Lappeenrannan teknillinen yliopisto
Teknillinen tiedekunta
KANDIDAATINTYÖ

BL10A1000 Kandidaatintyö ja seminaari
2.3.2009

PROFIBUS JA SERCOS II -KENTTÄVÄYLIEN NOPEUSVERTAILU

COMPARISON OF SPEED BETWEEN PROFIBUS AND SERCOS II FIELDBUSES

0278116 Hans Baumgartner SÄTE 4

SISÄLLYSLUETTELO

KÄYTETYT MERKINNÄT JA LYHENTEET

ADR	Target address
AT	Axis Telegram
B	tavu (*byte*)
bit	bitti
BOF	Beginning Of Frame
CNC	Computer Numerical Controlled
DA	Destination Address
D_{in}	isäntälaitteen vastaanottamien tavujen määrä
D_{out}	isäntälaitteen lähettämien tavujen määrä
DSAP	Destination Service Access Point
ED	End Delimiter
EOF	End Of Frame
f	tiedonsiirtonopeus
FC	Fuction code
FCS	Frame Checking Sequence
IDN	Identification Number
LE	Length of protocol data unit
LEr	Repetition of protocol data unit
MDT	Master Data Telegram
MST	Master Synchronization Telegram
NRZ	Non Return to Zero
NRZI	Non Return to Zero Inverted
n_s	laitteiden lukumäärä
OSI	Open System Interconnection
SA	Source Address
SD	Start Delimiter
SERCOS	SErial Real-time COmmunication System
SSAP	Source Service Access Point
t	aika
T	aika
T_{bit}	yhden bitin siirtämiseen kuluva aika

JOHDANTO

Kenttäväylien käyttö teollisuudessa kasvaa jatkuvasti. Korvaamalla vanhat analogiset tiedonsiirtotavat digitaalisilla kenttäväylillä, voidaan tiedonsiirtoa tehostaa ja yksinkertaistaa. Kenttäväylille ei kuitenkaan ole syntynyt esimerkiksi 4-20 mA virtaviestin tapaista *de facto* -standardia, vaan käytössä on useita erilaisia kenttäväyliä.

Kandidaatintutkielmassa on perehdytty PROFIBUS DP ja SERCOS II (SErial Real-time COmmunication System) -kenttäväyliin ja selvitetty kenttäväylien käyttäytymistä nopeassa, reaaliaikaista tiedonsiirtoa vaativassa järjestelmässä. Työssä on muodostettu suuntaa-antavat yhtälöt tutkittavien kenttäväylien suurimmasta käytettävästä nopeudesta sekä testattu teoreettisia yhtälöitä käytännössä.

Työ on tehty Lappeenrannan teknillisessä yliopistossa yhteistyössä ABB Oy:n kanssa ja se liittyy läheisesti hihnaservokäyttöisen pakkauskoneen koelaitteiston tutkimukseen. Työssä on vertailtu servokäyttöjen ohjauksessa vaadittavien kenttäväylien ominaisuuksia ja väylän vaikutusta ohjauksessa saavutettavaan nopeuteen.

1. YLEISTÄ KENTTÄVÄYLISTÄ

Kenttäväylät ovat teollisuudessa käytettyjä kenttätason tiedonsiirtoväyliä, joilta usein vaaditaan reaaliaikaista vastetta. Kenttäväylien avulla hoidetaan prosessien laitteiden välinen tiedonsiirto, joten kenttäväylien on oltava luotettavia ja riittävän nopeita. Tyypillisesti kenttäväylät yhdistävät säätävän laitteiston prosessissa sijaitseviin varsinaisiin toimilaitteisiin, kuten sensoreihin, venttiileihin ja sähkömoottoreihin.

Suosittu ratkaisu reaaliaikaiseen tiedonsiirtoon on pitkään ollut 4-20 mA virtaviestiin perustuva analoginen signaali. Tekniikka on edullinen, yksinkertainen ja varmatoiminen, mutta se soveltuu ainoastaan kahden laitteen keskinäiseen tiedonsiirtoon. Kyseisellä tekniikalla toteutettu monimutkainen prosessi vaatii siis jokaiselle toimilaitteelle oman kaapeloinnin, joka lisää järjestelmän monimutkaisuutta ja heikentää näin sen luotettavuutta. Myös prosessin laajennettavuus on näin ollen hankalaa ja kallista, kun jokaiselle uudelle laitteelle täytyy asentaa oman johdotus.

Kenttäväylien avulla pystytään yhdistämään lukuisia - väylästä riippuen, jopa useita satoja - toimilaitteita huomattavasti pienemmällä kaapelointimäärällä. Eri kenttäväylät sallivat erilaisia verkkotopologioita, joista yleisesti käytössä ovat mm. *daisy-chain*, tähti, rinki ja puu -verkkotopologiat. OSI-mallin (Open System Interconnection) fyysinen kerros voidaan eri kenttäväylillä toteuttaa eri tavoilla kuten optisella tiedonsiirrolla, RS-485 parikaapelilla tai koaksiaalikaapelilla. Suurin osa fyysisen kerroksen toteutuksista on tällä hetkellä tehty jollain parikaapeliratkaisulla tai optisesti.

Vaikka kenttäväylien kehitystyö on aloitettu jo 1980-luvulla, saatiin ensimmäinen standardiksi luokiteltava julkaisu aikaan vasta 1999. Tuolloin IEC 61158 standardi nimesi kahdeksan kenttäväyläprotokollaa. Nykyään standardin IEC 61558 versiossa 2.0 2007-11 on nimettynä 16 eri protokollaperhettä, jotka on listattuna taulukkoon 1.1.

Taulukko 1.1 Kenttäväyläteknologiat ja protokollat [1]

	Teknologia	Profiilin numero	Protokolla
1	FOUNDATION Fieldbus ™	1/1	FF H1
		1/2	FF HSE
		1/3	FF H2
2	CIP ™	2/1	ControlNet™
		2/2	EtherNet/IP™
		2/3	DeviceNet™
3	PROFIBUS, PROFINET	3/1	PROFIBUS DP
		3/2	PROFIBUS PA
		3/3	PROFIBUS CBA
		3/4	PROFINET IO CC-A
		3/5	PROFINET IO CC-B
		3/6	PROFINET IO CC-C
4	P-NET ®	4/1	P-NET RS-485
		4/2	P-NET RS-232
		4/3	P-NET on IP
5	WorldFIP ®	5/1	WorldFIP
		5/2	WorldFIP with subMMS
		5/3	WorldFIP minimal for TCP/IP
6	INTERBUS ®	6/1	INTERBUS
		6/2	INTERBUS TCP/IP
		6/3	INTERBUS minimal subset of CP 6/1
		6/4	
		6/5	
		6/6	
7	-	-	Poistettu
8	CC-Link	8/1	CC-Link/V1
		8/2	CC-Link/V2
		8/3	CC-Link/LT
9	HART	9/1	HART
10	Vnet/IP	10/1	Vnet/IP
11	Tcnet	11/1	Tcnet
		11/2	Tcnet-Loop
12	EtherCAT	12/1	
		12/2	
13	ETHERNET Powerlink	13/1	EPL
14	EPA	14/1	
		14/2	
15	MODBUS ® - RTPS	15/1	MODBUS TCP
		15/2	RTPS
16	SERCOS	16/1	SERCOS I
		16/2	SERCOS II
		16/3	SERCOS III

Taulukosta 1.1 havaitaan, että kenttäväylämarkkinoille ei ole syntynyt yhtä 4-20 mA viestin tapaista *de facto* -standardia, vaan lukuisia erillisiä tekniikoita ja niiden toteutustapoja. Erilaisten tekniikoiden lukumäärä onkin yksi kenttäväylien yleistymisen suuria haasteita. Järjestelmien suunnitteleminen hankaloituu, kun laitevalmistajilla ei ole oikealle väylälle sopivaa laitetta ja kun tulevaisuuden johtavaa kenttäväyläratkaisua on vaikea ennustaa.

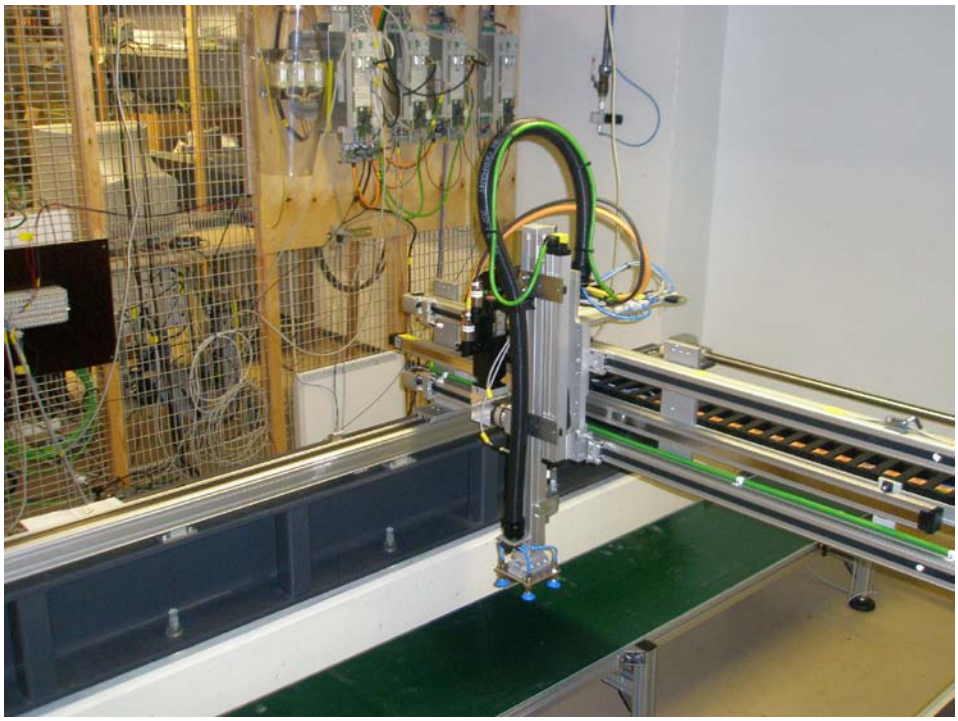
Seuraavassa keskitymme PROFIBUS DP ja SERCOS II -kenttäväyliin ja niiden nopeuksien tarkastelemiseen. Protokollien analysoinnissa tarkastellaan vain varsinaisen ohjaus- ja mittausdatan lii-

kennettä prosessin toiminnan aikana. Tutkimuksen ulkopuolelle on jätetty muu liikenne, joka verkossa mahdollisesti tapahtuu, kuten alustukset ja virheenkorjaukset. Saavutetut maksiminopeuksien arvot ovat siis teoreettisia maksiminopeuksia.

Työn tavoitteena on muodostaa tarkasteltaville kenttäväylille yhtälöt, joiden avulla voidaan arvioida, kuinka usein toimilaitteille voidaan lähettää uutta ohjausinformaatiota. Lisäksi muodostettuja yhtälöitä testataan käytännössä.

2. LAITTEISTO

Tutkimus liittyy läheisesti käytännön ongelmaan, vaikkakin tuloksia voi suoraan käyttää myös muihin sovelluksiin. Tutkittava sovelluskohde on kuvassa 2.1 esiintyvä lineaarisilla hammashihnaservokäytöillä ajettava pakkauskoneen koelaitteisto. Koneessa on kolme taajuusmuuttajaohjattua kestomagneettiservomoottoria, jotka ajavat hammashihnojen välityksellä pakkauskonetta x-, y- ja z-suunnissa. Lisäksi laitteistossa on taajuusmuuttajaohjattu liukuhihna.



Kuva 2.1 Tutkittava koelaitteisto

Kuvan 2.1 taustalla näkyvät neljä servomoottoreita ohjaavaa taajuusmuuttajaa on kytketty säätöjärjestelmään PROFIBUS DP tai SERCOS II -kenttäväylän avulla. Laitteiston tutkimuksessa on havaittu PROFIBUS DP -kenttäväylän viiveiden ja nopeuden olevan yksi rajoittava tekijä laitteen säätöä suunniteltaessa [2]. Tämän takia järjestelmässä on siirrytty käyttämään nopeampaa SERCOS II -väylää. Kandidaatintutkielman tarkoituksena onkin muodostaa yhtälöt kenttäväylien tiedonsiirtonopeuksille ja verrata PROFIBUS DP ja SERCOS II -kenttäväylien hyötydatan siirtonopeuksia.

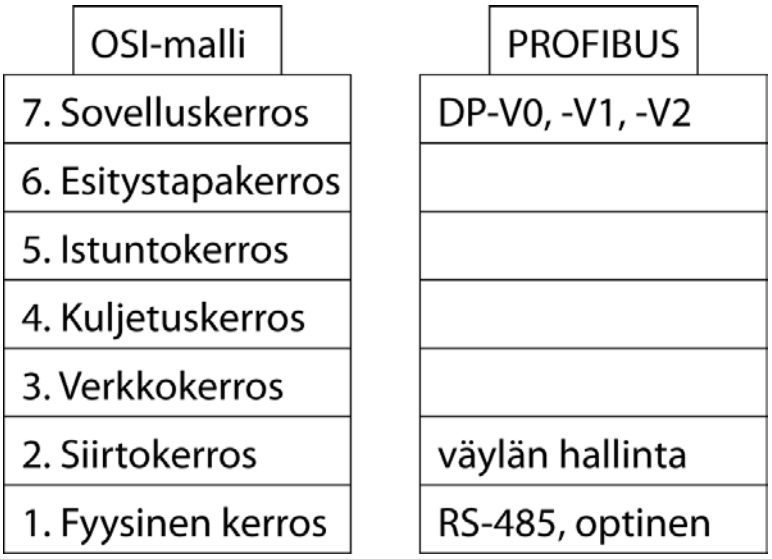
Nopeuden selvittämiseksi tutkitaan väylällä liikkuvan datan määrä ja muoto, sekä selvitetään, mikä osuus datasta on varsinaista ohjausinformaatiota. Ohjaussovelluksissa tiedonsiirto on muodoltaan syklistä, eli tietoa vaihdetaan tasaisin väliajoin, joten kun tiedetään väylällä liikkuvan datan määrä,

voidaan muodostaa suuntaa-antava yhtälö suurimmalle mahdolliselle ohjausarvojen ja mittausdatan päivitysnopeudelle.

Väylän nopeus ei toki ole ainoa ohjausinformaation päivitysnopeuteen vaikuttava seikka. Toimilaitteet vaativat oman aikansa tiedon käsittelyyn ja myös signaalien eteneminen piirissä vaativat omat aikansa, jotka voi olla hyvin huomattavia. Työssä on kuitenkin keskitytty tarkastelemaan väylän teoreettisen nopeuden vaikutusta säätöpiirien viiveisiin.

3. PROFIBUS

PROFIBUS on laaja kenttäväylästandardi, joka määrittelee kuvan 3.1 OSI-mallista fyysisen-, siirto- ja sovelluskerroksen. Fyysinen kerros määrittelee nimensä mukaisesti väylän fyysisen toteutuksen. Siirtokerros antaa väylää tarvitseville laitteille vuorollaan hallintaoikeuden väylän käyttöön, jotta dataa ei voida lähettää samanaikaisesti useasta toimilaitteesta ja sovelluskerros huolehtii datan oikeasta muodosta, kuittauksista, aikataulutuksesta, jne.



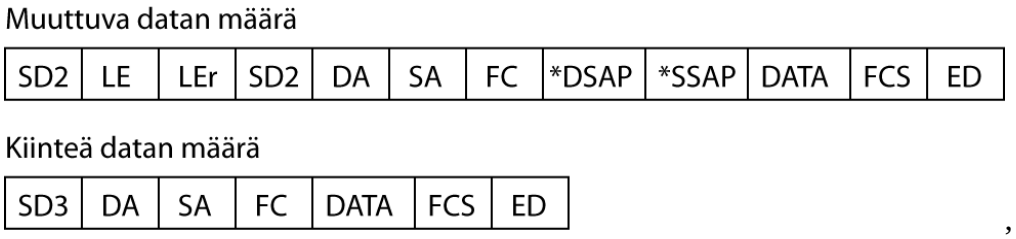
Kuva 3.1 PROFIBUS-kenttäväylän sijoittuminen OSI-mallin mukaisille kerroksille

Verkon laitteet jaetaan toimintansa mukaan isäntä- ja orjalaitteisiin, joilla on erilaiset verkonkäyttöoikeudet. Tyypillisesti vain isäntälaitteet saavat aloittaa verkon käytön ja orjalaitteet lähettää dataa vain pyydettäessä. Tämä kuitenkin riippuu sovelluskerroksesta, joten keskitymme työssä koelaitteiston tapaukseen, jossa vain isännällä on oikeus tiedonsiirron aloittamiseen.

PROFIBUS-standardi ei määrittele verkon fyysiselle toteutukselle vain yhtä kaapelointitapaa, vaan kaapeloinnissa voidaan käyttää sekä optista, että sähköistä tiedonsiirtoa. Myös verkon rakenne riippuu sovelluskohteesta, joten yhdessä järjestelmässä voi olla joko yksi tai useampia isäntälaitteita. Koelaitteiston tapauksessa fyysinen kerros on toteutettu tavallisella parikaapelilla ja järjestelmässä on yksi isäntä- ja neljä orjalaitetta.

3.1 Telegrammit

IEC 61158 -standardi määritellee PROFIBUS-väylän tiedonsiirtotelegrammit. Tarkastelun ulkopuolelle on jätetty väylän alustukset sekä mahdolliset virheenkorjaukset, joten keskitymme ainoastaan prosessin toiminnan aikaiseen, virheettömään tiedonvaihtoon. PROFIBUS määrittelee siirrettävän datan pituuden kiinteästi kahdeksan tavun pituiseksi tai vaihtoehtoisesti data-tavujen pituus voi olla 1..246 tavua. Datan määrästä riippuen telegrammien muoto poikkeaa hieman. Kuvassa 3.2 on esitetty kiinteän ja datamäärältään vaihtelevan telegrammin muoto.



,

jossa
SD: Start Delimiter (SD2 = 0x68, SD3 = 0xA2)
LE: Length of protocol data unit, (incl. DA, SA, FC, DSAP, SSAP)
LEr: Repetition of protocol data unit
DA: Destination Address
SA: Source Address
FC: Function Code
*DSAP: Destination Service Access Point
*SSAP: Source Service Access Point
FCS: Frame Checking Sequence
ED: End Delimiter (= 0x16 !)

Kuva 3.2 PROFIBUS-väylän datatelgrammityypit

PROFIBUS DP käyttää tyypillisesti muuttuvan datamäärän telegrammeja SD2, kuten myös koelaitteistossa oleva FPBA-01 -sovitinkortti [3]. Tämän vuoksi keskitymme tutkimaan vaihtuvan datamäärän telegrammeja. Kuvassa 3.2 esiintyvien lohkojen pituus on datalohkoa lukuun ottamatta yksi tavu eli kahdeksan bittiä. Telegrammin pituudeksi saadaan kaikki lohkot yhteenlaskettuna

$$\begin{aligned} &1\mathrm{B(SD2)} + 1\mathrm{B(LE)} + 1\mathrm{B(LEr)} + 1\mathrm{B(SD2)} + 1\mathrm{B(DA)} + 1\mathrm{B(SA)} \\ &+ 1\mathrm{B(FC)} + 1...246\mathrm{B(DATA)} + 1\mathrm{B(FCS)} + 1\mathrm{B(ED)} = 10...255\mathrm{B} \end{aligned} \qquad (3.1)$$

Tähdellä merkityt lohkot *DSAP ja *SSAP on varattu ISO-mallin sisäiseen tiedonsiirtoon. Lohkot lasketaan kuuluvaksi datalohkoon ja sovelluksesta riippuen lohkot voivat joko olla, tai olla olematta telegrammissa. Kyseisistä lohkoista johtuen kirjallisuudessa ilmoitetaan data-tavujen suurimmaksi

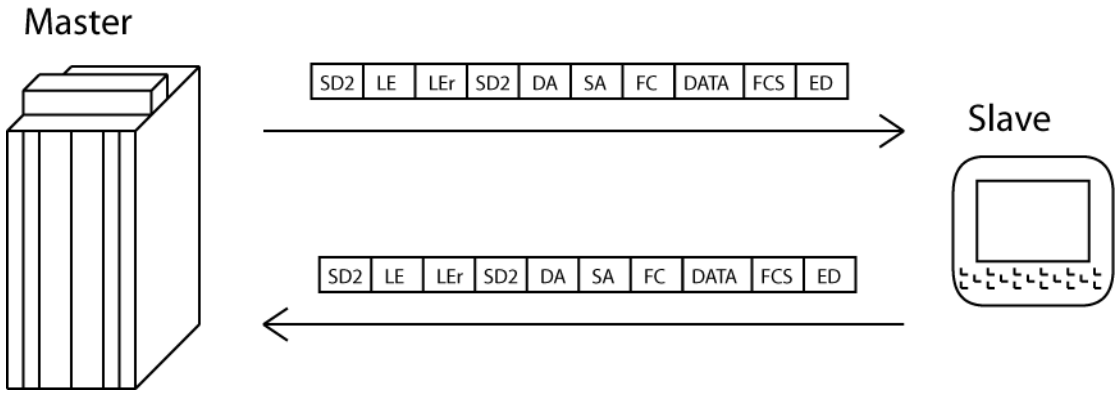
määräksi joko 244 tai 246, laskutavasta riippuen. Esimerkiksi koelaitteistossa käytettävän FPBA-01 -sovitinkortin tapauksessa kyseisiä lohkoja ei oletusarvoisessa, syklisessä tiedonsiirrossa käytetä, joten niitä ei myöskään lähetetä telegrammin mukana. Usein laitteet rajaavat myös siirrettävän datan määrää, kuten koelaitteiston sovitinkortissa, joka määrittelee datalohkon pituudeksi 4..28 tavua.

3.2 Tiedonsiirto

Edellisessä kappaleessa kerrotut telegrammien pituuden ovat vain telegrammissa siirtyvien tavujen lukumäärä. Fyysisessä tiedonsiirrossa jokaisen tavun siirtämiseen tarvitaan 11 bittiä. Tiedonsiirtolohko koostuu siis aloitusbitistä, kahdeksasta databitistä, pariteettibitistä ja lopetusbitistä. Näin ollen siirretyn telegrammin todellinen pituus saadaan, kun siirrettyihin tavuihin lisätään kolme kertaa siirrettävien tavujen lukumäärä bittejä. Esimerkiksi 20 tavun pituisen telegrammin koko bitteinä on

$$20\mathrm{B} + \left(3 \cdot 20\mathrm{bit}\right) = 220\mathrm{bit} \qquad (3.2)$$

PROFIBUS-kenttäväylä määrittelee useita tiedonsiirtometodeja isäntä- ja orjalaitteiden välille. Yksinkertaisimmillaan tiedonsiirto tapahtuu niin, että isäntälaite lähettää telegrammin, joka sisältää ohjausinformaation orjalaitteelle, jonka jälkeen orjalaite vastaa välittömästi, omalla telegrammillaan isäntälaitteelle. Orjan telegrammi sisältää tarvittavat mittaustiedot ja oloarvot orjalaitteelta. Oletusarvoisesti myös koelaitteiston käyttämä FPBA-01 -sovitinkortti käyttää kyseistä tiedonsiirtotapaa. Kuva 3.3 havainnollistaa PROFIBUS DP:n tiedonsiirtoa yksinkertaisimmillaan.



Kuva 3.3 Isäntä-orja-tiedonsiirto

Kuvan 3.3 mukaisen tiedonsiirron vaatima aikaa voidaan nyt laskea. PROFIBUS vaatii 33 bitin pituisen synkronointiajan ennen jokaista kyselyä, lisäksi sekä isäntä-, että orjalaitteella on jonkin-

lainen laitekohtainen vasteaika. Käsitellään aluksi kuitenkin vain väylän mahdollistamaa tiedonsiirtonopeutta.

Merkitään väylän baudinopeutta symbolilla *f*. Yhden bitin vaatimaa siirtoaikaa on silloin

$$T_{\text{bit}} = \frac{1}{f} \tag{3.3}$$

Jokaisessa siirrettävässä telegrammissa on 11 tai 9 *header*-lohkoa, sekä tarvittava määrä data-tavuja. Koelaitteiston tapauksessa, jossa orjalaite vastaa välittömästi isännän kutsuun, ei DSAP ja SSAP -lohkoja käytetä, jolloin *header*-lohkoja on yhdeksän kappaletta. Kun jokaista telegrammiparia edeltää 33 bittijakson synkronointiaika, saadaan kahden telegrammin tiedonsiirtoajaksi ilman dataa

$$33 \cdot T_{\text{bit}} + 2 \cdot 9 \cdot 11 \cdot T_{\text{bit}} = 231 T_{\text{bit}} \tag{3.4}$$

Yhtälön (3.4) mukainen aika vaaditaan vähintään telegrammien siirtämiseen. Yksi data-tavu koostuu siis aloitusbitistä, kahdeksasta databitistä, lopetusbitistä ja pariteettibitistä, eli yhteensä 11 bitistä. Tavallisesti orjalle lähetettävän datan määrä on eri kuin orjan isännälle lähettämän datan määrä. Merkitään isännän lähettämää dataa tavuina D_{out} ja orjan lähettämien data-tavujen määrää D_{in}. Väylän määrittelemän telegrammiparin siirtoaika on nyt

$$231 T_{\text{bit}} + D_{\text{out}} \cdot 11 \cdot T_{\text{bit}} + D_{\text{in}} \cdot 11 \cdot T_{\text{bit}} = \left(231 + 11 \cdot \left(D_{\text{out}} + D_{\text{in}}\right)\right) T_{\text{bit}} \tag{3.5}$$

Otetaan esimerkki, jossa neljän orjalaitteen ja yhden isännän järjestelmässä, isäntä lähettää kaikille orjille D_{out} = 10 data-tavua ja vastaanottaa orjilta D_{in} = 15 datatavua. Väylän nopeus on 12 Mbit/s, merkitään orjien lukumäärää n_{s}. Yhtälöistä (3.3) ja (3.5) saadaan:

$$\begin{aligned} T_{\text{bit}} &= \frac{1\text{bit}}{12000000 \frac{\text{bit}}{\text{s}}} = 83\text{ns} \\ T &= n_{\text{s}} \cdot \left(231 + 11 \cdot \left(D_{\text{out}} + D_{\text{in}}\right)\right) T_{\text{bit}} \\ T &= 4 \cdot \left(231 + 11 \cdot \left(10 + 15\right)\right) 83\text{ns} = 168\mu\text{s} \end{aligned} \tag{3.6}$$

Yhtälöstä (3.6) nähdään esimerkin järjestelmän yhden jakson pienimmäksi jaksonajaksi 168 µs. Näin pientä arvoa ei kuitenkaan voida koskaan käytännössä saavuttaa, sillä orjalaitteilla on aina jokin vasteaika. Myös isäntälaitteella on aika, joka laitteen täytyy odottaa ennen kuin voi synkronoida väylän ja lähettää uuden telegrammin seuraavalle orjalaitteelle. Nämä ajat ovat tyypillisesti luokkaa useita kymmeniä bittijaksoja ja niiden tarkat arvot löytyvät laitevalmistaja ohjeista. Yhtälössä (3.6) oli oletettu, että kaikille orjalaitteille lähetetään sama määrä ohjausinformaatiota ja kaikki orjat vastaavat samalla datamäärällä takaisinkytkentätietoa. Käytännössä asia ei kuitenkaan usein ole näin yksinkertainen, joten yhtälö (3.6) on hyvä muokata summamuotoon. Jos merkitään orjan reaktioaikaa T_{SR} ja isännän T_{ID}, saadaan yhden syklin jaksonajaksi

$$T = \sum_{k=1}^{n_s} \left(\left(231 + 11 \cdot \left(D_{\mathrm{out}(k)} + D_{\mathrm{in}(k)} \right) \right) T_{\mathrm{bit}} + T_{\mathrm{SR}(k)} + T_{\mathrm{ID}(k)} \right) \tag{3.7}$$

3.3 Koodaus

Tutkittu PROFIBUS DP laitteisto käyttää fyysisen tiedonsiirron koodaukseen NRZ-koodausta (Non Return to Zero), jossa siirrettävä bitti vastaa suoraan tiettyä jännitetasoa ja saman symbolin aikana ei palata mihinkään perusjännitteeseen. Käytännössä tämä tarkoittaa siis sitä, että minkäänlaista koodausta ei ole, vaan bitit liikkuvat tiedonsiirtolinjassa sellaisenaan. Ylhäällä oleva linja tarkoittaa ykkösbittiä ja alhaalla oleva linja nollabittiä.

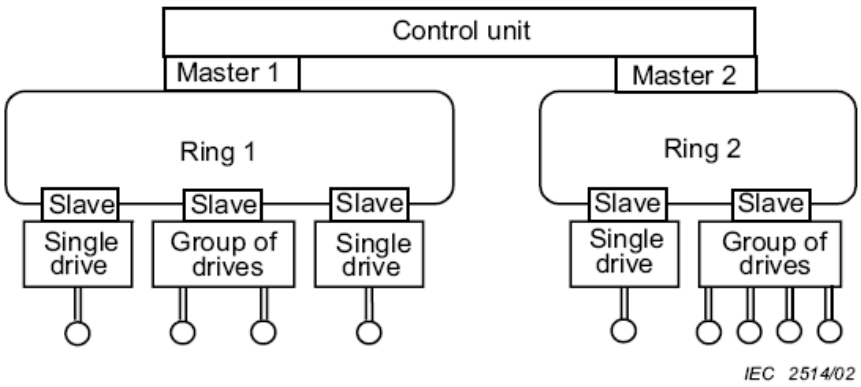
4. SERCOS II

Toinen tutkittava kenttäväylä on SERCOS II. Väylää on kehitetty erityisesti erilaisia tarkkaa säätöä vaativia servokäyttöjä varten, kuten CNC- (Computer Numerical Controlled), pakkaus- ja painokoneisiin. SERCOS-kenttäväylä voidaan jakaa kolmeen versioon. Kaikissa versioissa toimintaperiaate on sama, mutta OSI-mallin alemmat kerrokset ja siten myös nopeus vaihtelevat. Kahdessa ensimmäisessä versiossa, SERCOS I ja SERCOS II, tiedonsiirto on toteutettu optisella kaapelilla. Versiossa SERCOS III kenttäväylä perustuu pakettipohjaiseen ethernet-teknologiaan, jonka fyysinen toteutus voi olla Cat-parikaapeli tai mahdollisesti optinen kaapeli. SERCOS-väylän kaikissa versioissa laitteet on kytketty *daisy-chain*-rakenteella toisiinsa, eli laite A on kytketty laitteeseen B, laite B laitteeseen C jne.

SERCOS II on SERCOS I:n paranneltu versio, joka eroaa pikkuveljestään tiedonsiirtonopeuden osalta. Ensimmäisessä versiossa nopeus oli 2 tai 4 Mbit/s ja versiossa kaksi 2/4/8/16 Mbit/s. Vertailun vuoksi voidaan mainita SERCOS III:n nopeudeksi jopa 100 Mbit/s. Perehdymme seuraavassa tarkemmin SERCOS II -kenttäväylään, mutta koska toimintaperiaate on kaikilla versiolla sama, voidaan tuloksia hyödyntää myös muiden SERCOS-väylän versioiden kanssa.

PROFIBUS-väylän tapaan myös SERCOS-standardi määrittelee kuvan 3.1 OSI-mallin mukaiset kerrokset, joskin SERCOS-väylän tapauksessa fyysinen toteutus on joko optinen- tai parikaapeli ja sovelluskerrokselle on omat määrityksensä.

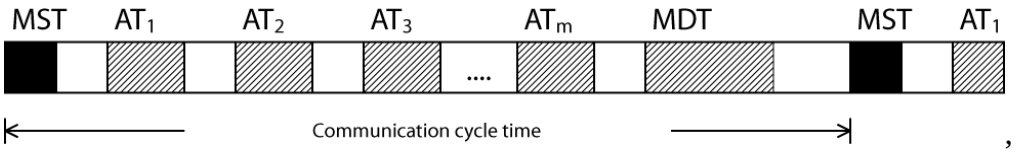
Yhdessä SERCOS-ringissä on yksi isäntälaite ja 1-254 orjalaitetta. Laitteet on kytketty toisiinsa optisella kaapeloinnilla ja *daisy-chain*-rakenteella. Kuvassa 4.1 on esitetty malli tyypillisestä SERCOS-kenttäväylän rakenteesta.



Kuva 4.1 SERCOS-väylän tyypillinen verkkotopologia [5]

4.1 Telegrammit

Toimintaperiaatteeltaan SERCOS II eroaa merkittävästi PROFIBUS DP:sta. SERCOS-väylän toiminta perustuu siihen, että jokainen verkon laite lähettää oman telegramminsa täsmälleen oikealla ajanhetkellä. Isäntä ei siis tiedustele jokaiselta orjalta erikseen tämän tilaa ja lähetä tarvittavia ohjearvoja, vaan väylässä liikkuvan datan lähettäjä ja vastaanottaja riippuvat ajasta, joka on kulunut syklin aloittavan MST-telegrammin lähettämisestä. Kuvassa 4.2 on esitetty yhden tiedonsiirtojakson rakenne.



,

jossa
MST: Master synchronization telegram
AT: Axis (drive) telegram
MDT: Master data telegram

Kuva 4.2 SERCOS-väylän tiedonsiirtojakso

Kuvan 4.2 mukaisen tiedonsiirtojakson jaksonaika on määritelty SERCOS-standardissa ja se voi olla 62.5, 125 tai 250 µs tai *n* x 250 µs. Suurin mahdollinen kesto tiedonsiirtojaksolle on 65 ms. Liikenteen alustuksessa määritellään jokaiselle orjalaitteelle aika MST:n ja AT:n välissä, jonka jälkeen orjalaite lähettää oman telegramminsa. Isäntälaite lähettää oman ohjausinformaationsa orjalaitteille MDT-telegrammissa. Jokainen orjalaite kuulee saman MDT-telegrammin, jossa on kullekin orjalle oma lohkonsa.

Kaikilla telegrammeilla MST, AT ja MDT on samanlainen runko, johon kuuluu neljä vakiolohkoa. Kuvassa 4.3 on esitetty telegrammin perusrakenne.

jossa
BOF: Beginning of frame (01111110_B)
ADR: Target address
FCS: Frame check sequence
EOF: End of frame (01111110_B)

Kuva 4.3 Telegrammien perusrakenne

Kuvan 4.3 telegrammin lohkojen pituudet ovat BOF – 1 tavu, ADR – 1 tavu, FCS – 2 tavua ja EOF – 1 tavu. Fyysisen, kaapelissa liikkuvan datan pituus ei kuitenkaan ole aina sama, kuin lähetetyn datan pituus. SERCOS-väylä käyttää *bit stuffing* -tekniikkaa väylän sykronointiin ja erottamaan BOF ja EOF-lohkot muista lohkoista. Koska on mahdollista ja todennäköistä, että myös muut lohkot sisältävät BOF ja EOF-lohkoille määritellyn tavun 01111110_B täytyy väylän ohjauselektroniikan jotenkin estää 01111110_B bittijonon pääsy väylälle. *Bit stuffing* -tekniikassa lähetin lisää jokaisen viiden ykkösbitin perään nollan, joka vastaavasti vastaanottopäässä poistetaan. Näin voidaan estää 01111110_B tavun pääsy väylälle. Tekniikka luonnollisesti aiheuttaa sen, että telegrammin bittien määrää ei voida tarkasti laskea, joten telegrammien väliin on jätettävä pieni marginaali *bit stuffing*:lle.

Varsinaista maksimipituutta DATA-lohkon pituudelle ei ole määritelty, joten siirrettävän datan määrää rajoittaa käytännössä tiedonsiirtojakson maksimipituus 65 ms.

4.1.1 MST

MST-telegrammi aloittaa jokaisen tiedonsiirtosyklin ja se lähetetään kaikille orjalaitteille, jolloin ADR-kentässä on *broadcast*-osoite 11111111_B. Telegrammin tarkoituksena on synkronoida orjalaitteet siten, että ne osaavat lähettää oman telegramminsa täsmälleen määrättynä aikana.

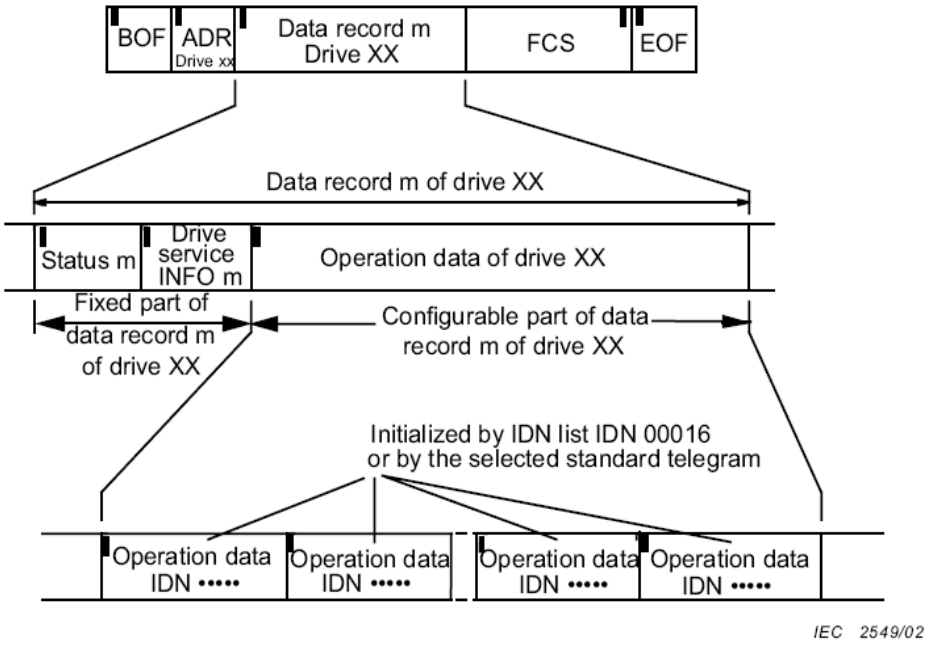
DATA-kenttä sisältää tiedon siitä, mitä operaatiota väylältä vaaditaan eli ollaanko väylää sulkemassa, alustamassa, lähettämässä tiedostoa, syklisessä tiedonsiirrossa jne. Tutkielmassa oletamme alus-

tukset suoritetuksi ja tiedonsiirron olevan syklistä. Syklisen tiedonsiirron tapauksessa DATA-kenttä sisältää tavun 00000100_B.

FCS on 16-bittinen CRC-16-CCITT-tarkistussumma, joka lasketaan telegrammin ADR- ja DATA-lohkojen bittien ja jakajapolynomin avulla.

4.1.2 AT

AT-telegrammien DATA-kenttä koostuu kiinteästä neljän tavun osasta sekä käyttökohteesta riippuvasta määriteltävästä osasta. Kuva 4.4 havainnollistaa AT-telegrammin rakennetta.

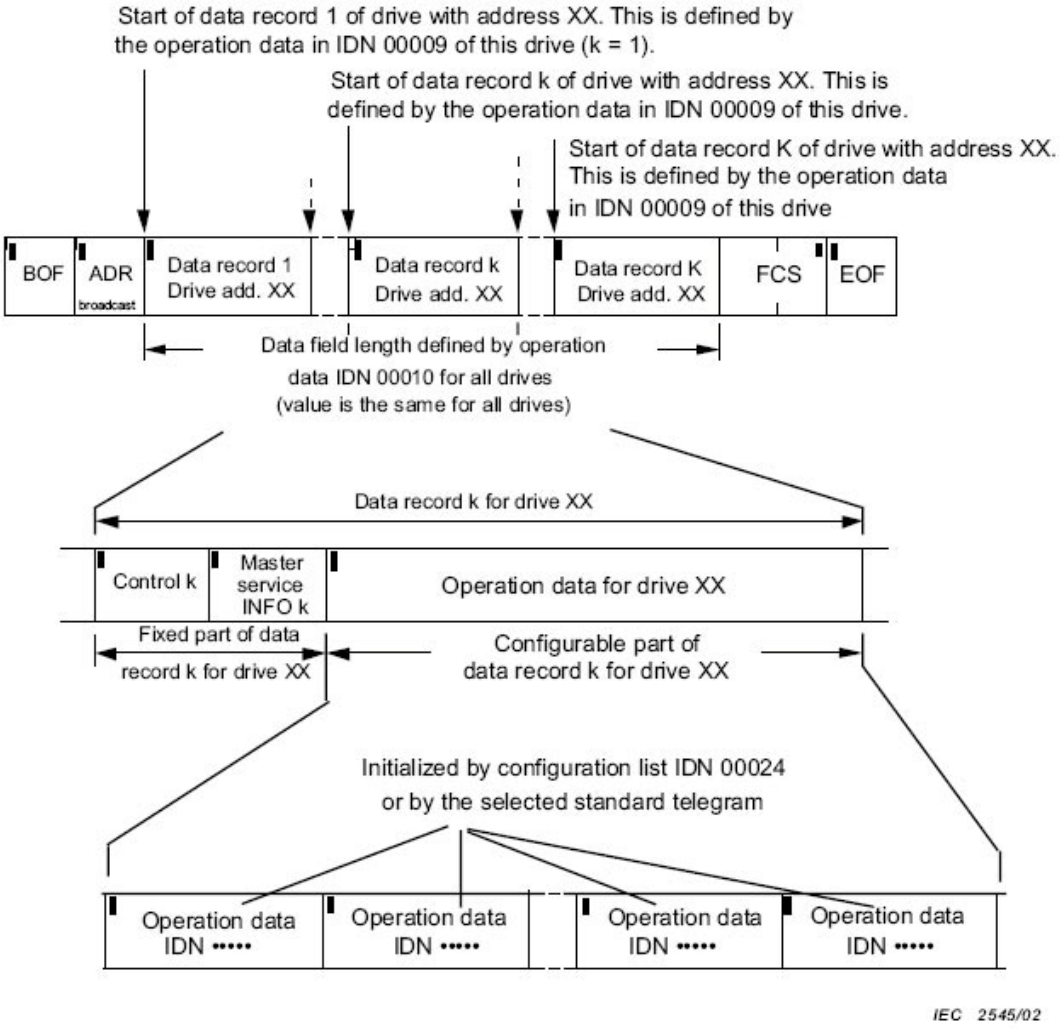


Kuva 4.4 AT-telegrammin rakenne [5]

Kuvassa 4.4 olevat lohkot *Status m* ja *INFO m* ovat pituudeltaan kaksi tavua kumpikin. Järjestelmäkohtainen data koostuu kahden tai neljän tavun pituisista lohkoista. SERCOS-standardi määrittelee käytön yksinkertaistamiseksi lukuisan joukon valmiita ohjauskomentoja, joille on annettu omat IDN-numeronsa (IDentification Number).

4.1.3 MDT

Viimeisenä, orjalaitteiden jälkeen, isäntälaite lähettää oman telegramminsa kaikille orjalaitteille. MDT:n DATA-kenttä sisältää oman lohkonsa jokaiselle orjalaitteelle kuvan 4.5 mukaisesti.



Kuva 4.5 MDT-telegrammin rakenne [5]

MDT-telegrammi on hyvin samankaltainen AT-telegrammien kanssa, ainoastaan datakenttiä on MDT-telegrammissa useampia.

4.2 Moodit

SERCOS-standardi määrittelee useita valmiita tiedonsiirtomoodeja. Servokäytössä tyypillisiä ohjausarvoja ovat paikka, nopeus ja vääntömomentti, joten esimerkiksi näille ohjaus- ja takaisinkytkentäarvoille on määritelty omat standarditelegrammit. Standardista IEC 61491 on listattuna IDN-numeroita eri datalle.

4.3 Tiedonsiirtoaika

Edellä olevien tietojen perusteella voidaan nyt laskea aika, joka kuluu kiinteiden telegrammien siirtämiseen ilman dataa. Kuvan 4.3 mukaisen telegrammin lohkot BOF, ADR, FCS ja EOF ovat pi-

tuudeltaan 1-, 1-, 2- ja 1-tavua eli yhteensä viisi tavua. Merkitään orjalaitteiden lukumäärää n_s. Yhdessä tiedonsiirtojaksossa on yhteensä $2 + n_s$ telegrammia. Kun jokaisessa telegrammissa on viisi tavua kiinteästi määriteltyä dataa, saadaan tämän datan määräksi tavuina

$$(2 + n_s) \cdot 5\text{B} \tag{4.1}$$

Lisäksi tiedetään, että MST-telegrammin pituus tavuina on vakio kuusi tavua. Sekä AT-, että MDT-telegrammin DATA-lohkon kiinteän datan osuus on kussakin AT-telegrammissa neljä tavua ja MDT-telegrammissa neljä tavua kutakin orjalaitetta kohden. Kiinteän datan yhteismääräksi tiedonsiirtojaksossa saadaan nyt

$$\begin{aligned}&(2 + n_s) \cdot 5\text{B} + 1\text{B} + n_s \cdot 4\text{B} + n_s \cdot 4\text{B} = \\ &n_s \cdot (5\text{B} + 4\text{B} + 4\text{B}) + 2 \cdot 5\text{B} + 1\text{B} = n_s \cdot 13\text{B} + 11\text{B}\end{aligned} \tag{4.2}$$

Yhtälöön (4.2) merkitsemällä isännän lähettämää dataa tavuina D_{out} ja orjan lähettämien datatavujen määrää D_{in}. Saadaan yleinen yhtälö datamäärälle.

$$n_s \cdot 13\text{B} + 11\text{B} + \sum_{k=1}^{n_s} \left(D_{\text{in(k)}} + D_{\text{out(k)}} \right) \tag{4.3}$$

Datan siirtämiseen vaadittava aika saadaan siirtonopeuden avulla. Merkitään baudinopeutta symbolilla *f*, jolloin yhden bitin siirtämiseen kuluva aika on

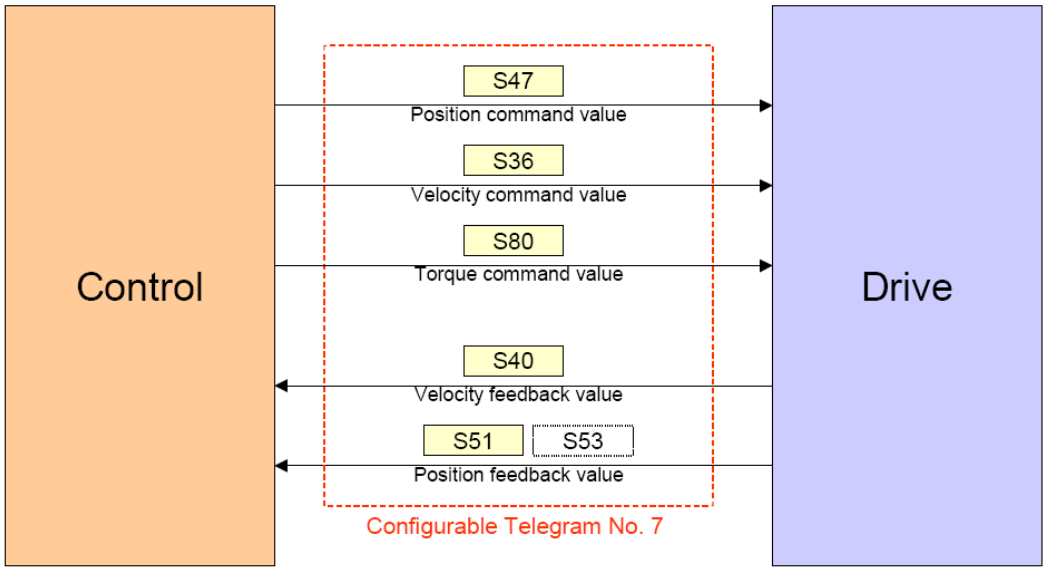
$$T_{\text{bit}} = \frac{1}{f} \tag{4.4}$$

Yhtälöistä (4.3) ja (4.4) saadaan nyt yhden tiedonsiirtosyklin jaksonajaksi

$$T = \left(n_s \cdot 13\text{B} + 11\text{B} + \sum_{k=1}^{n_s} \left(D_{\text{in(k)}} + D_{\text{out(k)}} \right) \right) \cdot T_{\text{bit}} \cdot 8\,\text{bit} \tag{4.5}$$

On kuitenkin otettava huomioon, että telegrammeja ei voida lähettää täysin peräkkäin edes teoriassa, sillä *bit stuffing* rajoittaa käytettävää nopeutta. Lisäksi myös muu elektroniikka rajoittaa telegrammien aikaväleille minimiarvon.

Otetaan lähemmin tarkasteltavaksi esimerkiksi *Configurable Telegram No. 7* [6]. Kuvassa 4.6 on esitetty kyseisen tiedonsiirron periaatekaavio.



Kuva 4.6 Esimerkkikuva mahdollisesta tiedonsiirtotavasta SERCOS II-kenttäväylässä [6]

Kuvassa 4.6 olevat merkinnät *S47*, *S36* jne. viittaavat standardin IEC 61491 määrittelemiin standardilohkojen IDN-numeroihin. Standardin IEC 61491 liitteistä löydetään kyseisten lohkojen kuvaukset sekä lohkon pituus tavuina. Lohkojen suomennokset ja pituus tavuina on taulukoitu taulukkoon 4.1.

Taulukko 4.1 Kuvan 4.6 lohkojen merkitys

IDN	Koko tavuina	Englanninkielinen määritelmä	Suomennos
36	4	Velocity command value	Nopeuden ohjearvo
40	4	Velocity feedback value	Nopeuden oloarvo
47	4	Position command value	Paikan ohjearvo
51	4	Position feedback value 1 (motor feedback)	Paikan oloarvo 1 (takaisinkytkentä moottorilta)
53	4	Position feedback value 2 (external feedback)	Paikan oloarvo 2 (ulkoinen mittaus)
80	2	Troque command value	Väännön ohjearvo

Taulukon 4.1 oloarvot ovat takaisinkytkentäsignaaleja orjalaitteilta eli AT-telegrammien DATA-lohkoja. Vastaavasti ohjearvot ovat isäntälaitteen ohjausarvoja, jotka kuuluvat MDT-telegrammiin Tutkittavassa pakkauskoneen koelaitteistossa orjat lähettävät takaisinkytkentäsignaaleina neljäbittiset nopeuden ja paikan oloarvot ja saavat isäntälaitteelta kaksibittisen vääntömomentin ohjausarvon. Yhteensä tietoa siirretään siis 10 tavua. Kyseisellä tiedonsiirrolla n_s orjalaitteen vaatima datamäärä on yhtälön (4.3) mukaan

$$n_s \cdot 13\text{B} + 11\text{B} + n_s\left(8\text{B} + 2\text{B}\right) = n_s \cdot 23\text{B} + 11\text{B} \tag{4.6}$$

ja tiedonsiirtoaika vastaavasti yhtälöllä (4.5)

$$T = \left(n_s \cdot 23\text{B} + 11\text{B}\right) \cdot T_{\text{bit}} \cdot 8\,\text{bit} \tag{4.7}$$

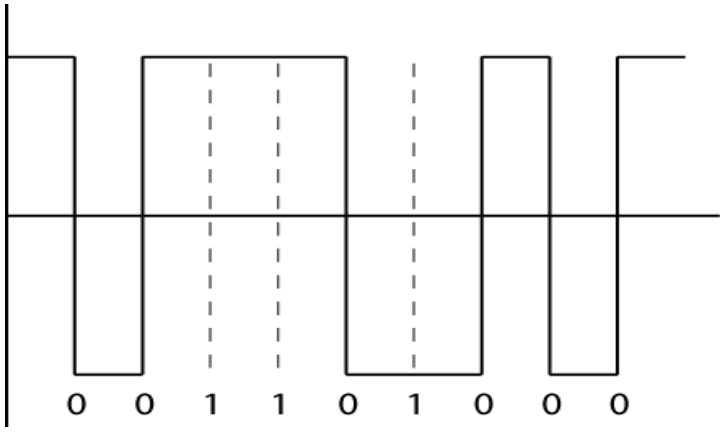
Valitsemalla neljän orjalaitteen ja isännän systeemi ja tiedonsiirtonopeudeksi 8 Mbit/s, saadaan sykliajaksi

$$T = \left(4 \cdot 23\text{B} + 11\text{B}\right) \cdot \frac{1}{16000000\,\frac{\text{bit}}{\text{s}}} \cdot 8\,\text{bit} = 103\,\mu\text{s} \tag{4.8}$$

Koska SERCOS-väylä määrittelee syklinajoiksi 62.5, 125 tai n x 250 µs, voidaan kyseisellä kokoonpanolla teoriassa saavuttaa 125 µs syklinaika.

4.4 Koodaus

SERCOS II -kenttäväylä käyttää fyysisen bittivirran koodaukseen NRZI-koodausta (Non Return to Zero Inverted), jossa nollabitti vaihtaa signaalin tilaa ja ykkösbitti pitää signaalin tilan ennallaan. Kuvassa 4.7 on esitetty graafisesti NRZI-koodauksen periaate.



Kuva 4.7 NRZI-koodaus

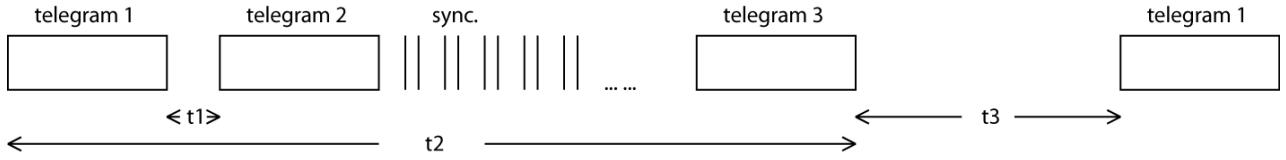
Käytettävästä koodauksesta ja *bit stuffing* -tekniikasta johtuen ei SERCOS-väylää seuraamalla nähdä suoraan väylällä liikkuvaa dataa, vaan haluttaessa seurata väylän liikennettä täytyy data dekoodata.

5. VÄYLIEN TESTAAMINEN

5.1 PROFIBUS

PROFIBUS DP -kenttäväylää testattiin kahden Siemens SIMATIC S7-300 -logiikan avulla. Toinen logiikoista toimi isäntä- ja toinen orjalaitteena. Laitteet konfiguroitiin niin, että isäntä lähetti orjalle jatkuvasti yhden sanan. Laitteen parametrit määrittelivät tyypilliseksi lähetystiheydeksi 1 ms. Suurin mahdollinen aika lähetysten välissä oli kuitenkin määritelty 4.6 ms:ksi, joten jitterin määrittäminen ei kyseisellä koelaitteistolla onnistunut. PROFIBUS-standardi määrittelee väylän jitteriksi mikrosekunnin luokkaa.

Väylän liikennettä tarkkailtiin logiikka-analysaattorin avulla. Syklin aikana väylällä liikkui kolme telegrammia. Syklin alussa isäntä lähetti kuvan 3.3 muotoisen telegrammin orjalle, joka vastasi samanmuotoisella telegrammilla, jonka datalohko sisälsi datatavun 0. Näiden kahden telegrammin jälkeen väylä synkronoi itseään ja lähetti kuusi tavua pitkän tyhjän telegrammin (*start delimiter* SD1). Kuvassa 5.1 on esitetty väylän syklin rakenne.



Kuva 5.1 Yhden syklin aikana liikkuvat telegrammit

Kuvassa 5.1 näkyvät ajat t_1, t_2 ja t_3 riippuvat väylän nopeudesta. Esimerkiksi 1.5 Mbit/s väylänopeudella t_1 = 9 Tbit, $t_2 \approx 819$ µs ja t3 ≈ 205 µs. 6 Mbit/s väylänopeudella kyseiset ajat olivat t_1 = 4 µs, $t_2 \approx 850$ µs ja t3 ≈ 112 µs. Viimeisen telegrammin (*telegram* 3) ja sitä seuraavan telegrammin (*telegram* 1) välillä väylällä ei ollut liikennettä ja väylä oli ylhäällä.

Telegrammit 1 ja 2 olivat kuvan 3.2 mukaisia muuttuvan datamäärän telegrammeja, joten niiden siirtämiseen kulunut aika on

$$t = 9 \cdot 11\text{bit} \cdot T_{\text{bit}} + 2 \cdot 11\text{bit} \cdot T_{\text{bit}} = 110 T_{\text{bit}} \qquad (5.1)$$

Telegrammi 3 oli kuuden tavun pituinen telegrammi, jonka *start delimiter* -lohko oli SD1. Kyseinen telegrammi määritellään PROFIBUS-standardissa tyhjäksi telegrammiksi, joka ei sisällä lainkaan informaatiota. Kyseisen telegrammin siirtämiseen vaadittu aika on tällöin

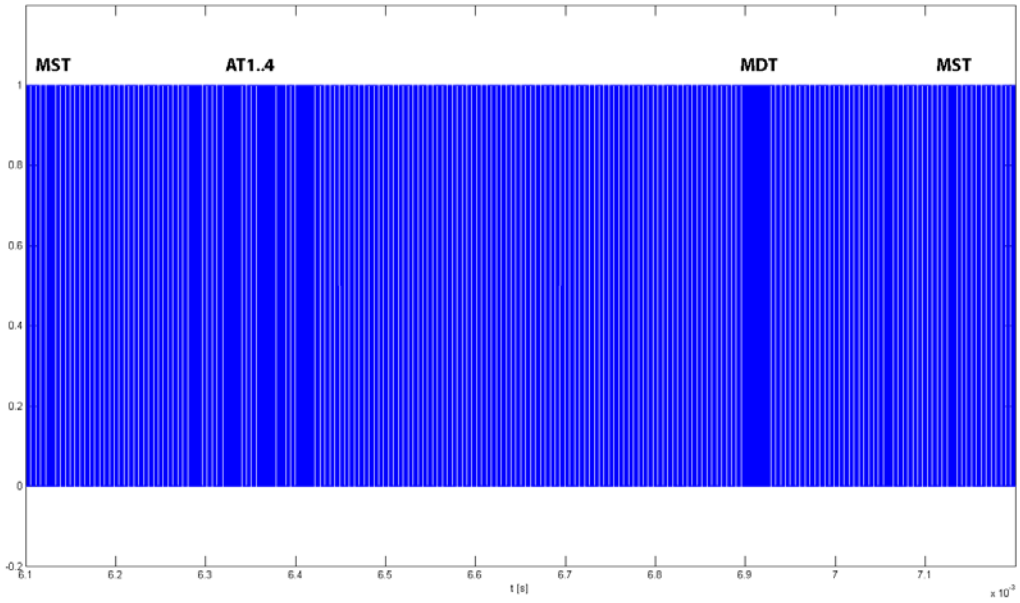
$$t = 6 \cdot 11\text{bit} \cdot T_{\text{bit}} = 66T_{\text{bit}} \tag{5.2}$$

Suuremmilla väylänopeuksilla eli 6 ja 12 Mbit/s oli selkeästi havaittavissa myös jitterin vaikutus. PROFIBUS-standardi määrittelee jitteriksi luokkaa µs, joka on huomattava erityisesti suhteessa pulssileveyteen. Logiikka-analysaattorilla olikin havaittavissa pelkkä pulssileveyden vaihtelu selvästi ja näin ollen myös yhden tavun siirtämiseen kulunut aika vaihteli.

5.2 SERCOS II

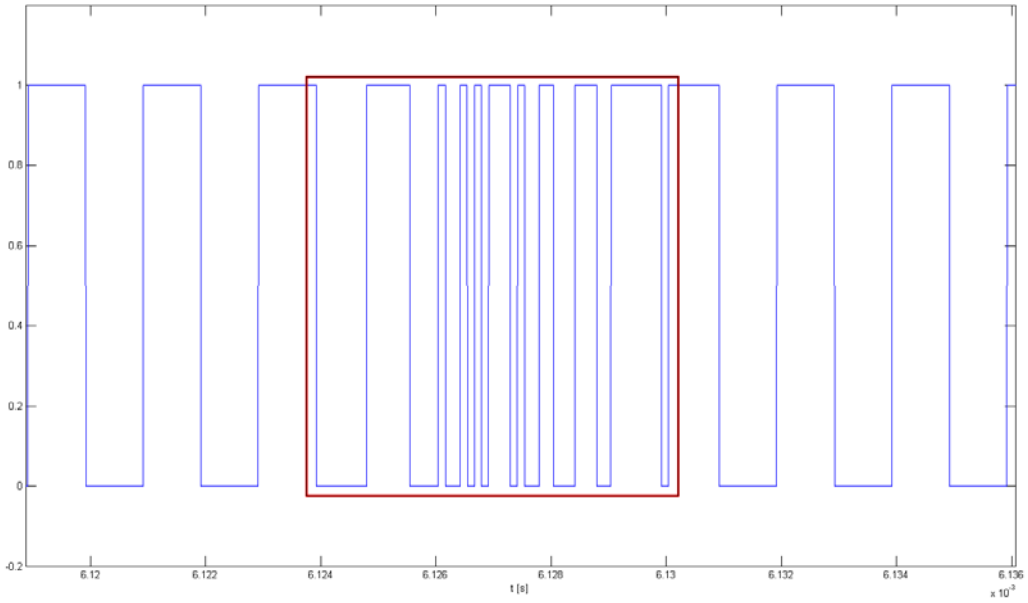
SERCOS-väylän toimintaa mitattiin kuvan 2.1 koelaitteistossa. Väylällä liikkuva signaali mitattiin logiikka-analysaattorilla yhden taajuusmuuttajan lähetinyksiköstä. Väylän tiedonsiirtonopeus oli 8 Mbit/s. Logiikkalaitteisto ajoi neljää taajuusmuuttajaa, jolloin logiikka toimi isäntälaitteena ja taajuusmuuttajat orjina. Tällöin isäntä lähetti kuvan 4.2 telegrammit MST ja MDT ja orjat kukin oman AT-telegramminsa.

Koelaitteistoa käyttävä säätö oli asetettu päivittämään uudet asetusarvot ja saamaan takaisinkytkentäarvot yhden millisekunnin välein. Kuvassa 5.2 on logiikka-analysaattorin mittausdata noin yhden millisekunnin ajalta, jossa on havaittavissa kuvan 4.2 mukaiset telegrammit.



Kuva 5.2 SERCOS-väylän tiedonsiirtosykli. Kuvassa näkyvät tummemmat kohdat ovat siirrettäviä telegrammeja. Ajanhetkellä noin 7.05 näkyvä tumma alue ei sisällä dataa, vaan kaksi pientä häiriöpiikkiä.

Kuvassa 5.2 näkyviä telegrammeja lähemmin tarkasteltaessa voidaan havaita telegrammien sisäisen rakenteen olevan standardin mukaisia. Esimerkiksi MST-telegrammin rakenne on esitetty kuvassa 5.3.



Kuva 5.3 MST-telegrammin rakenne

Kuvan 5.3 telegrammi on NRZI-koodattu. Fyysisessä siirrossa on käytetty myös *bit stuffing* -tekniikkaa, joten telegrammin sisältöä ei PROFIBUS-väylän tavoin voida nähdä suoraan mittausdatasta. Esimerkiksi liitteessä I olevaa C-kielistä ohjelmaa käyttämällä saadaan MST-telegrammin sisällöksi 01111110 11111111 00100000 11000101 01101101 01111110_B.

Vastaavasti saadaan esimerkiksi AT2-telegrammin sisältämälle datalle 0111110 01000000 10000000 00000001 00000000 00000000 11111001 00100010 00000000 00000000 10001000 10000000 00000000 00000000 10000110 11111011 01111110_B. Telegrammista voidaan löytää siirretyt tavut kuvaan 4.4 vertaamalla. Telegrammi sisältää siis kahdeksan datatavua, jotka ovat 32 bittiset paikan ja nopeuden oloarvot.

Kuvasta 5.1 voidaan havaita siirrettyjen telegrammien välissä olevan huomattavasti käyttämätöntä aikaa, joten teoriassa järjestelmän syklinaikaa voidaan pienentää merkittävästi. Myös väylän tiedonsiirtonopeuden kasvattaminen 16 Mbit/s puolittaa käytettävissä olevan syklinajan.

JOHTOPÄÄTÖKSET

Kandidaatintutkielman lähtökohtana ei sinänsä ollut selvittää kumpi tutkittavista väylistä on nopeampi, sillä etukäteen oli tiedossa SERCOS-väylän suurempi tiedonsiirtokapasiteetti. Työn tarkoituksena olikin muodostaa protokollien kehysrakenteiden pohjalta analyyttiset yhtälöt, joiden perusteella voidaan arvioida syklisen tiedonsiirron syklinajat.

PROFIBUS DP ja SERCOS II -kenttäväylille saatiin muodostettua suuntaa-antavat yhtälöt mahdollisille tiedonsiirtonopeuksille sekä todettua mittausten avulla muodostettujen yhtälöiden paikkansapitävyys.

SERCOS-väylän telegrammien koko on pienempi, kuin PROFIBUS-väylällä, jonka lisäksi SERCOS-väylän erilainen tiedonsiirtoperiaate tehostaa väylän toimintaa huomattavasti. Toisin, kuin PROFIBUS-väylän, SERCOS-väylän ei tarvitse lähettää omaa kehystä jokaiselle orjalaitteelle, jolloin tiedonsiirtosyklin aikana siirrettävän tiedon määrä vähenee oleellisesti.

LÄHTEET

[1] IEC/TR 61158-1, Edition 2.0, 2007-11

[2] Saarakkala, Seppo, Diplomityö, Lineaarisen hammashihnaservokäytön tilasäätö, LUT, 2008

[3] PROFIBUS DP Adapter Module FPBA-01, User's manual, ABB, 2005

[4] http://www.automation.com/images/article/Profibus_Introduction_698A.doc, 26.12.2008

[5] IEC 61491, Second edition, 2002-10

[6] Sercos Interface, Pack Profile Specification, Version 1.1 (Decempber 22, 2003)

LIITE I

```
/* Ohjelma dekoodaa NRZI ja bit stuffing -koodatun signaalin.
   Esimerkki ohjelman käytöstä:
     Encoded bits: 10000000111111000010010101110100110001110011111110

     Bit stuffing error!
     Bit stuffing error!

     Decoded bits: 011111B0 11111111 00100000 11000101 01101101 011111B0
*/

#include <iostream>

using namespace std;

int main(void) {

  int i=0, j=0, k=0, ones=0;
  char encoded[1000], decoded[1000];

  cout << "Encoded bits: ";
  cin >> encoded;

  while(encoded[i] != '\0') {
    if(encoded[i] == encoded[i+1]) {
      decoded[j] = '1';
      ones++;
      if(ones == 5) {
        ones = 0;
        i++;
        if(encoded[i] == encoded[i+1]) {
          decoded[++j] = 'B';
          k++;
          cout << endl << "Bit stuffing error!";
        }
      }
    }
    else {
      decoded[j] = '0';
      ones = 0;
    }
    i++;
    j++;
    k++;
    if(k == 8) {
      decoded[j] = ' ';
      j++;
      k = 0;
    }
  }

  decoded[j-1] = '\0';
  cout << endl << endl << "Decoded bits: " << decoded << endl << endl;
  return 0;


}
```