Open your mind. LUT.
Lappeenranta University of Technology

Moottorin paikkasäädön toteutus dsPIC-ohjaimella
Position control of a motor with dsPIC-controller

Jouni Vuojolainen

Kandidaatintyö
LUT Energia
Sähkötekniikan koulutusohjelma

22.04.2014

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknillinen tiedekunta
Sähkötekniikan koulutusohjelma

Jouni Vuojolainen
Moottorin paikkasäädön toteutus dsPIC-ohjaimella

2014

Kandidaatintyö.
24 s.

Tarkastaja: TkT Tuomo Lindh

Työssä suunniteltiin nostinsimulaattorin z-akselin suuntaiselle moottorille paikkasäätö ja sen ohjaus CAN-väylää pitkin. Työ toteutettiin projektiryhmässä, jossa eri henkilöt vastasivat eri osa-alueista. CAN-kommunikointi saatiin toimimaan ja lisäksi moottorilta pystyttiin lukemaan paikkatieto talteen. Moottorin säätö jäi vielä puuttumaan.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology
Degree Programme in Electrical Engineering

Jouni Vuojolainen
Position control of a motor with dsPIC-controller

2014

Bachelor's Thesis.
24 p.

Examiner: D.Sc Tuomo Lindh

In this project the idea was to design to an existing crane simulator a z-axis sided motor and design control for that motor and communication with CAN bus. Project was made in a project group, where different people were responsible for different tasks. CAN communication was finished and motor position could be read. Motor control was unfinished.

SISÄLLYSLUETTELO

KÄYTETYT MERKINNÄT JA LYHENTEET

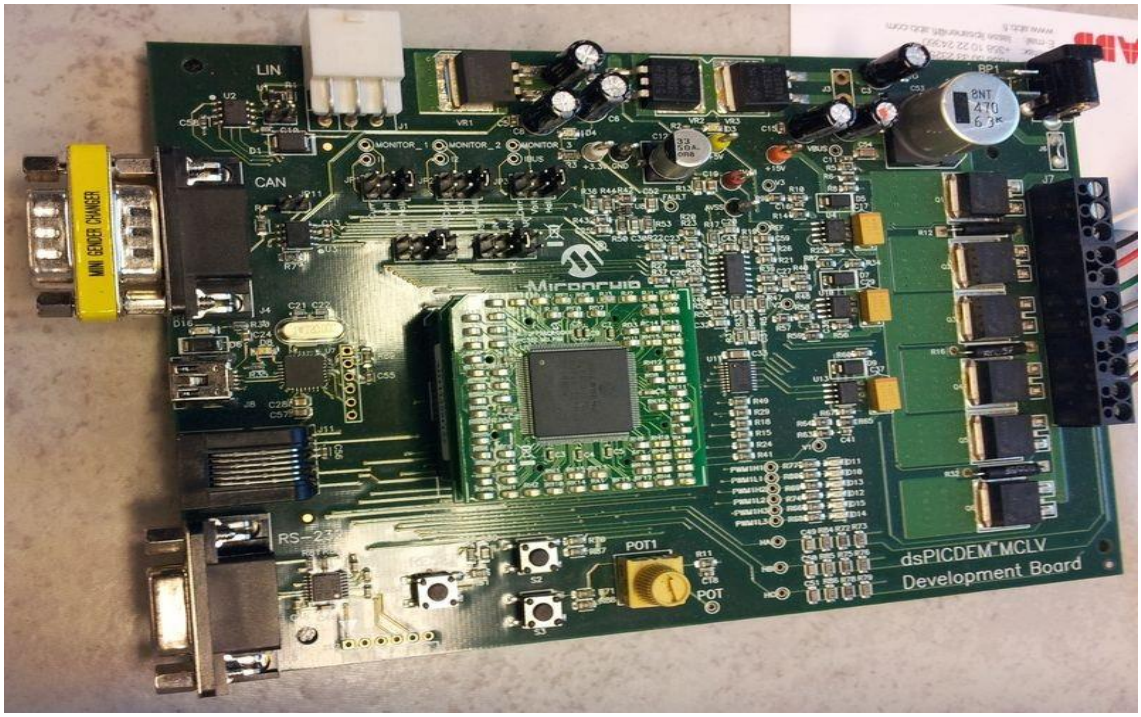
CAN	Controller Area Network, teollisuudessa hyödynnetty automaatioväylä
BLDC	Brushless DC, harjaton DC moottori
PLC	Programmable Logic Controller, ohjelmoitava logiikka
ANSI C	C- kielen standardi
PWM	(Pulse Width Modulation, pulssinleveysmodulaatio)
MIPS	(Million Instructions per second, miljoonaa käskyä sekunnissa)
QEI	(Quadrature Encoder Interface, paikanmittauksen pulssienkooderi)
dsPIC	(dsPIC33EP512MU814 mikroprosessori)
RAM	(Random-access memory, keskusmuisti)
DMA	(Direct Memory Access, keskusmuistin hallinta ilman prosessoria)

1. JOHDANTO

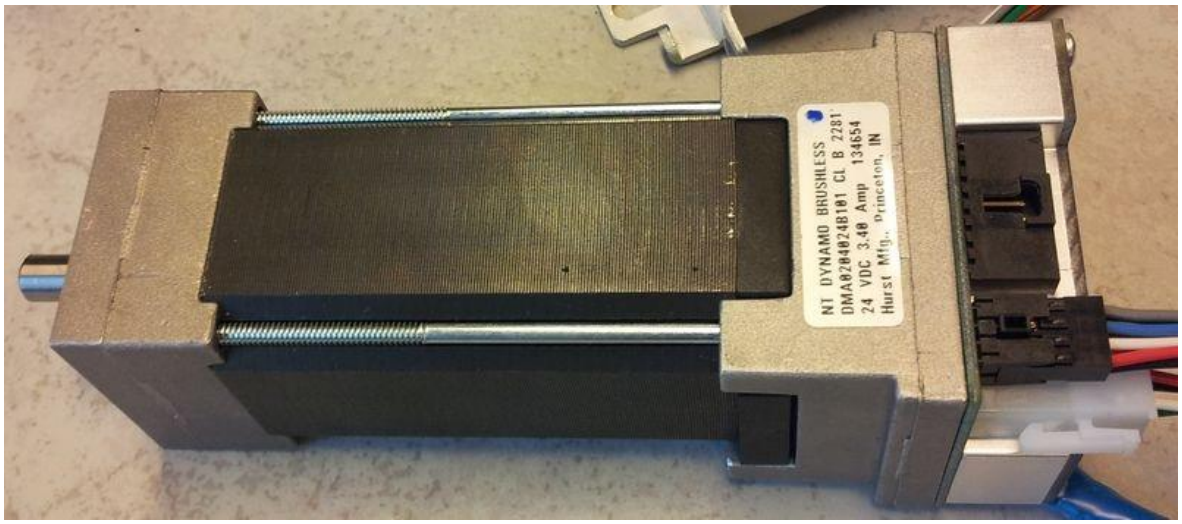
Tässä työssä suunnitellaan nostinsimulaattoriin paikkasäätö Z-akselin suuntaiselle moottorille ja sen ohjaus CAN (Controller Area Network, teollisuudessa käytetty automaatioväylä)-väylää pitkin. Laitteistona käytetään dsPICDEM MCLV kehitysalustaa, johon on kytketty dsPIC33EP512MU814 mikroprosessori. Moottorina käytettiin pientä 24 V jännitteellä toimivaa BLDC (Brushless DC, harjaton DC)-moottoria, josta löytyvän pulssienkooderin avulla paikkasäätö on mahdollista toteuttaa hyvin tarkasti. Työ rakennetaan osaksi oppilaskäyttöön ja tutkimukseen tulevaa laitteistoa, jonka tarkoituksena on mallintaa torninosturia. Työ toteutettiin projektiryhmässä, jossa vastasin CAN-väylästä, pulssienkooderista ja dsPIC:stä yleisesti.

2. LAITTEISTON KUVAUS

Mikroprosessori dsPIC33EP512MU814 on kytketty dsPICDEM MCLV kehitysalustaan, joka syöttää mikroprosessorille ja käytetylle moottorille tarvittavat käyttöjännitteet ja mahdollistaa mikroprosessorin toimintojen ja erilaisten kommunikointiväylien hyödyntämisen ilman juotoksia tai ylimääräisiä johdotuksia. Mikroprosessorin ohjelmointi tapahtuu C- kielellä, joka on täysin ANSI C (C kielen standardi) -yhteensopiva. Sitä on kuitenkin laajennettu paremman yhteensopivuuden takia (Microchip C30). Kehitysalusta on esitetty kuvassa 2.1 ja kuvassa 2.2 käytetty moottori.



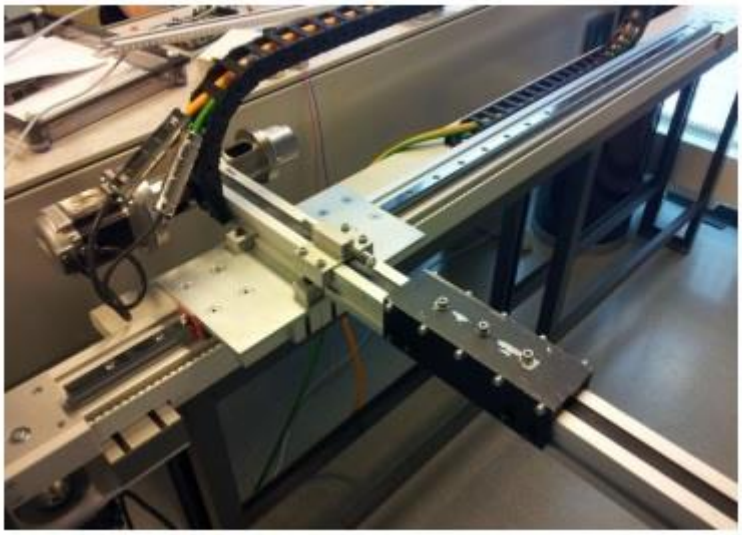
Kuva 2.1 käytetty kehitysalusta ja mikroprosessori. Mikroprosessori on kuvan keskellä näkyvä musta siru (Loppuraportti 2013).



Kuva 2.2 Työssä käytetty BLDC- moottori (Loppuraportti 2013).

Laitteisto kytkeytyy jo olemassa olevaan nostinsimulaattoriin siten, että moottorin pulssienkooderilta saatu paikkatieto lähetetään CAN-väylää pitkin PLC:lle. PLC lähettää CAN- väylää pitkin moottorin nopeusohjeen ja mikroprosessori hoitaa nopeuden säädön ja ohjaa tämän perusteella moottoria PWM (Pulse Width Modulation, pulssinleveysmodulaatio)-signaalien avulla.

Järjestelmä rakentuu kokonaisuudessaan kaksiakselimanipulaattorin päälle. Manipulaattoreihin kytkettyjen lineaarikuljettimien avulla on toteutettu nostinsimulaattorin ohjaus x- ja y-suunnassa. Lineaarikuljettimia ohjataan servomoottoreilla ja taajuusmuuttajilla tyypiltään ACSM1, joka on ABB:n valmistama. Taajuusmuuttajat kytkeytyvät EtherCAT (Ethernet for Control and Automation Technology, automaatiolaitteissa käytetty automaatioväylä)-väylää pitkin ABB:n valmistamaan AC500-logiikkaan. XY-suuntainen paikkasäätö tapahtuu AC500-logiikan avulla. AC500 antaa taajuusmuuttajille nopeusohjeen ja taajuusmuuttajat antavat logiikalle paikkatiedon. Nopeussäätö on toteutettu yksinkertaisella PI-säätimellä. Kuvassa 2.3 on esitetty manipulaattoriin liitetyt lineaarikuljettimet.

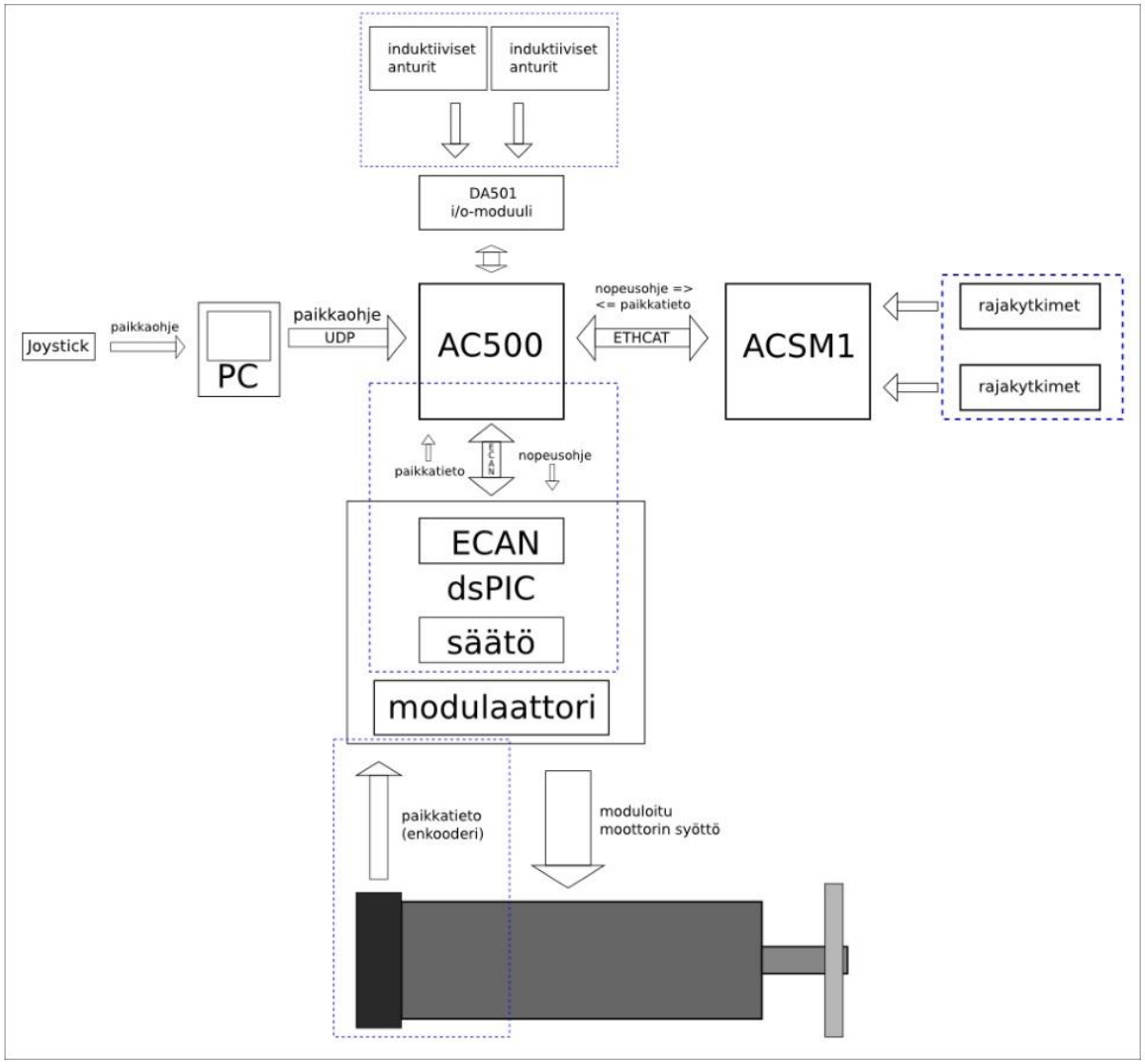


Kuva 2.3 Manipulaattoriin liitetyt lineaarikuljettimet (Loppuraportti 2013).

Logiikka on edelleen kytketty tietokoneeseen Ethernet-kaapelin välityksellä. Tietokoneella ajetaan MPLAB IDE -kehitysympäristöä, joka huolehtii dsPIC:n ohjelman ohjelmoinnista. Lisäksi ajetaan ABB:n Control Builder Plus -ohjelmaa, joka huolehtii AC500-logiikan määrittelystä ja CodeSys–ohjelmaa, joka huolehtii logiikan ohjelmoinnista. Tietokoneeseen kytketyn joystickin avulla on mahdollista ohjata koko nostinsimulaattoria XYZ-suunnassa. Logiikka ja siihen liitetyt moduulit näkyvät kuvasta 2.4.

Kuva 2.4 AC500-logiikka,Ethercat-,CAN- ja DA501-moduulit (Loppuraportti 2013).

AC500-logiikkaan on lisäksi kytketty DA501 I/O -moduulin avulla induktiiviset lähestymisanturit. Lähestymisanturien avulla rajataan lineaarikuljettimen liikealuetta eli sen täytyy pysyä kahden anturin välissä. Lisävarmistuksena järjestelmässä on myös mekaaniset rajakytkimet, jotka on kytketty ACSM1-digitaalituloon. Rajakytkimen aktivoituessa taajuusmuuttaja menee hätäseis-tilaan ja pysähtyy noin 0.1 s aikana. Koko laitteiston lohkokaavio on esitetty kuvassa 2.5.



Kuva 2.5 Laitteiston lohkokaavio (Loppuraportti 2013).

3. KÄYTETTY MIKROPROSESSORI

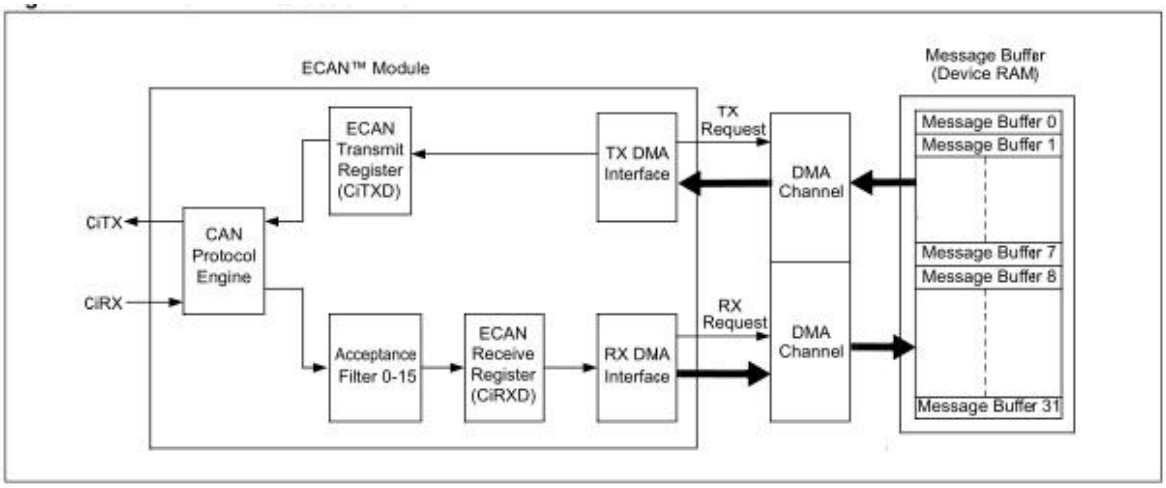
Työssä käytetty mikroprosessori dsPIC33EP512MU814 on Microchip:n valmistama. Arkkitehtuuriltaan se on 16-bittinen ja prosessorin maksinopeus on 70 MIPS (Million Instructions per second, miljoonaa käskyä sekunnissa). Arkkitehtuurina on modifioitu Harvard arkkitehtuuri, eli datalle ja käskyille on omat väylänsä. Lisäksi on lisätty tukea digitaaliseen signaalien käsittelyyn. Ohjelmamuistia on 512 kilotavua, joten laitteella on mahdollista toteuttaa melko pitkiäkin ohjelmia. Kommunikointiin on mahdollista käyttää esim. CAN-väylää. Mikroprosessorissa on myös QEI (Quadrature Encoder Interface, paikanmittauksen pulssienkooderi)-moduuli, jonka avulla tarkka paikan mittaus on mahdollista. PWM-generaattoreita löytyy yhteensä 7 kappaletta (dsPIC datasheet).

Lisäksi mikroprosessorista löytyy myös muitakin ominaisuuksia, mutta niitä ei käsitellä tässä työssä. Mikroprosessori valittiin tähän työhön, koska sillä voidaan mitata paikka tarkasti ja kommunikoida CAN-väylän välityksellä.

3.1 CAN-kommunikointi

CAN on sarjamuotoinen protokolla, joka tukee hyvin reaaliaikaisuutta. Alun perin CAN-väylää käytettiin autoissa liittämään sensorit, moottorin elektroniikka jne. yhteen yhdellä väylällä. CAN mahdollistaa maksimissaan 1 Mbit/s siirtonopeuden. Autoteollisuudesta CAN-väylä on levinnyt myös muuhun teollisuuteen. CAN-standardi käsittää kaksi eri osaa 2.0A ja 2.0B. Näiden eroina on se, että 2.0A standardissa käytetään 11-bittistä laiteosoitetta ja 2.0B standardissa käytetään 29-bittistä laiteosoitetta. 2.0B standardia tukevat laitteet voivat kommunikoida myös 2.0A standardia tukevien laitteiden kanssa, sillä 29-bittisen laiteosoitteen ensimmäiset 11-bittiä ovat samat kuin 2.0A standardin laiteosoite (Bosch CAN). Aina ei kuitenkaan verkkoon voi kytkeä niin montaa laitetta kuin laiteosoitteita olisi olemassa, sillä käytetyt CAN-lähettimet ja/tai -vastaanottimet voivat tuoda omia rajoitteita laitteiden maksimimääriin esimerkiksi dsPICDEM MCLV -kehitysalustalta löytyvä MCP2551 CAN transceiver tukee maksimissaan 112 laitetta samassa väylässä. Tässä työssä käytettiin 2.0A standardia johtuen kehitysalustan 112 laitteen rajoituksesta. dsPIC:n ECAN-moduulin avulla toteutettiin CAN-kommunikointi. Kommunikoinnin nopeutena on käytetty 250kb/s.

ECAN-moduuli mahdollistaa CAN-kommunikoinnin toteuttamisen dsPIC mikroprosessorissa. Moduuli tukee sekä 2.0A, että 2.0B standardia. Nopeudet 1Mbit/s asti ovat tuettuna. Moduuliin voi käyttää maksimissaan 32 viestibufferia, joista kaikkia voidaan käyttää viestien vastaanottamiseen, mutta vain 8 voidaan käyttää viestien lähetykseen. Viestibufferit sijaitsevat laitteen RAM (Random-access memory, keskusmuisti) -muistissa. Viestibufferien hallinta tapahtuu DMA (Direct Memory Access, keskusmuistin hallinta ilman prosessoria):n avulla, joka siirtää datan moduulilta laitteen keskusmuistiin (dsPIC ECAN). Tässä työssä käytettiin neljää viestibufferia, joista yksi vastaanottoon ja yksi lähetykseen. Kaaviokuva ECAN-moduulin ja DMA:n toiminnasta on esitetty kuvassa 3.1.1.



Kuva 3.1.1 kaaviokuva ECAN moduulista ja DMA:n toiminnasta. DMA siis välittää viestit ECAN moduulin ja laitteen keskusmuistin välillä (dsPIC ECAN).

Kuvasta 3.1.1 näkyvä CAN-protokollamoottori huolehtii ulkoisen CAN-väylää ajavan piirin kanssa kommunikoinnista, joka tässä tapauksessa olisi dsPICDEM MCLV kehitysalustan MCP2551 CAN transceiver. Protokollamoottori ohjaa lähetys- ja vastaanotto DMA-rajapintoja, jotka siirtävät DMA-kanavia pitkin viestejä joko keskusmuistista protokollamoottorille tai protokollamoottorilta keskusmuistiin.

ECAN-moduulia hallitaan erilaisten rekisterien avulla. Lisäksi hallintaa varten kirjoitettiin pari C-kielistä funktiota. ECAN-moduuliin liittyvät tärkeimmät rekisterit ja C-kieliset funktiot on esitetty taulukossa 3.1.2.

Tauluko 3.1.2 ECAN-moduuliin liittyvät tärkeimmät rekisterit ja niiden kuvaukset ja C-kieliset funktiot ja niiden kuvaus.

Rekisterin nimi	**Rekisterin kuvaus**
C1CFG1	siirtonopeuden määritysrekisteri 1
C1CFG2	siirtonopeuden määritysrekisteri 2
C1CTRL1	ECAN-moduulin hallintarekisteri 1
C1CTRL2	ECAN-moduulin hallintarekisteri 2
C1FCTRL	FIFO:n hallintarekisteri
C1mnCON	Lähetetys/vastaanottobufferin hallintarekisteri
C-kielinen funktio	**Kuvaus funktiosta**
ecan1WriteRxAcptFilter()	Vastaanottofiltterin määritys
ecan1WriteRxAcptMask()	Vastaanottomaskin määritys
rxECAN1()	CAN-viestien vastaanottofunktio
ecan1WriteMessage()	CAN-viestien lähetysfunktio

On olemassa kaksi eri tapaa kirjoittaa rekisteriin. Koko rekisteriin voidaan kirjoittaa ihan normaalilla C-kielisellä sijoituksella esimerkiksi esim_rekisteri = 0;. Toinen tapa on kirjoittaa rekisterissä sijaitsevia arvoja yksi kerrallaan. Käytetään esimerkkinä kuvassa 3.1.3 näkyvää rekisteriä CiFMSKSEL2.

Register 21-9: CiFMSKSEL2: ECAN Filter 15-8 Mask Selection Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
F15MSK<1:0>		F14MSK<1:0>		F13MSK<1:0>		F12MSK<1:0>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
F11MSK<1:0>		F10MSK<1:0>		F9MSK<1:0>		F8MSK<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-14 **F15MSK<1:0>:** Mask Source for Filter 15 bits
11 = Reserved; do not use
10 = Acceptance Mask 2 registers contain mask
01 = Acceptance Mask 1 registers contain mask
00 = Acceptance Mask 0 registers contain mask

bit 13-12 **F14MSK<1:0>:** Mask Source for Filter 14 bits (same values as bits 15-14)

bit 11-10 **F13MSK<1:0>:** Mask Source for Filter 13 bits (same values as bits 15-14)

bit 9-8 **F12MSK<1:0>:** Mask Source for Filter 12 bits (same values as bits 15-14)

bit 7-6 **F11MSK<1:0>:** Mask Source for Filter 11 bits (same values as bits 15-14)

bit 5-4 **F10MSK<1:0>:** Mask Source for Filter 10 bits (same values as bits 15-14)

bit 3-2 **F9MSK<1:0>:** Mask Source for Filter 9 bits (same values as bits 15-14)

bit 1-0 **F8MSK<1:0>:** Mask Source for Filter 8 bits (same values as bits 15-14)

Kuva 3.1.3 Yksittäisen muuttujan arvon muuttaminen rekisterissä (dsPIC ECAN).

Jos haluaisimme muuttaa muuttujan F15MSK arvoa se tapahtuisi näin CiFMSKSEL2bits.F15MSK=0; eli yleisesti rekisterin_nimibits.muuttujan_nimi=0;. Rekisterin arvon lukeminen on myös mahdollista muuttuja tai koko rekisteri kerrallaan.

ECAN moduulin käyttöä varten tulee tarvittaviin rekistereihin kirjoittaa oikeat arvot. Aluksi asetetaan C1CTRL1-rekisterin REQOP bitti arvoon 4. Tämä on konfigurointimoodi, jota tarvitaan C1CFG1- ja C1CFG2-rekisterien muokkaukseen. Kuvassa 3.1.4 on esitetty C1CTRL1-rekisteri.

Register 21-23: CiCTRL1: ECAN Control Register 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
—	—	CSIDL	ABAT	CANCKS	REQOP<2:0>		
bit 15							bit 8

R-1	R-0	R-0	U-0	R/W-0	U-0	U-0	R/W-0
OPMODE<2:0>			—	CANCAP	—	—	WIN
bit 7							bit 0

Legend:	r = Reserved		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **CSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 12 **ABAT:** Abort All Pending Transmissions bit
1 = Signal all transmit buffers to abort transmission
0 = Module will clear this bit when all transmissions are aborted

bit 11 **CANCKS:** ECAN Module Clock (FCAN) Source Select bit
1 = FCAN is equal to 2 * FP
0 = FCAN is equal to FP

bit 10-8 **REQOP<2:0>:** Request Operation Mode bits
111 = Set Listen All Messages mode
110 = Reserved; do not use
101 = Reserved; do not use
100 = Set Configuration mode
011 = Set Listen-Only mode
010 = Set Loopback mode
001 = Set Disable mode
000 = Set Normal Operation mode

bit 7-5 **OPMODE<2:0>:** Operation Mode bits
111 = Module is in Listen All Messages mode
110 = Reserved; do not use
101 = Reserved; do not use
100 = Module is in Configuration mode
011 = Module is in Listen-Only mode
010 = Module is in Loopback mode
001 = Module is in Disable mode
000 = Module is in Normal Operation mode

bit 4 **Unimplemented:** Read as '0'

bit 3 **CANCAP:** CAN Message Receive Timer Capture Event Enable bit
1 = Enable input capture based on CAN message receive
0 = Disable CAN capture

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **WIN:** SFR Map Window Select bit
1 = Use filter window
0 = Use buffer window

Kuva 3.1.4 C1CTRL1-rekisteri. Asetetaan REQOP bitin arvoksi 4(100_2) (dsPIC ECAN).

Tämän jälkeen konfiguroidaan C1CFG1- ja C1CFG2-rekisterit, joilla säädetään ECAN-moduulin nopeus. Seuraavaksi valitaan C1FCTRL-rekisterin avulla 4 viestibufferia dsPIC:n keskusmuistiin komennolla C1FCTRLbits.DMABS=0; . Kuvassa 3.1.5 on esitetty C1FCTRL-rekisteri.

Register 21-18: CiFCTRL: ECAN FIFO Control Register

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
DMABS<2:0>			—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FSA<4:0>				
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-13 **DMABS<2:0>:** Message Buffer Size bits
111 = Reserved; do not use
110 = 32 buffers in device RAM
101 = 24 buffers in device RAM
100 = 16 buffers in device RAM
011 = 12 buffers in device RAM
010 = Eight buffers in device RAM
001 = Six buffers in device RAM
000 = Four buffers in device RAM

bit 12-5 **Unimplemented:** Read as '0'

bit 4-0 **FSA<4:0>:** FIFO Start Area bits
11111 = Read buffer RB31
11110 = Read buffer RB30
•
•
•
00010 = TX/RX buffer TRB2
00001 = TX/RX buffer TRB1
00000 = TX/RX buffer TRB0

Kuva 3.1.5 C1FCTRL-rekisteri. Asetetaan DMABS muuttujan arvoksi 0, joka tarkoittaa 4 viestibufferin käyttämistä(dsPIC ECAN).

dsPIC on konfiguroitu siten, että se ottaa vastaan vain laiteosoitteella 2 olevat viestit vastaan. Tämä on toteutettu Microchipin CE427 koodiesimerkistä (MchipCE427) ja taulukosta 3.1.2 löytyvien esimerkkifunktioiden ecan1WriteRxAcptFilter() ja ecan1WriteRxAcptMask() avulla.

Nyt kun tarvittavat konfiguroinnit on suoritettu voimme siirtyä normaalimoodiin asettamalla C1CTRL1-rekisterin REQOP bitin arvoon 0 (rekisteri esitetty kuvassa 3.1.4). Nyt C1CFG1- ja C1CFG2-rekisterien arvoja ei voida muokata.

Tämän jälkeen konfiguroidaan viestibufferi 0 lähetysbufferiksi ja viestibufferi 1 vastaanottobufferiksi ja asetetaan molemmille buffereille sama prioriteettiarvo. Tämä onnistuu komennoilla C1TR01CONbits.TXEN0=1;, C1TR01CONbits.TX0PRI=0b11;, C1TR01CONbits.TXEN1=0; ja C1TR01CONbits.TX1PRI=0b11;. Esimerkkikuva viestibufferin hallinnasta on esitetty kuvassa 3.1.6.

Register 21-25: CiTRmnCON: ECAN TX/RX Buffer m Control Register (m = 0,2,4,6; n = 1,3,5,7)

R/W-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
TXENn	TXABTn	TXLARBn	TXERRn	TXREQn	RTRENn	TXnPRI<1:0>	
bit 15							bit 8

R/W-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
TXENm	TXABTm(1)	TXLARBm(1)	TXERRm(1)	TXREQm	RTRENm	TXmPRI<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8 See definition for bits 7-0, Controls Buffer n

bit 7 **TXENm:** TX/RX Buffer Selection bit
1 = Buffer TRBn is a transmit buffer
0 = Buffer TRBn is a receive buffer

bit 6 **TXABTm:** Message Aborted bit(1)
1 = Message was aborted
0 = Message completed transmission successfully

bit 5 **TXLARBm:** Message Lost Arbitration bit(1)
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERRm:** Error Detected During Transmission bit(1)
1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent

bit 3 **TXREQm:** Message Send Request bit
1 = Requests that a message be sent. Once the message is successfully sent, the bit is automatically cleared.
0 = Setting this bit to '0' while a message is being sent, aborts the message transmission.

bit 2 **RTRENm:** Auto-Remote Transmit Enable bit
1 = When a remote frame is received, TXREQ bit will automatically set
0 = When a remote frame is received, TXREQ bit will be unaffected

bit 1-0 **TXmPRI<1:0>:** Message Transmission Priority bits
11 = Highest message priority
10 = High intermediate message priority
01 = Low intermediate message priority
00 = Lowest message priority

Note 1: This bit is cleared when the TXREQ bit is set.

Kuva 3.1.6 Esimerkkikuva viestibufferin hallintarekisteristä. Muutetaan TXEN0 arvoksi 1, TXEN1 arvoksi 0 ja TX0PRI/TX1PRI arvoiksi $3(11_2)$.

Nyt tarvitsee enää kytkeä ECAN-moduuli dsPIC:n oikeisiin pinneihin. Tämä onnistuu komennoilla RPINR26bits.C1RXR = 100; ja RPOR9bits.RP101R = 14;. Nyt ECAN-moduulia voidaan käyttää CAN viestien lähetykseen ja vastaanottoon.

CAN-viestien vastaanotto tapahtuu automaattisesti. Kun ECAN-moduuli vastaanottaa viestin, se laukaisee keskeytyksen. Tämä keskeytys tarkistaa haluaako moduuli lähettää vai vastaanottaa CAN-viestin. Nyt kun halutaan vastaanottaa viesti, suoritetaan taulukosta 3.1.2 rxECAN1() funktio, joka tallentaa viestin (tässä tapauksessa haluttu nopeus) haluttu_nopeus muuttujaan.

CAN-viestin lähetys onnistuu seuraavalla funktiolla, joka löytyy taulukosta 3.1.2 ja joka on toteutettu laajentamalla Microchipin CE427 koodiesimerkistä (MchipCE427) ecan1WriteMessage(unsigned long long lahteva, short can_id_lahteva). Funktiolle siis annetaan syötteenä haluttu lähetettävä data, joka tässä tapauksessa on unsigned long long muodossa ja lisäksi can_id_lahteva joka kuvaa CAN- kommunikointiin liittyvää laiteosoitetta. Viestin lähetyksin laukaisee ECAN-moduulin keskeytyksen, mutta nyt ei ole tarvetta ohjata ohjelman suoritusta keskeytyksessä.

3.2 Pulssienkooderi ja paikan mittaus

dsPIC:n QEI-moduulin avulla on mahdollista toteuttaa tarkka paikan- ja nopeudenmittaus mikäli käytetystä moottorista löytyy pulssienkooderi. QEI-moduulin avulla saadaan paikkatieto 32 bitin tarkkuudella ja nopeus 16-bitin tarkkuudella (dsPIC QEI).

Moduulin toiminta perustuu neljään signaalin QEA, QEB, INDX ja HOME, joista INDX-signaalia käytetään nollaamaan paikkalaskuri eli yleensä kun moottori on pyörähtänyt yhden kierroksen. Tässä työssä ei kuitenkaan hyödynnetä INDX-signaalia. HOME-signaalin avulla on mahdollista vaikka ajaa moottori tiettyyn alkutilaan, jos sellaiseen on tarvetta. Tässä työssä ei kuitenkaan kotipaikkaa käytetä. QEA- ja QEB-signaaleja käytetään varsinaiseen paikanmittaukseen. QEA- ja QEB-signaalit ovat tyypillisesti 90° eri vaiheessa. QEA- ja QEB–signaalien nykyisten ja edellisten tilojen avulla on mahdollista päätellä liikkuiko moottori myötä- vai vastapäivään vai pysyikö se paikoillaan (dsPIC QEI). Taulukossa 3.2.1 on esitetty kuinka eri edelliset ja nykyiset tilat vaikuttavat paikanmittaukseen.

Taulukko 3.2.1 QEA- ja QEB-signaalien nykyisten ja edellisten tilojen vaikutus paikkalaskuriin. Count up tarkoittaa laskurin arvon kasvattamista yhdellä, Count down vähentämistä yhdellä, No count or direction change ja Invalid state change, ignore eivät vaikuta laskurin arvoihin (dsPIC QEI).

Current Quadrature State		Previous Quadrature State		Action
QA	QB	QA	QB	
1	1	1	1	No count or direction change
1	1	1	0	Count up
1	1	0	1	Count down
1	1	0	0	Invalid state change, ignore
1	0	1	1	Count down
1	0	1	0	No count or direction change
1	0	0	1	Invalid state change, ignore
1	0	0	0	Count up
0	1	1	1	Count up
0	1	1	0	Invalid state change, ignore
0	1	0	1	No count or direction change
0	1	0	0	Count down
0	0	1	1	Invalid state change, ignore
0	0	1	0	Count down
0	0	0	1	Count up
0	0	0	0	No count or direction change

Myös QEI-moduulia hallitaan erilaisten rekisterien avulla. Hallinnan helpottamiseksi on myös kirjoitettu C-kielinen funktio. Rekisterit ja niiden kuvaukset ja C-kielinen funktio on esitetty taulukossa 3.2.2.

Tauluko 3.2.2 QEI-moduuliin liittyvät tärkeimmät rekisterit ja niiden kuvaukset ja C-kielisen funktion kuvaus.

Rekisterin nimi	**Rekisterin kuvaus**
QEI1CON	QEI-moduulin hallintarekisteri
QEI1IOC	QEI-moduulin I/O hallintarekisteri
QEI1STAT	QEI-moduulin tilarekisteri
POS1CNTH	Paikkalaskurin ylimmät 16-bittiä
POS1CNTL	Paikkalaskurin alimmat 16-bittiä
POS1HLD	Paikkalaskurin 16-bittinen pitorekisteri
VEL1CNT	Nopeuslaskurirekisteri
C-kielinen funktio	**Kuvaus funktiosta**
lue_QEI_paikka()	Lukee paikkatiedon QEI-moduulilta 32-bittisenä

Jotta QEI-moduulia voitaisiin käyttää, täytyy seuraaviin taulukosta 3.2.2 löytyviin rekistereihin kirjoittaa tarvittavat arvot: QEI1CON, QEI1IOC, QEI1STAT. Rekisterien avulla siis valitaan QEI-moduulin toimintamoodi, kellosignaalin jako yms. alustukset. Lisäksi QEI-moduuli tarvitsee kytkeä dsPIC:n oikeisiin pinneihin.

dsPIC:n rekisterit ovat 16-bittisiä, joten 32-bittinen paikkatieto sijaitsee kahdessa eri rekisterissä. Tämän takia paikkatiedon lukeminen ja mahdollinen kirjoittaminen tulee hoitaa seuraavalla tavalla. Kun paikkatieto halutaan lukea, luetaan ensiksi rekisteri POS1CNTL, jossa sijaitsee 32-bittisen paikkatiedon alimmat 16-bittiä. Tämä saa aikaan POS1CNTH rekisterin arvon kopioinnin POS1HLD-rekisteriin. Luetaan seuraavaksi POS1HLD-rekisteristä ylimmät 16-bittiä. Tätä tarvitaan sen takia, että lukuoperaatiolla saataisiin oikea 32-bittinen arvo. Jos halutaan kirjoittaa paikkatieto rekistereihin, tulee ensiksi kirjoittaa ylimmät 16-bittiä POS1HLD-rekisteriin. Tämän jälkeen, kun kirjoitetaan alimmat 16-bittiä POS1CNTL-rekisteriin, samalla kellojaksolla automaattisesti siirretään POS1HLD-rekisterin sisältö POS1CNTH-rekisteriin (dsPIC QEI). Tässä työssä ei kuitenkaan tarvita paikkatiedon kirjoitusta rekistereihin.

Paikkatiedon lukemista varten on tehty C-kielinen taulukosta 3.2.2 löytyvä funktio unsigned long lue_QEI_paikka(). Funktio lukee edellä mainitun ohjeen mukaisesti rekisterien arvot ja bittisiirron avulla yhdistää rekisterien arvot yhdeksi unsigned long tyyppiseksi arvoksi.

QEI-moduulin avulla on lisäksi mahdollista tehdä nopeusmittauksia. Nopeustieto sijaitsee rekisterissä VEL1CNT. Rekisterin lukeminen aiheuttaa rekisterin resetoinnin. Kun rekisteriä luetaan 1-4kHz taajuudella, on mahdollista arvioida nopeutta hyvin tarkasti (dsPIC QEI).

3.3 Nopeuden säätö

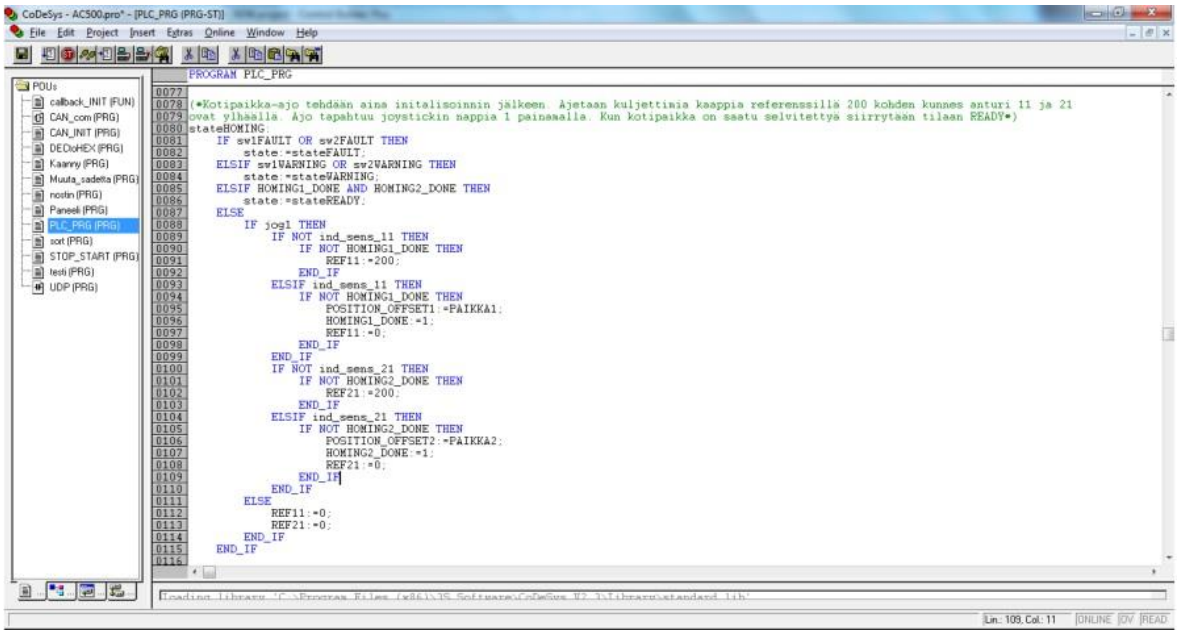
Microchipin AN1078 application noten(MchipAN1078) päälle rakennetun ohjelman avulla tarkoituksena oli muuttaa olemassa oleva anturiton vektorisäätö pelkäksi nopeussäädöksi. Enkooderin avulla oli tarkoitus lukea paikka ja lähettää paikkatieto PLC:lle ja PLC:ltä tulevan nopeusohjeen avulla toteuttaa moottorin nopeussäätö. Aluksi kokeiltiin viedä vektorisäädön estimoiduksi nopeudeksi enkooderilta saatu mitattu nopeus. Tätä ei kuitenkaan saatu toimimaan johtuen nopeusarvon heilahteluista ja olisi tarvittu jonkinlaista nopeuden arvon suodatusta. Lisäksi valmiissa ohjelmassa on välillä käytetty Q15-formaattia ja välillä tavallisia liukulukuja, joten emme saaneet selville minne ja missä muodossa mitattu nopeus tulisi ohjelmassa laittaa.

Säätöä kokeiltiin toteuttaa myös d-/q-irtikytketyllä säädöllä. Säätö huomioi roottorin kulmanopeuden, kyseisen suunnan induktanssin ja virran tulon ja kestomagneettisen käämivuon. Tämäkään säätötapa ei toiminut johtuen mahdollisesti moottorimallin arvojen epätarkkuudesta tai mahdollisesti myös Q15-formaatin ja liukulukujen käyttämisen samassa ohjelmassa.

Moottorin säätö jätettiin siis AN1078 application noten mukaiseksi. Tämä säätötapa on kuitenkin todella huono sillä anturiton vektorisäätö vaatii tietyn minimi pyörimisnopeuden, jonka jälkeen saadaan vasta tarkkoja estimaatteja säätöä varten. Moottoria ei siis ole mahdollista pyörittää pienillä nopeuksilla tai pieniä sykäyksiä, joten tältä osin ohjelma jäi hieman vajavaiseksi.

4. KOTIPAIKKA-AJO

Ilman kotipaikka-ajoa on kuljettimet pitänyt asettaa käsineen määrätyille paikoille, jotta systeemin paikanmittaus toimisi oikein. Kotipaikka-ajon tarkoituksena on saada kuljettimet aina asettumaan tiettyyn nollapisteeseen. Kotipaikka-ajossa käytetään apuna induktiivisia lähestymisantureita, jotka on sijoitettu lineaarikuljettimien molempiin päihin. Kotipaikka-ajo tapahtuu siten, että molempia kuljettimia ajetaan vakionopeudella kaappiin päin ja tämän avulla saadaan x- ja y-suunnassa paikalle tietty offset arvo. Kotipaikka-ajon toteuttava koodi on esitetty kuvassa 4.1.



Kuva 4.1 Kotipaikka-ajon toteuttava koodi.

5. YHTEENVETO / JOHTOPÄÄTÖKSET

Työn tarkoituksena oli lisätä jo olemassa olevaan nostinsimulaattoriin z-akselin suuntainen moottori ja tälle moottorille paikkasäätö. Kaikkiin osatavoitteisiin ei päästy, mutta jotkin pystyttiin toteuttamaan. Kommunikointi CAN-väylää pitkin saatiin toimimaan PLC:n ja dsPIC:n välillä. dsPIC:n enkooderilta saatiin myös paikkatieto talteen, jonka avulla yritettiin tehdä paikkasäätö järjestelmään. Kotipaikka-ajo saatiin myös toimimaan, eli kuljettimia ei tarvitse enää asettaa käsin määrätyille paikoille. Nopeussäätöä ei kuitenkaan saatu toimimaan kunnolla. Alkuperäisenä tavoitteena oli saada myös nopeussäätö ja paikkaohjaus myös toimimaan, mutta ajanpuutteen vuoksi niitä ei saatu valmiiksi.

Mahdollisia jatkotutkimus- /kehityskohteita olisi ainakin moottorin säädön toteuttaminen toimimaan. Lisäksi ohjelmakoodia voisi kokeilla optimoida tai muuten parantaa ja selkeyttää. Myös kunnollinen dokumentointi koko järjestelmästä voisi olla paikallaan. Ohjelman toteuttaminen olisi ehkä kannattanut aloittaa ihan ”puhtaalta pöydältä” eikä tehdä olemassa olevan koodin päälle, sillä se toi paljon hankaluuksia ja valmiin koodin toiminta oli hiukan huonosti selitetty.

LÄHTEET

(Bosch CAN) Bosch CAN Specification version 2.0 1991.

(dsPIC datasheet) Microchip. 2010. dsPIC33EP512MU814 datasheet.[verkko-dokumentti]. [viitattu 28.9.2013]. http://ww1.microchip.com/downloads/en/DeviceDoc/70616g.pdf

(dsPIC ECAN) Microchip, 2008-2011 dsPIC33E/PIC24E Family Reference Manual Section 21. Enhanced Controller Area Network(ECAN). [verkko-dokumentti]. [viitattu 18.10.2013]. http://ww1.microchip.com/downloads/en/DeviceDoc/70353C.pdf

(dsPIC QEI) Microhip, 2010 dsPIC33E/PIC24E Family Reference Manual Section 15. Quadrature Encoder Interface(QEI). [verkkodokumentti] [viitattu 20.10.2013]. http://ww1.microchip.com/downloads/en/DeviceDoc/S15.pdf

(MchipAN1078) Microchip, Zambada Jorge 2010 Sensorless Field Oriented Control of a PMSM.[verkko-dokumentti ja esimerkkikoodi]. Viitattu [4.10.2013].http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en530042

(MchipCE427) Microchip Coding Example 427 Crosswire Communication between ECAN 1 and ECAN 2 modules. viitattu[18.10.2013]. http://ww1.microchip.com/downloads/en/DeviceDoc/CE427_ECAN_Crosswire.zip

(Microchip C30) Microchip. 2007. MPLAB C30 C Compiler User's Guide. [verkkodokumentti]. [viitattu 28.5.2013]. http://ww1.microchip.com/downloads/en/DeviceDoc/70094E.pdf

(Loppuraportti 2013) Sulautettujen järjestelmien seminaarikurssi, loppuraportti. Matti Ruohonen 23.6.2013