Arbeidsnotater

STATISTISK SENTRALBYRÅ

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

IO 74/33

16. juli 1974

Makrosystemet

DATSY-notat nr. 1

Av

David Walker

INNHOLD

Dette notatet er det første i en serie arbeidsnotater om DATSY. Det er meningen at serien skal omfatte både teknisk og brukerorientert dokumentasjon av videreutvilking av systemet.

1. Innledning*

DATSY (Data Treatment System) er et brukerorientiert 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. Gjennom bruken av DATSY har det åpenbart seg behov for å utvikle systemet videre for å oppnå et mer hensiktsmessig verktøy for implementering av økonomiske modeller.

Dette notat er bl.a. en manual for bruk av en slik utvidelse, som er programmert av forfatteren og godkjent av Norsk Regnesentral. Utvidelsen kalles for makrosystemet, og er redegjort nærmere for i avsnitt 2. Notatet inneholder også to andre forslag til videreutvikling av DATSY som vil kunne utbedre noen av de svakheter en har funnet ved systemet (se avsnitt 9).

Det er forutsatt noe kjennskap til EDB i avsnitt 2, og til innholdet i [3] eller tilsvarende i avsnitt 3-9.

2. Formål

Makrosystemet er introdusert som én av flere mulige løsninger av et bestemt problem i forbindelse med produktiv drift av modellen MODIS IV (se [4], vedlegg 1 for en liste over dokumentasjon av denne). Dette problem er at ethvert DATSY-program er begrenset til 200 setninger, slik at programmet for MODIS IV er delt opp i 10-15 forskjellige kjøringer

*) Forfatteren vil gjerne takke avdelingsdirektør Erik Aurbakken, forsker Olav Bjerkholt og konsulent Ola Jacobsen for hjelp med tilrettelegging av dette notatet.

forbundet med hverandre gjennom utskrift på og innlesing fra arkivtaper. Dette er svært tungvint. Det forårsaker gjennomsnittlig 40 tapemonteringer for en enkelt kjøring av MODIS, mens bare 5 av disse egentlig kan sies å være påkrevet. (Selv det tallet er avhengig av en annen svakhet med DATSY, nemlig at DATSY pr. idag ikke kan bruke magnetiske platestasjoner.) Videre innebærer dette forhold at gjennomløpstiden for en fullstendig MODIS-kjøring øker betraktelig.

Det er ikke mulig simpelthen å utvide grensen på 200 setninger uten en meget omfattende omprogrammering av DATSY. Blant annet blir det slik forholdene er avsatt plass for alle setningene i core, og i tillegg øker de ressursene som kreves for kompilering mer enn proporsjonalt med antall setninger. Det er heller ikke aktuelt å programmere modellen i Fortran som en rekke sammensatte direktiver (se [2], s. 35), fordi dette for det første ville tatt for lang tid, for det andre ville det gjort det mye vanskeligere å forandre i modellen, og for det tredje lå det visse begrensninger på antall direktiver som kunne brukes i en bestemt DATSY-versjon. Den åpenbare løsning, nemlig en lettere overgang mellom de 10-15 programmene omtalt ovenfor, var for vanskelig.

Med dette som bakgrunn utviklet forfatteren makrosystemet, hvor to direktiver (MAKRO og VMAKRO) kompilerer og eksekverer direktivsetninger. De 10-15 programdeler til MODIS kan (etter små forandringer) med dette systemet bli kopiert over på arkivtape som data og kjørt av et overordnet DATSY-program. Ca. 150 linjer med forandringer var nødvendig i den overordnede delen av DATSY (se avsnitt 1). Åpenbart innbyr systemet til mer generelle videreutviklinger, og dette er det redegjort nærmere for i avsnitt 9.

Samtidig som makrosystemet løser problemet med unødvendig inndeling av MODIS i delprogrammer, er det også gode utsikter for at det - etter omprogrammering av en del av en rutine i assemblyspråk - skal løse et annet problem forbundet med kjøring av MODIS. Dette er at DATSY-programmer må kompileres hver gang de kjøres, selv når de står uforandret. Slik unødig omkompilering utgjør ca. 20-30% av de totale kostnader ved kjøring av modellen. (Utlisting av DATSY-program kan heller ikke undertrykkes og det vil også kunne inkluderes her.)

Det er altså aktuelt å bruke makrosystemet først og fremst når man ønsker å bruke flere enn 200 setninger i én eksekvering av DATSY, men andre anvendelser blir også antydet senere.

Systemet er implementert på maskinene UNIVAC 1108 og Honeywell-Bull H6060.

3. Generell Beskrivelse

En generell beskrivelse for brukere av makrosystemet blir gitt i dette avsnitt. Ved normal førstegangsbruk av direktivet MAKRO plukker man ut direktivsetninger som allerede er blitt testet i en DATSY-kjøring (i hvert fall med 'SLUTTKONT' og helst med 'SLUTTPROD') og legger dem bak DATSY-programmet som data i en liste. Denne listen kan skrives ut på en arkivtape slik at den blir lagret for senere bruk. Den kalles for et makroprogram.

Alle objekter som nevnes i makroprogrammer må enten deklareres eller brukes i hovedprogrammet hvor enten direktivet MAKRO eller VMAKRO brukes for å kjøre makroprogrammet. Disse objektene inkluderer objekter som blir lest inn fra kort, hentet fra en arkivtape, produsert i selve makroprogrammet, eller som blir brukt utelukkende som mellomobjekter i makroprogrammet.

Makroprogrammer kan gjerne inkludere tomme setninger ment som kommentar, men bør helst ikke inneholde deklarasjoner, deskripsjoner, eller hjelpesetninger. Disse tre siste setningstyper vil imidlertid bare bli betraktet som tomme setninger hvis de forekommer, og vil ikke forårsake noen feil. Alle tomme setninger blir skrevet ut hver gang MAKRO eller VMAKRO brukes for å kjøre et makroprogram.

Ved å skrive setningen:

MAKRO PROG. (f.eks.)

i DATSY-programmet kan et makroprogram i en liste som heter 'PROG' kjøres. Ordet 'PROG' er valgfritt.

Eksempel 1

DATSY-programmet, som er forutsatt kjørt uten feil før vi begynner, er som følger:

```
ARKIVTAPE STANDARD ARK1 ARK2.
MATRISE C E.
VI HENTER A, B, OG D AUTOMATISK FRA ARK1 (TOM SETNING).
ADDER A + B = C.
MULTIPLISER C * D = C.
SUBTRAHER A - E = E.
```

(eksemplet fortsetter på neste side)

```
SOM MAN SER BLIR C OG E PRODUSERT I KJØRINGEN.

SKRIV PÅ ARK2 A B C E.

SLUTTPROD.
```

Dette er et så lite program at det er ingen vits å bruke MAKRO for å kjøre det, men eksemplet vil illustrere bruken av makrosystemet. Et makroprogram, som vi kaller for PROG98, kan lages og kjøres ved å bruke de opprinnelige punchede kort (eller kortbilder), pluss noen nye kort slik:

```
ARKIVTAPE STANDARD ARK1 ARK2.

MATRISE C E.

MATRISE A B D.

MAKRO PROG98.

LISTE PROG98.

SKRIV PÅ ARK2 A B C E.

SLUTTPROD.

* PROG98,A80
VI HENTER A, B, OG D VED DEKLARASJON FRA ARK1 (TOM SETNING).
ADDER A + B = C.
MULTIPLISER C * D = E.
SUBTRAHER A - E = E.
SOM MAN SER BLIR C OG E PRODUSERT I KJØRINGEN.
```

I tillegg til matrisene C og E, som ble deklarert i det opprinnelige DATSY-program, er A, B og D blitt deklarert i eksemplet ovenfor. Som tidligere nevnt, må en del ekstra deklarasjoner til når MAKRO eller VMAKRO brukes. I dette tilfellet er deklarasjonene til A og B overflødige, og den til D nødvendig (se reglene om dette i avsnitt 4).

Et makroprogram som er skrevet ut på en arkivtape kan senere kjøres om i kjøringer hvor arkivtapen er deklarert uten at makroprogrammet blir deklarert på nytt. Det blir hentet automatisk fra arkivtapen, akkurat som vanlig med objekter i DATSY-programmer. Vi tar et nytt eksempel for å vise dette.

Eksempel 2

```
ARKIVTAPE STANDARD BÅND1.

LISTE PROGRAM.
```

(eksemplet fortsetter på neste side)

```
SKRIV PÅ BÅND1:  PROGRAM.

SLUTTPROD.

* PROGRAM,A80
        MULTIPLISER X * Y = W.
        ADDER W + Y = W.
        ADDER Y + W = X.
        INVERTER X = Z.
        PRINTVERDI Y.
```

Etter denne kjøringen kan man kjøre PROGRAM for å skrive BÅND3 slik, hvis X og Y er matriser som ligger på BÅND2:

```
ARKIVTAPE STANDARD BÅND1 BÅND2 BÅND3.

MATRISE W Y Z.

MAKRO PROGRAM.

SKRIV PÅ BÅND3:  X Z.

SLUTTPROD.
```

Fordelene med makrosystemet kommer ikke klart fram i eksemplet, bare bruksmetoden. Poenget er at man kan lese inn en rekke slike makroprogrammer, hvert av dem som en liste, og kjøre dem etter hverandre innen et DATSY-program slik:

```
.
.
.

MAKRO PROGA1.

MAKRO PROGA2.

MAKRO PROGC.

.
.
.
.
```

Det totale antall direktivsetninger som kan kjøres på denne måte innen ett DATSY-program ligger over 200 000 på en UNIVAC 1108. I store økonomiske modeller blir det da ikke nødvendig å dele modellen opp i en rekke DATSY-program forbundet med hverandre gjennom utskrift og innlesing av arkivtaper.

MAKRO og VMAKRO direktivene kan brukes innenfor makroprogrammer, uten noen nevneverdige begrensninger. Filekodene 14 og 16 blir normalt

brukt av makroprogrammer, og kan ikke brukes til noe annet. Når MAKRO og VMAKRO brukes innenfor makroprogrammer, blir også filekodene 17 og oppover opptatt, avhengig av hvor dypt dette går.

Det anbefales å bruke formatet (A80) som standard ved innlesing av makroprogrammer som lister, fordi dette er bredt nok til å ramme alle mulige direktivsetninger (husk at én setning kan gå over flere kort), og i tillegg kan direktivet SETTPÅTOPP da brukes for å sette sammen makroprogrammer. A80 ligger på dummyarkivtape STANDARD og kan brukes uten videre, som i eksemplene ovenfor.

Andre direktiver kan blandes fritt med MAKRO, som i eksemplet nedenfor.

Eksempel 3

```
ARKIVTAPE STANDARD ARK1 ARK2.
MATRISE C D E.
ADDER A + B = C.
MAKRO PROG97.
SUBTRAHER A - E = E.
LISTE PROG97.
SKRIV PÅ ARK2:  PROG97 E.
SLUTTPROD.
* PROG97,A80
MULTIPLISER C * D = E.
MATRISEN D ER FORUTSATT Å LIGGE PÅ ARKIVTAPE ARK1.
```

4. Ekstra deklarasjoner ved bruk av makroprogrammer

Behovet å deklarere (og evt. deskribere) alle objekter som nevnes bare i makroprogrammer er nødvendig slik som MAKRO er i dag, fordi:

(i) ellers ville ikke objekter brukt bare i makroprogrammer bli hentet fra arkivtaper og

(ii) ellers ville ikke objekter som skulle produseres i makroprogrammet bli etablert som DATSY-objekter og ville istedet bli tomme ord.

Regler for når det er nødvendig å deklarere (og evt. deskribere) objekter følger:

1) Deklarer hvis objektet opptrer for første gang som utobjekt i kjøringen, eller hvis objektet skal brukes som innobjekt i et makroprogram og ikke brukes i hovedprogrammet.

2) Deklarasjon er overflødig hvis objektet blir hentet automatisk fra en arkivtape av en setning i hovedprogrammet, som kan være direktiv- eller SKRIV-setning.

Reglene kan eksemplifiseres ved å finne ut hvorfor deklarasjonene til matrisene X og Z i eksempel 2 ovenfor er overflødige, og hvorfor matrisen W må deklareres.

I eksempel 3 ovenfor er det ikke nødvendig å deklarere A og B, siden de fins på ARK1 og brukes i hovedprogrammet. Matrisene C og E må som vanlig deklareres fordi de blir produsert som utobjekt i direktivsetninger i kjøringen. Matrisen D, som er forutsatt å ligge på arkivtape ARK1, må deklareres fordi den er nevnt bare i makroprogrammet.

5. VMAKRO-direktivet

VMAKRO er et direktiv som er identisk med MAKRO bortsett fra at den tar en liste (objektliste) i (A12) format som annet objekt. Objektlisten inneholder objekt- og direktivnavn som skal settes inn ved eksekvering av makroprogrammet. Hvis listen inneholder f.eks. fem navn, skal makroprogrammet inneholde fem forekomster av den spesielle erstatningsindikator, ----------, som består av mellom 6 og 12 bindestreker. Hvis alle objektnavn skal byttes ut i et makroprogram kalles det for en ren prosedyre. Da tilsvarer makroprogrammet - sett fra brukersiden - et vanlig sammensatt direktiv i DATSY.

Eksempel 4

```
ARKIVTAPE STANDARD ARK1 ARK2.
A, B OG D LIGGER PÅ ARKIVTAPE ARK1.
MATRISE C D E.
VMAKRO VPROG99 MED INNBYTTINGER.
SKRIV PÅ ARK2: A B C E.
```

(eksemplet fortsetter på neste side)

```
LISTE VPROG99, INNBYTTINGER.
SLUTTPROD.
* VPROG99, A80
DETTE ER VPROG99.
C OG E BLIR PRODUSERT NÅR VPROG99 KJØRES.
ADDER A + ------- = C.
MULTIPLISER C * --------- = E.
---------- A OG E = E.
* INNBYTTINGER,A12
B
D
SUBTRAHER
```

Elementene i objektlisten (INNBYTTINGER i eksemplet) må punches i fast format, fra 1. kolonne i kortet.

Dette eksemplet oppnår samme resultat som i eksempel 1, men innbyr til variasjoner ved å forandre i objektlisten INNBYTTINGER og i deklarasjonene.

Poenget med VMAKRO er:

i) Å løse mulige problemer som oppstår ved bruk av MAKRO, f.eks. å unngå å lage en rekke makroprogrammer som er nesten like.

ii) Å fungere som en basis eller idékilde til senere utvikling (se avsnitt 9).

6. Feilsøking og utskrifter

Direktivsetninger i DATSY-programmer blir normalt sjekket for tre mulige feiltyper, som vil forhindre at programmet blir kjørt i det hele tatt. Det samme gjelder direktivsetninger i makroprogrammer. To av disse tre feiltypene blir oppdaget før selve makroprogrammet blir kjørt, dvs. makroprogrammet blir ikke påbegynt når slike feil inntreffer. Den siste feiltypen oppdages (som MAKRO er i dag) bare under kjøring av direktivene i makroprogrammet. Hvis det skulle vise seg å være ønskelig kunne dette forandres, slik at alle tre typer feil blir oppdaget før makroprogrammer blir påbegynt. (Dette vil imidlertid kreve CPU-tid og det måtte derfor bli valgfritt om man brukte det eller ikke). Siden direktivsetninger forutsettes kjørt tidligere i et DATSY-program bør ikke

dette være nødvendig.

Feiltypene er beskrevet nedenfor.

1) Feil antall objekter i setningen (som ofte vil være punchefeil eller det at man har glemt en deklarasjon).

Siden nye deklarasjoner må skrives ut ved konvertering til MAKRO for å få inn objekter fra arkivtaper, kan feil av type 1) lett inntreffe selv når direktivsetningene allerede er blitt kjørt riktig i et vanlig DATSY-program på forhånd. Derfor er det bra at feil av denne typen blir oppdaget før en eventuell kjøring av makroprogrammet blir påbegynt.

2) Feil type objekt input til et direktiv (f.eks. forsøk på å multiplisere to lister sammen).

3) Objekt brukt som input til et direktiv, når objektet ennå ikke har fått noen verdi.

Som påpekt ovenfor blir ikke feil av type 3) oppdaget før makroprogrammet blir påbegynt (slik systemet er i dag), bare når det aktuelle direktiv skal kjøres.

Feil av type 2) kan forseres ved å bruke S-opsjon (se [3]), dvs. man kan få kjøringen til å fortsette likevel.

I tilfelle man får en feil av type 3), se først etter varsler om tomme setninger som kan tyde på f.eks. punchefeil.

Ved bruk av K-opsjon (se [3]) kan brukeren få skrevet ut et makroprogram med nummer på direktivsetninger, like før det kjøres av MAKRO eller VMAKRO. PRINTVERDI kan også brukes for å skrive ut programmet, men da refererer rekkefølgenumrene seg til kort og ikke nødvendigvis til direktivsetninger, som kan gå over flere kort. Tomme setninger blir alltid gjengitt like før kjøring av et makroprogram, slik at brukere skal kunne kontrollere at de ikke skyldes feilpunching.

7. Effektivitet

Når antall objekter med forskjellige navn som er brukt eller produsert i en DATSY-kjøring begynner å nærme seg 500, vil effektiviteten av visse beregninger (invertering, multiplisering av vanlige store matriser, osv.) bli betydelig dårligere. Så vidt forfatteren vet ligger den eneste faste grensen på antall objekter noe over 1000, og den er enda

høyere hvis få av disse objektene er rekordsett.

Ved å bruke et lite antall mellomberegningsobjekter om og om igjen kan antall objekter holdes ned. Det er fullt tillatt for ett objekt å ha mange forskjellige forekomster. F.eks. kan en matrise ha forskjellig antall rader og kolonner i forskjellige beregninger.

8. EDB-pålitelighet i store økonomiske modeller

Det kan være hensiktsmessig å begrense samkjøring av makroprogrammer når det gjelder den totale CPU-tiden som blir brukt, fordi hvis evt. utskrift av resultater på bånd går dårlig, må i så fall hele kjøringen kjøres om igjen. Derimot minsker man sjansen at slike feilskrivinger på bånd inntreffer ved å kjøre hele modellen i en kjøring ved bruk av MAKRO. Dette problemet ventes å bli mindre vesentlig på H6060 maskinen ved Statens Driftsentral, fordi utskrift på bånd er mye påliteligere der enn på eldre maskiner, og magnetiske platestasjoner kan kanskje bli brukt istedenfor magnetbånd.

Det kan være andre grunner enn feil ved utskrift på bånd som taler for at modellen kjøres i flere skritt, mens den er under utvikling. F.eks. vil manuell kontroll av data midt i en kjøring og testing av forandringer i modellen være to årsaker til forsiktighet når det gjelder store mengder av CPU-tid.

Konvertering til makroprogrammer forhindrer iallfall ikke at modellen blir kjørt skrittvis, siden hvert skritt normalt blir konvertert til ett makroprogram, og hvert av disse kan kjøres alene.

På lengre sikt, når forandringer er gjort og data input kontrolleres automatisk ved ekstra direktiver, faller grunnene for å kjøre modellen i flere skritt bort.

9. Forslag til framtidig utvikling av makrosystemet

Makrosystemet innbyr først til en utvikling hvor et nytt direktiv som kunne hete LØKKE blir laget. Dette direktivet kunne tenkes å kjøre et makroprogram et visst antall ganger, med mulighet for et betinget hopp fra et sted i makroprogrammet tilbake til det programmet som kjører makroprogrammet. Dette betingede hoppet kunne ordnes ved små

forandringer i direktiv SAMMENLIGN (se [3]). Direktivet LØKKE kunne lages ved små forandringer til direktiv MAKRO, og kunne tenkes å bruke de samme programmodulene.

Videre innbyr spesielt VMAKRO-direktivet til utvikling av en ny type sammensatt direktiv, som kan kalles for et makrodirektiv. Denne typen direktiv kunne tenkes laget av brukeren selv, ved å bruke DATSY. Direktivsetninger hvor makrodirektiver blir brukt kunne tenkes skrevet akkurat som andre direktivsetninger. Enhver makrodirektivsetning kunne tenkes byttet ut automatisk av DATSY med en tilsvarende VMAKRO-**setning** pluss evt. deklarasjoner (se avsnitt 4 og 5), uten at dette kom fram i programlistingen. Dette kunne skje straks etter at alle SUBSTITUER-setninger var blitt utført.

Bruk av M-opsjon (se [3]) ved eksekvering av DATSY-programmet ville da forårsake utskrift om VMAKRO og hver enkelt av de direktivene som utgjorde makrodirektivet.

Et slikt makrodirektivsystem ville passe utmerket for å lage direktiver basert på meget spesielle problemstillinger. Sammensetninger med mer generell anvendelse blir kanskje best utført med det eksisterende systemet for sammensatte direktiver, som må programmeres i Fortran og som derfor er lite brukt hittil av den grunn. Begge typer sammensetninger har sine respektive fordeler. Den eksisterende metode med sammensatte direktiver tillater i noen tilfelle en bedre utnytting av maskinen. Et makrodirektivsystem ville ha den fordelen at brukeren ikke behøvde å vente for å få laget sammensatte direktiver av spesielt trenede programmere, men kunne gjøre dette selv, kanskje i løpet av timer.

Makrodirektiver kunne tenkes oppbevart samlet i spesielle permanente filer på magnetisk disk, som ble deklarert i DATSY-programmer og åpnet av DATSY selv. Disse blir referert til her som makrofiler. VMAKRO-programmer med visse begrensninger kunne da kopieres ved et spesielt direktiv over til en bestemt makrofile. Forskjellige brukere burde da kunne bygge opp sine egne makrofiler med hundrevis av makrodirektiver beregnet på spesielle formål. Det bør kunne være mulig å lage og bruke makrodirektiver i forskjellige DATSY-programmer innenfor samme kjøring.

Eksempel 5

Første DATSY-program:

```
MAKROFILE FORSK.
```

(eksemplet fortsetter på neste side)

```
REGISTRER KJØR-MODELL1 PÅ FORSK.

PROGRAM KJØR-MODELL1.

FORMAT A80.

SLUTTPROD.

* A80 = (A80)
* KJØR-MODELL1, A80
```

Fullstendig DATSY-program uten datakort og med erstatningsindikatorer lagt inn (se avsnitt 5)

DATSY-programmet blir i dette eksemplet oppgitt som data i et nytt DATSY-program, og kunne tenkes å inneholde alle mulige slags setninger bortsett fra selve makrodirektivet KJØR-MODELL1. (Hvis et makrodirektiv kalte direkte eller indirekte på seg selv ville dette bli oppdaget ved bruk. En feilmelding ville komme fra direktivet VMAKRO.) KJØR-MODELL1 er deklarert her som PROGRAM, en tenkt ny objektklasse. Det tenkte direktivet, her kalt for REGISTRER, kopierer programmet ut på makrofilen FORSK. (MAKROFILE er også tenkt innført som egen objektklasse.) I et senere DATSY-program kan makrodirektivet KJØR-MODELL1 kjøres slik:

Eksempel 5 (forts.)

Andre DATSY-program:

```
MAKROFILE FORSK.

ARKIVTAPE INPUTBÅND OUTPUTBÅND.

KJØR-MODELL1 MED INPUT1, INPUT2, INPUT3 OG INPUT4, OG
  SETT RESULTATET I OUTPUT.

MATRISE OUTPUT.

SKRIV PÅ OUTPUTBÅND: OUTPUT.

SLUTTPROD.
```

Makrodirektivet KJØR-MODELL1 hentes her fra makrofilen FORSK og kjøres. Input til direktivet kommer fra INPUTBÅND, og en resultatmatrise blir skrevet på OUTPUTBÅND.

De begrensninger som syns å være nødvendig i VMAKRO-programmer som skal konverteres til makrodirektiver inkluderer bl.a. at direktivnavn ikke kan være med i objektlisten (se avsnitt 5). Derfor ville det fremdeles lønne seg enkelte ganger å bruke VMAKRO direkte som beskrevet

i avsnitt 5.

Forslaget om makrodirektiver krever vesentlig nyprogrammering. En teknisk beskrivelse av forslaget skal utarbeides. Forslaget om direktiv LØKKE og forandring i direktiv SAMMENLIGN er uavhengig av makrodirektivsystemet og krever relativt lite programmering for å få etablert.

Referanser

[1] S. Spurkland: Et brukerorientert datasprog. Norsk Regnesentral, Oslo, 1969. (Rapport nr. 0)

[2] S. Spurkland: DATSY - Generell rapport. Norsk Regnesentral, Oslo, 1971. (Rapport nr. 5)

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

[4] O. Bjerkholt, A. Hustveit, P. Sand: MODIS IV Dokumentasjonsnotat nr. 1. Behandling av eksogene variable og bruk av alternativer. Arbeidsnotater fra Statistisk Sentralbyrå IO 74/33, Oslo, 1974.