TAMPEREEN AMMATTIKORKEAKOULU
Tietokonetekniikan koulutusohjelma
Tietokonetekniikka

Tutkintotyö

Lassi Hakala

Sukelluslampun valonohjaus PIC 16F690 prosessorilla

Työn ohjaaja Ilkka Tervaoja
Työn teettäjä Lassi Hakala
Tampere 2007

Lassi Hakala

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Tietokonetekniikka

Hakala, Lassi Sukelluslampun valonohjaus PIC 16F690 prosessorilla
Tutkintotyö Sivut 48 Liitteitä 3
Työn ohjaaja Ilkka Tervaoja
Työn teettäjä Lassi Hakala
Toukokuu 2007
Hakusanat Mikroprosessori, lampunohjaus, PIC 16F690

TIIVISTELMÄ

Työn tarkoituksena on saada sukelluslampun paloaikaa lisättyä, jotta valo riittäisi paremmin sukelluksen ajaksi. Työssä tutustutaan ensin PIC mikroprosessori tekniikkaan ja erityisesti työssä käytettävään PIC 16F690 malliin. Itse työssä suunniteltiin PWM pohjalla toimiva kytkentä, jonka olisi tarkoitus lisätä lampun paloaikaa käytön aikana. Koodaus tehtiin mikroprosessorin omalla assembler- kielellä, jonka kääntämiseen ja ohjelmointiin käytettiin kehitysalustan mukana tullutta MBLAB IDE kääntäjää. Lisäksi käytetään PICKIT 2 v2.20 ohjelmointi laitetta, jolla siirrettiin käännetty koodi prosessorille.

TAMPERE POLYTECHNIC
Information technology
Computer engineering

Hakala, Lassi Diving lamp light controller with PIC 16F690
Engineering Thesis 48 pages , 3 appendices
Thesis Supervisor Ilkka Tervaoja
Commissioning Hakala Lassi
May 2007
Keywords Microcontroller, Light controller, PIC 16F690

ABSTRACT

Aims of this project were to get to know basics of PIC microcontroller and to design diving lamp light conroller based on this technology. First we get to know basics of the PIC microcontroller family and some history behind them. Then we take closer look to a PIC 16F690 model which we use in this project. After we have some knowledge from the PIC controllers we design circuit that increases lamp burning time. We do that with PWM signal that controls lamp. Coding was made with PIC's own assembler-language and building the source code was made with MBLAB IDE program.

ALKUSANAT

Työn löytymiseen vaikutti oma sukellus harrastus. Työn olisi voinut tehdä monella muullakin tavalla, mutta päädyin tekemään sen mikroprosessori ohjaimen avulla, jolloin saisin kokemusta niiden käytöstä. Työn tein varsinaisen työni ohessa aina, kun sille liikeni aikaa päätöiltä. Työssä pääsin hyödyntämään elektroniikan, digitaalitekniikan ja laitteistoläheisten kursseilla opittuja tietoja ja taitoja.

Tampereella 16.5.2007

_____

Lassi Hakala

SISÄLLYSLUETTELO

LYHENTEET

PIC	Peripheral Interface Controller tai alkuperäinen lyhenne Programmable Intelligent Computer
MSB	Vähiten merkitsevä bitti
LSB	Eniten merkitsevä bitti
USB	Universal Serial Bus
PWM	Pulse-width modulation
I/O	INPUT/OUTPUT
A/D	Analog to digital
DSP	Digitaalinen signaalin käsittely
EEPROM	Electrically Erasable Programmable Read-Only Memory
kt	Kilo tavu
Hz	Taajuus

1 JOHDANTO

Sukellettaessa monia peräkkäisiä sukelluksia tai pitkiä sukelluksia on vaarana, että lampun akuista voi loppua virta kesken. Aikaisemmin akun kestoon pystyi vaikuttamaan vain laittamalla lampun kiinni, jolloin kaikki valo katosi. Toinen vaihtoehto oli pitää lamppua päällä, jolloin virtaa kului koko lampun teholla. Lampun toki pystyi pinnalla vaihtamaan tehottomampaan, mutta silloin myös valoteho kärsi silloin, kun sitä olisi tarvinnut. Työn tarkoituksena on saada lamppuun säädin, joka mahdollisimman yksinkertaisesti säätää lampun tehoa. Myös säätimen ohjaus tarvitsee toimia yhden napin avulla, sillä sukelluslampun tulee toimia jopa 50m eli 6bar paineessa. Tällöin ylimääräiset läpiviennit koteloon nappeja varten tuovat aina yhden mahdollisen vuotokohdan lisää. Säätimen kanssa on mahdollista käyttää tehokkaita 50-70W lamppuja, joilla saadaan paljon valotehoa tarvittaessa. Kun lamppua ei tarvitse hyvissä olosuhteissa pitää täysillä, valoteho voidaan pitää puolessa, jolloin myös virran käyttö puolittuu ja paloaika lisääntyy. Huonoissa olosuhteissa, jossa tarvitaan taas valotehoa, pidetään lamppu täydellä teholla tarvittava aika ja pienennetään tehoja, jos se on mahdollista.

2. PIC PROSESSORIT

PIC mikroprosessoreita on nykyään hyvin monenlaisia 8- ja 16-bittisiä malleja. PIC prosessorit ovat hyvin suosittuja nykyään harrastelijoiden sekä suunnittelijoiden keskuudessa. Tämän aiheuttaa niiden halpa hinta, helppo saatavuus, laaja käyttäjäpohja, halvat kehitysalustat sekä mahdollisuudet ohjelmoida prosessori vielä, kun se on kytkennässä kiinni./3/

2.1 PIC:n ominaisuudet

PIC prosessori käyttää Harvard arkkitehtuuria, joten siinä on erilliset data ja ohjelma muistit. Tämä tarkoittaa, että ohjelmamuistit ovat 12-16-bittisiä ja datamuistit ovat 8-bittisiä. Tästä etuna varmistetaan, että käskyt voidaan suorittaa yhden käskyjakson aikana. /4/

Prosessorit ovat RISC (reduced instruction set computer) periaatteella toimivia, koska perus PIC prosessorin käskykantaan kuuluu vain 35 eri käskyä ja ne kaikki voidaan suorittaa yhden kellojakson aikana. /3/ Perus PIC mallien käskykanta on tuo 35 käskyä, mutta uusimmat ja lisää ominaisuuksia sisältävät ”high-end” mallit käyttävät jo 70:tä eri käskyä. /3/

PIC piirien mielenkiintoisiin ominaisuuksiin kuuluu myös ICSP (In-Circuit Serial Programming), joka tarkoittaa, että piiri on mahdollista ohjelmoida sarjamuotoisesti milloin vain. Tällöin piirit voidaan ennen ohjelmointia juottaa valmiiksi piirilevyille ja ohjelmoida myöhemmin. Tässä tapauksessa ainoastaan kytkennän on oltava jännitteetön. Piiriin kytketään käyttöjännite (Vdd) ja maan (Vss) lisäksi ohjelmointijännite (Vpp), ICSP-kellosignaali (ISCPCLK) ja ISCP-data (ISCPDAT)./3/

2.2 PIC mallit

PIC prosessorit voidaan jakaa neljään eri luokkaan ominaisuuksiensa ja osin mallinumeroidensa perusteella. Nämä luokat ovat: perusmallit, keskitason piirit, korkean tason piirit ja 16bittiset PIC . /6/

Perusmallit kuuluvat pääosin PIC10 sarjaan, mutta osa PIC12 ja PIC16 sarjoista kuuluu myös tähän perusmalli kategoriaan. Yhteistä näillä perusmalleilla on 12- bittin levyiset ohjelmamuistit sekä vain 2-tasoinen pino aliohjelmien paluuosoitteita varten. Näitä piirejä on saatavissa 6 jalan pienistä piireistä aina vähän suurempiin 28 jalkaisiin. Piireillä on myös matala toimintajännite, joten ne toimivat myös pattereilla toimivissa sovelluksissa. /6/

Keskitason arkkitehtuurin piirit eroavat vain vähän perusmallien piireistä. Näillä piireillä on isompi 8-tasoinen pino paluuosoitteita varten sekä niiden

ohjelmamuistit ovat 14-bittiä leveitä. Lisäksi osa näistä keskitason piireistä tarjoaa monikanavaisia A/D muuntimia, parempaa keskeytysten hallintaa ja EEPROM muistia datalle. Pääosa PIC12 ja PIC16 sarjoista kuuluvat tähän arkkitehtuuriin. Piirien koko vaihtelee 8-64 jalkaisten välillä. /6/

Korkean tason piirit eroavat edellisistä jo enemmän. Alun perin ne olivat PIC 17 sarjaa, mutta se ei menestynyt markkinoilla, joten se korvattiin PIC18 sarjalla. Siitä tuli lopulta hyvin suosittu. Piireillä oli entistä pidempi paluuosoite pino 31 tasoa. Lisäksi sitä pystyy myös käyttämään suoraan käskyillä. Aikaisemmin pinoa ei ole pystynyt lukemaan tai kirjoittamaan käskyjen avulla. Ohjelmamuistien leveys kasvoi myös 16-bittiin, joka mahdollistaa monien uusien käskyjen käytön. Lisääntyneen käskykannan ansiosta piireille on helpompi kirjoittaa ohjelmia C-kielellä. Tämän sarjan piireillä on paljon ominaisuuksia valmiina. On piirejä, joihin on lisättynä suora tuki USB- tai ethernet-protokolille . Piirien koko vaihtelee 18-100 jalan välillä. /6/

Viimeisenä luokkana on 16 bittiset PIC prosessorit. Tarjolla on sekä normaaleja 16-bittisiä mikrokontrollereita että DSP ominaisuudet sisältäviä kontrollereita. Piirit tarjoavat monia digitaalisessa signaalin prosessoinnissa tarvittavia käskyjä. Piirit ovat harvard-arkitehtuurin RISC prosessoreita, joten ne suorittavat käskyt nopeasti yhden käskyjakson aikana. Nämä dsPIC prosessorit tulivat markkinoille vasta vuonna 2004. Piirit tukevat entistä enemmän C-kielistö-ohjelmointia.

2.4 Historia

Ensimmäinen PIC prosessori oli tarkoittettu käytettäväksi uuden 16bittisen CP1600 prosessorin kanssa. CP1600 oli muuten hyvä prosessori, mutta se kärsi huonosta I/O suorituskyvystä. PIC1650 kehitettiin vuonna 1975 auttamaan CP1600 prosessoria ottamalla I/O tehtävät osin itselleen. Vuonna 1985 General Instrument myi osan toiminnoistaan ja PIC prosessorien kehitys myytiin samalla. Uusi omistaja lakkautti lähes kaikki toiminnot, jotka olivat jo tässä vaiheessa päässeet vanhentumaan. Microchip, joka oli uusi omistaja, onnistui

kuitenkin kehittämään prosessorista halvan OTP (one time programmable) mallin, jonka avulla valmistajat pystyivät päivittämään prosessorien koodia kesken tuotannon. Aikaisemmin nämä OTP ohjelmoitavat prosessorit olivat kalliita ja niiden sijasta käytettiin maski-ROM piirejä, jotka täytyi tehdä aina uudelleen, kun koodiin tehtiin muutoksia. Lisäksi piirin ICSP ominaisuus auttoi PIC-prosessoria tulemaan suosituksi eri suunnittelijoiden keskuudessa. Nykyään Microchip on toimittanut yli 5 miljardia PIC piiriä. /3/ /5/

2.3 PIC assembly käskyt

PIC-prosessorien käskykanta käsittää 35 erilaista käskyä, joilla piirien ohjelmointi tapahtuu. Käskyt voidaan jakaa kolmeen eri kategoriaan: tavua käsittelevät rekisterikäskyt, bittiä käsittelevät rekisterikäskyt ja kontrollioperaatiot. /1/

Kaikki käskyt käyttävät suorittimelta vain yhden käskyjakson suoritusaikaa. Ainoastaan hyppykäskyn sisältävät käskyt käyttävät kaksi käskyjaksoa, jos ne toteuttavat hypyn, muuten nekin käyttävät vain yhden käskyjakson.

Käskyissä käytetään seuraavia muuttujia:

f = Rekisteriosoite (0x00 ... 0x7F), tai rekisterille annettu nimi

d = Kohteen valinta. d = 0 palauttaa tuloksen W-rekiseriin d = 1 palauttaa tuloksen f-rekisteriin (käskyissä oletusarvona palautus f-rekisteriin)

b = bitti numero kohderekisterissä

k = luku 00h ... FFh, voidaan käyttää myös nimettyä rekisteriä.

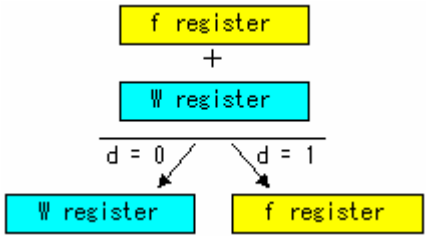
Käskyjen toiminta on ensin selitetty sanallisesti. Sen jälkeen osasta käskyjä on selventävä kuva, joka auttaa ymmärtämään käskyn toimintaa.

2.3.1 Tavua käsittelevät käskyt

Seuraavat käskyt vaikuttavat koko rekisterin bitteihin kerralla eikä niillä pääse muokkaamaan suoraan yksittäisiä bittejä.

ADDWF f ,d

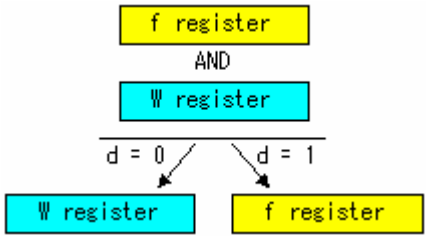
Toiminnolla lisätään W-rekisterin sisältö f-rekisterin sisältöön. F-rekisteri voi olla mikä tahansa rekisteri, joka on nimetty tai vain muistipaikan numerolla määrätty. D määrää, kirjoitetaanko tulos W- vai f-rekisteriin. Jos d on 0, tulos kirjoitetaan W-rekisteriin ja d:n ollessa 1, tulos menee f-rekisteriin. Käsky voi vaikuttaa STATUS-rekisterin kolmeen bittiin C-, DC- ja Z-bitteihin. (kuva 1)



Kuva 1. ADDWF-käskyn toiminta /8/

ANDWF f,d

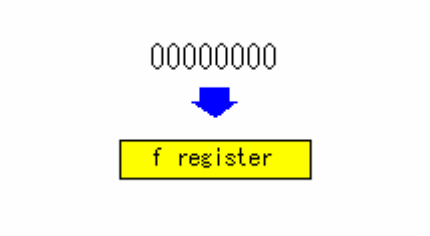
Toiminnolla vertaillaan W- ja f-rekisterin bittejä ja jälleen tulos palautetaan d-bitin määräämään paikkaan. Käsky voi vaikuttaa STATUS-rekisterin Z-bittiin. (kuva 2)



Kuva 2. ANDWF-käskyn toiminta /8/

CLRF f

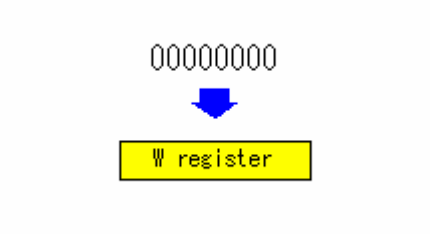
Toiminto tyhjentää rekisterin F-rekisterin sisällön ja asettaa STATUS-rekisterin Z bitin suoraan 1. (kuva 3)



Kuva 3. CLRF-rekisterin toiminta /8/

CLRW

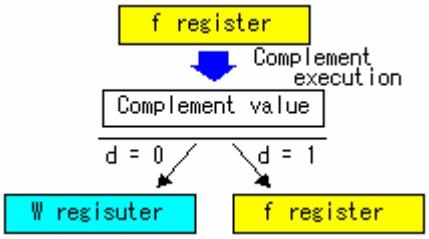
Toimii samoin kuin edellinen käsky CLRF, mutta vaikuttaa W-rekisteriin. (kuva 4)



Kuva 4. CLRW-käskyn toiminta /8/

COMF f, d

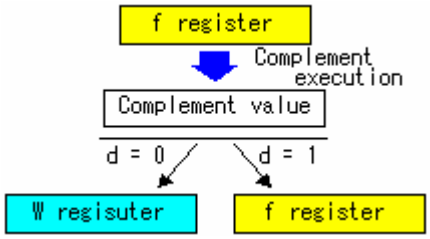
Muodostaa f-rekisterin komplementin eli 0->1 ja 1->0. D määrää jälleen mihin rekisteriin tulos kirjataan. Käsky voi vaikuttaa STATUS-rekisterin Z-bittiin.(kuva 5)



Kuva 5. COMF-käskyn toiminta /8/

DECF f ,d

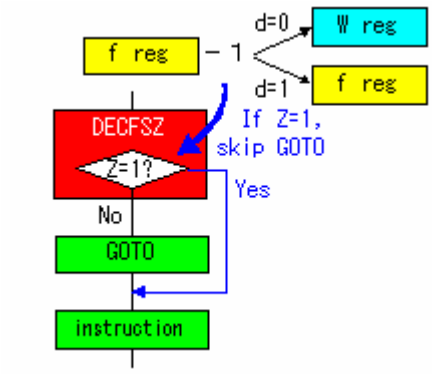
Käsky vähentää f-rekisterin arvosta yhden ja tallentaa käskyn d:n määräämään rekisteriin. Käsky voi vaikuttaa STATUS-rekisterin Z-bittiin. (kuva 6)



Kuva 6. DECF-käskyn toiminta /8/

DECFSZ f,d

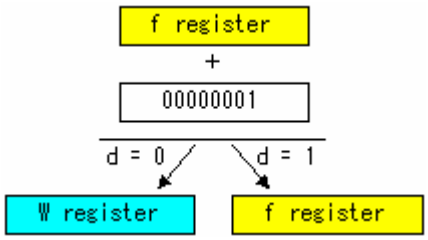
Käsky toimii kuten edellinen DECF-käsky mutta, jos tulokseksi tulee 0, hypätään seuraavan käskyn yli. (kuva 7)



Kuva 7. DECFSZ-käskyn toiminta /8/

INCF f,d

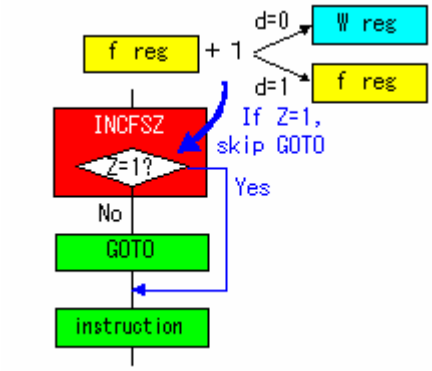
Lisätään F rekisterin arvoon 1 ja tulos siirretään d määräämään rekisteriin. Käsky voi vaikuttaa STATUS-rekisterin Z-bittiin. (kuva 8)



Kuva 8. INCF käskyn toiminta /8/

INCFSZ f,d

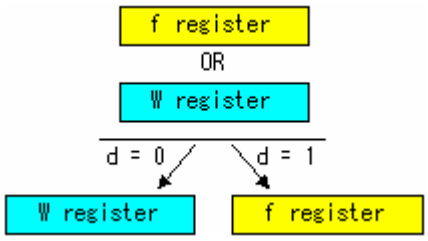
Toiminta sama kuin INCF, mutta tuloksen mennessä 0, hypätään seuraavan käskyn yli. (kuva 9)



Kuva 9. INCFSZ-käskyn toiminta /8/

IORWF f,d

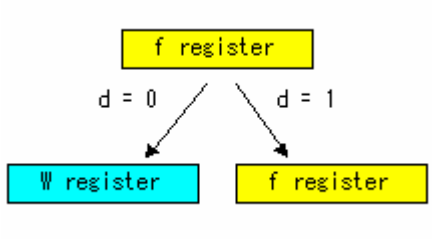
Käsky tekee OR-operaation f- ja W-rekisterien kanssa, ja tallentaa tuloksen d:n määräämään rekisteriin. Käsky voi vaikuttaa STATUS-rekisterin Z-bittiin. (kuva 10)



Kuva 10. IORWF-käskyn toiminta /8/

MOVF f,d

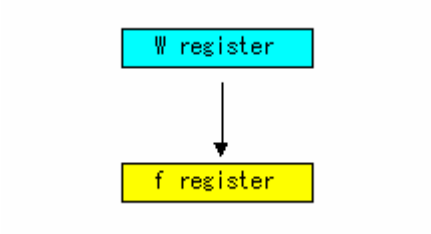
Siirretään f-rekisterin tiedot joko W- tai f-rekisteriin. Käsky voi vaikuttaa STATUS-rekisterin Z-bittiin. (kuva 11)



Kuva 11. MOVF-Käskyn toiminta /8/

MOVWF f

Käskyllä kopioidaan W-rekisterin tiedot rekisteriin f. (kuva 12)



Kuva 12. MOVWF-käskyn toiminta /8/

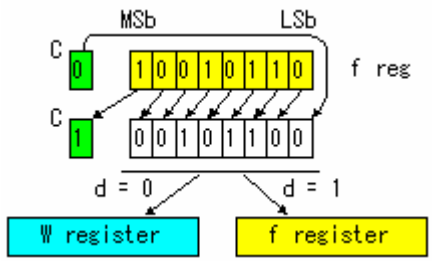
NOP

Käsky ei tee mitään. Tällä käskyllä voidaan tehdä yhden kellosyklin pituisia viivästyksiä. (kuva 13)

Kuva 13. NOP-käskyn toiminta /8/

RFL f,d

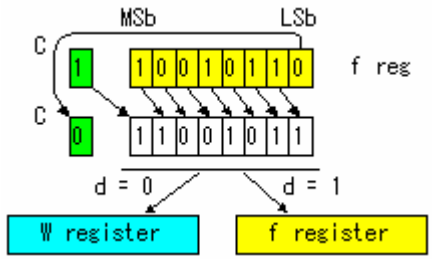
Käskyllä siirretään bittejä yksi vasemmalle ja tallennetaan MSB-bitti STATUS-rekisterissä olevaan Carry-bittiin. Täten uusi LSB-bitti saadaan STATUS-rekisterin Carry-bitin edellisestä tilasta. D määrää jälleen mihin tulos kyseisessä operaatiosta kirjoitetaan. (kuva 14)



Kuva 14. RFL-käskyntoiminta /8/

RRF f,d

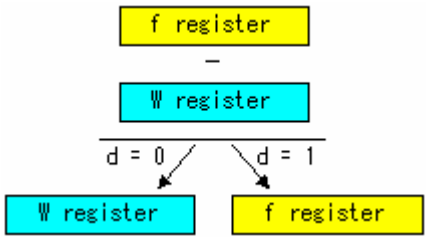
Toimii kuten RFL mutta kierrättää bittejä oikealle. (kuva 15)



Kuva 15. RFF-käskyn toiminta /8/

SUBWF f,d

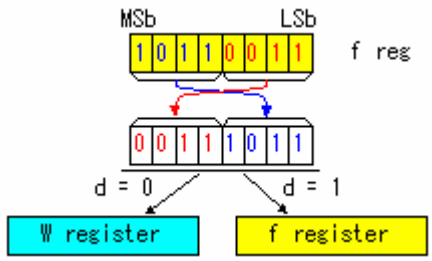
Käsky vähentää sekä W- että f-rekisterin sisällön ja tallentaa tuloksen d:n määräämään rekisteriin. Toiminto voi vaikuttaa STATUS-rekisterin C-, DC- ja Z-bitteihin. (kuva 16)



Kuva 16. SUBWF-käskyn toiminta /8/

SWAPF f,d

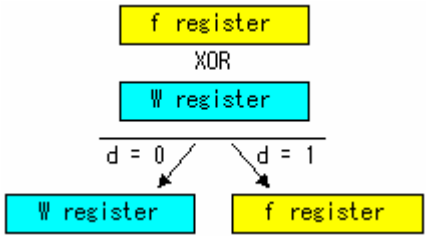
Käsky vaihtaa f-rekisterin neljän ylemmän ja neljän alimman bitin paikkoja ja tallentaa tuloksen d:n määräämään rekisteriin. (kuva 17)



Kuva 17. SWAPF-käskyn toiminta /8/

XORWF f,d

Käsky tekee loogisen vertailuoperaatio XOR:n rekisterien W ja f välillä ja tallentaa tuloksen d:n määräämään rekisteriin. Käsky voi vaikuttaa STATUS-rekisterin Z-bittiin. (kuva 18)

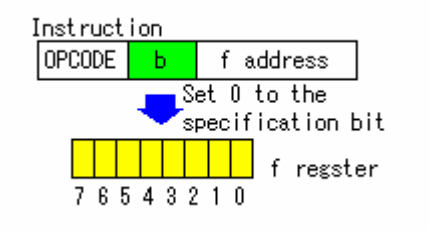


Kuva 18. XORWF-käskyn toiminta /8/

Seuraavat käskyt vaikuttavat suoraan rekisterin sisälle yksittäisen bitin tilaan.

BCF f,b

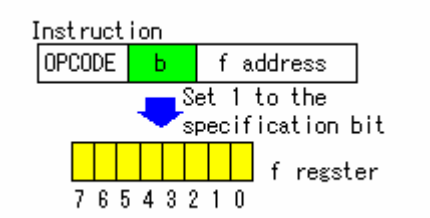
Käsky kirjoittaa 0 f-rekisterin bittiin, jonka b määrää. (kuva 19)



Kuva 19. BCF-käskyn toiminta /8/

BSF f,b

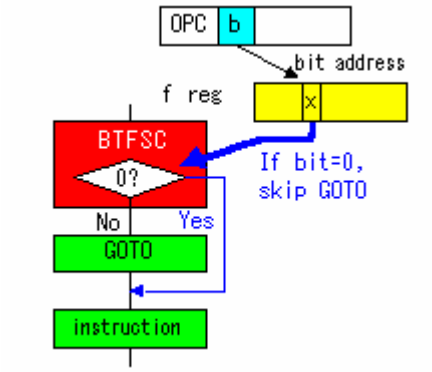
Käsky toimii kuten BCF, mutta se asettaa f-rekisterin b-bitin asennon tilaan 1. (kuva 20)



Kuva 20. BSF-käskyn toiminta /8/

BTFSC f,b

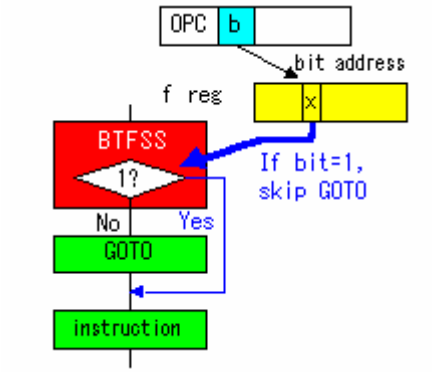
Käsky tarkistaa f-rekisterin b-bitin tilan. Jos Bitti b on nolla, hypätään yli seuraavan käskyn. (kuva 21)



Kuva 21. BTFSC-käskyn toiminta /8/

BTFSS f,b

Toimii kuten käsky BTFSC, mutta hyppykäsky aiheutuu, jos bitti on tilassa 1. (kuva 22)



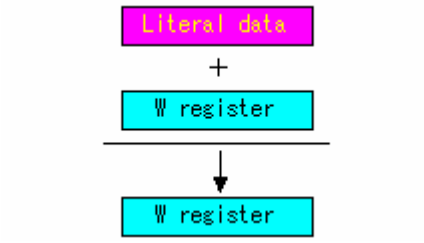
Kuva 22. BTFSS-käskyn toiminta /8/

3.3 Literaali- ja kontrollioperaatiot

Seuraavilla operaatioilla voidaan lisätä rekistereihin tietty luku tai niitä käytetään ohjelman ohjaukseen.

ADDLW k

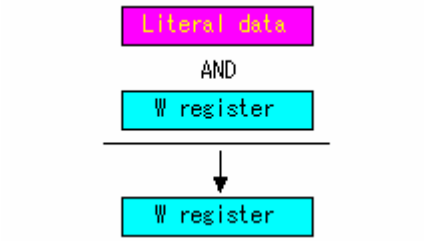
Käskyllä lisätään k:n arvo rekisterin W. Toiminto voi vaikuttaa STATUS-rekisterin C-, DC- ja Z-bitteihin. (kuva 23)



Kuva 23. ADDLW-käskyn toiminta /8/

ANDLW

Suorittaa AND-operaation k:n ja rekisterin W kesken. Toiminto voi vaikuttaa rekisterin STATUS bittiin Z. (kuva 24)



Kuva 24. ANDLW-käskyn toiminta /8/

CALL k

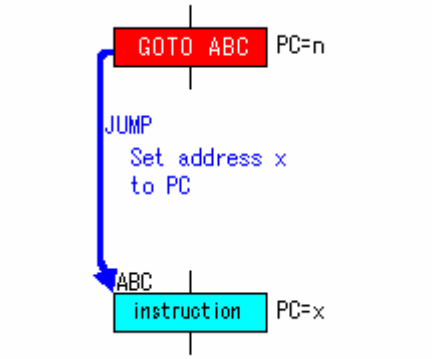
Käskyllä kutsutaan k nimistä tai paikasta löytyvää aliohjelmaa. Lisäksi käsky tallentaa nykyisen ohjelman suorituspaikan pinoon, jolloin siihen voidaan palata aliohjelman loputtua.

CLRWDT

Käskyllä alustetaan watchdog timer alkuarvoihinsa

GOTO k

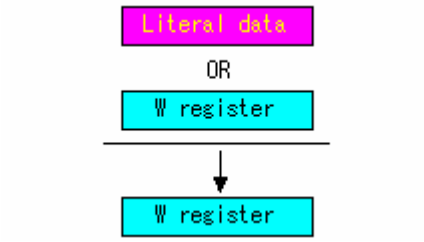
Käskyllä hypätään k:n määräämään paikkaan koodissa. (kuva 25)



kuva 25. GOTO-käskyn toiminta /8/

IORLW k

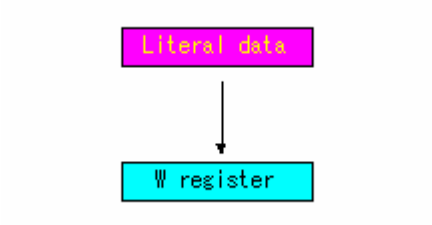
Käskyllä vertaillaan lukua k W-rekisterin kanssa ja tallenetaan tulos rekisteriin W. Toiminto voi vaikuttaa rekisterin STATUS bittiin Z. (kuva 26)



Kuva26. IORLW-käskyn toiminta /8/

MOVLW k

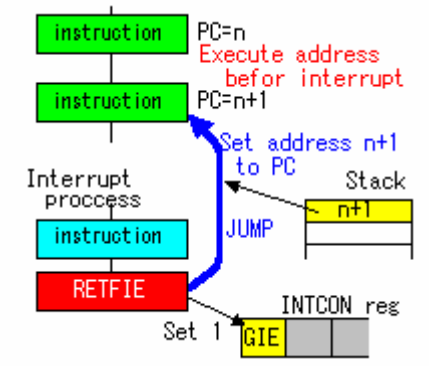
Käskyllä tallennetaan annettu data k rekisteriin W. (kuva 27)



Kuva 27. MOVLW käskyn toiminta /8/

RETFIE

Käskyllä palautetaan ohjelma keskeytyksestä. Käsky hakee pinosta päällimmäisen paluu osoitteen ja palaa suorittamaan ohjelmaa siitä kohdasta. (kuva 28)



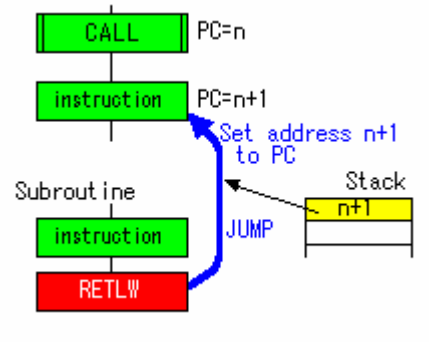
kuva 28. RETFIE-käskyn toiminta /8/

RETLW k

Sama käsky kuin edellinen, mutta luku k siirretään samalla W-rekisteriin

RETURN

Käskyllä palataan aliohjelmasta edelliseen koodin suorituspaikkaan. Käsky ottaa pinosta päällimmäisen paikan ja jatkaa siitä koodin suoritusta. (kuva 29)



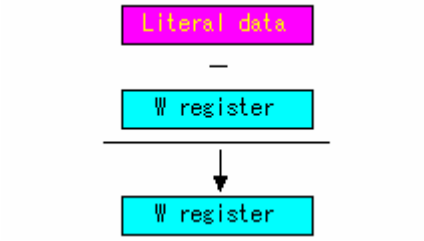
Kuva 29. RETURN-käskyn suoritus /8/

SLEEP

Käskyllä prosessori menee stand by tilaan. STATUS-rekisterin Time Out bitti 4 (TO) asetetaan 1 ja saman rekisterin Power Down bitti 3 (PD) asetetaan 0. Myös watchdog timer ja kello-oskillaattori sammutetaan.

SUBLW k

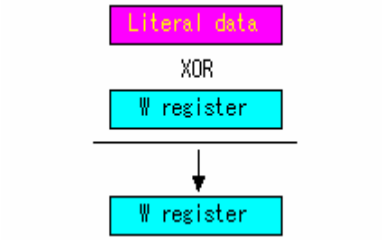
Käskyllä vähennetään k-arvo rekisteristä W ja se tallennetaan W-rekisteriin. Toiminto voi vaikuttaa STATUS-rekisterin C-, DC- ja Z-bitteihin.(kuva 30)



Kuva 30. SUBLW-käskyn toiminta /8/

XORLW k

Käsky suorittaa XOR-operaation k ja W-rekisterin arvon välillä. Toiminto voi vaikuttaa STATUS-rekisterin Z-bittiin. (kuva 31)



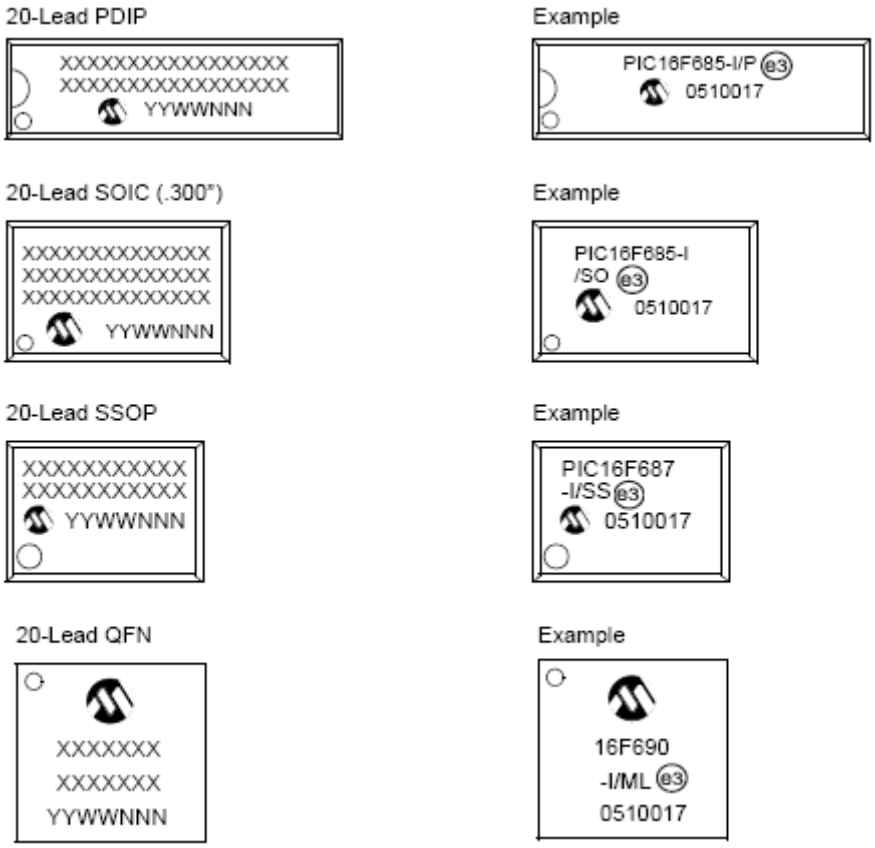
Kuva 31. XORLW-käskyn toiminta /8/

3 PIC 16F690

PIC16F690 on siis keskitason arkkitehtuuriin perustuva prosessori. Tämä tarkoittaa, että käskykanta käsittää edellisessä kappaleessa käsitellyt 35 eri käskyä. Prosessori on 8-bittisellä ytimellä ja sen ohjelmamuistien leveys on 14-bittiä. Nopeus on maksimissään 20 MHz, jolloin joudutaan käyttämään ulkoista kidettä. Prossessorissa on myös sisäinen kello-oskillaattori, jonka nopeuden voi säätää välillä 32 kHz-8Mhz. Jos käytetään sisäistä kello-oskillaattoria, oletuksena on piirin nopeus 4MHz, jota myös tässä työssä käytetään. Koska prosessori pystyy käyttämään sisäistä kello-oskillaattoria, sen vaatima minimikytkentä on todella pieni. Periaatteessa tarvitsee vain kytkeä maa- ja käyttöjännite kiinni. Käyttöjännitealue on laaja ja se ulottuu aina 2V:sta - 5.5V:iin, jolloin monet eri jännite lähteet pystyvät käyttämään tätä mallia. Muistia prosessorista löytyy 4096 tavua eli 4kt. Tämä muisti on siis jaettu 14-bittisiin osiin, jonka aiheuttaa harvard arkkitehtuuri. Lisäksi prosessorista löytyy 8-bittistä SRAM-muistia rekistereitä ja määriteltyjä muuttujia varten. Näiden kahden lisäksi löytyy myös 256 tavun mittainen EEPROM-muisti. Tähän muistiin voidaan ajon aikana tallentaa dataa ja se pysyy siellä myös käyttöjännitteen poiston jälkeen. Prosessorissa on myös 12 10-bittistä A/D-muunninta. /7/

3.1 Koteloinnit

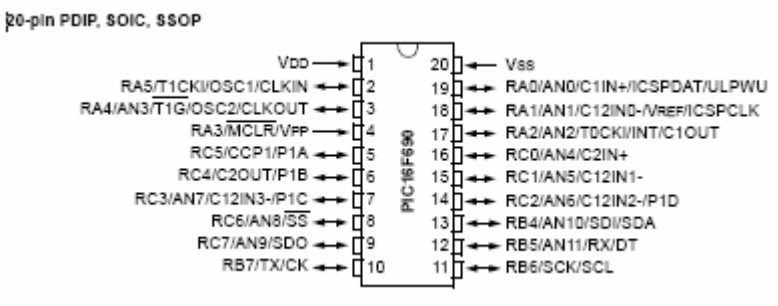
PIC 16F690 prosessoria toimitetaan neljällä eri koteloinnilla(kuva 32). Koteloinnit eroavat toisistaan jonkin verran. Eniten tietysti eroaa PDIP koteloitu reikäjuotettava muiden kotelointi tyyppien ollessa pintajuotettavia malleja. SOIC- ja SSOP-koteloinnit eivät eroa toisistaan muuten kuin kooltaan. Kummatkin ovat pintaliitoskomponentteja, joilla on jalat juotoksien tekemistä varten. QFN-kotelotyyppi taas eroaa näistä kahdesta eniten, sillä siinä ei ole jalkoja lainkaan, vaan se juotetaan suoraan piirilevyyn pintaliitoksena. /7/



Kuva 32. PIC 16F690-kotelointityypit /7/

3.2 Liitännät

PIC 16F690-prosessorissa on 20 jalkaa, joista kaksi on varattu käyttöjännitteelle ja maalle. Loput ovat ohjelmoijan käytössä. Kaikki nämä portit voidaan määrätä joko digitaalisiksi tuloiksi tai lähdöiksi. Lisäksi osa porteista voidaan määrätä myös analogisiksi, lähinnä A/D muunninta varten. Kuvasta 33 nähdään pinnien eri toimintamahdollisuudet. Tärkeimpänä näistä voidaan mainita pinnien jako kolmeen eri ryhmään RA0-RA5, RB4-RB7 ja RC0-RC7. Nämä portit voidaan TRIS-rekisterin avulla määrätä digitaalisiksi tuloiksi ja lähdöiksi. Eri rekisterien avulla taas näille jaloille päästään määräämään eri toimintatapoja. Erikseen voisi myös mainita pinnit 4, 18 ja 19. Näiden pinnien kautta pystytään hoitamaan piirin ohjelmointi.



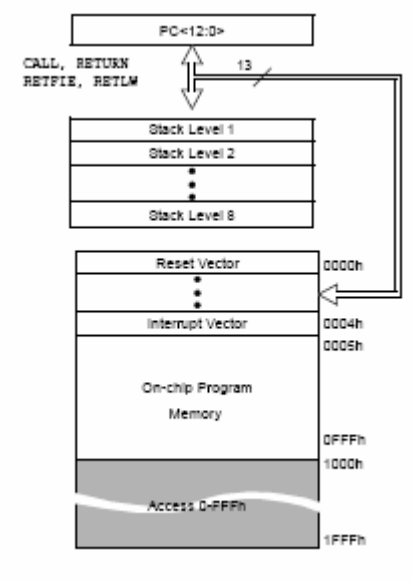
Kuva 33. PIC 16F690-liitännät

3.3 Muistialue

Koska PIC kuuluu harvard arkkitehtuuriin, sillä on erilliset ohjelma- ja datamuistit. Ohjelmamuistien leveydet ovat 14-bittiä ja datamuistien taas 8-bittiä. Ohjelmamuistiin tallennetaan kirjoitetun ohjelman käskyt. Datamuistissa sijaitsevat työrekisterit, tehdyt muuttujat sekä eri rekistereitä, joilla määrätään prosessorin toimintaa.

3.3.1 Ohjelma muistit

PIC 16F690-prosessorilla on 13-bittinen ohjelmalaskin, jolla pystyy osoittamaan yhteensä kahdeksaan kilotavuun 14-bittistä muistia. Piirin sisällä kyseisestä muistimäärästä on kuitenkin vain neljä kilotavua. Kuten kuvasta 34 nähdään piirin resetointiosoite on 0000h ja keskeytysvektori on osoitteesta 00004h. Piirin loppuohjelmamuisti on väliltä 0005h-0FFFh. Muistissa sijaitsee myös pino aliohjelmien paluuosoitteita varten. Täten tällä prosessorilla on mahdollista kutsua vain kahdeksaa sisäkkäistä aliohjelmaa. Lisäksi prosessorilla ei ole mitään käskyjä, joilla pinoon pääsisi itse vaikuttamaan. Ainoastaan call, return ,retfie ja retlw -käskyt vaikuttavat pinoon. /7/



Kuva 34. PIC16F690 Ohjelmamuistin ja pinon kartta /7/

3.3.2 Data muistit

Datamuistit jaotellaan neljään eri pankkiin, jotka koostuvat General Purpose Registers (GPR) ja The Special Function Registers (SFR) rekistereistä. SFR-rekisterit sijaitsevat 32 ensimmäisessä tilassa jokaisessa pankissa. Loput 96 -tilaa jokaisessa pankissa kuuluvat sitä vastoin GPR-rekisteihin(kuva 35). SFR-rekistereissä sijaitsevat prosessorin tarvitsemat ohjausrekisterit, jotka määräävät esimerkiksi porttien toiminnan. Pankkeja voidaan lukea joko suorasti tai epäsuorasti. Epäsuoralla osoituksella pankki 0 löytyy osoite alueelta 00h-7Fh, pankki1 80h-FFH, pankki2 100h-17Fh ja pankki 180h-1FFh. Suorasti voidaan osoittaa vain yhtä pankkia kerrallaan. Tällöin kyseisen pankin rekisterit löytyvät alueelta 00h-7Fh. Tämä kulloinkin käytössä oleva rekisteri määrätään STATUS-rekisterin RP1- ja RP0-biteillä. Taulukosta 1 nähdään, kuinka RP-bitit määräävät käytettävän pankin.

Taulukko 1. Pankkien valinta

RP1	RP0	Tila
0	0	pankki0
0	1	pankki1
1	0	pankki2
1	1	pankki3

	File Address		File Address		File Address		File Address
Indirect addr. (1)	00h	Indirect addr. (1)	80h	Indirect addr. (1)	100h	Indirect addr. (1)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	PORTA	105h	TRISA	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	PORTC	107h	TRISC	187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2(1)	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh		18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh		18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h		91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD(2)	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUA	95h	WPUB	115h		195h
CCPR1H	16h	IOCA	96h	IOCB	116h		196h
CCP1CON	17h	WDTCON	97h		117h		197h
RCSTA	18h	TXSTA	98h	VRCON	118h		198h
TXREG	19h	SPBRG	99h	CM1CON0	119h		199h
RCREG	1Ah	SPBRGH	9Ah	CM2CON0	11Ah		19Ah
	1Bh	BAUDCTL	9Bh	CM2CON1	11Bh		19Bh
PWM1CON	1Ch		9Ch		11Ch		19Ch
ECCPAS	1Dh		9Dh		11Dh	PSTRCON	19Dh
ADRESH	1Eh	ADRESL	9Eh	ANSEL	11Eh	SRCON	19Eh
ADCON0	1Fh	ADCON1	9Fh	ANSELH	11Fh		19Fh
General Purpose Register 96 Bytes	20h – 7Fh	General Purpose Register 80 Bytes	A0h – EFh	General Purpose Register 80 Bytes	120h – 16Fh		1A0h
		accesses 70h-7Fh	F0h – FFh	accesses 70h-7Fh	170h – 17Fh	accesses 70h-7Fh	1F0h – 1FFh
Bank 0		Bank1		Bank2		Bank3	

Kuva 35. PIC 16F690-datamuistienosoitteet.

3.3.3 EEPROM-muisti

Näiden lisäksi piiriltä löytyy 256 tavua eeprom muistia. Tämän muistin kirjoitus onnistuu ajonaikana ja ohjelmointivaiheessa. Muisti ei tyhjene käyttöjännitteen poistuessa, joten tänne pystytään säilömään datataulukoita, Muisteilla ei ole suoraan omia osoitteita, mutta niitä käytetään datamuistin SFR puolella olevien rekisterien kautta. Nämä rekisterit ovat EECON1, EECON2,

EEDAT, EEDATH, EEADR ja EEADRH. EEDAT rekisteriin asetetaan data joka halutaan joko kirjoittaa tai lukea muistista. EEADR rekisterissä taas on EEPROM alueen muistipaikan numero, josta EEDAT hakee tietonsa tai kirjoittaa tähän. Nämä ohjausrekisterit ovat 8-bittisiä ja ohjelmadata on prosessorista 14-bittisiä. Tämän takia joudutaan ottamaan toiset rekisterit yläpään bittejä varten. Nämä EEDATH ja EEADRH toimivat aivan samoin kuin alapään bittejä ohjaavat EEDAT ja EEADR. EECON1 ja EECON2 rekisterit ovat EEPROM muistin ohjausbittejä, joilla ohjataan EEPROM muistin kirjoitusta sekä sen kulloistakin tilaa. /7/

3.4 Tärkeimmät rekisterit

Seuraavaksi käsittelemme tärkeimmät rekisterit, jotka löytyvät datamuistista. Tärkein rekistereistä on W-rekisteri. W-rekisteriä käytetään yleisrekisterinä PIC prosessoreissa. Tätä työrekisteriä käytetään laskuissa sekä useissa muissa eri käskyissä. Tämän rekisterin avulla siirretään myös dataa muiden eri rekistereiden välillä.

3.4.1 STATUS rekisteri

Toinen tärkeimmistä rekistereistä on STATUS-rekisteri. Se sisältää tärkeimmät bitit prosessorin toiminnan kannalta. Taulukosta 2 voimme nähdä rekisterin sisällön. Taulukossa 3 puolestaan löytyvät näiden eri bittien toiminnot.

Taulukko 2 STATUS-rekisterin sisältö

Bitti	7	6	5	4	3	2	1	0
Toiminto	IRP	RP1	RP0	/TO	/PD	Z	DC	C

Taulokko 3 STATUS-rekisterin bittien merkitys

Bitti 7 IRP Bank-valinta	0	BANK0, BANK1	Epäsuora
	1	BANK2, BANK3	
Bitti 6-5 RP1-RP0	00	BANK0	Suora Bank Valinta
	01	BANK1	
	10	BANK2	
	11	BANK3	
Bitti 4 /TO	0	Piirin käynnistyksen jälkeen, CLRWDT- tai SLEEP-käsky	
	1	WDT-laskuri ympäri	
Bitti 3 /PD	0	Kun Sleep-operaatio on käynnissä	
	1	Piirin käynnistyksen jälkeen tai CLRWDT-käskyn jälkeen	
Bitti 2 Z	0	Aritmeettinen operaatio on muu kuin nolla	
	1	Aritmeettinen operaatio on nolla	
Bitti 1 DC	0	Ei ylivuotoa 5 bittiin	
	1	Ylivuoto 4 alimmasta bitistä viidenteen bittiin	
Bitti 0 C	0	Jos ei ole ylivuotoa	
	1	On ylivuoto, ylivuoto "yhdeksänteen" bittiin	

Erikseen näistä voidaan mainita vielä RP0- ja RP1-bitit. Näillä valitaan mitä datamuistin pankkia halutaan käyttää suoraan. STATUS-rekisteristä löytyy myös Z-, DC- ja C-bitit. Näitä bittejä ohjaavat monet eri prosessorin käyttämät käskyt. Z-bitti menee nollaksi aina, kun aritmeettisen käskyn lopputuloksen ollessa 0. DC-bittiä taas käytetään positiivisen ja negatiivisen alueen erottimena 8-bittisillä luvuilla. Kun tulos on suurempi kuin 0Fh on DC-bitti 1 ja vastaavasti tuloksen ollessa pienempi kuin 0Fh on DC-bitti 0. C-bitti on taas ylivuoto bitti eli jos laskutoimituksen mennessä yli luvun FFh, tällöin nousee C-bitti ykköseksi. /7/

3.4.2 PORT A-C, TRIS A-C, ANSEL- ja ANSELH-rekisterit

Nämä rekisterit ovat tärkeimpiä rekistereitä I/O toimintojen takia. PORTA-, PORTB- ja PORTC-rekistereistä voidaan sisääntuleva data lukea tai sinne voidaan kirjoittaa ulosmenevä data. Kuten taulukosta 4 voimme nähdä nämä rekisterit eivät ole samanpituisia. Ainostaan PORTC on täydet 8-bittiä PORTB ollessa vain 4-bittiä ja PORTA 6-bittiä. /7/

Taulukko 4 Rekisterit PORT A-C ja TRIS A-C

Rekitsteirt/bitit	7	6	5	4	3	2	1	0
PORTA			RA5	RA4	RA3	RA2	RA1	RA0
PORTB	RB7	RB6	RB5	RB4				
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
TRISA			RA5	RA4	RA3	RA2	RA1	RA0
TRISB	RB7	RB6	RB5	RB4				
TRISC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0

Rekisterit TRISA, TRISB ja TRISC määräävät taas porttien toiminnan eli ovatko ne sisään- vai ulostuloja. Jos TRISA-rekisterin bittiin RA5 kirjoitetaan 0, tarkoittaa tämä, että portti RA5 on ulostulo. Vastaavasti kirjoittamalla tähän bittiin 1, muuttuu portti RA5 sisääntuloksi. /7/

Prosessorin ANSEL- ja ANSELH-rekistereitä sitä vastoin käytetään määräämään onko portti analoginen vai digitaalinen. Tätä rekisteriä tarvitaan lähinnä vain A/D muunninta käytättäessä, jolloin sisääntulojen pitää olla analogisia. Rekistereitä on kaksi, koska PIC 16F690-prosessorilla on 12 kanavaa A/D muuntimia varten. Tällöin ANSEL-rekisterillä ohjataan kahdeksaa ensimmäistä porttia ja ANSELH-rekisterillä taas ohjataan loppuja neljää porttia.

3.4.3 Muut rekisterit

Prosessorilla on näiden tärkeimpien rekisterien lisäksi useita muita eri rekistereitä, joita käytetään aina eri toimintojen, kuten A/D muunnosten yhteydessä. Näitä rekistereitä on kuitenkin niin monta, että niitä kaikkia ei kannata käydä läpi tässä työssä. Kävin työn kannalta oleelliset ja samalla prosessorin toiminnan kannalta tärkeimmät rekisterit läpi.

4 OHJELMOINTI ALUSTA

Kappaleessa esitellään työssä käytetty piirien ohjelmointilaite ja tarvittavat ohjelmistot, joilla käännettiin assembler-kielinen koodi piirin ymmärtämään konekieliseen muotoon.

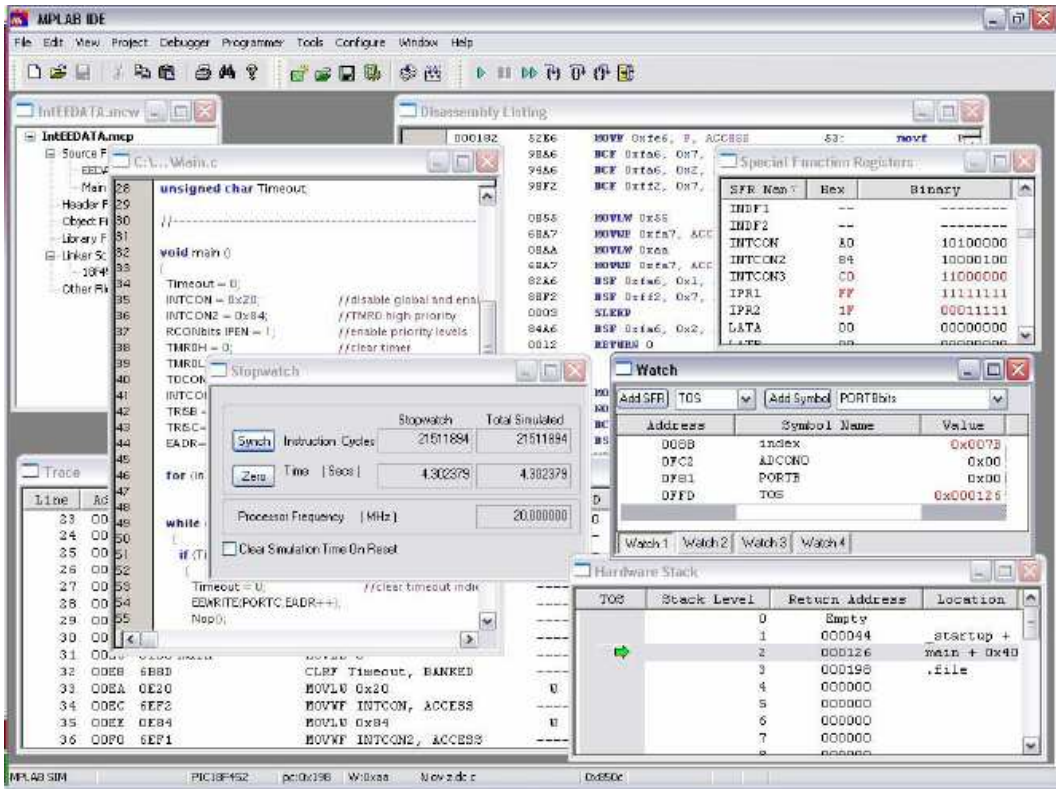
4.1 Käytettävä laite

Piirien ohjelmointi tapahtui Microchipin PICKIT2-laitteella (kuva36). PICKIT2 on USB-liitäntäinen ohjelmointialusta. Laitteella on mahdollista ohjelmoida PIC12F- ja PIC16F-sarjan kaikkia 8/14/20 pinnisiä mikroprosessoreita. Lisäksi mukana on kytkentälevy, johon mahtuu tekemään pieniä testikytkentöjä, ohjelmoidun piirin toiminnan testaamiseksi. Laitteen mukana tulevat myös kääntäjä c ja assemblerkieltä varten.

Kuva 36. PicKit2-piirinohjelmointilaite

4.2 Ohjelmisto

Itse ohjelmointi tapahtui MBLAB IDE 7.50 -ohjelmalla . Ohjelma on ilmainen ja haettavissa Microchipin kotisivulta. Ohjelma on monipuolinen kehitysohjelmisto PIC-piirejä varten. Ohjelmalla on mahdollista tietenkin kääntää assembler -kielistä koodia HEX -muotoon, jonka voi sitten siirtää piiriin. Lisäksi ohjelmassa on debug-toimintoja, koodivirheiden etsimiseksi. Ohjelmistolla voi myös simuloida ohjelmoidun piirin toimintaa. Käyttöliittymän kuva on kuvassa 37.



Kuva 37. MBLAB IDE 7.5 käyttöliittymä

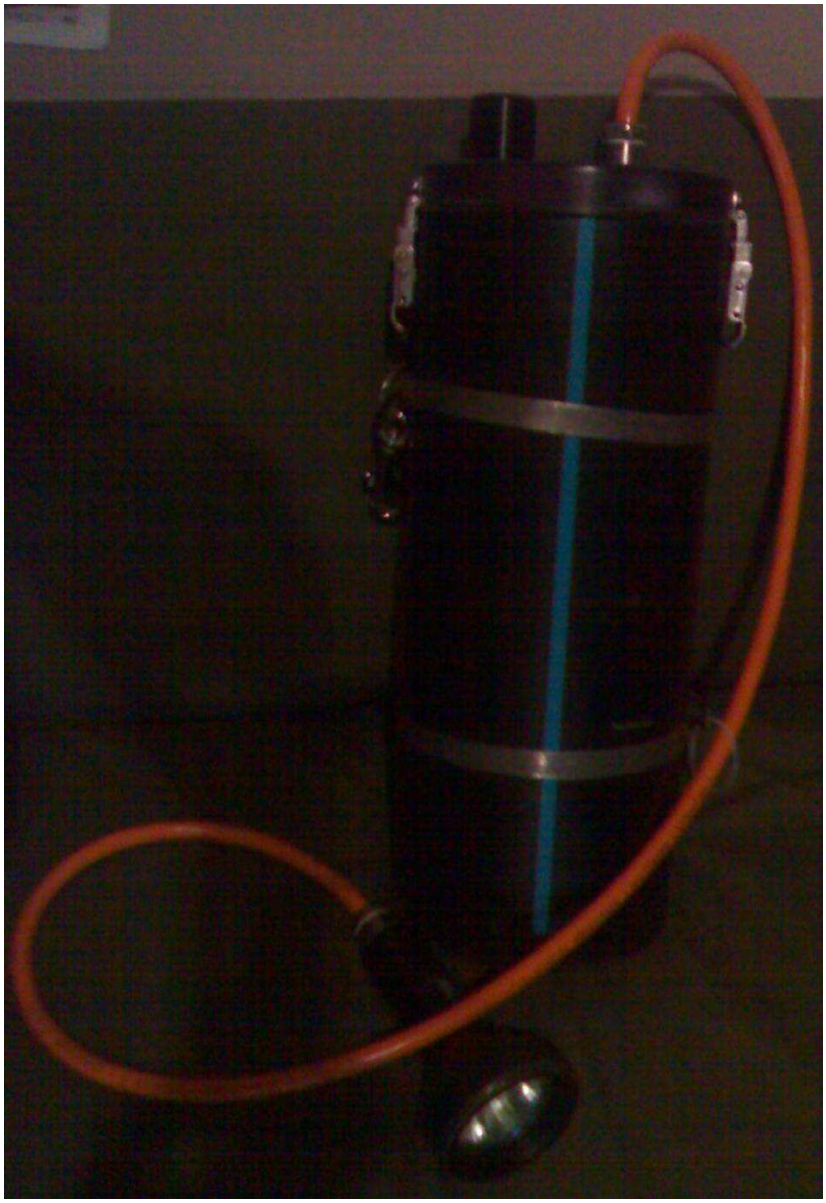
5. VALONOHJAUS

Tässä kappaleessa käymme ensin läpi lampun rakenteen, johon olen suunnittelemassa valonohjausta. Tämän jälkeen suunnittelemme tarvittavan kytkennän sekä piirille tarvittavan koodin pätkän. Lopuksi mittaamme lampunohjauksen toimintaa ja paloaikaa lampunohjauksen päällä ollessa.

5.1 Lampun esittely

Työ on suunniteltu omaa lamppuani varten (kuva38). Lamppu koostuu kahdesta osasta: akkukotelosta ja valopäästä. Lamppu ei sisällä mitään ylimääräistä elektroniikkaa, vaan akkukotelossa on kytkin, jota käyttämällä lampun johdot yhdistyvät suoraan akkuun. Akkukotelossa on kaksi 7.2Ah 12 voltin lyijyhyytelöakkua, jotka antavat lampulle sen tehosta riippuen 1-3h paloaikaa. Valopään lamppuna toimii normaali 12 volttinen heijastimella

varustettu halogeenilamppu, joita on saatavissa helposti 20 watista aina 75 wattiin asti. Lampussa on yleensä käytössä 50 watin lamppu, joka antaa parhaan valotehon paloaikaansa nähden.

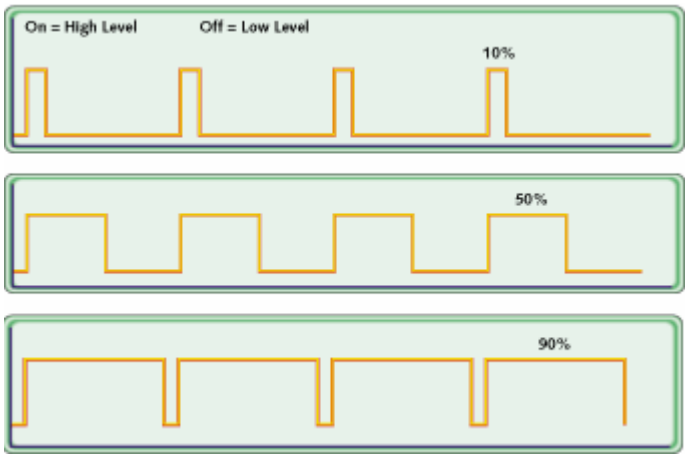


Kuva 38. Sukelluslamppu

5.2 Toiminnan periaate

Lampun ohjaus toteutetaan PWM-periaatteella eli pulssinleveysmodulaatiolla. Pulssinleveysmodulaatiossa kuormaan eli lamppuun menevää jännitettä katkotaan niin, että kuorma ei saa jatkuvasti täyttä jännitettä vaan osan ajasta jännite on nollissa. Tällöin, jos suhteeksi laitetaan 50 prosenttia eli signaali on puolet ajasta ylhäällä ja puolet alhaalla, lamppu toimii puolella sille

syötettävästä jännitteestä (kuva39). Tässä työssä käytetään 12V akkua joten 50 prosentin suhteella lamppu palaa 6 volttia vastaavalla teholla.



Kuva 39. PWM-ohjaussignaali

Pulssinleveysmodulaatiota käytetään muun muassa hakkurityyppisissä jännitelähteissä, D-luokan vahvistimissa ja sähkömoottoreiden sekä monenlaisten valaisimien tehonsäädössä. Tärkeän edun pulssinleveysmodulaation käytössä antaa kytkimenä toimiva komponentin oleminen suurimman osan ajasta joko johtavassa tilassa tai sen estäessä virran kulun kokonaan. Tällöin siinä ei tapahdu suurta tehohäviötä ja laitteen hyötysuhde säilyy korkeana verrattuna muuttuvaan resistanssiin perustuviin säätimiin, himmentimiin ja vahvistimiin./2/

5.3 Suunnittelu

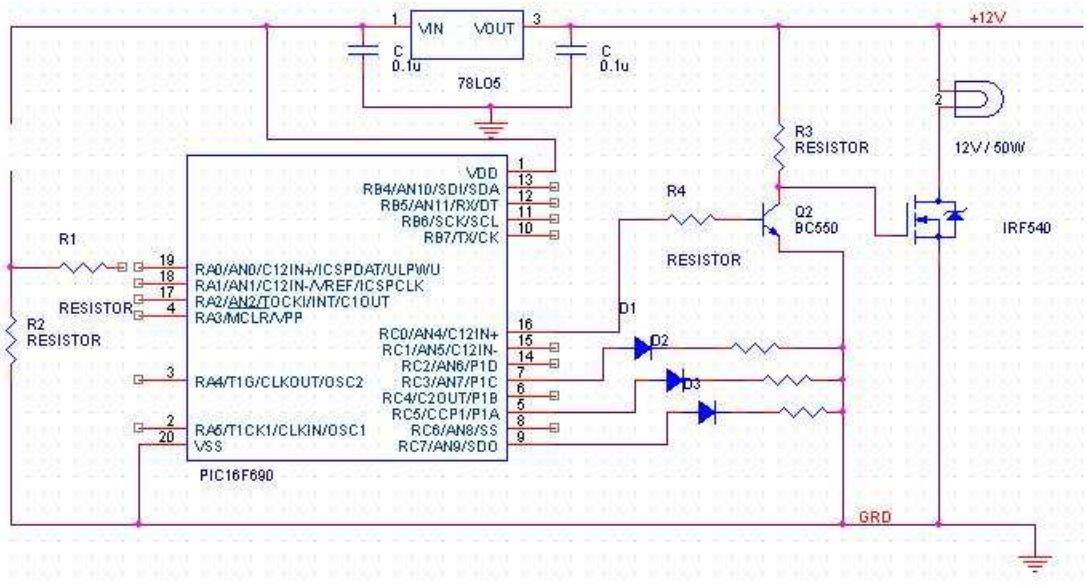
Suunnittelu aloitettiin kytkennässä tarvittavien komponenttien kartoituksella ja ohjauksen suunnittelun toteutuksella. Ohjauksen päätin toteuttaa yhden painikkeen avulla, koska se oli helpoin toteuttaa. Täten lamppuun ei tarvitse tehdä ylimääräisiä läpivientejä. Käytän siis alkuperäistä kytkintä, joka on toiminut ennestään lampun ja akkujen välillä suoraan. Seuraavaksi piti ratkaista mitä ulostuloja piiristä tarvittaisiin. Tietenkin valonohjausulostulo tarvitaan ohjaamaan valon PWM-signaalia. Lisäksi päätin tehdä valot osoittamaan

lampun ohjauksen kulloisenkin tilan. Nämä valot toteutetaan erivärisillä ledeillä, jotka kytketään piirin vapaisiin jalkoihin. Valot sijoitetaan akkukotelon sisälle, josta ne näkyvät hyvin läpinäkyvän polykarbonaatti kannen läpi. Koska PIC 16F690 on 20 jalkanen prosessori jää vapaita I/O portteja vielä tämänkin jälkeen vapaaksi.

Seuraavaksi suunnittelin itse kytkennän, jolla nämä toiminnot saadaan toimimaan. Seuraava kappale 5.4 käsittelee kytkentää ja sen suunnittelun vaiheita. Kytkennän jälkeen pääsin suunnittelemaan mikroprosessorille koodia, jolla halutut toiminnot saadaan kytkentään. Itse koodattu ohjelma löytyy kappaleesta 5.5 ja koodin selitys kohdittain kappaleesta 5.6.

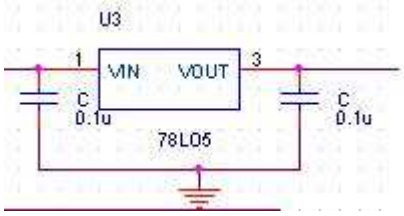
5.4 Kytkentä

Kytkentä tehtiin mahdollisimman vähillä komponenteilla (kuva40). Kytkennän voi jakaa kolmeen eri osaan: Virran syöttö, lampun valonohjaus ja merkkiledit.



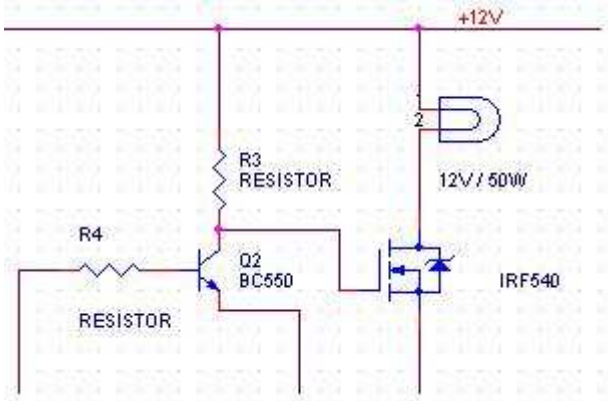
Kuva 40. Käytettävä kytkentä

Mikroprosessori käyttää viiden voltin jännitettä, joten akuista tuleva 12 voltin jännite tarvitsee pudottaa mainittuun viiteen volttiin (kuva41). Tämä tapahtuu regulaattorin avulla. Regulaattorin kummallekin puolelle asetetaan vielä kondensaattorit, jotka ovat tasaamassa jännitettä.



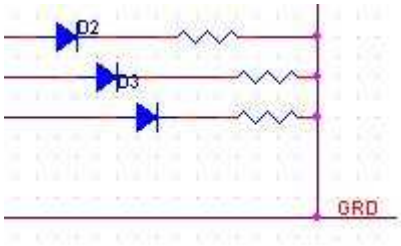
Kuva 41. piirin virransyöttö

Valonohjaus tehdään fetin avulla. Koska fetin ohjaukseen tarvitaan 12 voltin jännite, tarvitaan yksi transistori nostamaan jännite tarvittavalle tasolle(kuva 42).



Kuva 42. Valonohjaus

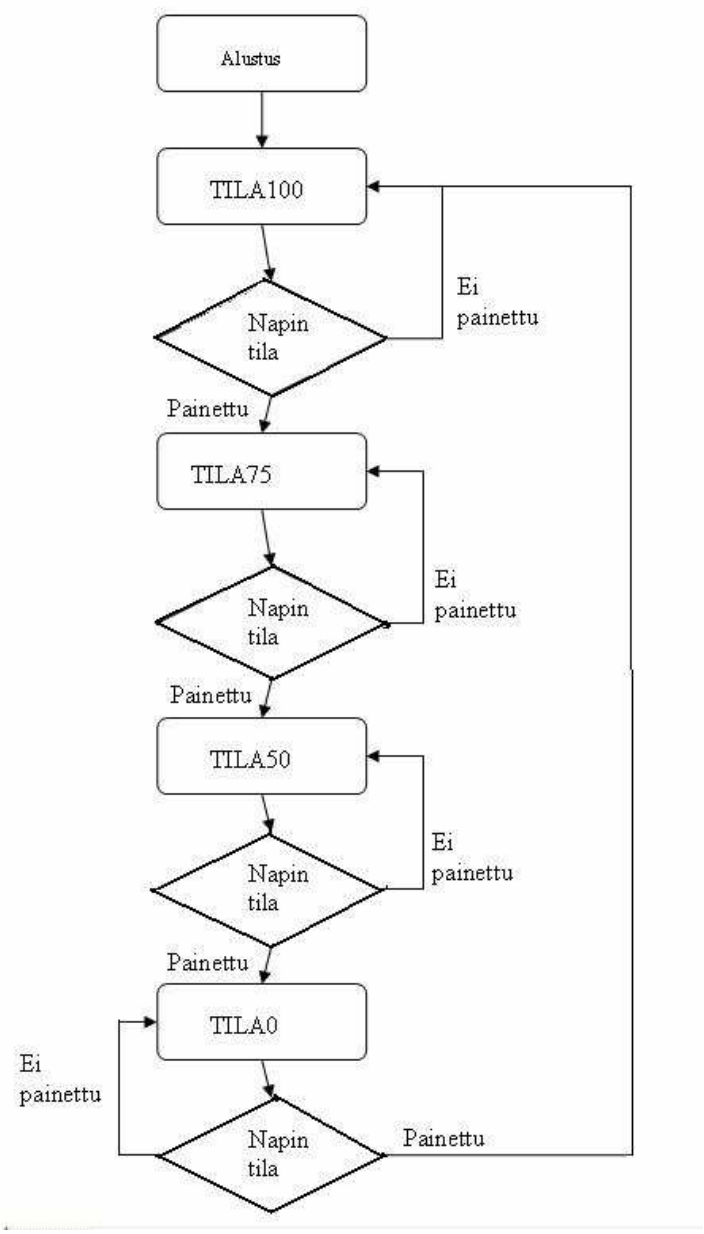
Ledit, jotka ilmoittavat tilan ovat kytkettynä suoraan piirin jalkoihin ja siitä edelleen 470 ohmin vastuksen kautta maihin(kuva 43).



Kuva 43. Merkki ledit

5.5 Koodi

Koodin suunnittelu lähti käyntiin vuokaavion suunnittelulla, josta näkee ohjelman kulun helposti. (Kuva 44). Ensin ohjelma alustaa itsensä ja sen jälkeen menee suoraan sadan prosentin tilaan. Tämän jälkeen ohjelma tarkastaa painikkeen tilan ja sen jälkeen joko jatketaan nykyisessä tilassa tai siirrytään seuraavaan 75% tehon tilaan. Näin jatketaan myös seuraavien tilojen kanssa.



Kuva 44. Koodin vuokaavio

Tämän jälkeen suunnittelin ohjelmasta pseudo-koodin, minkä avulla varsinaisen assembler-kielisen ohjelman kirjoitus helpottuu. Liitteessä 1 on ohjelma kirjoitutettuna pseudo-kielelle. Seuraavassa vaiheessa ohjelma kirjoitettiin kokonaan assembler-kielellä. Ohjelman ollessa oli jo kirjoitettuna pseudo-koodilla oli se helppo kirjoittaa PIC:n assembler käskyillä. Liitteessä 2 on ohjelman lähdekoodi kokonaisuudessaan.

5.6 Koodin esittely

Tässä osiossa käymme läpi kytkennässä käytettävän lähdekoodin, joka on siirretty kytkennän PIC 16F690 prosessoriin.

```
#include <p16F690.inc>
```

Ladataan prosessorin omat ohjaustiedot jotka löytyvät kääntäjän tiedoista.

```
__config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF &
_MCLRE_OFF & _CP_OFF  & _IESO_OFF & _FCMEN_OFF)
```

a

Käsky rivillä konfiguroidaan prosessorin asetukset, joita tässä koodissa käytetään. Käskyt asettavat prosessorin rekistereitä haluttuun muotoon. Tärkeimpänä käskyistä voidaan mainita _INTRC_OSC_NOCLKOUT, jolla asetetaan prosessori käyttämään sisäistä RC oskillaattoriaan. Koska koodissa ei säädetä tämän RC oskillaattorin nopeutta erikseen, toimii kytkentä 4Mhz taajuudella.

```
cblock 0x20
viive
```

Määritetään muuttujille rekisteri omat osoitteet. Tässä tapauksessa viive saa osoitteen 0x20, jos tarvitsisimme lisää omia rekistereitä, voisimme lisätä ne vain listana peräkkäin ja ne saisivat osoitteensa aina seuraavasta vapaasta rekisteripaikasta eli 0x20,0x22,0x24…

```
Alku
            bsf         STATUS,RP0
            movlw       0xFF
            movwf       TRISA
            clrf        TRISC
            bcf         STATUS,RP0
            movlw    b'00000000'
            movwf    PORTC
```

Aloitetaan ohjelma siirtymällä Pankkiin1, jossa sijaitsee TRISA JA TRISC rekisterit. Tarvitsemme yhden ulostulon ja 4 ulostuloa kytkentää varten. Portteja on tarjolla runsaasti, joten asetamme kaikki PORTA portit sisääntuloiksi kirjoittamalla TRISA rekisteriin pelkkää 1 ja samoin teemme PORTC porteista ulostuloja, kirjoittamalla niihin pelkkää nollaa. Lopuksi vielä kaikki ulostulo portit nollataan kirjoittamalla niihin nollaa.

```
Tila100
            BTFSS       PORTA,3
            GOTO        Tila100
```

Ensimmäisen tilan koodi alkaa tarkastamalla onko kytkin auki vai ei. Jos kytkin on vielä auki , hypätään takaisin alkuun ja tarkastetaan napin tila uudestaan. Tämä tehdään sen takia, että kytkintä kerran painamalla tila vaihtuu vain seuraavaan tilaan, eikä hypi tilojen välillä.

```
movlw   b'00100000'
movwf   PORTC
```

Seuraavaksi asetamme ulostulot toimimaan oikein tässä tilassa. Portti RC6 kytkettynä oleva ledi sytytetään ja toiset ledit sammutetaan. Portissa RC0 oleva lampun ohjaus taas kytketään päälle asettamalla portin tilaksi 0.

```
CALL viivesilmukka    ; kulutetaan aikaa
CALL viivesilmukka    ; kulutetaan aikaa
CALL viivesilmukka    ; kulutetaan aikaa
CALL viivesilmukka    ; kulutetaan aikaa
```

Kutsutaan viivesilmukka aliohjelmaa, jonka tehtävänä on kuluttaa aikaa PWM ohjausta varten. Tässä tilassa näitä käskyjä ei varsinaisesti vielä tarvitsisi, koska valonohjaus pysyy päällä kokoajan. Rivit on lisätty vain sen takia, että ohjelman eri tilojen suoritus veisi suunnilleen yhtä paljon aikaa.

```
BTFSS       PORTA,3
GOTO        Tila75
GOTO        Tila100
```

Ensimmäisen tilan lopussa tutkitaan jälleen kytkimen asentoa. Jos kytkin pysyy kiinni, hypätään saman tilan alkuun. Jos taas kytkin on avattuna, hypätään seuraavaan tilaan.

```
Tila75
            BTFSS       PORTA,3
            GOTO Tila75
            movlw   b'00001000'
            movwf   PORTC
            CALL viivesilmukka
            CALL viivesilmukka
            CALL viivesilmukka
            movlw   b'00001001'
            movwf   PORTC
            CALL viivesilmukka
            BTFSS       PORTA,3
            GOTO Tila50
            GOTO Tila75
```

Seuraavassa tilassa on tarkoitus polttaa lamppua 75 prosentin PWM suhteella. Tila toimii samoin kuin edellinen tila, mutta ennen viimeistä viivesilmukka aliohjelmaa valonohjaus signaali otetaan pois päältä, jolloin lamppu saa virtaa

noin kolme neljäsosaa aikaa. Samoin merkki ledi vaihdetaan RC3 porttiin kytkettynä olevaan keltaiseen lediin.

```
Tila50
            BTFSS      PORTA,3
            GOTO Tila50
            movlw   b'10000000'
            movwf   PORTC
            CALL viivesilmukka
            CALL viivesilmukka
            movlw   b'10000001'
            movwf   PORTC
            CALL viivesilmukka
            CALL viivesilmukka
            BTFSS      PORTA,3
            GOTO Tila0
            GOTO Tila50
```

Seuraavassa tilassa lampun ohjaus on päällä puolet ajasta. Tällöin lampun ohjaus on päällä kaksi viivesilmukkaa ja toiset kaksi se on pois päältä. Tilan merkki ledi on kytkettynä porttiin RC7.

```
Tila0
            BTFSS      PORTA,3
            GOTO Tila0
            movlw   b'10101001'
            movwf   PORTC
            CALL viivesilmukka
            CALL viivesilmukka
            CALL viivesilmukka
            CALL viivesilmukka
            BTFSS      PORTA,3
            GOTO Tila0
            GOTO Tila100
```

Viimeisessä tilassa lampun ohjaus on sammutettuna kokoajan. Tilan merkiksi kaikki kolme merkki lediä sytytetään.

```
viivesilmukka
            DECFSZ      viive,f
            GOTO        viivesilmukka
            RETURN
            END
```

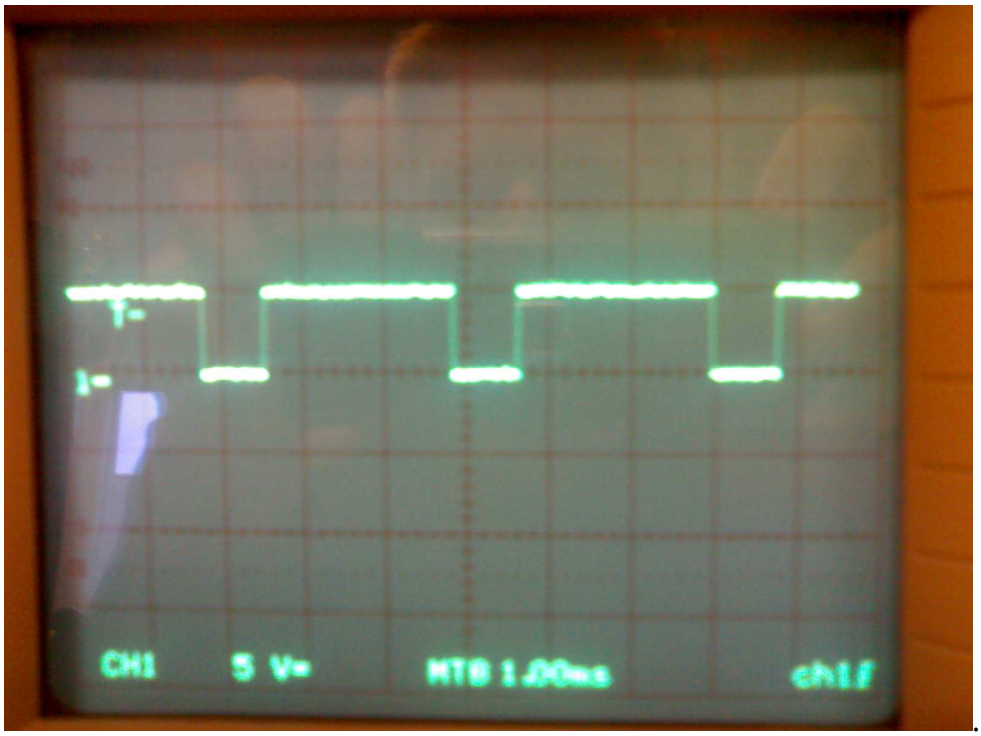
Viimeisenä lähdekoodissa on viivealiohjelma. Tämän koodin pätkän tarkoituksen on pelkästään kuluttaa aikaa, jolloin PWM tilojen jakson pituudet saadaan riittäviksi, jotta ulkoinen kytkentä pysyy mukana tahdissa.

6. MITTAUKSET

Tässä osiossa testaamme, kuinka paljon hyödymme tämän kytkennän käyttämisestä lampussa. Mittaamme lampun purkukäyrän jokaisella tilalla, josta näemme paljonko kukin tila tuo lisää paloaikaa lampulle. Tämän lisäksi mittaamme oskilloskoopin avulla PWM työ- ja lepojaksot.

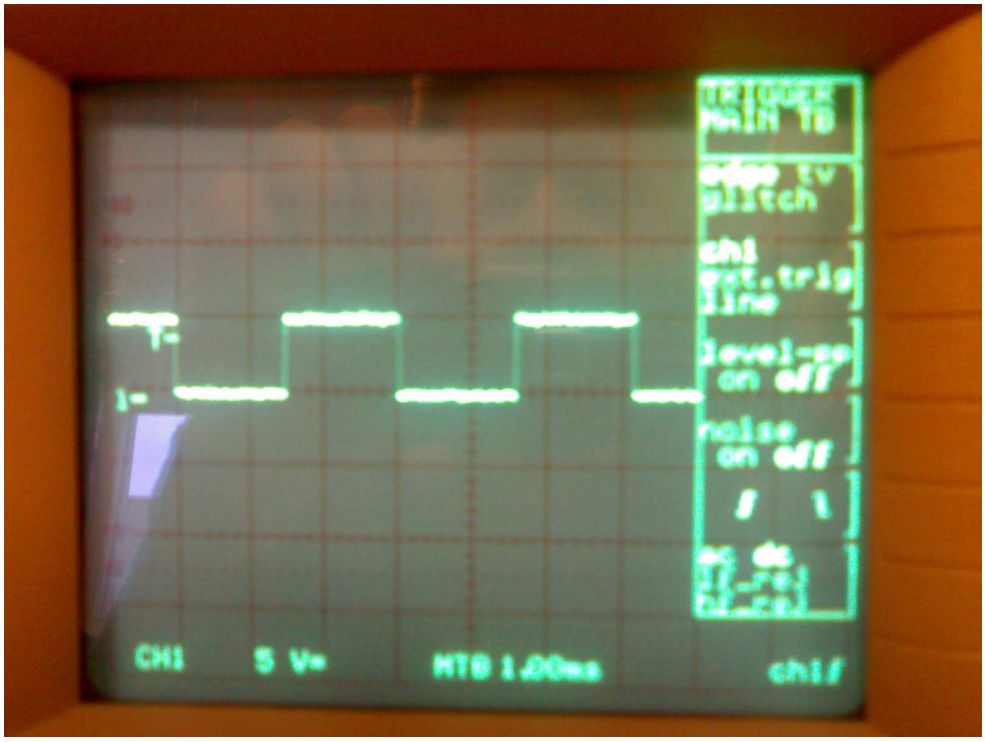
6.1 PWM tilojen pituudet.

Tarkastimme oskilloskoopilla eri tilojen toiminnan. Kuvassa 45 näkyy ohjaus signaali 75% tilalla. Kuten kuvasta nähdään, on ohjaus päällä ¾ osan ajasta ja pois päältä 1/4 ajan. Kuvasta voidaan myös katsoa yhden viivejakson pituus joka on noin sen pituinen, kuin signaalinohjaus on pois päältä. Kuvassa tämä on noin 0,75ms pituinen ja laskennallisesti sain pituudeksi 4Mhz kellotaajuutta käyttäessä saman noin 0,75ms. Yksi kierros viive aliohjelmaa sisältää 768 käskyä ja suoritin pystyy suorittamaan noin miljoona käskyä sekunnissa, joten viivesilmukan suoritus kestää 0.768 ms. Tällöin ohjaussignaalin taajuudeksi tulee maksimissaan 1,3kHz. Tämän kytkentä kestää vielä hyvin, joten viivesilmukan pituutta olisi mahdollista vielä lyhentää. Tämä ei ole kuitenkaan tarpeen, sillä lampun valossa ei huomaa mitään vilkuntaa.



Kuva 45. Ohjaus signaali 75% tilassa.

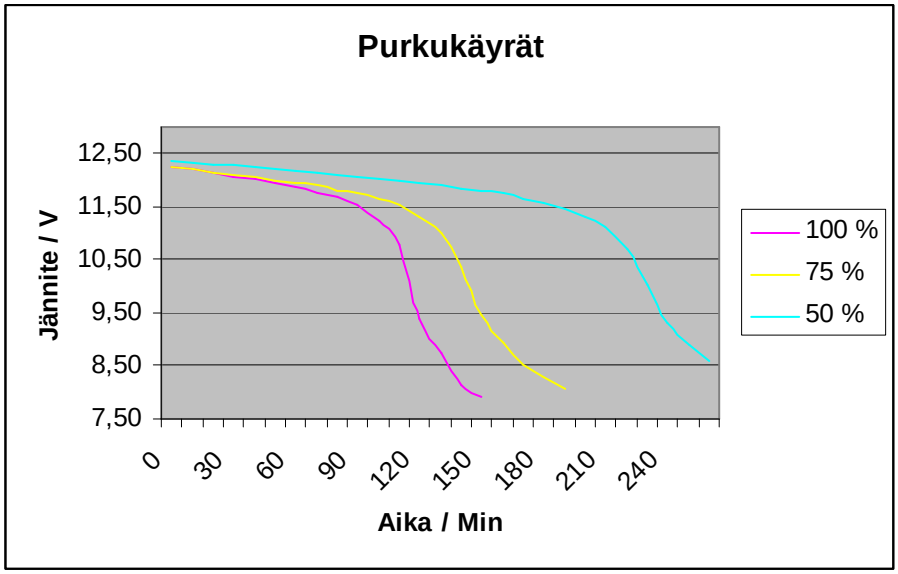
Kuvassa 46 on taas 50% tilan oskilloskooppi kuva. Tila toimii aivan oikein, sillä ohjaus signaali on puolet ajasta pois ja puolet päällä.



Kuva 46. Ohjaussignaali 50% tilassa.

6.2 Purkukäyrä

Mittasimme akkujen purkukäyrän eri tiloilla (kuva 47). Tästä mittauksesta näemme parhaiten, kuinka paljon paloaika lisääntyy kytkennän avulla.



Kuva 47. Purkukäyrät

Normaali jännite alkaa tippua rajummin noin 120minuutin jälkeen. Käytättäessä 75% tilaa saadaan paloaikaa lisää noin puolisen tuntia ja valotehossa ei ole suurta muutosta. Viimeisessä 50% tilassa taas paloaika pitenee huomattavasti lähemmäs neljää tuntia, mutta valoteho on huomattavasti huonompi kuin käytettäessä täyttä tehoa.

7 YHTEENVETO

Työssä saatiin tehtyä kytkentä, jolla saatiin lisättyä lampun paloaikaa tarvittaessa. Kytkennän tarkoitus siis täyttyi. Ohjaus toimii yhden kytkimen avulla jolloin pystyin hyödyntämään lampussa valmiiksi ollutta kytkintä ja samalla ohjaus pysyy yksinkertaisena, jolloin sen käyttö onnistuu myös pimeässä. Laitetta on toistaiseksi testattu vain pinnalla, jolloin sen käytössä ei

ole ilmennyt suurempia ongelmia. Kun kytkentää päästään käyttämään itse sukelluksen aikana, voidaan todeta onko eri tilojen valotehot riittäviä. PWM tilojen pituutta muuttamalla voidaan tämän jälkeen säätää tehoa, jos se on tarpeen.

LÄHTEET

Sähköiset lähteet

1 Guide to use the PIC.[WWW-sivu].[viitattu 17.4.2007] Saatavissa http://www.interq.or.jp/japan/se-inoue/e_pic.htm

2 Wikipedia. [WWW-sivu]. [viitattu 17.4.2007] Saatavissa http://fi.wikipedia.org/wiki/Pulssinleveysmodulaatio

3 Wikipedia. [WWW-sivu]. [viitattu 17.4.2007] Saatavissa http://en.wikipedia.org/wiki/PIC_microcontroller

4 Wikipedia. [WWW-sivu]. [viitattu 17.4.2007] Saatavissa http://en.wikipedia.org/wiki/Harvard_architecture

5 the P.I.C. page. [WWW-sivu]. [viitattu 17.4.2007] Saatavissa http://www.jpixton.dircon.co.uk/pic/history.html

6 Microchip kotisivu . [WWW-sivu]. [viitattu 17.4.2007] Saatavissa http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=74

7 PIC 16F690 Datalehdet. [WWW-sivu]. [viitattu 17.4.2007] Saatavissa http://ww1.microchip.com/downloads/en/DeviceDoc/41262D.pdf

8 Hobby of Electronic Circuit Engineering. [WWW-sivu]. [viitattu 17.4.2007] Saatavissa http://www.interq.or.jp/japan/se-inoue/e_menu.htm

LIITTEET

```
        Alku alustukset
        Aseta portit
        Mene “alku”
Alku
        Hyppää TILA100

TILA 100
        Tarkista napin tila
        Jos painettu  Hyppää takaisin TILA 100
        Asetetaan Lampun ohjauspäälle
        Merkki ledi 1 päälle ja edellinen pois
        Kulutetaan aikaa x4
        Tarkastetaan nappi
        Jos painettu hyppy TILA75
        Muuten hyppy TILA100

TILA 75
        Tarkista napin tila
        Jos painettu  Hyppää takaisin TILA 75
        Asetetaan Lampun ohjauspäälle
        Merkki ledi 2 päälle ja edellinen pois
        Kulutetaan aikaa x3
        Lampunohjaus pois
        Kulutetaan aikaa x1
        Tarkastetaan nappi
        Jos painettu hyppy TILA50
        Muuten hyppy TILA75

TILA 50
        Tarkista napin tila
        Jos painettu  Hyppää takaisin TILA 50
        Asetetaan Lampun ohjauspäälle
        Merkki ledi 3 päälle ja edellinen pois
        Kulutetaan aikaa x2
        Lampunohjaus pois
        Kulutetaan aikaa x2
        Tarkastetaan nappi
        Jos painettu hyppy TILA0
        Muuten hyppy TILA50
TILA 0
        Tarkista napin tila
        Jos painettu  Hyppää takaisin TILA 0
        Edellinen merkki ledi pois
        Lampunohjaus pois
        Kulutetaan aikaa x4
        Tarkastetaan nappi
        Jos painettu hyppy TILA100
        Muuten hyppy TILA
```

```
#include <p16F690.inc>
        __config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF &
_MCLRE_OFF & _CP_OFF  & _IESO_OFF & _FCMEN_OFF)

        cblock 0x20
viive                                   ;määritetään rekisteri viiveen laskennalle
        endc

        org 0
        GOTO Alku
Alku
        bsf         STATUS,RP0          ; Valitaan pankki 1 asetukset
        movlw       0xFF
        movwf       TRISA               ; porteista A tehdään sisäänmenoja
        clrf        TRISC               ; Porteista C tehdään taas ulostuloja
        bcf         STATUS,RP0          ; Palataan pankin 0 käsittelyyn
        movlw   b'00000000'             ;kaikki portit nollataan
        movwf   PORTC

Tila100
        BTFSS       PORTA,3     ; Ohitetaan seuraava jos painiketta 0 EI paineta
        GOTO Tila100            ; Jos painiketta painetaan odotetaan että sitä ei enää paineta
        movlw   b'00100000'     ;Vihreä ledi päälle valon ohjaus päälle
        movwf   PORTC
        CALL viivesilmukka      ; kulutetaan aikaa
        CALL viivesilmukka      ; kulutetaan aikaa
        CALL viivesilmukka      ; kulutetaan aikaa
        CALL viivesilmukka      ; kulutetaan aikaa
        BTFSS       PORTA,3     ; Ohitetaan seuraava jos painiketta 0 EI paineta
        GOTO Tila75             ; hypätään seuraavaan tilaan jos painiketta ei paineta
        GOTO Tila100            ; Hypätään takaisin tilan alkuun

Tila75

        BTFSS       PORTA,3     ; Ohitetaan seuraava jos painiketta 0 EI paineta
        GOTO Tila75             ; Jos painiketta painetaan odotetaan että sitä ei enää paineta
        movlw   b'00001000'     ;keltainen ledi päälle valon ohjaus päälle
        movwf   PORTC
        CALL viivesilmukka      ; kulutetaan aikaa
        CALL viivesilmukka      ; kulutetaan aikaa
        CALL viivesilmukka      ; kulutetaan aikaa
        movlw   b'00001001'     ;keltainen ledi päälle valon ohjaus pois päältä
        movwf   PORTC

        CALL viivesilmukka      ; kulutetaan aikaa
        BTFSS       PORTA,3     ; Ohitetaan seuraava jos painiketta 0 EI paineta
        GOTO Tila50             ; hypätään seuraavaan tilaan jos painiketta ei paineta
```

```
            GOTO Tila75             ; Hypätään takaisin tilan alkuun

Tila50
            movlw   b'11111110'     ;kaikki ledit päälle valon ohjaus pois päältä
            movwf   PORTC
            BTFSS       PORTA,3     ; Ohitetaan seuraava jos painiketta 0 EI paineta
            GOTO        Tila50      ; Jos painiketta painetaan odotetaan että sitä ei enää paineta
            movlw   b'10000000'     ; Punainen ledi päälle valon ohjaus päälle
            movwf   PORTC
            CALL viivesilmukka      ; kulutetaan aikaa
            CALL viivesilmukka      ; kulutetaan aikaa
            movlw   b'10000001'     ; Punainen ledi päälle valon ohjaus pois päältä
            movwf   PORTC
            CALL viivesilmukka      ; kulutetaan aikaa
            CALL viivesilmukka      ; kulutetaan aikaa
            BTFSS       PORTA,3     ; Ohitetaan seuraava jos painiketta 0 EI paineta
            GOTO Tila0              ; hypätään seuraavaan tilaan jos painiketta ei paineta
            GOTO Tila50             ; Hypätään takaisin tilan alkuun

Tila0

            BTFSS       PORTA,3     ; Ohitetaan seuraava jos painiketta 0 EI paineta
            GOTO Tila0              ; Jos painiketta painetaan odotetaan että sitä ei enää paineta
            movlw   b'00000001'     ;Valonohjaus pois, samoin ledit
            movwf   PORTC
            CALL viivesilmukka      ; kulutetaan aikaa
            CALL viivesilmukka      ; kulutetaan aikaa
            CALL viivesilmukka      ; kulutetaan aikaa
            CALL viivesilmukka      ; kulutetaan aikaa
            BTFSS       PORTA,3     ; Ohitetaan seuraava jos painiketta 0 EI paineta
            GOTO Tila100            ; hypätään seuraavaan tilaan jos painiketta ei paineta
            GOTO Tila0              ; Hypätään takaisin tilan alkuun


viivesilmukka
            DECFSZ      viive,f             ; Kulutetaan aikaa
            GOTO        viivesilmukka       ;
            RETURN


end
```