TAMPEREEN AMMATTIKORKEAKOULU
Kone- ja tuotantotekniikka
Kone- ja laiteautomaatio

Opinnäytetyö

Ville Ranta

MIKROKONTROLLERIN OPISKELUYMPÄRISTÖ JA OHJELMOINTIOPAS

Työn ohjaaja Lehtori Leo Sutinen
Työn teettäjä Tampereen ammattikorkeakoulu, kone- ja laiteautomaatiolaboratorio
Tampere 2010

TAMPEREEN AMMATTIKORKEAKOULU
Kone- ja tuotantotekniikka
Kone- ja laiteautomaatio
Ranta, Ville
Mikrokontrollerin opiskeluympäristö ja ohjelmointiopas
Opinnäytetyö 76 sivua
Työn ohjaaja Lehtori Leo Sutinen
Työn teettäjä TAMK, Kone- ja laiteautomaatiolaboratorio
Maaliskuu 2010
Hakusanat mikrokontrolleri, mikro-ohjain, flowcode, ohjelmointi, pic

TIIVISTELMÄ

Tämä opinnäytetyö on tehty Tampereen ammattikorkeakoulun kone-ja laiteautomaatiolaboratoriolle. Työhön kuuluu opetuslaitteiston järjestely ja mikrokontrollerien harjoitteluopas.

Jotta mikrokontrollereita voidaan käyttää automaatiojärjestelmien ohjauksessa, täytyy tuntea niiden rakenne, kuinka ohjaimet kytketään ulkoisesti johonkin järjestelmään ja mitä kaikkea ne tarvitsevat toimiakseen. Tämän opinnäytetyön tarkoitus on helpottaa ja selventää mikrokontrollerien opiskelua ja käyttöä.

Opinnäytetyön raportti toimii opiskeluympäristön käyttöohjeena. Opiskeluympäristöön kuuluu E-block-harjoittelulaitteisto, jonka käyttöä pyrittiin yksinkertaistamaan lajittelemalla irralliset osat salkkuihin. Laitteiston osat ja niiden käyttö on kuvattu ohjeiden alussa.

Ohjeissa kerrotaan pic-mikrokontrollereiden sisäisestä rakenteesta ja toiminnasta. Mikro-ohjainten kytkemiseen elektroniseen piiriin tutustutaan muutamilla esimerkeillä.

Ohjaimen ohjelmointiin käytetään Flowcode ohjelmaa. Ohjelman toiminnoista on pyritty kertomaan selkeästi ja kattavasti, jotta mikrokontrolleriin ladattavan ohjelman laatiminen olisi vaivatonta.

Raportin lopussa on esitelty monipuolinen esimerkkiohjelma. Esimerkkiohjelman esittelyssä on pyritty kertomaan enemmän ohjelman toimintaperiaatteesta. Esittelyä ei kerrota yksityiskohtia myöten, lukijalle on yritetty jättää jotain ajateltavaa.

TAMPERE UNIVERSITY OF APPLIED SCIENCES
Mechanical and Production Engineering
Mechanical Engineering
Ranta, Ville
Learning environment for microcontrollers and programming guide
Engineering thesis 76 pages
Thesis supervisor Sutinen Leo
Commissioning company TAMK, Automation laboratory
March 2010
Keywords microcontroller, flowcode, programming, pic

ABSTRACT

This engineering thesis is made for automation laboratory at Tampere University of applied sciences. This is a learning manual for microcontrollers.

Microcontrollers are used as brains in automated systems. To use these controllers one needs to know their structure. How to integrate the controller to some system and what they need for proper function. The purpose of this diploma work is to help reader to learn about microcontrollers and how to use them.

This report serves as a user guide of learning environment. Learning environment includes E-blocks for practical studies. The hardware components and their use are described at the beginning of the instructions.

PIC microcontroller's internal structure and functions are told in the instructions. How to connect the microcontroller with other electrical circuits are covered by few examples.

Flowcode program is used to program the controller. Flowcode program is presented closely, so that the microcontroller’s program would be easy to build.

At the end of the report there is an example program.

SISÄLLYSLUETTELO

1 JOHDANTO

Tämä teos on Tampereen ammattikorkeakoulun insinööriopintoihin kuuluva opinnäytetyö. Työn tilaaja on Tampereen ammattikorkeakoulun kone- ja laiteautomaatiolaboratorio.

Osana kone- ja laiteautomaatioinsinöörin opintoja tulee opiskella mikro-ohjaimien toimintaa, ohjelmointia ja käyttöä. Opiskelua varten Tampereen ammattikorkeakoulu käyttää Matrix Multimedian kehittämiä E-Block-sarjan tuotteita. E-Block-sarja koostuu erillisistä yksiköistä tai komponenteista, joita voi liittää toisiinsa kulloisenkin harjoituksen mukaan. Mikro-ohjainten ohjelmointiin käytetään Flowcode-ohjelmaa.

Ohjelmoinnin opiskelu ja ohjeiden lukeminen voi olla raskasta. Tämänkään oppaan teksti ei varmasti aukene kerrasta. Kehoittaisin opiskelijaa kuitenkin lukemaan tämän tekeleen kertaalleen läpi takertumatta liiaksi yksityiskohtiin. Lukemisen jälkeen tiedon löytäminen sisällysluettelon avulla on paljon helpompaa.

2 TAVOITE JA RAJAUKSET

Tämän opinnäytetyön yhtenä tehtävänä on järjestää kone- ja laiteautomaatiolaboratorion E-Block-komponentit opiskelua helpottaviksi sarjoiksi, myös kestävyys ja säilytys huomioon ottaen. Opinnäytetyön harjoittelulaitteistoon kuuluu E-Block-sarjan ohjelmointikortti (programmer), kytkinkortti (switch board), ledkortti (led board), anturikortti (sensor board), näyttökortti (lcd board) ja kytkentäkortti (proto board).

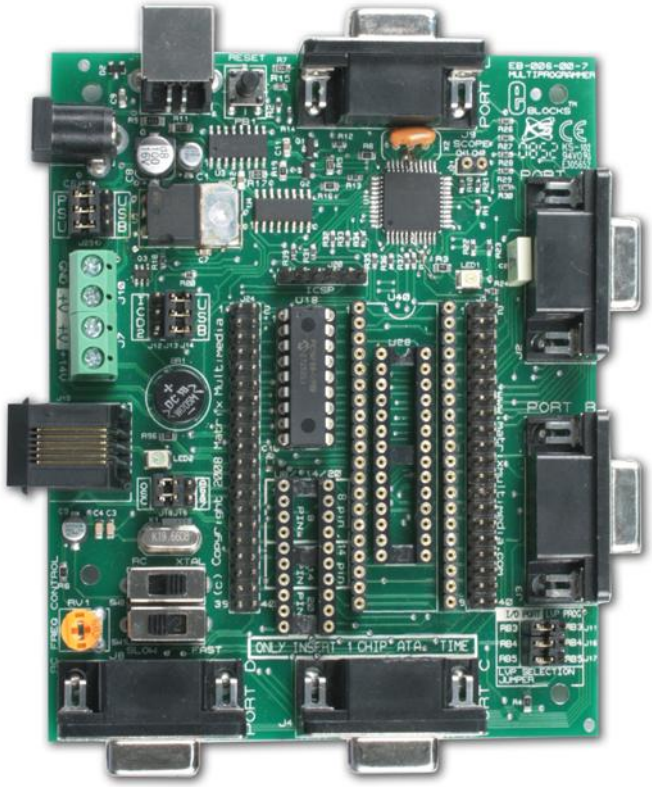
Opinnäytetyön toinen osuus on käyttöohjeistuksen laatiminen koulun E-Block-sarjoille. Ohjeistuksessa keskitytään tarkastelemaan Microchipin valmistamaa PIC-microkontrolleria ja sen ohjelmointia, esimerkkejä käyttäen. Ohjelmat laaditaan Flowcode-ohjelmistolla, joka on myös Matrix Multimedian kehittämä. Flowcode-ohjelman versio on 3. Flowcode-ohjelman sisältämistä toiminnoista käydään läpi vain ne, mitkä liittyvät edellä mainittuun laitteistoon.

Tämän opinnäytetyön tarkoituksena on siis toimia mikrokontrollereiden helppokäyttöisenä ja selkeänä opiskelumateriaalina kone- ja laiteautomaatio-opiskelijoille.

Työhön ei kuulu laitteiston ja ohjeistuksen testaus opiskelutilanteessa.

3 E-BLOCK-HARJOITTELULAITTEISTO

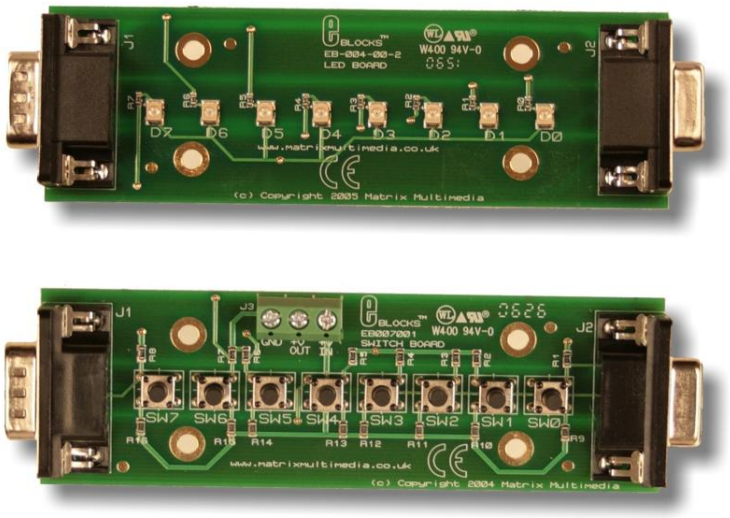
Laitteiston kokoaminen lähti liikkeelle pahvilaatikon sisällä pyörivistä, irrallisista E-Block-korteista. Tärkein ja välttämättömin näistä kaikista korteista on ohjelmointikortti (kuva 1). Käyttö mietittiin sen ympärille.



Kuva 1. Harjoittelulaitteiston ohjelmointikortti.

Ohjelmointikortti pitää sisällään, ohjelmoitavan mikrokontrollerin lisäksi, kaikki ohjelmointiin ja käyttöön tarvittavat piirit, komponentit ja liitännät. Muut oheiskortit saavat tarvittaessa käyttöjännitteensä ohjelmointikortilta. Tietokoneella laadittu ohjelma ladataan mikrokontrolleriin USB-liittimen (Universal Serial Bus) kautta. Kortti sisältää myös erilaisia tapoja antaa kellopulssia ohjelmoitavalle

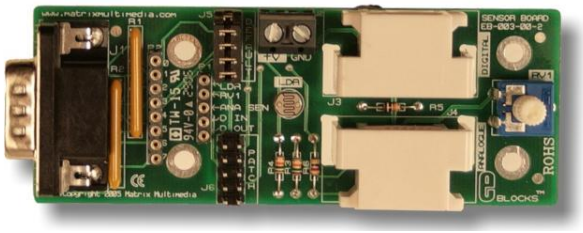
mikrokontrollerille. Oheiskorteista kytkin- ja ledkortit toimivat hyvin digitaalisten tulojen ja lähtöjen harjoituksissa (kuva 2).



Kuva 2. Ylhäältä ledkortti ja kytkinkortti.

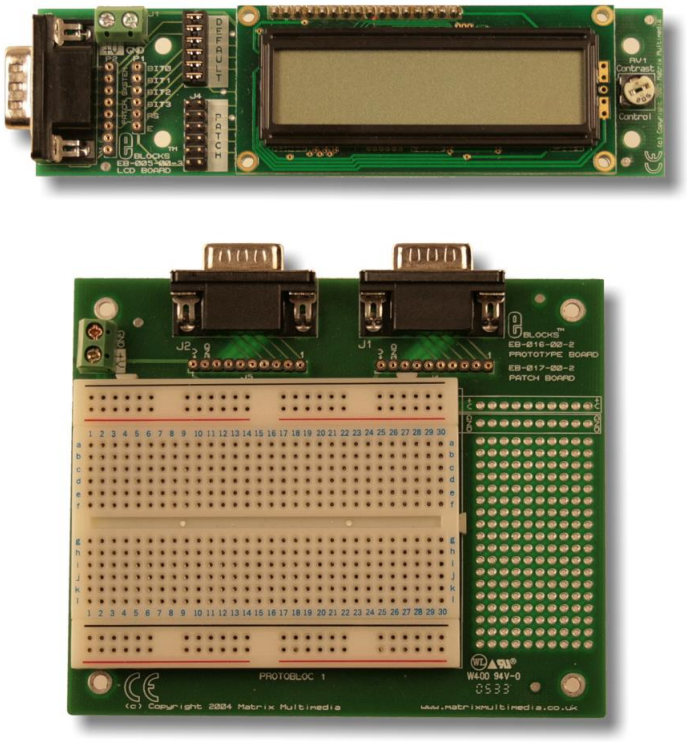
Ledkortilla on kahdeksan ledvaloa, joilla voidaan simuloida esimerkiksi moottorin käyttöä. Kytkinkortilla on kahdeksan painonappia, joita voidaan käyttää harjoituksissa kuvaamaan vaikka sylinterin rajakytkimiä.

Analogisignaalin lukemista ja käsittelyä voidaan harjoitella anturikortilla, joko valovastuksella tai potentiometrillä (kuva 3). Potentiometriä voidaan käyttää esimerkiksi lämpövastuksen simulointiin.



Kuva 3. Anturikortti.

Näyttökortille tulostetaan tekstiä tai numeroita (kuva 4). Kytkentäkortille voi tehdä omia kytkentöjä, jolloin oppii myös konkreettisesti liittämään esimerkiksi ohjattavia ledejä mikrokontrolleriin.

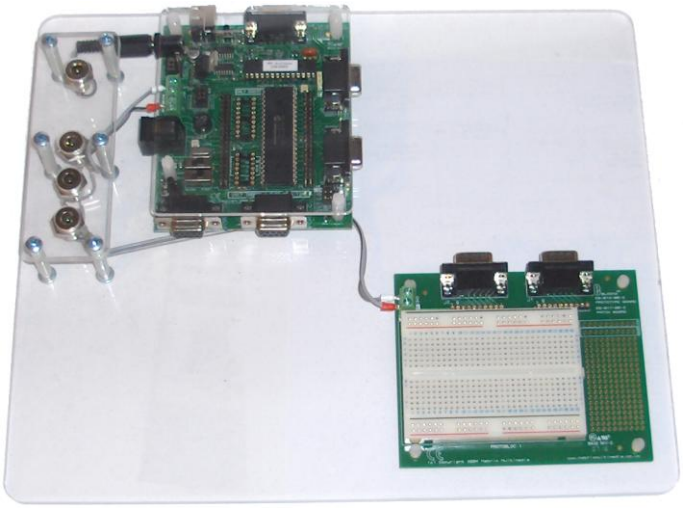


Kuva 4. Ylhäältä näyttökortti ja kytkentäkortti.

E-Block-korttien rikkoutumisen ehkäisemiseksi osaan korteista teetettiin läpinäkyvästä muovista suojalevy, jonka tarkoituksena on suojata herkät komponentit ja estää piirilevyn vääntyminen. Oikosulkujen välttämiseksi suojamuovit asennettiin nylon-ruuveilla (kuva 5).

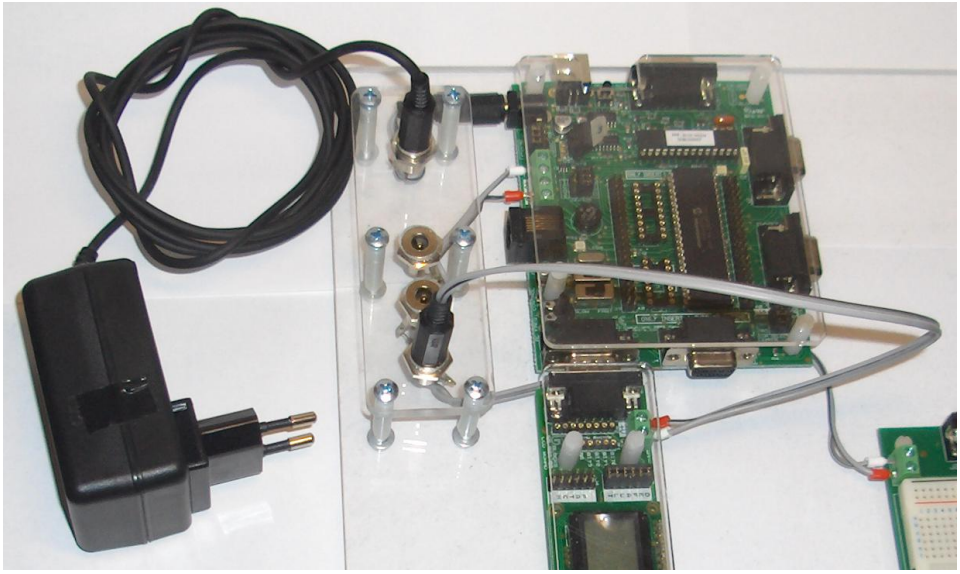
Kuva 5. Ledkortin suojaus.

Harjoittelualusta teetettiin samasta muovista. Alustaan asennettiin kiinteäksi ohjelmointikortti, kytkentäkortti sekä liitinpaneeli (kuva 6).



Kuva 6. Harjoittelualustan kiinteät osat.

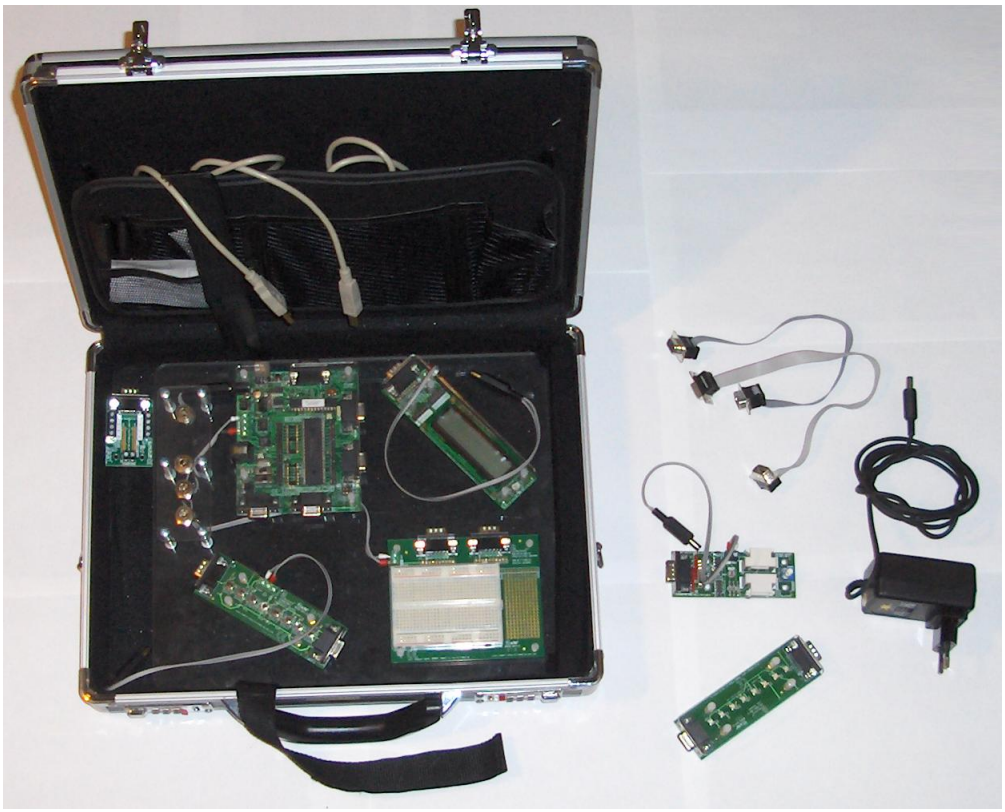
Ohjelmointikortin lisäksi osa muista korteista tarvitsee käyttöjännitteen (kuva 7). Harjoittelualustaan lisättyyn liitinpaneeliin kytketään ohjelmointikortin tarvitsema muuntaja sekä oheiskortteihin lisätyt virtajohdot.



Kuva 7. Liitinpaneliin kytketty muuntaja ja näyttökortti.

Normaalikäytössä 13,5 V muuntaja kytkettäisiin suoraan ohjelmointikorttiin. Muut kortit saisivat virtansa ohjelmointikortilla olevalta 5 V jännittelähdöltä. Kytkentä tehtäisiin sähköjohdoilla, jotka ruuvattaisiin korteilla oleviin ruuviliittimiin. Jatkuvassa käytössä ruuvien kannat voisivat kulua tai liitinten juotokset pettää. Johtojen ruuvailu ei ole käytön kannalta helppoa, ja vaarana on myös johdon kytkeminen väärään liittimeen. Liitinpaneelin ylin liitin on 13,5 V muuntajaa varten ja loput kolme liitintä ovat oheiskorteille. Liitinten koot ovat erilaiset, jotta verkkomuuntajaa ei voi kytkeä vahingossa väärään liittimeen ja samalla aiheuttaa korttien rikkoutumisen.

Kortit on jaettu harjoittelusarjoiksi. Esiteltyjen korttien lisäksi jokaiseen sarjaan kuuluu verkkomuuntaja, ohjelmointikaapeli sekä kaksi D-liittimillä varustettua kytkentäjohtoa. Harjoittelusarja säilytetään alumiinisalkussa (kuva 8).

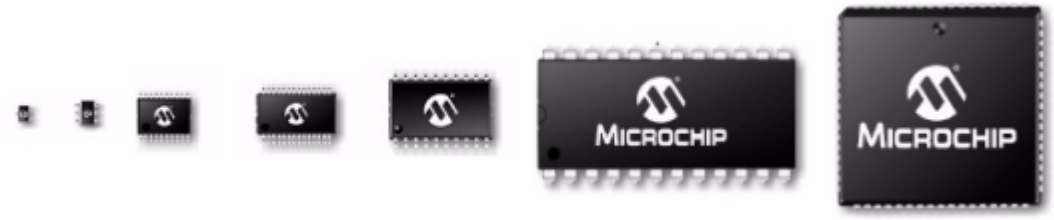


Kuva 8. E-block-harjoittelusarja.

4 MIKROKONTROLLERIT

Mikrokontrollerit ovat pieniä elektronisia piirejä, joiden toiminta määritellään tapauskohtaisella ohjelmalla. Mikrokontrollerit sisältävät mm. ohjelmamuistia, käyttömuistia ja suorittimen, joten ne ovat kuin pieniä tietokoneita. Niitä käytetään aivoina pienissä automaattisissa sovelluksissa.

Mikrokontrollerien valmistajia on monia, mutta suosittuja mikrokontrollereja ovat Atmelin valmistama AVR-tuoteperhe ja Microchipin valmistama PIC-tuoteperhe. Mikrokontrollereita on monia erikokoisia ja toiminnoiltaan hieman toisistaan poikkeavia malleja (kuva 9). Sovelluksesta riippuen valinta voidaan tehdä 6-nastaisen, tulitikun pään kokoisen pintaliitoskomponentin ja tulitikkuaskin kokoisen, 100-nastaisen mikrokontrollerin välillä.

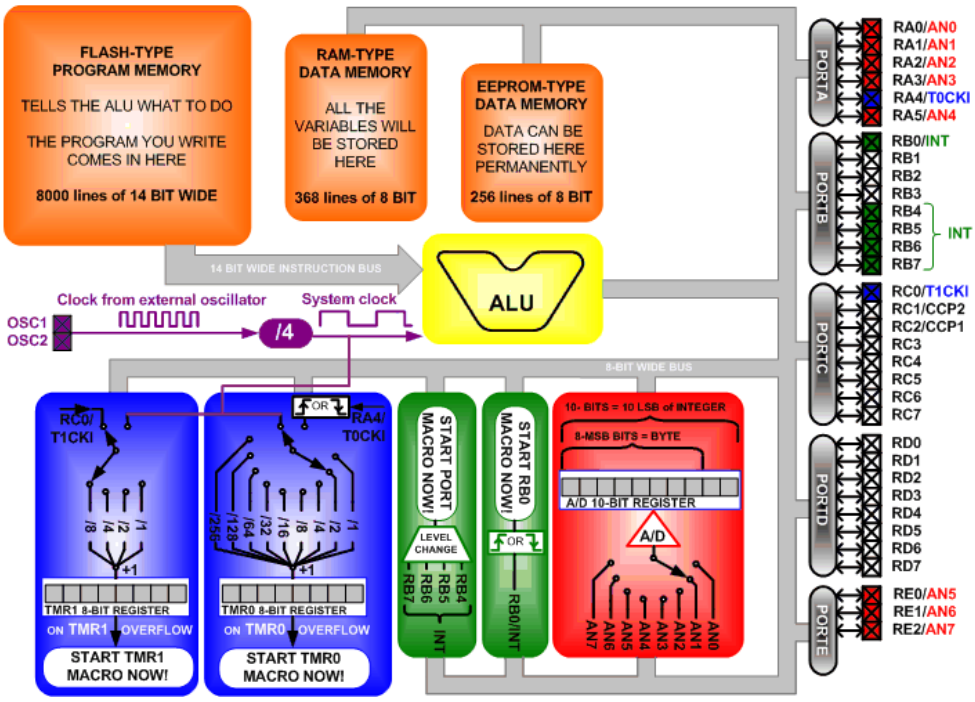


Kuva 9. Todellisessa koossa olevia mikrokontrollereita. /1/

Mikrokontrollereita käytetään hyvin yleisesti kodinelektroniikassa ja autoissa. Esimerkiksi television kaukosäätimessä mikrokontrolleri kerää käyttäjän napinpainallukset ja lähettää asianmukaisen signaalin infrapunaledillä itse televisiolle, jossa toinen kontrolleri purkaa saadun signaalin ja ohjaa halutun toiminnan. Toinen esimerkki voisi olla tavallisen henkilöauton moottorinohjaus: mikrokontrolleri seuraa moottorin kierroslukua, kaasupolkimen asentoa ja mahdollisia muita antureita. Näiden tietojen mukaan säädetään oikean polttoainemäärän suihkutuksen sekä sytytyksen ajoitusta.

4.1 Sisäinen rakenne ja toiminta

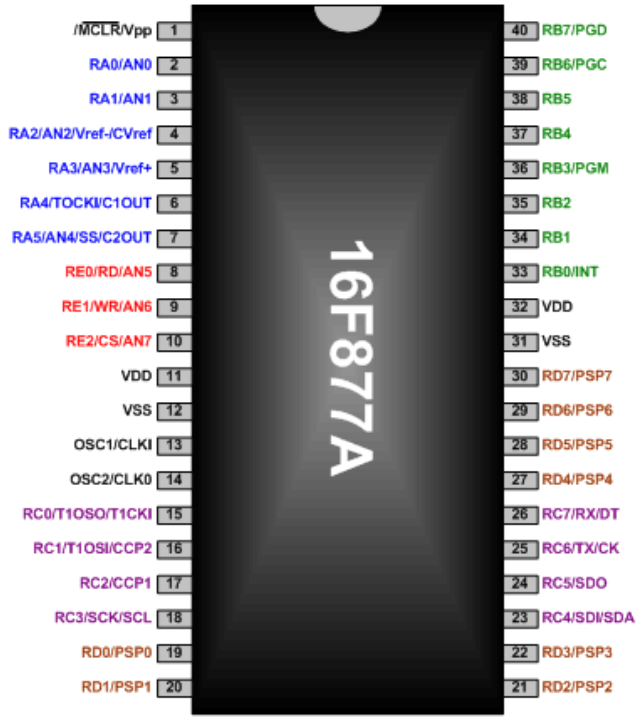
Mikrokontrollerille ladattu ohjelma ja käskyt tallentuvat ohjelmamuistiin (kuva 10). Mikrokontrollerin toimintoja ohjaa ALU (Arithmetic Logic Unit, aritmeettis-looginen yksikkö). Kaikki kontrollerin sisällä kulkeva data kulkee tämän yksikön kautta. ALU suorittaa ohjelmamuistissa olevaa ohjelmaa, jonka mukaan se kerää dataa, käsittelee ja suorittaa mahdolliset laskut ja lähettää tiedot muille lohkoille ja porteille. ALU tallentaa käyttömuistiin (RAM, random access memory) toiminnan aikana tarvittavia väliaikaisia tietoja ja muuttujia. EEPROM-muistiin (electronically erasable programmable read-only memory) tallennetaan pysyvää tietoa.



Kuva 10. PIC16F877A lohkokaavio /2/.

Portit ja niiden nastat ovat ulkoisia liitäntöjä. PIC16F877A sisältää näitä portteja viisi. Nastojen tai bittien porttikohtainen määrä vaihtelee, joka rajoittaa hieman ulkoista kytkemistä. Esimerkiksi näyttöä, joka tarvitsee kuusi nastaa mikrokontrollerilta, ei voida kytkeä E porttiin. Jokaisen portin nasta tai bitti voidaan

määritellä toimimaan digitaalisena sisääntulona tai uloslähtönä. Joillakin nastoilla on lisäksi muita lisätoimintoja (kuva 11).



Kuva 11. PIC:n nastojen ja porttien toimintoja /2/.

Porttien A ja E nastoja voidaan käyttää analogisena sisääntulona, esimerkiksi lämpötila-anturille. Analoginen signaali saa A/D-muuntimessa (analogi/digitaali-muunnin) 10-bittisen arvon. Osa B-portin nastoista voidaan käyttää keskeytyksiin (interrupt). Ulkoinen muutos keskeytys-nastassa, esimerkiksi napin painallus, keskeyttää pääohjelman suorituksen ja käynnistää keskeytysohjelman. Keskeytysohjelman jälkeen pääohjelman suoritus jatkuu siitä, mihin se jäi. Ajastettujen ohjelmien suoritus hoituu ajastinlohkoilla. Ajastuksessa tarvittavat kellopulssit otetaan joko järjestelmäkellolta tai omalta kellolta, joka kytketään joko A-portin viidenteen tai C-portin ensimmäiseen nastaan. Kun kellopulssit saavuttavat määritellyn arvon, pääohjelma keskeytyy ja ajastusohjelman suoritus alkaa.

Mikrokontrollereilla voi olla myös muita mahdollisia ominaisuuksia, kuten pulssinleveysmodulaatio (Pulse Widht Modulation, PWM), komparaattori eli vertailija-toiminto ja erilaisia sarjaliikenneprotokollia. Tarkemmat tiedot kaikista ominaisuuksista löytää mikrokontrollerikohtaiselta datalehdeltä, jonka voi ladata valmistajan internetsivuilta.

4.2 Ulkoiset kytkennät

Mikrokontrolleria kytkettäessä tulee ottaa huomioon kontrollerikohtaiset sähköiset maksimiarvot, jotka on kerrottu ohjaimen datalehdellä. Taulukkoon 1 on listattu PIC16F877A absoluuttisia maksimeja.

Taulukko 1. PIC16F877A sähköisiä maksimiarvoja.

Jännite VDD-nastassa verrattuna VSS-nastaan	-0,3 ... +7,5V
Jännite MCLR-nastassa verrattuna VSS-nastaan	0 ... +14V
Jännite RA4-nastassa verrattuna VSS-nastaan	0 ... +8,5V
Jännite muissa nastoissa verrattuna VSS-nastaan	-0,3V ... (VDD + 0,3V)
Maksimivirta VSS-nastasta	300 mA
Maksimivirta VDD-nastaan	250 mA
Yksittäisen I/O-nastan maksimivirta	25 mA
Maksimivirta yhteensä porteissa A, B ja E	200 mA
Maksimivirta yhteensä porteissa C ja D	200 mA

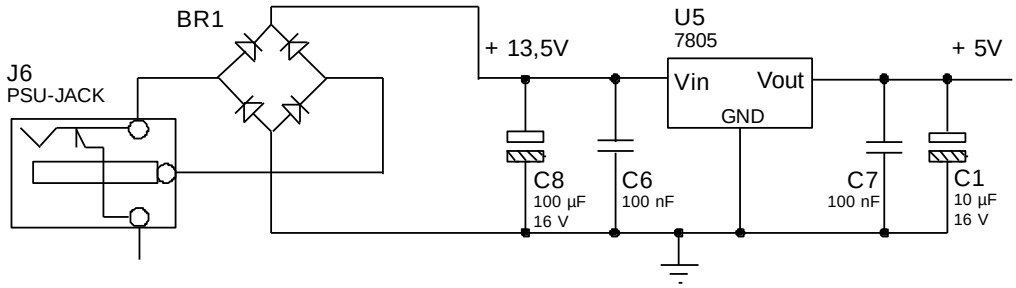
Taulukosta nähdään, että vaikka yksittäisen portin nasta pystyy käsittelemään 25 mA virran, niin maksimivirta on rajoitettu taulukossa kuvatulla tavalla.

4.2.1 Käyttöjännite

Mikrokontrollerin tarvitsema käyttöjännite on tyypillisesti 5 V tasajännite, mutta esimerkiksi PIC16F877A toimii myös 2 V ja 5,5 V välillä riippuen kellotaajuudesta. Käytössä olevan mikrokontrollerin arvot kannattaa tarkistaa valmistajan datalehdeltä.

Verkkolähteiden ulostuloissa tapahtuu jännitteen huojumista. Pattereissa taas jännite laskee siihen kytketyn kuorman mukaan. Mikrokontrollerin tulisi saada kuitenkin tasaista jännitettä.

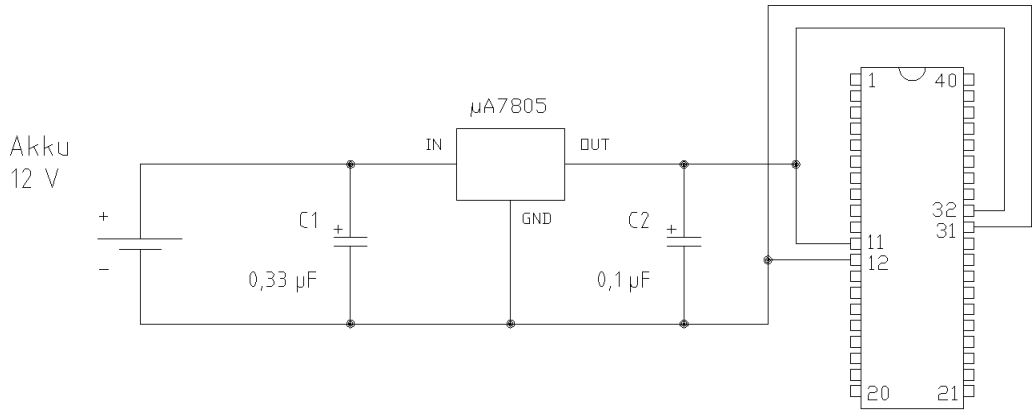
E-Block-ohjelmointikortilla muuntajakäytössä tasainen 5V jännite on tuotettu 7805-regulaattoripiirillä (kuva 12). Kuvassa muuntajan liittimen jälkeen tuleva tasasuuntaussilta huolehtii jännitteen napaisuudesta, muuntajan ulostulon napaisuudella ei ole siis väliä.



Kuva 12. Ohjelmoitavan mikrokontrollerin käyttöjännite ohjelmointikortilla.

Regulaattori 7805 hukkaa tulojännitteestä kaiken 5 V ylittävän jännitteen. Tulojännitteen tulee olla kuitenkin reilusti haluttua lähtöjännitettä suurempi, jotta regulaattorin sisällä tapahtuvat välttämättömät jännitehäviöt tulee ylitettyä. Valmistajat ilmoittaa komponenteilleen raja-arvot, esimerkiksi Texas Instrumentsin µA7805-piirin tulojännitteen tulee olla 7 V ja 25 V välissä, jotta lähtöön saadaan

5 V tasainen jännite. Texas Instrumentsin µA7805-piirillä ja muutamalla kondensaattorilla saadaan hoideltua esimerkiksi autoon sijoitettavan sovelluksen käyttöjännite (kuva 13).



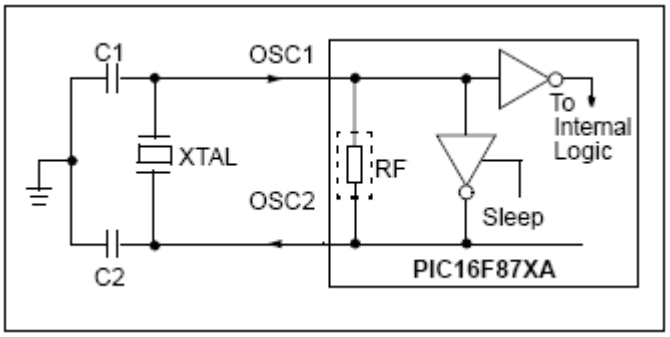
Kuva 13. Kontrollerin jännitelähde autokäyttöön.

4.2.2 Kellopulssi

Käyttöjännitteen lisäksi mikrokontrollerit tarvitsevat myös kellopulssin, kontrolleri suorittaa sisäisiä toimintoja ja ohjelmoitua ohjelmaa tämän pulssin tahtiin. Yhden käskyn suorittamiseen tarvitaan neljä kellopulssia, esimerkiksi 10 MHz kellotaajuudella toimiva kontrolleri suorittaa 2,5 miljoonaa käskyä sekunnissa (MIPS, million instructions per second). Värähtelytaajuudella eli kellotaajuudella vaikutetaan siis suoraan kontrollerin nopeuteen, samalla kuitenkin lisätään myös mikrokontrollerin virrankulutusta.

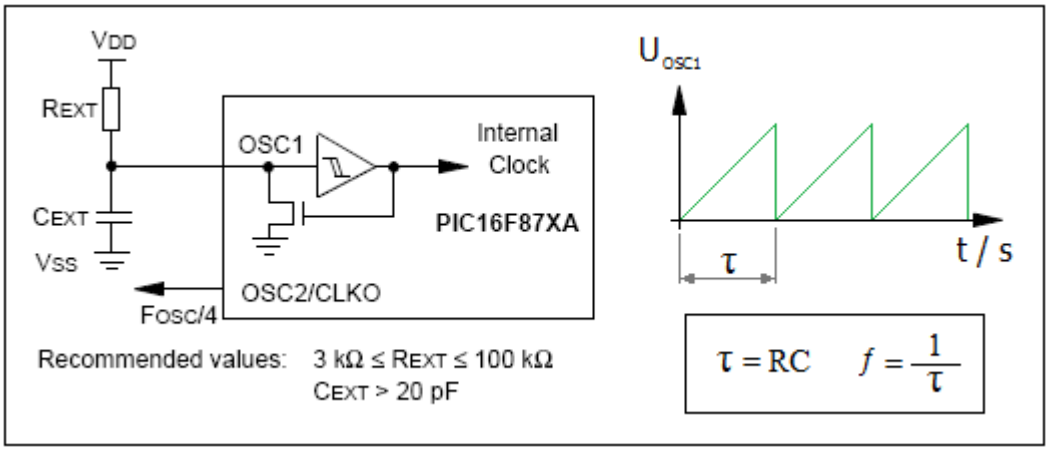
Mikrokontrolleri voi saada kellopulssinsa toiselta laitteelta, jolloin myös laitteiden välinen tiedonsiirtokin tahdistuu. Joissakin kontrollereissa on sisäisiä oskillaattoreita eli jaksollisesti värähteleviä piirejä. PIC16F877A:ssa niitä ei ole, laite tarvitsee siis ulkoisen pulssin tai oskillaattoripiirin.

Oskillaattoripiiri voidaan tehdä kide- (crystal) tai keraamisella resonaattorilla sekä kahdella kondensaattorilla (kuva 14). Kide värähtelee omalla toimintataajuudellaan, kun se liitetään kontrollerin OSC-nastoihin sekä kondensaattorien kautta maatasoon. Komponentin valmistaja ohjeistaa oikean kokoisten kondensaattoreiden valinnassa. Valmistajat tarjoavat myös valmiita resonaattoripiirejä, joihin kondensaattorit ovat sisällytetty.



Kuva 14. Kiteen ja kondensaattorien kytkentä. /3/

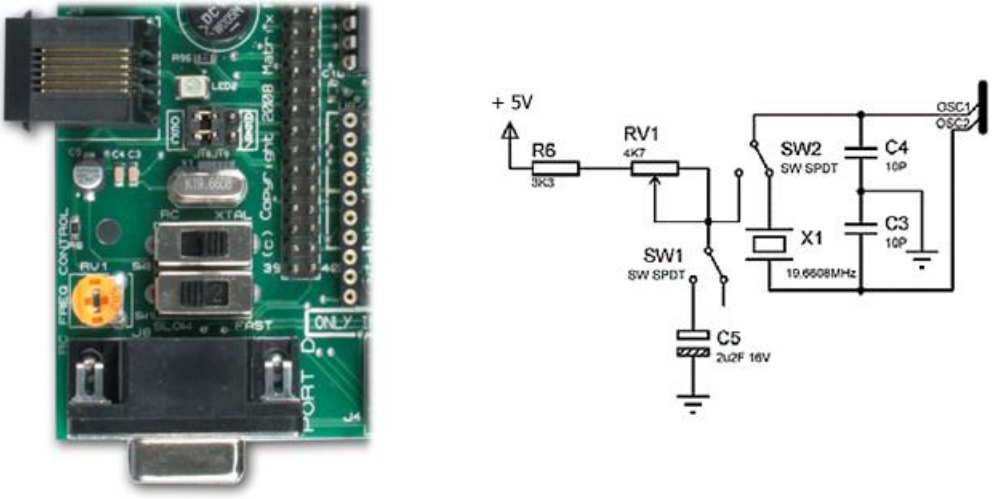
RC-kytkentää (resistor, capacitor) voidaan käyttää sovelluksissa, joissa pulsseilta ei vaadita ajallisesti tarkkaa toimintaa. Kondensaattoria varataan vastuksen kautta, jolloin jännite OSC1-nastassa alkaa nousta (kuva 15). Kondensaattori puretaan, kun jännitte on noussut riittävästi. Kytkentä tuottaa sahalaita-aaltoa OSC1-nastaan.



Kuva 15. RC-kytkennän käyttö oskillaattoripiirinä. /3/

RC-kytkennän taajuus lasketaan aikavakion (τ, tau) avulla. Tulos on kuitenkin vain suurpiirteinen, koska taajuuteen vaikuttavat komponenttien rakenne, laatu sekä käyttöympäristö. Komponenteilla on toleranssi, jonka sisällä niiden arvo vaihtelee. Lisäksi liitokset aiheuttavat oman kapasitanssinsa, joka pientä kondensaattoria käytettäessä tulee ottaa huomioon.

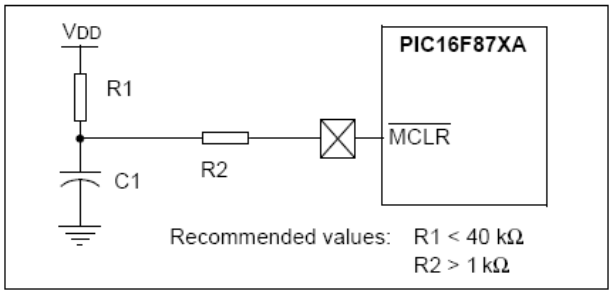
E-block-ohjelmointikortilla on kideoskillaattoripiiri sekä muunneltava RC-piiri. Käytettävä piiri valitaan SW2-kytkimellä (kuva 16). Käyttämällä kidettä kontrollerin kellotaajuus on 19,6608 MHz. Jos taas käytetään RC-piiriä, niin SW1-kytkimellä valitaan kapasitanssiarvo, ja potentiometrillä muutetaan vastuksen arvoa. Vastuksen arvo vaihtelee 3,3 kΩ ja 8 kΩ välillä. Kapasitanssiarvo on joko 2,2 µF tai 10 pF. RC-piiriä käyttämällä voidaan taajuutta vaihdella 12,5...30 MHz tai 57...138 Hz alueilla.



Kuva 16. Kellopulssin valinta- ja säätökomponentit ohjelmointikortilla. /4/

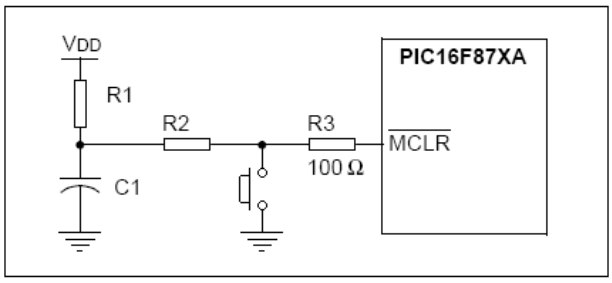
4.2.3 Nollauspiiri

Mikrokontrollerin ensimmäinen nasta on nollaus- eli reset-nasta (MCLR). Reset-nastan jännitteen ollessa 0 V on kontrolleri reset-tilassa, jolloin asetukset ja ohjelmakierto palaavat alkutilaan. Ohjelman suoritus alkaa vasta kun nastan jännite nousee ylös. Normaalin toiminnan aikana nastan tilan tulee olla käyttöjännitteen tasolla. Yksinkertaisin vaihtoehto on kytkeä nasta ylösvetovastuksen avulla suoraan käyttöjännitteeseen. Datalehdeltä kannattaa kuitenkin tarkistaa suositeltu reset-kytkentä. PIC16F877A tulisi kytkeä kahdella vastuksella ja kondensaattorilla, joilla varmistetaan reset-nastan toiminta (kuva 17).



Kuva 17. Reset-piirin kytkentä ilman painonappia. /3/

Käytön aikana voi sovellukseen tulla toimintahäiriöitä, jolloin järjestelmä halutaan käynnistää uudestaan. Reset-piiriin voidaan lisätä painonappi, jolla MCLR-nasta kytketään tarvittaessa maahan. Kytkennässä tulisi käyttää pientä vastusta, joka suojaa ohjainta virtapiikeiltä (kuva 18).

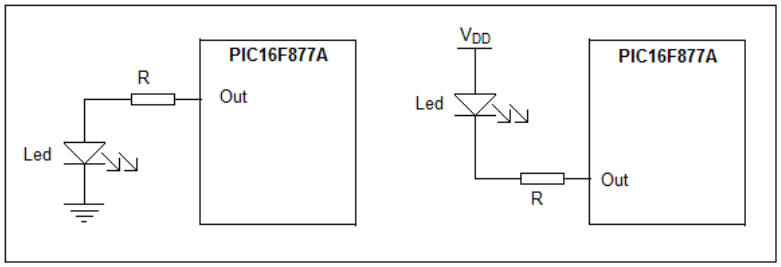


Kuva 18. Reset-napin kytkeminen mikrokontrolleriin.

4.2.4 Ohjattavat komponentit

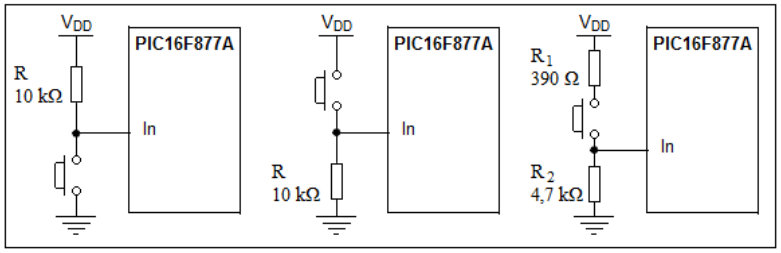
Mikrokontrollerin loput nastat ovat ohjelmoitavia nastoja. Ulkoisten laitteiden liittämisessä tulee huomioida maksimiarvot. Ledit, kytkimet ja muut pientä virtaa käyttävät piirit voidaan kytkeä suoraan mikrokontrolleriin.

Nastan toimiessa lähtönä se ohjaa ulkoista kuormaa. Nastan tilan mukaan mikrokontrolleri joko ”syöttää” (source) tai ”nielee” (sink) virtaa. Esimerkiksi led-valo voidaan kytkeä toimimaan kahteen eri suuntaan (kuva 19). Virta rajoitetaan etuvastuksella sallitun rajan alle.



Kuva 19. Ledin kaksi eri kytkentäsuuntaa.

Kytkimet, painonapit ja raja-anturit voidaan kytkeä usealla eri tavalla. Nasta on tulona ja tarkkailee jännitettä. Jännite on joko 0 V tai 5 V (kuva 20).

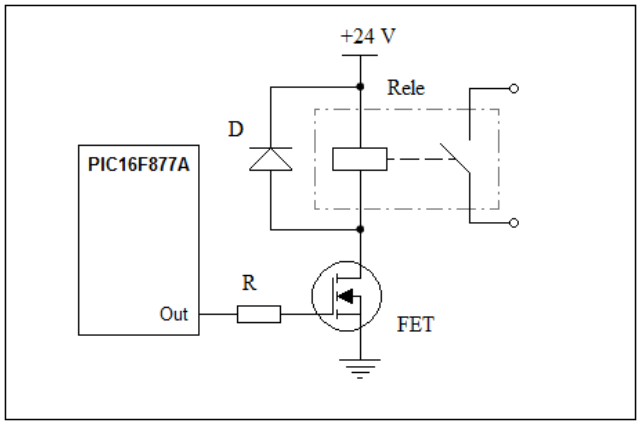


Kuva 20. Painonappien kytkentätapoja.

Kytkennöissä voidaan ottaa huomioon myös mahdolliset ohjelmointivirheet, jotka voivat pahimmassa tapauksessa rikkoa koko mikrokontrollerin. Esimerkiksi

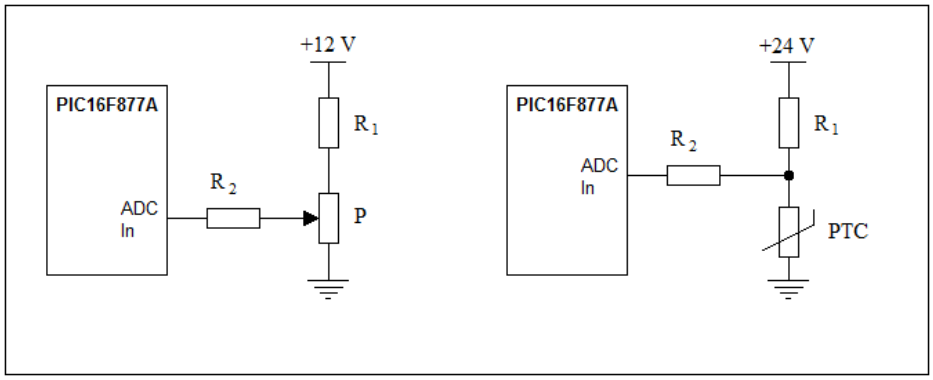
painonapin kytkeminen ja ohjelmointi lähdöksi voi aiheuttaa nappia painettaessa oikosulun. Tällaista oikosulkuvirtaa varten piiriin voidaan lisätä pieni vastus. E-block-kytkinkortin kytkentä on kuvan 20 viimeisen piirin mukainen.

Suurempaa virtaa tai jännitettä tarvitsevat piirit kytketään apupiireillä mikrokontrolleriin. Apupiireissä voidaan käyttään esimerkiksi transistoreja, optoerottimia ja releitä (kuva 21).



Kuva 21. Suuret virrat ja jännitteet ohjataan apupiireillä.

Analogisen signaalin kytkemisessä tarvitaan jännitteen sovitus (kuva 22). Sovitus kannattaa tehdä mieluummin kapeammaksi kuin sallitut ääripäät, näin pysytään varmasti esimerkiksi mittausalueella.



Kuva 22. Analogisignaali tarvitsee sovittaa 0 V ja 5 V väliin.

5 FLOWCODE JA OHJELMOINTI

Toimiakseen oikein ja halutulla tavalla mikrokontrolleri tarvitsee ohjeet eli ohjelmakoodin. Mikrokontrolleri suorittaa tätä ohjelmaa annetussa järjestyksessä. Ohjelman laatiminen tietokoneella ja sen lataaminen mikrokontrolleriin on ohjelmointia.

Mikrokontrollerin muistissa oleva ohjelma on sarjoina bittejä, binäärinumeroita eli ykkösiä ja nollia. Neljän bitin sarja voidaan ilmoittaa myös heksadesimaalilukuna, eli lukuna jolla voi olla 16 eri arvoa. Binäärinumeroina olevaa ohjelmaa sanotaan konekieleksi, ja tätä kieltä kontrolleri ymmärtää. Konekieli on ensimmäisen sukupolven ohjelmointikieli. Ohjelmointikortilla oleva PPP-piiri välittää konekielellä olevan ohjelman mikrokontrollerille. Ohjelman kirjoittaminen tällaisilla kielillä vaatisi melkoisen syvällistä osaamista ja yksittäisen laitemallin rakenteen tuntemusta. Ohjelmointivirheen etsiminen suuren ohjelman ja mahdollisesti kymmenientuhansien merkkien seasta voisi olla haastavaa. Tarvitaan jotain helpompaa.

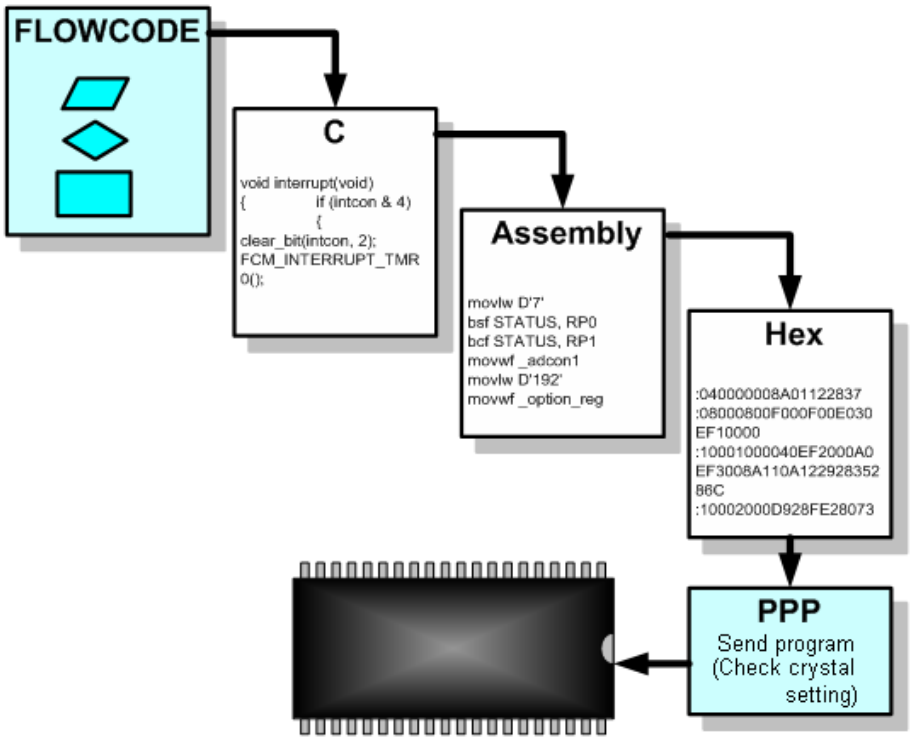
Assembly, eli symbolinen konekieli, on toisen sukupolven ohjelmointikieli. Se antaa usein toistuville heksadesimaalikäskyille helpommin ymmärrettävän nimen. Assembleri muuttaa assembly-kielellä kirjoitetun ohjelman konekielelle. Konekielestä ja assemblystä käytetään myös yhteistä nimitystä alemman tason ohjelmointikieli. Ohjelman kirjoittajan tarvitsee kuitenkin tuntea vielä paljon laitekohtaisia yksityiskohtia. /5/

Ylemmän tason C-ohjelmointikieli helpottaa ohjelman laatimista entisestään. C-kielen etuihin kuuluu sen yleiskäyttöisyys: haluttu yhteenlasku kirjoitetaan samalla tavalla - meni se sitten tavalliseen kotitietokoneeseen tai mikrokontrollerilla varustettuun pesukoneeseen. C-kieltäkin käyttämällä tarvitsee määritellä ohjelman

kohdelaite, mikä tapahtuu lisäämällä kirjoitettavaan ohjelmaan valmiiden laitekohtaisten kirjastojen nimet. Nämä kirjastot sisältävät tarvittavat tiedot laitteen arkkitehtuurista, rekistereistä ja käskykannasta. Valmis ohjelma muutetaan assemblyksi kääntäjällä, joka lisää kirjastot ohjelmaan ja huolehtii kaikista edellä mainituista yksityiskohdista.

Flowcode on korkean tason ohjelmointikieli. Kirjoittamisen sijaan ohjelma rakennetaan palikoista vuokaavioksi. Palikoilla on omat yksinkertaiset toimintonsa, joita säädetään muutamalla asetuksella. Flowcode on kuin kuvitettu C-kieli, jossa pystytään jopa simuloimaan ohjelman toimintaa lataamatta sitä mikrokontrolleriin.

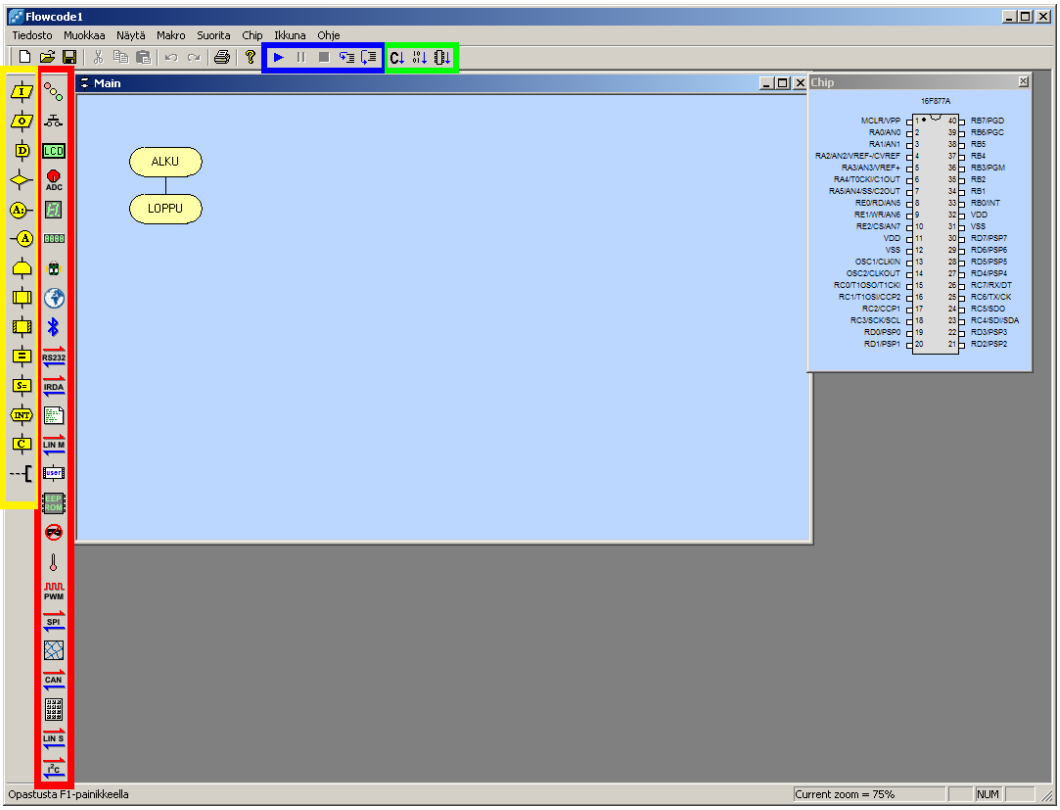
Mikrokontrolleria ohjelmoitaessa valmis Flowcode-ohjelma käännetään ensiksi C-kielelle, sitten assemblyksi ja edelleen konekieleksi, jonka mukaan PPP-piiri ohjelmoi lopulta itse mikrokontrollerin (kuva 23).



KUVA 23. Mikrokontrollerin ohjelmointi tapahtuu portaissa. /1/

5.1 Flowcode käyttöliittymä

Ohjelma laaditaan siis Flowcode-nimisellä ohjelmalla (kuva 24). Perusnäkymän keskellä on Main-ikkuna, jonka vuokaavioon laaditaan tuleva pääohjelma. Ohjelmassa pääohjelman lisäksi on usein aliohjelmia, joiden ikkunat sisältävät omat vuokaavionsa. Oikealla näkyvästä Chip-ikkunasta pystyy simuloitaessa seuraamaan nastojen tilaa, ja se toimii myös hyvänä muistilistana nastojen erityistoiminnoille.



Kuva 24. Flowcoden perusnäkymä. Työkalupalkit on kuvaan laatikoitu väreillä.

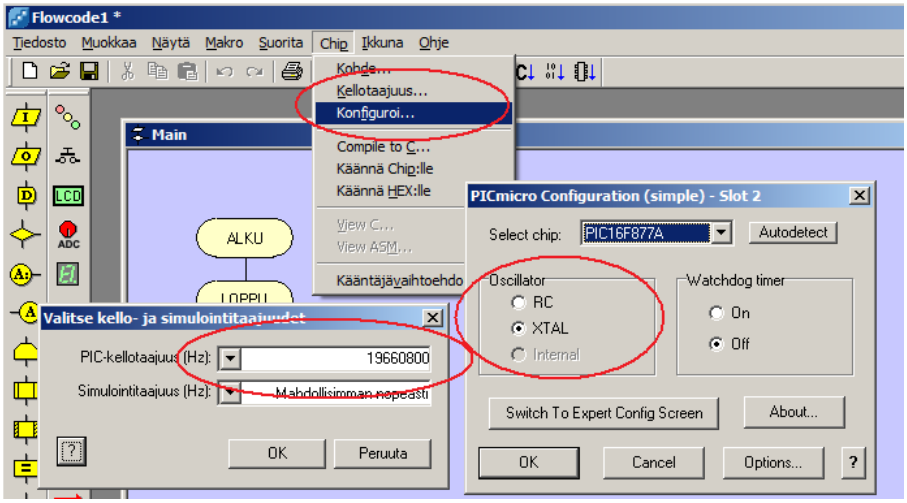
Kuvaan vasemmalle keltaisella laatikoituna on käskypalkki. Ohjelma rakennetaan raahaamalla näitä käskyjä vuokaavioon. Ohjelma suorittaa käskyt yksi kerrallaan aloittaen vuokaavion ylimmäisestä käskystä. Käskyjen vieressä punaisella laatikoituna on komponenttipalkki. Komponentit ovat kontrollerin sisäisten toimintojen ja ulkoisesti liitettyjen laitteiden mallinnuksia. Haluttu komponentti

lisätään klikkaamalla kuvaketta, jolloin ikkuna-alueelle ilmaantuu komponentin oma ikkuna. Ylhäällä työkalupalkissa ovat sinisellä laatikoidut simulointikuvakkeet, joilla voi suorittaa ohjelmaa virtuaalisesti tietokoneella ilman mitään kytkettyjä laitteita. Vihreällä laatikoiduilla kääntäjäkuvakkeilla suoritetaan ohjelman kääntäminen sekä ohjelmoidaan ohjelmointikortilla oleva mikrokontrolleri.

5.1.1 Asetukset ja ohjelman lataus

Aloitettaessa laatia uutta ohjelmaa Flowcode haluaa tietää, mihin mikrokontrolleriin tuleva ohjelma lopulta ladataan. Mikrokontrollerin mallin voi tarkastaa ohjelmointikortilla olevasta piiristä. Piirin voi myöhemmin vaihtaa Chip-valikosta.

Oikean piirin lisäksi tulee valita oikea oskillaattorin tyyppi ja sen kellotaajuus (kuva 25). E-Block-ohjelmointikortilla olevaa kideoskillaattoria käytettäessä oskillaattorin tyypiksi valitaan XTAL ja kellotaajuudeksi annetaan sen kellotaajuus, joka on 19,6608 MHz. Ohjelman latauksessa esiintyvät ongelmat johtuvat usein väärästä oskillaattorityypistä.



Kuva 25. Asetusten valinta.

Kun asetukset ovat kunnossa ja ohjelmointikortin virtalähde ja USB-kaapeli on kytketty, voidaan mikrokontrolleri ohjelmoida joko Chip-valikon ”Käännä Chip:lle” -kohdasta tai kääntäjäpalkin viimeisestä napista.

Ohjelman voi halutessaan kääntää C-kielelle joko kääntäjäkuvakkeiden ensimmäisestä napista tai Chip-valikosta ”Compile to C” -kohdasta. Konekielelle kääntäminen käy joko keskimmäisestä napista tai valikon ”Käännä HEX:lle” kohdasta. Konekielelle kääntäessä ohjelma luo myös assemblytiedoston. Käännettyjä koodeja pääsee katselemaan Chip-valikon kohdista ”View C” ja ”View ASM”. Konekielisen version ohjelmasta löytää tallennuskansiosta tallennetulla nimellä .hex-päätteisenä.

5.1.2 Simulointi

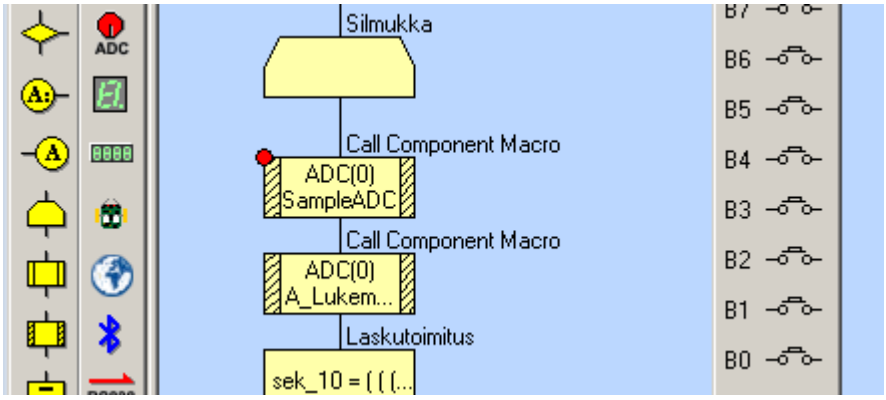
Ohjelmaa voidaan testata suoraan Flowcodessa. Rajoituksena ovat ohjelmat, joihin on lisätty C-koodia. Lisätyn C-koodin toiminta pitää ohjelmoida simulointia varten Flowcoden omilla käskyillä, jotta simulointi onnistuisi. Myöskään viiveet ja keskeytykset eivät toimi keskenään oikein. Simuloinnissa viiveen suorituksen aikana ei tapahdu keskeytyksiä, vaikka todellisuudessa keskeytys-käsky kumoaa kaiken muun ja suoritetaan heti, oli viive päällä tai ei. Viiveen aikana tapahtuva keskeytys on testattava oikeassa mikrokontrollerissa.

Simulointi käynnistetään simulointikuvakkeista, joille löytyvät myös vastineet Suorita-valikosta. Simuloinnin nopeutta pystyy muuttamaan Chip-valikon Kellotaajuus-kohdasta. Ensimmäinen nappi käynnistää ja jatkaa normaalia simulointia, toisesta napista simuloinnin saa keskeytettyä ja kolmas nappi lopettaa koko simuloinnin. Askellusnapeilla ohjelmaa pystytään simuloimaan käsky

kerrallaan. Neljäs nappi askeltaa kaikki käskyt, myös aliohjelmissa olevat käskyt. Viides nappi askeltaa pelkästään pääohjelman käskyt. Jos eteen tulee aliohjelma, se suorittaa sen sisältämät käskyt yhdellä kertaa.

Simuloinnin aikana ruudulla olevista ikkunoista voi seurata mikrokontrollerissa ja ohjelmassa tapahtuvia muutoksia. Poikkeuksena on simulointinopeuden vaihtoehto ”Mahdollisimman nopeasti”. Muuttujat-ikkunasta voi nähdä käytetyt muuttujat ja niiden sen hetkiset arvot. Kutsupino-ikkunassa näkyvät ajettavat aliohjelmat. Muuttujat-ikkunan jonkin muuttujan nimeä tuplaklikkaamalla voi muuttujan arvoa vaihtaa simuloinnin aikana.

Ohjelmaan voidaan lisätä myös pysäytyskohtia, joiden kohdalla simulointi keskeytyy (kuva 26). Esimerkiksi pitkän ohjelman pientä osaa voidaan tarkastella askeltamalla, mutta ohjelman muut osuudet ajetaan keskeytyksettä. Näitä simulointikesketyksiä lisätään ja poistetaan klikkaamalla hiiren oikealla napilla jotakin käskyä vuokaaviossa ja valitsemalla ”Vaihda keskeytyskohta”, jolloin simulointi pysähtyy valittua käskyä ennen. Valittuna olevan käskyn kohdalle voi keskeytyskohdan lisätä myös Muokkaa-valikosta. Samasta valikosta voi myös poistaa kaikki ohjelmassa olevat simulointikeskeytykset.

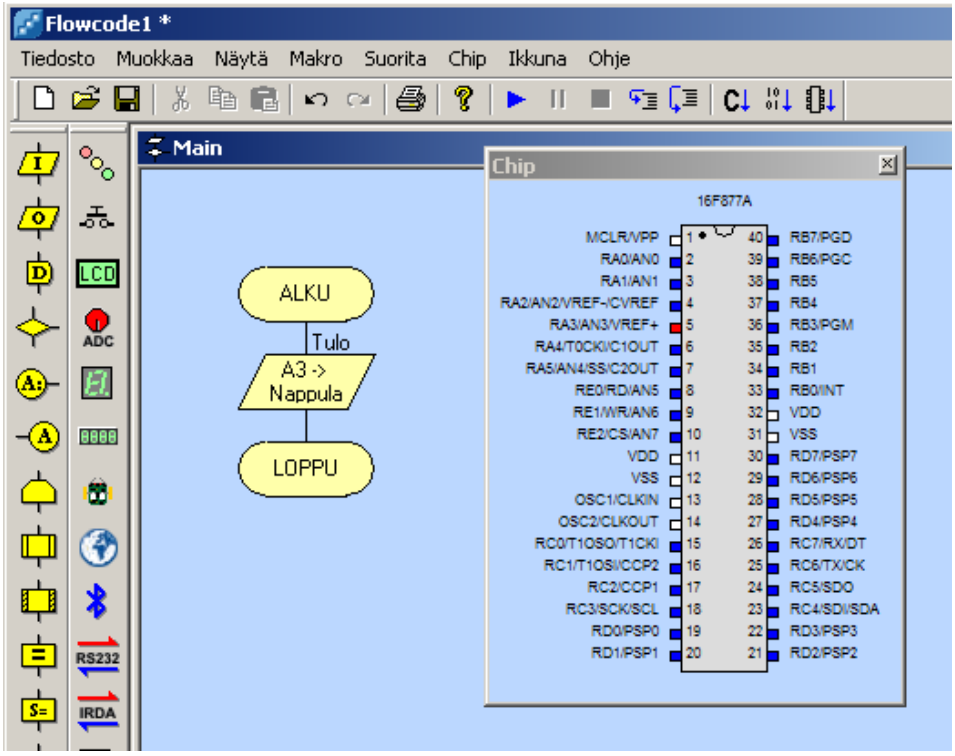


Kuva 26. Simulointi keskeytyy vuokaaviossa olevan punaisen merkin kohdalle.

Flowcoden suomennos voi antaa väärän käsityksen, mutta simulointikesketyksellä ei ole mitään tekemistä Keskeytys-käskyn kanssa - niiden toimintaa ei saa luulla samaksi.

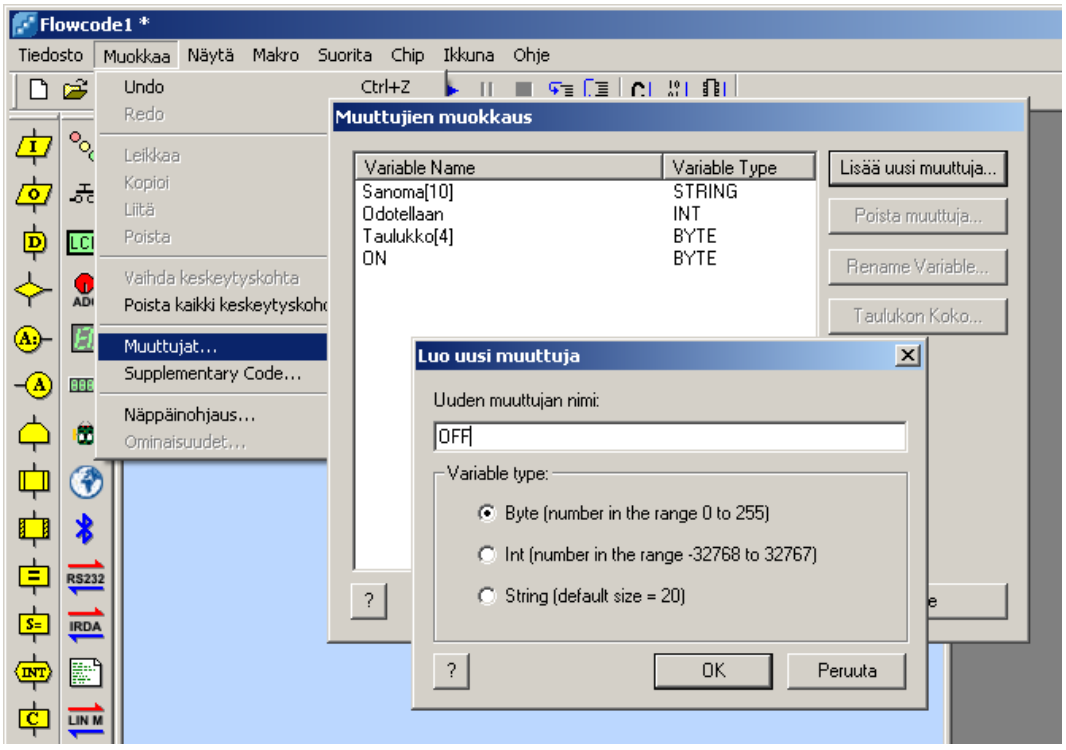
5.2 Muuttujat

Mikrokontrollerissa tietoihin ja arvoihin viitataan nimillä eli muuttujilla. Portin tai nastan tila tallennetaan muistiin johonkin nimettyyn muuttujaan. Lähdön tila voidaan määrätä muistissa olevan muuttujan arvon mukaan. Viiveen kesto voi määräytyä jonkin muuttujan arvon mukaan. Esimerkiksi kun mikrokontrollerin viidenteen nastaan kytketyn napin tila luetaan, voidaan se tallentaa vaikka Nappula-nimisen muuttujan arvoksi (kuva 27).



Kuva 27. Kytkimen tila luetaan Nappula-nimiseen muuttujaan.

Muuttujan nimessä saa käyttää englantilaisia aakkosia, alaviivaa ja numeroita. Nimessä täytyy olla vähintään yksi kirjain, jotta se eroaisi puhtaasta numerosta. Nimi saa olla enintään 32 merkin mittainen, eikä siinä saa käyttää välilyöntejä. Nimenä kannattaa tietenkin olla jokin sen toimintaa kuvaavaa sanayhdistelmä tai lyhenne. Kun Flowcode kääntää ohjelman C-kielelle, se muuttaa kaikkien muuttujien nimet isoiksi kirjaimiksi, jolloin vaarana on kahden eri muuttujan tahaton päällekkäisyys. Flowcodessa ”MUUTTUJA” ja ”muuttuja” on siis sama.



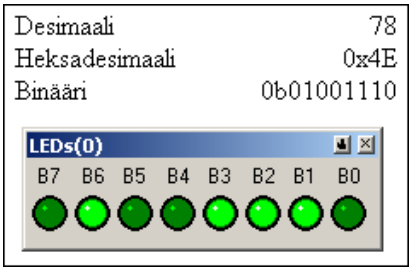
Kuva 28. Muuttujien lisäys muokkaus-ikkunasta.

Uutta muuttujaa luodessa määritellään sille nimen lisäksi tyyppi (kuva 28). Määrityksellä varataan muistista riittävä tila tyypin mukaan. Byte, eli tavu, on 8-bittinen luku, joka voi saada 256 eri arvoa. Int, eli sana, on 16-bittinen luku ja eri arvoja sillä voi olla 65536.

Tavusta ja sanasta voidaan tehdä taulukko lisäämällä muuttujan nimen perään hakasulkeissa taulukossa olevien arvojen määrä. Esimerkiksi int-tyypin lamppu[7]-muuttuja sisältää siis seitsemän sanan kokoista lukua, ja vastaavasti kuvassa oleva Taulukko[4]-muuttuja pitää sisällään neljä tavun kokoista lukua. Taulukoissa olevien lukujen osoitteet ovat saman muotoisia kuin muuttujan nimi, mutta numerolla määritellään tiedon paikka itse taulukossa, numeroinnin alkaessa nollasta. Jos siis halutaan käyttää kuvan taulukon ensimmäistä lukua, käytetään halutussa kohdassa Taulukko[0]-osoitetta. Vastaavasti viimeisen luvun osoite olisi Taulukko[3].

String, eli merkkijono, on taulukko, jossa arvo on ASCII-koodattu (American standard code for information interchange) merkki tai kirjain. Kuvan Sanoma[10]-muuttuja on kymmenen merkin taulukko tai lause.

Muuttujan arvoa muutetaan esimerkiksi tulo-, laskutoimitus- ja tekstinmuokkaus-käskyillä. Lukuarvo voidaan antaa desimaali-, heksadesimaali- tai binäärimuodossa. Lukumuodolla ei ohjelman kannalta ole väliä, vaan muoto kannattaa valita käyttötarkoituksen mukaan. Esimerkiksi portin tila kannattaa antaa suoraan binäärimuodossa (kuva 29).

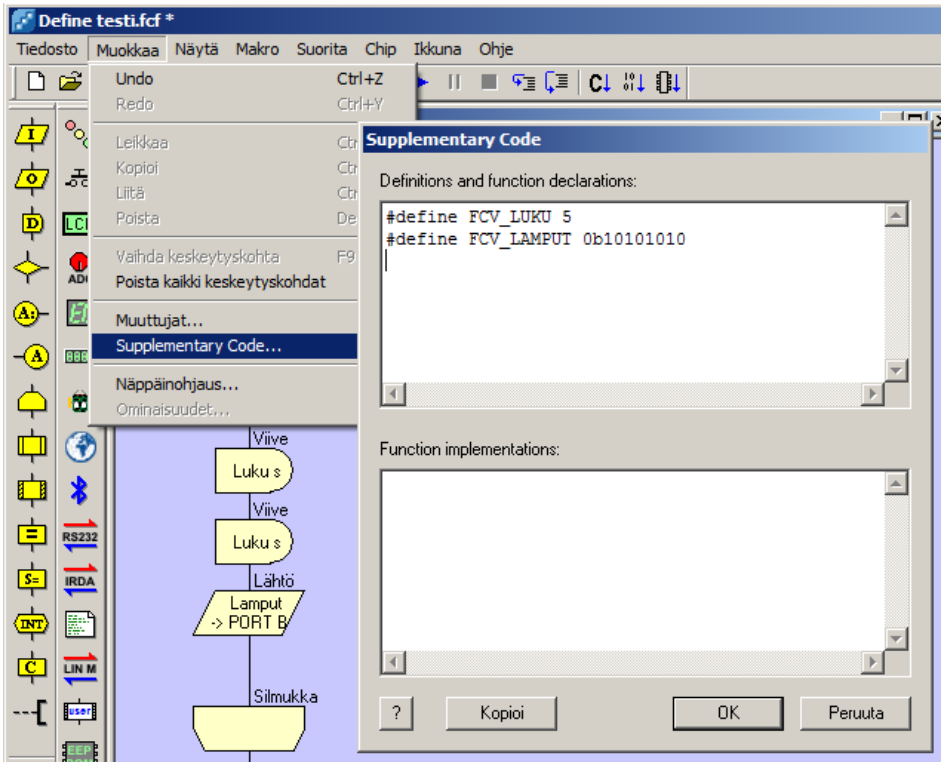


Kuva 29. Arvon voi antaa muuttujalle eri muodoissa.

Ohjelman kirjoittamisen ja tarkastelemisen takia muuttujia kannattaa käyttää myös pysyvien lukujen eli vakioiden nimeämiseen. Jonkin laskun ohjelmointi voi olla helpompaa, jos laskussa käyttää vaikkapa Kerroin-muuttujaa kuvaamaan pitkääkin

lukua. Ohjelmassa voi myös sama luku esiintyä useammassakin kohdassa, vaikka viiveen kestona. Esimerkiksi antamalla ohjelman alussa Odotellaan-muuttujan arvoksi 5 voidaan muuttujaa käyttää viiveiden kestona ja tarvittaessa myöhemmin muuttaa näiden kaikkien viiveiden kestoa suoraan määrityksestä. Ulkoisista kytkennöistä riippuen voi napin painallus aiheuttaa kytkettyyn nastaan tilan 0. Vastaavasti lamppu voi sammua, kun nastalle annetaan arvoksi 1. Epäselvyyksien välttämiseksi, varsinkin pitkissä ja monimutkaisissa ohjelmissa, kaikki tällaiset tilat kannattaa nimetä muuttujiksi, vaikka ne olisivatkin arvoltaan vakioita. Esimerkiksi ON-muuttujalle voidaan määrätä arvo 0 ja OFF-muuttujalle arvo 1.

Pienissä ohjelmissa, joissa muistinkäyttö ei ole ongelma, voidaan vakiomuuttuja määritellä myös laskutoimituksella. Muita vakion määritystapoja ovat vuokaavion käsky ”C-koodi”, Define-komponentti sekä Muokkaa-valikon ”Supplementary code”. Näistä Supplementary code on muistin käytön kannalta paras (kuva 30).



Kuva 30. Vakioden nimeäminen ja arvon määritys.

Supplementary code lisää sinne kirjoitetut C-kieliset rivit käännetyn C-koodin alkuun. Kääntäessään ohjelmaa C-kielelle Flowcode lisää muuttujiin myös FCV_ -etuliitteen. Määriteltävässä muuttujassa pitää siis käyttää etuliitettä ja isoja kirjaimia. Määrityksen alkuun pitää lisätä C-kielen komento #define ja loppuun annettava arvo. Jotta muuttujan nimeä voidaan vielä käyttää Flowcodessa, täytyy se lisätä ilman etuliitettä normaaliksi muuttujaksi. Esimerkiksi kuvan FCV_LAMPUT on lisätty tavun kokoiseksi Lamput-muuttujaksi ja FCV_LUKU sanan pituiseksi Luku-muuttujaksi.

Koska Supplementary code on C-koodiin lisättävä rivi, sitä ei pysty suoraan simuloimaan. Simulointia varten ohjelman alkuun lisätään ylimääräinen Laskutoimitus-käsky, jossa määritellään vakiomuuttujille arvot. Tämä ylimääräinen Laskutoimitus-käsky poistetaan ennen ohjelman latausta mikrokontrolleriin.

5.3 Operaattorit

Muuttujia muokkaavia laskutoimituksia ja ehtoina toimivia lausekkeita rakennetaan erilaisilla operaattoreilla. Flowcodessa operaattorit voidaan jakaa karkeasti matemaattisiin (taulukko 2), muokkaaviin, vertaileviin sekä loogisiin operaattoreihin.

Taulukko 2. Matemaattiset operaattorit

Operaattori	Käyttötapa	Esimerkki	Tulos
= + - *	muuttuja = luku1 + luku2	muuttuja = 1 + 2	muuttuja = 3
/	muuttuja = luku1 / luku2	muuttuja = 19 / 2 muuttuja = 31 / 8	muuttuja = 9 muuttuja = 3
MOD	muuttuja = luku1 MOD luku2	muuttuja = 19 MOD 2 muuttuja = 31 MOD 8	muuttuja = 1 muuttuja = 7

Matemaattisia operaattoreita käytetään laskutoimituksiin. Niitä käytettäessä täytyy kuitenkin muistaa, että laskeminen tapahtuu vain kokonaisluvuilla. Desimaalit pyöristyvät aina alaspäin, toisin sanoen katoavat.

Yhteen-, vähennys- ja kertolaskut ovat yksinkertaisia, kunhan pysytään muuttujan tyypin arvoalueella. Tavu tyypin muuttujaan ei siis voi syöttää arvoa, joka on suurempi kuin 255. Jakolaskujen kanssa ongelmaksi tulee pyöristys, esimerkiksi taulukon kummankin esimerkin oikeat tulokset ovat 9,5 ja 3,875. Oikein pyöristetyt tulokset olisivat siis 10 ja 4. Ongelma saadaan kierrettyä ”pilkku ja puolikas”-periaatteella. Kerrotaan jaettava kymmenellä, suoritetaan jako, lisätään viisi ja jaetaan takaisin kymmenellä. Taulukon esimerkkilaskut olisivat siis muotoa (((19*10)/2)+5)/10=10 ja (((31*10)/8)+5)/10=4,375. Desimaalien katoamisen jälkeen tuloksena olisi oikein pyöristyneet kokonaisluvut 10 ja 4. MOD operaattorilla saadaan jakolaskun jakojäännös käyttöön.

Muokkaavia operaattoreita käytetään laskutoimitus-käskyssä, ja niihin kuuluvat siirto- ja not-operaattorit (taulukko 3). Siirto-operaattorilla siirretään binääritietoa annetun askelmäärän verran. Siirrettäessä tietoa muuttujan lukutyypin ”ulkoa” tulevat ylimääräiset bitit ovat aina nollia ja ”ulos” siirtyvä tieto katoaa. Esimerkiksi ajattelemalla taulukon 3 siirtoesimerkit yhdeksi muuttujaksi, jota muokataan kaksi kertaa, huomataan kuinka muutama ykkönen katoaa.

Taulukko 3. Bittien siirto ja invertointi.

Operaattori	Käyttötapa	Esimerkki	Tulos
>> tai <<	muuttuja = luku << 4	muuttuja = 0b00001111 << 6 muuttuja = 0b11000000 >> 4	muuttuja = 0b11000000 muuttuja = 0b00001100
NOT	muuttuja = NOT luku	muuttuja = NOT 0b00101110 muuttuja = NOT 0b10011001 muuttuja = NOT 0b00000001	muuttuja = 0b11010001 muuttuja = 0b01100110 muuttuja = 0b11111110

Not-operaattori vaihtaa annetun luvun bittien tilan käänteiseksi, nollat muuttuvat ykkösiksi ja ykköset nolliksi. Not-operaattorin käyttö loogisena totuusehtona toimii

vain jos muuttujan kaikkien bittien tila on sama. Esimerkiksi tavu-muuttujan arvon ollessa 1 sen not-muunnos on 254. Kumpikin tila on nollasta poikkeava eli tosi, looginen toiminta ei siis päde. Kun muuttuja on 255, eli tavun kaikki bitit ovat ykkösiä, niin sen not-muunnos on 0, eli epätosi. Looginen totuus vaihtuu todesta epätodeksi. Taulukon lukujen paikalla voidaan käyttää myös muuttujia antamaan lähtöarvot.

Vertailevia ja loogisia operaattoreita käytetään pääasiassa silmukka- ja päätös-käskyjen ehtolauseissa (taulukko 4). Vertailu tarkastelee annettuja lukuja tai muuttujia ja antaa vastauksen ykkösenä tai nollana sen mukaan, onko annettu vertailuväite tosi vai epätosi.

Taulukko 4. Vertailevat ja loogiset operaattorit.

Operaattori	Käyttötapa	Esimerkki	Tulos (totuus)
< <=	luku1 >= luku2	(muuttuja =) 123 >= 234	0b00000000 (epätosi)
> >=		(muuttuja =) 123 >= 123	0b00000001 (tosi)
= <>	luku1 = luku2	(muuttuja =) 34 = 27	0b00000000 (epätosi)
	luku1 <> luku2	(muuttuja =) 34 <> 27	0b00000001 (tosi)
AND	luku1 AND luku2	0b01100110 AND 0b00110111	0b00100110 (tosi)
		0b10001100 AND 0b00110001	0b00000000 (epätosi)
		0b00000000 AND 0b00000000	0b00000000 (epätosi)
OR	luku1 OR luku2	0b01100110 OR 0b00110111	0b01110111 (tosi)
		0b10001100 OR 0b00110001	0b10111101 (tosi)
		0b00000000 OR 0b00000000	0b00000000 (epätosi)
XOR	luku1 XOR luku2	0b00000000 XOR 0b00110111	0b00110111 (tosi)
		0b10001100 XOR 0b00110001	0b10111101 (tosi)
		0b00000000 XOR 0b00000000	0b00000000 (epätosi)

And-, or- ja xor-operaattorit vertaavat annettujen arvojen binäärimuotoja ja niiden samoilla paikoilla olevia bittejä. Tuloksena on ykkönen tai nolla vertailtavaan kohtaan binäärilukua. Esimerkiksi and-operaattori vertaa annettujen lukujen ensimmäistä bittiä. Jos vertailtavat bitit ovat kumpikin ykkösiä, saa tulos ykkösen ensimmäisen bitin kohdalle binäärilukua, muutoin tulos on nolla. Tämän jälkeen verrataan kummankin binääriluvun toista bittiä ja annetaan vastaus tuloksen toiseen bittiin. And-operaattori antaa siis loogisen toden, jos yksikin bittipari on tosi.

Ongelmia voi kuitenkin tulla, kuten taulukon toisessa and-esimerkissä. Vaikka kumpikin vertailtava luku olisikin nollasta poikkeava, eli yksinään tosi, voi yksittäisten bittien vertailu antaa silti tuloksena epätoden.

Or-operaattori testaa samoin annettujen lukujen bittipareja. Tuloksena on nolla, jos kumpikin bitti on nolla. Muissa tilanteissa tulos on yksi. Or-operaattori toimii myös loogisesti oikein. Jos annetuista luvuista yksikin on nollasta poikkeava, niin silloin tuloksen binäärimuodossa on vähintään yksi ykkönen ja tulos on tosi.

Xor-operaattori antaa tuloksen bittiin ykkösen, jos bittiparissa on ykkönen ja nolla. Toisin sanoen tulokseen tulee nolla, jos vertailtavat bitit ovat kumpikin joko ykkösiä tai nollia. And-operaattorin tavoin tämäkin operaattori toimii loogisesti väärin. Esimerkiksi taulukon toisen xor-esimerkin pitäisi olla loogisesti epätosi, koska kumpikin annetuista luvuista on tosia.

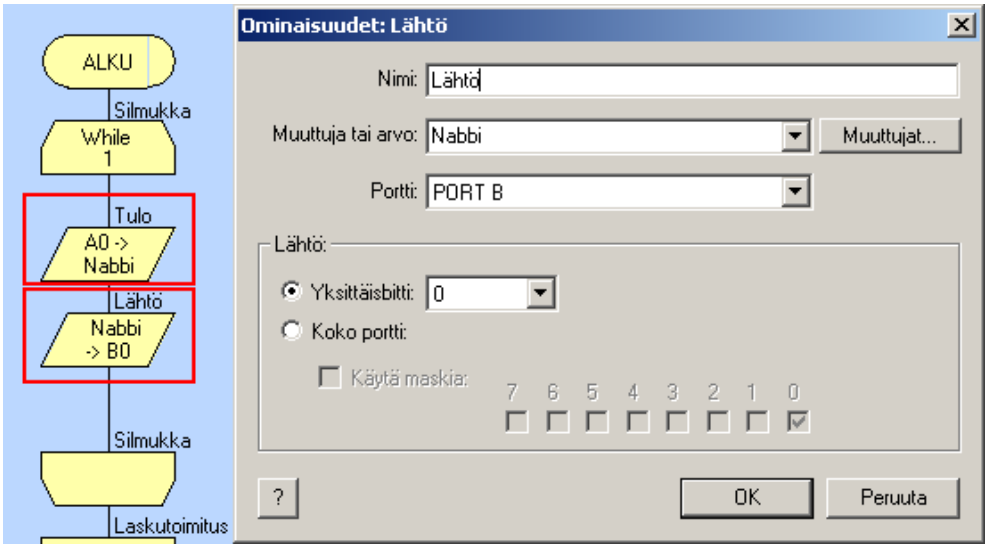
Laskutoimituksissa ja ehtolauseissa kannattaa käyttää sulkeita, joilla määrätään laskennan tai tarkastelun järjestys. Esimerkiksi ehto ”kierrosten ollessa 1500 ja 2000 välillä” voidaan kirjoittaa ”(Kierrokset-muuttuja > 1500) AND (Kierrokset-muuttuja < 2000)”. Ehto tarkastaa ensiksi suurempi ja pienempi kuin –vertailut, antaa näille tuloksena nollan tai ykkösen. Jos kumpikin vertailu on ykkönen, tulee koko lausekkeesta silloin tosi.

5.4 Käskyt ja niiden käyttö

Ohjelma koostuu käskyistä, jotka suoritetaan käsky kerrallaan, halutussa järjestyksessä. Suoritettuaan yhden käskyn ohjelma siirtyy seuraavaan käskyyn. Käskyt raahataan komentopalkista vuokaavioon haluttuun kohtaan. Kunkin käskyn ominaisuuksia pääsee muokkaamaan tuplaklikkaamalla kyseistä käskyä. Tässä kappaleessa käydään kaikki komento-palkin käskyt lävitse.

5.4.1 Tulo ja lähtö

Näiden käskyjen toimintatapa ja samalla myös ominaisuudet-ikkuna (kuva 31) on samankaltainen. Tulo (input) tarkastaa halutun nastan tai portin tilan ja tallentaa tiedon muuttujaan. Lähtö (output) taas asettaa nastat muuttujan määräämään tilaan. Ero tulee siinä, että tulo tarvitsee aina jonkin muuttujan mihin tieto tallennetaan, kun taas lähtöön voi muuttujan tilalle antaa luvun suoraan.



Kuva 31 Tulo ja lähtö

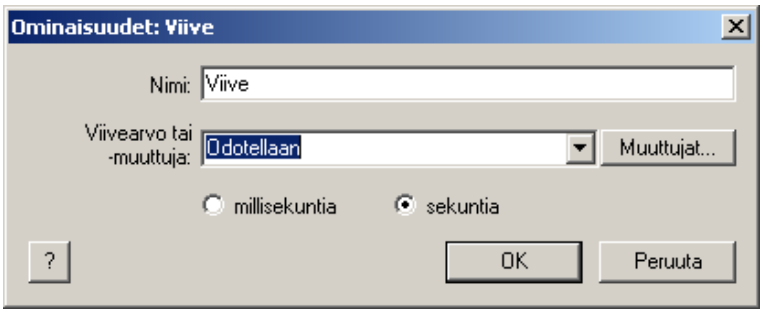
Kuvan lähtö antaa B-portin ensimmäiselle bitille (RB0, piirin 33. nasta) Nabbi-muuttujassa olevan arvon. Nastan jännite on 0 V, jos muuttujan arvo on nolla. Kaikilla muilla muuttujan arvoilla nastan tila on tosi eli 5 V, myös negatiivisilla arvoilla. Jos kyseessä olisi tulon ominaisuudet-ikkuna, tallentuisi B-portin ensimmäisen nastan tila, nolla tai ykkönen, Nabbi-muuttujaan. Tulossa sama toimii siis päinvastoin, nastojen sen hetkinen tila tallennetaan annetun muuttujan arvoksi.

Lähdössä Koko portti –vaihtoehtoa käytettäessä annettu arvo tai muuttujassa oleva luku muutetaan tarvittaessa tavun mittaiseksi (8-bittiä) binääriluvuksi, josta jokainen bitti määrää yhden nastan tilan.

Käytä maskia –valinnalla merkataan lähdössä, mitkä nastat ovat vaikutuksen alaisia, merkkaamattomien nastojen tila pysyy siis entisellään. Esimerkiksi merkkaamalla 0- ja 7-bitti ja antamalla lähdön arvoksi 255 (0b11111111), B0 ja B7 saisivat tilan 1 ja muut nastat säilyttäisivät edellisen tilansa, oli se sitten 0 tai 1. Käyttämällä maskia tulossa luetaan merkattujen nastojen tila ja annetaan merkkaamattomille nastoille arvoksi 0. Esimerkiksi merkkaamalla tuloon maskiksi 1. ja 4. bitti ja painamalla kaikkia porttiin kytkettyjä kytkimiä tallentuu muuttujaan luku 18 (00010010).

5.4.2 Viive

Mikrokontrolleri suorittaa ohjelmaa melkosella vauhdilla, joten ohjelmassa voidaan tarvita taukoja. Esimerkiksi kytkintä painettaessa voi ledi pysyä päällä vaikka 15 sekuntia. Viiveen (delay) lukuarvo annetaan lukuna tai muuttujana ja valitaan luku sekunneiksi tai millisekunneiksi (kuva 32).

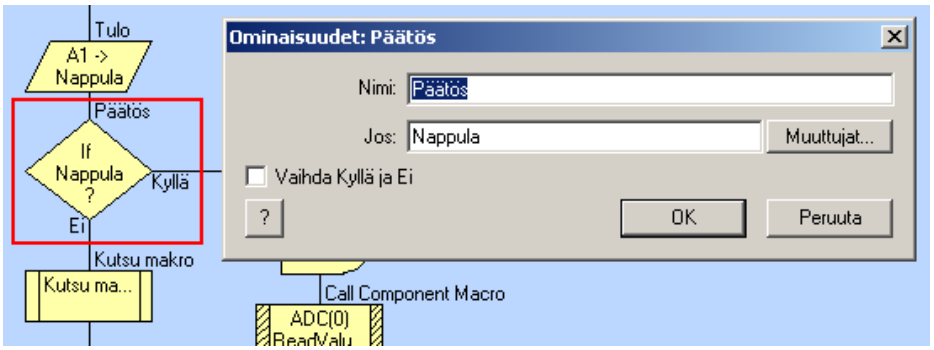


Kuva 32 Viiveen ominaisuudet.

Viiveen suorituksen aikana mikrokontrolleri ei suorita mitään muuta, poikkeuksena keskeytys-käskyt. Esimerkiksi napin painalluksen odottelu, jonka pitäisi tulla vaikka 10 sekunnin sisällä, voidaan ohjelmoida viivettä ja keskeytystä käyttäen.

5.4.3 Päätös

Ohjelmassa voi tulla eteen tilanne, jossa on tarpeen valita kahden eri toimintareitin väliltä. Päätös-käskyllä (decision) ohjelma testaa, onko ehto tosi. Ehto on ominaisuudet-ikkunan jos-rivillä oleva lauseke tai yhtälö. Esimerkiksi kytkimen tila voidaan tarkastaa, arvon ollessa nollasta poikkeava eli tosi valitaan Kyllä-reitti ja arvon ollessa nolla eli epätosi valitaan Ei-reitti (kuva 33).

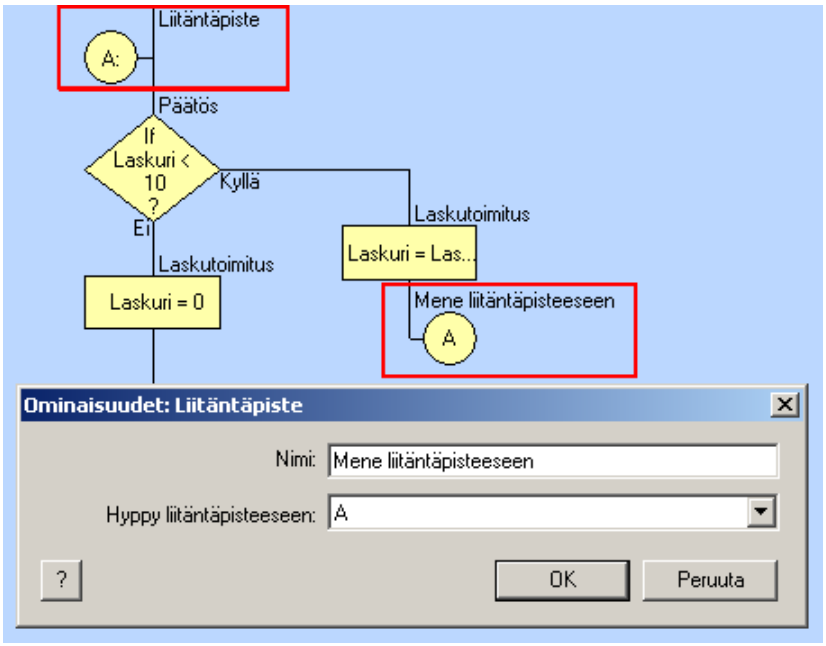


Kuva 33. Päätöksellä valitaan haluttu reitti.

Ehtona oleva lauseke voidaan muodostaa käyttämällä matemaattisia ja loogisia operaattoreita, esimerkiksi nappi1 AND nappi2.

5.4.4 Liitäntäpiste ja hyppy

Jossakin kohtaa ohjelmaa voi tulla tarve päästä hyppäämään joidenkin käskyjen yli, joko eteen- tai taaksepäin. Tämän kaksiosaisen käskyn toiminta on hyvin yksinkertainen ja suoraviivainen. Lisätään ohjelmaan liitäntäpiste (connection point) ja käsky mennä tuohon pisteeseen (kuva 34).

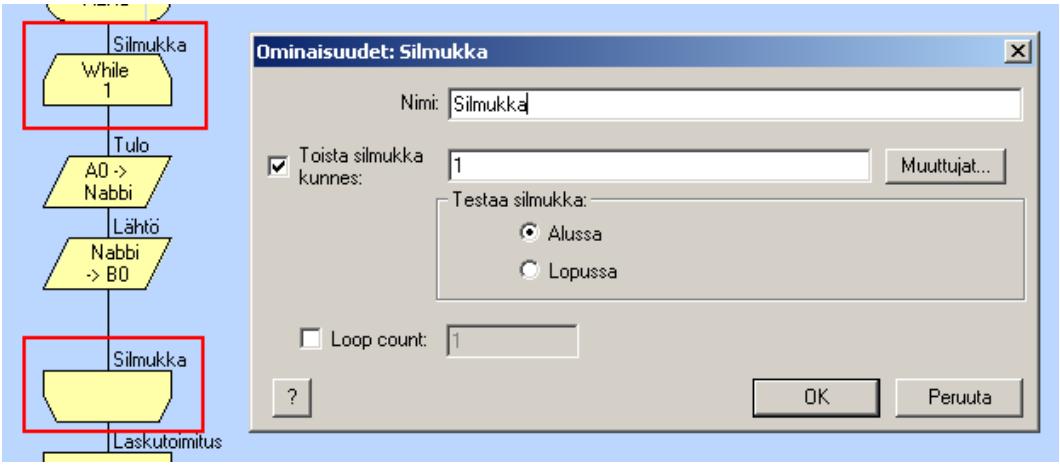


Kuva 34. Hyppykäskyllä siirrytään liitäntäpisteeseen.

Liitäntäpisteitä voi olla enintään 26, jotka erotellaan toisistaan aakkosilla. Hyppykäskyjä samaan pisteeseen taas voi olla useampia.

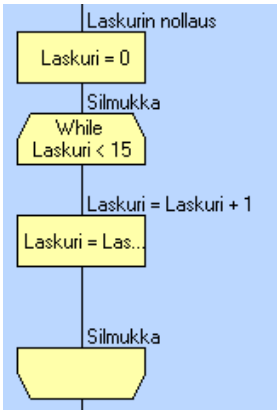
5.4.5 Silmukka

Mikrokontrolleriin ladattavan ohjelman halutaan lähes aina toimivan niin kauan kun kontrolleri on päällä. Joissakin tilanteissa halutaan, että jokin käskysarja toistuu useamman kerran peräkkäin. Kumpaankin näistä voidaan käyttää Silmukka-käskyä (kuva 35).



Kuva 35. Silmukalla saadaan käskysarja toistumaan useamman kerran.

Ominaisuudet-ikkunan ”Toista silmukka kunnes”-kohta on merkitykseltään väärä. Todellisuudessa tekstin pitäisi olla ”Toista silmukka jos”! Käsky toimii siis siten, että kentässä olevan ehdon ollessa tosi suoritetaan silmukassa olevat käskyt. Ehdon testaus alussa määrää sen, mennäänkö silmukkaan vai hypätäänkö silmukan yli. Jos testaus on silmukan lopussa, niin silmukkaan mennään ja suoritetaan käskyt ainakin kerran. Testaus lopussa määrää siis sen, hypätäänkö silmukasta ulos vai palataanko silmukan alkuun.



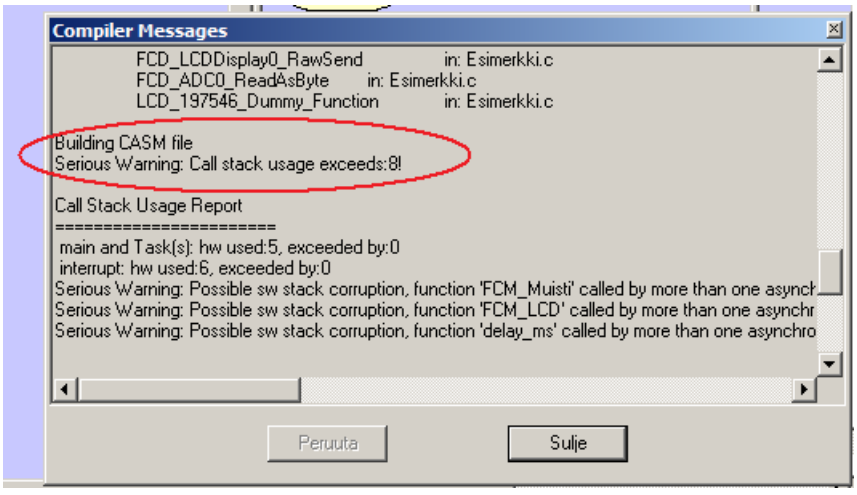
Kuva 36 Laskuri

Ehtona voidaan käyttää muuttujaa tai muuttujan vertailua loogisilla ja matemaattisilla operaattoreilla (kuva 36).

Käyttämällä silmukan ehtona ykköstä ehto on aina tosi ja saadaan päättymätön silmukka. Yleensä tällainen silmukka luodaan ensimmäisenä uutta ohjelmaa laadittaessa. Tuleva ohjelma laaditaan tällaisen sisään.

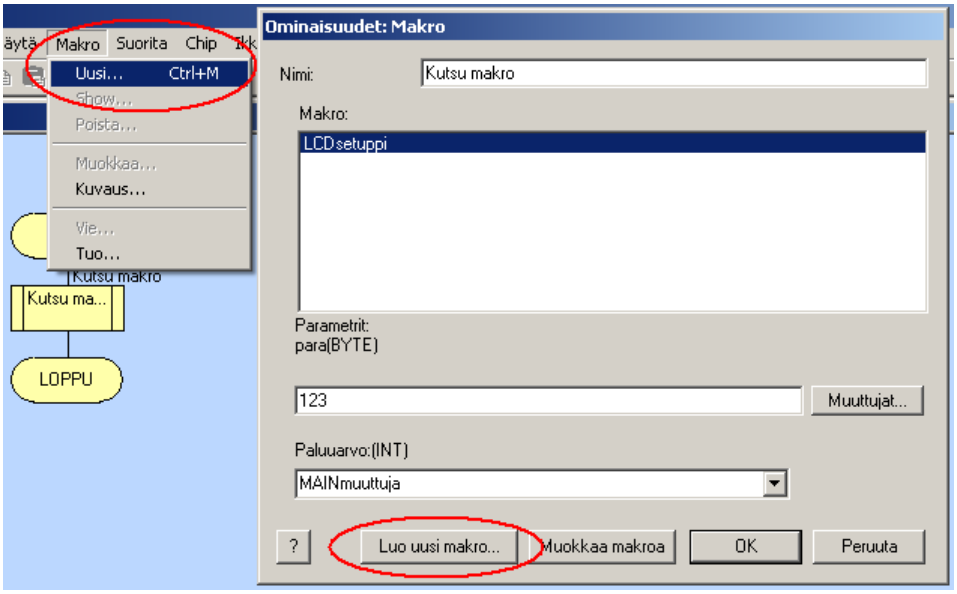
5.4.6 Kutsu makro

Aliohjelma on pääohjelman sisällä oleva oma ohjelmansa, joka sisältää oman vuokaavionsa ja toimintonsa. Aliohjelma eli makro käynnistetään pääohjelmasta Kutsu makro –käskyllä (call macro). Pitkä ja paljon komponentteja sisältävä ohjelma voi näyttää todella sekavalta. Makrojen käytöllä saadaan siistittyä pääohjelmaa, joka helpottaa niin ohjelman tekoa kuin mahdollisten virheiden etsimistä. Toinen hyödyllinen asia on makrojen sisäisten muuttujien eli paikallisten muuttujien käyttö, joka joissakin tilanteissa säästää käyttömuistia. Makroja ei saa kuitenkaan luoda liikaa. Ohjelman latauksen aikana ilmestyvä kääntäjä-ikkuna antaa varoituksen, jos sallittu maksimi on ylitetty (kuva 37).



Kuva 37. Kääntäjä varoittaa, jos makroja on liikaa.

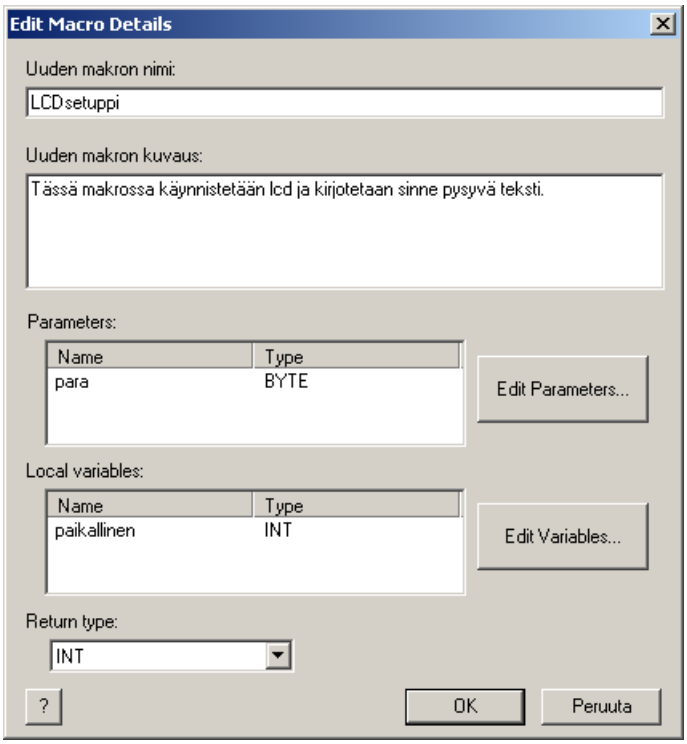
Uuden makron voi luoda joko Kutsu makro –käskyn ominaisuudet-ikkunasta tai Makro-valikosta (kuva 38). Itse ohjelman kirjoitus makroon toimii samalla tavalla kuin pääohjelmaankin, raahataan käskyjä makron omaan vuokaavioon ja määritellään niiden ominaisuudet.



Kuva 38. Uuden makron luonti ja Kutsu makro-käskyn ominaisuudet-ikkuna.

Käskyn ominaisuudet-ikkunasta valitaan kutsuttava makro. Jos makrolle on määritelty sisäisiä parametrimuuttujia, niille annetaan halutut arvot joko lukuna tai muuttujan arvona. Jos paluuarvoa käytetään, niin se syötetään johonkin pääohjelman muuttujaan.

Uuden makron luonti –ikkunassa annetaan makrolle välttämön nimi (kuva 39). Mahdollinen kuvaus näkyy makron omassa vuokaaviossa. Kolme muuta kohtaa on makrossa käytettäviä muuttujia, joita ei tarvitse välttämättä käyttää. Makro-muuttujien lisäys ja muokkaus valmiiseen makroon onnistuu Makro-valikon Muokkaa-kohdasta, kun haluttu aliohjelma on päällimmäisenä.



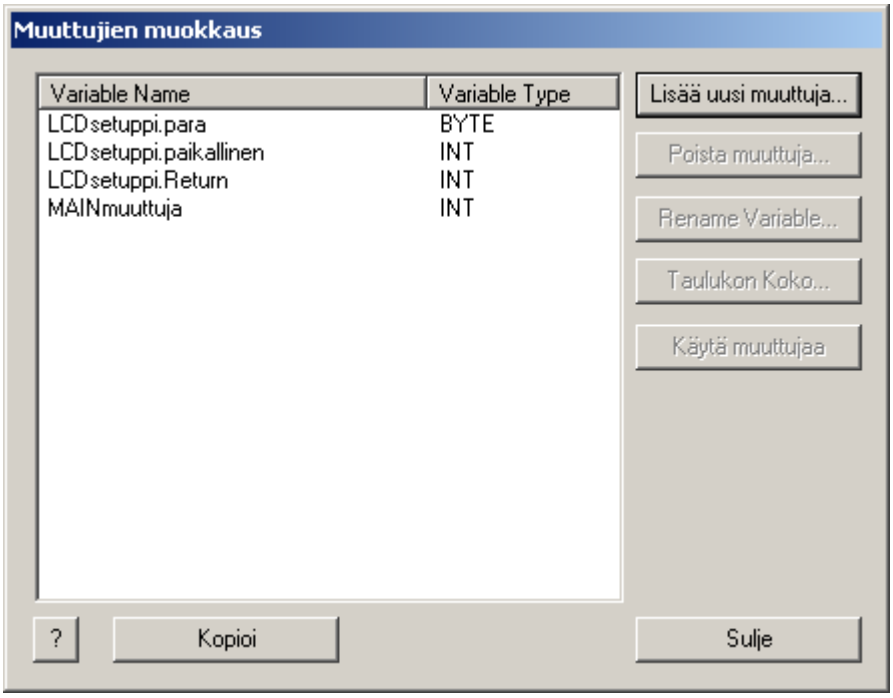
Kuva 39. Makrolle annetaan nimi, sekä haluttaessa kuvaus ja muuttujat.

Parametri-kohtaa käyttämällä voidaan luoda makroon muuttujia, joille annetaan Kutsu makro –käskyllä halutut arvot. Esimerkiksi moottorin käyntiaikaa ohjataan aliohjelmalla ja siellä olevalla viiveellä. Viiveen kestona käytetään parametri-muuttujaa, jolle annetaan kutsussa arvo. Näin eri kohdissa pääohjelmaa saadaan samalla aliohjelmalla pidettyä moottoria päällä eri ajan.

Paikallisia muuttujia (Local variables) käytetään parametri-muuttujien tapaan vain oman makronsa sisällä, mutta niiden arvo määritellään vasta aliohjelmassa. Ne varaavat käyttömuistia vain silloin, kun aliohjelma on käynnissä. Ohjelmakierron palatessa pääohjelmaan muistivaraus vapautuu muuhun käyttöön.

Paluuarvolla (Return type) luodaan makroon Return-niminen muuttuja, jonka sisältämä arvo annetaan jollekin pääohjelman muuttujalle makron suorituksen jälkeen.

Aliohjelman muuttujan nimi muodostuu annetun nimen ja aliohjelman nimen mukaan (kuva 40). Makrossa voidaan käyttää myös pääohjelman yleisiä muuttujia (Global variables), jotka näkyvät joka paikassa omalla nimellään.



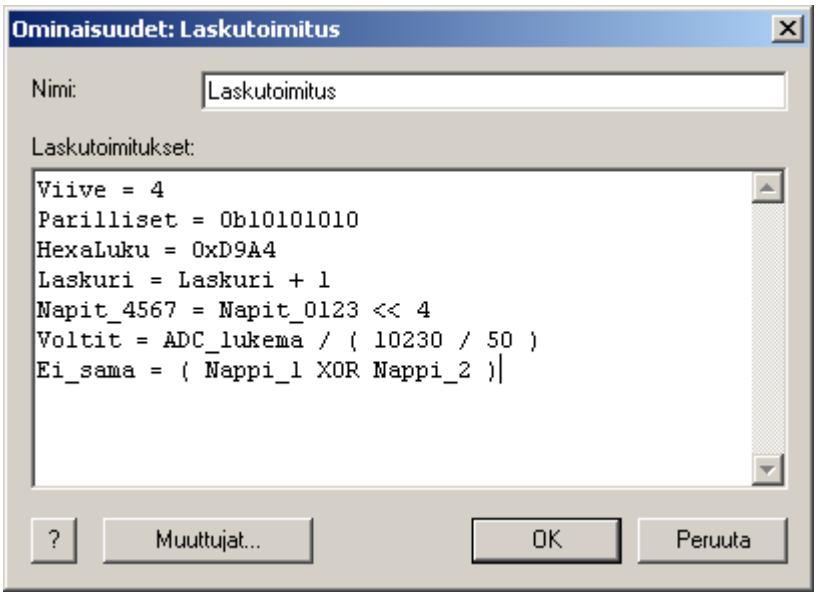
Kuva 40. Aliohjelmilla on käytössään paikalliset ja yleiset muuttujat.

5.4.7 Kutsu komponenttimakro

Mikrokontrolleriin liitettyjen laitteiden, kuten lcd-näyttö, ohjaaminen tarvitsee melko monimutkaisia ohjeita. Kutsu komponenttimakro-käskyllä (call component macro) ohjataan kytkettyjen komponenttien toimintoja. Komponenttimakrot ovat siis valmiiksi ohjelmoituja makroja. Tämän käskyn käytöstä lisää kappaleessa 4.6 Komponentit.

5.4.8 Laskutoimitus

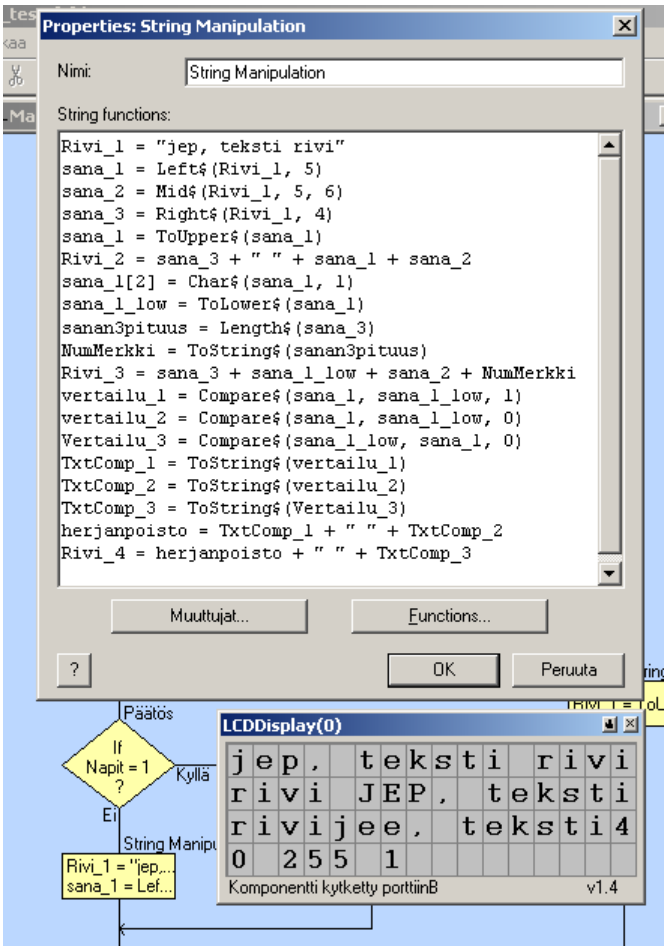
Muuttujien arvojen muuntaminen onnistuu tällä käskyllä. Pienemmissä ohjelmissa laskutoimitusta (calculation) voi käyttää myös kiinteän muuttujan määritykseen. Laskuissa voi käyttää kappaleessa 4.3 esiteltyjä operaattoreja. Saatu tulos sijoitetaan aina johonkin muuttujaan, joten kaikki laskut alkavat tuloksena olevan muuttujan nimellä ja yhtäsuuruusmerkillä (kuva 41). Ohjelma suorittaa käskyn sisältämät laskut yksi kerrallaan, aloittaen ylimmästä.



Kuva 41. Laskutoimituksella muokataan muuttujan arvoa.

5.4.9 Teksti-muuttujan muokkaus

String manipulation –käskyllä saadaan määriteltyä ja muokattua teksti-muuttujia. Kuten laskutoimituksessa, jokainen rivi tarvitsee muutoksen kohteen ja yhtä kuin –merkin. Tekstin muokkaus tapahtuu funktioilla (kuva 42).



Kuva 42. String-muuttujien käsittely.

Kuvan esimerkeistä Rivi_X-muuttujat on tulostettu näytölle havainnollistamaan muutoksia. Muuttujista sanan3pituus ja vertailu_X ovat tavallisia tavun pituisia numeromuuttujia. Kaikki loput ovat erikokoisia string-muuttujia. Kuvassa on käytetty kaikkia, functions-napin takaa löytyviä funktioita, jotka seuraavassa käydään läpi.

Kuvan ensimmäinen muutos on perusmääritys, syötetään haluttu teksti lainausmerkeissä muuttujaan. Rivi_1-muuttujasta poimitaan sana_1:een viisi ensimmäistä merkkiä Left$()-funktiolla, sana_2:een Mid$()-funktiolla kuusi merkkiä osoitteesta 5 lähtien (taulukon osoitteet alkavat nollasta) ja sana_3:een

neljä viimeistä merkkiä Right()-funktiolla. Sana_1-muuttujan kirjaimet on nostettu ToUpper$()-funktiolla isoiksi kirjaimiksi ennen Rivi_2-muuttujan muodostusta, jonka määritykseen on lisätty myös yksi välilyönti.

Char$()-funktiolla sana_1:n osoitteeseen 2 sijoitetaan saman muuttujan osoite 1:n merkki. Vaikka sana_1 muuttuu, niin muutos ei näy Rivi_2:ssa, koska ohjelma lukee rivit ylhäältä alas, rivi kerrallaan. ToLower$()-funktiolla sana_1_low-muuttujaan sijoitetaan sana_1:n sisältö pienillä kirjaimilla. Length$()-funktiolla sanan3pituus saa arvon, joka vastaa sana_3:n merkkien määrää. Lukuarvo muutetaan numerokirjaimeksi ToString$()-funktiolla, jossa voi käyttää myös numeroita muuttujan sijaan sulkeiden sisällä. Rivi_3-muuttujaan summataan muutokset.

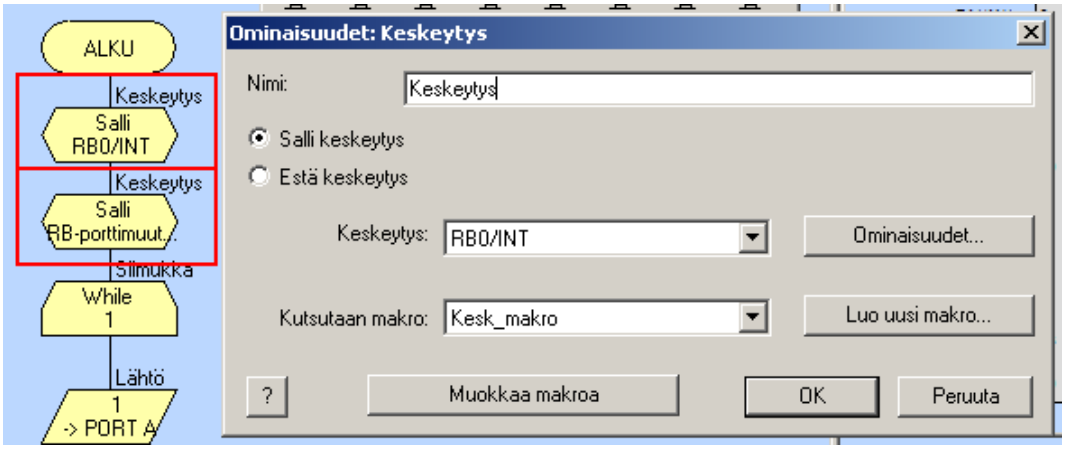
Compare$()-funktio vertaa kahta string-muuttujaa ja antaa vertailun päätteeksi tavu-arvona jonkin seuraavista: 0 jos muuttujat ovat identtisiä, 1 jos ensimmäinen muuttuja on myöhemmin ja 255 jos aikaisemmin ASCII-taulukossa. Funktion viimeisellä numerolla valitaan, erotellaanko isot ja pienet kirjaimet (0) vai ei (1).

Kuvassa, ennen Rivi_4:n muodostusta, on käytetty ylimääräistä muuttujaa. Flowcode ei pysty lisäämään kahta välilyöntiä kerralla.

5.4.10 Keskeytys

Normaalista ohjelmakierrosta poikkeavat aliohjelmat, joko ajastetut tai välittömästi käskyyn reagoivat, käynnistetään keskeytyksellä (interrupt).

Ohjelmaan sijoitetusta keskeytys-kuvakkeesta alkaen joko sallitaan keskeytys tai estetään aikaisemmin sallittu keskeytys. Keskeytyksen tyypistä riippuen itse keskeytysrutiini ja sen johdosta suoritettava makro alkavat myöhemmin (kuva 43).



Kuva 43. Keskeytykselle valitaan tyyppi ja kutsuttava makro.

RBO/INT-keskeytys käynnistyy B-portin ensimmäisestä nastasta. Ominaisuuksista valitaan käynnistyssignaalin nouseva tai laskeva reuna. Minkä tahansa B-portin neljän viimeisen nastan tilan muutoksesta käynnistyy RB-porttimuunnos-niminen keskeytys. Käynnistys tapahtuu siis bittien 4 - 7 noustessa ja laskiessa.

TMR0-ylivuoto (overflow) on ajastettu keskeytys, joka laskee joko sisäisen kellon (CLKO) tai T0CKI-nastan (A-portti, 4-bitti) ulkoisia pulsseja. Esiskaalauksesta ja kellotaajuudesta riippuen keskeytysrutiini käynnistyy tietyin väliajoin, kun pulsseja on kertynyt tarpeeksi. Keskeytys käynnistää siis asetetun makron, nollaa laskurin ja aloittaa laskemisen alusta. Ominaisuuksista valitaan sisäinen kello ja esiskaalauksen aste. Jos järjestelmän kellotaajuus on asetettu oikein, ominaisuus-ikkuna näyttää myös todellisen keskeytystaajuuden.

Keskeytysten tyypit ja ominaisuudet riippuvat käytettävästä mikrokontrollerista. Tarkemmat tiedot löytyvät kyseisen kontrollerin datalehdeltä.

5.4.11 C-koodi

Tällä käskyllä voi lisätä ohjelmaan C- tai assembler-kielellä kirjoitettua koodia. Lisättyä koodia ei pysty pysty simuloimaan. Lisäksi muuttujien käyttämisessä, sekä koodissa että vuokaaviossa, täytyy muistaa Flowcoden muuttujan nimen muodostuminen (luku 4.4)

5.4.12 Kommentti

Hyvä ohjelma ei ole pelkästään toimiva. Ohjelman tarkastus, korjaaminen ja päivitys ovat asioita, jolloin lukijana voi olla joku muu kuin ohjelman alkuperäinen laatija. Sen takia ohjelman pitää olla selkeä ja helposti ymmärrettävä. Ohjelman rakenteen tulisi olla järkevä, ja käytettyjen muuttujien nimien pitäisi kuvata muuttujien toimintaa. Kommentilla voi käskyjen väliin lisätä myös vapaamuotoista tekstiä, joka ei vaikuta ohjelman toimintaan. Kommenttiin voidaan kirjoittaa esimerkiksi sitä seuraavien käskyjen toiminnan kuvaus.

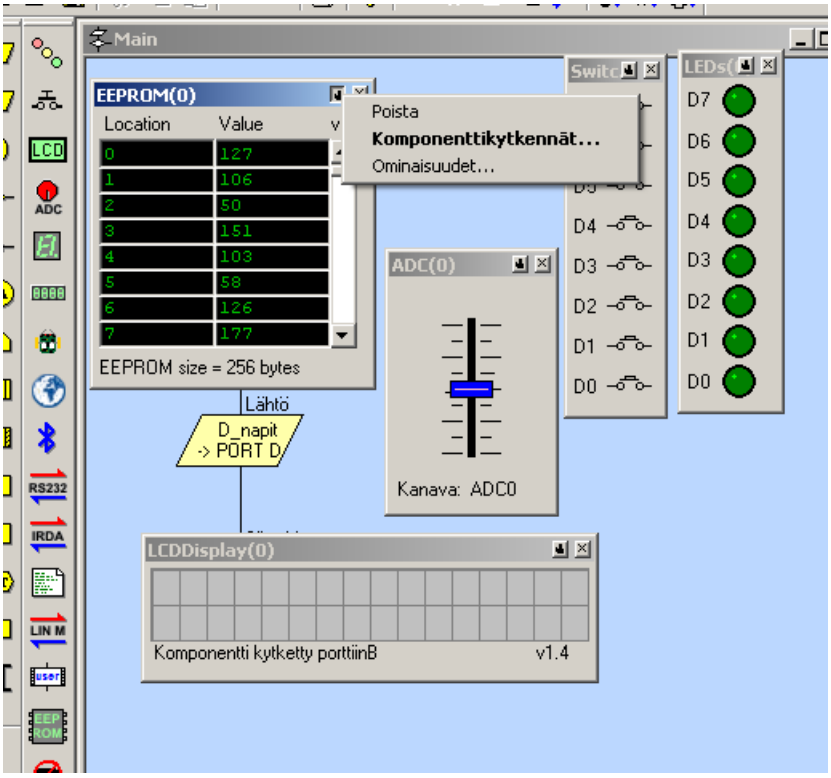
5.5 Komponentit

Flowcode sisältää erilaisia valmiiksi esiohjelmoituja makroja, joita mikrokontrolleri käyttää ohjaamaan siihen kytkettyjä laitteita (esim. LCD-näyttö) ja joitakin sisäisiä toimintoja (esim. EEPROM-muisti). Näitä laitteita ja toimintoja kutsutaan Flowcodessa komponenteiksi.

Komponentit saa liitettyä ohjelmaan komponentti-palkista, jonka jälkeen liitetyn komponentin ohjaus tapahtuu komponenttimakro –käskyllä. Jotkin komponentit

tarvitsee liittää, jotta niiden ohjaukseen tarvittavat käskyt tulevat esiin. Joitakin komponentteja taas ei tarvitse liittää, mutta simulointia ajatellen se kannattaa tehdä. Esimerkiksi kytkimien ja ledien ohjaus onnistuu myös suoraan tulo- ja lähtö-käskyillä, mikä kuluttaa myös huomattavasti vähemmän muistia kuin komponentti makron käyttö.

Liittämisen jälkeen Flowcodeen ilmestyy kyseisen komponentin ikkuna. Kaikkien komponenttien ikkunassa, oikeassa yläkulmassa, on kaksi nappia. Nuoli-valikosta säädetään komponentin ominaisuuksia ja kytkentäkohtaa sekä poistetaan tarpeeton komponentti (kuva 44).



Kuva 44. Komponenttien kytkentä ja ominaisuudet valitaan nuoli-valikosta.

Ruksia painamalla saadaan komponentti piilotettua. Tämä ei kuitenkaan poista komponenttia, vaan komponentti on edelleen liitettynä. Samaa komponenttia voi

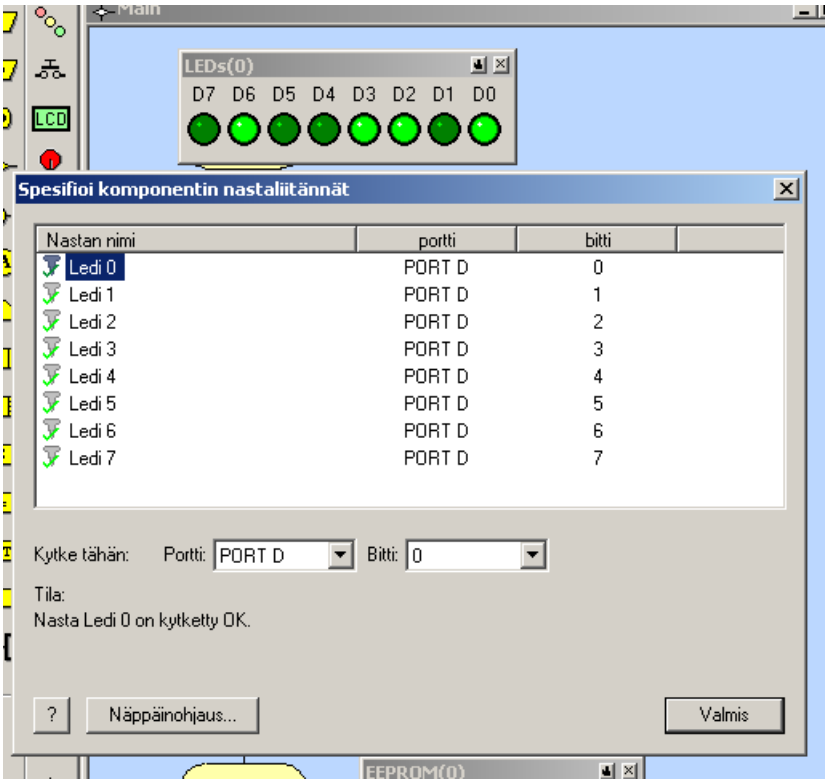
liittää useammankin kerran, komponentit erotellaan tällöin nimen perässä olevalla numerolla. Tämä voi aiheuttaa sekaannusta ja ohjelman toimimattomuutta, jos esimerkiksi luulee poistaneensa jonkin komponentin ruksista ja lisänneensä sen uudestaan ohjelmaan. Tällöin ohjelmassa on kaksi samaa komponenttia ja piilotettuna olevaa komponenttia voidaan ohjelmoida ”vahingossa”, eikä simuloitaessa tapahdu mitään. Liitettyjen komponenttien määrän näkee Näytä-valikosta, Kytketyt komponentit –kohdasta, josta myös piilotetut komponentit saa uudestaan näkyviin.

Joitakin komponentteja voidaan simuloinnin aikana ohjata tietokoneen näppäimistön numeronäppäimillä. Esimerkiksi kytkimen voi liittää vaikka näppäimeen 6. Näppäinohjausta pääsee muuttamaan Muokkaa-valikosta tai komponenttikytkennät-ikkunasta.

5.5.1 Led-komponentti

Tätä komponenttia voidaan käyttää kahteen tarkoitukseen, joko simuloinnin aikana nastojen tilan seuraamiseen tai yksittäisen nastan ohjaamiseen komponenttimakroa käyttämällä.

Komponentin ominaisuuksista voidaan valita tarvittava ledien määrä ja muita Flowcodessa näkyviä asioita, kuten nimi ja väri. Komponenttikytkennöistä valitaan liitettävä portti ja sen nastat (kuva 45).



Kuva 45. Komponenttikytkennät ja D-portin nastojen tila.

Komponenttimakroilla saadaan parametriin annetun nastan tila muutettua, esimerkiksi kuvan kahdeksas ledi saadaan syttymään LEDOn-makron parametrilla 7 ja kolmas ledi sammumaan LEDOff-makron parametrilla 2. Tätä ominaisuutta ei kuitenkaan kannata käyttää, koska yhdenkin ledin sytys komponenttimakrolla vie muistia yli neljä kertaa enemmän kuin kokonaisen portin tilan muutos lähtö-käskyllä! Ledien ja nastojen ohjaaminen Lähtö-käskyllä kannattaa siis opetella ensimmäisenä.

Simuloinnin aikana ledien ja nastojen tilan seuraamiseen Led-komponentti soveltuu hyvin. Led-komponentti näyttää siis siihen kytketyn nastan loogisen tilan. Kuvassa D-portin neljännen nastan tila on 1 (+5 V) ja viidennen 0 (0 V). Kuitenkin täytyy muistaa, että simulointi ja todellinen tilanne voi olla käänteinen. E-Block-sarjan

led-kortti on kuitenkin kytkennältään sellainen, että nastan tila 1 sytyttää ledin ja 0 sammuttaa sen, joten ledien simulointi vastaa ulkoista tilannetta.

5.5.2 Kytkin-komponentti

Liittämällä tämä komponentti voidaan tulo-nastojen tilaa muutella simuloinnin aikana, eli käyttää normaaleina painonappeina tai rajakytkiminä. Kytkimen komponentti-makroja voidaan käyttää myös ohjelmassa, joko lukemaan yksittäisen tulon tilaa tai odottelemaan sen muutosta.

Komponenttikytkennöistä valitaan, mihin porttiin komponentti on liitetty. Ominaisuuksista voidaan valita esimerkiksi kytkimen tyyppi ja määrä, sekä nimetä kytkimiä. Kytkinvärähtely-kohta (Debounce) vaikuttaa kytkin-komponentin WaitUntil-makroihin lisäämällä niihin viivettä, jolla taas saadaan mahdolliset virhetilat kuriin.

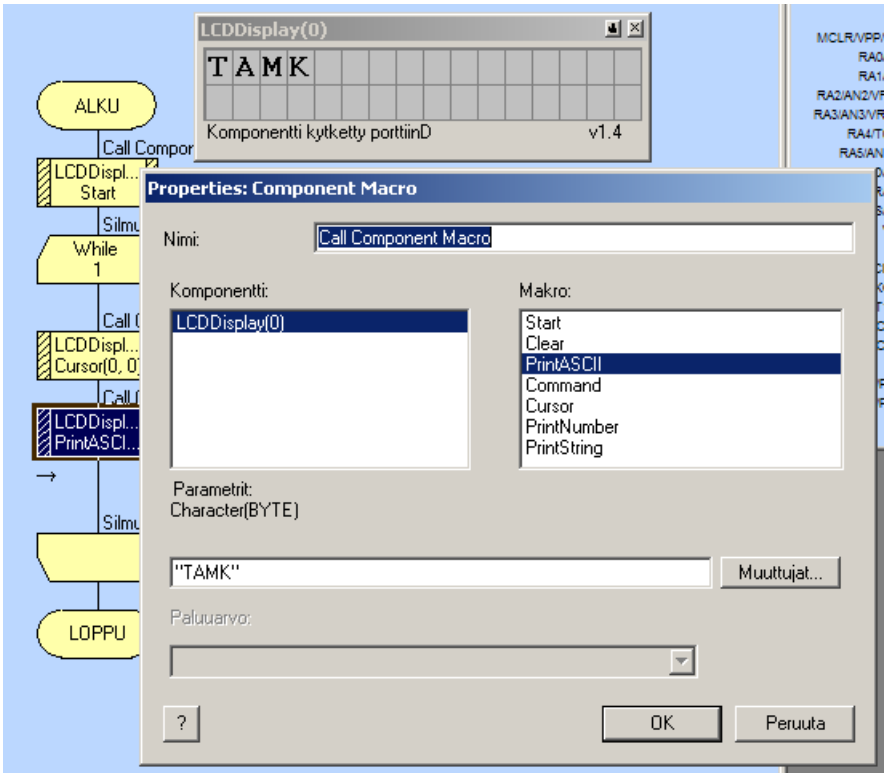
Kytkin-komponentin ReadState-makrolla luetaan parametrin määrittämä yhden nastan tila ja tallennetaan se paluuarvoksi annettuun muuttujaan. Esimerkiksi parametrilla 4 ja paluuarvolla ”nappi” C-porttiin kytketyistä napeista luetaan viides ja sen tila tallennetaan nappi-muuttujaan. WaitUntilHigh-makrolla odotellaan, kunnes parametrin määräämä nasta saa tilan 1. WaitUntilLow-makrolla odotellaan tilaa 0. Kuten Led-makrot, nämäkin vievät muistia turhan paljon. Nastan tila kannattaa siis lukea Tulo-käskyllä ja odotteluun käyttää Silmukka-käskyä.

Simuloinnissa kytkimen sulkeminen antaa nastalle tilan 1 ja avaaminen tilan 0. Led-komponentin tavoin mikrokontrolleriin liitettävät kytkimet voivat toimia päinvastoin kuin simuloinnissa. E-Block-sarjan kytkinkortin kytkentä toimii kuitenkin simuloinnin kanssa samaan suuntaan.

5.5.3 LCD-näyttö

Näyttö-komponentin ominaisuuksista valitaan käytettävän näytön koko, eli montako merkkiä on yhdellä rivillä, ja montako rivejä on. Liitettävä portti valitaan komponenttikytkennöistä. E-Block-näytön koko on 16x2 ja nastajärjestys on normaali, joten kytkettävä portti on ainoa valinnan kohde. Jos käytetään jotain muuta näyttöä, joudutaan varmistamaan kyseisen näytön datalehdeltä nastajärjestys.

Näytön ohjaaminen tapahtuu Komponenttimakro-käskyllä (kuva 46). Ensimmäisenä pitää käynnistää näyttö Start-makrolla, jota ei tarvita ohjelmassa kuin kerran. Näyttö tyhjennetään tekstistä Clear-makrolla. Kummallakaan näistä makroista ei ole parametreja.



Kuva 46. Näytön ohjauksessa käytettävät Komponentti-makrot.

Cursor-makrolla ja sen parametreilla (x,y) määritellään tekstin aloituskohta eli kursorin paikka. Parametrit voidaan antaa numeroina tai muuttujien arvoina. Esimerkiksi ”0,0” on ensimmäinen vasemmalta ja ensimmäinen rivi ylhäältä. ”6,line” on taas kuudes merkki vasemmalta ja rivin määrää line-muuttujan arvo. Kursori siirtyy kirjoituksen jälkeen aina yhden askeleen oikealle, esimerkiksi kirjoitettuaan kuvan nelikirjaimisen tekstin kursori siirtyy viidennen merkin kohdalle. Jos kursoria ei siirrettäisi, se kirjoittaisi uuden tekstin viidennestä merkistä alkaen.

PrintASCII-makrolla voidaan näytölle tulostaa tekstiä ja numeroita suoraan lainausmerkeissä. Parametriksi voidaan myös kirjoittaa tai antaa muuttujasta tavun mittainen luku, jolloin näytölle tulee numeroa vastaava ASCII-merkistön merkki.

PrintString-makrolla pystytään myös syöttämään lainausmerkeissä tekstiä ja numeroita suoraan näytölle. String-muuttujan nimeä käyttämällä näytölle tulostuu muuttujan sisältämä teksti.

PrintNumber-makrolla numerot kirjoitetaan suoraan ilman lainausmerkkejä tai käyttämällä muuttujaa. Parametriin voidaan antaa tekstiä ja numeroita lainausmerkeissä, jolloin näytölle tulostuu perätysten jokaista merkkiä vastaava ASCII-taulukon numero.

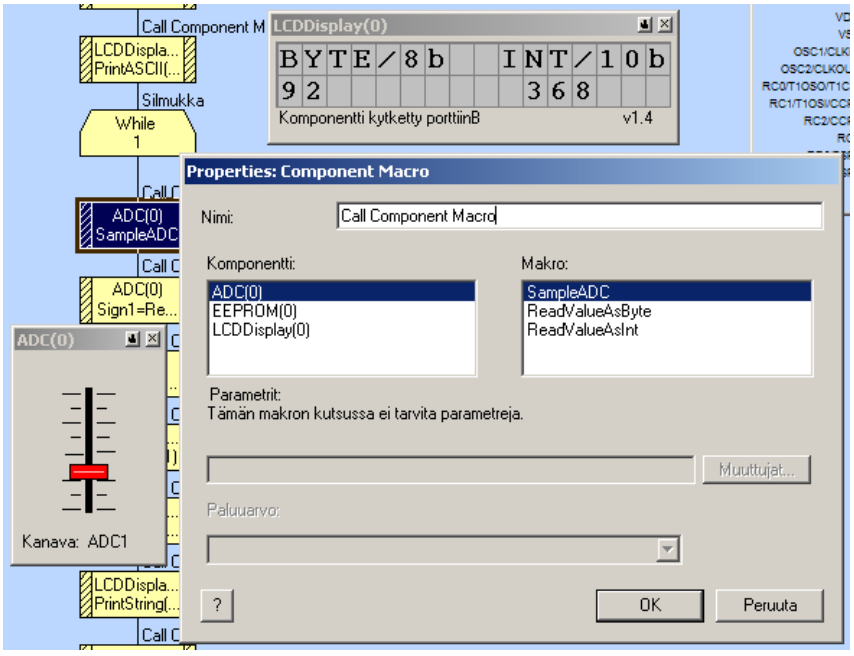
Näytölle kirjoitetaan usein vaihtelevia tekstejä tai lukuja. Tästä voi syntyä pieniä sekaannuksia ilman erityistoimenpiteitä. Esimerkiksi tulostetaan näytölle tavu-muuttujan lukua, joka voi siis vaihdella 0 ja 255 välillä. Näytölle kirjoitetaan ensiksi arvo 199, kursorin palautuksen jälkeen tulostetaan uusi arvo 4. Näytöllä näkyy lukema 499. Ongelma ratkaistaan joko tyhjentämällä koko näyttö ja kirjoittamalla kaikki merkit ja luvut uusiksi tai tulostamalla muutama välilyönti heti vaihtelevan luvun tulostuksen jälkeen. Esimerkin tapauksessa arvon tulostuksen jälkeen kaksi välilyöntiä riittäisi.

5.5.4 Analogitulo

ADC-komponentilla pystytään simuloimaan analogista signaalia, ja sen komponenttimakroilla ohjelmoidaan tarvittava signaalinkäsittely. Ominaisuuksista voidaan antaa komponentille nimi sekä vaihtaa sen väri ja tyyppi.

Komponenttikytkennästä valitaan kytketty nasta. Tässä kohtaa pitää olla tarkkana, esimerkiksi PIC16F877:n viides analogi-nasta (ADC4 tai AN4) on A-portin kuudes nasta (RA5). E-Block-sarjan anturi-kortilla olevan valovastuksen signaali menee kytketyn portin ensimmäiseen nastaan. Kortilla olevan potentiometrin signaali taas menee toiseen nastaan. Jos kortti liitetään A-porttiin, niin komponenttikytkennät ovat ADC0 ja ADC1. E-porttiin liitettynä kytkennät ovat ADC5 ja ADC6.

Komponentti-makroilla otetaan analogikanavasta ensin SampleADC-makrolla näyte, jonka jälkeen se tallennetaan johonkin muuttujaan ReadValueAs-makroilla (kuva 47).



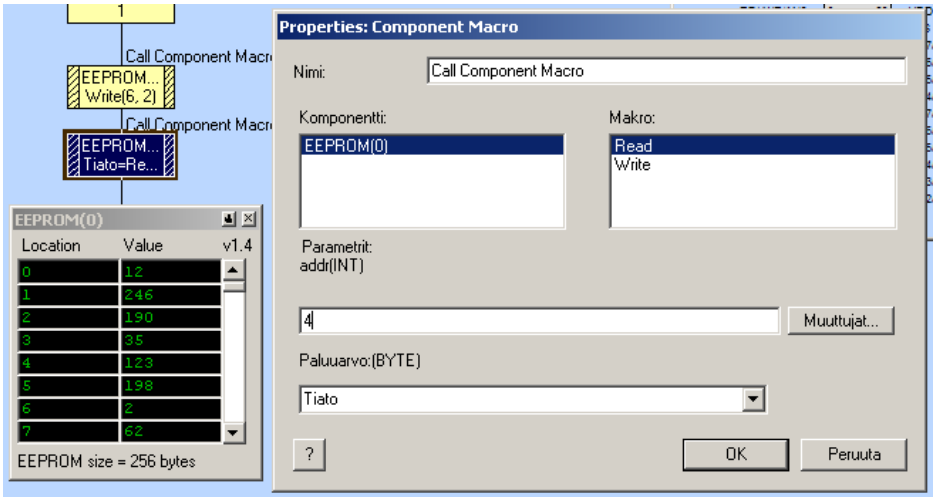
Kuva 47. Analogisignaali luetaan ja tallennetaan muuttujaan 8- tai 10-bittisenä.

Tavu-muuttujaan tallennus antaa 8-bittisen arvon, jolloin lukema on 0 ja 255 välillä. Lukema on 10-bittinen, jos se tallennetaan int-muuttujaan. Ääripäät ovat tällöin 0 ja 1023.

Simuloinnin aikana ADC-komponentilla säädetään analogi-tuloon tulevan signaalin tasoa, käyttöjännitteen ja maan välillä (olkoon se vaikka 0 V ja 5 V). Oikeassa käytössä mittausjännitteen ääripäät tulisi sovittaa siis käyttöjännitteen ja maan väliin.

5.5.5 Tallennus Eeprom-muistiin

Käyttömuistissa olevat muuttujien arvot katoavat, jos mikrokontrolleri sammutetaan. Joitakin sovelluksen muuttujien arvoja voidaan kuitenkin tarvita, kun laitte käynnistetään uudelleen. Nämä tallennetaan Eeprom-muistiin (kuva 48). Liitetyn Eeprom-komponentin makroilla saadaan tämä toiminto käyttöön. Komponentin ominaisuuksista voidaan vaihtaa muistin kokoa ja lukujen näyttötapaa.



Kuva 48. Arvo luetaan Eepromin osoitteesta 4 ja tallennetaan Tiato-muuttujaan.

Haluttu arvo tallennetaan eepromiin Write-makrolla, jonka parametreiksi annetaan muistipaikan osoite sekä tallennettava tieto, joko numeroina tai muuttujan arvona. Tallennetun tiedon saa käyttöön Read-makrolla, jonka parametriksi syötetään haettavan tiedon osoite ja paluuarvoon annetaan muuttujan nimi, jonne tieto tallennetaan käytön ajaksi.

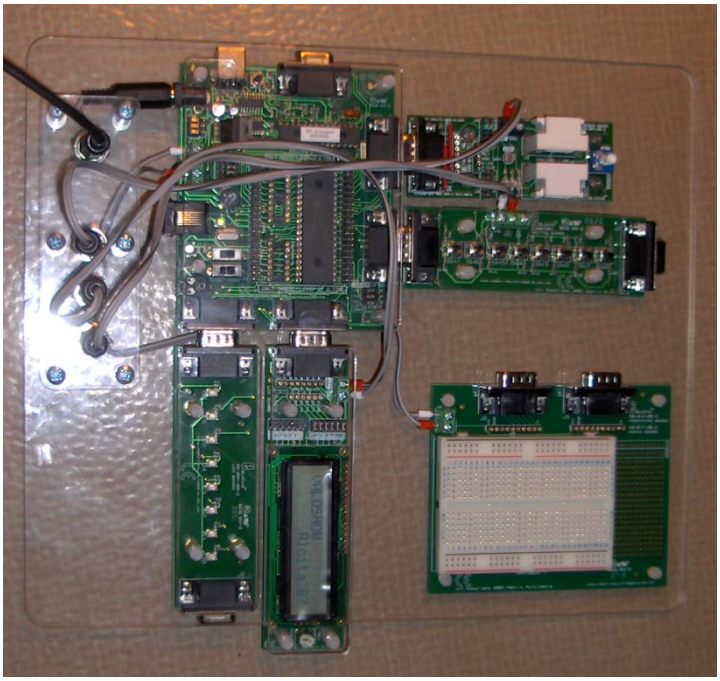
Jokainen muistiosoite pystyy tallentamaan vain tavun mittaisen luvun. Int-muuttujat tarvitsevat siis kaksi osoitetta. Tallennus voidaan suorittaa esimerkiksi siten, että tallennetaan int-muuttuja ensin johonkin osoitteeseen, jolloin ensimmäiset 8-bittiä tallentuvat. Laskutoimituksella siirretään loput 8 bittiä edellisten paikalle, minkä jälkeen sama int-muuttuja tallennetaan toiseen muistipaikkaan.

Int-luku voidaan palauttaa esimerkiksi siten, että bitit 9-16 luetaan muistista haluttuun int-muuttujaan ja siirretään laskutoimituksella bitit paikoilleen. Bitit 1-8 luetaan toiseen muuttujaan ja laskutoimituksella lasketaan muuttujat yhteen.

6 ESIMERKKIOHJELMA

Tässä luvussa esitellään yksinkertainen esimerkkisovellus, jossa käytetään monipuolisesti Flowcoden käskyjä ja E-Block-oheiskortteja sekä niiden Flowcode-komponentteja.

Esimerkissä E-Block-ohjelmointikortin A-porttiin kytketään anturikortti ja B-porttiin kytkinkortti (kuva 49). Näyttökortti liitetään C-porttiin ja ledkortti D-porttiin.

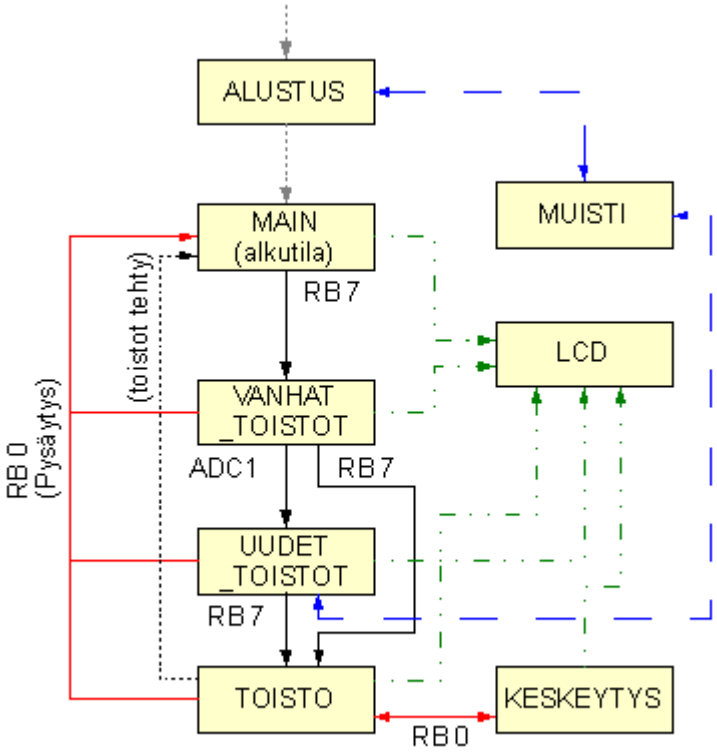


Kuva 49. Esimerkkisovelluksessa käytetyt lisäkortit.

Sovellus toimii siten, että ledkortilla näytetään haluttua valokuviota tai kiertoa, jonka toistokertoja voidaan vaihdella anturikortin potentiometrillä 1 ja 10 välillä. Sovelluksessa käytetään kahta painonappia, kytkinkortin kahdeksannella napilla (RB7) käynnistetään ja edetään ohjelmassa. Keskeytyskytkimenä toimii kortin ensimmäinen nappi (RB0), sillä voidaan pysäyttää toisto ja palata alkutilaan.

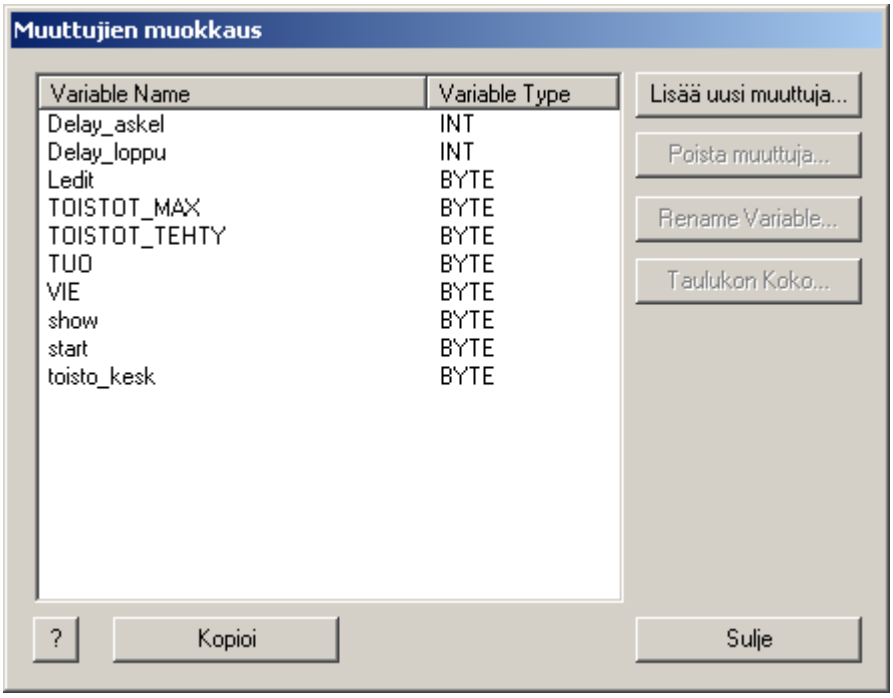
Sovelluksessa käytetään myös lcd-näyttöä, johon tulostetaan sen hetkinen ohjelman tila ohjeineen. Viimeisin säädetty toistokertojen määrä tallennetaan Eeprom-muistiin, josta se haetaan laitetta uudelleen käynnistettäessä.

Makroja käyttämällä saadaan ohjelman rakenne ja toiminta selkeämmäksi (kuva 50). Näyttökorttia sekä muistia käytetään niiden omilla makroillaan. Ohjelman alussa kerran käytetyt toiminnot on sijoitettu Alustus-makroon. Toiston käynnistys, säätö ja suoritus on sijoitettu kolmeen erilliseen makroon. Ensimmäisessä päätetään, kelpaako edellinen toistomäärä vai säädetäänkö uusi. Toisessa makrossa suoritetaan uuden toistomäärän säätö. Kolmannessa suoritetaan itse valokuvion toisto, kunnes toistomäärä on täyttynyt. Keskeytys-makro kutsutaan, kun halutaan pysäyttää toiston suoritus, jonka jälkeen ohjelma palaa alkutilaan.



Kuva 50. Esimerkkiohjelman toimintakaavio.

Esimerkkiohjelman joissakin makroissa käytetään sekä paikallisia että parametrimuuttujia, joiden toiminta selvitetään kyseisen makron yhteydessä. Yleisiä muuttujia taas käytetään useammassa kuin yhdessä makrossa (kuva 51).



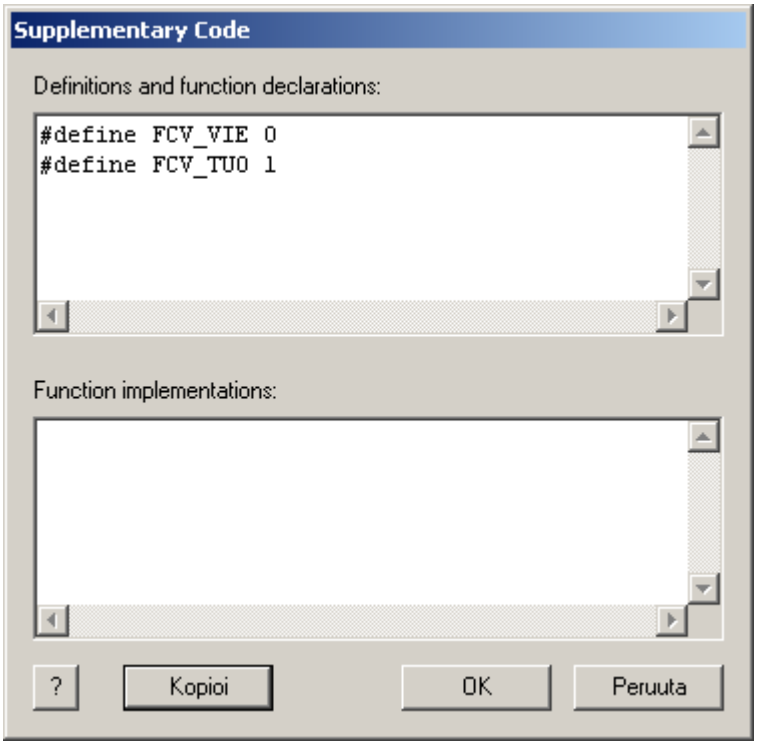
Kuva 51. Esimerkkiohjelman yleiset muuttujat.

Valokuvion toistossa käytetään Delay-muuttujien arvoa viiveiden kestona. Ledit-muuttujan arvolla määritellään, mitkä ledit loistavat. Säädetty toistokertojen määrä tallennetaan Toistot_max-muuttujaan. Toistot_tehty-muuttujaan lasketaan toiston aikana kertyneet toistot.

Show- ja start-muuttujia käytetään päävuokaaviossa ohjaamaan toiston käynnistämistä.

Toisto_kesk-muuttujalla merkataan ne ohjelman tilat, joissa keskeytyksen halutaan olevan voimassa.

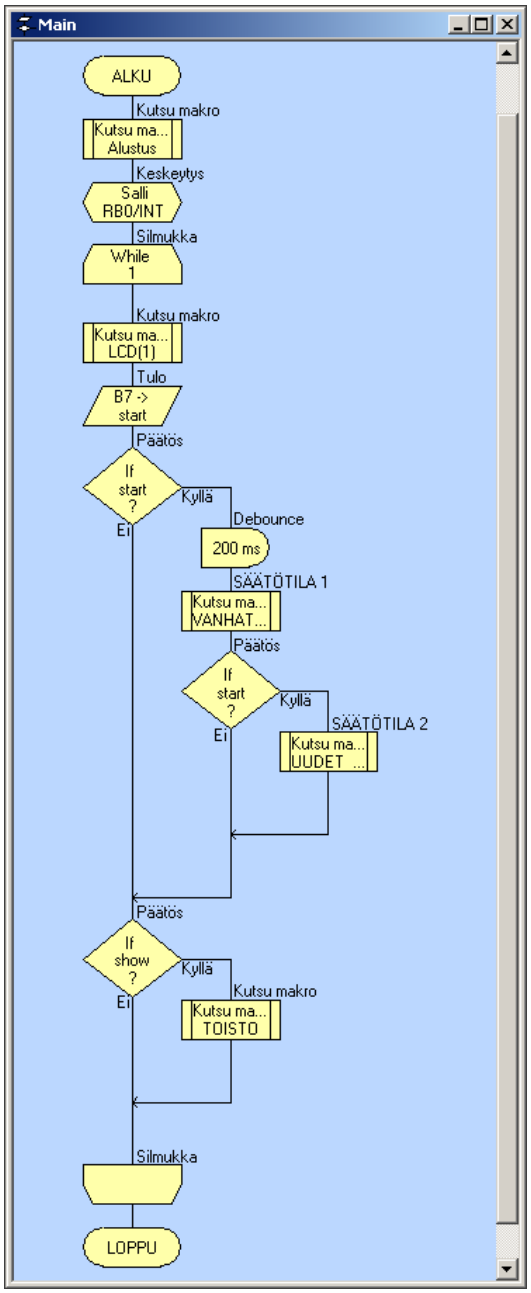
Tuo- ja Vie-muuttujia käytetään Muisti-makron kutsussa arvon sijasta antamaan kutsutun makron parametrille arvo. Nämä muuttujat määritellään vakiomuuttujiksi. Tuo-muuttujalle annetaan arvo 1 ja Vie-muuttujalle arvo 0. Määrittely tehdään Muokkaa-valikon Supplementary Code -kohdasta (kuva 52).



Kuva 52. Vakiomuuttujat esimerkkiohjelmassa.

6.1 Main-vuokaavio

Pääohjelman alussa kutsutaan Alustus-makro (kuva 53). Keskeytys-käskyllä sallitaan RB0-keskeytys, joka nappia painettaessa kutsuu Keskeytystila-makron. Tämän jälkeen ohjelma on alkutilassa ikuisessa silmukassa.



Kuva 53. Päävuokaavio.

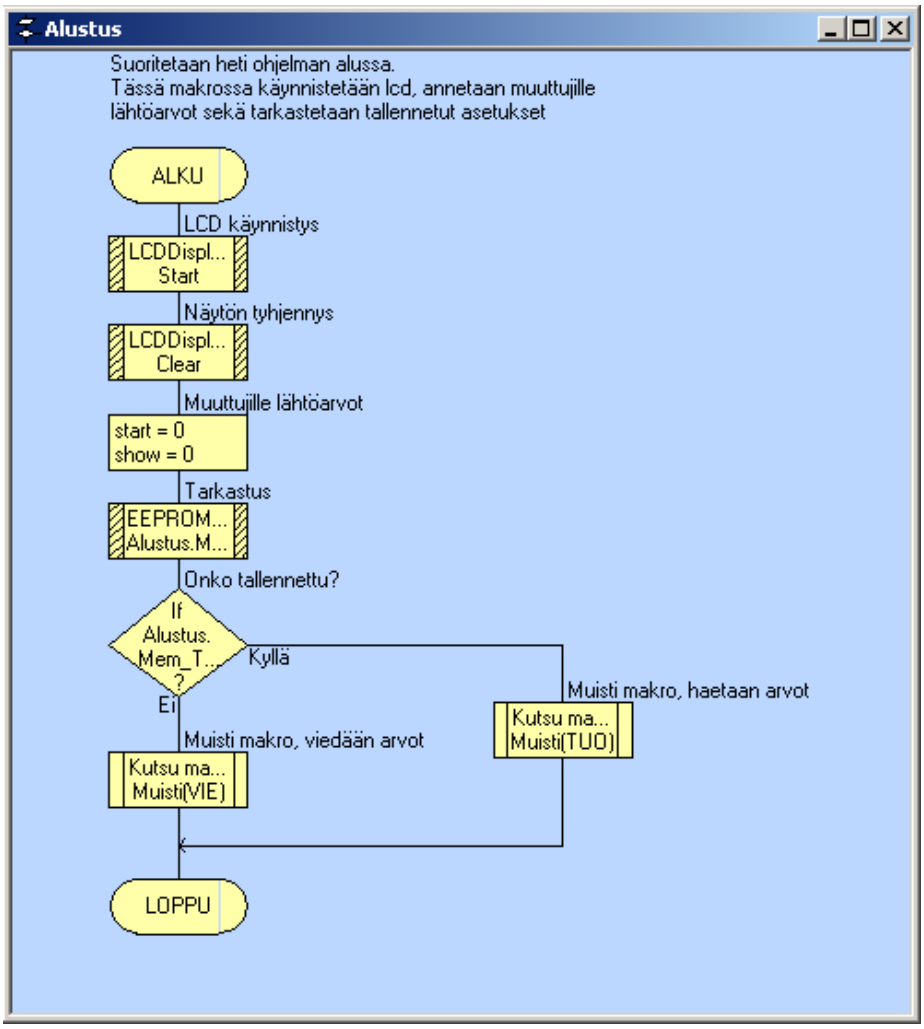
Alkutilassa ohjelma kutsuu LCD-makron, jonka parametrille annetaan arvo 1 (5.3 LCD-makro).

Tämän jälkeen ohjelma lukee käynnistysnapin tilaa ja tallentaa sen start-muuttujaan. Jos nappia painetaan, kutsutaan Vanhat_toistot-makro. Koska kutsutussa makrossa luetaan uusiksi painettu nappi, on ohjelmaan lisätty viive kytkinvärähtelyn takia.

Vanhat_toistot-makrosta poistuttaessa annetaan start- ja show-muuttujille arvot (5.4 Vanhat_toistot-makro). Jos start-muuttuja on 1, kutsutaan Uudet_toistot-makro. Kun säätö-makrot on suoritettu ja show-muuttujalla on arvo 1, valokuvion toisto käynnistetään kutsumalla Toisto-makro.

6.2 Alustuksen ja muistin makrot

LCD-näytön käynnistys ja tyhjennys suoritetaan komponenttimakroilla (kuva 54). Start, show ja toisto_kesk-muuttujille annetaan laskutoimituksella lähtöarvoiksi 0 ja Toistot_max-muuttujalle annetaan 2.

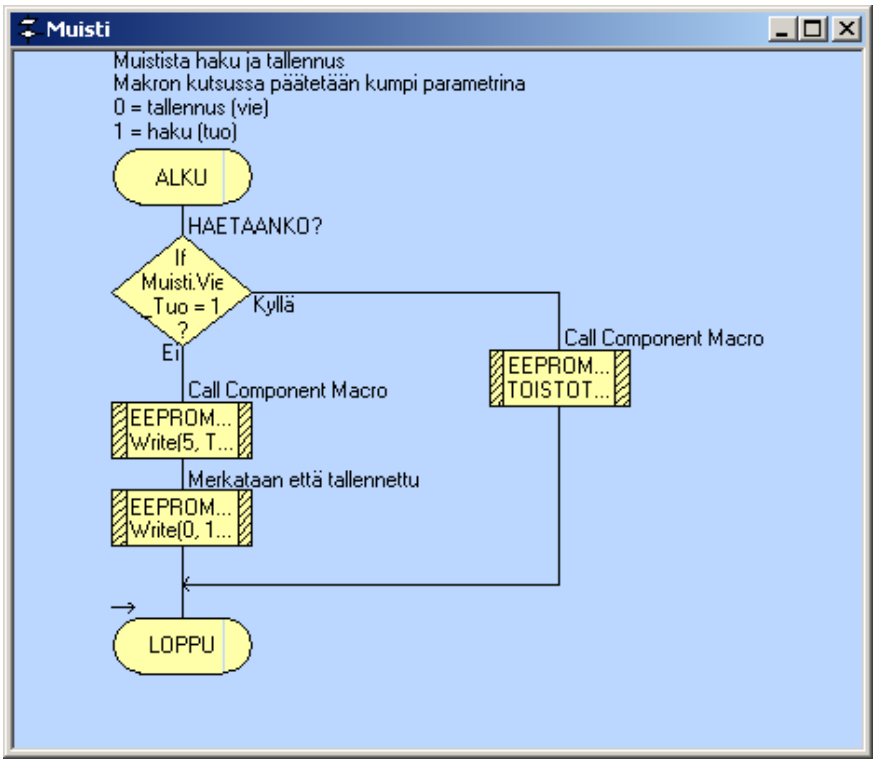


Kuva 54. Alustus-makron vuokaavio.

Kun Eeprom-muistiin tallennetaan toistomäärä, tallennetaan samalla toiseen muistipaikkaan jokin luku. Toinen tallennettu luku kertoo ohjelmalle, onko tallennusta koskaan tehty. Jos tarkistusta ei tehdä, voi ensimmäisellä

käynnistyksellä toistomäärä saada esimerkiksi arvon 255 rajatun maksimin ollessa kuitenkin vain 10. Tarkistukseen tässä esimerkissä on käytetty osoitetta 0 ja tarkistuslukua 135. Alustus-makroon luodaan tavun kokoinen paikallinen muuttuja Mem_Test, johon haetaan komponenttimakrolla arvo tarkistusosoitteesta. Jos haettu arvo on yhtä kuin 135, haetaan tallennettu toistomäärä kutsumalla Muisti-makro, jonka parametriksi annetaan muuttuja TUO. Jos Päätös-käskyn ehto ei täyty, niin laskutoimituksessa annettu arvo 2 tallennetaan muistiin kutsumalla Muisti-makro arvolla VIE.

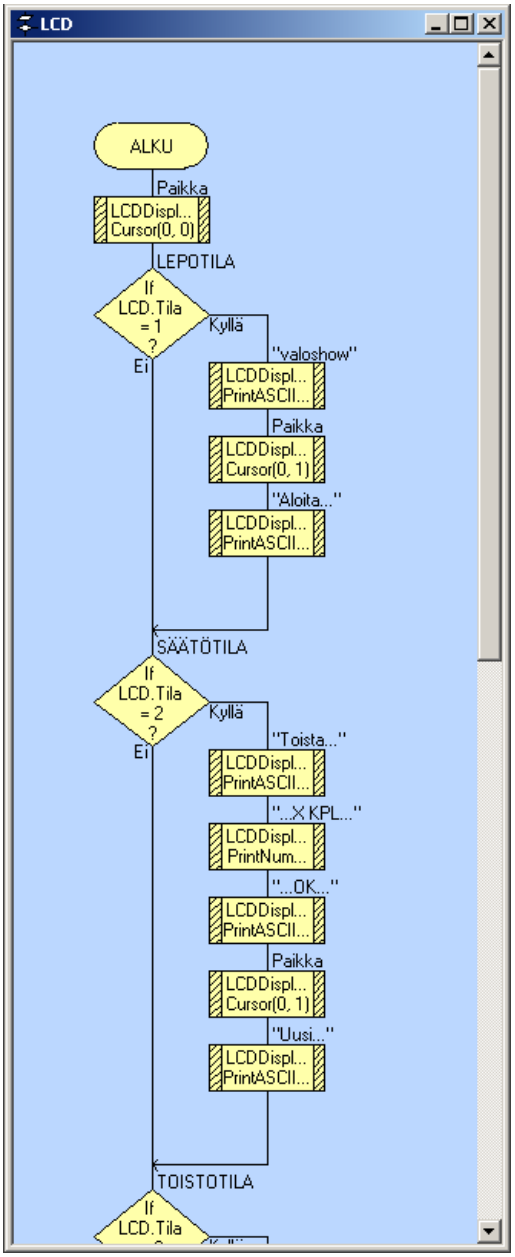
Muisti-makroon luodaan parametrimuuttuja Vie_Tuo, joka saa kutsussa arvon 0 tai 1. Kun Päätös-käskyllä tarkastettavan parametrin arvo on 0, kirjoitetaan Eeprom-muistin osoitteeseen 5 Toistot_max-muuttujan arvo ja osoitteeseen 0 tarkistusluku 135 (kuva 55). Jos parametrin arvo on 1, haetaan muistista tallennettu arvo Toistot_max-muuttujaan.



Kuva 55. Muisti-makrolla joko viedään tai tuodaan arvo.

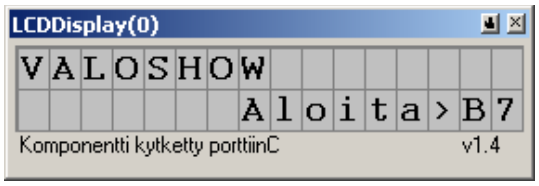
6.3 Näytön ohjaaminen LCD-makrolla

Esimerkkiohjelmassa näytöllä näytetään neljä eri tilaa. LCD-makroon luodaan Tila-parametri, jolle annetaan makroa kutsuttaessa arvo 1...4. Päätös-käskyjen ehdoilla ohjataan näytön kirjoitus haluttua reittiä (kuva 56).



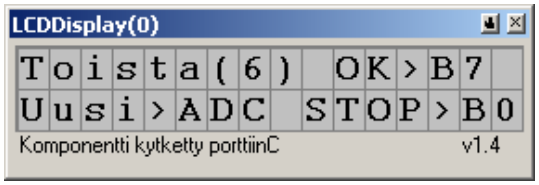
Kuva 56. LCD-makron ensimmäiset kaksi tilaa.

Kun LCD-makro kutsutaan parametrin arvolla 1, saa LCD.Tila-muuttujakin saman arvon. Tällöin näytölle kirjoitetaan lepotilan tekstit (kuva 57).



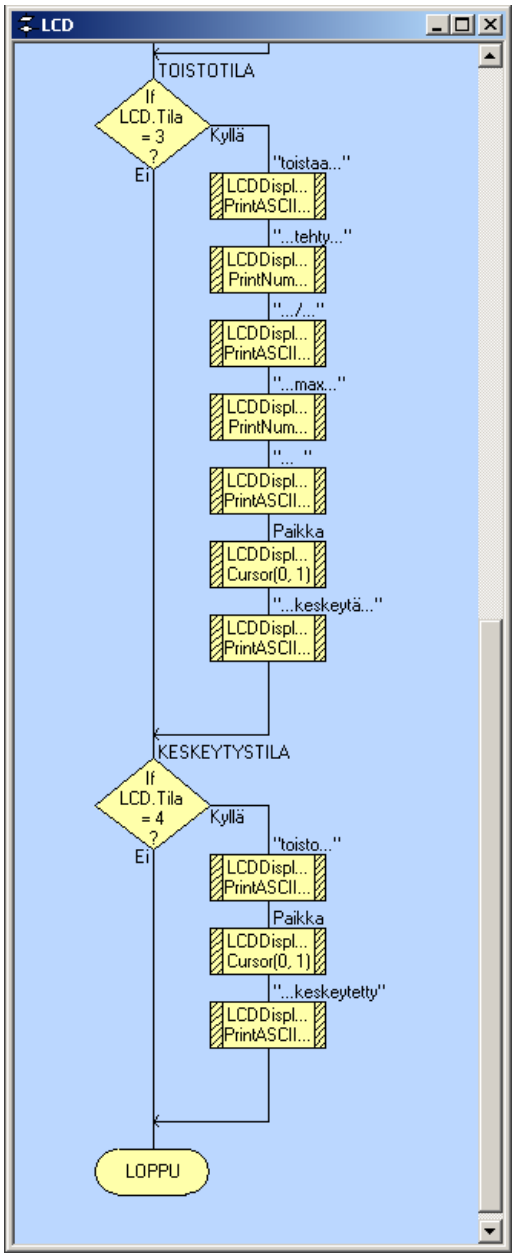
Kuva 57. Esimerkin alkutilan näyttö.

Parametrilla 2 näytölle tulostetaan myös Toistot_max-muuttujan arvo (kuva 58), joka uutta arvoa säädettäessä vaihtelee.



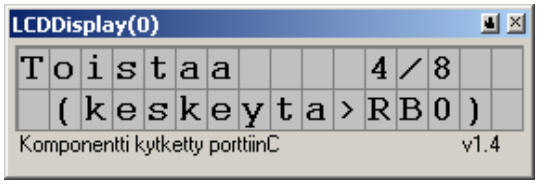
Kuva 58. Säätötilan näyttö.

LCD-makron viimeiset kaksi tilaa koskevat toistoa ja sen pysäytystä (kuva 59).



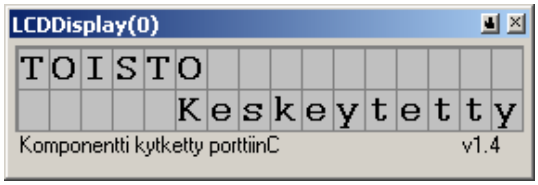
Kuva 59. Makron kaksi viimeistä tilaa.

Kolmatta näytön tilaa käytetään toiston aikana (kuva 60). Näytölle tulostetaan tekstien lisäksi Toistot_tehty- ja Toistot_max-muuttujien arvot. Ennen rivinvaihtoa on lukujen perään tulostettu vielä muutama välilyönti.



Kuva 60. Toiston seuranta näytöltä.

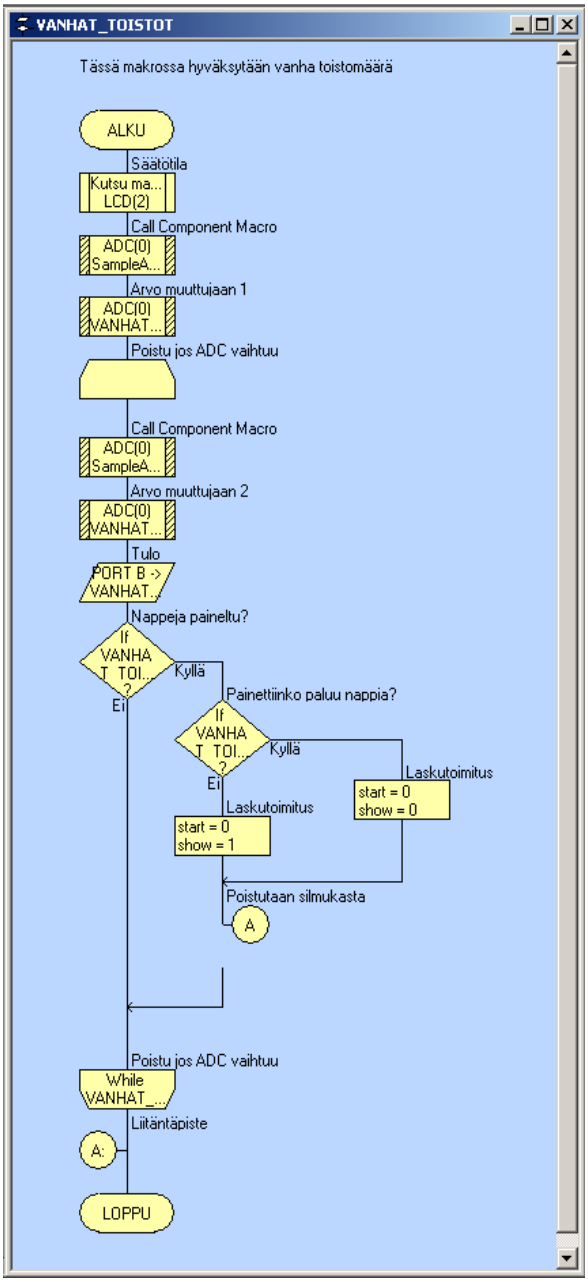
Näytön neljättä tilaa näytetään hetken aikaa, jos toisto keskeytetään napista ennen aikojaan (kuva 61).



Kuva 61. Näytön teksti kun toisto on keskeytetty.

6.4 Toistoa edeltävät säätö-makrot

Kun päävuokaavion alkutilassa painetaan käynnistysnappia, kutsutaan ensiksi Vanhat_toistot-makro (kuva 62). Tässä makrossa käytetään kolmea paikallista muuttujaa. LCD-makron kutsun jälkeen luetaan ensimmäiseen paikalliseen, Adc_1-muuttujaan, sen hetkinen potentiometrin tila. Silmukassa luetaan uusiksi potentiometrin tila, joka annetaan Adc_2-muuttujaan. Silmukan ehtona on, että Adc_1 ja Adc_2-muuttujat ovat yhtä suuret. Jos käyttäjä haluaa muuttaa toistomäärää ja potentiometrin asentoa vaihdetaan, poistutaan silmukasta ja makrosta. Tässä tilanteessa start-muuttujan arvo on edelleen 1 ja päävuokaaviosta kutsutaan Uudet_toistot-makro.

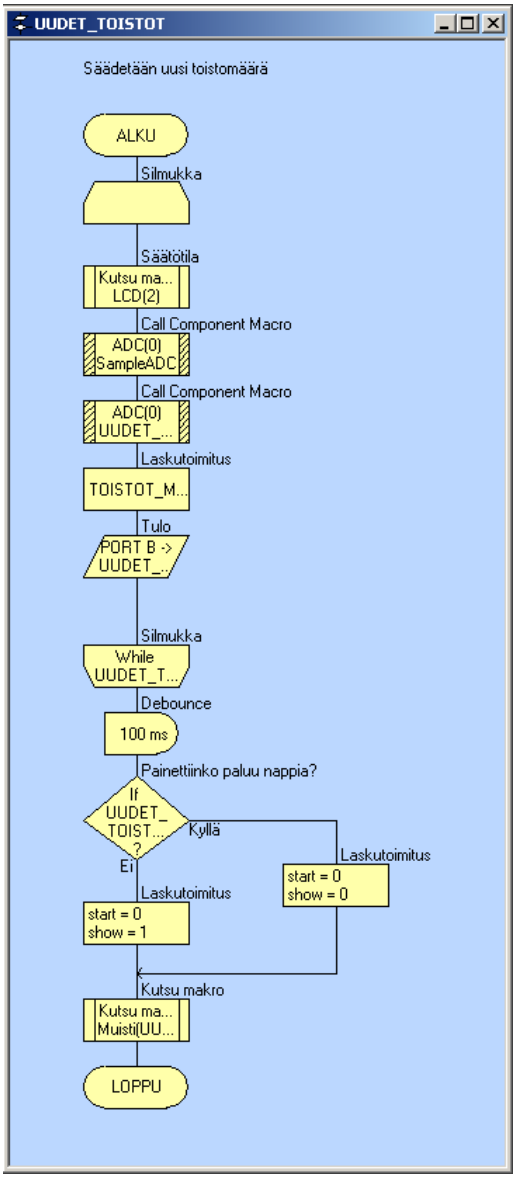


Kuva 62. Vanhat_toistot-makro.

Silmukassa luetaan myös maskia käyttäen B-portin bittejä 0 ja 7. Saatu arvo tallennetaan kolmanteen paikalliseen, napit-muuttujaan. Paluu-nappi antaa muuttujalle arvon 1, jolloin kummankin päätöksen ehto täyttyy. Ensimmäisen ehto on

nollasta poikkeavuus, eli jotain on painettu. Toisessa päätöksessä ehto toteutuu, jos nappi-muuttuja on yhtä kuin 1. Paluu-napista start- ja show-muuttujat nollataan, jolloin hyppy-käskyn ja makron sulkemisen jälkeen ohjelma palaa alkutilaan. Hyväksy-napista start-muuttuja saa arvon 0 ja show-muuttuja arvon 1, jolloin päävuokaaviosta kutsutaan Toisto-makro.

Uudet_toistot-makroon luodaan napit_u, Adc_lukema ja Muisti-muuttujat (kuva 63). Silmukassa luetaan potentiometrin tilaa, joka tallennetaan tavun kokoisena Adc-lukema-muuttujaan. Laskutoimituksella muutetaan lukema (0...255) toistokerroiksi (1...10), joka sijoitetaan Toistot_max-muuttujaan. Lcd-näyttöä päivittämällä nähdään laskettu lukema.

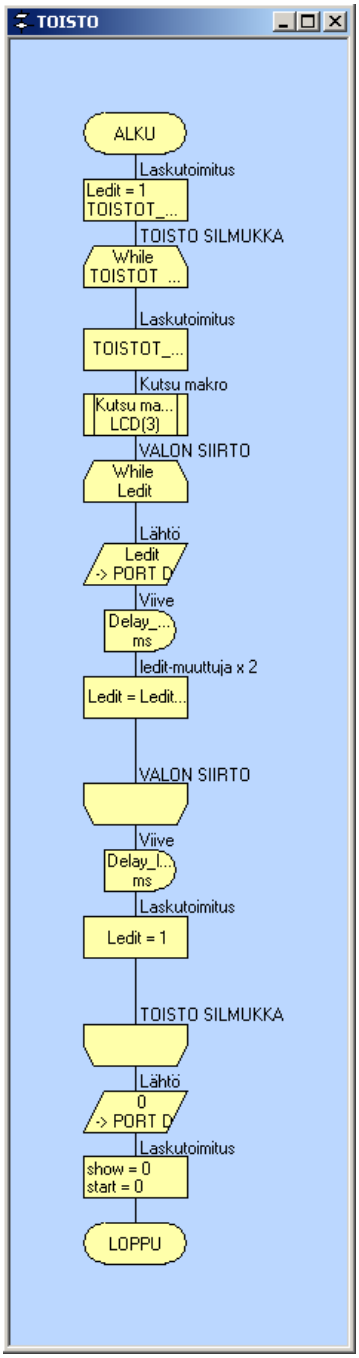


Kuva 63. Uudet_toistot-makro.

Silmukan ehto on ”napit_u-muuttuja on 0”. Maskia käyttäen luetaan peruuta- ja hyväksy-nappeja. Silmukan jälkeisen päätöksen ehto on ”napit_u on 1”. Jos poistumiseen on käytetty paluu-nappia, niin ehto täyttyy. Tällöin nollataan start- ja show-muuttujat sekä annetaan paikalliselle Muisti-muuttujalle arvoksi TUO. Hyväksy-napista show saa arvon 1 ja Muisti-muuttuja arvon VIE. Tämän jälkeen kutsutaan Muisti-makro tällä paikallisen Muisti-muuttujan arvolla.

6.5 Toisto ja toiston keskeytys

Pääohjelmaan paluun jälkeen kutsutaan Toisto-makro, jos show-muuttuja on 1 (kuva 64). Toisto-makron alussa annetaan laskutoimituksella Ledit ja toisto_kesk-muuttujille arvo 1 sekä nollataan Toistot_Tehty-muuttuja. Delay_-muuttujille annetaan myös halutut arvot millisekunteina. Delay_askel-muuttujalle annetaan arvo 150 ja Delay_loppu-muuttujalle arvo 600.



Kuva 64. Toisto-makro.

Uloimmassa Toisto Silmukassa lasketaan toisto-kertoja, päivitetään näyttöä ja alustetaan Ledit-muuttuja toistokertojen välissä. Silmukan ehtona on ”Toistot_Max on isompi kuin Toistot_tehty”.

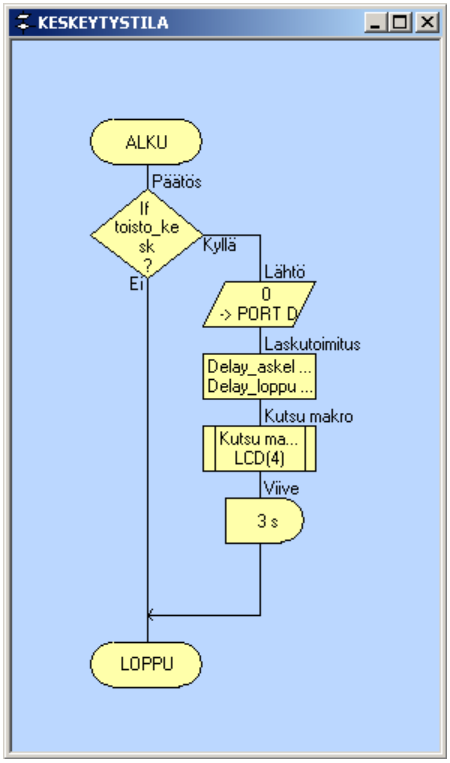
Valokuvio suoritetaan sisemmässä Valon Siirto silmukassa, jossa askelviiveen jälkeen Ledit-muuttuja kerrotaan kahdella. Silmukasta poistutaan kun laskutoimituksen tulos ylittää tavun koon, jolloin muuttuja saa arvon 0. Tämän jälkeen on loppuviive, jonka jälkeen valokuvio alkaa alusta jos toistokerrat eivät ole täynnä.

Toiston lopussa sammutetaan ledit ja nollataan show, start ja toisto_kesk-muuttujat, minkä jälkeen ohjelma palaa alkutilaan.

Toisto voidaan pysäyttää ennen aikojaan keskeytys-

napista. Tällä kertaa peruutus hoidetaan keskeytys-rutiinia käyttäen. Ohjelman alussa sallittiin RB0-keskeytys, joka kutsuu Keskeytystila-makron (kuva 65).

Keskeytys-rutiinia ei estetä missään kohtaa ohjelmaa, mistä seuraa se että aina kun ensimmäistä nappia painetaan, kutsutaan myös Keskeytystila-makro. Makron alussa olevan päätöksen ehtona on toisto_kesk-muuttuja. Tämän muuttujan määrittelyllä pystytään rajaamaan ohjelman alueet, jolloin keskeytys on voimassa. Esimerkkiohjelmassa tämä tehdään Toisto-makron alussa ja lopussa.



Kuva 65. Keskeytystilan makro.

Toiston aikainen keskeytys sammuttaa ledit, nollaa Ledit ja Delay_-muuttujat sekä antaa Toistot_tehty-muuttujalle Toistot_max-muuttujan lukeman. Näytölle tulostetaan keskeytysilmoitus kolmen sekunnin ajaksi.

Keskeytystila-makrosta poistumisen jälkeen Toisto-makron käskyt suoritetaan uusilla arvoilla, joten ohjelma palaa alkutilaan periaatteessa välittämästi.

7 JOHTOPÄÄTÖKSET

Opinnäytetyöhön kuului opiskelulaitteiston järjestely. Laitteiston säilytys ja käyttö pyrittiin saamaan helpoksi. Tässä mielestäni onnistuttiin. Opinnäytetyön pääpaino oli ohjelmointiohjeiden laatimisessa. Oppaassa on pyritty kertomaan eri toiminnoista siten, että lukija ymmärtäisi miksi jokin toimii sillain kuin se toimii. Tämän osalta onnistumisen voi kertoa vain tätä opasta käyttävä opiskelija.

Oppaaseen kuuluu myös esimerkkiohjelma, jonka tarkoitus on näyttää esiteltyjen käskyjen ja komponenttien toimintaa keskellä suurempaa ohjelmaa. Esimerkin esittely on yritetty tehdä sellaiseksi, että tehdessään esimerkin ohjelmaa lukija joutuisi myös ajattelemaan. Kaikki tarvittava tieto löytyy kuitenkin käskyjen ja komponenttien kuvauksista.

Jos koulu ottaa käyttöön uusia E-Block-sarjan oheiskortteja, niin niihin liittyvät ohjeistukset voidaan lisätä omina lukuinaan tämän oppaan sisälle. Opas tulisi myös päivittää, jos koulu alkaa käyttämään Flowcode-ohjelmiston uusinta versiota.

LYHENTEET JA MERKIT

A/D	Analogi/Digitaali
ALU	Arithmetic logic unit, aritmeettislooginen yksikkö
ASCII	American standard code for information interchange
C	Capacitor, kondensaattori
EEPROM	Electronically erasable programmable read-only memory
F	Faradi, kapasitanssin yksikkö
Hz	Hertsi, värähtelytaajuuden yksikkö
MIPS	Million instructions per second, miljoona ohjetta sekunnissa
PWM	Pulse width modulation, pulssinleveysmodulaatio
R	Resistor, vastus
RAM	Random access memory
USB	Universal serial bus
V	Voltti, jännitteen yksikkö
τ	Tau, aikavakio
Ω	Ohmi, vastuksen yksikkö

LÄHDELUETTELO

1 Microchip Technology Inc. [www-sivu]. [viitattu 25.2.2010.] Saatavissa: http://www.microchip.com/

2 Getting Started With Flowcode. [sähköinen dokumentti.] Matrix Multimedia. [viitattu 25.2.2010.] Saatavissa: http://matrixmultimedia.com/lc_microcontroller.php

3 PIC16F87XA datalehti. [sähköinen dokumentti.] Microchip Technology Inc. [viitattu 25.2.2010.] Saatavissa: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010242

4 E-block programmer datalehti. [sähköinen dokumentti.] Matrix Multimedia. [viitattu 25.2.2010.] Saatavissa: http://matrixmultimedia.com/level5.php?CAT=Browse%20All%20Products

5 Wikipedia. [www-sivu]. [viitattu 25.2.2010.] Saatavissa: http://fi.wikipedia.org/wiki/Assembly_(ohjelmointikieli)