CODEN:LUTFD2/(TFRT-5271)/1-086/(1982)

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
Document number
CODEN:LUTFD2/(TFRT-5271)/1-086/(1982)

Author(s)
Gunnar Sandin

Tord Wullt

Supervisor
LARS NIELSEN
Sponsoring organization

Title and subtitle

Experiment med datorstyrd kännande robot.

Abstract

Electronics and software for control of a two-wheeled, touchsensing vehicle, known as the Turtle.

Program applications.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
SWEDISH

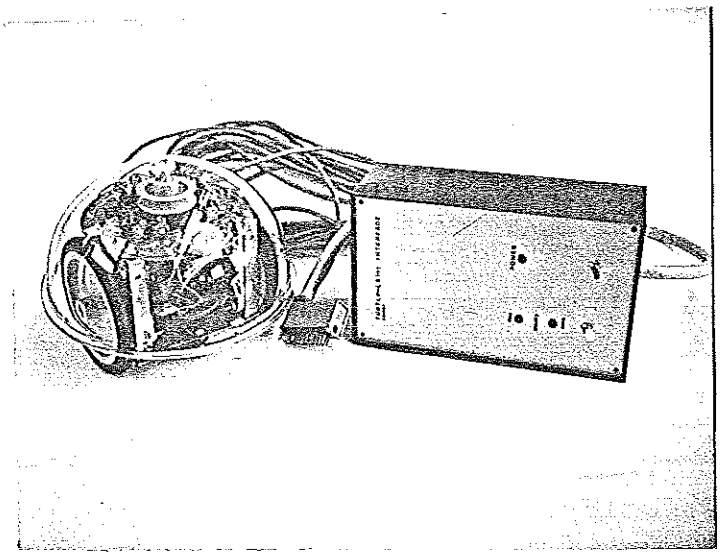
Number of pages
86

Recipient's notes

Security classification

Innehållsförteckning.

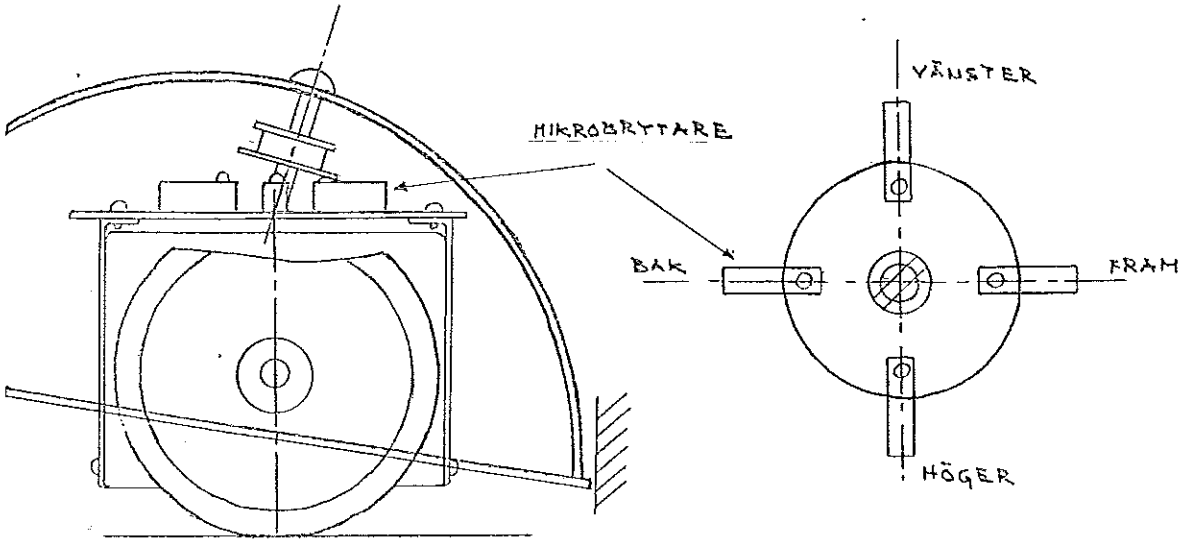
1.Beskrivning av Turtle.



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

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.

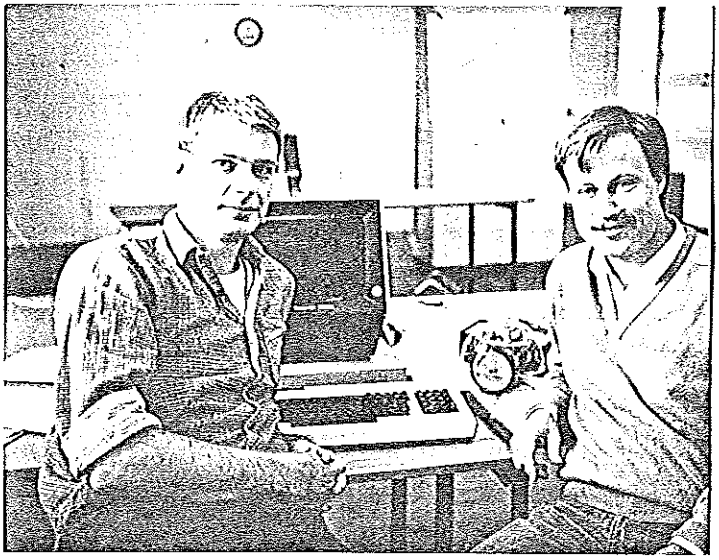
Man skulle kunna utföra lokal hastighetsåterkoppling på motorerna ute på Turtle och genom att mäta tiden få information om positionen på golvet.

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

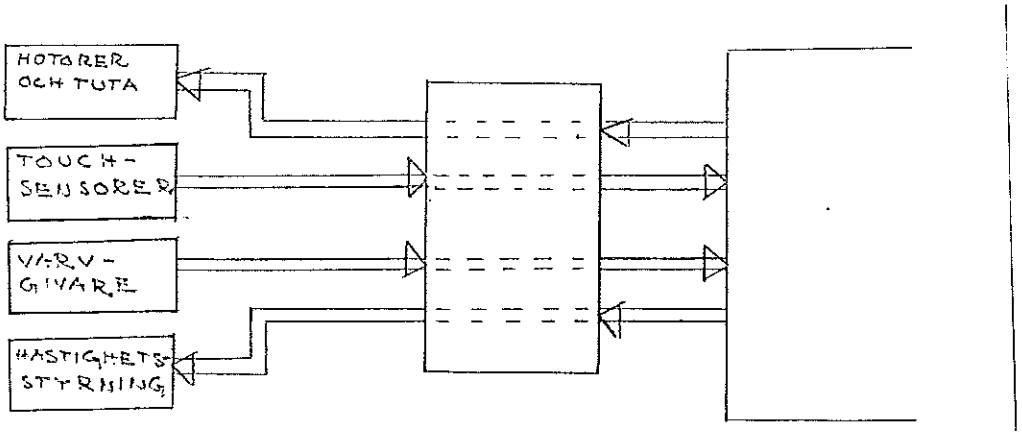
3.1 Inledning.

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

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

Från Turtle kunde vi få signaler från de fyra touchsensorerna (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 nivåbytena 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.



Turtle ^ Interface ^ Dator

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. Överföringen av signalerna är alltså ej kodad.

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

3.2 Styrsignaler till motorer och tuta.

Turtle's krav på signalerna:

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

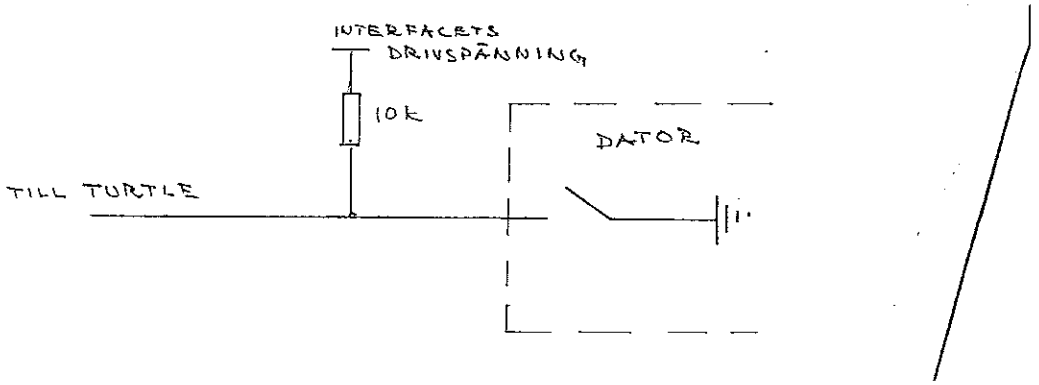
0 volt ------> OFF.

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:



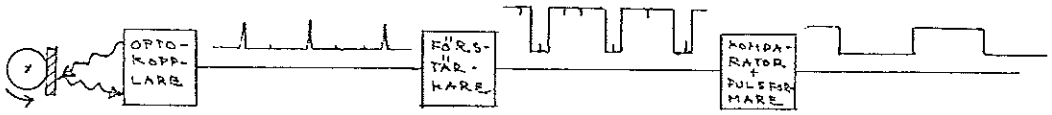
3.3 Signaler från touchsensorerna.

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.

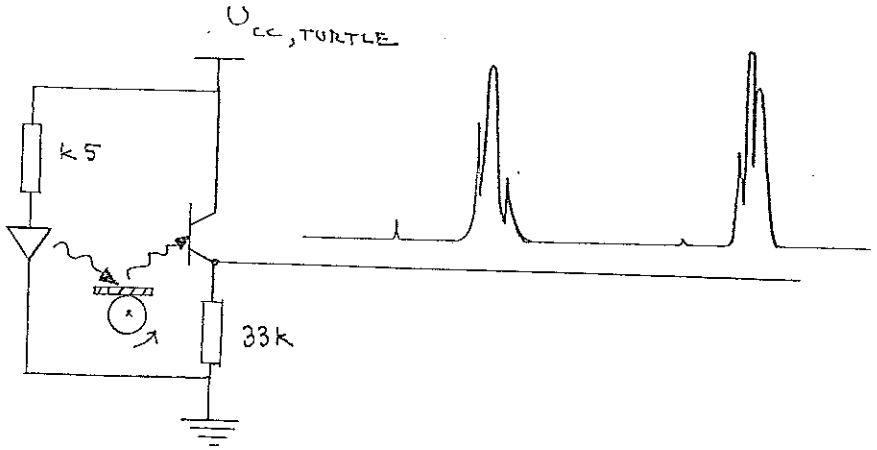
3.4 Varvgivare.

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



Optokopplare.

Följande koppling har använts:



1 Hjulvarv <===> 300 motorvarv.
1 Motorvarv <===> ca.3.6 mm rörelse på golvet.

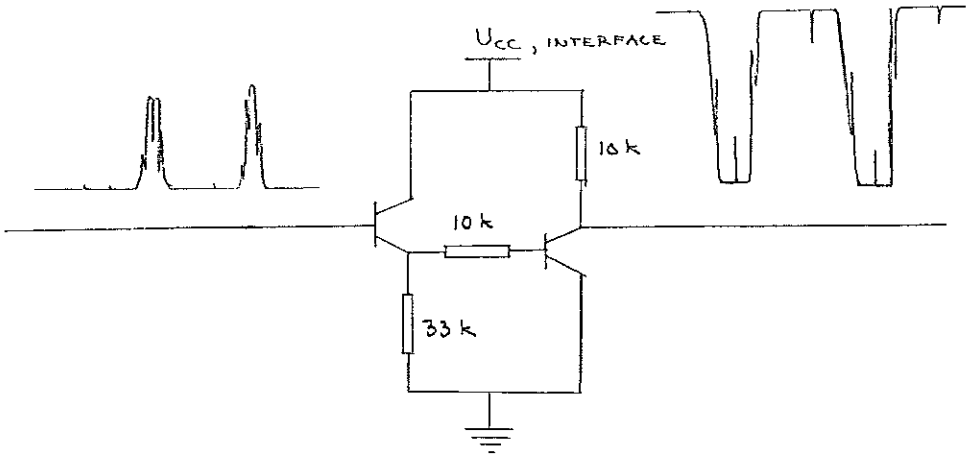
Drivspänning	varvtal
12 volt	40 varv per sekund
18 volt	60 varv per sekund

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 Reglerteknikś 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.

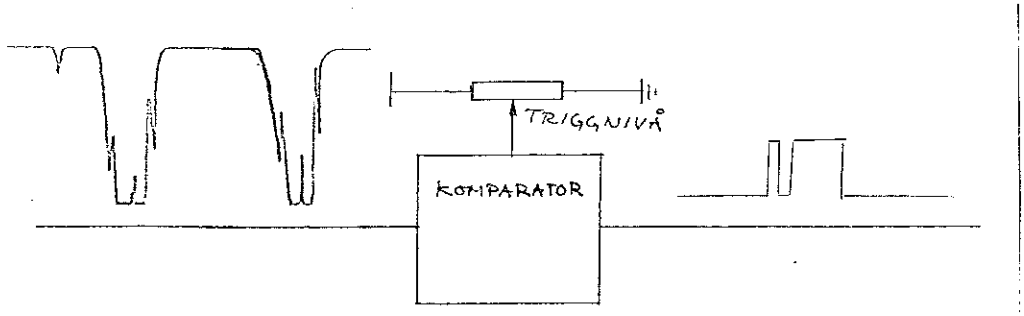
För att säkrare kunna trigga på pulsen från optokopplarna, och för att en försämring av reflektorerna inte skall påverka tillförlitligheten, har vi lagt in en switchförstärkare efter optokopplarna.

schema:

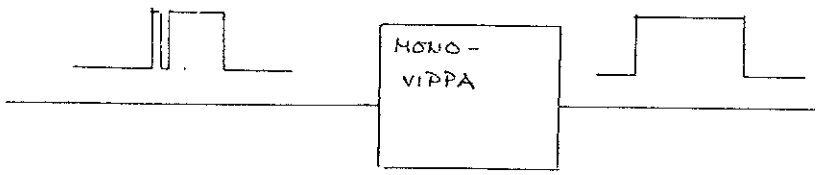


Komparator och pulsformare.

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

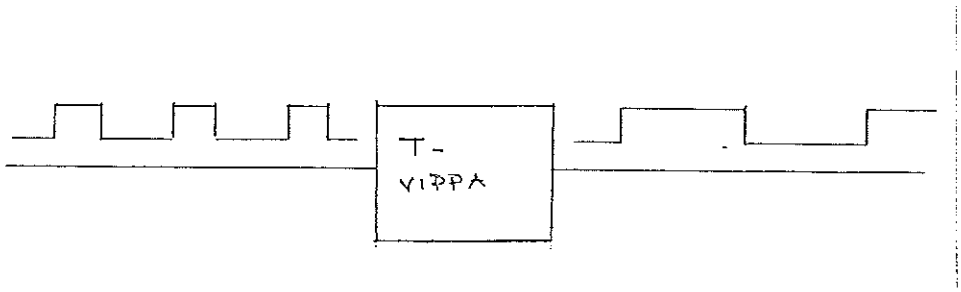


Därefter följer en monostabil vippa med en pulslängd som är något längre än pulsen från komparatorn.



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.

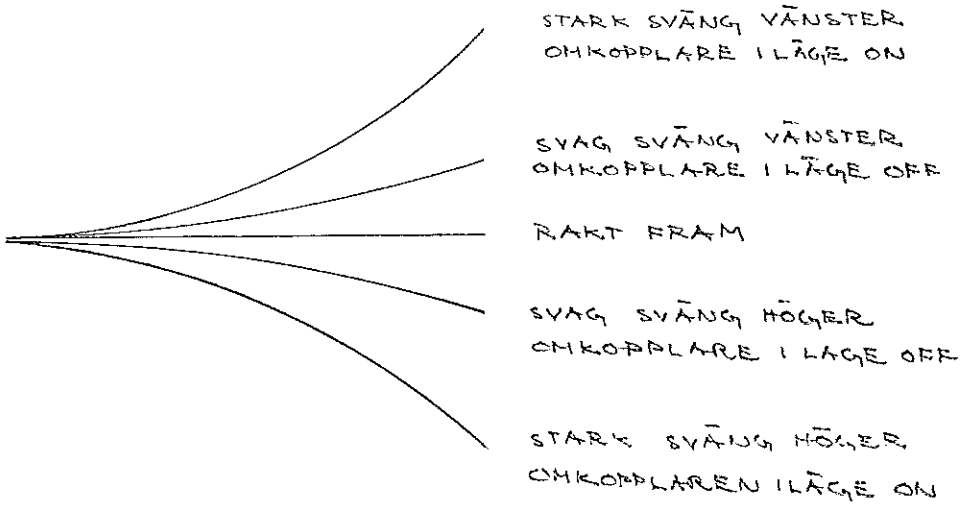
Vi antar att det bland annat beror på temperaturberoende hos motorerna och anordningen som skall ge motorerna konstant ström,samt parametervariation hos växellådan.

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

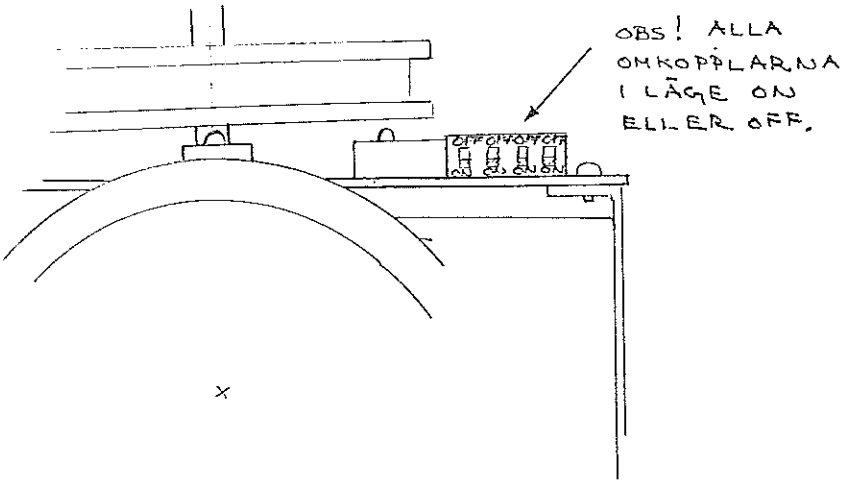
Detta ger tre olika rörelsetillstånd:

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

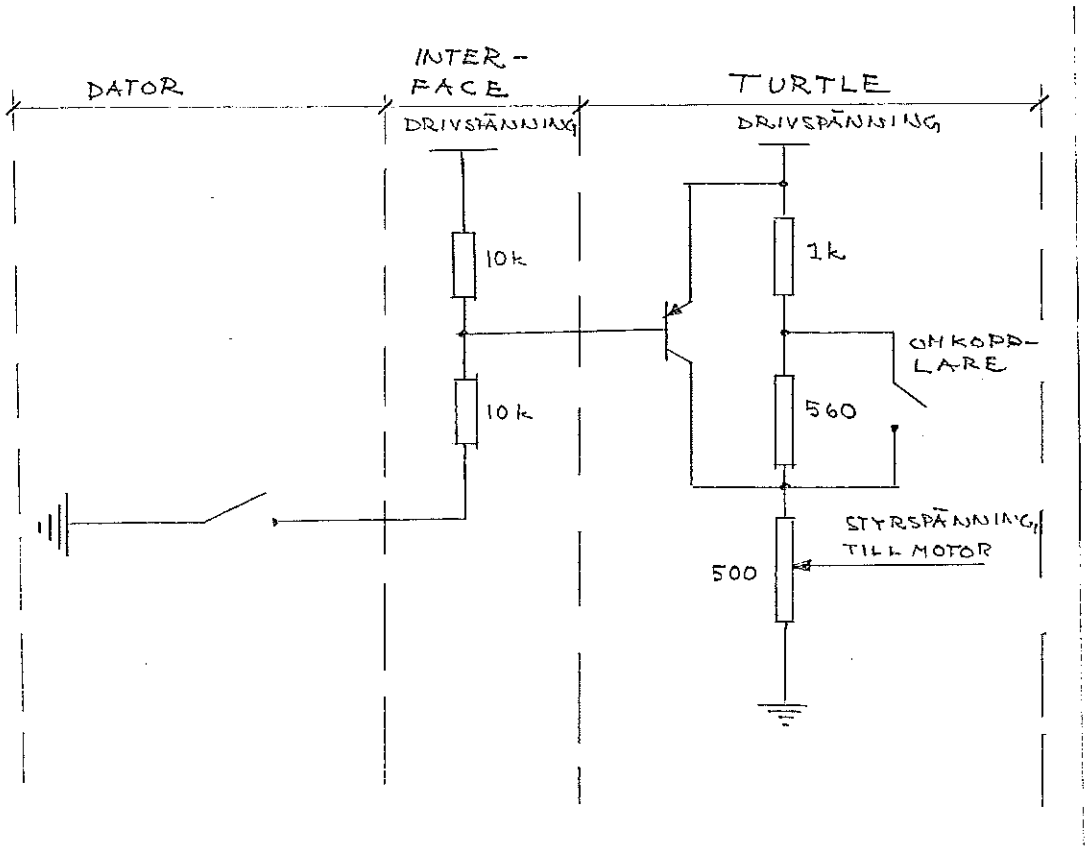
Med en omkopplare som sitter på Turtle kan man dessutom välja hur mycket de två hastigheterna skall skilja sig åt. Vi kan alltså välja hur mycket Turtle skall svänga höger respektive vänster.



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.

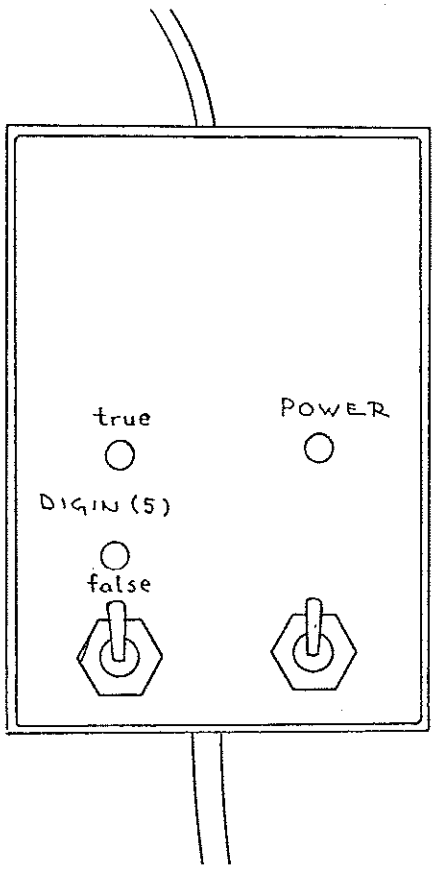
Från interfacet utgår två sladdar, en till datorn och en till Turtle.

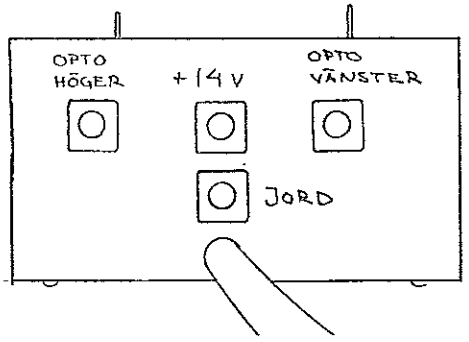
Två omkopplare finns. Med den ena kan man stänga av spänningen till elektronik och Turtle. Den andra sätter datorns ingång digital-in(5) sann eller falsk. Denna ingång svarar mot mjukvarufunktionen Bitin(0,5) och kan användas i programmen.

På baksidan av interfacet sitter fyra honkontakter 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.





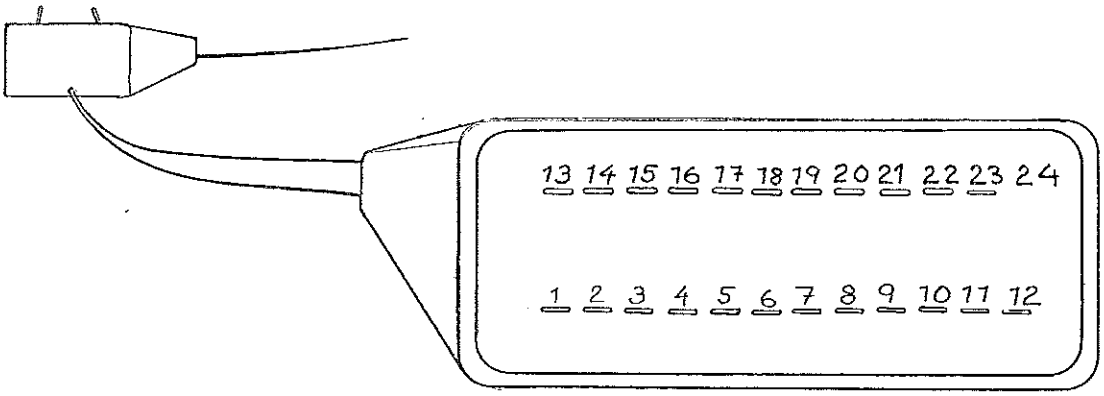
Turtle är förbunden med interfacet via en 10 meter lång sladd. Sladden har 15 ledare och dessa används enligt följande:

Funktion	Färg
Höger motor fram	Röd-grön
Höger motor back	Röd-gul
Vänster motor fram	Mörkgrön
Vänster motor back	Vit
Tuta	Skär
Hastighetsstyrning höger	Grå
Hastighetsstyrning vänster	Blå-röd
Mikrobrytare fram	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.



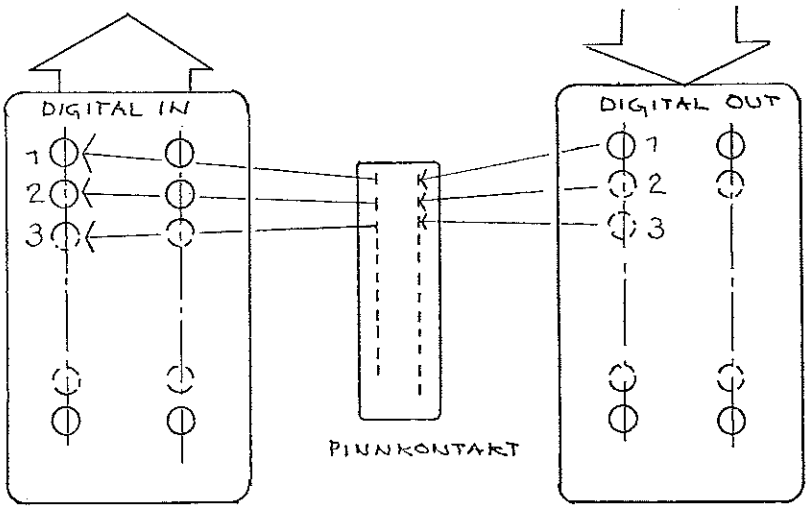
Kontaktens pinnar (obs alla används inte) överför följande signaler:

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

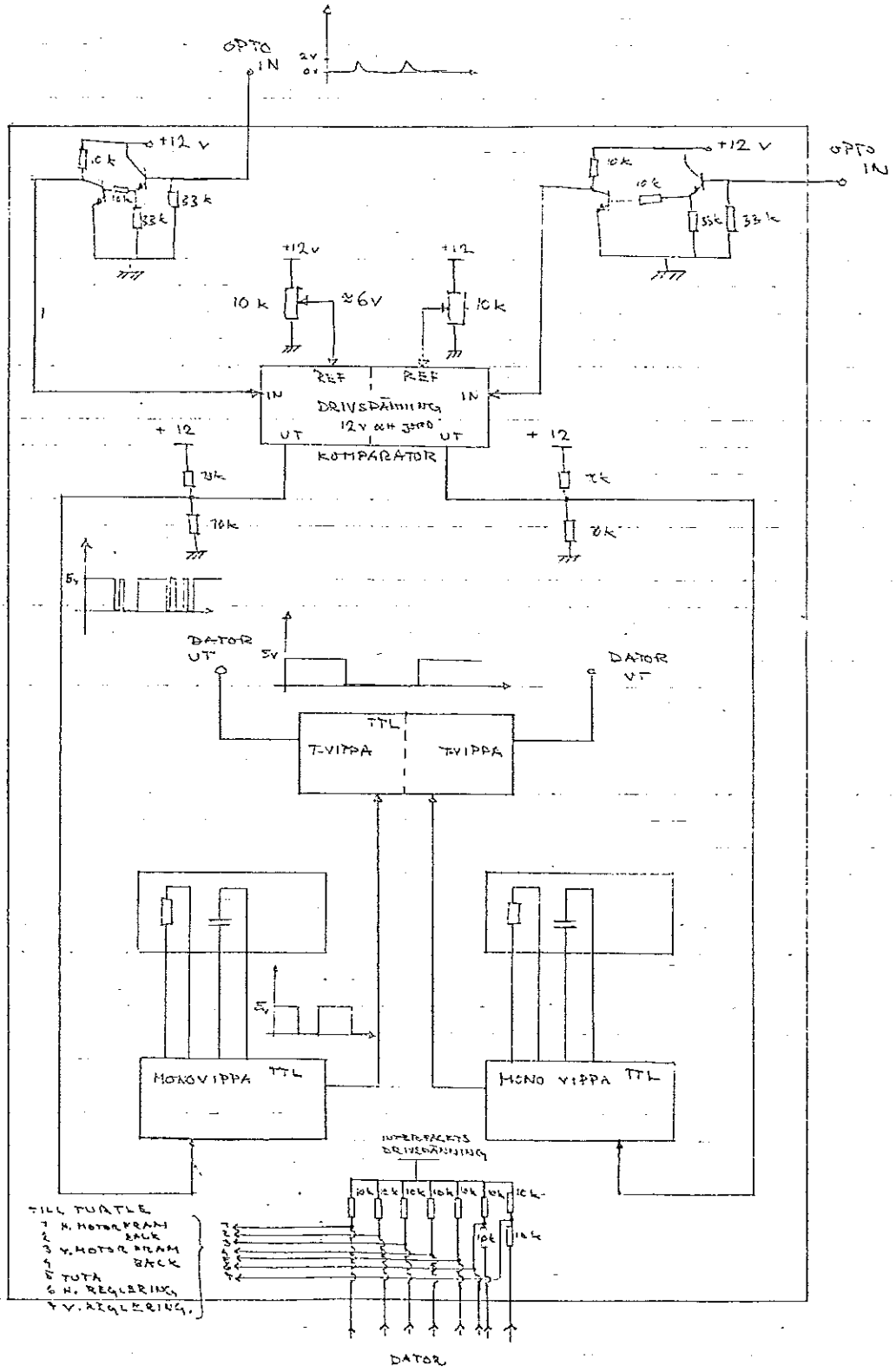
18	H.varvgivare	6
19	V.varvgivare	7
20	-	-
21	-	-
22	-	-
23	-	-
24	-	-

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.

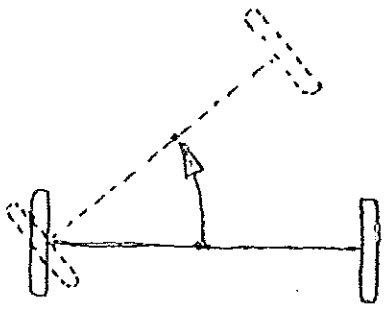
Följande proceduranrop kan göras i användarens Pascal-program:

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

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 ts, 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 konkurerande 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.

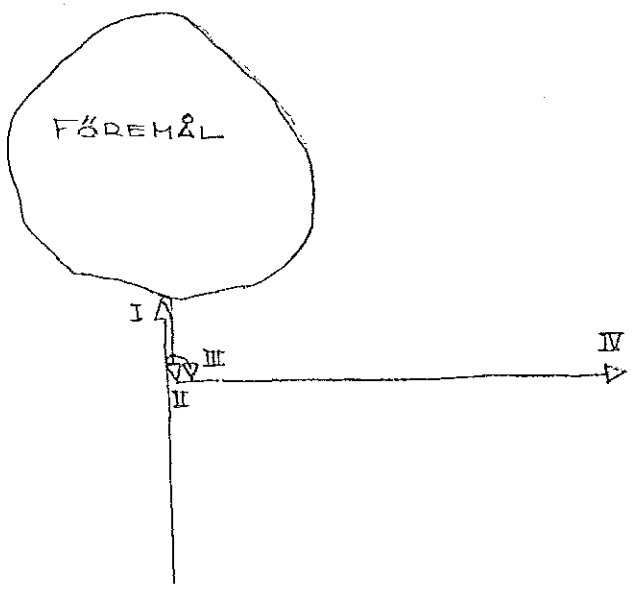
Vi illustrerar användningen av primitiverna med ett enkelt programexempel.

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:

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

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;


      dammi,
      dommy1,
      dommy2          :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;
      kl:=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;

procedure PutInMotion;                   {Satter igang de bada processerna}
begin
   SetOutsFalse;
   MotionToOuts;
   moving:=true;
   touchenabled:=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
         kl:=kl+1;
         dommy2:=false;
      end;
   if not dommy2 and not Bitin(0,7) then
      begin
         kl:=kl+1;
         dommy2:=true;
      end;
   with ts do
      begin
         RDIST:=kr;                           {Olika hjuldiameter korrigeras }
         LDIST:=round(WHEELFACTOR*kl);        {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                                         {ar 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;
      SignTo8itouts;
      Bitout(0,5,true);                          { tystar tutan}
      initsem(mutex,1);
      initsem(touchtimer,1);
      initsem(ready,0);
      initsem(move,0);
      initsem(user,1);
      createprocess(TouchdisableTimer,1000);
      createprocess(Control,2000);
end;
```

5. Tillämpningsexempel.

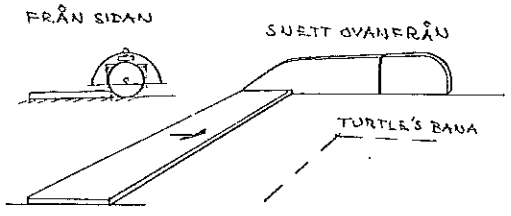
5.1 Inledning.

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.

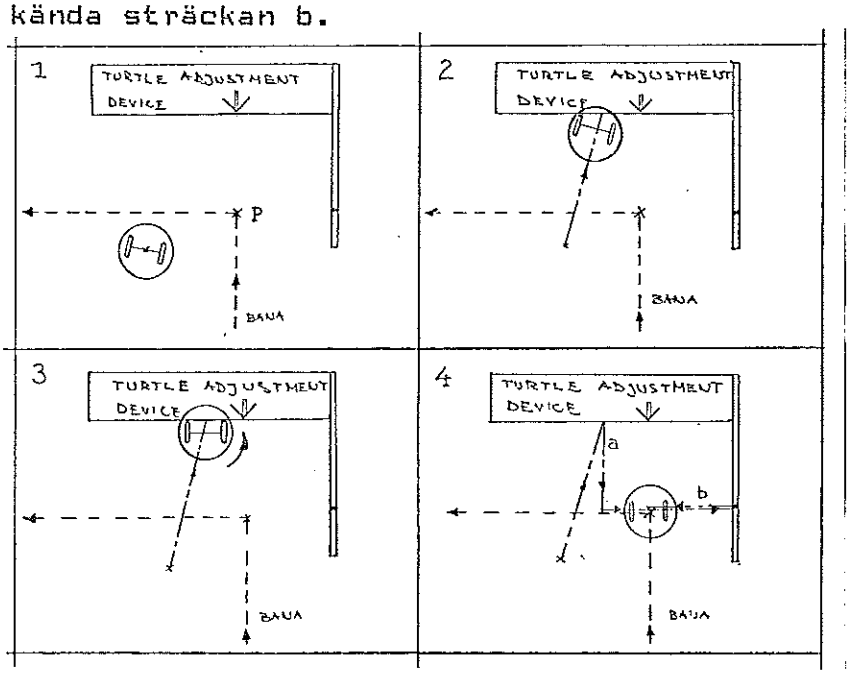
Under sin vandring på golvet tappar Turtle normalt sin position och riktning.De vanligaste orsakerna är slirning och icke-perfekta svängar.

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 lage m.hj.a}
begin                      {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;
```

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 mad primitiverna i filen TURTLE.

```
procedure TurtleOpcom;                    {Valjer terminal- eller program-  }
var com   :char;                            {korning av Turtle.               }
    dist  :integer;
    return:boolean;
begin
   writeln('TURTLE OPCOM');
   writeln('*********************************************************************');
   writeln('Turtle Operator Comunication 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         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;
```

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 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:

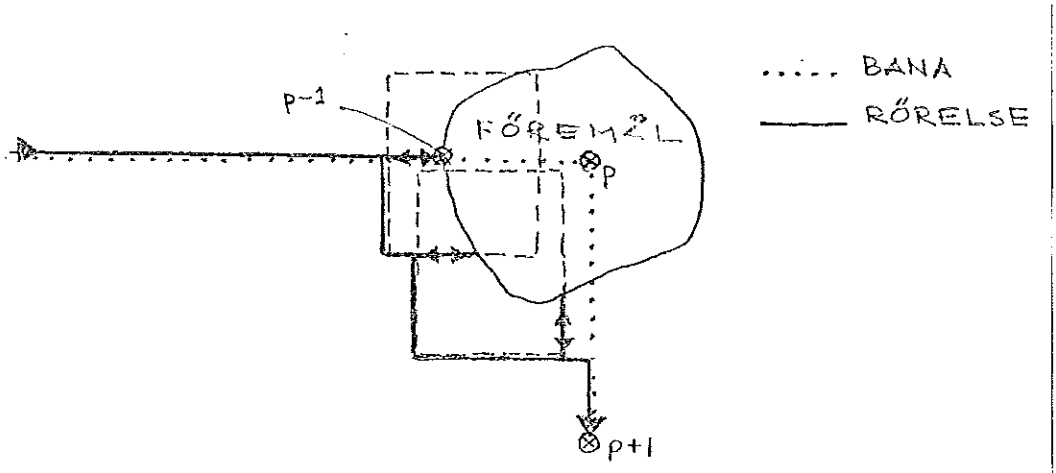
En bana lagras in från terminal.Några punkter kan vara justeringspunkter(se 5.3). Banan läses in i en matris med tre kolonner(x-koord, y-koord, siffra som anger om denna punkt är

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 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;
```

```
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,,]}
                                                    {till coor[pek+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;
```

```
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='l' 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;
```

```
procedure AvoidFirst;                          {Anropas nar Turtle for forsta     }
var  peksave,                                  {gangnen far fronttouch.           }
     savex,savey :integer;                     {Byter tillfalligt ut coor[pek-1,,] }
begin                                          {mot aktuell punkt for att korrekt }
   peksave:=pek-1;                             {sokning efter fortsatt bana skall }
   savex:=coor[peksave,1];                     {kunna ske.                        }
   savey:=coor[peksave,2];
   coor[peksave,1]:=x;
   coor[peksave,2]:=y;
   AVOID;
   coor[peksave,1]:=savex;
   coor[peksave,2]:=savey;
end;

procedure ChooseDirection;                        {Valjer hoger- eller vanster-   }
begin                                             {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;                         {Justerar vinkel och lage m.hj.a}
begin                                             {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;                 {Kor till punkterna i matrisen coor       }
var  ref,                              {och undviker foremal.                    }
     i,k    :integer;
begin
   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;
```

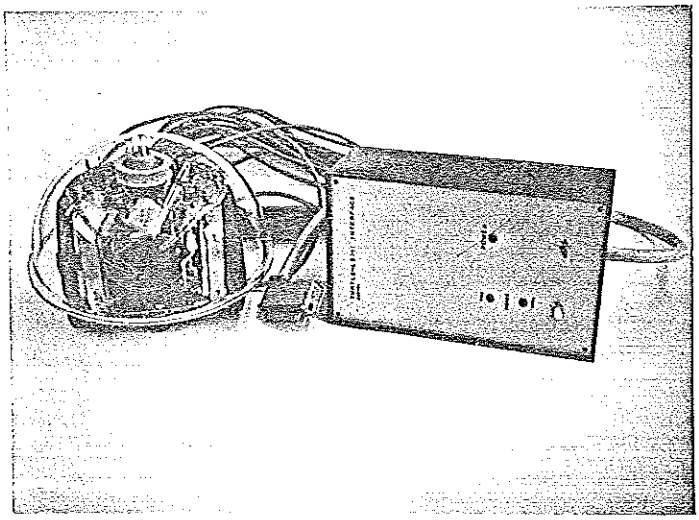
```
procedure LoadTrackandRun;              {Laser in punkter i matrisen coor och   }
var q      :char;                       {anropar FollowTrack.                   }
    datain :boolean;
    i,k,
    p      :integer;
begin
   writeln('New Track ?');
   readln(q);
   if q = 'y' then
      begin
         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 Comunication 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.
```

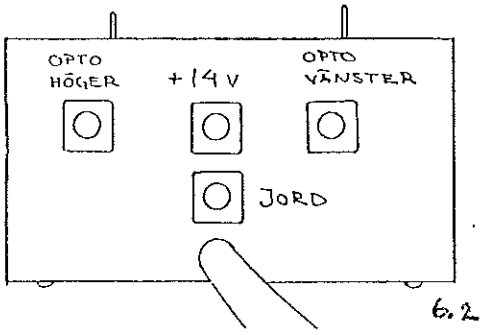
6. Bruksanvisning.



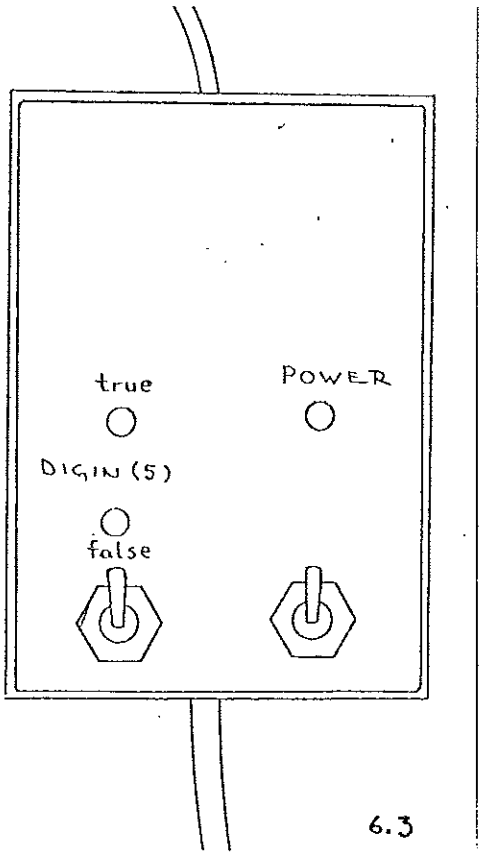
6.1

Interface.

Anslut interface till 14 Volt (1.5 A) och jord. Honkontakter finns på baksidan av interface (fig 6.2).
Som bekräftelse på att spänning är ansluten, lyser någon av DIGIN 5 indikatorerna. (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 operatörskommunikation 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 länkningen 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. Nedbromsningströgheten 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 nedbromsningsosäkerhet 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.

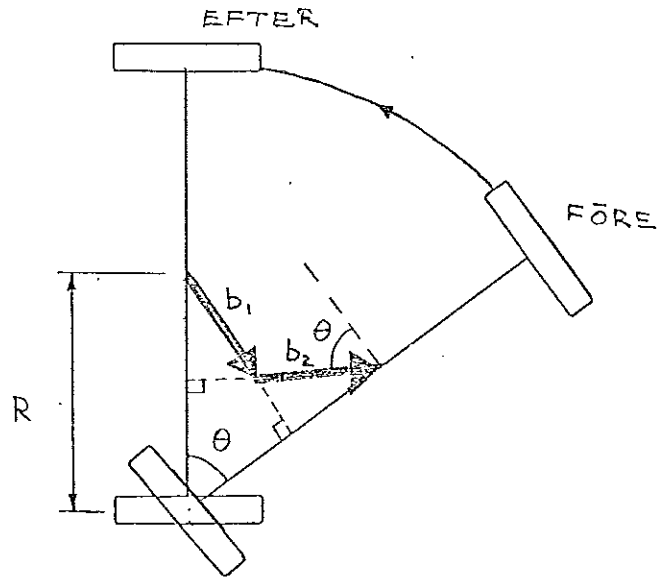
Problemen här är av samma natur som vid centrumsväng. De är emellertid mindre eftersom:

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 pga. 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

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

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

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

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 kommenderades 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 ≈ 2 pulser för centrumsvängar och C ≈ 1 puls för enhjulssvängar.

Den praktiska betydelsen av C är att om man utan korrektioner 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.

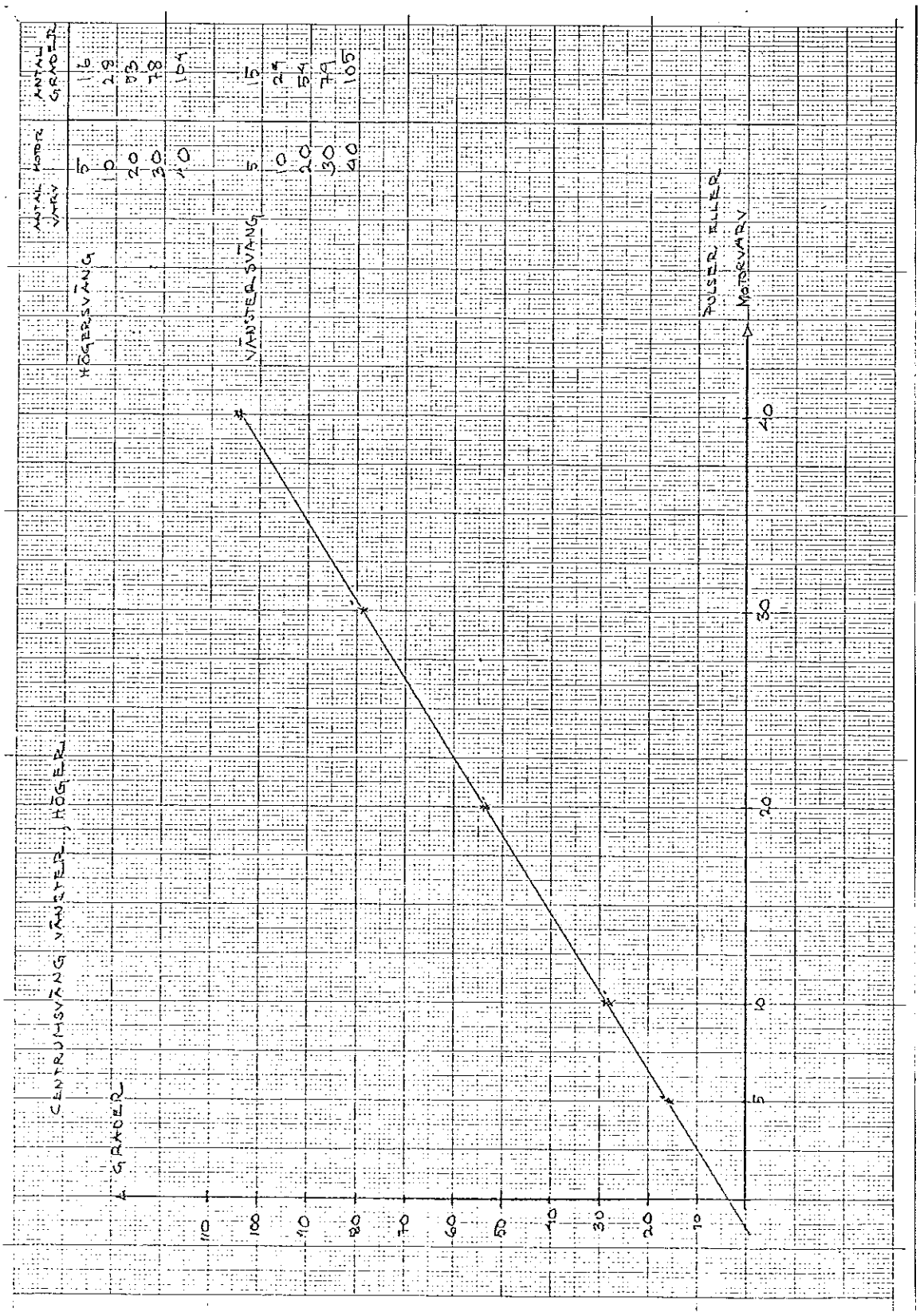
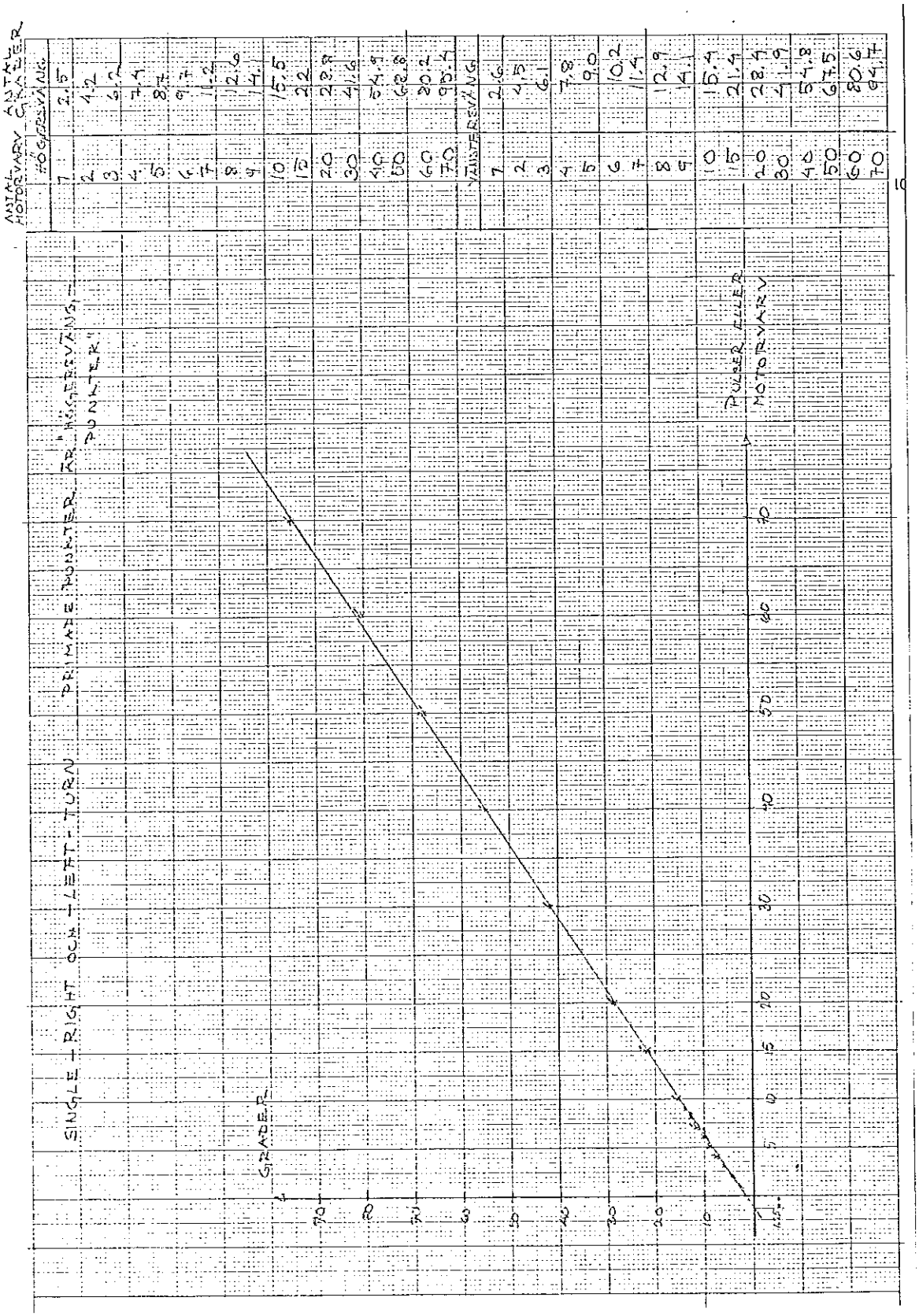

ANTAL GRADER
ANTAL HELA VARV
HÖGERSVÄNG
VÄNSTERSVÄNG
PULSER KLLED
MOTSVARAV
CENTRUMSVÄNG VÄNSTER, HÖGER
GRADER

DIAGRAM 2.


ANTAL MOTORVARV
ANTAL GRADER
HÖGERSVÄNG
VÄNSTERSVÄNG
SINGLE — RIGHT OCH LEFT - TURN
PRIMÄR POINTER
PRIMARE POINTER
POINTER
GRADER
PULLER
MOTORVARV

Appendix 3

Terrapin, Inc.

678 Massachusetts Avenue #205
Cambridge, Massachusetts 02139
617-492-8816

EDUCATIONAL APPLICATIONS OF THE TERRAPIN™ 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 Terrapin™ 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 touch	front touch	right touch	left 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 Control	Horn	Pen	Lights	LM2	LM1	RM2	RM1

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:

Turtle State	Binary Control Data	Decimal Equivalent
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 microcomputer 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:

Turtle Function	Bit	Place Value
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
200 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):

Turtle Motion	Binary Turtle State	Decimal Equivalent
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* (LTCONS)
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 (LTCONS) 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. darlingtons
   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

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.

Mechanical

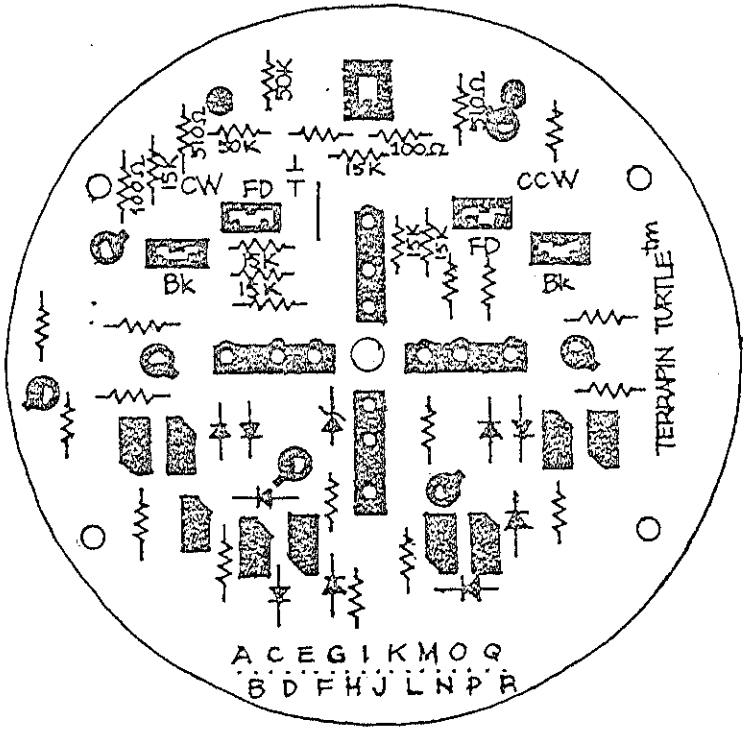
Part	Quantity
Motors	two
Wheels	two
Tires	two
Pen solenoid	one
Pen plunger	one
Pen spring	one
Plastic balls (Turtle toes)	two
Touch sensor pulley	one
Touch sensor ball joint	one
Allen wrench	one
Motor brackets	four
3-1/2" motor spacers	three
Rubber speaker mount	one
solenoid mounting brackets	two
Ball 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	one
1K POTS	four
.1uf capacitors	one
555 IC	one
GED40c4 darlingtons	nine
2n2222 transistors	seven
Micro switches	four
Male molex pins	15
Female molex pins	15
Male molex connector	one
Female molex connector	one
Ten foot cable	one

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.

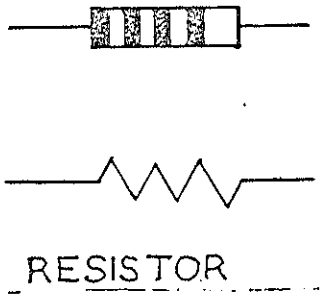
The digits are indicated by colors according to this chart:

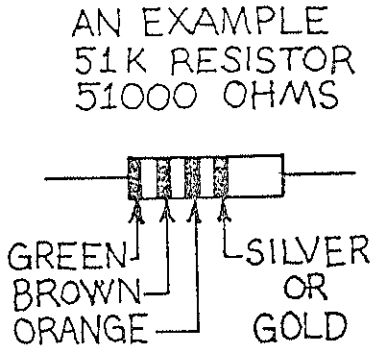
0 black	5 green
1 brown	6 blue
2 red	7 violet
3 orange	8 grey
4 yellow	9 white

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

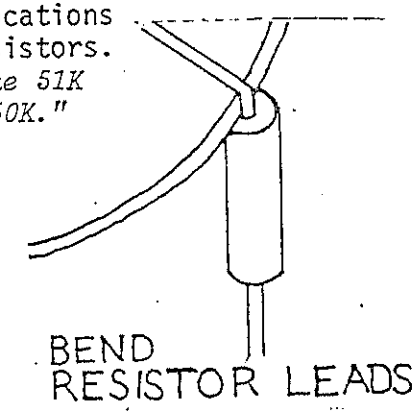
Note: There are three resistors (~~W~~) together, the top two labelled 15K, the bottom one unlabelled (following the convention of putting labels below symbols). The bottom one is the 1K resistor.

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





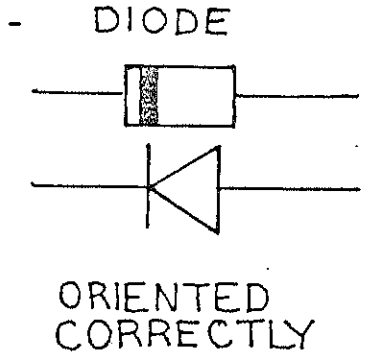
Resistor values can be identified by the colored bands painted on them (see box pg. 5). Bend the leads of the resistors so that they are spaced properly for insertion into the board. *Proper spacing is easy to achieve by bending the leads over the edge of the pc board as illustrated. The board is just the thickness required between the resistor body and the right angle bend in the lead.* Each resistor location is marked on the board. Resistor locations are marked with values on the board. Unmarked locations require 1K (1000 ohm) resistors. *Note: The location for the 51K resistor may be marked "50K."*

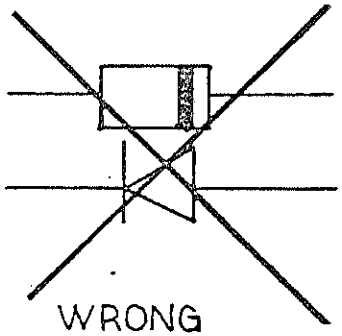


Resistors can be inserted into their board location in either direction; it does not matter which way the bands are oriented. Insert the resistors into the board and solder them.

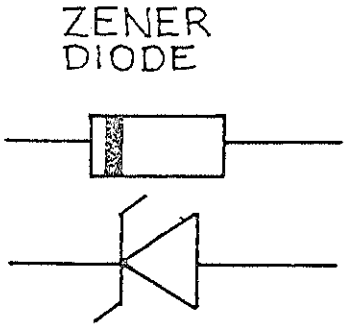
Diodes: There are nine diodes and one zener diode. *Be sure not to confuse the ordinary power diodes and the zener diode.* The zener is longer than the power diodes.

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.





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.



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.

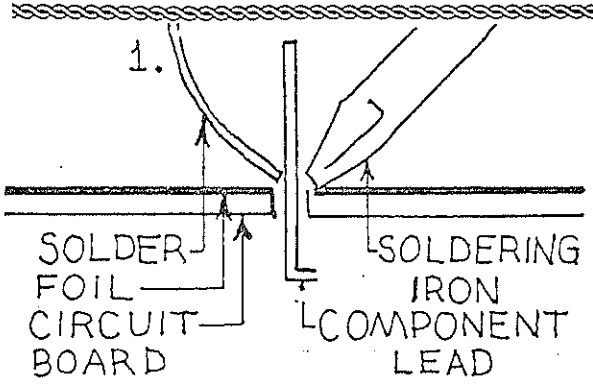
Exercise extreme care in soldering around the Turtle pc board. Forming a solder "bridge" between two foil areas of the board must be prevented for proper assembly and operation of the Turtle. A "bridge" is formed when two different foil lines are joined by solder. This may be the result of too much solder, 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.

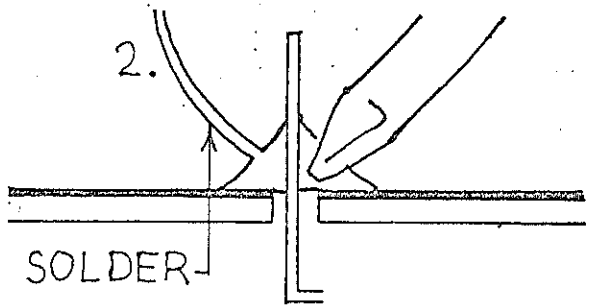
Use the minimum amount of solder necessary for a good connection. Should a bridge develop, turn the Turtle pc board upside down and apply the iron to the bridge. The solder will drip down onto the tip, thereby destroying the bridge.

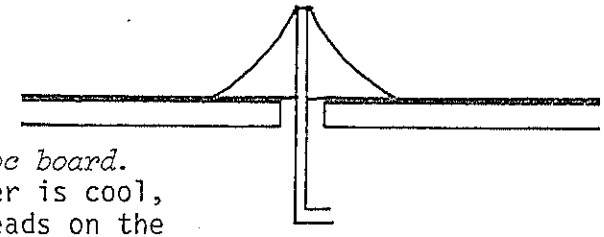
To solder: Once the electrical component is properly placed in the pc board, turn it so the lead is up. Place the soldering iron tip at 45° to both lead and pc board foil. Heat for approximately three seconds. Apply solder to the connection on *the other side* from the soldering iron. The foil and component 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







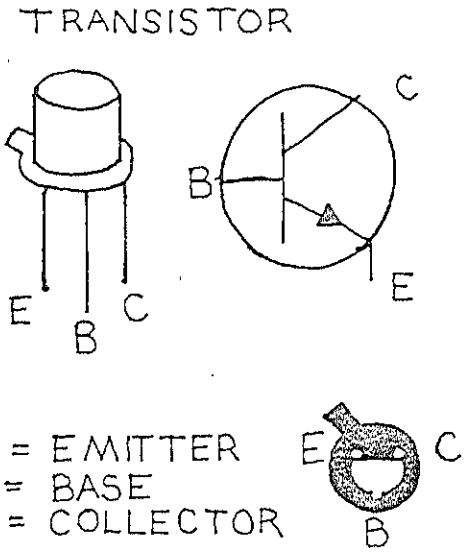
close to the pc board. When the solder is cool, cut off the leads on the bottom of the board close to the board. Save one excess lead for use as a jumper.

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 zener 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.

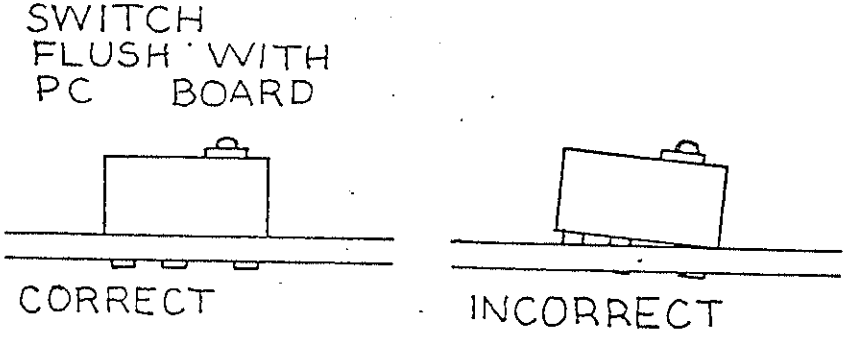
Transistors: Find the seven transistors. They are packaged in three-lead metal cans or black plastic packages. Insert the transistors into the board. If the transistors are metal cans the tabs on the board pattern must be aligned with the tabs on the cans. If the transistors are in plastic packages, 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.

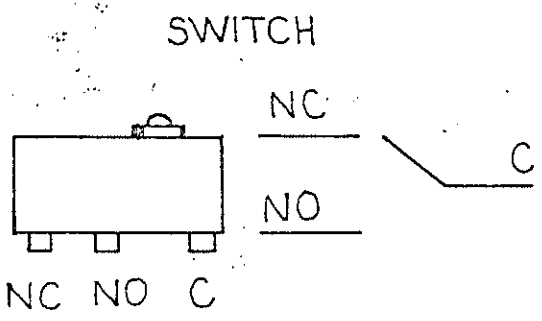


Switches: There are four switches in the Turtle pc board. They are activated by the touch disk to

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



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

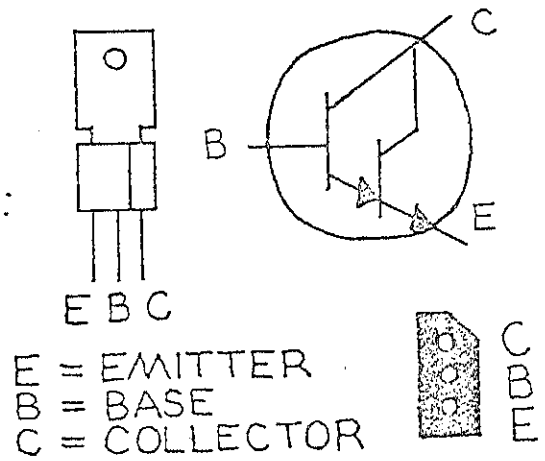


NC = NORMALLY CLOSED
NO = NORMALLY OPEN
C = CLOSED/COMMON

are thick, they may require substantially more heating than the leads of other components. If the swtiches 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

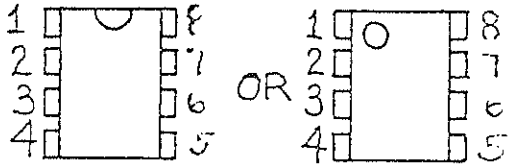


and clip leads close to board. The tabs at the top of the darlingtons should not touch each other while the Turtle is in operation.

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

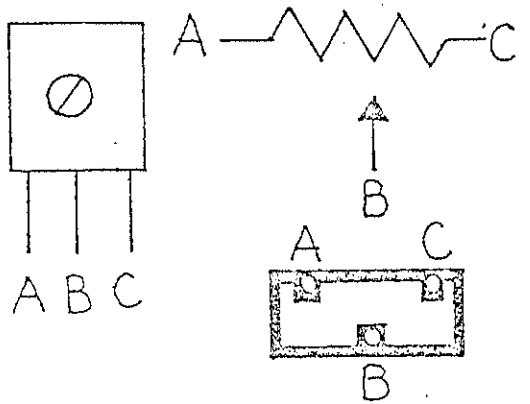
with the 555 improperly placed is likely to damage the 555. The pins are numbered (by convention) as shown in the drawing below. (The 555 is viewed from the top.) The notch or dot in the case indicates the "pin 1" end. The integrated circuit must be inserted so that this notch is at the same end as indicated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

INTEGRATED CIRCUIT



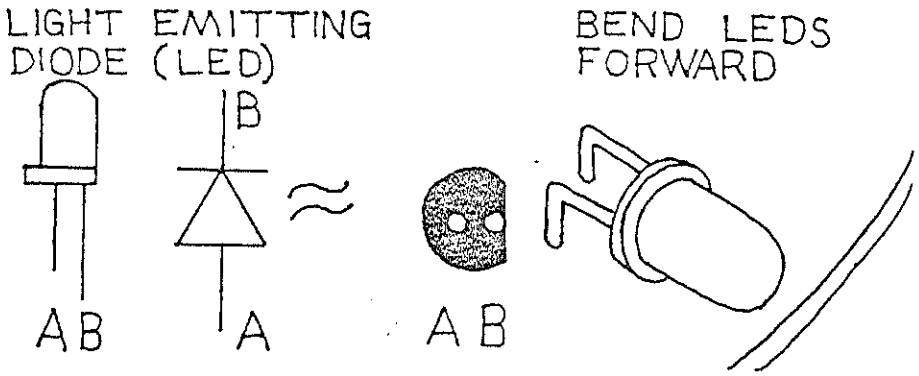
Pots (Variable Resistors): There are four trimmer pots (potentiometers, variable resistors) on the Turtle board for tuning the motor circuits. Insert the triangularly spaced leads into the top of the board. Solder and clip leads close. Make sure that the adjustment slots on the pots face the front of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

POTENTIOMETER



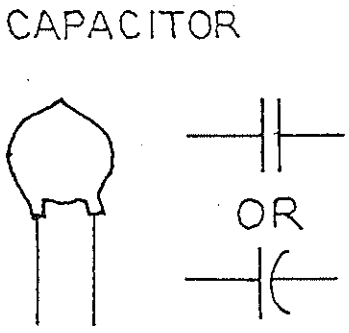
When you power up your Turtle, you will use the pots to adjust the Turtle motors so that your Turtle runs straight (see "Adjustments," p. 31).

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

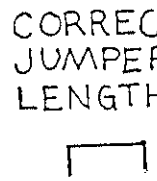


the symbol on the printed cirucit board pattern. Insert the LEDs in the board, leaving ½" of the leads on top of the board. Solder on to the pc board. Then bend the LED leads on top of the board at a right angle, so that the LEDs lie flat on the board, facing front. Clip leads close to bottom of board.

Capacitor: There is a single .1µf 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.



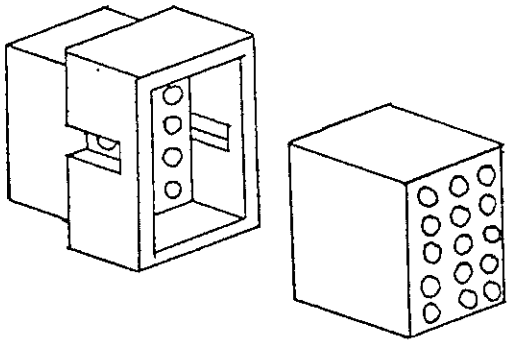
Jumper: A "jumper" is a wire that connects one part of the foil to another. There is one jumper wire on the pc board. It also functions as a test point for the supply voltage. This jumper can be made from an excess lead clipped from a resistor. The location of the jumper is indicated on the pc board by a line connecting two holes. Use pliers to bend the resistor lead to the size shown above and solder it between the holes.



Cable and Connector: The cable assembly consists of ten feet of Terrapin cable, a Waldom Molex™ 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.)

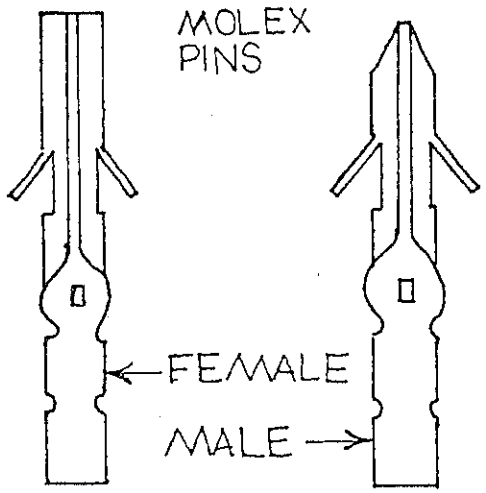
The way you assemble this part of the Turtle should reflect your own needs. You may assemble the cable exactly as recommended below. You may choose to omit the Molex connector and simply solder directly both the Turtle and interface ends of the cable. That would save some assembly time, but would necessitate 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

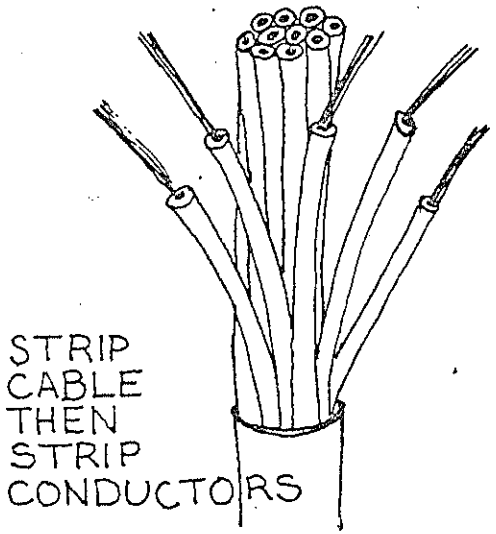
MALE PLUG



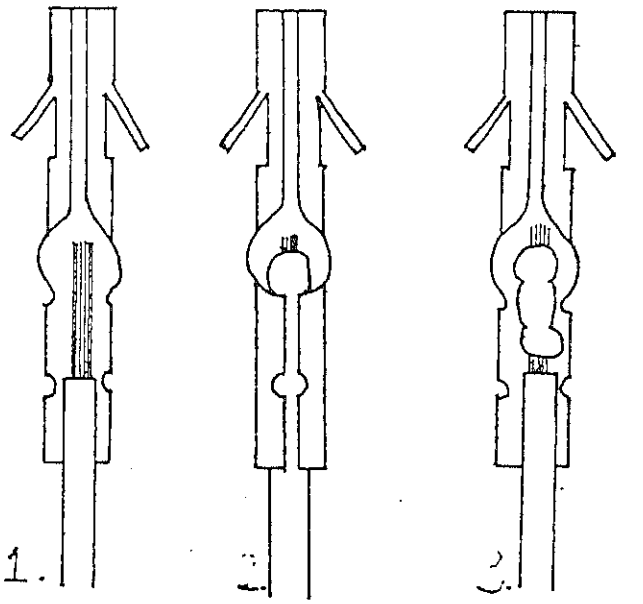
the suggested sequence of assembly, the cable ends are directly soldered to the interface card and to the Turtle board. The Molex connector is placed in the cable near the interface card.

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.*



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)

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

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 wiires 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 constructors. *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

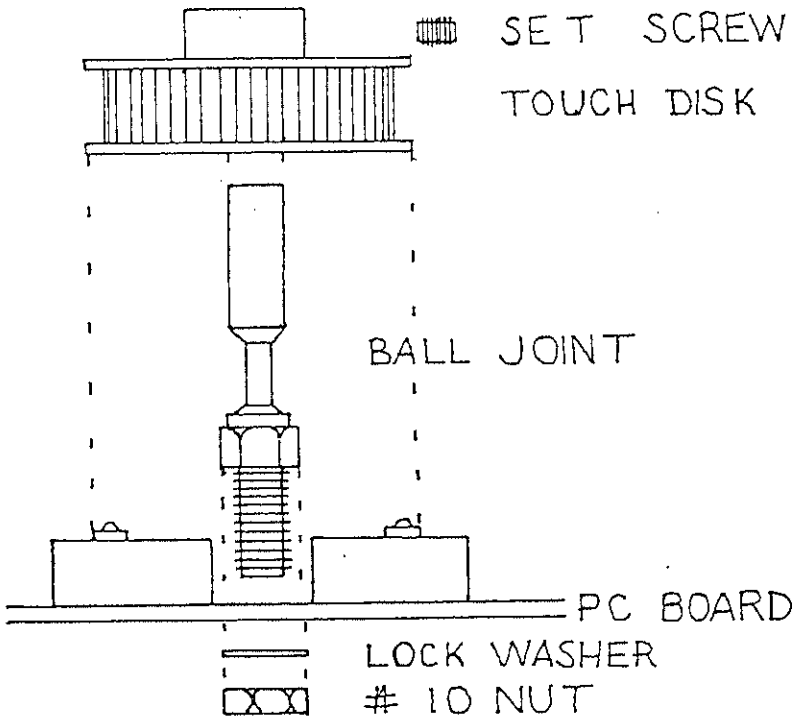
with VOM or Oscilloscope probe clips for testing purposes.

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

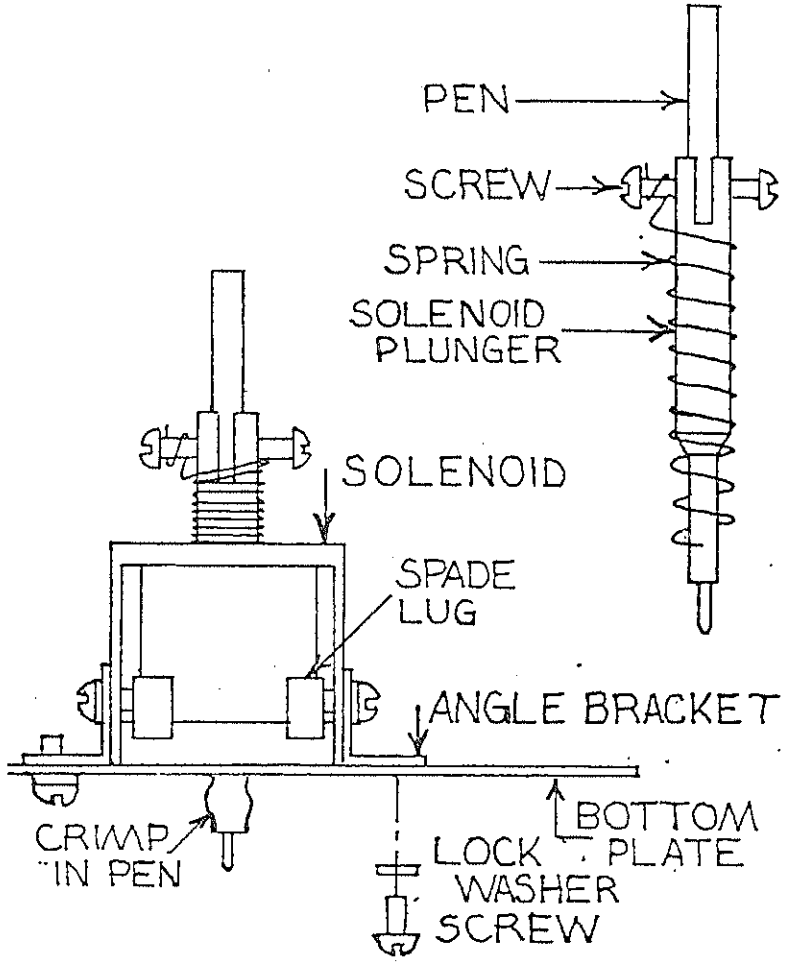
MECHANICAL ASSEMBLY

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

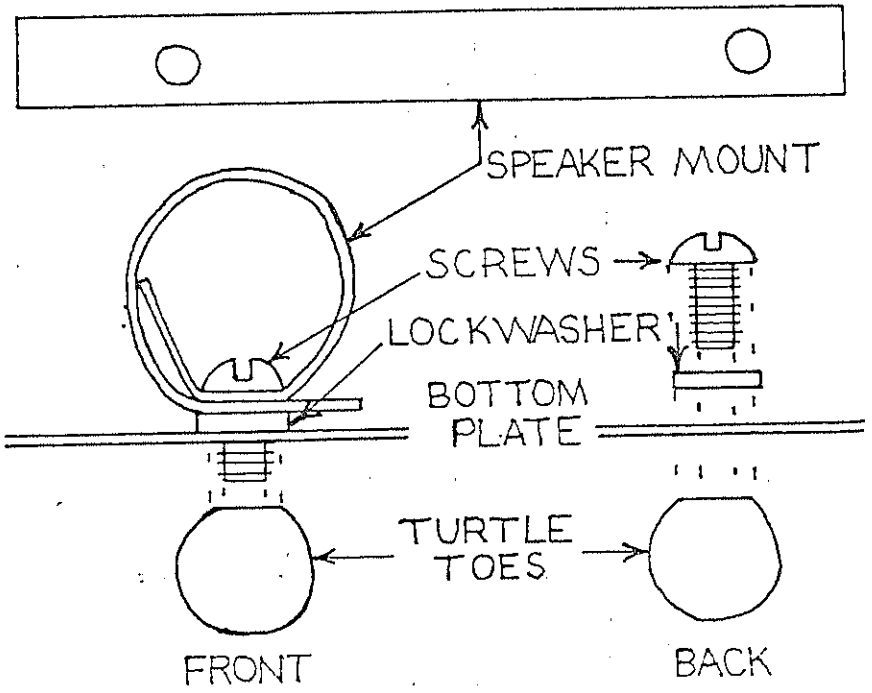


refill from its paper packet. (These refills are available at your local stationery store.) Following the instructions on the packet, break the refill off at the bottom line. Note that there is a crimp near the ball end of the pen. This must be rounded out with pliers so that the pen may be inserted 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.

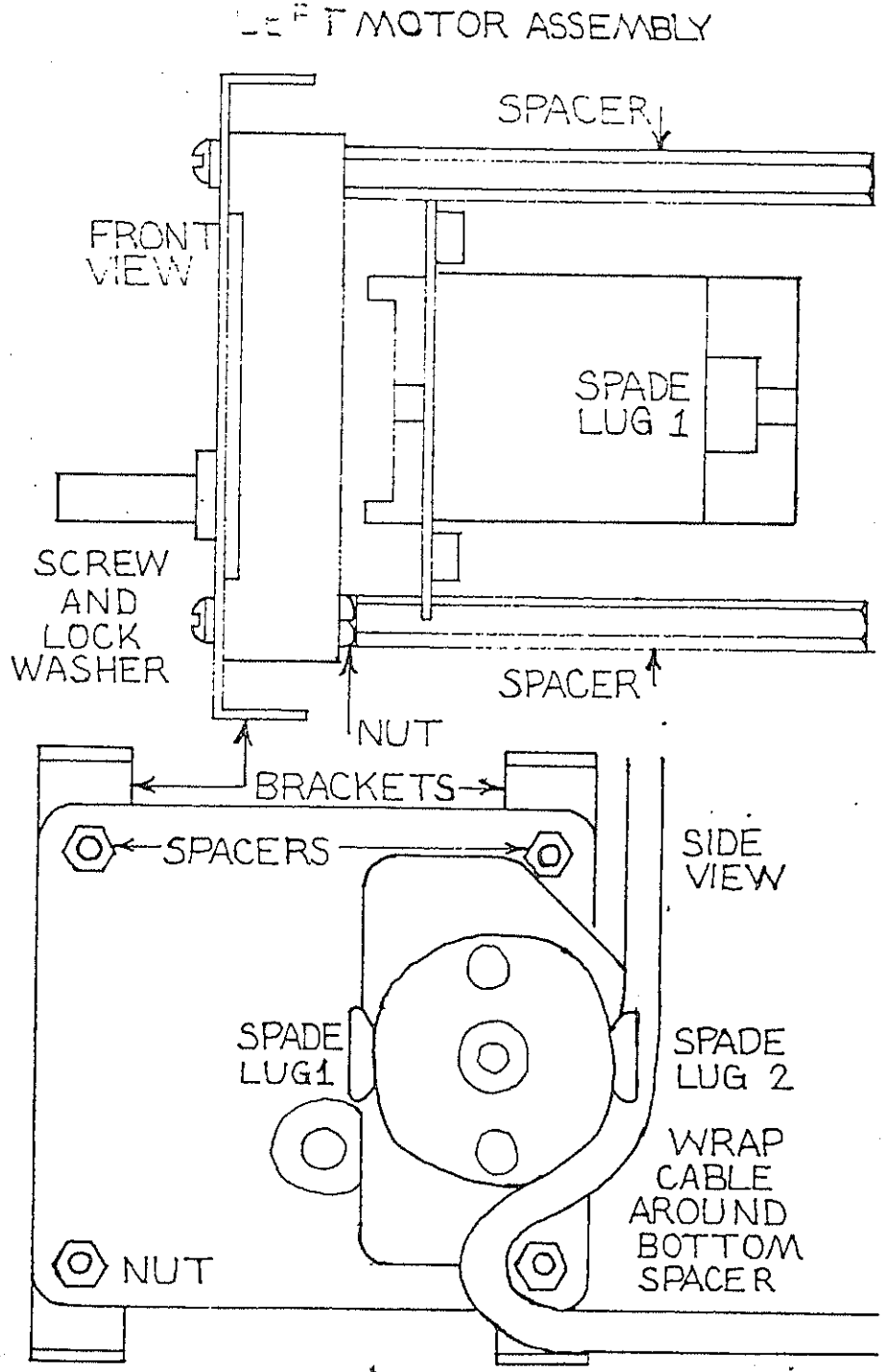


The ball end of the pen will protrude from the bottom of the bottom plate. To prevent the plunger assembly from falling out, use pliers to gently crimp the pen beneath the bottom plate. *In order to assure that you do not obstruct ink flow, place a screw or other piece of hardware between the handle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

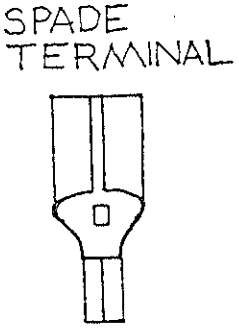


Right Motor Assembly: You will need one of the motors, the three aluminum hex cross section spacers, two motor brackets, four 6-32x1" screws, four #6 lock washers, one 6-32 nut, the pc board, two 6-32x1/4" screws, and two #6 nylon washers.



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.*



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.

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.

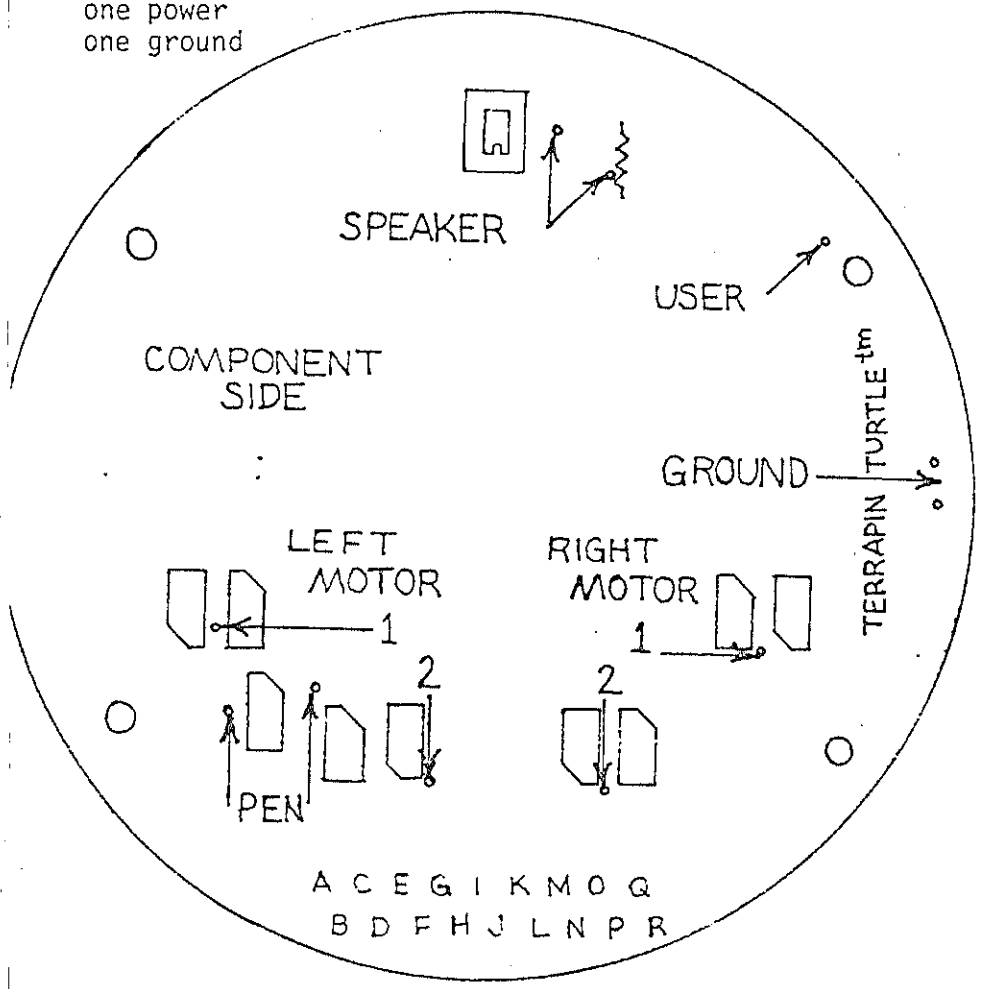
Dome: Screw the dome onto the shaft of the ball joint using the 10-32x1" screw, with the nylon shoulder washer under the dome, curved side down.

Behold your finished Terrapin Turtle!

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:

T	H	P	L	LM2	LM1	RM2	RM1

LSB

RM1, RM2 - Right motor
LM1, LM2 - Left motor
L - Lights
P - Pen
H - Horn
T - Tone control

RT - right
LT - left
FD - forward
BK - back

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 LT about left side
1	1	1	0	off	BK	turn RT 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

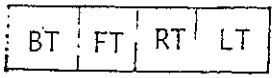
PEN "0" is up
"1" is down

HORN "0" is quiet
"1" is noisy

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

Touch Sensor Registor (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

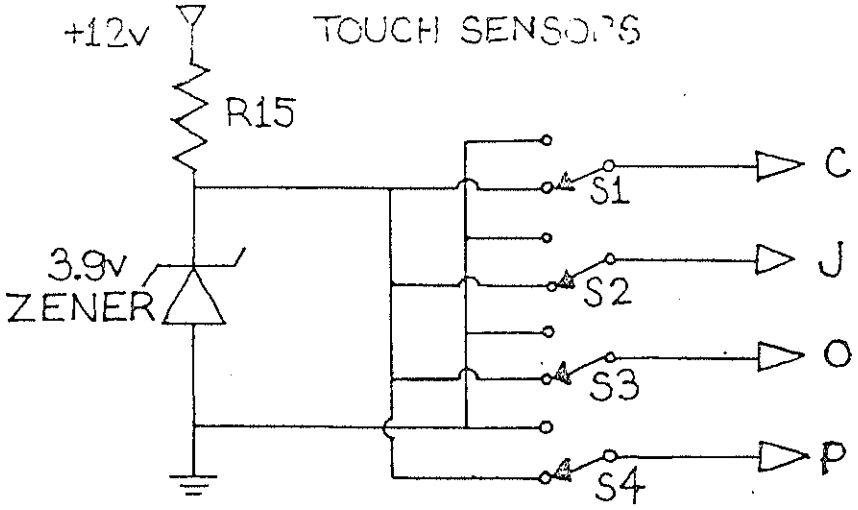
Pen: For best drawing ability the pen should be adjusted in the following manner: Connect the Turtle to a 12-18 volt power supply and to the parallel interface on your computer. Remove the Turtle bottom plate and turn the pen solenoid on by writing a logic "1" to the parallel interface bit connected 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 ¼". 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 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

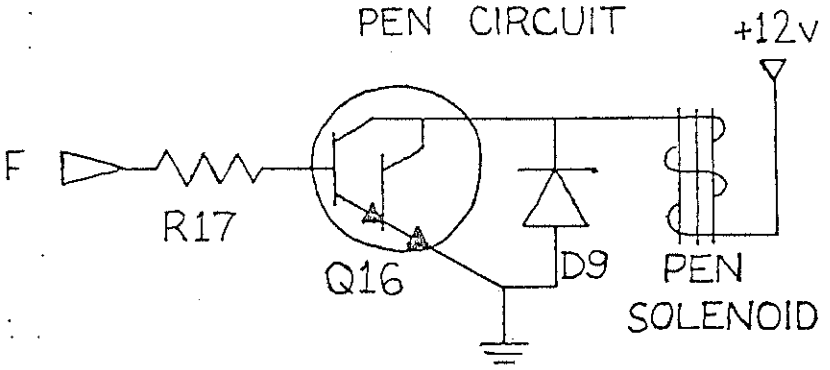
assembly sticks at any point try bending the refill so it travels freely. Now put the bottom plate back on the Turtle, place the Turtle on a sheet of paper, and send it forward with the pen down by writing logic 1's to the pen bit and to RM1 and LM1. If the Turtle does not draw properly, repeat the adjustment proceeding with the pen down in refill position slightly lower or higher than ¼" 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.

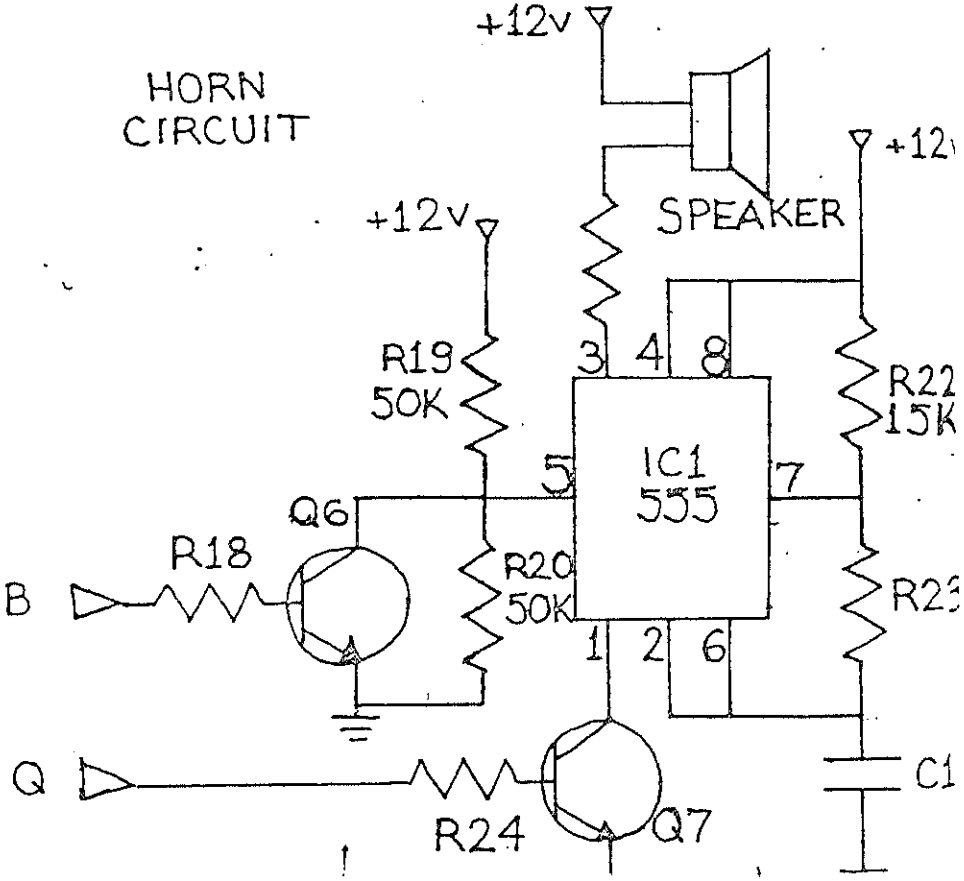
Motors: In order to make your Turtle run straight when going forward or backward and turn at the same speed when going right or left, you must adjust the 4 motor speed adjustment pots. First, use a small screwdriver to turn the adjustment slot on each pot clockwise until it stops. Then place your Turtle 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.

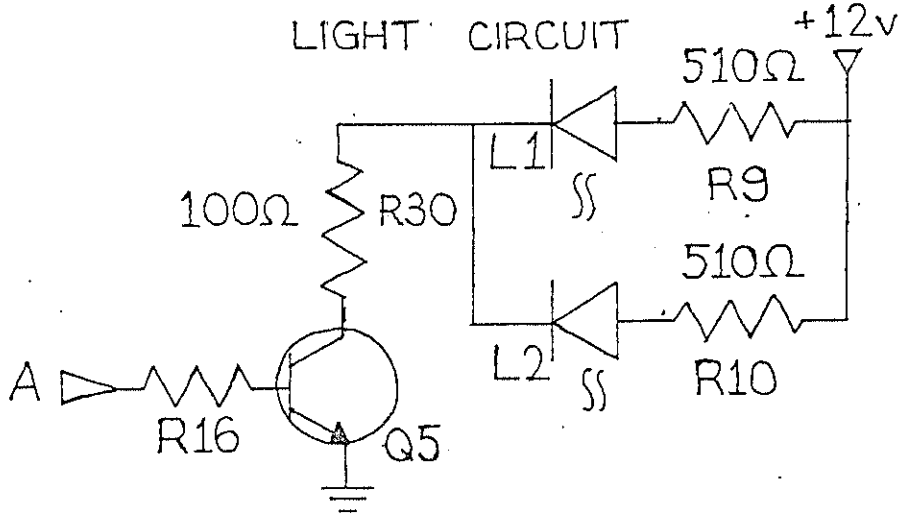
THEORY OF OPERATION

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.

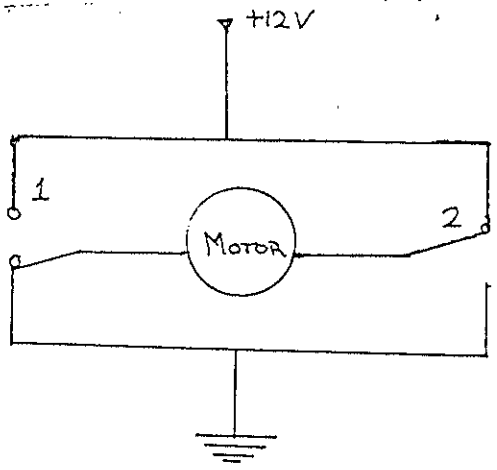

+12v
TOUCH SENSORS
R15
3.9v
ZENER
S1
S2
S3
S4
C
J
O
P

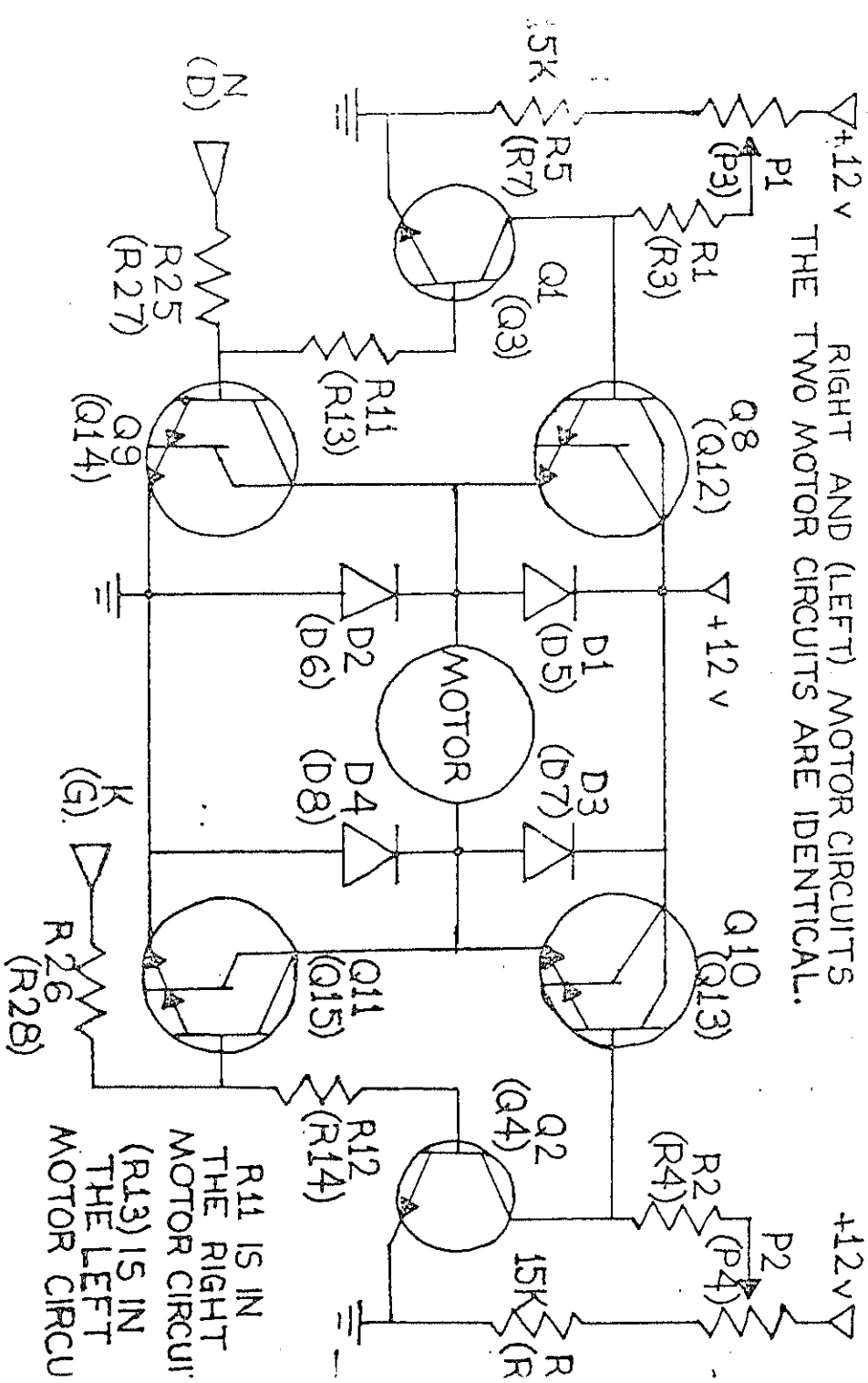

PEN CIRCUIT
+12v
F
R17
Q16
D9
PEN
SOLENOID


HORN
CIRCUIT
+12v
SPEAKER
+12v
R19
50K
3
4
8
R22
15K
IC1
555
5
7
Q6
R18
B
R20
50K
1
2
6
R23
C1
Q
R24
Q7


LIGHT CIRCUIT
+12v
510Ω
L1
R9
100Ω
R30
510Ω
L2
R10
A
R16
Q5

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 swtiches 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

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.

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 darlingtons	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.

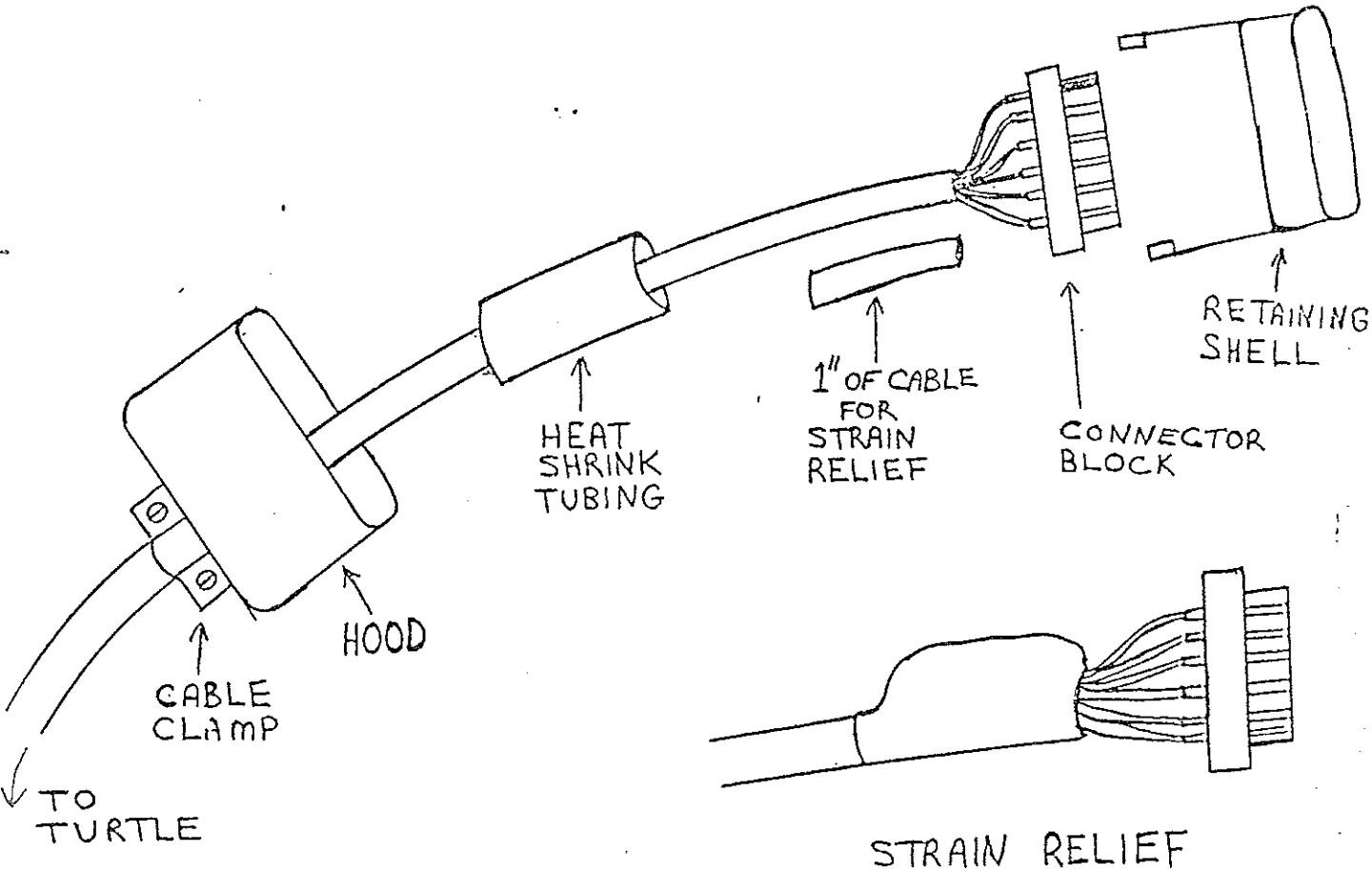
Cable and Connector (page 15-19)

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.

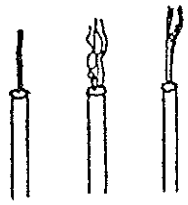
To prepare the cable for soldering to the connector, cut off approximately 1 inch of cable and save for use as a strain relief. Then cut and strip the cable as described in the manual on page 17. Tin each wire (heat the end and allow it to "wick" up a small amount of solder) immediately after stripping 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)



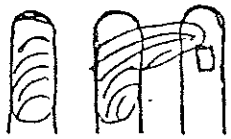
STRAIN RELIEF AFTER HEATING TUBING

TINNED WIRE



RIGHT WRONG

TERMINAL CONNECTIONS



RIGHT WRONG

AVOID SOLDER BRIDGES

PUSH WIRE ALL THE WAY INTO TERMINAL

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

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.

FIG. 1

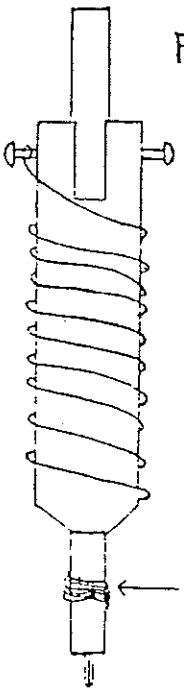
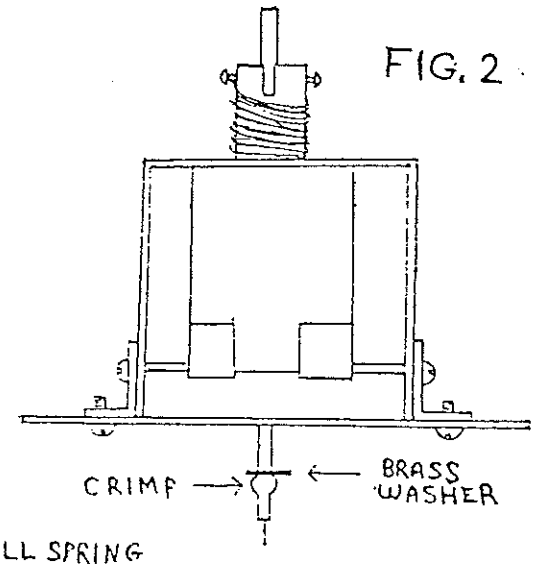


FIG. 2



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.