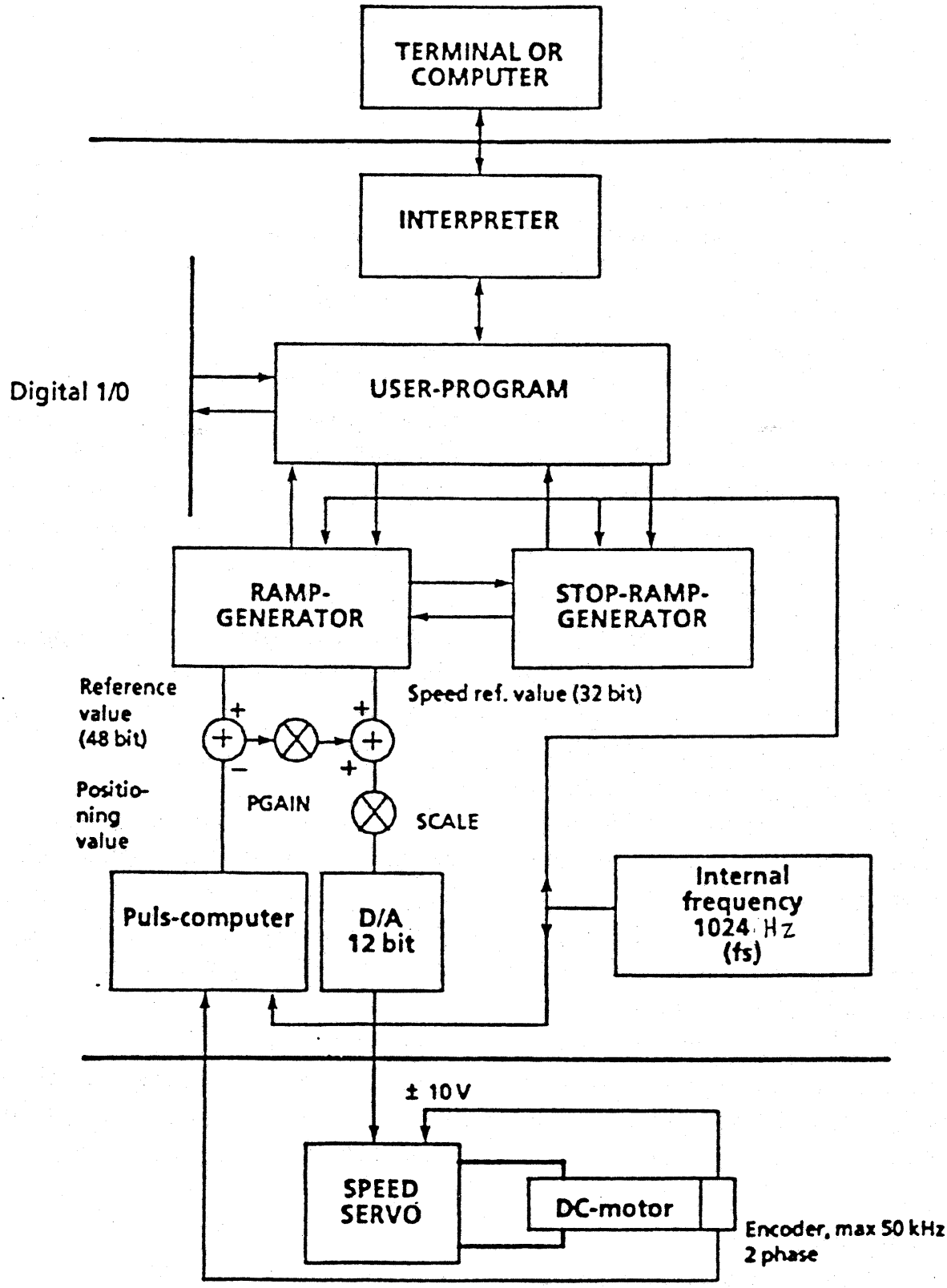
StackPointer

4-1988. Organ för Datorföreningen Stacken, KTH.

PDJ – en makalös manick att knäcka julnötterna med – julklappen till den som trodde han hade allt – sid 14.


TERMINAL OR COMPUTER
INTERPRETER
Digital 1/0
USER-PROGRAM
RAMP-GENERATOR
STOP-RAMP-GENERATOR
Reference value (48 bit)
Speed ref. value (32 bit)
Positio-ning value
PGAIN
SCALE
Puls-computer
D/A 12 bit
Internal frequency 1024 Hz (fs)
± 10 V
SPEED SERVO
DC-motor
Encoder, max 50 kHz 2 phase

Figur 1.

PDI—Peanut Defence Initiative

eller

Hur man bygger en jordnötskastare

av Thord Nilson

DET VAR på våren 1988 och NASACON IX närmade sig med stormsteg.

NASACON är en årlig SF-kongress som hålles någonstans i Fisksätra. Förutom det litterära, arrangeras även det så kallade jordnötsloppet.

Jordnötsloppet är en tävling där det gäller att förflytta en jordnöt minst två meter, över ett av tävligsledningen uppmätt område. Alla medel är tillåtna utom att över området tillföra mänsklig energi. Dvs, man får kasta över nöten, men inte bära den. Stilpoäng utdelas. Den som får högst stilpoäng vinner tävlingen.

På NASACON VIII, året innan, hade SAFA (Svenska Arbetsgruppen För Algoritmforskning) ställt upp med PDI i jordnötsloppet.

Den dåvarande PDI-konfigurationen bestod av en voice coil-magnet från en utrangerad RP03:a. Den används för att förflytta läs/skrivhuvudena. En sådan magnet är klart imponerande. Efter transport i bil, har de flesta lösa verktyg man hade i bagageluckan hittat dit. I stället för den medföljande spolen tillverkades en egen av en avsågad 1-liters T-spritflaska, som visade sig ha samma diameter som originalspolen. På toppen av flaskan (vid korken) monterades ett cigarretui, i vilket sedan jordnöten skulle placeras.

För att skjuta iväg spolen anslöts sedan ett kraftigt kondensatorbatteri via en kontaktor. För att öka imponatoref-

fekten ytterligare styrdes det hela från en portabel dator med talsyntesprogram.

Det hela blev mycket uppskattat och SAFA vann tävlingen.

Nu gällde det alltså att försöka hitta på något ännu häftigare.

På arbetet (Mikrologik AB) satt jag för tillfället och uppdaterade programvaran till GME-systems positioneringssystem Mikropos 300. Det var då jag kom att tänka på möjligheten att montera en sked direkt på motoraxeln och sedan snäppa iväg jordnöten med skeden på något intressant sätt.

Mikropos 300 är styrdatorn i ett generellt positioneringssystem för enklare maskinstyrningar etc. Ett sådant system består i huvudsak av: Servomotor med tachometer för hastighetsåterkoppling och en pulsgivare för positionsåterkoppling. Till detta kopplar man en drive och en Mikropos. En drive är i princip ett effektsteg med en hastighetsregulator. En analog insignal styr motors hastighet, 0 V = stå stilla, +5 V = halv fart framåt, +10 V = full fart framåt, –5 V = halv fart bakåt etc.

Mikropos är kopplad till pulsgivaren och denna styrsignal. Från Mikropos kan man sedan styra motorns position, hastighet och acceleration, se fig 1.

Ett timerinterrupt i Mikropos beräknar varje ms en ny önskad hastighet och position för motoraxeln. Den beräknade positionen jämförs med den verkliga positionen (från pulsgivaren). Denna skillnad kombineras med den önskade hastigheten för att ge ett nytt hastighetsbörvärde till driven.

För att sedan kunna styra det hela på ett effektivt och flexibelt sätt finns det en interpreter som interpreterar ett för positioneringsändamål specialutvecklat språk kallat P. Språket P påminner till utseendet om en blandning av Basic och Assembler, se listningen.

Nu skulle jag alltså testa ifall det skulle vara praktiskt genomförbart att montera en sked eller liknande på motoraxeln, lägga en jordnöt i skeden och sedan på ett konstfullt sätt skicka iväg jordnöten minst 2 meter.

En plastsked monterades provisoriskt och försök med att kasta muttrar påbörjades. Någon timma, ett antal plastskedar och många muttrar senare skickades den första muttern med en kraftfull forehand i väggen. Det fungerade alltså, med serve!

Nu behövde följande göras/byggas:

Ett riktigt fäste för skeden, en ställning för motorn, en laddare av något slag (det vore ju inget roligt att bara lägga jordnöten i skeden), skriva styrprogram i P för att kontrollera skedens rörelser, ett häftigt menyprogram till I*M PC med imponatoreffekt, affischer och ett

föredrag med OH-bilder visandes PDI:s funktion.

Vi delade upp arbetet så att Nils tog hand om menyprogrammet, Jörgen fick fixa föredraget, jag P-programmet, och jag + Jörgen de mekaniska bitarna.

En helg ca 1 månad före NASACON var undertecknad hos Jörgen för att bygga behövliga mek-bitar. En ställning till motorn svetsades samman och slipades till av ett par meter fyrkantsrör. Detta skedde under svåra protester från en finsk barnkör, som övade ute på gräsmattan. Vi fick bara använda slipmaskinen i korta intervaller, medan kören drack saft och åt bullar eller var allmänt religiösa.

Därefter skulle mataren för jordnötterna konstrueras. Vi tänkte oss att nötterna skulle ligga i ett magasin (à la k-pist) som ledde till en ränna och sedan flyttades därifrån med en kolv. Detta visade sig dock inte fungera i praktiken.

Problemet var att toleransen hos jordnötterna var betydligt sämre än hos 9 mm:s ammunition. Efter ett antal timmars provande och testande kunde vi klara max två till tre jordnötter i magasinet utan att de fastnade i varandra. Det är förvånansvärt på hur många olika sätt jordnötter kan trassla ihop sig. Vi gjorde en back-tracking (som det heter i Prolog) och hamnade till sist på den välkända Colt-principen. Den verkade tillräckligt enkel för att kunna fungera med jordnötter.

Alltså, en revolver med plats för tolv jordnötter sågades ut ur en spånplatta, en ställning löddes ihop av kretskort och stagades med tråd och ännu mera kretskort. Nu återstod bara rännan som nötterna skulle glida nedför för att hamna snyggt och prydligt i skeden. Det visade sig vara ett icke-trivialt problem. (För de som inte tror mig: Skaffa en påse oöppnade jordnötter och prova själv!) Någon gång vid midnatt var vi nöjda med resultatet.

En vecka senare fanns en preliminär version av styrprogrammet klart. Det var nu dags för samkörning mellan PC:n och Mikropos, dock fanns det inget menyprogram. Nils S höll på, men var inte klar.

Nils hade hittat en Modula-2-kompilator för PC på Kicki. Som demoprogram för en medföljande multitaskhanterare fanns ett terminalemulatorprogram. Nu höll Nils på att hacka om detta program till ett PDI/2-menyprogram. Detta för att enklare kunna köra seriekommunikationen mot Mikropos.

Medan jag trimmade positioneringsprofilerna till Mikropos skrev Jörgen följande spec till Nils:

Nya specar på de strängar som skall styra PDI/2, the sequel.

PDI Hardware Team har i sin godhet beslutat införa en del nya strängar att styra nötkastaren med. Dessutom har en ny form av kommunikation införts. Allt för att glädja Herr Programmeraren av användarinterfejset.

Ny kommunikation.

Prompten och loginmeddelandet fås som tidigare.

Nu gäller det att mata värden till och läsa värden ur register istället. Läser gör man på följande sätt:

DISP Rnn<CR>

Svaret blir:

Rnn= m

och en ny prompt.

Att stoppa ett värde i ett register går till på följande sätt:

LET Rnn, m<CR>

varvid svaret blir en ny prompt.

Nya strängar.

Alla kommandon enligt tidigare spec. har utgått och ersatts av att man matar in värden i register och skickar till Mikropos istället. Svaren får man genom att läsa andra register. Denna metod bryter inte exekveringen, som ^C gjorde. Värdena och åtgärderna för register R20 kvarstår dock i stort sett oförändrade.

Registren kan pollas när som helst och svar kan fås när som helst under körningen.

Vi har nu fyra register, R20, R21, R22 och R23.

Reg	Läsfunktion	Skrivfunktion
R20	status	N.A.
R21	= 0, kommandot accepterat. Nytt kommando kan ges omedelbart, ett kommando kan ligga pending.	Skjutochladdkommando.
R22	Antal nötter i revolvern.	Antal nötter kvar efter nyladdning.
R23	Hur många nötter kvar att automatskicka.	Antal nötter att automatskjuta.

Detaljer, R20.

Om R20 är	gäller följande status
0	MP måste startas, ge "^C RUN<CR>"
1	Klart att ge startkommando
2	Klart att ge startkommando

Detaljer, R21.

Om R21 läses till noll är det nyss givna kommandot accepterat. Alla andra värden skall ignoreras.

Värden skrivna till R21 är skjutkommandon. Dessa är följande:

Tal	Betydelse
1	Misslyckad laddning av jordnöt, sked för hög.
2	Misslyckad laddning av jordnöt, sked för låg.
3	Lyckad laddning, svagt kast (0.5 meter).
4	Kastar jordnöt och tar den igen, kastas därefter enl 3.
5	Lyckad laddning av jordnöt, men tappar den senare.
6	Vevar skeden fram och åter, winding up spoon power.
7	Enkelt kast av jordnöt. Tävlingskast utan serve.
8	Slag med serve, framåt, tävlingsslag.
9	Mycket hårt slag med serve, bakåt. Jordnöt sannolikt krossad

10	Livsfarligt kommando, skeden går 600 rpm samtidigt som magasinet töms. Jordnötter sprutar åt alla håll.
11	Automateld, tävlingskast, snabbare än om det görs som upprepade 7 eftersom, tja skit i det. Antal jordnötter som skjuts anges i R23, före skottet.
12	Skeden går som sekundvisare, tills annat kommando ges. Skjuter nötter varje minut, tills nötterna är slut.
13	Automateld med krossfunktion, mycket illusoriskt, jordnötsdimma.
14	To be determined.
...	

Detaljer, R22.

Häri skrivs hur många nötter det nu finns i revolvern. Om Mikropos skulle ha en avvikande mening, dvs när man skjutit slut och laddat. Max antal nötter är tolv (12).

Detaljer, R23.

Häri skrivs hur många nötter nästföljande automateldkommando skall skjuta. Automateld genom Mikropos försorg blir snabbare än om du skall ge flera skjutkommandon i serie på grund av att skeden inte behöver återställas efter varje skott.

Värdet som skrivs i R23 får naturligtvis inte vara högre än det som läses ur R22.

Jörgen

Två dagar innan tävlingen:

Nu var det hög tid för samkörning. Nils var dock inte riktigt klar ännu, några detaljer var kvar att fixa, bl a läsning av registervärden från Mikropos.

Samkörningen gick hyfsat tills läsning av registervärden skulle provas. Ett allvarligt problem upptäcktes: Om det blev ett enda överföringsfel på seriekanalen så hamnade alla processerna i PC:n i WAIT, och det fanns ingen timeout-möjlighet.

Olika alternativ övervägdes och förkastades. Det hela skulle gå att lösa snyggt om multitaskhanteraren hade varit styrd från timer-interruptet eller hade haft en funktion motsvarande

Simula Simulations HOLD. Vid midnatt var det dags att ge upp för dagen.

Hemma, med källkoden till multitaskhanteraren försökte jag laga till en HOLD-funktion. Efter någon timmas funderande fanns ett utkast klart.

En dag innan tävlingen:

Den nya HOLD-funktionen provas och visar sig fungera. Meny-programmet skrivs om för att utnyttja den, och allt verkar funka OK. Nils ritar nya skärmbilder, ändrar andra och fixar in nya finesser i programmet som diverse oljud mm.

Under tiden plockar jag och Jörgen ihop de saker som skall användas, fixar kablar till högtalare etc.

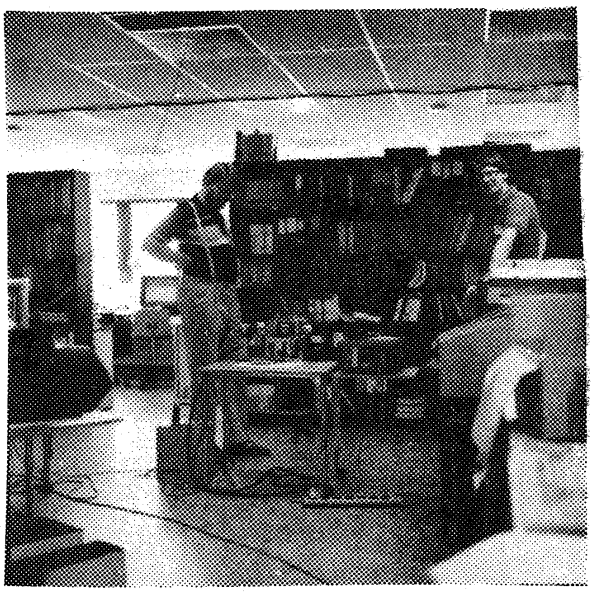


Bild 1: Testuppkopplingen på Mikrologik begrundas av Jörgen och mig.

Tävlingsdagen:

Vi har nu en konfiguration enl fig 2.

På förmiddagen ställs allt upp för en sista testskjutning. Allt fungerar OK. (Se bild 1 och 2) Vi monterade ned grejorna och lastade i två bilar för färd mot Fisksätra skola.

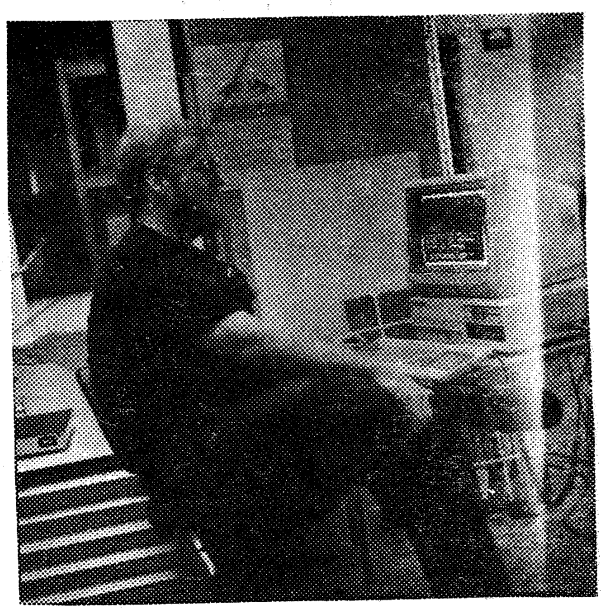
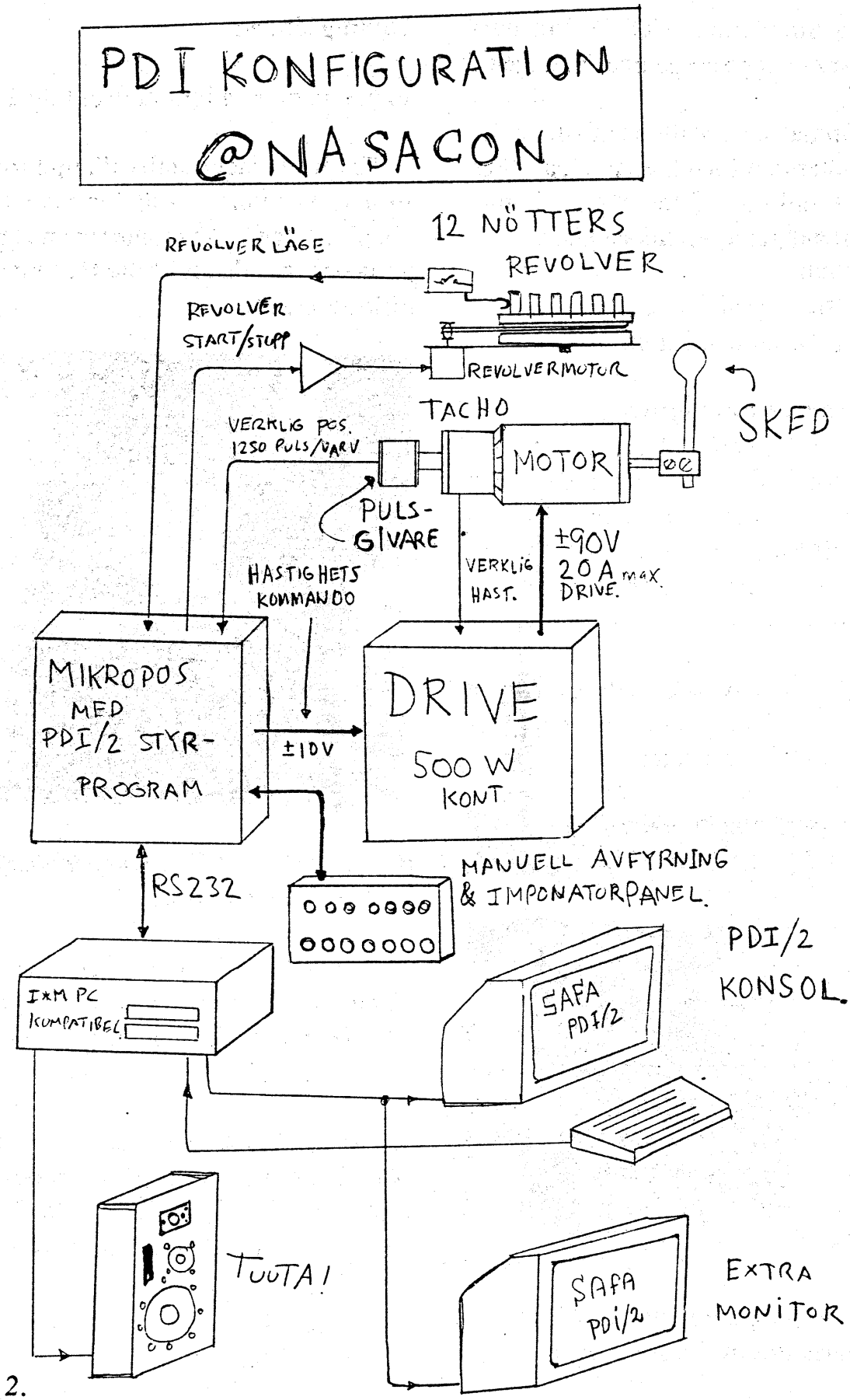


Bild 2: Nils som PDI Commander vid testkörning på Mikrologik.

Allt bars in i stora aulan i väntan på tävlingen. Vi hade ansökt om att bli sista deltagare, vilket tävlingsledningen också beviljade.

Jörgen, iklädd vit rock, och något svamligare än vanligt drog teorin bakom PDI. Han hade faktiskt utarbetat ett program som följdes i huvudsak:


PDI KONFIGURATION
@NASACON
12 NÖTTERS
REVOLVER
REVOLVER LÄGE
REVOLVER START/STOPP
REVOLVERMOTOR
TACHO
VERKLIG POS.
1250 PULS/VARV
MOTOR
SKED
PULS-
GIVARE
HASTIGHETS
KOMMANDO
VERKLIG
HAST.
±90V
20A max
DRIVE.
MIKROPOS
MED
PDI/2 STYR-
PROGRAM
±10V
DRIVE
500 W
KONT
RS232
MANUELL AVFYRNING
& IMPONATORPANEL.
I*M PC
KOMPATIBEL
SAFA
PDI/2
PDI/2
KONSOL.
TUUTA!
SAFA
PDI/2
EXTRA
MONITOR

Figur 2.

Peanut Defence Initiative
the sequel
på Nasacon 9, 1988

1. Sakerna ställs upp inför publik
2. Overheadpresentation
3. Skjutning
4. Undertecknande av Onlookers Award

2. Overheadpresentation

SAFA och PDI

PDI:s historia

PDI och framtiden

PDI i ett militärhistoriskt perspektiv, redan Lionardo da Vinci...

PDI är naturligtvis helt ofarligt för Sveriges befolkning. Som vi redan har visat med vår förra demonstration, den elektromagnetiska kanonen, är de förstörelsevapen SAFA framställer helt riskfria för den egna nationen. Naturligtvis är eldkraften så stor att ingen annan nation skulle våga ett anfall. Som vanligt framställer SAFA bara vapnen för forskningens skull och skall de sedan användas är det ju politikernas sak. Vi tar inte ställning till den saken. Vi gör bara vad som är bäst för landet.

(Overhead ON!)

Projektet studerades denna gång mycket ingående och under några hektiska veckor i juni färdigställdes och togs hela utrustningen och all programvaran fram. Det var The SAFA Night Hacking Team, vilka alla finns representerade här idag, som arbetat till långt in på nätterna vid terminaler och lödkolvar för att få allt färdigt till denna viktiga demonstration, idag.

För att ni skall kunna föreställa er vilka svårigheter detta projekt bjöd på skall jag ta er på en kort rundvandring genom årets jordnötskanons alla delar.

Gå igenom konstruktionen

Sluta med att utlova en Award

3. Skjutning

Och nu får jag be alla intresserade stiga fram och flockas kring bildskärmen men se upp för kanonen, eftersom den är farlig och kan ställa till med personskador om den vidrörs. Det visade sig bland annat under projektets gång, då skedenheten lossnade från motordelen och flög och studsade genom hela laboratoriet, slog märken i en printer och slutligen hamnade i ett angränsande rum.

Presentera maskinens olika delar

Tala om problemen som uppstått under projektets gång, olika skjutfel, nötterna hamnade på alla andra ställen än där de skulle. Det stod från början helt klart att nötterna måste automatmatas. Inte kunde man hålla på att lägga i dem för hand, inte. Vi hade på gång en helt annan matarteknik, nämligen en matare med liggande nötter, men det visade sig inte fungera så bra, revolvern fick bli lösningen. Många teorier prövades, rörande det optimala skottet och vi ska be att få visa några av dem, senare.

Spoon-enheten var ursprungligen av plast, men detta gav sådana problem med flygande splitter i lablokalen att den snabbt utbyttes mot en i ett mera beständigt material, nämligen ståååååål! Dessutom visade det sig att vi behövde ett specialfäste som tålde höga accelerationer, varför ett dylikt svarvades av Delrin.

Demo av The Spoon Unit. Nu måste fjädern spännas, så vi tar och vevar upp spoonenheten.

Veva skeden fram och åter-demon

Demo av olika laddnings-metoder

Misslyckad laddning, sked för hög LOAD TOO HI
Misslyckad laddning, sked för låg LOAD TOO LO
Tappar nöten LOAD 'N DROP
Släng upp nöten och fånga igen CATCH

Demo av eldkraften, tidigt försök

Svagt kast LOAD OK 50 CM

Tävlingskastet, får vi nu be tävlingsledningen frambära N Ö T E N och ladda den i revolvern!

Enkelt kast — tävlingsjordnöt — ONESHOT 1

Andra, kraftigare skott

Slag med serve ONESHOT 2
Hårt slag bakåt BACKFIRE

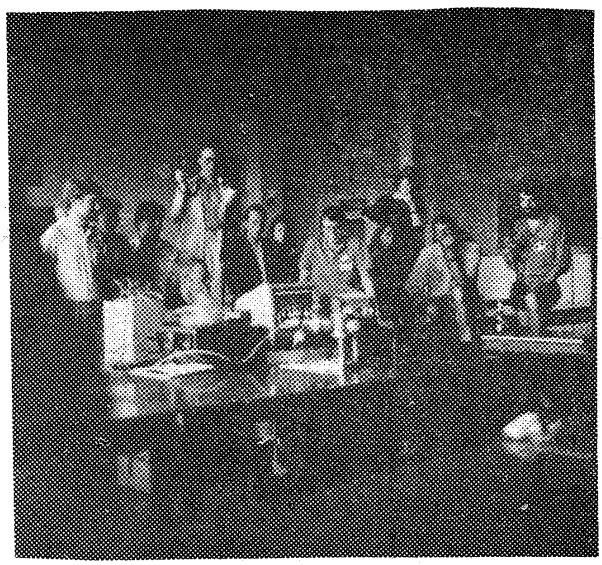
"All out superpower confrontation"

kommando 10, slumpmässigt nötsprut, s k asynkron kross LETHAL!!!
kommando 11, lyckad automateld AUTOFIRE

Medan domarna nu bestämmer sig för vilket bidrag som skall vara det vinnande, låter vi klockan gå! Observera med vilken precision vårt urverk spinner, etc.

Klock-kommandot CLOCK

Under tiden kan vi få be alla hugade åskådare att stiga fram och emottaga ett bevis på att ni stått ut med denna demonstration, the PDI Onlookers Award.



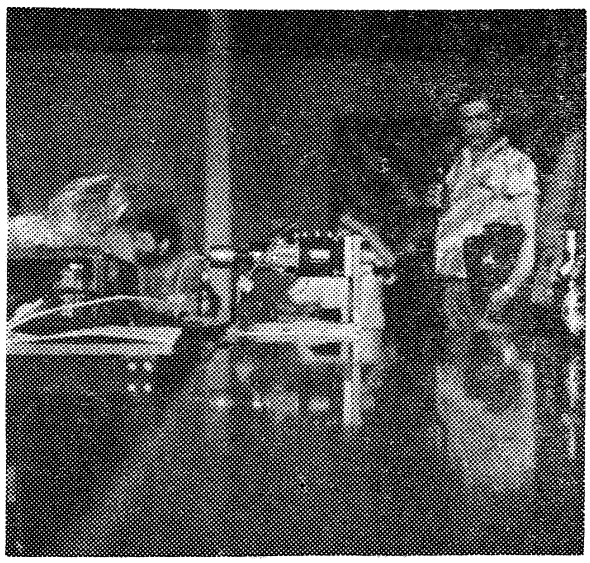


Bild 3 och 4: Vid NASACON, PDI utför CATCH, jordnöten kastas upp i luften, skeden backar ett varv för att sedan fånga nöten när den faller ner.

Det hela vart så lyckat att vi tyckte det var synd att bara deltagare på NASACON fick bevittna det hela, så efter ett tag fanns följande meddelande att läsa på diverse ställen:

SAFA visar PDI/2 på STACKEN-möte

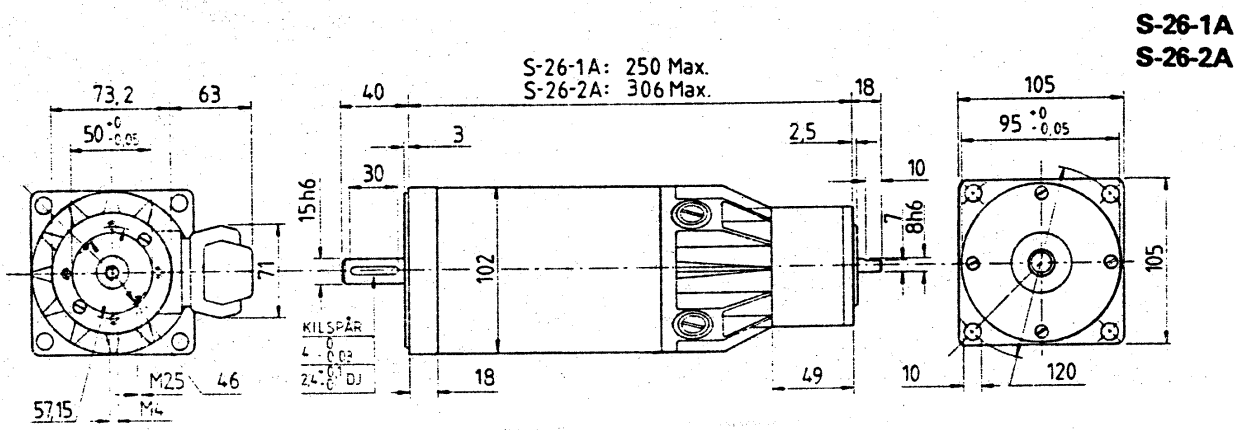
På STACKEN-mötet torsdagen den 7/7-1988 kl 19:15 i sal E7 kommer SAFA att demonstrera PDI/2, årets vinnare av jordnötsloppet på NASACON IX.

PDI står för "Peanut Defence Initiative" och PDI/2 består i huvudsak av en datorkontrollerad motordriven sked med vilken man på ett antal konstfulla sätt kan slunga iväg en jordnöt. Till denna sked är ett revolvermagain för 12 nötter anslutet. Maximala eldhastigheten är omkring 1 jordnöt/sekund.

Eftersom PDI/2 blev synnerligen uppskattad på NASACON och är kraftigt datoriserad så är det något som en sann hacker bara måste se.

Missa inte detta unika tillfälle!!!

Detta skedde också på utlovad tid, samt på kåren och vid ett senare tillfälle i STACKENs lokal.



Faktaruta:

Motor: ElectroCraft S-26-1A.

Kont moment:	2.5 Nm
Toppmoment:	20.7 Nm
Max varvtal:	3000 rpm
Max kont uteff:	700 W
Mek tidskonst:	6 ms
El tidskonst:	1.7 ms
Max kont ström:	7.5 A
Max toppstöm:	60 A
Vikt:	7 kg

Pulsgivare: Leine Linde, 1250 pulser/varv.
Detta pulstal multipliceras sedan med 4 internt i Mikropos vilket ger en effektiv upplösning på 5000 pulser/varv.

Drive: GME System PWE 6010.

Styrsystem: GME System Mikropos 300.

Och till sist, för de verkliga entusiasterna: Styrprogrammet i P för PDI/2:

```
;*****************************************************************
;*
;*      PDI/2 -- Peanut Defence Initiative, the sequel.
;*
;*      Peanut Throwing Program.
;*
;*      Motor/Drive:            Max RPM: 2500
;*      Pulse transducer:       1250 pulses/turn * 4
;*                              ==> 5000 pulses/turn.
;*
;*****************************************************************

.define status = r20    ; Ok for PC to restart prog if status=0
.define command = r21   ; Set by PC to execute a command.
.define peanuts = r22   ; Set by PC to indicate # of peanuts in revolver.
.define stsidle = 0     ; Program not running, PC must send ^C RUN
.define okstart = 1     ; Ok for PC to send command.
.define running = 2     ; Command in progress.

        .org    1               ; Start of program.

restart:
        serial mode 16          ; Enable XON/XOFF.
        let     status, running
        vector  10,ctrlc        ; Control-C trap....
        gosub   initpdi         ; Initialize.
        let     status, okstart
        goto    wcmd            ; Then  ready for command.
```

```
        .org    25
;       .linestep 2
        .linestep 1
;
;       This is the main lupe. -- Top level -- Waiting for command.
;

wcmd:   let     status, okstart
        gosub   rblink          ; Twinkle twinkle little led's.
        gosub   swdec           ; Decode switches.
        if      0 = command, wcmd ; Continue waiting if no command.

        gosub   cdec            ; go and decode command.
        goto    wcmd

;*****************************************************************
;*
;*      Decode input switches.
;*
;*****************************************************************

swdec:  if lo in and 1, wave    ; wave spoon.
        if lo in and 2, throw1  ; swing, mode 1.
        if lo in and 4, throw2  ; swing, mode 2.
        if lo in and 8, throw3
        if lo in and 16, throw4
        return

;*****************************************************************
;*
;*      Routines to take care of input switches.
;*
;*****************************************************************

wave:   goto    wavef

throw1: gosub   gohome          ; Start at home pos.
throw1a:
        gosub   swing1
        if lo in and 2, throw1a ; Do again if button pressed.
        return

throw2: gosub   swing2
        return

throw3: gosub   swing3
        return

throw4: gosub   swing4
        return

;*****************************************************************
;*
;*      Decode input command.
;*
;*****************************************************************

cdec:   sub     command, 1
        if      0 = command, cmd1               ; load point to high
        sub     command, 1
        if      0 = command, cmd2               ; load point to low
        sub     command, 1
        if      0 = command, cmd3               ; right but to slow.
        sub     command, 1
        if      0 = command, cmd4
        sub     command, 1
        if      0 = command, cmd5
        sub     command, 1
```

```
        if      0 = command, cmd6
        sub     command, 1
        if      0 = command, cmd7
        sub     command, 1
        if      0 = command, cmd8
        sub     command, 1
        if      0 = command, cmd9
        sub     command, 1
        if      0 = command, cmd10
        sub     command, 1
        if      0 = command, cmd11
        sub     command, 1
        if      0 = command, cmd12
        sub     command, 1
        if      0 = command, cmd13      ; This is the destructor command.

        return                          ; Noo good, wait for new command.

        .org    100

;*****************************************************************
;*
;*      Routines to take care of the individual commands.
;*
;*****************************************************************

cmd1:   let     r2,150          ; offset position -- load to hi.
        goto    faicmd

cmd2:   let     r2,-200         ; load to low.
        goto    faicmd

cmd3:   let     r2,0            ; load ok.
        goto    faicmd

faicmd: acc     50
        ret     50
        pos speed 30 000
        pos abs r2
        wait    rdy
        gosub   loadnut
        clr     tmr
        wait tmr > 1000
        gosub   xthrow
        return

;
;       CATCH.
;

cmd4:   gosub   gohome                  ; Goto home position.
        gosub   loadnut                 ; Load a nut.
        gosub   jiggle                  ; Jiggle it in position.
        pos     speed 40 000
        acc     50
        ret     50
        pos     rel -145                ; Backup to launch postion.
        wait    rdy
        acc     200
        ret     300
        pos     rel 500                 ; Throw nut into the air.
        wait    rdy
        pos     speed 60 000
        acc     600
        ret     400
        pos     abs -5000               ; Back spoon one turn when nut
        wait    rdy                     ; is in the air.
        clr     tmr
```

```
        wait    tmr > 1500              ; Wait for nut to land on spoon
        gosub   xthrow                  ; then lazy throw away.
        return

;
;       Load And Drop.
;

cmd5:   gosub   gohome                  ; Load nut and then drop it.
        gosub   loadnut
        acc     1                       ; slow backwards until drop.
        ret     1
        pos     rel -600
        wait    rdy
        return


cmd6:   goto    wave

cmd7:   goto    swing1

cmd8:   goto    swing2

cmd9:   goto    swing3

cmd10:  goto    swing4

cmd11:  gosub   gohome
cmd11n: if      0 = r23, cmd11e
        if      0 > r23, cmd11e
        gosub   swing1                  ; Eject it...
        sub     r23,1
        goto    cmd11N
cmd11e: return

;
;       CMD12 -- Second hand of watch, eject peanut every minute,
;                 until out of peanuts or other command.
;

cmd12:  gosub   gohome
        clr     tmr

cmd12l: let     r0,20/2         ; One minute.... in 30 seconds.
cmd12m: pos     rel - 83*2
        wait    tmr > 1000
        pos     rel -83*2
        wait    tmr > 2000
        pos     rel -84*2
        wait    tmr > 3000
        dec     tmr     3000
        if      0 < command, cmd12e     ; Exit if new command.
        loop    r0, cmd12m
        gosub   swing1                  ; Throw a nut.
        if      0 = command, cmd12      ; Start all over again if no new
cmd12e: return                          ; command.

;
;       CMD13 - Automateld med krossfunktion.
;

cmd13:  if      0 = r23, cmd13e
        if      0 > r23, cmd13e
        gosub   swing3                  ; Eject it...
        sub     r23,1
        goto    cmd13
cmd13e: return
```

```
;*****************************************************************
;
;       Perform complete swing, mode 1.
;

swing1: let     status, running
        gosub   loadnut
        gosub   fthrow
        return

;
;       Perform complete swing, mode 2.
;

swing2: let     status, running         ; Indicate status is running.
        Gosub   gohome                  ; Goto home position.
        Gosub   loadnut                 ; Load peanut.
        Gosub   jiggle                  ; Jiggle peanut in position.
        Gosub   ffire                   ; Fire!
        return                          ; Wait for next command.

;
;       Perform complete swing, mode 3.
;

swing3: let     status, running
        gosub   gohome
        gosub   loadnut
        gosub   jiggle
        gosub   bfire                   ; Backwards fire.
        return

;
;       Perform complete swing, mode 4 -- This empties revolver.
;

swing4: let     status, running
        acc     50
        speed   50 000
        wait    spd = 50 000

swi4l:  if      0 = peanuts, swi4e              ; Exit when no more peanuts.
        if      0 > peanuts, swi4e
        gosub   loadnut
        goto    swi4l

swi4e:  speed   0
        pos     speed   50 000
        pos abs 0
        wait    rdy
        return


;*****************************************************************
;*
;*      Wave the spoon and look silly...
;*
;*****************************************************************

wavef:  Acc      50             ; Just 80 turns forward.
        Ret      10
        Pos Speed  50000        ; This is 600 RPM.
        POS ABS 400000
        Wait    RDY

        clr     tmr
        wait    tmr > 1500      ; Pause...
```

```
        Acc       100           ; Then 80 turns back again.
        Ret       500
        POS ABS   0
        Wait     RDY
        CLR TMR
        Wait   TMR >  1000      ; Pause again.

        Acc       1500          ; 50 fast movements to impress audience.
        Ret       1500
        POS ABS   6250          ; To start postion...
        Wait   RDY
        CLR TMR
        Wait   TMR >  1000      ; 1 sec wait.
        LET       R1, 350       ; Inital time between movements.
        LET       R0, 50        ; Number of movements.

wavef1: POS REL   2500          ; 1/2 turn forward.
        CLR TMR
        wait     rdy            ; (to make sure it is not too fast)
        Wait   TMR >  R1        ; Wait
        SUB       R1, 5         ; Decrement wait time.

        Loop      R0,wavef1     ; Do it again.... 50 times.....

        wait     tmr > 1000     ; Puh!!!

        Acc       50            ; New parameters, for silly wave....
        Ret       50

wavef2: Push POS                ; Get current position.
        Pop       R0
        DIV2      R0            ; Div by 8.
        DIV2      R0
        DIV2      R0

        CLR TMR                 ; 100 ms wait...
        Wait   TMR >  100
        ABS       R0,-1         ; Make r0 negative.
        POS REL   R0            ; Go 1/8 of distance from current pos
                                ; towards 0.
        Wait   RDY
        If 0 >    R0,wavef2     ; If more to go..

        POS ABS   0             ; Then the last movement....
        wait     rdy
        return

;*****************************************************************
;*
;*      Forward FIRE with SERVE.
;*
;*****************************************************************

ffire:  acc       20            ; Backup to serve position.
        ret       20
        pos rel -130
        wait rdy

        Acc       200           ; Setup...
        Ret       250
        Pos Speed  30000
        CLR TMR                 ; Reference time.
        POS REL 500             ; Throw peanut into air.
        Wait     RDY

        Acc       1000          ; Setup for SMASH!
        Ret       1500
        Pos Speed  35000        ; This is the soft smash.
```

```
        POS ABS -1000           ; Backup....
        Wait  RDY

        Wait  TMR >  380        ; Wait until time to SMASH!
        POS ABS  4000           ; SMASH!!!!!
        Wait  RDY
        Return                  ; This is it!

;*****************************************************************
;*
;*      Backward FIRE with ZAPPING SMASH!
;*
;*****************************************************************

bfire:  acc     20              ; Goto serve position.
        ret     20
        pos rel 75
        wait rdy

        acc     200             ; Parameters for "throw into the air"
        ret     250
        pos speed 30 000

        clr     tmr             ; Ref-Time
        pos rel 500             ; Serve
        wait rdy

        acc     1000            ; Parameters for SUPER SMASH!
        ret     1500
        pos speed 100 000       ; This is 1200 RPM.

        wait    tmr > 350       ; Ready... Steady... GO!!!!
        pos abs -12000          ; SMAAAASHHHHH!
        wait    rdy
        return                  ; So, the world has one broken peanut more.

;*****************************************************************
;*
;*      Lazy throw away.... (forward)
;*
;*****************************************************************

fthrow: Acc     500             ; Good parameter for simple throw away.
        Ret     250
        Pos Speed  30000
        POS REL  5000           ; Go exactly one turn.
        Wait  RDY
        Return                  ; Then ready.

;*****************************************************************
;*
;*      eXtra Lazy throw away.... (forward)
;*
;*****************************************************************

xthrow: Acc     50      ; Good parameter for simple throw away.
        Ret     50
        Pos Speed  30000
        POS REL  2500           ; Go exactly 1/2 turn
        Wait  RDY
        pos rel -2500           ; and back again.
        wait rdy
        Return                  ; Then ready.
```

```
;*****************************************************************
;*
;*      Load peanut.
;*
;*****************************************************************

loadnut:
;
;       First we must check if there is any penuts in the revolver.
;
        if      0 = peanuts, nonuts     ; Error if nope.
        if      0 > peanuts, nonuts     ; Just to make sure....
        sub     peanuts, 1              ; Take a nut.
;
        SET Out  64             ; Start peanut revolvermotor.
ldn1:   If lo IN and   64,ldn1  ; Wait while sensor active
ldn2:   If hi IN and   64,ldn2  ; Wait while sensor inactive.
        CLR Out  64             ; Got active, stop motor.
        CLR TMR                 ; Timer = 0
        Wait  TMR >  150        ; Wait 150 ms for peanut to begin to fall.
        SET Out  64             ; Start motor again
        Wait  TMR >  350        ; Wait 200 ms more.
        CLR Out  64             ; Stop motor.
        Wait  TMR >  500        ; Wait 150 ms more for peanut to stabilize.
        Return
;
;       No Nuts, blink with the lamps....
;
nonuts: let     r0,63
nonu1:  clr out 63
        set out r0
        clr     tmr
        wait tmr > 15
        loop    r0, nonu1
        clr out 63
        goto    wcmd            ; Wait for new command.


;*****************************************************************
;*
;*      INIT PDI -- Find home position.
;*
;*****************************************************************

initpdi:
        Ref   POS    0          ; Setup system parameters etc...
        Scale    145
        Zero     192
        Pole     0
        P Gain   13
        vector  8, fposerr      ; Setup vector for position error.
        pos err 1250            ; 1/4 turn is fatal.....
        Enable REF              ; Want to look for ref-pulse.
        Acc      50
        Ret      50
        Speed    1000           ; Slow forward.
initp1: If lo IN and 128,initp1 ; Wait util i find it.
        Ref   POS    700        ; Yep, this is pos 700 rel to launch pos.

        Pos Speed  10000                ; Goto launch psition.
        POS ABS  0
        Wait  RDY
        Return
```

```
;*****************************************************************
;*
;*      GOHOME -- Goto launch position and setup standard ACC & RET.
;*
;*****************************************************************

gohome: Acc      100
        Ret      100
        Pos Speed 10000
        POS ABS  0                      ; go away...
        Wait  RDY
        Return

;*****************************************************************
;*
;*      JIGGLE -- Jiggle peanut in position.
;*
;*****************************************************************

jiggle: LET      R0, 40                 ; Number of times to jiggle.
        LET      R1, 15000              ; Jiggle force.
        LET      R2,-15000

jloop:  Profile ACC  440                ; Jiggle forwards
        Wait  RDY
        Profile ACC -440                ; and then back again.
        Wait  RDY
        CLR TMR
        SUB      R1, 375                ; Decremnt jiggle force.
        ADD      R2, 375
        Loop     R0,jloop               ; Then ready for next jiggle.
        clr     tmr                     ; Wait 0.5 seconds more for
        wait    tmr > 500               ; panut to stablize.
        Return

440     P Data   R1,  5                 ; Data for jiggle.
441     P Data   R2, 10
442     P Data   R1,  5
443     P Data   0,  0

;*****************************************************************
;*
;*      FPOSERR -- Fatal positioning error, close down shop.
;*
;*****************************************************************

fposerr:
        scale   0               ; Emergeny stop!

        pos     abort           ; Stop internal ramp gen etc...
        acc     1000
        speed   0

fposer1:
        let     status, stsidle ; Flag that we are offline.
        clr out 127
        clr     tmr
        wait    tmr > 200
        set out 63
        wait    tmr > 400
        goto    fposer1         ; Loop here until user aborts...
```

```
;*****************************************************************
;*
;*      Control-C trap, go here when user types Control-C.
;*
;*****************************************************************

ctrlc:  vector   10,ctrlc       ; Setup Control-C trap again.
        speed    0
        acc      500
        pos      abort          ; This stops main motor gracefully
        Out       0             ; This makes sure revolver motor is off.
        let      status, stsidle
        STOP                    ; Then we can stop program.
        Goto      restart


;*****************************************************************
;*
;*      RANDOM BLINK ROUTINE.
;*
;*****************************************************************

rblink: push     tmr
        pop      r0
        sub      r0,200
        if       0 > r0, rblret

        add      r10, 13611     ; This is a 16 bit random generator.
        mul      r10, 271
        and      r10, 65535     ; Do only keep the 16 lsb's
        let      r11, r10
        div2     r11
        div2     r11
        div2     r11
        div2     r11
        and      r11, 63        ; Want to random blink with 6 LED's
        clr out  63
        set out  r11
        let      r12,r10
        and      r12,127
        add      r12,50
        dec      tmr r12        ; Decrment the timer.

rblret: return                  ; No change, return.
;
;       SLUT!!!!
;
=================================================================
```

A boy scout was out doing his bob-a-job stint one Saturday in Farmborough. He walked up to the front door of one house and rang the doorbell. The owner appeared.
"Yes?"
"Bob a job week, sir!"
At first the man didn't want anything to do with the kid, but eventually he agreed to give him a job. "You can paint my porch for me", he said. "The paint and brushes are in the garage—here's the key." The boy scout toddled off to do the job. Two hours later, he rang the doorbell.
"Job's done", he said, his palm outstretched.
"Harumph. Took your time, didn't you ? Well, okay, here's your FIVE PENCE!"
The boy took the money and started to walk away, but after a few paces he turned around and said, "Oh, thanks for the donation, but by the way, it wasn't a porch, it was a Ferrari!"