

FÖRSTUDIE AV PROGRAMMET S L I R S I M.

NILS HEDER

TONI SPEIDEL

INSTITUTIONEN FÖR REGLERTEKNIK
LUNDS TEKNISKA HÖGSKOLA

MARS 1982

LUND INSTITUTE OF TECHNOLOGY DEPARTMENT OF AUTOMATIC CONTROL Box 725 S 220 07 Lund 7 Sweden	Document name	
	Master thesis	
	Date of issue	
	March 1982	
Author(s) Nils Heder Toni Speidel	Document number	
	CODEN: LUTFD2/(TFRT-5269)/1-086/(1982)	
	Supervisor	
	Björn Wittenmark	
	Sponsoring organization	
Title and subtitle Förstudie av programmet SLIRSIM. (Prestudy of the program SLIRSIM.)		
Abstract		
Key words		
Classification system and/or index terms (if any)		
Supplementary bibliographical information		
ISSN and key title		ISBN
Language	Number of pages	Recipient's notes
Swedish	86	
Security classification		

DOKUMENTATABLAD RT 3/81

Distribution: The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

ASEA

Författare - Author

Nils Hedar, Toni Speidel

Godkännare - Approved by

Per Leander, TKU *PL*

Uppdragsgivare - Requested by

T Nordin, TK *TN*

Titel - Title

Prestudy
Program SLIRSIM

Teknisk rapport Technical Report

Från - From Datum - Date

TKU 82-03-11

Utredning, teoretisk
undersökning - Analysis,
theoretical investigation

Provnings, experim. under-
sökning - Test, experi-
mental investigation

Delrapport

Slutrapport
Provnings/undersökning avslutad
Test/investigation finished

TR TK 8204

Reg. Sida - Page

61014,6135 1

Ordernr - Ref. No.

831.210-47

Debiteras ordernr

PKI/AKI

645

Antal textsidor - No. of pages of text

75

Antal bilagesidor - No. of supplm. pages

11

Sammanfattning - Summary

The purpose of this work has been to develop a computerprogram, which can simulate the action of slip, and the slip regulation on ASEAs electrical locomotives Rc4.

So far such studies has mainly been carried out by practical tests. However, because of all the unknown factors, which affect the behaviour of the locomotive, the results of these tests have a limited value for an optimization of the system.

For further development of the regulation system and study of its properties, ASEA has a desire to create a complete model of a locomotive's mechanical and electrical behaviour.

The mechanical part already existed in the shape of a FORTRAN program, called SIMSLIP.

The result of this work is a program called SLIRSIM. It contains the most important parts of the electrical main circuits and the regulation system and together these two programs make a first attempt in the long work towards a final model.

Distribution

TKMA YTKU Almqvist Inst. för Reglerteknik, LTH

Enbart sida 1 - Page 1 only

KSB

Nyckelord - Ämnesord

Lok
Simulering
Adhesion
Slirning

Keywords

Övriga nyckelord

Anm. Huvudregeln är att nyckelorden skall skrivas på svenska i vänstra kolumnen.
I kolumnen för keywords införs då så önskas engelska motsvarigheter till ämnesorden samt engelska uttryck utan svensk motsvarighet.
Med övriga nyckelord avses t ex materialbeteckning, produktbeteckning, leverantör, kund, etc.

Den här handling är ett avsnitt av ett större dokument. Den är ett avsnitt av ett större dokument. Den är ett avsnitt av ett större dokument.

This document must not be copied without our written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose. Contravention will be prosecuted. ASEA

Ansvarig: KSB

6917 0300-BA (930) 80-04 8000

Innehållsförteckning

Sida

1.	INLEDNING
2.	FORDONSBESKRIVNING
2.1	Allmänt
2.2	Huvudkretsar, principfunktioner
2.2.1	Tyristorn
2.2.2	Strömriktaren
2.2.3	Fältströmriktaren
2.2.4	Traktionsmotorn
2.2.5	Glättningsinduktorn
2.2.6	Principiella strömkurvformer
2.2.7	Telefilter
2.3	Reglerutrustning
2.4	Mekanisk del
2.4.1	Boggi
2.4.2	Motorupphängning
2.4.3	Kraftöverföring
2.4.4	Svängningar i mekaniskt system
2.5	Adhensionsutnyttjning
2.5.1	Adhesion
2.5.2	Dragkraftsuttag, adhesionsutnyttjning
2.5.3	Torsionssvängningar i hjulaxeln
3.	PROBLEMBESKRIVNING
4.	MODELLBESKRIVNING
4.1	Transformatorn
4.2	Glättningsinduktorer
4.3	Likströmsmotorn
4.4	Tyristorbryggorna
4.5	Ekvationsunderlag för rotorströmkringarna
4.5.1	Modell med 1 motor
4.5.1.1	Frihjulning
4.5.1.2	Ledning
4.5.1.3	Kommuteringsintervall
4.5.1.4	Bestämning av tändvinklar vid uppsty- ringens början
4.5.2	Ekvationsunderlag för modell med två motorer
4.6	Reglering
5	PROGRAMBESKRIVNINGAR
5.1	Simuleringsprogram SIMPAC/SIMPLOT
5.2	Programpaket SIMSLIP
5.3	Program 1MOTOR
5.4	Förändringar i SIMSLIP i samband med byte av MOTOR-modellen (1MOTOR)
5.5	Program 2MOTOR
5.6	Modificeringar av 2MOTOR-modellen för SIMSLIP
6	SIMULERINGAR

Bilagor

- 1 Allmänna instruktioner för körning av programmen
- 2 Program EMOTORX (modell av endast en motor)
- 3 Program SMOTORX (modell av en motor i SIMSLIP)
- 4 Program FILT (modell av Rc-huvudkrets med 2 motorer)
- 5 Program KVASI (modell av huvudkrets med två motorer i SIMSLIP)

1

INLEDNING

Examensarbetet har avsett att utveckla ett simuleringsprogram för studium av elloks slirbeteende och slirreglersystem. Framtagning av dessa system har hittills mestadels gjorts genom praktiska prov. Resultaten har emellertid varit svårtydda, eftersom mätbetingelserna inte kunnat renodlas tillräckligt. Simuleringsprogram efodras för att kunna göra tillräckligt idealiserade studier.

För vidareutveckling av reglersystem och studium av dessa egenskaper är det önskvärt att utarbeta en total simuleringsmodell av lokets mekaniska del, eldel och dess reglering. Mekaniska problem med boggier och drivanordning har tidigt tvingat fram en simuleringsmodell (SIMSLIP) för denna del. Målsättning med detta examensarbete har varit att i anslutning till SIMSLIP påbörja arbetet med modeller för elektriskt system och reglering. I ett första steg har huvudkrets och ankarströmreglering modellerats.

Ol 9257-1
Ol 930-1
Ol 9431-72

ASEA Projekt nr 831.210-47

Anslagsbegäran Nr T-1-0042

Original till	ÖK-chef	sektorchef	registratur	ERB	Anslagsbeholdning	Pos.	Ordernummer	Lev. avd.
	TK	T	TE		20 000	1	S 43.831.210-645-4710	TKU
					14 000	2	S 43."- -4720	TKU
Godkänd och suveräniserad anslagsbegäran (order) till					20 000	3	S 46."- -4730	KD
	TK	TKU	YTKU		6 000	4	S 46."- -4740	YTK, MO
Från	Datum	Utfärdare, fn-nr						
TKU	81-06-16	P Leander 1881						
Avrop från	investeringsbudget	År						
	utvecklingsbudget	19 81						
	OK-budget							
Beräknad ekonomisk livslängd	För pgr	Reg.						
	år	645						
Begäran avser	<input checked="" type="checkbox"/> grundenslag	<input type="checkbox"/> Nyanskaffning	<input type="checkbox"/> Ersättningsanskaffning för maskin nr	<input type="checkbox"/> Rep. och underhåll	<input type="checkbox"/> Experiment och utveckling			
	<input type="checkbox"/> tilläggsanslag nr							

Rubrik för anslag (30 pos.)
Programpaket SLIRSIM (1)
 Beskrivning av projektet
 För vidareutveckling av våra slirskydd och slirreglersystem behöver totala modeller (mekdel, eldel och reglering) för datormulering tas fram. Detta första steg i framtagning av programhjälpmedel avser modellering och preliminära studier av system Rc4.
 Arbetet bedrivs som ett examensarbete vid TKU sommaren 1981.

Pos.	Specifikation av anslagsanvändningen	Beräknad kostnad	Kontering (skall anges av utfärdare)
1	Examensarbete	20 000	S 43.831.210-645-4710
2	Handledning TK	14 000	S 43."- -4720
3	Datorkostnader	20 000	S 46."- -4730
4	Handledning andra sektorer	6 000	S 46."- -4740
	Summa kr	60 000	

Motivering/Lönsamhet
 Våra loks adhesionegegenskaper har studerats och försök till optimeringar gjorts under ett flertal praktiska prov genom åren. Som regel har resultaten blivit mycket svåra att utvärdera, eftersom mätbetingelserna inte kunnat renodlas tillräckligt. Digitalsimuleringar av mekaniskt, och vissa analogmaskinsimuleringar av elektriskt system, har gjorts. Studier på totala simuleringssystemer är nödvändiga för fortsatt utvecklingsarbete inom detta område.

Ansvärlig för genomförandet	TKU P Leander	<input type="checkbox"/> Investeringsskalkyl bifogas		
Beräknad kvängsamning av projektet	År, period 81-06	Anslagsbegäran granskad	Datum	Namn
Beräknat färdigställande (skall alltid anges)	81-09	<input type="checkbox"/> Tillstyrkes		
Arbetet utföres av/Vårn levereras av	TKU	<input checked="" type="checkbox"/> Avstyrkes		
		<input checked="" type="checkbox"/> Bevillat jfr Ol 9257-1	81-06-16	<i>[Signature]</i>
		<input type="checkbox"/> Avslås		

Ansvarig EHD
 6817 0230-AA (9257) 80-11 10 000
 BI 410
 Ansvarig EHD

2

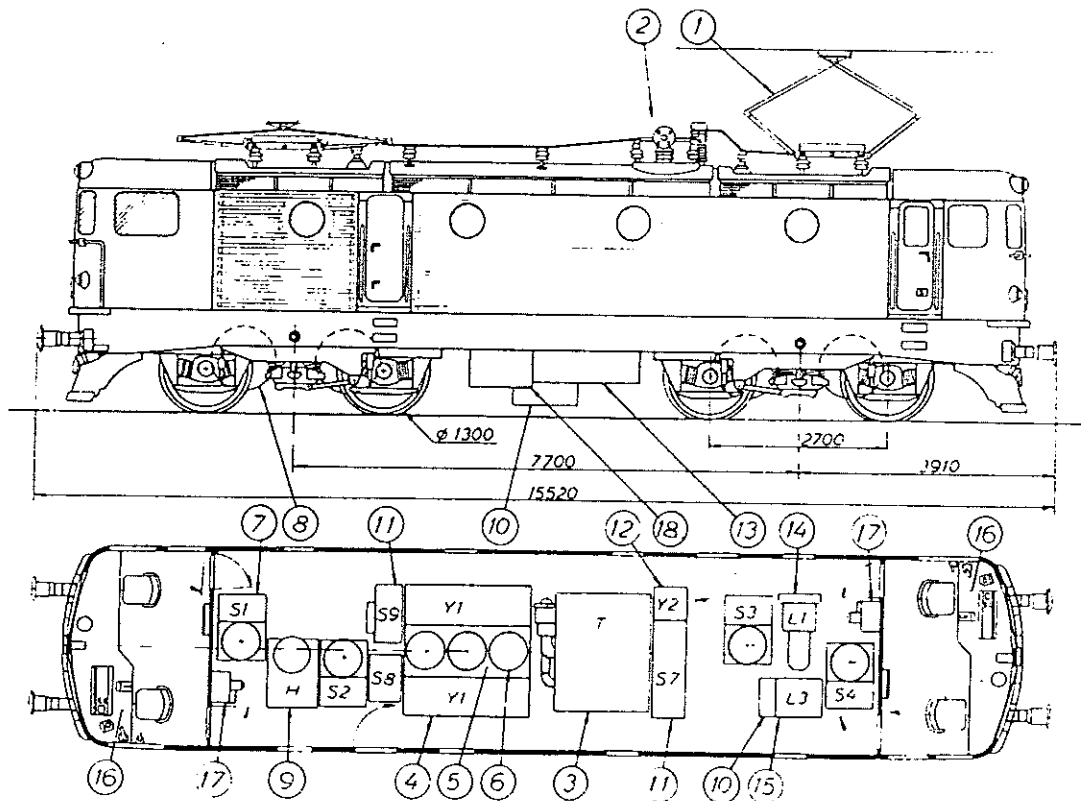
FORDONSBESKRIVNING

2.1

Allmänt

I form av några bilagda broschyrer ges här en mycket schematisk bakgrund till ASEAs lokteknik.

Uppbyggnaden av ett Rc4-lok framgår av nedanstående måttskiss, som också visar komponenterna i maskinrummet.



- | | |
|----------------------------------|---|
| 1 Strömvtagare | 11 Apparatstativ |
| 2 Huvudbrytare | 12 Skåp med styr- och reglerutrustning och laddningsdon |
| 3 Huvudtransformator med reaktor | 13 Batterier |
| 4 Strömriktare | 14 Kompressor |
| 5 Oljekylare | 15 Tryckluftapparatstativ |
| 6 Ventilator för 4 och 5 | 16 Förarplats |
| 7 Stativ med ventilator för 8 | 17 Hyttvärmeaggregat |
| 8 Traktionsmotor | 18 Kondensatorer för filter |
| 9 Statisk omriktare | |
| 10 ATC | |



(F 93597)

The experience obtained with the first twenty class Rc 1 locomotives has led to the fitting of oil-cooled thyristor converters and harmonic filters on 220 new locomotives. At the same time the number of converter bridges connected in the series has been reduced to two. These locomotives have been given the class designations Rc 2–Rc 4 (135 km/h) and Rc 3 (160 km/h).

The class Rc 4 differs from class Rc 2/3 as follows: The smoothing chokes are oil-cooled. The thyristor converter contains fewer and more powerful semi-conductors. The latest design of electronic equipment with integrated circuits is used and ASEA PRESSDUCTOR® Creep Control system is another innovation.

Common data Rc 2, Rc 3 and Rc 4

Line frequency and voltage		16 2/3 Hz, 15 kV
Gauge	mm	1,435
Driving wheel diam. (new)	mm	1,300
Bogie wheelbase	mm	2,700
Total wheelbase	mm	10,400
Height over pantograph, down	mm	4,500
Max. width	mm	3,150
Length over buffers	mm	15,520
Number of traction motors		4
Traction motor control		thyristors
Transmission		ASEA hollow-shaft motor drive
Auxiliary machines		50 Hz, 3-phase
Brake system		Disc brake with tread brake on all wheels

Special data

		Rc 2:	Rc 3:	Rc 4:
Delivered		1969	1970	1975
Max. speed	km/h	135	160	135
Gear ratio		87:26	106:37	87:26
Continuous transformer rating excluding heating and auxiliaries	kVA	3,910	3,910	4,680
Rating*) according to IEC 349	kW	3,600	3,600	3,600
Speed (wheel diam. 1,260 mm full field)	km/h	78	93	76
Tractive effort at wheel rims	kN	165	140	170
Max. starting effort	kN	280	235	290
Axle load max.	tonnes	19.4	19.4	19.5
Weight, total	tonnes	77	77	78

*) Tractive effort curve, see reverse

Rc4-loken är baserade på en Rc-serie som 1967 startade med Rc1 och följdes sedan av Rc2 och Rc3.

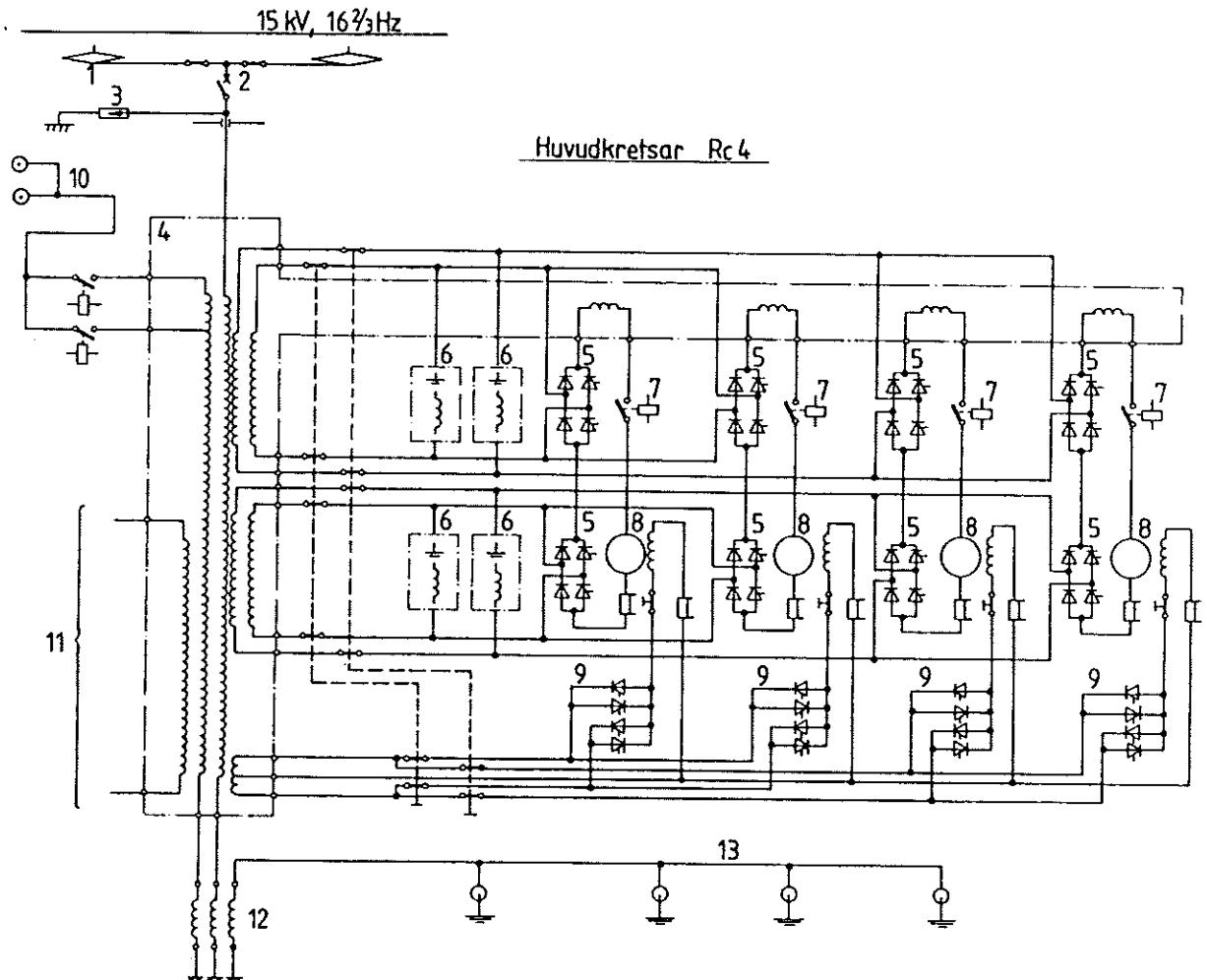
Skillnaden mellan Rc1 och Rc2/3 var att i Rc1 fanns en luftkyld strömriktare samt att antalet bryggor per motor var tre. I Rc2/3 blev strömriktaren oljestyld och antalet bryggor reducerades till två. På Rc4 infördes en ny generation apparatutrustning, i övrigt är det ingen större skillnad. Alla Rc-lok är växlade för en hastighet av 135 km/h utom 10 st Rc3-lok som är växlade för 160 km/h.

Fram till i dag har det tillverkats ca 250 Rc-lok.

2.2

Huvudkretsar, principfunktioner

Huvudkretsarna på Rc4-loken är uppbyggda enligt följande schema:



- | | |
|------------------------------|------------------------------|
| 1 Strömavtagare | 8 Sep.magn. traktionsmotorer |
| 2 Huvudbrytare | 9 Fältströmriktare |
| 3 Ventilavledare | 10 Tågvärme 800 eller 975 V |
| 4 Huvudtransformator | 11 Hjälpkraftanslutning |
| 5 Halvstyrda tyristorbryggor | 12 Jordningstransformator |
| 6 Telefilter | 13 Återledningsdon |
| 7 Motorkontaktor | |

2.2.1

Tyristorn

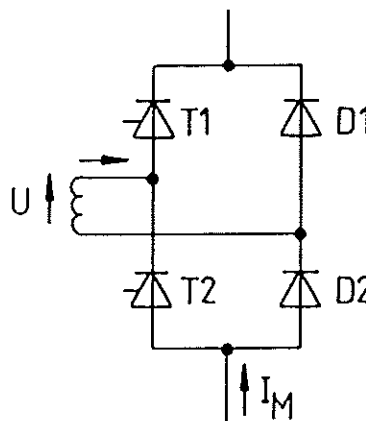
En tyristor fungerar som en styrd likriktare. Den "släckta" tyristorn spärrar för ström i båda riktningarna. Då man ger den en s k tändpuls börjar den fungera som en diod, tills dess att strömmen genom den blir noll. I den s k nätkommuterade tyristorströmriktare vi har här, sker släckningen automatiskt då nätspänningen (växelspänning) byter polaritet.

Genom att variera tidpunkten för tändpulsens kan man således styra medelströmmen genom lasten.

2.2.2

Strömriktaren

Ankarströmriktaren fungerar som en likriktarbrygga, där två dioder ersatts med styrda ventiler (T1 och T2).



Bryggans funktion kan delas in i fyra ledningstillstånd:

1. Motorströmmen frihjulrar genom D2 och D1. I_M drivs inte uppåt av nätspänngen U .
2. T1 tänds och I_M switchar över till D2 \rightarrow spänningskällan \rightarrow T1. Tillståndet kallas påkommutering. (Med kommutering avses det förlopp då strömmen byter väg. Jämför en motors kommutator).
3. I_M går genom D2 \rightarrow spänningskällan \rightarrow T1. Fullt bidrag från U fås. Tillståndet kallas ledning.
4. I_M switchar tillbaka till D2 och D1. Detta sista tillstånd kallas avkommutering.

Av praktiska skäl är strömriktarbryggorna för två motorer parvis parallellkopplade på transformatorns sekundärlindningar. För att undvika cirkulerande strömmar styrs bryggorna för respektive motor ut parallellt.

Huvudkretsarna består för varje motorankare av två stycken kaskadkopplade halvstyrda enfas tyristorbryggor. De matas med spänningen 487 V 16 2/3 Hz. Bryggorna ger maximalt 2080 A rms.

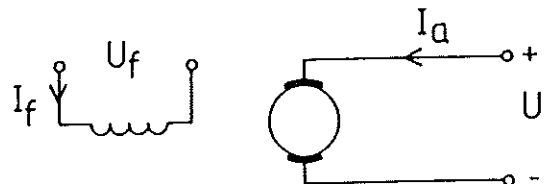
2.2.3

Fältströmriktaren

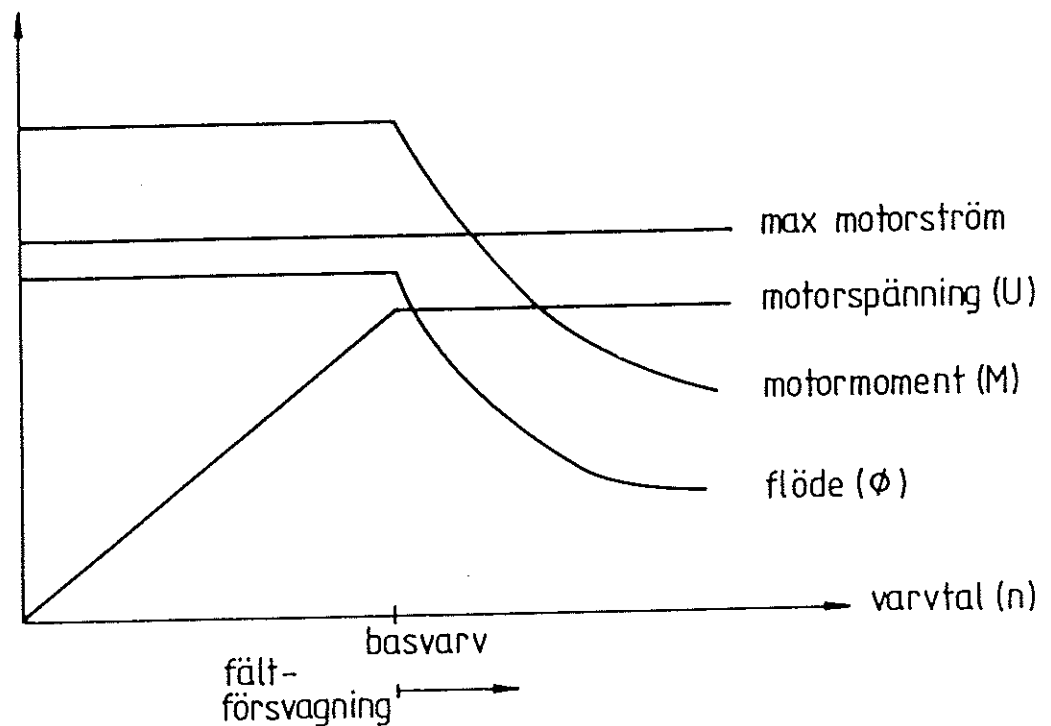
Varje motorfältlindning matas via en dubbelströmriktare. Tyristorbryggan matas med 2 x 37,5 V och ger 300 A kontinuerlig likström. Fältströmriktarna är bestyckade med antiparallellkopplade tyristorer, vilket gör det möjligt att styra fältströmmen i valfri riktning genom motors magnetiseringslindning. Detta för att kunna reversera lokets färdriktning.

2.2.4 Traktionsmotorn

Traktionsmotorerna är separatmagnetiserade likströmsmotorer och av typ L J H 108-6 (eller -2). Kontinuerlig effekt är 900 kW. Separatmagnetisering är en av likströmsmotorerna fyra grundkopplingar och innebär att den fordrar en separat spänningskälla för magnetiseringen. Principen är ganska ovanlig, men har fördelen att man kan styra motorn både via U_f och U (mer om detta följer senare).



Motorernas styrning som funktion av varvtal framgår av nedanstående principiella diagram.



Vid basvarv har max motorspänning uppnåtts. Detta innebär att bryggorna är maximalt utstyrda, vilket sker vid ungefär 70 km/h. Vill man då öka hastigheten ytterligare måste man fältförsvaga, dvs minska flödet. Detta beror på att $U \sim n \cdot \phi$ där U ej kan ökas, men dock hållas konstant. Eftersom $M \sim I \cdot \phi$ får man ett lägre moment, med sämre acceleration som följd. Fältförsvagningen sker via fältströmriktarna.

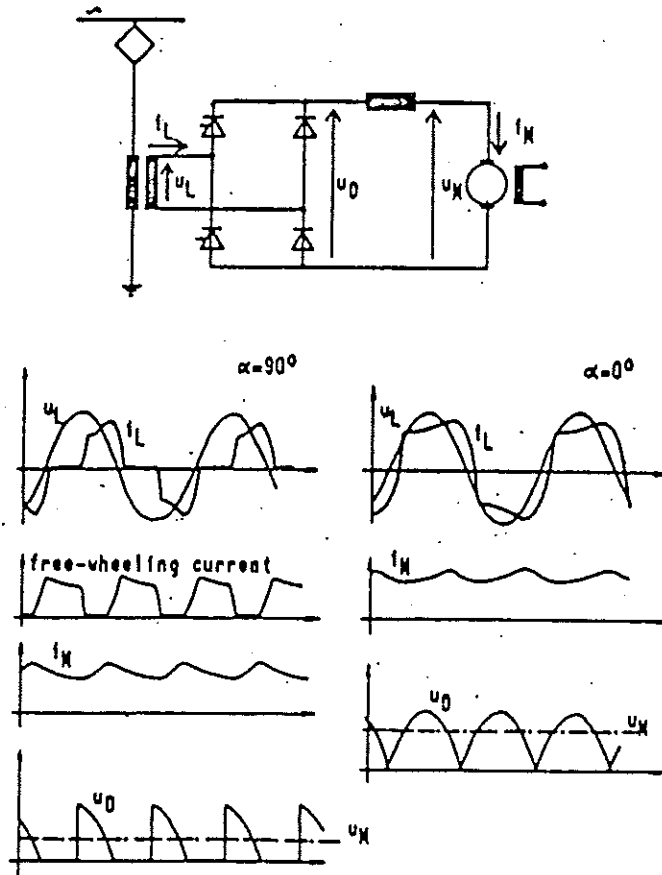
2.2.5 Glättningsinduktor

Enfas växelströmsmatning till loket representerar ett pulserande effektflöde. Likströmsmotorn kräver en närmast konstant effektmatning. För att strömmen genom motorn skall vara något så när konstant har man infört en s.k. glättningsinduktor i ankarströmkretsen. Under thyristorbryggans ledningsintervall driver nätspänningen upp strömmen genom induktorn, dess upplagrade magnetiska energi ökas. Under frihjulningsintervallen överförs energi från induktor till motor, strömmen minskar. Induktorn dimensioneras av att strömriplet i motorn inte får överstiga 40 %. Strömriplet ger upphov till ett motormoment, som pulserar med dubbel nätfrekvens (på SJ 33,3 Hz).

Såväl transformator som glättningsreaktorer är sammanbyggda i en enhet. Detta för att förenkla konstruktionen, då framför allt med tanke på kylningen vilken sker med olja. Denna sammanbyggnad medför att en magnetisk induktiv koppling uppstår mellan dels transformatorns lindningar och dels mellan de olika glättningsinduktorerna. Strömförändringar i någon motors ankarrets, t ex vid slirning, påverkar därigenom också de andra ankarströmmarna.

2.2.6 Principiella strömkurvformer

Nedanstående figur visar principiella ström- och spänningskurvformer vid styrvinklarna 90° respektive 0° för en thyristorbrygga.



2.2.7

Telefilter

Som framgår av figurena kommer linjeströmmen att innehålla mycket övertoner och kraftigt avvika från sinusform. För att undvika telestörningar från relativt högfrekventa övertoner finns telefilter (6) på transformatorns sekundärlindningar.

2.3

Reglerutrustning

Rc4-lokets reglerutrustning framgår av beskrivningen på följande sidor, vilken saxats ur ASEAs tidning.

Ankarströmregleringen är gemensam för två motorer. Fältströmsregleringen sker individuellt på varje motor.

I huvudsak styrs motorerna genom ankarströmreglering. Under basvarv sker det genom ankarströmriktarna. Över basvarv, då tyristorbryggorna är fullt utstyrda, sker ankarströmregleringen via fälten. Fältförsvagning görs också för att tillåta ökande motorvarv vid konstant motorspänning. Dessutom finns överordnade reglerloopar för lastdelningsreglering, max motorspänning och den nämnda ankarströmsregleringen via fältet. Dessa loppar verkar på fältströmsdelen. För ankarströmsdelen finns en loop för hastighetsreglering. Vidare finns ett antal slirregulatorer av skydds- eller reglerkaraktär.

ASEA AB, S-402 18, Göteborg, SWE

dernt lok håller själv inställd hastighet och dragkraft, oberoende av kontaktledningsspänning och banans utseende.

Reglerblockschemat (fig. 7) kan uppdelas i fyra funktionsenheter, nämligen en enhet för hastighetsreglering, två av varandra oberoende enheter för reglering av motorström och en enhet för slirreglering.

Reglerutrustning

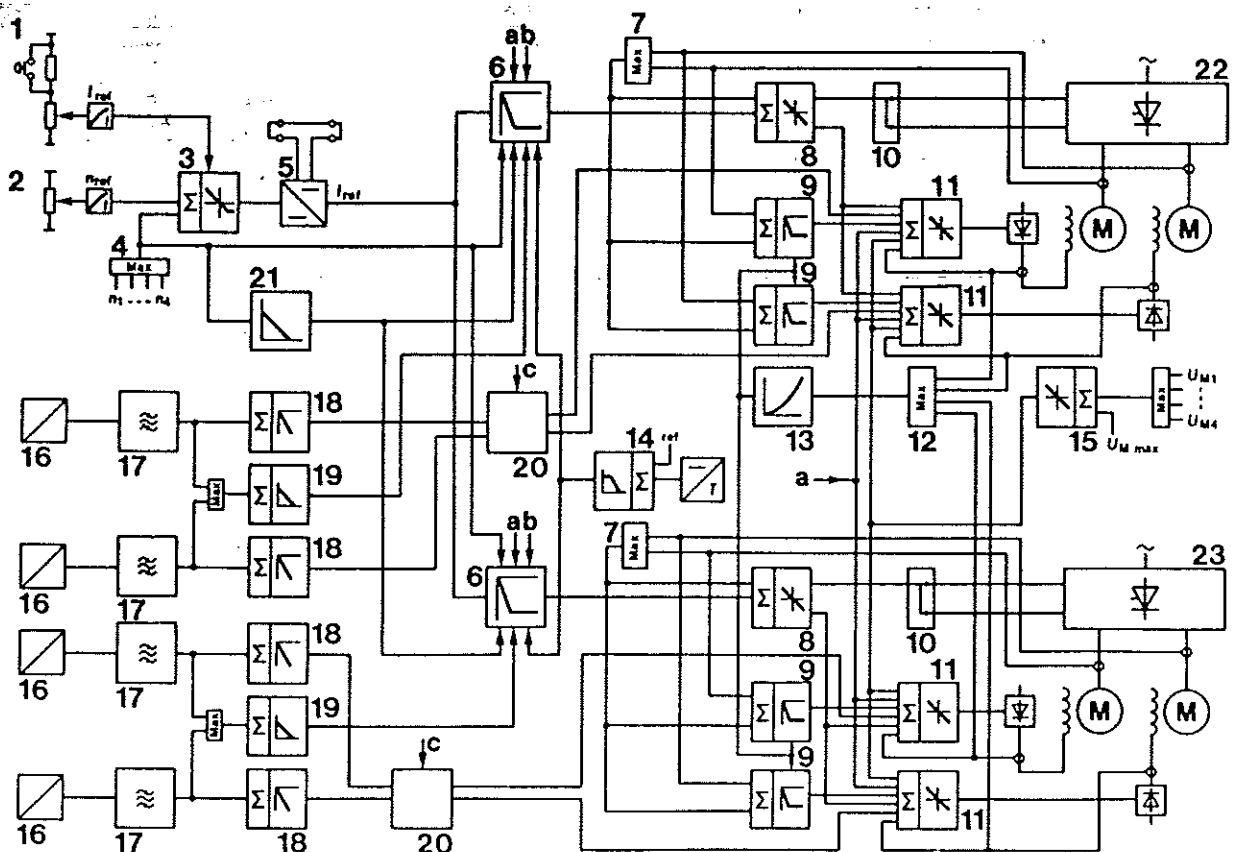
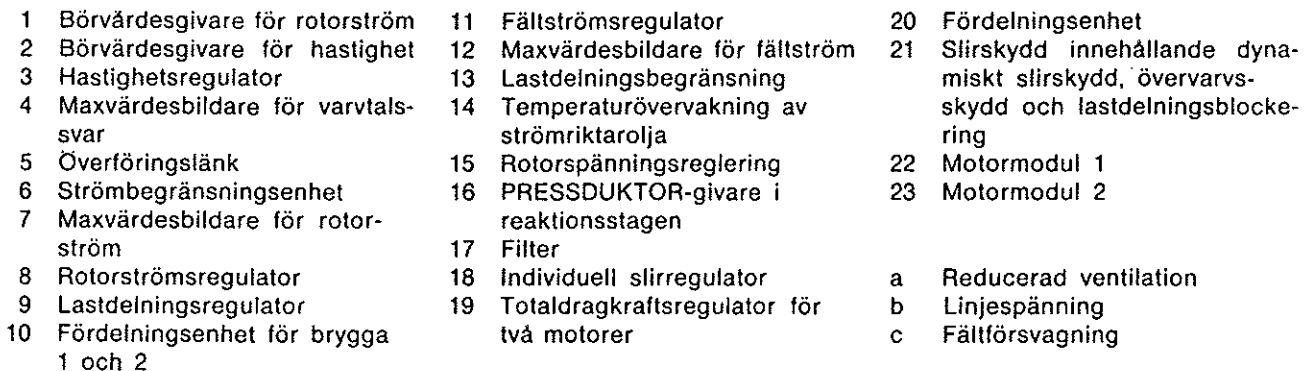
Elektronikutrustningen, som erfordras för styrningen av tyristorströmriktaren, ger stora möjligheter att påverka betydelsefulla egenskaper hos moderna tyristorfordon. I Rc-loken avlastas föraren många funktioner, som han tidigare rent rutinmässigt måste tänka på. Ett mo-

Hastighetsreglering

Loket styrs med hjälp av två reglage: dragkraftskontrollern (1) och

hastighetsreferensen (2). Hastighetsreferensen jämförs med ärvärdet av hastigheten, n_{max} (4), i hastighetsregulatorn (3), som bildar strömläsvärdet till strömregulatorerna (8). Strömläsvärdet begränsas redan i hastighetsregulatorn (3) av förarens inställning av max. dragkraft (1). Utsignalen från hastighetsregulatorn (3) går via en överföringslänk (5) till strömbegränsningsenheterna (6). Överföringslänken förmedlar strömläsvärdet även till övriga multipelkopplade lok. De olika dragenheterna kommer härigenom att dela lasten lika, dvs. alla drivande enheter har samma strömläsvärde och endast ledande lok har sin hastighetsregulator (3) i funk-

Fig. 7. Blockschemat för reglerutrustningen i ett Rc4-lok.



tion.

I begränsningsenheten begränsas strömledvärdet med hänsyn till aktuell kontaktledningsspänning, vid reducerad ventilation, vid hög hastighet, vid slirning och vid för hög strömriktartemperatur.

Rotorströmsreglering

Rotorströmsregleringen är uppdelad i två exakt lika moduler, vardera innehållande två motorer. Strömledvärdet från strömbegränsningsenheten (6) jämförs med $I_{A, \max}$ (7) för de två modulmotorerna i rotorströmsregulatorn (8). En sekvensstyrning (10) sörjer sedan för att först brygga 1 och sedan brygga 2 styrs ut. När de båda strömriktarbryggorna styrs ut fullt ges från strömregulatorn (8) fältförsvagningsignal till de fyra fältregulatorerna (11). Parallellkopplade, separatmagnetiserade likströmsmotorer fordrar speciell lastdelningsreglering för att dela lasten lika. Lastdelningsregulatorerna (9) ombesörjer detta inom varje tvåmotormodul. För att lastdelningen skall bli lika stor inom hela hastighetsområdet, alltså även inom fältförsvagningsområdet, finns en lastdelningsbegränsning (13), som på ett förenklat sätt tar hänsyn till motorernas magnetiseringskurva.

Slirreglersystem

Ofta anges ett loks teoretiska maxiprestanda som riktvärde på vad ett lok kan prestera. Maximidragkraften vid mycket goda friktionsförhållanden hjul-räl är egentligen helt ointressant. Mer väsentligt för den som skall utföra en transportuppgift är den dragkraft, som man alltid minst kan räkna med, även under de sämsta väderleksbetingelserna. Sedan flera år har ASEA därför koncentrerat sig på att ta fram ett helt automatiskt slirreglersystem.

År 1973 kunde det första slirreglersystemet med givare av typ PRESSDUKTOR[®] införas på ASEAs nylevererade tyristorlok till Österrike. Erfarenheterna av systemet var där så goda, att hela Rc-4-generationen nu beställts med detta unika slirreglersystem (se ASEA Tidning 1975:3, s. 71-73.

Slirreglersystemet bygger på att fordonets hjulsatser (hjuldrivaxlar) egensvänger med sin karaktäristiska frekvens, när ett av hjulen når maximum av överförbar kraft mellan hjul och räl. Denna svängning kan tack vare att ASEAs drivordning (fig. 8) sitter osymmetriskt på hjulaxeln upptäckas i växelns reaktionsstag. Det är alltså möjligt att mäta hjulsatsens svängningar i en icke roterande del av kraföverföringen. I reaktionsstaget är en kraftgivare av typ PRESSDUKTOR inmonterad.

Funktionen av slirreglersystemet är följande (se blockschema i fig. 7): Givarna (16) i de fyra reaktionsstagen (ett för varje motor) avger en signal till filterenheterna (17), som filtrerar och likriktar signalen. Slirregulatorerna (18) ingriper när slirsignalen blir större än ett angivet värde. Dessa individuella slirregulatorer (18) påverkar sin motors moment genom korrektioner i fältregulatorernas (11) börvärden. I fältförsvagningsområdet sker detta ingrepp genom höjning av fältströmmen i den slirande motorn, varvid rotorströmmen på grund av motorernas branta karaktäristik reduceras avsevärt, trots små förändringar i fältströmmarna. Vid ett varvtal lägre än motorernas basvarvtal kan fältströmmen inte höjas ytterligare, emedan motorerna redan har full fältström. Den motorindividuella slirregleringen måste då gå via rotorströmsregulatorerna, initierad av en sänkning av fältströmmen i icke slirande motor. Enheten (20) kopplar om mellan de båda styrförfarandena med fältförsvagningsignalen som styrsignal.

Den axelindividuella slirregleringen kompletteras av en modulindividuell reglering, som verkställs av en regulator (19), som har den maximala av modulens båda slirsignaler som ärvärde. Ingreppsnivån på regulatorn (19) är högre än motsvarande nivå på den axelindividuella regulatorn (18).

ASEAs slirreglersystem ger loken möjlighet till att under alla väderleksförhållanden maximalt utnyttja tillgänglig adhesion, utan att föraren behöver ingripa. Mätningar har visat, att systemet ger avsevärda förbättringar av dragkraftsuttaget.

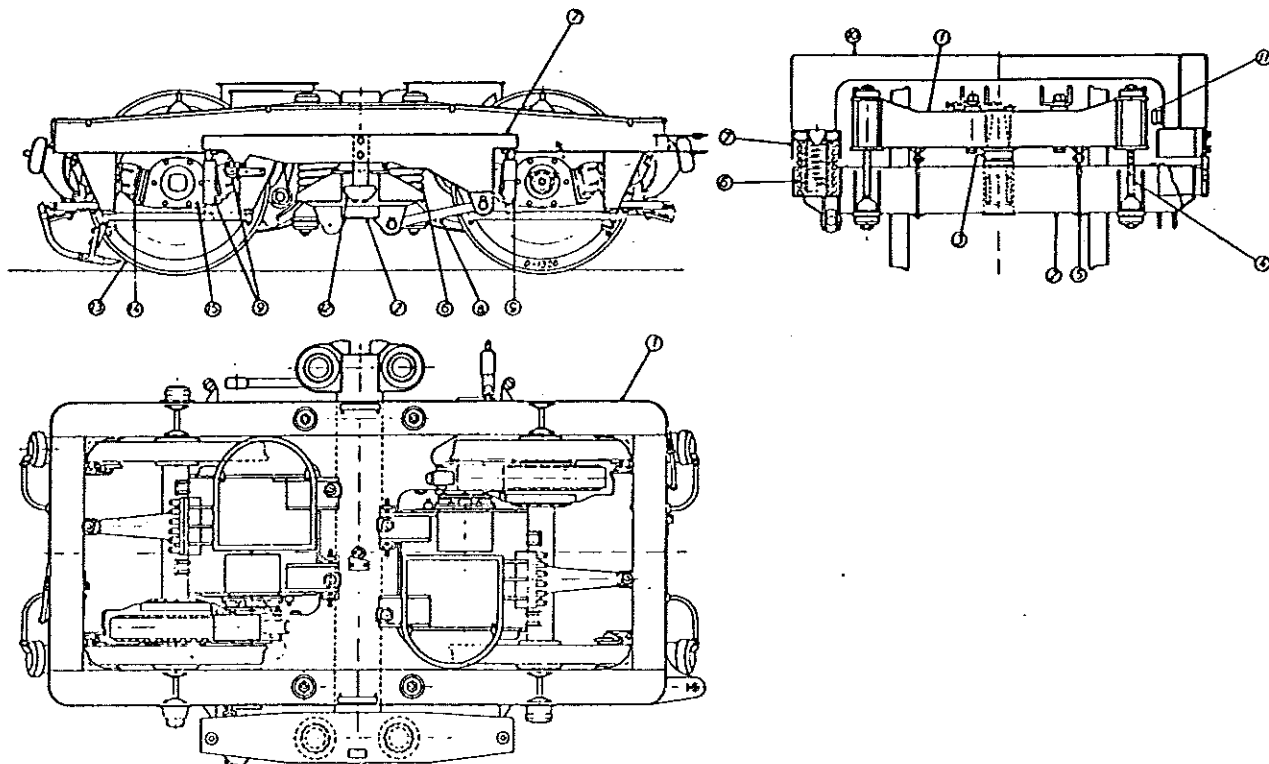
Förutom dessa fördelar, som enkelt handhavande och maximalt dragkraftsuttag, ger det beskrivna slirreglersystemet mycket mindre underhall på den mekaniska konstruktionen,

kopplingar, lagringar m.m. Dessa egenskaper erhålls därför att alla mekaniska vibrationer rent allmänt dämpas av slirreglersystemet.

2.4 Mekanisk del

Vi utnyttjar här i huvudsak ASEAs beskrivningar.

2.4.1 Boggi



Boggiramen (1) är förbunden med vaggbalken (2) dels genom en gummilagrad centruntapp (3) dels genom totalt fyra pendlar, två vid vardera sidan (4). Som extra säkerhet finns dessutom två säkerhetslinor (5).

Vaggbalken är i sina ändrar utformad så att den bildar stödytor för två dubbla skruvfjädrar vid vardera sidan (6). På dessa skruvfjädrar är sedan ett ok placerat (7), vilket är förbundet med vaggbalken med en gummilagrad dragstång (8) samt via 3 stötdämpare (9), två vertikala och en horisontell direkt med boggiramverket.

Lokkorgen (10) stöder sedan direkt på oket och styrs där av koniska tappar.

Dragkraften från boggin överförs via centruntappen till vaggbalken och därifrån via dragstängerna till oket och därmed till korgen.

Skruvfjädrarna tar upp både vertikala och horisontella rörelser mellan korg och boggi. För att begränsa sidorörelsen finns en gummibuffert (11) monterad i korgen, en på vardera sidan om boggin. Vridningen av boggin i förhållande till korgen vid körning i kurvor möjliggörs genom vridning i centruntappens gummibussning samt genom att vaggpendlarna med sina halvsfäriska gummielement tillåter en viss vridning.

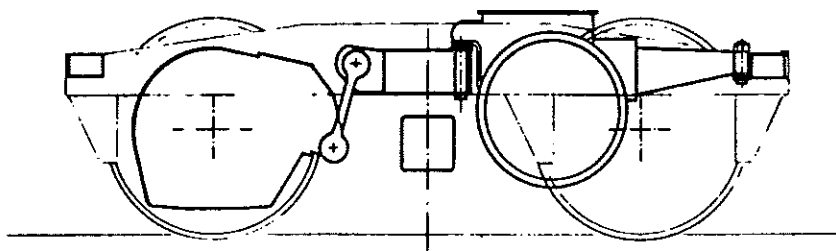
För att ge loket goda gångegenskaper, har 6 hydrauliska dämpare per boggi monterats. Fyra av dessa är vertikala medan två är monterade horisontellt.

Hjulsatserna (13) är avfjädrade och styrda i förhållande till boggiramverket med hjälp av snedställda, plåtarmade gummi-element (14), som sitter mellan lagerbox (15) och boggiramverk. Dessa gummi-element består av gummi med mellanliggande plåtar och dessa element möjliggör en erforderlig vertikal nerfjädring samtidigt som de även tillåter den axiella rörelse av axlarna i förhållande till ramverket som krävs vid kurvgång.

2.4.2

Motorupphängning

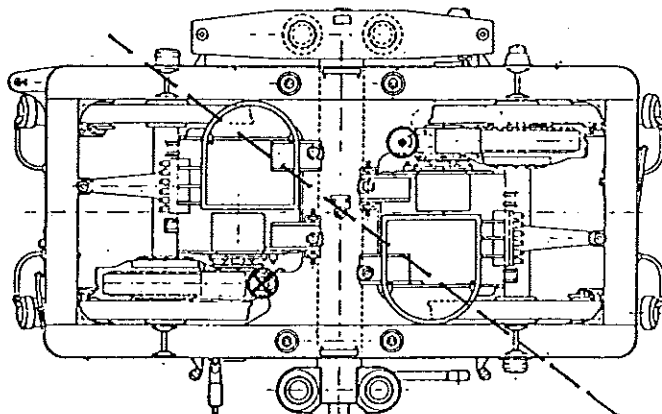
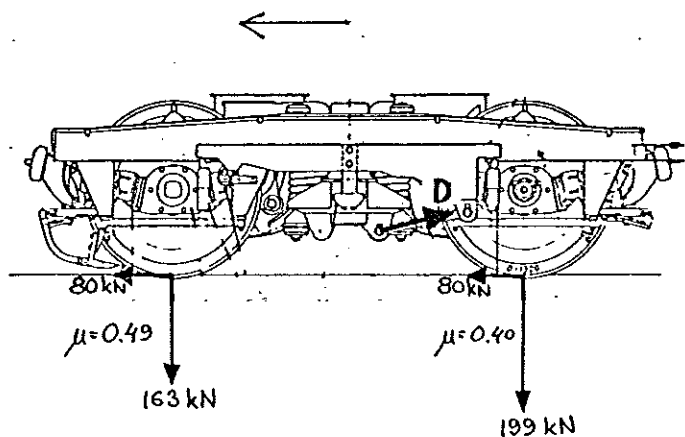
Motorerna, som väger vardera ca 3 ton, är upphängda i boggiramen avfjädrade från hjulsatserna. Detta ger lägre skovfjädrad vikt, vilket minskar spårkrafterna:



Växeln är fast monterad på hjulaxlen och för att ta upp reaktionskrafterna finns ett stag som håller växeln i förhållande till boggiramverket. Se ovanstående figur.

I detta stag är den PRESSDUKTOR inbyggd som ingår i det system som finns i Rc4-loken för att göra det möjligt att maximalt utnyttja tillgänglig adhesion.

Dragstångs- och reaktionsstagskrafterna i samband med dragkraftsuttag innebär en snedställning av boggiramen. Detta framgår av nedanstående figur, där ett fall med förhållandevis hög dragkraft (80 kN/axel) markerats med bl a några sifferangivelser om vertikala hjullaster och resulterande friktionskoefficienter.



Hjulaxlarnas av- och pålastning beror framförallt på dragkrafts- respektive bromskraftsuttag i staget D.

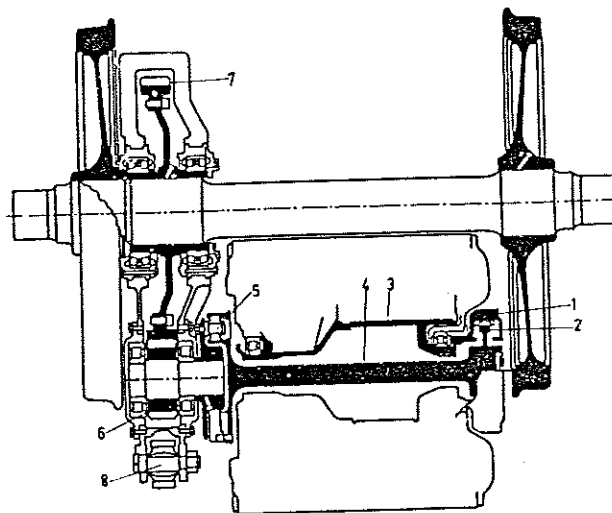
2.4.3 Kraftöverföring

Drivdon

Figuren på sidan 3 visar det system med rotorhåtaxeldrift som ASEA använder på stora elektriska linjelok för överföring av drivkraft från traktionsmotorn till växeln. Drivkraften från den i boggin upphängda motorn överförs via en tandkoppling till en genom rotorns centrum gående torsionsaxel och därefter över en gummikoppling till en fri på lokaxeln lagrad växel. Traktionsmotorns stator har på den ena sidan 2 st. upphängningstassar för fastskruvning i boggiramen och på den andra sidan (lokaxelsidan) fästen för en konsol som utgör den tredje fästpunkten i boggin. På samma sida finns dessutom 2 st. fångklackar ovanför lokaxeln.

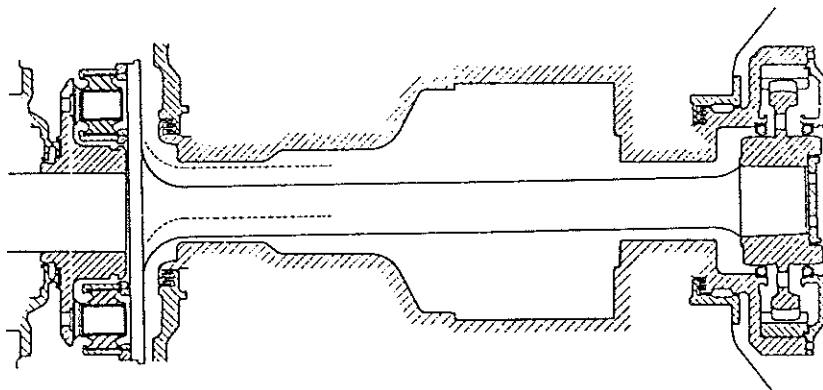
Växels drevände är rörligt upphängd i boggiramen medelst ett gummilagrät reaktionsstag. De flexibla kopplingarna i kraftöverföringen upptar de rörelser av olika slag som uppkommer mellan motor och växel vid boggens fjädring och hjulaxelns rörelser på grund av banans ojämnheter. Tandkopplingens axiella rörlighet medger dessutom att hjulaxeln kan röra sig transversellt relativt boggiramen. Axelns och gummikopplingens torsionsfjädring utjämnar eventuella pulsationer i rotormomentet och ger systemet ett önskvärt lågt egensvängningstal.

Lokmotor med drivdon (1378 088)



1. Oljefylld tandkoppling med kuggring
2. Tandkopplingens drev
3. Rotorhåtaxel
4. Torsionsaxel
5. Gummikoppling
- 6-7. Växel
8. Växellager

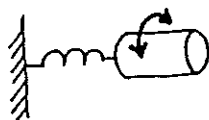
Lokmotorns momentöverföring (Fr 4015)



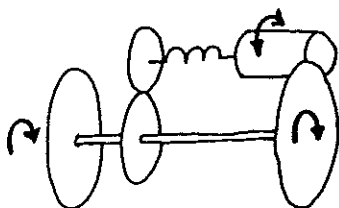
2.4.4

Svängningar i mekaniskt system

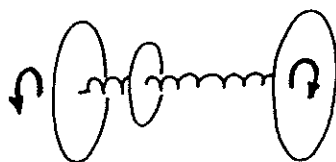
Kraftöverföringssystemet har ett antal olika svängningsformer, vilka är av betydelse bl a vid slirningstillfällena. Inom parentes anges ungefärlig resonansfrekvens för en konstruktion såsom Rc4.



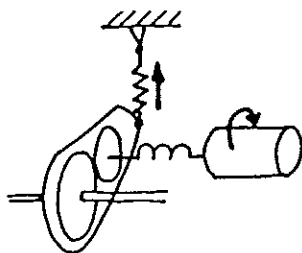
Koppling-rotor (13 Hz)



Hjulpar-koppling-rotor (19 Hz)



Hjul-axel-hjul (45 Hz)
-mer om denna svängningsform under avsnitt



Rotor-koppling-växelhuss-
reaktionsstag (12 Hz resp 86 Hz)

Förutom ovan beskrivna svängningar finns också ett antal svängningar i boggin, vilka är av intresse under slirförlopp.

Vi har boggiramens

- vertikalrörelse (5-7 Hz)
- nickning (vridning kring axel tvärs banan) (5-7 Hz)
- vaggning (" - längs ") (7-9 Hz)

Sedan finns egenfrekvenser (25-60 Hz) i det svängande systemet bana-hjul. Sådana svängningar kan t ex tänkas bli exiterade av reaktionsstagskrafter i samband med torsionssvängningar i hjul-axeln eller av det pulserande motormomentet (33 Hz). Även slipersstörningar (slipersavstånd 0,65 m) eller orundheter hos hjulen kan inverka.

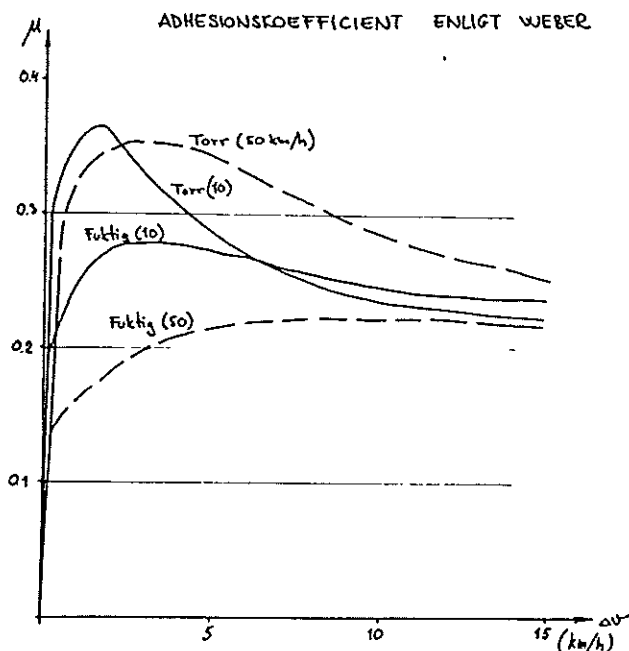
HÖR... AS... 1985... 100000... 100000... 100000...

2.5 Adhensionsutnyttjning

2.5.1 Adhesion

Adhesion är ett begrepp som ofta används i jämvägssammanhang och kan översättas med rullfriktion, och då speciellt i längsriktningen. I kontaktpunkten mellan hjul och räl finns dock generellt både längs- och tvärskrafter. Detta brukar illustreras med en s k adhesionskurva som illustrerar sambandet mellan friktionskoefficienten mellan hjul och räl som funktion av hjulperiferins överhastighet relativt spåret, vilket brukar kallas slip eller kryp. Överhastigheten brukar anges som absolutbelopp Δv eller i procent $S = \Delta v/v_0$

Några alternativa adhesionskurvor enligt en gammal referens ges av följande figur.



Som synes varierar adhesionskoefficienten kraftigt och beror av sådana faktorer som kontaktytans beskaffenhet och fordonets gångegenskaper.

2.5.2 Dragkraftsuttag, adhesionsutnyttjning

Eftersom $F = \mu \cdot N$ kan man i princip ersätta y-axeln med överförbar dragkraft. Därmed förstår man att det existerar ett optimalt slip. På gamla tiders lok var detta knappast något problem eftersom deras prestanda ej tillät adhesionsvärden i närheten av toppen. Däremot på ett modernt Rc-lok är det ett mycket påtagligt fenomen.

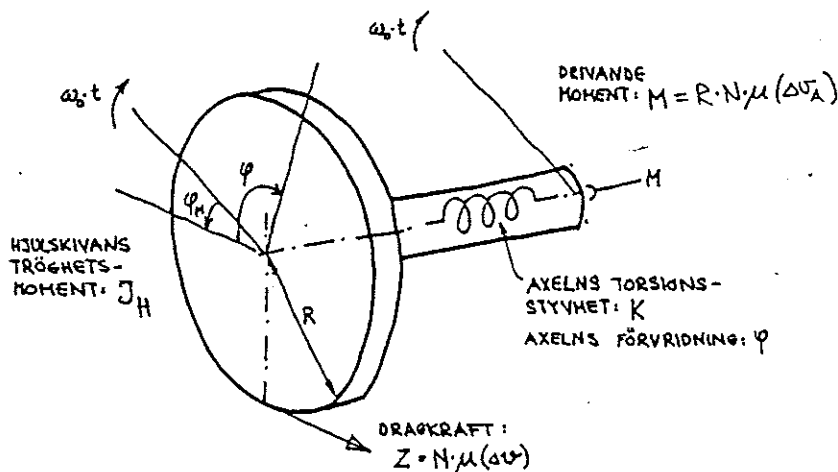
Önskvärt är att kunna utnyttja största tillgängliga adhesion, d v s att kunna nå adhesionskurvens topp, utan att för den skull kraftigt öka de mekaniska påkänningarna. Detta exempelvis genom alltför stora övervarv i samband med att hjulaxeln kan slira loss på grund av varierande adhesionsförhållanden längs spåret. M a p detta är det önskvärt att finna ett reglersystem och/eller en kretslösning, som ger snabb momentavlastning vid slirning. Bl a detta vill man t ex åstadkomma med separatmagnetiserade motorer.

Sammanfattningsvis talar man om adhesionsutnyttjning och avser då ett loks förmåga att ge bästa möjliga dragkraftsuttag i ett lite längre tidsperspektiv med varierande momentana adhesionsförhållanden.

2.5.3

Torsionssvängningar i hjulaxeln

En mycket påtaglig konsekvens av adhesionskurvens utseende är att dess negativa lutning vid högre slip innebär en negativ dämpning i det mekaniska systemet. Detta kan man inse i fall man ställer upp svängningsekvationen för hjulsatsen, d v s de två hjulskivorna med hjulaxeln som fjäder:



$$J \cdot \ddot{\varphi} + k \varphi = M - Z \cdot R$$

det gäller, att $\Delta v = \Delta v_A + R \cdot \dot{\varphi}$

ansätt för åskådlighetens skull, att

$$\mu = C \cdot \Delta v$$

$$\Rightarrow J \cdot \ddot{\varphi} + NR^2 C \dot{\varphi} + K \varphi = 0$$

Denna differentialekvation har lösningen

$$\varphi = \varphi_0 e^{-c \frac{NR^2}{2J} t} (e^{i\omega t} + e^{-i\omega t})$$

Svängningsrörelsen blir dämpad för C_0 och odämpad med växande amplitud för C_0 , vilket motsvarar en adhesionskurva med negativ lutning.

Torsionssvängningen har en väldefinierad frekvens inom intervallet 44-51 Hz beroende på hjul diameter. Svängningarna ger upphov till mycket svåra påkänningar på hjulaxlarna och kan leda till utmattningsbrott. Det är därför viktigt att styra ner momentet om dessa uppträder. Men samtidigt är svängningarna ett instrument för att reglera arbetspunkten mot optimum,

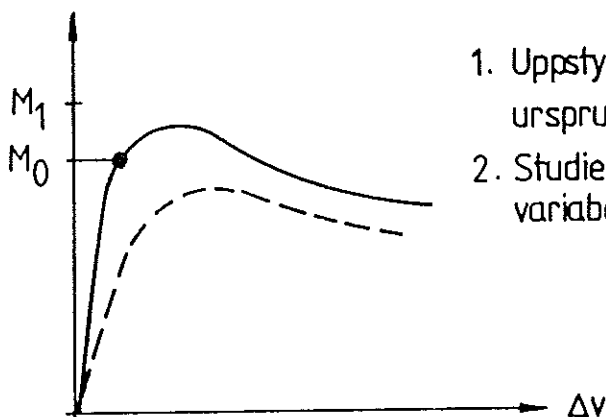
Då det gäller en allmän teori om rullfriktion är det mycket svårt att få tag på fakta. Vi har försökt bli på skolan samt hos SKF utan några egentliga resultat.

3 PROBLEMBESKRIVNING

Vi är intresserade av att studera lokets slirbeteende, dvs de dynamiska förloppen i mekanisk och elektrisk del, i samband med att adhesionförhållandena längs spåret varierar. Intressant är t ex att studera påkänningarna i det mekaniska systemet, och hur elsystemet skall byggas upp för att loket skall medge bästa möjliga utnyttning av tillgänglig friktion. I detta sammanhang är det önskvärt att kunna studera betydelsen av olika parametrar såväl som olika reglersystem.

Regleringen med fasvinkelstyrda nätkommuterade strömriktare är speciell så till vida, att regleringrepp kan göras endast en gång per halvperiod hos den växelspanning ($16 \frac{2}{3}$ eller 50 Hz) som matar loket. Efter tändning av en tyristor förblir tyristorbryggan ledande fram till växelspanningens påföljande nollgenomgång.

Eftersom modellbeskrivningen blir tämligen omfattande är det rimligt bara att studera relativt korta tidsförlopp, av storleksordning någon eller några sekunder. Det är därigenom önskvärt att indela simuleringsstudierna i två delar, först en relativt långsam uppstyrning till önskat startmoment (M_0) därpå ett flertal studier av slirförlopp från detta utgångsvärde.



1. Uppstyrning till M_0 på ursprunglig adhesionskurva
2. Studier av slirförlopp vid variabel adhesionskurva

Uppstyrmingsförloppet förutsätts ske så att alla svängningsförlopp i systemet antar stationärtillstånd, från vilket fortsatta studier sedan kan ske. I fall man i utgångsläget begär ett moment (M_1), som överskrider tillgänglig adhesion kommer en slirning naturligtvis att uppträda direkt.

I examensarbetet har det gällt att åstadkomma en modell för rotorströmkretsarna i en tvåmotormodul (boggi), samt för rotorströmregleringen. Denna modell skall knytas an till tidigare ASEA-modell av mekaniskt system, och den totala modellen skall verifieras genom uppstyrmingsförlopp. Här utarbetad modell och beräkningsmetodik skall sedan kunna användas för fortsatt utveckling.

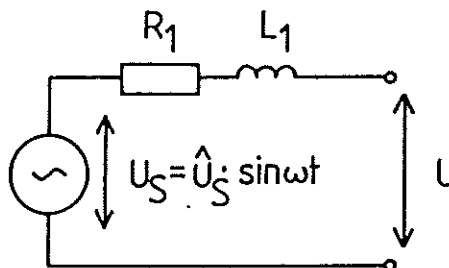
4 MODELLBESKRIVNING

Innan vi i detalj går in på de aktuella ekvationssystemen kan det vara lämpligt att beskriva de olika kretselementen, som ingår i modellen, samt grundläggande ekvationer och konstanter. Allmänna beskrivningar finns under kapitlet fordonsbeskrivning tidigare.

4.1 Transformatorn

Som framgår av kretsschemat är lokets huvudtransformator försedd med ett flertal lindningar. Lindningarna ligger på samma järnkärna, vilket innebär att de är magnetiskt kopplade.

Som en första arbetsansats har vi i vår modell ansatt att varje sekundärlindning ger en styv sinus-spänning ut. För att detta antagande inte skall bli allt för grovt, har vi lagt in en inre resistans och induktans i trafon:



Matematiskt formulerat ser det ut så här:

$$U = \hat{U}_S \sin \omega t - R_1 i_s - L_1 \frac{di_s}{dt}$$

U = spänning ut från trafon

\hat{U}_S = toppspänningen (tomgång) = 757 V

i_s = ström från (igenom) trafon

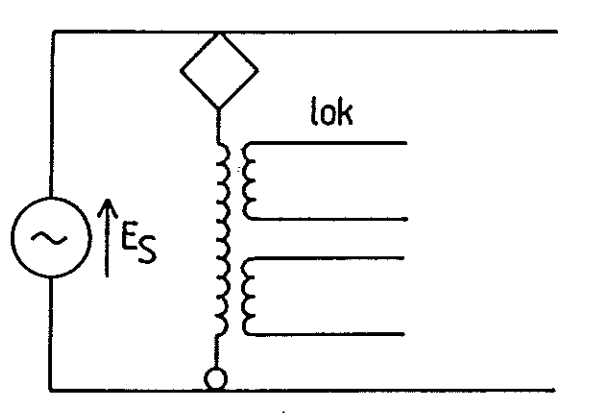
R_1 = trafons likströmsresistans reducerad till = 7,5 mΩ sekundärsidan

L_1 = trafons induktans reducerad till sekundärsidan = 0,192 mΩ

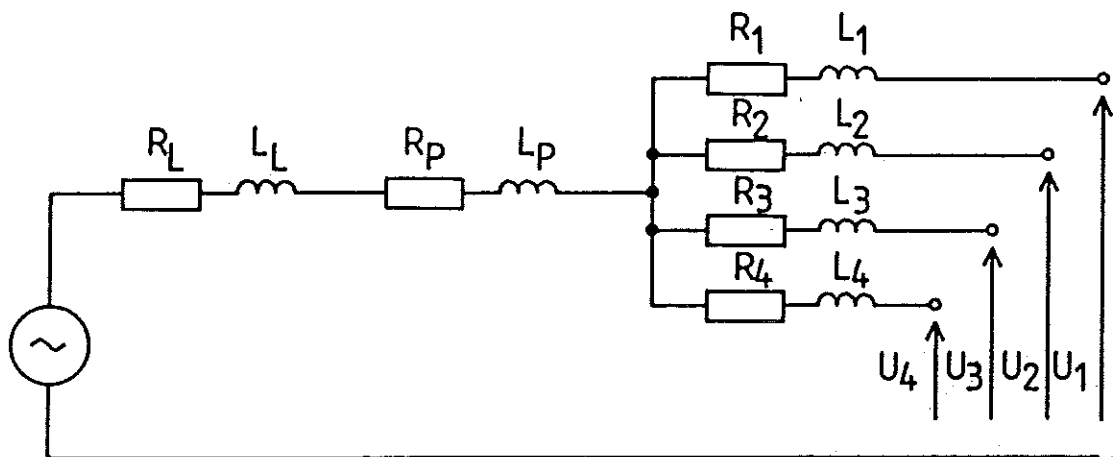
ω = nätvinkelfrekvensen = 104,7 rad/s

t = tiden

På sikt måste denna ansatts vidareutvecklas. En mer korrekt beskrivning av sekundärlindningarnas utspänningar inbegriper även kontaktledning och matningsstationer:

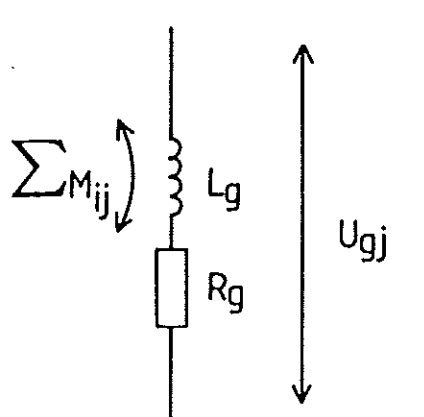


Transformatorn har fyra sekundärlindningar för rotorströmsmatning. Ett ekvivalentschema skulle få följande utseende:



4.2 Glättningsinduktorer

Av kyltekniska skäl och ur prissynpunkt har man kapslat in alla induktorer (4 st) i ett gemensamt kärl, i vilket kylolja cirkuleras. Härför uppstår en stark magnetisk koppling mellan induktorererna. I vår modell har vi tagit hänsyn till detta genom att införa ömsesidiga induktanser M_{ij} .



Matematiskt:

$$U_{Gj} = R_G i_j + \sum_{l=1}^4 (M_{lj} \frac{di_l}{dt})$$

U_{Gj} = spänning över induktor j

R_G = induktorns likströmsresistans = 13,3 m

L_G = induktorns induktans

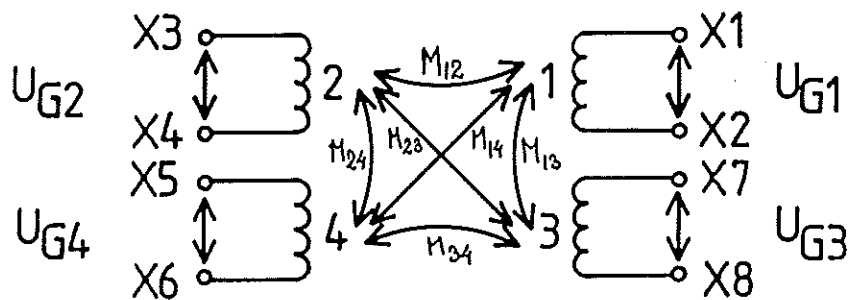
M_{lj} = ömsesidiga induktanser

Speciellt gäller $M_{ii} = L_G$ $i = 1, 2, 3, 4$ i modellen med en motor gäller $M_{ij} = 0$ då $i \neq j$.

För att representera induktorns induktanser väljer vi nu att gå över till en matris-representation:

$$\underline{\bar{U}}_G = \begin{pmatrix} U_{G1} \\ U_{G2} \\ U_{G3} \\ U_{G4} \end{pmatrix} \quad \underline{\bar{L}} = \begin{pmatrix} L_G & M_{12} & M_{13} & M_{14} \\ M_{21} & L_G & M_{23} & M_{24} \\ M_{31} & M_{32} & L_G & M_{34} \\ M_{41} & M_{42} & M_{43} & L_G \end{pmatrix} \quad \underline{\bar{I}} = \begin{pmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{pmatrix}$$

För att bestämma värdena på de olika elementen i induktansmatrisen skall vi här göra beräkningar med utgångspunkt från normala induktansmätningar på en Rc4-induktor:



Spole nr	Ström per sp A	Spänning per sp V 50 Hz	I	U	I	U	I	U
1	348	600	348	485	348	477	348	560
2	348	600	348	592	348	477	0	48
3	348	600	348	572	0	125	0	48
4	348	600	0	156	0	125	348	560

I figuren ovan skall egentligen ett R_g ingå i varje induktor. Detta förutsätter dock att de olika faslägena i mätningarna är kända. Eftersom vi nu inte känner till dessa försummar vi inverkan av R_g .

Följande samband sättes nu lätt upp:

$$\bar{U}_G = \underline{L} \cdot \frac{d\bar{I}}{dt}$$

Vi befinner oss i ett stationärt tillstånd och väljer \bar{U}_G som viktfas.

Fouriertransformation medför nu:

$$\bar{U}_G = \underline{L} \cdot j \cdot \omega \cdot \bar{I} \quad (A)$$

Tabellen ovan ger oss olika randvärden till denna ekvation:

$$\begin{array}{l}
 \text{RV} \quad 1. \quad \bar{I}_1 = \sqrt{2} \cdot \begin{pmatrix} 348 \\ 348 \\ 348 \\ 348 \end{pmatrix} \quad \bar{U}_{G1} = \sqrt{2} \cdot \begin{pmatrix} 600 \\ 485 \\ 477 \\ 560 \end{pmatrix} \\
 \\
 2. \quad \bar{I}_2 = \sqrt{2} \cdot \begin{pmatrix} 348 \\ 348 \\ 348 \\ 348 \end{pmatrix} \quad \bar{U}_{G2} = \sqrt{2} \cdot \begin{pmatrix} 600 \\ 592 \\ 477 \\ 48 \end{pmatrix} \\
 \\
 3. \quad \bar{I}_3 = \sqrt{2} \cdot \begin{pmatrix} 348 \\ 348 \\ 0 \\ 0 \end{pmatrix} \quad \bar{U}_{G3} = \sqrt{2} \cdot \begin{pmatrix} 600 \\ 572 \\ 125 \\ 48 \end{pmatrix} \\
 \\
 4. \quad \bar{I}_4 = \sqrt{2} \cdot \begin{pmatrix} 348 \\ 0 \\ 0 \\ 348 \end{pmatrix} \quad \bar{U}_{G4} = \sqrt{2} \cdot \begin{pmatrix} 600 \\ 156 \\ 125 \\ 560 \end{pmatrix}
 \end{array}$$

För varje randvärde (1-4) kan ekv A ovan sättas upp. Genom att sätta ihop alla randvärdes-vektorer erhålles följande matris-samband:

$$\begin{bmatrix} \bar{U}_{G1} \\ \bar{U}_{G2} \\ \bar{U}_{G3} \\ \bar{U}_{G4} \end{bmatrix} = \underline{L} \cdot j \cdot \omega \cdot \begin{bmatrix} \bar{I}_1 \\ \bar{I}_2 \\ \bar{I}_3 \\ \bar{I}_4 \end{bmatrix}$$

Utskrivet blir detta:

$$\begin{bmatrix} 600 & 600 & 600 & 600 \\ 485 & 592 & 572 & 156 \\ 477 & 477 & 125 & 125 \\ 560 & 48 & 48 & 560 \end{bmatrix} = \underline{L} \cdot j \cdot \omega \cdot \begin{bmatrix} 348 & 348 & 348 & 348 \\ 348 & 348 & 348 & 0 \\ 348 & 348 & 0 & 0 \\ 348 & 0 & 0 & 348 \end{bmatrix}$$

Genom att sätta absolutbelopp kring båda leden och via en invertering av I-matrisen erhålles.

$$\underline{L} = \begin{bmatrix} \bar{I}_1 \\ \bar{I}_2 \\ \bar{I}_3 \\ \bar{I}_4 \end{bmatrix}^{-1} \cdot \frac{1}{\omega} \cdot \begin{bmatrix} \bar{U}_{G1} \\ \bar{U}_{G2} \\ \bar{U}_{G3} \\ \bar{U}_{G4} \end{bmatrix}$$

$$\begin{bmatrix} L_g & M_{12} & M_{13} & M_{14} \\ M_{21} & L_g & M_{23} & M_{24} \\ M_{31} & M_{32} & L_g & M_{34} \\ M_{41} & M_{42} & M_{43} & L_g \end{bmatrix} = \frac{1}{348 \cdot \omega} \cdot \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 1 & -1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 600 & 600 & 600 & 600 \\ 485 & 592 & 572 & 156 \\ 477 & 477 & 125 & 125 \\ 560 & 48 & 48 & 560 \end{bmatrix}$$

$$\begin{bmatrix} L_g & M_{12} & M_{13} & M_{14} \\ M_{21} & L_g & M_{23} & M_{24} \\ M_{31} & M_{32} & L_g & M_{34} \\ M_{41} & M_{42} & M_{43} & L_g \end{bmatrix} = \frac{10^2}{348 \cdot 2\pi \cdot 50} \cdot \begin{bmatrix} 4,45 & 0,4 & 0,2 & 1,16 \\ 0,32 & 4,37 & 1,05 & 0,09 \\ 0,08 & 1,15 & 4,47 & 0,31 \\ 1,15 & 0,08 & 0,28 & 4,44 \end{bmatrix}$$

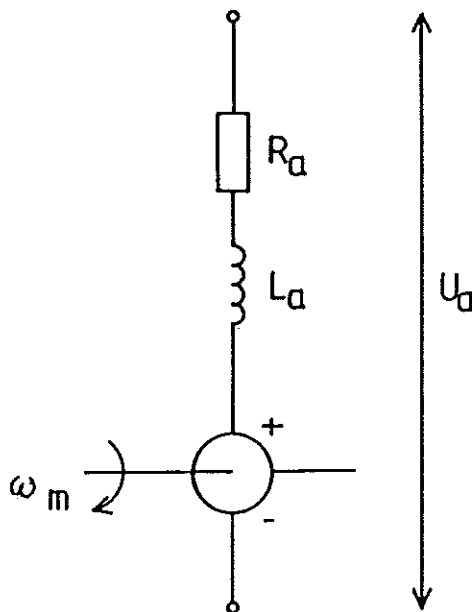
Diagonal-elementen i den högra matrisen skall vara lika, liksom även $M_{ij} = M_{ji}$ skall gälla. Detta fixar vi genom bestämma dom respektive medelvärdena.

Resultatet blir:

$$\begin{bmatrix} L_g & M_{12} & M_{13} & M_{14} \\ M_{21} & L_g & M_{23} & M_{24} \\ M_{31} & M_{32} & L_g & M_{34} \\ M_{41} & M_{42} & M_{43} & L_g \end{bmatrix} = \begin{bmatrix} 4,054 & 0,329 & 0,128 & 1,056 \\ 0,329 & 4,054 & 1,006 & 0,078 \\ 0,128 & 1,006 & 4,054 & 0,270 \\ 1,056 & 0,078 & 0,270 & 4,054 \end{bmatrix} \text{ mH}$$

4.3

Motorn kan sägas bestå av 2 delar, ankarkrets respektive fältkrets. Vi har använt oss av följande modell av rotor sidan:



$$U_a = R_a \cdot i_a + L_a \frac{di_a}{dt} - E$$

$$M_a = c \cdot \Psi \cdot i_a$$

$$\frac{d\omega_m}{dt} = \frac{1}{J_T} \cdot (M_a - M_b)$$

$$E = c \cdot \Psi \cdot \omega_m$$

U_a = spänningen över motorn

i_a = motorns ankarström

R_a = motorns inre resistans = 11,67 mΩ

L_a = motorns inre induktans = 0,54 m

ω_a = motorns vinkel varvtal (rad/s)

M_a = motorns vridande moment

E = motemk inducerat av fältsidan

Ψ = det av fältsidan bildade magnetiska flödet

c = motorkonstant = 118,2

Ur mekanisk synpunkt är vi även intresserade av:

$$J_T = \text{rotorns tröghetsmoment} = 58 \text{ kgm}^2$$

M_B = belastande moment

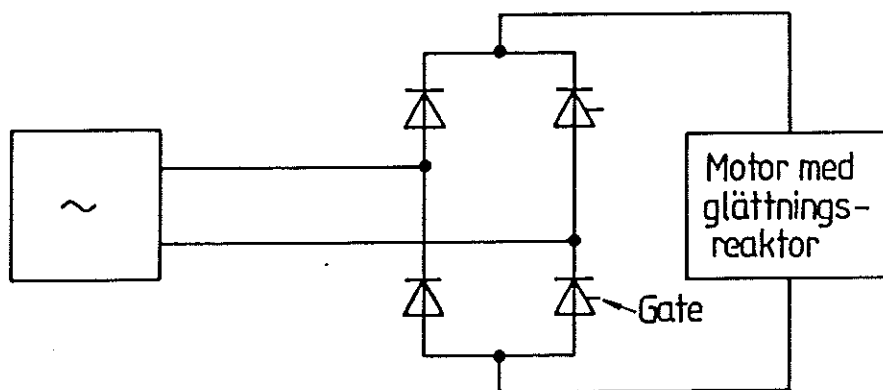
Som en första modell för motorns fält införs här en statisk betraktelse. Statiskt skall flödet styras såsom visas i figur sid 11 .

Vi tittar här på flödet under basvarv:

$$\Psi \text{ konst} = 5,5 \cdot 10^{-3} \quad (\text{T/m}^2)$$

4.4 Tyristorbryggorna

Varje tyristorbrygga styrs ut genom att en tyristor erhåller en tändpuls. För enkelhetens skull betraktar vi nu endast en tyristorbrygga:



Precis då en tändpuls trycks på tyristorns gate börjar den (dess stigtid försummas) att leda. Om spänningen är orienterad enligt figur kommer den nedre tyristor att erhålla tändpuls. Är polariteten omvänd så är det den övre tyristor som tändes. Att tyristor börjar leda innebär att strömmen "byter" väg igenom kretsen, varför ekvationerna (för spänningsfallet igenom kretsen), måste bytas.

I vår modell simulerar vi tändpulsen genom att införa en tändvinkel α som relateras till nollgenomgångarna för den sinusformade spänningen ut från trafon.

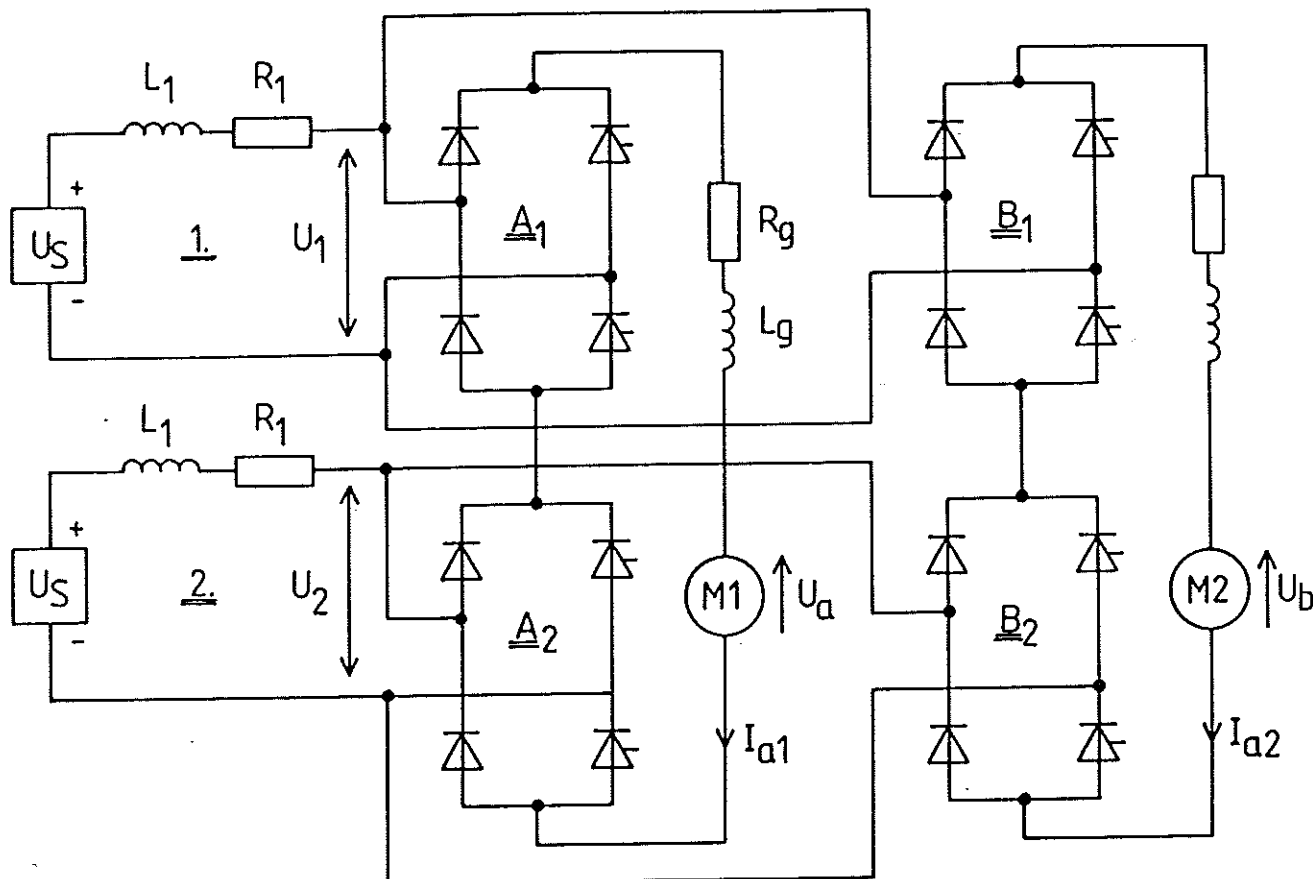
Då $\omega t < \alpha$ gäller vissa ekvationer för frihjulning. När ωt blir större än α måste vi byta ekvationer, varvid man måste ta hänsyn till de olika förloppen påkommutering, ledning, avkommutering. Dessa förlopp beskrives mera detaljerat senare.

Efter varje genomgången period (av spänningen U), kan α ändras, dvs bryggans utstyrning kan ändras. Denna utstyrning tar regler-systemet hand om.

4.5

Ekvationsunderlag för rotorströmkretsarna

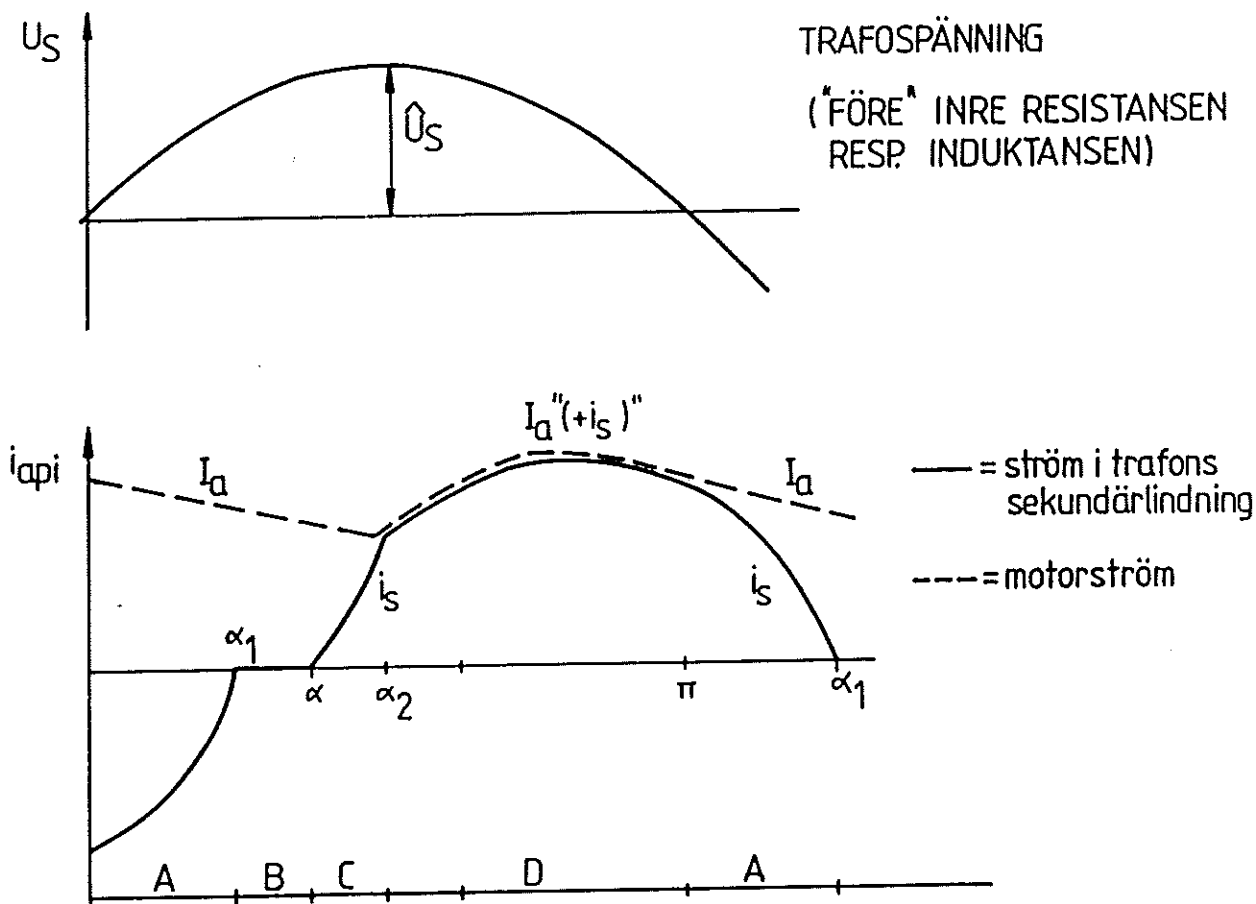
Huvudkretsen för en tvåmotormodul har följande utseende, där strömriktarna anslutits till trafons sekundärlidning i en s k rak parallellkoppling:



4.5.1

Modell med 1 motor

Vi skall nu ange ekvationerna för rotorströmmen under tyristorbryggornas olika ledintervall. För enkelhets skull studerar vi först en modell med endast en motor M1, samt förloppen med endast en brygga (A_1) ledande. Strömmens principiella tidsförlopp ges då av följande figur:



A : Avkommutering (+ frihjulning)

B : Frihjulning

C : Påkommutering

D : Ledning

Vi inför följande beteckningar i anslutning till figuren ovan:

t = tiden

φ = vinkelläge = $\omega \cdot t$

α = tändvinkel hos den översta brygga

α_2 = påkommuteringsvinkel övre brygga

α_1 = avkommuteringsvinkel (samma för båda bryggorna)

β = tändvinkel undre brygga

β_2 = påkommuteringsvinkel undre brygga

β_1 = avkommuteringsvinkel undre brygga

Vidare betecknar vi:

i_{s1} = kortslutningsström i övre brygga

i_{s2} = kortslutningsström i undre brygga

Ur programmeringsteknisk synpunkt är den elektriskt intressanta delen (motorn) helt "likgiltig" när det gäller vilket diod-tyristorpar som leder. Motorsimuleringen utföres därför endast vid ekvivalenta positiva halvperioder med def-riktningar enligt figur. För varje period kommer följande att gälla:

$$N \cdot \pi \leq \omega t \leq (N+1) \cdot \pi$$

Detta förutsatt att vi efter varje halvperiod (sett från nätet) ökar N med 1. Från början sättes då givetvis N till 0.

Ekvationerna ovan hänför sig till ekvivalenta positiva halvperioder genom att de trigonometriska funktionerna ges med absoluttecken.

4.5.1.1 Frihjulning

"Glättnings"-strömmen (I_a) frihjulnar i framriktningen genom dioderna varvid tyristorerna är stängda och strömmen drivs ej upp av nätspänningen.

$$(L_a + L_g) \frac{di_a}{dt} + (R_a + R_g) i_a + E = 0$$

$$E = c \cdot \psi \cdot \omega_m, \quad \frac{d\omega_m}{dt} = (c \cdot \psi \cdot i_a - M_B) / J_T$$

$$\text{Gäller då } N \cdot \pi \leq \varphi \leq N \cdot \pi + \alpha_2 \quad (\varphi = \omega t)$$

4.5.1.2 Ledning

Översta bryggan:

$$(L_1 + L_a + L_g) \frac{di_a}{dt} + (R_1 + R_a + R_g) i_a + E = \sqrt{2} \cdot U_s \cdot |\sin \varphi|$$

$$\text{Gäller då } N \cdot \pi + \alpha_2 \leq \varphi \leq N \cdot \pi + \beta_2$$

Båda bryggorna:

$$(2L_1 + L_a + L_g) \frac{di_a}{dt} + (2R_1 + R_a + R_g) i_a + E = 2\sqrt{2} U_s |\sin \varphi|$$

$$\text{Gäller då } N \cdot \pi + \beta_2 \leq \varphi \leq (N+1) \cdot \pi$$

4.5.1.3

Kommuteringsintervall

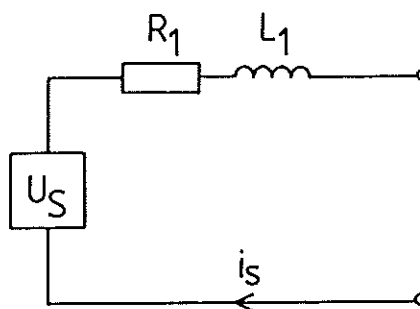
Då strömmen frihjular genom en brygga erhålles inget spänningsbidrag från nätet över bryggan. Vid påkommatering gäller samma sak fast med den skillnaden att trafons sekundärsida kan betraktas som kortsluten en tid t_d , varvid nätströmmen i_s dras upp. Till denna tid t_d relaterar vi vinklarna φ_d och α_2 enligt:

$$\varphi_d = \omega \cdot t_d = \alpha_2 - \alpha$$

där α_2 är vinkel-läget då sekundärsidan börjar ge ström till motorn. Vid avkommatering sjunker på samma sätt strömmen i trafons sekundär-sida till 0. Även detta förlopp tar en viss tid. Vi relaterar nu en vinkel α_1 till den tid (i vinkelmått) det tar för strömmen i_s att sjunka till 0.

Vid full utstyrning (minsta möjliga α) kan påkommateringen ej starta innan avkommateringen är klar ($i_s = 0$). Med andra ord kan man säga att avkommateringen sätter en minsta gräns för tändvinklarna i nästkommande period.

Av- och påkommateringsvinklarna kan vi bestämma analytiskt vid den här aktuella trafomodellen med kortsluten sekundärsida.



Detta ger samma typ av differentialekvation som tidigare, men där vi tittar på dess analytiska lösning:

$$L \cdot \frac{di}{dt} + R \cdot i = H \cdot \sin \omega t \quad i\left(\frac{\varphi_0}{\omega}\right) = i_{\varphi_0}$$

$$\text{Homogen lösning: } \frac{L}{R} \cdot \frac{di}{dt} + i = 0 \Rightarrow i = K e^{-\frac{R}{L} \cdot t}$$

$$\text{Partikulär } i_{sg}, \text{ ansats: } i = A \sin(\omega t + \theta) + D$$

insättning \Rightarrow

$$L \cdot A \cdot \omega \cos(\omega t + \theta) + R(A \sin(\omega t + \theta) + D) = H \sin \omega t$$

\Leftrightarrow

$$A \sqrt{(L\omega)^2 + R^2} \cdot \sin(\omega t + \theta + \delta) + R \cdot D = H \sin \omega t - C$$

$$\delta = -\arctan\left(\frac{\omega L}{R}\right)$$

identifiering \Rightarrow

$$D = 0$$

$$\theta = \delta$$

$$A = \frac{H}{\sqrt{(L\omega)^2 + R^2}}$$

$$\Rightarrow i = A \sin(\omega t + \delta) + K e^{-\frac{R}{L}t}$$

$$i\left(\frac{\varphi_0}{\omega}\right) = i_{\varphi_0} \Rightarrow i_{\varphi_0} = A \sin(\varphi_0 + \delta) + K e^{-\frac{R\varphi_0}{L\omega}}$$

$$\Rightarrow K = \left[i_{\varphi_0} - A \sin(\varphi_0 + \delta) \right] e^{\frac{R\varphi_0}{L\omega}}$$

Resultat:

Lösningen till eku. ges av:

$$i(t) = A \sin(\omega t + \delta) + K e^{-\frac{R}{L}t}$$

$$A = \frac{H}{\sqrt{(L\omega)^2 + R^2}}$$

$$\delta = -\arctan\left(\frac{\omega L}{R}\right)$$

$$K = \left[i_{\varphi_0} - A \sin(\varphi_0 + \delta) \right] e^{\frac{R\varphi_0}{L\omega}}$$

Vi ställer nu upp ekvationerna för på- och avkommuterande ström:

För påkommuteringen i övre brygga gäller:

$$L_1 \frac{di_{s1}}{dt} + R_1 i_{s1} = U_s \sqrt{2} \cdot \sin \varphi$$

$$\text{B.V. } i_{s1}(\alpha + N \cdot \pi) = 0$$

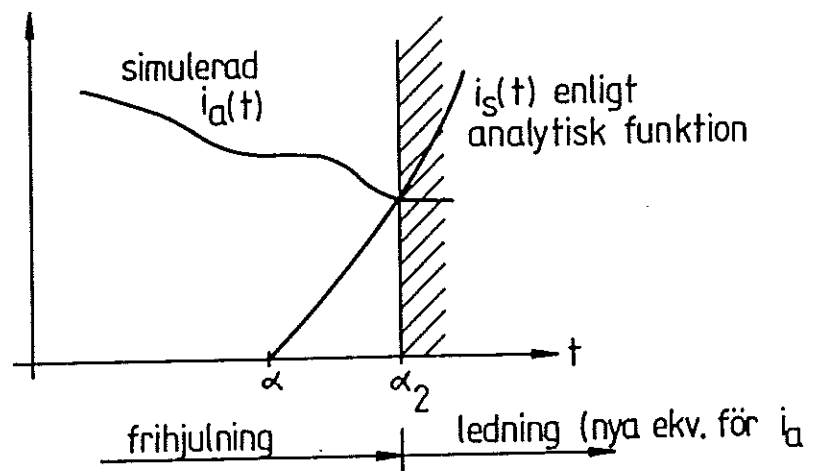
$$\Rightarrow i_{s1}(t) = A \sin(\omega t + \gamma) + K e^{-\frac{R_1}{L_1} t}$$

$$A = \frac{\sqrt{2} \cdot U_s}{\sqrt{(\omega L_1)^2 + R_1^2}}$$

$$\gamma = -\arctan\left(\frac{\omega L_1}{R_1}\right)$$

$$K = \left[-A \sin(\alpha + N \cdot \pi + \gamma) \right] e^{\frac{R_1}{L_1 \omega} (\alpha + N \cdot \pi)}$$

Påkommuteringsvinkeln α_2 ges nu av det tidsögonblick då påkommuterande ström når nivån för frihjulande ström.



Dvs vid den tid t som $i_a = i_{s1}$ erhålles α_2 enligt:

$$\alpha_2 = \omega t - N \cdot \pi$$

Noggrannheten med vilken α_2 kan bestämmas kommer att bero av vilka tidssteg, som utnyttjas i simuleringen. I det tidssteg, då $i_s(t) \geq i_a(t)$ anses bryggan ha övergått i ledande tillstånd med nya ekvationer för $i_a(t)$.

Påkommuteringen (i_{s2}) i undre brygga erhålles på samma sätt med begynnelsevärde

$$i_{s2}(\alpha + N \cdot \pi) = 0$$

och med påkommuteringsvinkeln $\beta_2 = \omega t - N \cdot \pi$ given av det tidsögonblick då $i_{s2} = i_a$.

För avkommuteringarna gäller samma differentialekvation som tidigare

$$L_1 \frac{di_s}{dt} + R i_s = \sqrt{2} U_s \sin \omega t$$

Begynnelsevärdet till denna diffekv ges av värdet på i_a vid tiden $\frac{N \cdot \pi}{\omega}$

Vi kallar nu därför
$$i_a \left(\frac{N \cdot \pi}{\omega} \right) = i_{api} = i_{beg} \left(\frac{\pi}{\omega} \right)$$

Lösningen av diffekvationen ges av:

$$i_s(t) = A \sin(\omega t + \delta) + (i_{api} + A \sin \delta) \cdot e^{-\frac{R_1}{L_1 \omega} (T - \omega t)}$$

OBS Tiden t i detta uttryck är inte den absoluta tiden utan den tid som erhålles relativt $\frac{N \cdot \pi}{\omega}$

Vinkeln α_1 ges nu av $i_s = 0$ vid tiden $t = (\alpha_1 + \pi) / \omega$

$$0 = A \sin(\alpha_1 + \pi + \delta) + (i_{api} + A \sin \delta) \cdot e^{-\frac{R_1 \alpha_1}{L_1 \omega}}$$

$$\sin(\alpha_1 + \delta) = \left[\frac{i_{api}}{A} + \sin \delta \right] e^{-\frac{R_1 \alpha_1}{L_1 \omega}}$$

$$\alpha_1 = -\delta + \arcsin \left[\left(\frac{i_{api}}{A} + \sin \delta \right) e^{-\frac{R_1 \alpha_1}{L_1 \omega}} \right]$$

Denna ekvation löses iterativt med t ex α_1 som begynnelsevärde.

Avkommuteringen kommer att bli samtidig i både övre och nedre brygga. Samma motorström i_{api} flyter i bägge trafolindningarna och skall kommuteras av under lika förhållanden. Likadana matningsförhållanden till de båda bryggorna har ju antagits.

Beräknat α_1 ger minsta möjliga tändvinkel i nästa period.

4.5.1.4

Bestämning av tändvinklar vid uppstyrningens början

Vi avser att börja simulera en uppstyrning av motorströmmen med att ange ett motorvarvtal ω_m som indata, och utgå från att strömmen genom motorn är noll. Vi antar även $\omega_m =$ konstant:

Dvs $i_a \equiv 0$

$$\omega_m = \text{"valfritt" begynnelsevarvtal, (rad/s)}$$

För att spara simuleringstid är det ointressant att studera den tid det tar att låta tändvinklarna styra upp till ett sådant läge att ström börjar flyta i motorkretsen. I_a kommer att förbli noll så länge trafons sekundärspänning i tändögonblicket är mindre än motorns motriktade emk, dvs

$$i_a = 0 \quad \text{då} \quad \hat{U}_s \sin \alpha < E$$

$$\text{eller} \quad \hat{U}_s (1 + \sin \beta) < E$$

Motorns emk ges här av valt varvtal och tillhörande flöde Ψ :

$$E = c \cdot \Psi \cdot \omega_m$$

Vi vill nu starta simuleringen i det läge, då ovanstående inte längre gäller. Beroende av E skall följande startvärden för tändvinklarna väljas:

$$\frac{E}{\sqrt{2} \cdot U_s} \leq 1 \quad \Rightarrow \quad \begin{cases} \alpha_0 = \pi - \arcsin \frac{E}{\sqrt{2} \cdot U_s} \\ \beta_0 = \pi \end{cases}$$

$$1 < \frac{E}{\sqrt{2} \cdot U_s} \leq 2 \quad \Rightarrow \quad \begin{cases} \alpha_0 = 0 \\ \beta_0 = \pi - \arcsin \frac{E}{\sqrt{2} \cdot U_s} \end{cases}$$

$$2 < \frac{E}{\sqrt{2} \cdot U_s} \quad \Rightarrow \quad \begin{cases} \alpha_0 = 0 \\ \beta_0 = 0 \end{cases}$$

I det sista fallet är båda bryggorna fullt utstyrda utan att någon ström kan drivas upp i motorkretsen. Motorns emk måste här minskas genom minskning av Ψ (fältförsvagning), för att kunna köra motorn i det sk fältförsvagningsområdet.

4.5.2

Ekvationsunderlag för modell med två motorer

I denna modell har vi lagt in 2 st motorer. Modellen är anpassad för induktiv koppling mellan alla glättningsreaktorer (4 st). Genom att variera på kopplingskonstanterna kan man simulera valfria par av parallellkopplade motorer.

Vidare kan man lätt erhålla en modell för 4 st motorer genom att parallellkoppla 2 st av ovanstående modeller. Detta kräver en trivial anpassning av variabler (endast justering av variabelnamnen).

Modellen förutsätter att bryggor som är kopplade till samma trafo, styrs ut parallellt, dvs dessa bryggor erhåller tändpulser samtidigt. Vidare måste de övre bryggorna vara fullt utstyrda innan de undre bryggorna "tändes". Detta är verkliga förutsättningar i den aktuella kretslösningen.

Behandlingen av detta problem blir helt analog med vad som tidigare skisserats för en motorkrets. Enda egentliga skillnaden är att transformatorlindningarna kommer att föra summan av de båda motorströmmarna i_{a1} och i_{a2} . Tiderna för av- och påkommuteringsarna bestäms också av tiden för att kommutera strömmarnas summa. Kravet om samtidiga tändpulser till parallellkopplade thyristorbryggor förenklar behandlingen avsevärt.

Med beteckningar enligt figuren på sid 30 kan vi teckna ekvationerna för kretsen, under frihjulning resp ledning. Vi tecknar först spänningsfallen i kretsarna.

Motorkretsar:

$$\Delta U_A = (R_A + R_G) i_{a1} + L_A \frac{di_{a1}}{dt} + \sum_{r=1}^4 \frac{di_{ar}}{dt} M_{1r}$$

$$\Delta U_B = (R_A + R_G) i_{a2} + L_A \frac{di_{a2}}{dt} + \sum_{r=1}^4 \frac{di_{ar}}{dt} M_{2r}$$

speciellt gäller $M_{11} = M_{22} = L_G$

Trafokretsar:

$$\Delta U_1 = L_1 \left(\frac{di_{a1}}{dt} + \frac{di_{a2}}{dt} \right) + R_1 (i_{a1} + i_{a2})$$

$$\Delta U_2 = L_1 \left(\frac{di_{a1}}{dt} + \frac{di_{a2}}{dt} \right) + R_1 (i_{a1} + i_{a2})$$

Sedan kan vi enkelt ställa upp önskade ekvationer.

Frihjulning ($K = 0$):

$$E_A = -\Delta U_A \quad (\text{motoremk} = \text{spänningsfall})$$

$$E_B = -\Delta U_B$$

Ledning övre brygga ($K = 1$):

$$\hat{U}_s |\sin \omega t| - \Delta U_1 - \Delta U_A = E_A$$

$$\hat{U}_s |\sin \omega t| - \Delta U_1 - \Delta U_B = E_B$$

Ledning i övre och undre brygga ($K = 2$):

$$2\hat{U}_s |\sin \omega t| - \Delta U_1 - \Delta U_2 - \Delta U_A = E_A$$

$$2\hat{U}_s |\sin \omega t| - \Delta U_1 - \Delta U_2 - \Delta U_B = E_B$$

Vi skriver nu om ekvationerna på en generell form, som också är anpassad för SIMPAC. Lite räknearbete ger då:

$$\frac{di_{a1}}{dt} = \frac{1}{(L_A + L_G + KL_1)} \cdot \left\{ K \cdot \hat{U}_s |\sin \omega t| - K \cdot L_1 \frac{di_{a2}}{dt} - K \cdot R_1 (i_{a1} + i_{a2}) - (R_A + R_G) i_{a1} - \sum_{r=2}^4 \frac{di_{ar}}{dt} M_{1r} - E_A \right\}$$

$$\frac{di_{a2}}{dt} = \frac{1}{(L_A + L_G + KL_1)} \cdot \left\{ K \cdot \hat{U}_s |\sin \omega t| - K L_1 \frac{di_{a1}}{dt} - K \cdot R_1 (i_{a1} + i_{a2}) - (R_A + R_G) i_{a2} - \sum_{r=1}^4 \frac{di_{ar}}{dt} M_{2r} - E_B \right\}$$

Frihjulning : $K = 0$

Ledning övre bryggor : $K = 1$

Ledning övre och undre bryggor : $K = 2$

Det återstår nu att ta reda på när K i ekvationerna skall anta respektive värden. Detta gör vi, liksom vid tidigare behandlade modell för 1 motor, genom att ställa upp ekvationerna för på- och avkommuterande nätströmmar. Detta skall göras m a p summan av motorströmmarna i_{a1} och i_{a2} .

Ekvationen för påkommuterande kortslutningsström i_s i respektive brygga blir som tidigare:

$$i_{s1}(t) = A \left[\sin(\omega t - N \cdot \pi + \gamma) - \sin(\alpha + \gamma) e^{\frac{R_1}{L_1 \omega} (-\omega t + N \cdot \pi + \alpha)} \right]$$

$$A = \frac{\sqrt{2} U_s}{\sqrt{(\omega L_1)^2 + R_1^2}}$$

$$\gamma = -\arctan\left(\frac{\omega L_1}{R_1}\right)$$

Påkommuteringsvinkeln α_2 ges av tidsögonblicket, då

$$i_{s1}\left(\frac{\alpha_2 + N \cdot \pi}{\omega}\right) = i_{a1}(t) + i_{a2}(t)$$

På samma sätt erhålles påkommuteringsvinkeln β i undre bryggan.

Ekvationen för avkommuterande ström kan ställas upp som tidigare:

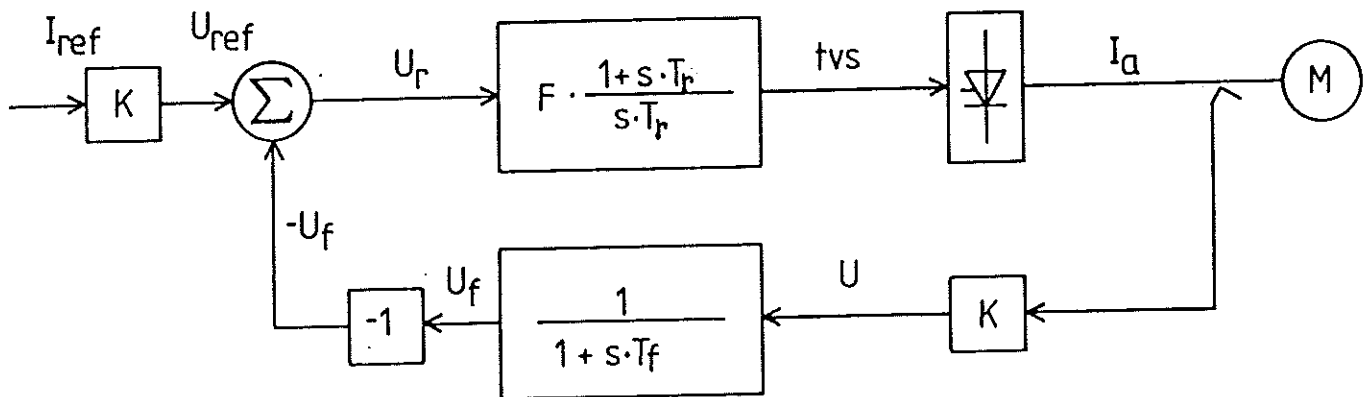
$$\alpha_1 = -\gamma + \arcsin \left[\left(\frac{i_{a11} + i_{a12}}{A} + \sin \gamma \right) e^{-\frac{R_1 \alpha_1}{L_1 \omega}} \right]$$

Beräknat α_1 ger minsta möjliga tändvinkel i nästa period.

4.6 Reglering

Som ett första steg beträffande modellering av lokets reglerloopar har vi tagit med strömregleringen.

Strömregleringen har följande utseende:



I_a = ankarström

tvs = tändvinkelsignal

U = spänning

U_f = filterad spänning

I_{ref} och U_{ref} är referensvärden (börvärden)

$$U_f = \frac{1}{1+sT_f} \cdot U$$

$$\Rightarrow \frac{dU_f}{dt} = \frac{1}{T_f} (U - U_f) \quad (1)$$

$$tvs = F \cdot \frac{1+sT_r}{sT_r} \cdot U_r$$

$$U_r = U_{ref} - U_f$$

med $x = \frac{F \cdot U_r}{sT_r}$ erhålles:

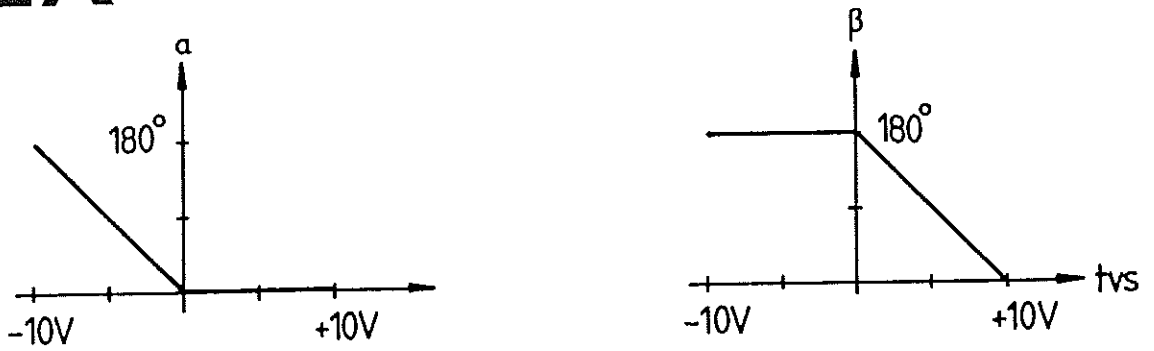
$$\frac{dx}{dt} = \frac{F}{T_r} \cdot U_r \quad \left. \vphantom{\frac{dx}{dt}} \right\} (2)$$

$$tvs = x + F \cdot U_r$$

I tvåmotormodellen filtreras båda spänningarna. Reglering sker därefter på den maximala (momentanvärden) av dessa.

tvs appliceras på följande sätt på tändvinklarna α och β :

tvs	α	β
-10 V	180°	180°
-5 V	90°	180°
0	α_1	180°
+5 V	α_1	90°
+10 V	α_1	α_1



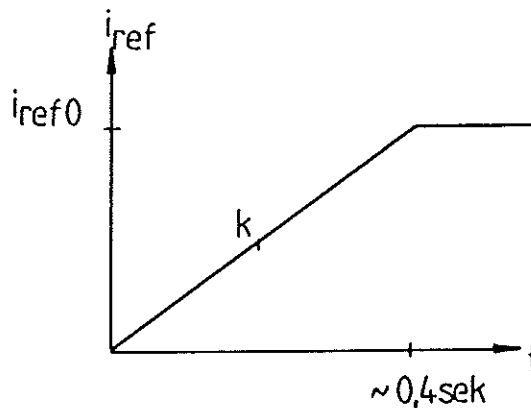
Begränsningar i tvs:

$$|I\text{-delen}| \leq 10$$

$$\Sigma(I \& P \text{ del}) \leq 10$$

Detta om linjäriseringen i styripulsdonet utesluts, vilket man utan problem kan göra.

För att undvika strömrusning vid uppstyrningens början införs en ramp för strömreferensen:



5 PROGRAMBESKRIVNINGAR

5.1 Simuleringsprogram SIMPAC/SIMPLOT

Datorprogram SIMPAC används för att lösa system av linjära eller icke linjära differentialekvationer av första ordningen, och program SIMPLOT för att presentera resultat i diagramform via en plotter. (Differenkvationen behöver ej vara ordinär).

Någon användarhandledning till SIMPAC finns tyvärr inte. Här skall ges en mycket kort beskrivning.

För att använda SIMPAC lägger man in två styrkort SELECT WB 1150/SIM1 samt SELECT WB 1150/SIM2 i sin indatalista. Mellan dessa placeras styrkortet för det program som SIMPAC skall lösa.

5.1.1

Parameterbeteckningar

Man skiljer i SIMPAC på i princip fyra olika typer av parametrar. De beroende variablerna, som i vårt fall var tidsberoende, ingår i Y-vektorn. Vidare finns dessas derivator i F-vektorn. Övriga varierande, men ej tidsberoende, variabler läggs i Z-vektorn. Konstanter slutligen, ingår i A-vektorn. När man har satt upp sitt ekvationsunderlag får man således "översätta" detta enligt ovanstående regler.

För att man i tabellutskrifter och diagram skall få variabelbeteckningar som är lättare att tolka än A(1), Y(1), F(1), Z(1) etc kan i indata anges önskade variabelnamn, via fälten ANAME, YNAME, DYNAME och ZNAME. Exempel: ANAME = namn 1, Namn 2,.....

5.1.2

Subrutin DE

Ekvationsunderlaget skrivs som en SUBROUTINE DE (T, Y, F), där T står för tiden, vilken ökar successivt under körningen av SIMPAC. I denna sk DE-rutin får ingå alla typer av FORTRAN-satser. Differentialekvationerna skall dock skrivas på en bestämd form:

$$F(1) = A(12) \times Y(1) + A(25) \times Y(2) + Z(10) \times Y(3)$$

$$F(2) = A(1) \times Y(1) + \dots$$

dvs med F-variabeln till vänster och övriga till höger om likhetstecknet. Dock kan F-variabler som tidigare beräknats även stå till höger. Data som man, i DE-rutinen, bara vill skall beräknas en gång kan skrivas under satsen IF (NFIRST.EQ.1) GOTO 1. NFIRST blir nämligen 1 så fort DE-rutinen har genomlöpts en gång.

5.1.3

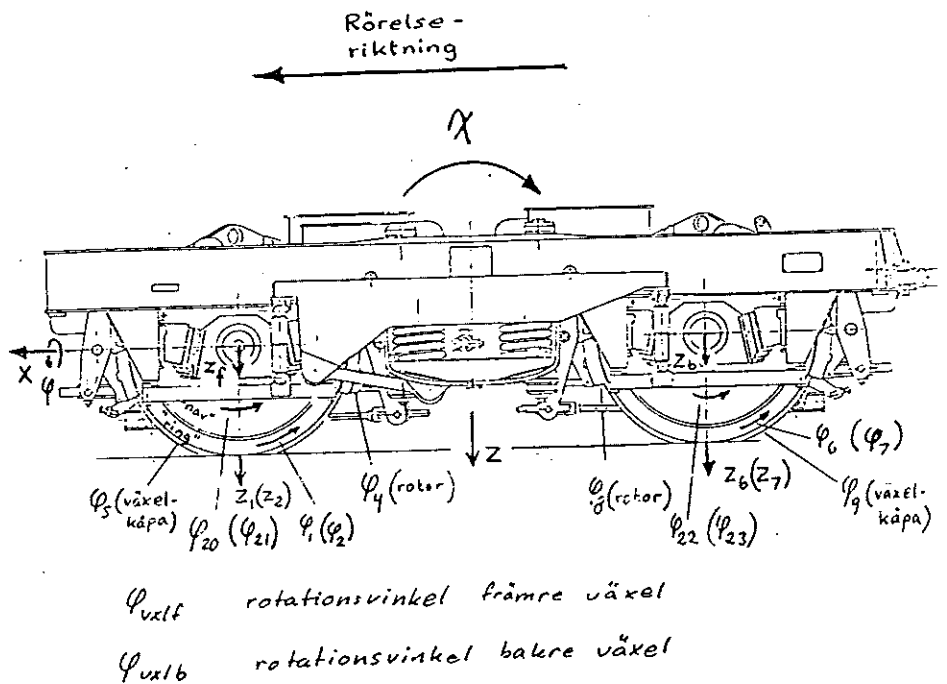
Integrationsmetod

Vid beräkningen av Y-storheterna kan man välja mellan tre olika integrationsrutiner. Vi valde den vanligaste, RKGSL, vilket är en Runge-Kutta-metod, utökad med en prediktor-korrektor beräkning.

Man kan välja relativa eller absoluta toleranser på sina tidsberoende variabler. Integrationsrutinen arbetar nämligen med variabelt tidssteg, och om inte de satta toleranserna uppfylls, halveras steget. (Kortaste tillåtna integrationssteg motsvarar 10 steglängdshalveringar).

Detta innebär att vid speciellt svåra beräkningsavsnitt minskas steget, medan det vid enklare avsnitt kan hållas större. Då en beräkning, dvs en fullständig genomlöpning av DE-rutinen, är godkänd, ställs en variabel, IGOOD, till 1. Man kan bestämma det maximala steget som skall tas med DELTA = Tidsteget, SIMPAC arbetar då med maximalt dubbla detta steg om integrationsrutin RKGSL användes. Om man vill studera hur halveringen av tidsteget arbetar kan man titta på, och plotta variabeln Z(IZHAPL). Då anger man i indata IZHAPL = 5 (t ex) och ZNAME(5) = NHAPL, som betecknar "antalet steglängdshalveringar under plottintervallet".

Variabelbeteckningar i SIMSLIP:



(parentes = höger sida)

Man kan simulera på olika nivåer genom att byta ut olika subrutiner. I den sk DE-rutinen sker anrop av de olika subrutiner som SIMSLIP är uppbyggt av. Här sker också en överföring av indatavärden från SIMPAC-kodade adresser till interkodade adresser, som lagts upp i SIMSLIP-specifika COMMON-areor. Denna överföring av indata sker endast första gången som DE anropas.

Själva beräkningarna börjar med att DRAG tillkallas. Här beräknas dragstångskräfterna. Modellen förutsätter att det är en boggi med vaggbalk. Dragkrafterna betecknas med FD.

Därefter tillkallas STAG, där de reaktionskrafter, som uppstår i reaktionsstaget, beräknas. I reaktionsstaget kan man möta den 45 Hz-svängning med vars hjälp man kan upptäcka slirning. Reaktionsstagskrafterna betecknas DV.

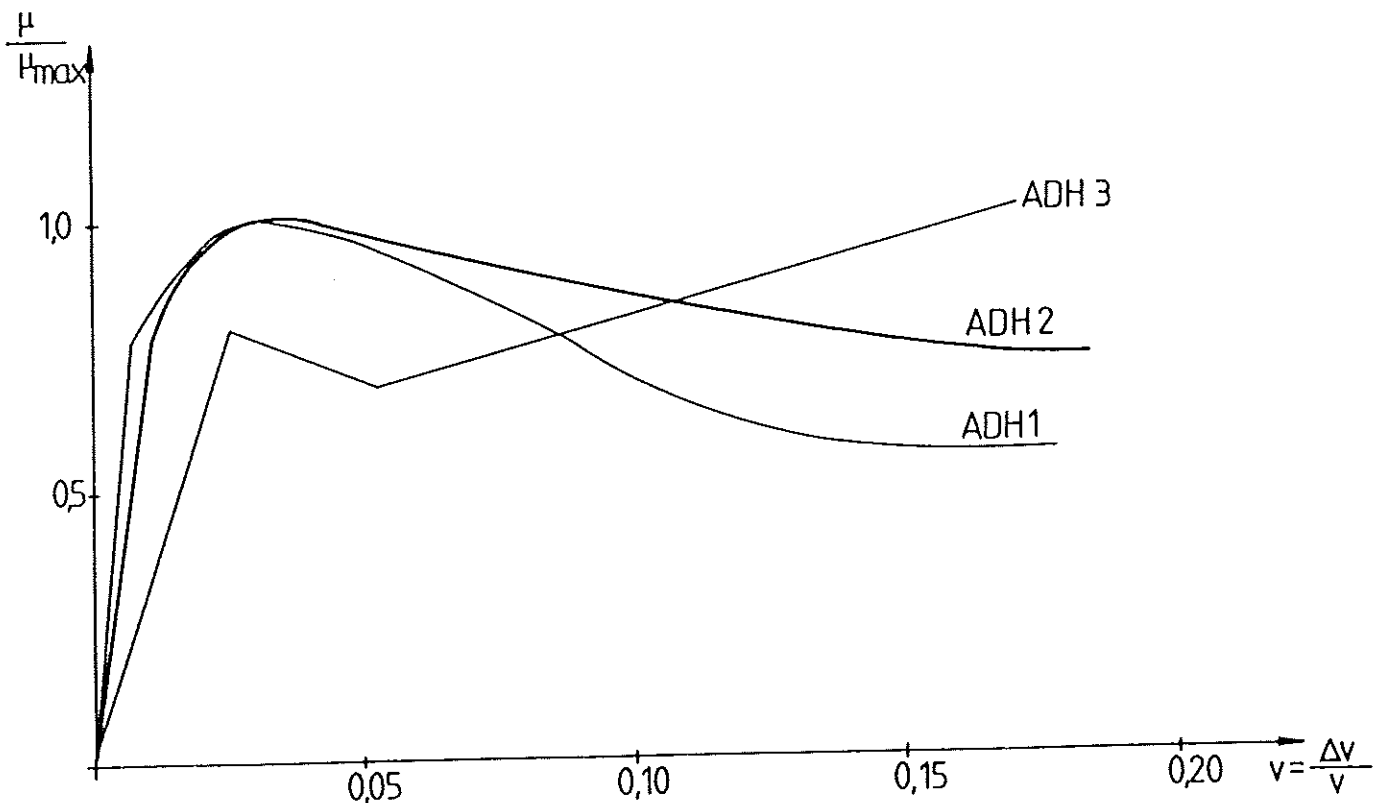
Därefter tillkallas KOP som beräknar vilka moment som uppstår i den elastiska motorkopplingen och torsionsaxeln. Här beräknas också momentstörningar i motorkopplingar. Momenten betecknas MI. Detta är alltså belastande moment. Momentstörningarna betecknas MDIS. Vill man simulera momentstörningar sätter man MDIS = A (60) \neq 0. 33 Hz och 45 Hz svängningar bör kunna lokaliseras i kopplingsmomentet.

Nästa subrutin är MOM. Här sker helt enkelt en tillväxt av det s k nollmomentet. Detta har inget med verkligheten att göra utan är en konstruktion för att man skall kunna starta från nollmoment. I princip är detta moment en del av det moment som motorn simuleras att åstadkomma. Mer om detta när vi kommer till subrutin MOTOR.

Härefter kommer subrutin MBR där man kan välja olika typer av bromsning. Detta gör man med parametern NBROMS. Sätter man NBROMS = 1 fås ingen broms, 2 ger bromsmoment på hjulskivorna och 3 ger bromsmoment på motoraxeln. Man kan här välja en mängd olika konstanter beroende på vilken broms man vill ha.

Hjulparens och spårets vertikallrörelser beräknas i VERT. Här finns fyra olika versioner. VERT-A, som vi har använt, sätter hjulaxlarnas och spårets vertikalläge till noll. I VERT-B kan man bl a simulera orunda hjul och krokiga spår i vertikalled. Kontinuerlig kontakt mellan hjul och spår förutsätts. VERT-C är som VERT-B med undantaget att kontinuerlig kontakt ej förutsätts. VERT-D slutligen har fjädrande hjul.

I KRYP beräknas hjulens kryp eller slip. KRYP anropar ADH vilket är en adhesionskruva. Man kan välja bland tre stycken. Formen på de standardiserade adhesionskruvorna 1 och 2 framgår av figuren. De specificeras med u_{max} (indata A(93) resp A(94) för axel 1 resp 2). Typ 3 består av styckvis linjära delar och finns i flera versioner för speciella studier. Vi valde ADH-2, vilket är den mest "normala" kurvan. Kryptet betecknas NY.

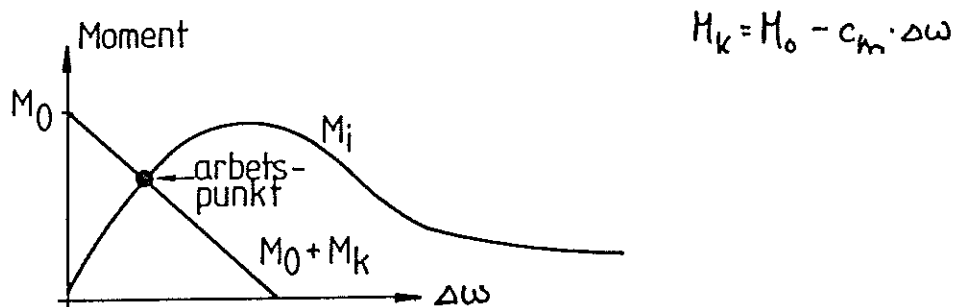


Typ av adhesionskurva väljes med SELECT-kort,

ex: SELECT CTKMO2/ADH1- ϕ

Nästa subrutin är AXEL. I denna finns rörelseekvationer för hjulparens rotation. Det finns tre olika versioner. AXEL-A simulerar en helt stel axel. AXEL-B, som vi använde, tar hänsyn till torsionsstyvheten i axeln, samt AXEL-C som är framtagen för speciella hjul med inbyggda fjäderelement av gummi.

Härefter kommer subrutin MOTOR, som i ett slirsimuleringsprogram skall ersättas med en komplett beskrivning av det elektriska systemet. Versionen i SIMSLIP bygger på en enkel modell, där motorn betraktas som matad med konstant spänning. Man kan då ansätta en motor med linjär momentkaraktäristik



Motorkarakteristik inritad i adhesionsdiagram:

där ω = nom varvtal hos hjulei $(\omega = \frac{v}{3.6} \cdot \frac{1}{r})$

Q_{nom} = nom hjullast

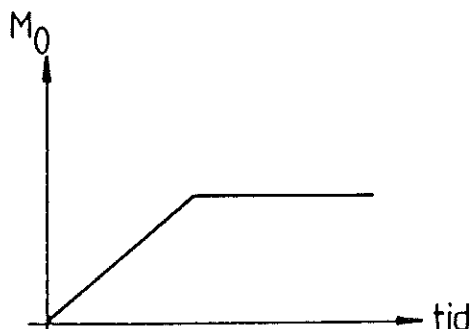
r = hjulradie

M_0 = nom motormoment vid överhastighet noll.

U = utväxling (> 1)

C_m = motordämpning

Vid simuleringar utgår man från en nominell hastighet i stationärt tillstånd. För att studera systemet styr man ut M_0 som en ramp, till ett visst värde där det sedan ligger konstant.



Detta kan symbolisera ett pådrag från föraren. Det hela kommer då att styra in sig mot arbetspunkten enligt figur.

Boggins och korgens rörelser beräknas slutligen i rutinen RAM.

För en grundligare genomgång av SIMSLIP hänvisas till pärmen SIMSLIP-D på TKM.

5.3

Program 1MOTOR

Avsnittet på följande sidor beskriver programmets uppbyggnad och själva idé.

Här visas också ett märktest av en motor för verifikation av programmet.

5.3.1

Programbeskrivning

Programmets uppbyggnad framgår i och för sig ganska klart dels av flödesschemat (sid 52) och av själva programlistan. För tydlighets skull följer här en förklaring i ord. Beteckningarna hänför sig till den tidigare modellbeskrivningen.

Programmet utmärks av att vinkeln, dvs tiden, bestämmer vilka ekvationer som skall gälla. Vi har ekv I vid frihjulning, ekv II då endast brygga 1 leder, och ekv III då båda bryggorna leder.

I programmet görs först i vanlig ordning deklARATIONER, och COMMON areor bestäms. Därefter kommer en variabel som blir 1 när programmet gått igenom första gången. Här kan man alltså sätta sådana beräkningar som bara behöver göras en gång per program. Här bestäms begynnelsevärden på tändvinklar, ledningsvinklar samt variabler för regleringen. Tändvinklarna bestäms i BEGTV. De bestäms av att man önskar börja strömmen lika med noll för ett givet utgångsvarvtal.

Nu börjar själva programmet. Till att börja med bestäms ett belastningsmoment. Det bestäms till $MB = Y(2) \times JT \times T$ vilket man naturligtvis kan ändra. Sedan görs en filtrering av strömmen. Det görs för att det kan vara lämpligt att reglera på denna. En integrering av felet mellan referensvärdet och det riktiga värdet sker också om man önskar I-reglering.

Nu kommer frågan om en ny period har inletts. Om så har skett skall α och β få nya värden. Det kan ske antingen med direktutstyrning eller med anropet CALL REGL. α_1 och β_1 ges värdena π för att man skall försäkra sig om att ekvation I skall gälla. Det är nämligen där som den "riktiga" α_2 bestäms. Sedan kollas så att inte α och β är mindre än α_1 .

Nästa steg blir att beräkna flödet, APSI, samt momentekvationen och motormomentet MV.

Nu skall det undersökas i vilket drifttillstånd bryggorna befinner sig. Det görs alltså genom att kolla var i tiden strömmen befinner sig. Vi börjar med att se om brygga 1 leder. Det görs genom att testa om $\varphi < \alpha_2 + N \cdot \pi$, där N står för antal perioder-1, om svaret är nej leder alltså inte bryggan och ekvation I gäller. I annat fall har alltså bryggan börjat påkommutteringen och vi testar om påkommutteringsströmmen har blivit större än ankarströmmen. I så fall kan vi bestämma ett nytt α_2 . Vid denna tidpunkt började ju bryggan att leda och vi vill inte gå in i ekvation I fler gånger denna period. Påkommutteringsströmmen beräknas i FUNCTION IS (FI, N, ALFA). Har bryggan inte börjat leda gäller tydligen fortfarande ekvation I.

ANSVARIG: FOR

Vi kollar nu också om möjligen brygga 2 blivit ledande. I så fall blir ju $\alpha_2 = \beta_2$. Observera att, med nuvarande styrfilosofi, β alltid är större än eller lika med α . Om nu FI skulle ha passerat $\alpha_2 + N \cdot \pi$ leder alltså brygga 1 och vi får fortsätta med att utröna om möjligen β_2 också har blivit passerad. Har den inte det gäller ekvation II, och vi nödgas göra om proceduren med β som tidigare för α . Vi testar alltså om β har passerats av FI och i så fall också om IS blivit större än ankarströmmen i vilket fall vi kan bestämma över β_2 . Skulle nu FI ha passerat över β gäller ekvation III och vi återvänder till SIMPAC.

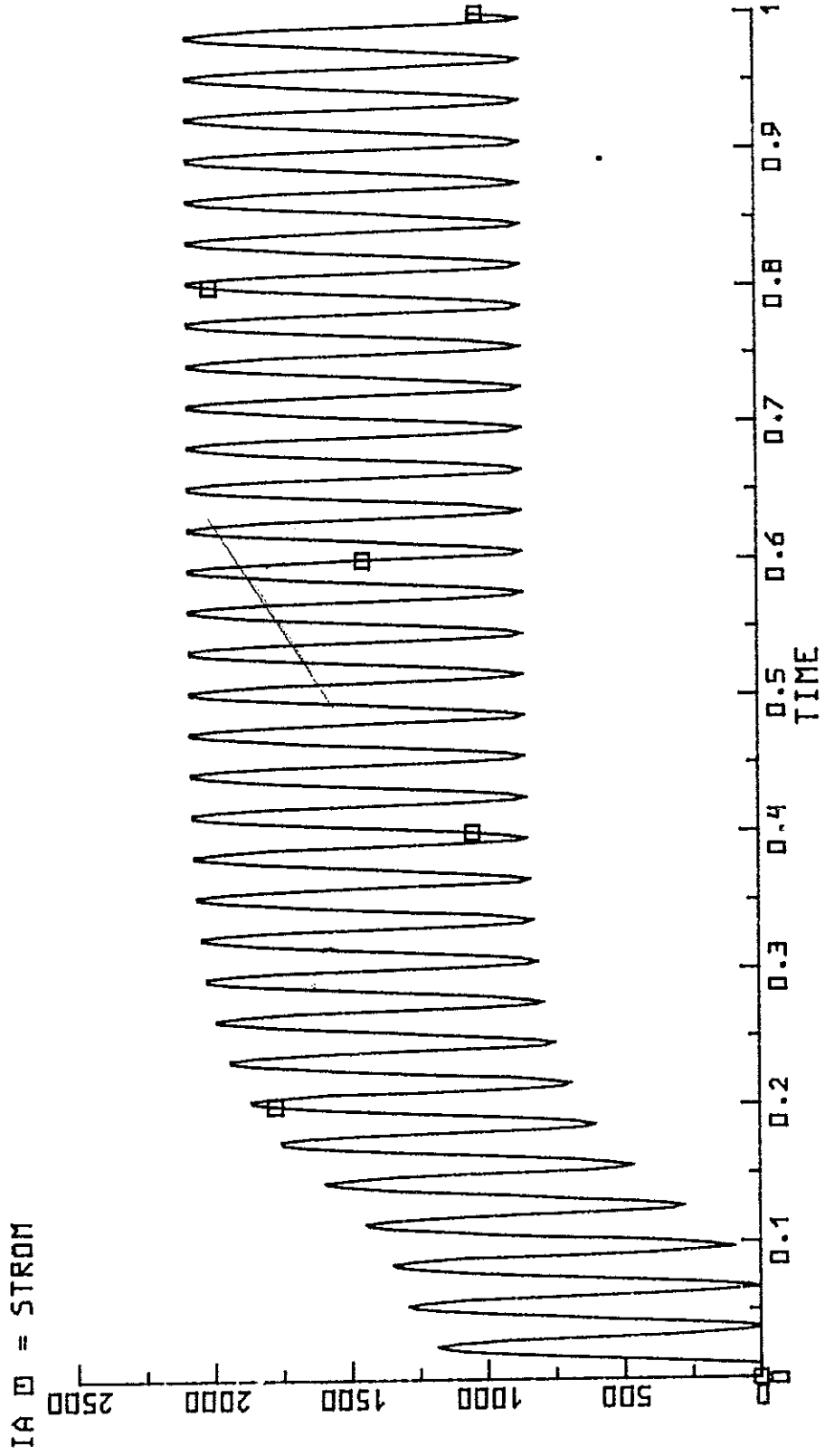
DIFF och DIFF2 är ett par kontrollutskrifter som görs för att se om skillnaden mellan IS och motorströmmen antagit rimliga värden då α_2 resp β_2 bestämts. Detta beror ju av valda integrationssteg.

Något verkligt belastningsmoment finns inte, utan man får lägga på ett eget fiktivt. Detta gör man med MB, Z(10), och måste alltså göras inne i programmet. Ett alternativ är att "nollställa" den diff ekv som bestämmer, F(2), dvs derivatorn för rotorns hastighet (Rad/s).

5.3.2 Märkttest

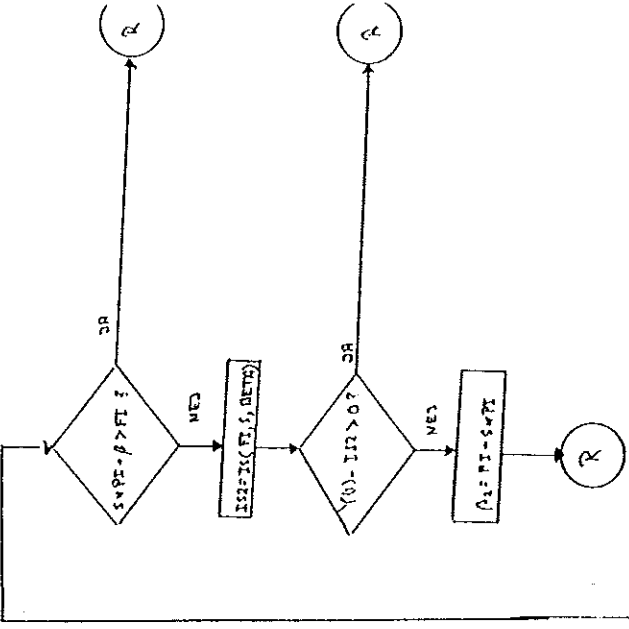
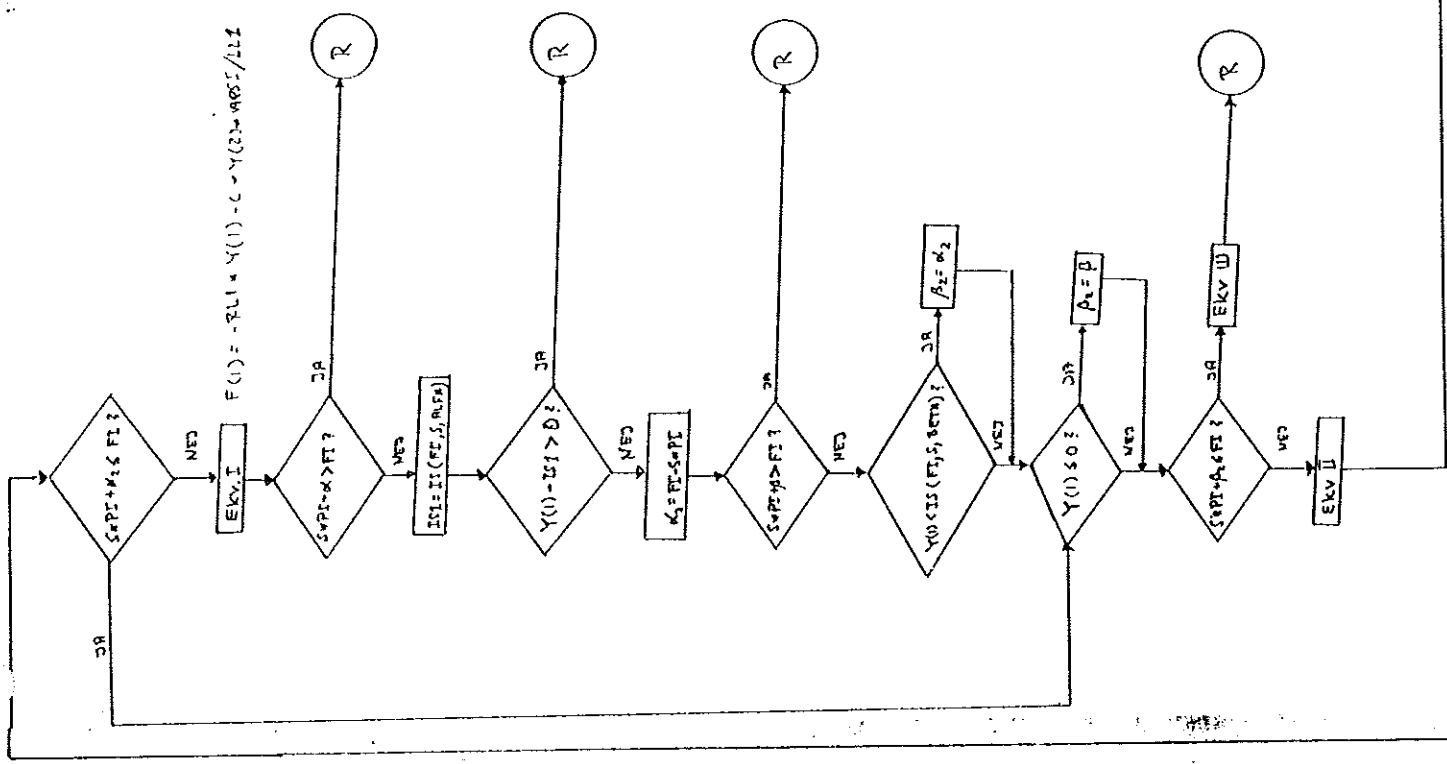
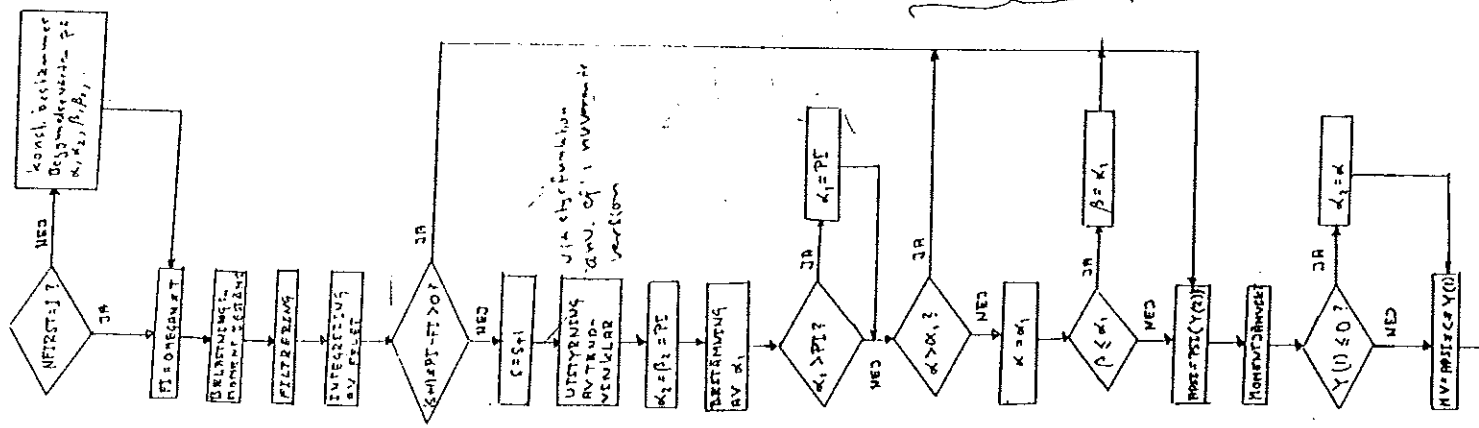
För programmet 1MOTOR har ett par simuleringar gjorts. Här bifogas resultatet av en verifierande simulering, 525K9. Denna (diagram 2) visar resulterande motorström vid fullt utstyrda tyristorbryggor och vid motorns märkvarvtal (116,2 rad/s) och märkflöde. Loket skall vara så dimensionerat, att denna punkt ger märkström i motorerna. Vi har här erhållit ca 1400 A, medan märkström är 1250 A. Ansatserna i vår trafomodell torde ge denna avvikelse eftersom kommuteringsintervallen blir för korta.

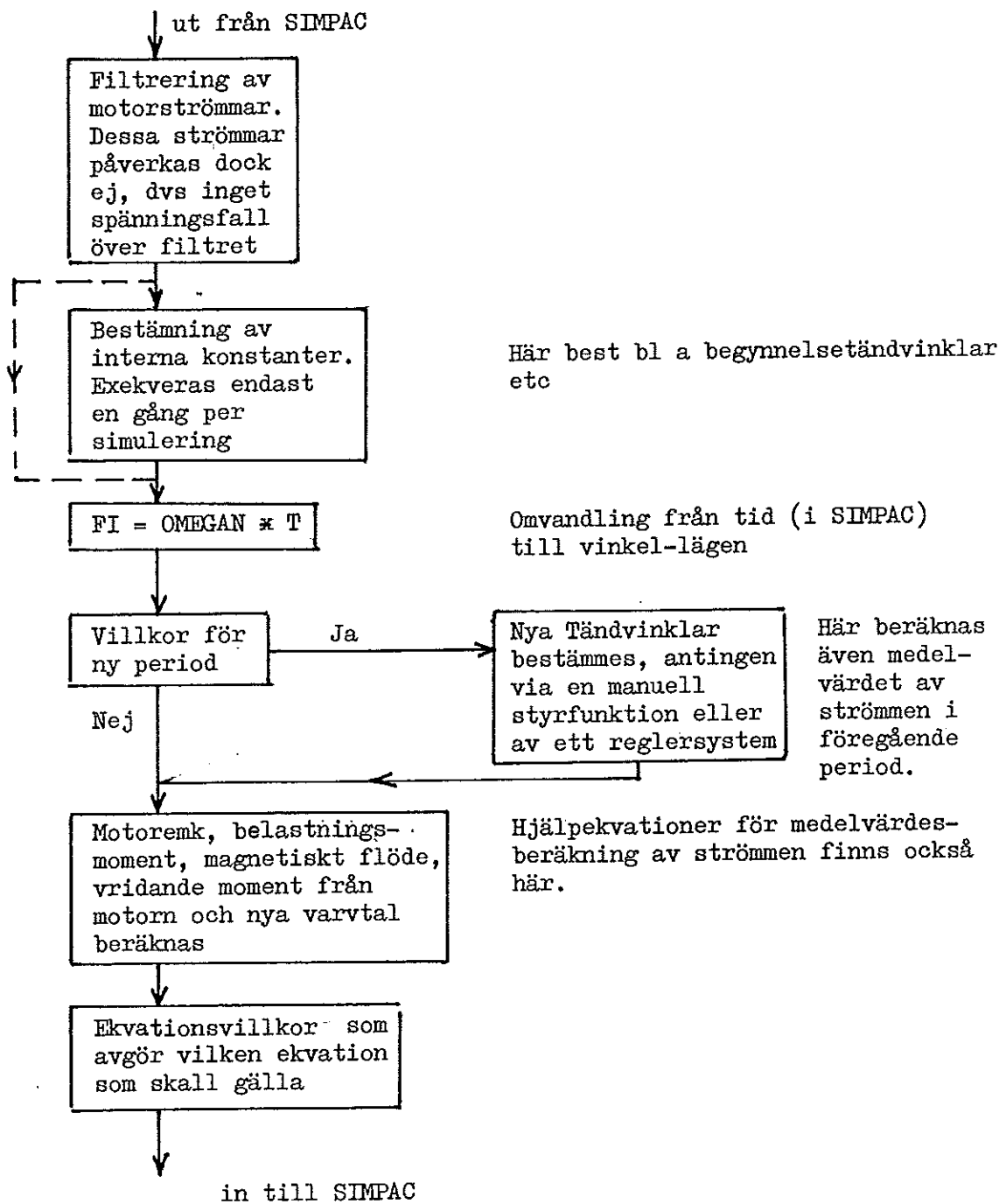
MOTOR
F, NR2
N.H, T.S, TKU



Santiajo
betekningur
for hlaua 65
framt.

Deila gætt
; JUST þá
bestættur au
nig - hændumlin





5.5.1

Strömekvationer

I stället för att efter varje villkor skriva ut ekvationerna har vi överfört dessa på en generell form vilket medför att vi kan skriva en ekvations-subrutin. Vi har härvid använt oss bl a av en matris MEKV som en formell parameter. Detta är tyvärr "lite" omständigt, eftersom det hade räckt med en vektor. Ekvationerna i subrutinen ser ut så här:

$$\frac{dI_{a1}}{dt} = [MEKV(1,I) \cdot \hat{U}_s \cdot |\sin(FZ)| - I_{a1}(R_a + R_g + MEKV(3,I)) - I_{a2}M_{21} - E_{uka} - I_{a3}M_{13} - I_{a4}M_{14}] / (L_a + L_g + MEKV(5,I))$$

$$\frac{dI_{a2}}{dt} = [MEKV(2,I) \cdot \hat{U}_s \cdot |\sin(FZ)| - I_{a2}(R_a + R_g + MEKV(4,I)) - I_{a1}M_{21} - E_{ukb} - I_{a3}M_{32} - I_{a4}M_{24}] / (L_a + L_g + MEKV(6,I))$$

MEKV =	<table style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">2</td></tr> <tr><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">2</td></tr> <tr><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">R1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">2R1</td></tr> <tr><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">R1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">2R1</td></tr> <tr><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">L1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">2L1</td></tr> <tr><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">L1</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">0</td><td style="padding: 5px;">2L1</td></tr> </table>	0	0	0	1	0	0	0	0	2	0	0	0	1	0	0	0	0	2	0	0	0	R1	0	0	0	0	2R1	0	0	0	R1	0	0	0	0	2R1	0	0	0	L1	0	0	0	0	2L1	0	0	0	L1	0	0	0	0	2L1	1. 2. 3. 4. 5. 6.
0	0	0	1	0	0	0	0	2																																																
0	0	0	1	0	0	0	0	2																																																
0	0	0	R1	0	0	0	0	2R1																																																
0	0	0	R1	0	0	0	0	2R1																																																
0	0	0	L1	0	0	0	0	2L1																																																
0	0	0	L1	0	0	0	0	2L1																																																
I =	1	4	0	0	0	0	0	9																																																
Ekv =	MI	MIII					MVI																																																	

(Matrisen MEKVs dimension har av en olycklig slump blivit allt för stor).

Det framgår nu klar varför vi infört parametern I i avsnittet "Definitionsområden". Variabeln STEG finns enbart till för att man skall kunna spåra programmets väg genom ekvationsvillkoren.

I programmet finns det möjlighet att välja på vilket sätt man vill styra ut bryggorna. Om variabeln JMAN (inparameter) sättes till 0, kommer de nya tändvinklarna (efter varje period) att ges av 4 st funktions. Dessa väljes valfritt. Dock måste styrrestriktionerna enligt tidigare gälla. Följande funktions och tändvinklar hör ihop:

α_A AFALFA	β_A ABALFA
α_B BFALFA	β_B BBALFA

Då JMAN = 1 styrs bryggorna ut av reglersystemet.

Läsaren bör nu vara mogen för en mer detaljerad beskrivning av programmet. Detta beskrivning ges bäst av det kompletta flödes-schemat, vilket följer på nästkommande sida.

5.6

Modifieringar av 2MOTOR-modellen för SIMSLIP

Inläggningen av 2 motorer i SIMSLIP har skett på exakt samma sätt som för 1 motor. Vi anser därför att det är onödigt att än en gång upprepa detta förfarande.

6

SIMULERINGAR

Här skall två representativa simuleringar presenteras. Båda omfattar modellen för en boggi med två motorer.

6.1

Simulering 662K9

Strömreferensen styr de båda motorströmmarna upp till 2000 A på 0,4 sek. Med adhesionskruvans maximum satt till 0,5 kommer utvecklade dragkraft inte att överskrida adhesionsmaximum och vi får ingen slirning. Vi har:

Diagram 1 Ström i motor 1. I början av uppstyrningen syns hur (sid 60) frihjulande ström hinner gå ned till noll innan tändpuls ges. Strömpulsationerna får 25 % rippel vid 2000 A.

Diagram 3 Effektiv utstyrning brygga 1. Övre bryggan styrs ut (sid 61) direkt. Avkommateringarna begränsar effektiv utstyrning.

Diagram 4 Utstyrning brygga 2. 60 km/h ligger under märkhastig- (sid 62) het, dvs önskad ankarström uppnås innan brygga 2 blivit fullt utstyrd.

Diagram 6 Momentet i kraftöverföringens gummikoppling. Motorns (sid 63) momentpulsationer syns.

Diagram 5 Moment i kraftöverföringens gummikoppling som ett (sid 64) exempel plottad som funktion av kryptet mellan hjul och räl.

Diagram 8 Reaktionsstagskraft mellan växelkåpa och boggiram (sid 65) pga kopplingsmomentet MI2 (diagram 6).

Diagram 9 Boggiramens vridning kring en axel i dess längsrikt- (sid 66) ning. Vridningen uppstår pga de båda reaktionsstagskrafterna.

Diagram 10 Motsvarande diagram 9, men med boggiramens vridning (sid 67) kring en axel i dess tvärsriktning. Nickningsvinkeln stabiliseras efter ca 1 sekund.

6.2

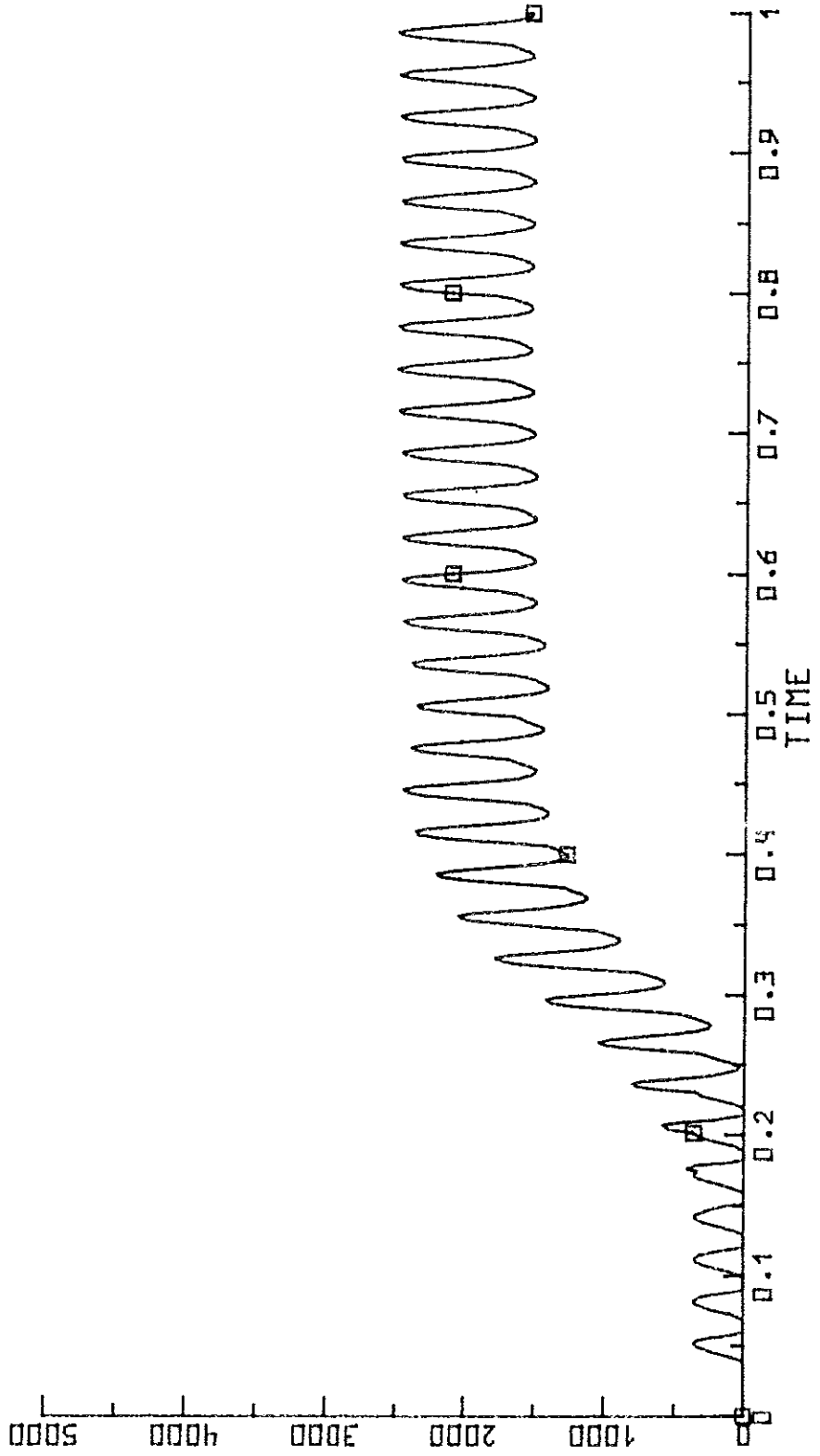
Simulering 865S5

Denna simulering avser att på prov studera händelserna vid ett slirförlopp på båda hjulaxlarna.

Simuleringen är gjord på så sätt, att motorströmmen först styrs upp som vid föregående simulering, och mot en adhesionskruva med $\max = 50\%$. Vid $t = 0,5$ har adhesionskruvan, momentant ändrats till $\max =$, för att sedan vid $t = 0,85$ ändras tillbaka till ursprunglig adhesionskruva. Under hela slirförloppet har bryggornas tändvinklar hållits konstanta, dvs slirförloppet motsvarar förhållandena vid matning av motorerna med konstant spänning.

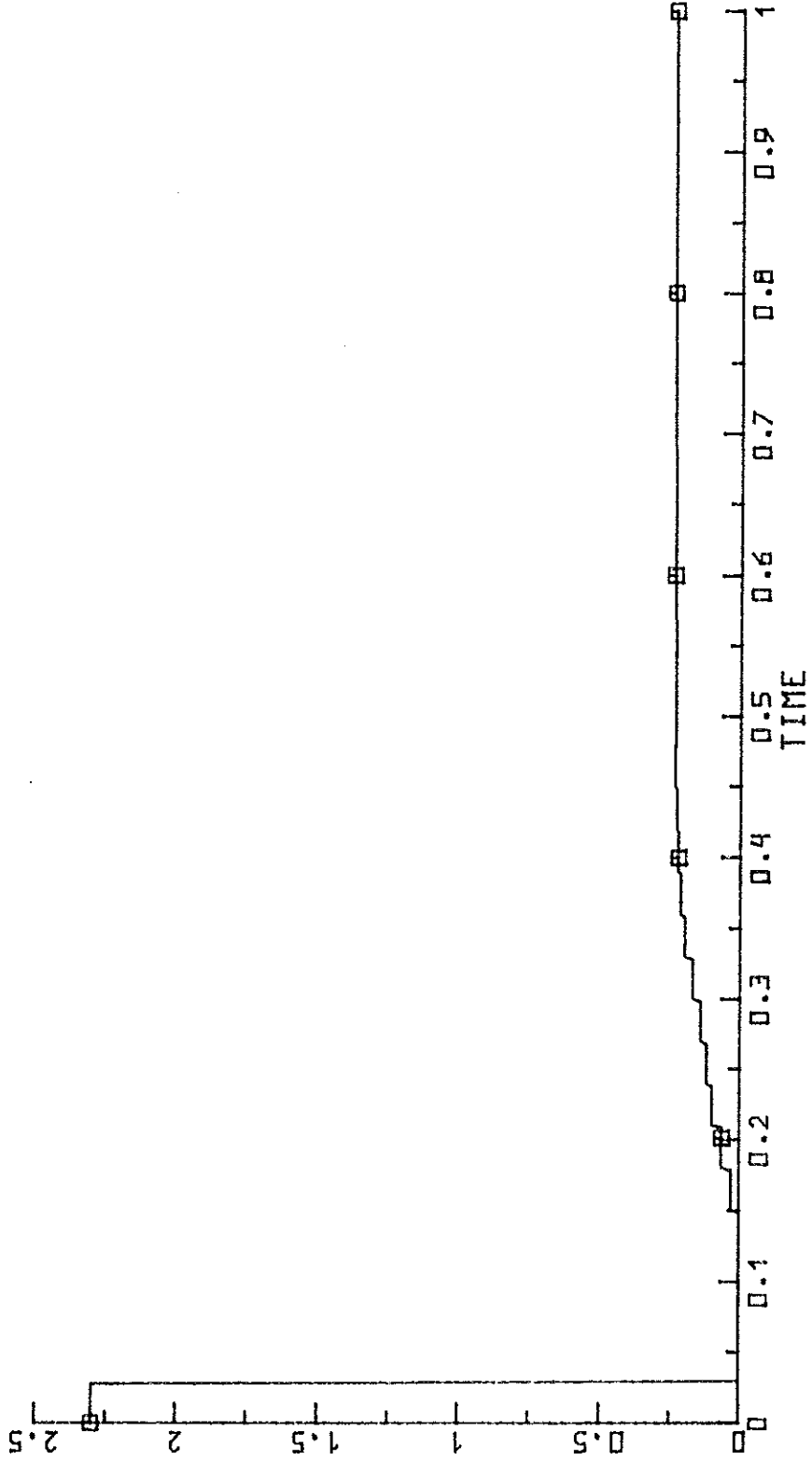
SIMSLIP,
RC-8066I, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A.

IA1 □ = MOTORSTROM 1

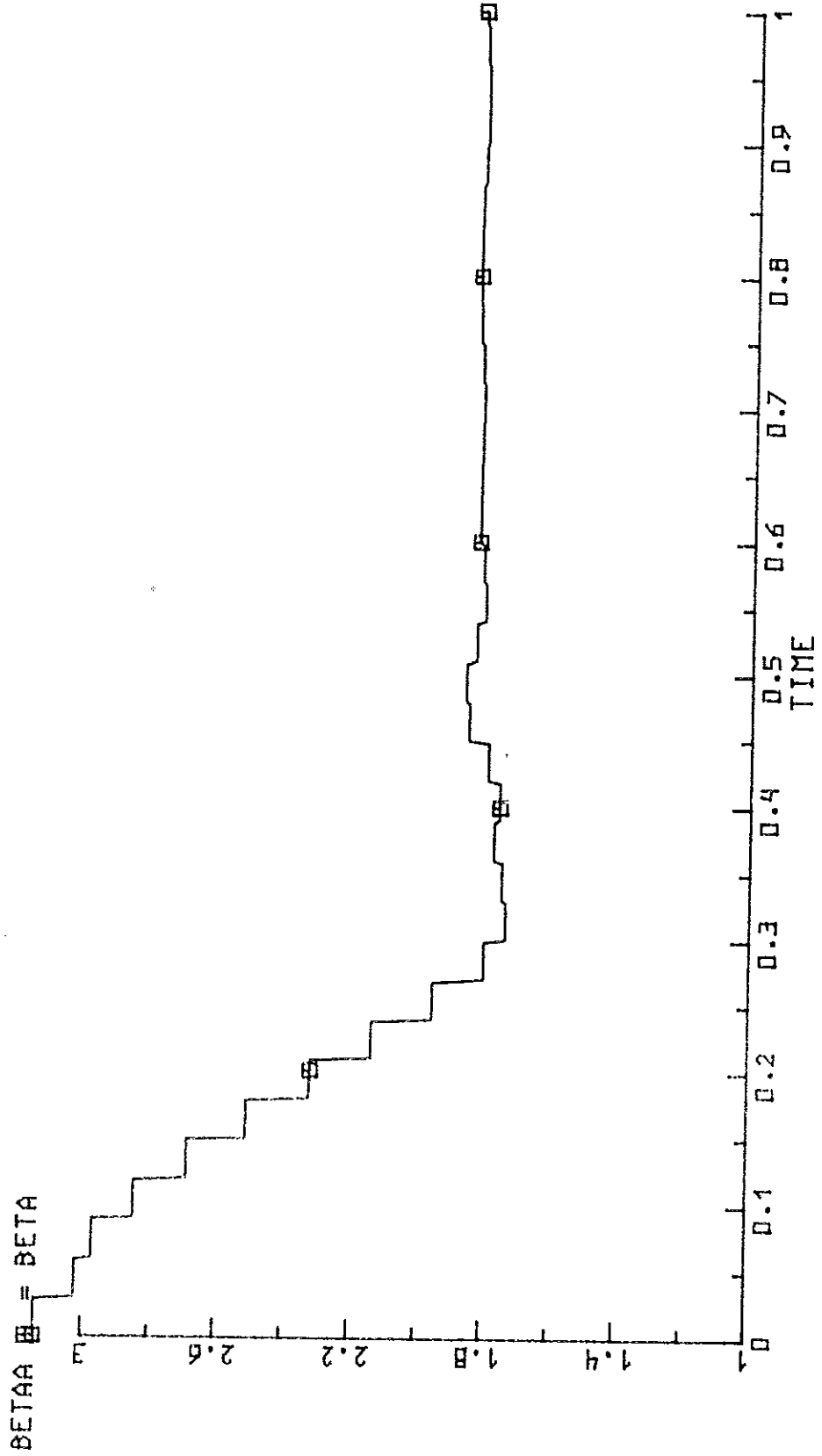


SIMSLIP,
RC-80561, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

ALFAA α = ALFA

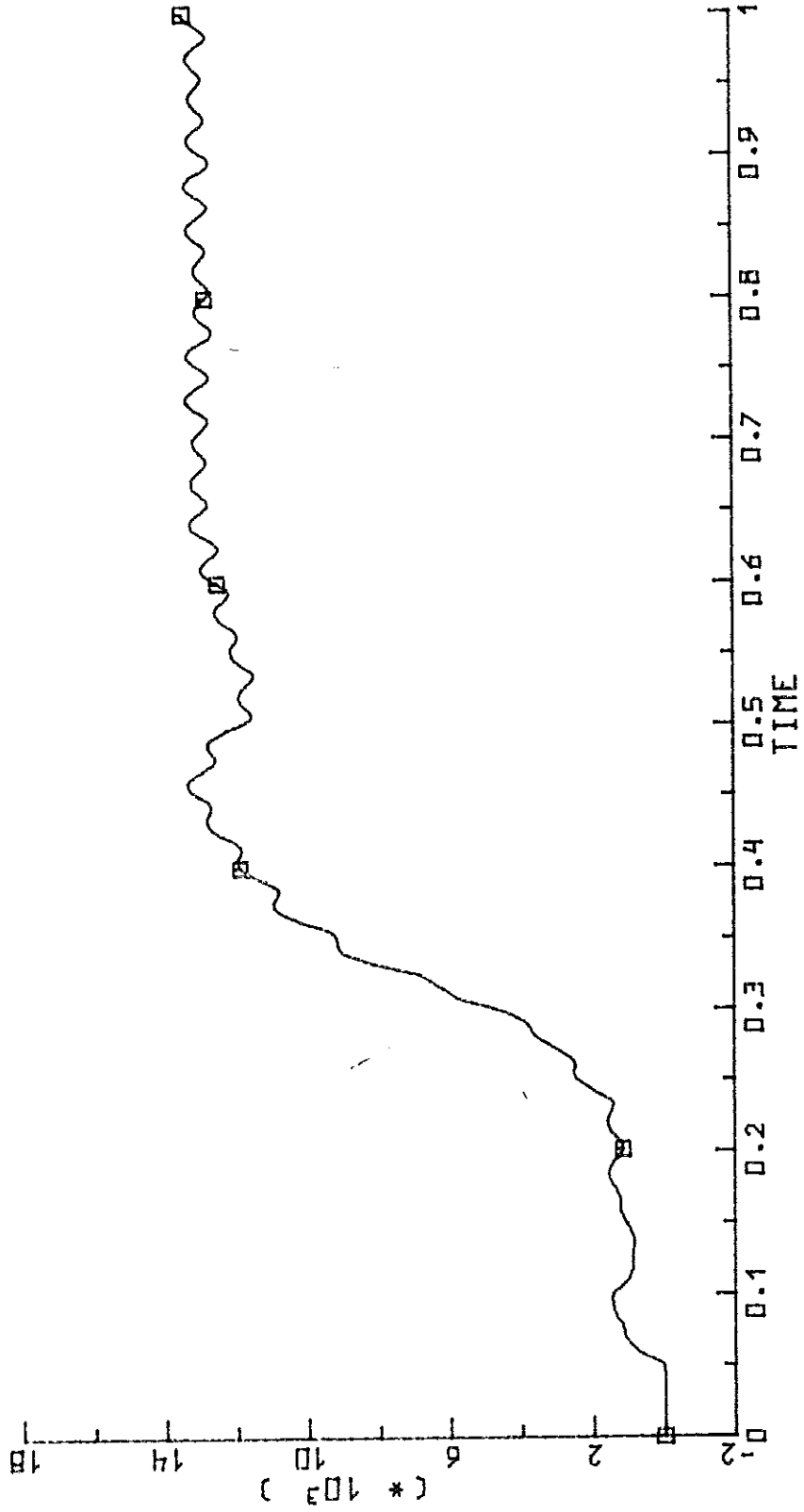


SIMSLIP,
RC-80661, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A.



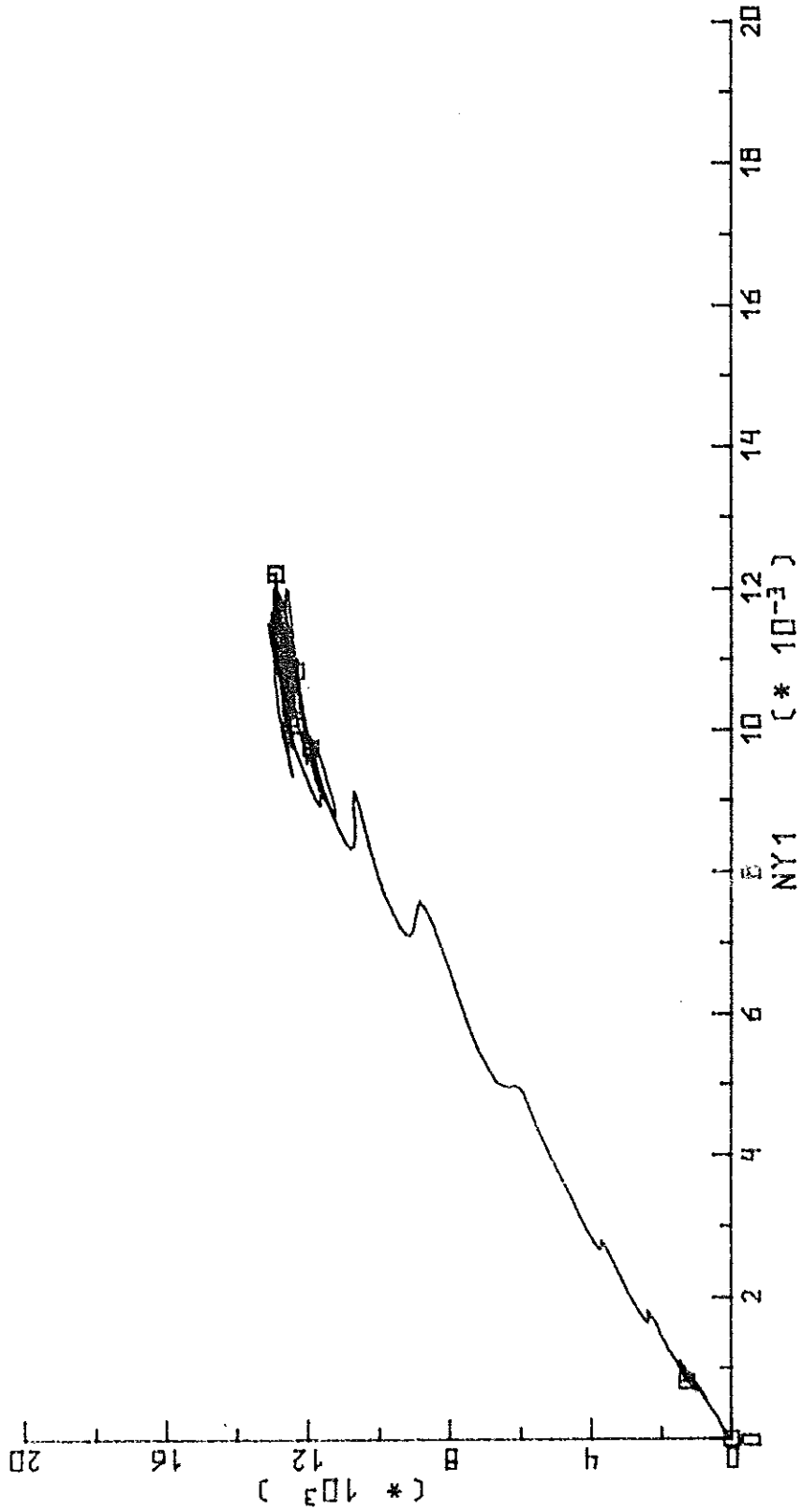
SIMSLIP,
RC-BÖGGEI, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

MI2 □ = KOPPLINGSMOMENT 2



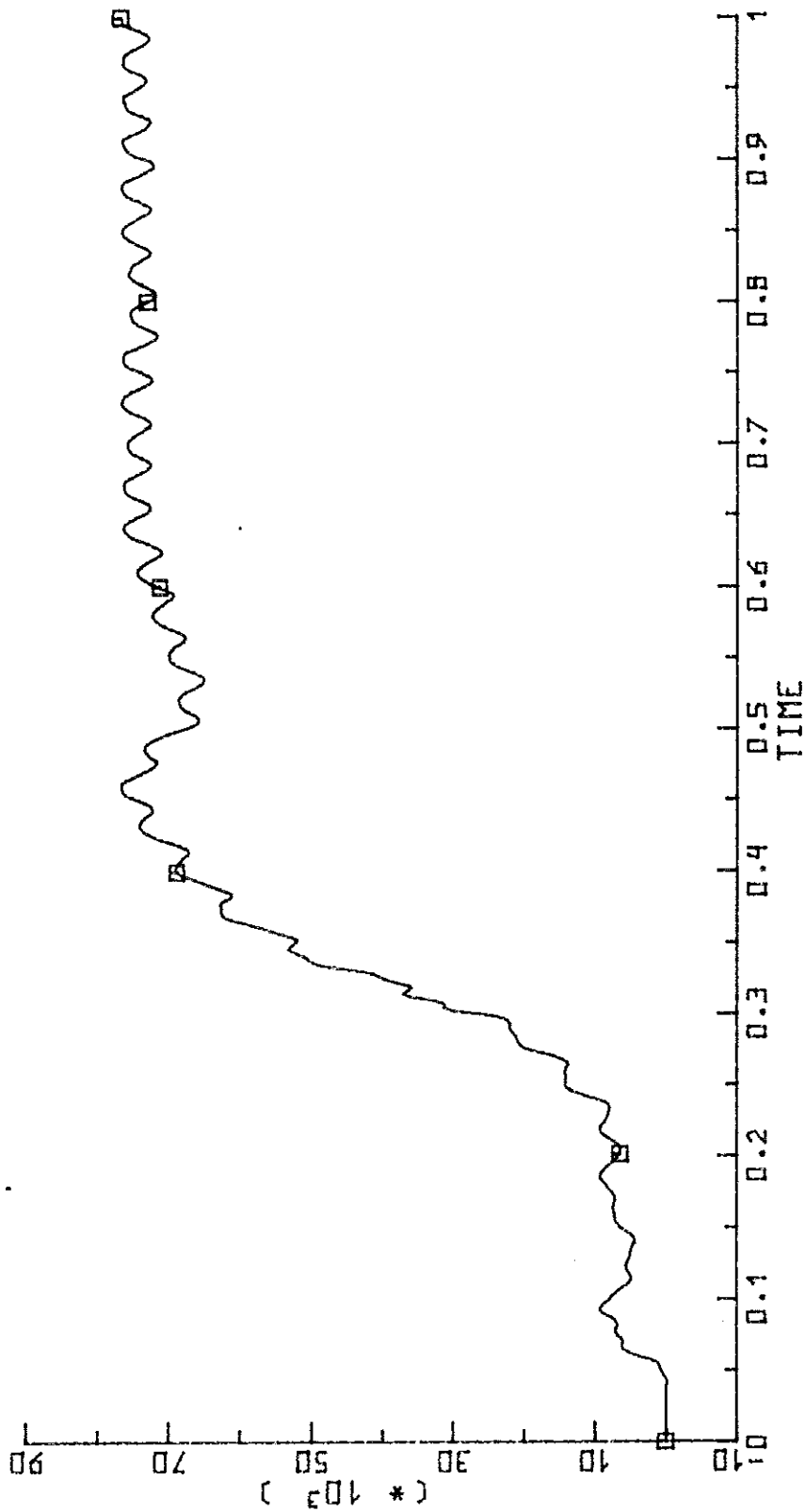
SIMSLIP,
RC-B066I, VERSION MED TVA MOTORER,
V=60KM/H OCH JREFA=2000A.

MI1 □ = KOPPLINGSMOMENT 1



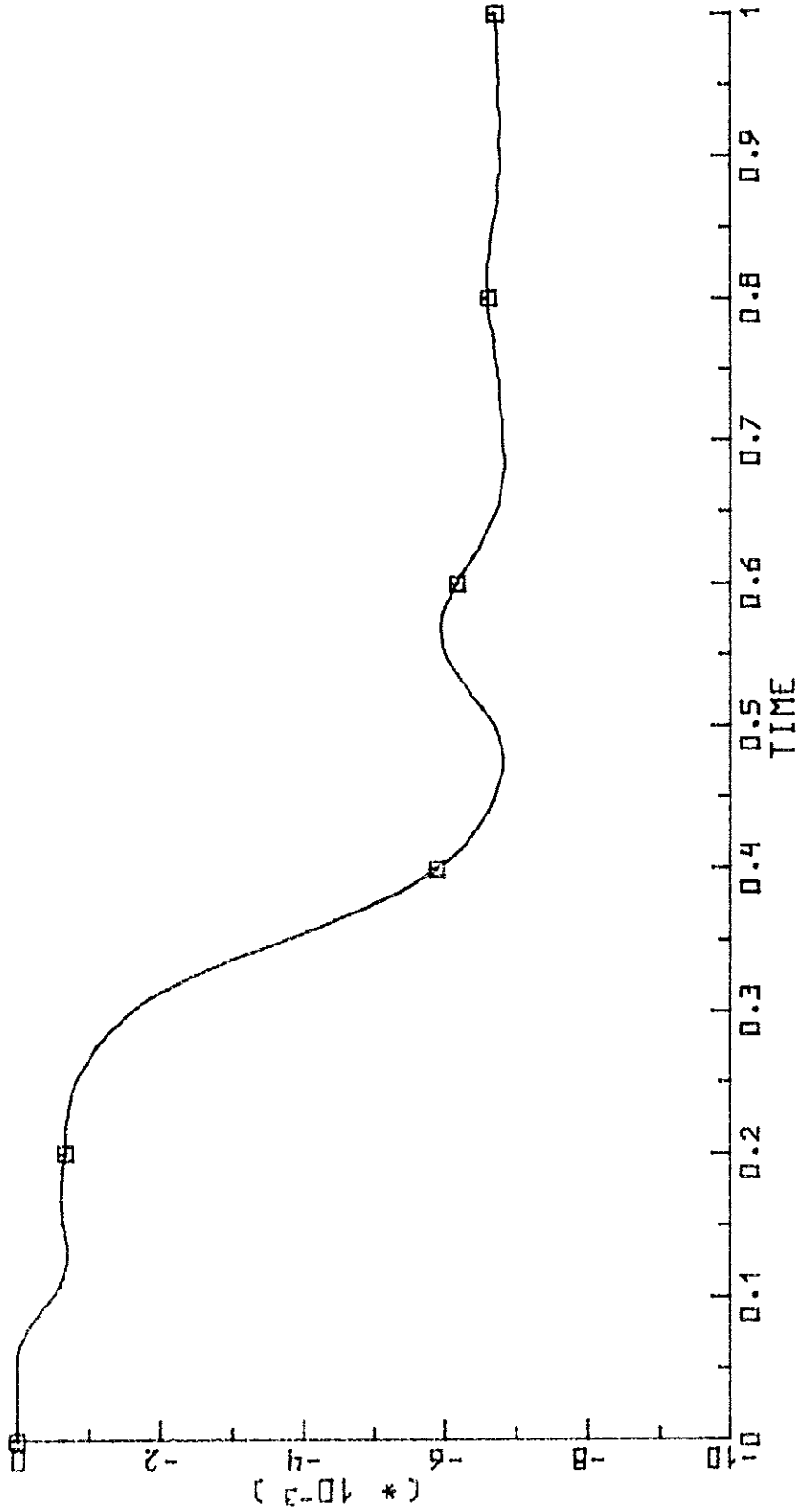
SIMSLIP,
RC-80661, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

DV2 □ = REAKTIONSTAGSKRAFT 2



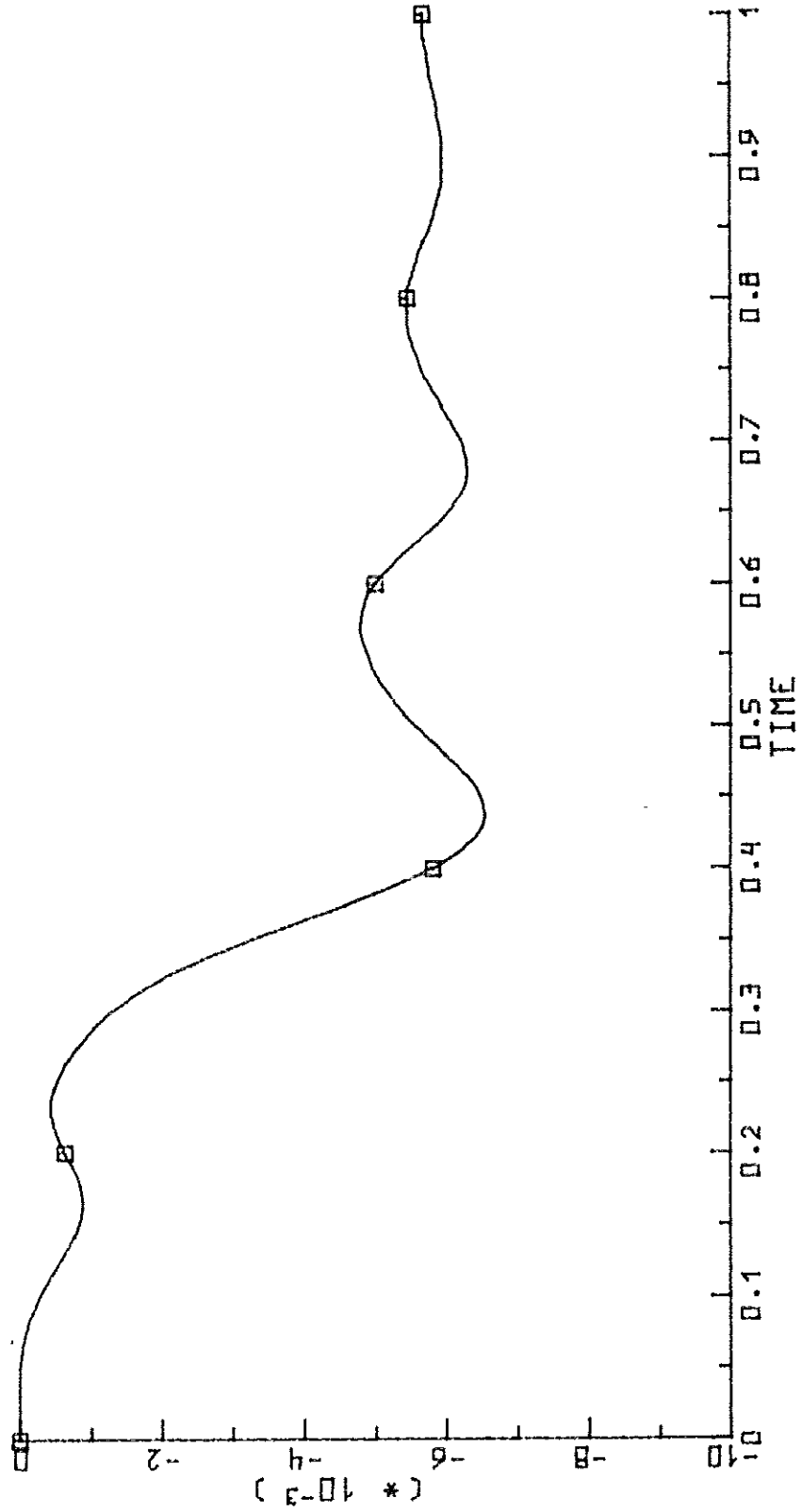
SIMSLIP,
RC-B066I, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

FI □ = VÄGNINGSVINKEL B066IRAM



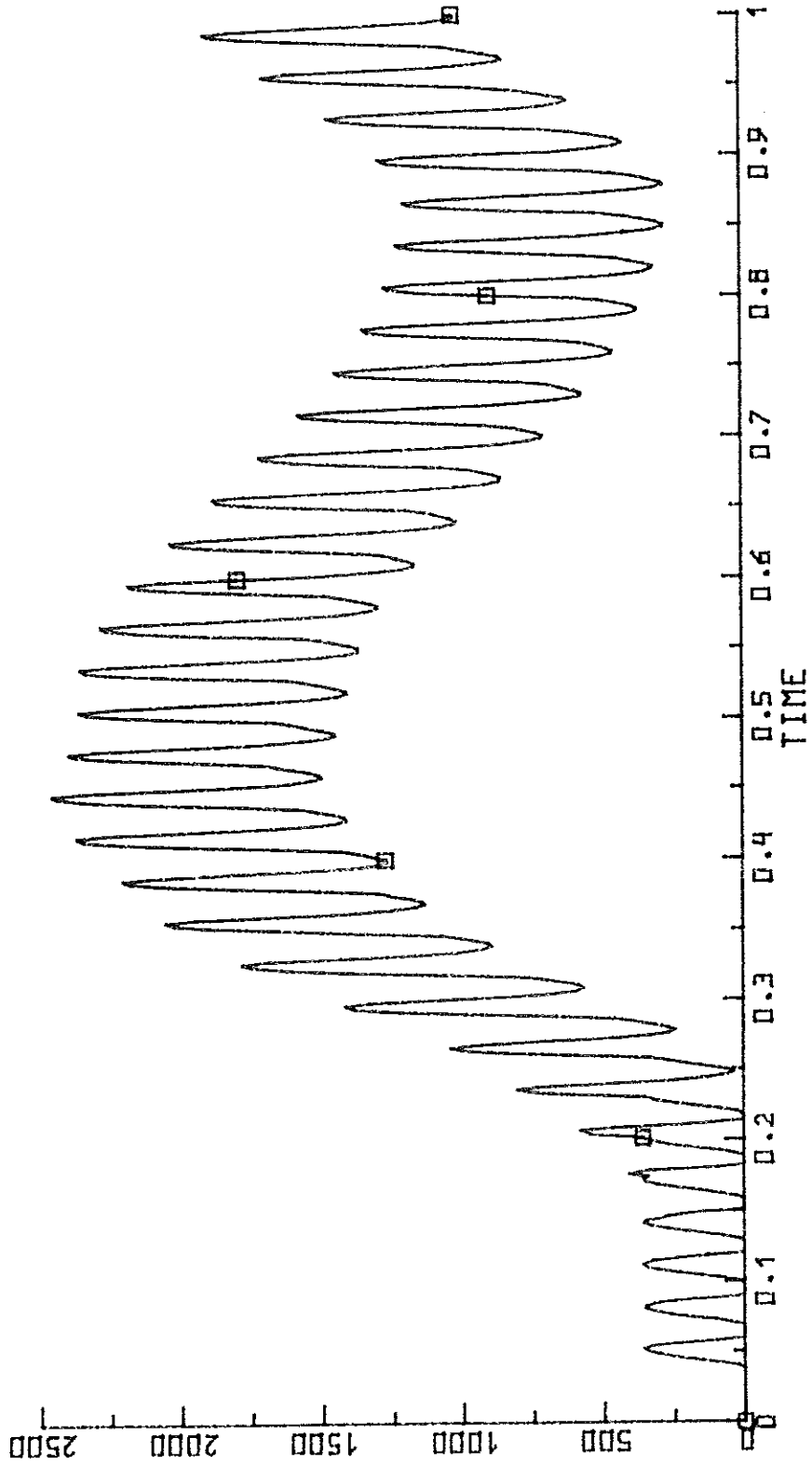
SIMSLIP,
RC-BOGGI, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

CHI □ = NICKNING BOGGIRAM



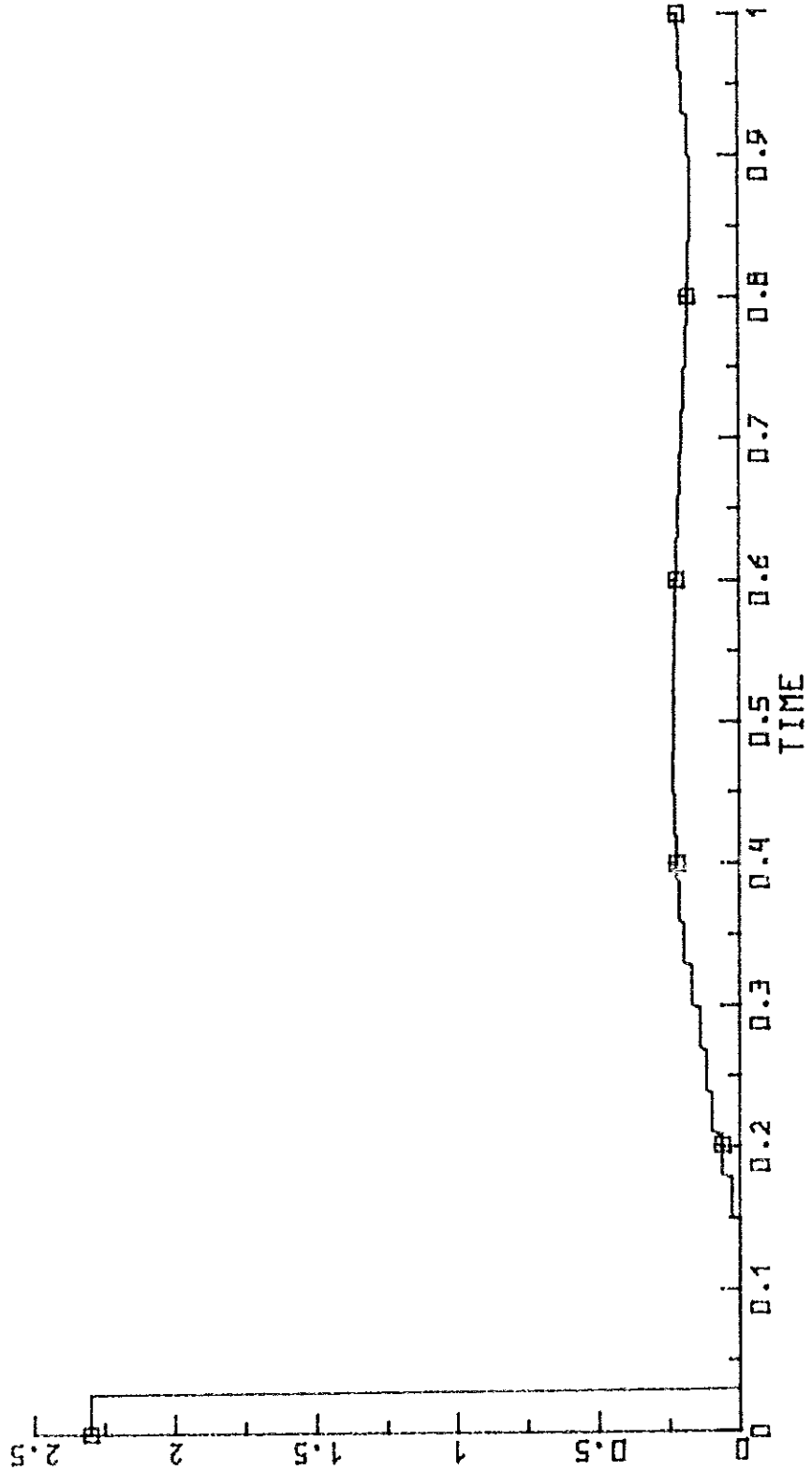
SIMSLIP,
RC-BOGGI, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

IA1 □ = MOTORSTROM 1

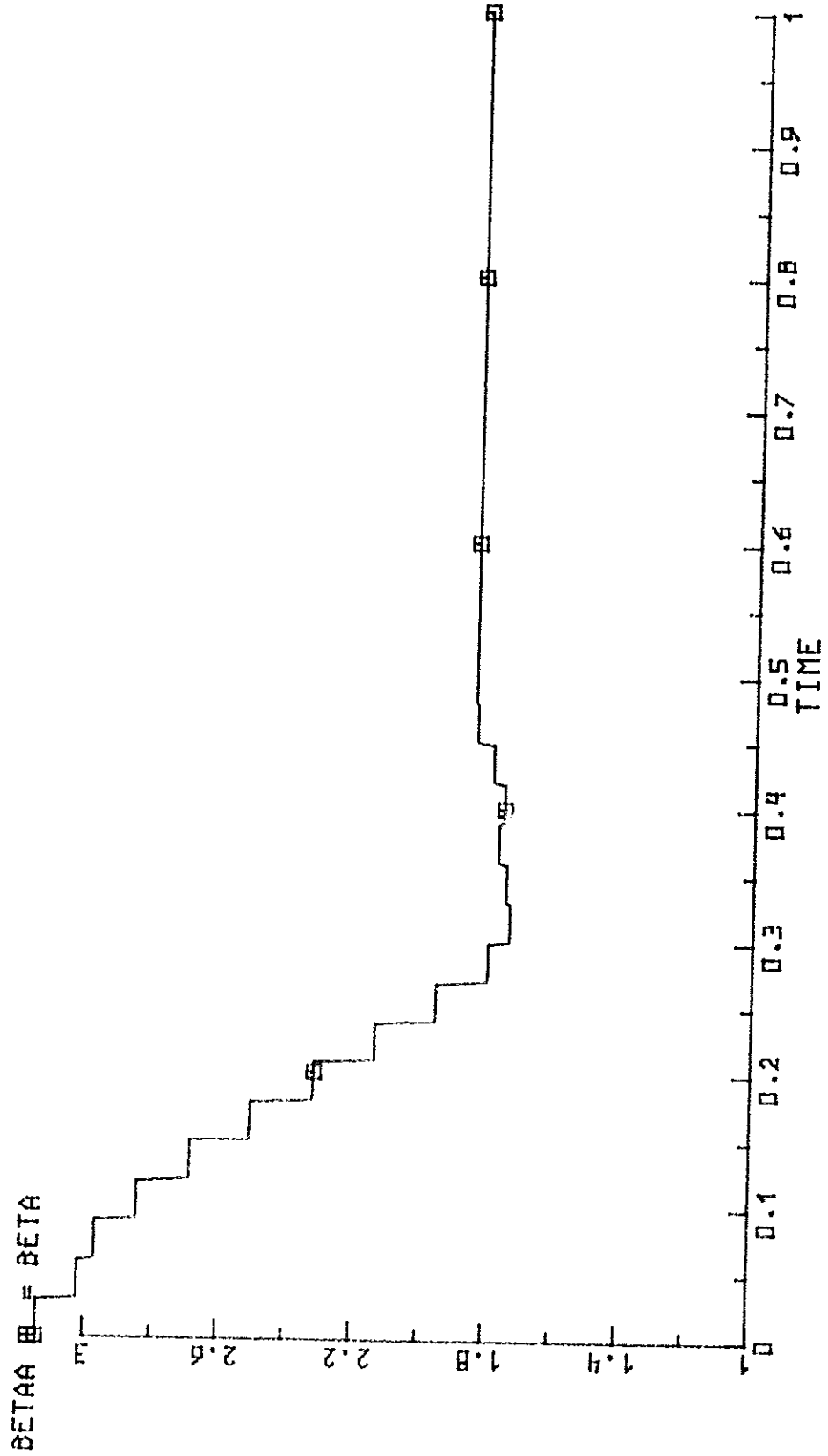


SIMSLIP,
RC-80661, VERSION MED TVA MOTORER,
V=60KM/H OCH JREFA=20000A.

ALFAA α = ALFA

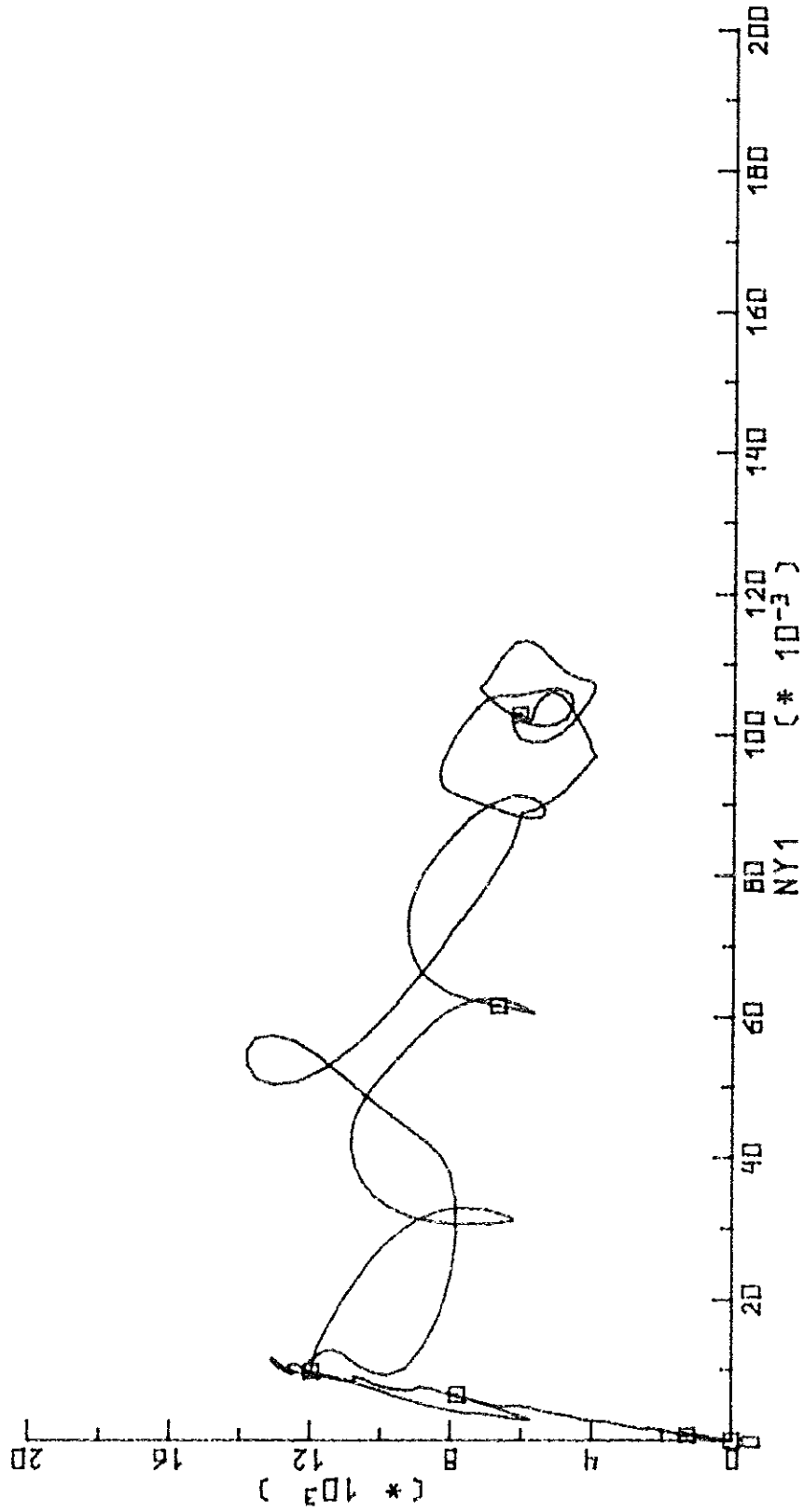


SIMSLIF,
RC-B0661, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,



SIMSLIP,
RC-80661, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

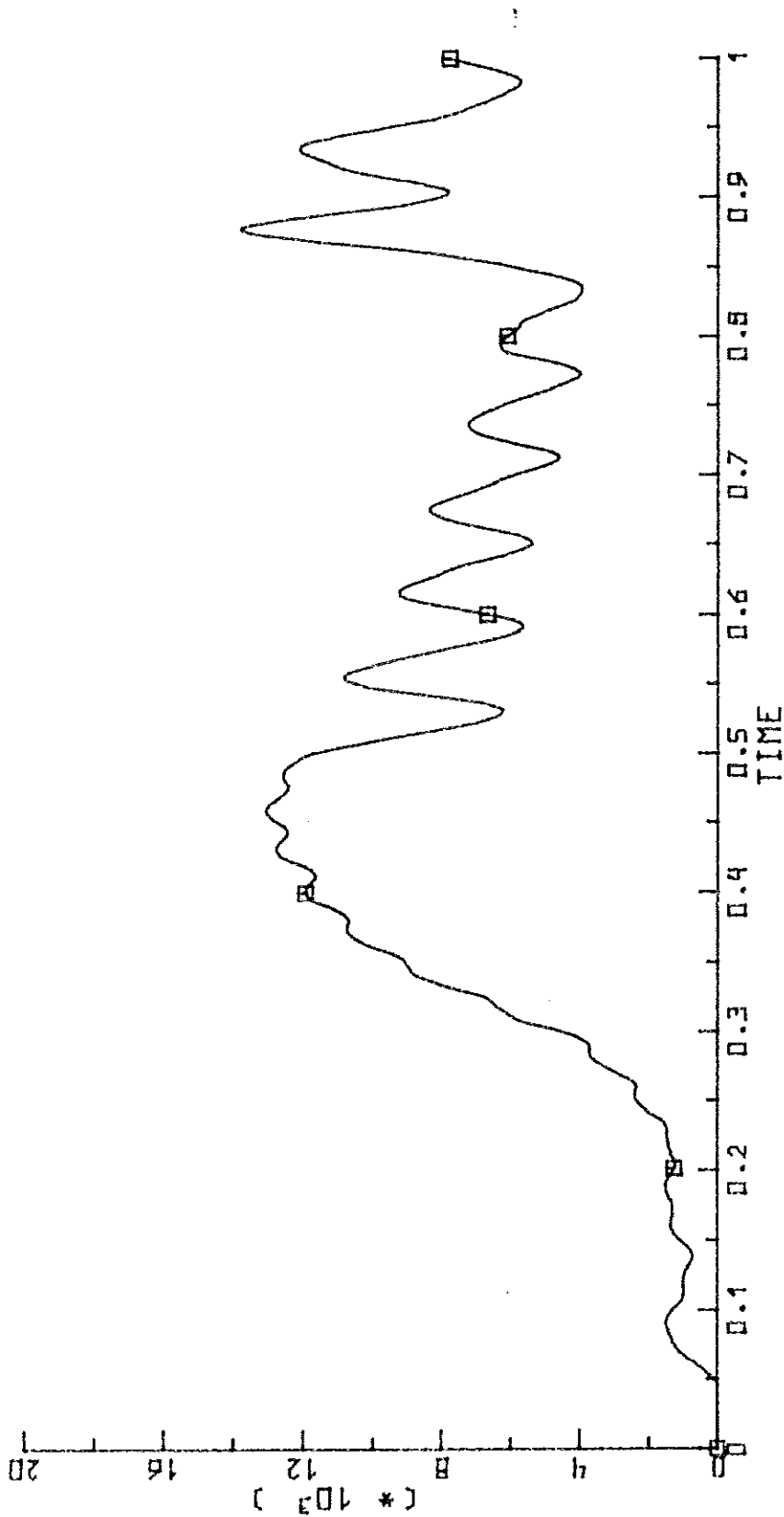
MI1 □ = KOPPLINGSMOMENT 1 SOM FUNKTION AV KRYP



691 / 0907 AA (999) 81-04 100 000 Ansvarig: POH

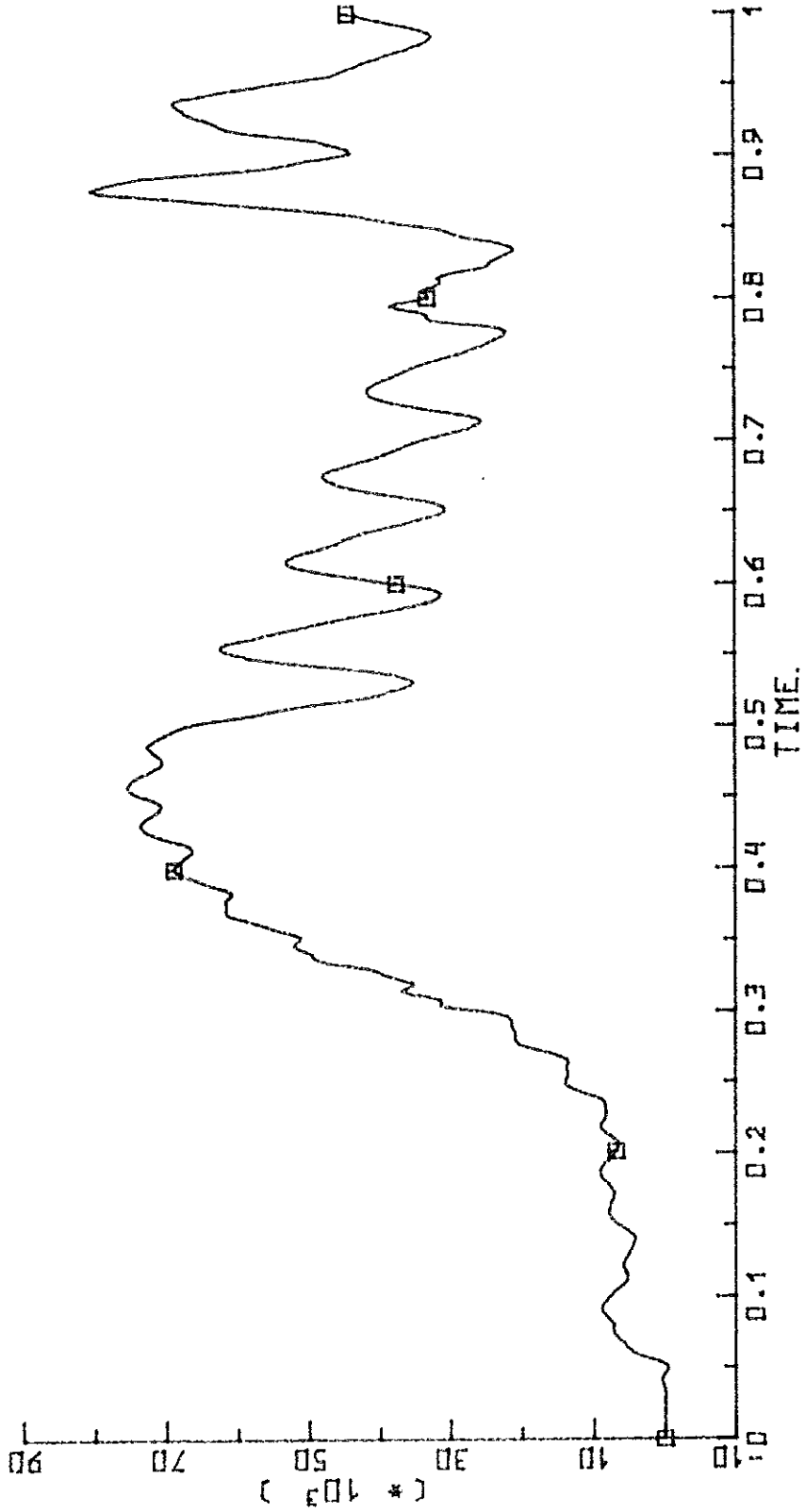
SIMSLIP,
RC-80661, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

M11 □ = KOPPLINGSMOMENT 1



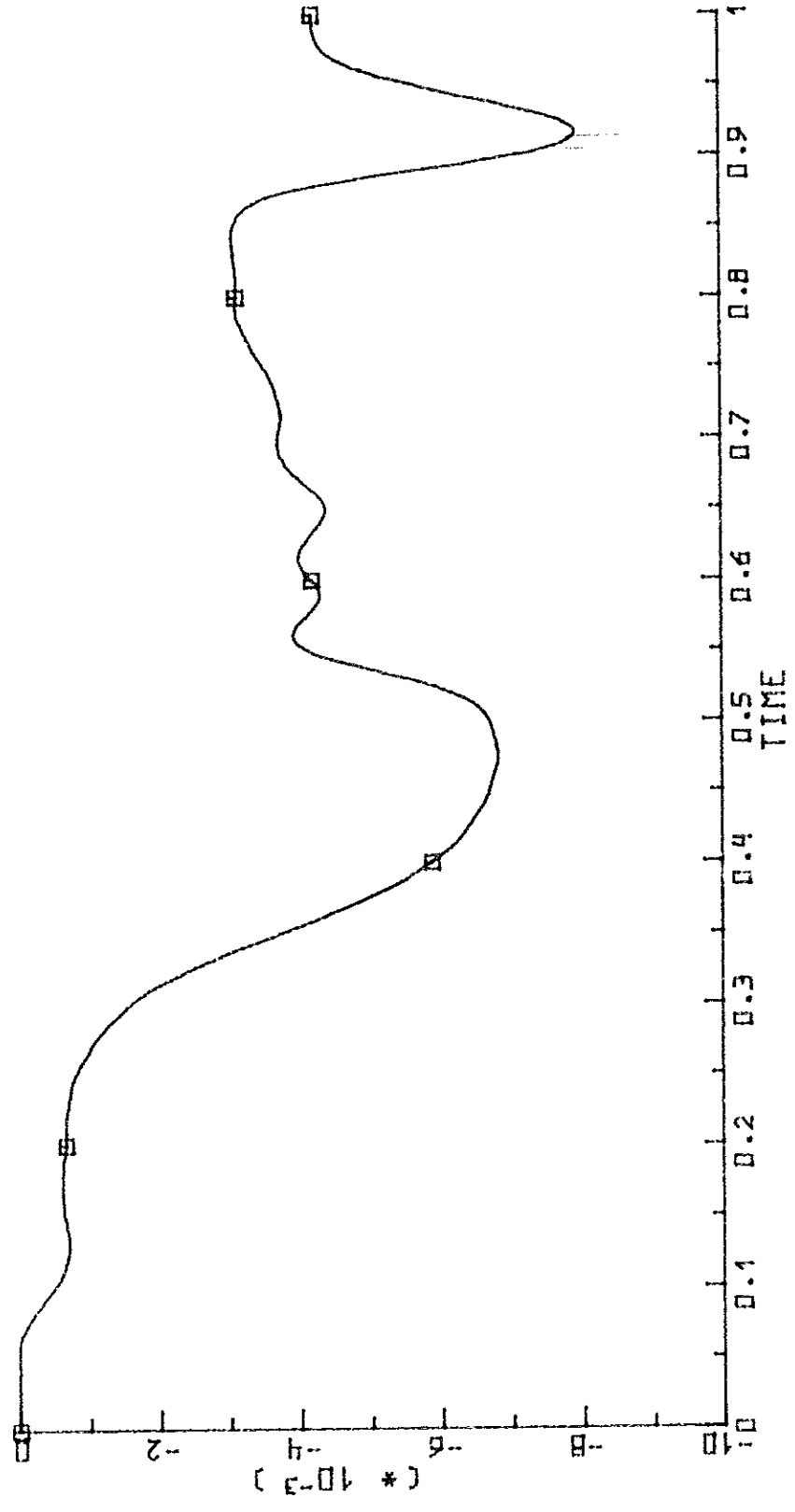
SIMSLIP,
RC-BÖGGI, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=20000g.

DV1 □ = REAKTIONSTAGSKRAFT 1



SIMSLIP,
RC-80661, VERSION MED TVA MOTORER,
V=60KM/H OCH JREFA=2000A,

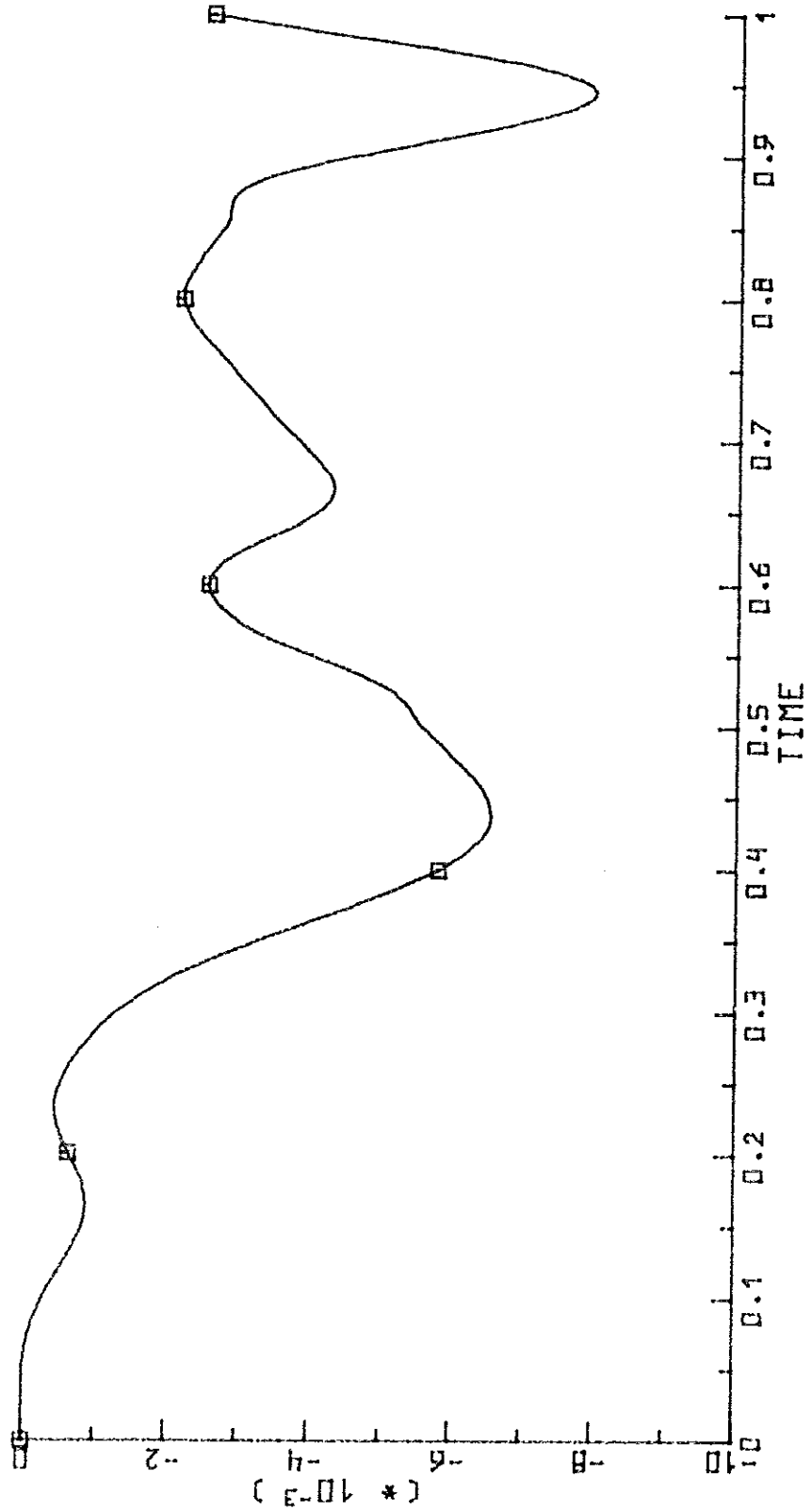
FI □ = VAEGNINGSVINKEL B0661RAM



6917 0507-AA (989) 81-04 100 000 Ansvarig: FOR

SIMSLIP,
RC-8066GI, VERSION MED TVÅ MOTORER,
V=60KM/H OCH JREFA=2000A,

CHI □ = NICKNING 8066GIRAM



Plottning

1. Logga in på en terminal som är ansluten till en plotter.
2. Skriv `.RPLOT`.
3. Datorn frågar nu: `FRAME-NUMBER, OPTION?`
4. Beroende på vilken terminal du har svarat du: T4663, 1200, S4096 eller T4662, 1200, S4096
(Detta är enda gången som du svarar med ovanstående vid frågan framenummer, options?)
5. Nu frågar datorn efter `SNUMBNR?`, varvid du då skriver ditt `snumbnr`.
6. Än en gång frågar datorn: `FRAME-NUMBER?`
Du svarar då med ett nummer, vilket då motsvarar det diagram i indatafilen, där diagram 1 innehåller variabeln efter första `DAPLOT` etc. Alternativt kan man svara `ALL` på frågan om vilka diagram som skall plottas, varvid alla diagram ritas.
7. Datorn frågar nu: `PAPERFORMAT?`
Svara då med `M`

Efter vagnretur börjar nu plottern att rita. Det är alltså viktigt att plottpenna och papper har laddats på plottern. När diagrammet är klart börjar datorn om från punkt 6. (Dvs om du inte svarat `ALL` innan).

8. Om du vid användning av T4663 svarat `ALL` på frågan `frame-number, options?`, ställer datorn ytterligare en fråga (`ROTATE?`) efter det att du svarat `M` (på frågan `paperformat`), Du svarar då med `Y` varvid plottern börjar arbeta.
9. När alla diagram är plottade (eller efter 2 vagnreturer), "frågar" datorn: `DIRECT KEEP OR RELEASE!` Beroende på om du vill behålla eller suddas ut plottdata svarar du `KEEP` eller `RELEASE`.
10. Datorn upprepar nu förfarandet från punkt 5 ovan. Om du vill gå ur `Rplot` slår du bara vagnretur varvid du återigen befinner dig i katalogen.

Ändring av program

Ändringar och omkompileringar av våra programpaket sker på följande sätt:

1. Logga in på `CTKU17` och ta fram den aktuella filen (källkod) på arbetsarean.
2. Gör dina ändringar.
3. Skriv `RESAVE` filnamn.
4. Kompilera (i `BCD`-kod) genom att skriva `FRN* = ;KTSBIN(NOGO, BCD)`.

Om ingen felutskrift kommer fram på skärmen är ändringarna skrivna på korrekt Fortran. Vid felutskrift upprepas förfarandet från punkt 2 ovan.

Program EMOTORX (modell av endast en motor)

Katalog: CTKU17
 Filnamn källkod: EMOTORX (XJOB81/EMOTOR)
 " kompilerad: EMOTORX-0
 Indatafil: INDATA (XJOB81/INDATA)
 Körinstruktion: RUN INDATA

Back-up filer ges inom parentes

Parameterlista

Y-storheter:

IA = Y(1) = Ankarström
 OMEGA = Y(2) = Rotorvarvtal
 UF = Y(3) = Filtreerad spänning (dvs från motorström omvandlad).
 UFR = Y(4) = Integrationsdel vid reglering

F-storheter:

F(1) = Ankarströmmens derivata
 F(2) = rotoracceleration

Z-storheter:

ALFA = Z(1) = tändvinkel brygga 1
 ALFA1 = Z(2) = avkommuteringsvinkel
 ALFA2 = Z(3) = påkommuteringsvinkel brygga 1
 BETA = Z(4) = tändvinkel brygga 2
 BETA2 = Z(5) = påkommuteringsvinkel brygga 2
 APSI = Z(6) = motorflöde
 MV = Z(7) = motormoment
 IS1 = Z(8) = påkommuteringsström brygga 1
 IS2 = Z(9) = " brygga 2
 MB = Z(10) = belastningsmoment
 S = Z(11) = uppräkningsvariabel
 FI = Z(12) = löpande tidsstorhet

40-100-81-02 100000

IAPI = Z(12) = ankarström vid periodbyte
IR = Z(14) = referensvärde för ankarström
DIFF = Z(15) = storhet för kontrollutskrift
DIFF2 = Z(16) = storhet för kontrollutskrift
FEL = Z(17) = reglerfel
TVS = Z(18) = tändvinkelsjustering

A-storheter:

OMEGAN = A(1) = nätvinkelfrekvens
RA = A(2) = resistans motorns ankare
RG = A(3) = resistans glättningsinduktor
R1 = A(4) = resistans transformator
IA = A(5) = induktans motor
LG = A(6) = induktans glättningsinduktor
L1 = A(7) = induktans transformator
JT = A(8) = tröghetsmoment för rotorn
C = A(9) = motorkonstant
US = A(10) = spänning från trafo (max)
K = A(11) = resistor för omvandling av ström till spänning
TF = A(12) = tidskonstant för filter
TR = A(13) = tidskonstant för regulator
PK = A(14) = propotionalitetskonstant
IRO = A(15) = avsedd strömreferens

Z(179) = IAPI = strömvärde som tas 1 ggr/period

Z(180) = JAM = summan av alla hastigheter för rotorn

Z(181) = IR = referensvärde för strömmen

Z(182) = FEL = reglerfel

Z(183) = TVS = tändvinkelsignal

A-storheter:

A(148) - A(161) = A(1) - A(15) i EMOTORX (en motor)

FILT	KVASI	ANAME	MODELL-BESKRIVNING
A(11)	A(160)	M12	M12
A(12)	A(161)	M13	M13
A(13)	A(162)	M14	M14
A(14)	A(163)	M23	M23
A(15)	A(164)	M24	M24
A(16)	A(165)	IREFA	Strömreferens till reglersystemet
A(17)	A(166)	IREFB	Används inte
A(18)	A(167)	TRACE	T (Filtrets tidskonstant)
A(19)	A(168)	K1	K1
A(20)	A(169)	K2	K2
A(21)	A(170)	K3	K3
A(22)	A(171)	K4	K4
A(23)	A(172)	K5	K5
A(24)	A(173)	IMAN	IMAN (Val av utstyrning)

Induktiva kopplingskonstanter mellan
Glättningsreaktorerna

(Regulator-konstanter för strömreglering)

OBS! Används ej i nuvarande version

SIMPAC-beteckningar i program.

FILT	KVASI	ZNAME	MODELL-BESKRIVNING
Z(1)	Z(167)	ALFAA	α_A
Z(2)	Z(168)	BETAA	β_A
Z(3)	Z(169)	ALFAB	α_B
Z(4)	Z(170)	BETAB	β_B
Z(5)	Z(171)	ALFA2A	α_{2A}
Z(6)	Z(172)	BETA2A	β_{2A}
Z(7)	Z(173)	ALFA2B	α_{2B}
Z(8)	Z(174)	BETA2B	β_{2B}
Z(9)	Z(175)	ALFA1A	α_{1A}
Z(10)	Z(176)	ALFA1B	α_{1B}

FILT	KVASI	ZNAME	MODELL-BESKRIVNING
Z(11)	Z(177)	MVA	Vridande moment ut från motor A resp. B. Se vidare avsnittet Fältsidan (modell 1).
Z(12)	Z(178)	MVB	
Z(13)	Z(179)	JAPIA	i_{api1}
Z(14)	Z(180)	JAPIB	
Z(15)	Z(181)	PSIA	Ψ_A
Z(16)	Z(182)	PSIB	
Z(17)	-	MB1	M_{B1}
Z(18)	-	MB2	
Z(19)	Z(183)	JS1A	i_{S1A}
Z(20)	Z(184)	JS2A	i_{S2A}
Z(21)	Z(185)	JS1B	i_{S1B}
Z(22)	Z(186)	JS2B	i_{S2B}
Z(23)	Z(187)	S	S (2 motorer) (N i 1 motor)
Z(24)	Z(188)	EMKA	E_{mkA}
Z(25)	Z(189)	EMKB	E_{mkB}
Z(26)	Z(190)	DJA3	i_{a3}
Z(27)	Z(191)	DJA4	i_{a4}
Z(28)	Z(192)	FI	$\varphi(\omega t)$
-	Z(193)	OMEGAA	ω_{mA}
-	Z(194)	OMEGAB	ω_{mB}
Z(29)	Z(195)	JMEDA	Medelvärdet av strömmen genom motor A resp B i den föregående perioden
Z(30)	Z(196)	JMEDB	
Z(31)	Z(197)	JZMA	Hjälpvariabler för bestämning av strömmarnas medelvärden föregående period.
Z(32)	Z(198)	JZMB	
Z(33)	Z(199)	HJSIG	Variabler i strömregulatorn (se reglersystemet)
Z(34)	Z(200)	TVSIG	

FILT	KVASI	YNAME	MODELL-BESKRIVNING
Y(1)	Y(73)	IA1	i_{a1}
Y(1)	Y(74)	IA2	i_{a2}
Y(3)	-	OMEGA1	ω_{mA} motorvarvtal (rad/s)
Y(4)	-	OMEGA2	ω_{mB}
Y(5)	Y(75)	IAINT1	Hjälpvariabler för medelvärdesberäkningar av strömmarna föregående period
Y(6)	Y(76)	IAINT2	
Y(7)	Y(77)	UAF1	Filtrerade spänningar med vilka reglersystemet bestämmer nya tändvinklar
Y(8)	Y(78)	UAF2	
Y(9)	Y(79)	UIFR	Max filtrerad spänning uppintegrerad. (Används av regl-syst).
Y(10)	Y(80)	DUMMY	Används inte

Program KVASI (modell av huvudkrets med två
motorer i SIMSLIP)

Katalog: CTKU17
Filnamn källkod: KVASI
Filnamn kompilerad: KVASI-0
Indatafil: PTH (XJOBBS1/PTV)
Körinstruktion: RUN PTH

Observera, att fil RAM-0 (omskriven subrutin i SIMSLIP) måste finnas på katalogen.
Backup-filer ges inom parentes.

Programmet körs via indatafilen, som bl a anropar erforderliga huvudprogram (SIMPAC etc) och subrutiner. I indatafilen sättes de olika A-konstanterna och eventuella startvärden på valfria Y-variabler.

COMMON/CKRYP/MYO1, MYO2, SOME1, SOME2 har införts för att göra det möjligt att i motorsubrutinen ändra på maximala friktioner i ADH-rutinen.

Parameterlista elektrisk modell

Se parameterlista för program FILT.

EXPERIMENT MED DATORSTYRD KÄNNANDE ROBOT

GUNNAR SANDIN

TORD WULLT

INSTITUTIONEN FÖR REGLERTEKNIK
LUNDS TEKNISKA HÖGSKOLA

AUGUSTI 1982

LUND INSTITUTE OF TECHNOLOGY DEPARTMENT OF AUTOMATIC CONTROL Box 725 S 220 07 Lund 7 Sweden		Document name	
		MASTER THESIS	
		Date of issue AUG 1982	
Author(s) Gunnar Sandin Tord Wullt		Document number	
		CODEN:LUTFD2/(TFRT=5271)/1-086/(1982)	
		Supervisor	
		LARS NIELSEN	
		Sponsoring organization	
Title and subtitle			
Experiment med datorstyrd kännande robot.			
Abstract			
<p>Electronics and software for control of a two-wheeled, touchsensing vehicle, known as the Turtle.</p> <p>Program applications.</p>			
Key words			
Classification system and/or index terms (if any)			
Supplementary bibliographical information			
ISSN and key title			ISBN
Language	Number of pages	Recipient's notes	
SWEDISH	86		
Security classification			

DOKUMENTOATABLAD RT 3/81

Distribution: The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

Innehållsförteckning.

1. Beskrivning av Turtle.

2. Idéer och utveckling i kronologisk följd.

3. Elektronik.

3.1 Inledning.

3.2 Styr signaler till motorer och tuta.

3.3 Signaler från touchsensorer.

3.4 Varvgivare.

3.5 Hastighetsstyrning.

3.6 Interface-box.

4. Programvara-primitiver.

4.1 Inledning.

4.2 Användarkommandon.

4.3 Principer vid användning av primitiverna.

4.4 Primitivernas uppbyggnad.

4.5 Programutskrift med kommentarer.

5. Tillämpningsexempel.

5.1 Inledning.

5.2 Positionsjustering.

5.3 Enkel operatörskommunikation.

5.4 Bana med okända föremål.

6. Bruksanvisning.

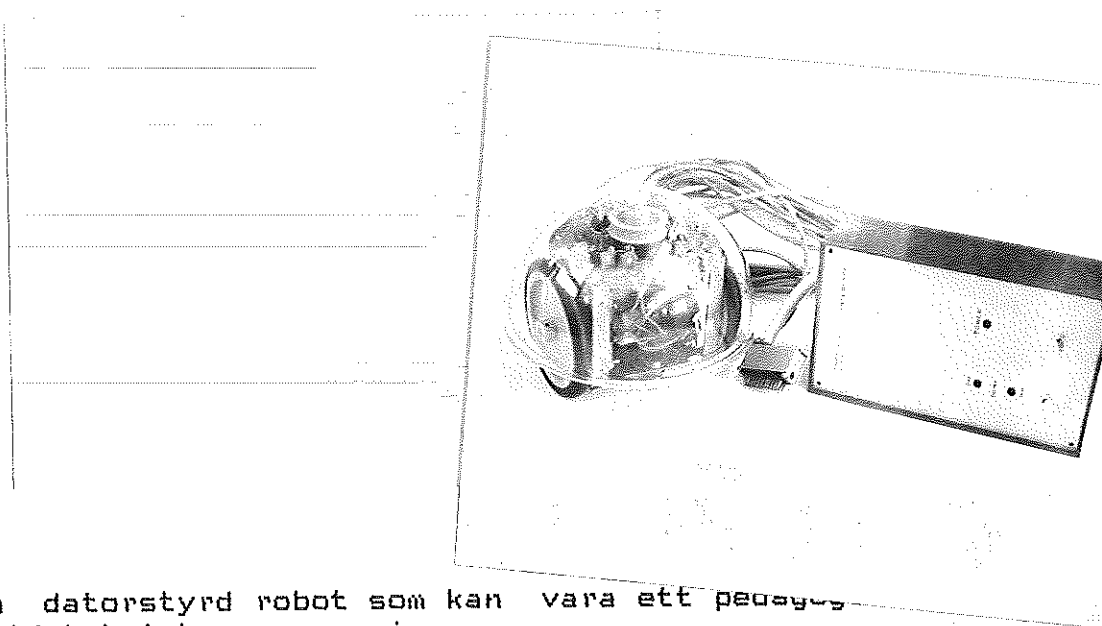
7. Slutsatser om rörelserna samt förslag till förbättringar.

Appendix 1. Centrumkorrigerings vid enhjulssväng.

Appendix 2. Omvandling mellan grader och pulser vid svängar.

Appendix 3. Turtle-beskrivningar från Terrapin, Inc.

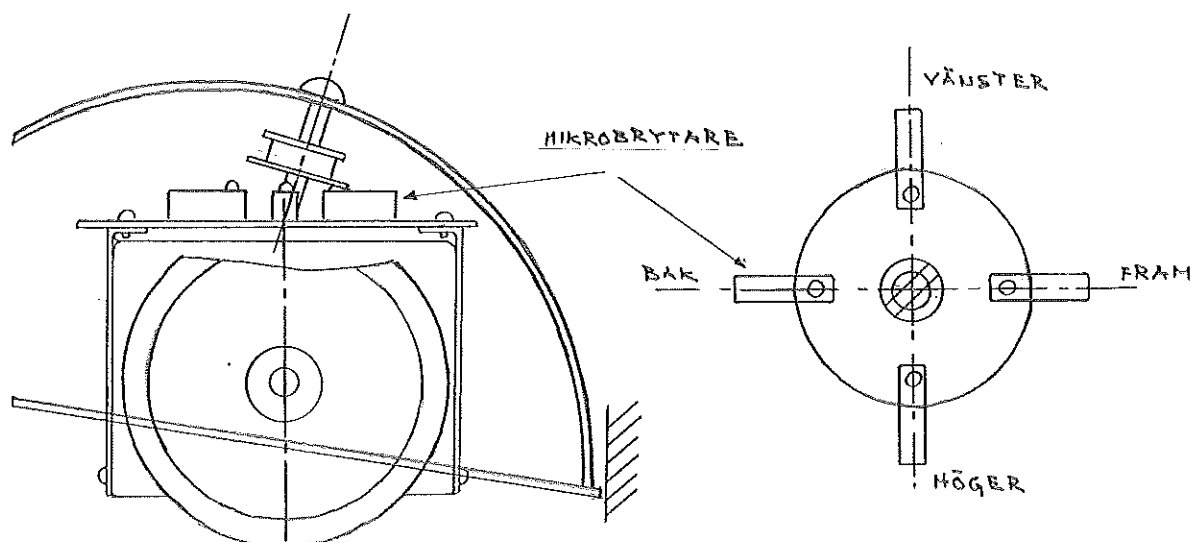
1. Beskrivning av Turtle.



Turtle är en datorstyrd robot som kan vara ett pedagogiskt hjälpmedel vid teknisk programmering.

Turtle har två stora hjul som drivs av var sin motor med växellåda. Varje motor kan styras separat framåt eller bakåt. Hastigheten kan dock inte väljas utan styrningen är av on/off-typ.

Under "skalet" sitter fyra mikrobrytare, se bilden



Beroende på i vilken riktning Turtle kör in i något hinder, kommer en eller två av de fyra mikrobrytarna att tryckas in.

Det finns också en penna som ritar på golvet, en tuta och ett par lysdioder som skall föreställa Turtle's ögon.

Turtle är lämplig att pröva tex. sökningsalgoritmer med.

Drivspänning: 12-18 v.

Strömbehov: < 1 A.

Storlek: ung. 30 cm i diameter.

2. Ideer och utveckling i kronologisk följd.

När vi i januari 1982 hade bestämt oss för att göra examensarbete inom området robotstyrning och då speciellt styrning av Turtle, fick vi fria händer att bestämma vilka frågor som kunde vara intressanta att ta itu med. Mest med tanke på industrirobotar som förflyttar sig på golv, satte vi upp följande inledande mål:

Konstruktion av interface.

Programskrivning för

- a. inlärning av rörelsebanor,
- b. algoritmer för att undvika föremål i banorna,
- c. att minnas föremålens placering och utseende,
- d. labyrintsökning.

Praktiska begränsningar hos Turtle gjorde dock att målsättningen ändrades något. Det visade sig nämligen att Turtle's båda motorer gick med ojämn hastighet, vilket innebar oacceptabel reproducerbarhet för de mål som vi satt upp. Detta gav oss två valmöjligheter, nämligen att

1. anpassa målen och problemställningarna efter Turtle's begränsningar, utnyttja enbart dess förmåga att känna när den rör vid något, eller
2. införa någon form av reglering som gör att man säkrare vet vilka rörelser (sträckor och vinklar) som Turtle gör på golvet.

Vi tyckte att de ursprungliga målen var intressantast, och valde därför att på något sätt införa återkoppling. Två regleringsmöjligheter diskuterades.

Man skulle kunna utföra lokal hastighetsåterkoppling på motorerna ute på Turtle och genom att mäta tiden få information om positionen på golvet.

Den andra möjligheten vore att införa lägesgivare som talar om hur långt varje hjul har gått. Då vet man hur långt Turtle har gått, eller vilken vinkel Turtle har vridit. Man kan dessutom reglera Turtle's kurs med hjälp av datorn. Detta senare alternativ lät spännande och ledde oss fram till följande idéer:

Lägesgivare blev egentligen varvgivare genom att vi med optokopplare och reflektorer på motoraxeln fick information om de båda motorernas varvantal och därmed Turtle's läge på golvet. Med denna information tillgänglig och med lämplig programvara tänkte vi göra det möjligt för datorn att

påverka respektive motors drivning och därmed få en kursreglering.

Nu började arbetet med att få en vettig och störningsfri signal från varvgivarna/optokopplarna till datorn. Samtidigt funderade vi på olika sätt att reglera kurs och läge med pulsantalet som underlag.



Vår första tanke var att påverka respektive hjuls hastighet genom att snabbt sätta av och på styrsignalerna till motorerna. Möjligheten med denna pulsbreddsmoduleringsstyrning förkastade vi därför att Turtle's mekanik uppfattade så korta avstängningar som 1-2 ms, medan realtidssystemet ej tillåter kortare avstängningar än 20 ms.

En annan möjlighet i detta läge var att låta Turtle utföra sin rörelse okorrigerad, samt använda datorn till att, med hjälp av antal pulser från respektive hjul, utföra död räkning på banan. Vi hade dock svårt att hitta någon enkel modell för rörelsemönstret p.g.a. att Turtle's drift var oregelbunden och snabbt varierande i tiden.

Istället valde vi att införa en möjlighet att under gång justera de båda motorernas hastigheter med två extra styrsignaler till Turtle. (Detalj-information om elektroniken finns i kapitel 3.)

Vid utveckling av mjukvara för styrning av Turtle hade vi som mål att skapa enkla och naturliga kommandon/primitiver av typen Framåt(sträcka), Sväng höger(vinkel), Tuta, Hämta aktuellt status, Ändra omvandlingskonstanter, etc.

Dessa kommandon utformas som procedurer som anropas från t.ex. program för sökning eller operatörskommunikation.

(Beskrivningen av primitiverna finns i kapitel 4.)

Som avslutning på examensarbetet använde vi primitiverna i ett program där Turtle följer en inmatad bana och undviker föremål som ligger i banan. Vi skrev också en enkel operatörskommunikation samt en procedur där Turtle justerar sin position och riktning i rummet med hjälp av en speciell Turtle Adjustment Device.

3. Elektronik

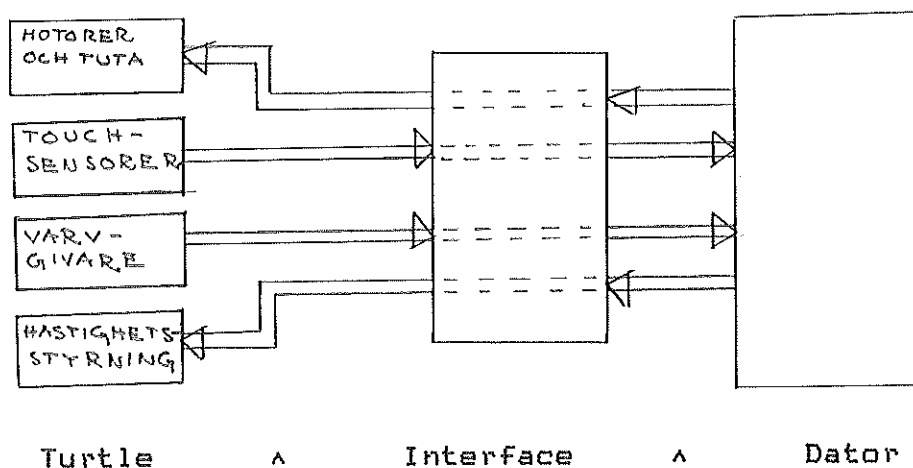
3.1 Inledning.

Till Turtle fanns vid examensarbetets början följande styr-
signaler av on/off -typ:

- I. Motorerna, fram respektive back.
- II. En tuta med valmöjlighet mellan två tonlägen.
- III. En penna som ritar på golvet.
- IV. Turtle's ögon (lysdioder).

Från Turtle kunde vi få signaler från de fyra touchsenso-
rerna (mikrobrytarna).

För att få Turtle att gå rakt införde vi en reglering av
kursen. Denna reglering fungerar på följande sätt:
En varvgivare ger pulser som byter nivå för varje motorvarv.
Signalen samplas och nivåbytarna räknas av datorn. Detta görs
för båda hjulen. När antalet pulser skiljer sig åt för de
båda hjulen, sänks hastigheten på det hjul som ligger
"före". I form av elektronik krävs alltså en varvgivare och
en hastighetsstyrning för varje hjul.



För att inte behöva ha en onödigt tjock kabel bestämde vi
oss för att avstå från styrningen av pennan, Turtle's ögon
och valmöjligheten av tonhöjden för tutan.

Observera att en ledare finns för varje funktion. Över-
föringen av signalerna är alltså ej kodad.

Samtliga dessa signaler har anpassats i ett interface till
reglertekniks datorer LSI-11.

3.2 Styr signaler till motorer och tuta.

Turtle's krav på signalerna:

5-18 volt -----> ON.

0 volt -----> OFF.

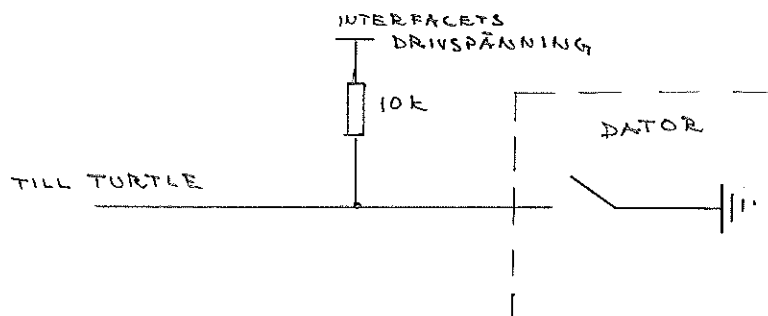
Datorns utsignaler.

Vi har använt funktionen Bitout(true /false):

Bitout(true) -----> Datorns utgång jordad.

Bitout(false) -----> Datorns utgång frikopplad och Turtle rör sig eller tutar.

Anpassningen av Turtle till dator har skett med följande koppling:



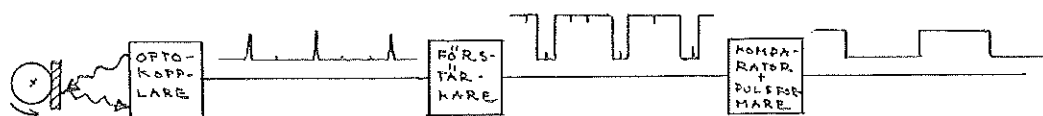
3.3 Signaler från touchsensorerna.

Under "skalet" sitter fyra mikrobrytare. Se kap.1. När en mikrobrytare inte är nedtryckt befinner sig motsvarande utgång på en nivå lika med drivspänningen. Om turtle kör in i något och en mikrobrytare trycks ned, kommer utgången istället att bli jordad.

Dessa signaler kan kopplas direkt till datorns "Bitin"-ingångar.

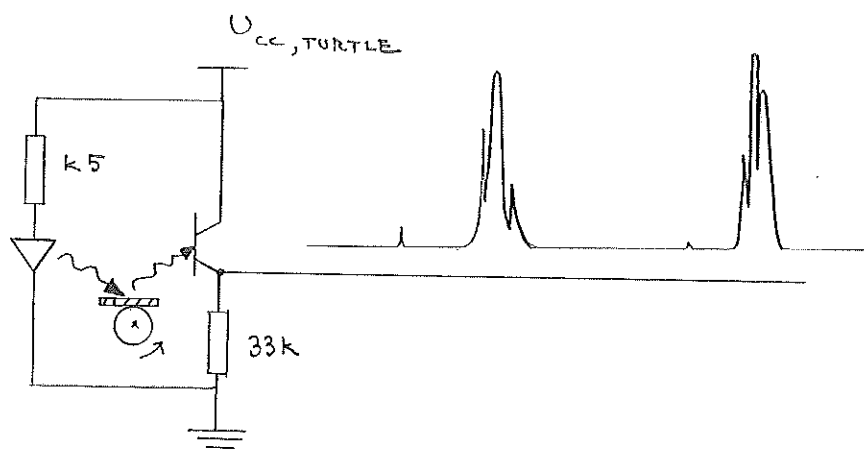
3.4 Varvgivare.

Varvgivaren består av en optokopplare med reflektor fäst på motoraxeln och ett puls-formande nät.



Optokopplare.

Följande koppling har använts:



1 Hjulvarv \Leftrightarrow 300 motorvarv.

1 Motorvarv \Leftrightarrow ca. 3.6 mm rörelse på golvet.

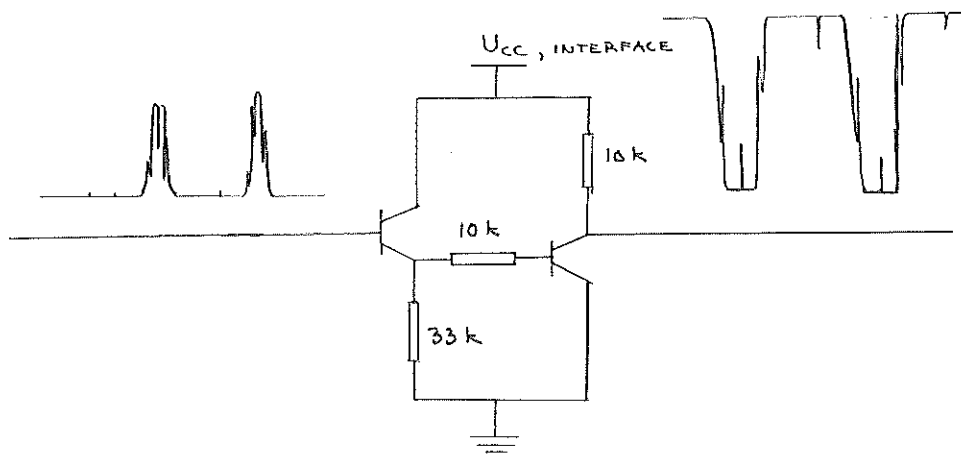
Drivspänning	varvtal
12 volt	40 varv per sekund
18 volt	60 varv per sekund

Observera att om datorn direkt skall räkna antalet motorvarv så måste man vid val av drivspänning ta hänsyn till hur ofta datorn kan sampla en insignal. Med Reglertekniks LSI-11:or och realtidskärna kan vi ha en frekvens på högst 50 hz. Detta gör att vi inte kan använda en högre drivspänning än 12.5 volt.

Förstärkaren.

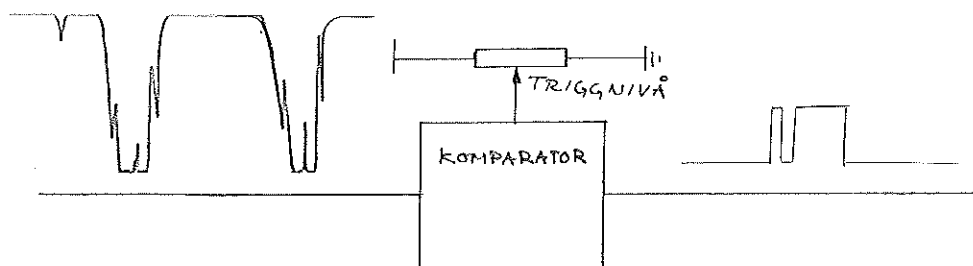
För att säkrare kunna trigga på pulsen från optokopplarna, och för att en försämring av reflektorerna inte skall påverka tillförlitligheten, har vi lagt in en switchförstärkare efter optokopplarna.

schema :

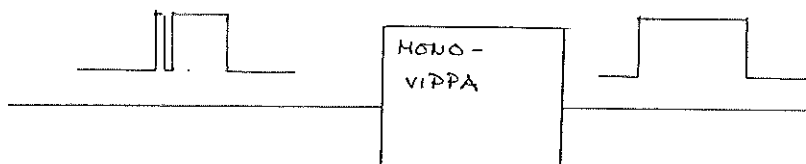


Komparator och pulsformare.

Det är önskvärt att, med tanke på brus och störningar, kunna välja den bästa trignivån för efterkommande nät. Därför har vi lagt en komparator på ingången till pulsformaren.

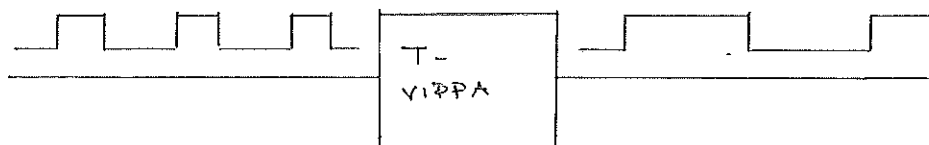


Därefter följer en monostabil vipa med en pulslängd som är något längre än pulsen från komparatorn.



Monovippan används för att undvika mer än en triggning på varje puls.

Som sista länk ligger en t-vippa som ger en fyrkantpuls med nivåbyte för varje inkommande puls från monovippan.



3.5 Hastighetsstyrningen.

Momentet hos en motor, och vid konstant belastning varvtalet, bestäms av den ström som går genom motorn.

För att få Turtle att gå rakt fanns från början en anordning för att ge motorerna en konstant ström. Strömmen kunde väljas med en styrspänning över en potentiometer. Potentiometrarna (fram höger/vänster, back höger/vänster) justeras så att Turtle går så rakt som möjligt. Denna anordning visade sig inte fungera så bra.

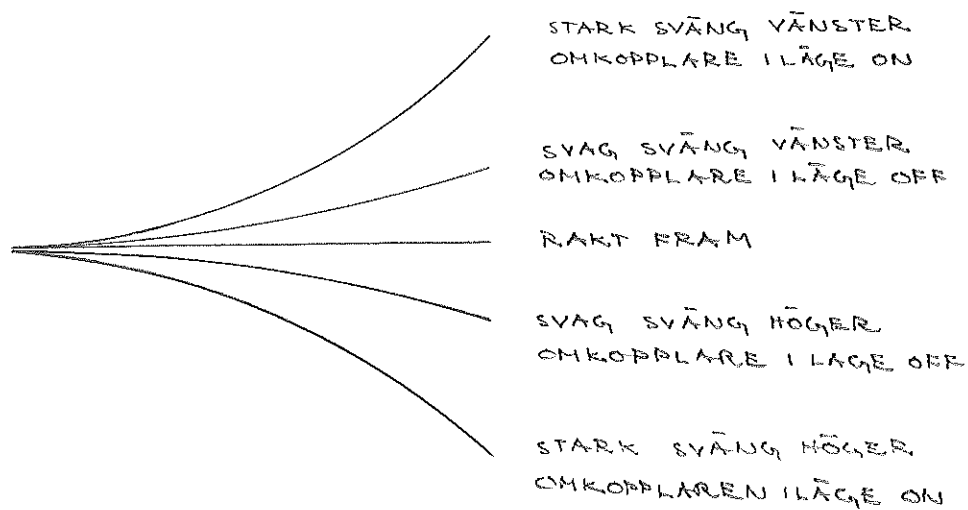
Vi antar att det bland annat beror på temperaturberoende hos motorerna och anordningen som skall ge motorerna konstant ström, samt parametervariation hos växellådan.

För att få Turtle att gå rakt införde vi den i kap. 3.1 beskrivna kursregleringen. Vi gjorde det möjligt att med en digital styrsignal från datorn välja mellan två något åtskilda hastigheter.

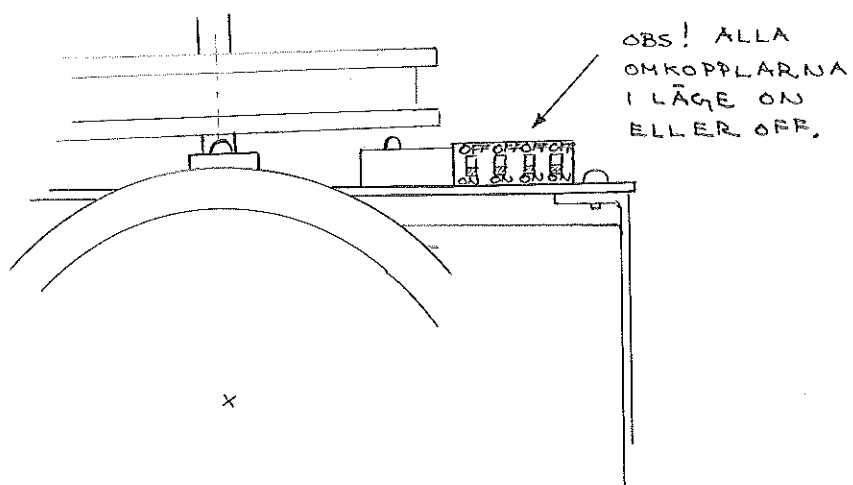
Detta ger tre olika rörelsetillstånd:

- I. Svag sväng vänster.
- II. Ungefär rakt fram.
- III. Svag sväng höger.

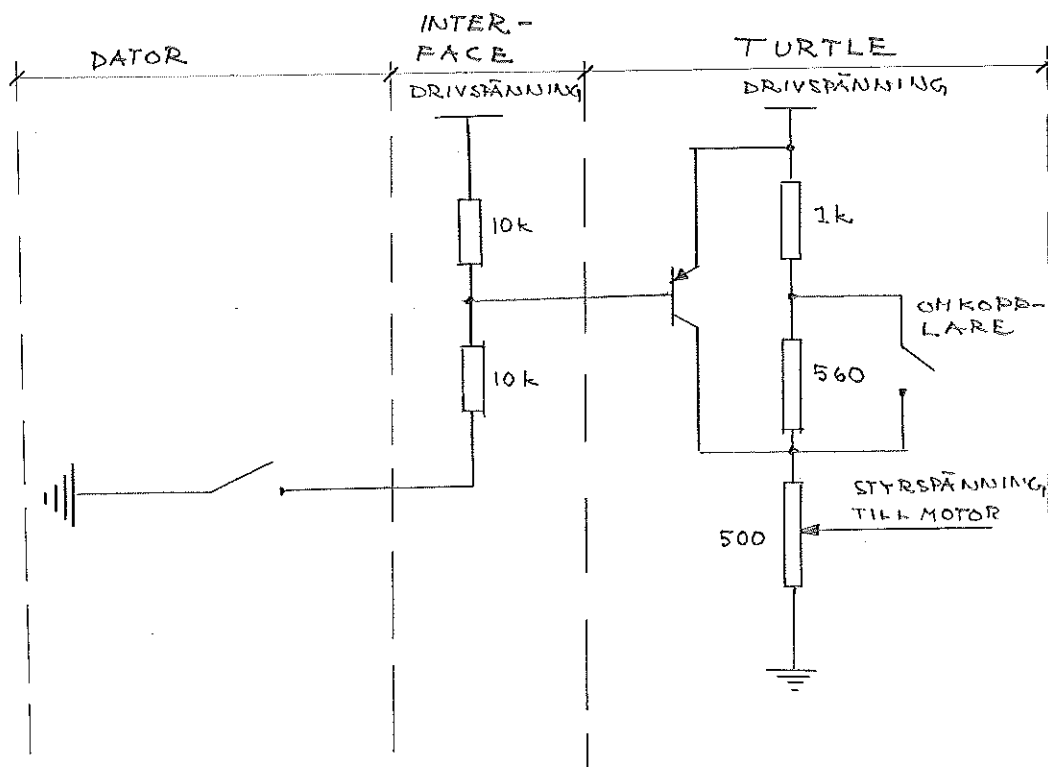
Med en omkopplare som sitter på Turtle kan man dessutom välja hur mycket de två hastigheterna skall skilja sig åt. Vi kan alltså välja hur mycket Turtle skall svänga höger respektive vänster.



Omkopplarens placering på Turtle:



Schema över styrspänningsväljaren:



Observera att när Turtle kör blir det ett spänningsfall på ungefär 2 volt över kabeln mellan interface och Turtle.

Injusteringen av styrningen görs lämpligen så att man med oscilloskop studerar reglersignalerna från datorn till Turtle. Potentiometrarna ställs så att reglering sker lika ofta på de båda hjulen. Se nästa kapitel.

3.6 Interface-box.

I interfacet har vi placerat all elektronik som anpassar signaler mellan Turtle och dator. Till detta räknas också förstärkare och pulsformare för optokopplarna.

Från interfacet utgår två sladdar, en till datorn och en till Turtle.

Två omkopplare finns. Med den ena kan man stänga av spänningen till elektronik och Turtle. Den andra sätter datorns ingång digital-in(5) sann eller falsk. Denna ingång svarar mot mjukvarufunktionen Bitin(0,5) och kan användas i programmen.

På baksidan av interfacet sitter fyra kontakter för banankontakter.

De två mittersta skall anslutas till spänning och jord. Spänningen skall vara 14 volt vilket ger 12.5 volt till Turtle efter spänningsfallet över kabeln. Strömförbrukningen är mindre än 1 ampere.

Från de två andra kan man få signaler från förstärkarna efter optokopplarna. Det kan vara bra om man vill kontrollera optokopplarnas funktion. Observera att ett på dessa kontakter inkopplat mätinstrument kan ge störningar så att t-vippan slår om för ofta.

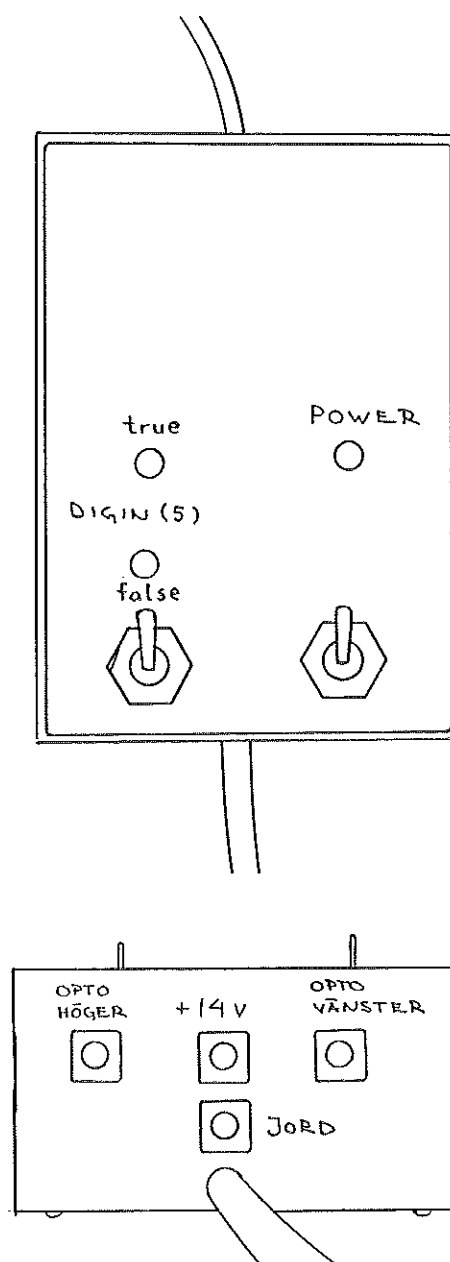
Turtle är förbunden med interfacet via en 10 meter lång sladd. Sladden har 15 ledare och dessa används enligt följande:

Funktion

Höger motor fram
 Höger motor back
 Vänster motor fram
 Vänster motor back
 Tuta
 Hastighetsstyrning höger
 Hastighetsstyrning vänster
 Mikrobrytare fram

Färg

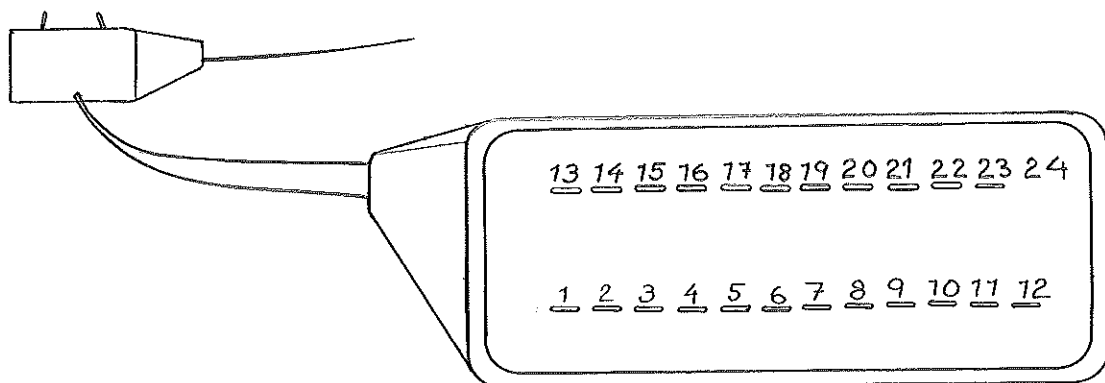
Röd-grön
 Röd-gul
 Mörkgrön
 Vit
 Skär
 Grå
 Blå-röd
 Orange



Mikrobrytare höger	Lila
Mikrobrytare bak	Röd
Mikrobrytare vänster	Ljusgrön
Pulser från höger optokopplare	Gul
Pulser från vänster optokopplare	Blå
Spänning	Brun och svart
Jord	Skärmen

Till spänningen har vi använt två ledare för att minska spänningsfallet över kabeln.

Interfacet kopplas till LSI-11 med en 24-polig kontakt som är ansluten till en redan klar busskommunikation.



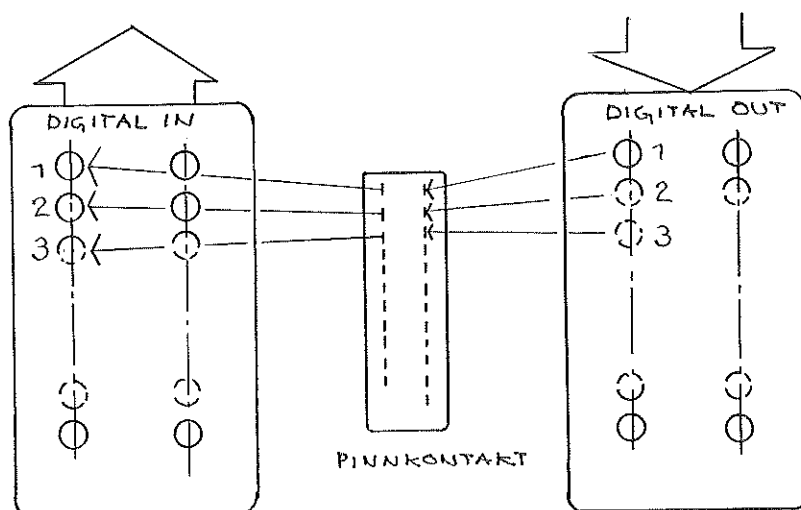
Kontaktens pinnar (obs alla används inte) överför följande signaler:

Pinne	Funktion	Mjukvarufunktion
1	H.motor fram	Bitout(0,1,true/false)
2	H.motor back	2
3	V.motor fram	3
4	V.motor back	4
5	Tuta	5
6	H.hast.styrsignal	6
7	V.hast.styrsignal	7
8	-	-
9	-	-
10	-	-
11	-	-
12	Jord	-
13	Mikrobr.fram	Bitin(0,1)
14	Mikrobr.höger	2
15	Mikrobr.bak	3
16	Mikrobr.vänster	4
17	Digin(5) väljare	5

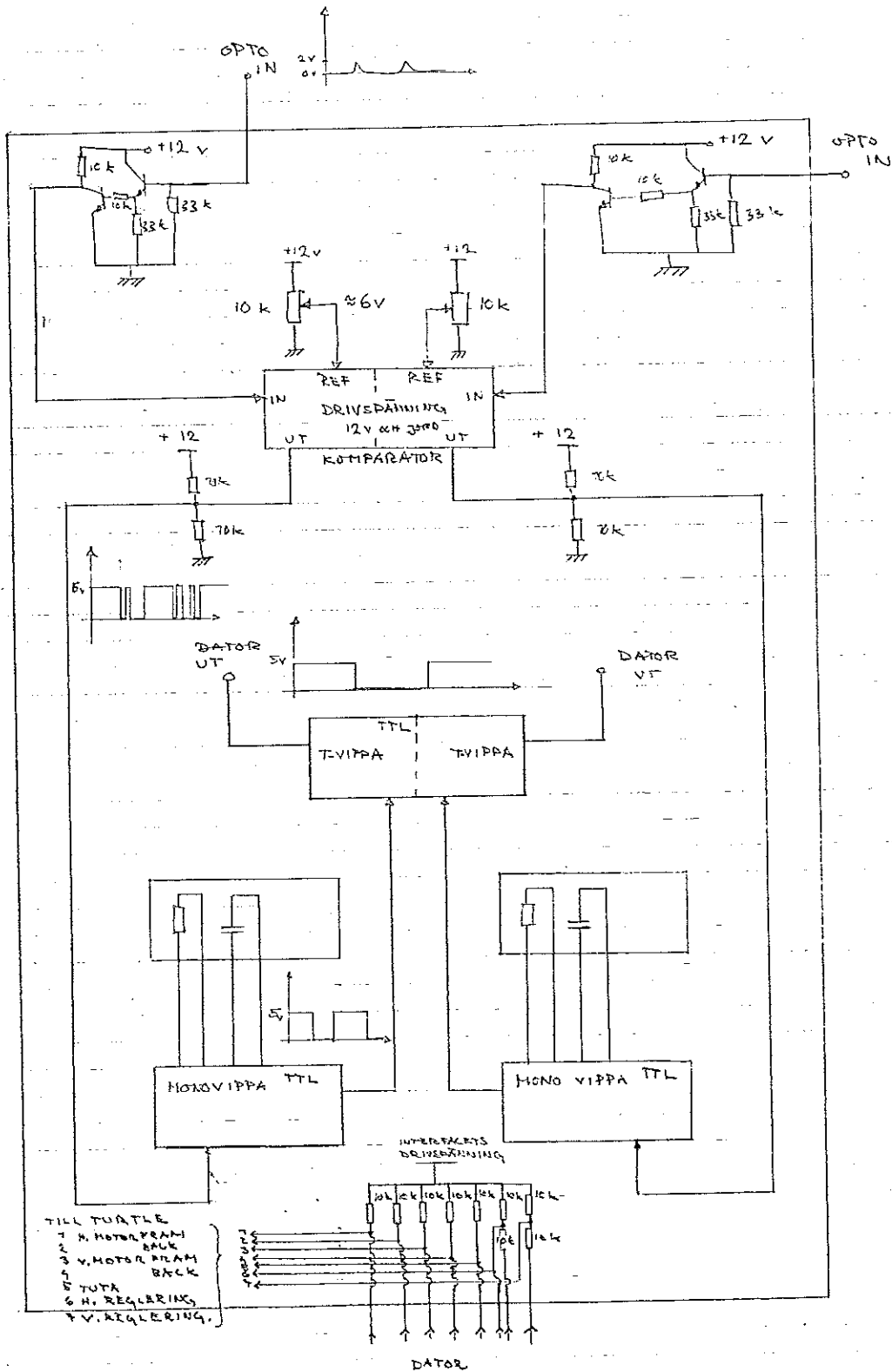
18	H.varvgivare	6
19	V.varvgivare	7
20	-	-
21	-	-
22	-	-
23	-	-
24	-	-

Genom att koppla ett oscilloskop till en kontaktpinne kan man studera t.ex. reglersignalerna från datorn.

PÅ LSI-11 är pinnkontakten parallellkopplad med utgångarna digital-out respektive ingångarna digital-in. Man kan alltså koppla in sitt oscilloskop på dessa in och utgångar när man vill studera datorns in och utsignaler.



SCHEMA ÖVER INTERFACE



4. Programvara - primitiver.

4.1 Inledning.

När en användare får Turtle i sin hand och slår sig ner vid datorn, vill han ha möjligheten att från sitt program ge kommando om fundamentala rörelser, samt få reda på vad som hänt efter kommandot.

Vårt mål har därför varit att skapa primitiver som gör det enkelt att styra Turtle. Styrningen tänkes ske genom att användaren till sitt förfogande har ett antal procedurer som han anropar med lämpliga argument.

4.2 Användarkommandon.

Följande proceduranrop kan göras i användarens Pascal-program:

Forw(length)
Gå framåt length cm.

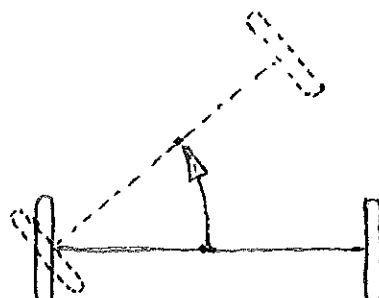
Back(length)
Backa length cm.

Turnright(angle)
Sväng höger angle grader. Ena hjulet går framåt, andra bakåt, vilket medför att Turtle's centrum behåller sin position under vridningen.

STurnright(angle)
Sväng höger angle grader. S står för Single. Endast ena hjulet roterar. Därigenom kommer Turtle's centrum att förflyttas under vridningen (1 → 2 i figuren). Vill man använda denna vridning (som är något exaktare) kan man göra en korrektion för ändringen av centrum. Se vidare i appendix 1.

Turnleft(angle)
Som Turnright fast vänster.

STurnleft(angle)
Som STurnright fast vänster.



Tut(number)

Genererar number korta ljudsignaler.

Getstatus

Hämtar information om Turtle's aktuella tillstånd. Lämplig att använda efter en kommenderad rörelse t.ex. för att kontrollera om Turtle kört in i något föremål.

När Getstatus anropas kopieras ett antal variabler och läggs i en record status. Öppnar man denna kan man testa på eller använda följande variabler:

FORWON	Booleska variabler som vi-
BACKON	sar den senaste rörelsen.
RIGHTON	
LEFTON	
SINGLERIGHTON	
SINGLELEFTON	
FRONTTOUCHON	Booleska variabler som vi-
BACKTOUCHON	sar om någon touchsensor
RIGHTTOUCHON	(mikrobrytare) är intryckt.
LEFTTOUCHON	
REACHED	Sann om kommenderad sträcka
	eller vinkel blev uppnådd.
DIST	Heltal som anger till-
	ryggalagd sträcka i cm.
RDIST	Heltal. Anger antal pulser
LDIST	från höger resp. vänster
	varvgivare.
WHEELFACTOR	Reellt tal som korrigerar
	skillnad mellan hjuldiam.
CONVERTF	Reella tal för omvandling
CONVERTB	mellan antal pulser och
CONVERTR	cm(grader) för de olika
CONVERTL	rörelserna. (se appendix 2)
CONVERTSR	
CONVERTSL	

Setconstants

Ger möjlighet att från terminal ändra samtliga kalibreringskonstanter (de sju sista variablerna i listan ovan). Se vidare InitTurtle.

InitTurtle

Initierar variabler och skapar de processer som används av primitiverna. Tystar tutan. Anropas från huvudprogrammet.

Man kan sätta kalibreringskonstanterna till egna värden. Detta görs efter InitTurtle i huvudprogrammet genom att öppna en record tg, som primitiverna använder, och göra

tilldelningar på de 7 sista variablerna ovan.

4.2 Principer vid användning av primitiverna.

En kommenderad rörelse slutförs alltid, antingen genom att den kommenderade sträckan/vinkeln uppnås eller genom att Turtle får en touch och stannar. Om två eller flera konkurrerande aktiviteter vill ge Turtle kommando kommer dessa att exekveras färdigt i tur och ordning (prioritet bestäms av användaren). En aktivitet kan alltså ej avbryta en pågående rörelse, ej heller en pågående parameteröverföring.

Vi illustrerar användningen av primitiverna med ett enkelt programexempel.

Order till Turtle: Gå framåt 100 cm. Om du stöter på ett föremål, backa 10 cm och tag av åt höger.

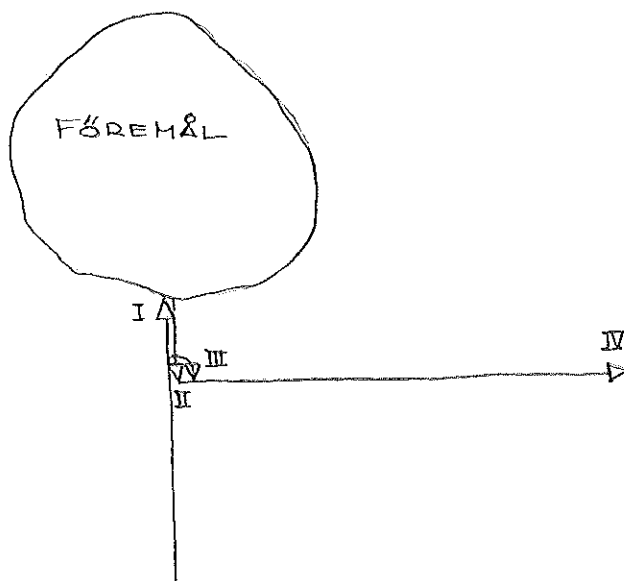
Detta ger följande program:

```

*
*
*           I
Forw(100);
Getstatus;
if status.FRONTTOUCHON then
  begin
    Back(10);           II
    Turnright(90);     III
    Forw(100);         IV
  end;
*
*
*

```

Antag att Turtle stöter i ett föremål under det första Forw(100)-anropet. Rörelsen på golvet blir då:



4.3 Primitivernas uppbyggnad.

När ett anrop av typen Forw(100) görs, händer följande:

1. Rörelsen skyddas genom att varje användarprocedur i början gör wait(user) och i slutet signal(user).
2. Booleska variabler falskställes och räknarvariabler nollställes.
3. Lämpliga booleska variabler som indikerar rörelsen sättes.
4. Den kommenderade sträckan i cm omvandlas till antal pulser.
5. En process TouchDisableTimer, som under ett litet tidsintervall stänger av touchsensorerna, startas genom att semaforen touchtimer signaleras.
6. En annan process Control, som varje tick kontrollerar touchsensorer, räknar pulser, reglerar kursen, samt kontrollerar om kommenderad distans är nådd, startas med hjälp av semaforen move.
Denna process bestämmer när rörelsen är klar.
7. När rörelsen är slutförd (signaleras från processen Control med en semafor ready), väntar vi ytterligare 3 tick för att få med eventuella pulser under nedbromsningen. Därefter stannas processen Control genom att den booleska variabeln moving sätts falsk.
8. Den tillryggalagda distansen för aktuell rörelse räknas ut.
9. Signal(user) gör det möjligt för nya primitivanrop.
10. Återhopp till användar programmet.

Samtliga primitiver dvs. Forw, Back, Turnright, Turnleft, STurnright, STurnleft, Tut, Getstatus, Setconstants är skyddade med semaforen user.

Synkronisering_i_huvuddrag.

```

procedure Forw(dist);
begin
  wait(user);
  .
  .
  moving:=true;
  signal(touchtimer);
  signal(move);
  wait(ready);
  moving:=false;
  .
  .
  signal(user);
end;

{process} procedure TouchDisableTimer;
begin
  .
  .
  while true do
    begin
      wait(touchtimer);
      .
      .
    end;
end;

{process} procedure Control
begin
  .
  .
  while true do
    begin
      wait(move);
      while moving do
        begin
          .
          .
          vid touch eller uppnådd distans
          görs här signal(ready);
          .
          .
        end;
      end;
    end;
end;

```

Med processen TouchDisableTimer gör man det möjligt att t.ex. backa igång, trots att någon touchsensor är nertryckt. Man undviker också kontaktstudsar hos fronttouchsensorerna vid igångsättning av en rörelse. Det som händer är att

variabeln touchenable sätts sann efter 30 tick. Touchenable (som används i processen Control) skyddas med semaforen mutex.

I processen Control görs följande:

1. Touchsensorerna kontrolleras. Om någon är intryckt görs signal(ready) som indikerar att rörelsen är slut.
2. Antal pulser räknas eventuellt upp. (Varje flank på en fyrkantvåg räknas.)
3. Kursen regleras. Om antal pulser från de båda motorerna skiljer sig åt, väljes en av två möjliga hastigheter (egentligen styrspänningar) på lämplig motor.
4. Om det antal pulser som kommenderades i användaranropet, har uppnåtts, görs signal(ready).
5. Punkt 1. tom. 4. görs varje tick, så länge rörelsen pågår.

4.4 Programutskrift med kommentarer.

```

(PRETUR)

type statustype = record
    FORWARDON,
    BACKON,
    RIGHTON,
    LEFTON,
    SINGLERIGHTON,
    SINGLELEFTON,
    FRONTTOUCHON,
    BACKTOUCHON,
    RIGHTTOUCHON,
    LEFTTOUCHON,
    REACHED      :boolean;
    DIST,
    RDIST,
    LDIST        :integer;
    WHEELFACTOR,
    CONVERTF,
    CONVERTB,
    CONVERTR,
    CONVERTL,
    CONVERTSR,
    CONVERTSL    :real;
end;

var touchenabled,
    moving      :boolean;
    out         :array[1..4] of boolean;

    kr,
    kl,
    movelimit   :integer;

    ts,
    status      :statustype;

    mutex,
    user,
    move,
    ready,
    touchtimer  :semaphore;

    dummy,
    dummy1,
    dummy2     :boolean;

```

PRIMITIVER

```

function Bitin(chan,bitnr:integer):boolean;external;

procedure Bitout(chan,bitnr:integer;value:boolean);external;

procedure Init;
begin
  with ts do
    begin
      FRONTTOUCHON:=false;
      RIGHTTOUCHON:=false;
      BACKTOUCHON:=false;
      LEFTTOUCHON:=false;
      FORWARDON :=false;
      RIGHTON :=false;
      BACKON:=false;
      LEFTON:=false;
      SINGLERIGHTON:=false;
      SINGLELEFTON:=false;
      REACHED:=false;
      RDIST:=0;
      LDIST:=0;
      DIST:=0;
    end;
    moving:=false;
    dommy1:=not Bitin(0,6);
    dommy2:=not Bitin(0,7);
    dammi:=true;
    kr:=0;
    k1:=0;
  end;

  procedure SetOutsFalse; {Out bestammer varje hjuls}
  var k:integer; {rorelse }
  begin
    for k:=1 to 4 do
      out[k]:=false;
    end;

  procedure MotionToOuts;
  begin
    with ts do
      begin
        out[1]:=FORWARDON or LEFTON or SINGLELEFTON;
        out[2]:=FORWARDON or RIGHTON or SINGLERIGHTON;
        out[3]:=RIGHTON or BACKON;
        out[4]:=BACKON or LEFTON;
      end;
    end;

  procedure SignToBitouts; {Aktuell rorelse laggs pa utgangarna}
  var k:integer;
  begin
    with ts do
      begin
        for k:=1 to 4 do
          if out[k] and (not REACHED)
            and (not FRONTTOUCHON)
            and (not RIGHTTOUCHON)
            and (not BACKTOUCHON)
            and (not LEFTTOUCHON) then
            Bitout(0,k,false)
          else
            Bitout(0,k,true);
        end;
      end;
    end;

  procedure PutInMotion; {Satter igang de bada processerna}
  begin
    SetOutsFalse;
    MotionToOuts;
    moving:=true;
    touchenabed:=false;
    signal(touchtimer);
    signal(move);
    SignToBitouts;
  end;

```

```

procedure Getstatus;
begin
  wait(user);
  status:=ts;
  signal(user);
end;

procedure Tut(number:integer);
var j:integer;
begin
  for j:=1 to number do
    begin
      Bitout(0,5,false);
      waittime(1*tick);
      Bitout(0,5,true);
      waittime(10*tick);
    end;
end;

procedure Forw(length:integer);
begin
  wait(user);
  if length > 0 then
    begin
      Init;
      ts.FORWARDON:=true;
      movelimit:=round(length*ts.CONVERTF-2); {subtraktion med 2 pulser }
      PutInMotion; {eftersom Turtle inte stan-}
      wait(ready); {nar omedelbart. }
      waittime(3*tick);
      moving:=false;
      ts.DIST:=round(ts.RDIST/ts.CONVERTF);
    end;
  signal(user);
end;

procedure Back(length:integer);
begin
  wait(user);
  if length > 0 then
    begin
      Init;
      ts.BACKON:=true;
      movelimit:=round(length*ts.CONVERTB-2); { Se Forw. }
      PutInMotion;
      wait(ready);
      waittime(3*tick);
      moving:=false;
      ts.DIST:=round(ts.RDIST/ts.CONVERTB);
    end;
  signal(user);
end;

procedure Turnright(angle:integer);
begin
  wait(user);
  if angle > 0 then
    begin
      Init;
      ts.RIGHTON:=true;
      movelimit:=round(angle*ts.CONVERTR-2); { Se Forw. }
      PutInMotion;
      wait(ready);
      waittime(3*tick);
      moving:=false;
      ts.DIST:=round(ts.RDIST/ts.CONVERTR);
    end;
  signal(user);
end;

```

```

procedure Turnleft(angle:integer);
begin
  wait(user);
  if angle > 0 then
    begin
      Init;
      ts.LEFTON:=true;
      movelimit:=round(angle*ts.CONVERTL-2);      { Se Forw.      }
      PutInMotion;
      wait(ready);
      waittime(3*tick);
      moving:=false;
      ts.DIST:=round(ts.RDIST/ts.CONVERTL);
    end;
  signal(user);
end;

procedure STurnRight(angle:integer);
begin
  if angle > 0 then
    begin
      wait(user);
      Init;
      ts.SINGLERIGHTON:=true;
      movelimit:=round(angle*ts.CONVERTSR-1);      { Se Forw. }
      PutInMotion;
      wait(ready);
      waittime(3*tick);
      moving:=false;
      ts.DIST:=round(ts.LDIST/ts.CONVERTSR);
      signal(user);
    end;
  end;

procedure STurnLeft(angle:integer);
begin
  if angle > 0 then
    begin
      wait(user);
      Init;
      ts.SINGLELEFTON:=true;
      movelimit:=round(angle * ts.CONVERTSL-1);      { Se Forw.  }
      PutInMotion;
      wait(ready);
      waittime(3*tick);
      moving:=false;
      ts.DIST:=round(ts.RDIST/ts.CONVERTSL);
      signal(user);
    end;
  end;

procedure SetConstants;
var a,b,c,d,e,f,g :real;
begin
  wait(user);
  writeln('WHEELFACTOR, CONVERTF, CONVERTB, CONVERTR, CONVERTL,
          CONVERTSR, CONVERTSL');
  with ts do
    begin
      writeln('Old values:');
      writeln(WHEELFACTOR, CONVERTF, CONVERTB, CONVERTR, CONVERTL,
              CONVERTSR, CONVERTSL);
      writeln('New values ! > ');
      readln(a,b,c,d,e,f,g);
      WHEELFACTOR:=a;
      CONVERTF:=b;
      CONVERTB:=c;
      CONVERTR:=d;
      CONVERTL:=e;
      CONVERTSR:=f;
      CONVERTSL:=g;
    end;
  signal(user);
end;

```

```

procedure Count;                                {Raknar upp vid varje flank}
begin
  if dommy1 and Bitin(0,6) then
  begin
    kr:=kr+1;
    dommy1:=false;
  end;
  if not dommy1 and not Bitin(0,6) then
  begin
    kr:=kr+1;
    dommy1:=true;
  end;
  if dommy2 and Bitin(0,7) then
  begin
    k1:=k1+1;
    dommy2:=false;
  end;
  if not dommy2 and not Bitin(0,7) then
  begin
    k1:=k1+1;
    dommy2:=true;
  end;
  with ts do
  begin
    RDIST:=kr;                                {Olika hjuldiameter korrigeras }
    LDIST:=round(WHEELFACTOR*k1);           {med WHEELFACTOR. }
  end;
end;

procedure Regulate;                             {Reglerar vid skillnad i varv- }
begin                                           {antal. }
  with ts do
  begin
    if RDIST > LDIST then
    begin
      Bitout(0,6,true);
      Bitout(0,7,false);
    end;
    if LDIST > RDIST then
    begin
      Bitout(0,6,false);
      Bitout(0,7,true);
    end;
    if RDIST = LDIST then
    begin
      Bitout(0,6,true);
      Bitout(0,7,true);
    end;
  end;
end;

procedure CheckDistance;                       {Kollar om den kommenderade strackan}
begin                                           {är nadd. }
  with ts do
  begin
    if (RDIST >= movelimit) or
       (LDIST >= movelimit) then
    begin
      if dammi then
      begin
        dammi:=false;
        REACHED:=true;
        SignToBitouts;
        signal(ready);
      end;
    end
    else REACHED:=false;
  end;
end;

```

```

procedure Touchcontrol;
begin
  with ts do
    begin
      wait(mutex);
      if touchenabled then
        begin
          FRONTTOUCHON:=Bitin(0,1);
          RIGHTTOUCHON:=Bitin(0,2);
          BACKTOUCHON:=Bitin(0,3);
          LEFTTOUCHON:=Bitin(0,4);
          if(FRONTTOUCHON or RIGHTTOUCHON or BACKTOUCHON
            or LEFTTOUCHON) and dammi then
            begin
              dammi:=false;
              SignToBitouts;
              signal(ready);
            end;
        end
      else
        begin
          FRONTTOUCHON:=false;
          RIGHTTOUCHON:=false;
          BACKTOUCHON:=false;
          LEFTTOUCHON:=false;
        end;
      signal(mutex);
    end;
end;

{process} procedure Control;
begin
  setpriority(3);
  while true do
    begin
      wait(move);
      while moving do
        begin
          Touchcontrol;
          Count;
          Regulate;
          CheckDistance;
          waittime(1*tick);
        end;
    end;
end;

{process} procedure Touchdisabletimer; {Gor det mojligt att kora en liten }
begin                                     {stracka aven om touchsensorerna ar}
  setpriority(15);                         {intryckta eller vid kontaktstudsar}
  while true do
    begin
      wait(touchtimer);
      waittime(20*tick);
      wait(mutex);
      touchenabled:=true;
      signal(mutex);
    end;
end;

```

```
procedure InitTurtle;
begin
  with ts do
    begin
      FORWARDON:=false;
      RIGHTON :=false;
      BACKON :=false;
      LEFTON :=false;
      FRONTTOUCHON:=false;
      RIGHTTOUCHON:=false;
      BACKTOUCHON:=false;
      LEFTTOUCHON:=false;
      REACHED:=false;
      SINGLERIGHTON:=false;
      SINGLELEFTON:=false;
      RDIST:=0;
      LDIST:=0;
      DIST:=0;
      CONVERTF:=3.24545;
      CONVERTB:=3.24545;
      CONVERTR:=0.37778;
      CONVERTL:=0.37778;
      CONVERTSR:=0.7444;
      CONVERTSL:=0.7444;
      WHEELFACTOR:=1.0055;
    end;
  moving:=false;
  movelimit:=maxint;
  SetOutsFalse;
  SignToBitouts;
  Bitout(0,5,true);
  initsem(mutex,1);
  initsem(touchtimer,1);
  initsem(ready,0);
  initsem(move,0);
  initsem(user,1);
  createprocess(TouchdisableTimer,1000);
  createprocess(Control,2000);
end;
```

(tystar tutan)

5. Tillämpningsexempel.

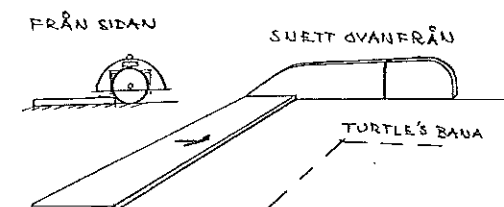
5.1 Inledning.

Examensarbetet avslutades med att vi skrev några enkla tillämpningsprogram. Vi hade då som mål att det vore bra om Turtle kunde justera sin position och riktning, vidare att det skulle vara lätt att styra Turtle från terminal. Slutligen skrev vi ett program för sökning.

5.2 Positionsjustering.

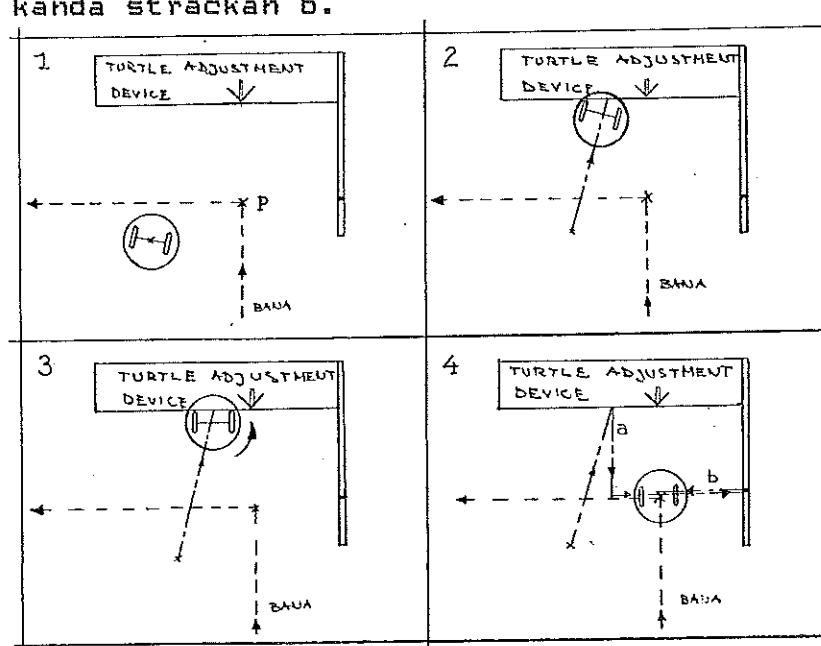
Under sin vandring på golvet tappar Turtle normalt sin position och riktning. De vanligaste orsakerna är slirning och icke-perfekta svängar.

Det behövs alltså någon eller några kända anordningar i rummet som Turtle kan använda sig av om den "vill justera in sig". Vi tillverkade för detta ändamål en Turtle Adjustment Device (se fig) som skall användas tillsammans med proceduren AdjustPosition.



Funktionen framgår av nedanstående bildserie.

1. Turtle har, under sin väg till punkten p, tappat sin position och riktning. I detta läge antar vi att AdjustPosition anropas.
2. Turtle går då in i en rörelsefas, där den först kör fram till Turtle Adjustment Device. Observera att den senare måste vara placerad i ett väl definierat läge i förhållande till den kända punkten p. På anordningen finns markeringar(pilar) som underlättar utplaceringen..
3. När Turtle kommer fram till den första listen slirar den mot denna, och får på så sätt den riktning som den skulle haft i punkten p.
4. Den slutliga inställningen av positionen görs genom att Turtle nu backar en känd sträcka a, vrider 90 grader, kör in i väggen på Turtle Adjustment Device och sedan backar den kända sträckan b.



Programlista AdjustPosition

```

procedure AdjustPosition; {Justerar vinkel och läge m.hj.a}
begin
  Tut(4);
  Forw(50);
  Back(20);
  Turnright(90);
  Forw(100);
  Getstatus;
  if status.FRONTTOUCHON then
  begin
    Back(30);
    Turnleft(90);
  end
  else writeln('I cant find no referencepoint!');
  Tut(4);
end;

```

5.3 Enkel_operatörskommunikation.

För att göra det möjligt att från terminal köra Turtle , samt ändra dess parametrar, skrev vi den procedur som listas nedan.

Vi har lagt denna procedur tillsammans med primitiverna i filen TURTLE.

```

procedure TurtleOpcom;
var com :char;
    dist :integer;
    return:boolean;
begin
  writeln('TURTLE OPCOM');
  writeln('*****');
  writeln('Turtle Operator Comunication System ger dig mojlighet att ');
  writeln('valja mellan foljande kommandon: ');
  writeln('f heltal => Fram          heltal cm ');
  writeln('b          Back          cm ');
  writeln('r          Centrumsvang hogre          grader ');
  writeln('l          Centrumsvang vanster        grader ');
  writeln('h          Enhjulssvang hogre          grader ');
  writeln('v          Enhjulssvang vanster        grader ');
  writeln('t          Heltal korta ljudsignaler   ');
  writeln('c          0          Andra omvandlingskonstanter ');
  writeln('p          0          Aterhopp till programmet ');
  writeln('*****');
  writeln;
  return:=false;
  while not return do
    begin
      writeln('Kommando ! ( f b r l h v t c p )');
      readln(com,dist);
      case com of
        'f': Forw(dist);
        'b': Back(dist);
        'r': Turnright(dist);
        'l': Turnleft(dist);
        'h': STurnright(dist);
        'v': STurnleft(dist);
        'c': SetConstants;
        'p': return:=true;
        't': Tut(dist);
      end;
    end;
  end;
end;

```

5.4 Bana med okända föremål.

Problemet som ska lösas är följande:

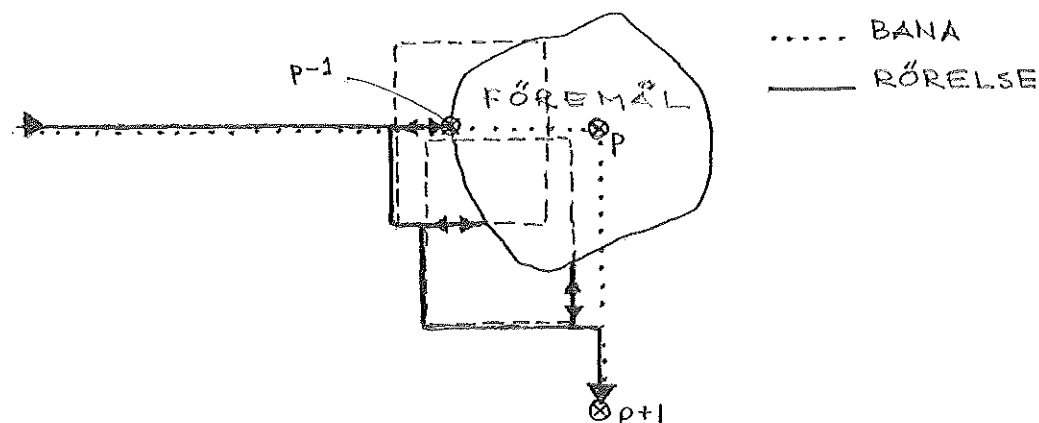
Turtle ska i tur och ordning kunna gå mellan ett antal punkter i rummet. I Turtle's väg kan ett okänt antal föremål finnas. Vi begränsade problemet genom att låta banan vara rätvinklig, samt att ett föremål får täcka högst en av de givna punkterna. Föremålen måste dessutom vara fristående från väggar. Däremot finns ingen begränsning på antalet föremål eller deras utseende.

I programmet görs följande:

En bana lagras in från terminal. Några punkter kan vara justeringspunkter (se 5.3). Banan läses in i en matris med tre kolonner (x-koord, y-koord, siffra som anger om denna punkt är

Turtle försöker gå succesivt från punkt till punkt. Antag att Turtle är på väg till punkten p när den stöter i ett föremål. Då görs följande:

1. Kollisionspunkten lagras. Låt oss kalla denna punkt $p-1$.
2. Om punkten $p+1$ ligger till vänster, väljes sökning åt vänster; annars högersökning.
3. Turtle börjar söka sig runt föremålet och letar samtidigt efter banan enligt följande princip:
 - a. Turtle försöker gå i en fyrkant. Om den stöter i något, upprepar den försöket.



- b. Före varje förflyttning kontrolleras om Turtle under förflyttningen kommer att passera någon av banlinjerna $p-1 - p$ eller $p - p+1$. Om så är fallet kör Turtle fram till banan och fortsätter som förut.

Program_med_kommentarer.

(T8 FollowTrack)

```

var pek: {Pekar pa nasta rad i coor }
endp; {pekar pa sista raden i coor}
xnext: {X-koord. for nasta punkt }
ynext: {Y-koord. for nasta punkt }
x,y,fi :integer; {Aktuell koord. o. riktning }
coor :array[0..50,1..3] of integer; {Har lagras bana + ref.instr}
linefound :boolean; {Sann nar bana hittats }
d :char; {l/r,Hoger-/Vanster-sokning }

```

```
function Bitin(chan,bitnr:integer):boolean;external;
```

```
procedure Forw(length:integer);external;
```

```
procedure Back(length:integer);external;
```

```
procedure Turnright(angle:integer);external;
```

```
procedure Turnleft(angle:integer);external;
```

```
procedure STurnRight(angle:integer);external;
```

```
procedure STurnLeft(angle:integer);external;
```

```
procedure Getstatus;external;
```

```
procedure SetConstants;external;
```

```
procedure InitTurtle;external;
```

```
procedure Tut(number:integer);external;
```

```

procedure Run; {Kor Turtle till xnext,ynext el- }
var dx; {ler till fronttouch fas. }
dy :integer; {Uppdaterar x,y. }
begin

```

```

  dx:=xnext-x;
  dy:=ynext-y;
  status.Fronttouchon:=false;
  if dx = 0 then
    begin
      if dy > 0 then
        begin
          if fi = 0 then Turnleft(90)
          else if fi = 180 then Turnright(90)
          else if fi = 270 then Turnright(180);
          fi:=90;
          Forw(dy);
          Getstatus;
          if status.FRONTTOUCHON then y:=y+status.DIST else y:=ynext;
        end
      else if dy < 0 then
        begin
          if fi = 0 then Turnright(90)
          else if fi = 180 then Turnleft(90)
          else if fi = 90 then Turnright(180);
          fi:=270;
          Forw(-dy);
          Getstatus;
          if status.FRONTTOUCHON then y:=y-status.DIST else y:=ynext;
        end;
      end
    else if dy = 0 then
      begin
        if dx > 0 then
          begin
            if fi =90 then Turnright(90)
            else if fi = 270 then Turnleft(90)
            else if fi = 180 then Turnright(180);
            fi:=0;
            Forw(dx);
            Getstatus;
            if status.FRONTTOUCHON then x:=x+status.DIST else x:=xnext;
          end
        else if dx < 0 then
          begin
            if fi = 90 then Turnleft(90)
            else if fi = 270 then Turnright(90)
            else if fi = 0 then Turnright(180);
            fi:=180;
            Forw(-dx);
            Getstatus;
            if status.FRONTTOUCHON then x:=x-status.DIST else x:=xnext;
          end;
        end;
      end;
    end;
  end;
end;

```

```

procedure Avoid; {Anropas av Avoidfirsttime samt }
                 {rekursivt av Check och Avoid. }
                 { Undviker ett }
                 {foremal genom att forsoka ga }
var i,n,p,r,s,t,xsave,ysave :integer; {runt det med hoger eller vans- }
                 {ter-sokning.Fore varje forflyt-}
                 Add :array[0..50,1..2] of integer; {tning anropas Check. }

procedure Check; {Kontrollerar fran coor[pek-1,1]}
                 {till coor[pek+1,1] om en bana }
var k,direktion :integer; {skar linjen x,y --> xnext,ynext}

procedure DecideDirektion(var direktion:integer); {Bestammer vinkeln for}
var dx,dy :integer; {linjen x,y --> xnext,ynext}
begin
  dx:=xnext-x;
  dy:=ynext-y;
  if (dx>0) then direktion:=0;
  if (dx<0) then direktion:=180;
  if (dy>0) then direktion:=90;
  if (dy<0) then direktion:=270;
  if (dx=0) and (dy=0) then writeln('DecideDirektion anropad utan bana');
end;

procedure GoToLine; {Forsoker kora Turtle till }
begin {en av Check hittad linje. }
  RUN;
  if status.FRONTTOUCHON then Avoid
  else
    begin
      pek:=k+1;
      linefound:=true;
      Tut(2);
      writeln('Turtle now on Track again!');
    end;
end;

begin {Check}
  DecideDirektion(direktion);
  k:=pek-1; if (k<0) then k:=0;
  while (k < pek+1) and (k <(endp-1)) do
    begin
      case direktion of
        0:
          if (x<coor[k,1]) and (xnext>=coor[k,1]) and
            ( ((y<=coor[k,2]) and (y>=coor[k+1,2])) or
              ((y>=coor[k,2]) and (y<=coor[k+1,2])) )
            then
              begin
                xnext:=coor[k,1];
                ynext:=y;
                GoToLine;
              end;
          90:
          if (y<coor[k,2]) and (ynext>=coor[k,2]) and
            ( ( (x<=coor[k,1]) and (x>=coor[k+1,1]) ) or
              ( (x>=coor[k,1]) and (x<=coor[k+1,1]) ) )
            then
              begin
                xnext:=x;
                ynext:=coor[k,2];
                GoToLine;
              end;
          180:
          if (x>coor[k,1]) and (xnext<=coor[k,1]) and
            ( ( (y<=coor[k,2]) and (y>=coor[k+1,2]) ) or
              ( (y>=coor[k,2]) and (y<=coor[k+1,2]) ) )
            then
              begin
                xnext:=coor[k,1];
                ynext:=y;
                GoToLine;
              end;
          270:
          if (y>coor[k,2]) and (ynext<=coor[k,2]) and
            ( ( (x<=coor[k,1]) and (x>=coor[k+1,1]) ) or
              ( (x>=coor[k,1]) and (x<=coor[k+1,1]) ) )
            then
              begin
                xnext:=x;
                ynext:=coor[k,2];
                GoToLine;
              end;
            end;
          k:=k+1;
        end;
      end;
    end;
  end;
end;

```

```

begin (Avoid)
  for i:=0 to 50 do
    for r:=1 to 2 do Add[i,r]:=0;

  r:=40;
  p:=50;
  s:=80;
  t:=40;

  Add[0,1]:=0;   Add[0,2]:=-r;      {Innehaller strackor somm skall lag-}
  Add[1,1]:=p;   Add[1,2]:=0;      {gas till x,y for att fa xnext,ynext}
  Add[2,1]:=0;   Add[2,2]:=s;      {nar Turtle undviker ett foremal. }
  Add[3,1]:=-p;  Add[3,2]:=0;
  Add[4,1]:=0;   Add[4,2]:=-t;

  Add[5,1]:=r;   Add[5,2]:=0;
  Add[6,1]:=0;   Add[6,2]:=p;
  Add[7,1]:=-s;  Add[7,2]:=0;
  Add[8,1]:=0;   Add[8,2]:=-p;
  Add[9,1]:=t;   Add[9,2]:=0;

  Add[10,1]:=0;  Add[10,2]:=r;
  Add[11,1]:=-p; Add[11,2]:=0;
  Add[12,1]:=0;  Add[12,2]:=-s;
  Add[13,1]:=p;  Add[13,2]:=0;
  Add[14,1]:=0;  Add[14,2]:=t;

  Add[15,1]:=-r; Add[15,2]:=0;
  Add[16,1]:=0;  Add[16,2]:=-p;
  Add[17,1]:=s;  Add[17,2]:=0;
  Add[18,1]:=0;  Add[18,2]:=p;
  Add[19,1]:=t;  Add[19,2]:=0;

  Add[20,1]:=0;  Add[20,2]:=r;
  Add[21,1]:=p;  Add[21,2]:=0;
  Add[22,1]:=0;  Add[22,2]:=-s;
  Add[23,1]:=-p; Add[23,2]:=0;
  Add[24,1]:=0;  Add[24,2]:=t;

  Add[25,1]:=-r; Add[25,2]:=0;
  Add[26,1]:=0;  Add[26,2]:=p;
  Add[27,1]:=s;  Add[27,2]:=0;
  Add[28,1]:=0;  Add[28,2]:=-p;
  Add[29,1]:=-t; Add[29,2]:=0;

  Add[30,1]:=0;  Add[30,2]:=-r;
  Add[31,1]:=-p; Add[31,2]:=0;
  Add[32,1]:=0;  Add[32,2]:=s;
  Add[33,1]:=p;  Add[33,2]:=0;
  Add[34,1]:=0;  Add[34,2]:=-t;

  Add[35,1]:=r;  Add[35,2]:=0;
  Add[36,1]:=0;  Add[36,2]:=-p;
  Add[37,1]:=-s; Add[37,2]:=0;
  Add[38,1]:=0;  Add[38,2]:=p;
  Add[39,1]:=t;  Add[39,2]:=0;

  linefound:=false;
  Back(15);
  case fi of
    0: begin n:=0; x:=x-15; end;
    90: begin n:=5; y:=y-15; end;
    180: begin n:=10; x:=x+15; end;
    270: begin n:=15; y:=y+15; end;
  end;
  if d='1' then n:=n+20;
  for i:=n to n+4 do
    if not linefound then
      begin
        xnext:=x+Add[i,1];
        ynext:=y+Add[i,2];
        Check;
        if not linefound then
          begin
            Run;
            if status.FRONTTOUCHON then
              begin
                Avoid;
                if not linefound then
                  begin
                    writeln('FEL? Turtle har nu gatt ett HELT varv!');
                    Tut(6);
                  end;
                end;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;

```



```

procedure AvoidFirst;
var peksave,
    savex,savey :integer;
begin
    peksave:=pek-1;
    savex:=coor[peksave,1];
    savey:=coor[peksave,2];
    coor[peksave,1]:=x;
    coor[peksave,2]:=y;
    AVOID;
    coor[peksave,1]:=savex;
    coor[peksave,2]:=savey;
end;

procedure ChooseDirection;
begin
    {Valjer hoger- eller vanster- }
    {sokning. }
    if fi = 0 then
        if (coor[pek,2]-coor[pek+1,2]) > 0 then d:='r' else d:='l'
    else if fi = 90 then
        if (coor[pek,1]-coor[pek+1,1]) > 0 then d:='l' else d:='r'
    else if fi = 180 then
        if (coor[pek,2]-coor[pek+1,2]) > 0 then d:='l' else d:='r'
    else if fi = 270 then
        if (coor[pek,1]-coor[pek+1,1]) > 0 then d:='r' else d:='l';
end;

procedure AdjustPosition;
begin
    {Justerar vinkel och lage m.hj.a}
    {Turtle Adjust Device. }
    Tut(4);
    Forw(50);
    Back(20);
    Turnright(90);
    Forw(100);
    Getstatus;
    if status.FRONTTOUCHON then
        begin
            Back(30);
            Turnleft(90);
        end
    else writeln('I cant find no referencepoint!');
    Tut(4);
end;

procedure FollowTrack;
var ref,
    i,k :integer;
begin
    {Kor till punkterna i matrisen coor }
    {och undviker foremal. }
    fi:=90;
    x:=coor[0,1];y:=coor[0,2];
    pek:=1;
    while pek <= endp do
        begin
            xnext:=coor[pek,1];
            ynext:=coor[pek,2];
            ref:=coor[pek,3];
            Run;
            Getstatus;
            if status.FRONTTOUCHON then
                begin
                    Tut(3);
                    writeln('I*v had a Touch ,man!');
                    ChooseDirection;
                    AvoidFirst;
                end
            else
                begin
                    if ref = 1 then AdjustPosition;
                    pek:=pek+1;
                end;
            end;
            Tut(5);
        end;
end;

```

```

procedure LoadTrackandRuni      (Laser in punkter i matrisen coor och
var q      :char;              {anropar FollowTrack.
    datain :boolean;
    i,k;
    p      :integer;
begin
  writeln('New Track ?');
  readln(q);
  if q = 'y' then
    begin
      writeln('Coordinates x,y,r      (-1,-1,0 => endpoint)
              (x,y,1 => referencepoint)');
      for i:=0 to 50 do for k:=1 to 3 do coor[i,k]:=-1;
      datain:=true;
    end
  else if q = 'n' then datain:=false;
  p:=0;
  while datain do
    begin
      readln(coor[p,1],coor[p,2],coor[p,3]);
      if (coor[p,1] = -1) and (coor[p,2] = -1) then
        begin
          endp:=p;
          datain:=false;
        end
      else p:=p+1;
    end;
  Tut(1);
  FollowTrack;
end;

```

```

{process} procedure Opcom;      {Valjer terminal- eller program-
var com      :char;           {korning av Turtle.
    d      :integer;
begin
  setpriority(4);
  writeln('TURTLE OPCOM');
  writeln('*****');
  writeln('Turtle Operator Communication System ger dig mojlighet att ');
  writeln('valja mellan foljande kommandon: ');
  writeln;
  writeln('f  heltal => Fram          heltal cm ');
  writeln('b          Back          cm ');
  writeln('r          Centrumsvang hoger   grader ');
  writeln('l          Centrumsvang vanster  grader ');
  writeln('h          Enhjulssvang hoger    grader ');
  writeln('v          Enhjulssvang vanster  grader ');
  writeln('t          Heltal korta ljudsignaler ');
  writeln('c          0          Andra omvandlingskonstanter ');
  writeln('p          0          Anropa LoadTrackandRun ');
  writeln('*****');
  writeln;
  while true do
    begin
      writeln('Kommando ! ( f b r l h v t p c )');
      readln(com,d);
      case com of
        'f': Forw(d);
        'b': Back(d);
        'r': Turnright(d);
        'l': Turnleft(d);
        'h': STurnright(d);
        'v': STurnleft(d);
        'c': SetConstants;
        'p': LoadTrackandRun;
        't': Tut(d);
      end;
    end;
end;

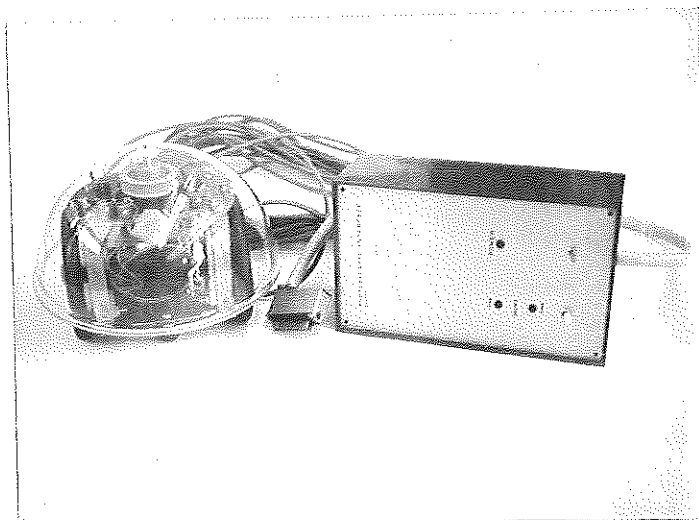
```

```

{main program}
begin
  initkernel(2000);
  initio;
  InitTurtle;
  createprocess(Opcom,2000 );
  setpriority(maxpriority);
end.

```

6. Bruksanvisning.



6.1

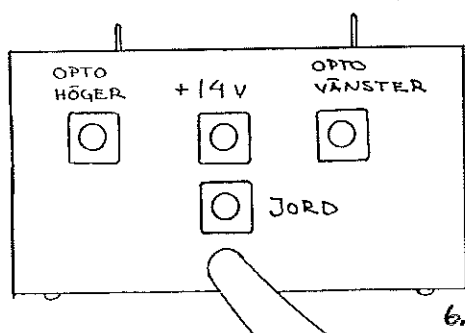
Interface.

Anslut interface till 14 Volt (1.5 A) och jord.
Hönskontakter finns på baksidan
av interface (fig 6.2).

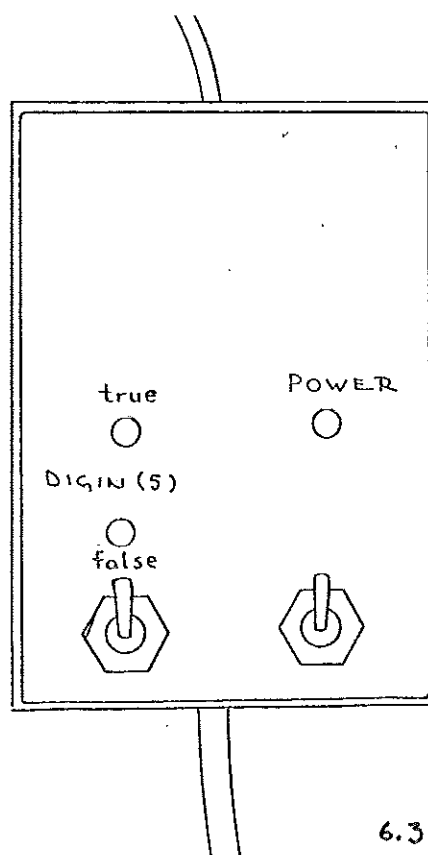
Som bekräftelse på att
spänning är ansluten, lyser
någon av DIGIN 5 indika-
torerna. (se fig. 6.3).

Spänning till Turtle fås genom
att sätta POWER-strömbrytaren
i läge on.

Turtle's lysdioder (ögon)
skall nu lysa och Turtle ger
ifrån sig ett förtvivlat tjut,
som varar tills programmet
(primitiverna) börjar
exekveras (Initturtle
anropas).



6.2



6.3

Mjukvara.

Ett program som använder Turtle kan innehålla följande externdeklarerade procedurer:

```

procedure Forw(length){external};
procedure Back(length){external};
procedure Turnright(angle){external};
procedure Turnleft(angle){external};
procedure STurnright(angle){external};
procedure STurnleft(angle){external};
procedure Tut(number){external};
procedure Getstatus{external};
procedure Setconstants{external};
procedure InitTurtle{external};
procedure TurOpc{external};

```

(Beskrivning av dessa finns i avsnitt 4.2.)

Huvudprogrammet kan ha följande utseende, om vi antar att användaren vill skapa en egen process Proc.

```

begin
  initkernel(1000);
  initio;
  InitTurtle;
  createprocess(Proc,2000);
  setpriority(maxpriority);
end.

```

Följande instruktioner används vid kompilering och länkning. De nödvändiga programmen PRETUR.VAR och TURTLE finns på en skiva med namnet TURTLE.

Kompilering:

```

R PASCAL
*PROG,TT:=PREDAT,DX1:PRETUR.VAR,DX1:PROG

```

Länkning:

```

LINK/EXE:DX1: PROG,DX1:TURTLE,KERNEL,PASLIB

```

Filen TURTLE innehåller primitiver + en enkel operatörs-kommunikation TurOpc. (För beskrivning av TurOpc se kap 5.2)

Filen PRETUR.VAR innehåller globala variabler och skall kompileras tillsammans med användarens program. Den skall också ingå om man vill ändra i primitiverna

Kompileringskommandot blir då:

```
R PASCAL
```

```
*TURTLE,TT:=PREDAT,DX1:PRETUR.VAR,DX1:TURTLE/E
```

```
MACRO/OBJ:DX1: TURTLE
```

Efter detta finns en ny objektfil TURTLE.OBJ att användas vid länknigen av huvudprogrammet.

Att tänka på.

Turtle går lite krokigt och svänger något mer eller något mindre än vad man ger order om. Detta har olika orsaker, där den största är slirning mot underlaget. Tvätta därför golvet och Turtle's hjul noga.

7. Slutsatser om rörelserna samt förslag till förbättringar.

Hur väl fungerar de kommenderade rörelserna i praktiken ?

Körning rakt fram.

Turtle är känslig för slirning, dvs. golvet måste vara rent. Parametern Wheelfactor, som tar hänsyn till hjulens olika diametrar, måste ibland ändras. Eventuellt finns ett samband mellan Wheelfactor och friktion mot golvet.

Sammanfattningsvis kan vi säga att Turtle fungerar tillfredsställande vid körning rakt fram.

Centrumsvängar.

För varje motorvarv räknas en puls och under denna tid hinner hjulen röra sig 3 mm. Detta motsvarar en vridning på ca. 3 grader. Hur stor blir osäkerheten om vi räknar en puls per motorvarv?

I starten får vi en osäkerhet på 1 puls. Nedbromsnings-trögheten hos Turtle är i medeltal 1.75 pulser. Vi antar att osäkerheten här är 1 puls. Dessutom är det rimligt att anta att slirningen bidrar med en osäkerhet på 1 puls.

Vi är nu uppe i en osäkerhet på 3 pulser. Detta ger en maximal osäkerhet av 9 grader vid en centrumsväng. Detta är liktydigt med att om Turtle svänger ett antal pulser som motsvarar ett väntevärde på tex. 90 grader och därefter kör en meter rakt fram, kommer den i värsta fall att befinna sig 8 cm från den punkt där den borde vara.

Resonemanget ovan har visat sig vara rimligt i praktiken.

Ett annat problem om man vill göra en 90 graders-sväng är att det i praktiken inte finns något antal pulser som motsvarar en sväng med väntevärde 90 grader. Detta medför ytterligare några graders fel.

Hur kan centrumsvängarna förbättras?

En lösning är att fästa fler reflektorer på motoraxeln samt göra det möjligt att sänka Turtles hastighet kraftigt vid svängarna. Hastigheten måste sänkas så att datorn hinner räkna pulserna, samt så att slirning och nedbromsnings-osäkerhet minimeras.

Kommer detta att räcka?

En puls motsvarar nu ca. $3/4$ grader eller $3/4$ mm hjulförflyttning.

Antag nedbromsningsosäkerhet och osäkerhet p.g.a slirning vid långsam sväng vardera bidrar med 1 puls.

Detta ger totalt 3 pulser eller 2.25 grader i osäkerhet, vilket gör att Turtle kan komma 2 cm fel efter att ha svängt en sväng med väntevärdet tex. 90 grader och därefter kört en meter rakt fram.

Enhjulssväng.

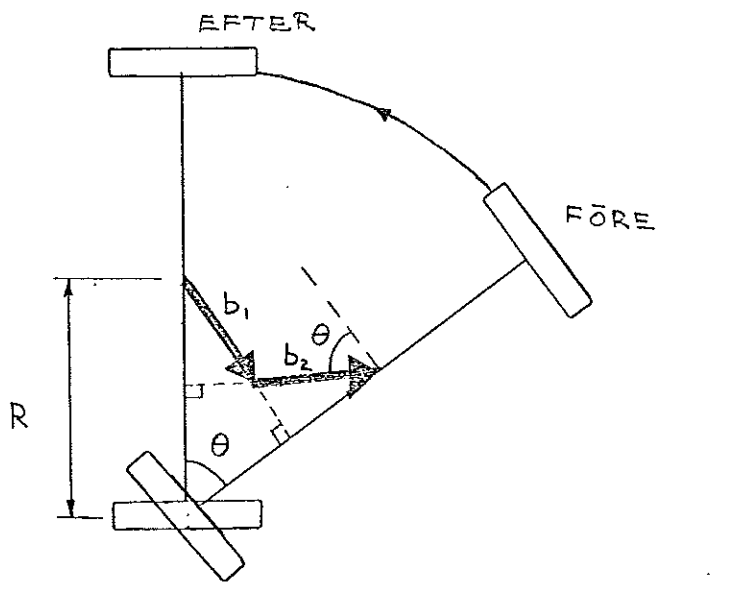
Problemen här är av samma natur som vid centrumsväng. De är emellertid mindre eftersom:

1. Ungefär dubbelt så många pulser räknas, vilket innebär att 1 puls motsvarar ungefär 1.5 grader (i stället för 3 grader).
2. Svängen sker långsammare. Det betyder att osäkerheten p.g.a. slirning och nedbromsningströghet minskar.
3. Ett visst antal pulser motsvarar en sväng med väntevärde mycket nära 90 grader, vilket förbättrar 90 graders-svängar.

Appendix 1

Centrumkorrigering vid enhjulssvängar.

Vid enhjulssväng kommer centrums position att förflyttas. Detta kan man ta hänsyn till genom att förkorta sträckorna före och efter svängen.



θ = kommenderad vridning
 R = halva axellängden
 b_1 = korrigering 1
 b_2 = korrigering 2

$$b_2 = \frac{R - R \cos \theta}{\sin \theta} = \frac{R(1 - \cos \theta)}{\sin \theta}$$

$$b_1 = R \sin \theta - \frac{R - R \cos \theta}{\sin \theta} \cos \theta = \frac{R(1 - \cos \theta)}{\sin \theta}$$

$$\therefore b_1 = b_2 = \frac{\text{axellängd}}{2} \cdot \frac{1 - \cos \theta}{\sin \theta}$$

Appendix 2.

Omvandling mellan grader och antal pulser vid svängar.

För att få en så god omvandlingsfaktor som möjligt gjordes en försöksserie på golvet. Ett visst antal pulser kommanderades och sedan mättes det antal grader (θ) som Turtle vridit sig.

Diagram 1. visar Centrumsväng.

Diagram 2. visar Enhjulssväng.

Det framgår att kurvan är av typen :

$$\theta = k \cdot (\text{antal pulser} + C)$$

$$\text{antal pulser} = \theta/k - C$$

Diagrammen visar att $C \approx 2$ pulser för centrumsvängar och $C \approx 1$ puls för enhjulssvängar.

Den praktiska betydelsen av C är att om man utan korrek- tioner kommenderar Turtle att gå n pulser, så går hon i själva verket $n + C$ pulser. Detta beror på att Turtle behöver C st pulser på sig för att stanna efter det att den kommenderade sträckan blev nådd.

DIAGRAM 1.

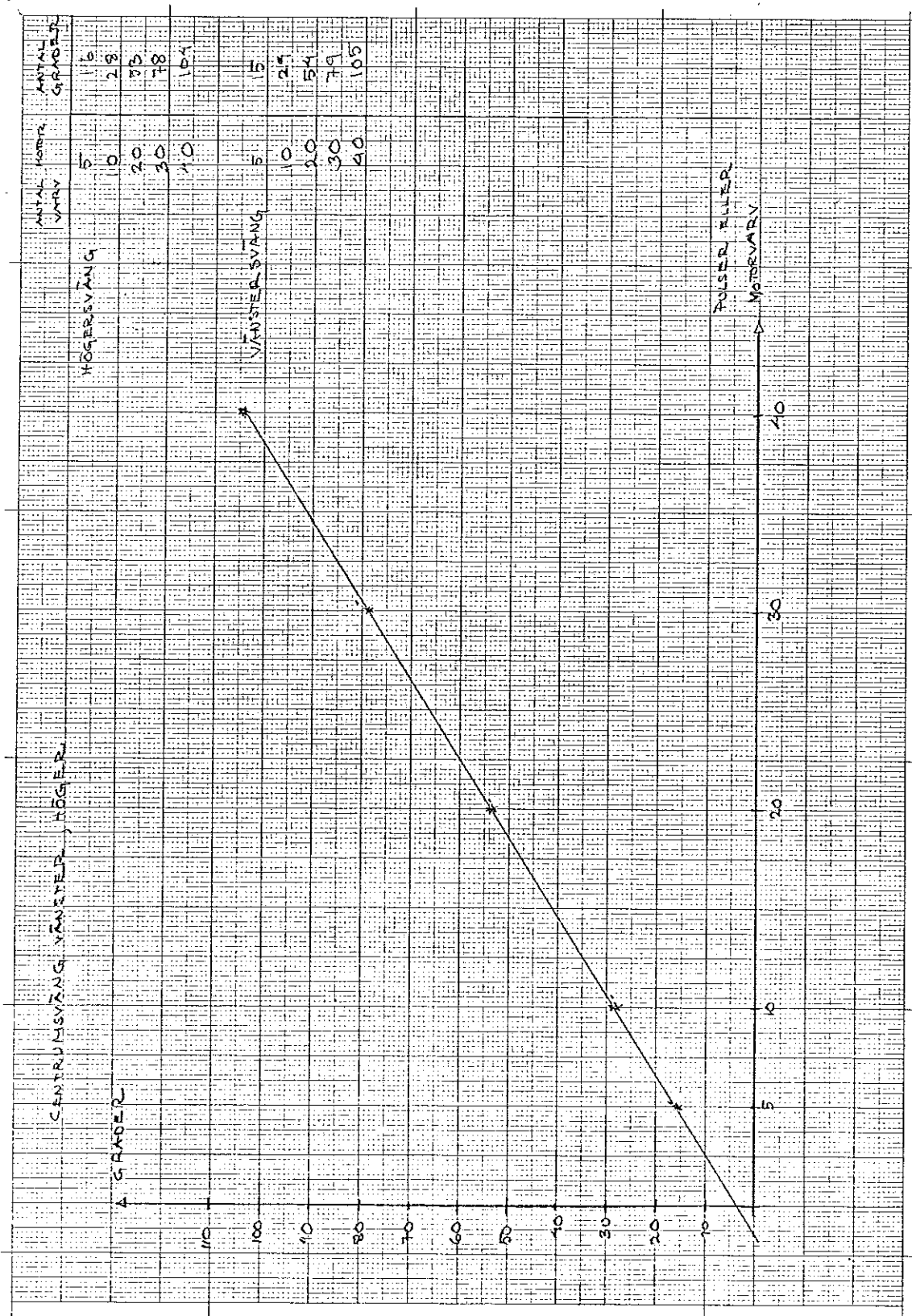
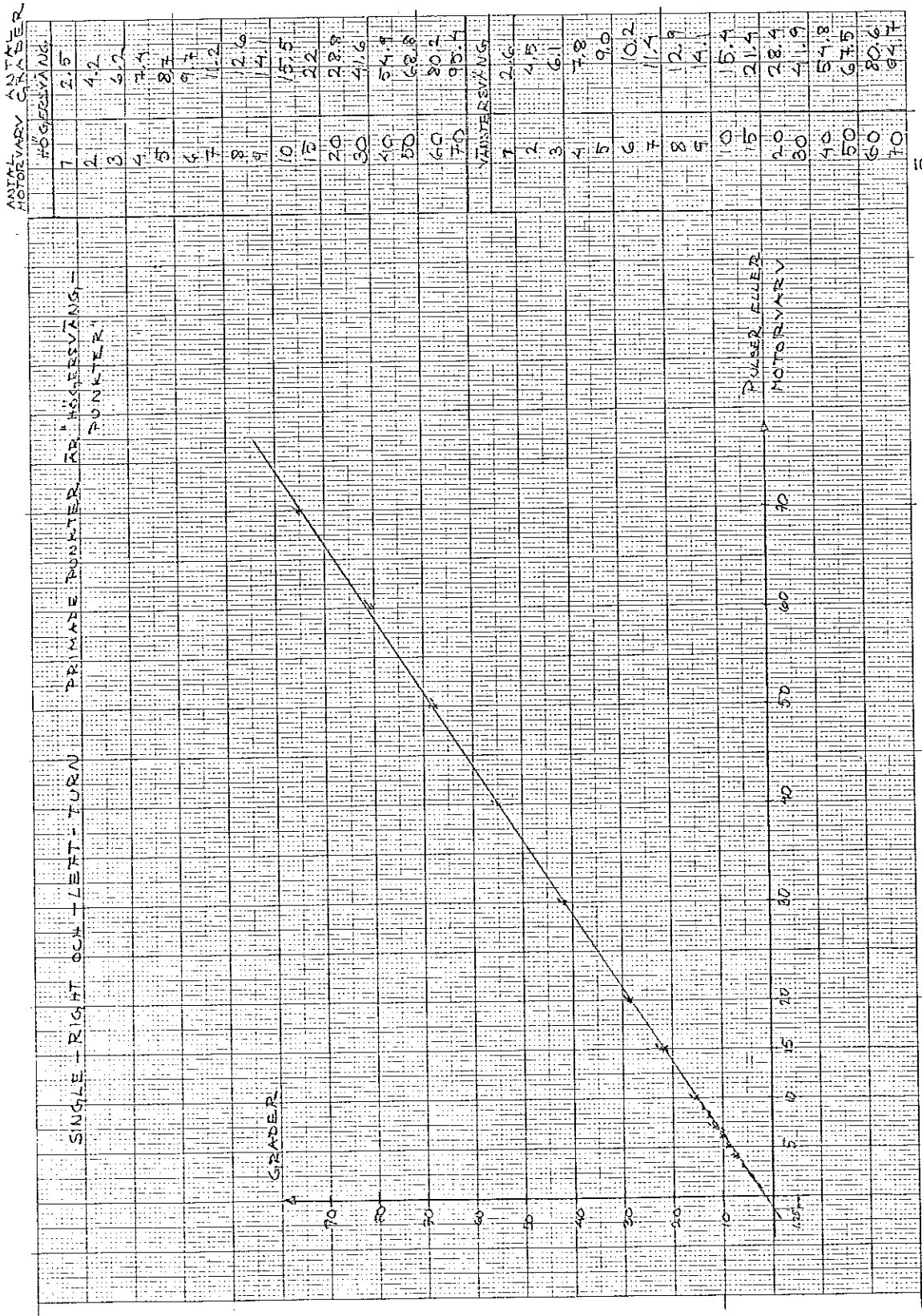


DIAGRAM 2.



10

EDUCATIONAL APPLICATIONS OF THE TERRAPINtm TURTLE

The Terrapin Turtle is a powerful learning tool. Educators use it from elementary to graduate levels, to teach everything from geometry to programming to artificial intelligence. A visible, mobile extension of the computer, the Turtle puts a friendly face on the computer, making it less of a black box. Most importantly, Turtles are fun.

TURTLE GEOMETRY AND MATHEMATICS - Programming Turtle movement gives rise to a whole new style of mathematics dubbed Turtle Geometry. In a book by that title, Harold Abelson and Andrea diSessa of MIT use Turtle movement to explore geometry, vectors, number theory, topology, and other mathematical concepts. The authors found that students perform better by "debugging" problems themselves, rather than simply learning the right answer by rote. The approach allows for active participation in mathematical discovery and exploration.

Seymour Papert's book Mindstorms discusses Turtles and the LOGO language developed at MIT. Papert worked with the famous child psychologist Jean Piaget for a number of years. Upon coming to MIT, Papert began developing a language (LOGO) that allowed children to interact with computers without having to understand programming. Since young children have trouble with abstract concepts, the Turtle was developed as a concrete, visible, three dimensional extension of the computer. The Turtle helps children quickly master the concepts of distance, number line, and angle, since they can relate the Turtle's movements to their own. Advanced children can explore more complex ideas: How do you make the Turtle draw geometric figures? How do you scale up a polygon? Do you change the sides or the angles? How do you draw a circle? a parabola? an ellipse?

Simple Turtle demonstrations explain base two. For example, a single touch sensor is either on or off, represented by 1 or 0. An interesting demonstration prints the state of all four touch sensors as a single binary number or the equivalent in base ten. When you push against the dome, affecting the touch sensor state, the number changes. A similar exercise examines signals sent along the eight control wires between the computer and the Turtle. These exercises also introduce the concepts of state and control.

SPECIAL NEEDS - Educators use Turtles with handicapped people. In one study of students with cerebral palsy, a program allowed "manual" control of the Turtle and recorded the students' motor development. Aiming for a particular goal, the students used four keys or groups of keys to direct the Turtle forward, right, back, or left, while the computer recorded their choices. The students were very excited about the project because they were able to control the Turtle, making it move and draw, if even in a crude fashion. Some even learned how to program. (See Dr. E. Paul Goldenberg's book, Special Technology for Special Children) Work with autistic children using the Turtle has been very promising. Most autistic children relate better to mechanical objects than to the social world. The children seem not to fear failure with inanimate objects. Playing with the Turtle allows a bridge of common experiences between the child and teachers or parents, which they can use to break the communication barrier. Amazingly, the children will often start verbalizing spontaneously about the Turtle's movements.

LIFE SCIENCES - Many people consider the Turtle a pet and, in fact, Turtles can model animal behavior. For instance, a Turtle can simulate the random movements of an ant, or perhaps a turtle, which cannot see where it is going. By placing constraints on a random number generator to control the size and direction of the Turtle's turns, a user can explore the effects on the Turtle's forward movement. Does the Turtle go straight or in circles and why? This experiment is also an illustrative example in the study of probability and statistics. (See Abelson and diSessa, Turtle Geometry) Like a real animal, the Turtle gets touch feedback from the world around it. The dome allows a Turtle to determine contact in any of eight directions. Patrick Winston in Artificial Intelligence describes how to use the Turtle's sense of touch to avoid obstacles on a migratory route across a room. This same logic serves as the basis for a program to solve mazes.

SOFTWARE TECHNOLOGY - Turtles have been used in teaching all levels of programming to people of all ages. Papert's Mindstorms demonstrates how even very young children have been introduced to programming and mathematical concepts by using the Turtle. The visible, non-abstract nature of Turtles is useful not only with little children, but in introducing anyone to new concepts. For instance, to teach the importance of correct syntax in a beginning programming course, first show students how to use the Turtle interactively or write a simple program to instruct the Turtle to move or blink or beep. Turtles also demonstrate basic flow control principles. The touch sensors provide an excellent means of explaining conditional statements. For example:

```
IF FRONTTOUCH THEN RIGHTTURN ELSE FORWARD 1
```

The Turtle will make a right turn if something is touching its front touch sensor. If not, the Turtle will move forward 1 unit and stop. Repeating this same command inside a loop instructs the Turtle to move forward until it touches something, then turn right and continue moving forward. More complex branching using AND, OR, and NOT naturally leads to a discussion of Boolean logic. Throughout all, students learn the concept of debugging. The visible nature of the Turtle makes programming mistakes obvious. In more advanced projects, students can recreate or add to the Turtle-computer software interface using assembly language or a high level language such as Basic. Studying how the computer controls the Turtle illustrates how software and hardware interact in more complex systems.

HARDWARE TECHNOLOGY - Terrapin's Instruction and Assembly Manual helps you and your students understand how electronic components inside the Turtle work together. Components of the assembled Turtle are visible under the clear plastic dome. The Turtle is designed to be easy to understand. It contains components and circuits common in computer hardware and robotics. The touch sensor circuits contain a zener regulated voltage source. The motor circuitry uses common emitter - common collector voltage translation switching circuitry in bridge configuration. The light and horn circuits include LED's, an integrated circuit timer and a speaker.

SUMMARY - Turtles are versatile, easy-to-use, and most importantly, Turtles are fun. They allow exploration of mathematics, programming and electronics. Although superb for beginners, the Turtle's versatility makes it challenging for advanced users as well. Turtles are only limited by your imagination.

Terrapin, Inc.

33 Edinborough Street, Sixth floor
 Boston, Massachusetts 02111
 617-482-1033

HOW TO USE BASIC TO CONTROL YOUR TURTLE

Except for die-hard assembly language fanatics, it is quicker and more fun to write programs in a higher level language than to actually play with the bits on an assembly language level. The following discussion should give you an understanding of how to interface a Turtle to BASIC. When you are through, you will understand how to write subroutines to control the Turtle, leaving you free to program in BASIC.

The TerrapinTM Turtle has four binary outputs which transmit the state of its touch sensors and eight binary inputs which control its pen, horn, lights and motors. First we will deal with reading the touch sensor outputs with a BASIC program. Each of the four binary touch sensor outputs comes from one of the four touch sensor switches activated by the Turtle's dome. When the Turtle dome is not in contact with any object, the output of each touch sensor is (a TTL high level corresponding to) a binary "1." If the dome is contacting something and one or two of the touch sensors switches is depressed, the output of each activated switch is (a TTL low level corresponding to) a binary "0." If the four wires from the touch sensor switches are connected to the four least significant bits of a parallel input port on a computer they form a four bit binary number with decimal values between 0 and 15. Assume the touch sensor lines are connected in the following manner to the input port:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	back	front	right	left
				touch	touch	touch	touch

and that bits 4-7 are connected to TTL low levels and therefore have binary 0's on them. Since the touch sensor switches are at the front and back, and on the sides of the Turtle, there are nine possible combinations of touch sensor outputs (8 octants and no touch at all). This is because at most two contiguous switches may be on at once. These possibilities and their outputs in binary and decimal are shown below:

Touch sensor activated	Binary number	Decimal equivalent
Nothing	00001111	15
Front only	00001011	11
Front & right	00001010	10
Right only	00001101	13
Right & back	00000101	5
Back only	00000111	7
Back & left	00000110	6
Left only	00001110	14
Left & front	00001001	9

With most microcomputer BASIC interpreters, it is possible to read the binary data at a parallel input port in some way. TRS-80 level II BASIC does this in a typical way with its "IN" function. This function returns the decimal value of the binary number present at the input port specified in its argument. Thus, if the Turtle's touch sensor lines were connected to parallel port number 30 as specified above, it would be possible to print the decimal equivalent of the Turtle's touch sensor output with the following simple program:

```
10 REM THIS PROGRAM PRINTS TOUCH SENSOR OUTPUT
20 PRINT IN (30)
30 END
```

Note that the IN function takes as an argument the parallel input port(s) to be read from and returns as its output the data present there. If we ran the program while the front touch sensor was being depressed the output would look as follows:

```
RUN (typed by user)
11 (typed by computer)
READY (typed by computer)
```

Of course, the output of the IN function can be used to change the flow of control in a program with the use of if - then (conditional) statements. One simple example of this is shown below:

```
100 IF IN (30) = 11 THEN GO TO 200
110 PRINT "FRONT TOUCH SENSOR NOT DEPRESSED"
120 GO TO 210

200 PRINT "FRONT TOUCH SENSOR DEPRESSED"
210 END
```

Controlling the Turtle's state with BASIC is somewhat harder than reading its touch sensor outputs. Controlling the pen, horn and eyes is fairly simple as they are set either on or off; however, controlling the distance the Turtle moves and the angle it turns is more complicated. This is because the distance each wheel moves is a function of what direction the motor is turning and how long the motor is turned on. Even this is not too much of a problem for a BASIC program once Turtle input control is determined.

Assume the Turtle's eight input lines are connected to the eight output lines of a parallel output port in the following manner:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Tone	Horn	Pen	Lights	LM2	LM1	RM2	RM1
Control							

Where RM1 = Right Motor 1, LM1 = Left Motor 1, etc.

When arranged in this fashion, the Turtle's eight control bits form an 8 bit binary number with possible decimal values from 0 to 255. Some possible Turtle states and their corresponding control bit states are shown below:

<u>Turtle State</u>	<u>Binary Control Data</u>	<u>Decimal Equivalent</u>
Horn, pen, lights off Turtle forward	00000101	5
Horn on, high tone, pen, lights, and motors off	11000000	192
Lights on, pen on, horn off, motors backward	00111010	58
Horn on, low tone, lights and pen off, right turn	01001001	73

The parallel output port connected to the Turtle can be controlled by a micro-computer BASIC interpreter in much the same way that data is read from the Turtle's parallel input port. Once again, TRS-80 Level II basic provides a typical example of how this is done with its OUT statement. This statement takes two arguments; a number from 0 to 255 which is the data to be output to the port, and another number from 0 to 255 which specifies the port to be output to. Thus, again assuming that the Turtle is connected to port #30, the following short program will send the Turtle forward:

```
100 REM THIS PROGRAM MOVES THE TURTLE FORWARD
110 OUT 30 (port address), 5 (data)
120 END
```

And this one will stop it and turn on the horn:

```
100 REM BEEP
110 OUT 30, 192
120 GND
```

With 256 possible Turtle states, remembering which number went with which state would be tedious at best. Fortunately, this is not necessary, since one can control each Turtle function independently.

This can be done easily using a binary expansion of decimal numbers. If we expand the third Turtle state in the above table the result is:

$$0x2^7 + 0x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 =$$

$$0 \times 128 + 2 \times 64 + 1 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 = 58$$

This suggests a way of independently controlling each Turtle bit: We start off with a Turtle state variable (say "T") initialized to zero. Then, any time we want to turn on a specific Turtle function, we add 1 times its binary place value to the variable T and then output the variable to the Turtle. If we later want to turn that Turtle control line off without affecting the other control lines, we simply subtract the binary place value from the variable and again output it to the Turtle. The control functions and their binary place values are summarized below:

<u>Turtle Function</u>	<u>Bit</u>	<u>Place Value</u>
Tone control	7	$2^7 = 128$
Horn off/on	6	$2^6 = 64$
Pen	5	$2^5 = 32$
Lights	4	$2^4 = 16$

Left Motor 2	3	$2^3 = 8$
Left Motor 1	2	$2^2 = 4$
Right Motor 2	1	$2^1 = 2$
Right Motor 1	0	$2^0 = 1$

The following program illustrates how this method might be used:

```

90 LET T = 0
100 PRINT "WHAT TURTLE BIT DO YOU WANT TO"
110 PRINT "TURN ON?"
120 INPUT A
130 IF A < 0 THEN GO TO 170
140 IF A > 7 THEN GO TO 170
150 LET T = T + 2 ^ A
160 OUT 30, T
170 PRINT "WHAT TURTLE BIT DO YOU WANT TO"
180 PRINT "TURN OFF?"
190 INPUT A
200 IF A < 0 THEN GO TO 100
210 IF A > 7 THEN GO TO 100
220 LET T = T - 2 ^ A
230 OUT 30, T
240 GO TO 100
250 END

```

Of course, there is a problem with this method; if you try to turn off a Turtle function that has not been previously turn on, you run into trouble. For example, suppose that the horn is on and emitting a low tone and all other "6" bits are off. The Turtle register and variable T would contain the binary number 0100000 or 64 decimal. If one then tries to turn RM1 off by subtracting its binary place value of 1 from T, the result is 00111111 or 63 , which would turn off the horn and turn the lights and pen on. To remedy this problem, it would be necessary to check if a Turtle bit was on before turning it off and vice-versa. This is left as the proverbial "exercise for the reader."

One problem with controlling the Turtle remains: How do we use BASIC to control the distance the Turtle moves, and the degree which it turns. Since the direct current motors on the Turtle are (hopefully) connected to a constant supply voltage, they rotate at a constant speed and move or turn the Turtle at a constant velocity. Therefore, to control the distance the Turtle moves, we must control the time the Turtle motors are on. To do this a BASIC program must perform the following steps:

1. Set the Turtle motor state to forward, back, right or left
2. Wait for a time proportional to the desired distance or angle of turn
3. Set the Turtle motor state to STOP

Setting the Turtle motor state is easy, but waiting for a specific time is tricky. On microcomputing where time sharing is not being done, this can be easily accomplished by use of a timing loop. Since, in such systems, it takes a constant length of time to interpret each BASIC statement, by executing a controlled number of statements it is possible to wait a controlled length of time. The easiest way to do this in BASIC is with the familiar FOR-NEXT loop as shown below:

```

100 REM THIS PROGRAM WAITS A LENGTH OF TIME DEPENDING ON THE VALUE OF N
110 N = 1000
120 PRINT "START"
130 FOR M = 1 to N STEP 1
140 NEXT M
150 PRINT "DONE"
160 END

```

The length of time elapsed between the printing of START and DONE on the terminal will be very closely proportional for N for this program since the program goes through the for-next loop N times, taking a constant length of time per loop. It is therefore possible to write the following subroutines for controlling the Turtle: FORWARD and BACK, which take an argument of how far to move the Turtle in inches, and RIGHT and LEFT which take an argument of what angle to turn the Turtle. All these subroutines are called by GOSUB statements with the distance to be moved or angle to be turned contained in variable A. They use the following constants for FORWARD and BACK (these will be correct if the Turtle is wired as previously described):

<u>Turtle Motion</u>	<u>Binary Turtle State</u>	<u>Decimal Equivalent</u>
FORWARD	00000101	5
BACK	00001010	10
RIGHT	00000110	6
LEFT	00001001	9
STOP	00000000	0

```

20000 REM FORWARD SUBROUTINE
20010 LET A = A* (FDCONS)
20020 OUT(PORT), 5
20030 FOR N = 0 TO A STEP 1
20040 NEXT N
20050 OUT(PORT), 0
20060 RETURN

```

```

20100 REM BACK SUBROUTINE
20110 LET A = A* (BKCONS)
20120 OUT(PORT), 10
20130 FOR N = 0 TO A STEP 1
20140 NEXT N
20150 OUT(PORT), 0
20160 RETURN

```

```

20200 REM RIGHT SUBROUTINE
20210 LET A = A* (RTCONS)
20220 OUT(PORT), 6
20230 FOR N = 0 TO A STEP 1
20240 NEXT N
20250 OUT(PORT), 0
20260 RETURN

```

```
20300 REM LEFT SUBROUTINE
20310 LET A = A* (LITCONS)
20320 OUT(PORT), 9
20330 FOR N = 0 TO A STEP 1
20340 NEXT N
20350 OUT(PORT), 0
20360 RETURN
```

In these subroutines (port) should be replaced by the number of the parallel IO port the Turtle is connected to and (FDCONS), (BKCONS), (RTCONS), and (LITCONS) should be replaced by experimentally determined constants which make the Turtle go a distance in inches (or centimeters or whatever) equal to the arguments to the FORWARD and BACK subroutines and turn the number of degrees specified in the arguments to the RIGHT and LEFT subroutines. To make them easier to understand, these subroutines do not preserve the state of the pen, lights, and horn when they are called but this could be easily done by the previously-discussed method of controlling the Turtle bits independently of one another. Subroutines such as these could then be used in BASIC programs to solve mazes, map rooms, and draw geometric figures (to name only a few possibilities).

TERRAPIN TURTLE ASSEMBLY MANUAL

The Turtle is assembled in the following order:

1. Printed Circuit (PC) Board
 - a. resistors
 - b. diodes
 - c. transistors
 - d. switches
 - e. darlington
 - f. integrated circuit (IC)
 - g. potentiometer (pots)
 - h. light emitting diodes (LEDs)
 - i. capacitor
 - j. jumper
 - k. solder in cable
 - l. leads
2. Attach ball joint and touch disk
3. Assemble solenoid assembly
4. Mount solenoid to bottom plate
5. Attach one motor electrically
6. Screw spacers, brackets, top plate onto one motor.
7. Wind cable around spacers
8. Connect other motor electrically, attach with brackets.
9. Screw on top plate (pc board) fully
10. Attach toes, speaker mounting to bottom plate
11. Attach speaker electrically
12. Attach solenoid electrically, mount bottom plate
13. Mount speaker
14. Put on tires
15. Mount wheels
16. Screw on dome

Copyright © 1979 by Terrapin, Inc.
Drawings by Erlyne Gee
rev. 8/79

The Terrapin Turtle can be assembled in a few hours by anyone experienced in electronic kit assembly. This manual covers all of the steps necessary to assemble a fully-working Turtle. All Turtle parts are included in the Terrapin kit. The tools you will need are:

soldering iron (fine point)
 longnose pliers - helpful
 wirecutter, diagonal cutter, or shears
 screwdriver - flat bladed
 volt ohm meter (VOM) - helpful for debugging
 1/16" allen wrench (included in kit)

The main steps in the Turtle assembly are 1) assembly of the pre-cut, pre-drilled printed circuit board that functions as the electronic control for all of the Turtle functions, and 2) mechanical assembly of the Turtle body.

Most of the working components of the Turtle are also structural. The motors which drive the wheels form the main body of the Turtle. They are held together by rigid spacers. A metal bottom plate, and a top plate (which is the pc board) are connected by vertical brackets. The bottom plate holds the pen solenoid assembly. All hardware for assembling the frame is included in your kit.

Assemble the Turtle following the suggested procedure below. Special procedures and tips are included. Instructions for the novice are included under special headings. Important details and warnings are *in italics*.

Please inspect your kit to make sure that you have *all* of the parts listed in the proper (listed) quantities and that you can identify each part.

Parts List

Mechanical

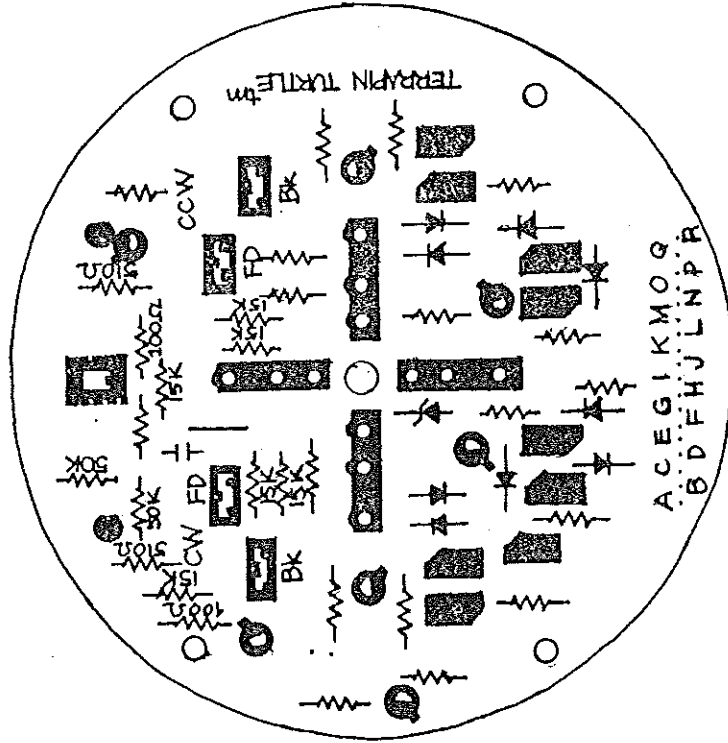
Part	Quantity
Motors	two
Wheels	two
Tires	two
Pen solenoid	one
Pen plunger	one
Pen spring	one
Plastic balls (Turtle toes)	two
Touch sensor pulley	one
Touch sensor ball joint	one
Allen wrench	one
Motor brackets	four
3-1/2" motor spacers	three
Rubber speaker mount	one
solenoid mounting brackets	two
Ball point pen refill	one
Speaker	one
6-32x1/4" screws	ten
Lock washers for 6-32 screws	ten
6-32x1" screws	eight
6-32 nuts	two
Bottom plate	one
10-32 dome screw	one
10-32 dome washer	one
6-32 nylon washers	four
1/4 inch screws	two
1/4 inch lock washers	two

Electrical

Part	Quantity
Printed circuit board	one
1K resistors	18
510 ohm resistors	two
100 ohm resistors	two
15K resistors	six
50K resistors	two
1N4000 diodes	nine

- 3.6 zener diodes
- 1K POTS
- .1µf capacitors
- 555 IC
- GED40c4 darlingtons
- 2n2222 transistors
- Micro switches
- Male molex pins
- Female molex pins
- Male molex connector
- Female molex connector
- Ten foot cable

ELECTRONIC ASSEMBLY:
ASSEMBLING THE PRINTED CIRCUIT BOARD



Preparation: Remove the printed circuit (pc) board from your Terrapin Turtle kit. It is a pre-cut 5" diameter circle with drilled holes for all components. Insert the components from the top of the board (the side printed with component pictures) and solder the component leads to the pre-tinned copper foil on the bottom of the board.

Assure yourself that all electronic components from the parts list are in your kit. Insert them in the recommended order following the printed pattern on the board and the diagrams in this booklet.

Resistors: Find the resistors.

Resistor Value	Quantity
50K resistors	2
15K resistors	6
1K resistors	18
510 ohm resistors	2
100 ohm resistors	2

Resistor Color Code: Resistors are printed with standard color code bands. There are three bands indicating the value of the resistor, and one band indicating the precision of the indicated value (this last band may be ignored). The resistor value is (1st digit)(2nd digit) x 10^(3rd digit). The third digit indicates the number of zeros according to this chart:

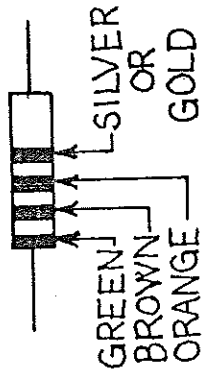
0 black	5 green
1 brown	6 blue
2 red	7 violet
3 orange	8 grey
4 yellow	9 white

The letter "K" stands for kilo-ohms, or one thousand ohms.

Note: There are three resistors (W) together, the top two labelled 15K, the bottom one unlabelled (following the convention of putting labels below symbols). The bottom one is the 1K resistor.

Note: Clip the lead on the 100ohm resistor level with the pc board to prevent shorting with the screw.

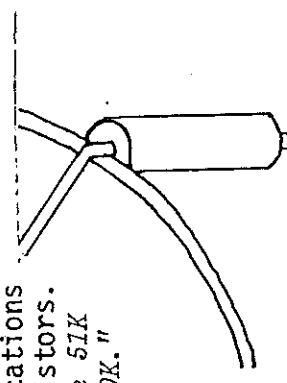
AN EXAMPLE
51K RESISTOR
51000 OHMS



RESISTOR

Resistor values can be identified by the colored bands painted on them (see box pg. 5). Bend the leads of the resistors so that they are spaced properly for insertion into the board. Proper spacing is easy to achieve by bending the leads over the edge of the pc board as illustrated. The board is just the thickness required between the resistor body and the right angle bend in the lead. Each resistor location is marked with values on the board. Unmarked locations require 1K (1000 ohm) resistors.

Note: The location for the 51K resistor may be marked "50K."



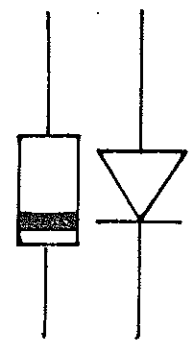
BEND RESISTOR LEADS

Resistors can be inserted into their board location in either direction; it does not matter which way the bands are oriented. Insert the resistors into the board and solder them.

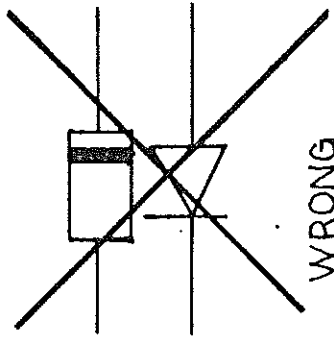
Diodes: There are nine diodes and one zener diode. Be sure not to confuse the ordinary power diodes and the zener diode. The zener is longer than the power diodes.

Diode cases have a single black band printed at one end. This corresponds to the straight line in the diode symbol (or zener diode symbol). Diodes must be oriented with the band on the case corresponding to the line on the pc board.

DIODE



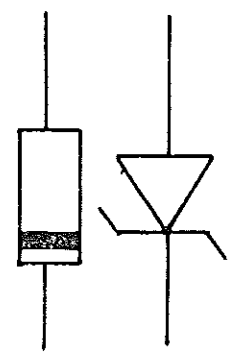
ORIENTED CORRECTLY



WRONG

Bend the diode leads for insertion into the board. Proper spacing may be achieved by gripping the diode lead next to the body with a long-nose plier and bending the lead over the plier jaws. Insert the diodes as illustrated.

ZENER DIODE



Soldering

SOLDERING: Proper soldering is essential to the proper assembly and operation of your Terrapin Turtle. To solder correctly, use a small 25 to 40 watt pencil soldering iron. The solder tip must be kept clean. Clean it by wiping the tip with a damp sponge or cloth. Then "tin" the tip by coating it with solder. If the solder won't adhere to the iron or "cakes," clean and retin the tip.

Exercise extreme care in soldering around the Turtle pc board. Forming a solder "bridge" between two foil areas of the board must be prevented for proper assembly and operation of the Turtle. A "bridge" is formed when two different foil lines are joined by solder. This may be the result of too much solder, of accidentally dragging the iron across several foils or of carelessly touching a prior joint with the iron. Inspect the soldering area before and after a soldering joint is made.

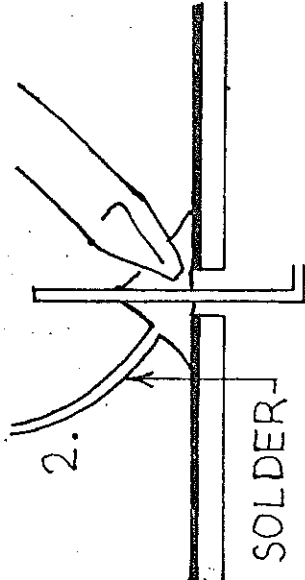
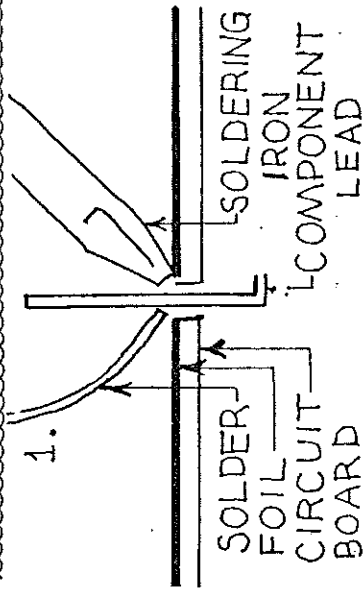
Use the minimum amount of solder necessary for a good connection. Should a bridge develop, turn the Turtle pc board upside down and apply the iron to the bridge. The solder will drip down onto the tip, thereby destroying the bridge.

To solder: Once the electrical component is properly placed in the pc board, turn it so the lead is up. Place the soldering iron tip at 45° to both lead and pc board foil. Heat for approximately three seconds. Apply solder to the connection on the *other side* from the soldering iron. The foil and component lead should melt the solder, not the iron directly. Let the solder flow around the joint, then lift the soldering iron straight up to avoid forming any bridge.

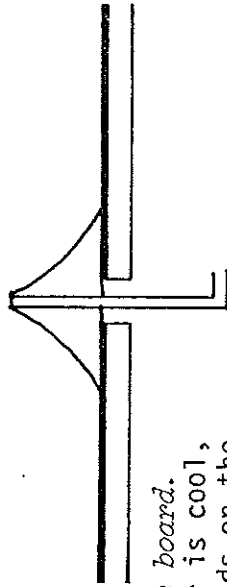
NOTE: Do not use acid core solder or paste flux.

Insertion: Be sure that the components are inserted *all* the way down, so that their bodies are

Soldering



3. CLIPPED OFF LEAD.



close to the pc board.
When the solder is cool, cut off the leads on the bottom of the board close to the board. Save one excess lead for use as a jumper.

Be sure that component leads are cut close to the board, especially those near the mounting holes.

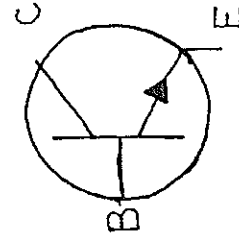
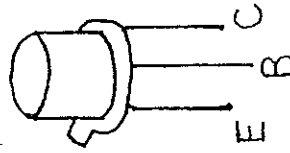
Important: The diodes must be inserted with the proper orientation. The sensor diode must be inserted in the proper board position. Solder in the diodes and clip leads close to board.

WARNING: To avoid eye injury when cutting off excess lead lengths be sure to hold the leads so that they cannot fly towards your eyes.

Transistors: Find the seven transistors. They are packaged in three-lead metal cans or black plastic packages. Insert the transistors into the board. If the transistors are metal cans the tabs on the board pattern must be aligned with the tabs on the cans. If the transistors are in plastic packages, the flat side of the board pattern must be aligned with the flat side of the transistor. Their three leads are emitter, collector and base. The tab is always near the emitter. The base is in the center. *Transistors must be oriented correctly.*

Solder the transistors to the pc and clip leads close to board.

TRANSISTOR

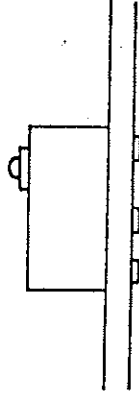


E = EMITTER
B = BASE
C = COLLECTOR

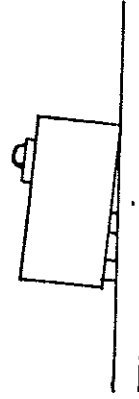
Switches: There are four switches in the Turtle pc board. They are activated by the touch disk to

indicate that the Turtle's shell is in contact with an object. The switches are to be inserted in a star in the center of the board. Since the switch leads are short and wide, the holes for them are drilled rather large. You should not expect a right fit for all the leads; the switches will not hold themselves in position while being soldered. Insert the switches in the board. They are indicated on the printed pattern by large white rectangles. Since the switches may fall out of the board when it is turned to the clad side for soldering, place a book behind the switches to hold

SWITCH
FLUSH WITH
PC BOARD



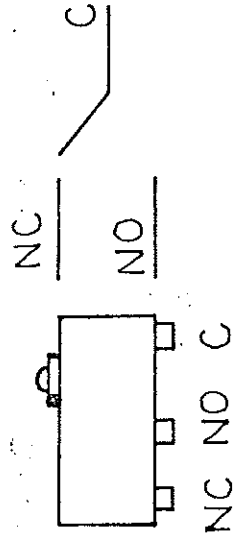
CORRECT



INCORRECT

them in while you turn the board around. Solder in the leads. *Make sure that the bodies of the switches are flush with the top of the board.* Since the leads

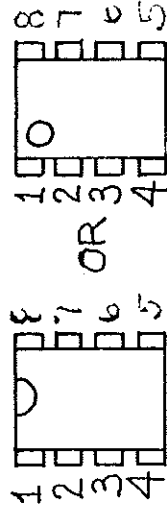
SWITCH



NC = NORMALLY CLOSED
NO = NORMALLY OPEN
C = CLOSED/COMMON

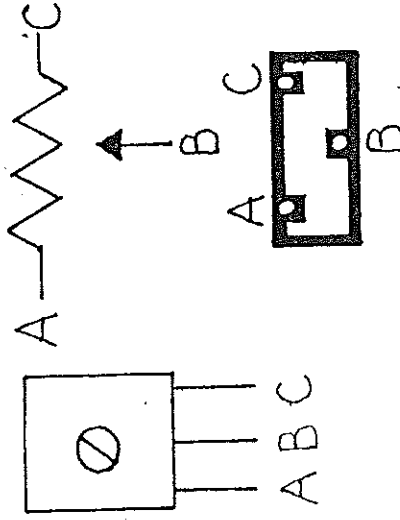
with the 555 improperly placed is likely to damage the 555. The pins are numbered (by convention) as shown in the drawing below. (The 555 is viewed from the top.) The notch or dot in the case indicates the "pin 1" end. The integrated circuit must be inserted so that this notch is at the same end as indicated on the printed circuit board pattern.

INTEGRATED CIRCUIT



Pots (Variable Resistors): There are four trimmer pots (potentiometers, variable resistors) on the Turtle board for tuning the motor circuits. Insert the triangularly spaced leads into the top of the board. Solder and clip leads close. Make sure that the adjustment slots on the pots face the front of the board.

POTENTIOMETER

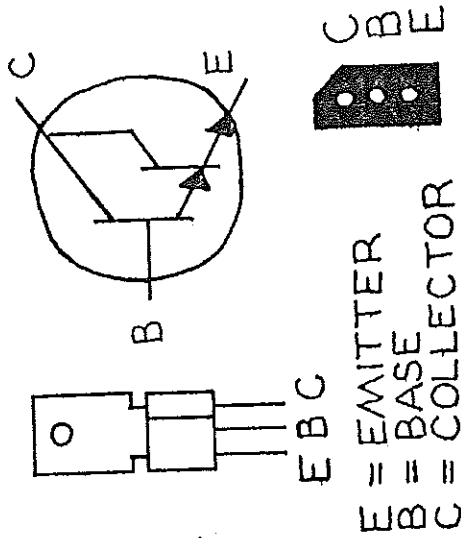


When you power up your Turtle, you will use the pots to adjust the Turtle motors so that your Turtle runs straight (see "Adjustments," p. 31).

are thick, they may require substantially more heating than the leads of other components. If the switches are not flush with the board, the touch sensors will not function properly. Make sure that your solder joints make good contact with the printed circuit foil at these holes.

Darlingtons: The darlingtons are transistor-like components. Insert the nine darlingtons in the board in the appropriate positions. The notched edges of the darlington cases must be aligned with the notched edges on the symbols. The darlingtons have three relatively short leads. Be sure that the leads contact the pc foil. Solder to the pc

DARLINGTON

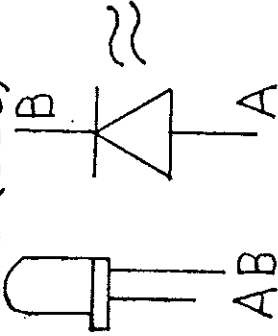


and clip leads close to board. The tabs at the top of the darlingtons should not touch each other while the Turtle is in operation.

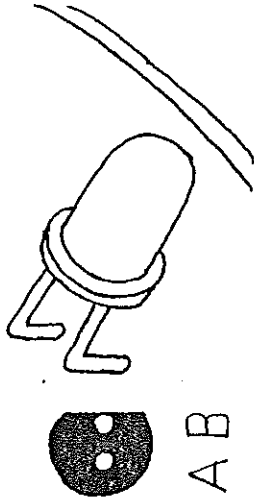
555 IC: Insert the 555 integrated circuit (IC) into the board. It is essential that the 555 be oriented properly. Attempting to operate the Turtle

LEDs: There are two light emitting diodes (LEDs) which serve as the Turtle's eyes. The cathode of these diodes is marked by the flat side of their case. This must coincide with the flat side of

LIGHT EMITTING DIODE (LED)



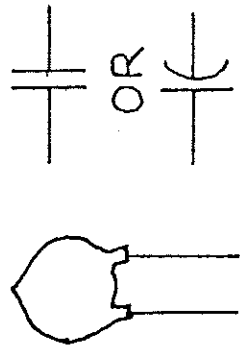
BEND LEADS FORWARD



the symbol on the printed circuit board pattern. Insert the LEDs in the board, leaving $\frac{1}{4}$ " of the leads on top of the board. Solder on to the pc board. Then bend the LED leads on top of the board at a right angle, so that the LEDs lie flat on the board, facing front. Clip leads close to bottom of board.

Capacitor: There is a single $.1\mu\text{F}$ capacitor in the Turtle. Insert it in the board at the proper position. The capacitor leads come with approximately the right spacing for these holes. Gently adjust them until they fit. Orientation is unimportant. Solder in the capacitor and clip leads close to the board.

CAPACITOR



Jumper: A "jumper" is a wire that connects one part of the foil to another. There is one jumper wire on the pc board. It also functions as a test point for the supply voltage. This jumper can be made from an excess lead clipped from a resistor.

The location of the jumper is indicated on the pc board by a line connecting two holes. Use pliers to bend the resistor lead to the size shown above and solder it between the holes.

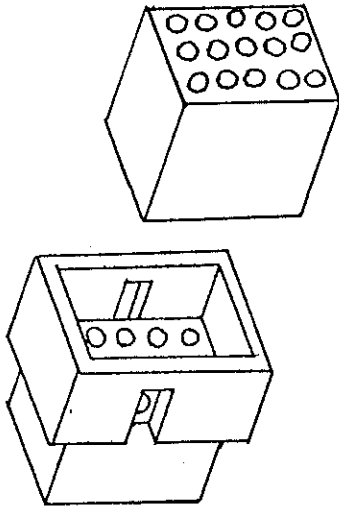
CORRECT JUMPER LENGTH



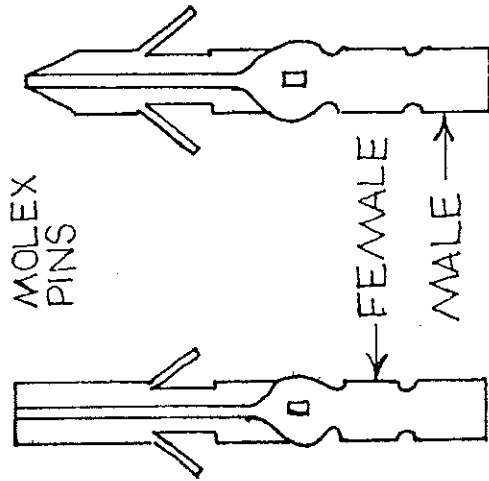
Cable and Connector: The cable assembly consists of ten feet of Terrapin cable, a Waldom MolexTM miniature connector and connector pins (30 pins, 15 male, 15 female). The connector is included to provide you with a way of easily disconnecting your Turtle from its computer interface for purposes of travel, switching your Turtle from computer to computer, etc. The cable assembly also provides an easy way to extend the length of your cable-- simply buy more Terrapin cable or other 15 conductor flexible cable plus another Molex connector and plug the new section in between the two original connector halves. (The connectors are available through many electronic distributors.)

The way you assemble this part of the Turtle should reflect your own needs. You may assemble the cable exactly as recommended below. You may choose to omit the Molex connector and simply solder directly both the Turtle and interface ends of the cable. That would save some assembly time, but would necessitate desoldering a cable any time you wanted to remove your Turtle. You may prefer to use another connector system at your own interface, mounting your connector at the end of the Terrapin cable and on your interface card or board or in place of the Molex connector. In

MOLEX CONNECTOR



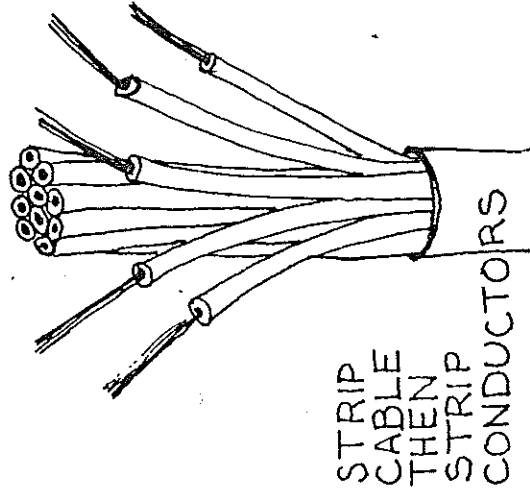
FEMALE RECEPTACLE MALE PLUG



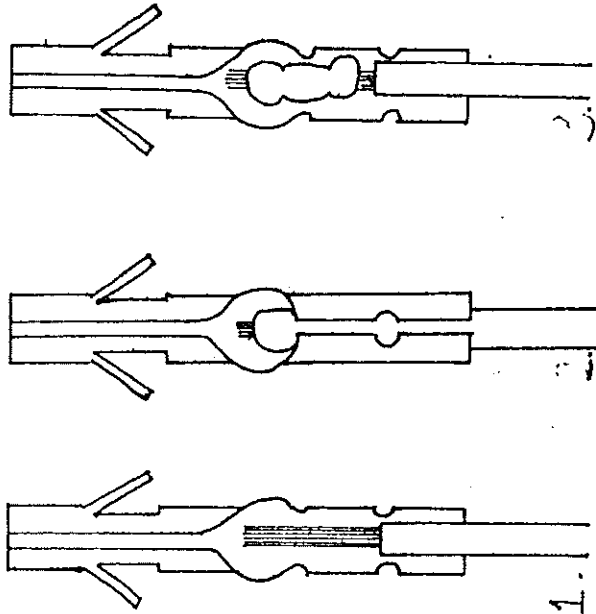
the suggested sequence of assembly, the cable ends are directly soldered to the interface card and to the Turtle board. The Molex connector is placed in the cable near the interface card.

Cut the cable and install the receptacle at a distance convenient for mounting. If you do not mount the receptacle solidly, plan for some type of strain relief for the interface end of the cable. The short piece of cable will be the interface cable; the remainder will be the Turtle cable.

On the Turtle cable, strip back the outer (grey) cable jacket about one inch. Strip the individual conductors of 1/4" of insulation. *The individual conductors are stranded. Be careful of breaking the fine strands as you strip.* The white-grey wire is unused. It can be used to replace another wire if needed.



Solder, then crimp the conductors to female Molex pins in the order below. Be sure to solder the multiple wires (power and ground) together to a single Molex pin. The ground pin should have three conductors from the cable. The power pin should have two.



INSERT WIRE

CRIMP

SOLDER

Molex connector hole	Wire color	Turtle hole	Function
1	black	A	lights (L)
2	brown	B	horn high/low (T)
3	red	C	left touch (LT)
4	orange	D	left motor 1 (LM1)
5	yellow, white-brown	E, L	supply, power
6	green	F	pen (P)
7	blue	G	left motor 2 (LM2)
8	purple, grey, white-red	H, I, M	ground
9	white	J	back touch (BT)
10	white-black	K	right motor 2 (RM2)

	white-orange	N	right motor 1 (RM1)
11	white-yellow <td>O</td> <td>right touch sensor</td>	O	right touch sensor
12	white-green <td>P</td> <td>front touch (FT)</td>	P	front touch (FT)
13	white-blue <td>Q</td> <td>horn on/off (H)</td>	Q	horn on/off (H)
14	white-purple <td>R</td> <td>user</td>	R	user
15			

Crimp connections: The Molex pins have a divided tube of metal designed to be crimped to hold in the wire. Tin the wire. Tinning makes soldering easier. Insert wire into the crimp tube. Firmly crush the tube, folding the flaps over the insulated part of the wire with longnose pliers or crimping tool. Solder the wire to the tube on the inside with a small amount of solder. *Be sure not to get large drops of solder on the outside of the Molex pins, as it will make insertion into the connector halves difficult.*

Cut off the grey-white wire if you haven't used it. Insert the pins in the receptacle half of the connector in the holes indicated on the chart above. The numbers of the connector holes appear in raised plastic lettering on the connector itself; each number is next to its hole. Push the pin in from the side with the numbers until they click and offer resistance to attempts to remove them. On the interface cable (short piece) strip back one end of the jacket, continue as for Turtle cable, except use male pins and insert in the plug half of the connector. When these connectors are put together, the wire colors should match.

The free ends of the cables must now be soldered. The Turtle cable must be soldered to the Turtle pc board. The interface cable must be soldered to your interface. If you are using the Terrapin S-100 bus interface card, refer to the interface manual. If you have your own interface, solder the wires such that the appropriate functions are connected to their wires as in the chart above. At the back of the Turtle pc board

is a row of holes for the cable wire connections. The holes are labeled on the printed pattern with the letters A through R. Solder the wires according to the chart. (It follows the resistor color code.) Strip back the jacket at the free end of the Turtle cable two inches, strip about 1/4" of the insulation from the individual conductors. Twist the strands and tin the conductors. *Do not overheat while tinning or soldering as the thin insulation will boil off.* Insert each conductor from the component side of the board and solder to the pc traces underneath.

Simply push the two halves of the Molex connector together to complete the Terrapin cable assembly. Pull the halves apart to disconnect the cable at any time.

Motor, Speaker and Pen Solenoid Hookup to

PC Board: The motors, pen solenoid and speaker must be electrically connected to the pc board by hookup wire. You will need the red hookup wire and the spade terminals. Cut the wire into eight equal pieces. Strip both ends of each piece of about 1/4" of insulation. Solder, then crimp the spade terminals onto one end of six of the wires. The wires without spade terminals go to the speaker. Solder the eight wires onto the pc board *on the foil side, not on the component side.* Clip excess wire. The diagram gives the functions of the various leads. Connect the speaker to the speaker hook-up wires by soldering one wire to each tab. Use as little heat as possible. The speaker is easily damaged. Carefully clip the excess wire.

Ground Clips: Make a loop about 1/4" high out of the excess from a component lead. Solder, as indicated, into the two holes in the ground foil on the pc board. Insert from the component side. This provides an easy way to access ground

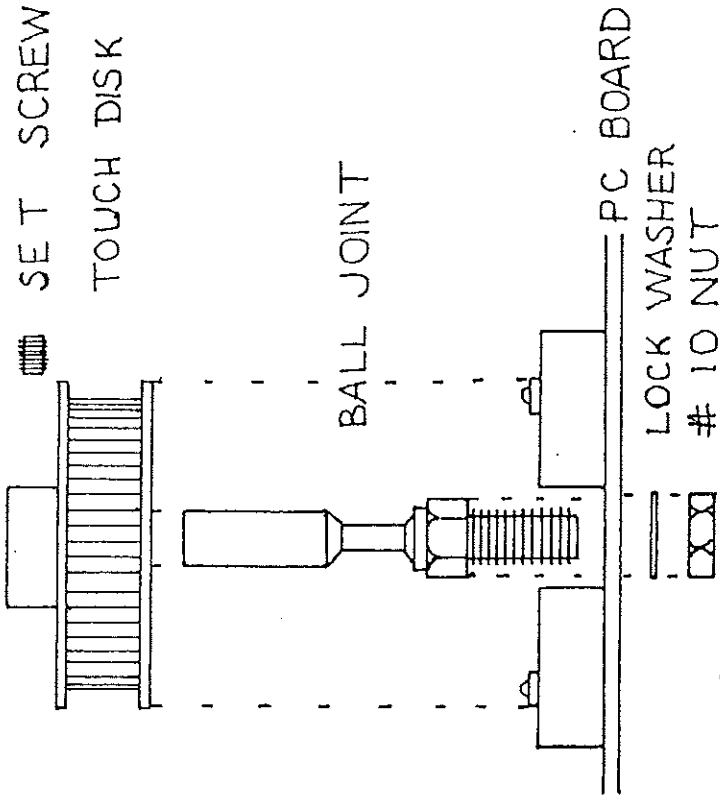
with VOM or Oscilloscope probe clips for testing purposes.

This concludes the electrical assembly of the Turtle. You may now put away your soldering iron.

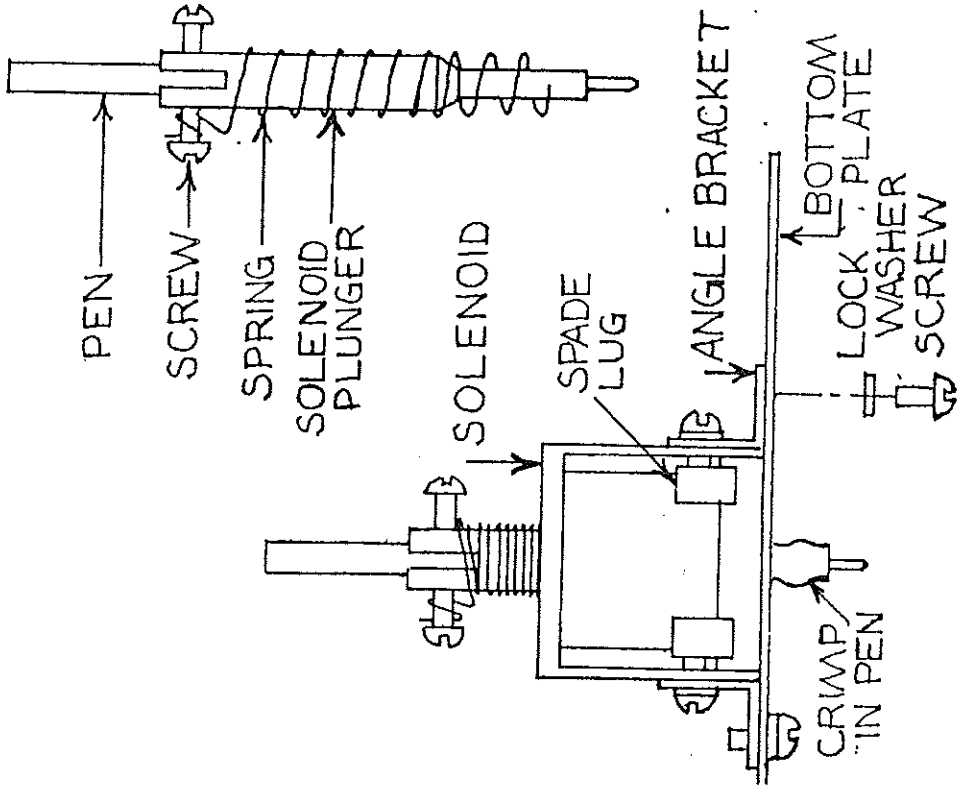
MECHANICAL ASSEMBLY

Ball Joint and Touch-Disk: Find the ball joint assembly, yellow plastic touch-disk, set screw (small black Allen screw), #10 nut and lock washer. Insert the ball joint assembly, threaded end down, through the center of the pc board. (Insert from the component side of the board.) Fasten on pc board with lock washer and #10 nut. Then place the disk on the (top) shaft of the ball joint assembly. The projecting hub should be on top. Use the enclosed Allen wrench to screw the set screw into the threaded hole in the side of the hub. Do not tighten yet. Adjust the height of the touch-disk until it is lightly touching the red buttons on the touch switches. Tighten down the set screw to keep the disk in place. Pressure on the ball joint in any direction should cause the corresponding switch to click.

Pen Solenoid: The pen assembly consists of the solenoid, solenoid plunger, ball point pen refill cartridge, four 6-32x1/4" screws, two angle brackets, two 4-40x3/8" screws, four #6 lock washers, the bottom plate and a spring. Screw the angle brackets (non-threaded hole) onto the solenoid body with the 6-32x1/4" screws, and lock washers. Then screw the angle brackets onto the bottom plate, using the same size screws and lock washers. The sides of the solenoid body should be parallel to the bottom plate edges. Remove the standard ball point pen

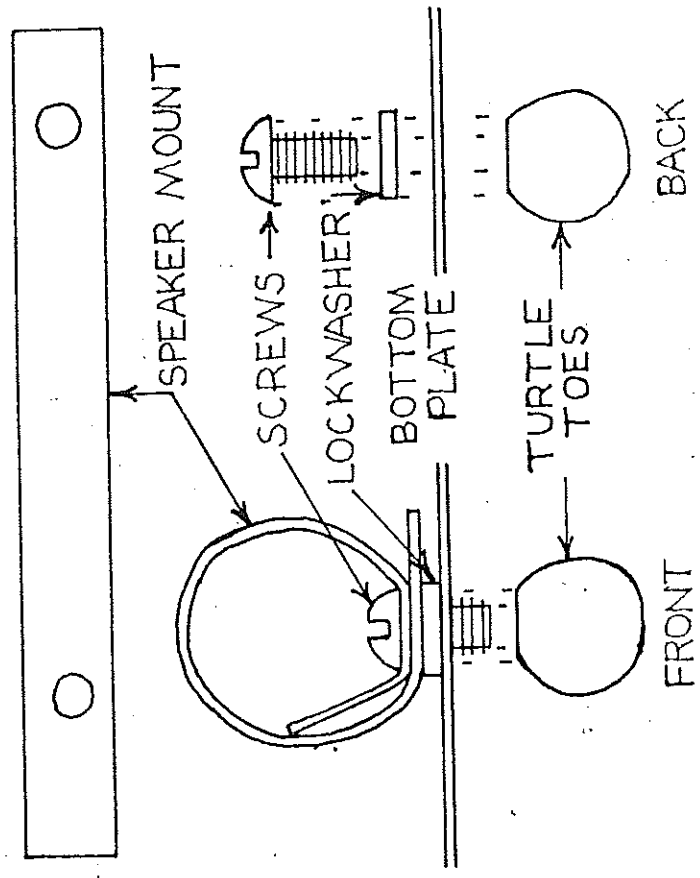


refill from its paper packet. (These refills are available at your local stationery store.) Following the instructions on the packet, break the refill off at the bottom line. Note that there is a crimp near the ball end of the pen. This must be rounded out with pliers so that the pen may be inserted into its holder. Insert the pen through the center hole of the plunger, with the ball end of the pen emerging from the pointed end of the plunger. Screw 4-40x3/8" screws into the two holes at the top end of the plunger. Adjust the pen until it sticks out about 7/8" from the pointed end of the plunger. Adjust the 4-40 screws, as set screws, to grip the pen and hold its position. Do not crush the pen. Slip the spring over the pointed end of the plunger and with pliers wrap the end of the spring around one screw. Insert this assembly into the hole in the center of the solenoid body.



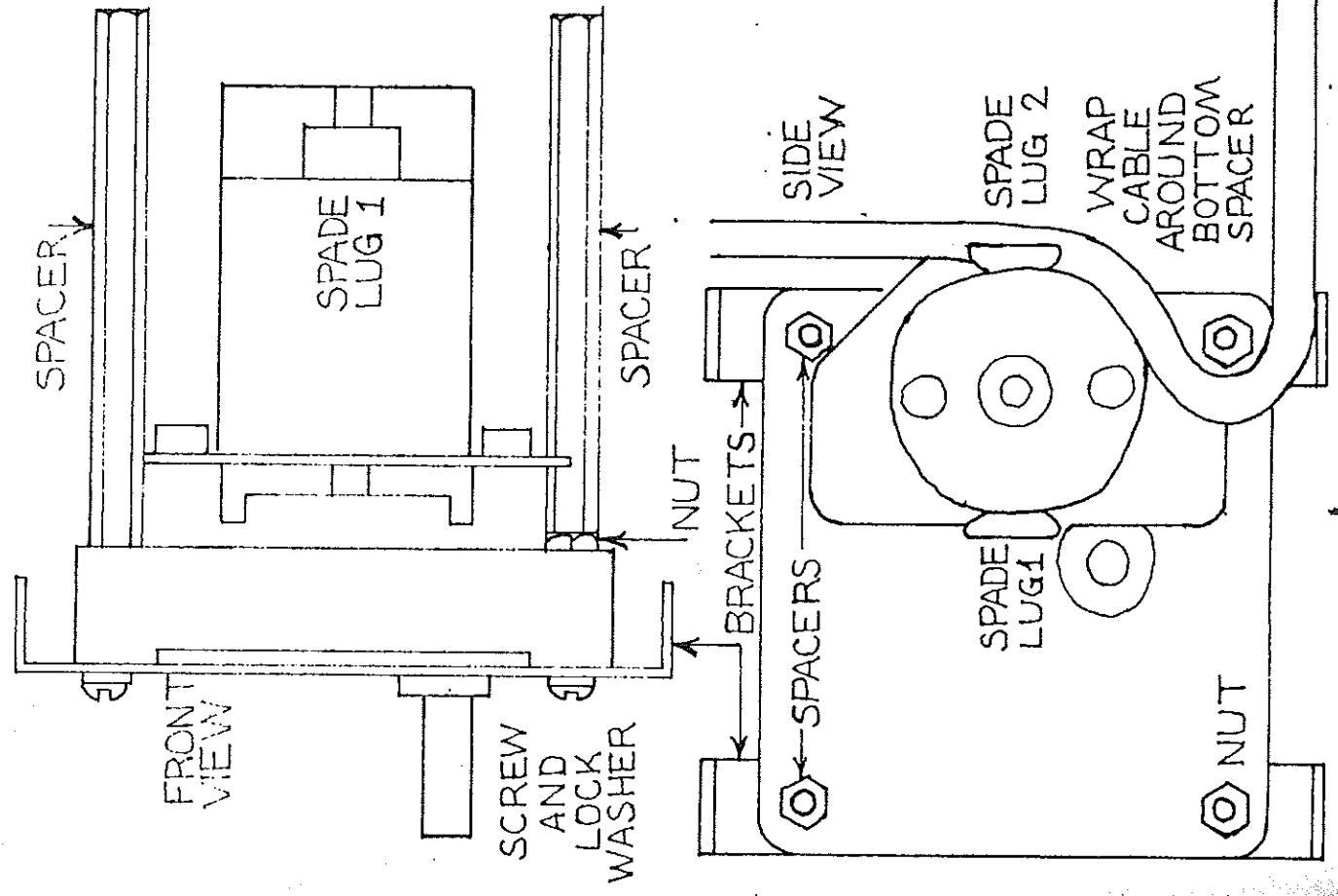
The ball end of the pen will protrude from the bottom of the bottom plate. To prevent the plunger assembly from falling out, use pliers to gently crimp the pen beneath the bottom plate. In order to assure that you do not obstruct ink flow, place a screw or other piece of hardware between the handle end of your pliers such that the jaws of the pliers cannot close completely.

Turtle Toes and Speaker Mount: You will need the two black threaded balls (Turtle toes), two 1/4" split ring lock washers, two 1/4-20x1/2" screws, speaker mount (belting with pre-punched holes), and the bottom plate. Screw one Turtle toe onto the bottom (non-solenoid) side of the bottom plate using one screw and lockwasher. Mount the other toe including the speaker mount. Form the speaker mount into a ring, matching the holes. Insert the screw through the holes, holding the loop together, with the head of the screw inside the ring. The speaker mount now defines the front of the Turtle.



Right Motor Assembly: You will need one of the motors, the three aluminum hex cross section spacers, two motor brackets, four 6-32x1" screws, four #6 lock washers, one 6-32 nut, the pc board, two 6-32x1/4" screws, and two #6 nylon washers. Attach three spacers and two brackets to the motor

LEFT MOTOR ASSEMBLY



NOTE: For right motor assembly, see page 25.

Attach Pen Solenoid Leads: Be sure that the pen solenoid leads come through the center of the Turtle, between the two motors. Attach the spade terminals to the space lugs on the solenoid.

Attach Bottom Plate: Screw the bottom plate onto the motor brackets with 6-32x1/4" screws and lock washers. The solenoid inserts straight up into the center of the Turtle, between the two motors. *Be careful not to break the speaker leads.*

Mount Speaker: Push the speaker firmly into the speaker mount, so that it is gripped tightly. Do not put your fingers through the speaker. Push on the rim.

Tires and Wheels: Stretch the tires onto the wheels. This may be somewhat difficult, as the tires must stretch substantially. Loosen set screws with the Allen wrench (Turtle tuner). Put wheels on the motor shafts (sticking out of the sides of the Turtle). Make sure that the set screw is on the *outside* of the wheel. Rotate each wheel until the set screw is over the flat portion of the shaft. Tighten set screw onto shaft to secure wheel.

Dome: Screw the dome onto the shaft of the ball joint using the 10-32x1" screw, with the nylon shoulder washer under the dome, curved side down.

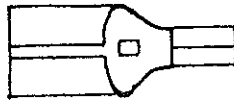
Behold your finished Terrapin Turtle!

with long screws and lock washers. In the unused space insert a long screw, lock washer and nut. Attach the pc board to the top of the motor brackets with short screws and nylon washers. The top is the side with two spacers. The motor shaft should be near the bottom. The front is the side with the nut. The front of the pc board is the side of the Turtle eyes (LEDs). *It is important to use nylon washers between the bottom of the pc board and the brackets. If nylon washers are neglected, electrical shorts may develop.* Keep all electrical leads in the center of the Turtle, between front and back spacers. To provide strain relief for the Terrapin interface cable, wind it around the back spacers.

Motor Electrical Leads:

Attach right motor leads RM1 to right spade lug 1 and RM2 to right spade lug 2 by sliding the spade terminals onto the spade lugs, with spade lug 1 on the outside and spade lug 2 on the inside. Attach the left motor leads to the left motor: with LM1 on the inside, LM2 on the outside. Tape the spade terminals with electrical tape to insulate them. *Bend the spade terminals so that they avoid contacting the motor case and causing electrical shorts.*

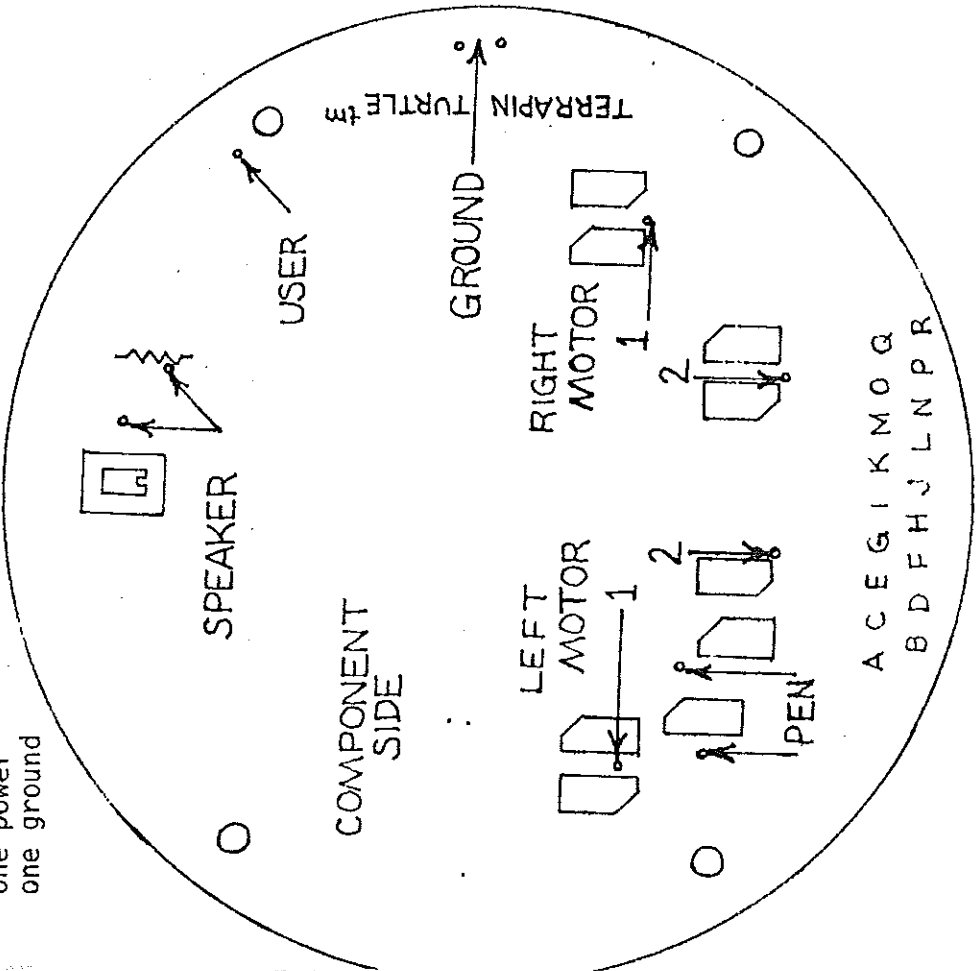
SPADE TERMINAL



Left Motor Assembly: The left motor is oriented symmetrically opposite the right motor. The motor shaft goes near the bottom. Attach it to the spacers and the remaining two brackets using 6-32x1" screws and lock washers. Use a 6-32 nut where there isn't a spacer. Screw the pc board onto the brackets with 6-32x1/4" screws and nylon washers between the pc board and the brackets. You may have to bend the brackets to make things fit.

OPERATING INSTRUCTIONS:
TALKING TO THE TURTLE

Communication between the Turtle and your computer takes place over the Turtle cable. The 15 connections on the connector provide:
eight wires for Turtle control
four wires for the touch sensors
one spare for user expansion
one power
one ground

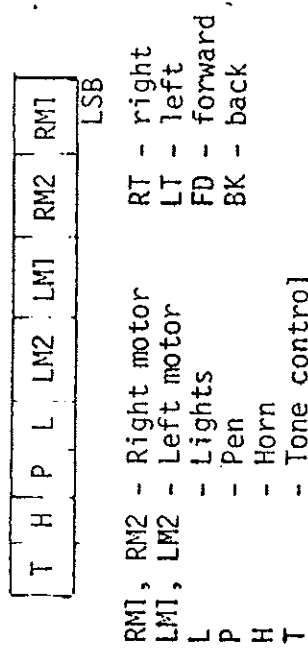


See the chart on page 18.

Power Supply: The Turtle will operate with a supply voltage anywhere between 12-18v. It requires one amp while moving, a maximum of two-three amps while pushing an object and less while idling. There is no need for a separate logic supply.

Logic 0 = 0v. - 0.6v.
Logic 1 = 2v. - supply.
This is TTL compatible.

Turtle Control Register (inputs to the Turtle):
The eight wires that control the Turtle should be connected to a parallel output port serving as a Turtle control register. The Turtle itself contains no memory. A change in state in any of these eight lines is immediately reflected in the Turtle. The bit assignments and functions are as follows:



Motors: Each motor is controlled by two bits in the control register. If the bits are the same (i.e. both 0 or both 1), the motor will be off. One being "1" and the other "0" will turn the motor on with the direction being controlled by the bit that is a "1." Slower speeds can be had by feeding a square wave of different duty cycles to the motors. This will have the effect of trying to turn the motors on and off very rapidly, but due to mechanical inertia, they will just spin more slowly.

LM2	LM1	RM2	RM1	LM	RM	Turtle Actions
0	0	0	0	off	off	off
0	0	0	1	off	FD	turn LT about left side
0	0	1	0	off	BK	turn RT about left side
0	0	1	1	off	off	off
0	1	0	0	FD	off	turn LT about right side
0	1	0	1	FD	FD	move forward
0	1	1	0	FD	BK	turn right about center
0	1	1	1	FD	off	turn LT about right side
1	0	0	0	BK	off	turn RT about right side
1	0	0	1	BK	FD	turn LT about center
1	0	1	0	BK	BK	move backward
1	0	1	1	BK	off	turn LT about right side
1	1	0	0	off		
1	1	0	1	off	FD	turn LJ about left side
1	1	1	0	off	BK	turn RJ about left side
1	1	1	1	off	off	off

Note: the polarity of forward vs. back on each motor depends on how the pc is connected to the motors. If your Turtle's motors go back when the table says forward, you can correct this by simply exchanging the two spade terminals and leads, one for the other.

LIGHTS "0" turns them off
"1" turns them on

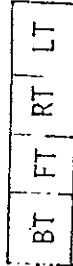
PEN "0" is up
"1" is down

HORN "0" is quiet
"1" is noisy

STONE "0" is the low tone on the horn
CONTROL "1" is the high tone on the horn.

Touch Sensor Register (outputs from the Turtle): There are four wires that reflect the

state of the touch disk. These should be wired to a parallel input port. They are always valid, and need not be enabled.



BT - back touch/FT - front touch
FT - right touch/LT - left touch

Each bit corresponds to one of the four touch sensors. If the Turtle bumps into something, the dome will cause the touch disk to press on one of the four microswitches.

"0" means something is touching.
"1" means no touch.

ADJUSTMENTS

Pen: For best drawing ability the pen should be adjusted in the following manner: Connect the Turtle to a 12-18 volt power supply and to the parallel interface on your computer. Remove the Turtle bottom plate and turn the pen solenoid on by writing a logic "1" to the parallel interface bit connected to the pen. The soft iron solenoid core should be pulled all the way down into the solenoid body by the magnetic field. Loosen the two screws holding the pen refill and adjust the pen refill so that when the solenoid is down the pen refill projects about $\frac{1}{4}$ " below the bottom of the two plastic balls ("Turtle toes"). Now use a software program to write a logic "0" to the pen bit on the parallel interface. The pen solenoid core should move up out of the solenoid body. Adjust the crimp point on the pen refill so that the upper position of the pen refill is about $\frac{1}{8}$ " above the bottom of the two plastic balls. If problems are experienced with the pen refill sticking in the hole in the bottom of the pen solenoid, try inserting a 6/32 foothead lockwasher on the pen refill and the bottom plate. Now verify that the pen moves up and down freely by alternately writing logic 1's and 0's to the pen control bit and observing that the pen extends and retracts. If the pen

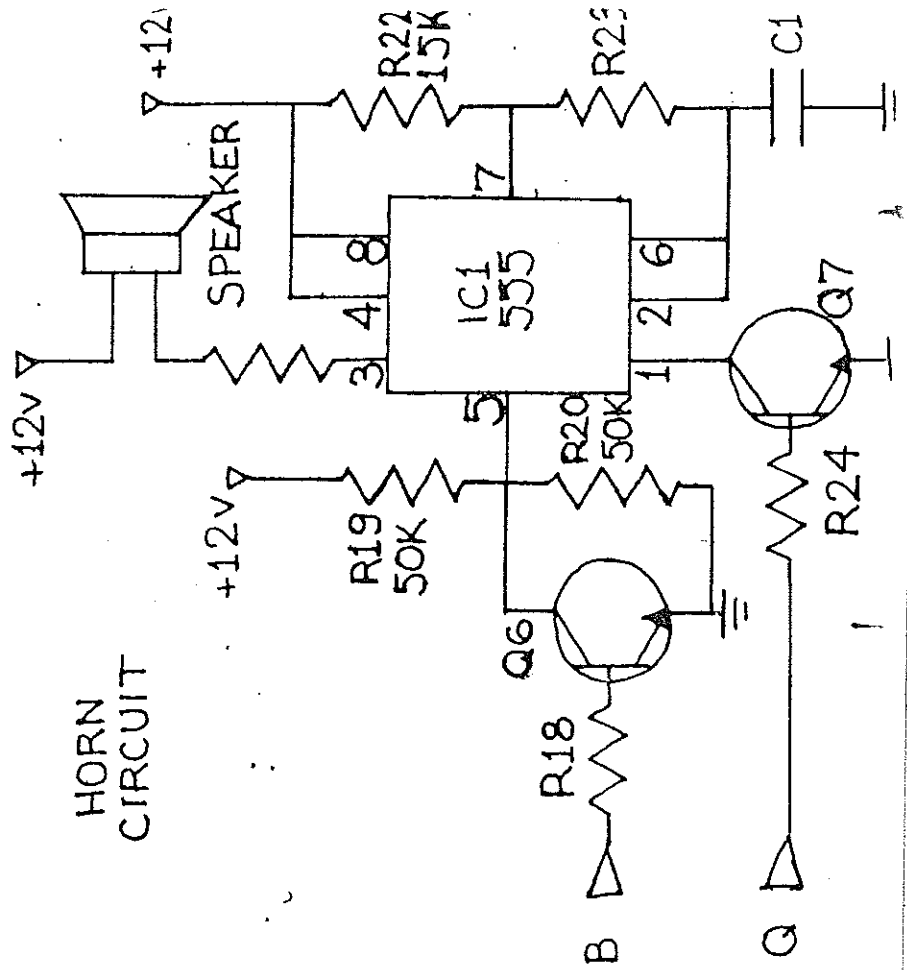
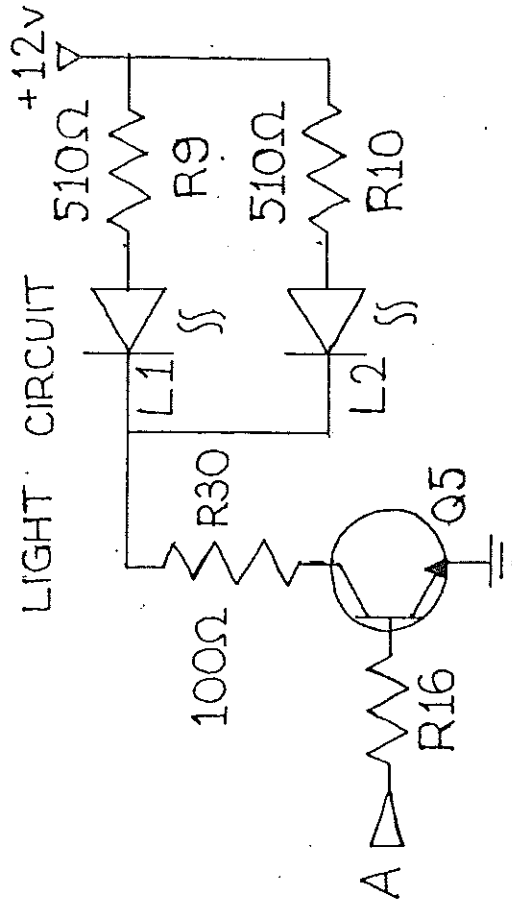
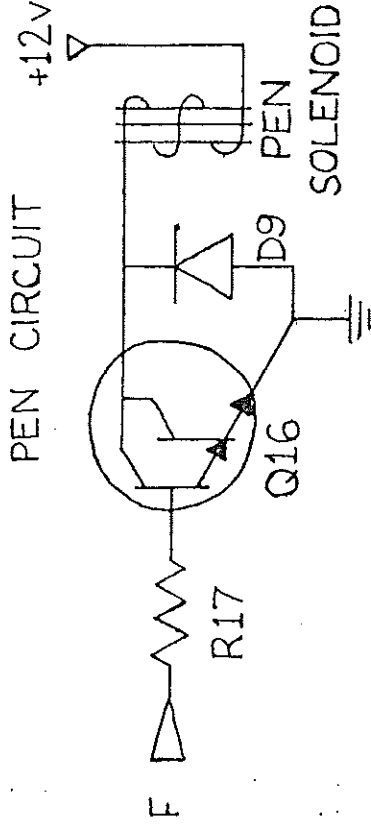
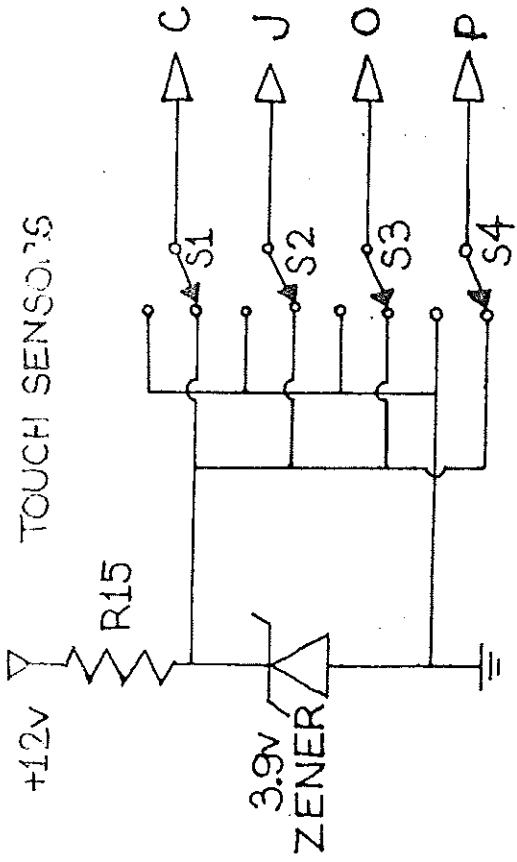
assembly sticks at any point try bending the refill so it travels freely. Now put the bottom plate back on the Turtle, place the Turtle on a sheet of paper, and send it forward with the pen down by writing logic 1's to the pen bit and to RM1 and LM1. If the Turtle does not draw properly, repeat the adjustment proceeding with the pen down in refill position slightly lower or higher than $\frac{1}{2}$ " below the Turtle toes. When properly adjusted the pen solenoid core should not "bottom out" in the pen down position; rather, it should push the pen against the paper with a constant force from the magnetic field inside the solenoid. This allows the pen refill and core to move up and down slightly in response to irregularities in paper height and wheel diameter.

Motors: In order to make your Turtle run straight when going forward or backward and turn at the same speed when going right or left, you must adjust the 4 motor speed adjustment pots. First, use a small screwdriver to turn the adjustment slot on each pot clockwise until it stops. Then place your Turtle on the floor (preferably one with straight floorboards or tile lines) and set it going forward by writing logic 1's to the parallel interface bits connected to RM1 and LM1 and logic 0's to RM2 and LM2. After this is done the Turtle should start going forward (if it travels backward or turns the motors are hooked up incorrectly and should be reconnected as shown earlier in the manual). If, as the Turtle moves forward, it curves to the right or left, turn the motor speed control pot marked FD on the side opposite the direction of turn counter clockwise until the Turtle travels straight. Once the Turtle travels forward properly use your computer to write logic 0's to the parallel interface bits connected to RM1 and LM1 and logic 1's to the bits connected to RM2 and LM2, sending the Turtle backwards. If the Turtle does not travel straight back, adjust the BK pot on the side opposite the direction of turn in the same way the FD pots adjusted until proper backward travel is obtained.

THEORY OF OPERATION

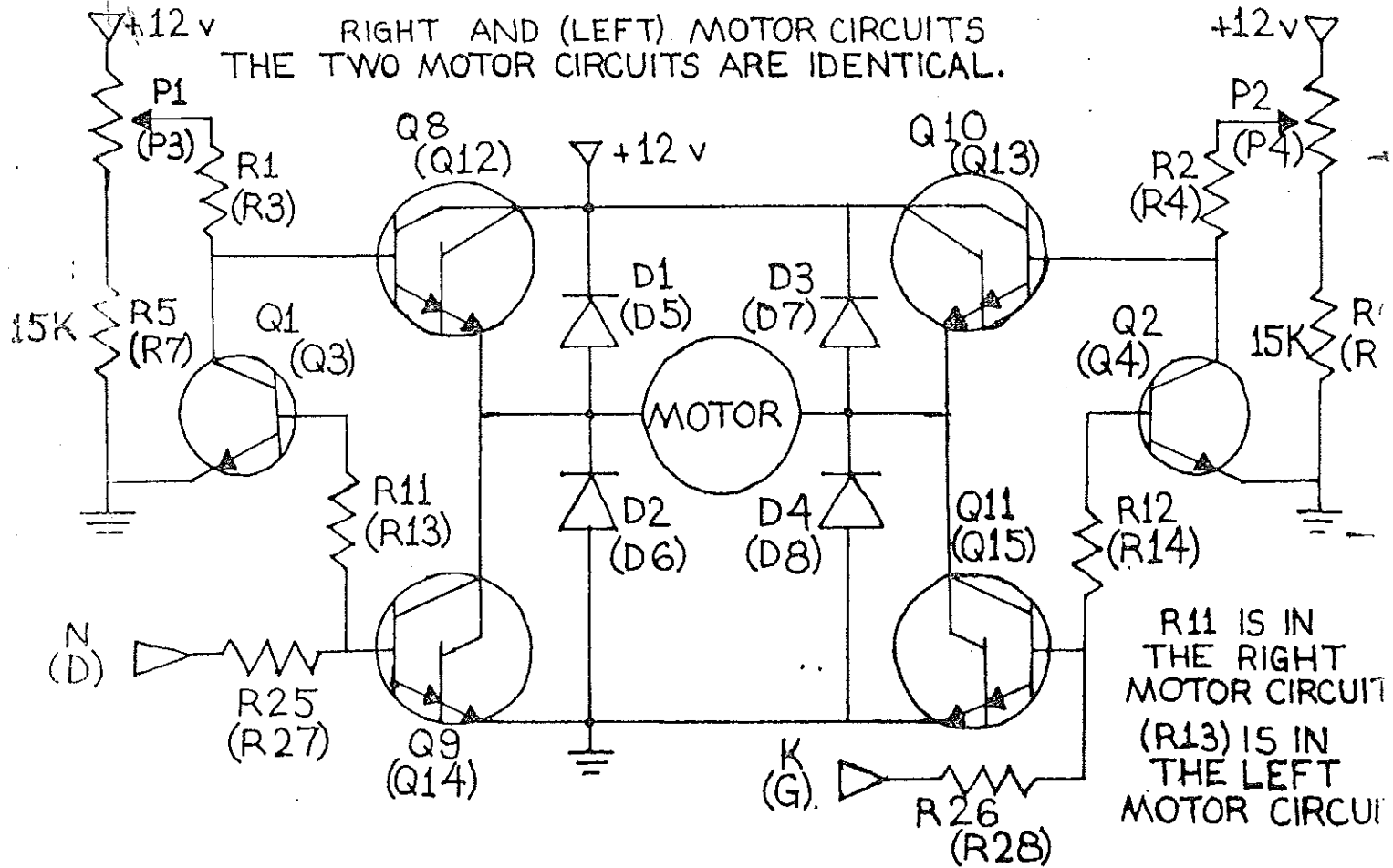
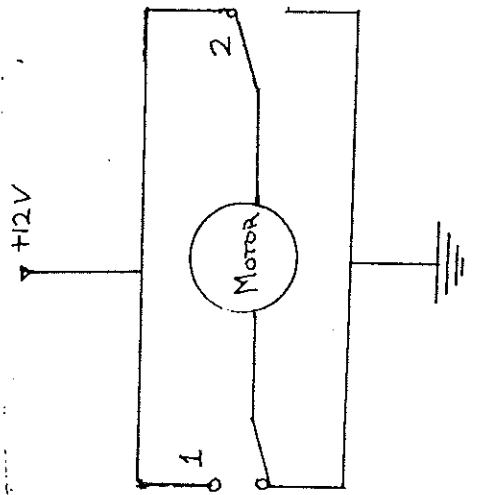
Pen and Light Circuits: The pen and light circuits on your Turtle both work in the same way. For the light circuit, if the voltage at pin A on the Turtle connector (refer to the schematic diagram, next page) is less than .6 volts (a TTL low level) then no current flows in the 1K resistor R16 or in the base of transistor Q5. Since Q5 has no base current, its collector current is zero and the LEDs do not light. If connector pin A is at a TTL high level (three volts or so) current flows through R16 into the base of transistor Q5, saturating it. Current can then flow through current limiting resistors R9, R10, and R11 and also through the two LEDs into the collector of Q5. This lights the LEDs brightly. In the pen circuit, if connector pin F is at less than 1.2 volts, darlington transistor Q16 is turned off and no current flows in the pen solenoid. With a TTL high level on pin F, Q16 saturates, causing current to flow through the solenoid and lowering the pen. Diode D9 absorbs the inductive transient produced when current in the pen solenoid is abruptly cut off.

TOUCH SENSORS



Theory of Operation

Motor Circuit: The motor circuit on your Turtle has a more complex job to do than the pen or light circuits. Not only must the motors be turned on and off, they must also be able to rotate in either direction. To do this the motor circuit must be able to switch the polarity of the motor voltage. The way the circuit functions is modeled using switches (see figure below). If switch 1 is in the lower position, and switch 2 is in the upper position, the motor will rotate in one direction. If switch 1 is in the upper position and switch 2 is in the lower position, the motor will rotate in the other direction. If the switches are either both up or both down, the motor will not rotate. The switches on your Turtle are made of two sets of three transistors so they can be easily controlled by your computer. If pin N on the Turtle connector is at a TTL low level, transistors Q9 and Q1 are turned off. Resistor R1 pulls the base of transistor Q8 and the collector of Q1 up to nearly the supply voltage. (The actual value is set by potentiometer P1.) Since darlington Q8 is connected in



an emitter-follower configuration, its emitter voltage is approximately 1.2 volts lower than its base voltage. (This corresponds to switch 1 being in the upper position.) If, on the other hand, pin N is at a TTL high voltage, both Q1 and Q9 are turned on by the current flowing through resistors R25 and R11. Since transistor Q1 is turned on, its collector is pulled down to nearly ground level, effectively turning Q8 off and allowing Q9 to ground the motor lead. (This corresponds to switch 1 being in the lower position.) Since the other side of the circuit works in exactly the same way, the Turtle motor can be controlled in both directions using the two control lines N and K.

Horn Circuit: The horn circuit on your Turtle uses a 555 integrated circuit precision timer hooked up in an astable configuration that continuously oscillates when power is applied. When pin R on the Turtle connector is at a TTL low level, transistor Q7 is cut off, depriving the IC of power and silencing the horn. When pin R is at a high level, Q7 is saturated, the IC is supplied with a ground connection, and the horn sounds. The tone of the horn is controlled by transistor Q6. If pin B is low, Q6 is cut off and the voltage at the modulation input to the 555 is 1/2 the supply voltage. When pin B is high, Q6 is turned on, thereby placing R29 in parallel with R20. This lowers the voltage at pin 5 of the IC to about 1/4 of the supply voltage and changes the frequency of the horn.

Touch Sensors: Your Turtle sends information about its contacts with objects back to your computer with the touch sensor circuitry. When the Turtle's "shell" strikes an obstruction, it activates one or two of the touch sensor switches mounted on the circuit board. The touch sensor output lines are normally connected to a 3.6 volt

supply regulated by zener diode D13. This supply puts out a constant TTL high level in the electrically noisy Turtle environment. When a touch sensor switch is depressed, the appropriate output line is switched from the 3.6 volt supply to ground, giving a TTL low output.

DEBUGGING TIPS

If your Turtle does not operate properly check the circuit board carefully for bad solder connections, incorrectly inserted components, "solder bridges" between adjacent tracks on the circuit board, and loose wires. Make sure that the metal tabs on top of the darlington transistors are not touching each other and that no component leads (particularly the leads of the 100 ohm light resistor) or motor or pen-spade lugs are contacting the metal frame of the Turtle. If the Turtle electronics assembly looks OK make sure that the parallel interface is functioning and that the power supply is providing 12-18 volts at 2 amps.

If you still haven't found the problem, refer to the paragraphs below listing common problems and their solutions:

Symptom: Nothing on the Turtle functions.

Possible Problem and Suggested Solution: Power supply not working. Test supply with meter; repair or replace if not working.

PP & SS: Parallel interface not working. Test for data at interface with logic probe; repair or replace.

PP & SS: Power or ground connections open. Test for voltage at Turtle; if not present, correct bad connection.

PP & SS: Power to ground short circuit. If Turtle draws excessive current or blows fuses or supply does

Debugging

not work with Turtle connected, check carefully for a power to ground short.

Symptom: Interaction between pen and motor control; i.e. when motors are turned on pen operates and vice-versa.

PP & SS: Short circuit between motor or pen leads and Turtle frame or incorrectly wired pen and motor connections. Carefully check that the pen connections on the Turtle board are not wired to the motors; correct any short circuits between the motor spade lugs and the Turtle frame.

Symptom: Interaction between the "eyes" and pen or motor.

PP & SS: Almost certainly caused by the leads of the 100 ohm current limiting resistor for the LEDs being long enough to touch the motor bracket and short-circuit it. Trim leads on the resistor.

Symptom: Pen doesn't work.

PP & SS: Shorts between pen and solenoid leads and frame, bad connection in cable, pen solenoid plunger up too high in rest position, friction between pen refill and hole in bottom of solenoid. Check for shorts and bad connections. The rest (pen off) position of the solenoid plunger should be no more than about 1/2 inch above the pen down position or the magnetic flux in the solenoid will not be strong enough to pull the pen down.

Also for sale by TERRAPIN, INC.:

The TST-1, an RS-232 serial-to-parallel interface; completely self-contained and has its own power supply. Enabled by an escape character, the TST-1 can run off the same serial line as and independently of, another serial device, such as a computer terminal. Write to Terrapin for a complete description.

Price...\$150.00. When purchased with a Turtle...\$125.00

Listing and documentation for a 250-byte Turtle Interpreter and a 250-byte executive and sample programs suitable for use on a KIM-1. The programs enable the user to manually control the Turtle, to draw triangles and pentagons, to walk along a wall, to push an object around.

Price...\$10.00;

With cassette...\$15.00.

Send for these helpful TERRAPIN PUBLICATIONS:

How to Use BASIC to Control Your Turtle
(with special attention to TRS-80)... 6 pp.

Education Applications of the Terrapin Turtle... 2 pp.

Interfacing your Turtle with the Apple... 3 pp.

(When requesting publications, please add 75¢ to cover postage and handling)

Terrapin, Inc.

678 Massachusetts Avenue #205

Cambridge, MA 02139



Terrapin, Inc.

678 Massachusetts Avenue #205
Cambridge, Massachusetts 02139

ADDENDUM TO THE ASSEMBLY MANUAL

Below are changes and clarifications which have been made to the Terrapin Turtle Instruction and Assembly Manual.

Parts List (page 3-4)

Mechanical

Part	Quantity
----	-----
Motors	two
Wheels (yellow plastic)	two
Tires	two
Pen solenoid	one
Pen plunger	one
Pen spring	one
Plastic balls (Turtle toes)	two
Touch sensor pulley (yellow plastic)	one
Touch sensor ball joint	one
Allen wrench	one
Motor brackets	four
3.5" motor spacers	three
Rubber speaker mount	one
Solenoid mounting brackets	two
Ball point pen refill	one
Speaker	one
6/32 x 1/4" screws	twelve
Lock washers for 6/32 screws	sixteen
6/32 x 1" screws	eight
6/32 nuts	two
Bottom plate	one
10/32 dome screw	one
10/32 nylon dome washer	one
6/32 nylon washers	four
1/4-20 x 1/2" screws	two
1/4 lock washers	two
10/32 nut	one
#10 star washer	one
Allen (or set) screw	one
4/40 x 1/4" screws	two
#4 flat brass washer	one

Electrical

Part	Quantity
Printed circuit board	one
1K resistors	eighteen
510 ohm resistors	two
100 ohm resistors	two
15K resistors	two
51K resistors	two
6.9K resistors	four
IN4000 diodes	nine
3.6 zener diode (IN748)	one
LED's	two
500 ohm POTS	four
.1 f capacitor	one
555 IC	one
GED40c4 darlington	nine
2n2222 transistors	seven
Micro switches	four
Solderless spade connectors	six
40" red hookup wire	one
Amphenol male connector	one
Amphenol female connector	one
Heat shrink tubing	one
Ten foot cable	one

Resistors (page 5-6)

Resistor Value	Quantity
51K resistors	two
15K resistors	two
6.9K resistors	four
1K resistors	eighteen
510 ohm resistors	two
100 ohm resistors	two

Four of the six 15K resistors have been replaced with 6.9K resistors. These four are all in the motor speed adjustment circuit and have been changed to increase the adjustment range. The resistors to be replaced are the four marked 15K nearest the speed adjustment pots and closest to the center of the board. (Refer to the board or to the diagram on page 4.) Two are located behind the FD speed adjustment pot on the turtle's left side and the other pair is located to the left of the FD pot on the turtle's right side.

The motor circuit diagram is on page 36. The four 15K resistors (R5,R6,R7,R8) should all be 6.9K resistors.

Diodes (page 7)

In many turtle kits the IN4001 diodes are provided in a black epoxy package with a silver band at one end to indicate orientation as explained on page 7 of the manual. The zener diode is in a glass package marked as described in the manual, but it is smaller and has IN748 printed around the middle of the case.

Capacitor (page 14)

The capacitor may come in a variety of packages: the disc shape described in the manual, a rectangular plastic block or a trapezoidal block.



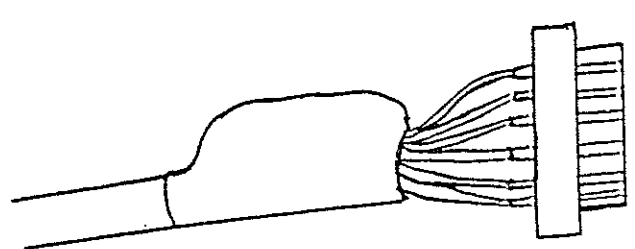
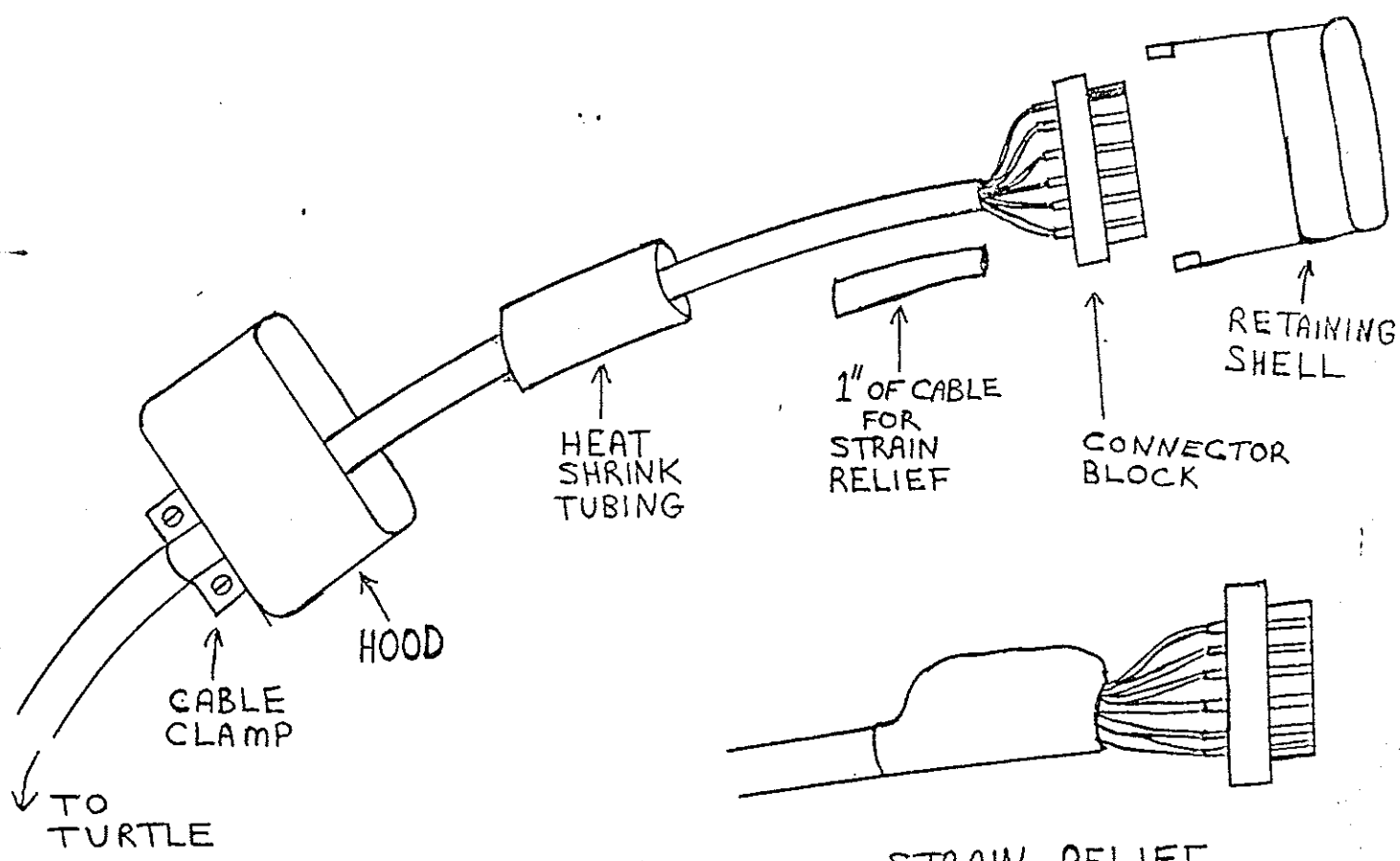
Cable and Connector (page 15-19)

The Molex connectors have been changed to Amphenol Ribbon Connectors for greater reliability and ease of use. A male and female connector are included. The pins are fixed in the Amphenol connector instead of coming separately, as they did with the Molex connector.

The male Amphenol cable connector consists of several parts (see diagram): the blue plastic connector block, a hood with cable clamp that slips onto the cable side of the connector block, the retaining shell that holds the connector block in the hood, and two very small screws (be careful when opening the bag) that hold the hood and retaining shell together. Terrapin includes a short length of heat shrinkable tubing for attaching a small piece of cable to the main cable just behind the connector block to serve as a strain relief.

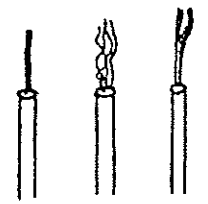
To prepare the cable for soldering to the connector, cut off approximately 1 inch of cable and save for use as a strain relief. Then cut and strip the cable as described in the manual on page 17. Tin each wire (heat the end and allow it to "wick" up a small amount of solder) immediately after stripping to avoid fraying the ends before the tinning can be done. If you do fray the end, before you tin it "point" the end by twisting it as you would a thread.

If you have already soldered one end of the cable to the turtle, be sure to slip first the hood (with the cable clamp towards the turtle)



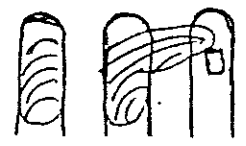
STRAIN RELIEF
AFTER
HEATING TUBING

TINNED WIRE



RIGHT WRONG

TERMINAL CONNECTIONS



RIGHT WRONG
AVOID
SOLDER BRIDGES



PUSH WIRE ALL
THE WAY INTO
TERMINAL

11001
+ btb

and then the piece of heat shrink onto the cable BEFORE starting to solder the cable to the connector block.

The terminals on the connector block should also be tinned before soldering the wires to the terminal. It is best to heat the terminal by touching the soldering iron to the side of the terminal facing the inside of the two rows and apply a small amount of solder to the groove. To connect the wire to the terminals, heat the terminal again in the same way. When the solder melts, slip the tinned wire into the groove, remove the soldering iron and hold the wire in place until cool. The pins should not be crimped as mentioned in the manual for the Molex connector. FOLLOW THE CONNECTOR PATTERN BELOW.

Amphenol Pin Number	Wire Color	Turtle Hole	Function
1	Black	A	Lights (L)
2	Brown	B	Horn hi/lo (T)
3	Red	C	Left Touch (LT)
4	Orange	D	Left Motor 1 (LM1)
5	Yellow	E	Power
6	Green	F	Pen (P)
7	Blue	G	Left Motor 2 (LM2)
8	Purple	H	Ground
9	Grey	I	Ground
10	White	J	Back Touch (BT)
11	White-Black	K	Right Motor 2 (RM2)
12	White-Brown	L	Power
13	White-Red	M	Ground
14	White-Orange	N	Right Motor 1 (RM1)
15	White-Yellow	O	Right Touch (RT)
16	White-Green	P	Front Touch (FT)
17	White-Blue	Q	Horn on/off (H)
18	White-Purple	R	for user



Pennan

A strain relief for the cable must be formed. The cable is too small for the clamp on the hood to work. Therefore, put the one inch section of cable cut off earlier alongside the cable, even with the end of the cable insulation, and slip the heat shrink tubing over it. The tubing can be shrunk over the double section of cable with heat from a match or the soldering iron. Care should be taken not to heat excessively the cable underneath, as the insulation could melt and allow wires in the cable to short.

The hood and retaining shell can now be slipped onto the connector block and the position of the strain relief can be adjusted. The cable clamp is then tightened over the double thickness of cable and the shell and hood are screwed together.

If you are making your own interface, the female Amphenol connector can be attached to your interface with flat cable or a length of turtle cable. You should determine how long a piece you need and cut it off the cable before soldering the cable to the turtle. Soldering the cable to the female connector is the same as soldering the cable to the cable connector. Follow the wiring pattern above.

Motor, Speaker and Pen Solenoid Hookup to PC Board (page 20)

Cut the hookup wire into eight pieces. Two should be about an inch longer than the others. Use the two longer pieces for hooking up the speaker. Do not put spade terminals on the two longer wires. Place the eight wires as shown in the diagram on page 28. The USER hole should be ignored. The two GROUND holes are explained in the following paragraph in the manual.

Pen Solenoid (page 21-23)

1. Pen Float Modification (see Fig. 1 below)

Cut an approximately 1/4" length of the retraction spring included in the pen refill package and slide it onto the pen cartridge on the bottom of the pen cartridge/solenoid plunger assembly before inserting the assembly into the pen solenoid. (It may be necessary to increase the length of the pen cartridge projecting from the plunger by 1/8" to 1/4" to obtain adequate writing pressure.) This change and the next one can be seen by comparing the diagrams below with the ones on page 23 in the manual.

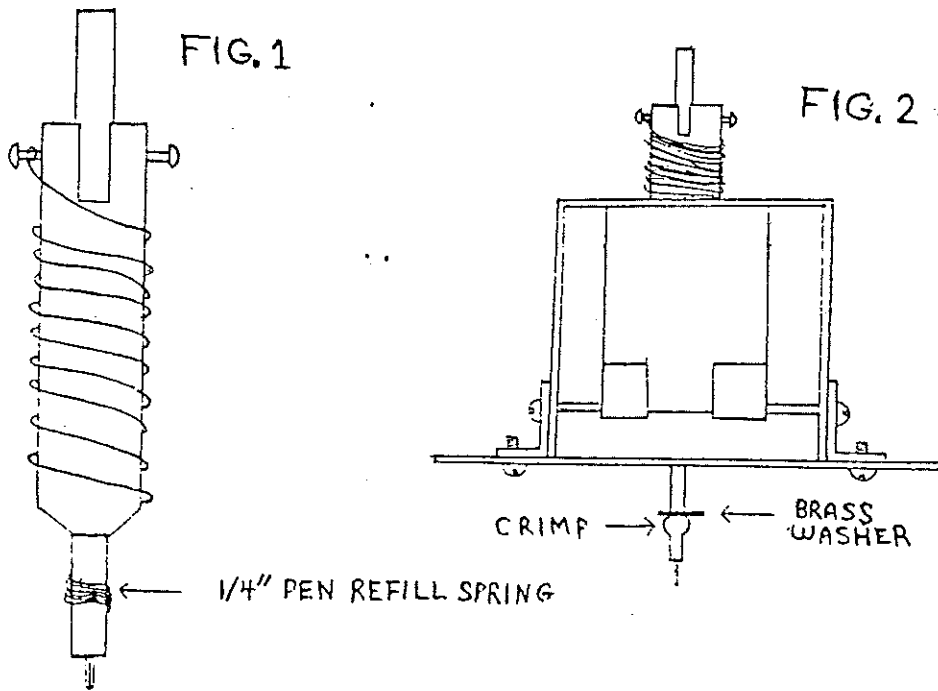
2. Anti-Stick Modification (see Fig. 2 below)

The #4 flat brass washer is slipped onto the pen cartridge projecting from the bottom of the bottom plate before crimping the pen. This modification prevents the pen from getting stuck in the up position when the shoulders of the crimp are jammed into the hole in the bottom of the solenoid as the pen is retracted.

Also, the second sentence on page 23 makes more sense if it is changed to: To prevent the plunger assembly from retracting too far, use pliers to gently crimp the pen beneath the bottom plate.

Turtle Toes and Speaker Mount (page 24)

The speaker mount should be placed on the side opposite the solenoid spade terminals.



Right Motor Assembly (page 24-26)

The two diagrams on page 25 are actually the right motor assembly, not the left. Also, for easier assembly, first glue the nylon washers to the top of the brackets or the bottom of the PC board.

Horn Circuit (page 34)

The horn circuit diagram in the manual is incorrect and has been duplicated below. The resistor near the speaker was in the wrong place, and a 15K resistor was missing near transistor Q6.

