C O M P A S

Common Pascal Language System

Version 2

T I L L Æ G T I L B R U G E R M A N U A L

Poly-Data microcenter ApS
Strandboulevarden 63
2100 København Ø

Dette addendum beskriver de ændringer og udvidelser der er foretaget i COMPAS V2.2 i forhold til COMPAS V2.1. I teksten anvendes forkortelsen BM for Brugermanualen og PM for Programmeringsmanualen.

Tillæg til Brugermanualen

Det Tillæg til Brugermanualen, der refereres til i Brugermanualens kapitel 3 (beskrivelsen af editoren), findes ikke i COMPAS V2.2. I stedet anvendes den nye HELP kommando eller den nye <HELP> editorfunktion til at vise en tabel, der oversætter fra symbolske editorfunktioner til nøgler på tastaturet. Se mere herom nedenfor.

On-line hjælp (BM)

For at gøre betjeningen af COMPAS endnu nemmere, er der i COMPAS V2.2 tilføjet en HELP (hjælp) kommando og en <HELP> editorfunktion. Begge funktioner viser hjælpetekster, der læses fra filen COMPAS.HLP.

Mulige HELP kommandolinier er:

```
HELP
HELP <kommando>
HELP *
```

hvor <kommando> er en af de kommandoer, der findes i kommandoniveauet. Bemærk, at HELP kommandoordet kan forkortes til blot H. Ligeledes kan <kommmando> forkortes til det første bogstav i det relevante kommandoord. Når <kommando> angives, udskriver COMPAS en detaljeret beskrivelse af denne kommando. Hvis HELP anvendes uden parameter, udskrives en oversigt over de tilgængelige kommandoer, og hvis stjerne efterfølger HELP kommandoen, udskrives en oversigt over editorens kommandoer.

I editoren kan man på ethvert tidspunkt anvende <HELP> funktionen til at få udskrivet en kommandooversigt (svarende til den der vises af 'HELP *' kommandoen). Brug HELP kommandoen uden argument til at finde ud af, hvilken nøgle der aktiverer <HELP> funktionen.

Hjælpekommandoerne virker kun, hvis COMPAS.HLP filen ligger på den diskette, der var auto-disketten, da COMPAS blev startet. Hvis COMPAS.HLP ikke findes, udskriver hjælpekommandoerne:

```
No COMPAS.HLP file on disk
```

Hvis man angiver en ukendt kommando ved HELP, udskrives:

```
No such help screen
```

Udvidelser af konfigurationstabellen (BM, Appendix A)

Med den nye <HELP> kommando er der nu 24 indgange i keyboard konfigurationstabellen. Strengen, der gælder for <HELP>, er placeret mellem strengene for <AUTO> og <LEFT>.

Efter de 24 funktionsstregne kommer der nu yderligere 24 strenge (i nøjagtig det samme format), der angiver de tekster, der står skrevet på tasterne, for eksempel RETURN, BACKSP, ESC, F1, eller nogle andre forklarende tekster, for eksempel CTRL/A for CONTROL og "A" på en gang. Når man bruger HELP kommandoen eller <HELP> funktionen, anvendes disse strenge i stedet for symbolske funktioner til at beskrive editorens funktioner.

Bemærk, at ingen af strengene i keyboard konfigurationstabellen må være længere end 8 tegn.

For at skabe plads til den ovenfor beskrevne udvidelse, er der nu afsat 256 bytes til keyboard konfigurationstabellen (offset $09 til $108). Insert line, delete line, reverse on, og reverse off sekvenserne er flyttet tilsvarende (læg $80 til det oprindelige offset).

Overlay procedurer og funktioner (PM, Kapitel 15)

I COMPAS V2.2 er der mulighed for at adskille grupper af procedurer og funktioner fra hovedprogrammet. Når man udvikler store programmer er dette en stor fordel, idet et sådant program fylder mindre i lageret når det udføres, men dog stadig er et enkelt program, og ikke flere kædede (chainede) programmer.

Alle procedurer og funktioner, der erklæres med det reserverede ord OVERLAY, bliver under oversættelsen adskilt fra hovedprogrammet og placeret i en eller flere separate overlay-filer. Under kørsel af programmet bliver overlay-underprogrammerne kun indlæst fra disketten, når de kaldes. Da flere overlay-underprogrammer kan dele det samme lagerområde i hovedprogrammet, dvs. indlæses og udføres på samme adresse i lageret, kan man nu have adskillige procedurer og/eller funktioner i et program, der kun optager plads for en (nemlig den største).

Det er her vigtigt at lægge mærke til forskellen mellem overlay-underprogrammer og kædede (chainede) programmer: Overlay-underprogrammer oversættes sammen med hovedprogrammet, og udgør således et hele, mens kædede programmer oversættes separat.

I lighed med chaining kan overlay-underprogrammer ikke anvendes i programmer, der startes med RUN kommandoen. Overlay-underprogrammer fungerer kun i programfiler, altså programmer, der er oversat med PROGRAM eller OBJECT kommandoerne, og startes fra CP/M.

Erklæring af overlay-underprogrammer foregår på samme måde som erklæring af almindelige underprogrammer, bortset fra, at erklæringen startes med det reserverede ord OVERLAY. Eksempler:

```
OVERLAY PROCEDURE initialize;
VAR
  i: integer;
BEGIN
  FOR i:=1 TO 10 DO data(.i.):=0;
  count:=0;
END;
```

```
OVERLAY FUNCTION average(d: datalist): real;
VAR
  i: integer;
  a: real;
BEGIN
  a:=0;
  FOR i:=1 TO 25 DO a:=a+d(.i.);
  average:=a/25;
END;
```

Når et program oversættes, bliver overlay-underprogrammer, der står umiddelbart efter hinanden, samlet i en enkelt overlay-fil. I hovedprogrammets maskinkode bliver der derefter afsat et overlay-område, der er stort nok til det største af overlay-underprogrammerne. Et eksempel:

```
PROGRAM overlaydemo_1;

OVERLAY PROCEDURE p1;
BEGIN writeln('procedure 1'); END;

OVERLAY PROCEDURE p2;
BEGIN writeln('procedure 2'); END;

OVERLAY PROCEDURE p3;
BEGIN writeln('procedure 3'); END;

BEGIN p1; p2; p3; END.
```

Antag, at det ovenfor viste program oversættes med PROGRAM kommandoen og gives navnet OVDEMO. Compileren producerer da to filer, nemlig OVDEMO.COM og OVDEMO.000. OVDEMO.COM indeholder hovedprogrammet, og dermed også det område hvori overlay-underprogrammerne indlæses. OVDEMO.000 indeholder maskinkoden for procedurerne p1, p2 og p3. Når OVDEMO.COM udføres, bliver p1, p2 og p3 på skift læst ind i lageret fra OVDEMO.000 og udført.

Ved et kald af et overlay-underprogram undersøger systemet først, om underprogrammet allerede er tilstede i overlay-området. I så fald udføres underprogrammet med det samme, uden læsning fra disketten. Hvis hovedprogrammet i eksemplet ovenfor havde udført fem kald til proceduren p1, uden at kalde p2 eller p3 ind i mellem, ville p1 altså kun blive indlæst ved det første kald.

Et program er ikke begrænset til kun at have en overlay-fil. Faktisk kan der være op til 100 overlay-filer (nummereret fra 000 til 099) knyttet til et enkelt program. Hver overlay-fil har da et bestemt overlay-område i hovedprogrammets kode, hvori underprogrammerne indlæses. På denne måde kan flere overlay-underprogrammer være tilstede i lageret på samme tid. Som nævnt ovenfor bliver overlay-underprogrammer, der står umiddelbart efter hinanden, samlet i en enkelt overlay-fil. Hvis der derimod foretages andre erklæringer mellem overlay-underprogrammerne, produceres der flere filer, og dermed flere overlay-områder. Et eksempel:

```
PROGRAM overlaydemo_2;

OVERLAY PROCEDURE p1;
BEGIN END;

OVERLAY PROCEDURE p2;
BEGIN END;

PROCEDURE p3;
BEGIN END;

OVERLAY PROCEDURE p4;
BEGIN END;

OVERLAY PROCEDURE p5;
BEGIN END;

BEGIN END.
```

Hvis det ovenfor viste program oversættes med navnet OVDEMO, producerer compileren tre filer, nemlig OVDEMO.COM, OVDEMO.000 og OVDEMO.001. OVDEMO.000 indeholder koden for p1 og p2, OVDEMO.001 indeholder koden for p4 og p5, og OVDEMO.COM indeholder koden for p3 og hovedprogrammet, samt to overlay-områder. Under kørsel af programmet kan p1 eller p2 være tilstede i lageret samtidig med p4 eller p5. Bemærk, at den erklæring (eller de erklæringer), der adskiller de to overlay-grupper, ikke nødvendigvis skal være et underprogram; det kunne lige så godt være en LABEL, CONST eller VAR erklæring, men en kommentar er ikke nok, da denne helt udelukkes under compileringen.

Som det er tilfældet med almindelige underprogrammer, kan overlay-underprogrammer også nestes (overlays inden i overlays). Et eksempel:

```
PROGRAM overlaydemo_3;

OVERLAY PROCEDURE p1;
BEGIN END;

OVERLAY PROCEDURE p2;

OVERLAY PROCEDURE p21;
BEGIN END;

OVERLAY PROCEDURE p22;
BEGIN END;

BEGIN p21; p22; END;

OVERLAY PROCEDURE p3;
BEGIN END;

BEGIN p1; p2; p3; END.
```

Compileren vil da producere to overlay-filer. 000-filen indeholder koden for p1, p2 og p3, og 001-filen indeholder koden for p21 og p22. I lighed med hovedprogrammet, er der i proceduren p2 afsat et overlay-område, hvori p21 og p22 kan indlæses.

Når compileren oversætter et program med overlays, placerer den overlay-filerne på samme diskette som hovedprogrammet.

Under kørsel af et program med overlays antager systemet, at overlay-filerne ligger på auto-disketten (den diskette der vælges, når intet angives). Dette kan imidlertid ændres ved hjælp af Y compilerdirektivet. Syntaksen for dette direktiv er et Y umiddelbart efterfulgt af et bogstav mellem A og P, eller et 0 (nul), for eksempel (*$YA*), (*$YE*) og (*$Y0*). Et bogstav angiver en bestemt diskettestation, og et nul angiver auto-disketten. Direktivet skal anvendes før det første underprogram i en overlay-gruppe, og gælder for alle efterfølgende overlay-grupper, indtil et nyt direktiv mødes.

Når en variabel eller en parameter erklæres, bliver denne tildelt en plads i lageret, der aldrig benyttes af andre variable. Da overlay-underprogrammer i samme overlay-fil normalt ikke kalder hinanden, er dette spild af plads, idet underprogrammerne lige så godt kunne dele det samme data-areal. Ved hjælp af et O (bogstavet O) compilerdirektiv er det derfor muligt at instruere compileren om, at underprogrammerne i en overlay-gruppe skal dele det samme data-areal. Ved start af compileren vælges (*$O-*), og i denne stilling deler overlay-underprogrammerne ikke det samme data-areal. I (*$O+*) stillingen er data-arealet derimod fælles. Direktivet skal anvendes før det første underprogram i en overlay-gruppe, og gælder for alle efterfølgende overlay-grupper, indtil et nyt direktiv mødes.

Når et program efter opstart udfører det første kald til et af overlay-underprogrammerne i en overlay-fil, bliver filen åbnet af systemet, og den forbliver herefter åben sålænge programmet kører. I tilfælde af, at en diskette, hvorpå der ligger åbnede overlay-filer, skal udskiftes mens programmet kører, er det derfor nødvendigt at gennemtvinge en gen-åbning af overlay-filerne. Dette gøres ved at udføre de følgende sætninger:

```
mem(.addr(p)+36.):=0; mem(.addr(p)+37.):=0;
```

hvor p er identifieren for en af overlay-underprogrammerne i den relevante overlay-fil. I forbindelse med en gen-åbning af en overlay-fil er det også muligt at flytte til et andet diskettedrev. Dette gøres ved at udføre sætningen:

```
mem(.addr(p)+3.):=drive;
```

hvor drive er 0 (nul) for auto-disketten, eller mellem 1 og 16 for diskettedrev A til P.

Ved anvendelsen af overlay-underprogrammer opnås store besparelser først, når underprogrammerne er relativt store og kaldes relativt sjældent. Desuden gælder der, at så mange underprogrammer som muligt bør samles i hver overlay-fil.

Programafbrydning under indlæsning (PM)

Under indlæsning fra skærmterminalen (CON: enheden) er det i COMPAS V2.2 muligt at afbryde programmet ved at trykke CTRL/C. Herved fremkommer meddelelsen:

```
USER INTERRUPT AT PC=aaaa
Program terminated
```

hvorefter der returneres til COMPAS ('>>' klartegnet) eller til CP/M. I lighed med kørselsfejl og I/O fejl kan det sted, hvor programmet blev afbrudt, findes med FIND kommandoen.

I visse tilfælde er det ikke ønskeligt, at brugeren kan afbryde programmet som beskrevet ovenfor. Denne facilitet kan derfor slås fra ved hjælp af C compilerflaget. Ved start af compileren vælges (*$C+*), og i denne stilling kan programmet afbrydes under indlæsninger. Hvis et (*$C-*) direktiv placeres i begyndelsen af programmet (før erklæringsdelen), er det ikke muligt at afbryde under indlæsning.

Programafbrydning under kørsel (PM)

Hvis et fejlbehæftet program går i en uendelig løkke, vil den eneste måde at afbryde det på normalt være et tryk på systemets RESET-knap, hvilket i værste tilfælde kan betyde, at kildeteksten går tabt. I COMPAS V2.2 er det imidlertid muligt at instruere compileren om, at den programkode der genereres lejlighedsvis skal checke tastaturet for at se, om brugeren ønsker at afbryde programmet. Dette gøres ved hjælp af U compilerdirektivet. Ved start af compileren vælges (*$U-*), og i denne stilling vil det færdige program ikke kunne afbrydes. I den modsatte stilling, (*$U+*), genererer compileren et kald til et check-rutine før hver programsætning. I modsætning til C direktivet kan U direktivet frit anvendes gennem kildeteksten, og alle sætninger der oversættes i (*$U+*) stillingen vil kunne afbrydes under kørslen ved at taste CTRL/C.

Bemærk, at sætninger, der er oversat i (*$U+*) stillingen, udføres mærkbart langsommere end sætninger, der ikke kan afbrydes.

Når en sætning afbrydes, udskriver systemet:

```
USER INTERRUPT AT PC=aaaa
Program terminated
```

hvorefter der returneres til kommandoniveauet (eller til CP/M). Den sætning, der blev afbrudt, kan derefter findes med FIND kommandoen.

Compilerdirektiver og include-filer (PM, Kapitel 17)

Når compileren møder et include-fil direktiv, gemmer den status af alle compilerflag (A, I, O, R, S, U og V) og registre (W og Y) og efter endt læsning af include-filen, tilbagestilles flagene og registrene. Det er således muligt for en include-fil at definere en lokal stilling af flag og registre, uden at berøre den stilling, der gælder for "hovedprogrammet".

Ny I/O fejl (PM, Appendix F)

12 Ulovlig brug af assign. Returneres hvis programmet kalder assign standardproceduren med en af de prædefinerede filer (input, output, con, trm, kbd, lst, aux eller usr).

Nye Compilerfejl (PM, Appendix H)

96 Overlays tillades kun i programfiler. Det er ikke muligt at anvende overlays i programmer der oversættes med COMPILE eller RUN kommandoerne.

97 Overlays kan ikke forward-erklæres. Det er ikke muligt at anvende FORWARD specifikationen i forbindelse med overlay-underprogrammer.

98 PROCEDURE eller FUNCTION forventet.

99 Kan ikke åbne overlay-fil. Compileren kan ikke åbne en ny overlay-fil, enten fordi diskettens directory er fuldt, eller fordi disketten er skrivebeskyttet.