STACKPOINTER

5-1986. Organ för Datorföreningen STACKEN, KTH.

Datorföreningen STACKEN

STACKEN är datorföreningen på KTH. Vi har en mängd intressanta verksamheter på gång. Dock hänger det ytterst på den enskilde medlemmen att avgöra vad han eller hon vill göra för föreningen. STACKEN är en idéell förening, där intresse för datorer är den gemensamma faktorn. Sedan föreningen grundades 1978 har vi (bland annat) åstadkommit:

- samköp av mikrodatorbyggsatser
- diverse kurser och föredrag
- studiebesök på intressanta ställen
- AMIS, den portabla EMACS-kompatibla editorn för TOPS-10, VMS, PRIME, NORD, ...
- STACKPOINTER, föreningens tidning
- en egen datorhall för våra stordatorer
- en egen DEC-10:a (se nedan)

Vi har sedan ett år tillbaka en egen DEC-10, som vi installerat, felsökt och kör på. Det är en gammal modell med KA10-processor. Vi kallar henne KATIA. Hon står i vår nya klubblokal och maskinhall "B30", på Brinellvägen 30 (V-sektionen) på gaveln mot Lill-Jansskogen (där vi har en egen ingång). En våning ovanför finns en hörsal, där vi håller till vid större möten.

Ordinarie möten är första torsdag i varje månad kl 19 i antingen lokalen eller hörsalen. Är Du intresserad av föreningen, är Du välkommen till något av våra möten. Vill Du sedan bli medlem, lämnar Du in en skriftlig ansökan till styrelsen eller skickar den till vår postadress:

Datorföreningen STACKEN
c/o NADA
KTH
100 44 STOCKHOLM

Postgiro: 433 01 15-9.

Bankgiro: 344-3595.

Kallelse till höstmöte

Härmed kallas till höstmöte i Datorföreningen STACKEN. Mötet kommer att hållas torsdagen 1986-12-04, med början klockan 19, antingen i hörsal V4 eller i STACKENs lokaler i våningen under, på Brinellvägen 32 på Kungliga Tekniska Högskolan. Förslag till dagordning:

§1. Mötets öppnande.
§2. Val av justeringsmän.
§3. Val av mötesordförande.
§4. Val av mötessekreterare.
§5. Tillkännagivande av uppgjord röstlängd.
§6. Frågan om mötet är stadgeenligt utlyst.
§7. Frågan om dagordningens godkännande.
§8. Val av styrelse.
§9. Fastställande av firmatecknare.
§10. Val av revisorer.
§11. Utseende av ny valberedning.
§12. Fastställande av medlemsavgift för 1987.
§13. Fastställande av budget för 1987.
§14. Fastställande av mötesdagar för 1987.
§15. Övriga frågor.
§16. Mötets avslutande.

För styrelsen, *Hans Nordström*

Richard M Stallman i Stockholm

Torsdagen den 30:e oktober höll Richard M Stallman en föreläsning på KTH. Föreläsningssalen var helt fylld, det var många som var nyfikna på vad han skulle säga, "den siste riktige hackern", som han kallas. RMS, som han kallas, är ju ganska känd som mannen bakom editorn EMACS och den har ju ett grundmurat rykte. Talet handlade om den gamla goda tiden på MIT, vad som var speciellt med den, och om GNU-projektet, varför det startades, och lite om vad det innebär, samt lite om varför programvara skall vara fri. Han såg verkligen ut som man hade väntat sig och fått höra, långt mörkt hår, skägg och mustasch, ett stort märke "GNU-isance" på bröstet, och en flöjt i fickan. Efter föreläsningen och frågestunden visade STACKEN sin DEC-10 (KA10) och demonstrerade operativsystemet ITS på en DEC-2020. ITS har skrivits på MIT.

Han kom till MIT AI-lab (Artificial Intelligence Lab) 1969, då datorterminaler fortfarande var ganska sällsynta, flera fick dela på dem. Inställningen på MIT var att terminalerna var allmänna, så att de skulle kunna användas av så många som möjligt för att skriva program. Om en professor låste in en terminal i sitt rum, så kunde studenterna slå in dörren till rummet för att komma åt terminalen, som läxa åt den som låser in så allmännyttiga tingestar. För maskinerna fanns ett serviceavtal, men oftast reparerades maskinerna av MIT-folk. Servicekillarna fick bara leverera delarna. På 60-talet utvecklade MIT-folk maskinerna från Digital, lade till nya instruktioner, index register, nya time-share features... Idag är det "omöjligt" att lägga till nya saker, eftersom det inte går att ändra kretsarna som när det var transistorer.

Hackertraditionen att arbeta på natten kom sig ursprungligen av att PDP-1 maskinen var en enanvändarmaskin, så man fick boka tid, och det fanns mest tid på natten. Detta har fortsatt till nutid. Ni kanske har suttit en natt själva? En annan hackertradition var att ha platser att sova på. RMS bodde själv några månader på MIT när han inte hade annan bostad. Det fanns ingen file protection på

deras maskiner, källkoderna var fritt tillgängliga. Det gjorde att det var lätt att lära sig bra programmering, och det gick mycket fortare att fixa fel i programvaran. Den mest lämpade gjorde ändringarna utan att behöva vänta på leverantören. Allt var öppet för alla, och vid ett tillfälle klagade en chockad ITS-användare över att han hade mailat en fråga till en person, och en stund senare fått svar från någon annan!! Visserligen var svaret riktigt – men inte läser man andras mail!? Det var en väloljad anarki, och den utövade stor dragningskraft på andra studenter. De var rädda för att någon skulle komma och förstöra den fina anda som de hade på MIT-AI. I synnerhet trodde man hotet kom från MIT-Computer Science, där chefen ville bli större, och han var en sån typ som ville ge order.

Det verkliga hotet mot MIT-andan visade sig inte komma från andra institutioner eller personer, utan från ett håll som de aldrig hade kunnat förutse: 80-talets kommersialism. Alla kompetenta programmerare köptes iväg, och nybörjarna hade nu inga erfarna hackers att fråga och få inspiration från, och dragningskraften avtog från MIT-AI. De kommersiella programvarorna gjorde sitt intrång nu när ingen längre kunde underhålla programvaran, de kommersiella operativsystemen som Twenex med sitt filskydd kom, och ingen kunde längre ändra systembuggar snabbt. Dessa system var dessutom sämre skrivna, "monkey-code". Idag är MIT inte alls vad det en gång var...

Det var just förlusten av den fina stämningen på MIT som ledde fram till GNU-projektet. RMS hoppas kunna skapa en ny gemenskap av fritt hackande. Arbetet började för två och ett halvt år sedan, och idag är ungefär hälften av arbetet gjort. RMS såg i begynnelsen två alternativ.

1) ett LISP-maskin-system, på standard hårdvara,

2) ett standard OS, Unix-likt, för att göra det lätt att byta.

Valet föll på alternativ 2, eftersom en LISP-mskin måste innehålla speciell hårdvara, eller ett operativssytem som tar hand om fel. LISP-alternativet skulle i fallet standardmaskin inneburit att han behövt skriva både ett OS och en LISP, nu valde han det senare alternativet.

Namnet betyder "Gnu's Not Unix" om ni undrade. Först

behövde han en kompilator, och sökte efter en som var fri. Han hittade ingen, så det första projektet var en C-kompilator. Efter det kom GNU-Emacs, skriven i Lisp. Lispen är skriven i C, och innehåller sin egen Lisp-interpretatator. Han fick tag på Gosling EMACS, som var skriven i Mock-Lisp, och hoppades kunna använda den. Det visade sig att det mesta behövde skrivas om, när man fick en riktig Lisp, och ville få med Lisp-objekt. När det var färdigt fick han problem med Gosling, och var tvungen skriva om även resten... Det tog 1 1/2 vecka. Nu utvecklade han en debugger för C. Den verkar verkligen ha fina egenskaper; användardefinierade kommandon, kommandofiler, pekare, temporära variabler mm. Man kan även förse sina egna kommandon med dokumentation, styra flödet i programmet, ja den tillåter dig att finna ut nästan allt som man önskat sig. C-kompilatorn ger den bästa koden av C-kompilatorer på VAX, och den är enkel att portera. Det tog en och en halv dag att få den till VAX. Den släpps om några månader. Kärnan TRIX baseras på remote procedure calls, att anropa ett domänprogram. Systemet kommer till slut att innehålla: fil-version, undelete, recognition, backup information, nya (inkompletta) versioner kommer att vara osynliga och finnas under ett temporärt namn, 8-bit-support, atomic update on a file, nätverkssupport TCP/IP, Kermit kanske kan användas som en UUCP. Det finns redan en hel del programvara, och inte så mycket återstår. Det enda som inte finns, och som han inte beslutat om han skall sätta in, det är filskydd...

RMS tycker att programvara skall vara fri, av flera skäl. Det finns några olika sätt att se på ett program:

1). Program= Matematisk formel
2). Program= Recept (matrecept)
3). Program= Bok, med copyright

Av alla dessa valde man det konstigaste, att kalla ett program för en bok. I början fanns det mycket fri programvara, ja till och med Digitals operativssytem var fria. Att belägga programvara med spridningsförbud är : skadligt för moralen, ondsint, cyniskt, och förorsakar förfall i samhället. Idén att man äger information är skadlig : (Passar det universiteten?)

1. Skapar färre användare av programmet, men inte mindre arbete med att skriva det.

2. Problem när användare vill ändra. Folk gillar att ändra eller lägga till effekter, och det kan du vanligen inte.

3. Ett område expanderar snabbast när kunskapen inom området delas. Nybörjare behöver något att starta med. Programmerare slösar sin tid på dubbelarbete, och det har skapats en brist på programmerare. Vi verkar behöva dubbet så många som vi verkligen behöver. Det talas om att öka produktiviteten hos programmerarna, men inte om att minska deras antal. Det är samma anda som får oss att tyst åse ett mord på gatan!

RMS beslutade sig för att försöka kämpa mot detta, på sitt sätt. Det han är bra på är ju programmering. Han tycker att man kan dra paralleller mellan maffians beskydd, och programvarusäljarnas beskydd mot polisanmälan, mot rundlig avgift. Man skall skilja på information och egendom. Om vi gör oss fri från konstgjorda brister (artificial scarcity) så kommer vi inte att behöva arbeta lika mycket!

Efter föreläsningen bidrog NADA med en check till Free Software Found! Under frågestunden fick vi höra ett förslag från RMS, en "software tax", som skulle fungera så att man får rösträtt i förhållande till hur mycket maskinvara man köpt, och man röstar om hur mycket skatten skall vara. Skatten man skall betala får man betala till fria stiftelser, och därigenom kan man välja vart bidraget skall gå. Vill man, så kan man sättta upp en egen stiftelse för utvecklig av programvara på det område man själv önskar. Det är väl inte helt realistiskt idag, ansåg han själv. För den som inte har skall man skaffa X-Windows från MIT, en fri programvara, bättre än de kommersiella. GNU skrivs för 32-bitsmaskiner med byte-adressering, dvs kommer troligen inte på DEC-10/20. En liten diskution om maskinarkitekturer gav vid handen att PDP-1 och PDP-10 var bra, men även följande kan gå an (i ordning nedåt): VAX - 32000 - 6800 - Intel (just piss). Vi fick veta att "C combines the power of assembler language with the convenience of assembler language".

Free Software Foundation
1000 Mass. Ave.
Cambridge, MA 02138
USA

Richard M Stallman
545 Tech Sq rm 703
Cambridge, MA 02139
USA

rms@prep.ai.mit.edu

enea!seismo!prep.ai.mit.edu!rms

När RMS hade gått, visades STACKENs DEC-10 Katia, och operativsystemet ITS visades på en DEC-2020, där vi kunde provköra. ITS har inget filskydd, alla kommer åt allt. Många roade sig med att logga in som RMS, även i USA brukade det sitta en 2-3 st inloggade som RMS... Nu är det fler på MIT som vill hacka ITS än Twenex, och trots vissa svårigheter att komma överens med OS kan jag förstå det. Alla processer är självständiga, om man vill, så man kan byta process, eller tom jobb precis när man vill. ITS har ingen kommandoavtolkare, så man kör genom DDT. För nybörjarna finns det färdiga kommandon, som RMAIL, FINGER, LISTF och mera. Det finns den riktiga EMACSen. Man måste själv KILLa en process. Det som saknas är recognition och versioner, men det mesta som saknas i ITS finns ju inte ens i kommersiella Os, som VMS... ITS kan köra alla TOPS-10 program förutom ITS program, så det finns gott om kompilatorer och övrig programvara.

Jan Lien

Hur man trimmar sin PDP-11

Även riktiga datorer kan ha användning för upptrimning och ditskruvade saker från HOBBEX. Så här går det till.

När vi först fick vår egen PDP-11/34 med 128k minne, dubbla 80 MByte CDC hårddiskar och tre Hazeltine 1500 terminaler tyckte vi att vi hade lyckats lägga vantarna på någonting väldigt speciellt. FNYYS, mikrodatorer är bara leksaker, det är därför vi bara kan få två av trettio system att fungera samtidigt. Det här däremot, är en riktig dator. Den kostar en halv miljon, mer än vad nån av oss tjänar på ett år. Den här kommer inte göra oss besvikna.

När Control Data Corporation grymt hade tagit oss ur denna villfarelse kunde vi se 11:an som den verkligen var: en hög med plastförpackade EPA-produkter, precis så många att man kunde få igång en eländig version av fleranvändar STAR TREK skriven i virtuell LISP.

Detta förde oss fram till krisen. Vi noterade att när alla tre spelarna försökte docka sitt skepp vid basen medan LISP interpretatorn gjorde Garbage Collection så uppstod en fördröjning på (i värsta fall) 0.2 sekunder. Detta var som alla förstår extremt irriterande och krävde omedelbar eller tidigare åtgärd.

Det första steget var att korrigera ett stukat minneskort. Detta görs bäst med en kofot genom att placera kofoten under kortet och bända. Var inte rädd för att använda våld. När kretskortet har rätats ut kan elektronerna flyta längs räta linjer i stället för att tvingas sakta ner i kurvorna. Det är välkänt att ljus (och elektroner) går snabbare längs räta linjer.

Detta hjälpte inte upp problemet i så hög grad så nästa steg var att rensa upp de igenkläggade kretsarna i CPUn. När datorn kör samlas överskottselektroner upp i undangömnda skrymslen och vrår, speciellt inuti de integrerade kretsarna. Dessa felaktiga elektroner kan slöa ner dessa kretsar och minska drivförmåga och tillförlitlighet. Den bästa lösningen på det problemet är att skaffa en elektronextraktor. (HOBBEX 47-6665-3). Tyvärr är det ingen lagervara. Denna kan man

använda för att suga upp elektronerna.

Det pågår en intensiv forskning på Chalmers Tekniska Högskola för att få reda på var och varför elektronerna fastnar. En teori går ut på att vissa datavägar används enbart av lågfrekvent exekverade instruktioner . Det kommer alltså sällan något tillflöde av elektroner vilket stagnerar det interna flödet.

Olyckligtsvis hjälpte inte heller denna teknik upp tillräckligt och slutligen kom vi fram till att en vanlig PDP 11/34 har kanske inte den prestanda man vill ha för att köra vår sofistikerade fleranvändar-STAR TREK skriven i virtuell LISP. Man behöver någonting vassare.

Vi hade läst i några mindre respektabla datormagasin att man kan skruva fast en turbotillsats till datorn och då kan förvänta sig en ökning av hastigheten med 100 b.p.s. Då företagets finansiella resurser var i botten efter en ännu icke uppklarad förskingringshärva, fick vi låna av en av företagets kamrerer som hade vunnit massor med pengar på Lotto. Det är skönt att ha nån tillförlitlig hjälp när man verkligen vill ha en turbotillsats till sin PDP-11/34. Vi beställde då turbotillsatsen och väntade under svordomar över datorn varje gång alla tre spelarna försökte docka sina rymdskepp medan den virtuella LISPen gjorde Garbage Collection. Programmet är egentligen inte skrivet i virtuell LISP utan i TRAC. TRAC är ett språk som interpreteras av den virtuella LISPen.

Vi blev genast väldigt nöjda med turbotillsatsen när den anlände, komplett med en och en halv sida dokumentation som förklarade dess olika finesser och hur man kopplar in den. Installationen gjordes snabbt av en av våra servicekillar. Den tog bara en timme eller så. Tyvärr dök en representant för den lokala servicebyrån upp under tiden och han gillade inte alls vårt initiativ. (Det är välkänt att DEC inte har servat system på många år). Då vi insåg redan från start att han skulle dyka upp eftersom det var den enda gången sen leveransen av systemet vi verkligen inte ville ha honom där var vi förberedda och lyckades avleda honom till en IBM PC/AT som hade ett defekt batteri så att konfigureringen försvann. Detta gav oss en chans att göra klart installationen av turbotillsatsen. Senare kom representanten tillbaka och sa att batteriet hade

helt enkelt tagit slut efter tre år och att IBM tyvärr bara producerade dessa specialbatterier för nya maskiner.

När installationen var komplett lät vi vår residenta bitfiddlare ta över och efter ett (nära) förstörande test ansåg han den säker för användning. Nu är det på gränsen till upphetsande att sitta vid konsolen på det här kraftpaketet. Det högst upprörande problemet med STAR TREK spelet har försvunnit totalt kan vi rapportera. Vårt nästa projekt blir att köra en SPICE modell för en Ada kompilator i hårdvara. Tyvärr är SPICE skriven i Fortran, för vilket vi inte har en kompilator, men Fortran kan man interpretera i virtuell LISP. Vi har nämligen ett avancerat ADA program för att göra direkt cursor addressering på en Hazeltine.

Fascinerande, eller hur!

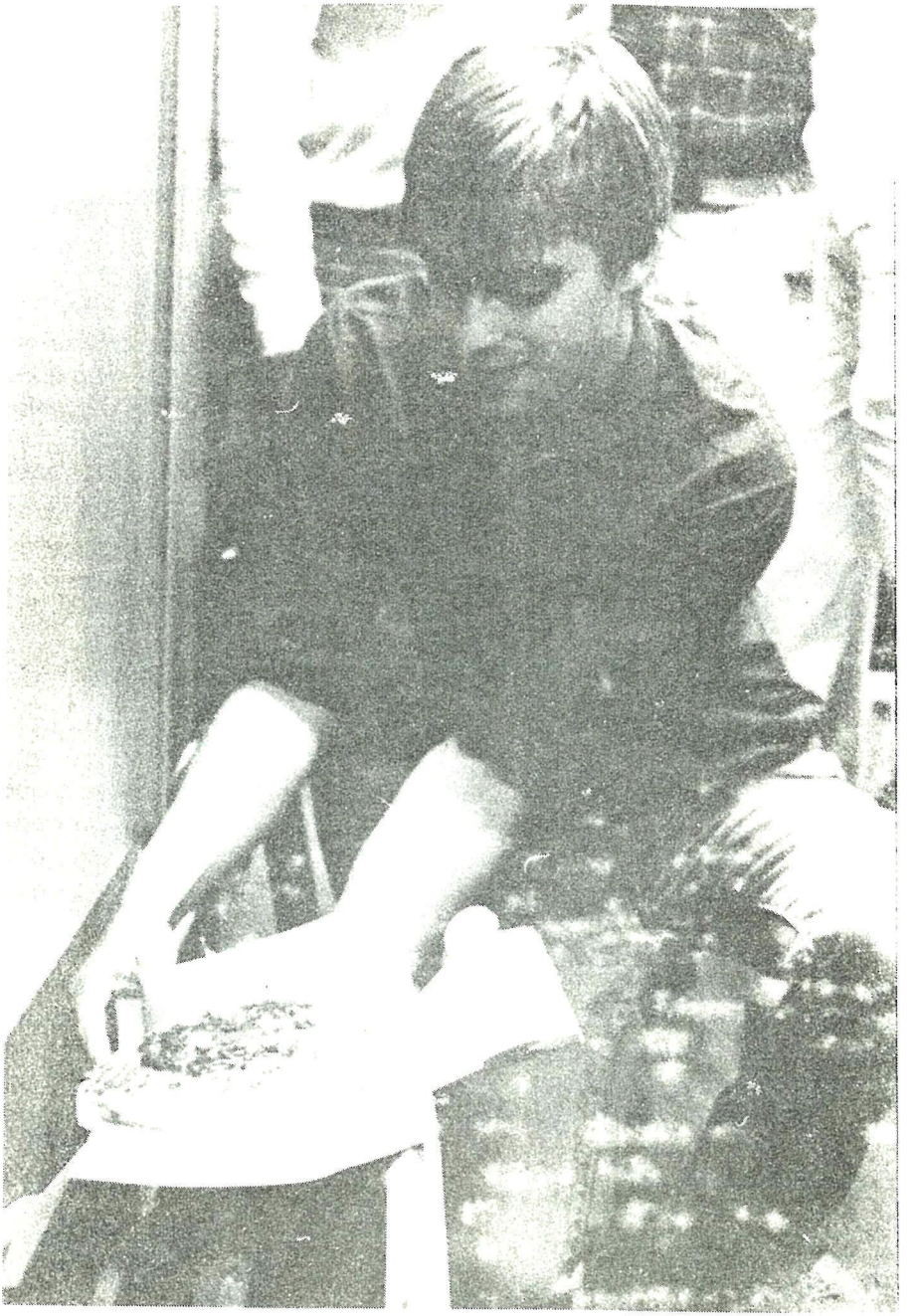
Har vi tjudrat Elvira?

Alfa-Laval hade en PDP-11/40, som de inte längre använde och den tog vi naturligtvis tacksamt emot. Det var ett komplett, fungerande PDP-11-system, med diskar, bandstation och 8 terminalportar. Vi fick även med kablar, manualer, ritningar och allt annat som hör till. Där var ordning och reda, heder åt dem som haft hand om den maskinen! När vi hade lastat in alltihopa i lastbilen fick vi en mycket Alfa-Lavalsk fråga: "Har ni tjudrat datorn ordentligt?"

Vi har genererat ett RSTS/E-system till den och läst ner de senaste backuperna från Elvira. Nu är frågan: Vad ska vi kalla den? Det är ju inte Elviras hårdvara, den har Hedindata.

Elvira var en PDP-11/45, E-sektionens teknologdator, känd som den första datorn på KTH där teknologerna kunde köra så mycket de ville, om det var någon som inte visste det.

Jan Michael Rynning



Katia ett år

Katias ettårskalas firades – inte i dagarna tre – men väl med tårtorna tre, och Katia själv fick blåsa ut ljuset. Den egentligen något drygt tioåriga damen var vid god vigör, men kunde detta till trots inte få i sig någon tårta, trots att Lars Höglund öppnade ventilationsgallret och nästan försökte mata henne.

Jan Michael Rynning

Katia på ANF10

Natten mellan den 10 och 11 november fick vi igång Calle – bestående av en DL10 vi fått från Medicindata i Göteborg och en PDP-11/40 från SSAB i Borlänge – och kunde på så sätt koppla in Katia på ANF10-nätet, tillsammans med Kicki, Nadja och alla de andra. Och kan ni tänka er, vad gör hon då som tack för det? Jo, hon sätter igång att glufsa i sig alla terminallinjer hon kan få tag på! Ty så gjorde man i Tops-10 version 6.03A, om man var en väluppfostrad dator. Det har vi nu halvt om halvt lärt henne av med och efter ytterligare justeringar av programvaran ska vi nog få filöverföringar att fungera och göra henne till en rumsren medlem av ANF10-gänget.

Jan Michael Rynning

```
[ANF10 network: connected to KICKI(20), located at KICKI(20), 14 nodes]
Node    ODEN    (1)     Oden/QZ Stockholm 7.02  22-May-86
Node    HUGIN   (2)     DN87SX V23(210) 1986-04-10
Node    MUNIN   (3)     DN87SX V23(210) 1986-09-29
Node    BALDER  (4)     DN87SX V23(210) 1986-09-29
Node    AURORA  (6)     Aurora SU Dec-2020 7.02 6-Nov-85
Node    NADJA   (7)     Nadja KTH Dec-2020 7.02 10-Apr-86
Node    VENUS   (10)    Venus Dec-2020 7.02     10-Apr-86
Node    FILIP   (11)    FOA3 Link|ping 7.02     06-13-86
Node    FILIPA  (12)    DN20 V23(210)   9-JAN-84
Node    KICKI   (20)    Kicki Stockholm 7.02A   10-05-86
Node    JENJEN  (21)    DN87 V24(227)   3-SEP-85
Node    JANUS   (41)    DN200 V23(210)  9-JAN-84
Node    KATIA   (50)    Stacken, DEC-1040, 6.03A        10-27-86
Node    CALLE   (51)    DN87 V23(210)   9-JAN-84
```

SM i programmering

När "SM i programmering", med underrubrik "administrativa tillämpningar", avhölls i Södertälje, helgen 4–5 oktober – 24 timmars kontinuerlig programmering – så var STACKEN där och visade upp sig.

Vi ställde inte upp i själva tävlingen – vem vill väl betala 5000 för att få skriva administrativa tillämpningar på sin fritid – men vi var inbjudna att delta i en utställning som fanns i anslutning till den. Så vi tog dit ett modem, en grafisk terminal med skrivarutgång och en POSTSCRIPT-laserskrivare och satt och körde grafik, TEX och AMIS på Kicki hela helgen och passade samtidigt på att sätta ihop förra numret av STACKPOINTER.

Detta väckte viss förundran, inte minst hos några av deltagarna, som var helt inställda på administrativa program och fjärde generationens databashanterare och hade något svårt att koppla det till vad vi satt där och gjorde.

Jan Michael Rynning

Datorers användning i VERKLIGHETEN

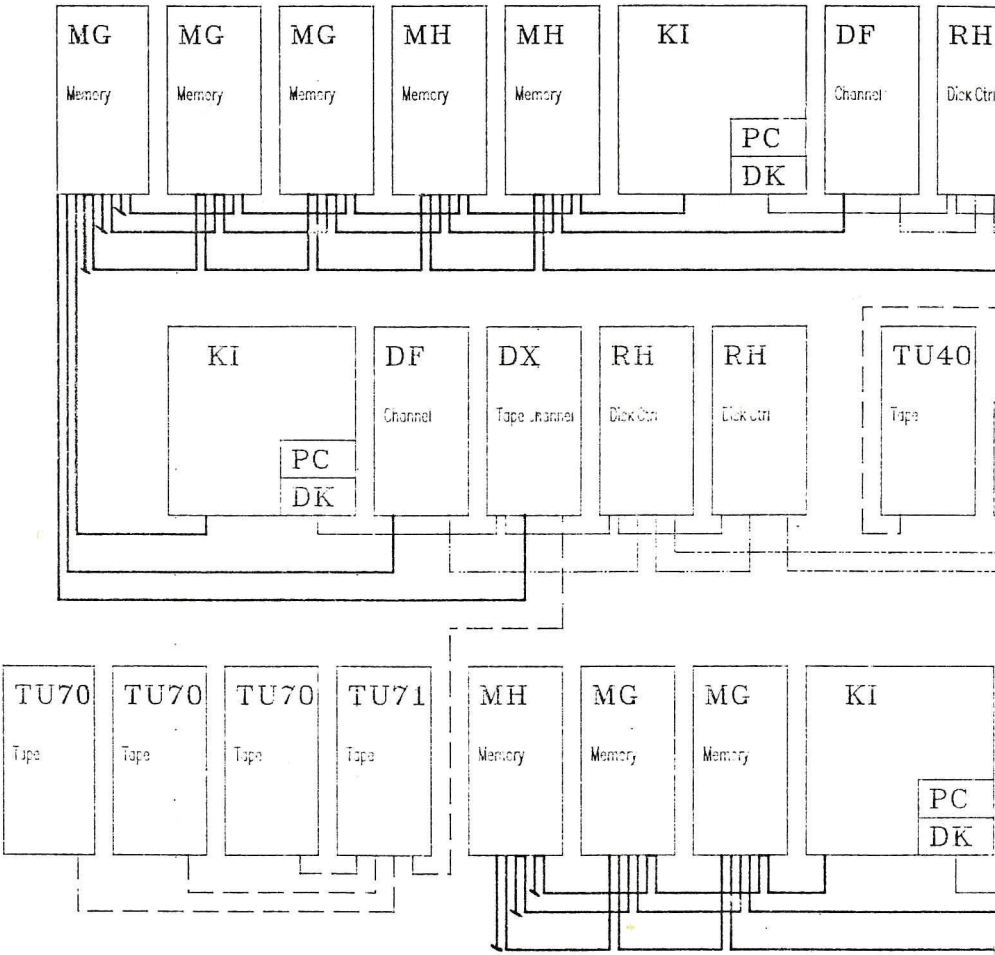
Ryktet förtäljer att företaget FÖRETAGET hade köpt in ett stort och fint materialstyrningssystem för att slippa veta vad man skulle beställa och när man skulle beställa. På så sätt skulle man spara in MASSOR med PENGAR. Leverantören LEVERANTÖREN hade sagt att det här var det bästa pengar kunde köpa och det var nog bra för kostade, det gjorde det.

En dag bestämde man sig på FÖRETAGET att man skulle skrota ett antal maskiner som man hade fått tillbaks i utbyte för nya maskiner man hade levererat. De var i och för sig dyra, de kostade runt en miljon styck, men var gammalmodiga. Någon talade om för det fina materialstyrningssystemet att det skulle skrota 100 maskiner. Det här ställde till vissa problem för det fina materialstyrningssystemet för hur det än räknade så kunde det bara hitta 68 stycken. Nåja, tänkte det fina materialstyrningssystemet. Problem är till för att lösas. Den tänkte ett bra tag... Efter en stund hade den kommit på en lösning.

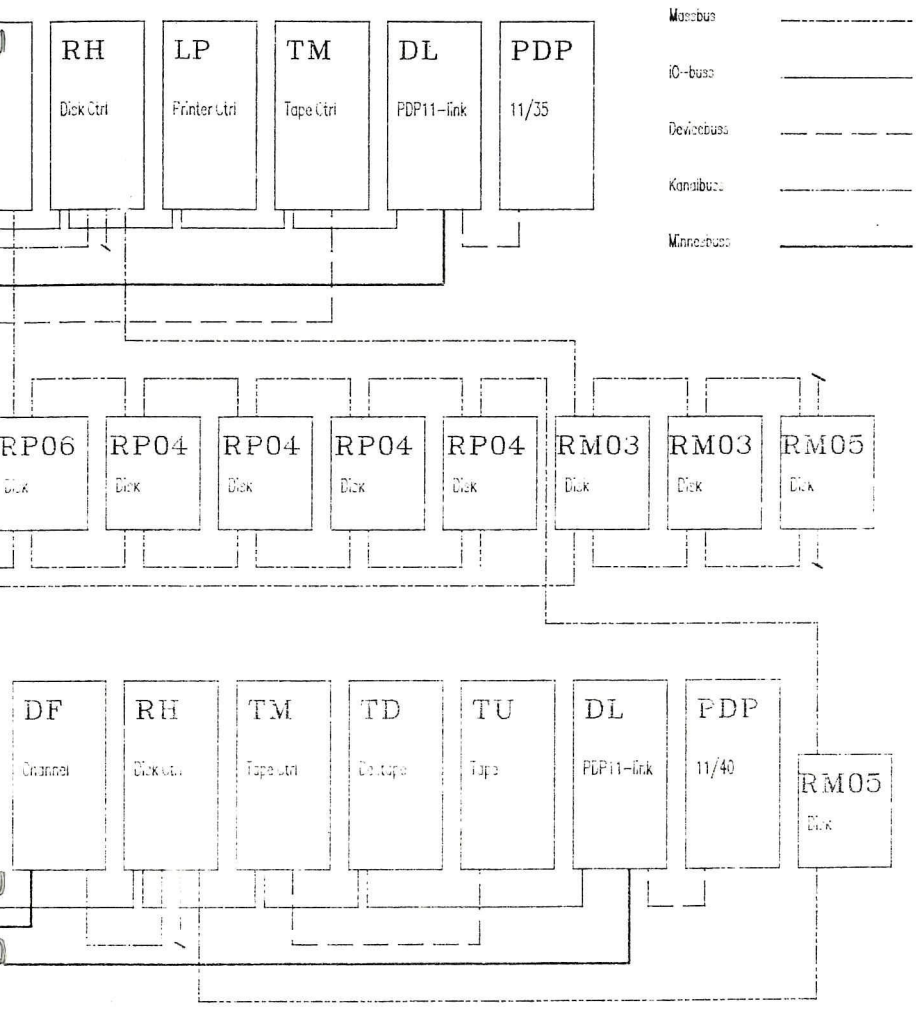
En tid senare så gick fabrikschefen på FÖRETAGET omkring och kände sig verkligt nöjd med det nya, visserligen dyra men fina materialstyrningssystemet då han plötsligt passerade förbi de skrotade maskinerna. Han fortsatte ett par steg och hajade plötsligt till. Det var någonting som inte stämde. Han kallade till sig några tekniker för att fråga ut dem men insåg i samma ögonblick vad som var fel. De här maskinerna såg ju alldeles nya ut! En kontroll visade vad som hade hänt. Det fina materialstyrningssystemet hade räknat ut att om man producerade 32 maskiner så skulle man kunna skrota alla 100 maskiner som var begärda.

Ulf Samuelsson

Colossal Cave Computi



ng Center Connections



Känns de igen?

Följande stycken har stulits från ett uråldrigt nummer av DATAMATION, och är delar av en större artikel som behandlar (data)personalen på en firma med finansiell inriktning.

Crypto

One of the applications programmers is named "Crypto" Guess. Crypto loves a good puzzle above all. He has ben in data processing for years, and is a product of the times when a programmer's skill was measured by how obscurely he wrote his programs.

He remembers the good old days fondly, and every program he writes is a tribute to them. His life is dedicated to the quest for the ultimate puzzle– to solve it or create it. Rubik's Cube was like a honeymoon to him. He still has the first one he solved, lacquered and sitting on his desk. Seen before 8 a.m., he will be working the Times crossword puzzle.

Crypto is a truly gifted programmer, provided that someone does not have to modify or try to understand one of is programs. Unfortuanately, his ideal career match, that of designing burglar-proof tombs for Egyptian pharaos, died out 30 centuries ago. He loves the esoteric and was enraptured when he discovered that writing data of various lenght on the new tape drives actually caused them to sing. The weekend that he was supposed to be helping Hashmark on the payroll tax problem he spent writing a program to play "Ballad of the Green Beret" on the tape drive.

His speech is peppered with acronyms and he is obsessed with DCBs, DFTs, ASCBs, CVTs, and other such creatures lurking in the bowels of the computer. He does not care for financial applications or things financial in general. He is the rare employee who forgets to cash his paycheck and must be reminded to do so later by irate payroll clerks struggling with bank reconcilliations.

Crypto's peculiar skills are very useful in one application– security. He can locate the weakness in any package's security system. He should not,

however, write code.

Bits

The final data processing person is the systems programmer, "Bits" Flogger. Bits is not often sighted in daytime. He shuns bright lights and prefers the company of his beloved computer. Bits is easy to recognize. He wears a rumpled white shirt with shirttail out, slacks with shiny knees and seat, wing-tip shoes, and he drives a car with a SUPPORT YOUR LOCAL POLICE bumper sticker. (A couple of years ago Bits wore hiking boots and sported a SAVE THE WHALES bumper sticker. He is the same person; he just crossd the generation gap.)

Bits carries a battered briefcase and has a plastic penholder in his pocket with a full complement of colored felt-tip pens, along with an IBM 370 Reference Summary card. He often carries a calculator strapped to his belt, too.

Bits refers to computer users as "abusers" and only grudgingly accepts them as a necessary evil to allow him to have access to a computer. He is rather abrasive with people, but no one speaks to him about it because hi might split to another company and boost his salary to over $50000 a year.

Bits never gets involved in a package implementation of his own volition. He should, however, be cultivated, perhaps by bribing him with old Buck Rogers comics of something else he values. Bits knows the system and can solve problems. He can make things work. He is one ot the few people in the company who is a true expert in his job.

Datorer och oväsen

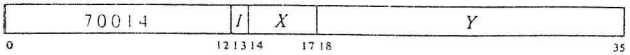
Den som har lusläst sin PDP-10 Processor Reference Manual vet att i en KI10-processor (centralenheten i Kicki) finns en inbyggd digital-till-analog-omvandlare, utspänningen kan hittas på pinne 2S02V2 i bakplanet. Visserligen är det bara sex bitars upplösning, men ändå... Efter ett tag väcks tanken: Kan man månne använda den till att göra något som påminner om musik med? Jo, kanske.

Det första problemet, att koppla in förstärkare/högtalare löses med hjälp av ett litet hembygge. Nästa problem är hur man gör ljud. Genom att med jämna mellanrum, och dessutom ganska fort skicka ut data på D/A-omvandlaren borde man kunna få det att låta. Mellanrummen åstakommer man med hjälp av en intervallräknare, den heter DK10 och bor i CPUn. Den kan generera interrupt med $n \times 10$ mikrosekunders mellanrum, här väljs $n=10$, vilket ger 10 kHz samplingsfrekvens. Sen kommer det roligaste, Tops-10 har funktioner som tillåter användarjobb att göra fysisk I/O samt ta hand om interrupt, under pågående timesharing! Inget strul med att stänga av maskinen för att spela på den.

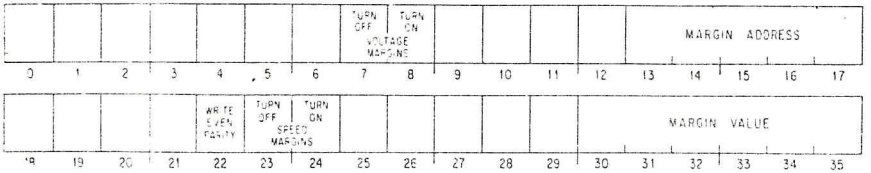
Den första versionen av programmet som för oväsen består av tre delar: Ett huvudprogram, med interruptrutin, samt två tabeller, den ena beskriver hur olika toner ser ut, den andra är en sekvens av toner och hur länge. Med hjälp av ett snabbt hopkladdat program i PASCAL (och flyttal, ugh!) genererades tabellen som besriver hur olika toner ser ut. Till uruppförandet valdes en vanlig skala, 12 toner efter varandra i stigande ordning. Dax att länka ihop och prova. Jodå, det låter, men... Det visade sig att det fanns rutiner i monitorn som fick för sig att ibland, om man hade otur, peta på DK10an, och ändra på saker och ting. En dos FILDDT fixade den detaljen. Nästa försök. Jodå, nu spelar den faktiskt en tolvtonsskala, som önskat. Som nästa stycke att spela valdes "Ack Värmeland Du Sköna", och efter mycket pusslande med längden och höjden på tonerna blev det rätt likt...

Johnny Eriksson

DATAO APR, Maintenance Data Out, Arithmetic Processor

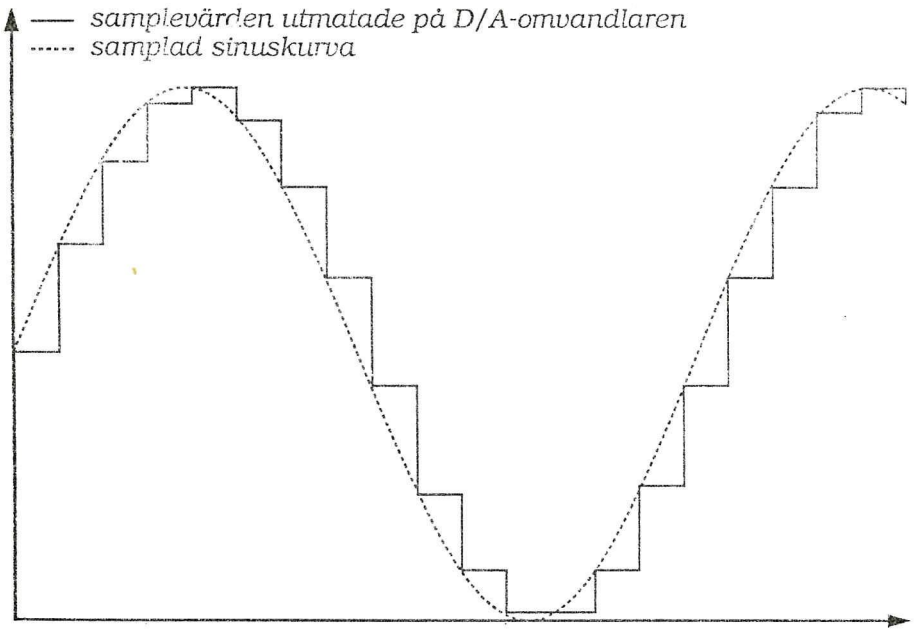


Supply diagnostic information and perform diagnostic functions according to the contents of location *E* as shown.



The margin value specified by bits 30–35 of the output word is translated to a voltage in the range 0–10 volts by a D-A converter, whose output is available at pin 2S02V2. Running margins requires a slowdown capacitor in the converter. But turning off the margin enable switch cuts out the capacitor, making the converter output suitable for external use, such as for operating audio equipment to play Bach or rock or Bacharach.

Notes. This instruction is primarily for maintenance, for which further information is given in Chapter 10 of the *KI10 Maintainance Manual*.



Ljudtillsats till hemdatorn

Följande ritning kan komma väl till pass för den som vill börja leka med sin hemdator. Passande nog är den integrerade krets som används, i likhet med KI10, utgången från marknaden...

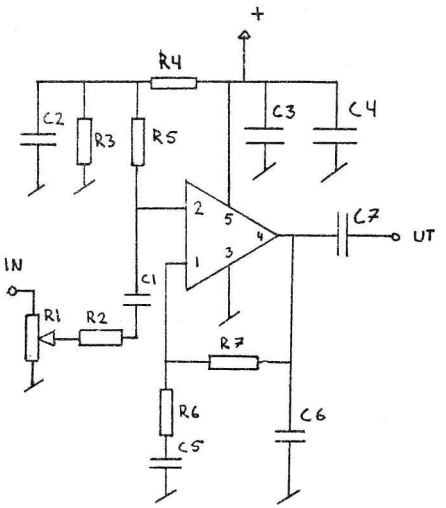
Komponentförteckning:

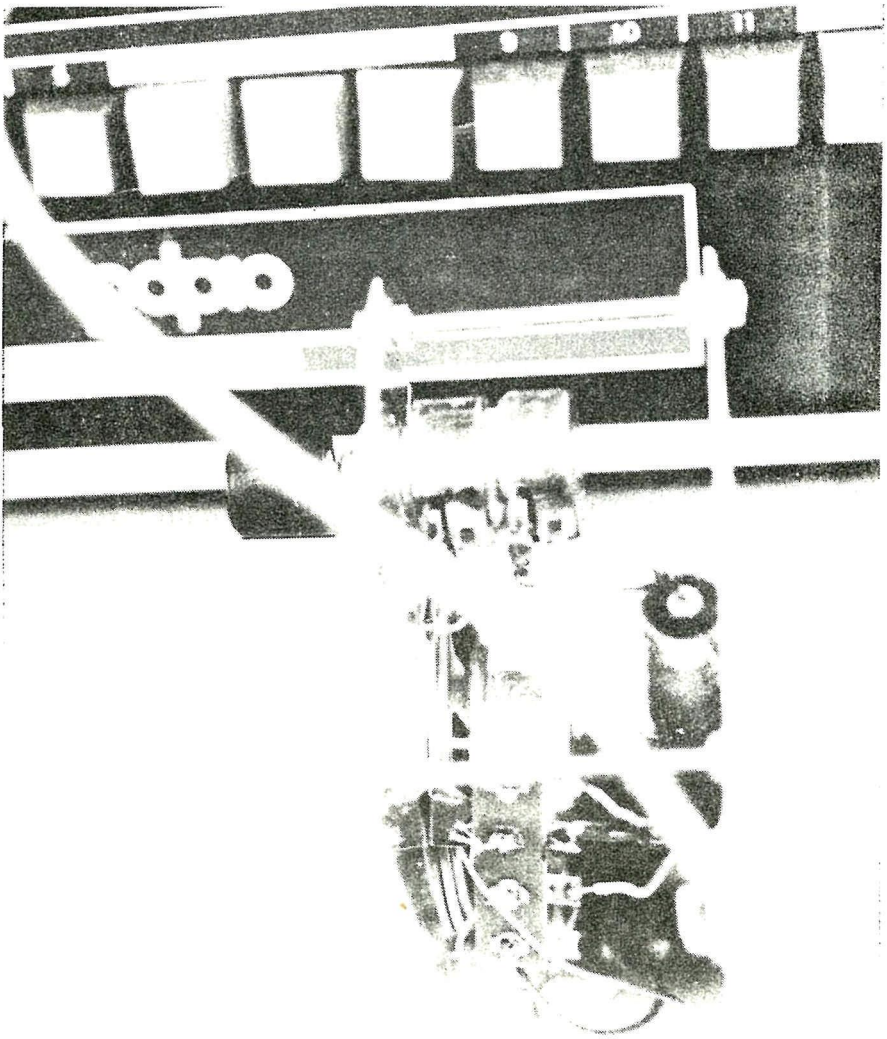
IC1	SN76008
R1	Pot, 100 kΩ log.
R2	10 kΩ
R3	10 kΩ
R4	10 kΩ
R5	17 kΩ
R6	1 kΩ
R7	47 kΩ
C1	0.1 µF
C2	100 µF
C3	100 µF
C4	0.1 µF
C5	5 µF
C6	0.33 µF
C7	1000 µF

Uppbyggnad:

På enklaste sätt, en bit veroboard duger gott. Insignal anslutes till "in", högtalare till "ut", och volt (10-15 stycken) till "+".

Johnny Eriksson





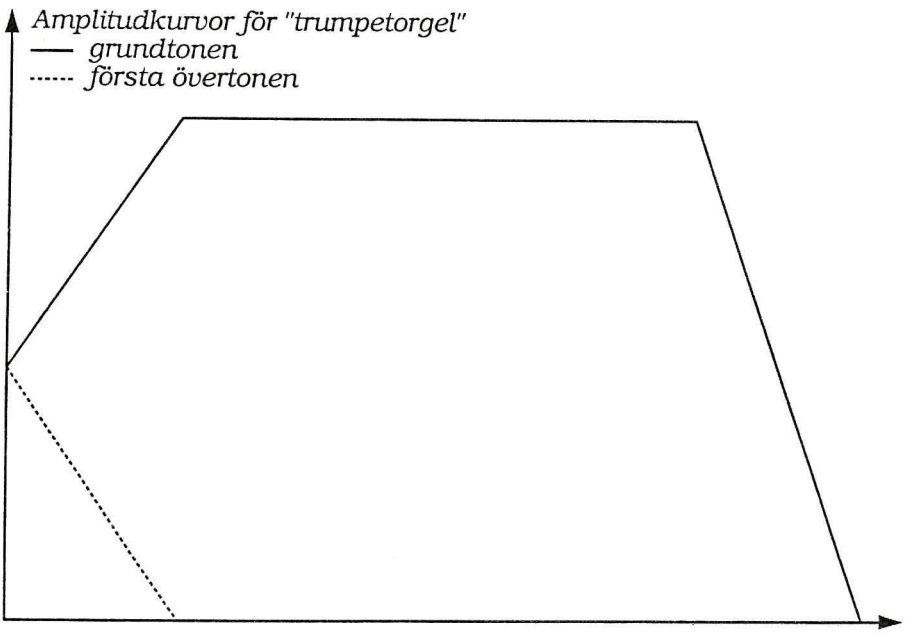
Duo av spikpiano och orgel

Vill man göra något mer avancerat än enstämmig sinusmusik, spela flerstämmigt, ekoeffekter, mänskligt tal, ... så kan man slippa göra tidskrävande beräkningar i realtid, om man lägger alla samplevärdena i en fil och sedan spelar upp den på D/A-omvandlaren. Vi räknade ut att man måste läsa 13 diskblock i sekunden för att klara det, och det gör Kicki utan att anstränga sig. Så jag skrev ett sådant program.

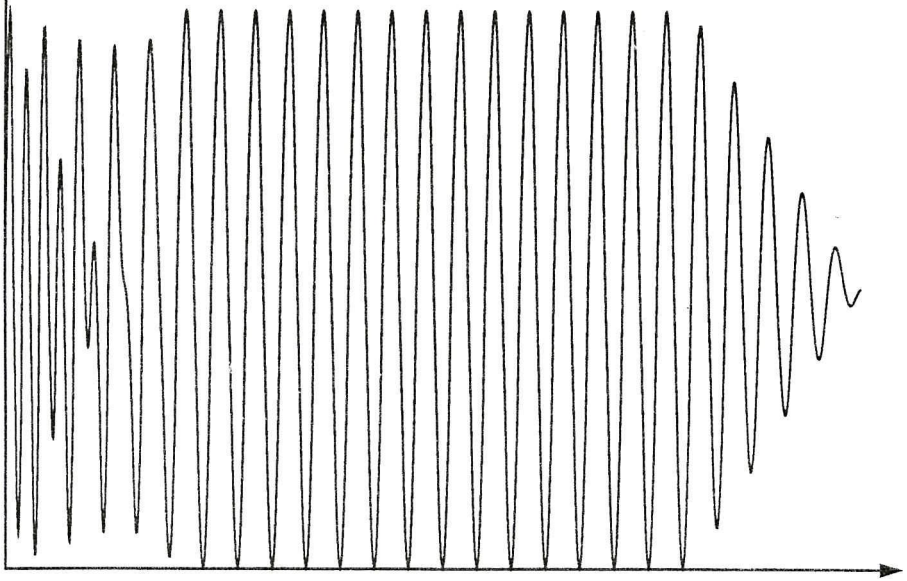
"Ack Värmeland Du Sköna"

utsågs till offer. Vad händer om man blandar grundtonen och första övertonen, så att varje ton börjar med enbart överton, som snabbt avklingar? Det lät som två personer hade spelat låten samtidigt, den ena på spikpiano, den andra på orgel. Usch! Ett mer njutbart ljud, något som liknade en trumpetorgel, fick man om man drog ner övertonshalten, så man började med lika delar grundton och första överton.

Jan Michael Rynning



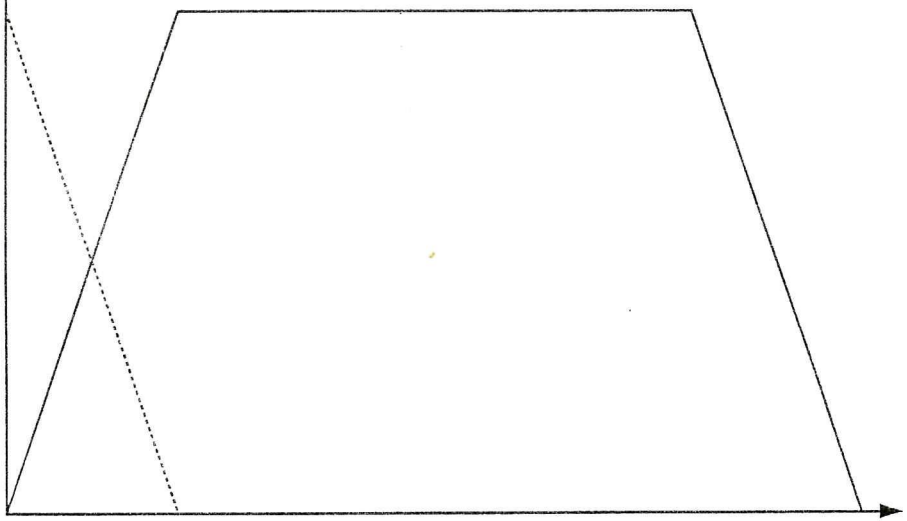
Vågform för "duo av spikpiano och orgel"



Amplitudkurvor för "duo av spikpiano och orgel"

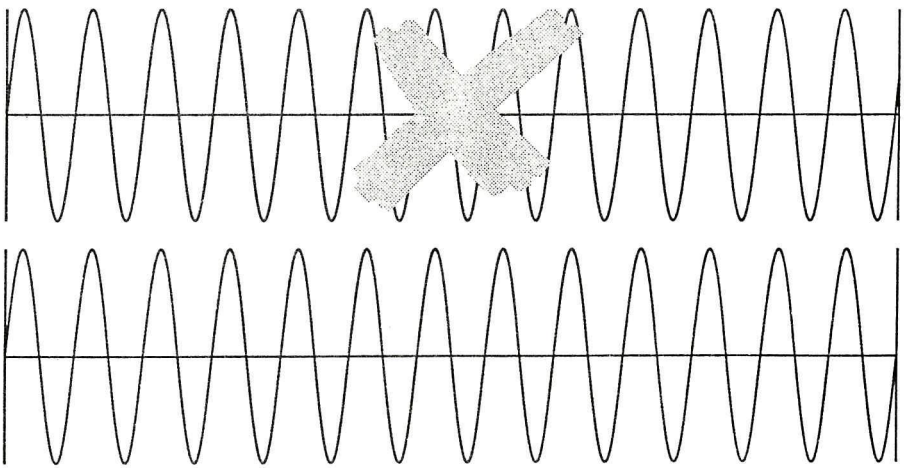
—— *grundtonen*

----- *första övertonen*



```
program wawe(output);
const
  interval = 100;                (* # microseconds between interrupts. *)

function sinus(arg: real): real;
const
  pi = 3.141592654;
begin
  sinus := sin(2*pi*arg);
end;

function twotone(arg: real): real;
const
  pi = 3.141592654;
begin
  twotone := sin(2.0*pi*arg) + sin(6.0*pi*arg)/3;
end;

function Tolerance(t: integer): integer;
begin
  Tolerance := 1 + (t div 500);
end;

procedure generate(function func(x: real): real; note: integer; freq: real);
label 9;
var
  samples, wawes: integer;
  eps: integer;
  i, t: integer;
  actual: real;
begin
  t := round(1E6 / freq);
  eps := Tolerance(t);
  for i := 1 to interval do begin
    if abs((interval div 2) - ((interval div 2)+(i*t)) mod interval) < eps
    then goto 9;
  end;
  9:
  samples := ((i*t)+(interval div 2)) div interval;
  wawes := i;
  actual := ((1E6/interval) * wawes) / samples;
  write('        xwd ', samples:0, ',[');
  write('   ;Note # ', note:0, ', f = ', actual:7:3, ' Hz');
  eps := round(1000.0*actual/freq - 1000.0);
  if eps <> 0 then write(',  Error = ', eps:0, '%.');
  for i := 1 to samples do begin
    if (i mod 18) = 1 then begin
      writeln; write('            byte(6)');
    end else write(',');
    write(round(32+31*func((wawes*i)/samples)):2);
  end;
  writeln(']');
end;
```

```
procedure work(first, last: integer; f: real; function func(x: real): real);
var
  i: integer;
begin
  for i := first to last do begin
    generate(func, i, f);
    f := f * 1.059463094;
  end;
end;

procedure fill(first, last: integer);
var
  i: integer;
begin
  for i := first to last do begin
    writeln('        xwd 6,fill  ;Note # ', i:0);
  end;
end;

begin
  writeln('radix 10');
  writeln;
  writeln('fill:   byte(6) 32,32,32,32,32,32');
  writeln;
  writeln('note::  xwd 6,fill');
  work(1, 50, 55.0, sinus);
  fill(51, 100);
  work(101, 150, 55.0, twotone);
  fill(151, 255);
  writeln;
  writeln('end');
end.
```



```
title   play
search  jobdat, uuosym, macten

pichn==2                        ;Interrupt level to use.

t1==1
t2==2

b==10

define error(msg),<
        jrst[   outstr[asciz "?
?'msg'.
"]
                reset
                exit]
>;error

play:   jfcl                    ;Ignore runoffset.
        reset                   ;Stop the world.

; Set up the tune to play.

        move    t1,[point ^D36,tune##(b)]
        movem   t1,tptr
        setzm   tctr
        move    t1,[point 6,[byte(6) 40,40,40,40,40,40](b)]
        movem   t1,cptr
        movem   t1,dptr
        setzm   cctr
        setzm   dctr

; Set up realtime stuff.

        move    t1,[lk.hls+lk.lls]
        lock    t1,             ;Lock this job in core.
         error(LOCK UUO failed)
        lsh     t1,^D9          ;Convert page # to addr.
        hrrzm   t1,base         ;Set up base register.
        move    t1,[EXCH B,BASE]
        add     t1,base
        movem   t1,fix1         ;Set up first relocated instruction.
        move    t1,[JRST @DKINT]
        add     t1,base
        movem   t1,fix2         ;Set up second relocated instruction.
        movei   t1,[
                xwd     pichn,dkint
                xwd     1,0
                conso   clk,3B32
                exp     0]
        rttrp   t1,             ;Link in device.
         error(RTTRP UUO failed)
        setzm   donflg          ;Clear doorbell.
        cono    clk,1B26        ;Clear interval timer.
        datao   clk,[^D10]      ;Interrupt each 100 microseconds.
        cono    clk,7B32+pichn  ;Start playing.
```

```
wait:   movei   t1,^D1
        sleep   t1,
        skipn   donflg
         jrst   wait
        exit


tctr:   block   1               ;Number of "ticks" left of current waweform.
tptr:   block   1               ;Pointer to sequence of notes...

cctr:   block   1               ;Current counter.
cptr:   block   1               ;Current pointer.

dctr:   block   1               ;Descriptor counter.
dptr:   block   1               ;Descriptor pointer.

base:   block   1               ;Base adress, for relocating at exec level.
acsave: block   1               ;Save T1 here on interrupt.
donflg: block   1               ;Doorbell from exec to user level.

; Here when DK10 interrupts.

dkint:  0                       ;JSR here.
fix1:!  exch    b,0             ;RH patched to exec addr of BASE at init.
        movem   t1,acsave(b)    ;Save T1.
        ildb    t1,cptr(b)      ;Set new output level.
        datao   apr,t1
        cono    clk,1B31+1B32+pichn ;Allow further interrupts.
        sosle   cctr(b)         ;Time to refresh pointer/counter?
         jrst   next(b)
        move    t1,dptr(b)      ;Yes.
        movem   t1,cptr(b)
        move    t1,dctr(b)
        movem   t1,cctr(b)
next:   sosle   tctr(b)         ;Time to change waweform?
         jrst   quit(b)
        ildb    t1,tptr(b)      ;Yes.
        jumpn   t1,next2(b)     ;End of tune?
        cono    clk,1B26        ;Yes, clear interval timer.
        setom   donflg(b)       ;Ring doorbell.
        jrst    quit(b)         ;Dismiss interrupt.

next2:  hlrzm   t1,tctr(b)      ;Set up time to use next note.
        andi    t1,377          ;Only eight bits of note number.
        addi    t1,note##(b)    ;Get exec addr of note descriptor.
        move    t1,(t1)         ;Get data.
        hlrzm   t1,cctr(b)      ;Set up repeat counters.
        hlrzm   t1,dctr(b)
        hrli    t1,(point 6,(b));Make relocated byte pointer.
        movem   t1,cptr(b)      ;Set up byte pointers.
        movem   t1,dptr(b)
quit:   move    t1,acsave(b)    ;Restore everything.
        exch    b,base(b)
fix2:!  jrst    @0              ;RH patched to exec addr of DKINT at init.

end     play
```

```
radix 10

.speed==700                     ;Lenght of /16-note, in 100uS ticks.

%16==   1*.speed
%8==    2*.speed
%.8==   3*.speed
%4==    4*.speed
%.4==   6*.speed
%2==    8*.speed
%.2==   12*.speed
%1==    16*.speed
%.1==   24*.speed

...==28
c==...+0
d==...+2
e==...+4
f==...+5
g==...+7
a==...+9
h==...+11
oct==12

define note(pitch,len<%4>),<
        <len-120>,,pitch
        0120,,0
>;note

tune::  note g-oct,%2
        note c,%.2
        note d,%4
        note e,%2
        note f,%2
        note g,%1
        note h,%2
        note d+oct,%2
        note d+oct,%2
        note c+oct,%2
        note c+oct,%2
        note a+1,%2
        note g,%.1
        note 0,%2
        note g,%2
        note f,%1
        note a-1,%2
        note g,%2
        note e-1,%1
        note g,%2
        note f,%2
        note d,%.1
        note e-1,%2
        note c,%.1
        0

end
```

STACKPOINTER

STACKPOINTER är organ för Datorföreningen STACKEN.

STACKPOINTER utkommer när material i tillräcklig mängd finns, förhoppningsvis 4–5 gånger per år. Återgivande av delar av innehållet är tillåtet när källan anges.

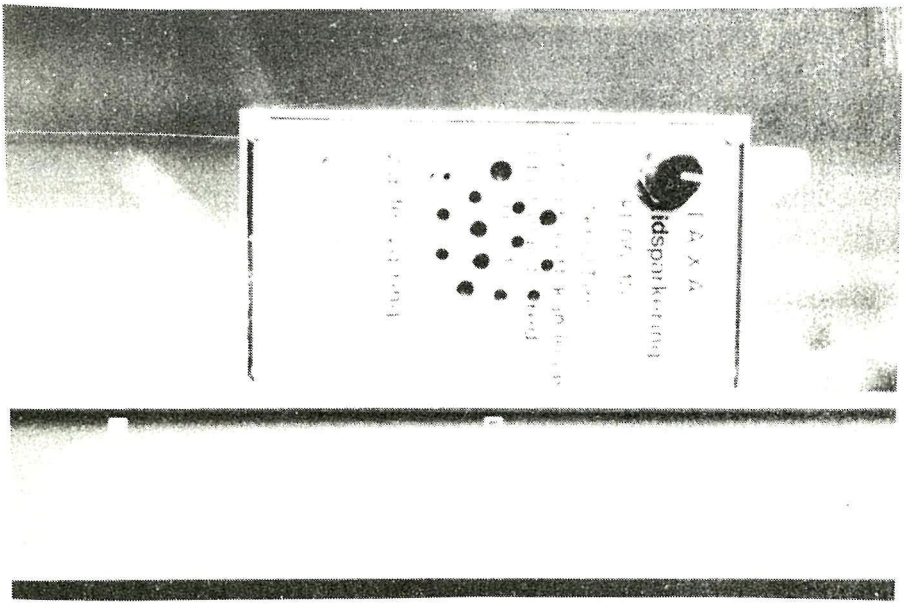
Redaktör: Hans Nordström.

I redaktionen: Jan Michael Rynning, Johnny Eriksson, Stellan Lagerström och Ulf Samuelsson.

Fotografer: Henning Croona och Jan Michael Rynning.

Ansvarig utgivare: Mats O Jansson.

Färdigställd: 1986-11-14.



Datorcentralen i Århus har en speciell taxa för parkerade jobb (=detachade). Betalningen görs direkt till CPU:n. Ja, se danskarna.

Redaktörn

Allt oftare nuförtiden upptäcker jag barn på arbetet. Bra, blir min tanke. Föräldrarna tillämpar det här med att barnen skall få komma och se de vuxnas värld. Men ack, så fel jag har. Sedan en tid tillbaks finns Macintosh på arbetsplatsen. Där placeras barnen. En halv dag kan man lyckas bli av med dom på detta sätt. Barnvakt. Häpp!

Hört att man har D/A i hårdvara. Det blir ljud av det. Vilka möjligheter! På den gamla goda tiden (50-talet) så fanns inte debugger och annat tjafs. Men det fanns högtalare ansluten i maskinen. Alltså lyssnade man sig till när programmet gick in i evigheten (loopade). Eller ännu äldre, relämaskiner, då diagnosticerades utan högtalare. Hans Riesel har berättat. Vilka kommentarer till program, en D/A-mojäng kan ge upphov till:

"Tycker inte du att Foo-kompilatorn fått en lite råare ton i senaste versionen?"

"Har du hört på Maken!" från Teco-folk.

"ThinkTank låter More nu."

Nu blir det en evig visa att lyssna på datorn.

Ja, det är det hele.

/hn

Häng Macintosh

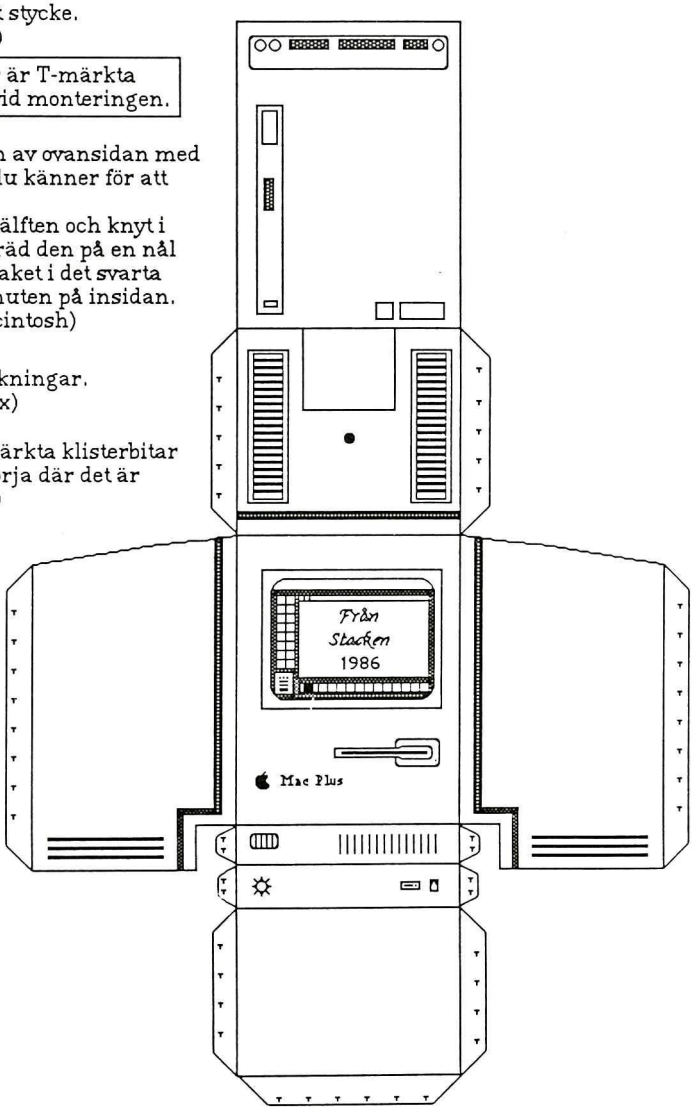
1. Kapa Macen i ett stycke.
(Ref QuickDraw)

   Alla klisterbitar är T-märkta och försvinner vid monteringen.

2. Förstärk insidan av ovansidan med en tape-bit ifall du känner för att hänga Macen.
Vik en tråd på hälften och knyt i andra ändan. Träd den på en nål och för igenom taket i det svarta hålet. Lämna knuten på insidan.
(Ref. Inside Macintosh)

3. Vinkla alla utvikningar.
(Använd ToolBox)

4. Klistra alla T-märkta klisterbitar med insidan. Börja där det är toppen. (Refse 2)



A.A. -86 hn


ORIGINAL
HACKS
HACKS