Datamaskinsystemet

D21

DAC 1

Programmeringssystem

Svenska Aeroplan Aktiebolaget

Programmeringssystemet DAC 1
för
datamaskinen Saab D21

Denna publikation utgör en sammanställning av sådana uppgifter, som är av väsentligt intresse vid programmering med programmeringssystemet DAC 1 för datamaskinen SAAB D21.

Det är utarbetat för ett datamaskinsystem av den omfattning, som angives i systembeskrivningen för SAAB D21 och kommer att utbyggas i samma takt som denna. För närmare uppgifter om D21:s funktion hänvisas till systembeskrivningen.

För handhavande av D21 vid översättningen av DAC-program hänvisas till publikationen för detta ändamål.

0128-0101-0:2

Innehållsförteckning

0128-0101-0

Kapitel 1

Programmering med DAC 1

Allmänt

DAC 1 är ett programmeringssystem, som användes vid programmering av datamaskinen SAAB D21. Det är utformat med tanke på, att programmeringen skall kunna ske på ett bekvämt sätt. Sålunda används symbolisk adressering med alfanumeriska namn, och operationsbeteckningar skrives i symbolisk form. Genom en lämplig strukturell uppbyggnad har stor frihet erhållits vid valet av operationsbeteckningar, och svårtydda förkortningar har härigenom kunnat undvikas. Detta medför ej enbart, att DAC-programmering är lätt att lära, utan även att läsbarheten av programmen i hög grad förbättras. Vid utformningen av DAC 1 har också stor hänsyn tagits till framtida utveckling, och utvidgningar kommer därför att kunna ske på ett enkelt sätt.

I DAC 1 ingår såväl pseudoinstruktioner som instruktioner, som har direkt motsvarighet i en maskininstruktion. Genom att man har tillgång till alla de direkta maskininstruktionerna, kan DAC-program utarbetas för full maskineffektivitet.

I datamaskinsystemet SAAB D21 kan långsamma yttre enheter arbeta självständigt samtidigt som beräkningar utföres i centralenheten. Genom avbrottssignaler erhåller centralenheten information från de yttre enheterna, då dessa avslutat beordrad verksamhet. Detta arbetssätt medför, att datamaskinen kan utnyttjas synnerligen effektivt men medför samtidigt, att stora krav ställes på organisationen av programmet för ett sådant effektivt utnyttjande.

I DAC 1 finns en stor uppsättning pseudoinstruktioner för in- och utmatning, vilka utformats med tanke på att göra användningen av yttre enheter enkel, samtidigt som denna sker på ett effektivt sätt. Sålunda befrias programmeraren helt från den administration, som uppkommer i samband med avbrottssignalerna, och han behöver för effektivt utnyttjande av in- och utorganen endast följa nedanstående enkla regler:

1 Beordra aktivitetet vid den yttre enheten så tidigt som möjligt. Om exempelvis data skall inläsas från hålremsa, bör inläsningsinstruktionen ej ges på den plats i programmet, där inlästa data skall börja behandlas.

2 Då programmet för det fortsatta genomloppet kräver att en tidigare beordrad verksamhet vid en yttre enhet skall vara avslutad, skrives en frågeinstruktion i programmet. Denna instruktion medför att fortsatt genomlopp av programmet ej sker, förrän den yttre enheten avslutat beordrad verksamhet. Man skall alltså i det ovan nämnda exemplet skriva en frågeinstruktion omedelbart före den plats i programmet, där inlästa data skall börja behandlas.

Programstruktur

I ett DAC-program ingår deklarationer och instruktioner.

Deklarationerna användes dels för att namnge konstanter, dels för att ge DAC-översättaren upplysningar om vilka variabler m m, som an-

0128-0101-0:3/4/5

vändes i programmet. Utläggning av och platsreservation för dataområden i snabbminnet sker också med ledning av deklarationerna.

Exempel:

Deklarationen REAL: A, Tn, PI: 3.14159; innebär, att A, Tn och PI är reella, d v s de skall representeras med flytande komma. A och Tn är variabler, PI är en konstant med värdet 3.14159. Vid utläggning av dataområdet av programmet kommer A, Tn och PI att utläggas i angiven ordning.

I instruktionsdelen av DAC-programmet skrives instruktionerna sekvensmässigt. Därvid sättes operationsbeteckningen först, varefter kan följa ett variabelnamn eller ett tal.

Exempel:

Instruktionen +, Alfa;

innebär, att det aktuella värdet av Alfa skall adderas till innehållet i ackumulatorregistret. Additionen skall ske med fast komma och i enkel precision.

I DAC 1 förekommer ett flertal pseudoinstruktioner, vilka vid översättningen ger upphov till, att en eller flera maskininstruktioner utläggas i programmet. Detta sker i många fall genom att programhopp utlägges till underprogram. Härvid kommer DAC-översättaren automatiskt att tillse, att berört underprogram tillfogas det översatta DAC-programmet.

Vid planläggningen av programmeringen för en viss större uppgift är det alltid lämpligt att uppdela problemet i mindre delproblem, som programmeras och kontrolleras var för sig. Sådana delar kan sedan sammanfogas i större sammanhörande enheter o s v, tills hela programmet är klart. I DAC 1 kan deklarationer och instruktioner sammanfogas till sådana enheter, kallade block. Flera block kan sedan i sin tur sammanfogas till block på en högre nivå. Genom att ange en programdel som ett block, kommer de variabler, som deklareras i blocket, att bli interna för detta block och har inget samband med variabler med samma namn, som deklarerats i ett överordnat block. Detta medför, att egna underprogram, som skrives i DAC och anges som block, kan tillfogas ett DAC-program utan speciell anpassning med hänsyn till använda namn.

Då DAC-programmet översättes till maskinkod, kommer varje block att betraktas som en enhet och utläggas i konsekutiva celler. För utläggningen av de olika blocken i snabbminnet och på magnetband har programmeraren full frihet att disponera dessa minnen på lämpligaste sätt. I de fall ett DAC-program är för stort att helt rymmas i snabbminnet, kan överföringen från magnetband av ett programblock ske med hjälp av DAC-instruktioner för detta ändamål, varvid endast blocknamn behöver anges.

0128-0101-0:5/6

Kapitel 2

Grundläggande begrepp

Symboler

För inmatning av DAC-program till D21 användes 8-kanals hålremsa, vars remskod framgår av appendix 2. I DAC användes endast de tecken och symboler, som har motsvarighet i tecken i hålremskoden.

I ett DAC-program är tecknen för mellanslag, vagnretur och tabulator insignifikanta och ignoreras av DAC-översättaren. Det bör dock observeras, att de har betydelse vid inläsning av dataremsor med de pseudoinstruktioner för remsinläsning, som ingår i DAC.

Stoppkod användes för att ge stopp i remsinläsningen av ett DAC-program. I övrigt ignoreras detta tecken.

För att i alfanumeriska strängar, exempelvis tabellrubriker, som utmatas från DAC-programmet, utmärka sådana typografiska operationer, som ej har motsvarighet i ett skrivet tecken, användes följande symboler:

SP användes för mellanslag
CR användes för vagnretur
TAB användes för tabulator
ST användes för stoppkod
TF användes för utplåning (tape feed)

Då de förekommer i en sträng, kan de föregås av ett heltal, som anger antalet av den typografiska operationen. Detta heltal understrykes därvid också. Förekommer ingen heltalsangivelse, utföres den typografiska operationen en gång.

Exempel ---CR 4SP---

betyder, att vagnretur utföres, varefter följer fyra mellanslag.

Namn

För namngivning av variabler, konstanter, programplatser, deklarationer och block användes alfanumeriska namn definierade på följande sätt:

En bokstav följd av ett godtyckligt antal bokstäver eller siffror.

Exempel:

A

gamma
al7B2
Max funktionsvärde

Både stora och små bokstäver får användas, och betraktas därvid som olika tecken.

Samma namn kan ej användas för att beteckna två olika storheter i ett block. Samma namn kan däremot användas för att beteckna olika storheter i olika block. I detta fall blir storheten intern för blocket och har inget gemensamt med storheten med samma namn, som finns i det andra blocket.

I namn på variabler, konstanter, programplatser och deklarationer är endast de nio första tecknen identifierande, medan överskjutande ignoreras. Två storheter i ett block måste alltså ha namn, som åtminstone har ett tecken av de nio första olika.

Namn på block får högst innehålla två tecken, d v s en bokstav följd av en bokstav eller siffra.

Tal

I ett DAC-program kan tal förekomma dels i deklarationerna för att ange numeriska värdet av en konstant eller en följd av konstanter (kapitel 5) och dels i adressdelen av en DAC-instruktion (kapitel 3).

Följande symboler användes i samband med tal:

Siffror	0-9
Decimalpunkt	.
Plustecken	+
Minustecken	-
10-exponent	10

Vid positiva tal kan plustecknet utelämnas.

Vid exponenttal måste 10-exponenten vara ett heltal. Även i de fall mantissan är 1 måste denna angivas, t ex 1_{10}-5.

Följande typer av tal förekommer:

Heltal

Ett heltal får endast innehålla siffror och eventuellt förtecken.

Exempel:
0
19
-32869

Ett heltal utlägges i ett D21-ord.

Reella tal

Ett reellt tal representeras i D21 med flytande komma (kapitel 4). Det måste förutom siffror och eventuellt förtecken även innehålla decimalpunkt eller 10-exponent eller bådadera.

Exempel:
0.0
-193.06
14_{10}-5
$0.125_{10}6$

Ett reellt tal kommer att utläggas i två konsekutiva D21-ord, varvid mantissan upptar 40 bitar inklusive teckenbit och exponenten representeras med en 8 bitars karakteristik.

Maskintal

Ett maskintal är ett tal, som uppfyller relationen

$-1 \leqslant$ maskintalet < 1

Ett maskintal kan innehålla både komma och 10-exponent. Om talet skall utläggas med enkel precision, skrives (M) efter talet, medan om det utlägges med dubbel precision (DM) skrives efter talet.

Exempel:
-1 (M)
0.33333 (M)
0.98765_{10}-5 (DM)
0 (DM)

Maskintal i enkel precision utläggas i ett D21-ord, medan maskintal i dubbel precision utlägges i två konsekutiva D21-ord. I detta senare fall representeras talet med 47 bitar inklusive teckenbit. Teckenbiten i andra ordet är alltid 0.

Då maskintal förekommer i deklarationerna, utelämnas (M) och (DM).

Oktal information

Oktal information kan förekomma i ett DAC-program, dels som oktala adresser, dels som oktala ord, d v s data eller instruktioner. Den oktala informationen kan endast innehålla siffrorna 0-7.

Oktala adresser i DAC-instruktioner

Oktala adresser kan skrivas i adressdelen av en DAC-instruktion. Därmed avses, att instruktionen skall utföras på innehållet i den cell, som anges av den oktala adressen. Den oktala adressen kan skrivas med utelämnande av nollor i början av adressen och skall följas av (OAS).

Exempel:
123 (OAS)
10034 (OAS)

avser den oktala adressen 00123 respektive 10034.

0128-0101-0:8/9/10/11

Oktala ord i adressdelen av en DAC-instruktion

Oktala ord kan skrivas i adressdelen av en DAC-instruktion. Därmed avses att instruktionen skall utföras på ordet i adressdelen. Det oktala ordet kan skrivas med utelämnade av nollor i början av ordet och skall följas av (OCT).

Exempel: ∧ , 3077 (OCT) ;

Denna DAC-instruktion innebär, att innehållet i ackumulatorregistret skall logiskt multipliceras med det oktala ordet 00003077.

Oktala instruktioner

Oktala ord kan ingå som instruktioner i DAC-programmet. Den oktala instruktionen kan skrivas med utelämnande av nollor i början av ordet. Den åtskiljes från andra instruktioner på samma sätt som en DAC-instruktion.

Exempel:
```
01 4 10537 ;
     6321 ;
```
innebär de båda instruktionerna
```
01 4 10537
00 0 06321
```

Oktala ord i deklarationer

Genom en deklaration kan oktala ord namnges. Det oktala ordet kan skrivas med utelämnande av nollor i början av ordet.

Exempel: 643201

innebär det oktala ordet 00643201.

Strängar

En sträng utgöres av en godtycklig följd av tecken och kan ingå i vissa deklarationer i DAC (kapitel 7). En sträng kan innehålla alla de tecken och symboler, som är representerade i hålremskoden, med undantag av tecknen semikolon och understrykning. Mellanslag, vagnretur, tabulator, stoppkod och utplåning representeras på det sätt, som angivits under "Symboler" i detta kapitel. Om tecknen semikolon och understrykning skall ingå i strängen, måste symbolerna SC respektive UL användas för dessa tecken. I en REGISTER-deklaration kan dessutom tecknet kolon ej ingå i strängen, utan måste representeras genom symbolen CO.

Exempel:
```
TEXT: UL F UL e UL l SP parametrar;
```
Strängen är
```
Fel parametrar
```
och ingår i en TEXT-deklaration.

En sträng lagras med tre tecken per D21-ord och upptar alltid ett helt antal ord.

Kommentarer

Kommentarer kan skrivas i ett DAC-program. De skrives efter COMMENT: och kan innehålla alla tecken med undantag av semikolon, som utgör avslutningstecken för kommentaren. Kommentarer ignoreras helt av DAC-översättaren vid inläsningen till D21.

Exempel:
```
COMMENT: Programblock för beräkning av
         Tmax;
```

0128-0101-0:11/12/13

Kapitel 3

DAC-programmets struktur

I ett DAC-program ingår deklarationer och instruktioner. Dessa kan sammanfogas till större enheter kallade block. Block kan i sin tur ingå i block av högre nivå.

Avslutningstecken

Semikolon (;) användes genomgående som avslutningstecken för en deklaration och en instruktion.

Deklarationer

Deklarationer användes i DAC för följande funktioner:

1 För att ge DAC-översättaren upplysningar om vilka variabler, som införes i ett block.

2 För namngivning av numeriska konstanter i ett block.

3 För att i ett program införa icke-numerisk information eller reservera plats härför.

4 För att eventuellt ge DAC-översättaren upplysningar om hur vissa i DAC ingående underprogram önskas utplacerade i blocken.

Deklarationer skrives på följande sätt:

Deklarationsnamnet, d v s namnet på deklarationstypen understrykes och avslutas med kolon.

Exempel:

REAL :
LIBRARY:

Den eller de storheter, som avses att deklareras, följer därefter, varefter deklarationen avslutas med semikolon. Om mer än en storhet deklareras, åtskiljas de olika storheterna genom komma.

Exempel: INTEGER: N;
REAL : A, B, C;

I de fall det i en storhet ingår två begrepp åtskiljas dessa genom kolon. Detta är exempelvis fallet vid namngivning av konstanter, där dels namnet och dels numeriska värdet ingår.

Exempel: REAL: A, B4: 1.93, nz;

B4 är en konstant med värdet 1.93.

I ett block kan samma deklarationstyp förekomma mer än en gång. Ett namn får däremot ej deklareras mer än på ett ställe bland alla de deklarationer, som ingår i blocket. Deklarationerna redovisas i kapitel 5, 7 och 8.

Instruktioner

En DAC-instruktion består i allmänhet av en operationsdel (OP) och en adressdel (AS). Den skrives på följande sätt:

OP, AS;

Operationsdelen skrives först och åtskiljes från adressdelen genom komma. Semikolon avslutar instruktionen. Man kan i en DAC-instruktion utelämna operationsdelen eller adressdelen eller skriva en helt tom instruktion.

, AS;
OP , ;
;

Vid översättningen utlägges 00 (oktalt) för utelämnad operationsdel, 00000 för utelämnad adressdel och 00000000 för en tom instruktion.

Vissa DAC-instruktioner behöver för sin funktion ingen adressdel och skrives därvid

OP ;

En DAC-instruktion har högst en adressdel. Då en operation fordrar flera adressuppgifter, anges dessa i en följd av DAC-instruktioner.

Exempelvis beordras framspolning av M block på magnetbandsaggregat N på följande sätt:

```
FORW TAPE , N;
AS TO MR  , M;
```

En DAC-instruktion kan antingen ha en direkt motsvarighet i en maskininstruktion eller vara en pseudoinstruktion, som ger upphov till att flera maskininstruktioner genomlöpes. För en sådan instruktion kommer en eller flera maskininstruktioner att utläggas på motsvarande plats i maskinprogrammet. I de fall pseudoinstruktionen utföres genom ett hopp till ett underprogram kommer detta att utläggas i det bildade maskinprogrammet genom DAC-översättarens försorg. Vilken typ av instruktion en DAC-instruktion utgör framgår ej av operationsbeteckningen, och båda typerna är ur programmeringssynpunkt i de flesta avseenden ekvivalenta.

I DAC-programmet kan också oktalt skrivna maskininstruktioner tillfogas. Mera härom se "Oktala instruktioner" i kapitel 2.

Operationsdelen av DAC-instruktionen

En fullständig redovisning kommer att ges för de i DAC 1 använda operationsbeteckningarna och utgör i och för sig fullt tillräcklig information. Det kan dock vara av intresse att något redogöra för strukturen i operationsdelen.

För operationsbeteckningar i DAC kan bokstäver, siffror och andra tecken användas. Undantagna är kolon, semikolon, komma, understrykning och parenteser. Dessutom kan en operationsbeteckning ej börja på en siffra.

En operationsbeteckning kan innehålla ett godtyckligt antal tecken. I DAC 1 är endast de nio första tecknen identifierande och överskjutande ignoreras, men denna begränsning kommer att avlägsnas, då instruktionslistan senare utökas.

Ett flertal av DAC-instruktionerna för aritmetik och överföring till och från snabbminnet är av sådan typ, att de direkt arbetar på ackumulatorregistret eller multiplikatorregistret. I de fall instruktionen endast berör multiplikatorregistret, ingår förkortningen MR i operationsbeteckningen, medan de instruktioner, som saknar registerbeteckning förutsättes arbeta på ackumulatorregistret. För flytande räkning användes ett pseudoackumulatorregister, vilket på analogt sätt ej anges i operationsbeteckningen.

För vissa av de vanligaste instruktionerna har i DAC använts ett system med prefix och suffix för att utmärka olika varianter av samma instruktionstyp. Exempelvis användes + för att beteckna addition med fast komma, enkel precision. Dessutom finns andra additionsinstruktioner, nämligen addition med fast komma, dubbel precision och addition med flytande komma. Dessa instruktioner skrives D+ respektive F+. Detta beteckningssystem användes genomgående för att utmärka dubbel precision respektive flytande räkning. Ett annat exempel på prefix utgöres av L, som användes för s k långa operationer.

Exempel: L* , BETA;

betecknar den multiplikation, som ger en produkt av dubbellängd vid multiplikation av två tal i enkel precision. Om multiplikationen skulle utförts på två tal i dubbel precision och givit ett resultat i dubbel precision, skulle operationsbeteckningen D* använts.

För att utmärka registertömning, d v s nollställning, användes C (Clear). Beroende på om tömningen sker i början eller slutet av operationen sättes C som prefix respektive suffix.

```
Exempel:   C+   ,   GAMMA;
           MR=C,    DELTA ;
```

C+ betyder töm och addera till ackumulatorregistret AR, d v s det aktuella numeriska värdet av GAMMA hämtas till AR. MR=C innebär, att innehållet i multiplikatorregistret MR överföres till DELTA, varefter MR nollställes.

Adressdelen av DAC-instruktionen

I adressdelen av en DAC-instruktion kan förutom namn på variabler, konstanter och platser i programmet även tal förekomma. Dessutom kan även oktala adresser skrivas i adressdelen.

Vissa storheter, som namnges i DAC, kommer då de skall utläggas i snabbminnet att uppta mer

0128-0101-0:15/16/17/18

än en cell, exempelvis utlägges tal med flytande komma i två konsekutiva celler. I sådana fall kommer namnet att svara mot den första cellen i följden. Om man refererar till en sådan storhet med en instruktion, som opererar på ett ord, kommer endast den första cellen att beröras.

Exempel: C+, GAMMA;

Om GAMMA är deklarerad som reell, kommer denna instruktion att medföra, att endast det första ordet av de två, som GAMMA upptar, överföres till ackumulatorregistret. Om motsvarande instruktion för reell aritmetik användes erhålles korrekt resultat.

Då tal skrives i adressdelen av en DAC-instruktion, skrives dessa så, som anges i kapitel 2.

Exempel:

Talen 6, 0.95 (M), 0.95 (DM), $6.95_{10}{}^{-3}$ och 76543210 (OCT) utgör heltal, maskintal i enkel precision, maskintal i dubbel precision, tal med flytande komma och oktalt tal respektive.

Oktala adresser i en instruktion skrives så som anges i kapitel 2.

Exempel: +, 21630 (OAS) ;

Namngivning

Då man i en DAC-instruktion önskar referera till en deklaration eller en plats i programmet, d v s en annan instruktion, måste detta ske genom att deklarationen eller platsen i programmet åsättes ett namn. Detta sker genom att namnet sättes före deklarationen eller programplatsen och åtskiljes från denna genom kolon.

Exempel:

Normaltypografi: TEXT : 2 SP 4.2 ;
L4: C+, GAMMA;

Namnen är här Normaltypografi respektive L4. För namngivning användes namn enligt definitionen i kapitel 2. Ett godtyckligt antal tecken kan ingå i namnet, men endast de nio första är identifierande och överskjutande ignoreras.

En deklaration eller programplats kan erhålla mer än ett namn.

Exempel: F7: Felutskrift: TEXT: CR Fel
SP parameterremsa;
ABC: K3: t: MR=C, DELTA;

Samma namn får däremot ej förekomma mer än på en plats i programmet eller på en deklaration. Vid namngivning av en pseudoinstruktion, som ger upphov till att mer än en maskininstruktion utlägges i programmet, kommer namnet att åsättas den första av de utlagda instruktionerna. Detta medför, att exempelvis adressinsättning ej kan ske i en pseudoinstruktion, som vid utläggningen ej erhåller adressen i den första maskininstruktionen. I DAC förekommer endast ett fåtal instruktioner av denna typ ("Aritmetik med fast komma, dubbel precision" kapitel 5), och i sådana fall kan adressinsättning ske till en plats i programmet, som sedan indirekt adresseras av pseudoinstruktionen. Av instruktionslistan framgår, vilka dessa instruktioner är. För sådana pseudoinstruktioner, som i DAC skrives som mer än en instruktion, kan namngivning direkt ske av varje delinstruktion.

Exempel: A1 : FORW TAPE, N;
DEF : AS TO MR , M;

Vektorer

Det är i DAC möjligt att räkna med indicerade storheter med ett index. Därvid kan index vara ett heltal eller en heltalsvariabel.

Indicerade storheter namnges enligt de regler, som anges för namn i kapitel 2. Index sättes efter namnet inom klammer.

Exempel: C [3]
a7T [k]

Vektorerna deklareras genom en ARRAY-deklaration, varvid undre och övre indexgräns numeriskt specificeras (kapitel 5).

Då en vektorkomponent med numeriskt index ingår i en DAC-instruktion, kommer vid översättningen inga extra maskininstruktioner att utläggas härför. Detta är emellertid fallet, då index är variabelt. Beroende på typen av vektor kommer 5 eller

0128-0101-0:18/19/20/21

6 maskininstruktioner att utläggas. Eventuell spillindikation, som förefinns före utförandet av instruktionen, kommer därvid att gå förlorad. Spillindikation, som däremot uppkommer som ett resultat av instruktionen, kvarstår emellertid.

Adressinsättning kan ej ske av ett indicerat namn, där index ej är numeriskt givet. Detta kan ej heller ske till en instruktion, som redan innehåller ett indicerat namn med variabelt index.

Räkning med indicerade storheter kan ske både för det fall DAC-instruktionen motsvaras av en direkt maskininstruktion eller en pseudoinstruktion.

Exempel:
```
+, C [3]    ;
F+, a7T [k] ;
```

Indirekt adressering

I D21 förekommer två former av indirekt adressering, dels med och dels utan framstegning med ett. Detta utmärkes i DAC genom att adressdelen av instruktionen sättes inom olika typer av parenteser. Vid indirekt adressering med framstegning med ett användes parenteserna [ ].

Exempel:
```
C+, [ALFA] ;
```

Sker den indirekta adresseringen utan framstegning användes parenteserna ( ).

Exempel:
```
+, (B);
```

Om platsen B i programmet innehåller DELTA, d v s

```
B: , DELTA;
```

blir alltså effekten av instruktionen densamma som

```
+ , DELTA;
```

Block

Ett block måste alltid namnges i DAC, varvid endast ett namn får förekomma. Vid namngivningen kan högst två tecken användas, varvid det första måste vara en bokstav och det följande en bokstav eller siffra.

I ett block skrives deklarationerna först, varefter instruktionerna följer. Ett block kan innehålla enbart deklarationer eller instruktioner. Mera härom se kapitel 8.

Blockbörjan anges genom BEGIN;.

Exempel:
```
A1: BEGIN;
```

Blockslut anges genom END varefter följer blocknamnet. Detta åtskiljes från END genom komma.

Exempel:
```
END, A1;
```

Om ett block skall ingå i ett annat block måste det helt ligga inom detta block.

Exempel:
```
A: BEGIN ;
   - - - -
B: BEGIN ;
   - - - -
   END, B;
   - - - -
   END, A;
```

Två block får däremot ej delvis ingå i varandra. Man får således ej skriva:

```
A: BEGIN ;
   - - - -
B: BEGIN ;
   - - - -
   END, A;
   - - - -
   END, B;
   - - - -
```

Mer än ett block kan ingå i ett annat block.

Exempel:
```
A: BEGIN ;
   - - - -
B: BEGIN ;
   - - - -
   END, B;
   - - - -
C: BEGIN ;
   - - - -
D: BEGIN ;
   - - - -
   END, D;
   - - - -
   END, C;
   - - - -
   END, A;
```


0128-0101-0:21/22/23

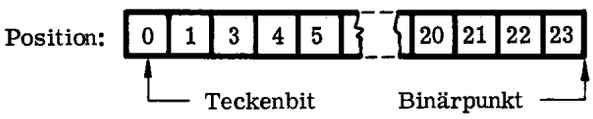
Kapitel 4

Tal- och teckenrepresentation

I DAC ingår olika typer av tal, som i D21 kommer att representeras på olika sätt.

Heltal

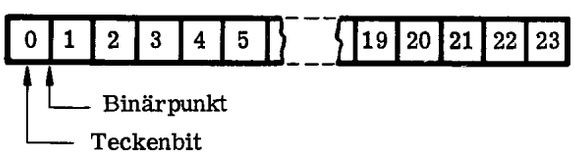
Ett heltal upptar alltid ett D21-ord och representeras på följande sätt:



Då det i DAC 1 ej ingår särskilda instruktioner för heltalsmultiplikation och division, måste instruktionerna för fast komma användas. Detta medför, att binärpunktens läge måste justeras genom skiftinstruktioner.

Maskintal, enkel precision

Ett maskintal i enkel precision upptar alltid ett D21-ord och representeras på följande sätt:



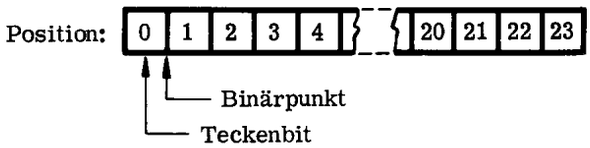
Talet måste alltid uppfylla relationen

$$-1 \leqslant \text{Talet} \leqslant 1-2^{-23}$$

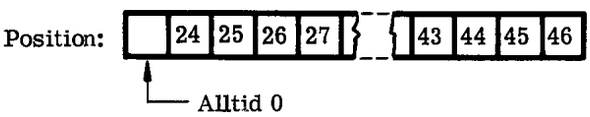
Maskintal, dubbel precision

Ett maskintal i dubbel precision upptar alltid 2 konsekutiva D21-ord och representeras på följande sätt:

Ord 1.



Ord 2.



Talet måste alltid uppfylla relationen

$$-1 \leqslant \text{Talet} \leqslant 1-2^{-46}.$$

Reella tal

Ett tal, som deklareras reellt, representeras med flytande komma. Detta innebär, att ett tal $x \neq 0$ erhåller representationen

$$x = M \cdot 2^{e+128}$$

där mantissa M uppfyller relationen

$$1/2 \leqslant M \leqslant 1-2^{-39}$$

om talet är positivt och

$$-1 \leqslant M \leqslant -1/2-2^{-39}$$

om talet är negativt.

I stället för exponenten e användes en karakteristik k där

$$k = e+128$$

Karakteristiken måste uppfylla relationen

$$0 \leqslant k \leqslant 255$$

Talet 0 representeras med både mantissa och karakteristik = 0.

Ett reellt tal upptar alltid två konsekutiva D21-ord, varvid mantissan representeras med 40 bitar, inklusive teckenbit, och karakteristiken med 8 bitar. I vissa fall kan aritmetik med flytande komma ske med 24 bitars mantissa. Detta innebär, att de 16 minst signifikanta bitarna ignoreras. I D21 representeras ett tal med flytande komma på följande sätt:

Ord 1.

Ord 2.

Alfanumeriska tecken

Alfanumerisk information (strängar) lagras med 3 tecken per ord, där varje tecken upptar 7 bitar. Resterande 3 bitar användes för information om innehållet i ordet. Tecknen representeras på samma sätt som i hålremskoden, med de 6 kanaler, som kvarstår då paritetskanalen och kanalen för vagnretur borttagits. Dessutom har en bit tillfogats för att markera nedre eller övre läge. För vagnretur användes ett speciellt tecken.

0128-0101-0:25

Kapitel 5

Aritmetik och administration

Deklaration av konstanter och variabler

Heltal

En variabel deklareras som heltal och en heltalskonstant namnges genom en INTEGER-deklaration.

Exempel:
```
INTEGER : n;
INTEGER : i, j:16, k:-8, M;
```

n, i och M är variabler medan j och k är konstanter.

Maskintal, enkel precision

En variabel deklareras som ett maskintal i enkel precision och en konstant namnges genom en FIXED-deklaration. Det tal, som anger det numeriska värdet av konstanten, skrives utan att (M) tillägges efter talet, vilket var fallet, om det förekom i adressdelen av en DAC-instruktion (kapitel 2).

Exempel:
```
FIXED: A;
FIXED: Ne:-1, Tx, H7:0.987;
```

Maskintal, dubbel precision

Maskintal i dubbel precision deklareras genom en DFIXED-deklaration. För tal gäller att (DM) utelämnas efter talet i deklarationen.

Exempel:
```
DFIXED: B;
DFIXED: A1, A2, A3:-0.1234_10^-5;
```

Reella tal

En variabel deklarareras som reell och en konstant namnges genom en REAL-deklaration. Talen representeras därvid med flytande komma.

Exempel:
```
REAL: C;
REAL: X, p:1.936, x;
```

Oktal information

Införande av oktala variabler och namngivning av oktala konstanter sker genom en OCTAL-deklaration. Ett oktalt begrepp upptar alltid ett D21-ord. Värdet av konstanter skrives utan att (OCT) tillägges efter. Nollor i början av ordet kan utelämnas.

Exempel:
```
OCTAL: Q;
OCTAL: L:7654 3210, M, N: 123;
```

Deklaration av vektorer

Genom ARRAY-deklarationen definieras undre och övre indexgränserna för de indicerade variablerna eller konstanterna. I det fall vektorkomponeterna är konstanter angives numeriska värdet av dem i deklarationen.

Indexgränserna skrives efter namnet och sättes inom parenteserna [ ]. Undre indexgräns skrives

först, varefter övre gränsen följer, åtskild genom kolon. Indexgränserna måste alltid vara heltal.

Exempel: REAL ARRAY: A [1:8] , B [-2:2]

Vektorn A innehåller 8 komponenter, medan vektorn B har 5 komponenter.

För att specificera talrepresentationen hos vektorkomponenterna föregås ARRAY av ett typnamn, som är analogt med det, som användes i de ovan nämnda deklarationerna. Följande vektortyper kan förekomma:

INTEGER ARRAY:
FIXED ARRAY :
DFIXED ARRAY :
REAL ARRAY :
OCTAL ARRAY :

I det fall vektorkomponenterna är konstanter angives deras numeriska värden så, som framgår av följande exempel:

```
REAL ARRAY: A   [1:8]   , T1   [0:2] : -1,
                0.93, 3.76, B   [-2:2] ;
```

Här har alltså vektorn T1 konstanta komponenter, och värdet av dem skrives som en följd av tal, åtskilda genom komma och åtskilda från gränsdefinitionen genom kolon. Det första talet svarar mot den undre indexgränsen, det därpå följande mot nästa index, o s v. Antalet tal måste överensstämma med det antal vektorkomponenter, som angives genom deklarationen.

Om ett variabelt index under en beräkning antar ett värde, som ligger utanför det indexintervall, som angives genom deklarationen, kommer detta att passera opåtalat.

I DAC 1 ingår ej aritmetik med vektorer utan endast med vektorkomponenter. Ett vektornamn kan därför ej förekomma i en instruktion utan att index samtidigt angives, antingen numeriskt eller som variabelnamn (se "Vektorer" kapitel 3). Då en vektor deklareras i ett block, får den ej ha samma namn som en annan storhet, som deklareras, eller på annat sätt införes i blocket, exempelvis namn på en plats i blocket.

Instruktioner

I instruktionslistan ingår såväl instruktioner med direkt motsvarighet i maskininstruktioner som pseudoinstruktioner. För maskininstruktionerna anges i de följande instruktionslistorna den oktala koden i kolumnen "Utlägges". För ytterligare information om dessa instruktioner hänvisas till systembeskrivningen för datamaskinsystemet SAAB D21.

I samma kolumn redovisas också hur en pseudoinstruktion utlägges i programmet. Härvid har det valts att redovisa denna utläggning i DAC-form, men vid översättningen kommer naturligtvis motsvarande maskininstruktioner att utläggas. I det fall pseudoinstruktionen utföres genom hopp till underprogram har det relevanta underprogrammet kallats P.

Då DAC-instruktionen skall ha en adressdel har denna redovisats, medan ingen sådan förekommer, då sådan ej ingår i instruktionen. För adressdelar har två beteckningar använts:

A anger att adressdelen är ett namn
N anger att adressdelen är ett heltal.

Härmed menas, att DAC-instruktionen i sin grundform har en adressdel an denna typ. För aritmetiska instruktioner kan ju ett tal mycket väl anges i adressposition. I de fall adressdelen för grundformen är ett tal, kan genom indirekt adressering ett namn stå i adressposition.

```
Exempel:    ←, (B) ;
            - - - - -
            B: , 3 ;
```

Denna instruktion ger vänsterskift 3 steg. Alla instruktioner kan indirektadresseras, såvida annat ej särskilt angives.

För pseudoinstruktioner gäller, att tider endast är approximativt angivna.

0128-0101-0:27/28/29

Förekommande förkortningar

AR	= ackumulatorregistret
ar	= innehållet i AR
AR_{0-11}	= ackumulatorregistrets positioner 0-11
ar_{0-11}	= innehållet i AR_{0-11}
MR	= multiplikatorregistret
mr	= innehållet i MR
ARMR	= det sammanhängande 47 bitars registret bestående av AR och MR
armr	= innehållet i ARMR
PAR	= det pseudoackumulatorregister, som användes vid flytande räkning. PAR ingår i DAC-basprogrammet för flytande räkning.
par	= innehållet i PAR
A	= namn
a	= A:s numeriska värde
A+1	= den cell i snabbminnet, som följer efter cellen, som motsvaras av platsen A i programmet
(a, a+1)	= numeriska värdet av talet i A och A+1 med hänsyn tagen till representationssättet
Irr	= namn på för programmeraren irrelevant plats i DAC-basprogrammet
B	= namn på plats i DAC-basprogrammet
P	= namn på ett underprogram, som är relevant i sammanhanget
N	= decimalt heltal

0128-0101-0:30

Aritmetik med fast komma, enkel precision

Instruktion		Verkan	Tid i μs	Utlägges
C+,	A	a till AR	9.6	01
C-,	A	-a till AR	9.6	41
+ ,	A	ar+a till AR	9.6	00
- ,	A	ar-a till AR	9.6	40
* ,	A	ar · a till AR avrundat mr förstöres	35.2-40.8	04
/ ,	A	$\frac{ar}{a}$ till AR Rest i MR	44.0	05
= ,	A	ar till A och kvar i AR	9.6	06
=C,	A	ar till A, 0 till AR	9.6	46
+=,	A	ar+a till A och AR	12.0	07
+=C,	A	ar+a till A, 0 till AR	12.0	47
L*,	A	ar · a till ARMR	35.2-40.8	44
L/,	A	$\frac{armr}{a}$ till AR Rest i MR	44.0	45

0128-0101-0:31

Aritmetik med fast komma, dubbel precision

Instruktion	Verkan	Tid i μs	Utlägges
DC+ , A	(a,a+1) till ARMR	35.2	=C , Irr MR=C, Irr D+ , A
DC- , A	-(a,a+1) till ARMR	35.2	=C , Irr MR=C, Irr D- , A
D+ , A	armr+(a,a+1) till ARMR	16.0	02
D- , A	armr-(a,a+1) till ARMR	16.0	42
D* , A	armr · (a,a+1) till ARMR	300	GO TO PR, P AS TO MR, A
D/ , A	$\frac{\text{armr}}{(a,a+1)}$ till ARMR	440	GO TO PR, P AS TO MR, A
D= , A	armr till A, A+1 och kvar i ARMR	96	GO TO PR, P AS TO MR, A

Adressinsättning kan ej ske i instruktionerna DC+, DC-, D*, D/ och D=. I fall där detta måste ske, kan adressinsättningen göras till en plats i programmet, som indirekt adresseras i instruktionen.

Exempel:
```
D= , (B) ;
- - - -
- - - -
B: ,     ;
```

Adressinsättningen göres till B.

Aritmetik med flytande komma

Instruktion	Verkan	Tid i μs	Utlägges
FC+ , A	(a,a+1) till PAR	115	AS TO MR, A GO TO PR, P
FC- , A	-(a,a+1) till PAR	135	AS TO MR, A GO TO PR, P
F + , A	par+(a,a+1) till PAR	265	AS TO MR, A GO TO PR, P
F - , A	par-(a,a+1) till PAR	280	AS TO MR, A GO TO PR, P
F* , A	par $\cdot$ (a,a+1) till PAR	430	AS TO MR, A GO TO PR, P
F / , A	$\frac{\text{par}}{(a,a+1)}$ till PAR	460	AS TO MR, A GO TO PR, P
F = , A	par till A, A+1 och kvar i PAR	142	AS TO MR, A GO TO PR, P

Aritmetiken utföres med 40 bitars mantissa.

För flytande aritmetik med 24 bitars mantissa hänvisas till underprogram för detta ändamål.

0128-0101-0:33

Allmänna administrativa instruktioner

Instruktion	Verkan	Tid i μs	Utlägges
AS= , A	ar_{8-23} överföres till adressdelen för A	9.6	10
MAS= , A	ar_{6-23} överföres till märk- och adressdelen för A	9.6	50
LEFT= , A	ar_{0-11} överföres till motsvarande positioner i A	9.6	11
RIGHT= , A	ar_{12-23} överföres till motsvarande positioner i A	9.6	51
MR= , A	mr till A och kvar i MR	9,6	12
MR=C , A	mr till A, 0 till MR	9,6	52
C+MR , A	a till MR	9,6	13
AS TO MR , A	effektiv adressdel, dvs den adressdel, som kvarstår, då alla led av indirekt adressering genomlöpts, till MR. Op.delen AS TO MR, dvs oktalt 53, medföljer även till MR.	9.6	53
$\wedge$ eller , A	ar $\wedge$ a till AR, dvs ar multipliceras logiskt med a	9.6	14
AND , A		————	
← , N	ar vänsterskiftas N steg	7.2 om N$\leqslant$4	15
L ← , N	armr vänsterskiftas N steg	7.2+ 0.8 (n-4) om N>4	55
→ , N	ar högerskiftas N steg. Teckenreproduktion		16
L → , N	armr högerskiftas N steg. Teckenreproduktion	————	56
C → , N	ar högerskiftas N steg. 0 inmatas i AR_0	7.2 om N$\leqslant$3	17
LC → , N	armr högerskiftas N steg. 0 inmatas i AR_0	7.2+ 0.8 (N-3) om N>3	57
NORM , A	ar normaliseras Antal skiftsteg till A	———— 10.4+ N · 0.8	20
LNORM , A	armr normaliseras Antal skiftsteg till A		60

Instruktion	Verkan	Tid i μs	Utlägges
+1 , A	a+1 (heltal) till A pos 9-23 Tecknet till teckenindikatorn I	9.6	21
GO TO , A	Hoppa till A	4.8	22
I GO TO , A	Hoppa till A om teckenindikatorn I är negativ. Annars fortsättning med nästa instruktion.	4.8 om hopp sker, 6.4 om hopp ej sker	62
+ GO TO , A	Hoppa till A om AR $\geqslant$ 0. Annars fortsättning med nästa instruktion.		23
- GO TO , A	Hoppa till A om ar $<$ 0. Annars fortsättning med nästa instruktion.		63
OV GO TO , A	Hoppa till A om spill föreligger. Annars fortsättning med nästa instruktion.		24
NOV GO TO, A	Hoppa till A om ej spill föreligger. Om spill föreligger skiftas armr 1 steg höger med ar_{00} till AR_0 och fortsättes med nästa instruktion.		64
GO TO PR , A	Om denna instruktion har namnet L kommer L+1 att överföras till A:s adressposition, varefter hopp sker till A+1. AR och MR beröres ej av instruktionen.	9.6	26
DO , A	Instruktionen med namnet A utföres	4.8+ tiden för instruktionen	27

Då instruktionen I GO TO användes måste den följa omedelbart efter den instruktion, som medför att tecknet placeras i teckenindikatorn I.

0128-0101-0:35

Administrativa instruktioner i samband med flytande räkning

Instruktion	Verkan	Tid i μs	Utlägges
F+GO TO , A	Hoppa till A om par $\geqslant 0$. Annars fortsättning med nästa instruktion.	38.4 eller 40.0	AS TO MR , A MR = C , Irr C + , B + GOTO , (Irr)
F-GO TO , A	Hoppa till A om par < 0. Annars fortsättning med nästa instruktion.		AS TO MR , A MR = C , Irr C + , B - GO TO , (Irr)
IF $\geqslant$, A	Om par $\geqslant$(a,a+1) utföres den efterföljande instruktionen, om ej, den därpå följande. Den närmast följande instruktionen måste vara av sådan typ att endast en maskininstruktion utlägges. Par påverkas ej av instruktionen	60-240	AS TO MR, A GO TO PR, P
IF $<$, A	Om par $<$ (a,a+1) utföres den efterföljande instruktionen, om ej, den därpå följande. Samma villkor i övrigt som för IF $\geqslant$	60-240	AS TO MR, A GO TO PR, P

Stoppinstruktioner

Instruktion	Verkan	Utlägges
STOP , A	Stopp. Vid start sker hopp till A.	25
HOLD	Då alla yttre enheter avslutat beordrad verksamhet, fortsättes med nästa instruktion.	GO TO PR, P

Eftersom yttre enheter arbetar självständigt i förhållande till centralenheten, medför ett stopp i denna, att informationsöverföring till och från en yttre enhet, som påbörjat sin verksamhet, ej kan äga rum. Information kan på så sätt gå förlorad, och vid start kommer programmet ej att fungera på avsett sätt. Vid programmering bör därför HOLD-instruktionen skrivas före en STOP-instruktion.

Samma är förhållandet vid manuellt stopp av D21. För att erhålla samma manuella funktion som HOLD-instruktionen plus STOP-instruktionen användes decimaltablåregistret (DTR), i vilket talet -.00000 insättes. Detta tal är reserverat för denna funktion.

0128-0101-0:36/37

Kapitel 6

Konvertering och matematiska funktioner

Beteckningarna i detta kapitel är analoga med dem i kapitel 5, varför läsaren hänvisas till teckenförklaringen där.

Angivna tider är approximativa genomsnittstider.

Instruktioner för konvertering mellan olika talrepresentationer

Instruktion	Verkan	Tid i μs	Utlägges
ENTIER	Heltalsdelen av par till ARMR som heltal	115	GO TO PR, P
REAL TO FIXED	par till ARMR som maskintal	100	GO TO PR, P
INT TO REAL	Heltal i ARMR till PAR som flytande tal	135	GO TO PR, P
FIXED TO REAL	Maskintal i ARMR till PAR som flytande tal	120	GO TO PR, P

Heltalsdelen av x definieras som det största heltal, som ej är större än x.

0128-0101-0:37

Funktioner för tal med fast komma

Instruktion	Verkan	Tid i μs	Utlägges
ABS	|armr| till ARMR	6.4 eller 11.2	03
-ABS	-|armr| till ARMR		43
SQRT	$\sqrt{ar}$ till AR	400	GO TO PR, P
DSQRT	$\sqrt{armr}$ till ARMR	585	GO TO PR, P
SIN	sin(ar) till AR Vinkeln anges i varv	340	GO TO PR, P
DSIN	sin(armr) till ARMR Vinkeln anges i varv	1500	GO TO PR, P
COS	cos(ar) till AR Vinkeln anges i varv	370	GO TO PR, P
DCOS	cos(armr) till ARMR Vinkeln anges i varv	1475	GO TO PR, P
ARCTAN	arctg(ar) till AR Vinkeln erhålles i radianer	405	GO TO PR, P
DARCTAN	arctg(armr) till ARMR Vinkeln erhålles i radianer	1440	GO TO PR, P
EXP	e^{ar} till AR	275	GO TO PR, P
DEXP	e^{armr} till ARMR	1370	GO TO PR, P

0128-0101-0:38

Funktioner för tal med flytande komma

De matematiska funktionerna kan dels beräknas med 40 bitars mantissa och dels med 24 bitars mantissa. I det förra fallet börjar funktionsnamnet alltid med F och i det senare fallet med SF.

Instruktion	Verkan	Tid i μs	Utlägges
FABS	|par| till PAR	65	GO TO PR, P
-FABS	-|par| till PAR	60	GO TO PR, P
FSIGN	Tecknet av par till PAR, dvs -1 till PAR om par $<$ 0 0 till PAR om par =0 +1 till PAR om par $>$ 0	95	GO TO PR, P
FSQRT	$\sqrt{par}$ till PAR	560	GO TO PR, P
SFSQRT	$\sqrt{par}$ till PAR	400	GO TO PR, P
FSIN	sin(par) till PAR Vinkeln i radianer	1875	GO TO PR, P
SFSIN	sin(par) till PAR Vinkeln i radianer	500	GO TO PR, P
FCOS	cos(par) till PAR Vinkeln i radianer	1875	GO TO PR, P
SFCOS	cos (par) till PAR Vinkeln i radianer	545	GO TO PR, P
FARCTAN	arctg(par) till PAR Vinkeln i radianer	1740	GO TO PR, P
SFARCTAN	arctg(par) till PAR Vinkeln i radianer	590	GO TO PR, P
FLN	$^{e}\log$ (par) till PAR	1005	GO TO PR, P
SFLN	$^{e}\log$ (par) till PAR	650	GO TO PR, P
FEXP	e^{par} till PAR	1320	GO TO PR, P
SFEXP	e^{par} till PAR	415	GO TO PR, P
FENTIER	Heltalsdelen av par till PAR	150	GO TO PR, P

0128-0101-0:39

Kapitel 7

In- och utmatning

Genom att de långsamma yttre enheterna i D21 kan arbeta samtidigt med att annan verksamhet pågår i centralenheten kommer pseudoinstruktionerna för in- och utmatning i DAC ej att slutföras innan nästa instruktion i programmet påbörjas. Dessa pseudoinstruktioner, som är utformade som underprogram, utföres i stället efter följande filosofi:

Då det i underprogrammet uppkommer ett läge, som skulle föranleda väntan på att den yttre enheten avslutat tidigare beordrad verksamhet, kommer denna väntetid att utnyttjas för fortsatta beräkningar i huvudprogrammet. När den yttre enheten senare blir klar, sker återhopp till underprogrammet, då ett programavbrott erhålles. Underprogrammet genomlöpes tills ett nytt vänteläge uppkommer, o s v. På detta sätt förfares tills den genom pseudoinstruktionen beordrade verksamheten är avslutad, varvid information härom lämnas till huvudprogrammet. De yttre enheter, som beskrives i detta kapitel och arbetar efter denna metodik, är remsläsare, remsstans, skrivmaskin och bandminne.

Om man för det fortsatta genomloppet av huvudprogrammet är beroende av, att en pseudoinstruktion för en yttre enhet blivit utförd, skrives en frågeinstruktion

```
READY, L;
```

L anger här namnet på den plats i programmet, där pseudoinstruktionen finns. READY-instruktionen ger upphov till att den efterföljande instruktionen ej genomlöpes, förrän den angivna pseudoinstruktionen är helt utförd.

Exempel:

```
        - - - - -
    L: READ  , A;
       - - - - -
       - - - - -
       READY , L;
    B: - - - - -
```

READ-instruktionen innebär maskintalsinläsning från hålremsa. Om man vid platsen B behöver använda det eller de till A inlästa talen, måste READY, L skrivas omedelbart före.

READY-instruktionen medför, att två instruktioner utlägges i maskinprogrammet, som skrivna i DAC-form, är

```
T: C+  , L+2;
  +GO TO, T;
```

L+2 är det tredje ordet av dem, som utlägges för pseudoinstruktionen READ, A. Då pseudoinstruktionen påbörjats, sättes 0 i detta ord, och då den är avslutad insättes -1.

Om en yttre enhet beordras genom en pseudoinstruktion utan att en föregående instruktion avslutats, kommer den senast beordrade instruktionen ej att påbörjas förrän den föregående avslutats. Under väntetiden härpå kommer inget fortsatt genomlopp av programmet att äga rum. Pseudoinstruktioner, som beordrats vid andra yttre enheter, utföres dock i enlighet med ovan angiven metodik vid varje programavbrott.

I DAC-programmet kan samtidigt följande funktioner äga rum: inläsning från remsa, utmatning på remsstans eller skrivmaskin, aktivitet vid två bandaggregat samt genomlopp av ett beräkningsprogram.

0128-0101-0:40/41

Deklarationer

Genom en TEXT-deklaration kan alfanumeriska strängar skrivas eller plats reserveras härför i DAC-programmet. Då strängarna anges direkt i DAC-programmet, skrives de enligt de konventioner, som anges i kapitel 2. För inläsning av strängar kan READ TEXT- och READ STRING-instruktionerna användas. Platsreservation göres med hjälp av tecknet BL, föregånget av antalet tecken, för vilka reservation skall göras. Antalet understrykes även.

Exempel: TEXT: CR Fel SP parameterremsa;
TEXT: 17BL;

I en TEXT-deklaration kan endast en sträng skrivas.
För att möjliggöra identifiering av inlästa strängar användes en REGISTER-deklaration tillsammans med en LOOK UP-instruktion. I REGISTER-deklarationen angives en sträng och tillsammans med denna ett namn eller heltal. Om genom en LOOK UP-instruktion (avsnittet "Blandad sträng- och heltalsinläsning" i detta kapitel) en inläst sträng identifierats i en registerdeklaration, erhålles heltalet eller namnet, d v s den plats i programmet, som motsvaras av namnet, som utgång ur LOOK UP-instruktionen. Strängarna i REGISTER-deklarationen skrives enligt konventionerna i kapitel 2 och åtskiljes från det efterföljande namnet eller heltalet genom kolon. I en REGISTER-deklaration kan flera sådana par av strängar och namn eller heltal ingå. De åtskiljes därvid genom komma.

Exempel: REGISTER: SUM: -3, x+7: L1, z:z;

Det z, som förekommer till vänster om kolon, är strängen z, medan det högra är namnet z.

Inläsning från hålremsa

Som standard vid remsinläsning användes 8-kanals remsa. Detta innebär, att då inget annat angives i programmet, sker inläsningen med 8-kanals remsa. Det är emellertid möjligt att även utföra decimal inläsning av telexremsa och 5-kanals talremsa. De decimala inläsningsinstruktionerna är utformade oberoende av remsrepresentation. För att ändra denna användes instruktionen

R- CODE, N;

N är här ett heltal som antar följande värden:

N = 0 Inläsningen sker i forsättningen med telexremsa.
N = 5 Inläsningen sker i fortsättningen med 5-kanals talremsa
N = 8 Inläsningen sker i fortsättningen med 8-kanals remsa.

Instruktionen utlägges

AS TO MR, N;
GO TO PR, P;

Genom denna instruktion är det enkelt att ändra remsrepresentationen. Detta kan göras vid körningstillfället, om man utformar sitt program så, att heltalet N tillföres programmet innan den egentliga inläsningen startas.

Inläsning av strängar

För inläsning av denna typ av information finns två instruktioner.

READ TEXT innebär inläsning och utläggning av en sträng, som innehåller de tecken, som är representerade i hålremskoden för 8-kanals remsa med undantag av tecknen mellanslag, vagnretur, tabulator, stoppkod, utplåning, semikolon och understrykning. Dessa tecken måste representeras genom:

SP	för	mellanslag
CR	för	vagnretur
TAB	för	tabulator
ST	för	stoppkod
TF	för	utplåning
SC	för	semikolon
UL	för	understrykning

Dessa tecken kan även föregås av en antalsangivelse, som även understrykes.

Strängen avslutas med tecknet semikolon, vilket ej lagras upp vid inläsningen. Tecknen mellanslag, vagnretur, tabulator ignoreras vid inläsningen.

0128-0101-0:42/43/44

Instruktionen skrives:

```
READ TEXT, A;
```

Strängen inläses till A och följande.

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
```

Den tomma tredje platsen är den, som användes för READY-funktionen.

READ STRING innebär inläsning och utläggning av en sträng med alla 8-kanals remsans tecken med undantag av semikolon. Inläsning sker fram tills ett semikolon påträffats. Detta semikolon lagras ej i strängen.

Instruktionen skrives:

```
READ STRING, A;
```

Strängen inläses till A och följande.

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
```

Inläsning av heltal

Instruktionen skrives:

```
READ INT, A;
```

Denna instruktion utför inläsning av ett heltal till A.

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
```

Inläsning av maskintal

För inläsning av maskintal finns instruktioner, som dels utför inläsning av ett specificerat antal tal och dels utför inläsning av en talföljd, tills en avslutningssymbol uppträder på remsan. Dessa typer av inläsning kan ske för maskintal i såväl enkel som dubbel precision.

Genom instruktionerna

```
READ , A;
DREAD, A;
```

sker inläsning av det antal tal, som finns angivet i ackumulatorregistret, då instruktionen påbörjas. Inläsningen sker till A och följande platser i programmet. READ medför inläsning av maskintal i enkel precision och DREAD i dubbel precision.

Instruktionerna utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
```

Inläsning av talföljder (appendix 1) tills en avslutningssymbol uppträder på remsan utföres med instruktionerna:

```
READ SEQ , A;
AS TO MR , B;

DREAD SEQ, A;
AS TO MR , B;
```

för maskintal i enkel respektive dubbel precision. Inläsningen sker till A och följande medan antalet tal, som inlästs, angives i B som heltal.

Instruktionerna utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, B;
```

Om kontrollsumma finns på talföljdsremsan, utföres summakontroll, om den saknas sker ingen kontroll.

Inläsning av reella tal

Inläsning av reella tal sker på analogt sätt som maskintalsinläsning, d v s antingen inläsning av ett specificerat antal tal eller en talföljd. Vid inläsningen kommer talen alltid att representeras med 40 bitars mantissa.

Instruktionen FREAD, A;

medför inläsning av det antal tal, som finns angivet i ackumulatorregistret, då instruktionen påbörjas. Inläsningen sker till A och följande platser i programmet.

0128-0101-0:44/45/46/47

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
```

Inläsning av en talföljd tills en avslutningssymbol uppträder på remsan utföres med instruktionen

```
FREAD SEQ, A;
AS TO MR , B;
```

Inläsningen sker till A och följande, medan antalet inlästa tal angives i B som heltal. Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, B;
```

Om kontrollsumma finns på talföljdsremsan, utföres summakontroll, om den saknas sker ingen kontroll.

Summakontroll

Automatisk summakontroll kan endast erhållas vid instruktionerna READ SEQ, DREAD SEQ och FREAD SEQ, varvid eventuella summafel indikeras.

Vid instruktionerna READ INT, READ, DREAD och FREAD sker emellertid en ackumulerande summabildning till en plats i programmet, som kan angivas genom instruktionen

```
SUM TO, A;
```

Summabildningen sker med taldelsumman till A (ett D21-ord) och exponentsumman till A+1 (ett D21-ord). SUM TO-instruktionen medför ej nollställning av dessa platser i programmet, utan detta måste ombesörjas separat i programmet. Om längre fram i programmet instruktionen

```
SUM TO, B;
```

uppträder, sker all summabildning i fortsättningen till B och B+1.

Instruktionen SUM TO, A utlägges:

```
AS TO MR , A;
MR = C   , P;
```

där P är en plats i DAC-basprogrammet. Angives ingen summaplats, sker summeringen till en plats i DAC-basprogrammet.

Inläsningen av kontrollsumma och efterföljande summakontroll måste ombesörjas separat i programmet.

Summakontroll för instruktionerna av READ SEQ-typ sker ej på den plats, som angives genom SUM TO-instruktionen. Angående beräkning av kontrollsumma, se appendix 1.

Blandad sträng- och heltalsinläsning

Genom READ INFORM-instruktionen kan strängar, som ej innehåller enbart siffror och heltal blandat inläsas. Vilken typ av begrepp som inlästs indikeras.

Instruktionen skrives:

```
READ INFORM , A;
```

Om ett heltal inlästs, sättes detta i A+1, d v s den cell som följer efter den, som motsvaras av A, medan A kommer att innehålla 0. Om en sträng inlästs, hamnar denna i A+1 och följande, medan A göres negativt. I strängen kan ingå de tecken, som angives för instruktionen READ TEXT. Vid inläsningen användes dock mellanslag, vagnretur eller tabulator och ej semikolon som avslutningstecken för strängen.

För att reservera plats för den sträng, som skall inläsas, kan TEXT-deklarationen användas. Det är därvid att observera, att antalet tecken i den längsta strängen skall ökas med 3 då ett ord åtgår för att lagra typindikation.

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
```

Med hjälp av en LOOK UP-instruktion kan en inläst sträng identifieras i en REGISTER-deklaration (se "Deklarationer" i detta kapitel). Instruktionen skrives:

```
LOOK UP , A;
AS TO MR, L;
```

A är här samma plats, som angives i inläsningsinstruktionen ovan.

0128-0101-0:47/48/49/50

L är namnet på den REGISTER-deklaration, i vilken sökning skall ske.

Exempel: L: REGISTER: SUM: -3, x+7: L1, z:z;

Om den sträng, som sökes i REGISTER-deklarationen, ej återfinnes i denna, sker hopp till den instruktion, som följer efter AS TO MR, L. Denna måste då vara av sådan typ, att den endast ger upphov till att en maskininstruktion utlägges. Återfinnes strängen däremot, sker hopp till den därpå följande platsen, varvid det till strängen hörande högerledet finns i ackumulatorregistret, dvs antingen ett heltal eller den plats i programmet, som svarar mot namnet.

Exempel: I en programdel ingår bland annat följande:

```
L: REGISTER: SUM: -3, x+7: L1,
   z:z ;
A: TEXT: 6BL ;
         - - - - - -
      P: READ INFORM ,   A;
         - - - - - -
         READY       ,   P;
         LOOK UP     ,   A;
         AS TO MR    ,   L;
         GO TO       , Fel;
      R: - - - - - -
```

Genom READ INFORM-instruktionen inläses strängen x+7. Den efterföljande LOOK UP-instruktionen medför då att hopp sker till platsen R, varvid den mot L1 svarande absolutadressen står i ackumulatorregistret.

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
AS TO MR, L;
```

Utmatning på hålremsstans eller skrivmaskin

Utmatning på remsstans och skrivmaskin kan i D21 ej försiggå samtidigt. Det finns två olika former för stans- och skrivmaskinsutmatning, dels en som ger skrivmaskinsutmatning eller ingen utmatning, beroende på om en knapp (SM-A) på manöverpanelen är nedtryckt eller ej, och dels en annan, som ger utmatning på skrivmaskin eller stans, beroende på hur en av två knappar (SM-B respektive RS) på manöverpanelen är nedtryckt. Den förra utmatningsformen kallas i fortsättningen utmatning på skrivmaskin (TYPE-instruktion) och den senare utmatning på hålremsstans (PUNCH-instruktion).

Som standard vid utmatning på hålremsstans användes 8-kanals remsa. Då inget annat angives i programmet sker alltså stansutmatning på 8-kanals remsa. Decimal utmatning kan emellertid även ske med telexremsa eller 5-kanals talremsa. De decimala stansinstruktionerna är utformade oberoende av remsrepresentation. För att ändra denna användes instruktionen:

```
P- CODE, N;
```

N är ett heltal som kan anta följande värden:

N = 0 utmatningen sker i fortsättningen med telexremsa.

N = 5 utmatningen sker i fortsättningen med 5-kanals talremsa.

N = 8 utmatningen sker i fortsättningen med 8-kanals remsa.

Ändring av remsrepresentationen vid utmatning påverkar ej remsrepresentationen för inläsning.

Instruktionen utlägges:

```
AS TO MR, N;
GO TO PR, P;
```

P- CODE påverkar ej representationen vid utmatning till skrivmaskin. Stansutmatning, som sker på skrivmaskin (knapp SM-B på manöverpanelen) kommer endast att ge meningsfull utskrift, då 8-kanals kod användes.

Utmatning av alfanumeriska strängar

Utmatning kan ske av strängar i TEXT-deklarationer. Härvid utmatas exakt de tecken, som ingår i strängen fram till semikolon, vilket ej utmatas. Semikolon representerade med SC utmatas dock.

För utmatning på skrivmaskin skrives

```
TYPE TEXT, A;
```

där A är namnet på TEXT-deklarationen.

Exempel: A: TEXT: CR Fel SP parameterremsa;

0128-0101-0:50/51/52/53

Instruktionen utlägges

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, 10010(0AS);
```

För utmatning på stans skrives

```
PUNCH TEXT, A;
```

där A är namnet på TEXT-deklarationen.

Instruktionen utlägges

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, (B);
```

B är en plats i DAC-basprogrammet.

Specifikation av format vid decimal utmatning

För att kunna typografiskt redigera decimal information vid utmatningen användes TEXT-deklarationen tillsammans med en FORMAT-instruktion. I TEXT-deklarationen specificeras önskat format vid utmatning av ett tal, medan FORMAT-instruktionen tolkar TEXT-deklarationen till en för utmatningsinstruktionerna lämplig form. En FORMAT-instruktion medför, att alla efterföljande utmatningsinstruktioner utföres enligt den typografi, som angivits genom denna, tills typografin ändras av en ny FORMAT-instruktion.

I en TEXT-deklaration, som skall tolkas av en FORMAT-instruktion, kan förutom information om tryckningsformen för talet även ingå information om de typografiska operationerna mellanslag, vagnretur, tabulator, stopp och utplåning. Då information härom skrives direkt i DAC-programmet, användes de symboler, som angivits i kapitel 2.

Information om typografi kan även inläsas vid körningstillfället med hjälp av READ TEXT-instruktionen.

Exempel:

```
TEXT: CR TAB 4.3 3SP ;
```

Deklarationen innebär, att först utföres vagnretur och tabulator, varefter tryckning sker med 4 heltalssiffror och 3 decimaler och avslutas med 3 mellanslag.

För att specificera taltypografi användes följande:

1) för heltal

   h.0 Heltalet tryckes med h heltalssiffror.

2) för maskintal

   h.d Talet multipliceras före tryckningen med 10^h varefter det tryckes med h heltalssiffror och d decimaler.

3) för reella tal

   h.d Talet tryckes med h heltalssiffror och d decimaler.

   h. $d_{10}N$ Talet tryckes normaliserat enligt h.d, d v s tryckes alltid med h signifikanta heltalssiffror och d decimaler och med en exponent, som beräknats för att ge rätt numeriskt värde av talet. N är här bokstaven och står ej som symbol för någon storhet.

   h. $d_{10}e$ Talet tryckes med h heltalssiffror och d decimaler och med den uppgivna exponenten e.

Exempel:

Om talet 1234,567 skall tryckas enligt

$1.5_{10}N$ erhålles $1.23457_{10}3$

$1.5_{10}5$ erhålles $0.01235_{10}5$

h, d är heltal $\geqslant 0$ medan e är ett godtyckligt heltal.

Angives h.0 kommer ingen decimalpunkt att tryckas

0.d kommer en heltalssiffra alltid att tryckas.

Insignifikanta nollor i början av heltalet ersättes av mellanslag. Dock tryckes alltid minst en heltalssiffra. Före positiva tal göres mellanslag, medan vid negativa tal ett minustecken tryckes omedelbart före första signifikanta heltalssiffran. Om talet innehåller flera heltalssiffror än vad som angivits, kommer talet att tryckas med korrekt värde, men de typografiska kraven kan givetvis ej uppfyllas.

0128-0101-0:53/54/55

Instruktionen för att tolka typografispecifikationen i en TEXT-deklaration skrives

```
FORMAT, L;
```

L är här namnet på TEXT-deklarationen.

Exempel: L: TEXT : 2SP 1.4_{10}N;

Instruktionen utlägges

```
AS TO MR, L;
GO TO PR, P;
```

Utmatning av heltal

Utmatning på skrivmaskin av ett heltal A utföres genom instruktionen

```
TYPE INT, A;
```

Instruktionen utlägges

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, 10010 (0AS);
```

Utmatning på stans av ett heltal A utföres genom instruktionen

```
PUNCH INT, A;
```

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, (B);
```

B är en plats i DAC-basprogrammet.

Utmatning av maskintal

Utmatning kan ske av maskintal i såväl enkel som dubbel precision.

Skrivmaskinsutmatning av ett tal utföres genom instruktionerna

```
TYPE , A;
DTYPE, A;
```

av ett maskintal i A i enkel precision respektive ett maskintal i A och A+1 i dubbel precision.

Instruktionerna utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, 10010(0AS);
```

Stansutmatning utföres på analogt sätt genom instruktionerna

```
PUNCH , A;
DPUNCH, A;
```

av maskintal i enkel respektive dubbel precision.

Instruktionerna utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, (B);
```

B är en plats i DAC-basprogrammet.

Utmatning av reella tal

Utmatning på skrivmaskin av ett reellt tal A utföres genom instruktionen FTYPE, A;

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, 10010 (OAS);
```

Utmatning på stans av ett reellt tal A utföres på analogt sätt genom instruktionen FPUNCH, A;

Instruktionen utlägges:

```
AS TO MR, A;
GO TO PR, P;
          ;
AS TO MR, (B);
```

B är en plats i DAC-basprogrammet.

Magnetbandsinstruktioner

I D21 kan vissa magnetbandsfunktioner utföras samtidigt på två skilda bandaggregat. De möjliga funktionerna är:

1) Inläsning och skrivning
2) Inläsning och blockräkning
3) Skrivning och blockräkning
4) Blockräkning på två bandaggregat

0128-0101-0:56/57/58

Oberoende härav kan spolning till ändläge ske på ett eller flera bandaggregat.

Då en magnetbandsinstruktion skall utföras i DAC-programmet, kontrolleras att

1) Inläsning ej kommer att ske på skilda bandaggregat samtidigt.
2) Skrivning ej kommer att ske på skilda bandaggregat samtidigt.
3) Ett bandaggregat ej beordras, på vilket en pågående instruktion ej är avslutad.
4) Båda utkanalerna ej är upptagna.

Instruktionen påbörjas ej förrän så kan ske med hänsyn till villkoren ovan. Under väntetiden kommer inget fortsatt genomlopp av programmet att äga rum. Då spolning till ändläge beordrats, anses denna instruktion avslutad i och med att beordring skett. Detta innebär, att en ny instruktion för samma bandaggregat kan komma att avbryta ändlägesspolningen innan ändläget uppnåtts. Stoppinstruktionen följer speciella regler, som angives, där denna instruktion beskrives.

Inläsning av ett magnetbandblock

Instruktionen skrives:

```
READ TAPE, N;
AS TO MR  , A;
AS TO MR  , M;
```

Genom instruktionen sker inläsning av M (decimalt) ord från det magnetbandblock, framför vilket läshuvudet står, då beordring sker. Inläsningen sker från bandaggregat N till A och följande platser i programmet.

Exempel:

```
READ TAPE,     4;
AS TO MR  , BETA;
AS TO MR  , 980 ;
```

Instruktionen utlägges:

```
AS TO MR, N;
GO TO PR, P;
          ;
AS TO MR, A;
AS TO MR, M;
```

Då hela block önskas inlästa, kan M alltid sättas = 1023, oberoende av om blocket i själva verket ej har denna längd. Inläsning kommer att ske endast av det antal ord, som blocket innehåller.

Skrivning av ett magnetbandblock

Instruktionen skrives:

```
WRITE TAPE, N;
AS TO MR  , A;
AS TO MR  , M;
```

Genom instruktionen kommer M st. konsekutiva ord från A och följande platser i programmet att utläggas som ett magnetbandblock på bandaggregat N.

Instruktionen utlägges:

```
AS TO MR, N;
GO TO PR, P;
          ;
AS TO MR, A;
AS TO MR, M;
```

Blockräkning

Blockräkning innebär, att magnetbandet spolas framåt eller bakåt ett önskat antal block.

Instruktionen för blockräkning i framriktningen skrives:

```
FORW TAPE, N;
AS TO MR  , M;
```

och innebär, att framspolning sker av M (decimalt) block på bandaggregat N.

Instruktionen för blockräkning i backriktningen skrives:

```
BACKW TAPE, N;
AS TO MR  , M;
```

och innebär analogt, att backspolning sker av M block på bandaggregat N.

Instruktionerna utläggas:

```
AS TO MR, N;
GO TO PR, P;
          ;
AS TO MR, M;
```


0128-0101-0:58/59/60/61

Ändlägesspolning

Ändlägesspolning kan ske såväl i fram- som backriktningen.

Ändlägesspolning i framriktningen på bandaggregat N skrives:

```
UNWIND TAPE, N;
```

medan ändlägesspolning i backriktningen på bandaggregat N skrives:

```
REWIND TAPE, N;
```

Instruktionerna utläggas:

```
AS TO MR, N;
GO TO PR, P;
70000000;
```

respektive

```
AS TO MR, N;
GO TO PR, P;
74000000;
```

Dessa instruktioner är av sådan typ att de ej ger upphov till programavbrott, då de avslutade. Man kan alltså ej använda READY-instruktionen för att konstatera, om de är avslutade. Man kan emellertid genom frågeinstruktionen

```
END TAPE, N;
```

konstatera, om spolningen är klar eller ej. Denna instruktion medför, att ackumulatorregistret blir positivt, om spolningen är klar, medan det blir negativt, om spolningen ej är avslutad. Instruktionen kan för övrigt användas i samband med alla bandinstruktioner för att konstatera, om aktivitet pågår vid magnetbandet eller ej. Pågår aktivitet, erhålles ackumulatorregistret negativt, medan det blir positivt, om ingen aktivitet pågår.

Instruktionen utlägges:

```
L: AS TO MR, N;
   01 1 00027;
   ←——— , (L);
```

Indirekt adressering med framstegning med ett kan ej användas i samband med denna instruktion.

Stoppinstruktion

Genom instruktionen

```
STOP TAPE, N;
```

kommer pågående magnetbandsaktivitet vid bandaggregat N omedelbart att stoppas.

Denna instruktion åtgärdas omedelbart, oberoende av om den aktivitet, som pågår vid bandaggregatet är avslutad eller ej. Då instruktionen beordras, undersöker ett administrationsprogram på vilken av de två bandkanalerna beordring skall ske. Därvid utväljes bandkanalerna efter följande ordning:

1) Den bandkanal, på vilken en magnetbandsinstruktion pågår på samma bandaggregat, som den beordrade.
2) En ledig bandkanal.
3) En bandkanal, på vilken blockräkning pågår.
4) En bandkanal, på vilken inläsning pågår.

Fallet 3 och 4 kan medföra att en pågående instruktion på ett annat bandaggregat avbrytes.

Stoppinstruktionen utlägges

```
AS TO MR, N;
GO TO PR, P;
```

In- och utmatning via decimaltablåregistret

Decimaltablåregistret (DTR) kan dels användas för utmatning och dels för inmatning. Vid inmatning erhålles ett programavbrott, då DV-knappen på D21:s manöverpanel nedtryckes.

Inmatning från DTR

Genom att inmatning från DTR ej styres via programmet utan genom operatören måste följande programmeringsmetodik användas. I programmet utarbetas en programdel, som hämtar in innehållet i DTR och vidtager de åtgärder, som föranleds av inmatningen. Då programavbrottet sker i huvudprogrammet, måste emellertid platsen för denna programdel vara definierad för det avbrottsadministrationsprogram, som ingår i DAC.

Efter det att DTR-programdelen genomlöpts kan det vara önskvärt att återvända till den plats i huvudprogrammet, där avbrottet skedde.

0128-0101-0:61/62/63/64

Genom instruktionen

DTR GO TO, A;

definieras platsen A för den programdel, som åtgärdar programavbrottet. Denna instruktion måste vara genomlöpt innan första DTR-programavbrottet erhålles. Då detta erhålles sker hopp till platsen A.

Instruktionen utlägges:

AS TO MR, A;
GO TO PR, P;

Innehållet i DTR erhålles till ackumulatorregistret genom instruktionen

GET DTR;

Denna instruktion utlägges:

01 1 00036;

Om innehållet i DTR är ett decimalt tal, som skall utläggas till en plats i programmet, sker detta med hjälp av instruktionerna

DTR TO INT , A;
DTR TO FIXED, A;
DTR TO REAL , A;

Då en av dessa instruktioner genomlöpes, måste innehållet i DTR, som erhölls genom GET DTR, finnas i ackumulatorregistret. Om ett heltal inmatats, användes den första instruktionen, varigenom heltalet placeras i A i den representation, som användes vid heltal. Den andra eller tredje instruktionen användes på analogt sätt vid inmatning av ett maskintal respektive ett reellt tal.

Instruktionerna utläggas:

AS TO MR, A;
GO TO PR, P;

Då återhopp önskas till den plats i huvudprogrammet, där DTR-programavbrottet skedde, skrives instruktionen

DTR RETURN;

Instruktionen utlägges:

GO TO, P;

P är en plats i administrationsprogrammet.

Observera att teckenkombinationen -.00000 vid inmatning till DTR är reserverad för den speciella stoppfunktion, som framgår av kap 5.

Utmatning till DTR

Utmatning till DTR utföres genom någon av de följande instruktionerna:

DTR INT, A;
DTR , A;
FDTR , A;

Den första instruktionen användes om A är heltal, den andra om A är maskintal och den tredje om A är ett reellt tal.

Instruktionerna utläggas:

AS TO MR, A;
GO TO PR, P;

Yttre val

Genom instruktionen

GET YVR;

erhålles innehållet i yttre valregistret (YVR) till ackumulatorregistrets 4 sista positioner. Övriga positioner är 0.

Instruktionen utlägges:

01 1 00035;

J128-0101-0:64/65/66

Kapitel 8

Programblock

Då ett antal deklarationer och instruktioner i ett DAC-program sammanfogas till ett block (kapitel 3) innebär detta, att en ny nomenklaturnivå införes i DAC-programmet. Härvid gäller att:

1) De namn, som införes genom deklarationerna, är interna för detta block och har inget gemensamt med en storhet med samma namn i ett överordnat block. Denna storhet blir oåtkomlig från blocket.
2) De namn, som förekommer i instruktionerna i blocket men ej är deklarerade där, måste vara deklarerade i ett yttre block och bibehåller sin gamla betydelse i det nya blocket.
3) Hopp kan ske till godtyckliga platser inom samma block eller överordnade block. Hopp till underordnade block kan endast ske till blockbörjan.

Utläggning av block

Utplacering av block i snabbminnet och på magnetband ombesörjes av programmeraren. För den skull anges förutom ordningsföljden mellan blocken på magnetband och i snabbminnet dessutom absolutadressen för blockbörjan eller en adress relativt början eller slutet av ett annat block. Sådan information om utläggningen tillhör emellertid ej DAC-programmet och behandlas därför ej här.

Vid översättningen kommer ett block att utplaceras i minnet som en sammanhängande enhet. Med block menas då de delar av blocket, som ej innehåller något underblock. Detta kan innebära, att översättaren utför en viss omdisposition av programmet. I nedanstående figurer visas blockstrukturen av ett DAC-program; i den vänstra i DAC-form och i den högra i en något annan form.

Beteckningen

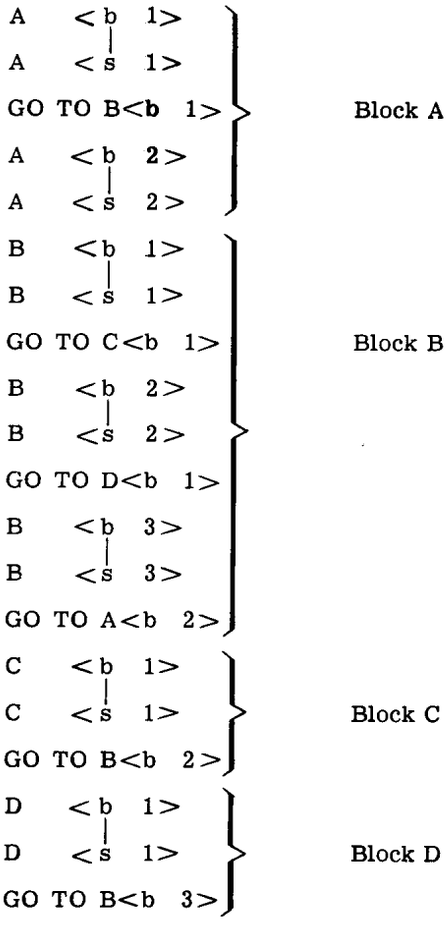
B < b > 2
|
B < s > 2

skall tolkas som andra delen av block B, där B < b 2 > utmärker början och B < s 2 > utmärker slutet.

```
A:  BEGIN ;          |     A < b  1 >
                     |       |
     ———             |     A < s  1 >
B:   BEGIN ;         |     B < b  1 >
     ———             |     B < s  1 >
C:   BEGIN ;         |     C < b  1 >
                     |       |
     END, C ;        |     C < s  1 >
     ———             |     B < b  2 >
                     |     B < s  2 >
D:   BEGIN ;         |     D < b  1 >
                     |       |
     END, D ;        |     D < s  1 >
                     |     B < b  3 >
     ———             |       |
     END, B ;        |     B < s  3 >
                     |     A < b  2 >
     ———             |       |
     END, A ;        |     A < s  2 >
```


0128-0101-0:66/67

Om ordningsföljden mellan blocken skall vara A, B, C och D kommer vid översättningen följande disposition att erhållas:



De sammanbindande hoppinstruktionerna utlägges av DAC-översättaren. Även om ordningsföljden mellan blocken ändras, kommer programgenomloppet alltid att ske på korrekt sätt, förutsatt att start alltid sker i A<b 1> .

Utläggning inom block

I ett block utlägges instruktionerna i början av blocket, varefter följer alla deklarationerna. För deklarationerna kommer minnesutläggning att ske i samma ordning som deklarationerna har i blocket och med variabler och konstanter i den ordning de angivits i deklarationen.

Utläggning av underprogram för pseudoinstruktioner

I och med att en pseudoinstruktion, som ger uppnov till ett programhopp till ett underprogram, användes i DAC-programmet, kommer vid översättningen underprogrammet automatiskt att medtagas. Dessa underprogram kommer då att utläggas utanför DAC-programmet och kommer att vara tillgängliga i snabbminnet under hela programgenomloppet.

Om den tillgängliga snabbminneskapaciteten är för liten för att hela DAC-programmet skall samtidigt rymmas i snabbminnet, måste vissa block utplaceras på magnetband och hämtas in vid behov. Det kan då vara önskvärt att låta ett pseudoinstruktionsunderprogram ingå i blocket, om exempelvis pseudoinstruktionen endast användes i detta block. Man kan för den skull styra utläggningen av sådana underprogram genom att en LIBRARY-deklaration skrives i blocket och i denna anges pseudoinstruktionens operationsdel. De pseudoinstruktioner, som kan utplaceras på detta sätt, är instruktionerna för de elementära matematiska funktionerna och in- och utmatning via hålrmsa, skrivmaskin och DTR.

För att kunna LIBRARY-deklarera en pseudoinstruktion i ett block får instruktionen ej förekomma i ett yttre block.

För utläggning inom blocket gäller vad som sagts i närmast föregående avsnitt.

Exempel: LIBRARY : FSIN, FPUNCH;

Överföring av programblock från magnetband

I DAC kan överföring av programblock från magnetband ske med tillhjälp av två instruktioner.

```
PREPARE BLOCK, A;
INPUT   BLOCK, A;
```

A är här namnet på blocket.

PREPARE-instruktionen innebär, att magnetbandet spolas till det magnetbandblock, som innehåller programblock A. En PREPARE-instruktion kan beordras från en godtycklig plats i DAC-programmet.

0128-0101-c-68/69/70

Genom INPUT-instruktionen utföres inläsning av programblocket. Denna instruktion förutsätter, att magnetbandet står framspolat i rätt läge. INPUT kan ej beordras från en plats i DAC-programmet, som överläses av det nya blocket.

En förutsättning för att denna programblocksöverföring skall fungera på korrekt sätt är att det bandaggregat, som innehåller programblocket, ej användes genom andra magnetbandsinstruktioner än de här nämnda. Om så sker, måste programmeraren själv svara för PREPARE-funktionen, medan INPUT-instruktionen fortfarande kan användas.

För att konstatera om PREPARE- och INPUT-instruktionerna är utförda kan READY-instruktionen (kapitel 7) användas.

0128-0101-0:70

Appendix 1

Konventioner för stansning av decimala dataremsor

Decimala data, som inläses med någon av pseudoinstruktionerna i DAC, skall på hålremsa representeras enligt de konventioner, som följer här nedan. Vid inläsningen kan 8-kanals remsa, telexremsa eller 5-kanals talremsa användas. Som standard användes 8-kanals remsa, och önskas annan representation måste detta beordras i DAC-programmet genom R- CODE-instruktionen (kapitel 7).

Decimala tal

Ett tal, som ej föregås av förtecken, betraktas som positivt. Plustecken kan även användas som förtecken för positiva tal. Före negativa tal stansas minustecken. Punkt användes för decimalpunkt. I talet måste alla tecken komma i en följd, ett mellanslag är alltså ej tillåtet mitt i talet. För att utmärka slut på talet kan alla tecken, som ej har någon speciell funktion vid decimal datastansning användas t ex mellanslag, vagnretur och tabulator. Ett förtecken för ett efterföljande tal kan ej användas som avslutningssymbol. Vid tal med 10-exponent måste exponenttecknet följa omedelbart efter sista siffran i mantissan. Därefter kan exponenten följa eventuellt föregången av irrelevanta tecken.

Exempel vid 8-kanals remsa

7

-3.187

$2.3_{10}{}^{-7}$

Tecknen övre och nedre läge vid 8-kanals remsa och bokstavs- och sifferskift vid telexremsa kan ej användas för att utmärka talavslutning. Dessa tecken användes endast för omskiftningsfunktionen.

Talföljder

Genom pseudoinstruktionerna av READ SEQ-typ erhålles inläsning av de konsekutiva tal, som är stansade på hålremsan fram till en symbol, som utmärker talföljdsslut. Vid sådan inläsning kommer summakontroll att utföras, om symbolen för "kontrollsumma följer" är stansad på remsan. Kontrollsumman måste sedan följa efter denna symbol och avslutas med symbolen för talföljdsslut.

Summakontroll

Vid instruktionerna av READ SEQ-typ (kapitel 7) kan summakontroll automatiskt erhållas, om sådan beordras på hålremsan. Vid decimal inläsning genom andra pseudoinstruktioner kommer en ackumulerande summabildning att ske till en plats i programmet, som kan anvisas av programmeraren. Om summakontroll önskas, måste tillses, att nollställning utföres, innan inläsningen ingångsättes, inläsning av kontrollsumman beordras och kontroll utföres i programmet.

Kontrollsumman beräknas på följande sätt:

Alla tal, som skall kontrolleras, förvandlas först till heltal och förses med en 10-exponent, som gör att talet blir numeriskt korrekt. Därefter summeras alla taldelar och alla exponentdelar

var för sig, och de båda heltal, som så erhålles, utgör kontrollsumman. I denna skrives taldelssumman först.

Exempel:

Talet	Heltal	Exponent
7	7	0
-1.86	-186	-2
$2.3.10^{-7}$	23	-8
-12	-12	0
0.3	3	-1
Summa	-165	-11

Kontrollsumman utgör alltså de båda heltalen -165 och -11. Om talföljden skulle stansats på 8-kanals remsa, skulle denna få följande utseende:

7 -1.86 2.3_{10}-7 -12 0.3 Λ -165 -11 /

Radering

Genom ett raderingstecken kan det tal, som är under inläsning eller helt har inlästs, borttagas och ersättas med det efter raderingstecknet följande talet. Då inläsningen ej sker genom talföljdsinläsning, måste raderingstecknet komma omedelbart efter siffrorna i talet, dvs ett talavslutningstecken får ej ha föregått. Om vid talföljdsinläsning ett helt inläst tal raderas, måste det raderade talet ingå i kontrollsumman för att denna skall stämma.

För radering av ett tecken finns speciella tecken vid 8-kanals remsa och 5-kanals talremsa, som stansas över det felaktiga tecknet. Dessa raderingstecken ignoreras helt under inläsningen.

Remsstopp

Genom ett stopptecken på remsan erhålles stopp under den decimala inläsningen. Vid start fortsättes inläsningen. Om stopptecknet står i slutet på en remsa, måste ytterligare ett tecken följa, ty i annat fall spolas remsan genom remsläsaren.

Tecknens funktion

I nedanstående tabell anges de olika tecken, som har en speciell funktion vid inläsning av decimala data. Alla övriga tecken med undantag av omskiftningstecken kan användas för att utmärka slut på ett tal och ignoreras i övrigt.

Funktion	8-kanals remsa	telex-remsa	5-kanals talremsa
Siffror	0-9	0-9	0-9
Decimalpunkt	.	.	.
10-exponent	10	:	E
Plustecken	+	+	+
Minustecken	-	-	-
Talföljdsslut	/	/	A
Kontrollsumma följer	Λ	?	C
Radering av tal	)	)	B
Radering av tecken	Utplåning	Saknas	F
Remsstopp	Stoppkod	Klocka	Stopp

Om det vid inläsning av det första talet ej förekommit ett omskiftningstecken, förutsätter inläsningsinstruktionerna att nedre läge gäller vid 8-kanals remsa och sifferskift vid telexremsa.

0128-0101-0:72/73/74

Appendix 2

Teckenrepresentation på hålremsa

På 8-kanals hålremsa användes en av kanalerna för paritetskontroll. Udda paritet användes, d v s antalet hål för ett tecken skall vara ett udda antal. Vid inläsningen till D21 sker paritetskontroll, och då pariteten är korrekt, överföres de återstående 7 kanalerna till ackumulatorregistret.

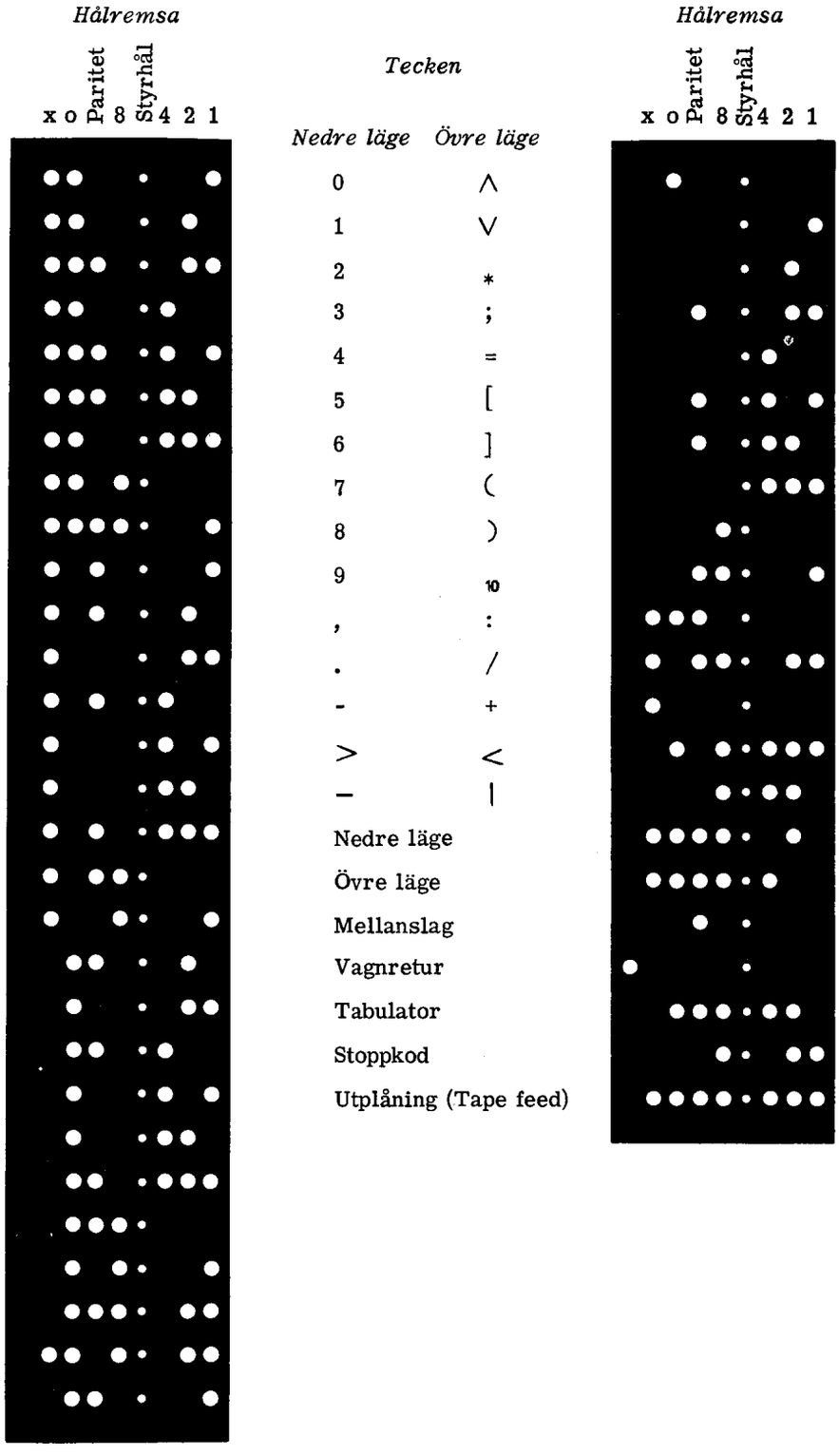
Vid inläsning av telexremsa och 5-kanals talremsa sker ingen paritetskontroll. Hela den inlästa raden överföres till ackumulatorregistret.

Tecknen __ och | leder ej till någon frammatning av vagnen på skrivmaskinen vid 8-kanals remsa. Samma gäller för tecknet __ (understrykning) vid 5-kanals talremsa.

Tecknet "Utplåning" (Tape feed) ignoreras av D21-remsläsaren, och frammatning sker till nästa tecken.

0128-0101-0:75

8-kanals remsa

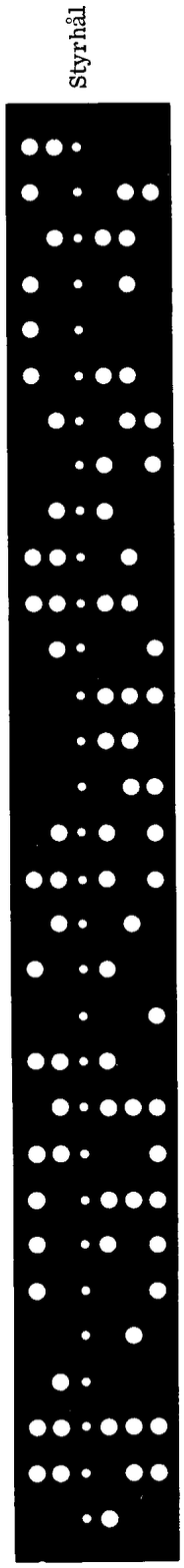


0128-0101-0

Telexremsa

Bokstav	*Siffra*
A	-
B	?
C	:
D	Vem där
E	3
F	Å
G	Ä
H	Ö
I	8
J	Klocka
K	(
L	)
M	.
N	,
O	9
P	0
Q	1
R	4
S	'
T	5
U	7
V	=
W	2
X	/
Y	6
Z	+
Vagnretur	
Ny rad	
Bokstavsskift	
Sifferskift	
Mellanslag	

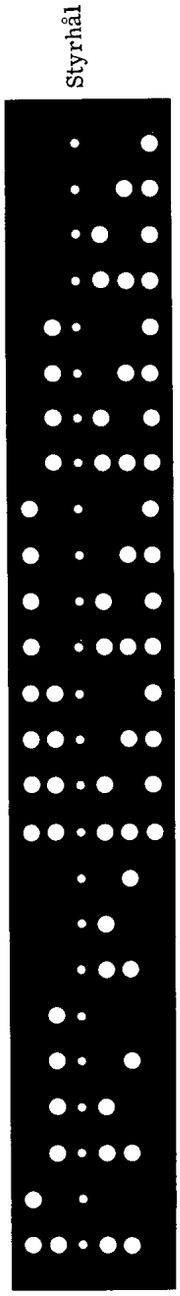
Hålremsa



5-kanals talremsa

Tecken
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
Vagnretur
Mellanslag
Tabulator
-
+
_ (understrykning)
.
*
Stopp

Hålremsa



Sakregister

0128-0101-0:1a/1b

0128-0101-0:1b/1c

0128-0101-0:1d

SVENSKA AEROPLAN AKTIEBOLAGET
Linköping

0128-0101
Layout: Publikationsavd. Saab
Tryck: Stålhammar/Zetterqvist
Linköping 1964