Uptime

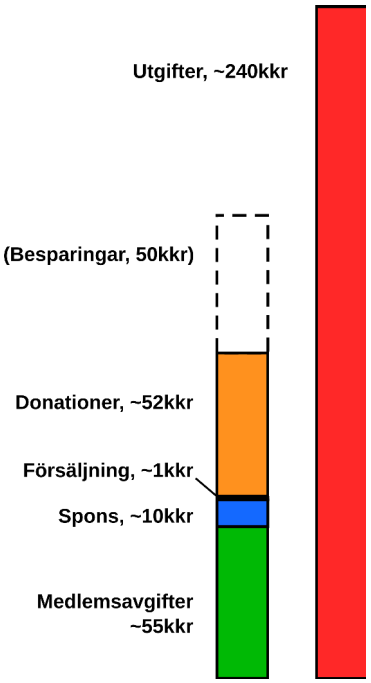
Datorföreningen Updates nyhetsblad

Redaktör: Bjarni Juliusson

Nummer 7 2023

Updates ekonomi

Sökandet av spons går för trögt. Våra förväntade utgifter 2024 är runt 240.000kr, mestadels hyra. Våra nuvarande återkommande årliga intäkter är 65.000kr, varav 10.000kr är spons från två sponsorer. Därtill har vi fått in donationer på runt 52.000kr. Som det ser ut just nu behöver vi alltså ytterligare omkring 72.000kr för att klara nästa år, men jag understryker att detta inte räcker för långsiktig ekonomi! Det är nödvändigt att vi innan nästa års slut inte bara har återkommande inkomster som fullt motsvarar utgifterna, utan dessutom har pengar över att spara inför den flytt som kommer. Vi bor på ett kraftigt rabatterat rivningskontrakt som löper ut vid slutet av 2024 om det inte kan förlängas. Donationer och besparingar bör ses som en tidsfrist för arbetet med insamlandet av återkommande spons. Det ska noteras att medlemsantalet stiger — med åtta nya medlemmar hittills detta år — vilket bidrar något till intäkterna, men det kommer trots allt att vara nödvändigt att skriva många fler sponsoravtal. Hjälp till!



Vill du komma i kontakt med Update kan du skriva till styrelsen@dfupdate.se eller till vår allmänna diskussionslista update@lists.dfupdate.se. Du är också välkommen att besöka vår IRC-kanal #update på EFnet eller lokalen på Svartbäcksgatan 65 i Uppsala. Uptime nås på uptime@dfupdate.se.

Närmast planerade händelser

Lördagssweatshop varje lördag kl 14-
Vi samlas i lokalen och arbetar med spons, stjärnuppgifter, månadens dator, etc.

Höstmöte måndag 13/11 kl 18ak
Plats: Svartbäcksgatan 65. Efter mötet bjuder vi på våfflor!

Updatering lördag 18/11 kl 19:00
Data management and digital preservation of (research) data med Herbert Lange från Göteborgs universitet. Plats: Svartbäcksgatan 65 och via BBB. Se wikin för mer information.

Plenumsmöte onsdag 22/11 kl 18:30
Välkommen att delta i vår löpande planering och utvärdering. Plats: Svartbäcksgatan 65.

Onsdagsmöte 6/12 kl 18
Restaurang bestäms på IRC och Updatelistan.

Utställningen på Svartbäcksgatan 65 är också öppen varje lördag från klockan 14 till kvällen.

Nyheter

■ Den största nyheten just nu är att Helene Elisabeth Bunkelmann Marckwardts fond även iår beslutat att stöda Update med en donation. Deras bidrag på 50.000kr är en mycket stor hjälp, och ger oss nödvändig tid att hitta mer långsiktig spons. Stort tack till H.E.B.M.F!
■ Plenumsmöte hölls den 25/10. Protokollet dyker förhoppningsvis upp här.
■ Vi öppnade vår utställning på Kulturnatten 9/9, med mycket gott resultat. Totalt kom 166 besökare. Datorerna som var igång var mycket populära.
■ Vi sände två representanter till Engagemangsmässan 12/9, en två timmars minimässa på Blåsenhus för människor som söker föreningar att engagera sig i. Det var delvis givande, och väldigt liten ansträngning.
■ Den 1/9 gjorde Folkrörelsearkivet besök hos Update på sin kick-off-dag.
■ Vi hade en introduktionsdag för nya medlemmar 16/9. Fyra nya medlemmar deltog.
■ Vi har också fått fyra nya medlemmar: Välkomna, obscurity, hansit, senap och SM5DEH!

Apptajm: expect

Expect automatiserar körandet av sådana *interaktiva* verktyg, som inte kan styras endast från kommandoraden utan också kräver att användaren därefter reagerar på prompter för att t.ex. skriva in lösenord och liknande. Det fungerar som en programmerbar stand-in för dig vid terminalen, som läser vad det interaktiva verktyget skriver ut och reagerar med att "skriva" rätt svar på "tangentbordet". Rätt najs.

I praktiken fungerar det så att man skriver ett skript i vilket man använder en handfull kommandon för att 1) starta det interaktiva programmet, 2) vänta på output som matchar något mönster, och 3) skicka input. Sedan kör man skriptet med:

```
$ expect ditt_skript.exp
```

Expect är dessutom skrivet som en utökning till programmeringsspråket Tcl, och man kan också använda Tcl-kod i sina skript. Tcl är ett rätt märkligt språk, men de saker man oftast vill göra i Expect är inte så krångliga, som vi ska få se. Vi kanske kan prata mer om Tcl nån annan gång.

```
spawn <prog> <arg1> <...>        # Starta ett program, som blir "current process".

expect <text>                    # Vänta tills programmet skriver ut denna text

send <text>                      # Skicka programmet denna text som input

interact                         # Lämna över till användaren
```

Ett enkelt Expect-skript för att starta en telnet-session och logga in skulle kunna se ut såhär:

```
spawn telnet min.hemliga.server.nu
expect "Användarnamn: "
send "bob\n"
expect "Lösenord: "
send "golvsylt\n"
interact
```

Den sista raden lämnar över interaktionen med det startade programmet till dig, så att du kan ta över och fortsätta telnet-sessionen själv efter att Expect har skött början. Därefter kommunicerar du med Telnet som om Expect inte vore där.

Det går också att ha flera mönster med respektive kommandon som ett enda `expect`-kommando. Detta `expect` väntar då tills något av mönstren matchar input och kör det mönstrets handling:

```
expect ping {send pong} ding {send dong}
```

Det går också bra att lämna den sista handligen tom, vilket är vad vi gjorde i det första exemplet, på förra sidan, som bara hade ett mönster och ingen handling. Mönstret matchas, men gör ingenting, och skriptet går vidare till nästa rad.

Har man många mönster-kommando-par kan man radbryta mellan dem med backslash sist på raderna, eller genom att stoppa alltihop inom ett par måsvingar:

```
expect ping {send pong} \
       ding {send dong}
```

eller

```
expect {
  ping {send pong}
  ding {send dong}
}
```

Av default förväntar sig `expect` mönster i glob-format, dvs med '`*`' för "vilken sträng som helst", '`?`' för "ett tecken vilket som helst", och '`[`*`abc..`*`]`' för "något av dessa tecken". Det går också att använda reguljära uttryck med parametern `-re`:

```
expect -re hejsan|svejsan  {send "Hej hej"}
```

Man kan också använda variabler i sina skript. Dessa sätter man med `set`, och evaluerar med `$`:

```
set goddag hej
puts $goddag                          # Skriver ut "hej"
```

`puts` är ett Tcl-kommando som skriver ut ett meddelande. Detta skrivs alltså ut i terminalen; det skickas inte till current process.

Det går också att komma åt miljövariabler och kommandoradsparametrar:

```
expect "Vad är din TERM? " {send "$env(TERM)\n"}

puts "$argc parametrar: $argv"

puts "parameter 1 är: [lindex $argv 1]"
```

Här ser vi också användandet av `[hakparenteser]`, en Tcl-syntax som substituerar värdet som returneras av ett kommando. I det här fallet använder vi kommandot `lindex`, som indexerar en lista.

Vi har använt både måsvingar och citationstecken i exemplen, och de gör likartade saker; båda förhindrar ordsplittning: `{Detta är en sträng}` och `"Detta är också en sträng"`. Skillnaden är att måsvingar också förhindrar substitutioner inuti strängen. Så om man vill använda värdet på en variabel inuti en citerad sträng så använder man citationstecken, och vill man förhindra att något som ser ut som en variabel substitueras så använder man måsvingar:

```
puts {$goddag}                        # Skriver ut "$goddag"
puts "$goddag"                        # Skriver ut "hej"
```

En sträng som bara är ett ord, som t.ex. "hej", behöver inte citeras alls eftersom den inte splittas; och omvänt gör det inget om man citerar t.ex. `puts` som `"puts"` eller `{puts}`, det ändrar inte betydelsen.

Förutom att matcha text kan expect-kommandot också detektera End Of File eller en timeout, och köra kommandon som svar. EOF uppstår om programmet Expect kommunicerar med avslutar sig, och timeout inträffar om ingenting har matchat inom en viss tidsrymd. Default är 10 sekunder, men detta kan man ändra med variabeln `timeout` eller parametern `-timeout`:

```
expect foo {send bar}                  # 10s timeout
expect -timeout 3 foo {send bar}       # 3s timeout bara den här gången
set timeout -1                         # Oändlig timeout fr.o.m. nu
expect timeout {send Tempo!\n} foo     # Vänta på foo, hantera timeout
expect eof {puts "Fick EOF"} foo       # Vänta på foo, hantera EOF
```

Man kan också använda mönstret `"default"` för att matcha antingen EOF eller timeout. Vill man matcha själva strängarna "default", "timeout" och "eof" kan man använda `-ex` före mönstret, vilket betyder "exact match". Det kan också användas för att matcha t.ex. en sträng som innehåller `'*'`.

```
expect -ex ******      # Matchar 6 asterisker
expect ******          # Matchar vad som helst
```

För att plocka ut texten som matchade, t.ex. för att använda den i svaret, kan man använda `$expect_out(0,string)`. Det ser lite kryptiskt ut, men `expect_out` är en associativ array som innehåller output från `expect`-kommandot, och "0,string" är en nyckel. Har man matchat ett reguljärt uttryck med grupper i kan man också få ut grupperna med nycklar från "1,string" till "9,string". Notera att nycklarna är strängar, och att det inte är något mellanslag mellan siffran och "string".

Man kan förresten göra `expect` och `send` mot sig själv istället för det anslutna programmet, vilket kan vara användbart för att testa kommandon. Detta gör man med kommandona `expect_user` och `send_user`. Det förra väntar alltså på input från dig, som du skriver in på tangentbordet, och det senare fungerar i princip som `puts`. Testa t.ex. att starta Expect och skriva in följande rad:

```
expect_user -re "hej(san)?" \
                {set h $expect_out(0,string); send_user "$h $h!\n"}
```

Om du sen skriver in "hej" eller "hejsan" svarar skriptet med antingen "hej hej!" eller "hejsan hejsan!".

Skillnaden mellan `puts` och `send_user`, förutom att `puts` automatiskt lägger till ett nyradstecken på slutet, är att text som skrivs ut med `send_user` kommer med i loggen om man loggar sessionen. Det kan man göra med kommandot `log_file`:

```
log_file telnet-log.txt        # Börja logga
log_file                       # Sluta logga
```

Tcl har såklart också helt vanliga kontrollstrukturer, som man kan använda i Expect:

```
while <villkor> <kropp>
for <start> <villkor> <inkrement> <kropp>
if <villkor> <kropp> else <kropp>
```

Exempel:

```
for {set i 0} {$i < 10} {incr i} {
  send_user "i = $i"
  if {$i % 2} {send_user " (udda)"}
  send_user \n
}
```

Precis som många andra språk har loopar i Tcl också `break` och `continue`. Men dessutom har expect-kommandot ett eget continue-kommando som heter `exp_continue`, som går tillbaka och fortsätter matcha efter en match:

```
expect_user {
  killevippen {puts "Du sa det magiska ordet!"}
  \n {puts "Nej det är fel. Försök igen."; exp_continue}
}
```

Det finns också en mängd andra finesser i Expect. Jag tänker inte gå igenom allihop i den här korta introduktionen, men för att ge en uppfattning om vad mer som finns har ni här en liten lista:

```
interact <mönster> <kommando>          # Överlåter interaktionen till dig men
                                       # tjuvlyssnar och reagerar på vad som sägs

$spawn_id                              # Handtag för senast spawnade processen
interact -u <spawn-handtag>            # Låter två program prata med varann

spawn -open [open <filnamn> <mode>] # Öppnar t.ex. en devicefil för en serieport
spawn -pty                             # Skapar en PTY men ingen process
$spawn_out(slave,name)                 # Sökväg till PTY-slaven

send -s                                # Send slowly, valfritt antal tecken per sekund
set send_slow {<tecken> <sekunder>} # Konfiguration för send -s
send -h                                # h för human, imiterar mänsklig variation

fork                                   # Forkar Expect
disconnect                             # Kopplar loss processen från terminalen
wait                                   # Väntar på att en process avslutas
system <kommando>                      # Kör ett shellkommando

sleep <sekunder>                       # Pausar en stund
exit                                   # Avslutar Expect
```

Därtill kommer all den funktionalitet som man kan förvänta sig av ett vanligt skriptspråk, eftersom hela Tcl står till ens förfogande. Det finns alltså funktionalitet för aritmetik, stränghantering, listor, funktioner, filer, GUI-programmering, osv.

Juni månads dator: Compis

Det svenska skoldatorprojektet COMPIS (för *COMPuter In School*) startade i början av 1980-talet med syftet att ersätta den likaledes svenska ABC 80 som användes i skolorna vid den tiden. Datorn kom i produktion 1985 och såldes enbart till skolor. Vid en tid då det fanns flera populära persondatormodeller tillgängliga för allmänheten, t.ex. IBM PC, Apple II, Apple Macintosh och Atari ST, kan beslutet att tillverka helt egna datorer bara för de svenska skolorna verka illa genomtänkt. Det blev inte heller nån stor framgång. Inte mycket mjukvara skrevs, produktionen upphörde 1988, och skolorna bytte snart ut sina datorer mot modeller som folk var bekanta med från hem och kontor. Eftersom Compis aldrig såldes till allmänheten har den inte blivit föremål för någon större nostalgi, och få hobbyister har bevarat dem.

Egentligen är Compis en rätt OK persondator för tiden. Den har 128Kbyte eller 256Kbyte RAM (expanderbart till 768Kbyte), en 8MHz 16-bitars CPU av modell 80186 (snarlik och kompatibel med föregångaren 8086 som satt i IBM PC), samt överraskande högupplöst grafik: 1280x800 pixlar svartvitt, eller 640x400 i färg. Datorn är inrymd i en låda av typen pizzakartong, som staplas tillsammans med en lika stor låda innehållande två 5,25" diskettenheter. Skärmen är separat och kan ställas ovanpå datorn, och tangentbordet är också separat.

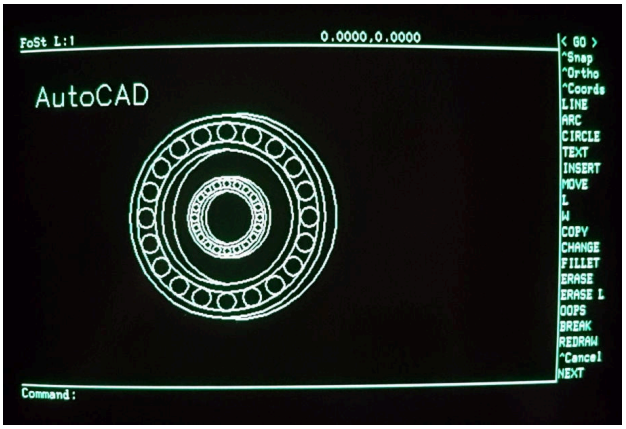
Compis kör normalt operativsystemet CP/M-86, men kan också köra MSDOS. Programmeringsspråket Comal, en sorts aningen mer vuxen motsvarighet till BASIC, framfördes som det rekommenderade programmeringsspråket, och det fanns också implementationer av Pascal, COBOL och Fortran.

Update har tre Compis-datorer i sin ägo. Två av dessa fungerar. En är testad men trasig. Vi har också fyra tillhörande diskettstationschassin (två enheter per chassi), som alla är testade och fungerar (med en enhet som är lite otillförlitlig). Vi har också två skärmar, varav en fungerar och en är trasig.

Vidare har vi två tangentbord, båda fungerande men inte i toppskick; sex exemplar av strömsladdarna som går mellan skärmen och datorn/diskettenheten; ett grafikkort som förefaller vara "32-colour direct access bit-mapped graphics board" samt manual till detta.

Även några donglar: Data Term Programvarunyckel; Compis Pascal Programvarunyckel (2st); HarmoniTvå Programvarunyckel; Millikans försök Programvarunyckel; HarmoniEtt (alfa).

Därtill ett stort antal disketter i boxar, bland annat: COMAL, KalkTvå-Plus, AutoCAD 1.40, MSDOS 3.20, MSDOS 3.10, Harmoni2, TextEtt, LOGISIM, TextTvå Alfa 10 Plus, en 6800-korsassembler, en VT52-emulator, diverse typsnitt, olika demo- och undervisningsprogram, samt dokumentfiler med skoluppgifter etc. Vi har inte gått igenom alla disketterna.



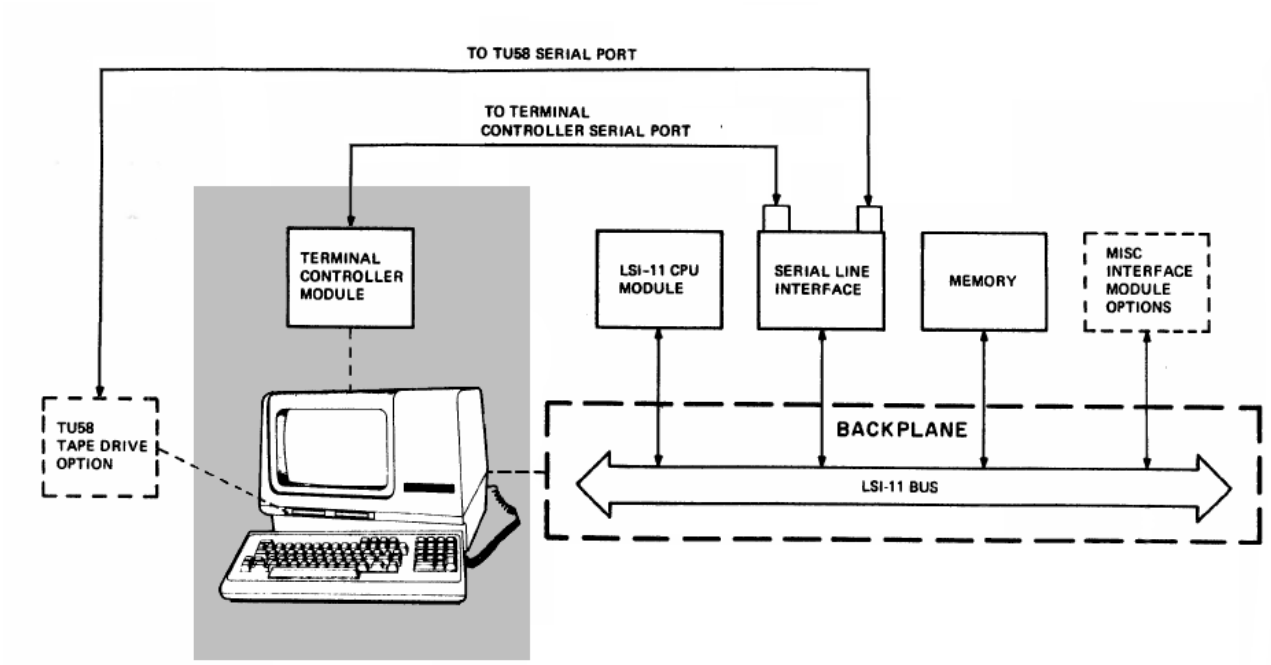
Oktober månads dator: VT103

Efter ett sommaruppehåll har vi fortsatt med månadens dator i oktober. Vi valde en modell som Update har flera exemplar av som vi visste var i behov av reparation. Det mesta av arbetet har gått till reparationer, och vi har nu en i huvudsak fungerande VT103 och två som är helare än de var innan, men fortfarande i behov av mer reparationer. Vi har inte kommit så långt att vi har bootat något operativsystem på någon av datorerna. Nu är det ny månad och ny dator, och det fortsatta arbetet med VT103 hamnar i bakgrunden.

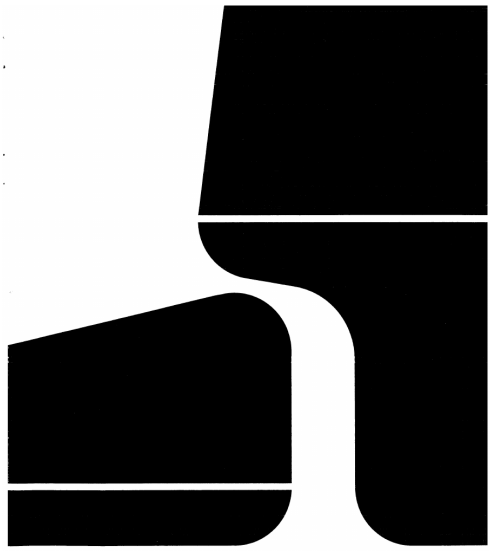
VT103 är en variant av DECs populära terminalmodell VT100 från 1978. VT103 verkar, att döma av datumen i manualen, ha kommit ut 1979. Till det yttre ser VT103 precis ut som en VT100, men på insidan har man byggt in en kortram för en liten PDP-11-dator. Förutsatt att den har bestyckats med en CPU (LSI-11), minne, serieportar osv, är VT103 således en komplett PDP-11 i desktopformat.

Jag har försökt illustrera det hela i bilden nedan, som är en redigerad version av en bild från användarmanualen. Den gråa rutan är VT100, som alltså bara är en terminal, avsedd att kopplas till en dator via serieport. I VT100 finns ett enda logik-kretskort: "Terminal controller module", som implementerar all terminalfunktionalitet. I VT103 finns dock inbyggt i samma chassi även ett Q-bus-bakplan (kallat "LSI-11 Bus" i bilden), med en komplett dator i. Denna är ansluten till VT100-terminalen via serieporten precis som vanligt, men internt i chassit istället för via kontakten på baksidan. I chassit sitter också (som tillval) en TU58 bandenhet, som även den ansluts till datorn via en serieport. Q-bus är en uppföljare till PDP-11-bussen Unibus, och introducerades med processorn LSI-11, den dittills minsta medlemmen i PDP-11-familjen.

En av Updates VT103:or

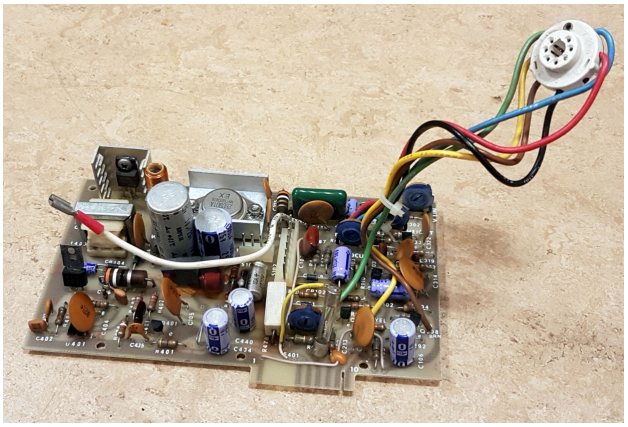


VT100



Alltihop baserar sig alltså på terminalen VT100, som DEC introducerade 1978. Det är en videoterminal med en 12-tums skärm som visar 80x24 tecken i svartvitt. Det finns också ett läge som visar 132 kolumner, men då endast 14 rader om man inte har installerat en "advanced video"-option. Terminalen kommunicerar med en dator via en serieport, i upp till 19200 baud. VT100 var enormt kommersiellt framgångsrik, och är idag känd för att ha populariserat ANSI-escapesekvenserna som fortfarande används för att flytta markören, skrolla, visa inverterad text osv, även i moderna terminalemulatorer som XTerm, Windows Terminal, macOS Terminal.app, osv.

Ett mycket vanligt förekommande fel som uppstår på dessa terminaler är dock att några komponenter på kretskortet som genererar spänningar och styrsignaler till bildröret brinner upp. Så var också fallet i två av våra tre VT103-maskiner. Den tredje saknade kortet helt och hållet. Lyckligtvis hittade vi ett par reservkort i förrådet, så vi tog ett av dem och så började vi felsöka.



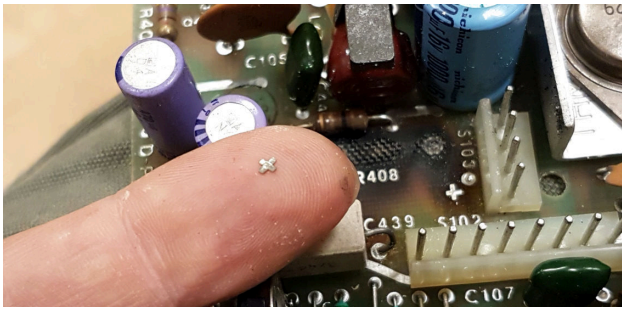
Såhär ser korten ut.

Ganska snart var det uppenbart att det fanns ett par fullkomligt förintade komponenter i mitten av varje kort: en diod och en kondensator.

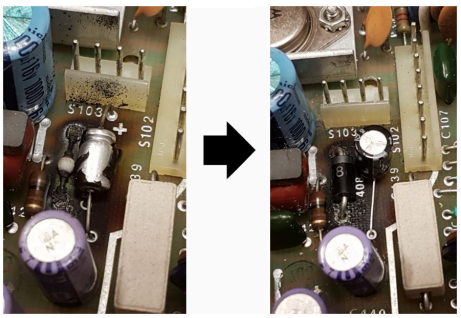
Runtikring dessa var själva kretskortet i samtliga fall förkolnat, och ledningsbanorna var i en del fall helt försvunna, eller förvandlades omedelbart till smulor vid beröring. På ett av korten lossnade "+"-tecknet som angav korrekt orientering för den tillintetgjorda dioden. Hålen i kortet som benen på dioden ska sitta i hade ökat några snäpp i storlek till följd av en karbonifikativ kantförkexning.

F.d. diod

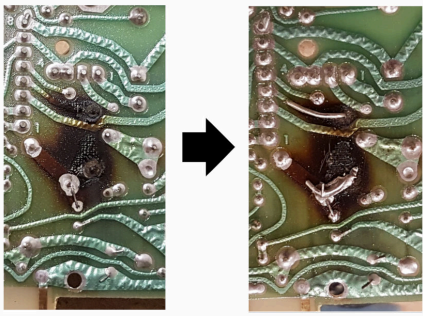
Både dioder och kondensatorer var dock lättersatta komponenter som genast skakades fram nya, och efter visst skrubbande av kretskorten med isopropanol och tops monterades dessa istället för de förstörda, och de saknade ledningsbanorna ersattes med trådbitar. Därefter peps alla transistorer med multimetern, och åter hade alla korten likadana fel: horisontalutgångstransistorn, som driver flybacktransformatorn, var trasig. Den typen av komponenter är inte riktigt lika frekvent lösdrivande i reservdelslådorna, men vi hade tre.



The Bad Plus

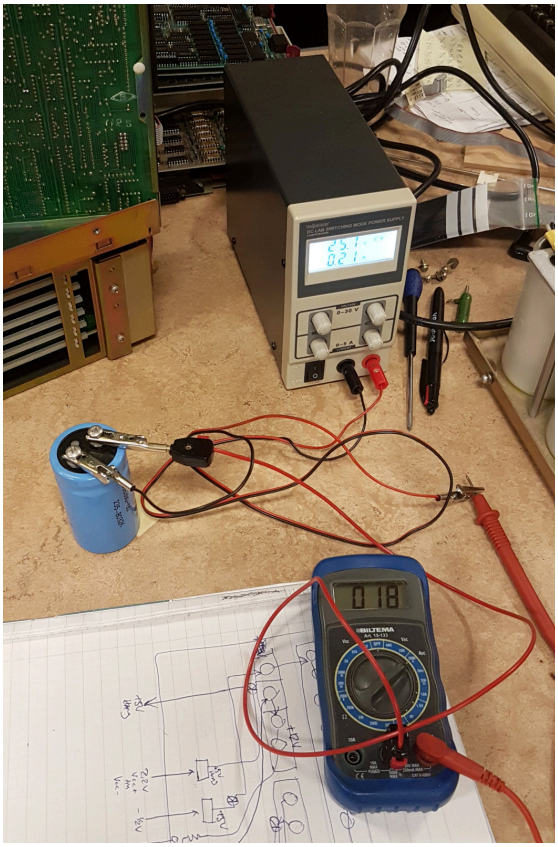


Dock i en annan, större, kapsel, så det fick böjas lite ben. Men det gick bra! Nu har vi tre fungerande kort.

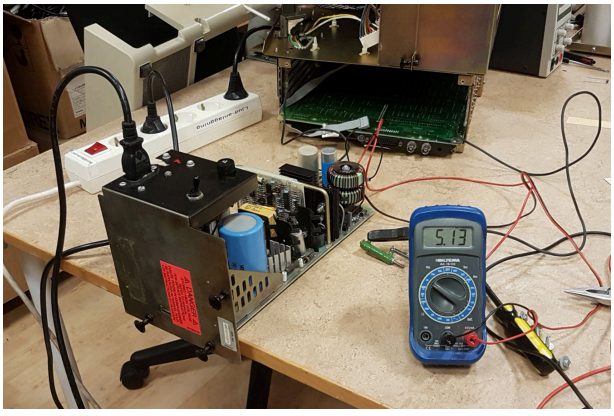


Sen var det dags att testa nätaggregaten. Två av VT103:orna hade nätaggregat med trasiga säkringar, men i den tredje var säkringen hel. Vi började med den. Det första vi gjorde var att skruva ur de två stora glättningskondensatorerna som sitter på primärsidan i nätaggregatet, för att testa deras läckström. Datorerna har inte varit påslagna på, ptja, årtionden troligtvis, och det vore jobbigt om de här största kondensatorerna hade blivit läckiga och sedan sprängdes så fort vi slog på strömmen. Värt att kolla dem. Och de var OK!

Så då kopplade vi helt enkelt in nätagget i väggen (men inte i datorn) och slog på strömmen. Inga fyrverkerier utlöstes. Spänningarna var dock inte alls rätt. Switchade nätaggregat är bortom undertecknads kompetens, så vi kontaktade Mattis på Dalby Datormuseum, som hjälpt oss med sådana förut. Han var, som alltid, mycket hjälpsam, och gav exakta instruktioner för vilka punkter på kretskortet som skulle mätas med oscilloskopet, var man skulle mata in spänningar, och hur signalerna skulle se ut. Allting såg dock ut precis som det skulle, och vi kliade oss i huvudet. Efter lite mer epostande och mätande konstaterade vi hur som helst att spänningarna nu var rätt, och att felet hade löst sig självt. Nöjda/missnöjda med detta stoppade vi tillbaka nätagget i datorn, och slog på strömmen. Terminalen fungerade finfint. Tangentbordskontakten krävde lite spray och bildstorleken krävde lite skruvande på en potentiometer, men sedan var allt tårta.



Kondensator testas

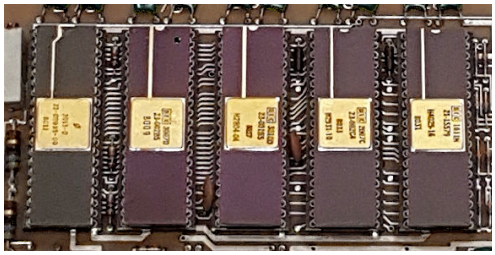


Hehu.

Hehé!

LSI-11

1975, några år innan VT103 kom ut, introducerade DEC sin första PDP-11 implementerad i LSI-teknologi, dvs Large-Scale Integration — ett litet antal mikrochip större än 10.000 transistorer. Tidigare PDP-11:or bestod av flera kretskort med ett stort antal små logikchip: grindar, flip-floppar, etc. LSI-11 däremot består i huvudsak av fyra chip: kontrollenhet, datapath, och två mikrokods-ROM. Det finns också en sockel för ett tredje mikrokods-ROM, i vilken våra exemplar har KEV11-A, en utökning för multiplikation och division samt flyttalsaritmetik. I själva verket består LSI-11 av Western Digitals MCP-1600, ett mikroprogrammerbart CPU-chipset som också användes av andra datortillverkare vid tiden för att implementera deras respektive arkitekturer, med hjälp av egen mikrokod.

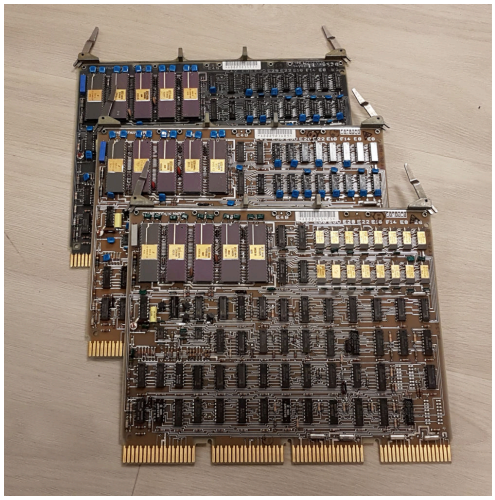


LSI-11-CPU; fem chip inkl. KEV11-A

De fem mikrochippen sitter på ett kretskort som också innehåller klockgenerator, bussgränssnitt, och 8Kbyte RAM. Kortet heter KD11-F, är av "quad"-storlek på DEC-språk, och avsett att stoppas i ett Q-bus-bakplan. Det fanns också en variant utan minne, KD11-H, och en senare minneslös version på ett hälften så stort kort, KD11-HA, även känd som LSI-11/2.

Istället för en frontpanel med spakar och lampor som på större PDP-11:or har LSI-11 ett enkelt monitorprogram implementerat i mikrokoden. Detta heter ODT, och tillåter läsning och skrivning av minne och register i oktal notation via konsolterminalen, som ansluts via ett separat serieportskort.

Av Updates tre LSI-11:or är det två som verkar fungera, men vi har inte lyckats boota något system eller någon diagnostik under oktober.



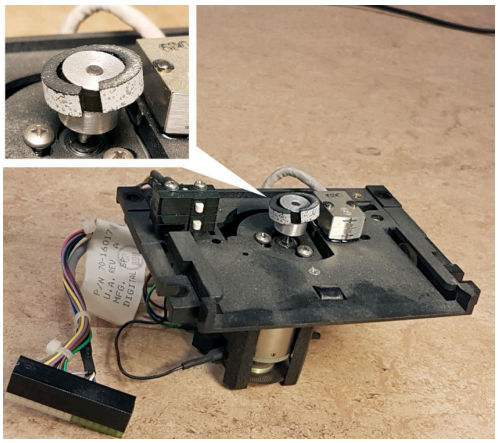
Updates tre LSI-11:or

TU58

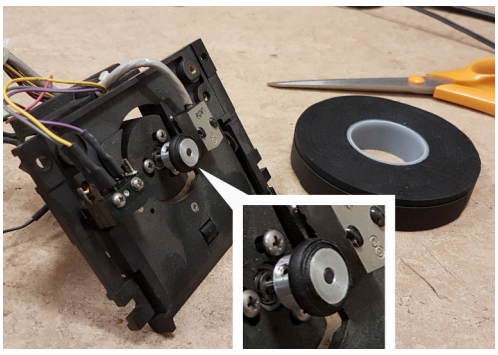
Bandenheten som sitter i framkanten under skärmen på VT103 heter TU58. Den har två separata drivar för band av typen DECtape II, ett kassettband ungefär lika stort som en musikkassett, med en kapacitet på 256Kbyte. Data är organiserade i fasta 512-bytes block, och åtkomsten är random-access, i likhet med det ursprungliga DECtape-formatet.

TU58-enheten ansluts till datorn via en vanlig serieport. I en baskonfigurerad VT103 bör man alltså ha två serieportar: en konsol, som pratar med VT100-terminalen, och en andra port för att prata med bandaren. TU58-kontrollern pratar ett paketorienterat protokoll där datorn kan be att få läsa eller skriva block nummer si och så; bandaren spolar bandet automatiskt till rätt ställe.

DECtape II användes främst för mjukvarudistribution, eftersom åtkomsten är långsam i mer allmänt bruk där man gör mycket sökningar. Update har en låda med ett antal DECtape II-band, men samtliga innehåller mjukvara för VAX, inget för PDP-11.



Gamla bandenheter som inte har varit i bruk på länge kan man räkna med har olika mekaniska problem som är mer eller mindre jobbiga att fixa. TU58 har dock en uppfriskande enkel mekanism, bestående av en enda rörlig del: en motor med ett gummihjul direkt på axeln, som matar fram bandet. Kassetten trycks bara rakt in i ett spår. Det är ingen lucka, ingen arm som flyttar läshuvudet, inga bandsträckare eller växlar eller slirkopplingar eller andra komplikationer. Gummihjulet hade såklart förgåtts, men på ett uttorkande sätt, och spruckit och fallit av, istället för att förgås på ett drypande sätt och täcka insidan av bandarmekanismen med smetig tjära som på en del andra bandspelare. Detta uppskattas! Då behövde vi bara ett nytt hjul istället för det trasiga. Så jag gick till Biltema och köpte en rulle vulktejp. Det kanske funkar?



Det ser rätt bra ut i alla fall. Men det måste sandpappras ner lite där änden av tejpen är, så att det inte är ett litet trappsteg. Men det hanns inte med i oktober. Det får bli nån annan gång. Då får vi också titta på de två andra nätaggregaten, som äter säkringar, och så får vi göra nya försök att boota nåt. Men VT103 är kul, så det kanske blir snart!

Spöket av forna tiders hjul

Updates tre VT103-datorer, varav en nu åter fungerar

Om du vet nånting om VT103, särskilt var Updates exemplar kom ifrån och vad de använts till, eller om du har mjukvara på DECtape II, skriv gärna till Uptime.

November månads dator är Elsa, vår PDP-12, som gick sönder för några år sen men som vi nu ska försöka återställa till körbart skick.

Vi ses i nästa nummer!