TEKNIIKKA JA LIIKENNE

Tietotekniikka

Tietoverkot

INSINÖÖRITYÖ

FLOWCODE JA E-BLOCKS OPETUSTYÖKALUINA

Työn tekijä: Sakari Paasonen
Työn ohjaajat: Janne Salonen

Työ hyväksytty: 11. 5. 2010

Janne Salonen
Yliopettaja

ALKULAUSE

Tämä insinöörityö tehtiin Metropolia Ammattikorkeakoululle. Työn valvojana toimi yliopettaja Janne Salonen. Kiitän häntä ohjauksesta sekä kaikesta yhteistyöstä niin lopputyöhön kuin muihinkin opintoihin liittyen.

Haluaisin myös kiittää perhettäni tuesta ja kannustuksesta opinnäytetyötä tehdessäni sekä koko opiskeluaikanani.

Helsingissä 11.5.2010

Sakari Paasonen

TIIVISTELMÄ

Työn tekijä: Sakari Paasonen

Työn nimi: Flowcode ja E-Blocks opetustyökaluina

Päivämäärä: 11.5.2010 **Sivumäärä:** 27 s. + 1 liitettä

Koulutusohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Tietoverkot

Työn valvoja: Yliopettaja Janne Salonen

Työn ohjaaja: Yliopettaja Janne Salonen

Työn tarkoituksena on testata 50 tunnin pituinen ”An introduction to microcontroller programming” kurssi, johon käytettiin E-Blocks-laitteistoa sekä Flowcode v3-ohjelmistoa. Työn tavoitteena oli suorittaa kurssi ja tutkia kuinka laitteet ja ohjelmisto toimivat käytännössä.

Työn teoriaosiossa esitellään E-Blocks-laitteisto ja Flowcode v3-ohjelmisto sekä käydään läpi niiden käyttötarkoituksia ja ominaisuuksia. Käytännönosiossa sovelletaan kurssilla opittuja asioita ja toteutetaan pienimuotoinen Flowcode v3:lla tehty sovellus, joka toimii E-Blocks-laitteistossa tarvittaessa.

Työn lopputuloksena saatiin tutkielma E-Blocks-laitteesta ja Flowcode v3-ohjelmistosta opetustyökaluina.

Avainsanat: Flowcode, E-Blocks, Mikrokontrolleri, Ohjelmointi

Metropolia

ABSTRACT

Name: Sakari Paasonen

Title: Flowcode and E-Blocks as teaching tools

Date: 11.5.2010 **Number of pages:** 27 + 1

Department: **Study Programme:**

Information Technology Data Networks

Supervisor: Janne Salonen, Principal lecturer

Instructor: Janne Salonen, Principal lecturer

The purpose of this graduate study was to test 50 hours long course called “An introduction to microcontroller programming” which uses E-Blocks hardware and Flowcode v3 software as its basis. The main objective of this project was to complete the course and examine how the hardware and software work in action.

In the theoretical part of this study E-Blocks hardware and Flowcode v3 software are introduced in detail. Also the main operational principles and possibilities are explored. In the practical part I will put to use what I’ve learned during the course and I will create a small application with Flowcode v3 that works in E-Blocks.

As a result, I created a study about E-Blocks hardware and Flowcode v3 software as teaching tools.

Keywords: Flowcode, E-Blocks, Microcontroller, Coding

SISÄLLYS

1 JOHDANTO

Matrix Multimedian E-Blocks ja Flowcode ovat mainioita opetustyökaluja mikrokontrolleriohjelmoinnin opettamiseen. Sen tärkein ero muihin mikrokontrolleriohjelmointitapoihin verrattuna on ehdottomasti sen helppokäyttöisyys.

Tässä työssä tutkitaan E-Blocks-laitteistoa sekä Flowcode-ohjelmistoa niin teoriassa kuin käytännössäkin. Työn tavoitteena on luoda tutkielma edellä mainituista laitteista ja ohjelmista, sekä testata laitteiston mukana tuleva kurssi.

Työn alussa esitellään Matrix Multimediayritys yleisesti sekä sen tuotteet Flowcode ja E-Blocks pikaisesti. Sen lisäksi tarkastellaan E-Blocksin ja Flowcoden ominaisuuksia ja vahvuuksia tarkemmin.

Kolmannessa luvussa esitellään työssä käytettävä E-Blocks-laitteisto tarkemmin, johon kuuluu käyttötarkoitusten tutkiminen, sekä laitteen ominaisuuksien esittely.

Neljännessä luvussa esitellään Flowcode-ohjelmisto, jossa selvitetään tarkemmin mitä ohjelmalla tehdään ja mihin sitä voidaan käyttää, sekä Flowcoden moninaiset ominaisuudet esitellään.

Viidennessä luvussa käydään läpi vähän teoriaa PICmicro-mikrokontrollereista, kuten mihin niitä käytetään oikeassa elämässä ja mistä jokapäiväisistä laitteista ja koneista niitä löytyy.

Kuudennessa luvussa esitellään mikrokontrollerit yleisesti sekä käsitellään hieman tarkemmin, miten mikrokontrollerit toimivat.

Seitsemäs luku on käytännön osio, jossa asennetaan E-Blocks-laitteet ja Flowcode-ohjelmisto sekä kommentoidaan työn eri vaiheita. Tähän osioon kuuluu myös Flowcodella luotava mikrokontrolleriohjelma, joka ladataan E-Blocks-laitteessa olevaan mikrokontrolleriin.

Kahdeksannessa ja samalla viimeisessä luvussa pohditaan työn edistymistä ja onnistumista sekä arvioidaan kurssi kokonaisuutena.

2 MATRIX MULTIMEDIA

Matrix Multimedia on johtava tekniikan alan yritys, jonka päämaja on Halifaxissa, Englannissa. Se on kehittänyt vuodesta 1993 asti tuotteita, joita käytetään opetuksessa, teollisuudessa ja kotona. Matrix Multimedian tunnetuimmat ja suosituimmat tuotteet ovat.

Flowcode, joka on erittäin kehittynyt graafinen ohjelmointikieli mikrokontrollereille. Se tarjoaa mahdollisuuden kehittää monipuolisia elektronisia järjestelmiä henkilöille, joilla ei ole vielä paljoa kokemusta ohjelmoinnista tai mikrokontrollereista.

E-Blocksit ovat pieniä piirilevyjä, joista jokainen sisältää lohkon elektroniikkaa joita voi normaalisti löytää elektronisista tai sulautetuista järjestelmistä.

Vuonna 2008, Matrix loi Loctronics-nimisen tuotevalikoiman, joka sisältää tuotteita, jotka yksinkertaistavat oppimis- ja opetus prosessin elektroniikan ja sähköopin alalla. Tuotevalikoimasta löytyy sovelluksia elektroniikan-, tieteen-, ja tekniikan aloille. Locktronicilla on asiakkaana useita oppilaitoksia peruskouluista yliopistoihin, ja sitä käytetään yli 10 000 koulussa maailmanlaajuisesti. [1.]

3 E-BLOCKS

E-Blocksissa on yli 40 erilaista piirilevyä yksinkertaisista LED-levyistä monimutkaisempiin laiteohjelmoijiin sekä Bluetooth- ja TCP/IP-piirilevyihin. Sen etuina ovat nopea asennus ja helppokäyttöisyys. Se myös mahdollistaa laajojen elektronisien ja sulautettujen järjestelmien pystyttämisen.

E-Blocksit voidaan liittää toisiinsa jolloin voidaan luoda suuri valikoima erilaisia järjestelmiä, joita voidaan käyttää elektroniikan opettamiseen ja oppimiseen.

E-Blocksin tärkeimpiä ominaisuuksia ovat

- PICmicro tuki, AVR, ARM ja Altera CPLD & FPGA laitteet
- Erinomainen kestävyys
- Tukee laajaa valikoimaa kommunikaatioyhteyksiä mukaan lukien SPI, RS232, I2C, TCP/IP, USB ja niin edelleen. [2.]

Kuva 1. E-Blocks-piirilevyjä [1]

4 FLOWCODE

Flowcode on yksi edistyneimmistä graafisista ohjelmointikielistä. Sen kaksi tärkeintä ominaisuutta ovat drag and drop, joka mahdollistaa selkeän ja yksinkertaisen ohjelmien luomisen, sekä automaattinen ohjelman kääntäjä, joka kääntää vuokaavio ohjelman C:ksi ja sitten vielä HEX-koodiksi. Flowcode-ohjelmistolla voi simuloida E-Blocks-laitteita ja periaatteessa kaikki ohjelmat, joita käyttäjä luo Flowcodella, voidaan testata ennen E-Blocksiin siirtämistä. Valmiit ohjelmat on helppo siirtää E-Blocksiin muutamalla hiiren painalluksella.

4.1 Ominaisuudet

Flowcoden käyttö on erittäin helppoa ja yksinkertaista ja ilman mitään opetusta ohjelmalla pystyy luomaan yksinkertaisia ohjelmia. Sen käyttöliittymää on helppo käyttää eikä ohjelman käyttö vaadi minkäänlaisia ohjelmointi taitoja. Ohjelma kuitenkin mahdollistaa luodun vuokaavio-ohjelman tarkistamisen tai muokkaamisen C-muodossa, jos käyttäjä niin haluaa.

Flowcodessa on useita korkean tason aliohjelmia jotka nopeuttavat järjestelmien kehitystä. [3, s. 2.]

4.1.1 Simulointi

Flowcodella voi suoraan simuloida samaa E-Blocks-alustaa, joka koululla on käytössä. Luotuasi ohjelman flowcodella voit simuloida sitä suoraan ilman mitään lisä säätelyjä. Menu kontrollit mahdollistavat jokaisen ohjelmaikonin tarkastamisen ja simuloinnin yksitellen ohjelmassa ja vaikutuksen näkee suoraan.

Simulointi on muutenkin erittäin tärkeä osa flowcodea ja mikrokontrolleriohjelmointia, koska tämä mahdollistaa lyhyemmän suunnittelujakson sekä mahdollisten ongelmien havaitsemisen ennen mikrokontrollerilaitteelle lataamista. Simulointi myös avustaa ymmärtämään ja oppimaan mikrokontrolleriohjelmointia järkevällä tavalla. [3, s. 3.]

4.1.2 Lataus

Kun ollaan tyytyväisiä suunniteltuun ohjelmaan, voidaan lähettää ohjelma yhdellä napinpainalluksella suoraan mikrokontrollerilaitteeseen. Flowcode tuottaa normaalia HEX-koodia AVR- ja PICmicro-mikrokontrollereihin ja on suoraan yhteensopiva E-blocksin laajan modulaaristen elektroniikka moduulien kanssa. [3, s. 3.]

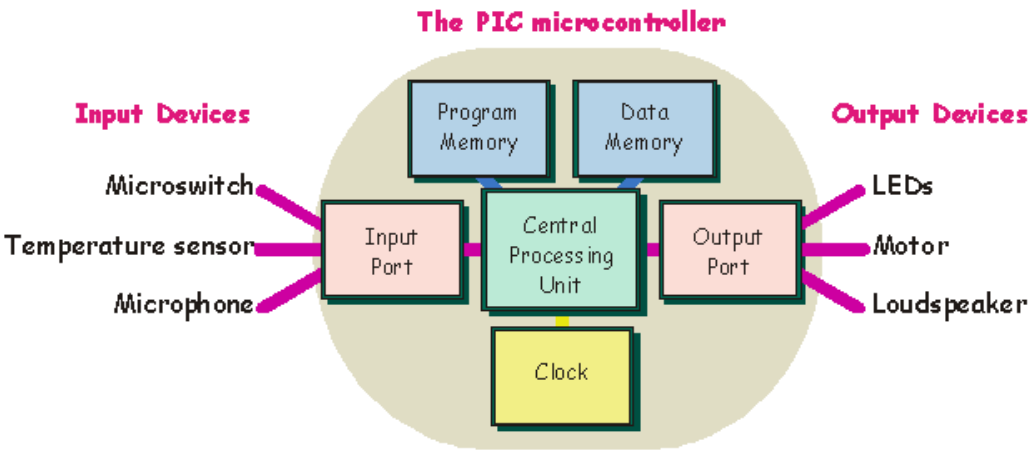
5 PICMICRO

PIC tulee sanoista (Peripheral Interface Controller), joka viittaa mikrokontrolleri ryhmään jonka on valmistanut Arizona Microchip. Kuten nimi ehdottaa, mikrokontrolleri on pieni laite, joka kontrolloi toisia elektronisia laitteita.

Niitä löytyy monista laitteista kuten:

- auton moottorista, lukkiutumattomista jarruista ja ilmastoinnista
- Millä tahansa laitteella, jolla on kaukosäädin, on melko varmasti mikrokontrolleri esim. televisiossa, videoissa, digitaalisissa kameroissa, kännyköissä, printtereissä, mikroaaltouuneissa, tiskikoneissa, pesukoneessa jne.

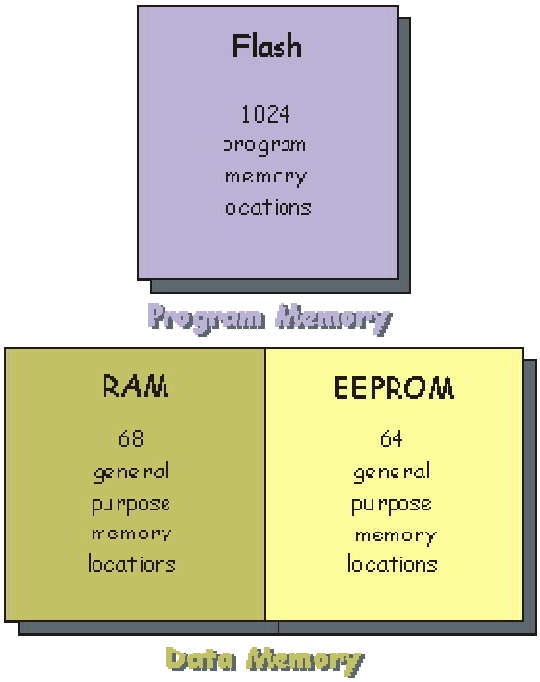
Mikrokontrolleri on digitaalinen integroitu piiri, ja se koostuu keskus prosessointi yksiköstä, muistista sekä tulo ja lähtö porteista. [4.]



Kuva 2. PIC-mikrokontrolleri [5]

PICmicron muisti

PICmicro-siruilla on kolme erillistä muistialuetta: ohjelmamuisti, käyttäjästä riippuva RAM, ja EEPOROM. Nämä nimet antavat vahvoja vihjeitä alueiden käyttötarkoituksista.



Kuva 3. PICmicro muisti 18pin 16F84 [8]

Ohjelmamuistia eli EPROM:ia käytetään ohjelman tallentamiseen. Joissakin PICmicro siruissa, kuten 16F84:ssa, tämä muisti käyttää flash-tekniikkaa. Tämä tarkoittaa, että sen voi ohjelmoida ja tyhjentää monta kertaa. Vanhemmat PICmicrot käyttävät PROM-ohjelmamuistissa ja monet näistä voi ohjelmoida vain kerran.

Datamuistia käytetään tietysti datan tallentamiseen. Osa tästä käyttää RAM muistia ja osa EEPROM:ia. EEPROM mahdollistaa tärkeän datan säilyttämisen, vaikka järjestämän virtalähde olisi suljettu. Esimerkiksi, ajatellaan, että PICmicro on osa lämmönsäätö kontrolleria, joka pitää inkubaattorin tietyssä lämpötilassa. Voisi olla järkevää tallentaa tarvittava lämpötila EEPROM:iin, jotta sitä ei tarvitsisi syöttää järjestelmään joka kerta, kun inkubaattori on päällä.

16F84 on verrattain pieni laite. Siinä 8192 sanaa ohjelma muistia (jokaisella on 13 bittiä), 368 bittiä käyttäjä muistia ja 256 bittiä EEPROM:ia. [8.]

6 MIKROKONTROLLERIT

Mikrokontrollerin ytimessä on Central Processing Unit (Keskus Prosessointi Yksikkö) eli CPU. Tämä prosessoi digitaaliset signaalit tekevät laskelmat ja loogiset operaatiot, luovat aikaviiveitä, aloittavat signaalisekvenssit jne.

Keskusyksikkö seuraa toimintaohjeita, jotka on tallennettu tiettyyn osaan muistia, jota kutsutaan ”program memory” eli ohjelmamuistiksi, joka sijaitsee PICmicron sisällä.

Toisinaan keskusyksikön täytyy tallentaa dataa ja myöhemmin hakea se. Tähän keskusyksikkö käyttää toista aluetta muistista, jota kutsutaan ”data memoryksi” eli datamuistiksi.

Kello synkronoi keskusyksikön toiminnot. Se lähettää jännitepulsseja keskusyksikölle, joka kontrolloi milloin dataa liikutetaan järjestelmässä ja milloin ohjelman toimintaohjeet suoritetaan. Mitä nopeampi kello, sitä nopeammin PICmicro suorittaa ohjelman. Yleisesti kello käy 20 MHz taajuudella (kaksikymmentä miljoonaa jännite pulssia joka sekunti.)

Voidakseen keskustella ulkopuolisen maailman kanssa mikrokontrollerissa on portteja. Jokaisessa portissa on 8 liitäntää, joihin usein viitataan sanalla ”bitti”, koska jokainen liitäntä vastaa yhtä bittiä 8 pinnisessä tulossa, joka puolestaan vastaa yhtä tavua dataa. Sensoreista lähtevä informaatio syötetään järjestelmälle tulo porttien kautta. Mikrokontrolleri prosessoi tämän datan ja käyttää sitä kontrolloidessaan laitteita, jotka ovat kytkettyinä lähtö portteihin. Portit itsessään ovat monimutkaisia elektronisia piirejä, eivät siis pelkästään päätteitä, joihin voi liittää komponentteja.

Kun PICmicro mikrokontrolleria käytetään, täytyy ensin tarkentaa, kuinka porttien halutaan käyttäytyvän. Portit ovat kaksisuuntaisia, joka tarkoittaa sitä, että ne voivat toimia joko tulo tai lähtö portteina. Kun luodaan ohjelma PICmicro:lle, aloitetaan konfiguroimalla ensin portit eli päätetään, toimivatko ne tulo- vai lähtöportteina.

Tuloportti voi vastaanottaa dataa (informaatiota) yhdessä kahdesta tilasta, joko analogisena signaalina tai digitaalisena signaalina. On tärkeää ymmärtää tarkalleen ero näiden kahden välillä. [5.]

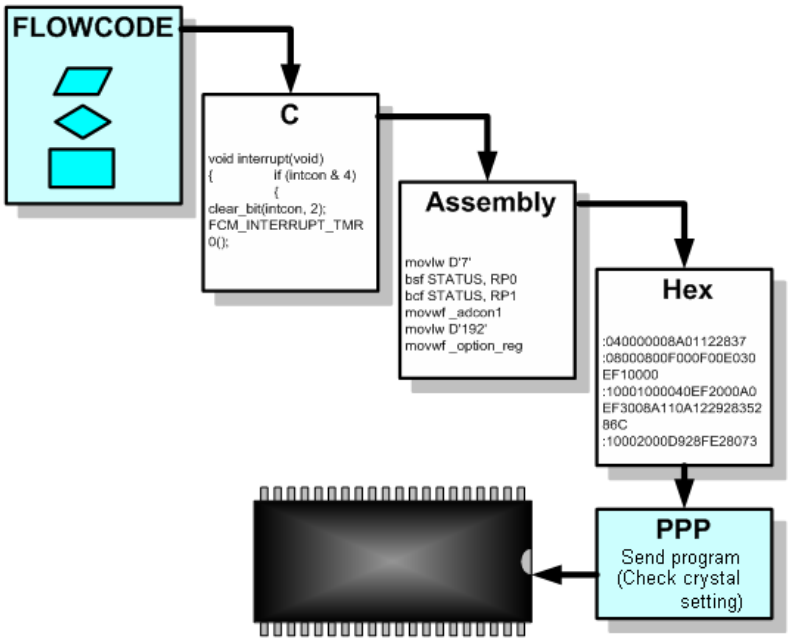
6.1 Mikrokontrolleriohjelmointi

Mikrokontrollerit ovat ohjelmoitavia laitteita. Ne tekevät juuri sen mitä ohjelmoija käskee, eikä mitään muuta. Ohjelma on lista ohjeita, mukaan lukien se data, mikä tarvitaan toimintojen toteuttamiseen.

Mikrokontrollerit ymmärtävät ainoastaan numeroita. Tämän takia on olemassa kaksi vaihtoehtoa ja niistä molemmat vaativat jonkinlaisen käännöksen. Voimme kirjoittaa ohjelman korkean tason ohjelmointikielellä, kuten C++ ja sitten kääntättää kyseinen ohjelma numeroiksi. Toinen vaihtoehto on käyttää assembler-ohjelmointia. Tästä eteenpäin ohjelma on helppo muuttaa numeriseksi koodiksi, jota mikrokontrolleri ymmärtää.

Nämä kaksi ääripäätä tunnetaan nimillä korkeantason ohjelmointikieli (kieli joka on lähellä englantia) sekä matalantason ohjelmointikieli (joka on assembler). Ensimmäinen näistä kahdesta tavasta on yleensä helpompi ja nopeampi ohjelmoijalle, mutta ohjelman ajamiseen menee pitempi aika, koska ohjelma pitää kääntää mikrokontrollerille samalla, kun se ajetaan. Jälkimmäinen tapa on paljon hitaampi ohjelmoijalle, mutta loppujenlopuksi sen ajaminen laitteessa on huomattavasti nopeampaa. Flowcode käyttää korkeamman tason ohjelmointia, joka tekee ohjelmoinnista äärimmäisen helppoa ja nopeaa.

Flowcodessa PICmicro sirun ohjelmoimien on helppoa. Kun vuokaavio on luotu, yhdellä painalluksella ohjelmisto kääntää ohjelman numeeriseen muotoon. Flowcode välittää luodun ohjelman erilaisten prosessien läpi ennen, kun se päätyy PICmicro-laitteeseen: vuokaavio käännetään ensin C-koodiksi, sitten assembleriksi ja viimeinen hexadesimaaliluvuiksi. Tämän jälkeen hexakoodi lähetetään PICmicro-laitteeseen, joka ymmärtää ja osaa tulkita hexakoodia PPP-tytärohjelman avulla.



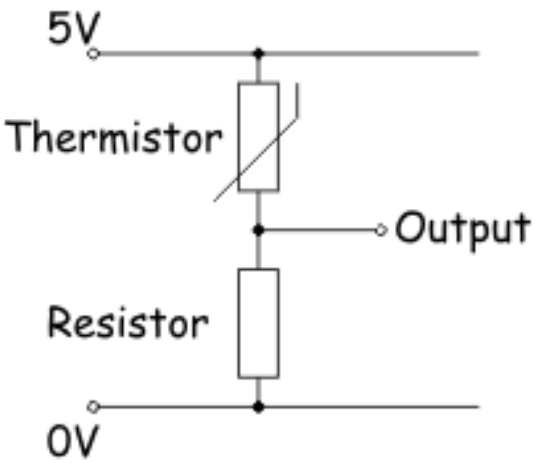
Kuva 4. Flowcodeprosessi vuokaaviosta E-Blocks-sirulle [9]

HEX-koodi tavallaan poltetaan PICmicro-sirun EPROM-muistiin, eli ohjelma ei katoa, vaikka kyseinen siru poistettaisiin. [9.]

Analoginen data

Monet elektroniset sensorit tarjoavat signaaleja analogisessa muodossa. Esimerkiksi, mikrofoni tarjoaa sähköisen ”kopion” ääniaallosta.

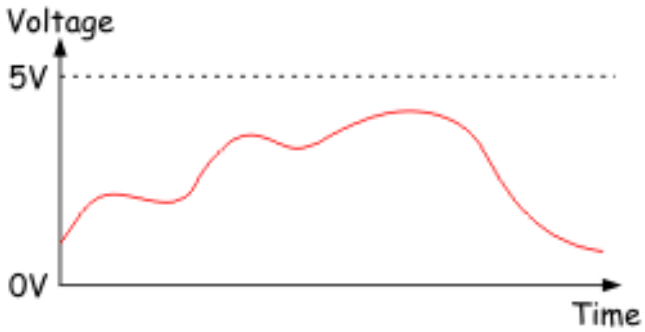
Seuraavassa kuvassa on piirikaavio lämpö sensorista.



Kuva 5. Lämpösensori [6]

Tuotettu volttimäärä nousee sitä mukaan, kun lämpötila nousee. Se on analoginen signaali, koska volttimäärä kopioi lämpötilan käytöksen. Analoginen signaali voi saada minkä tahansa volttiarvon, riippuen ainoastaan virtalähteestä, jota käytetään.

Tässä tapauksessa lämpötila sensorin tuotto voisi teoriassa nousta niinkin ylös kuin 5 volttiin, tai laskea niin matalalle kuin 0 volttiin. [6.]



Kuva 6. Analoginen signaali [6]

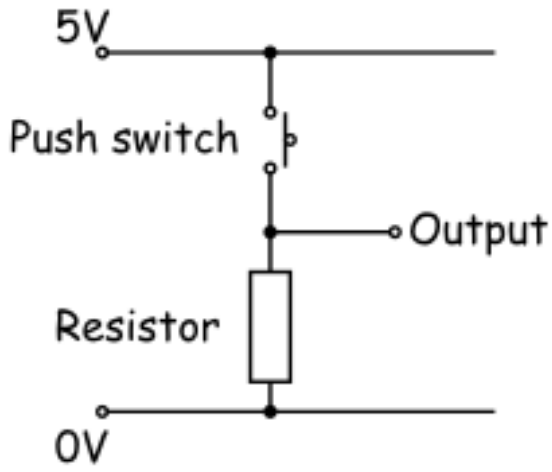
Digitaalinen data

Digitaalinen signaali kantaa tiedon numero muodossa. Elektroniset järjestelmät käyttävät binäärinumerojärjestelmää, joka käyttää ainoastaan numeroita 0 ja 1. Nämä kaksi numeroa on koodattu volteiksi.

Voidaan esimerkiksi päättää että:

0 = alhainen volttimäärä.
1 = korkea volttimäärä.

Digitaalisilla signaaleilla on siis vain kaksi volttiarvoa. Ne ovat virtalähteen maksimi ja minimi voltti määrä. Seuraavaksi pitää miettiä, miten nämä numerot saadaan syötettyä elektroniseen järjestelmään. Yksi erittäin hidas tapa olisi käyttää vaihdetta (digitaalinen sensori).



Kuva 7. Digitaalinen sensori [7]

Kun vaihde on auki (ei painettuna,) vastus painaa lähtötehon alas 0 volttiin. Tämä lähtöteho voisi kuvata numeroa 0. Kun vaihde on kiinni (painettu), lähtöteho on yhdistettynä positiiviseen lähteeseen ja on 5 V tässä tapauksessa. Tämä voisi kuvata numeroa 1. [7.]

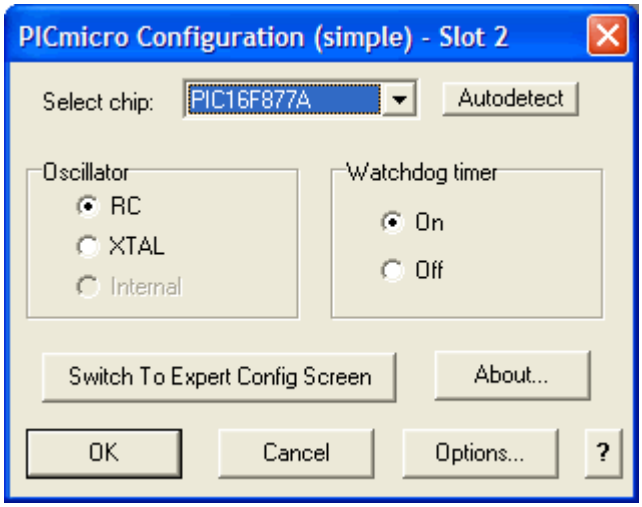
6.2 Mikrokontrollerin kellotaajuuden asettaminen

Jokainen mikrokontrolleri tarvitsee kellosignaalin toimiakseen. Laitteen sisäinen kellosignaali kontrolloi operaatioiden nopeuksia ja jokainen ohitettava kellojakso synkronoi useiden eri laiteosien operaatiot. Kellotaajuuden valinnassa on otettava huomioon useita seikkoja ja ennen ohjelman siirtämistä PICmicro laitteeseen täytyy ymmärtää valinnat ja tehdä muutamia päätöksiä.

On tärkeää muistaa, että kellotaajuus täytyy asettaa kahteen eri paikkaan: Flowcode-ohjelmaan sisäisesti sekä itse E-Blocks-laitteeseen, koska muuten ohjelma ei toimi. Mikrokontrolleriin voidaan asettaa kellotaajuus usealla eri tavalla mutta tässä työssä keskitytään vain kristallikelloon, rc-kelloon ja sisäiseen kelloon. PICmicro Multiprogrammer tukee näitä kaikkia tapoja. [11.]

Rautapuolella pitää varmistua siitä, että haluttu kellopiiri on kytketty mikrokontrolleriin. Esimerkkinä voidaan käydä nopeasti läpi RC metodi kellotaajuuden säätämiseksi. Eli ensimmäinen askel on varmistaa E-Blocks-laitteesta, että SW2 on RC asennossa ja SW1 SLOW asennossa, jotta saadaan kellotaajuudeksi 100Hz (joka on todella hidas kellotaajuus, mutta mahdollistaa ohjelman tarkkailun askel askeleelta). Tämän jälkeen suunnataan Flowcode-ohjelmistoon josta navigoidaan Chip-valikkoon ja valitaan Configure. Eteen aukeaa PICmicro Configuration ikkuna (kuva 8), jonka kautta voi-

daan asettaa ohjelmalle sisäiset asetukset, jotta ohjelma toimii oikein E-Blocks-laitteessa. Kun Flowcode lataa ohjelman PICmicro-laitteeseen, se muuttaa sisäisen kellon sirun kytkennän niin, että se kuvastaa ulkoista kellosirua, joka on kytkettynä E-Blocks-laitteeseen.



Kuva 8. PICmicro Configuration ikkuna [12]

Tästä ikkunasta voidaan varmistaa, että Flowcode muistaa asetukset oikein ja tarvittaessa voidaan tehdä muutoksia. [12.]

7 KÄYTÄNTÖ

Tämän työn käytännönosiossa on tarkoitus tarkastella Flowcode-sovellusta ihan perusteista lähtien ja myöhemmin tehdä ja testata käytännössä pienimuotoinen esimerkki sovellus. Tarkoituksena ei ole luoda monimutkaista sovellusta jota on vaikea ymmärtää, vaan keskittyä perusasioihin ja nähdä pieni välähdys siitä, mihin Flowcode pystyy.

Flowcode-ohjelmiston sekä E-Blocks-laitteen asennus käsitellään perinpohjaisesti alusta loppuun helposti ymmärrettävällä tavalla. Tämän jälkeen käydään läpi Flowcode-ohjelmiston useat monipuoliset työkalut joiden avulla lopuksi tehdään yksinkertainen Flowcode-sovellus ja perehdytään ohjelman tekemisen eri askeleisiin ja vaiheisiin.

7.1 Asentaminen

Tässä laitteiston ja ohjelmiston asennusta käsittelevässä osiossa käydään nopeasti Flowcoden ja E-Blocks-laitteen asennusprosessit läpi. Ohjelmiston ja laitteiston asentaminen on äärimmäisen helppoa ja erityistesti ohjeiden kanssa käyttäjä ei oikeastaan voi tehdä mitään väärin.

Asennusprosessissa on kuitenkin mahdollista tehdä helppoja virheitä, jotka myöhemmin voivat aiheuttaa erilaisia ongelmia, kuten ohjelman toimimattomuuden tai esimerkiksi ongelmia laitteen kytkemisessä. Tämän takia asennusohjeita tulee aina noudattaa tarkasti. Jos johonkin kohtaan ei tiedä oikeaa vaihtoehtoa on syytä ottaa asiasta selvää ennen asennuksen viimeistelyä.

7.1.1 Flowcoden asentaminen

Flowcoden asentaminen alkaa samalla tavalla kuin minkä tahansa muunkin ohjelmiston, eli ensimmäiseksi asetetaan ohjelmiston asennuslevy CD-asemaan. Ohjelman asennus alkaa, jonka alussa Flowcode-asennusrutiini lataa tytärasennusrutiinit laitteistolle. Flowcode-ohjelmisto asentuu ensimmäisenä.

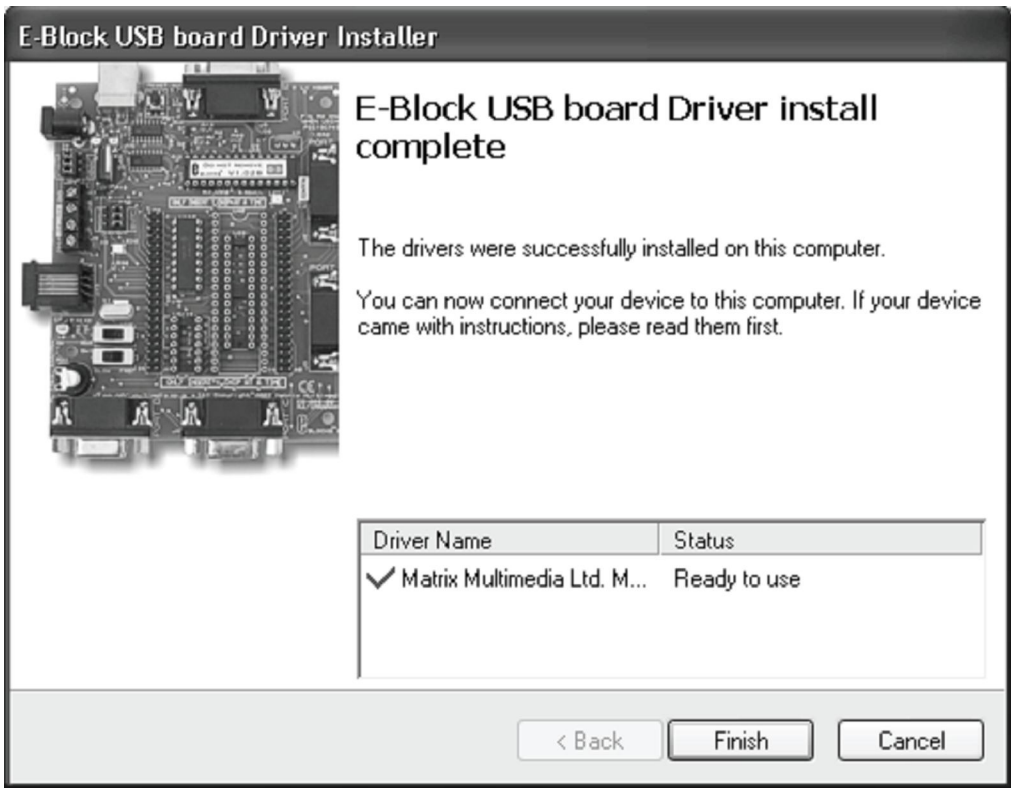
Ensimmäisen ikkunan jälkeen käyttäjälle esitetään mahdollisuus asentaa PPP. PPP on ohjelma, joka mahdollistaa Flowcodella HEX-tiedostoon käännetyn ohjelman siirtämisen E-Blocks-laitteeseen. PPP-asennuksen jälkeen alkaa kolmas asennus rutiini, joka lisää tuen Matrix Multimedian USB-laitteille. [13, s. 2.]

Seuraava askel on valita, mihin Flowcode-ohjelmisto asennetaan. Voidaan valita joko kohde itse tai vain painaa ”Next”-painiketta. Seuraavassa ikkunassa ohjelma varmistaa, että käyttäjä on tyytyväinen valintoihinsa. Tästä jatketaan painamalla ”Next”.

Seuraavassa ikkunassa kysytään, mihin PPP-ohjelmisto halutaan asentaa ja samoin kuin Flowcoden kanssa voidaan valita, mihin ohjelmisto asentuu tai vain painaa ”Next”. Tämän jälkeen aukeaa ”Select Features”-ikkuna josta käyttäjä voi päivittää Matrix Multimedia-ohjelmistopaketit, jotka käyttävät PPP:tä. Jos käyttäjällä ei ole tarkkaa tietoa siitä, mitä tässä tulee tehdä, niin

valitaan vain ”Next”. Tässä työssä esimerkiksi ohitettiin tämä askel ”Next”-painikkeella.

Tämän jälkeen aukeaa ”Software Installation”-varoitusikkuna, joka on tuttu muun muassa erinäisten ajurien asennuksesta. Eli tämä ei tarkoita mitään muuta kuin, että Flowcode ei ole rekisteröity Microsoftin kanssa. Valitaan ”Continue anyway”, Flowcoden asennus on täysin turvallista eikä vahingoita konetta.



Kuva 9. Flowcode-asennusohjelman viimeinen ikkuna [13, s. 5.]

Tämä on asennusohjelman viimeinen ikkuna. Kun painetaan ”Finish”, E-Blocks-laitteen ajuriohjelmisto asentuu koneelle. Tämä tarkoittaa sitä, että ensikertaa liitettäessä E-Blocks-laite koneeseen kiinni, tietokone tunnistaan laitteen automaattisesti. [13, s. 5.]

Seuraavassa ikkunassa (Enter Flowcode V3 License Key) käyttäjältä pyydetään ohjelmiston lisenssiä, joka löytyy Flowcode asennus CD:n kansien sisältä. Jos avainta ei ole, voi käyttäjä ottaa käyttöön 30-päivän demo-version ohjelmistosta valitsemalla ”Do not enter key”. Seuraavassa ikkunassa ilmoitetaan asennuksen onnistumisesta ja ainoa painike, mitä voidaan painaa on ”Finish”.

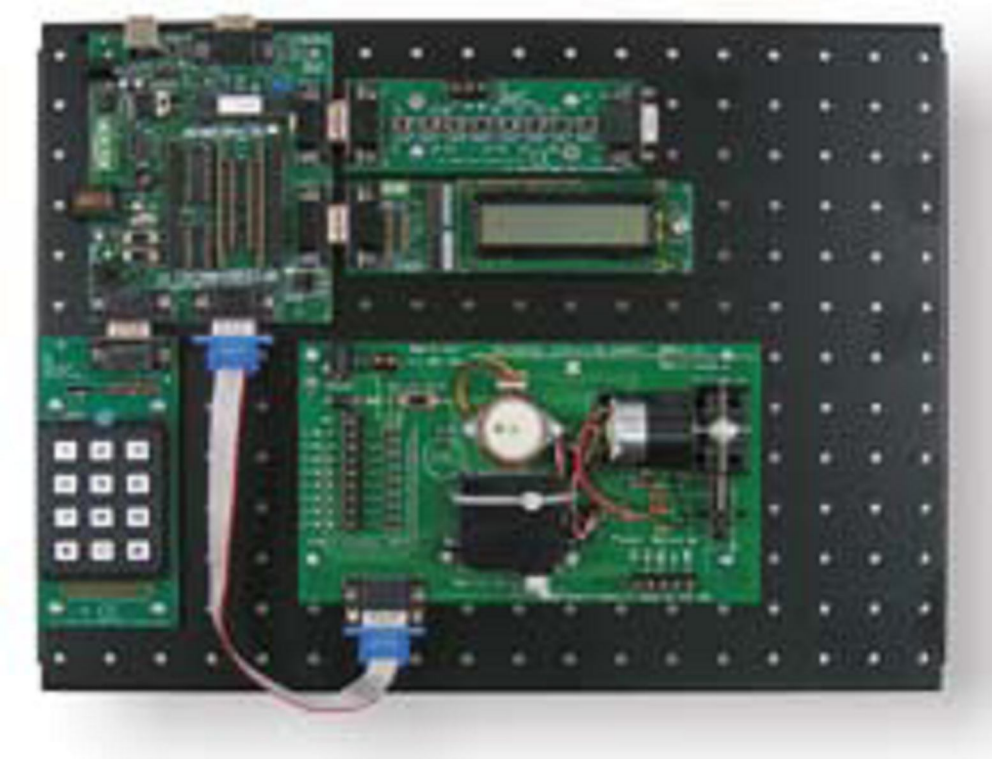
Seuraavassa ikkunassa käyttäjä voi vielä miettiä onko tehnyt oikeat valinnat PPP:n asentamisen suhteen. Jos on niin, tästä jatketaan painamalla ”Next”. Viimeinen ikkuna ilmoittaa asennuksen onnistuneen ja ainoa painike, mitä käyttäjä voi painaa, on ”Finish”. [13, s. 6.]

7.1.2 E-Blocks-laitteen kytkeminen

E-Blocks-laite kytketään tietokoneeseen suoraan USB-porttiin USB-kaapelin avulla. Kun laite on kytketty, tietokone tunnistaa laitteiston, jos käyttäjä on asentanut Flowcode-ohjelmiston ja sen mukana tulleet ohjelmat ja ajurit onnistuneesti.

Laitteen kytkemiseen ei liity oikeastaan mitää sen isompaa eli kyseessä on plug-and-play laite. Kytkettäessä kone tunnistaa laitteen ja ilmoittaa uuden komponentin asentamisesta tehtävien hallinnassa.

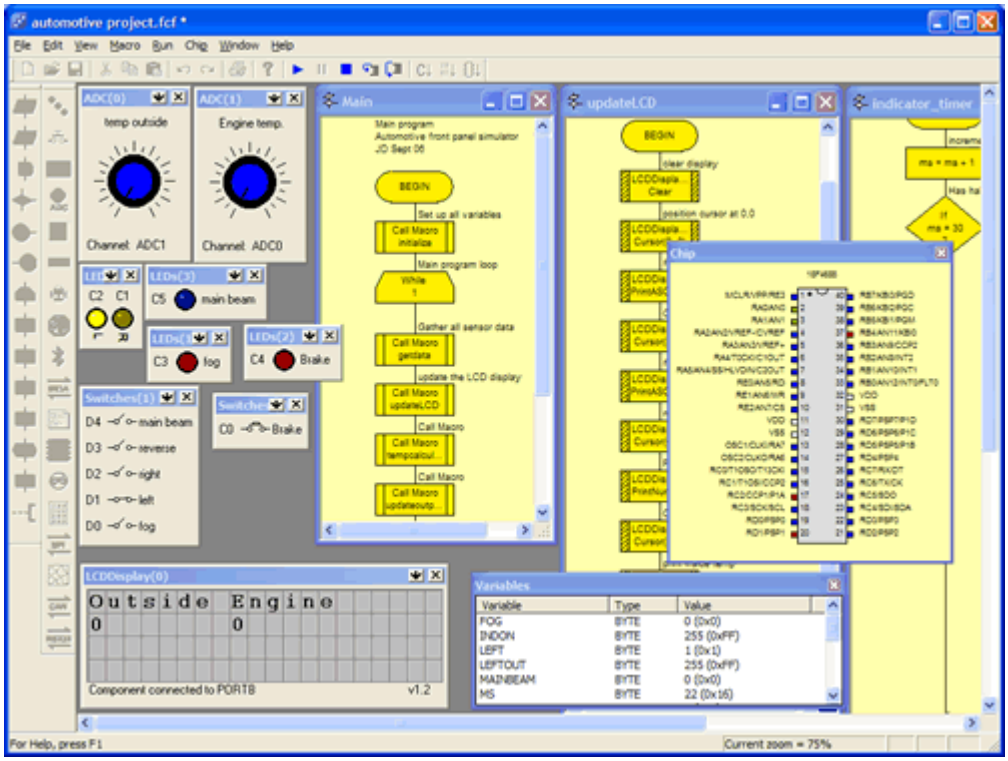
Tämän jälkeen E-Blocks-laitteisto on valmis käytettäväksi. PPP:n kautta voi samantien lähettää ohjelman levylle, jos näin tahtoo tehdä.



Kuva 10. E-Blocks-laitteisto [14, s. 4]

7.2 Flowcoden työkalut

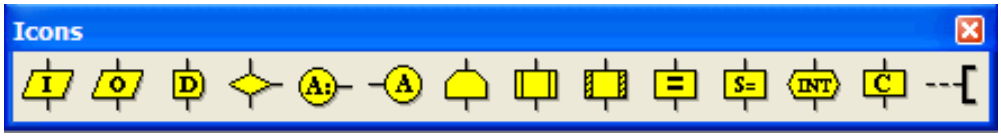
Flowcoden työtila koostuu työalueesta, jossa vuokaavioikkunat sekä useat erilaiset työkalut ovat näkyvillä. Tähän kuuluu kaksi ikonityökalupalkkia, joista drag and drop menetelmällä voi vetää vuokaaviokomponentteja ohjelmaikkunaan. Mukana on myös ikkunoita, joista näkee mikrokontrollerin tilan ja siihen liitetyissä komponenteissa tapahtuvat muutokset.



Kuva 11. Flowcode-ohjelman eri työkalut [10]

Ikoni työkalupalkki

Tämä työkalupalkki koostuu drag-and-drop-ikoneista, josta vedetään vuokaavioon erilaisia elimenttejä, joita yhdistelemällä syntyy ohjelma. Normaali sijainti on ohjelman vasemmassa laidassa (kuva 11), mutta sen sijantia voi myös vapaasti muokata. [10.]



Kuva 12. Ikoni työkalupalkki [10]

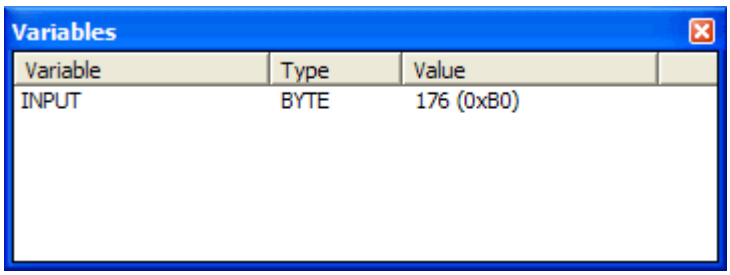
Komponentti työkalupalkki

Tässä työkalupalkissa on esillä kaikki erilliset komponentit, jotka voidaan liittää mikrokontrolleriin. Painamalla komponettia se ilmestyy ohjelman pääikkunaan. Pinniyhteydet ja komponenttien ominaisuuksia voidaan siten muokata. Normaali sijainti on ohjelman vasemmassa laidassa työkalupalkin vieressä (kuva 11), mutta myös tätä voi liikutella vapaasti. [10.]

Kuva 13. Komponentti työkalupalkki [10]

Muuttujaikkuna

Kun vuokaaviota simuloidaan, kaikkien muuttujien arvot voidaan nähdä tässä ikkunassa. Muuttujien arvot päivitetään joka kerta, kun komento simuloidaan, mutta ikkuna ei päivity, jos simulaatiota ajetaan täydellä nopeudella. Jos vuokaaviota simuloidaan ja samalla painetaan pause painiketta, voidaan suoraan tästä ikkunasta muokata muuttujien arvoja helposti hiirenpainalluksella. Tämä mahdollistaa vuokaavion testaamisen tunnetuissa olosuhteissa. [10.]



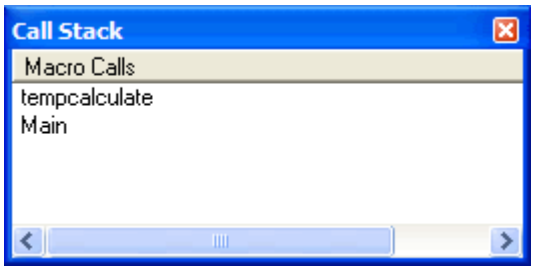
Kuva 14. Muuttuja ikkuna [10]

Ulkoinen komponentti ikkuna

Tässä ikkunassa näytetään laitteen status, joka on kiinni mikrokontrollerissa. Komponentti muuttuu aktiiviseksi, kun vuokaaviota simuloidaan. Tämä ikkuna myös mahdollistaa esimerkiksi porttien avaamisen ja sulkemisen simuloitavasta laitteesta. [10.]

Kutsu pinoikkuna

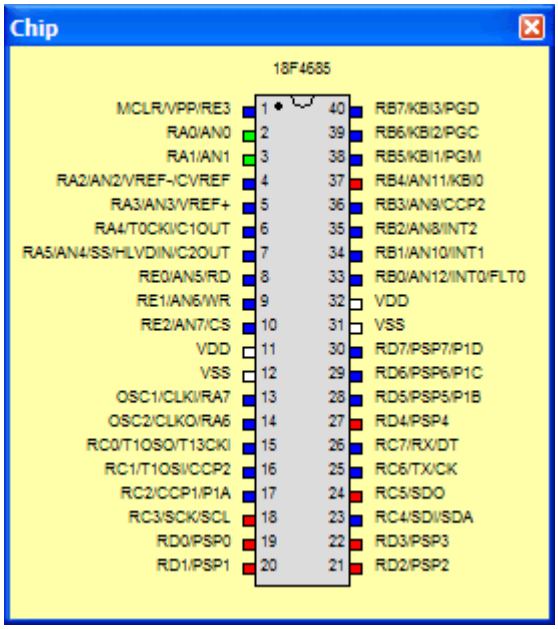
Tämä ikkuna näyttää tämän hetkisen makron jota simuloidaan. Tämä on hyödyllinen työkalu silloin, kun makro kutsuu toista makroa samalla, kun simulointi on käynnissä. [10.]



Kuva 15. Kutsu pinoikkuna [10]

Mikrokontrolleri-ikkuna

Tällä hetkellä käytössä oleva siru tai mikrokontrolleri näytetään tässä ikkunassa. Kun vuokaaviota simuloidaan, voidaan nähdä mikrokontrollerin I/O-porttien tilat punaisena ja sinisenä, jotka tarkoittavat korkeaa ja matalaa ulostuloa.



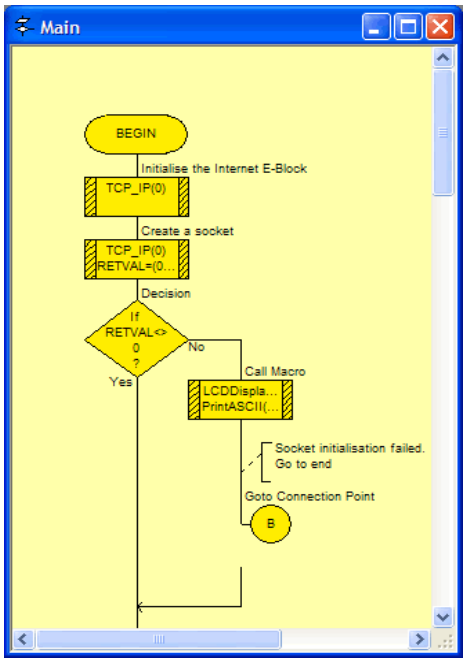
Kuva 16. Mikrokontrolleri-ikkuna [10]

Työkalupalkkien irroittaminen

Kaikki työkalupalkit voi irrottaa alkuperäisiltä paikoiltaan ja joko jättää vapaasti liikuteltaviksi, tai kiinnittää menu palkkiin, laitoihin tai Flowcode-ikkunan alalaitaan. Työkalupalkki on helppo irrottaa alkuperäiseltä paikaltaan painamalla hiiren nappi pohjaan palkin kohdalla ja vetämällä se irti. [10.]

Vuokaavioikkuna

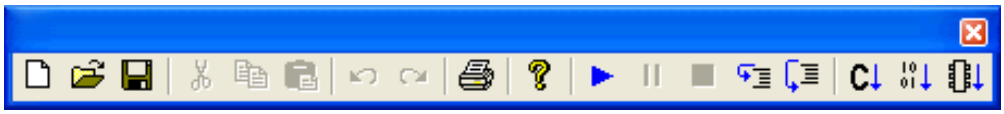
Ikonit, jotka muodostavat vuokaavion näytetään tässä ikkunassa. Lisäksi ikonit, jotka voivat muodostaa makron näytetään omassa erillisessä ikkunassa. Päävuokaavioikkuna on aina näkyvissä ja macro-ikkuna voidaan tarpeen vaatiessa näyttää tai piilottaa. [10.]



Kuva 17. Vuokaavioikkuna [10]

Menu- ja simulaatiotyökalupalkki

Tämän työkalupalkin painikkeet mahdollistavat normaalit toiminnot, jotka löytyvät melkein kaikista Windows-sovelluksista, kuten: uusi tiedosto, avaa tiedosto, tallenna, tulosta, ohje ja niin edelleen. Tämän lisäksi palkissa on simuloimiseen ja lataamiseen tarvittavat painikkeet. Palkin normaali sijainti on ruudun yläosassa (kuva 8). [10.]



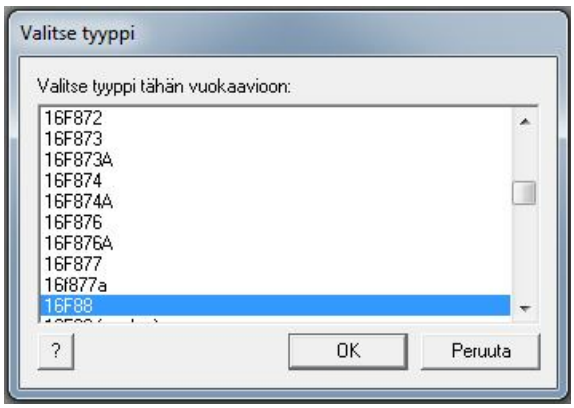
Kuva 18. Flowcode-menu ja simulaatio-työkalupalkki [10]

7.3 Kätännön esimerkki Flowcode-ohjelmasta

Käytännön esimerkkiä varten luin lukuisia ohjeita ja satoja sivuja opettajille tarkoitettua materiaalia, mutta loppujenlopuksi jouduin tyytymään yksinkertaiseen esimerkkiin joka soveltaa kurssilla opittuja taitoja. Tarkoitus on käydä yksityiskohtaisesti läpi ohjelman luomisprosessi, uuden työn aloittamisesta E-Blocks-laitteeseen siirtämiseen asti.

Toteutettu ohjelma on yksinkertainen sekuntikello joita kurssin eri oppitunneilla tehtiin useita erilaisia. Ohjelma näyttää ajan sadasosien tarkkuudella ja toimii suoraan E-Blocks-laitteessa, kun ohjelma on ensin käännetty HEX-tiedostoon. Oli ohjelma millainen hyvänsä, sen toteuttamiseen käytetään aina samoja askelia, ainoastaan komponenttien määrä ja toimintatavat vaihtelevat.

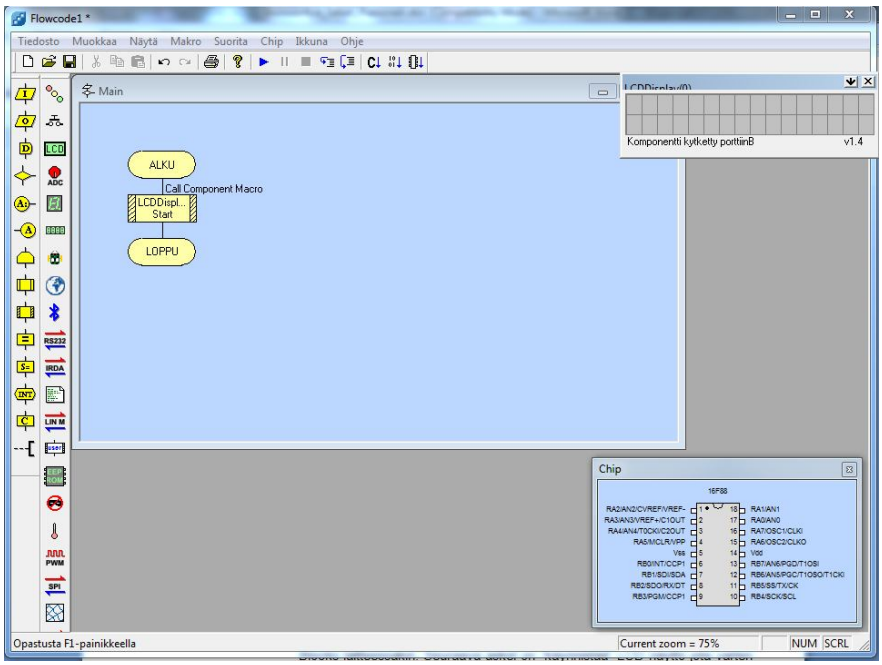
Ensimmäiseksi avattiin Flowcode-ohjelmisto, josta hyppää esiin ikkuna jossa kysytään halutaanko aloittaa uusi projekti vai jatkaa vanhaa. Valittiin uusi projekti, jonka jälkeen vuokaavion tyypiksi valitaan 16F88, joka vastaa koululla olevaa E-Blocks-laitetta ja sen eri komponentteja.



Kuva 19. Vuokaavion tyyppi

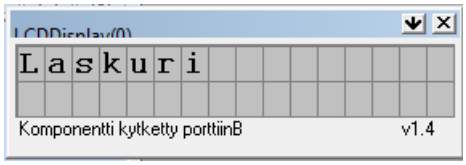
Valinnan jälkeen eteen aukeaa vuokaavion pohja jossa on tällähetkellä vain aloitus ja lopetus pisteet. Koska päädyin tekemään esimerkin joka käyttää LCD-näyttöä, ensimmäiseksi valitaan LCD-komponentti (kuva 13). Tämä mahdollistaa LCD-näytön käyttämisen niin Flowcode-simulaatiossa, kuin E-Blocks-laitteessakin. Seuraava askel on ”käynnistää” LCD-näyttö, jota varten

täytyy lisätä komponentti makro (kuva 12). Makroa oikealla näppäimellä painamalla saadaan esiin liukuikkuna josta valitaan ”ominaisuudet”. Vasemmalla puolella ikkunaa on lista käytössä olevista komponenteisesta ja tässä tapauksessa ohjelman pitäisi tarjota ainoastaan ”LCDDisplay(0)” komponenttia joka valitaan. Oikealle puolelle ruutua ilmestyy makro ikkunaan erilaisia toimintoja ja vaihtoehtoja. Valitaan ”Start”, joka käynnistää LCD-näytön ohjelman eri toimintoja varten, jotka halutaan näyttää näytöllä. Tässä vaiheessa ohjelman tulisi näyttää seuraavalta (kuva on hieman epäselvä mutta ajaa asian):



Kuva 20. Ohjelman alku

Tässä vaiheessa ohjelma ei vielä tee mitään ja ainoa toiminnallisuus on LCD-näytön käynnistys. Seuraavassa askeleessa lisätään ohjelmaan LCD-näytön ensimmäiselle riville tavallaan otsikko teksti joka on helppo toteuttaa lisäämällä uusi komponentti macro josta samalla tavalla, kuin aikasemminkin valitaan komponentiksi LCDDisplay(0). Tällä kertaa valitaan oikeasta makro ikkunasta ”PrintString” makro, jonka avulla LCD-näytölle lähetetään teksti. Tässä tapauksessa kirjoitetaan ”muuttujat” painikkeen vasemmalla puolella olevaan tyhjään kenttään ”Laskuri:” joka on siis tämän ohjelman otsikko ja se tulostuu LCD-näytön ensimmäiselle riville.



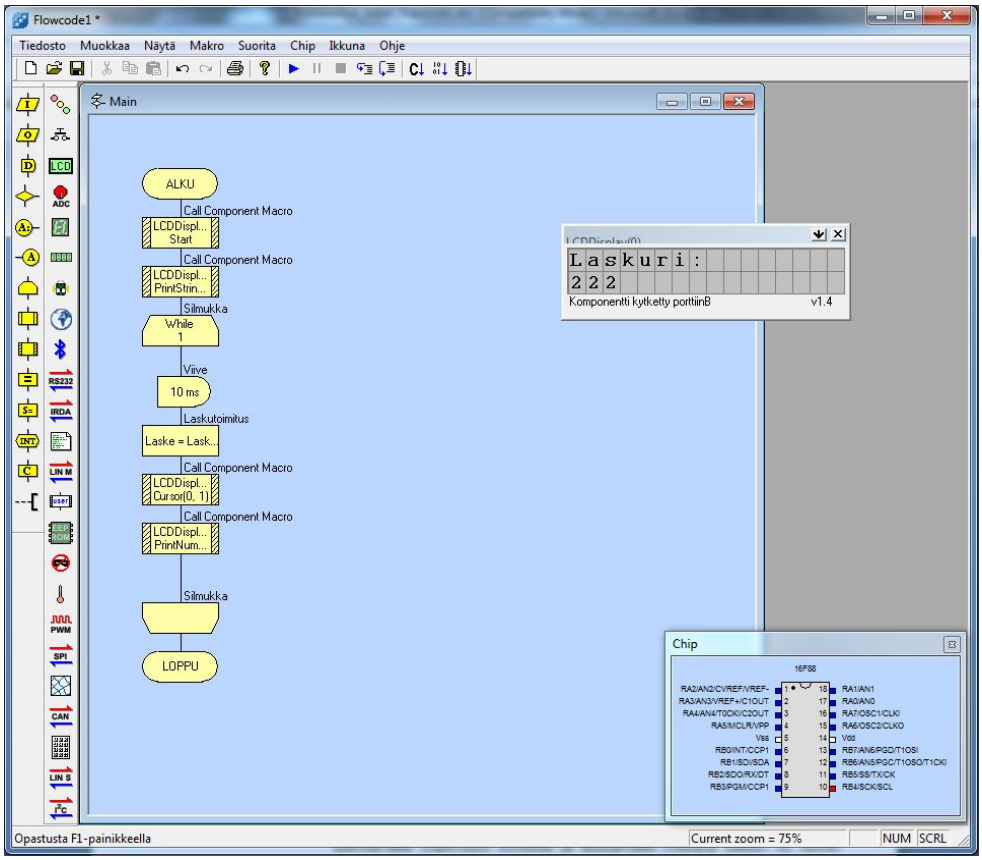
Kuva 21. LCD näyttö

Tämän jälkeen aloitetaan laskurin tekeminen, joka jatkaa samalla suoraviivaisella tavalla, kuin tähänkin asti, eli ensiksi valitaan silmukka työkalu (kuva 12) ikoni työkalupalkista. Tämän jälkeen silmukan sisälle lisätään ”laskutoimitus” ikoni, jonka avulla luodaan yksinkertainen laskuri. Laskutoimitusikonia oikealla painikkeella painamalla aukeaa ominaisuusikkuna, josta painetaan ”muuttuja” painiketta. Eteen aukeaa ikkuna, jossa on listattu kaikki kyseisessä ohjelmassa käytössä olevat muuttujat ja tässä tapauksessa ikkunan pitäisi vielä olla tyhjä. Valitaan oikeasta yläkulmasta ”lisää uusi muuttuja” jolle annetaan nimeksi ”Laske” ja tyypiksi valitaan INT. Seuraavaksi valittiin ”Laske” muuttuja ja painettiin ”käytä muuttujaa” painiketta joka ottaa valitun muuttujan käyttöön. Laskutoimistusta varten kirjoitetaan laskutoimitus kentään pieni plus lasku joka tässä tapauksessa on ”Laske = Laske + 1”. Tämä aiheuttaa sen, että laskuri lähtee laskemaan nollasta eteenpäin, kunnes ohjelma pysäytetään.

Ohjelman kaksi tärkeintä ominaisuutta on nyt paikoillaan. Seuraavaksi lisätään ohjelmalle viive, eli tässä tapauksessa käytetään 10 millisekuntia, jotta saadaan sadasosien tarkuudella rullaava ”kello”. Sitten kutsutaan ”Laske” muuttujaa silmukassa LCD-näytön ruudulle, jotta saadaan ohjelma toistamaan haluttua käskyä ja näyttämään se LCD-näytön ruudulla. Tämä tapahtuu lisäämällä silmukan sisään komponentti makro josta taas tuttuun tapaan valitaan ”LCDDisplay(0)”, mutta tällä kertaa valitaan makroksi ”PrintNumber”, jonka muuttujaksi joko valitaan tai suoraan kirjoitetaan ”Laske”.

Tämän jälkeen voidaan valita ohjelman suorituksen kannalta tärkeä komponentti joka mahdollistaa LCD-näytöllä rivin vaihdon, joka tapahtuu lisäämällä komponentti makro josta edelleen valitaan LCD-näyttökomponentti mutta nyt valitaan makroksi ”Cursor”, joka saa muutuja arvoiksi 0 ja 1 (0, 1).

Nyt ohjelman pitäisi olla toimintavalmiudessa ja se näyttää seuraavalle:



Kuva 22. Valmiin ohjelman vuokaavio

Ohjelma on erittäin yksinkertainen ja muutaman oppitunnin suorittamisella jokainen pystyisi itse toteuttamaan vastaavan, vaikka ilman apua. Alun perin tarkoitus oli tehdä yksinkertainen TCP/IP-sovellus, joka esimerkiksi hakisi oman koneen IP-osoitteen LCD-näytölle. Vaikka tällaisen ohjelman luominen olisi todennäköisesti helppo homma asiansa osaavalle Flowcode-ohjelmoijalle, ei se kuitenkaan allekirjoittaneelta onnistunut edes syvällisen perehtymisen jälkeen.

8 JOHTOPÄÄTÖKSET

Työn tekeminen oli tarkoitus aloittaa jo kesällä 2009, mutta kesätöiden ja muiden kiireiden takia itse työn aloittaminen siirtyi syksyyn asti. Aloitin työn asentamalla ohjelmistot koneelle ja aloittamalla mukana tulleen 50 tunnin mittaisen kurssin. Muutamien viikkojen jälkeen kuitenkin huomasin olevani jumissa, sekä itse työn kirjoittamisessa, kuin myös kurssin suorittamisessa. Tästä syystä päätin jättää kurssin suorittamisen myöhempään ajankohtaan

ja keskittyä täysipainoisesti itse työn kirjoittamiseen sekä lähdemateriaalien etsimiseen ja lukemiseen..

Aluksi asetin itselleni erittäin realistisen tavoitteen kirjoittaa yhden sivun päivässä, mutta pian huomasin päivien kuluvan eri lähteiden ja materiaalien etsimiseen ja lukemiseen. Kuinka ollakkaan olikin kulunut jo kolme kuukautta ja sivuja oli kirjoitettu noin kymmenen.

Jouluun mennessä olin kirjoittanut noit puolet suunnittelemistani aiheista, ja asetin tavoitteeksi saada työn tehdyksi helmikuun loppuun mennessä. Valitettavasti talviolympialaiset tulivat sotkemaan kuviot sekä unirytmin niin pahasti, että homma jatkui vasta maaliskuun alussa.

Työn aikana luin useaan eri otteeseen siihen asti kirjoitetut materiaalit läpi ja useammin, kuin kerran, tein päätöksen poistaa sivukaupalla tavaraa. Jopa itse työn aihe vaihtui pelkästä TCP/IP-sovelluksen suunnittelemisesta ja tekemisestä paljon laajempaan aiheeseen.

Työn tarkoituksena oli esitellä E-Blocks-laitteen ja Flowcode-ohjelmiston ominaisuuksia ja tutkia, kuinka ne toimivat mikrokontrolleriohjelmoinnissa opetustyökaluna.

Teoria osioiden aineistojen hankkimiseen kului huomattavasti enemmän aikaa, kuin itse ohjelmiston asentamiseen ja ominaisuuksiin perehtymiseen. Tarvittavan tiedon hankkiminen hoidettiin itseopiskeluna perehtymällä ja lukemalla useita eri lähdeaineistoja. Suurin osa potentiaalisesta lähdemateriaalista oli kuitenkin enemmän tai vähemmän ”mainos” tyyliin kirjoitettu, jonka takia työssä käytettiin paljon ohjelmiston mukana tulleen kurssin opetusmateriaalia lähteenä.

Flowcode v3-ohjelmiston asennus onnistui helposti ohjeita seuraamalla ilman suurempia ongelmia. E-Blocks-laitteen käyttöönotto oli myös äärimmäisen yksinkertaista.

Kurssin suorittaminen alkaa erittäin suoraviivaisesti ihan perusasioista. Kurssilla käsitellään periaatteessa kaikki mikrokontrolleriohjelmointiin liittyvä alusta loppuun ja annetaan oppilaalle mahdollisuus oivaalta itse asioita. Uskallan sanoa, että itse kurssi on toteutettu erinomaisesti, vaikka allekirjoittanut ei itse koskaan ihan loppuun asti päässytkään.

Kurssin heikkouksia ovat mielestäni muutamassa kohdassa silmään pistäneet ohjeiden vajaavaisuudet. Kyse ei ole mistään vakavasta, mutta joitakin kertoja jouduin kyllä googlesta tarkistamaan mistä on kyse ja miten nyt tulisi toimia.

Vahvuuksia taas ovat suurimmalta osin tarkat ohjeet ja helposti ymmärrettävä ohjelmisto, jonka avulla harjoitussovellusten tekeminen sujui kohtalaisen helposti. Opettajan oma materiaali on myös varsin kattava ja läpikotainen perusteista asti.

Kurssia voin hyvällä omallatunnolla suositella kurssille, joka käsittelee mikrokontrolleriohjelmointia joko E-Blocks-levyn kanssa käytettäväksi tai pelkästään Flowcodella suoritettavaksi. Huomattuani, kuinka Flowcode pystyy simuloimaan erilaisia mikrokontrollerilaitteita, mieleen tuli nopeasti kysymys, mihin itse laitteistoa viimekädessä edes tarvitsee?

Opinnäytetyölle asettamani tavoitteet saavutin kohtalaisesti. En saanut ”An introduction to microcontroller programming” kurssia täysin suoritettua, mutta sain erittäin hyvän kuvan siitä mihin kurssi sekä laitteet ja ohjelmisto pystyvät opetustyökaluna. Työn lopussa ohjeiden mukaan laadittu sovellus onnistui yllättävän hyvin muutamia alussa tehtyjä virheitä lukuunottamatta. Tiedän myös nyt aikalailla läpikotaisin Flowcode-ohjelmiston ominaisuudet, sekä hallitsen mikrokontrolleri ohjelmminoin alkeet. Voisin siis sanoa, että kaikenkaikkiaan työn tekeminen oli erittäin palkitsevaa mutta henkisesti rankkaa ja vaativaa hommaa josta mielestäni suoriuduin kunnialla.

VIITELUETTELO

[1] Matrix Multimedia, About [web-sivu], Matrix Multimedia kotisivu [luettu 18.7.2009].
Luettavissa: http://www.matrixmultimedia.com/about.php.

[2] Matrix Multimedia, About E-Blocks [web-sivu], Matrix Multimedia kotisivu [luettu 23.8.2009].
Luettavissa:http://www.matrixmultimedia.com/abouteblocks-X.php?C1=Browse%20All%20Products&CAT=E-Blocks%20hardware.

[3] Matrix Multimedia, Introduction [pdf-artikkeli], Matrix Multimedia kotisivu [luettu 10.11.2009].
Luettavissa: http://www.matrixmultimedia.com/datasheets/TEFLC-60-3.pdf.

[4] An introduction to microcontroller programming, Matrix Course Viewer, About PICmicro chips [välilehti] [luettu 19.11.2009].

[5] An introduction to microcontroller programming, Matrix Course Viewer, About PICmicro chips, Microcontrollers [välilehti] [luettu 20.11.2009].

[6] An introduction to microcontroller programming, Matrix Course Viewer, About PICmicro chips, Digital Versus analogue, Analogue data [välilehti] [luettu 15.1.2010].

[7] An introduction to microcontroller programming, Matrix Course Viewer, About PICmicro chips, Digital Versus analogue, Digital data [välilehti] [luettu 16.1.2010].

[8] An introduction to microcontroller programming, Matrix Course Viewer, About PICmicro chips, Memory [välilehti] [luettu 18.1.2010].

[9] An introduction to microcontroller programming, Matrix Course Viewer, About PICmicro chips, Programming, The Flowcode Process [välilehti] [luettu 15.2.2010].

[10] An introduction to microcontroller programming, Matrix Course Viewer, Flowcode step-by-step, Basic Flowcode functions, Flowcode overview [välilehti] [luettu 20.2.2010].

[11] An introduction to microcontroller programming, Matrix Course Viewer, Clocking your PICmicro device, Introduction [välilehti] [luettu 1.3.2010].

[12] An introduction to microcontroller programming, Matrix Course Viewer, Clocking your PICmicro device, Clock settings [välilehti] [luettu 2.3.2010].

[13] Matrix Multimedia, Manual [pdf-artikkeli], Matrix Multimedia kotisivu [luettu 20.4.2010].
Luettavissa:
http://www.matrixmultimedia.com/software/FLOWCODE%203%20MANUAL pq.pdf.

[14] Matrix Multimedia, Brochure for Education [pdf-artikkeli], Matrix Multimedia kotisivu [luettu 2.5.2010].
Luettavissa:
http://www.matrixmultimedia.com/articles/brochure%20-%20E-blocks%20for%20Education.pdf.

Lisään vielä tähän loppuun Flowcodella toteutetun sekuntikello sovelluksen C-koodin nähtäville kommentteineen. Jokainen voi helposti nähdä kuinka paljon koodia vaadittaisiin saman vuokaavion toteuttamiseen C-kielellä.

Flowcode siis itse generoi tehdystä vuokaaviosta seuraavan C-koodin:

```
//********************************************************************************
//**
//**  File name:     Z:\Programs\Matrix Multimedia\Flowcode V3\Examples\laskuri.c
//**  Generated by:  Flowcode v3.2.0.36
//**  Date:          Tuesday, May 10, 2010 16:07:40
//**  Licence:       Professional
//**  Registered to: Metropolia
//**
//**
//**  http://www.matrixmultimedia.com
//********************************************************************************

//Määrittää mikro-ohjaimen
#define P16F88
#define MX_EE
#define MX_EE_TYPE2
#define MX_EE_SIZE 256
#define MX_SPI
#define MX_SPI_B
#define MX_SPI_SDI 1
#define MX_SPI_SDO 2
#define MX_SPI_SCK 4
#define MX_UART
#define MX_UART_B
#define MX_UART_TX 5
#define MX_UART_RX 2
#define MX_I2C
#define MX_I2C_B
#define MX_I2C_SDA 1
#define MX_I2C_SCL 4
#define MX_PWM
#define MX_PWM_CNT 1
#define MX_PWM_TRIS1 trisb
#define MX_PWM_1 0
#define MX_PWM_TRIS1a trisb
#define MX_PWM_1a 3

//funktiot
#include <system.h>
#pragma CLOCK_FREQ 19660800

//Configuration data

//Internal functions
#include "Z:\Programs\Matrix Multimedia\Flowcode V3\FCD\internals.h"

//Makro-funktion kuvaukset

//Muuttujakuvaukset
```

```
short FCV_LASKE;

//LCDDisplay0: //Makro-funktion kuvaukset
void FCD_LCDDisplay0_GetDefines();
void FCD_LCDDisplay0_Start();
void FCD_LCDDisplay0_Clear();
void FCD_LCDDisplay0_PrintASCII(char Character);
void FCD_LCDDisplay0_Command(char in);
void FCD_LCDDisplay0_RawSend(char in, char mask);
void FCD_LCDDisplay0_Cursor(char x, char y);
void FCD_LCDDisplay0_PrintNumber(short Number);
void FCD_LCDDisplay0_PrintString(char* String, char MSZ_String);

//Supplementary defines


//Makro-toteutukset


//LCDDisplay0: //Makro-toteutukset


void FCD_LCDDisplay0_GetDefines()
{

}  //Dummy end of function to allow defines to be added correctly

//component connections
#define LCD_1443832_PORT    portb
#define LCD_1443832_TRIS    trisb
#define LCD_1443832_BIT0    0
#define LCD_1443832_BIT1    1
#define LCD_1443832_BIT2    2
#define LCD_1443832_BIT3    3
#define LCD_1443832_RS      4
#define LCD_1443832_E       5

#ifdef _BOOSTC
#define LCD_1443832_DELAY   delay_10us(10)
#endif
#ifdef _C2C_
#define LCD_1443832_DELAY   delay_us(100)
#endif
#ifndef LCD_1443832_DELAY
#define LCD_1443832_DELAY   delay_us(100)
#endif


//internal function prototypes
void LCD_1443832_RawSend(char nIn, char nMask);

//internal function implementations
void LCD_1443832_RawSend(char nIn, char nMask)
{
unsigned char pt;
```

```
unsigned char outVal;
outVal = LCD_1443832_PORT;
clear_bit(outVal, LCD_1443832_BIT0);
clear_bit(outVal, LCD_1443832_BIT1);
clear_bit(outVal, LCD_1443832_BIT2);
clear_bit(outVal, LCD_1443832_BIT3);
clear_bit(outVal, LCD_1443832_RS);
clear_bit(outVal, LCD_1443832_E);
pt = ((nIn >> 4) & 0x0f);
if (pt & 0x01)
set_bit(outVal, LCD_1443832_BIT0);
if (pt & 0x02)
set_bit(outVal, LCD_1443832_BIT1);
if (pt & 0x04)
set_bit(outVal, LCD_1443832_BIT2);
if (pt & 0x08)
set_bit(outVal, LCD_1443832_BIT3);
if (nMask)
set_bit(outVal, LCD_1443832_RS);
LCD_1443832_PORT = outVal;
set_bit (LCD_1443832_PORT, LCD_1443832_E);
LCD_1443832_DELAY;
clear_bit (LCD_1443832_PORT, LCD_1443832_E);
pt = (nIn & 0x0f);
LCD_1443832_DELAY;
outVal = LCD_1443832_PORT;
clear_bit(outVal, LCD_1443832_BIT0);
clear_bit(outVal, LCD_1443832_BIT1);
clear_bit(outVal, LCD_1443832_BIT2);
clear_bit(outVal, LCD_1443832_BIT3);
clear_bit(outVal, LCD_1443832_RS);
clear_bit(outVal, LCD_1443832_E);
if (pt & 0x01)
                set_bit(outVal, LCD_1443832_BIT0);
if (pt & 0x02)
                set_bit(outVal, LCD_1443832_BIT1);
if (pt & 0x04)
                set_bit(outVal, LCD_1443832_BIT2);
if (pt & 0x08)
                set_bit(outVal, LCD_1443832_BIT3);
if (nMask)
                set_bit(outVal, LCD_1443832_RS);
LCD_1443832_PORT = outVal;
LCD_1443832_DELAY;
set_bit (LCD_1443832_PORT, LCD_1443832_E);
LCD_1443832_DELAY;
clear_bit (LCD_1443832_PORT, LCD_1443832_E);
LCD_1443832_DELAY;
}

// Dummy function to close the defines section off
void LCD_1443832_Dummy_Function();
void LCD_1443832_Dummy_Function()
{
```

```
}

void FCD_LCDDisplay0_Start()
{

clear_bit(LCD_1443832_TRIS, LCD_1443832_BIT0);
clear_bit(LCD_1443832_TRIS, LCD_1443832_BIT1);
clear_bit(LCD_1443832_TRIS, LCD_1443832_BIT2);
clear_bit(LCD_1443832_TRIS, LCD_1443832_BIT3);
clear_bit(LCD_1443832_TRIS, LCD_1443832_RS);
clear_bit(LCD_1443832_TRIS, LCD_1443832_E);

delay_ms(12);

LCD_1443832_RawSend(0x33, 0);
delay_ms(2);
LCD_1443832_RawSend(0x32, 0);
delay_ms(2);
LCD_1443832_RawSend(0x2c, 0);
delay_ms(2);
LCD_1443832_RawSend(0x06, 0);
delay_ms(2);
LCD_1443832_RawSend(0x0c, 0);
delay_ms(2);

//clear the display
LCD_1443832_RawSend(0x01, 0);
delay_ms(2);
LCD_1443832_RawSend(0x02, 0);
delay_ms(2);

}

void FCD_LCDDisplay0_Clear()
{

LCD_1443832_RawSend(0x01, 0);
delay_ms(2);
LCD_1443832_RawSend(0x02, 0);
delay_ms(2);

}

void FCD_LCDDisplay0_PrintASCII(char Character)
{

LCD_1443832_RawSend(Character, 0x10);

}

void FCD_LCDDisplay0_Command(char in)
{

LCD_1443832_RawSend(in, 0);
delay_ms(2);
```

```
}

void FCD_LCDDisplay0_RawSend(char in, char mask)
{
//Error Reading Code For LCD-näyttö::Macro_RawSend

}

void FCD_LCDDisplay0_Cursor(char x, char y)
{

#if (2 == 1)
y=0x80;
#endif

#if (2 == 2)
                if (y==0)
                                y=0x80;
                else
                                y=0xc0;
#endif

#if (2 == 4)
                if (y==0)
                                y=0x80;
                else if (y==1)
                                y=0xc0;
                else if (y==2)
                                y=0x94;
                else
                                y=0xd4;
 #endif

LCD_1443832_RawSend(y+x, 0);
delay_ms(2);

}

void FCD_LCDDisplay0_PrintNumber(short Number)
{

short tmp_int;
char tmp_byte;
if (Number < 0)
{
LCD_1443832_RawSend('-', 0x10);
Number = 0 - Number;
}

tmp_int = Number;
if (Number >= 10000)
{
tmp_byte = tmp_int / 10000;
LCD_1443832_RawSend('0' + tmp_byte, 0x10);
```

```
while (tmp_byte > 0)
{
                tmp_int = tmp_int - 10000;
                tmp_byte--;
}
}
if (Number >= 1000)
{
tmp_byte = tmp_int / 1000;
LCD_1443832_RawSend('0' + tmp_byte, 0x10);

while (tmp_byte > 0)
{
tmp_int = tmp_int - 1000;
tmp_byte--;
}
}
if (Number >= 100)
{
tmp_byte = tmp_int / 100;
LCD_1443832_RawSend('0' + tmp_byte, 0x10);

while (tmp_byte > 0)
{
tmp_int = tmp_int - 100;
tmp_byte--;
}
}
if (Number >= 10)
{
tmp_byte = tmp_int / 10;
LCD_1443832_RawSend('0' + tmp_byte, 0x10);

while (tmp_byte > 0)
{
tmp_int = tmp_int - 10;
tmp_byte--;
}
}
LCD_1443832_RawSend('0' + tmp_int, 0x10);

}

void FCD_LCDDisplay0_PrintString(char* String, char MSZ_String)
{

char idx;
for (idx=0; idx<MSZ_String; idx++)
{
                if (String[idx]==0)
                {
                                break;
                }
                LCD_1443832_RawSend(String[idx], 0x10);
```

```
                }

}

//Supplementary implementations

void main()
{

//alustus
ansel = 0;
cmcon = 0x07;

//Keskeytyksen alustuskoodi
option_reg = 0xC0;

//Call Component Macro
//Call Component Macro: LCDDisplay(0)::Start
FCD_LCDDisplay0_Start();

//Call Component Macro
//Call Component Macro: LCDDisplay(0)::PrintString("Laskuri:")
FCD_LCDDisplay0_PrintString("Laskuri:",8);

//Silmukka
//Silmukka: While 1
while( 1 )
{
//Viive
//Viive: 10 ms
delay_ms(10);


//Laskutoimitus
//Laskutoimitus:
//  Laske = Laske + 1
FCV_LASKE =  FCV_LASKE + 1 ;


//Call Component Macro
//Call Component Macro: LCDDisplay(0)::Cursor(0, 1)
FCD_LCDDisplay0_Cursor(0, 1);


//Call Component Macro
//Call Component Macro: LCDDisplay(0)::PrintNumber(Laske)
FCD_LCDDisplay0_PrintNumber(FCV_LASKE);

}

mainendloop: goto mainendloop;
}
```