Arbeidsnotater

STATISTISK SENTRALBYRÅ

Dronningensgt. 16, Oslo-Dep., Oslo 1. Tlf. 41 38 20

IO 75/10 3. mars 1975

VERSJONSGENERERINGSSYSTEMET

DATSY-notat nr. 2

av

Ola Jacobsen

INNHOLD

1. INNLEDNING

DATSY (Data Treatment System) er et brukerorientert dataspråk utviklet av Norsk Regnesentral på kontrakt med Finansdepartementet. Byrået tok i sin tid initiativet til dette prosjektet for å få utviklet et hensiktsmessig programmeringssystem for større modellprosjekter og har samarbeidet med Norsk Regnesentral om utforming av systemet.

Mens Norsk Regnesentral i første rekke har utviklet den overordnede del av systemet har Byrået ansvaret for de fleste direktivene, dvs. de brukerorienterte operasjoner som systemet kan utføre. DATSY er beskrevet i en rekke rapporter fra Norsk Regnesentral, hvorav de viktigste er [1] og [2]. Byrået har utarbeidet en mer brukerorientert håndbok for bruk av DATSY, se [3].

DATSY er nå i produktiv bruk som implementeringsspråk for MODIS IV (se [4]) og har også vært anvendt for visse andre formål.

For å forenkle og gjøre innføringen av nye direktiver i DATSY pålitelig, er det laget en versjonsgenerator, dvs. et system som genererer en ny versjon av DATSY med spesifiserte direktiver. Versjonsgeneratoren ble først implementert på UNIVAC 1108 (se [5]). Ved konvertering av versjonsgeneratoren til Statens Driftssentrals datamaskin HONEYWELL BULL H6060 var det nødvendig å foreta enkelte forandringer og utvidelser fra det som er beskrevet i [5]. Disse forandringene er beskrevet i et eget brukernotat som heter "Vedlikehold av versjonsgenereringssystemet" [8].

Dette arbeidsnotatet er ment som en manual for bruk av versjonsgeneratoren på H6060. Arbeidsnotatet forutsetter at [1] er kjent, og ord og begreper fra denne rapporten brukes uten å bli nærmere definert. Andre spesielle ord og begreper understrekes første gang de forekommer i teksten. Notatet forutsetter også en viss kjennskap til styrekort på H6060.

Enkelte deler av stoffet i dette arbeidsnotatet er hentet fra [5].

Referanse til kapitler og avsnitt i arbeidsnotatet er gitt på formen (kap. avsnitt).

Forfatteren vil takke konsulent David Walker ved Byrået for hjelp under implementeringen av systemet.

2. BEGREPER I DATSY

Vi vil først ta for oss en del begreper som blir brukt senere i notatet, men som ikke er definert i [1]. Dette er begreper som er definert ved implementeringen av DATSY og ved implementering av versjonsgeneratoren. De begreper som vi skal ta for oss i dette kapittel er beskrevet i [2], [5] og [6].

2.1. Arbeidsrutinen

En arbeidsrutine er skrevet i Fortran og utfører en bestemt operasjon som f.eks. matrisemultiplikasjon, ekstrahering eller sortering. Det er tilgjengelig en rekke hjelperutiner for bruk ved skriving av arbeidsrutinene (se [7]).

2.2. Direktivrutinen

Direktivrutinen til et direktiv kan kalle på arbeidsrutiner og andre direktiv. Direktiv og arbeidsrutiner kalles indirekte via hjelperutinene DID og SID henholdsvis. DID står for direktiv i direktivrutine, SID står for subrutine i direktivrutine.

Det følgende eksempel illustrerer hvordan et direktiv kan være bygget opp av andre direktiver.

Eksempel

```
      SUBROUTINE D202
C     Direktivrutine til direktivet SORTER.
C     Skrevet av Ola Jacobsen.
C     Dette direktivet sorterer en file med vilkårlig mange rekords og sorteringsfelter. Det kaller
C     først på direktivet CSORT som leser inn maksimalt antall rekords i core fra file REK1, sorterer
C     disse innbyrdes og skriver dem ut på en annen file D1. Dermed har man fått en file med bunke-
C     vis sorterte rekords. To og to bunker merges sammen og skrives ut på en ny file D3. Denne
C     prosessen fortsetter fram og tilbake mellom D1 og D3 til alt er sortert. Den sorterte filen
C     vil ligge i file REK2.
      COMMON/TXP/REK1,SFELT,REK2/TXN/D1,D3,IARRM
      COMMON/OLA/IBUNKE,IANT,IARRN(24)
      CALL DID ('CSORT',REK1,SFELT,D1,IARRM)
      J=IDIMA(REK1)
      IF(IBUNKE*2.GE.J)GOTO2
   1  CALL DID ('MERGE',D1,D1,D2,IARRM)
C     Tester om vi er ferdige.
      IF(IBUNKE*2.GE.J)GOTO3
      CALL DID ('MERGE',D3,D3,D1,IARRM)
      IF(IBUNKE*2.LT.J)GOTO1
   2  CALL DID ('MERGE',D1,D1,REK1, IARRM)
      RETURN
   3  CALL DID ('MERGE',D3,D3,REK2,IARRM)
      END
```

Common blokken TXP brukes til å gi mnemoniske navn til objekter som er nevnt i direktivsetningen, mens common blokken TXN gir mnemoniske navn til objekter som brukes internt i direktivet. Direktivet SORTER kaller på direktivene CSORT og MERGE og er altså et sammensatt direktiv.

2.3. Testrutinen

Testrutinen til et direktiv kan inneholde en vilkårlig mengde tester som er aktuelle for direktivet og den kan gi verdier til parametre som beskriver objekter som er output fra direktivet, f.eks. objekttype og lengde (2.6).

Det er tilgjengelig en rekke hjelperutiner for bruk ved skrivning av testrutine (kan også brukes fra direktivrutinene) til henting og redefinering av objektenes parametre (2.6). Testrutinene kan derfor enkelt utføre ikke-standard kontroller, innvirke på allokeringen i core m.m.

2.4. Versjon

En versjon i DATSY består av en fast kjerne av systemrutiner samt ett eller flere direktiver som er koplet til systemrutinene og et sett arkivtaper som er beskrevet i arkivdeskripsjonsfilen.

2.5. Objekter

Et objekt kan være av tre typer.

Reelle objekter er de objekter som er med i kallet på direktivet. F.eks. i kallet på direktivet "MATMULT A*B=C" er A, B og C reelle objekter.

Dummy objekter er objekter som kun eksisterer inne i direktivet. Deres vesentligste anvendelseområde er:

- å reservere kladdeplass i core
- å være kopier av reelle objekter.

Interne objekter er objekter som forekommer i sammensatte direktiv. De oppstår ved at direktivrutinen for direktivet kaller på et annet direktiv som har som ut-objekt, et objekt, som ikke hører til de objekt som finnes i direktivsetningen til det første direktiv.

Hvert objekt tilhører en objektklasse som angir om et objekt er arkivtape, heltallsmatrise, rekordsett, navn etc. (jfr. Appendix A).

Det er også tillatt for et objekt å tilhøre en alternativ klasse som er et utvalg av objektklasser (jfr. Appendix A).

2.6. Direktivinformasjon

For hvert direktiv er det nødvendig å gi verdier til en del parametre. Disse beskriver direktivet og benyttes av DATSY systemet under eksekvering av direktivprogrammet.

I dette notatet vil vi betegne disse parametre for direktivinformasjonen.

Følgende parametre inngår i direktivinformasjonen:

Direktivnavn	Navnet på direktivet slik det forekommer i et direktivprogram (inntil 12 karakterer).
Direktivrutinenavn	Navnet på direktivrutinen (inntil 6 karakterer).
Testrutinenavn	Navnet på testrutinen (inntil 6 karakterer).
NPARA	Antall reelle objekt. Dvs. antall objektnavn som skal stå etter direktivnavnet i et direktivprogram.
IPARA	Totalt antall objekter = Antall reelle objekter (NPARA) + dummy objekter.
IVEKT	Direktivvekt. Denne angir i promille hvor stor del av ledig core plass som forutsettes allokert til dette direktiv. Det er igjen (1000-IVEKT) til innenforliggende direktiver. Hvis direktivet ikke er sammensatt skal IVEKT=1000.

For alle reelle og dummy objekt til et direktiv er det fire parametre som formelt beskriver hvert av disse.

KLASP	Objektklasse (2.5)
MINLP	Lengdekode som angir minimum plass et objekt må få allokert i core.
IPVEKT	Objektvekt. Angir forholdsmessig hvor mye objektet skal tildeles av ledig plass i core etter at det som er spesifisert ved MINLP er tildelt.
IUTYP	Objekttype. Angir om objektet er - Dummy objekt, IUTYP=0 - Innobjekt , IUTYP=1 - Utobjekt , IUTYP=2 - Innutobjekt , IUTYP=3

2.7. TABELL-rutinen

Ved eksekvering av et direktivprogram i DATSY, ligger direktivinformasjonen, for hvert direktiv som er med i versjonen, lagret i fem arrayer i DATSY-systemrutinen TABELL.

Denne rutinen blir laget av versjonsgeneratoren ved generering av ny versjon (se [5] - 2.3.6).

2.8. TEST-rutinen

Rutinen TEST er en Fortran rutine som inneholder en lang switch som organiserer kall til riktig testrutine for et direktiv.

TEST har følgende form:

```
      SUBROUTINE TEST
      INCLUDE COMMON
      IF(THT) CALL SKRIV(IDIRNR, 'GÅR INN I TEST.')
      GOTO (1,2,...,K), IDIRNR
C     Det er K direktiver med i versjonen.
      α1
      α2
      αK
1111  IF(THT) CALL SKRIV(IDIRNR, 'GÅR UT AV TEST.')
      RETURN
      END
```

α_L er to linjer koding:

```
      L  CALL TL
         GOTO 1111
```

hvor T_L er testrutine for direktiv L.

2.9. EKSEi-rutinene

Eksekveringsrutinen EKSEi kaller på riktig direktivrutine (Det finnes en EKSE rutine - EKSEi - for hver direktivorden i. Største verdi for i er 4), samt en del andre rutiner og holder orden på referanser og pekere som skal tilbakestilles etter eksekveringen av direktivet. EKSEi-rutinene inneholder en stor switch på samme måte som rutinen TEST.

3. BRUKERMANUAL FOR VERSJONSGENERERINGSRUTINENE PÅ H6060

I det etterfølgende gis det en veiledning i bruken av versjonsgenereringsrutinene på H6060. Den gir en oversikt over input til de forskjellige rutiner, og styrekort som er nødvendige for å kalle opp rutinene. For datakort som er punchet på IBM 29 punch, er det nødvendig å legge styrekortet $ INCODE IBMEL foran.

3.1. Manipulering med brukerrutiner i biblioteket

All manipulering med brukerrutiner utføres via systemrutinen PRE. Denne rutinen vedlikeholder informasjonen om hver brukerrutine i biblioteket, samtidig som den genererer de styrekort som er nødvendig for FILEDIT ved oppdatering av brukerrutinebiblioteket. Informasjonen som angir manipuleringer av rutinene må ligge sortert på rutinenavn. Felles for manipuleringen med rutinene er at følgende styrekort må angis.

```
      $ SELECT SSB/DATSY/VG-INFO/PRE-STRT
     [$ INCODE IBMEL]
```

Deretter følger de datakort som er nødvendig og disse avsluttes med styrekortet

$ SELECT SSB/DATSY/VG-INFO/ { PRE-1-2 / PRE-2-1 }

avhengig av om man skal oppdatere biblioteket fra file 1 til file 2 eller omvendt.

Hvis man skal innføre en rutine som er skrevet i et annet sprog enn Fortran, kan man bare sette det tilsvarende kompilatornavn fra kolonne 73 på stjernekortet dvs. ett datakort som inneholder en '*' i kolonne 1 (3.1.1). På denne måten kan man få innført rutiner som f.eks. er skrevet i GMAP.

Hvis det er ønskelig med opsjoner på kompilatorkortet, kan opsjonene punches fra kolonne 16 på stjernekortet.

Ved valg av navn på direktivrutiner gjelder følgende regler:

1) Navnet må begynne med bokstaven 'D' etterfulgt av et tall
2) Hvis navnet består av færre enn 6 karakterer må man sette karakteren 'D' til slutt.

EKS: D101D
D10119

Det samme gjelder for testrutiner men med 'T' i stedet for 'D'

EKS: T1T
T11999

En meget vanlig nybegynnerfeil er å få feil rekkefølge på de kortene som er input til PRE, dvs. de er ikke alfabetisk sortert etter rutinenavn.

Det er videre nødvendig å føre inn brukerrutinene til et direktiv før direktivinformasjonen innføres.

3.1.1. Innføring av rutiner

Ved innføring av ny rutine i biblioteket benyttes et datakort hvor det punches fra kolonne 1:

*(ny-rutine

etterfulgt av den symbolske teksten til Fortran-rutinen hvor subrutinenavnet må være det samme som 'ny-rutine'.

Det er fullt mulig å innføre flere rutiner samtidig.

Rutinen(e) vil alltid kompileres og utlistes når FILEDIT innfører rutinen i biblioteket.

3.1.2. Modifisering av rutiner

Ved modifisering av rutiner som tidligere er innført i biblioteket benyttes datakortet hvor det punches fra kolonne 1 (Λ betyr blank)

*Λnavn

hvor 'navn' er rutinenavnet til den tidligere innførte rutinen.

For hver rettelse som skal utføres i rutinen puncher man et datakort som begynner med et minustegn i kolonne 1.

Ved innsetning av en ny linje i Fortran-teksten skriver man nummeret på den linjen som den nye skal ligge foran, f.eks.

-5
ΛΛΛΛΛΛIANT=1

Skal man endre en eller flere linjer i Fortran-teksten skriver man linjenummeret fra og med den linjen som skal endres til og med den linjen som skal endres etter minustegnet, f.eks.

-5,6
ΛΛΛΛΛΛA=1.0
ΛΛΛΛΛΛB=2.0
ΛΛΛΛΛΛC=3.0

De Fortran linjer som kommer etter datakortet som inneholder minus i første kolonne, blir satt inn i Fortran-teksten på den tilsvarende plass som man refererer til via tallene som følger minustegnet.

Det er fullt mulig å modifisere flere rutiner i samme kjøring.

Rutiner som modifiseres vil alltid kompileres og utlistes når FILEDIT oppdaterer biblioteket.

3.1.3. Fjerning av rutine

Ved fjerning av en rutine benyttes datakort hvor det punches fra kolonne 1:

*DΛnavn

og hvor 'navn' er rutinenavnet til den tidligere innførte rutinen.

Rutinen og den tilsvarende rutineinformasjon vil dermed fjernes fra biblioteket. Direktiv-informasjonen vil ikke bli fjernet.

3.1.4. Innføring av rutiner som kaller på andre brukerrutiner

Hvis direktiv og/eller arbeidsrutinen kaller på andre brukerrutiner (gjelder ikke for direktivrutiner som kaller rutiner via SID eller DID) er det nødvendig å spesifisere navnene på de kallede rutiner i input til PRE. Dette gjøres med datakortene

*(navn

(som er det samme som er nevnt under 'innføring av rutiner') etterfulgt av

**{D / A} Λ navnspesifikasjoner

hvor D benyttes ved direktivrutine og A ved arbeidsrutine.

For navnspesifikasjoner gjelder følgende regler (karaktersekvensen som skal punches er gitt mellom apostroffer):

- innføring av nytt navn: '/Nyttnavn'
- endring av tidligere innført navn: 'Gammeltnavn/Nyttnavn'
- fjerning av tidligere innført navn: 'Gammeltnavn/'

Spesifisering av flere navnkombinasjoner atskilles med komma, og de må punches uten blanke imellom.

EKS:

a) Innføring av direktivrutinene DIOD som ikke kaller på noen brukerrutine.

*(DIOD
[symbolsk rutinetekst]

b) Innføring av arbeidsrutine TULL som kaller på brukerrutine ARB1 og ARB2

*(TULL
**AΛ/ARB1,/ARB2
[symbolsk rutinetekst]

c) Endring av navnet på en brukerrutine som kalles fra arbeidsrutine TULL, fra ARB1 til ARB3 samt innføring av et nytt kall på arbeidsrutine ARB4

*ΛTULL
**AΛARB1/ARB3,/ARB4
[Modifiseringskort for rutinen TULL]

3.2. Initiering av direktivinformasjonsfilen

Ved initiering av direktivinformasjonsfilen DIRINFO blir det opprettet pekerkjeder i informasjonsarrayer. Informasjon om objektklasser samt alternativ klasser i DATSY innsettes også inn i arrayer ved hjelp av DATA-statementer (se [5] - 2.2.1.3, 2.2.2.3).

3.3. Initiering av rutineinformasjonsfilen

Ved initiering av rutineinformasjonsfilen RUTINFO blir informasjonsarrayene dimensjonert og initiert med pekerkjeder. Strukturering av denne filen er vist i [8].

3.4. Innføring av direktivinformasjon

Ved innføring av et direktiv i biblioteket er det nødvendig å gi følgende data:

Kort 1

Kolonne	Variabel
1-12	Direktivnavn
13	Navntest for direktivrutinen (se nedenfor)
14-19	Direktivrutinenavn
20	Navntest for testrutinen (se nedenfor)
21-26	Testrutinenavn
27	Navntest for arbeidsrutinen (se nedenfor)
28-29	NPARA
31-32	IPARA
34-39	IVEKT
41	SERTIFISERINGSNØKKEL

og for hvert objekt direktivet har, må følgende data opplyses:

Kort 2	Variabel
1-12	KLASP
13-18	MINLP
19-24	IPVEKT
25-30	IUTYP

Verdiene må angis av de som skriver direktivet siden man ikke kan komme fram til verdiene på noen annen måte.

I posisjonen navntest for direktiv-, test- og arbeids-rutinen, punches en '*' (asteriks) hvis direktivet skal benytte en rutine som allerede brukes av et annet direktiv i biblioteket. Hvis ikke skal den være blank.

Verdier for SERTIFISERINGSNØKKELEN kan inntil videre ignoreres, da dette ennu ikke er implementert.

Navn i direktivinformasjonen punches venstrejustert mens tall punches høyrejustert.

3.4.1. Rutinen INNFOR innfører direktivinformasjonen i biblioteket (Corestørrelsen ved eksekvering 40 K)

Rutinen kalles ved styrekortet

```
$ SELECT SSB/DATSY/VG-INFO/INNFOR
[$ INCODE IBMEL]
```

Etter dette kortet kommer et antall datakort med direktivinformasjon punchet i fast format, som er vist ovenfor.

Rutinen foretar bl. annet følgende tester på input-data:

- Kolliderer det nye direktivnavnet med et tidligere innført direktivnavn. I dette tilfellet blir direktivinformasjonen ignorert.
- Er det navnkollisjon med noen tidligere innførte brukerrutinenavn.
  Hvis dette er tilfelle og det ikke er spesifisert tilsvarende '*' (se navntest ovenfor) vil ikke ny direktivinformasjonsfile bli skrevet ut når programmet terminerer.
- Input-verdiene kontrolleres. F.eks. gyldige objektklasser.

Verdiene som er input til rutinen skrives ut i tabellform og det skulle være lett for programmereren å kontrollere om de er riktige. Dette vil være en ekstra test på at det er riktige verdier som er innført i biblioteket. Hvis det er kommet feil verdier inn i biblioteket kan disse rettes opp etter en måte som er beskrevet i (3.5).

3.5. Fjerning av direktivinformasjon

3.5.1. Rutinen TAVEKK fjerner direktivinformasjon i biblioteket. (Corestørrelsen ved eksekvering er 40K).

Rutinen kalles ved styrekortet

```
$ SELECT SSB/DATSY/VG-INFO/TAVEKK
[$ INCODE IBMEL]
```

etterfulgt av datakort hvor navnet på direktivet er punchet fra kolonne 1.

Rutinen tar vekk all informasjon om direktivet i informasjonsarrayene i filen DIRINFO og frigjør det området hvor informasjonen var lagret slik at plassen kan benyttes av nye direktiv.

3.6. Hjelperutiner for biblioteket

3.6.1. Rutinen BIBINFO kan skrive ut informasjon om direktiv som er ført inn i biblioteket. En mulighet for input til rutinen er navnet til det direktivet man er interessert i informasjon om. Det er ingen grense på hvor mange direktivnavn som kan angis etter styrekortet (Corestørrelsen ved eksekvering er 32K):

```
$ SELECT SSB/DATSY/VG-INFO/BIBINFO
[$ INCODE IBMEL]
```

Ønsker man informasjon om alle direktiver som er innført er det kun nødvendig å ha et datakort med '*' (asteriks) punchet i kolonne 1. I dette tilfellet blir direktivinformasjonen for alle direktiver skrevet ut sortert etter direktivnavn (se Appendix B).

3.6.2. Rutinen UTTAB skriver ut hvilke objektklasser og alternativklasser som til nå er innført i systemet. (Corestørrelsen ved eksekvering er 32K).

Rutinen aktiveres ved styrekortet

```
$ SELECT SSB/DATSY/VG-INFO/UTTAB
```

uten etterfølgende datakort.

Det blir her produsert to tabeller. Den ene inneholder hvilke objektklasser det finnes i DATSY f.eks. TALL-H, PARAMETER-F osv. Den andre tabellen er en oversikt over hvilke objektklasser som tilhører de enkelte alternativ klasser, som f.eks. (se Appendix A):

```
KTREK:  REKORDSET  REKORD
```

3.6.3. Rutinen NYKLAS fører en ny objektklasse eller alternativklasse inn i biblioteket. Den påkalles ved styrekortet (Corestørrelsen ved eksekvering er 32K)

```
$ SELECT SSB/DATSY/VG-INFO/NYKLAS
[$ INCODE IBMEL]
```

Ved innføringen av en ny objektklasse skriver man navnet på den nye objektklassen fra kolonne 1 på hullkortet og legger det etter styrekortet.

Ved innføring av ny objektklasse i DATSY må man i tillegg foreta forandringer i enkelte systemrutiner. Brukerne kan altså ikke uten videre innføre nye objektklasser.

Brukerne kan godt innføre nye alternativ klasser dersom denne kun består av allerede innførte og godkjente objektklasser. Navnet på den nye alternativ klassen samt antall objektklasser som hører inn under denne punches på det første hullkortet (Navnet fra kolonne 1 og antallet høyrejustert i kolonne 13 og 14). Deretter følger navnene på de objektklasser som skal tilhøre alternativ klassen, ett navn pr. hullkort og punchet fra kolonne 1.

3.6.4. Rutinen Direktiv gir informasjon om hvilke rutiner et direktiv kaller på. Den kalles ved styrekortet (corestørrelsen ved eksekvering er 36K)

```
$ SELECT SSB/DATSY/VG-INFO/DIREKTIV
```

Følgende opplysninger blir skrevet ut sortert på direktivnavn:

- Direktivnavn
- Direktivrutinenavn
- Testrutinenavn
- Arbeidsrutinenavn (kan være flere)
- Hvilke direktiv som kalles fra direktivrutinen.

Direktiv-, direktivrutine- og testrutine-navn er angitt av brukeren ved innføring av direktivinformasjon i biblioteket.

Kall på arbeidsrutiner og direktiv er funnet ved gjennomsøkning av Fortran-teksten til direktivrutinen når denne føres inn i biblioteket (se Appendix C).

3.6.5. Rutinen RUT-NAVN skriver ut opplysninger om hvilke rutinenavn som til nå er brukt og innført i biblioteket. Denne informasjonen blir hentet ut fra filen RUTINFO som er beskrevet i [8]. Rutinen påkalles ved styrekortet

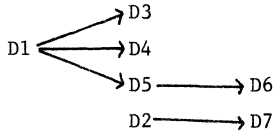
```
$ SELECT SSB/DATSY/VG-INFO/RUT-NAVN
```

3.7. Versjonsgenerering

Vi skal først beskrive hvilke brukersituasjoner som kan tenkes oppstå ved versjonsgenerering. I dette arbeidsnotatet er det ikke medtatt beskrivelse av hvorledes versjonsgeneratoren virker (se [5] kap. 2.3).

3.7.1. Brukersituasjoner

1) Det skal lages en versjon på grunnlag av en liste med direktivnavn. Listen vil være input til versjonsgeneratoren (rutinen som genererer ny versjon). Det er tilstrekkelig at direktivnavnene danner et genererende sett for versjonen. Med dette menes at det blant de direktiv som inngår i et sammensatt direktiv er nok å angi det direktiv som har høyest orden. F.eks. anta at vi skal lage en versjon med direktivene D1-D7 og at disse kaller på hverandre slik:



Da vil D1 og D2 være et genererende sett. Har man laget mange sammensatte direktiver, vil det genererende sett med direktivnavn kunne bli liten.

2) To versjoner skal slås sammen til en. Den nye versjonen vil da inneholde unionen av direktivene til de to andre versjonene. Input til versjonsgeneratoren vil være to genererende sett.

3) En versjon skal utvides med et nytt direktiv. Input vil være et genererende sett for versjonen og navnet på det nye direktivet.

I tillegg til det genererende settet med direktivnavn er det også nødvendig å gi opplysninger om segmenteringen til direktivene. Dette skal vi se på i neste kapittel.

3.7.2. Segmenteringsinformasjon

Segmenteringen styrer oppdelingen av absoluttelementet (programmet) i segmenter slik at de forskjellige segmenter kan overlappe hverandre i primærlageret. Informasjon om segmenteringen gis sammen med direktivlisten som er input til versjonsgeneratoren.

Segmentene bestemmes ved å legge et datakort med en "*" (asteriks) punchet i første kolonne etter de direktivnavn som utgjør hvert segment. F.eks. hvis direktivene MERGE, CSORT og MULTIPLISER skal ligge i samme segment og SORTER i et annet, vil inputlisten til versjonsgeneratoren være (ett navn pr. datakort):

```
MERGE
CSORT
MULTIPLISER
*
SORTER
*
```

Segmenteringsprogrammet lager dermed de nødvendige styrekort som loaderen trenger ved loadingen av rutinene.

Segmenteringsprogrammet er til nå laget etter enkle prinsipper, siden det er ønskelig å vinne erfaring ved segmenteringen før det utvikles et bedre program.

Slik som programmet virker idag vil det være en fordel å først spesifisere alle første ordens direktiver i segmenter slik at det største segmentet ligger sist blant disse. Lengden på segmentene kan finnes ut på to forskjellige måter:

- Beregn den totale lengden av segmentet ved hjelp av utskriften fra programmet RANLIB hvor lengden til hver brukerrutine er spesifisert.
- Se på utskriften fra en SYSEDIT-listing hvor hvert linkelements lengde er beskrevet.

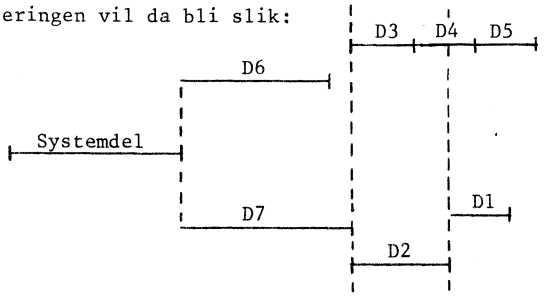
Det er helt nødvendig at det største elementet for hver direktivorden kommer sist, ellers kan det oppstå feil ved kjøring av versjonen. Deretter spesifiserer man annen ordens segmenter på samme måte som første ved å legge disse etter første ordenssegmentene.

Tilsvarende for høyere ordens direktiver.

Segmenteringsprogrammet vil imidlertid klare å segmentere et genererende sett med direktivnavn hvis man spesifiserer grupper av genererende sett, men man må være oppmerksom på virkemåten til programmet. F.eks. hvis man har den kallsekvens mellom direktivene som er vist i (3.7.1), og spesifiserer input til versjonsgeneratoren på følgende måte:

```
D1
*
D2
*
```

Segmenteringen vil da bli slik:

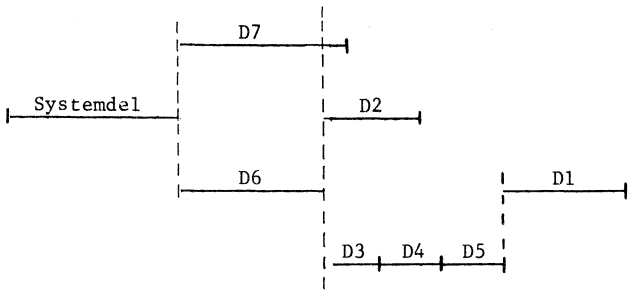


Først blir alle første ordensdirektivene lagt etter hverandre i segmenter. Deretter tar man annen ordens direktiver og deretter tredje ordens direktiver. Man ser av segmenteringen at det blir overlapping av D1 og D4 og D5 siden D1 blir lagt like etter D2 som er siste segment.

Input listen kan heller ikke spesifiseres som følger

```
D2
*
D1
*
```

Resultatet av segmenteringen ville da blitt:

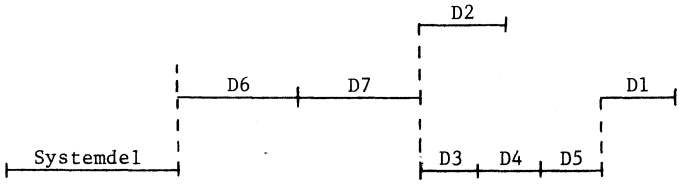


Her er resultatet det at D2 overlapper D7 og vi ville fått feil ved eksekvering.

Vi ser derfor at det sikreste er først å spesifisere første ordens direktiver i segmenter. Deretter kommer annen ordens direktiver osv. Inputlisten til versjonsgeneratoren kan da f.eks. bli som følger

```
D6
D7
*
D2
*
D3
D4
D5
*
D1
*
```

Vi ville da fått følgende segmentering



4. STYREKORTEKSEMPEL FOR BRUK AV VERSJONSGENERERINGSRUTINENE

Vi skal ved et styrekorteksempel vise hvorledes man kan generere en ny versjon med spesifiserte direktiver på H6060. Følgende styrekort benyttes

```
1       $ SNUMB ———
2       $ IDENT ——————————
3       $ LIMITS 10,67K,,10K
4       $ USERID ———
```

Hvis alle direktivene som skal være med i versjonen er riktig innført i biblioteket på forhånd, så trenger man ikke styrekortene 5, 6, 7, 8 nedenfor.

Har man direktiv-, test- eller arbeids-rutiner som skal inn i biblioteket fra hullkort føres disse inn via rutinen PRE. Det samme gjelder forandringer i eksisterende rutiner.

```
5       $ SELECT  SSB/DATSY/VG-INFO/PRE-STRT
       [$ INCODE  IBMEL]
       [Datakort se (3.1)]

6       $ SELECT  SSB/DATSY/VG-INFO/ {PRE-1-2}
                                     {PRE-2-1}
```

Ved forandring av direktivinformasjon som allerede er ført inn i biblioteket, må man ha styrekortet

```
7       $  SELECT  SSB/DATSY/VG-INFO/TAVEKK
       [$  INCODE  IBMEL]
       [Datakort med direktivnavn punches fra kolonne 1]
```

Deretter må man innføre de nye verdiene i biblioteket.

```
8       $  SELECT  SSB/DATSY/VG-INFO/INNFOR
       [$  INCODE  IBMEL]
       [Datakort]        For punching av datakort se (3.4)
```

Ny versjon lages ved styrekortet

```
9       $  SELECT  SSB/DATSY/VG-INFO/NYVERS
```

Datakort til versjonsgeneratoren kan enten være:

```
10     [Datakort med segmenteringsinformasjon]
```

eller hvis datakortene ligger på file

```
10      $  SELECT  SSB/DATSY/filenavn
```

Det er også mulig å legge datakort før eller etter select-kortet for å øke det genererende settet.
Deretter følger styrekortet

```
11      $  SELECT  SSB/DATSY/VG-INFO/SYSLD- {P}
                                            {T}
```

som lager en absoluttversjon som heter DATSY og legger denne på file 'P' (DATSPROD) eller 'T' (DATSTEST).

Eksekvering av absoluttversjonen skjer ved styrekortene

```
12      $  PROGRAM  DATSY
13      $  PRMFL    H*,R,R,SSB/DATSY/filenavn
14      $  PRMFL    **,R,R,SSB/DATSY/filenavn
15      $  DATA     I*
       [$  INCODE   IBMEL]
       [DATSY-program]
```

Filenavn i kort 13 og 14 er navnet på den filen hvor absoluttprogrammet ligger (f.eks. DATSTEST).

```
16      $  ENDJOB
17      ***EOF
```

5. ADMINISTRERING AV RUTINENE SOM OPERERER PÅ BIBLIOTEKET

5.1. Dataadministrator

Programbiblioteket vil være felles for Statistisk Sentralbyrå. Det er derfor viktig at det ikke foregår ukontrollert operering på biblioteket (ved f.eks. bruk av rutinene TAVEKK, INNFOR, PRE). Alle operasjoner bør utføres sentralisert f.eks. av en dataadministrator som samler oppdateringer og rettelser. Dataadministrator vil da bestemme tidspunktene for oppdatering av biblioteket og produksjon av nye DATSY-versjoner.

REFERANSER

[1] Spurkland, Sverre: DATSY, GENERELL BESKRIVELSE. DATSY-rapport nr. 5, Norsk Regnesentral. Februar 1971.

[2] Totland, Helge: DATSY, MANUAL FOR INNFØRING AV NYE DIREKTIV. DATSY-rapport nr. 6, Norsk Regnesentral. Januar 1971.

[3] Håndbok for bruk av DATSY. Statistisk Sentralbyrås Håndbøker 33, Oslo, 1974.

[4] Bjerkholdt, Olav, Hustveit, Anne, Sand, Paal: MODIS IV Dokumentasjonsnotat nr. 1. Behandling av eksogene variable og bruk av alternativer. Arbeidsnotater fra Statistisk Sentralbyrå IO 74/33, Oslo, 1974.

[5] Jacobsen, Ola: DATSY, VERSJONSGENERATOR. DATSY-rapport nr. 12, Norsk Regnesentral. Februar 1974.

[6] Totland, Helge: DATSY, IMPLEMENTERING AV ET BRUKERORIENTERT DATASPRÅK. DATSY-rapport nr. 7, Norsk Regnesentral. September 1971.

[7] Kristoffersen, Eva, Krogdahl, Stein, Totland, Helge: DATSY, SYSTEM OG HJELPERUTINER. DATSY-rapport nr. 10, Norsk Regnesentral. Januar 1971

[8] Jacobsen, Ola: Vedlikehold av versjonsgenereringssystemet. Internt notat fra Systemkontoret nr. A255-0. Statistisk Sentralbyrå. Februar 1975.

```
SNUMB = SBC37, ACTIVITY # = 02, REPORT CODE = 06, RECORD COUNT = 000357

                                  ****************************
                                  *  MULIGE KLASSER I DATSY  *
                                  *  ----------------------  *
                                  *          TALL-H          *
                                  *          TALL-F          *
                                  *          PARAMETER-H     *
                                  *          PARAMETER-F     *
                                  *          VEKTOR-H        *
                                  *          VEKTOR-F        *
                                  *          MATRISE-H       *
                                  *          MATRISE-F       *
                                  *          NAVN            *
                                  *          NAVNORD         *
                                  *          LISTE           *
                                  *          FELT            *
                                  *          REKORD          *
                                  *          REKORDSET       *
                                  *          SUBLISTE        *
                                  *          ARKIVTAPE       *
                                  *          FORMAT          *
                                  *          PARAMETER-K     *
                                  *          VEKTOR-K        *
                                  *          MATRISE-K       *
                                  *          PARAMETER-D     *
                                  *          VEKTOR-D        *
                                  *          MATRISE-D       *
                                  *          NYTAPE          *
                                  *          EKSTERNTAPE     *
*************************************************************************************************************
*          MULIGE ALTERNATIVE KLASSER I DATSY                                                               *
*          ----------------------------------                                                               *
*  KTFLYT  : PARAMETER-F  VEKTOR-F     MATRISE-F                                                            *
*  KTALLE  : TALL-H       TALL-F       PARAMETER-H  PARAMETER-F  VEKTOR-H     VEKTOR-F      MATRISE-H     *
*            MATRISE-F    NAVN         NAVNORD      LISTE        FELT         REKORD        REKORDSET     *
*            SUBLISTE     ARKIVTAPE    FORMAT       PARAMETER-K  VEKTOR-K     MATRISE-K     PARAMETER-D   *
*            VEKTOR-D     MATRISE-D    NYTAPE       EKSTERNTAPE                                           *
*  KTREK   : REKORDSET    REKORD                                                                           *
*  KTFNT   : FELT         TALL-H       TALL-F       NAVN                                                    *
*  KTLNVP  : LISTE        NAVNORD      VEKTOR-H     PARAMETER-H  VEKTOR-F     PARAMETER-F   VEKTOR-K      *
*            PARAMETER-K  VEKTOR-D     PARAMETER-D  REKORDSET                                             *
*  KTLIVE  : LISTE        VEKTOR-H     VEKTOR-F     VEKTOR-K     VEKTOR-D     REKORDSET     REKORD        *
*  KTVML   : VEKTOR-H     VEKTOR-F     MATRISE-H    MATRISE-F    LISTE        REKORDSET                   *
*  KTMTRK  : PARAMETER-F  VEKTOR-F     MATRISE-F    REKORDSET                                              *
*  KTNAVN  : NAVNORD      NAVN                                                                             *
*  KTMTVK  : MATRISE-F    VEKTOR-F                                                                         *
*  KTPAR   : PARAMETER-H  PARAMETER-F                                                                      *
*  KTSDM   : VEKTOR-F     MATRISE-F    VEKTOR-D     MATRISE-D                                              *
*  KTPT    : TALL-H       PARAMETER-H                                                                      *
*  KTMAT   : MATRISE-H    MATRISE-F    MATRISE-D                                                           *
*  KTSV    : VEKTOR-H     SUBLISTE                                                                         *
*  KTMATP  : MATRISE-H    MATRISE-F    MATRISE-K    MATRISE-D                                              *
*  KTMVLR  : MATRISE-H    MATRISE-F    VEKTOR-H     VEKTOR-F     LISTE        REKORDSET                   *
*  KTNHP   : PARAMETER-H  NAVNORD                                                                          *
*  KTPNR   : PARAMETER-H  PARAMETER-F  NAVNORD      REKORD                                                 *
*  KTARK   : PARAMETER-H  NAVN         ARKIVTAPE    NYTAPE       EKSTERNTAPE                               *
*  KTPALI  : PARAMETER-H  PARAMETER-F  NAVNORD      LISTE                                                  *
*************************************************************************************************************
```

SNUMB = SBL37, ACTIVITY # = 01, REPORT CODE = 06, RECORD COUNT = 001161

****

DIREKTIVNAVN	:	ABS					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D1L53D	OBJEKT NR	1	:		KTFLYT	1	1	1
TESTRUTINENAVN	:	T56T	OBJEKT NR	2	:		KTFLYT	1	1	2
NPARA	:	2								
IPARA	:	2								
IVEKT	:	1000								
SERTIFISERINGSNØKKEL:		0								

****

DIREKTIVNAVN	:	ADDER					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D1L40D	OBJEKT NR	1	:		KTMTRK	1	1	1
TESTRUTINENAVN	:	T41T	OBJEKT NR	2	:		KTMTRK	1	1	1
NPARA	:	3	OBJEKT NR	3	:		KTMTRK	1	1	2
IPARA	:	3								
IVEKT	:	1000								
SERTIFISERINGSNØKKEL:		0								

****

DIREKTIVNAVN	:	ADDERKUMULA					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D1L1D6	OBJEKT NR	1	:		KTFLYT	1	1	1
TESTRUTINENAVN	:	T1D9T	OBJEKT NR	2	:		KTFLYT	1	1	2
NPARA	:	2								
IPARA	:	2								
IVEKT	:	1000								
SERTIFISERINGSNØKKEL:		0								

****

DIREKTIVNAVN	:	ADDRADADDER					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D2L7D	OBJEKT NR	1	:		PARAMETER-H	0	1	1
TESTRUTINENAVN	:	T64T	OBJEKT NR	2	:		PARAMETER-H	0	1	1
NPARA	:	5	OBJEKT NR	3	:		MATRISE-F	1	5	1
IPARA	:	5	OBJEKT NR	4	:		MATRISE-F	1	1	1
IVEKT	:	0	OBJEKT NR	5	:		MATRISE-F	1	5	2
SERTIFISERINGSNØKKEL:		0								

****

DIREKTIVNAVN	:	ADDTILDELKOL					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D1L36D	OBJEKT NR	1	:		MATRISE-F	1	4	1
TESTRUTINENAVN	:	T37T	OBJEKT NR	2	:		SUBLISTE	0	1	1
NPARA	:	4	OBJEKT NR	3	:		PARAMETER-F	0	1	1
IPARA	:	4	OBJEKT NR	4	:		MATRISE-F	1	4	2
IVEKT	:	1000								
SERTIFISERINGSNØKKEL:		0								

****

DIREKTIVNAVN	:	ADDTILDELMAT					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D1L35D	OBJEKT NR	1	:		MATRISE-F	1	4	1
TESTRUTINENAVN	:	T36T	OBJEKT NR	2	:		SUBLISTE	0	1	1
NPARA	:	5	OBJEKT NR	3	:		SUBLISTE	0	1	1
IPARA	:	5	OBJEKT NR	4	:		PARAMETER-F	0	1	1
IVEKT	:	1000	OBJEKT NR	5	:		MATRISE-F	1	4	2
SERTIFISERINGSNØKKEL:		0								

****

DIREKTIVNAVN	:	ADDTILDELRAD					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D1D37D	OBJEKT NR	1	:		MATRISE-F	1	4	1
TESTRUTINENAVN	:	T38T	OBJEKT NR	2	:		SUBLISTE	0	1	1
NPARA	:	4	OBJEKT NR	3	:		PARAMETER-F	0	1	1
IPARA	:	4	OBJEKT NR	4	:		MATRISE-F	1	4	2
IVEKT	:	1000								
SERTIFISERINGSNØKKEL:		0								

****

DIREKTIVNAVN	:	ANTDOBLIST					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D1L44D	OBJEKT NR	1	:		LISTE	3	1	1
TESTRUTINENAVN	:	T45T	OBJEKT NR	2	:		DUMMY	0	1	0
NPARA	:	1								
IPARA	:	2								
IVEKT	:	1000								
SERTIFISERINGSNØKKEL:		0								

****

DIREKTIVNAVN	:	BEREGN-KUMUL					KLASP	MINLP	IPVEKT	IUTYP
DIREKTIVRUTINENAVN	:	D2L12D	OBJEKT NR	1	:		MATRISE-F	1	0	1
TESTRUTINENAVN	:	T131T	OBJEKT NR	2	:		MATRISE-F	1	0	1
NPARA	:	3	OBJEKT NR	3	:		MATRISE-F	1	0	2
IPARA	:	3								
IVEKT	:	0								
SERTIFISERINGSNØKKEL:		0								

****

SNUMB = SBC37, ACTIVITY # = 03, REPORT CODE = 06, RECORD COUNT = 000257

DIREKTIV	DIREKTIVRUTINE	TESTRUTINE	ARBEIDSRUTINE	KALLER PÅ DIREKTIV
ABS :	D1D53D	T56T	SUBABS	
ADDER :	D1D40D	T41T	GENADD	
ADDERKUMULA :	D1D1D6	T1D9T	KUMADD	
ADDRADADDER :	D2D7D	T64T		NAVNO PARA GENOBJEKT KOPI DELKOLVIS ADDER RADADDER SETTVEDSIDE
ADDTILDELKOL :	D1D36D	T37T	ADELKO	
ADDTILDELMAT :	D1D35D	T36T	ADDEMA	
ADDTILDELRAD :	D1D37D	T38T	ADELRA	
ANTDOBLIST :	D1D44D	T45T	ANTDOB	
BEREGN-KUMUL :	D2D12D	T131T		PARA DELHORISONT ELEMULT ADDERKUMULA
BEREGN-MARPI :	D2D9D	T130T		KOPI PARA DELKOLVIS TRAPESMETODE TRANSPONER SETTVEDSIDE ELEMULT SUBTRAHER
BEREGN-PAALO :	D2D13D	T132T		PARA DELHORISONT ELEMULT ADDER
BIGRADSKALER :	D2D3D	T52T		KOPI DELKOLVIS RADSKALER SETTVEDSIDE
BMDSTATRED :	D1D98D	T101T	BMD	
BYTTFELT :	D1D112	T115T	BYTTF	
BYTTKOL :	D1D22D	T23T	KOLBYT	
BYTTMAT :	D1D21D	T22T	MATBYT	
BYTTRAD :	D1D23D	T24T	RADBYT	
BYTTREK :	D1D113	T115T	BYTTR	
CSORT :	D1D122	T125T	ACSORT	
DELHORISONT :	D1D111	T114T	DELH	
DELKOLVIS :	D1D59D	T62T	DELKOL	
DELLISTE :	D1D109	T112T	DELLIS	
DIAGONALISER :	D1D56D	T59T	DIAG	
DIVIDER :	D211D	T211T		INVERS ELEMULT
DOKUMENTER :	D1D102	T105T		
EKSAGGREDREK :	D1D114	T117T	AGGREG	
EKSKOL :	D1D17D	T18T	KOLEKS RADEKS	
EKSMAT :	D1D16D	T17T	MATEKS	
EKSRAD :	D1D18D	T19T	RADEKS	
EKSREKORD :	D1D48D	T49T	EKSREK	
ELEMULT :	D1D54D	T41T	ELEMUL	
ENDMAKRO :	D1D196	T196T		
ENDRING :	D1D117	T120T	ENDR	