COMAL KAPSEL 2.02

for Commodore 128

TILLÆG TIL COMAL FOR COMMODORE 64

Commodore

COPYRIGHT

ASSISTANCE

Hvis De har nogle kommentarer angående denne COMAL manual eller selve programmeringssproget, være venlig at give disse videre til Deres forhandler. Commodore Data A/S har bestræbt sig på at gøre indholdtet af manualen korrekt og sikre at selve programmeringssproget virker efter hensigten. Fejl opdaget af brugere rettes så hurtigt som muligt. Deres evt. hjælp i så henseende vil blive værdsat meget af andre brugere.

ANSVAR

Hverken Commodore Data A/S eller dette firmas forhandlere eller leverandører påtager sig noget ansvar, direkte eller underforstået, vedrørende det heri omtalte programmeringssprogs kvalitet, ydeevne, værdi eller anvendbarhed til specielle opgaver. Programmellet sælges "som beset". Risiko med hensyn til kvalitet og ydeevne tages af køberen. Såfremt programmellet skulle udvise defekter efter anskaffelsen, må køberen (og ikke UniComal, Commodore Data A/S, dets forhandlere, distributører eller leverandører) påtage sig alle nødvendige reparations- og serviceomkostninger i forbindelse med enhver tilfældig skade forårsaget af defekter i programmellet eller tilhørende dokumentation.

INDHOLDSFORTEGNELSE

Sats og layout: Silkeborg Soldata 0684-1196

Tryk: Silkeborg Bogtryk A/S 0682-1655

1. udgave, 1. oplag, oktober 1986

COMMODORE 128 COMAL

I dette **C-128 Tillæg** beskrives de nye og udvidede muligheder, som brugeren af en C-128 COMAL kapsel version 2.02 kan udnytte. Dette **Tillæg** er et *supplement* til manualen **COMAL for Commodore 64** og bør indsættes bagest i ringbindet.

Med en C-128 COMAL kapsel til rådighed har man ikke blot et fremragende *EDB-sprog* til disposition. I kapslen følger redigeringsmuligheder og et *arbejdsmiljø* for programmøren, som kan måle sig med de bedste, der findes i verden i dag.

OPSTIL DATAMATEN

Har man tidligere benyttet C-64 COMAL kapslen, skal man blot slukke for datamaten og tilbehør og udskifte C-64 kapslen med den nye C-128 kapsel.

Er Commodore 128 maskinen en nyanskaffelse, følg da vejledningen i Deres C-128 manual. Opstil datamaten, tilslut skærmen og diskettestationen (eller båndoptageren).

DE TO TEKSTSKÆRME

Hvis De bruger en farveskærm, læg da mærke til, om Deres skærm er en "RGB" skærm, som er i stand til at vise 80 tegn på hver linie, eller en "PAL" type, som kun kan vise 40 tegn pr. linie. Hvis De bruger Commodores 1901 skærm, kan De tilslutte to sæt kabler, og ved hjælp af en omskifter t.v. på skærmens betjeningspanel, kan De skifte frem og tilbage mellem RGB (80-tegnsskærm) og PAL (40-tegnsskærm).

Begge skærme kan bruges med Deres C-128 kapsel. Der er mange fordele ved at benytte 80-tegnsskærmen, især ved programredigering og til programmer, der håndterer tekst. Hvis man vil vise sprites eller anden farvegrafik, er det nødvendigt at skifte over til 40-tegnsskærmen. Bemærk også, at de programmer, der beskrives i

COMAL for Commodore 64, altid bruger 40-tegnsskærmen.

Det er også muligt at køre et program, der udnytter begge skærme på én gang. Er to skærme tilsluttet, kan der vises farvegrafik på 40-tegnsskærmen, evt. med sprites i bevægelse, mens der vises tekst på 80-tegnsskærmen.

Bemærk, at procedurerne **textmode(1)** og **textmode(0)** i systempakken gør det muligt at skifte mellem 80-tegns- og 40-tegnsskærmen under programudførelse. Fra tastaturet kan man taste *sekvensen* **<ESC><X>** for at skifte frem og tilbage mellem de to skærme. Funktionen **inqsys(5)** fra systempakken kan bruges til at oplyse om den aktuelle skærmtilstand. Se mere herom under afsnittet **Tillæg til systempakken**.

Tasten **<40/80 Display>** øverst på C-128 kan også bruges til at vælge skærm. Hvis tasten er *nede*, når datamaten tændes, vælges 80-tegnsskærm. Er tasten *oppe*, vælges 40-tegnsskærm.

OPSTART

Er datamaten nu opstillet med kapslen på plads, kablerne tilsluttet og den rigtige skærm valgt, kan De tænde for datamaten og tilbehøret og begynde at bruge C-128 COMAL. Hvis en diskettestation er til rådighed, indsæt da C-128 Demodisketten, og skriv:

LOAD "C-128 demo" <Return>

Skriv så **RUN** og tast **<Return>** (eller tryk bare **<F7>** ligesom på 64'eren).

Når programmet har kørt lidt, afbryd det ved hjælp af **<RUN/STOP>** tasten. Prøv så at indtaste **LIST** (evt. **<F6><Return>**), og bemærk udskriften, gerne på 80-tegnsskærmen, hvis den er til rådighed.

C-128 SKÆRMREDIGERING

Har man nogensinde ønsket, at det kunne lade sig gøre at *rulle* en programlistning både op og ned på skærmen, får man sit ønske

opfyldt nu. Tast **<Alt-Markør Op>** (altså hold **<ALT>** nede - den er øverst på tastaturet, tredie taste fra venstre - og tryk på pil opad- også øverst, længere mod højre). Voila! Nye programlinier kommer til syne ovenfra og nedefter. Tryk **<Alt-Markør Ned>**, og nye programlinier kommer nedfra, programudskriften ruller opad. De andre markørtaster (nederst) kan også bruges til dette formål.

Ligesom med C-64 COMAL kan man starte/stoppe udskriften ved at taste **<Mellemrum>**. På den anden side kan man ikke længere holde **<Ctrl>**-tasten nede, mens programmet listes, for at få en langsom listning. Brug **<Alt>** og markørtasterne for at køre langsomt op og ned i udskriften.

Med den nye COMAL-kapsel har UniComal tilstræbt en standardisering i forhold til den eksisterende skærmredigering på C-128. Derfor er der en del ændringer i forhold til C-64 kapslen. Se **Appendiks B** for en oversigt over alle **<Alt>**-, **<Ctrl>**- og **<Esc>**-koder og de ændringer, der er sket.

NYE MULIGHEDER MED C-128 COMAL

Man lærer mest om C-128 COMALs mange muligheder ved at bruge sproget og dets faciliteter til løsning af nogle konkrete programmeringsopgaver. De korte programeksempler i forbindelse med beskrivelsen af de enkelte nøgleord og de længere eksempler på C-128 Demodisketten kan måske give idéer og inspiration hertil. Udover det, at man får 40K i stedet for 30K til brugerprogrammer, adgang til brugen af hele C-128 tastaturet, og at disketteoperationer med C-128 kapslen er blevet meget hurtigere, er der mange andre nye muligheder. Her er højdepunkterne af de nye faciliteter:

* *Udvidede INPUT faciliteter:* Nu kan programmøren bruge nye procedurer fra systempakken til at gøre programmer mere brugervenlige. Under indtastning kan man bruge forskellige taster, f.eks. **<Markør Op>** og **<Markør Ned>**, til at afslutte en indtastning, så markøren hopper op eller ned til andre inputfelter. Se under **inputpos**, **termchars**, **termpos** og **termchar$** for mere herom.

* *C-128 Fontpakken:* C-128'eren har jo mulighed for at bruge

enten 40-tegns- eller 80-tegnsskærme. Derfor er fontpakken udvidet, så der også kan bruges brugerdefinerede tegnsæt på 80-tegnsskærmen. Se mere herom under **Andre nøgleord**.

* *Nye redigeringsmuligheder:* Der er kommet mange flere redigeringsfaciliteter. Nu kan man rulle programmer opad eller nedad, indsætte ekstra linier på skærmen eller i programmer, sætte vinduer og skifte mellem 40- og 80-tegnsskærm. Hvis man f.eks. får brug for at indskyde flere programlinier lige efter programlinien, hvor markøren står, skal man blot taste **<Alt-N>**, og nye linienumre genereres automatisk! Se **Appendiks B** for en oversigt over alle disse muligheder.

* *RAM-filer:* Én af de virkelige store nyskabelser i C-128 kapslen i forhold til C-64 kapslen er muligheden for at lagre filer på op til 40K i RAM. UniComal har udnyttet et ubrugt 40K lagerområde til dette formål og forsynet brugeren med over 20 procedurer og funktioner i COMAL-pakken *ramfiles*. Der findes en nærmere beskrivelse af de mange muligeheder i afsnittet, **Den nye ramfiles pakke**. Se også demonstrationprogrammet "RAMFILES" på C-128 Demodisketten.

* Kommunikation både med databaser og med andre datamater er blevet gjort lettere med C-128 kapslen. Flere procedurer og funktioner er blevet føjet til systempakken til håndtering af RS-232 grænsefladen. Et brugervenligt kommunikationsprogram vil hermed blive meget lettere at lave.

KOMPATIBILITET MED C-64 COMAL

Under udviklingen af C-128 COMAL har UniComal søgt at gøre det let at føre C-64 COMAL programmer over til den nye COMAL. (Da C-128 COMAL rummer mange nye procedurer og funktioner, vil det normalt ikke være så let at gå den anden vej.)

Hvis man har et C-64 program, som skal bruges med den nye kapsel, kan man normalt blot indlæse programmet på normal vis ved hjælp af LOAD kommandoen.

Er der undtagelsesvis tale om et program, hvortil der er knyttet et

maskinkodeprogram ved hjælp af LINK kommandoen, skal følgende fremgangsmåde benyttes:

* Start med C-64 eller C-128 kapslen på plads i C-128'eren. Indlæs C-64 programmet ved hjælp af LOAD kommandoen. LIST så programmet til diskette:

  LIST "programnavn"

* Sørg nu for, at den nye C-128 COMAL kapsel er på plads. Det vil måske være en god idé at initiere en *helt ny diskette* ved hjælp af **PASS** kommandoen, inden man fortsætter. Man kan f.eks. skrive:

  PASS "n0:C-128 programmer,01"

  Bemærk, at man hermed opnår, at der bliver plads til 1328 blokke (á 254 tegn), hvis man har en 1571 diskettestation tilsluttet. Der bliver altså plads til dobbelt så mange blokke som på C-64'erens 1541 disketter.

* Sæt C-64 disketten på plads, og hent C-64 programmet ind ved hjælp af ENTER:

  ENTER "programnavn"

* Brug C-128'erens redigeringsmuligheder for at tilpasse programmet, så det kører korrekt på 80-tegnsskærmen, mm. Når man er færdig, kan det lagres på C-128 disketten ved hjælp af SAVE kommandoen.

* Benyt så "C128SYMB" på C-128 Demodisketten med Deres assembler i stedet for "C64SYMB" for at tilpasse maskinkodeprogrammet til 128'eren. Se vejledningen om maskinkode i **Appendiks C** for mere herom. Når COMAL-programmet og maskinkoden er i orden, kan man bruge LINK for at tilknytte maskinkodeprogrammet, så det kan blive gemt med programmet på en samlet fil ved hjælp af SAVE.

NØGLEORDSOVERSIGT

I dette afsnit finder De en kort beskrivelse af alle *nye* eller *reviderede* nøgleord, der findes i C-128 COMAL. Øvrige faciliteter står allerede beskrevet i **COMAL for Commodore 64**. Her er alle nye/reviderede nøgleord (**s**=sætning, **c**=kommando, **p**=procedure, f=funktion) emnegrupperede. Desuden anføres, om nøgleordet hører til COMAL-kernen, er en udvidelse, eller om det hører til en pakke.

I efterfølgende afsnit, **Tillæg til systempakken, Den nye ramfiles pakke** og **Andre nøgleord**, findes en nærmere beskrivelse af de enkelte nøgleord, ordnet alfabetisk inden for hvert afsnit.

<u>Nye muligheder med PRINT og INPUT sætninger</u>

PRINT	s,c	kernen	separatorerne ; og , får ny betydning.
ZONE	s,c	kernen	virker nu på ; ikke ,.
option(1)	p	system	tillader brugen af den gamle definition på ZONE.
INPUT	s,c	kernen	; og , får ny betydning, når de undertrykker linieskift i slutningen af en INPUT sætning.
inputpos	p	system	placerer markøren i INPUT-feltet.
termchars	p	system	bestemmer afslutningstegn(ene) i INPUT-sætningen.
termpos	f	system	oplyser om markørposition ved INPUT afslutning.
termchar$	f	system	returnerer tegnkoden for INPUT afslutningstegnet.
GET$	f	kernen	henter et specificeret antal tegn fra en given fil (er med både i C-64 og C-128 kapslen).

<u>RAM-filer i C-128'erens 40 KB ekstra hukommelse</u>

ramfiles	pakke	ramfiles	navn på ny pakke til ramfiler i 40K RAM område.
deleteallrf	p	ramfiles	fjerner alle ramfiler.
deleterf	p	ramfiles	fjerner bestemt ramfil.

createrf	p	ramfiles	opretter en ramfil.
eofrf	f	ramfiles	indikerer slut på en ramfil.
posrf	p	ramfiles	anbringer filpointer i en ramfil.
writenum	p	ramfiles	skriver et tal til en ramfil.
readnum	p	ramfiles	henter et tal fra en ramfil i form af en REF parameter.
getnum	f	ramfiles	returner et tal fra en ramfil.
writestr	p	ramfiles	skriver en streng til en ramfil.
writerefstr	p	ramfiles	lagrer en REF streng i en ramfil.
readstr	p	ramfiles	henter en streng fra en ramfil i form af en REF parameter.
getstr$	f	ramfiles	returnerer en streng fra en ramfil.
writerec	p	ramfiles	skriver en formatteret post til en ramfil.
writerefrec	p	ramfiles	skriver en formatteret REF post til en ramfil.
readrec	p	ramfiles	læser en formatteret post fra en ramfil.
getrec$	f	ramfiles	henter en formatteret post fra en ramfil.
inqrf	f	ramfiles	kan bruges til at afsløre formatet af ukendt ramfil.
saverf	p	ramfiles	lagrer en ramfil til disk.
loadrf	p	ramfiles	henter en hel ramfil fra disk.
saveallrf	p	ramfiles	lagrer alle ramfiler til disk.
loadallrf	p	ramfiles	henter alle ramfiler fra disk.
fieldtype$	f	ramfiles	returnerer feltbeskrivelsesteksten for en ramfil.
freerf	f	ramfiles	returnerer antal frie bytes i ramfil-området.
SIZE	c	kernen	udvidet så der også angives antal brugte og ledige tegn i ramfiles området.

Tegnkonvertering, kommunikation og filer

setmapping	p	system	styrer tegnkonvertering i filsystemet når /**a+** bruges.

resetmapping	p	system	sætter normal Commodore mapping.
char'in'buffer	f	system	returnerer antal tegn i RS-232 sende- eller modtagebuffer.
clearbuffer	p	system	tømmer RS-232 sende- eller modtagebuffer.
rs232status	f	system	returnerer værdien af seriel-portens statusregister.
GET$	f	kernen	ændret for forbedret seriel kommunikation.
IN	s,c	kernen	"" **IN streng$** returnerer nu FALSE (altså 0).
option(2)	p	system	anvendes ved relative filer for at undgå fejl i 1541/1571 drev.
setserialport	p	system	indstiller serielportens kom-munikationsparametre.
bin$	f	system	returnerer streng svarende til binær repr. af et tal.
hex$	f	system	returnerer streng svarende til hexadecimale repr. af et tal.

Brugen af fonts

discardfont	p	font	genopretter standardtegnsættet.
selectfont	p	font	skifter til det angivne tegnsæt.

Håndtering af skærme og farvevalg

textmode	p	system	skifter mellem 40- og 80-tegns-skærm.
textwindow	p	system	sætter tekstvindue.
getscreen2	p	system	kopierer indholdet af et tekst-vindue til en streng.
setscreen2	p	system	overfører informationer fra en streng til et tekstvindue.
textcolor	p	graphics	ændrer markørfarven på 40-tegnsskærmen.
textborder	p	graphics	ændrer kantfarven på 40-tegns-skærmen.
textbackground	p	graphics	ændrer baggrundsfarven på 40-tegnsskærmen.

textcolors	p	system	ændrer farverne på den aktuelle skærm.
plottext	p	graphics	skriver tekst på 40-tegnsskærmen men ikke 80-tegnsskærmen.
vdcpoke	p	system	skriver direkte til registrene i 80-tegnsskærmens styreenhed.
vdcpeek	f	system	læser registre i 80-tegnsskærmens styreenhed.
inqsys	f	system	returnerer diverse systemparametre.
cursormode	p	system	tænder, slukker eller ændrer markøren på 80-tegnsskærm.

Hop til og fra COMAL, mm.

BASIC	c	udvidelse	systemet forlader COMAL og går ind i Commodore BASIC 7.
monitor	p	system	systemet går til maskinkodemonitoren.
cpuspeed	p	system	bruges til at vælge clockfrekvens.
TRACE	c	kernen	bruges til at iværksætte en sporing af programforløbet.

Klokken, sprites og grafikdump

bell	p	system	giver en lyd. Kan nu afbrydes.
getshape$	f	sprite	returnerer en streng med en tegning.
hardcopymode	p	system	styrer udskrift af tekstskærmen til printer (hardcopy).

TILLÆG TIL SYSTEMPAKKEN

Når man skriver **USE system** som kommando direkte fra tastaturet, eller sætningen **USE system** udføres i et program, bliver et stort antal nyttige procedurer og funktioner gjort tilgængelige. Også i C-128 kapslen har man adgang til alle de faciliteter, som er beskrevet i **COMAL for Commodore 64, Kapitel 5.**

Udover alle disse faciliteter er der kommet mange nye og udvidede muligheder. Disse kan i store træk inddeles som:

* forbedret udnyttelse af RS-232 grænsefladen, herunder mulighed for avancerede kommunikationsprogrammer, så Deres C-128 kan bruges med databaser;

* fuld udnyttelse af C-128'erens 80-tegnsskærm, inklusiv brugen af vinduer, styring af cursormåde, skærmdump til printer, trykning af anførselstegn på Commodore printere, mm.;

* specielle funktioner og procedurer til at skabe kompatibilitet med C-64 programmer (se **option**).

Her er en oversigt, ordnet alfabetisk, af de nye og ændrede procedurer og funktioner, som Deres COMAL kapsel version 2.02 rummer.

bell(varighed#)

er en procedure, som anvendes til at lave COMAL lydsignalet (klokken). *Varighed#* lig med 1 svarer til den normale COMAL-klokke. Som noget nyt kan lyden slås fra og slås til.

Eksempel: Tast sekvensen **<Esc><H>** - klokken lyder nu ikke ved fejl eller ved brug af **bell** som procedure. Tast sekvensen **<Esc><G>** - klokken er slået til igen.

bin$(værdi#)

er en strengfunktion, som returnerer strengen svarende til den binære repræsentation af parameteren *værdi#* (0 til 65536). F.eks. returnerer **bin$(255)** strengen "0000000011111111".

char'in'buffer(retning#)

er en funktion, som returnerer antallet af tegn i RS-232 bufferne. Parameteren *retning#* lig med **0** betyder modtagebuffer, og lig med **1** betyder sendebuffer.

clearbuffer(retning#)

er en procedure, som tømmer RS-232 modtage- hhv. sendebuffer, når *retning#* er **0** hhv. **1**.

cpuspeed(tilstand)

er en procedure, som kan anvendes til at bestemme clockfrekvensen. *Tilstand* kan være lig med **0**, **1** eller **2**. Standardværdien er **0**.

Eksempler:	**cpuspeed(0)**	COMAL vælger selv clockfrekvensen: 1 MHz vælges for 40-tegnsskærm eller grafik, ellers vælges 2 MHz.
	cpuspeed(1)	clockfrekvensen vælges til konstant 1 MHz.
	cpuspeed(2)	clockfrekvensen vælges til konstant 2 MHz. 40-tegnsskærmen blankes og kan ikke bruges i dette tilfælde.

cursormode(tilstand#)

er en procedure, som kan anvendes til at tænde, slukke og ændre markøren på 80-tegnsskærmen. Parameteren *tilstand#* har følgende betydning:

tilstand#

0: slukker markøren
1: tænder markøren i aktuel tilstand
2: tænder ikke-blinkende blokmarkør
3: tænder blinkende blokmarkør
4: tænder hurtigblinkende blokmarkør
5: tænder ikke-blinkende stregmarkør
6: tænder blinkende stregmarkør
7: tænder hurtigblinkende stregmarkør

Eksempel:

```
0010 PAGE
0020 USE system
0030 cursormode(5)
0040 LOOP
0050   PRINT KEY$,
0060 ENDLOOP
```

Ovenstående eksempel viser en ikke-blinkende stregmarkør på skærmen. Tegn fra tastaturet bliver trykt (uden vognretur) indtil programmet afbrydes med **<Run/Stop>**. (Under en INPUT-sætning blive markøren altid tændt.)

getscreen2(skærm$)

er en procedure, som gemmer en del af (evt. hele) tekstskærmen. Billedinformationen svarende til det aktuelle vindue gemmes i strengen *skærm$*. Det lagrede billede kan hentes frem igen ved hjælp af proceduren **setscreen2**(*skærm$*). Bemærk, at **getscreen2** og **setscreen2** ikke er kompatible med **getscreen** og **setscreen**, som kun arbejder på 40-tegnsskærmen og som altid arbejder med hele skærmen. Bemærk, at **setscreen2** og **getscreen2** ikke kan bruges til at overføre skærmbilleder fra 40-tegnsskærmen til 80-tegnsskærmen eller omvendt. Et eksempel på brugen af **getscreen2** kan ses på C-128 Demodisketten (se "demo.window.80" og "demo.window.40").

hardcopymode(tegnsætmåde#,quotemåde#)

er en procedure, som bruges til at styre udskrift af tekstskærme til en printer. Parameteren **tegnsætmåde#** kan være lig med **0** eller **1**:

tegnsætmåde#=0 er defaultværdien og angiver, at COMAL vælger *store og små bogstaver* eller *store bogstaver og grafik* afhængigt af, hvad der aktuelt er på skærmen. I 40-tegnstilstand gælder samme tegnsæt for hele skærmen, mens der i 80-tegnstilstanden gælder, at de to tegnsæt kan blandes.

tegnsætmåde#=1 angiver, at tegnsæt ikke ændres ved skift på skærmen, men alene ud fra **setprinter** proceduren i systempakken.

Størrelsen *quotemåde#* angiver, hvorledes skrivning af anførselstegnet " klares på printer. På Commodore printere resulterer skrivning af et ulige antal " i, at printeren skifter over til quote måde. Dette resulterer i, at kontroltegn skrives i negativ skrift. Dette indebærer, at det ikke er muligt at skifte tegnsæt og slå negativ skrift fra og til. I C-64 COMAL er dette søgt løst ved at konvertere tegn skrevet i negativ på skærmen efter et ulige antal " til kontrolkoder, som jo skrives i negativ skrift. Herved kan nogle af tegnene skrives korrekt på printeren. C-128 COMAL er udvidet med nogle andre muligheder for at løse dette problem. Disse kan vælges ved hjælp af parameteren **quotemåde#**.

quotemåde#=0 angiver, at tegnet " skrives som grafik og derved ikke aktiverer quotemåde i printeren. Dette kræver dog, at printeren er kompatibel med Commodores MPS-801 printer til grafikudskrift. Denne værdi vælges ved opstart.

quotemåde#=1 angiver C-64 COMAL tilstand. I denne tilstand bør *quotemåde#* sættes lig med **1**, da skift mellem tegnsæt ellers kan resultere i udskrift af nogle ekstra tegn i negativ skrift.

quotemåde#=2 til 255 angiver, at der i stedet for tegnet " skrives **CHR$(quotemåde#)**. På denne måde er det muligt at skrive et andet tegn i stedet for ", f.eks. apostrof ' eller **CHR$(254)**, som er et programmerbart tegn på en Commodore MPS 802 printer. På printere, hvor quotemåde ikke er

indført eller kan slås fra, vælges **quote-måde#=34**, som giver udskrift af det normale " tegn.

Ved opstart udføres der automatisk **hardcopymode(0,0)**.

hex$(værdi#)

er en strengfunktion, som returnerer strengen svarende til den hexadecimale repræsentation af parameteren *værdi#* (0 til 65536). F.eks. returnerer **hex$(255)** strengen "00ff".

inputpos(position#)

er en procedure, som sætter markørens relative startposition i et inputfelt. Et 1-tal angiver første position. **Inputpos** gælder for den efterfølgende INPUT sætning og sættes herefter til **1** igen. Den sættes også til 1 ved RUN. Se programmet "demo.termchar.80" på C-128 Demodisketten.

inqsys(type#)

er en funktion, som returnerer følgende værdier afhængig af parameteren *type#*:

type#:
- 1: tekstvindue række min.
- 2: tekstvindue række maks.
- 3: tekstvindue søjle min.
- 4: tekstvindue søjle maks.
- 5: teksttilstand - returnerer **0** hhv. **1** for 40- hhv. 80-tegnsskærm.
- 6: aktuel cpuspeed - returnerer **1** hhv. **2** for 1 hhv. 2 MHz.

monitor

er en procedure, som kan bruges til at kalde maskinsprogsmonitoren

fra COMAL. Der returneres fra monitoren på normal måde med en X-kommando. Returneringen foregår ved hjælp af en fejlmeddelelse "monitor terminated". Det er derfor også muligt at kalde monitoren fra et kørende program og så fange retur fra monitor ved hjælp af fejlhåndtering. Dette kan være en stor hjælp under fejlretning af maskinsprogsprogrammer. Bemærk, at hvis der anvendes et "soft-loadet" tegnsæt i 40-tegnsskærmen, kan monitoren ikke kaldes fra denne skærm; det samme gælder desuden fra splitscreen. Dette skyldes, at skærmeditor kaldt fra monitoren ikke kan anvende skærmhukommelsen, der ligger "under" monitor ROM'en. Hvis det forsøges, gives en fejlmeddelelse.

option(1,gammel#)

er en procedure, som gør det muligt at skifte til den gamle definition af printskilletegn komma (,) og semicolon (;). (Se også ZONE.) Hvis *gammel#* er TRUE, virker PRINT efter den gamle definition, idet default for ZONE sættes til 0, og komma benyttes som skilletegn for tabuleringer. Semicolon betyder så, at der laves et enkelt mellemrum. Ved opstart af maskinen og ved RUN udføres **option(1,FALSE)**. Hvis **option** indføres i et program, som skal kunne udføres både under COMAL 2.01 og 2.02, må **option** kun kaldes, når ZONE er 1 ved opstart af et program.

Eksempel:
```
0010 IF ZONE THEN option(1,1)
0020 ...
```

Det er en fordel at bruge den nye ZONE definition, bl.a. fordi man hermed undgår, at komma, når det forekommer i ASCII-filoperationer, får en uventet betydning.

option(2,postilstand#)

er en procedure, der er indført på grund af fejl i 1541/1571 diskettestationerne. (Denne procedure erstatter den tidligere **setrecord-delay**.) Parameteren *postilstand#* har følgende betydning:

postilstand#:	bit 7	(værdi 128) angiver, at status fra diskettestationen læses efter alle positioneringer

af læse/skrivehovedet.

bit 6 (værdi 64) angiver, at der også positioneres efter alle skrive/læseoperationer.

bit 0 (værdi 1) angiver, at der før positionering kontrolleres, at foregående diskoperation er afsluttet. Dette sker ved at læse i diskdrevets hukommelse. Denne operation kan være inkompatibel med andre drev end 1541 eller 1571.

Når maskinen tændes, udføres **option(2,128)**. Indtil fejlene rettes i 1541/1571, er det derfor nødvendigt at udføre **option(2,128+64+1)**. Dette gør diskoperationer langsommere, men fejlfrie. Hvis der kun læses fra en relativ fil, er det tilstrækkeligt at udføre **option(2,128+64)**.

resetmapping

er en procedure, som sætter "normal" afbildning (*eng: mapping*) af tegn, altså ligesom i C-64 kapslen:

normal afbildning ved modtagelse:

97...125	=>	65...93	(små bogstaver)
65...93	=>	193...221	(store bogstaver)
95	=>	164	(understreg tegnet)

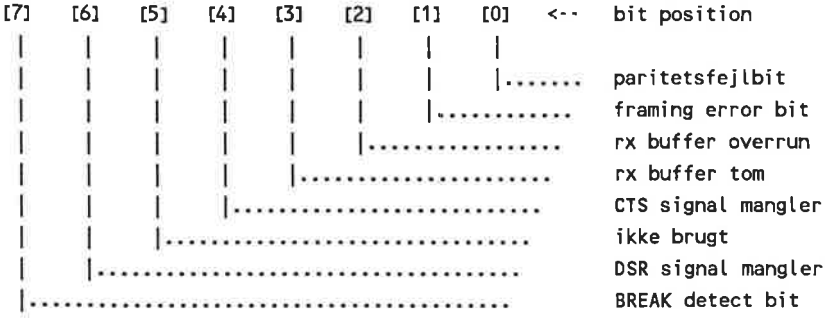
normal afbildning ved afsendelse:

65...93	=>	97...125	(små bogstaver)
193...221	=>	65...93	(store bogstaver)
164	=>	95	(understreg tegnet)

Bemærk, at når afbildningen ændres, gælder den for alle filer, der er åbnet med **/a+**, og afbildningen gælder, indtil der slukkes for makinen, eller den ændres med **setmapping** eller **resetmapping**. Se også **setmapping**.

rs232status

er en funktion, som returnerer værdien af serielportens statusregister og derefter nulstiller registeret. Funktionen returnerer følgende værdier:



Bemærk, at bit [3] (rx buffer tom) kan aflæses efter GET$. Den vil være lig med 1, hvis det ønskede antal tegn ikke var til stede i rx-bufferen. Rx er en forkortelse for det engelske ord *receive*, som betyder *modtage*.

setmapping(retning#,tegn$,til_tegn$)

er en procedure, som letter konvertering af tegn ved kommunikation med andre systemer, f.eks. via en RS-232 grænseflade. Konvertering var muligt i C-64 COMAL i filsystemet, f.eks. ved at skrive **/a+** efter et filnavn, og **setmapping** er en udvidelse af denne facilitet. Parameteren *retning#* er **0** for modtagelse (READ, GET eller INPUT FILE) og **1** for sende (WRITE eller PRINT FILE). Proceduren **setmapping** præciserer, at tegnet *tegn$* bliver erstattet af *til_tegn$* under efterfølgende overførsler.

setscreen2(skærm$)

er en procedure, som overfører en del af tekstskærmen, som tidligere var gemt ved hjælp af **getscreen2**. Billedinformationen er gemt i strengen *skærm$*. Billedet placeres med øverste venstre hjørne i det aktuelle tekstvindue og beskæres om nødvendigt. Des-

uden sættes markørfarve og position som angivet i *skærm$*. Bemærk, at **getscreen2** og **setscreen2** ikke er kompatible med **getscreen** og **setscreen**, som kun arbejder på 40-tegnsskærmen, og som altid arbejder med hele skærmen. Bemærk, at **setscreen2** og **getscreen2** ikke kan bruges til at overføre skærmbilleder fra 40-tegnsskærmen til 80-tegnsskærmen eller omvendt. Et eksempel på brugen af **getscreen2** kan ses i programmerne "demo.window.40" og "demo.window.80" på C-128 Demodisketten.

setserialport(attributer$)

er en procedure, som ændrer standardtilstanden for serielporten (ligesom **setprinter** gør for printere). Parameteren *attributer$* er en streng, som rummer informationer om serielportens kommunikationsparametre. Specifikationerne kan også forekomme i forbindelse med en OPEN FILE sætning eller kommando. Udover *transmissionshastigheden*, *databits*, *stopbits* og *paritet* som i C-64 COMAL, kan man nu også specificere *halv* eller *fuld duplex*, *handshake* og ekstra *paritetsparametre*. Alle muligheder vises i nedenstående tabel:

parameter syntaks		*mulige værdier*	*standardværdi*
baudrate	b<baud>	50-2400	b300
databits	d<antal>	5-8	d7
stopbits	s<antal>	0-2	s2
paritet	p<type>	n=ingen e=lige o=ulige m=mark s=space	pn
dupleks	d<type>	h=halv f=fuld	df
handshake	h<type>	3=3-linie x=x-linie	h3

termchar$

er en funktion, som angiver, hvilket tegn der blev brugt til at afslutte indtastningen i det sidste inputfelt, som brugeren har reageret på. F.eks. returnerer **termchar$** CHR$(13), når der er afsluttet

med **<Return>**. Se også INPUT, **termpos**, **termchars$**, og **inputpos**. Se C-128 Demodisketten for et programeksempel: "demo.termchar.80".

termchars(tegn$)

er en procedure, som definerer de tegn, der kan afslutte indtastning i det næste inputfelt. Standardværdien for **tegn$** er **chr$(13)**, som betyder, at indtastning afsluttes ved tryk på **<Return>**. Følgende tegn har en speciel betydning:

chr$(128)	afslutter ved **<markør tv.>**, når markøren står helt til venstre i inputfeltets første position.
chr$(130)	afslutter ved **<markør th.>**, når markøren står helt til højre i inputfeltets sidste position.
chr$(132)	afslutter, når der skrives et tegn i sidste position af inputfeltet.

Termcharstegnene gælder indtil en ny **termchars** procedure forekommer i programmet eller indtil RUN udføres igen.

Eksempel:
```
0010 USE system
0020 termchars(""13""27"")
0030 INPUT "Testing termchars... ": svar$
0040 PRINT "Du tastede på <Return> eller <Esc>!"
0050 END
```

termpos

er en funktion som angiver, hvor i inputfeltet markøren var ved afslutning af indtastningen. Værdien 1 svarer til første position i feltet. Se også **inputpos**, **termchars**, **termchar$** og INPUT.

textcolors(kant#,baggrund#,tekst#)

er en procedure, som kan anvendes til at ændre på tekstfarven på den aktuelle skærm. Parametrerne *kant#*, *baggrund#* og *tekst#* er hele tal fra 0 til 15. De bestemmer farverne for skærmens kant, baggrund og tekst, der skrives på skærmen. Bemærk dog, at kant-

farven på 80-tegnsskærmen altid følger baggrundsfarven. Farvekoderne kan ses på bageste side i dette tillæg.

textmode(tilstand#)

er en procedure, som tillader skift mellem 40- og 80-tegnsskærmen. *Tilstand#* lig **0** angiver 40-tegnsskærmen, og *tilstand#* lig **1** svarer til 80-tegnsskærmen. Bemærk, at når der skiftes til 80 tegn blankes 40-tegnsskærmen, medmindre **cpuspeed** er sat til **1**, eller grafikpakken er aktiv.

textwindow(rækkemin,rækkemax,søjlemin,søjlemax)

er en procedure, som sætter tekstvinduet og placerer markøren i øverste venstre hjørne. Man kan ophæve vinduet og få adgang til hele tekstskærmen igen ved at taste **<Alt-F>** eller **<Home><Home>**.

vdcpeek(register)

er en funktion som muliggør, at man kan læse direkte fra registrene i 80-tegnsskærmens styreenhed.

vdcpoke(register,tal)

er en procedure, som gør det muligt at skrive til registrene i 80-tegnsskærmens styreenhed.

DEN NYE RAMFILES PAKKE

Pakken *ramfiles* findes kun i C-128 COMAL kapslen. Denne nyskabelse er mulig, fordi ca. 40K er ledig i en Commodore 128. Det var oplagt for UniComal at udnytte denne plads til en avanceret form for datalagring.

Ramfiles pakken åbner mange spændende muligheder for hurtig datahåndtering. Brugeren kan naturligvis selv bestemme, hvordan data skal struktureres. Det er tilladt inden for de 40K, som er til rådighed, at definere op til 32 RAM-filer i samme program:

RAM-fil 1	**RAM-fil 2**	. . .	**RAM-fil 32**
datasæt 1	*datasæt 1*	. . .	*datasæt 1*
datasæt 2	*datasæt 2*	. . .	*datasæt 2*
.	.		.
.	.		.
datasæt N_1	*datasæt* N_2	. . .	*datasæt* N_{32}

Hvert datasæt (eller post) kan indeholde flere felter, som kan bestå af *reelle tal*, *hele tal*, *oktetter* (bytes), *Booleske variabler* (enkelte bits) eller *strenge*. For eksempel, hvis De gerne vil lave en database med følgende informationer:

kundenummer	**(et helt tal)**
kundens navn	**(en streng på maks. 38 tegn)**
kundens adresse	**(en streng på maks. 78 tegn)**
det samlede køb	**(et reelt tal)**
rabat gives	**(ja eller nej)**
30 dages kredit	**(ja eller nej)**

kunne De strukturere hvert datasæt som følger:

datasæt:	**helt tal**	**streng 1**	**streng 2**	**reelt tal**	**bit 1**	**bit 2**

Et helt tal fylder 2 bytes, streng 1 fylder 40 bytes (38 + 2, der angiver strengens længde), streng 2 fylder 78 + 2 = 80, det reelle

tal fylder 5, og de to logiske variabler fylder tilsammen 2 bits (fra 1 til 8 bits optager altid 1 byte). Dette giver tilsammen 128 bytes for hvert datasæt. Man kan beregne lagerforbruget ved formlen:

bytes brugt = antal poster x postlængden + antal felter x 4 + 16

I dette tilfælde vil f.eks. 300 datasæt fylde: 300 x 128 + 6 x 4 + 16 = 38440 bytes. Der er ialt 40.896 bytes til rådighed for RAM-filer.

Som det vil fremgå af følgende alfabetiske beskrivelsen af de enkelte procedurer og funktioner i pakken ramfiles, kan datasættets struktur defineres ved hjælp af proceduren **createrf(1,321,"I S38 S78 R LL")**.

I det følgende beskrives hver ramfiles procedure og funktion, ofte suppleret med korte eksempler. Se C-128 Demodisketten for et større sammenhængende eksempel ("RAMFILES") på brugen af RAM-filer.

createrf(filnr#,antal_poster#,beskrivelse$)

er en procedure, som anvendes til at oprette en RAM-fil betegnet *filenr#* (1-32). Filpointeren sættes til 1. felt i post 1. Denne filstruktur ligner på flere måder en direkte fil på diskette, da man på forhånd skal angive, hvor mange poster filen skal rumme (*antal_poster#* - maksimum 32767), og hvordan de enkelte felter i posten skal være (*beskrivelse$*). *Beskrivelse$* er en streng, som rummer oplysninger om felterne i hver post. Hvert felt beskrives af et bogstav (stort eller lille), der angiver felterne som følger:

"R" eller "r"	står for et *reelt tal*. Et reelt tal fylder 5 oktetter i RAM-lageret.
"I" eller "i"	står for et heltal (integer). Hvert tal fylder 2 oktetter.
"B" eller "b"	står for *en byte* (oktet), som kan have en heltalsværdi fra 0 til 255.
"L" eller "l"	står for *en logisk* (eller Boolesk) variabel, altså en enkelt bit (som er 0 eller 1). Otte Booleske variabler fylder 1 oktet.
"S" eller "s"	efterfulgt af et tal angiver den maksimale længde af en streng (højst 32767). F.eks. be-

tyder "S20", at der bliver plads til 20 tegn i feltet. Ved beregning af lagerforbruget må man huske, at to oktetter er afsat for hvert felt til information om strengens længde.

Bemærk, at mellemrum er tilladte i strengen *beskrivelse$* for at øge læselighed.

Eksempler: **createrf(1,100,"s5s10s40")** opretter RAM-fil nummer 1 med plads til 100 poster. Hver post består af tre strenge, hhv. på 5, 10 og 40 tegn.

createrf(2,50,"I S10 BBB R LLL") opretter en RAM-fil (fil nummer 2) med plads til 50 poster. Hver post består af et helt tal, en streng med maks. 10 tegn, tre bytes, et reelt tal og tre logiske variable (bits).

Se under **eofrf** for et kort programeksempel, og se Demodisketten for et større eksempel på brugen af RAM-filer.

deleteallrf

er en procedure, som sletter alle RAM-filer i lageret. Data i en RAM-fil forbliver i RAM, selv om et program afbrydes. De slettes først, når datamaten slukkes, eller **deleteallrf** eller **deleterf** udføres.

deleterf(filnr#)

er en procedure, som sletter data i RAM-filen nummer *filnr#*. Bemærk, at data i en RAM-fil ellers ikke bliver slettet, før datamaten slukkes. Bemærk, hvis et program, hvori en RAM-fil bliver skabt, køres mere end én gang, f.eks. under programudviklingsarbejdet, skal filen slettes, før en fil med samme nummer kan oprettes. Se også **deleteallrf**.

Eksempel: **deleterf(2)** sletter RAM-fil nummer 2 fra lageret.

Se under **eofrf** for et kort programeksempel, og se C-128 Demodisketten for et større eksempel på brugen af RAM-filer.

eofrf(filnr#)

er en funktion. Når en RAM-fil skabes, oprettes der også en *pointer* eller *pegepind*, som i starten peger på det første felt i den første post. Denne pointer følger med filen, også når filen gemmes på diskette eller hentes ind igen. Pointeren opdateres til næste felt, hver gang der skrives eller læses et enkelt felt. Når sidste felt i en post læses eller skrives, skifter pointeren til første felt i næste post. Når sidste felt i sidste post læses eller skrives, sættes et flag for *slut på fil*. Dette flag kan testes med funktionen **eofrf(filnr#)** analogt med funktionerne EOF og EOD. Når flaget er sat, vil yderligere læsning eller skrivning til RAM-filen resultere i en fejludskrift "slut på ram fil".

Eksempel:
```
0010 USE ramfiles
0020 DIM text$(5) OF 10
0030 DATA "Volvo","Mercedes","Saab","Ford","Nissan"
0040
0050 nr:=0
0060 WHILE NOT EOD DO
0070   nr:+1
0080   READ text$(nr)
0090 ENDWHILE
0100
0110 deleterf(2)
0120 createrf(2,nr,"s10")
0130 nr:=1
0140 WHILE NOT eofrf(2) DO
0150   writestr(2,text$(nr))
0160   nr:+1
0170 ENDWHILE
0180
0190 posrf(2,1,1)
0200 REPEAT
0210   readstr(2,biler$)
0220   PRINT biler$
0230 UNTIL eofrf(2)
```

fieldtype$(filnr#,feltnr#)

er en strengfunktion, som returnerer beskrivelsesteksten for de enkelte felter i en RAM-fil.

Eksempel 1:

```
0010 USE ramfiles
0020 deleterf(1)
0030 createrf(1,50,"rbls10")
0040 print fieldtype$(1,1)
```

Ovenstående lille program vil skrive "r" på skærmen, svarende til at første felttype i beskrivelsesstrengen for RAM-fil nummer 1 er et reelt tal. Hele beskrivelsesstrengen kan konstrueres ved hjælp af følgende program:

Eksempel 2:

```
0010 USE ramfiles
0020 beskrivelse$:=""
0030 deleterf(1)
0040 createrf(1,50,"rbls10")
0050
0060 FOR felt:=1 TO inqrf(1,3) DO // antal felter
0070   beskrivelse$:=beskrivelse$+fieldtype$(1,felt)
0080 ENDFOR felt
0090
0100 PRINT beskrivelse$
```

freerf

er en funktion, som returnerer antal ledige bytes i det lagerområde, som RAM-filerne udnytter. Ved opstart er der 40896 bytes ledige.

Eksempel: Efter kørsel af programmet i Eksempel 2 under **fieldtype$** giver **PRINT freerf** tallet 39914 som resultat.

getnum(filnr#)

er en funktion, som anvendes til at hente et tal fra RAM-filen *filnr#*. Funktionen returnerer det tal, som filpointeren peger på, og opdaterer filpointeren til næste felt.

getrec$(filnr#,post#)

er en strengfunktion, som returnerer en streng i internt format svarende til posten *post#* i RAM-filen nummer *filnr#*. Filpointeren opdateres ikke.

getstr$(filnr#)

er en funktion, som anvendes til at hente en streng fra filen nummer *filnr#*. Filpointeren skal forinden være placeret korrekt ved hjælp af **posrf(filnr#,postnr#,feltnr#)** eller som konsekvens af sekventiel læsning eller skrivning til RAM-filen. Fejlmeddelelsen "forkert ram felt-type" kommer frem, hvis feltet ikke svarer til en streng. Filpointeren opdateres til næste felt.

inqrf(filnr#,tal#)

er en funktion, som returnerer følgende værdier for at undersøge formatet af en "ukendt" RAM-fil.

tal#:
1: antal reserverede poster
2: længden af de enkelte poster
3: antal felter i en post
4: aktuelt postnummer
5: aktuelt feltnummer

Hvis RAM-filen *filnr#* ikke er oprettet, returneres altid 0.

loadallrf(filnavn$)

er en procedure, som sørger for at hente alle de RAM-filer fra diskette, som tidligere blev lagret under *filnavn$*. Se også **saveallrf**.

Eksempel: **saveallrf("@0:alle filerne")**
loadallrf("alle filerne")

loadrf(filnr#,filnavn$)

er en procedure, som gør det muligt at hente RAM-fil nummer *filnr#*, som tidligere var lagret under *filnavn$*, fra diskette. En evt. tidligere RAM-fil med samme nummer skal slettes først. For et eksempel på en procedure, der kan bruges til at hente en RAM-fil med navnet "ramfil 1", se programmet "RAMFILES", som findes på C-128 COMAL Demodisketten.

posrf(filnr#,post#,felt#)

er en procedure, som lader brugeren placere filpointeren efter behag i et bestemt *felt#* inden for en bestemt *post#* inden for et bestemt *filnr#*, klar til RAM-fil læse- eller skriveoperationer. See også **eofrf**, **getstr$**, **writenum** og **writestr** for eksempler på brugen af **posrf**. Se også demonstrationsprogrammet "RAMFILES" på C-128 COMAL Demodisketten.

readnum(filnr#,værdi)

er en procedure, der anvendes til at hente et tal fra fil nummer *filnr#* og overføre den til variablen *værdi*. Se **posrf** for et program-eksempel, der bruger **readnum**. Filpointeren opdateres automatisk til næste felt.

readrec(filnr#,post#,streng$)

er en procedure, som anvendes til at hente postnummer *post#* fra RAM-filen *filnr#* frem og overføre den til REF-parameteren *streng$*. Bemærk, at posten i givet fald er *i internt format*. Dvs. at strenge, hele tal, bits, bytes og reelle tal forekommer i den form, som de har i en RAM-fil. Filpointeren opdateres *ikke* til næste felt.

readstr(filnr#,streng$)

er en procedure, som kan bruges til at hente en *streng$*, som tidligere var gemt ved hjælp af **writestr**. Se **posrf** for et program-

eksempel, der viser anvendelsen af **readstr**. Filpointeren opdateres automatisk til næste felt.

saveallrf(filnavn$)

er en procedure, som lagrer alle eksisterende RAM-filer på diskette under navnet *filnavn$*. Alle filerne kan hentes ind igen ved hjælp af **loadallrf**(*filnavn$*). Det er også muligt at lagre enkelte RAM-filer. Se **saverf** og **loadrf**. Bemærk, at hvis **saveallrf** er blevet brugt til lagring, kan **loadrf**(*filnr#,filnavn$*) ikke bruges til at hente en fil ind igen. Tilsvarende for **saverf** og **loadallrf**. Filformatet er forskelligt ved lagring af en enkelt eller alle filer. Hvis formatet ikke passer, gives fejlmeldingen "forkert format i load-fil".

saverf(filnr#,filnavn$)

er en procedure, som lagrer filen med nummeret *filnr#* på diskette under filnavnet *filnavn$*.

Eksempel: saverf(1,"adresser") gemmer RAM-fil nummer 1 på disketten under filnavnet "adresser".

writenum(filnr#,værdi)

er en procedure, som bruges til at lagre en talværdi i en RAM-fil. Filpointeren skal først placeres ved hjælp af **posrf** eller i kraft af sekventiel læsning/skrivning, før der skrives til filen. Efter operationen opdateres filpointeren til næste felt.

Eksempel:
```
0010 USE ramfiles
0020 deleterf(1); createrf(1,10,"ir")
0030
0040 FOR tal#:=1 TO 10 DO
0050   writenum(1,tal#)
0060   writenum(1,SQR(tal#))
0070 ENDFOR tal#
0080 posrf(1,1,1)
```

```
0090 REPEAT
0100   readnum(1,værdi)
0110   readnum(1,kvadratrod)
0120   PRINT værdi;kvadratrod
0130 UNTIL eofrf(1)
```

writerec(filnr#,post#,streng$)

er en procedure som gør, at hele poster kan skrives til en RAM-fil på én gang. Den overførte *streng$* vil være pakket i et internt format og **LEN**(*streng$*) vil være postlængden. Den modsatte operation udføres med proceduren **readrec**(*filnr#,post#,streng$*). Den tekstvariabel, som der læses til ved hjælp af **readrec**, skal være dimensioneret til mindst denne længde. Filpointeren ændres ikke af denne procedure.

writerefrec(filnr#,post#,streng$)

er en procedure, som kan bruges til at skrive en hel post ad gangen til en RAM-fil. Brugen af denne procedure i stedet for **writerec** er tids- og lagerbesparende. **LEN**(*streng$*) er postlængden. Bemærk, at det er muligt at udføre den modsatte operation ved hjælp af **readrec**(*filnr#,post#,streng$*). Strengen, som posten læses til, skal være mindst lige så lang som postlængden. Operationen ændrer ikke på filpointeren.

writerefstr(filnr#,streng$)

er en procedure, som anvendes til at lagre en *streng$* i en RAM-fil. Brugen af **writerefstr** i forhold til **writestr** er tids- og lagerbesparende. Den modsatte operation kan udføres ved hjælp af proceduren **readstr**. Filpointeren opdateres til næste felt.

writestr(filnr#,streng$)

er en procedure, som kan bruges til at lagre en *streng$* til en RAM-fil. Forinden skal **posrf** være udført således, at filpointeren

står ved den rigtige post og det rigtige felt. Efter operationen opdateres filpointeren til næste felt.

Eksempel:
```
0010 USE ramfiles
0020 deleterf(1);createrf(1,6,"s8s10")
0030
0040 DATA "Boeing","Lockheed","Douglas","Rockwell",
0045 DATA "Cessna","Piper"
0050 DATA " 747"," Electra"," DC-3"," Commander",
0055 DATA " 172"," Cherokee"
0060
0070 FOR post#:=1 to 6 DO
0080   posrf(1,post#,1)
0090   READ fabrik$
0100   writestr(1,fabrik$)
0110 ENDFOR post#
0120
0130 FOR post#:=1 TO 6 DO
0140   posrf(1,post#,2)
0150   READ fly$
0160   writestr(1,fly$)
0170 ENDFOR post#
0180
0190 posrf(1,1,1)
0200 WHILE NOT eofrf(1) DO
0210   readstr(1,fabrik$); readstr(1,fly$)
0220   PRINT fabrik$,fly$
0230 ENDWHILE
```

Følgende fejlmeddelelser er nye og vedrører pakken ramfiles:

37:	ram file already created	(ram fil allerede oprettet)
38:	ram file not created	(ram fil ikke oprettet)
39:	invalid ram file number	(ulovlig ram fil-nummer)
40:	invalid ram record number	(ulovlig ram post-nummer)
41:	error in descriptor string	(fejl i beskrivelses-tekst)
42:	wrong ram field type	(forkert ram felt-type)
43:	invalid ram field number	(ulovlig ram felt-nummer)
44:	end of ram file	(slut på ram fil)
45:	wrong file type	(forkert fil-type)

ANDRE NØGLEORD

Udover systempakken og den nye ramfiles pakke er der sket forbedringer, udvidelser eller ændringer i følgende dele af C-64 COMAL:

* COMAL-kernen
* grafikpakken
* spritepakken
* fontpakken

En del af disse ændringer skyldes, at den nye COMAL version 2.02 kan udnytte C-128'erens 80-tegnsskærm. Desuden skyldes en del ændringer, at , og ; har fået en ny betydning i COMAL efter det sidste internationale standardiseringsmøde. Også IN blev ændret lidt ved dette møde, som blev afholdt i Danmark i september, 1986.

NØGLEORD FRA COMAL-KERNEN

BASIC

er en kommando, som kan bruges til at forlade COMAL og gå over til Commodore BASIC 7. Det er muligt at komme tilbage igen med kommandoen SYS 14*4096. Udfør BANK 15 før SYS kommandoen, hvis BANK i BASIC har været ændret.

Eksempel: **BASIC** hopper fra COMAL til BASIC.
SYS 14*4096 hopper tilbage til COMAL igen.

GET$(filnr#,antal_tegn#)

er en funktion i COMAL kernen. Det er ændret således, at når GET$ anvendes på en fil med nummeret *filnr#*, der er åbnet til serielporten ("sp:") for at hente *antal_tegn#*, hentes kun det antal tegn, som der er i RS-232 modtagebufferen. Dvs. at programmer ikke vil "blive hængende", når der intet tegn er i bufferen.

IN

er et nøgleord, som har fået en lidt anden betydning i C-128 COMAL. Nu vurderes **streng1$ IN streng2$**, hvor **streng1$** er den tomme streng (""), til FALSE altså **0**. I C-64 COMAL blev dette vurderet til **LEN(streng2$)+1** altså sandt.

Eksempel:
```
INPUT "Skal vi gå videre (j,n)?":svar$
IF svar$ IN "jJ" THEN DELETE "program"
```

I C-64 COMAL vil programmet eksekvere **DELETE "program"**, selv om brugeren *ikke* svarer "j" eller "J" men blot taster **<Return>**, så **svar$** bliver den tomme streng. Dette sker ikke i C-128 COMAL, da **svar$ IN "jJ"** bliver vurderet til FALSE, når **svar$=""**. Nu *skal* brugeren taste **j** eller **J**, ellers bliver THEN-delen ikke udført.

INPUT

er et nøgleord fra COMAL kernen, som har fået udvidede muligheder med C-128 kapslen. Attributter (understregning, omvendt tekst og blink) bevares i et inputfelt. F.eks. bevares omvendt tekst, selv om feltet slettes med **<Clr/Home>**. I lighed med PRINT sætningen virker , og ; nu anderledes, når de benyttes som afslutning af en INPUT sætning.

Eksempel:
```
0010 PAGE
0020 INPUT "Er vejret OK (j/n)? ":svar$;
0030 IF svar$="j" THEN
0040   PRINT "Godt, vi tager afsted."
0050 ELSE
0060   PRINT "Så bliver vi hjemme."
0070 ENDIF
```

Semikolon betyder nu altid, at der bliver netop ét mellemrum efter brugerens svar, medmindre ZONE sættes til noget andet. Vognretur bliver ikke udført, og teksten efter brugerens svar kommer på samme linie.

To procedurer er indført i systempakken for at lette arbejdet med

indlæsning af data: **inputpos**, som sætter markøren ved en given startposition i et inputfelt, og **termchars**, som bestemmer hvilke tegn, som kan afslutte indtastning i et inputfelt. Desuden er der to nye systempakke-funktioner: **termpos**, som angiver, hvor i inputfeltet markøren var ved afslutning af en indtastning, og **termchar$**, som angiver tegnkoden for afslutningstegnet i et inputfelt. Se under hver af disse for flere informationer samt programmet "demo.termchar.80" på C-128 Demodisketten.

PRINT

er et nøgleord, som kan bruges som sætning eller kommando til at skrive data på en udskriftsenhed som en skærm eller en printer. I forbindelse med brugen af PRINT-sætningen har komma (,) og semicolon (;) fået en lidt anden betydning. Se INPUT og ZONE for en nærmere omtale herom samt programeksempler.

SIZE

er en kommando som bruges til at vise, hvordan lageret er anvendt til programmer, data og RAM-filer:

Eksempel: **SIZE**

```
prog      data      free
00065     00000     40889

ramfiles:
used      free
00982     39914
```

ZONE [værdi]

er et nøgleord, som kan bruges som sætning, kommando eller funktion. ZONE har fået en lidt anden betydning i C-128 COMAL, end den har i C-64 COMAL. Nu er semicolon (;) det skilletegn (i stedet for komma), som ZONE virker på. Standardværdien er nu **1** (tidligere **0**), således at semicolon får samme betydning som før, når ZONE ikke er sat. Når ZONE sættes til en *værdi*, bliver der dette antal

pladser til hver tabulering. Skilletegnet komma (,) virker således nu altid kun som et skilletegn og bliver ikke påvirket af ZONE.

Eksempel:
```
0010 huskzone:=ZONE // ZONE kan bruges som funktion
0020 ZONE 10 // eller som sætning.
0030
0040 FOR nr:=1 TO 5 DO
0050   PRINT nr;nr;nr
0060 ENDFOR nr
0070
0080 ZONE huskzone
```

Denne ændring er gennemført for at sikre sig et skilletegn - nemlig , - som ingenting betyder. Dette er værdifuldt bl. a. fordi komma også bruges som dataskilletegn ved ASCII-filoperationer. Tidligere kunne man risikere, at uventede mellemrum dukkede op i en fil, på grund af en sætning som **PRINT FILE a$,b$,c$**, der blev udført, mens ZONE var forskellige fra nul. I C-128 COMAL kan dette ikke ske. Ændringen er tiltrådt af COMAL Standardiseringsgruppen i september 1986.

NØGLEORD FRA GRAFIKPAKKEN

textcolor(farve#)
textborder(farve#)
textbackground(farve#)

disse procedurer fra grafikpakken, der hver har en *farve#* (0-15) som parameter, virker kun på 40-tegnsskærmen. Bemærk, at alle øvrige muligheder fra tastatur og sytempakken gælder for den aktuelle skærm. Proceduren **textcolors** fra systempakken skal bruges, når man vil ændre tekstfarven på 80-tegnsskærmen. Se sidste side af dette tillæg for en farvekodeoversigt.

NØGLEORD FRA SPRITEPAKKEN

getshape$(tegningnr#)

er en strengfunktion i spritepakken, som returnerer en streng med

indholdet af tegningen med nummeret *tegningnr#*. Den er den inverse operation af **define(***tegningnr#,tegning$***)**.

NØGLEORD FRA FONTPAKKEN

discardfont

er en procedure i pakken font. Den restorerer standardtegnsættet, også selv om der har været udført en **keepfont**. Bemærk, at i 40-tegnstilstand frigives arbejdslager, som har været anvendt til fonten, først efter NEW eller DISCARD.

selectfont(tegnsæt#)

er en procedure i fontpakken, som bevirker, at der skiftes til font nummer *tegnsæt#* på den aktuelle skærm. Lovlige værdier for *tegnsæt#* er **0** og **1** på 80-tegnsskærmen og **0**, **1**, **2** og **3** på 40-tegnsskærmen (dog ikke **0** eller **1**, hvis der ikke er et brugerdefineret tegnsæt).

Eksempel:
```
USE font
USE system
textmode(1)          //  vælger 80-tegnsskærmen.
loadfont("font1")    //  henter en font-fil fra diskette.
selectfont(1)        //  aktiverer font 1 til brug.
```

Appendiks A: PROBLEMLØSNING

PROGRAMMET KØRES, MEN INTET SES PÅ SKÆRMEN:

Efterse, at alle de nødvendige kabler er på plads. Det kan være, at programmet er beregnet til kørsel på 80-tegnsskærm, og datamaten er tilsluttet en 40-tegnsskærm eller omvendt. Hvis det er muligt, kan man skifte til den anden skærm. Bemærk også, at 40-tegnsskærmen ikke kan bruges, hvis man har valgt en clockfrekvens på 2 MHz ved hjælp af systempakke proceduren **cpuspeed(2)**. Brug i stedet **cpuspeed(0)** (som er standard ved opstart).

Hvis man kun har den ene type skærm, kan følgende procedurer benyttes til at vælge skærmtype:

USE system	
textmode(1)	80 tegnsskærm vælges
textmode(0)	40 tegnsskærm vælges

Man kan evt. anbringe én af disse i en opstartsprocedure i programmet, så man er sikker på, hvilken skærm, der bliver brugt.

INPUT ELLER OUTPUT STÅR IKKE PÆNT PÅ SKÆRMEN:

Det er muligt, at programmet er skrevet til en 80-tegns skærm og køres på 40-tegns skærm, eller omvendt. Redigér programmet eller skift skærm. Brug evt. **textmode(1)** eller **textmode(0)** for at sikre, at programmet altid kører på den rigtige skærm.

DISKETTESTATIONENS AKTIVITETSLAMPE BLINKER:

Når diskettestationens aktivitetslampe blinker, og drevet arbejder, uden at der umiddelbart sker noget, kan det være fordi, at systemet omstiller sig til en anden drevtilstand. Den gamle 1541 diskettestation kan godt bruges sammen med C-128 og den nye COMAL kapsel. På den anden side får man megen glæde af at have adgang til en 1570 (en hurtig, enkeltsidet diskettestation) eller en 1571

(hurtig, dobbeltsidet). Hvis man har C-64 disketter med COMAL programmer, som man gerne vil føre over til C-128 COMAL, kan man godt bruge de gamle disketter i den hurtige disk drive. Diskettestationen skal dog arbejde lidt (aktivitetslampen blinker) for at konstatere, at disketten var formateret på en 1541, før et CATALOG kan vises eller programfiler kan hentes. Det er muligt fra C-128 COMAL at skifte mellem 1570/1571 tilstand til 1541 tilstand eller omvendt. Til dette formål kan man bruge:

PASS "u0>m0" skifter *fra* 1570 (1571) *til* 1541 tilstand
PASS "u0>m1" skifter *fra* 1541 *til* 1570 (1571) tilstand

PROGRAMLINIER HÆNGER IKKE SAMMEN PÅ SKÆRMEN:

Når man vil redigere et program, er det jo muligt med COMAL version 2.02 at placere markøren på en programlinie og foretage ændringer. Når man så taster **<Return>**, bliver programlinien optaget i programmet. Man kan evt. taste **<Ctrl-A>** for at "trække en linie sammen", hvis den er brudt med nogle ekstra mellemrum, fordi den blev listet og fylder mere end én linie på skærmen.

Hvis et program forårsager en rulning af skærmen, kan der opstå situationer, hvor en del af en brudt programlinie, ikke opfattes som sammenhængende med resten af linien. Dette kan afhjælpes ved at flytte markøren op til første del af linien (den del, hvor linienummeret står) og trykke på **<Ctrl-A>**.

NOGLE CTRL-, ATL- ELLER ESC-KODER VIRKER IKKE:

UniComal har udnyttet nogle indbyggede funktioner i C-128'eren. Har man én af de første udgaver af computeren, var nogle af disse tastatursekvenser endnu ikke implementeret. Dette gælder f.eks. **<Alt-Inst>**, men her kan man bruge **<Esc><A>** i stedet for.

Appendiks B - Alt-, Ctrl- og Esc-koder

Deres Commodore 128 tastatur rummer ekstra muligheder for styring af redigeringsprocessen og brugen af tekstskærmene. De har måske allerede prøvet kombinationerne **<Alt-Markør op>** (rul listning nedad) og **<Alt-Markør ned>** (rul listning opad), som er en god hjælp, når man arbejder med en programudskrift på skærmen.

Nogle af følgende kombinationer kan udføres direkte fra tastaturet (**d**), nogle kan udføres fra et program (**p**), og nogle kan udføres på begge måder.

fra tastaturet:

For at udføre f.eks. **<Ctrl-R>** fra tastaturet, skal man holde **<Ctrl>** tasten nede og samtidig tryk på **<R>**. Prøv følgende:

Eksempel: **<Ctrl-R>print<Return>** teksten "print" bliver vist med omvendt skrift.

Det samme gælder **Alt**-koderne som **<Alt-I>**. Hold **<Alt>** tasten nede og tast **<I>** tasten for at indsætte en tom linie på skærmen.

fra et program:

Hvis man ønsker omvendt tekst i et program, skal man blot sørge for, at **<Ctrl-R>** svarende til **CHR$(18)** kommer med lige før den tekst, der skal vises med omvendt skrift. Omvendt tekst afsluttes med **CHR$(146)**. Et linieskift slår også omvendt skrift fra.

Eksempel:
```
0010 inv$=CHR$(18); nor$=CHR$(146)
0020 PRINT inv$+"PRØVETEKST"+nor$+" 1 2 3 4"
```

ÆNDRINGER I FORHOLD TIL C-64 COMAL

Ny kode:	Gl. kode:	Funktion:
<Ctrl-H>	<Ctrl-B>	flytter markøren et ord tilbage.
<Esc><Q>	<Ctrl-K>	sletter resten af en linie.

<Esc><K> <Ctrl-L>	flytter markøren til slutningen af linien.
<Ctrl-G> <Ctrl-X>	ændrer kantfarven.

Alt-KODER

Bemærk, at Alt-koderne kun kan udføres direkte fra tastaturet.

<Alt-markør venstre>	**d**	flytter markøren et ord tilbage.
<Alt-markør højre>	**d**	flytter markøren et ord frem.
<Alt-markør op>	**d**	ruller en listning nedad.
<Alt-markør ned>	**d**	ruller en listning opad.
<Alt-Inst>	**d**	skifter mellem indsæt/erstat måde. Indsæt måde vises med ikke-blinkende markør. (Ikke på tidlige C-128'ere.)
<Alt-Del>	**d**	fjerner tegnet under markøren.
<Alt-T>	**d**	definerer øverste venstre hjørne af vindue.
<Alt-B>	**d**	definerer nederste højre hjørne af vindue.
<Alt-F>	**d**	skifter til fuld skærm som vindue.
<Alt-I>	**d**	indsætter en linie på skærmen.
<Alt-D>	**d**	sletter en linie på skærmen.
<Alt-E>	**d**	sletter resten af skærmen.
<Alt-R>	**d**	sletter en linie fra program og skærm.
<Alt-N>	**d**	indsætter en ny tom programlinie med linienummer +1.

Ctrl-KODER

<Ctrl-A>	1	**d**	lister en programlinie. God til at rette en linie tilbage med, hvis en rettelse fortrydes.
<Ctrl-B>	2	**d,p**	starter understregning (kun) på 80 tegnsskærmen.
<Ctrl-C>	3	**d**	svarer til **<Run/Stop>**.
<Ctrl-D>	4	**d**	dumper grafikside til printer (se notat 1).
<Ctrl-E>	5	**d,p**	skifter markørfarven til hvid.
<Ctrl-F>	6	**d**	flyt markøren et ord frem.
<Ctrl-G>	7	**d**	ændrer kantfarven på 40-tegnsskærm.

<Ctrl-G>	7	**p**	ringer klokken.
<Ctrl-H>	8	**d**	flytter markøren et ord tilbage.
<Ctrl-I>	9	**d,p**	tabulerer.
<Ctrl-J>	10	**d,p**	flytter markør til ny linie (linefeed).
<Ctrl-K>	11	**d,p**	sætter lock out <C=><Shift> på 40-tegnsskærm.
<Ctrl-L>	12	**d,p**	sætter unlock <C=><Shift> på 40-tegnsskærm.
<Ctrl-M>	13	**d,p**	genererer vognretur (carriage return) og ny linie.
<Ctrl-N>	14	**d,p**	sætter små/store bogstaver (på 40-tegnsskærm).
<Ctrl-O>	15	**d,p**	sætter blinkende tegn (kun 80-tegnsskærm).
<Ctrl-P>	16	**d**	sender kopi af aktuel tekstskærm til printer (se notat 2).
<Ctrl-Q>	17	**d,p**	flytter markøren nedad én linie.
<Ctrl-R>	18	**d,p**	sætter omvendt tekst på den aktuelle skærm.
<Ctrl-S>	19	**p**	flytter markøren til linie 1, søjle 1 <Home>.
<Ctrl-T>	20	**d,p**	sletter et tegn <Del>.
<Ctrl-U>	21	**d**	sætter grafikfunktionstasterne ud af drift.
<Ctrl-V>	22	**d**	udfører **textcolors(6,6,1)** - blå skærm med hvid tekst på aktuel skærm.
<Ctrl-W>	23	**d**	udfører **textcolors(11,15,0)** på den aktuelle skærm.
<Ctrl-X>	24	**d,p**	tænder/slukker tabulatorstop.
<Ctrl-Y>	25	**d**	ændrer baggrundsfarve.
<Ctrl-Z>	26	**d,p**	ændrer standardfarver.
<Ctrl-[>	27	**d,p**	<Esc>.
<Ctrl-]>	29	**d,p**	flyt markør til højre.

notat 1: *<Ctrl-D> gælder kun Commodore MPS-801 kompatible printere, hvor grafikdump er mulig.*

notat 2: *Hvis <Ctrl-P> giver dump af den aktive tekstskærm uden korrekt vognretur, prøv da* **USE system** *og proceduren* **setprinter("lp:/l+")**.

Esc KODER

Alle C-128 Escape-koder kan anvendes under redigering undtagen <Esc><L> og <Esc><M>, som slå skærmrulning til og fra, samt <Esc><E> og <Esc><F>, som slår markør blink til og fra. Bemærk, at **Esc**-koderne - i modsætning til **Ctrl**-koderne - tastes som en *sekvens* af tastninger. F.eks. for at skifte fra 40-tegns- til 80-tegnsskærmen eller tilbage igen tastes <Esc> og bagefter <X>.

<Esc><O>	ophæver anførsels- og indsæt-tilstand.
<Esc><Q>	sletter til slutningen af aktuel linie.
<Esc><P>	sletter til start af den aktuelle linie.
<Esc><@>	sletter til slutningen af skærmen.
<Esc><J>	flytter markør til start af aktuel linie.
<Esc><K>	flytter markør til slutning af aktuel linie.
<Esc><A>	starter auto-indsæt funktion.
<Esc><C>	slutter auto-indsæt funktion.
<Esc><D>	sletter den aktuelle linie.
<Esc><I>	indsætter en linie.
<Esc><Y>	sætter standard tabulatorstop (8 mellemrum).
<Esc><Z>	sletter alle tabulatorstop.
<Esc><V>	ruller skærmbilledet opad.
<Esc><W>	ruller skærmbilledet nedad.
<Esc><G>	aktiverer klokken (se <Ctrl-G>).
<Esc><H>	slår klokken fra så den ikke reagerer på <Ctrl-G>.
<Esc><E>	undertrykker markørens blink.
<Esc><F>	sætter markøren i gang med at blinke.
<Esc><B>	sætter skærmvinduebund til markørpositionen.
<Esc><T>	sætter skærmvinduetop til markørpositionen.
<Esc><X>	skifter mellem 40- og 80-tegnsskærm.

Følgende gælder kun 80-tegnsskærmen:

<Esc><U>	skifter til understregningsmarkør.
<Esc><S>	skifter til blokmarkør.
<Esc><R>	sætter skærmen til negativ udskrift.
<Esc><N>	sætter skærmen tilbage til normal skrift.

Alle Esc-koder kan bruges under programudførsel.

Appendiks C: MASKINEKODE VEJLEDNING

Dette er et tillæg til **kapitel 8** i **COMAL for Commodore 64**, som handler om, hvordan man knytter et maskinekodeprogram til COMAL version 2.02 for Commodore 128. Dette tillæg beskriver kun forskellene mellem C-64 og C-128.

OPDELING AF RAM:

RAM bank 0:

0-8 KB rummer systemvariabler for kernen, COMAL, processorstakken, og skærmhukommelse (1-2 KB er 40-tegnsskærmhukommelsen).
8-48 KB rummer lager for COMAL-program, navntabel samt stak. Her kan også anbringes pakker, hvorved man tager af brugerlageret. Hvis tegnsæt anvendes, anbringes de for 40-tegnsskærmen fra 43-48 KB.
48-64 KB kan anvendes til maskinsprogspakker, som ikke tager af brugerlageret (området fra $FF00 til $FFFF må dog ikke anvendes).

RAM bank 1:

0-8 KB er fælles RAM (*common RAM area*), dvs. at RAM 0 også adresseres i dette område.
8-48 KB anvendes til RAM-filer.
48-64 KB bruges til grafik samt diverse tabeller (filmapping, mm).

Input/Output- (I/O-) området ligger som i C-64 fra **52-56 KB**.

FØLGENDE ROM'ER FINDES I COMMODORE 128:

16-48 KB	BASIC-fortolker
48-52 KB	EDITOR (40/80-tegnsskærme)
52-56 KB	Standard dobbelttegnsæt
56-64 KB	KERNEL

COMAL-kapslen er opdelt i 6 sider af hver 16 KB, som alle befinder sig i adresseområdet 48-64 KB. Ved hjælp af bank-omskiftning opnås, at et 96 KB system kun optager 16 KB i Commodore 128. Bemærk, at COMAL-kapslen ligger i samme adresseområde som EDITOR og KERNEL.

LAGERSTYRING:

Styring af C-128'erens hukommelsesområde foregår ved hjælp af en speciel MMU (*memory management unit*) kreds. Styring af kapselbanks foregår på samme måde som i C-64, dog er porten (OVERLAY) ændret, så den ikke mere er en "write-only" port. Nu kan indholdet også læses.

OPBYGNING AF MODULER:

Modulerne er opbygget på helt samme måde som i C-64 kapslen, dog ændres **.lib c64symb** til **.lib c128symb** (**.lib c128symb** svarer til **.lib c128symb1** + **.lib c128symb2** uden kommentarer, mm.). Desuden vil <startadresse> typisk være $C000 i stedet for $8009. Nu angiver **.byte <map>** hukommelsesområder, og betydningen af forskellige mapværdier fremgår af følgende tabel. Samme hukommelsesområder gælder også for SETPAGE i systempakken:

mapværdi	*lagerområde (memory map)*
1	RAM 0
2	RAM 0 + I/O
3	RAM 1
4	RAM 1 + I/O
5	COMAL-kapsel + RAM 0
6	COMAL-kapsel + RAM 0 + I/O
7	COMAL-kapsel + RAM 1
8	COMAL-kapsel + RAM 1 + I/O
9	COMAL-kapsel + RAM 2
10	COMAL-kapsel + RAM 2 + I/O

mapværdi	*lagerområde (memory map)*
11	COMAL-kapsel + RAM 3
12	COMAL-kapsel + RAM 3 + I/O
13	KERNEL + RAM 0 + I/O
14	KERNEL + RAM 0 + CHARROM
15	KERNEL + RAM 0 + I/O + BASIC + MONITOR

Af disse hukommelsesområder kan alle anvendes som parameter til SETPAGE, men kun de områder, som angiver RAM 0, kan anvendes som <MAP> i moduler. Når DEFPAG er lig med 2, dvs. RAM 0 + I/O, skal man være opmærksom på, at KERNEL ikke er aktiv i dette område. De mest anvendte rutiner kan dog kaldes via indirekte kald, som er defineret i C128SYMB. Når standardområdet (2) anvendes, kan maskinkode ikke placeres i området $D000-$DFFF (I/O området). Hvis adgang til I/O porte ikke er nødvendig, kan map 1 anvendes uden begrænsninger i dette område.

Før COMAL-rutiner som f.eks. LOAD og STORE kaldes, skal rutinen SPAGEX kaldes (A registret skal være lig med det ønskede område). Rutinen sætter variablerne PAGEX og MAPX, som angiver hukommelsesområdet.

HVOR MODULER KAN ANBRINGES:

Moduler kan anbringes i RAM 0 fra $2100-$FEFF. Hvis start-adressen er mindre end $C000, begrænses dog brugerlageområdet. Området $AC00-$BFFF må dog ikke anvendes, hvis FONT-pakken anvendes til 40-tegnsskærmen.

HVOR MODULETS VARIABLER KAN ANBRINGES:

Variabler, som skal overleve fra kald til kald, kan anbringes i selve modulet eller i området $1200 til $12FF. Desuden kan tape- og RS-232-buffere bruges, forudsat at de ikke bruges i forvejen. I zero-page er kun lagercellen $FF ledig. (Adresserne $4C og $56 er fejlagtigt angivet som frie i C-64.) Adresserne på zero-page variabler

(INF1...TXTHI) er uændrede, adressen på COPY2 og COPY3 er de samme i C-64 og C-128: COPY2 $0047-$0048, COPY3 $0049-$004A. RANGES er flyttet til $1768-$1787 og TXT til $09A1-$09F0.

PAKKEEKSEMPEL:

Pakkeeksemplet på C-128 Demodisketten er ændret som beskrevet. Det vil sige:

```
    .lib C64symb
    .opt list           ;list dette modul
;
    *=$8009             ;startadresse
```

er ændret til:

```
    .lib C128symb
    .opt list           ;list dette modul
;
    *=$C000             ;startadresse
```

Som det ses af ovenstående, vil det i de fleste tilfælde være let at tilpasse en C-64 pakke, så den også kan køre på en C-128.

LITTERATUR:

For flere oplysninger om COMAL pakker, se den udmærkede bog:

Jesse Knight, *COMAL 2.0 Packages*, udgivet af COMAL Users Group, 5501 Groveland Terrade, Madison, WI 53716-3251 USA.

Bogen kan fås i Danmark ved henvendelse til:

COMAL TODAY DK
Postboks 122
DK-2300 København S

STIKORDSREGISTER

EGNE NOTATER

FARVEKODER

farve-kode	*farve på 40-tegnsskærm*	*farve på 80-tegnsskærm*
0	sort	sort
1	hvid	hvid
2	rød	mørkerød
3	cyan	lys cyan
4	violet	lys violet
5	grøn	mørkegrøn
6	blå	mørkeblå
7	gul	lysegul
8	orange	violet
9	brun	mørkegul
10	lyserød	rød
11	mørkegrå	cyan
12	grå	mørkegrå
13	lysegrøn	lysegøn
14	lyseblå	lyseblå
15	lysegrå	lysegrå