPSR određuje koliko se skalira glavni takt mikrokontrolera za periferiju SSP0. Glavni takt mikrokontrolera iznosi 100 MHz, tako da dijeljenjem s 10 takt koji je doveden na periferiju SSP0 iznosi 10 MHz (što je i maksimalni iznos za A/D pretvornik). CR0 i CR1 su kontrolni registri serijske veze. CR0 određuje veličinu podataka koji se šalju, vrstu veze (u ovom slučaju SPI), u kojem se stanju nalazi takt serijske veze između slanja podataka i na kojoj promjeni brida takta serijske veze dolazi do dohvaćanja podataka. Pomoću registra CR1 omogućujemo SSP kontroler. Funkcije pisanja i čitanja s A/D pretvornika biti će objašnjene u nastavku. Kako bi A/D pretvornik mogao započeti s radom potrebno je obaviti rutinu nakon uključivanja napajanja.

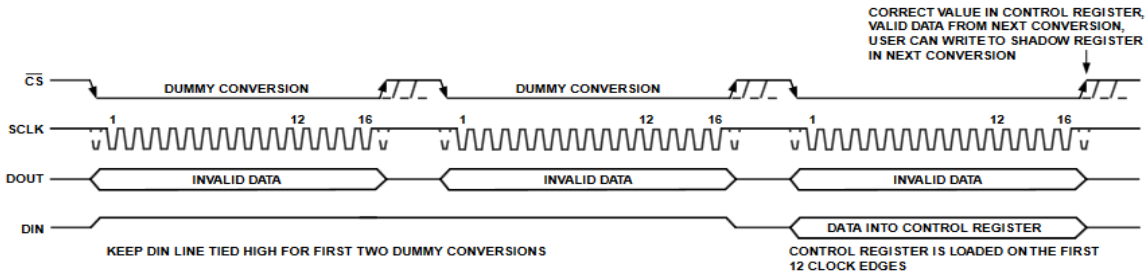


Figure 24. Three-Dummy-Conversions to Place AD7927 into the Required Operating Mode After Power Supplies Are Applied

Slika 3.3. Rutina inicijalizacije A/D pretvornika nakon uključivanja napajanja, [7] str. 21.

```
void AD7927_Init(void)
{
  LPC_PINCON->PINSEL3 &= 0xFFFC30FF;        // kako bi bitove postavili u nulu
  LPC_PINCON->PINSEL3 |= 0x0003CF00;        // Pin Function Select Register

  LPC_SSP0->CPSR |= 10;                     // SSP0 Clock Prescale Register, max brzina 10MHz
  LPC_SSP0->CR0  = 0x4F;                    // SSP0 Control Register 0  , 16 bitni transfer, 00-
                                            //spi, CPOL=1-serial clock izmedu frameova high,
                                            //CPHA=0-na prvom bridu clocka dogadja se
                                            //konverzija
  LPC_SSP0->CR1  = 0x02;                    // SSP0 Control Register 1  - ssp controller enable

  us_delay(5);

  write_ADC_SSP0(0xFFFF);                   //dummy cycle
  write_ADC_SSP0(0xFFFF);                   //dummy cycle
  write_ADC_SSP0(0xDF9 << 4);               //prvi MOSI, write=1, range 0-2xVref, straight
                                            //binary, svi kanali-od nultog do zadnjeg
  us_delay(5);

  SSP0_Go_Idle();
  read_ADC_SSP0();                          // nakon treceg pisanja svejedno dobijemo
                                            //invalid data, pa kako bi pri sljedecem pozivu
                                            //reada mogli citati valid dana sada procitamo
                                            //invalid dana
}
```

Isječak koda 3.3. Funkcija inicijalizacije A/D pretvorbe

Treće po redu slanje podatka unutar navedene rutine određuje način rada A/D pretvornika. U konkretnom slučaju odabran je binarni prijenos, interval pretvorbe 0-2xVref (0-5 V), prolazak svih ulaza po redu od nultog do sedmog, te normalan način rada što se tiče potrošnje. Funkcija unutar isječka koda 1.2. us_delay() osigurava kako bi između svakog slanja podataka prošlo barem 5 us, što je uvjet rada

konkretnog A/D pretvornika. Nakon trećeg pisanja potrebno je pročitati beskorisne podatke koji dolaze s A/D pretvornika kako ne bi u kasnijim čitanjima prouzročili pogrešan rad.

```
uint16_t read_ADC_SSP0()
{
  uint16_t value;

  LPC_SSP0->DR = 0x0000;                  //da bi  nesto dobili moramo i poslati

  while(!((LPC_SSP0->SR >> 0) & 0x1));   //  cekaj dok transmit fifo nije prazan
  value = LPC_SSP0->DR;                   //  citaj

  while(((LPC_SSP0->SR >> 4) & 0x1));    //cekaj dok je busy

  return value;
}
```

Isječak koda 3.4. Funkcija čitanja s A/D pretvornika

```
void write_ADC_SSP0(uint16_t value)
{
  us_delay(5);

  SSP0_Go_Idle();
  LPC_SSP0->DR = value;

  while(!((LPC_SSP0->SR >> 0) & 0x1));  // cekanje dok transmit fifo nije prazan

  while(((LPC_SSP0->SR >> 2) & 0x1))    // cekaj dok je recieve fifo pun// ocisti fifo recieve
// samim pozivanjem
     LPC_SSP0->DR;

  while(((LPC_SSP0->SR >> 4) & 0x1));   // cekanje dok je busy
}
```

Isječak koda 3.5. Funkcija pisanja na A/D pretvornik

LPC_SSP->DR je podatkovni registar serijske veze između mikrokontrolera i AD pretvornika. Pisanjem se puni FIFO odašiljački registar, dok se čitanjem (pozivanjem) prazni FIFO prijemni registar. LPC_SSP->SR provjerava stanje FIFO prijemnog i odašiljačkog registra.

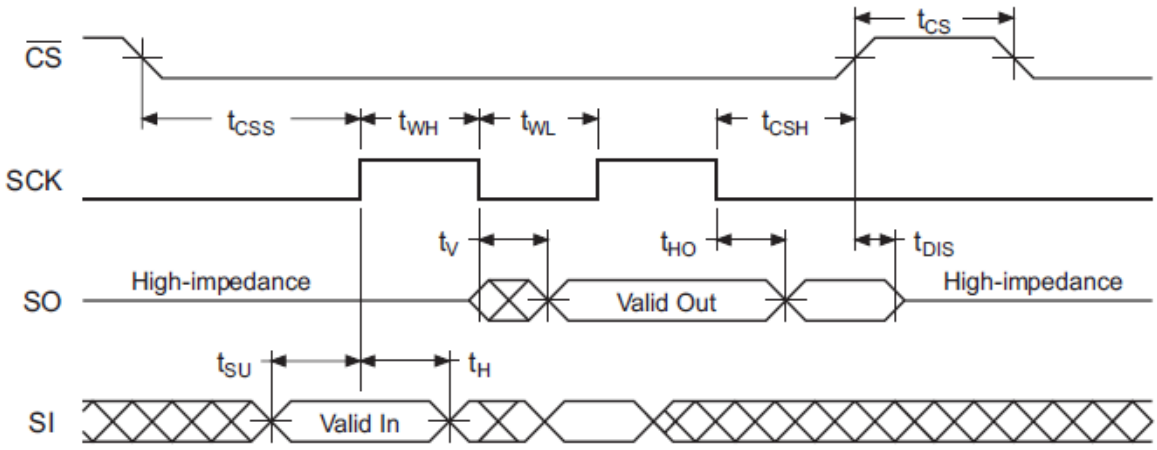
Bit	Symbol	Description	Reset Value
0	TFE	Transmit FIFO Empty. This bit is 1 is the Transmit FIFO is empty, 0 if not.	1
1	TNF	Transmit FIFO Not Full. This bit is 0 if the Tx FIFO is full, 1 if not.	1
2	RNE	Receive FIFO Not Empty. This bit is 0 if the Receive FIFO is empty, 1 if not.	0
3	RFF	Receive FIFO Full. This bit is 1 if the Receive FIFO is full, 0 if not.	0
4	BSY	Busy. This bit is 0 if the SSPn controller is idle, or 1 if it is currently sending/receiving a frame and/or the Tx FIFO is not empty.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Slika 3.4. Registar LPC_SSP0/1->SR, [6] str. 435.

3.3.3 Povezivanje mikrokontrolera s vanjskom Flash memorijom

Namjena vanjske *flash* memorije u modelu je spremanje rezultata umjeravanja koja se odvija na 230V/5A, te korištenja tih rezultat prilikom daljnjih računa mjerenih vrijednosti.

Memorija podržava SPI0 i SPI3 način rada. SSP1 kontroler sa strane mikrokontrolera podešen je da radi u SPI0 načinu rada.



Slika 3.5. SPI0 način rada, [8] str. 52.

Kako bi se uspješno prenosili podaci preko SPI sučelja potrebno je stanjem pina *Chip Select* upravljati manualno, tj. ne koristiti SSP1 kontroler sa strane mikrokontrolera za upravljanje njime. Razlog tomu je što SSP0/1 kontroler neposredno nakon slanja podatka (u konkretnom slučaju podatka veličine 8 bita)

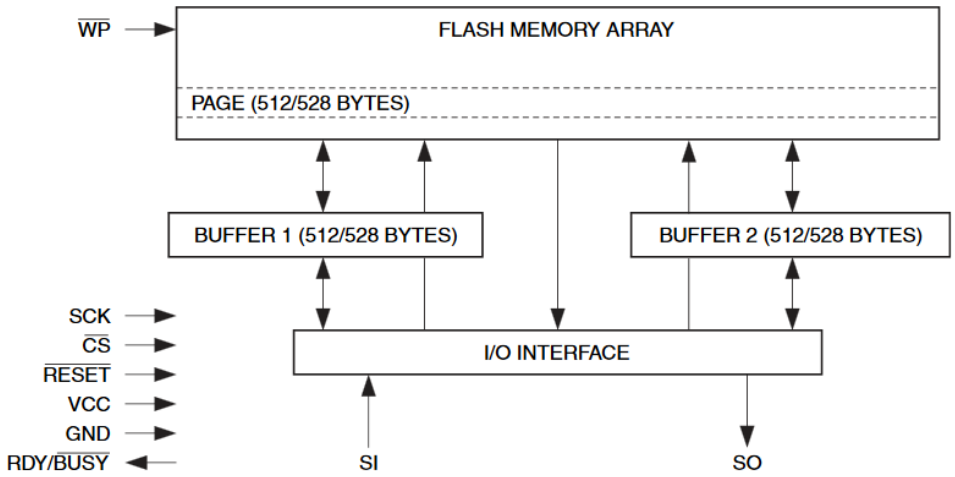
postavlja *Chip Select* u visoko stanje. Takvom izvedbom nije moguće poslati naredbe niti čitati/pisati podatke memorije. *Chip Select* označava početak i kraj cjelokupne operacije (čitanja/pisanja), koja može sadržavati veći broj osam bitnih prijenosa.

```
void set_cs(void) {
        LPC_SSP1->DR;
        LPC_GPIO0>FIOCLR=(1<<6);
        us_delay(5);
}
```

```
void clr_cs(void) {
        us_delay(5);
        LPC_GPIO0->FIOSET=(1<<6);
        us_delay(5);
}
```

Isječak koda 3.6. Funkcije za upravljanje *Chip Selectom*

Zbog sličnosti naredbi upisa i čitanja preko SPI sučelja SSP1 s naredbama za upis i čitanje preko SPI sučelja SSP0, opis tih rutina neće biti istaknut. Jedina razlika postoji u veličini podataka koja se čita/šalje tokom jedne narede čitanje/upisa. U slučaju AD pretvornika to je 16 bita, dok se kod *flash* memorije vrši prijenos iznosa 8 bita po operaciji. Razlog tomu je struktura naredbi *flash* memorije.



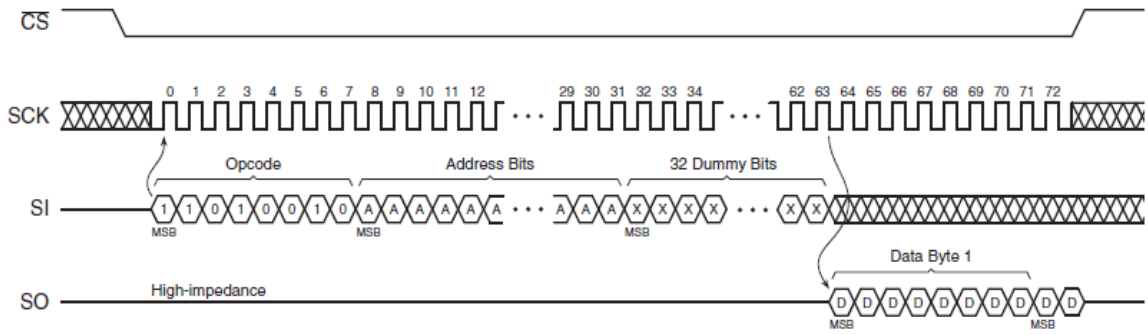
Slika 3.6. Blok dijagram *flash* memorije, [8] str. 4.

Dva osnovna pristupa memoriji, bilo da se radi o čitanju ili pisanju u nju, su pristup preko *buffera* i direktni pristup. Također stranice memorije mogu biti tretirane kao 512 ili 528 *byte*-ne. U konkretnom slučaju stranica je tretirana kao 528 *byte*-na.

Kontinuirano čitanje cjelokupne memorije kao polja moguće je odraditi na četiri maksimalne razine frekvencije takta (do 15 MHz, 50 MHz, 85 MHz, 104 MHz).

Kontinuirano čitanje cjelokupne memorije ne uzrokuje nikakva kašnjenja prilikom prelaska između stranica ili prelaska s kraja na početak memorije. Deaktivacijom *Chip Selecta* proces kontinuiranog čitanja memorije prestaje. Osim kontinuiranog čitanja cjelokupne memorije *flash* memorija nudi mogućnost čitanja pojedine stranice unutar memorije, te čitanje samih SRAM *buffera*. Kada proces čitanja pojedine stranice dođe do kraja stranice, ne prelazi na sljedeću već ponovno čita istu stranicu, samo od početka.

Prilikom čitanja sadržaja vanjske *flash* memorije u konkretnoj izvedbi korištena je operacija čitanja pojedine stranice memorije. Nakon postavljanja *Chip Select*-a u nisko stanje prvo se na SI pin šalje osam bitna naredba. Nakon nje slijede dva *dummy* bita, dvanaest bitna adresa stranice i osam bitna adresa *byte*-a unutar stranice. Trideset i dva *dummy* bita slijede nakon adresa, te time završava prijenos na pinu SI. Nakon toga slijedi osam bitna transmisija s pina SO. Transmisija se može obaviti nebrojeno puta, ali za konkretnu namjenu se obavlja šest puta (korekcijski faktori za tri struje i tri napona).



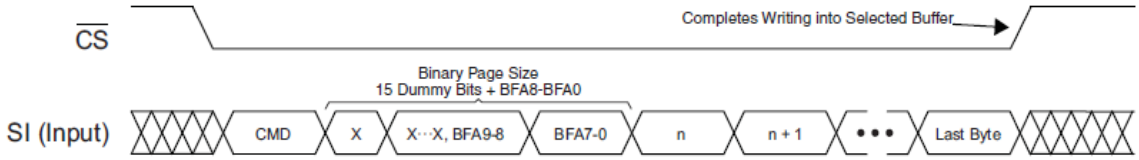
Slika 3.7. Operacija čitanja stranice memorije, [8] str. 57.

```
void read_mem(uint16_t page, uint16_t byte, int quant)  //quant=6
{
	unsigned char Char1;		// nizi byte
	unsigned char Char2;		// visi byte
	uint8_t i;
	set_cs();

	write_SSP1(0xD2);		 //op code
	i=(page>>6)&(0x3F);
	write_SSP1(i);			//2 dummya, 6 MSBs page add
	i=((page<<2)&(0xF3))|((byte>>8)&(0x03));
	write_SSP1(i);			//6 LSBs page add, 2 msb byte add

	i=byte&0xFF;
	write_SSP1(i);			//8 LSBs byte add
	write_SSP1(0xFF);		//
	write_SSP1(0xFF);		//
	write_SSP1(0xFF);		 //
	write_SSP1(0xFF);		//4 dummy bytea
	read_SSP1();			//ciscenje recieve buffera od nepotrebnog podatka

	for(i=0;i<quant;i++) {
			Char1= read_SSP1();
			Char2= read_SSP1();
			devParams.corr[i]=(Char2 << 8) | Char1;
			}
	clr_cs();
}
```

Isječak koda 3.7. Funkcija čitanja korekcijskih faktora s jedne stranice memorije

Pisanje/brisanje memorije moguće je izvesti na različite načine. Postoje posebne operacije upisivanja u *buffer*, operacije koje upisuju podatke iz *buffera* u glavnu memoriju (sa i bez brisanja prethodno postojećih podataka) i operacije koje izravno upisuju podatke u stranice/byteove (ponovno sa i bez brisanja prethodno postojećih podataka na željenim lokacijama). Kao što je napomenuto u poglavlju 2.6. memorijski prostor može biti adresiran stranicama, blokovima i sektorima. Takva adresiranja koriste se kod tri različita načina brisanja memorije.

Upis u memoriju ostvaren je koristeći dvije operacije. Prva operacija je upis u *buffer1*.



Slika 3.8. Prikaz upisa u *buffer* u slučaju stranice veličine 512 *byte*-a (uglata zagrada), te 528 *byte-a* (dijagram), [8] str. 54.

Operacija počinje slanjem osam bitne naredbe nakon koje slijede četrnaest *dummy* bitova, te deset bitova koji označavaju adresu unutar *buffer*-a od koje želimo zapisivati podatke u *buffer*. Ako se prilikom upisa popuni cijeli buffer, sljedeći upisani *byte* bi bio smješten na početak *buffer*-a, odnosno prvu adresu.

```
void write_buff1(uint16_t add)
{
        int i;
        unsigned char Char1;     // nizi byte
        unsigned char Char2;    // visi byte
        set_cs();

        write_SSP1(0x84);        //op.code
        write_SSP1(0x00);        //8 dummy

        i=(add>>6)&(0x3);        // 6 dummy+2 LSB-a adrese za lokaciju u bufferu
        write_SSP1(i);

        i=add & (0xFF);             // 8 LSB-a adrese
        write_SSP1(i);

        for(i=0;i<6;i++) {
                Char1 = devParams.corr[i] & 0xFF;
                Char2 = devParams.corr[i] >> 8;

                write_SSP1(Char1);
                write_SSP1(Char2);
        }
        clr_cs();
}
```

Isječak koda 3.8. Funkcija pisanja u *buffer*

Nakon upisa u *buffer* potrebno je prebaciti podatke iz *buffer*-a u glavnu memoriju. Takvo prebacivanje moguće je ostvariti na dva načina. Duži način je prebacivanje iz *buffer*-a u memoriju bez brisanja, zbog toga što zahtjeva jednu operaciju više (brisanje). Kraći, a samim time i korišten postupak je prebacivanje iz *buffer*-a u glavnu memoriju s ugrađenim brisanjem. Operacija kreće osam bitnom komandom, nakon koje slijedi dva *dummy* bita, dvanaest bitna adresa, te se operacija zaključuje s deset *dummy* bitova.

```
void write_buff1_into_main(uint16_t add)
{
        uint8_t i;

        set_cs();

        write_SSP1(0x83);       //  op code

        i=(add>>6)&(0x3F);      //2 dummy + 6 addrese pagea
        write_SSP1(i);

        i=(add&0x3F)<<2;        //6 addrese pagea + 2 dummy
        write_SSP1(i);

        write_SSP1(0x00);       //8 dummy

        clr_cs();
}
```

Isječak koda 3.9. Funkcija za prijenos podataka iz *buffer*-a u stranicu memorije

Prilikom korištenja prethodno definiranih funkcija potrebno je voditi računa o *Status Register*-u memorije. Zauzetost kontrolera sa strane mikrokontrolera tokom prijenosa pojedinog *byte*-a provjerena je unutar naredbi za pisanje i čitanje preko SSP1, ali također provjeravati zauzetost ili grešku sa strane same memorije.

Bit	Name		Type[1]	Description	
7	RDY/$\overline{\text{BUSY}}$	Ready/Busy Status	R	0	Device is busy with an internal operation.
				1	Device is ready.
6	COMP	Compare Result	R	0	Main memory page data matches buffer data.
				1	Main memory page data does not match buffer data.
5:2	DENSITY	Density Code	R	1011	16-Mbit
1	PROTECT	Sector Protection Status	R	0	Sector protection is disabled.
				1	Sector protection is enabled.
0	PAGE SIZE	Page Size Configuration	R	0	Device is configured for standard DataFlash page size (528 bytes).
				1	Device is configured for "power of 2" binary page size (512 bytes).

Bit	Name		Type[1]	Description	
7	RDY/$\overline{\text{BUSY}}$	Ready/Busy Status	R	0	Device is busy with an internal operation.
				1	Device is ready.
6	RES	Reserved for Future Use	R	0	Reserved for future use.
5	EPE	Erase/Program Error	R	0	Erase or program operation was successful.
				1	Erase or program error detected.
4	RES	Reserved for Future Use	R	0	Reserved for future use.
3	SLE	Sector Lockdown Enabled	R	0	Sector Lockdown command is disabled.
				1	Sector Lockdown command is enabled.
2	PS2	Program Suspend Status (Buffer 2)	R	0	No program operation has been suspended while using Buffer 2.
				1	A sector is program suspended while using Buffer 2.
1	PS1	Program Suspend Status (Buffer 1)	R	0	No program operation has been suspended while using Buffer 1.
				1	A sector is program suspended while using Buffer 1.
0	ES	Erase Suspend	R	0	No sectors are erase suspended.
				1	A sector is erase suspended.

Slika 3.9. *Status Register*, *byte*-ovi 1 i 2, [8] str. 27./28.

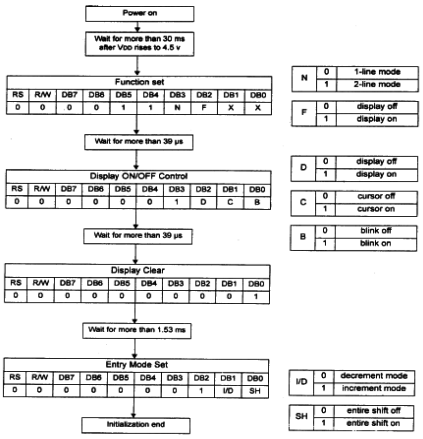
Status Register potrebno je čitati nakon svakog upisa ili čitanja kako bi se omogućio ispravan rad *flash* memorije.

```
int flash_SR() {
                    uint8_t reg,i;
                    set_cs();
                    write_SSP1(0xD7);
                    read_SSP1();
                    reg=read_SSP1();        //prvi byte
                    if(reg==0xAD)
                            i=1;             //512 byteova
                    else if(reg==0xAC)
                            i=2;             //528 byteova
                    else
                            i=0;
                    if(i==0) {
                            clr_cs();
                            return i;
                            }
                    reg=read_SSP1();         //drugi byte

                    if(reg==0x88)
                            i=i;
                    else
                            i=0;
                    clr_cs();
                    return i;
}
```

Isječak koda 3.10. Provjera *Status Register*-a

Pisanje u memoriju vrši se tokom umjeravanja, kada se izračunati kalibracijski faktori pohranjuju u *flash* memoriju. Čitanje iz memorije odvija se tokom inicijalizacije modela, te se korekcijski parametri spremaju u globalnu varijablu za daljnje korištenje.

3.3.4 LCD prikaznik i tipkovnica



Slika 3.10. Rutina inicijalizacije LCD modula, [4] str. 12.

Kako bi LCD modul bio ispravno inicijaliziran prilikom uključenja potrebno je proći kroz rutinu inicijalizacije danu unutar uputa vezanih za LCD modul.

```
void display_init(void)
{
  uint8_t i;

   ms_delay(250);

  LPC_PINCON->PINSEL0 &=   0x3FFFFFFF;        // P0.15 (DB0) kao GPIO
  LPC_PINCON->PINSEL1 &= (~0x00003FFF);       // P0.16 - P0.22 kao GPIO
  LPC_PINCON->PINSEL4 &= (~(0x3F << 14));     // P2.7 - P2.9 kao GPIO

  LPC_PINCON->PINSEL1 &=  ~(0x3  << 14);
  LPC_GPIO0->FIODIR |= DISP_LED;
  LPC_GPIO0->FIOSET  = DISP_LED;


  LPC_GPIO0->FIODIR |= LCD_DATA;            // postavi LCD_DATA pinove kao izlazne

  LPC_GPIO2->FIODIR |= LCD_CTRL;            // postavi LCD_CTRL pinove kao izlazne
  LPC_GPIO2->FIOCLR  = LCD_CTRL;            // postavi LCD_CTRL pinove u nulu

  ms_delay(1);

  // 2nd init instruction + NFXX (N = lne mode, F = Display OCTRL)
  for(i = 0; i < 3; ++i)
  {
    lcd_instr(0x3C, 0);   //2line, disp on
    ms_delay(5);
  }
```

```
  // Display ON/OFF Control; lower nibble == 1DCB (Display, Cursor, Blink)
  lcd_instr(0x0C, 0);          //display on, cursor off, blink off
  ms_delay(2);

  // ocisti prikaz
  lcd_clear();
  ms_delay(2);

  // Entry Mode Set; lower nibble == 1 I/D SH X (DEC/INC mode, shift ON/OFF)
  lcd_instr(0x06, 0); //inc mode, shift off
  ms_delay(5);


  lcd_print("   PM   ", "       ");
}
```

Isječak koda 3.11. Implementacija inicijalizacijske rutine LCD modula

Pinovi P0.15-P0.22 korišteni su kao podatkovni pinovi LCD modula. Pinovi P2.7-P2.9 korišteni kao kontrolni pinovi, P2.7 kao *read/write select*, P2.8 kao *enable* te P2.9 kao *data/instruction select.* Funkcije lcd_instr(), lcd_clear() i lcd_print neće biti dodatno opisane unutar izvješća.

Prikaz veličina na LCD-u periodično se osvježava, te se pritiskom na tipke mijenja ovisno o želji prikazivanja konkretne veličine. Funkcija promjene prikaza na LCD modulu ostvarena je automatom stanja koji kao ulazne parametre koristi pritisnute tipke, odnosno stanje varijabli lcd.section i lcd.screen. Pritiscima na lijevu tipku povećava se varijabla lcd.section i time se mijenja prikaz THD-a pojedine faze, fazni i linijski naponi i fazne struje. Konačno stanje prije izlaska iz izbornika lijeve tipke je umjeravanje koja je detaljnije opisana u poglavlju numeričke obrade signala. Pritiscima na srednju tipku povećava se varijabla lcd.screen i time se mijenja prikaz snaga, faktora snage te energija. Lijeva tipka služi kako bi se vratili unutar jednog od dva načina prikaza

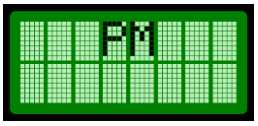
```
void keyboard_init(void)
{
  LPC_PINCON->PINSEL4 &=  ~(0x3F << 6);    // TPK1, TPK2, TPK3 kao GPIO
  LPC_GPIO2->FIODIR   &=  ~(0x07 << 3);    // TPK1, TPK2, TPK3 kao ulazni pinovi
  key.value = 0;
}
void keyboard (void)
{
        switch(key.value)
        {
                case ENTER:
                        if (lcd.screen>0)
                                lcd.section=lcd.section;
                        else
                                lcd.section=(lcd.section+1)%17;
                        break;
                case DOWN:
                        if(lcd.section==0)
                                lcd.section=(lcd.section);
                        else if (lcd.section>0)
                                lcd.section=lcd.section-1;
                        break;
        }
        switch(key.value)
        {
                case UP:
                        if(lcd.section>0)
                                 lcd.screen=lcd.screen;
                        else
                                lcd.screen=(lcd.screen+1)%15;
                        break;
                case DOWN:
                        if(lcd.screen==0)
                                lcd.screen=(lcd.screen);
                        else
                                lcd.screen=lcd.screen-1;
                        break;
}}
```

Isječak koda 3.12. Funkcija inicijalizacije tipkovnice i glavna funkcija tipkovnice

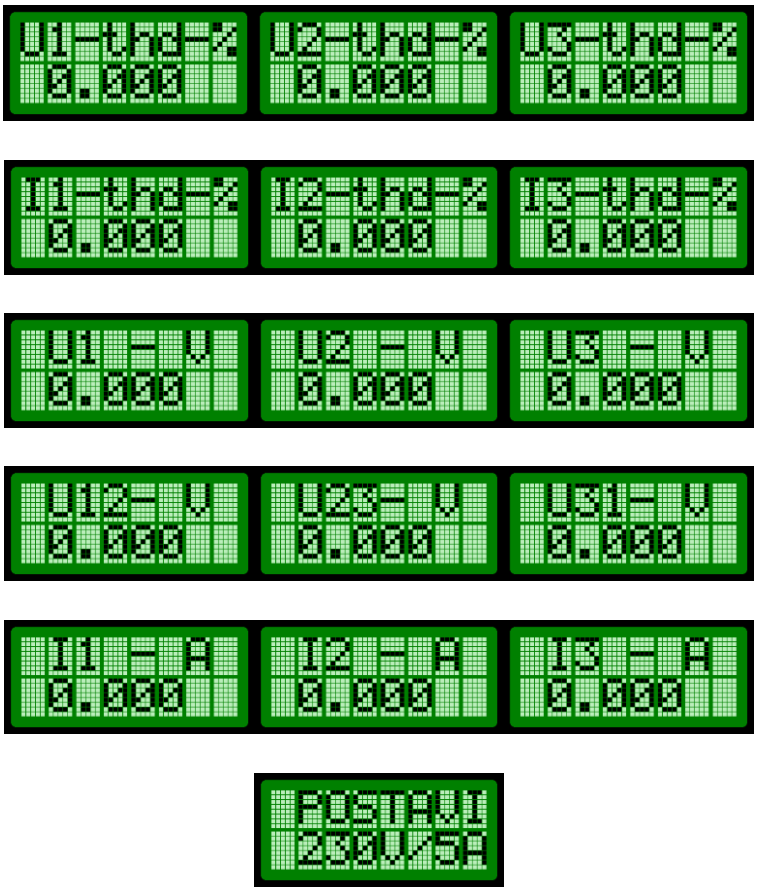
```
void display (void)
{
	if(lcd.section>0) {
		switch(lcd.section)
	{
		case 0:
		display_default_screen(0);
		break;
		case 1:
		display_voltage_thd(0);
		break;
		case 2:
		display_voltage_thd(1);
		break;
		case 3:
		display_voltage_thd(2);
		break;
		case 4:
		display_current_thd(0);
		break;
		case 5:
		display_current_thd(1);
		break;
		case 6:
		display_current_thd(2);
		break;
		case 7:
		display_phase_voltage(0);
		break;
		case 8:
		display_phase_voltage(1);
		break;
		case 9:
		display_phase_voltage(2);
		break;
		case 10:
		display_line_voltage(0);
		break;
		case 11:
		display_line_voltage(1);
		break;
		case 12:
		display_line_voltage(2);
		break;
		case 13:
		display_phase_current(0);
		break;
		case 14:
		display_phase_current(1);
		break;
		case 15:
		display_phase_current(2);
		break;
		case 16:
		calibrate();
		break;
	}
	}
```

```
	else {
	switch(lcd.screen)
	{
		case 0:
		display_default_screen(0);
		break;
		case 1:

	display_apparent_power(0);;
		break;
		case 2:
		display_apparent_power(1);
		break;
		case 3:
		display_apparent_power(2);
		break;
		case 4:
		display_active_power(0);
		break;
		case 5:
		display_active_power(1);
		break;
		case 6:
		display_active_power(2);
		break;
		case 7:
		display_reactive_power(0);
		break;
		case 8:
		display_reactive_power(1);
		break;
		case 9:
		display_reactive_power(2);
		break;
		case 10:

	display_active_power_factor(0);
		break;
		case 11:

	display_active_power_factor(1);
		break;
		case 12:

	display_active_power_factor(2);
		break;
		case 13:
		display_active_energy(0);
		break;
		case 14:
		display_reactive_energy(0);
		break;

	}
}
}
```

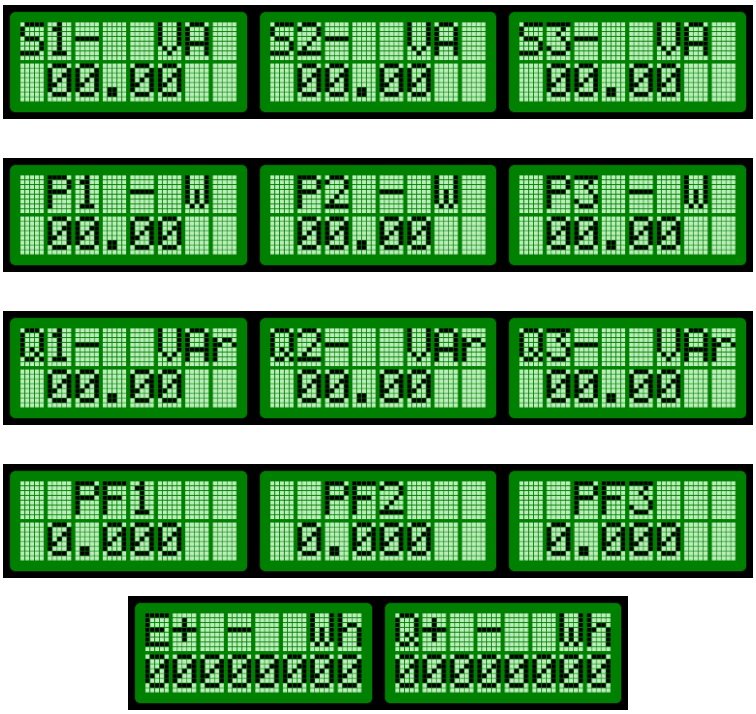
Isječak koda 3.13. Funkcija prikaza veličina na LCD modulu



Slika 3.11. Početni zaslon



Slika 3.12. Slijedni prikazi izbornika lijeve tipke



Slika 3.13. Slijedni prikazi izbornika lijeve tipke

3.4 Prekidne rutine

3.4.1 RIT prekid

Repetetive Interrupt Timer prekid služi za uzorkovanje ulaznih signala. Posebnu pažnju treba posvetiti što kraćem trajanju prekidne rutine kako se ne bi narušila funkcionalnost cijelog modela. Ulazni signali su uzorkovani frekvencijom 3200 Hz, koja je višekratnik osnovne frekvencije signala od 50 Hz. Prekidna rutina je zamišljena tako da prikupi 50 perioda (3200 uzoraka) svakog od ulaznih signala tijekom jedne sekunde. Nakon jedne sekunde kreće računanje nad uzorcima izvan prekidne rutine, dok unutar prekidne rutine prikuplja novih 3200 uzoraka. Interval od jedne sekunde postignut je time što RIT broji uzorke do vrijednosti od 3200, što u sekundama predstavlja jednu sekundu.

```
void rit_init(void)  {
  disable_rit();
  LPC_RIT->RICOMPVAL = 31250; // 312..5 us
  NVIC_SetPriority(RIT_IRQn, 0);  }
```

Isječak koda 3.20. Inicijalizacija RIT-a

```
for(i = 0; i < 8; ++i)                                          // citanje kanala sa AD7927 [200 kSPS]
  {
        sample = read_ADC_SSP0();                  // minimalno 5 us mora biti izmedu dva
                                                                    //uzastopna citanja
        if(((sample & 0xF000) >> 12) != i)  {          // potrebna reinicijalizacija ADC-a, izlaz iz
                                                  //prekidne rutine, (provjera adrese kanala adc-a)
                adc.reinit_adc = 1;
                return;   }
                tmp1=(sample & 0x0FFF)-adc.gnd[7-i];                //korekcija nule
                adc.fft[adc.index + 2 * ((7 - i) * BLOCK)]=tmp1;
                tmp1=adc.last[7-i]*devParams.tau[7-i]+tmp1*(65536-devParams.tau[7-i]);
                tmp1/=65536;                                                //korekcija pomaka u fazi
                i64Temp=tmp1*devParams.corr[7-i];
                i64Temp/=4096;                                             //umjeravanje
                adc.last[7-i]=tmp1;
                adc.val[adc.index + 2 * ((7 - i) * BLOCK)]  = i64Temp;
 }
```

Isječak koda 3.21. Dio prekidne RIT rutine, čitanje s kanala A/D pretvornika i spremanje veličina

Uzorci za FFT (*Fast Fourier Transformation*) spremaju se u zasebno polje. Također FFT se računa nad 256 uzoraka, te prvih 256 uzoraka predstavlja uzorke za FFT. Nad tim uzorcima jedina korekcija koja se primjenjuje je korekcija pomaka nule. Korekcija pomaka nule implementirana je zbrojem prvih 256 uzoraka unutar prekidne rutine, te dijeljenjem s brojem uzoraka i dodavanjem rezultata prethodnom iznosu nule. Dodavanje dobivenog rezultata odvija se nakon pune sekunde kako se ne bi narušilo mjerenje. Inicijalni iznos nule je 2047.

Osim kompenzacije pomaka nule, unutar prekidne rutine se nad ostalim uzorcima vrši i korekcija pomaka faze uzrokovana različitim trenutcima uzorkovanja parova struje i napona. Također uzorci su podvrgnuti množenju s kalibracijskim faktorima. Korekcija pomaka faze i umjeravanje objašnjeni su u poglavlju numeričke obrade signala.

```
if(accum.count<256)
        for(i = 0; i < 6; ++i)
                cal.sum[i] +=(adc.val[adc.index + 2 * i * BLOCK]); //pomak nule-uzorci

//////////////////////////////////////preskocen dio koda////////////////////////////////
if(accum.count == SAMPLES_COUNT)                    // ulazak svakih 3200 uzoraka T = 1 s
  {
    accum.block_select = ~(accum.block_select | 0xFE);
    for(i = 0; i < 3; ++i)
    {
      tmp1 = i + accum.block_select * 3;
      accum.ULE[tmp1]   = 0;                        // cisti sljedeci acc blok!!
      accum.ULL[tmp1]   = 0;
      accum.ILE[tmp1]   = 0;
      accum.P[tmp1]     = 0;
    }
    for(j = 0; j < 6; ++j)                          //računanje pomaka nule
        {
        tmp2=cal.sum[j]/256;
        tmp3=cal.sum[j] % 256;
        tmp1=adc.gnd[j] + tmp2;
        if((tmp3>=128))
         adc.gnd[j] = tmp1 + 1;
        else if((tmp3<=-128))
         adc.gnd[j] = tmp1 - 1;
        else
          adc.gnd[j] = tmp1;
        if(adc.gnd[j] > 2048 + 200)
          adc.gnd[j] = 2048 + 200;                   //limit
        else if(adc.gnd[j] < 2048 - 200)
          adc.gnd[j] = 2048 - 200;                  //limit
        cal.sum[j]=0;
        }
        accum.data_ready = 1;                 // podaci su spremi za racunanje RMS vrijednosti
        accum.count=0;
}
```

Isječak koda 3.22. Prikaz kompenzacije pomaka nule unutar RIT prekida

3.4.2 TIMER0 prekid

Prilikom inicijalizacije frekvencije takta mikrokontrolera u sistemskoj datoteci također je moguće odabrati frekvenciju takta pojedinih periferija mikrokontrolera. U slučaju Timer0, uz pomoć registra PCLKSEL0, frekvencija takta namještena je na frekvenciju takta mikrokontrolera, odnosno 100 MHz.

```
void timer0_init(void)
{
  LPC_TIM0->IR = 0x0F;              // ocisti MATCH interrupt flagove za kanale 0...3
  LPC_TIM0->PR  = (100000000 / 1000000) - 1;        // 100tickova na 100MHz (1us)
  LPC_TIM0->MR0 = 2000;         // (2000 * 1 us)=2ms   MATCH registar
  LPC_TIM0->MCR = 0x3;          // interrupt i reset na MR0 (MATCH ctrl registar)
  NVIC_SetPriority(TIMER0_IRQn, 1);  // drugi najvisi prioritet
}
```

Isječak koda 3.23. Inicijalizacije Timer0 periferije

Inicijalizacija Timer0 realizirana je na način da se brojač inkrementira svaku mikrosekundu te kad izbroji do vrijednosti od dvije tisuće (što odgovara vremenu od dvije milisekunde) dogodi se prekid i resetira se vrijednost brojača.

```
void TIMER0_IRQHandler (void)
{
  time.tick += 2;
  time.tick_value += 2;
  time.tcpip_timer += 2;
          time.pass+=2;

  if (time.tcpip_timer > 8)
  {
    timer_tick();                    /*          posluzivanje TCP/IP stacka          */
    time.tcpip_timer = 0;
  }
  LPC_TIM0->IR = 0x1;
}
```

Isječak koda 3.24. Prekidna rutina Timer0

Prekidna rutina Timer0 ima dvije svrhe. Prva je brojanje milisekundi za *debounce* tipki. Druga je određivanje intervala posluživanja TCP/IP *stack*-a. Interval definiran unutar NET_CONFIG.C mora odgovarati intervalu između pozivanja funkcije timer_tick(). Interval za posluživanje TCP/IP *stack*-a je 10ms i tako je definiran na oba mjesta. Prioritet nije najviši zbog toga što se prednost daje mjerenju, odnosno RIT prekidu, radi potrebe preciznosti vremenskih razmaka između uzoraka.

3.5 Web (HTTP) poslužitelj

3.5.1 Protokoli komunikacija

Ethernet (ili bolje rečeno *Ethernet* II) je najčešći oblik mrežnog prijenosa u LAN-ovima (*Local Area Network*). Zaglavlje *Ethernet* paketa sadrži sinkronizacijsku preambulu, nakon koje slijede adrese izvora i odredišta, te polje dužine koje označuje veličinu podatkovnog paketa. *Ethernet* paket je transportni mehanizam za TCP\IP podatke tokom prijenosa LAN-om.

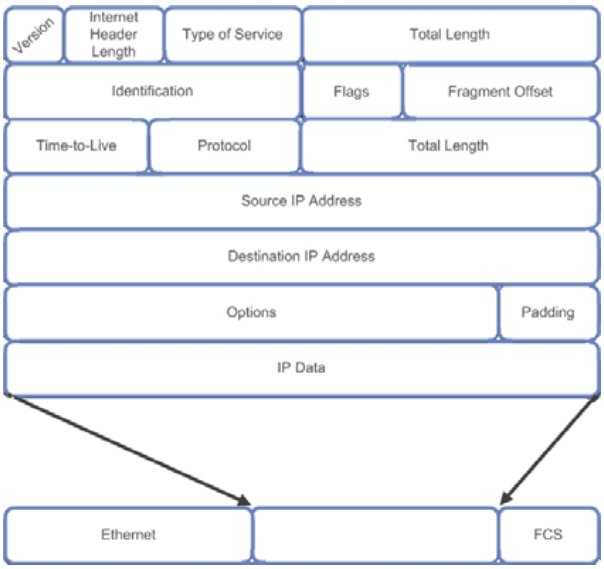
Slika 3.15. Ethernet paket, [9] str. 65.

Polje podatkovnih byte-ova mora biti veličine od 46 do 1500. Konačno polje u dana paketu je *Frame Sequence Check*, što je zapravo ciklička provjera redundancije (CRC). Takva ciklička provjera redundancije omogućava provjeru paketa od početka adrese odredišta do kraja podatkovnog polja.

Za slanje i primanje podataka kroz čvorove mreže, polje podatkovnih *byte*-ova *Ethernet* paketa sadrži TCP/IP datagram. *Layer2* okvir enkapsulira TCP/IP datagram.

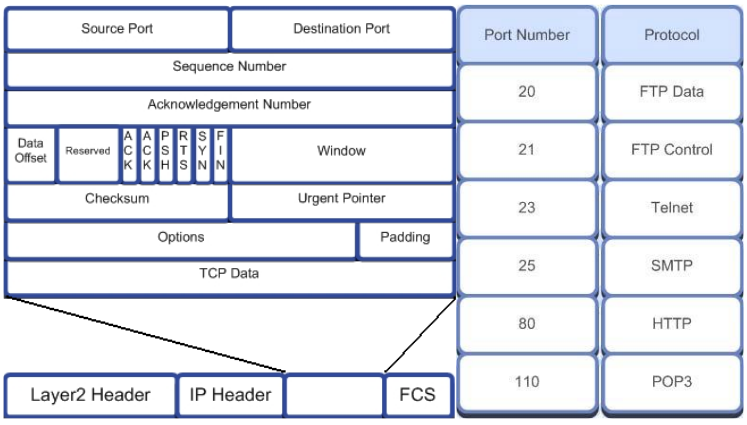
Slika 3.16. Enkapsulirani TCP/IP datagram, [9] str. 65.

Internet Protocol se koristi za prijenos podataka između dvije logičke IP adrese. IP paketi mogu se izgubiti, kasniti ili biti duplicirani. Uzrok tomu je to što nema upravljanja tokom podataka. IP protokol omogućava prijenosni mehanizam za slanje između dva čvora na TCP/IP mreži. Zaglavlje IP-a sadrži izvornu i odredišnu IP adresu. IP adresa je 32-bitni broj koji se koristi kako bi se jedinstveno definirao čvor unutar interneta. IP adresa neovisna je o fizičkoj adresi mreže. Kako bi IP paketi bili dostavljeni odredištu potreban je algoritam otkrivanja kako bi se uskladila IP adresa s fizičkom adresom. Na lokalnoj mreži se za takvu namjenu koristi ARP (*Address Resolution Protocol*).



Slika 3.17. IP paket, [9] str. 66.

TCP protokol je izveden tako da je sadržan unutar podatkovnog polja IP paketa. IP paket omogućava transportni mehanizam. TCP datagram omogućava logičku povezanost između računala i aplikacijskog sučelja. IP adresa koristi adresu odredišnog računala. TCP koristi izvorni i odredišna vrata (*port*), omogućava provjeru grešaka, fragmentaciju velikih poruka, i potvrdu pošiljatelju. Mehanizam potvrde i ponovnog slanja unutar TCP-a koristi metodu pomičnog prozora. Takva metoda iziskuje više *buffer*-a za držanje podataka koje je potencijalno potrebno ponovno poslati. Takva mogućnost otežava implementaciju manjih TCP/IP stogova, a glavni razlog je korisnički RAM. Broj mrežnih vrata TCP-a (*TCP port number*) povezuje podatkovni dio TCP-a s ciljanom aplikacijom. Standardne TCP/IP aplikacije imaju poznate brojeve vrata zbog toga da se udaljeni klijenti mogu lakše spajati na standardne usluge.

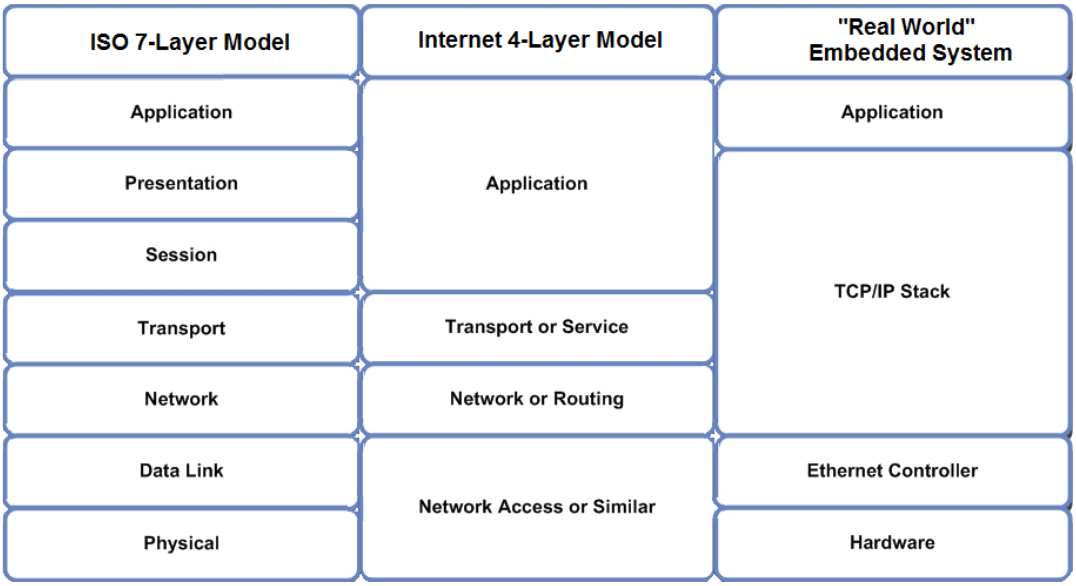


Slika 3.18. Lijevo: TCP, desno: tipizirani brojevi mrežnih vrata (*portova*), [9] str. 69.

3.5.2 Uvod u RL-TCPnet

Povezivanje pomoću *Ethernet*-a, odnosno uspostava *web server*-a, potpomognuta je RL-TCPnet *middleware*-om, koji je dio paketa RL-ARM, kojeg razvojno okruženje *KEIL* nudi svojim korisnicima kako bi lakše implementirali kompleksnija, ali tipska programska rješenja.

TCP/IP je mrežni model razdijeljen na četiri sloja koji se protežu kroz sedam slojeva ISO modela. Pristupni mrežni sloj sadrži fizičku vezu s mrežom, pakiranje podataka iz konkretne aplikacije u pakete i kontrolu protoka paketa kroz mrežu. U klasičnom mikrokontrolerskom sustavu PHY čip i *Ethernet* kontroler predstavljaju pristupni mrežni sloj, dok TCP/IP stog vodi računa o mrežnom i prijenosnom sloju.



Slika 3.14. Usporedba ISO, TCP/IP, mikrokontrolerski sustav, [9] str. 64.

Mrežni sloj upravlja prijenosom podatkovnih paketa između mrežnih postaja koristeći *Internet Protocol* (IP). Prijenosni sloj omogućava vezu između aplikacijskih slojeva različitih postaja te koristi jedan od dva protokola; UDP (*User Datagram Protocol*), TCP (*Transmission Control Protocol*). Aplikacijski sloj omogućava pristup komunikacijskoj okolini kroz korisničko sučelje. Sam pristup definiran je protokolima aplikacijskog sloja kao što su HTTP, Telnet, SMTP.

Iako se ponekad na manjim mikrokontrolerima implementira samo dio TCP/IP stoga, RL-TCPnet je potpuna implementacija koja omogućava punu funkcionalnost internet postaje na mikrokontroleru.

3.5.3 Uvod u implementaciju web (HTTP) poslužitelja

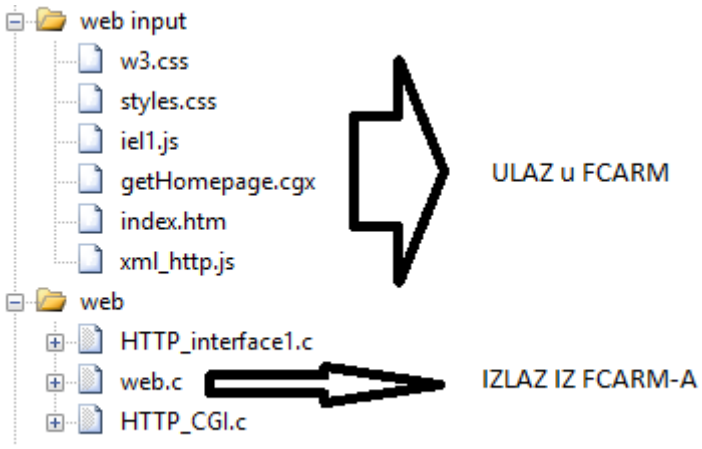
Unutar poglavlja naglasak će biti na predefiniranoj programskoj podršci (i njenom opisu) za realizaciju *web* poslužitelja koja stoji na raspolaganju unutar programskog paketa RL-ARM i načinu dodavanja ostalih datoteka potrebnih za konkretan slučaj.

HTTP ili *web* poslužitelj obrađuje zahtjeve kroz HTTP (*Hypertext Transfer Protocol*), što je mrežni protokol korišten za prijenos informacija na *World Wide Web*-u (WWW). Glavna svrha HTTP servera je spremanje, obrada i slanje *web* stranica klijentu. Dostavljene stranice su obično HTML (*Hypertext Markup Language*). Koristeći HTML opisuje se kako će stranica izgledati. Koristeći SSL (*Secue Sockets Layer*)/TLS (*Transport Layer Security*) moguće je ostvariti sigurniju vezu s HTTP poslužiteljem putem HTTPS-a.

Postoje dvije vrste *web* stranica koje su spremljene na *web* poslužitelju i slane *web* klijentu. Statične *web* stranice ne mijenjaju svoj sadržaj. Dinamične *web* stranice generirane su kad se zatraži takva stranica ili su generirani dijelovi stranice. Pomoću *JavaScript*-a generira se dinamični sadržaj *web* stranice. Primjer dinamične *web* stranice obrađen je u sklopu rada.

HTTP poslužitelj koji je djelomice preuzet iz RL-ARM-a posjeduje CGI (*Common Gateway Interface*) koji omogućava slanje i primanje podataka od mikrokontrolerske C aplikacije.

Sadržaj *web* poslužitelja može biti bilo koji format kojeg je moguće prikazati *web* preglednikom. Ograničavajuća okolnost je jedino memorija mikrokontrolera. Sve datoteke koje opisuju *web* stranicu dodajemo u projekt kroz alat za pretvorbu datoteka FCARM. FCARM nalazi se unutar programskog paketa MDK-ARM. Prilikom dodavanja datoteka pomoću FCARM-a potrebno ih je definirati kao *Image file*, omogućiti *Image File Compression*, definirati izlaznu datoteku (.C datoteka) i njenu lokaciju, definirati korijensku mapu svih ulaznih datoteka te omogućiti *Generate Listing*. Prilikom izrade projekta FCARM stvara naredbenu datoteku Auto_FCArm_Cmd.inp u projektnoj mapi koja sadrži datoteke i naredbe za pretvorbu datoteka. Prozor izlaza GRADNJE pokazuje sastavljanje *image* datoteka.



Slika 3.19. Raspored ulaznih i izlaznih datoteka FCARM-a

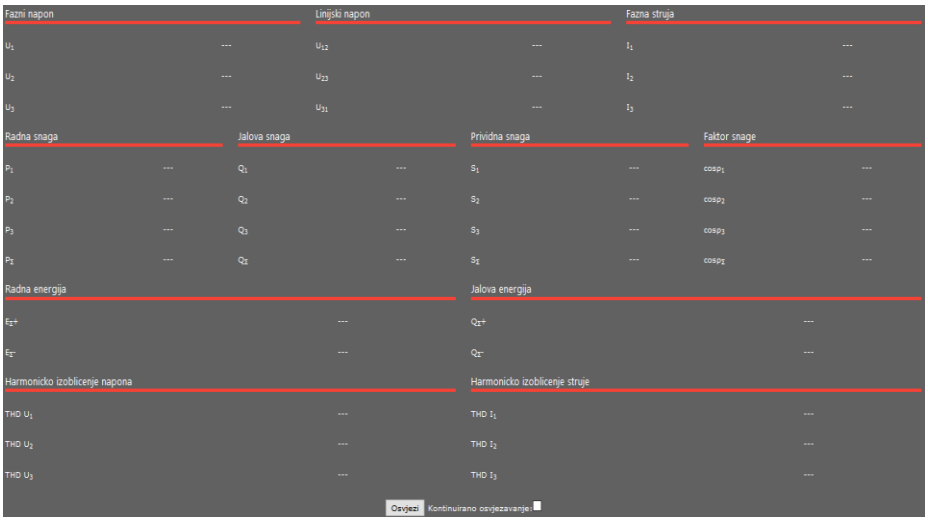
CGI protokol (*Common Gateway Interface*) omogućava izradu dinamičkih *web* stranica. Radi na principu dohvaćanja korisnikovih naredbi i slanja istih aplikaciji koja se nalazi na mikrokontroleru. Također omogućava slanje podataka iz aplikacije kroz TCP/IP server. Dodavanje HTTP_CGI.C (C:\KEIL\ARM\RL\TCPNET\SRC) datoteke omogućava CGI sučelje. Navedena datoteka povezuje događaje na *web* poslužitelju s .C aplikacijom. CGI sučelje izvedeno je kao skriptni jezik.

Skriptni jezik je jednostavan te svaka linija kreće s jednom od pet naredbi. Naredba „i“ služi za uključivanje i dodavanje datoteke iz virtualnog sustava datoteka na *web* preglednik. Naredbu „t“ slijedi tekst koji će bit prikazan na *web* pregledniku. Nakon naredbe „c“ slijedi određeni tekst koji je proslijeđen funkciji cgi_func(). Tekst služi kao naredba pomoću koje definiramo zahtjev. Unutar funkcije cgi_func() potrebno je odrediti što se treba odviti za određeni slučaj skriptnog jezika nakon „c“ naredbe. Naredba „#“ predstavlja komentar dok „.“ predstavlja zadnju liniju skripte.

Skriptni jezik koristi se i kod AJAX podrške koja služi kako bi dohvaćali željene podatke s mikrokontrolera i prosljeđivali ih *JavaScript* objektima. Za razliku od izvedbe s .cgi datotekom koja predstavlja potpunu *web* stranicu, stranica prilikom korištenja XML datoteke s ekstenzijom .cgx (koja sadrži isključivo liniju s skriptnom naredbom „c“) može biti obična HTML stranica uz dodatak *JavaScript* skripte koja upravlja dinamičnim dijelom stranice. Upravo je na taj način ostvarena konkretna *web* stranica. Prednost ovakve izvedbe je prijenos isključivo dinamičkih podataka unutar jednog TCP paketa, što omogućava brz prijenos koji koristi uzak frekvencijski spektar. Time je izbjegnuto osvježavanje cijele stranice čime se uklanja „titranje“ stranice prilikom osvježavanje i slanje velike količine statičnih podataka.

Osim omogućavanja HTTP servera, potrebno je omogućiti i *Ethernet* kontroler, te konfigurirati parametre mreže. U sklopu RL-ARM paketa sadržan je EMAC_LPC17xx.C koji predstavlja kod za cjelokupni rad *Ethernet* kontrolera sa strane mikrokontrolera. Unutar koda podržana je potpuna komunikacija između kontrolera i PHY čipa. Konfiguracija parametara mreže odvija se uz pomoć NET_CONFIG.C datoteke. Parametri koje je moguće postaviti pomoću *Configuration Wizard*-a su među ostalima ime poslužitelja, interval sistemskog brojača, MAC adresa, IP adresa i IP adresa mrežnog pristupnika.

3.5.4 Implementacija *web* (HTTP) poslužitelja



Slika 3.20. Izgled *web* stranice

HTML stranica izrađena je uz pomoć w3.css datoteke unutar koje je definirano mnošto stilova za različite prikaze. Iz datoteke w3.css preuzete su tablice. Unutar datoteke styles.css definirana je boja pozadine, font te sekundarna boja.

Tablice su zapravo redovi koji se dijele ovisno o broju stupaca za određeni prikaz. Kako bi bilo lakše pristupati lokacijama unutar stranice koje predstavljaju dinamičke podatke, potrebno je tim lokacijama dodijeliti „ID“, odnosno identifikacijsku oznaku.

Dinamički prikaz s korisničke strane odvija se pritiskom na gumb osvježi (koja momentalno obnovi prikaz svih veličina) te postavljanjem kvačice u kvadrat koji predstavlja kontinuirano osvježavanje. Kontinuirano osvježavanje odvija se svake dvije sekunde. Implementacija gumba i kvadrata realizirana je pomoću *JavaScript* skripte unutar .htm datoteke, uz potporu *JavaScript* skripta koje su uključene u .htm datoteku.

```
<script>
var periodicObjects  = [new periodicObj("getHomepage.cgx", 1000)];
var interval;
function refreshOnce() {
updateMultiplePass(periodicObjects[0]); }
function refreshMul() {
var x=document.getElementById("refreshChkBox").checked;
if (x==true) {
interval = setInterval(function(){refreshOnce();},2000); }
else {
clearInterval(interval); }}
</script>
```

Isječak koda 3.14. Skripta unutar index.htm

GetHomepage.cgx predstavlja periodični objekt koji sadrži jednu komandu pisanu skriptnim jezikom objašnjenim u prethodnom poglavlju. Ukoliko je potrebno koristiti više različitih naredbi najbolje je definirati više .cgx datoteka. U konkretnom slučaju gumb za osvježavanje i kontinuirano osvježavanje obavlja istu funkciju, samo je pristup vremenu različit. Pomoću funkcije refreshOnce() realizan je gumb, dok funkcija refreshMul() predstavlja implementaciju kvadratića za kontinuirano osvježavanje.

```
function updateMultiplePass(formUpd, callBack, userName, userPassword) {
 xmlHttp = GetXmlHttpObject();    //generiranje http zahtjeva
 if(xmlHttp == null) {
                        alert("XmlHttp not initialized!");
                        return 0; }
xmlHttp.onreadystatechange = responseHandler;   //obrada odgovora
 xmlHttp.open("GET", formUpd.url, true, userName, userPassword);
 xmlHttp.send(null);

 function responseHandler(){      //funkcija za obradu odgovora
  if(xmlHttp.readyState == 4)
  {
   if(xmlHttp.status == 200)
   {
    var data = xmlHttp.responseXML;
    processResponse(data);              // funkcija processResponse obradjuje ispravan odgovor
   }
   else if(xmlHttp.status == 401)
    alert("Error code 401: Unauthorized");
   else if(xmlHttp.status == 403)
    alert("Error code 403: Forbidden");
   else if(xmlHttp.status == 404) ("Error code 404: Not Found"); }}}
```

Isječak koda 3.15. updateMultiplePass() funkcija

Funkcija koja je ključna za obje spomenute opcije je updateMultiplePass(). Spomenuta funkcija nalazi se unutar datoteke xml_http.js. Služi za generiranje zahtjeva, obradu odgovora te u slučaju da je odgovor iskoristiv poziva funkciju processResponse() koja služi za obradu primljenih podataka. Funkcija processResponse() nalazi se unutar xml_http.js datoteke.

```
function processResponse(xmlDoc) {        // procesiraj xml fajlu koja je dosla s uC-a
  var response = xmlDoc.getElementsByTagName("t");   // uz pomoc DOM-a (getelements...)
  //dobivamo što smo s uC poslali,  podatke smo odvojili s <t><t\>
  //response su upravo podaci za homepage
    setHomepageValues(response);
}
```

Isječak koda 3.16. Funkcija za obradu odgovora

Funkcija processResponse() poziva funkciju setHomepageValues() koja se nalazi unutar datoteke iel1.js, te ona odrađuje dodjelu primljenih vrijednosti željenim dijelovima *web* stranice.

```
function setHomepageValues(rows) {
  var types = ["U1", "U2", "U3", "U12", "U23", "U31", "I1", "I2", "I3", "P1", "P2", "P3", "Ptot",
"Q1", "Q2", "Q3", "Qtot", "S1", "S2", "S3", "Stot", "cos1", "cos2", "cos3", "costot", "E+", "E-",
"Q+", "Q-", "ThdU1", "ThdU2", "ThdU3", "ThdI1", "ThdI2", "ThdI3"];

  for (var i = 0; i < types.length; ++i) {
    var row = rows[ i].childNodes[0].nodeValue;  /* dom pristupanje podataka*/
    var splits = row.split(";");     /* .split služi za dijeljenje jednog polja u više manjih polja */

    var element = document.getElementById("id" + types[i]); /* //koje elemente želimo */
    element.innerHTML = formatValue(splits[1], splits[0]); /* //u te elemente upisujemo,
splits0 je prefiks (k,M,G) */
  }
}
```

Isječak koda 3.17. Dodjela primljenih vrijednosti dijelovima stanice

Prikazano je kako teče primanje podataka s mikrokontrolera i njihovo prikazivanje. Pripremanje podataka za slanje s mikrokontrolera odvija se uz pomoć funkcija unutar datoteke HTTP_interface1.C. Funkcije služe za punjenje *buffer*-a koji služi za HTTP odgovor.

```
strncpy((char *) &buf[http->out_length], "<t>", 3);
strncpy((char *) &buf[http->out_length + 3],       (const char *) _line, size);//KOPIRANJE LINE-A
                                                                  //U HTTP OUTPUT BUFFER
strncpy((char *) &buf[http->out_length + 3 + size], "</t>", 4);
```

Isječak koda 3.18. Odvajanje vrijednosti za slanje s <t> i <\t>

Isječak koda 3.18. prikazuje isječak iz funkcije append_data(), funkcije koja je ključna u funkcijama web_append_selected_energies() i web_append_selected_values(). Te funkcije se koriste unutar datoteke HTTP_CGI.C kao odgovor na zahtjev koji sadržava .cgx naredbu koju upravo funkcija cgi_func() unutar datoteke HTTP_CGI.C obrađuje.

```
U16 cgi_func (U8 *env, U8 *buf, U16 buflen, U32 *pcgi) {
  HTTP_REQUEST http_req;
  memset((void *) &http_req, 0, sizeof(http_req));

   http_req.buf = &buf;
   http_req.pcgi = &pcgi;
   http_req.max_length = buflen;

   switch (env [0]) {
          case 'a':
                    switch (env[2]) {
                                        case 'a':

          web_append_selected_values(&http_req, 0, 3, (float*)&rms.ULE, "V");
          web_append_selected_values(&http_req, 0, 3, (float*)&rms.ULL, "V");
          web_append_selected_values(&http_req, 0, 3, (float*)&rms.ILE, "A");
          web_append_selected_values(&http_req, 0, 4, (float*)&rms.P, "W");
          web_append_selected_values(&http_req, 0, 4, (float*)&rms.Q, "VAr");
          web_append_selected_values(&http_req, 0, 4, (float*)&rms.S, "VA");
          web_append_selected_values(&http_req, 0, 4, (float*)&rms.Pf, "");
          web_append_selected_energies(&http_req, 12, 13, (int*)&rms.energies, "Wh");
          web_append_selected_energies(&http_req, 13, 14, (int*)&rms.energies, "Wh");
          web_append_selected_energies(&http_req, 14, 15, (int*)&rms.energies, "VArh");
          web_append_selected_energies(&http_req, 15, 16, (int*)&rms.energies, "VArh");
          web_append_selected_values(&http_req, 0, 3, (float*)&rms.thd_ULE, "%");
          web_append_selected_values(&http_req, 0, 3, (float*)&rms.thd_ILE, "%");
                                        }
                              break;
                    }
          return (U16) http_req.out_length;
 }
```

Isječak koda 3.19. Funkcija cgi_func()

Isječak koda prikazan je s dvije varijable jer je takva i naredba u .cgx datoteci. Mogla se koristiti i jedna varijabla, ali čest je slučaj da je potrebno više varijabli kako bi se definirao svaki slučaj i funkcija koju *web* server ima.

U slučaju korištenja MDK5, potrebno je koristiti *legacy pack* za MDK4. MDK5 omogućava ovakvu izvedbu *web* poslužitelja isključivo za RTOS (*Real-Time Operating System*) izvedbu.

3.6 Numerička obrada signala trofazne niskonaponske mreže

3.6.1 Račun RMS vrijednosti linijskih i faznih napona, struja i skaliranje mjerenih veličina

Izrazi za mjerene efektivne vrijednosti napona i struja:

$$U_{MJ} = \sqrt{\frac{1}{N}\sum_{0}^{3199} u_{MJ}^2(n)} \tag{3.2}$$

$$I_{MJ} = \sqrt{\frac{1}{N}\sum_{0}^{3199} i_{MJ}^2(n)} \tag{3.3}$$

Naponski ulazi u trofaznom NN sustavu su fazni naponi nazivnog iznosa U_N = 230Vef. Dozvoljeni mjerni opseg mora obuhvatiti veće područje, do 300Vef ≈ 1,3 U_N. Naponski ulazi obrađuju se u naponskom prilagodnom sklopu. Prijenosni odnos prilagodnog naponskog sklopa dobivamo pomoću izraza za pojačanje diferencijalnog pojačala definirano unutar opisa naponskog prilagodnog sklopa, a iznosi 300 Vef → ~ 2Vpp.

Strujni ulazi u trofaznom NN sustavu su fazne struje nazivnog iznosa I_N = 5Aef. Dozvoljeni mjerni opseg mora obuhvatiti veće područje do 6Aef = 1,2 I_N. Strujni ulazi obrađuju se u sklopu za strujno naponsku pretvorbu. Prijenosni odnos prilagodnog strujno naponskog sklopa dobivamo pomoću izraza za pojačanje operacijskog pojačala definirano unutar opisa strujno-naponskog prilagodnog sklopa, a iznosi je 6 Aef → ~ 3.2Vpp.

```
offset = (accum.block_select == 0) ? 3 : 0; //mijenjamo akumulacijski blok pri izlazu iz prekida

for(i = 0; i < 3; ++i)    {

      rms.ULE[i] = 308.4f*sqrt(accum.ULE[i+offset])/32768.0f;
     rms.ULL[i] = 308.4f*sqrt(accum.ULL[i+offset])/32768.0f;
     rms.ILE[i] = 3.75f*sqrt(accum.ILE[i+offset])/32768.0f;

           ///snage
           ///energije
}
```

Isječak koda 3.25. Dio funkcije calculate_rms(), skaliranje mjerenih veličina

Nad računom efektivnih vrijednosti s uzorcima uzetim s AD pretvornika potrebno je napraviti kvantizaciju kako bi efektivna vrijednost odgovarala mjerenoj efektivnoj vrijednosti.

Efektivna vrijednost dobiva se korjenovanjem akumulacije uzoraka za pojedinu veličinu i dijeljenjem s korijenom broja uzoraka (u skladu s formulama 3.2 i 3.3). Broj uzoraka iznosi 3200.

U skladu s gornjim zahtjevom za kvantizaciju potrebno je fazne i linijske napone pomnožiti s 750 (amplituda koja odgovara punoj skali na AD pretvorniku), s korijenom iz dva i s dva (pretvorba iz efektivne u vrijednost od vrha do vrha). Tako dobiveni iznos predstavlja punu skalu te je za kvantizaciju bita još jedino potrebno podijeliti dobiveni iznos s $2^{\text{broj bitova uzorka}}$.

Množenjem fazne struje potrebno je pristupiti na isti način kao i kod faznih i linijskih napona, uzimajući u obzir specifičnosti ulaznog sklopa. Efektivna vrijednost množi se s 9.375, s dva i s korijen i iz dva te se dijeli s $2^{\text{broj bitova uzorka}}$. Broj bitova uzoraka iznosi dvanaest.

3.6.2 Račun snaga i faktora snage

Račun snage i faktora snage temelji se na trokutu snage

Izraz za prividnu snagu:

$$S_{MJ} = U_{MJ} I_{MJ} \tag{3.4}$$

Izraz za radnu snagu:

$$P_{MJ} = \frac{1}{N}\sum_{0}^{3199} u_{MJ}(n) i_{MJ}(n)$$

(3.5)

Izraz za jalovu snagu:

$$Q_{MJ} = \sqrt{S_{MJ}^2 - P_{MJ}^2}$$

(3.6)

Izraz za faktor snage:

$$\cos(\rho) = \frac{P_{MJ}}{S_{MJ}}$$

(3.7)

Akumulacija i množenje uzoraka struje i napona, koje prikazuje izraz (3.5), odvija se unutar RIT prekida.

Unutar funkcije calculate_rms(), koja se nalazi izvan prekidne rutine, računa se prividna, radna i jalova snaga, te faktor snage. Akumulirana radna snaga na izlazu iz prekidne rutine množi se koeficijentima kao što je to slučaj u računu pojedinačnih efektivnih vrijednosti struje i napona (poglavlje 3.4.1.).

```
for(i = 0; i < 3; ++i)
  {
        rms.S[i] = rms.ULE[i] * rms.ILE[i];
        rms.P[i] = 6750.0f*accum.P[i+offset]/4194304.0f;  //kalibriran samo napon
        rms.P[i] *=1002.0f*1009.0f/(1024.0f*1024*256.0f); //1024*sqrt(256)
        rms.Q[i] = sqrt(fabs(rms.S[i]*rms.S[i]-rms.P[i]*rms.P[i]));

        rms.Pf[i] = rms.P[i] / rms.S[i];
  }
```

Isječak koda 3.26. Dio funkcije calculate_rms(), račun snaga i faktora snage (napomena: isječci kodova 3.25. i 3.26. nalaze se u istoj *for* petlji, ali su zbog preglednosti zapisani odvojeno)

3.6.3 Račun radne i jalove energije

Radna i jalova energija računa se kao akumulacija radne i jalove snage. Za mjerenje radne snage koristi se mjerna jedinica W, a za mjerenje radne energije Wh. Isto tako za jalovu snagu koristi se mjerna jedinica VAr, a za jalovu energiju VArh.

```
void accumulateEnergy(EUNIT *data, float Power, uint16_t index)
{
  float total  = (Power + data->residue);
  int32_t factor;
  int32_t tmp = (uint32_t) total;
   factor = 3600;
   data->value   += (tmp / factor);
   data->residue  = (tmp % factor) + (total - tmp);
   rms.energies[index] = data->value;
}
```

Isječak koda 3.27. Funkcija za akumulaciju energije

Zadaća funkcije za akumulaciju energije je zabilježiti i akumulirati svaku promjenu energije veću ili jednaku Wh. Druga zadaća je akumulirati ostatak nastao zaokruživanjem ulaznog iznosa snage i modularnim dijeljenjem cjelobrojnog iznosa snage s faktorom 3600 te pribrojavanjem ostatka ulaznoj snazi u sljedećem pozivanju funkcije. Učestalost pozivanja funkcije je jedna sekunda.

3.6.4 Umjeravanje

Korisnički pristup umjeravanju objašnjen je u poglavlju opisa tipkovnice i prikaznika. Funkcija umjeravanja služi većoj točnosti mjerenja u području nazivnih veličina. Korištenje funkcije preporuča se isključivo uz dobar referentni kalibrator kojeg treba postaviti na 230V/5A.

Nakon uspješno obavljenog umjeravanja, korekcijski faktori upisuju se u vanjsku *flash* memoriju te se prikupljeni uzorci množe njima i dijele s 4096. Prilikom svake slijedeće inicijalizacije učitavaju se te koriste kao korekcijski faktori pri mjerenju.

```
void calibrate(void)
{
  int32_t i, wait = 3;
  float tmp;
  display_calibration();              //   prikazi da je umjeravanje u tijeku    („POSTAVI 230V/5A“)
  while(1)
  {
    for(i = 0; i < 8; ++i)
          devParams.corr[i] = 4096;

          // cekaj 3000 ms prije pocetka umjeravanja
          if((accum.data_ready == 1) && (wait <= 0))
          {
                    accum.data_ready = 0;
                    calculate_rms();            // izracunaj sve RMS vrijednosti

                    for(i = 0; i < 6; ++i)
                    {
                    if(i < 3)    tmp = 4096.0f * 230.0f /rms.ULE[i];
                    else         tmp = 4096.0f * 5.0f/rms.ILE[i - 3];
                    if((tmp - (uint16_t)tmp) >= 0.5)     devParams.corr[i] = (uint16_t) tmp + 1;
                    else                                 devParams.corr[i] = (uint16_t) tmp;
                    }
          write_buff1(0);                              //pocetak upisa u vanjski Flash
          while(!flash_SR());
          write_buff1_into_main(0);
          while(!flash_SR());
          cal.calibrate = 0;
          break;
          }
          else if((accum.data_ready == 1) && (wait > 0))        // cekanje 3000ms
          {
          wait--;
          accum.data_ready = 0;
}}}
```

Isječak koda 3.28. Funkcija umjeravanja

3.6.5 Kompenzacija pomaka faze

Kompenzacija pomaka faze ključna je u povećanju točnosti mjerenja faktora snage, tj. točnosti proračuna razmaka u fazi između napona i struja. Parovi signala struje i napona međusobno su razmaknuti za ~10 µs nakon pretvorbe, iako su uzorkovani s razmakom od 15 µs. Takav slučaj dovodi do zaključka da signali strujnog prilagodnog sklopa dulje propagiraju kroz sklop do AD pretvornika nego što to čine signali naponskog prilagodnog sklopa. U konkretnom slučaju razlika iznosi ~5 µs.

U teoriji ovaj problem je rješiv tako da se promjeni redoslijed uzimanja uzoraka s AD pretvornika. Umjesto uzorkovanja tri faze struje pa tri faze napona, moguće je uzrokovati slijedno parove struje i napona. Takva realizacija ostvariva je promjenom načina rada AD pretvornika ili promjenom rasporeda izlaza prilagodnih sklopova na ulaze AD pretvornika. Takvo rješenje je teoretsko i nije ispitano unutar ovog rada.

Kako bi bilo moguće kompenzirati takav pomak u konkretnoj izvedbi potrebno je nakon svake pretvorbe (svakih 312.5 µs) skup uzoraka s ulaza prilagoditi na vremenskoj osi, kako se parovi struje i napona ne bi razlikovali u fazi prilikom izračuna rezultata. Cilj je struje I2 i I1 te napone U3, U2, U1 translatirati u vremenu kako bi bili u ravnini s prvim uzorkom, odnosno I3. Prilikom translacije u obzir se uzima prethodni uzorak pojedinog ulaza te odnos vremenskog razmaka (faznog pomaka) između struje I3 i ostalih pet uzoraka koji slijede i vremenskog razmaka između uzorka iste faze (312.5 µs).

Vremenski razmaci između struje I3 i ostalih ulaza (redom koji su uzorkovani) iznose 5 µs, 10 µs, 10 µs, 15 µs i 20 µs. Dijeljenjem navedenih razmaka s vremenskim razmakom između uzorka iste faze (312.5 µs), množenjem s 2^{16} i cjelobrojnim zaokruživanjem dobiveni su korekcijski faktori za pojedini ulaz.

$$\frac{(kor.fakt * x_i(n-1) + (2^{16} - kor.fakt) * x_i(n)}{2^{16}} = x_i(n) \tag{3.8}$$

Izrazom (3.8) prikazan je postupak translacije uzoraka.

3.6.6 Frekvencijska analiza ulaznih signala, račun harmonika i harmonijskog izobličenja

Unutar prekidne rutine sprema se po 256 uzoraka svake ulazne veličine kako bi se izvršila frekvencijska analiza nad njima. Za frekvencijsku analizu korištena je funkcija vF_dspl_fftR4b16N256() unutar NXP-ove DSP biblioteke. Unutar funkcije nalazi se Radix-4 implementacija FFT-a u 256 točaka, s 16-bitnim ulaznim i izlaznim vrijednostima. Širina pojedinog frekvencijskog područja nakon poziva funkcije može se izračunati kao odnos frekvencije uzrokovanja (3200 Hz) i broja točaka FFT-a (256), te iznosi 12.5 Hz. Broj ulaznih i izlaznih parametara u funkciju iznosi 512, 256

realnih i 256 imaginarnih. U konkretnom slučaju ulazni parametri su isključivo realni, bez imaginarnih komponenti. Rezultat transformacije u takvom slučaju je frekvencijsko područje od 1600 Hz podijeljeno u frekvencijska područja širine 12.5 Hz. Unutar dobivenog frekvencijskog područja pronalaze se osnovni i viši harmonici (do 31. harmonika) te se računa THD (*Total Harmonic Distortion*) prema sljedećem izrazu:

$$THD = \sqrt{\frac{U_{2rms}^2 + U_{3rms}^2 + \cdots + U_{Nrms}^2}{U_{1rms}^2}}\, x100[\%] \tag{3.9}$$

```
void calculate_fft(uint16_t index)
{
  int16_t i, offset;
  int16_t in[512], out[512];
  float accumulation, limit = 0.0f;
if(index<3)
 limit=20.0f;
 else if((index>=3) && (index<6))
limit=1.0f;
for(i = 0; i < 256; ++i)
  {
    in[2 * i + 0] = fft.data[offset + i]; //dijeljenje s brojem uzoraka prije ffta
    in[2 * i + 1] = 0;
  }
            vF_dspl_fftR4b16N256((int16_t *) &out[0], (int16_t *) &in[0]); //radix 4, 256 tocaka
            offset = index * 32;  //HARMONICI
 for(i = 0; i < 32; ++i)
    rms.harm[offset + i] = sqrt(pow(out[8 * i], 2) + pow(out[8 * i + 1], 2));
//suma harmonika reda [2, 31]
  for(i = 2; i < 32; ++i)
    accumulation += pow(rms.harm[offset + i], 2);

*(&rms.thd_ULE[0] + index) =(rms.harm[offset + 1] < limit) ? 0 : 100.0f * sqrt(accumulation) / rms.harm[offset + 1];
}
```

Isječak koda 3.29. Funkcija za računanje THD-a

3.7 Karakteristična trajanja i zauzeće memorije

Prikupljanje svih 3200 potrebnih uzoraka traje jednu sekundu. Nakon prikupljanja uzoraka, usporedno se računaju željene veličine i prikupljaju novi uzorci. Račun željenih veličina započinje s računom RMS vrijednosti koji traje fiksnih 110 µs, nakon čega slijedi računanje FFT-a čije trajanje varira između 8 i 10 ms jer je ovisno o ulaznim veličinama.

Ukupno zauzeće ROM-a iznosi 75kB, dok ukupno zauzeće RAM-a iznosi 47kB.

4 Ispitivanje s trofaznim kalibratorom

U ispitivanju je primijenjen trofazni kalibrator OMICRON CMC256 i piripadajući Test Universe 3.20. Nominalni naponski ulaz za model simuliran je trofaznim naponom iznosa 230Vef, a nominalni strujni ulaz za model simuliran je trofaznom strujom 5A. U sljedećim tablicama računata je pogreška referirana na mjerenu vrijednost.

Relativna pogreška u mjerenju računa se prema izrazu:

$$\varepsilon(\%) = \frac{X_{MJ} - X_S}{X_S} * 100\% \qquad (4.1)$$

4.1 Ispitivanje linearnosti mjerenja RMS vrijednosti napona i struja s priključenim sinusnim signalom

Tablica 4.1. Fazni naponi i relativne pogreške u mjerenju

U_N	U_N(V)	U_1(V)	U_2(V)	U_3(V)	**ε(%)**		
0	0	0.275	0.358	0.296	x	X	x
0.1	23	22.815	22.790	22.808	-0.80	-0.91	-0.83
0.25	57.5	57.287	57.246	57.350	-0.37	-0.44	-0.26
0.5	115	114.760	114.613	114.876	-0.21	-0.34	-0.11
0.75	172.5	172.257	172.098	172.352	-0.14	-0.23	-0.09
1	230	229.791	229.784	229.931	-0.09	-0.09	-0.03
1.25	287.5	287.320	287.301	287.479	-0.06	-0.07	-0.01

Tablica 4.2. Linijski naponi i relativne pogreške u mjerenju

U_N	U_N(V)	U_{12}(V)	U_{23}(V)	U_{31}(V)	**ε(%)**		
0	0	0.495	0.441	0.417	x	x	x
0.1	40	39.506	39.506	39.466	-1.24	-1.24	-1.39
0.25	100	99.165	99.299	99.242	-0.84	-0.77	-0.76
0.5	200	198.668	198.694	198.890	-0.67	-0.65	-0.56
0.75	300	298.237	298.299	298.423	-0.59	-0.57	-0.53
1	400	397.978	398.141	398.137	-0.51	-0.46	-0.47
1.25	500	497.629	497.833	497.737	-0.47	-0.43	-0.45

Tablica 4.3. Fazne struje i relativne pogreške u mjerenju

I_N	I_N(A)	I_1(A)	I_2(A)	I_3(A)	**ε(%)**		
0	0	0.006	0.004	0.008	x	x	x
0.1	0.5	0.497	0.496	0.499	-0.60	-0.80	-0.20
0.25	1.25	1.247	1.247	1.249	-0.24	-0.24	-0.08
0.5	2.5	2.497	2.497	2.498	-0.12	-0.12	-0.08
0.75	3.75	3.748	3.748	3.749	-0.05	-0.05	-0.03
1	5	4.998	4.998	4.999	-0.04	-0.04	-0.02
1.2	6	5.999	5.998	5.998	-0.02	-0.03	-0.03

4.2 Ispitivanje točnosti mjerenja snaga (P, S, Q) s priključenim sinusnim signalom

U mjerenjima snaga napon je nominalne vrijednosti (U_N), dok se vrijednosti struja mijenjaju.

Tablica 4.4. Radne snage i relativne pogreške u mjerenju

I_N	P_N(W)	P_1(W)	P_2(W)	P_3(W)	**ε(%)**		
0	0	-0.125	0.040	0.026	x	x	x
0.1	115	114.090	114.002	114.632	-0.79	-0.87	-0.32
0.25	287.5	286.411	286.460	287.018	-0.38	-0.36	-0.17
0.5	575	573.728	573.694	574.403	-0.22	-0.23	-0.10
0.75	862.5	861.227	861.048	861.676	-0.15	-0.17	-0.10
1	1.150k	1.148k	1.148k	1.149k	-0.17	-0.17	-0.09
1.2	1.380k	1.378k	1.378k	1.379k	-0.15	-0.15	-0.07

Tablica 4.5. Prividne snage i relativne pogreške u mjerenju

I_N	S_N(VA)	S_1(VA)	S_2(VA)	S_3(VA)	**ε(%)**		
0	0	1.381	0.864	1.813	x	x	x
0.1	115	114.105	114.016	114.645	-0.78	-0.86	-0.31
0.25	287.5	286.419	286.468	287.025	-0.38	-0.36	-0.17
0.5	575	573.733	573.699	574.407	-0.22	-0.23	-0.10
0.75	862.5	861.233	861.052	861.679	-0.15	-0.17	-0.10
1	1.150k	1.148k	1.149k	1.149k	-0.17	-0.09	-0.09
1.2	1.380k	1.378k	1.379k	1.379k	-0.15	-0.07	-0.07

Tablica 4.6. Jalove snage

I_N	Q_1(VAr)	Q_2(VAr)	Q_3(VAr)
0	1.375	-0.864	1.813
0.1	1.845	1.763	-1.670
0.25	2.119	2.031	-2.025
0.5	-2.550	-2.352	-2.229
0.75	-3.162	2.850	2.398
1	-3.588	-3.518	-2.894
1.2	-4.138	3.937	-3.335

4.3 Ispitivanje točnosti mjerenja faktora snage

Simulator trofaznog sustava korišten je kako bi bilo moguće postaviti fazne odnose između napona i struja. Uz idealan slučaj kada između napon i struja pojedine faze ne postoji fazni pomak, analiziran je slučaj kada su napon i struja pojedine faze razmaknuti za nekoliko karakterističnih vrijednosti. Takvim ispitivanjem moguće je utvrditi točnost kompenzacije pomaka faze uzrokovanog vremenskim razmakom uzorkovanja između pojedinog para struje i napona.

Tablica 4.7. Ispitivanje faktora snage ($I=I_N$,$U=U_N$)

∠(rad)	$\cos(\rho)_1$	$\cos(\rho)_2$	$\cos(\rho)_1$	$\cos(\rho)_{IDEALNO}$
$\pi/2$	0.000	-0.000	-0.000	0
$\pi/3$	0.500	0.500	0.500	0.500
$\pi/4$	0.707	0.707	0.707	0.707
$\pi/5$	0.809	0.809	0.809	0.809

4.4 Ispitivanje točnosti mjerenja akumulacije energija (E_p, E_q)

Ispitivanje akumulacije energije odrađeno je uz pomoć *PQ Signal Generator* modula *Test Universe*-a. Odabrana su tri slučaja faznih odnosa između struja i napona te je napon nominalan za sva odabrana mjerenja. Iz razloga što je u pitanju akumulacija, prikazani slučajevi dovoljni su kako bi se ocijenila preciznost mjerenja energije.

Tablica 4.8. Ispitivanje mjerenja ukupnih energija ($\cos(\rho)$=1.00, U=U_N)

Postavke mjerenja		Izmjerene energije				Pogreške mjerenja			
I(A)	t(h)	E+(Wh)	E-(Wh)	Q+(VArh)	Q-(VArh)	E+(%)	E-(%)	Q+(%)	Q-(%)
5	0.5	1722	0	2	0	-0.17	0	X	0
2.5	0.5	861	0	1	0	-0.17	0	X	0
0.5	1	342	0	1	0	-0.87	0	X	0
0.25	1	169	0	1	0	-2.02	0	X	0

Tablica 4.9. Ispitivanje mjerenja ukupnih energija ($\cos(\rho)$=0.5 (induktivan), U=U_N)

Postavke mjerenja		Izmjerene energije				Pogreške mjerenja			
I(A)	t(h)	E+(Wh)	E-(Wh)	Q+(VArh)	Q-(VArh)	E+(%)	E-(%)	Q+(%)	Q-(%)
5	0.5	861	0	1492	0	-0.17	0	-0.13	0
2.5	0.5	430	0	745	0	-0.29	0	-0.26	0
0.5	1	171	0	296	0	-0.87	0	-0.93	0

Tablica 4.10. Ispitivanje mjerenja ukupnih energija ($\cos(\rho)$=0.5 (kapacitivan), U=U_N)

Postavke mjerenja		Izmjerene energije				Pogreške mjerenja			
I(A)	t(h)	E+(Wh)	E-(Wh)	Q+(VArh)	Q-(VArh)	E+(%)	E-(%)	Q+(%)	Q-(%)
5	0.5	861	0	0	1491	-0.17	0	0	-0.19
2.5	0.5	430	0	0	745	-0.29	0	0	-0.26
0.5	1	171	0	0	296	-0.87	0	0	-0.93

4.5 Ispitivanje harmonika i harmonijskog izobličenja

Ispitivanje harmonika odrađeno je pomoću *Harmonics* modula *Test Universe*-a. Na nazivnoj vrijednosti napona (230V) i struja (5A) mjereni su iznosi THD-a (*Total Harmonic Distortion*-a). U svim mjerenjima prvi viši harmonik (100 Hz) iznosi 10% vrijednosti osnovnog harmonika, dok ostali viši harmonici iznose 3%.

Prvo mjerenje predstavlja isključivo utjecaj prvog višeg harmonika na ukupni THD. Viši harmonik iznosi 10% osnovnog harmonika te ujedno toliko iznosi i THD.

Tablica 4.11: Mjerenje s prvim višim harmonikom (100 Hz)

Fazni naponi	THD(%)	ε(%)	Fazne struje	THD(%)	ε(%)
U_1	9.980	-0.20	I_1	9.977	-0.23
U_2	10.029	0.29	I_2	10.029	0.29
U_3	10.001	0.01	I_3	10.024	0.24

Slijedeće mjerenje predstavlja utjecaj viših harmonika na frekvencijskom području do 300 Hz. Prvi viši harmonik iznosi 10%, dok ostali harmonici, koji se nalaze na 150 Hz, 200 Hz, 250 Hz i 300 Hz iznose 3% osnovnog harmonika. Teoretski THD iznosi 11.662%.

Tablica 4.12: Mjerenje do 300 Hz

Fazni naponi	THD(%)	ε(%)	Fazne struje	THD(%)	ε(%)
U_1	11.633	-0.25	I_1	11.643	-0.16
U_2	11.634	-0.24	I_2	11.659	-0.03
U_3	11.644	-0.15	I_3	11.639	-0.20

Slijedeće mjerenje predstavlja utjecaj viših harmonika na frekvencijskom području do 450 Hz. Prvi viši harmonik iznosi 10%, dok ostali harmonici, koji se nalaze na 150 Hz, 250 Hz, 350 Hz i 450 Hz iznose 3% osnovnog harmonika. Teoretski THD iznosi 11.662%.

Tablica 4.13: Mjerenje do 450 Hz

Fazni naponi	THD(%)	ε(%)	Fazne struje	THD(%)	ε(%)
U_1	11.688	0.22	I_1	11.694	0.27
U_2	11.660	-0.02	I_2	11.655	-0.06
U_3	11.654	-0.07	I_3	11.652	-0.09

Slijedeće mjerenje predstavlja utjecaj viših harmonika na frekvencijskom području od 550 Hz do 850 Hz. Prvi viši harmonik iznosi 10%, dok ostali harmonici, koji se nalaze na 550 Hz, 650 Hz, 750 Hz i 850 Hz iznose 3% osnovnog harmonika. Teoretski THD iznosi 11.662%.

Tablica 4.14: Mjerenje od 550 Hz do 850 Hz

Fazni naponi	THD(%)	ε(%)	Fazne struje	THD(%)	ε(%)
U_1	11.655	-0.06	I_1	11.676	0.12
U_2	11.649	-0.11	I_2	11.644	-0.15
U_3	11.673	0.09	I_3	11.645	-0.15

Završno mjerenje predstavlja utjecaj viših harmonika na frekvencijskom području od 950 Hz do 1250 Hz. Prvi viši harmonik iznosi 10%, dok ostali harmonici, koji se nalaze na 950 Hz, 1050 Hz, 1150 Hz i 1250 Hz iznose 3% osnovnog harmonika. Teoretski THD iznosi 11.662%.

Tablica 4.15: Mjerenje od 950 Hz do 1250 Hz

Fazni naponi	THD(%)	ε(%)	Fazne struje	THD(%)	ε(%)
U_1	11.613	-0.42	I_1	11.653	-0.08
U_2	11.701	0.33	I_2	11.688	0.22
U_3	11.650	-0.10	I_3	11.657	-0.04

5 Zaključak

U realizaciji numeričkih algoritama mjerenja postignuti su zadovoljavajući rezultati u vidu zadane točnosti mjerenja parametara niskonaponske trofazne mreže uz dinamiku struje i napona od 10% do 125% nominalne vrijednosti. Rezultate je moguće poboljšati primjenom složenijih numeričkih metoda obrade signala i redizajnom dijelova modela. Neka od mogućih poboljšanja su:

1. Obuhvaćanje većeg dinamičkog područja AD pretvornika redizajnom ulaznog sklopovlja.
2. Primjena AD pretvornika veće razlučivosti.
3. Mjerenje frekvencije ulaznih signala.

Kako bi model mogao zadovoljiti većinu standarda koji se primjenjuju u području mjerenja u radu navedenih veličina, najviše pažnje potrebno je posvetiti mjerenju frekvencije ulaznih signala. Standardi propisuju mjerenje na frekvencijama od 45Hz do 65Hz, dok je modelom moguće zadovoljavajuće mjeriti na frekvenciji od 50Hz.

6 Literatura

[1] LPC1769/68/67/66/65/64/63 32-bit ARM Cortex-M3 Microcontroller; up to 512 kB Flash and 64 kB SRAM with Ethernet, USB 2.0 Host/Device/OTG, CAN, Rev. 9.6 — 18 August 2015, Product Data Sheet

[2] MAX3222/MAX3232/MAX3237/MAX3241*3.0V to 5.5V, Low-Power, up to 1Mbps, True RS-232 Transceivers Using Four 0.1µF External Capacitors, Maxim Integrated 160 Rio Robles, San Jose, CA 95134 USA 1-408-601-1000

[3] Quad-Channel Digital Isolators ADuM1400/ADuM1401/ADuM1402 Data Sheet, ANALOG DEVICES, Rev. L

[4] AC-082A Datasheet, Ampire co.,ltd., 2001.

[5] DP83848C/I/VYB/YB PHYTER™ QFP Single Port 10/100 Mb/s Ethernet Physical Layer Transceiver, REVISED MARCH 2015, Texas Instruments

[6] UM10360LPC176x/5x User Manual, Rev. 4. 1 — 19 December 2016, User Manual

[7] AD7927 8-Channel, 200 kSPS, 12-Bit ADC with Sequencer in 20-Lead TSSOP Datasheet, Analog Devices, 2003.

[8] AT45DB161E 16-Mbit DataFlash (with Extra 512-Kbits), 2.3V or 2.5V Minimum, SPI Serial Flash Memory, © 2017 Adesto Technologies. All Rights Reserved. / Rev.: 8782K–DFLASH–7/2017

[9] Buliding Applications with RL-ARM, for ARM Processor-Based Microcontrollers, KEIL, Tools by ARM

[10] Power Quality Measurement and Troubleshooting, Glen A. Mazur, 1999, 2nd editon

[11] TI Designs. Reference Design to Measure AC Voltage and Current in Protection Relay With Delta-Sigma Chip Diagnostics, 2016.

[12] Moulin E. RMS Calculation for Energy Meter Applications Using the ADE7756. Analog Devices, AN-578 Application Note, 2003.

[13] Mitra S. K. Digital Signal Processing: A Computer Based Approach. Fourth Edition. Department of Electrical and Computer Engineering, University of California, Santa Barbara. McGraw-Hill, 2010.

[14] Dobrenić D. Digitalna regulacija električkih strojeva: Analogni podsistem mjerenja varijabli sinhronog generatora. SOUR Rade Končar, OOUR elektrotehnički institut, 1987.

[15] The Defintive Guide to the ARM Cortex-M3, Joseph Yiu, Elsevier inc., 2007.

[16] Programming Industrial Embedded Systems, 2016./2017. - Laboratory Exercise 1, Hrvoje Džapo

[17] AN10943 Decoding DTMF Tones Using M3 DSP library FFT function Rev. 1 — 17 June 2010 Application Note, NXP B.V. 2010.

[18] Mjerenje parametara trofazne niskonaponske izmjenične mreže numeričkom obradom signala mreže, Davor Štefok, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2018. (Projekt)

[19] Mjerenje parametara trofazne niskonaponske izmjenične mreže numeričkom obradom signala mreže, Davor Štefok, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2017. (Seminar)

Ugradbeni računalni sustav za mjerenje kvalitete električne energije

Sažetak

Diplomski rad obrađuje temu mjerenja električnih parametara niskonaponske mreže 230/400V, 50 Hz (struje, naponi, prividne, radne i jalove snage, faktora snage, radne i jalove energije) pomoću mikrokontrolera NXP LPC1768 s potrebnim periferijama za punu funkcionalnost. Mjerenje harmoničkog izobličenja je implementirano zbog sve češće pojave viših harmonika u niskonaponskoj mreži. Mjerenje električnih parametara temelji se na mjerenju prave efektivne vrijednosti tri struje i tri napona. Ostali parametri računaju se numeričkom obradom signala. Detaljno je opisan mjerni sustav kojeg čini analogno-digitalna pretvorba, numerička obrada signala, prikaz dobivenih veličina na samom modelu pomoću LCD-a te prikaz dobivenih veličina pomoću web poslužitelja.

Ključne riječi: mjerenje, niskonaponska mreža, kvaliteta energije, harmoničko izobličenje, mikrokontroler, digitalna obrada signala, web poslužitelj

Embedded Computer System for Electrical Energy Quality Measurement

Abstract

The purpose of this master thesis is to present the implementation of an electrical parameters measurement system in a low-voltage network 230/400V, 50 Hz (currents, voltages, apparent, active and reactive powers, active and reactive energies) on NXP LPC1768 microcontroller with all the peripherals for full functionality. The measurement of Total Harmonic Distortion is implemented because of the more frequent occurrences of higher harmonics in low-voltage networks. The measurement of electrical parameters is based on the measurement of True Root Mean Square values of three currents and three voltages. The rest of the parameters are calculated by digital signal processing methods. The thesis includes a detailed description of a measuring system that consist of analog to digital conversion, digital signal processing, presentation of measured values on the model's LCD module and presentation of measured values using a web server.

Keywords: measurement, low-voltage netowork, power quality, harmonic distortion, microcontroller, digital signal processing, web server