Aspecten van de conversie van X-8 (THE) systeemprocedures

Dijkstra, E.H.H.

Award date:
1972

A F S T U D E E R V E R S L A G

Aspecten van de conversie van X-8 (THE) systeemprocedures

door

E.H.H. DIJKSTRA

Afstudeerhoogleraar Prof.Dr. R.J. Lunbeck

september 1972

Technische Hogeschool Eindhoven
Afdeling der Technische Natuurkunde

BIJLAGEN

I Karakterrepresentatie van Algol 60 symbolen voor EL X-8, CD-6000, IBM-360 OS, Burroughs B6700, IBM 72 59 char-set.

II Compatibiliteit van Algol 60 teksten voor EL X-8, CD-6000, IBM-360 OS, Burroughs B6700 met het Revised Report on the Algorithmic Language Algol 60 als referentie.

III Tabellen van OCT, CHARF en SYM waarden.

IV Literatuuroverzicht.

Inleiding.

Er is onderzoek verricht naar de problemen die zich voordoen wanneer X-8 (THE) Algolteksten op andere computersystemen verwerkt worden.
In eerste instantie is de compatibiliteit van Algol-60 teksten voor EL X-8, CD-6000, IBM-360 OS, Burroughs B6700 vergeleken.
Het Revised Report on the Algorithmic Language Algol-60 is als referentie aangehouden.
Om de studie van de compatibiliteit te vergemakkelijken zijn de karakter verzamelingen van de verschillende machines vergelijkenderwijs in tabelvorm opgenomen; Bijlage I.
Vervolgens zijn EL X-8, CD-6000 en IBM-360 OS ten opzichte van het Revised Report getoetst op:
- Syntactische verschillen
- Uitbreidingen
- Beperkingen

De Burroughs B6700 wordt ten opzichte van het Revised Report getoetst op:
- Syntactische verschillen
- Beperkingen

Vanwege het grote aantal uitbreidingen van de Burroughs B6700 ten opzichte van het Revised Report zijn alleen die uitbreidingen opgenomen die relevant zijn voor de uitwerking van de conversie van X-8 (THE) Algoltekst naar Burroughs B6700 Algoltekst.
Uitgangsteksten voor de studie van de compatibiliteit van Algol-60 zijn de ten tijde van onderzoek laatst bekende publicaties.

EL X-8 — Technische Hogeschool Eindhoven
Rekencentrum Informatie nr. 30, 32 d.d. 1-12-1971

CD-6000 — Control Data Corporation
Algol generic reference manual
Publicatie nr. 60214900 d.d. 1-5-1968

IBM-360 OS IBM System/360
Operating System Algol Language
File nr. S360-26
Form C28-6615-1 d.d. 15-4-1969

Burroughs B6700 Burroughs B6700 Extended Algol Language
Information Manual nr. 5000128 d.d. 1-7-1971.

De resultaten van dit onderzoek zijn opgenomen in bijlage II.
In de paragrafen van het Revised Report met meerdere metalinguistische formules zijn deze formules genummerd gedacht, derhalve stelt 2.3.2 de tweede formule van paragraaf 2.3 voor.

Een aantal aspecten van de conversie van X-8 Algolteksten naar Burroughs B6700 Algolteksten kunnen hier genoemd worden.
THE-Algol is een variant van Algol-60 terwijl Burroughs B6700 Extended Algol het Algol-60 omvat. Onderlinge syntactische verschillen zijn voor zover ze relevant zijn ten aanzien van het doel, verwerking van THE-Algolteksten met behulp van de Burroughs B6700 in bijlage II opgenomen. Uit de systeembeschrijvingen van beide systemen (1 en 4) blijken de overige verschillen. Men komt zo tot een indeling in twee categorien.

A. De verschillen die overbrugd kunnen worden door nabootsing van (onderdelen van) het THE systeem in termen van het B6700 systeem.
Deze verschillen zijn gelegen in:
   1. de procedures voor invoer en uitvoer
   2. de procedures voor het werken met strings
   3. library procedures
   4. de mogelijkheid tot invoeging van stukken Algoltekst van magneetbanden.

   de procedures genoemd onder 1, 2 en 3 zijn systeemprocedures d.w.z. procedures die de gebruiker niet zelf hoeft te programmeren en te declareren.

B. De verschillen die overbrugd kunnen worden door manipulatie van de actuele THE-Algoltekst.
Deze verschillen zijn gelegen in:

1. het type complex in THE-Algol
2. de label declaratie in B6700-Algol
3. de forward reference declaratie in B6700-Algol
4. de dimensie specificaties bij formele arrays in B6700-Algol
5. de string als actuele parameter en string als specificator in THE-Algol
6. de designational expression in THE-Algol
7. de repeat statement in THE-Algol
8. standaardkop en standaardstuk van de THE-Algoltekst.

In dit verslag is het onderzoek verder toegespitst op aspecten van de omzetting van invoer en uitvoerprocedures van de X-8 (THE). Deze zijn daartoe herschreven in Algol-60 uitgebreid met:

1. Een routine om één enkele bandponsing te lezen.
2. Een routine om één enkel karakter af te drukken op een regeldrukker en/of een overgang op een nieuwe regel/pagina te bewerkstelligen.
3. Een routine om één enkele bandponsing te ponsen
4. Een routine die een volgband aanvraagt.
5. Een tweetal routines om een string op te splitsen in zijn samenstellende delen.

Omschrijvingen van in dit verslag gebruikte termen.

bandponsing

-elke configuratie van wel of geen ponsgaatjes in de bandstrook.

zichtbare positiebedrukking

-elke toetsaanslag op een flexowriter, die aanleiding geeft tot een wagenverplaatsing.

Toetsaanslagen, die aanleiding geven tot bedrukkingen met spaties direkt voor een overgang op een nieuwe regel, gelden echter niet als zichtbare positiebedrukkingen.

opmerkingen: 1. Herhaling van dezelfde toetsaanslag, zonder dat er een wagenverplaatsing plaats vindt, wordt gelijkwaardig geacht met de enkele toetsaanslag.

2. Toetsaanslagen zonder wagenverplaatsing, zoals omschakeling van kleine-letter naar hoofd-letter en omgekeerd, onderstreping, doorbalking, worden gecombineerd met de eerstvolgende toetsaanslag, die wel aanleiding geeft tot een wagenverplaatsing.

basissymbool

-wat in het Algol-rapport <basic symbol> wordt genoemd.

HOOFDSTUK 1

Nabootsing van systeeminvoerprocedures voor een ponsbandlezer.

1.1 Deze paragraaf handelt over de volgende systeeminvoerprocedures, waarbij structuur en onderlinge samenhang aan de orde komen.

integer procedure REHEP - leest één bandponsing
integer procedure CHARF - leest één zichtbare positiebedrukking
integer procedure SYM - leest één basissymbool
real procedure NUM - leest één getal

1.1.1 Inleiding

Op een ponsband kan men informatie vastleggen. De kleinste te beschouwen informatie-eenheid is de enkele ponsbandponsing. Wanneer de ponsband een 7-kanaalsband is, dan zijn er voor een enkele bandponsing 128 verschillende configuraties van wel of geen ponsgat mogelijk. Deze configuraties zijn binair te interpreteren, vanaf 0 (geen gat in de enkele bandponsing) t/m 127 (in elk van de 7 kanalen een ponsgat). Deze binaire interpretaties beschouwen we als het te onzer beschikking staande alfabet, waaruit grotere gehelen gevormd kunnen worden. Dat deze binaire interpretaties oorspronkelijk van de ponsband komen is dan niet meer ter zake dienend. Aan het fysisch einde van de band wordt ook een karakteristieke waarde toegekend.

1.1.2 procedure REHEP

De integer procedure REHEP tast één bandponsing af en voegt aan de gelezen informatie een getalwaarde, en wel de binaire interpretatie van de bandponsing, toe. Voor de gebruikte 7-kanaalsband ligt deze getalwaarde in het interval (0,127); aan "einde band" wordt echter de getalwaarde -1 toegekend.

1.1.3 procedure CHARF

Van de ponsband leest men één of meer bandponsingen als één zichtbare positiebedrukking. De integer procedure CHARF leest met gebruikmaking van REHEP de ponsband.
Bandponsingen leggen éénduidig een zichtbare positiebedrukking vast.

Omgekeerd hoort bij een zichtbare positiebedrukking niet éénduidig een combinatie van één of meer bandponsingen omdat:

1. de hoofdlettertoets en de kleine-lettertoets op zichzelf wel een ponsing geven maar nog geen zichtbaar resultaat.
2. bij een niet transporterende toets (onderstreping,doorbalking) de aanslag wel een ponsing geeft, terwijl in de zichtbare positiebedrukking, enkele onderstreping of doorbalking, niet van dubbele is te onderscheiden.

De eerste aanroep van CHARF dient te worden voorafgegaan door een aanroep van INICHARF welke onder andere de case initialiseert.
Onder case wordt hier verstaan lowercase (kleine-lettertoets) of uppercase (hoofd-lettertoets).
Na een aanroep van REHEP dient een aanroep van CHARF ook weer te worden voorafgegaan door een aanroep van INICHARF omdat na een aanroep van REHEP de case ongedefinieerd is.

1.1.4 procedure SYM

Van een ponsband kan men op verschillende wijzen een basissymbool lezen. Mogelijkheden zijn:

1. De integer procedure SYM leest met gebruikmaking van CHARF de ponsband. Men leest één of meer zichtbare positiebedrukkingen als één basissymbool.
2. De integer procedure SYM leest met gebruikmaking van REHEP de ponsband. Nu leest men één of meer bandponsingen als één basissymbool.

Zichtbare positiebedrukkingen leggen éénduidig een basissymbool vast.
Omgekeerd hoort bij een basissymbool niet éénduidig een aantal zichtbare positiebedrukkingen: naast Boolean wordt ook boolean en naast go to wordt ook goto en go to toegestaan.
Bij de X-8 (THE) wordt een basissymbool gelezen als een aaneenschakeling van zichtbare positiebedrukkingen.

1.1.5 procedure NUM

Van een ponsband kan men op verschillende wijzen een getal lezen. Mogelijkheden zijn:

1. De real procedure NUM leest met gebruikmaking van SYM de ponsband. Het getal wordt dan gelezen als een aaneenschakeling van basissymbolen. Toegestane tekens zijn $0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ .\ +\ -_{10}$
2. De real procedure NUM leest met gebruikmaking van CHARF de ponsband. Het getal wordt dan gelezen als een aaneenschakeling van uitsluitend zichtbare positiebedrukkingen. In dit geval kan men een getal verwerken dat niet is opgebouwd uit een rij basissymbolen. Een voorbeeld hiervan is een getal samengesteld uit onderstreepte cijfers.
3. De real procedure NUM leest met gebruikmaking van REHEP de ponsband. Het getal wordt dan gelezen als een aaneenschakeling van bandponsingen.

De gebruiker hoeft niet te weten volgens welke van deze drie mogelijkheden getallen van de band gelezen worden. Het leidende beginsel is, dat het de gebruiker zo gemakkelijk mogelijk gemaakt dient te worden.
Bij de X-8 (THE) wordt een getal gelezen als een aaneenschakeling van basissymbolen, waarin de tekens $0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ .\ +\ -_{10}$ kunnen voorkomen.

1.1.6 Enkele opmerkingen

Uit het voorafgaande blijkt dat de procedure SYM op twee manieren een basissymbool kan lezen. Als er een procedure SYM gerealiseerd is waarin beide mogelijkheden aan bod komen dan kan men door middel van een integer toestandsvariabele "basisinterpretatie" de wijze van lezen vastleggen.

basisinterpretatie = 1

Van de ponsband leest men één of meer bandponsingen. Hier is uitsluitend REHEP van toepassing.

basisinterpretatie = 2

Van de ponsband leest men één of meer zichtbare positiebedrukkingen. Hier is uitsluitend CHARF van toepassing.

basisinterpretatie = 3

Er is niet bekend volgens welke mogelijkheid de ponsband gelezen moet worden.

Als NUM met gebruikmaking van REHEP de ponsband leest is de wijze waarop van een ponsband door NUM een getal wordt gelezen éénduidig bepaald. De procedure NUM kan verder nog op twee manieren een getal lezen. Als er een procedure NUM gerealiseerd is waarin beide mogelijkheden aan bod komen dan kan men door middel van een integer toestandsvariabele "numerieke interpretatie" de wijze van lezen vastleggen.

numerieke interpretatie = 1

Van de ponsband leest men één of meer zichtbare positiebedrukkingen. Hier is uitsluitend CHARF van toepassing.

numerieke interpretatie = 2

Van de ponsband leest men één of meer basissymbolen. Hier is uitsluitend SYM van toepassing.

numerieke interpretatie = 3

Er is niet bekend volgens welke mogelijkheid de ponsband gelezen moet worden.

Daar bij de X-8 (THE) een getal van een ponsband gelezen wordt als een aaneenschakeling van basissymbolen en elk basissymbool gelezen wordt als een aaneenschakeling van zichtbare positiebedrukkingen hoeft men hier geen gebruik te maken van het begrip numerieke interpretatie.

1.2 Systeeminvoerprocedures voor de bandlezers bij de X-8 (THE)

Elk gebruikersprogramma heeft de beschikking over twee invoerbuffers, bandlezerstromen genaamd. Elke systeemprocedure leest gebruikersinformatie over één van deze bandlezerstromen in. Een parameter I identificeert de bandlezerstroom. De waarde van deze parameter moet overeenkomen met het nummer van de stroom waarover men werkt. De parameter I heeft de waarde 1 of 2. Indien I hieraan ongelijk is volgt de dynamische melding van een fout, de uitvoering van het programma wordt dan terstond beeindigd door een sprong naar de nooduitgang L END. De bandleesprocedures oorspronkelijk in "ELAN" geschreven zijn nagebootst en beschreven in Algol-60 uitgebreid met OCTET, een routine om één enkele bandponsing te lezen, een routine INIPAPERTAPE die een volgband aanvraagt, en een tweetal routines om een string op te splitsen in zijn samenstellende delen. Bestaande regels zoals verstrekt in de RC-informatie 30 en 32 zijn hierbij in acht genomen. Omzetting van deze uitgeschreven bandleesprocedures in Burroughs B6700 Algol is eenvoudig.

1.2.1

Voorkomende bandleesprocedures, die later nog besproken zullen worden, zijn:

```
procedure INIPATA(I); value I; integer I;
integer procedure OCT(I); integer I;
integer procedure OCT1;
integer procedure OCT2;
integer procedure REHEP;
procedure BACKOCT(I,X); value I,X; integer I,X;
integer procedure CHARF(I); integer I;
integer procedure CHARF1;
integer procedure CHARF2;
procedure BACKCHARF(I,X); value I,X; integer I,X;
procedure INICHARF(I,POS,LOWER); value I,POS,LOWER; integer I,POS;
   boolean LOWER;
integer procedure SYM(I); value I; integer I;
integer procedure SYM1;
integer procedure SYM2;
integer procedure RESYM;
```

```
procedure BACKSYM(I,X); value I,X; integer I,X;
real procedure NUM(I); integer I;
real procedure NUM1;
real procedure NUM2;
real procedure READ;
real procedure read;
procedure SKIPPATA(I); value I; integer I;
integer procedure SYMINSP(I); value I; integer I;
integer procedure NUMINSP(I); value I; integer I;
```

Voor een gebruikersprogramma verwerkt gaat worden, worden door het hier te beschrijven systeem een aantal systeemvariabelen (zie 1.2.8), op een vaste manier geinitialiseerd. Hieronder is de heersende case. De aanhef van het te lezen bandstuk van het gebruikersprogramma moet dan duidelijk maken of de heersende case ongedefinieerd blijft. In het THE-systeem wordt de stopcode gebruikt om de waarde van de heersende case ongedefinieerd te maken. De stopcode heeft dus de betekenis dat de heersende interpretatie tot nader order is afgelopen. Als CHARF een stopcode gelezen heeft dan levert een aanroep van REHEP de eerste heptade, die volgt op stopcode.
Bij de beschrijving van de bandleesprocedures is voorondersteld dat de routine OCTET aan elke eerstvolgende bandponsing een binaire interpretatie geeft. Hierbij dient ook op "einde band" gereageerd te worden. Hieraan wordt de waarde -1 toegekend.

1.2.1.1 procedure INIPATA(I); value I; integer I;

INIPATA vraagt door middel van INIPAPERTAPE volgband. Als volgband beschikbaar is vult INIPATA een buffer BUFi met toegekende OCTET waarden van gelezen bandponsingen vanaf de allereerste plaats in de buffer. (Als er een variabele staat gevolgd door een i wordt bedoeld de variabele gevolgd door een 1 of de variabele gevolgd door een 2.) Altijd wordt de gehele ponsband ingelezen. Daar op ponsbanden blanks, spaties of erases veelal in veelvoud achter elkaar voorkomen, worden in de buffer voor deze bandponsingen waardenparen opgeborgen. Eerst de toegekende OCTET waarde van de gelezen blank, spatie of erase en vervolgens het aantal malen dat deze blank, spatie of erase achtereenvolgens voorkomt.
Voorkomende locale variabelen in INIPATA zijn:

COUNTER A bepaalt de bufferplaats waar een OCTET waarde van een gelezen bandponsing wordt opgeborgen.
BINARY VALUE hieraan wordt de OCTET waarde van een gelezen bandponsing toegekend.
COUNTER B telt het aantal opeenvolgende spaties, blanks of erases.
NEXT BINARY VALUE hieraan wordt de OCTET waarde van een gelezen bandponsing volgend op een spatie, blank of erase toegekend.

Voorkomende globale variabelen in INIPATA zijn:

P1 wijst de eerstvolgende te lezen plaats in buffer BUF1 aan.
P2 wijst de eerstvolgende te lezen plaats in buffer BUF2 aan.
TAPE END heeft de OCTET waarde voor bandeinde.
BLANK heeft de OCTET waarde voor een blanke ponsing.
SPACE heeft de OCTET waarde voor een spatie ponsing.
ERASE heeft de OCTET waarde voor een uitpoetsponsing.
De booleans BOPASTi, BCPASTi, BSPASTi worden false; dit zijn booleans die verder in dit verslag nog nader toegelicht zullen worden.

```
procedure INIPATA(I);value I;integer I;
if I=1 ∨ I=2 then
begin integer DUMMY,COUNTER A,BINARY VALUE;
   if (I≠1 ∨ BUF1[P1]=-1) ∧ (I≠2 ∨ BUF2[P2]=-1) then
   begin if I=1 then P1:=0 else P2:=0;
      COUNTER A:=-1;INIPAPERTAPE(I);BINARY VALUE:=OCTET(I);
      for DUMMY:=0 while BINARY VALUE≠TAPE END do
      begin COUNTER A:=COUNTER A + 1;
      if I=1 then BUF1[COUNTER A]:=BINARY VALUE
             else BUF2[COUNTER A]:=BINARY VALUE;
      if BINARY VALUE=BLANK ∨ BINARY VALUE=SPACE ∨ BINARY VALUE=ERASE then
      begin integer COUNTER B,NEXT BINARY VALUE;
         COUNTER B:=1;NEXT BINARY VALUE:=OCTET(I);
         for DUMMY:=0 while NEXT BINARY VALUE=BINARY VALUE do
         begin COUNTER B:=COUNTER B + 1;NEXT BINARY VALUE:=OCTET(I) end;
         COUNTER A:=COUNTER A + 1;
         if I=1 then BUF1[COUNTER A]:=COUNTER B
                else BUF2[COUNTER A]:=COUNTER B;
         BINARY VALUE:=NEXT BINARY VALUE
      end else BINARY VALUE:=OCTET(I)
      end;
      COUNTER A:=COUNTER A + 1;
      if I=1 then
      begin BUF1[COUNTER A]:=-1;BOPAST1:=BCPAST1:=BSPAST1:=false end
            else
      begin BUF2[COUNTER A]:=-1;BOPAST2:=BCPAST2:=BSPAST2:=false end
   end
end else
begin PRINTTEXT(≤DYNAMIC ERROR IN STREAMNUMBER≥);
   go to L END
end INIPATA(I);
```

1.2.1.2 integer procedure OCT(I); integer I;

De integer procedure OCT(I) pelt uit de buffer BUFi per aanroep de volgende OCTET waarde. Bij het lezen blijft de buffer BUFi niet ongewijzigd op die plaatsen waar aantallen opeenvolgende spaties, erases en blanks vermeld staan. Deze aantallen worden door het lezen afgelaagd tot één. Levert de aanroep van OCT -1 (dat is einde band) op dan levert elke volgende aanroep van OCT ook -1 op tot aan een volgende actualisering door een aanroep van INIPATA. Door OCT wordt de waarde van de heersende case ongedefinieerd.

Voorkomende locale variabelen in OCT zijn:

OCTX hieraan wordt de OCTET waarde van de inhoud van de bufferplaats in BUFi aangewezen door pointer Pi toegekend.

Voorkomende globale variabelen in OCT zijn:

OCTBUFi de waarde van deze variabele wordt aan OCT toegekend als BOPASTi=true. Zie ook de later te bespreken procedure BACKOCT.
De booleans BOPASTi, BCPASTi en BSPASTi worden false.
De booleans LOWERCASEi en UPPERCASEi worden false.
Als INICHARF wordt aangeroepen, een procedure welke later besproken wordt, zal hetzij LOWERCASEi hetzij UPPERCASEi true worden.
De boolean OCTPASTi wordt door OCT true. Als INICHARF wordt aangeroepen zal OCTPASTi false worden.

```
integer procedure OCT(I);integer I;
begin integer OCTX;
   if I=1 then
   begin if ¬BOPAST1 then
      begin OCTX:=BUF1[P1];P1:=P1 + 1;
         if OCTX=BLANK ∨ OCTX=SPACE ∨ OCTX=ERASE then
         begin if BUF1[P1]=1 then P1:=P1 + 1 else
            begin BUF1[P1]:=BUF1[P1]-1;P1:=P1 - 1 end
         end else if OCTX=TAPE END then P1:=P1 - 1;
         OCT:=OCTX;
         BCPAST1:=BSPAST1:=LOWERCASE1:=UPPERCASE1:=false;
         OCTPAST1:=true
      end else
      begin OCT:=OCTBUF1;BOPAST1:=false end
   end else
   if I=2 then
   begin if ¬BOPAST2 then
      begin OCTX:=BUF2[P2];P2:=P2 + 1;
         if OCTX=BLANK ∨ OCTX=SPACE ∨ OCTX=ERASE then
         begin if BUF2[P2]=1 then P2:=P2 + 1 else
            begin BUF2[P2]:=BUF2[P2]-1;P2:=P2 - 1 end
         end else if OCTX=TAPE END then P2:=P2 - 1;
         OCT:=OCTX;
         BCPAST2:=BSPAST2:=LOWERCASE2:=UPPERCASE2:=false;
         OCTPAST2:=true
      end else
      begin OCT:=OCTBUF2;BOPAST2:=false end
   end else
   begin PRINTTEXT(‹DYNAMIC ERROR IN STREAMNUMBER›);
      go to L END
   end
end OCT(I);
```

1.2.1.3 integer procedure OCT1;

De waarde aan deze procedure toegekend is gelijk aan de binaire waarde zoals deze door OCT geleverd wordt.

```
integer procedure OCT1;OCT1:=OCT(1);
```

1.2.1.4 integer procedure OCT2;

De waarde aan deze procedure toegekend is gelijk aan de binaire waarde zoals deze door OCT geleverd wordt.

```
integer procedure OCT2;OCT2:=OCT(2);
```

1.2.1.5 integer procedure REHEP;

De waarde aan deze procedure toegekend is gelijk aan de binaire waarde zoals deze uitsluitend via de eerste lezerstroom door OCT geleverd wordt.

```
integer procedure REHEP;REHEP:=OCT(1);
```

1.2.1.6 procedure BACKOCT(I,X); value I,X; integer I,X;

Bij een eerstvolgende aanroep van OCT komt de waarde van X weer ter beschikking. Dit wordt ook wel de voorgift voor OCT genoemd.
Met de boolean BOPASTi wordt bijgehouden of er al dan niet een voorgift aanwezig is. Indien aan OCT een voorgift gegeven wordt terwijl een nog niet teruggegeven voorgift aanwezig is dan overschrijft de laatste voorgift de eerste. De voorgift wordt toegekend aan de globale variabele OCTBUFi.

```
procedure BACKOCT(I,X);value I,X;integer I,X;
begin if X<0 ∨ X≥256 then X:=BLANK;
   if I=1 then begin OCTBUF1:=X;BOPAST1:=true end else
   if I=2 then begin OCTBUF2:=X;BOPAST2:=true end else
   begin PRINTTEXT(‡DYNAMIC ERROR IN STREAMNUMBER‡);
      go to L END
   end
end BACKOCT(I,X);
```

1.2.2.1 integer procedure CHARF(I); integer I;

De procedure leest "zichtbare positiebedrukkingen" direct uit de buffer BUFi. De waarde aan de procedure toegekend is de CHARF waarde van het eerste MC-flexowritersymbool die een wagenverplaatsing tengevolge zou hebben met uitzondering van niet-zichtbare positiebedrukkingen (spaties aan het einde van een regel, spaties voor een stopcode, spaties voor bandeinde). Voor onderstreepte, doorbalkte en onderstreept doorbalkte symbolen is de CHARF waarde gelijk aan de CHARF waarde van het basissymbool vermeerderd met respectievelijk 128, 256 of 384.
Indien door CHARF een "stopcode" of "einde band" wordt gelezen krijgt CHARF de waarde -1. Wordt door CHARF een "foute pariteit" of "onbekende ponsing" gelezen dan krijgt CHARF de waarde -2.
Indien CHARF de waarde -1 gekregen heeft dient bij een volgende aanroep van een leesprocedure die van CHARF gebruik maakt, CHARF eerst geinitialiseerd te worden door een aanroep van INICHARF. Deze procedure wordt later besproken. Als CHARF de waarde -2 gekregen heeft kan een volgende aanroep van CHARF zonder meer plaatsvinden (dit in tegenstelling met de beschrijving van de werking van CHARF in RC-Inf 32 punt 4.1.5.1).
Als OCTPAST= true door gebruik van OCT heeft noch het aanroepen van CHARF noch het geven van voorgift aan CHARF effect. CHARF krijgt dan de waarde -1 en blijft deze behouden. CHARF maakt gebruik van de hulpprocedures SKIP, ASSIGN LOWER CHARFVALUE, ASSIGN UPPER CHARF VALUE.
SKIP "kijkt" of spaties al dan niet zichtbare positiebedrukkingen zijn, kunnen spaties overgeslagen worden dan wordt de pointer aangepast.
ASSIGN LOWER CHARFVALUE kent aan CHARF, bij een heersende LOWERCASE, een waarde toe. Deze waarde is afhankelijk van de gelezen OCTET waarde.
ASSIGN UPPER CHARFVALUE kent aan CHARF, bij een heersende UPPERCASE, een waarde toe. Deze waarde is afhankelijk van de gelezen OCTET waarde.
De booleans BOPASTi, BCPASTi en BSPASTi worden false.

De boolean BOPASTi wordt true in BACKOCT.
De boolean BCPASTi wordt true in BACKCHARF.
De boolean BSPASTi wordt true in BACKSYM.
BACKCHARF en BACKSYM worden later besproken.

Voorkomende locale variabelen in CHARF zijn:

OCTX hieraan wordt de OCTET waarde van de inhoud van de bufferplaats in BUFi aangewezen door pointer Pi toegekend.
CHAX hieraan wordt tijdelijk een CHARF waarde toegekend.
PP is een hulppointer die in SKIP gebruikt wordt om na te gaan of spaties al dan niet zichtbare positiebedrukkingen zijn.
De booleans UNDER en STROKE geven aan of er al dan niet een onderstreping of doorbalking aanwezig is.

Voorkomende globale variabelen, voor zover zij nog niet in voorafgaande procedures besproken zijn, zijn:

CHARFBUFi de waarde van deze variabele wordt aan CHARF toegekend als BCPASTi=true. Zie ook de later te bespreken procedure BACKCHARF.
UNDERSTROKE heeft de OCTET waarde voor een onderstreping of doorbalking.
LOWCASE heeft de OCTET waarde voor een lowercase ponsing.
UPCASE heeft de OCTET waarde voor een uppercase ponsing.
STOPCODE heeft de OCTET waarde voor een stopcode.
CAR RETURN heeft de OCTET waarde voor een overgang op nieuwe regel.
FLEXOi deze boolean wordt false als CHARF "stopcode" of "einde band" leest, FLEXOi wordt true als CHARF een casedefinitie of een overgang op nieuwe regel leest.
NEGONEi deze boolean wordt true als CHARF "stopcode" of "einde band" leest, NEGONEi wordt false als CHARF geinitialiseerd wordt door een aanroep van INICHARF.
LOWERCASEi en UPPERCASEi deze booleans dienen voor het bijhouden van de case.

```
integer procedure CHARF(I);integer I;

begin Boolean procedure SKIP;
   begin integer DUMMY,PP;
      boolean LLOWERCASE,UUPPERCASE,FFLEXO;
      if I=1 then
      begin LLOWERCASE:=LOWERCASE1;UUPPERCASE:=UPPERCASE1;
         FFLEXO:=FLEXO1;PP:=P1 + 2;
         for DUMMY:=0 while BUF1[PP]=BLANK ∨ BUF1[PP]=ERASE ∨
         BUF1[PP]=SPACE ∨ BUF1[PP]=LOWCASE ∨ BUF1[PP]=UPCASE do
         begin if BUF1[PP]=LOWCASE then
            begin LLOWERCASE:=FFLEXO:=true;UUPPERCASE:=false;
               PP:=PP + 1
            end else
            if BUF1[PP]=UPCASE then
            begin UUPPERCASE:=FFLEXO:=true;LLOWERCASE:=false;
               PP:=PP + 1
            end else PP:=PP + 2
         end;
         if BUF1[PP]=STOPCODE ∨ BUF1[PP]=TAPE END then
         begin P1:=PP;
            SKIP:=true
         end else
         if BUF1[PP]=CAR RETURN then
         begin P1:=PP;FLEXO1:=FFLEXO;LOWERCASE1:=LLOWERCASE;
            UPPERCASE1:=UUPPERCASE;SKIP:=true
         end else
         SKIP:=false
      end else
      begin LLOWERCASE:=LOWERCASE2;UUPPERCASE:=UPPERCASE2;
         FFLEXO:=FLEXO2;PP:=P2 + 2;
         for DUMMY:=0 while BUF2[PP]=BLANK ∨ BUF2[PP]=ERASE ∨
         BUF2[PP]=SPACE ∨ BUF2[PP]=LOWCASE ∨ BUF2[PP]=UPCASE do
         begin if BUF2[PP]=LOWCASE then
            begin LLOWERCASE:=FFLEXO:=true;UUPPERCASE:=false;
               PP:=PP + 1
```

```
        end else
        if BUF2[PP]=UPCASE then
        begin UUPPERCASE:=FFLEXO:=true;LLOWERCASE:=false;
           PP:=PP + 1
        end else PP:=PP + 2
      end;
      if BUF2[PP]=STOPCODE ∨ BUF2[PP]=TAPE END then
      begin P2:=PP;SKIP:=true end else
      if BUF2[PP]=CAR RETURN then
      begin P2:=PP;FLEXO2:=FFLEXO;LOWERCASE2:=LLOWERCASE;
        UPPERCASE2:=UUPPERCASE;SKIP:=true
      end else
      SKIP:=false
   end
end SKIP;

procedure ASSIGN LOWER CHARFVALUE;
begin CHAX:=if OCTX=1 ∨ OCTX=2 ∨ OCTX=4 ∨ OCTX=7 ∨ OCTX=8 then OCTX else
            if OCTX=19 ∨ OCTX=21 ∨ OCTX=22 ∨ OCTX=25 then OCTX-16 else
            if OCTX=97 ∨ OCTX=98 ∨ OCTX=100 ∨ OCTX=103 ∨ OCTX=104
            then OCTX-87 else
            if OCTX=115 ∨ OCTX=117 ∨ OCTX=118 ∨ OCTX=121 then OCTX-103 else
            if OCTX=81 ∨ OCTX=82 ∨ OCTX=84 ∨ OCTX=87 ∨ OCTX=88
            then OCTX-62 else
            if OCTX=67 ∨ OCTX=69 ∨ OCTX=70 ∨ OCTX=73 then OCTX-46 else
            if OCTX=50 ∨ OCTX=52 ∨ OCTX=55 ∨ OCTX=56 then OCTX-22 else
            if OCTX=35 ∨ OCTX=37 ∨ OCTX=38 ∨ OCTX=41 then OCTX-6 else
            if OCTX=32 then 0 else
            if OCTX=112 then 64 else
            if OCTX=107 then 88 else
            if OCTX=64 then 65 else
            if OCTX=91 then 87 else
            if OCTX=49 then 72 else
            if OCTX=59 then 89 else -2;
            if I=1 then P1:=P1 + 1 else P2:=P2 + 1;
            if UNDER ∧ CHAX≠-2 then CHAX:=CHAX + 128;
```

```
            if STROKE ∧ CHAX≠-2 then CHAX:=CHAX + 256;
            CHARF:=CHAX;
            UNDER:=STROKE:=false
end ASSIGN LOWER CHARFVALUE;

procedure ASSIGN UPPER CHARFVALUE;
begin CHAX:=if OCTX=97 ∨ OCTX=98 ∨ OCTX=100 ∨ OCTX=103 ∨ OCTX=104
            then OCTX-60 else
            if OCTX=115 ∨ OCTX=117 ∨ OCTX=118 ∨ OCTX=121 then OCTX-76 else
            if OCTX=81 ∨ OCTX=82 ∨ OCTX=84 ∨ OCTX=87 then OCTX-35 else
            if OCTX=67 ∨ OCTX=69 ∨ OCTX=70 ∨ OCTX=73 then OCTX-19 else
            if OCTX=50 ∨ OCTX=52 ∨ OCTX=55 ∨ OCTX=56 then OCTX+5 else
            if OCTX=35 ∨ OCTX=37 ∨ OCTX=38 ∨ OCTX=41 then OCTX+21 else
            if OCTX=32 ∨ OCTX=19 then OCTX+48 else
            if OCTX=1 ∨ OCTX=22 then OCTX+78 else
            if OCTX=112 then 121 else
            if OCTX=107 then 90 else
            if OCTX=64 then 76 else
            if OCTX=91 then 122 else
            if OCTX=49 then 74 else
            if OCTX=59 then 120 else
            if OCTX=2 then 66 else
            if OCTX=4 then 70 else
            if OCTX=21 then 91 else
            if OCTX=7 then 101 else
            if OCTX=25 then 99 else -2;
            if I=1 then P1:=P1 + 1 else P2:=P2 + 1;
            if UNDER ∧ CHAX≠-2 then CHAX:=CHAX + 128;
            if STROKE ∧ CHAX≠-2 then CHAX:=CHAX + 256;
            CHARF:=CHAX;
            UNDER:=STROKE:=false
end ASSIGN UPPER CHARFVALUE;

integer OCTX,CHAX;
boolean UNDER,STROKE;
```

```
UNDER:=STROKE:=false;
if I=1 then
begin if OCTPAST1 then CHARF:=-1 else
   if BCPAST1 then CHARF:=CHARFBUF1 else
   begin LO: OCTX:=BUF1[P1];
      if OCTX=BLANK ∨ OCTX=ERASE then
      begin P1:=P1 + 2;go to LO end;
      if ¬LOWERCASE1 ∧ ¬UPPERCASE1 ∧ OCTX=UNDERSTROKE
      then CHARF:=-2
      else if OCTX=LOWCASE ∨ OCTX=UPCASE ∨ OCTX=UNDERSTROKE then
      begin if OCTX=LOWCASE then
         begin LOWERCASE1:=FLEXO1:=true;UPPERCASE1:=false end else
         if OCTX=UPCASE then
         begin UPPERCASE1:=FLEXO1:=true;LOWERCASE1:=false end else
         if LOWERCASE1 then UNDER:=true else
         if UPPERCASE1 then STROKE:=true;
         P1:=P1 + 1;
         go to LO
      end else
      if OCTX=STOPCODE ∨ OCTX=TAPE END then
      begin if ¬UNDER ∧ ¬STROKE then
         begin FLEXO1:=false;NEGONE1:=true;CHARF:=-1 end else
         begin if UNDER then CHARF:=221 else
            if STROKE then CHARF:=349 else
            if UNDER ∧ STROKE then CHARF:=477;
            UNDER:=STROKE:=false
         end
      end else
      if OCTX=SPACE ∧ ¬UNDER ∧ ¬STROKE then
      begin if SKIP then
         begin if BUF1[P1]=STOPCODE ∨ BUF1[P1]=TAPE END then
            begin FLEXO1:=false;NEGONE1:=true;CHARF:=-1 end else
            begin if ¬LOWERCASE1 ∧ ¬UPPERCASE1 then
               FLEXO1:=LOWERCASE1:=true;
               CHARF:=119;
```

```
            P1:=P1 + 1
         end
      end else
      begin if ¬LOWERCASE1 ∧ ¬UPPERCASE1 then CHARF:=-2
                                          else CHARF:=93;
         P1:=P1 + 1;BUF1[P1]:=BUF1[P1]-1;
         if BUF1[P1]=0 then P1:=P1 + 1 else P1:=P1 - 1
      end
   end else
   if OCTX=SPACE then
   begin if ¬LOWERCASE1 ∧ ¬UPPERCASE1 then CHARF:=-2 else
      begin CHAX:=93;
         if UNDER then CHAX:=CHAX+128;
         if STROKE then CHAX:=CHAX+256;
         CHARF:=CHAX
      end;
      P1:=P1 + 1;BUF1[P1]:=BUF1[P1]-1;
      if BUF1[P1]=0 then P1:=P1 + 1 else P1:=P1 - 1;
      UNDER:=STROKE:=false
   end else
   if ¬NEGONE1 then
   begin if OCTX=CAR RETURN then
      begin if ¬LOWERCASE1 ∧ ¬UPPERCASE1 then
         FLEXO1:=LOWERCASE1:=true;
         if ¬UNDER ∧ ¬STROKE then
         begin CHARF:=119;P1:=P1 + 1 end else
         begin if UNDER then CHARF:=221 else
            if STROKE then CHARF:=349 else
            if UNDER ∧ STROKE then CHARF:=477;
            UNDER:=STROKE:=false
         end
      end else
      if LOWERCASE1 then ASSIGN LOWER CHARFVALUE else
      if UPPERCASE1 then ASSIGN UPPER CHARFVALUE else
      CHARF:=-2
   end else
```

```
      CHARF:=-1
   end;
   BOPAST1:=BCPAST1:=BSPAST1:=false
end else
if I=2 then
begin if OCTPAST2 then CHARF:=-1 else
   if BCPAST2 then CHARF:=CHARFBUF2 else
   begin L1: OCTX:=BUF2[P2];
      if OCTX=BLANK ∨ OCTX=ERASE then
      begin P2:=P2 + 2;go to L1 end;
      if ¬LOWERCASE2 ∧ ¬UPPERCASE2 ∧ OCTX=UNDERSTROKE
      then CHARF:=-2
      else if OCTX=LOWCASE ∨ OCTX=UPCASE ∨ OCTX=UNDERSTROKE then
      begin if OCTX=LOWCASE then
         begin LOWERCASE2:=FLEXO2:=true;UPPERCASE2:=false end else
         if OCTX=UPCASE then
         begin UPPERCASE2:=FLEXO2:=true;LOWERCASE2:=false end else
         if LOWERCASE2 then UNDER:=true else
         if UPPERCASE2 then STROKE:=true;
         P2:=P2 + 1;
         go to L1
      end else
      if OCTX=STOPCODE ∨ OCTX=TAPE END then
      begin if ¬UNDER ∧ ¬STROKE then
         begin FLEXO2:=false;NEGONE2:=true;CHARF:=-1 end else
         begin if UNDER then CHARF:=221 else
            if STROKE then CHARF:=349 else
            if UNDER ∧ STROKE then CHARF:=477;
            UNDER:=STROKE:=false
         end
      end else
      if OCTX=SPACE ∧ ¬UNDER ∧ ¬STROKE then
      begin if SKIP then
         begin if BUF2[P2]=STOPCODE ∨ BUF2[P2]=TAPE END then
            begin FLEXO2:=false;NEGONE2:=true;CHARF:=-1 end else
            begin if ¬LOWERCASE2 ∧ ¬UPPERCASE2 then
```

```
            FLEXO2:=LOWERCASE2:=true;
            CHARF:=119;
            P2:=P2 + 1
         end
      end else
      begin if ¬LOWERCASE2 ∧ ¬UPPERCASE2 then CHARF:=-2
                                          else CHARF:=93;
         P2:=P2 + 1;BUF2[P2]:=BUF2[P2]-1;
         if BUF2[P2]=0 then P2:=P2 + 1 else P2:=P2 - 1
      end
   end else
   if OCTX=SPACE then
   begin if ¬LOWERCASE2 ∧ ¬UPPERCASE2 then CHARF:=-2 else
      begin CHAX:=93;
         if UNDER then CHAX:=CHAX+128;
         if STROKE then CHAX:=CHAX+256;
         CHARF:=CHAX
      end;
      P2:=P2 + 1;
      BUF2[P2]:=BUF2[P2]-1;
      if BUF2[P2]=0 then P2:=P2 + 1 else P2:=P2 - 1;
      UNDER:=STROKE:=false
   end else
   if ¬NEGONE2 then
   begin if OCTX=CAR RETURN then
      begin if ¬LOWERCASE2 ∧ ¬UPPERCASE2 then
         FLEXO2:=LOWERCASE2:=true;
         if ¬UNDER ∧ ¬STROKE then
         begin CHARF:=119;P2:=P2 + 1 end else
         begin if UNDER then CHARF:=221 else
            if STROKE then CHARF:=349 else
            if UNDER ∧ STROKE then CHARF:=477;
            UNDER:=STROKE:=false
         end
      end else
      if LOWERCASE2 then ASSIGN LOWER CHARFVALUE else
```

```
            if UPPERCASE2 then ASSIGN UPPER CHARFVALUE else
            CHARF:=-2
         end else
         CHARF:=-1
      end;
      BCPAST2:=BCPAST2:=BSPAST2:=false
   end else
   begin PRINTTEXT(<DYNAMIC ERROR IN STREAMNUMBER>);
      go to L END
   end
end CHARF(I);
```

1.2.2.2 integer procedure CHARF1;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze door CHARF geleverd wordt.

```
integer procedure CHARF1;CHARF1:=CHARF(1);
```

1.2.2.3 integer procedure CHARF2;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze door CHARF geleverd wordt.

```
integer procedure CHARF2;CHARF2:=CHARF(2);
```

1.2.2.4 procedure BACKCHARF(I,X); value I,X; integer I,X;

Bij een eerstvolgende aanroep van CHARF komt de waarde van X weer ter beschikking. Dit wordt ook wel de voorgift voor CHARF genoemd. Met de boolean BCPASTi wordt bijgehouden of er al dan niet een voorgift aanwezig is. De boolean BCPASTi wordt in die procedures op false gezet die noch direct noch indirect CHARF aanroepen maar waarbij door deze procedures wel verder in de buffer gelezen wordt. Indien aan CHARF een voorgift gegeven wordt terwijl een nog niet teruggegeven voorgift aanwezig is dan overschrijft de laatst gegeven voorgift de eerste. Aanroep van BACKCHARF heeft geen effect als de laatste aanroep van CHARF de waarde -1 heeft opgeleverd. De laatst toegekende waarde aan CHARF wordt opgeslagen in CHARFBUFi.
CHARFBUFi is een globale variabele. Van de opgegeven waarde van X wordt bij

BACKCHARF niet gecontroleerd of deze tot het waardebereik van de betreffende leesprocedure behoort.

```
procedure BACKCHARF(I,X);value I,X;integer I,X;
if I=1 ∨ I=2 then
begin if I=1 then
   begin if ¬NEGONE1 ∧ ¬OCTPAST1 then
      begin CHARFBUF1:=X;BCPAST1:=true end;
   end else
   begin if ¬NEGONE2 ∧ ¬OCTPAST2 then
      begin CHARFBUF2:=X;BCPAST2:=true end;
   end
end else
begin PRINTTEXT(⊰DYNAMIC ERROR IN STREAMNUMBER⊱);
   go to L END
end BACKCHARF(I,X);
```

1.2.2.5 procedure INICHARF(I,POS,LOWER);value I,POS,LOWER;integer I,POS;
boolean LOWER;

Met INICHARF wordt CHARF geinitialiseerd.
Als parameters worden meegegeven:
- het geldende stroomnummer I
- de positie op de regel. Daar echter uitgegaan wordt van een MC-flexowriter zonder tabulatortoetsen, speelt de positie op de regel geen rol.
- de heersende case, dit in verband met de interpretatie van case afhankelijke ponsingen, voorafgaand aan de eerste op de band gegeven case definitie.

Als er geen ponsband meer voorhanden is, zal de procedure INICHARF volgband aanvragen. Ook worden door INICHARF de booleans OCTPASTi,NEGONEi,BOPASTi, BCPASTi,BSPASTi op false en FLEXOi op true gezet.

```
procedure INICHARF(I,POS,LOWER);value I,POS,LOWER;integer I,POS;
          Boolean LOWER;
if I=1 then
begin if BUF1[P1]=TAPE END then INIPATA(1) else
   if BUF1[P1]=STOPCODE then P1:=P1 + 1;
   NEGONE1:=OCTPAST1:=BOPAST1:=BCPAST1:=false;
   FLEXO1:=true;
   LOWERCASE1:=LOWER;
   UPPERCASE1:=¬LOWER
end else if I=2 then
begin if BUF2[P2]=TAPE END then INIPATA(2) else
   if BUF2[P2]=STOPCODE then P2:=P2 + 1;
   NEGONE2:=OCTPAST2:=BOPAST2:=BCPAST2:=false;
   FLEXO2:=true;
   LOWERCASE2:=LOWER;
   UPPERCASE2:=¬LOWER
end else
begin PRINTTEXT(‹DYNAMIC ERROR IN STREAMNUMBER›);
   go to L END
end INICHARF(I,POS,LOWER);
```

1.2.3.1 integer procedure SYM(I)

De waarde aan deze procedure toegekend is de SYM waarde van het eerstvolgende basissymbool op de band. SYM leest "de ponsband" door middel van CHARF. Als CHARF het einde van de heersende interpretatie meldt, (CHARF=-1) dan zoekt SYM onder het overslaan van blanks, stopcode en erases naar een nieuwe toe te kennen interpretatie ("UPCASE", "LOWCASE","CAR RETURN"). Wordt nu op de ponsband geen TAPE END aangetroffen en niet één van bovengenoemde waarden, dan levert SYM de waarde -1 af. Wordt echter "einde band" aangetroffen dan vraagt SYM volgband aan. Op de nieuwe band wordt het zoeken naar een geldende interpretatie toestand voortgezet. Een konsekwentie hiervan is, dat iedere door middel van SYM te lezen "band" moet beginnen met een case definitie of een overgang op een nieuwe regel. Als de ponsing eèn overgang op een nieuwe regel is, wordt de eerstvolgende ponsing geinterpreteerd alsof de lowercase toestand heerst. Bij het optreden van een foute pariteit of een onbekende ponsing levert een aanroep van SYM de waarde -1 af. Als er fouten in worddelimiters of onbekende CHARF waarden optreden, dan levert een aanroep van SYM de waarde -2 af. Worddelimiters welke hier worden toegelaten zijn de volgende onderstreepte basissymbolen uit het Algol rapport:
array begin Boolean comment do else end false for go to if integer label own procedure real step string switch then true until value while, en de volgende toevoegingen van onderstreepte symbolen aan het THE-systeem: algol and boolean complex correction goto go to invslash library mtape nlcr non not or progend.
Wanneer een fout in een worddelimiter wordt aangetroffen zal SYM via CHARF zover doorlezen (skippen) totdat een volgend symbool als SYM waarde is te interpreteren, mits dit symbool geen worddelimiter is. Een volgende aanroep van SYM levert dan de waarde toegekend aan dit herkenbare symbool. Het proces worddelimiter lezen wordt ingezet zodra CHARF één van de volgende waarden aflevert:
a b c d e f g i l m n o p r s t u v w B
Dit proces wordt beeindigd:

- Wanneer opeenvolging van zulke CHARF waarden een worddelimiter heeft opgeleverd.
- Zodra 11 onderstreepte letters opeenvolgend zijn gelezen en dit geen worddelimiter heeft opgeleverd wordt verder gelezen totdat een volgend symbool als SYM waarde is te interpreteren mits dit

symbool geen worddelimiter is.
- Zodra Charf een waarde aflevert die valt in het bereik van cijfers, letters en tekens.

Spaties, direkt voorafgaand aan een overgang op een nieuwe regel of een stopcode, worden door SYM niet afgeleverd; ze worden overgeslagen daar CHARF ze overslaat.
SYM maakt gebruik van de hulpprocedure WORDDELIMITER waarbij WORDDELIMITER weer gebruik maakt van WORDLENGTHk.
WORDDELIMITER leest zoveel onderstreepte karakters totdat een SYM-waarde toegekend kan worden ofwel leest tot het eerstvolgende symbool, wat geen onderstreept karakter is. SYM krijgt dan de waarde -2.
WORDLENGTHk "kijkt" of er een worddelimiter gevormd is, bestaande uit k karakters, zo ja dan wordt hieraan een SYM waarde toegekend, zo nee dan wordt afhankelijk van de waarde van B aan SYM wel of niet de waarde -2 toegekend.

Voorkomende locale variabelen in SYM zijn:

COUNTER deze variabele, houdt het aantal gelezen CHARF waarden bij, wanneer getracht wordt een worddelimiter te vinden, en de variabele wordt als index van het array gebruikt waarin de gelezen CHARF waarden worden bewaard.

arrayA[1:12] dit array dient als opslagplaats voor gelezen CHARF waarden, nodig om een worddelimiter te vinden.

CHAS aan deze variabele wordt tijdelijk de gelezen CHARF waarde toegekend.

B deze boolean wordt <u>false</u> als er geen CHARF waarden meer gelezen hoeven te worden om tot een SYM waarde te komen.

Voorkomende globale variabelen in SYM welke nog niet in de voorafgaande procedures besproken zijn, zijn:

SYMBUFi de waarde van deze variabele wordt aan SYM toegekend als BSPASTi=<u>true</u>. Zie ook de later te bespreken procedure BACKSYM.

De boolean BSPASTi wordt door SYM <u>false</u>.

```
integer procedure SYM(I);value I;integer I;
begin procedure WORDDELIMITER;
   begin procedure WORDLENGTH2;
      begin if A[1]=146 ∧ A[2]=143 then begin SYM:=94;B:=false end else
         if A[1]=141 ∧ A[2]=152 then begin SYM:=86;B:=false end else
         if A[1]=152 ∧ A[2]=155 then begin SYM:=79;B:=false end else
         if A[1]=144 ∧ A[2]=152 ∧ A[3]=93 then B:=true else
         if ¬B then SYM:=-2
      end;
      procedure WORDLENGTH3;
      begin if A[1]=142 ∧ A[2]=151 ∧ A[3]=141 then
         begin SYM:=105;B:=false end else
         if A[1]=143 ∧ A[2]=152 ∧ A[3]=155 then
         begin SYM:=82;B:=false end else
         if A[1]=138 ∧ A[2]=151 ∧ A[3]=141 then
         begin SYM:=80;B:=false end else
         if A[1]=151 ∧ A[2]=152 ∧ A[3]=151 then
         begin SYM:=76;B:=false end else
         if A[1]=151 ∧ A[2]=152 ∧ A[3]=157 then
         begin SYM:=76;B:=false end else
         if A[1]=152 ∧ A[2]=160 ∧ A[3]=151 then
         begin SYM:=106;B:=false end else
         if ¬B then SYM:=-2
      end;
      procedure WORDLENGTH4;
      begin if A[1]=157 ∧ A[2]=145 ∧ A[3]=142 ∧ A[4]=151 then
         begin SYM:=95;B:=false end else
         if A[1]=142 ∧ A[2]=149 ∧ A[3]=156 ∧ A[4]=142 then
         begin SYM:=96;B:=false end else
         if A[1]=156 ∧ A[2]=157 ∧ A[3]=142 ∧ A[4]=153 then
         begin SYM:=83;B:=false end else
         if A[1]=155 ∧ A[2]=142 ∧ A[3]=138 ∧ A[4]=149 then
         begin SYM:=107;B:=false end else
         if A[1]=157 ∧ A[2]=155 ∧ A[3]=158 ∧ A[4]=142 then
         begin SYM:=116;B:=false end else
         if A[1]=144 ∧ A[2]=152 ∧ A[3]=157 ∧ A[4]=144 then
```

```
    begin SYM:=81;B:=false end else
    if A[1]=151 ∧ A[2]=149 ∧ A[3]=140 ∧ A[4]=155 then
    begin SYM:=128;B:=false end else
    if ¬B then SYM:=-2
end;
procedure WORDLENGTH5;
begin if A[1]=139 ∧ A[2]=142 ∧ A[3]=144 ∧ A[4]=146 ∧ A[5]=151 then
    begin SYM:=104;B:=false end else
    if A[1]=143 ∧ A[2]=138 ∧ A[3]=149 ∧ A[4]=156 ∧ A[5]=142 then
    begin SYM:=117;B:=false end else
    if A[1]=138 ∧ A[2]=155 ∧ A[3]=155 ∧ A[4]=138 ∧ A[5]=162 then
    begin SYM:=111;B:=false end else
    if A[1]=160 ∧ A[2]=145 ∧ A[3]=146 ∧ A[4]=149 ∧ A[5]=142 then
    begin SYM:=85;B:=false end else
    if A[1]=158 ∧ A[2]=151 ∧ A[3]=157 ∧ A[4]=146 ∧ A[5]=149 then
    begin SYM:=84;B:=false end else
    if A[1]=149 ∧ A[2]=138 ∧ A[3]=139 ∧ A[4]=142 ∧ A[5]=149 then
    begin SYM:=114;B:=false end else
    if A[1]=159 ∧ A[2]=138 ∧ A[3]=149 ∧ A[4]=158 ∧ A[5]=142 then
    begin SYM:=115;B:=false end else
    if A[1]=144 ∧ A[2]=152 ∧ A[3]=221 ∧ A[4]=157 ∧ A[5]=152 then
    begin SYM:=81;B:=false end else
    if A[1]=144 ∧ A[2]=152 ∧ A[3]=93 ∧ A[4]=157 ∧ A[5]=152 then
    begin SYM:=81;B:=false end else
    if A[1]=150 ∧ A[2]=157 ∧ A[3]=138 ∧ A[4]=153 ∧ A[5]=142 then
    begin SYM:=123;B:=false end else
    if A[1]=138 ∧ A[2]=149 ∧ A[3]=144 ∧ A[4]=152 ∧ A[5]=149 then
    begin SYM:=130;B:=false end else
    if ¬B then SYM:=-2
end;
procedure WORDLENGTH6;
begin if A[1]=156 ∧ A[2]=157 ∧ A[3]=155 ∧ A[4]=146 ∧ A[5]=151 ∧
    A[6]=144 then begin SYM:=110;B:=false end else
    if A[1]=156 ∧ A[2]=160 ∧ A[3]=146 ∧ A[4]=157 ∧ A[5]=140 ∧
    A[6]=145 then begin SYM:=113;B:=false end else
    if ¬B then SYM:=-2
end;
```

```
procedure WORDLENGTH7;
begin if A[1]=146 ∧ A[2]=151 ∧ A[3]=157 ∧ A[4]=142 ∧ A[5]=144 ∧
   A[6]=142 ∧ A[7]=155 then begin SYM:=108;B:=false end else
   if A[1]=139 ∧ A[2]=152 ∧ A[3]=152 ∧ A[4]=149 ∧ A[5]=142 ∧
   A[6]=138 ∧ A[7]=151 then begin SYM:=109;B:=false end else
   if A[1]=166 ∧ A[2]=152 ∧ A[3]=152 ∧ A[4]=149 ∧ A[5]=142 ∧
   A[6]=138 ∧ A[7]=151 then begin SYM:=109;B:=false end else
   if A[1]=140 ∧ A[2]=152 ∧ A[3]=150 ∧ A[4]=150 ∧ A[5]=142 ∧
   A[6]=151 ∧ A[7]=157 then begin SYM:=97;B:=false end else
   if A[1]=153 ∧ A[2]=155 ∧ A[3]=152 ∧ A[4]=144 ∧ A[5]=142 ∧
   A[6]=151 ∧ A[7]=141 then begin SYM:=118;B:=false end else
   if A[1]=140 ∧ A[2]=152 ∧ A[3]=150 ∧ A[4]=153 ∧ A[5]=149 ∧
   A[6]=142 ∧ A[7]=161 then begin SYM:=92;B:=false end else
   if A[1]=149 ∧ A[2]=146 ∧ A[3]=139 ∧ A[4]=155 ∧ A[5]=138 ∧
   A[6]=155 ∧ A[7]=162 then begin SYM:=124;B:=false end else
   if ¬B then SYM:=-2
end;
procedure WORDLENGTH8;
begin if A[1]=146 ∧ A[2]=151 ∧ A[3]=169 ∧ A[4]=156 ∧ A[5]=149 ∧
   A[6]=138 ∧ A[7]=156 ∧ A[8]=145 then
   begin SYM:=125;B:=false end else
   if ¬B then SYM:=-2
end;
procedure WORDLENGTH9;
begin if A[1]=153 ∧ A[2]=155 ∧ A[3]=152 ∧ A[4]=140 ∧ A[5]=142 ∧
   A[6]=141 ∧ A[7]=158 ∧ A[8]=155 ∧ A[9]=142 then
   begin SYM:=112;B:=false end else
   if ¬B then SYM:=-2
end;
procedure WORDLENGTH10;
begin if A[1]=140 ∧ A[2]=152 ∧ A[3]=155 ∧ A[4]=155 ∧ A[5]=142 ∧
   A[6]=140 ∧ A[7]=157 ∧ A[8]=146 ∧ A[9]=152 ∧ A[10]=151 then
   begin SYM:=131;B:=false end else
   if ¬B then SYM:=-2
end;
```

```
    integer COUNTER,DUMMY;
    integer array A[1:12];
    Boolean B;
    B:=true;COUNTER:=0;
    for DUMMY:=0 while B do
    begin COUNTER:=COUNTER+1;
       A[COUNTER]:=CHAS:=CHARF(I);
       B:=¬((0≤CHAS ∧ CHAS≤35) ∨ (37≤CHAS ∧ CHAS≤62) ∨ (64≤CHAS ∧ CHAS≤67) ∨
          CHAS=70 ∨ CHAS=72 ∨ CHAS=74 ∨ CHAS=76 ∨ CHAS=79 ∨ CHAS=80 ∨
          (87≤CHAS ∧ CHAS≤91) ∨ CHAS=93 ∨ (98≤CHAS ∧ CHAS≤101) ∨
          (119≤CHAS ∧ CHAS≤122) ∨ CHAS=198 ∨ CHAS=200 ∨ CHAS=202 ∨ CHAS=204 ∨
          CHAS=215 ∨ CHAS=218 ∨ CHAS=221 ∨ CHAS=326 ∨ CHAS=328 ∨ CHAS=330 ∨
          CHAS=336 ∨ CHAS=349);
       if COUNTER=2 ∧ ¬B then SYM:=-2 else
       if COUNTER=3 then WORDLENGTH2 else
       if COUNTER=4 then WORDLENGTH3 else
       if COUNTER=5 then WORDLENGTH4 else
       if COUNTER=6 then WORDLENGTH5 else
       if COUNTER=7 then WORDLENGTH6 else
       if COUNTER=8 then WORDLENGTH7 else
       if COUNTER=9 then WORDLENGTH8 else
       if COUNTER=10 then WORDLENGTH9 else
       if COUNTER=11 then WORDLENGTH10 else
       if COUNTER=11 ∧ B then
       begin SYM:=-2;
          for DUMMY:=0 while B do
          begin CHAS:=CHARF(I);
             B:=¬((0≤CHAS ∧ CHAS≤35) ∨ (37≤CHAS ∧ CHAS≤62) ∨ (64≤CHAS ∧ CHAS≤67)
                ∨ CHAS=70 ∨ CHAS=72 ∨ CHAS=74 ∨ CHAS=76 ∨ CHAS=79 ∨ CHAS=80 ∨
                (87≤CHAS ∧ CHAS≤91) ∨ CHAS=93 ∨ (98≤CHAS ∧ CHAS≤101) ∨
                (119≤CHAS ∧ CHAS≤122) ∨ CHAS=198 ∨ CHAS=200 ∨ CHAS=202 ∨
                CHAS=204 ∨ CHAS=215 ∨ CHAS=218 ∨ CHAS=221 ∨ CHAS=326 ∨
                CHAS=328 ∨ CHAS=330 ∨ CHAS=336 ∨ CHAS=349)
          end
       end
    end;
    BACKCHARF(I,CHAS)
end WORDDELIMITER;
```

```
if I=1 ∨ I=2 then
begin if I=1 ∧ BSPAST1 then begin SYM:=SYMBUF1;BSPAST1:=false end else
   if I=2 ∧ BSPAST2 then begin SYM:=SYMBUF2;BSPAST2:=false end else
   begin integer CHAS;
      L2: CHAS:=if I=1 then CHARF(1) else CHARF(2);
      if CHAS=-2 then SYM:=-1 else
      if CHAS=-1 then
      begin INICHARF(I,POS,LOWER);
         if I=1 then
         begin LOWERCASE1:=UPPERCASE1:=FLEXO1:=false end else
         begin LOWERCASE2:=UPPERCASE2:=FLEXO2:=false end;
         go to L2
      end else
      if (0≤CHAS ∧ CHAS≤35) ∨ (37≤CHAS ∧ CHAS≤62) ∨ (64≤CHAS ∧ CHAS≤67) ∨
      CHAS=70 ∨ CHAS=72 ∨ CHAS=74 ∨ CHAS=76 ∨ CHAS=79 ∨ CHAS=80 ∨
      (87≤CHAS ∧ CHAS≤91) ∨ CHAS=93 ∨ (98≤CHAS ∧ CHAS≤101) ∨
      (119≤CHAS ∧ CHAS≤122) then SYM:=CHAS else
      if CHAS=200 ∨ CHAS=202 then SYM:=CHAS-127 else
      if CHAS=221 then SYM:=126 else
      if CHAS=349 then SYM:=127 else
      if CHAS=218 then SYM:=68 else
      if CHAS=336 then SYM:=69 else
      if CHAS=326 then SYM:=71 else
      if CHAS=198 then SYM:=77 else
      if CHAS=204 then SYM:=78 else
      if CHAS=328 then SYM:=102 else
      if CHAS=330 then SYM:=103 else
      if CHAS=215 then SYM:=129 else
      if (138≤CHAS ∧ CHAS≤144) ∨ CHAS=146 ∨ (149≤CHAS ∧ CHAS≤153) ∨
         (155≤CHAS ∧ CHAS≤160) ∨ CHAS=166 then
      begin BACKCHARF(I,CHAS);WORDDELIMITER end else SYM:=-2
   end
end else
begin PRINTTEXT(‹DYNAMIC ERROR IN STREAMNUMBER›);
   go to L END
end
end SYM(I);
```

1.2.3.2 integer procedure SYM1;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze door SYM afgeleverd wordt.

```
integer procedure SYM1;SYM1:=SYM(1);
```

1.2.3.3 integer procedure SYM2;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze door SYM afgeleverd wordt.

```
integer procedure SYM2;SYM2:=SYM(2);
```

1.2.3.4 integer procedure RESYM;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze uitsluitend via de eerste lezerstroom door SYM afgeleverd wordt.

```
integer procedure RESYM;RESYM:=SYM(1);
```

1.2.3.5 procedure BACKSYM(I,X);value I,X;integer I,X;

Bij een eerstvolgende aanroep van SYM komt de waarde van X weer ter beschikking. Dit wordt ook wel de voorgift voor SYM genoemd. Met de boolean BSPASTi wordt bijgehouden of er al dan niet een voorgift aanwezig is. Indien aan SYM een voorgift gegeven wordt terwijl een nog niet teruggegeven voorgift aanwezig is dan overschrijft de laatste voorgift de eerste. De laatst toegekende waarde aan SYM wordt opgeslagen in SYMBUFi.SYMBUFi is een globale variabele. Van de opgegeven waarde van X wordt bij BACKSYM niet gecontroleerd of deze tot het waardebereik van de betreffende leesprocedure behoort.

```
procedure BACKSYM(I,X);value I,X;integer I,X;
if I=1 then
begin SYMBUF1:=X;BCPAST1:=true end else
if I=2 then
begin SYMBUF2:=X;BCPAST2:=true end else
begin PRINTTEXT(‹DYNAMIC ERROR IN STREAMNUMBER›);
   go to L END
end BACKSYM(I,X);
```

1.2.4.1 real procedure NUM(I); integer I;

De procedure NUM leest uit de buffer BUF door middel van SYM. De waarde aan deze procedure toegekend is de NUM waarde van het eerstvolgende getal uit de buffer BUF. Deze getallen moeten voldoen aan de X-8 THE conventies voor getallen. Deze conventies zijn:

- een getal begint met een getalopener.
- als getalopener fungeren + - . ₁₀ 0 1 2 3 4 5 6 7 8 9
- als lay-out symbool is een enkele spatie binnen een getal overal toegestaan.
- twee of meer spaties binnen een getal zijn slechts toegestaan na het teken van het getal, een laag tientje en na het teken van de exponent.

Het lezen van een getal wordt beeindigd bij het aantreffen van een getal-afsluiter. Als getalafsluiter fungeren:

- het teken van het volgende getal.
- een overgang naar een nieuwe regel.
- twee of meer spaties, behalve op de binnen een getal toegestane plaatsen.
- een reeks algolsymbolen tussen apostrofes.
- een reeks basissymbolen waarin geen cijfer, teken, punt, laag tientje of een oneven aantal apostrofes in voorkomt.

Daar spaties, voorafgaand aan een stopcode of einde band door CHARF niet meer worden doorgemeld zullen twee spaties niet zonder meer kunnen fungeren als afsluiter van het laatste getal uit de buffer. Het laatste getal dient derhalve te worden afgesloten met een overgang op een nieuwe regel, dan wel met een of ander zichtbaar symbool, zodat aan de print-out van de buffer de correctheid is te verifieren.

Een getal kan dus de volgende gedaante hebben:

teken gevolgd door
meerdere spaties gevolgd door
meerdere cijfers onderling gescheiden door maximaal 1 spatie gevolgd door
punt al dan niet door een spatie gescheiden van het voorafgaande en volgende cijfer gevolgd door
meerdere cijfers onderling gescheiden door maximaal 1 spatie gevolgd door
laag tientje al dan niet gescheiden door een spatie van het voorafgaande cijfer gevolgd door
meerdere spaties gevolgd door
teken gevolgd door
meerdere spaties gevolgd door
meerdere cijfers onderling gescheiden door maximaal 1 spatie

Het ontwikkelde algorithme gaat successievelijk al deze mogelijkheden na. Indien bij aanroep van NUM een openingssymbool niet gevolgd wordt door een syntactisch correct getal slaat NUM alarm. Getalopeners en andere willekeurige reeksen basissymbolen voorkomend tussen apostrofes ' ' worden door NUM overgeslagen als commentaar. Een aanroep van SYM na NUM levert de SYM waarde van het eerste symbool van de getalafsluiter af. Indien de getalafsluiter bestaat uit 2 of meer spaties zal aanroep van SYM na NUM de SYM-waarde van de tweede spatie afleveren.
Bij de X-8 THE is de waarde aan de procedure toegekend gelijk aan die van het eerstvolgende getal uit de buffer met een relatieve precisie van 12 decimalen. Gehele waarden, absoluut kleiner 2/40-1 (=1099511627775), worden exact toegewezen.
Daar het teken van het volgende getal als getalafsluiter gebruikt kan worden dient het teveel gelezen basissymbool als voorgift van SYM te worden beschouwd. Dit houdt in dat als men de buffer sequentieel wil lezen "zonder overslaan", dat men na aanroep van NUM eerst een aanroep van SYM moet doen en daarna pas een aanroep van b.v. CHARF.
Door NUM wordt ook het aantal succesvolle aanroepen van NUM bijgehouden. Dit gegeven zal bij het optreden van een dynamische fout worden uitgevoerd ter precisering van het punt waar de fout optrad.

Voorkomende locale variabelen in NUM zijn:

NUMBUF de waarde van deze variabele, het te lezen getal, wordt aan NUM toegekend.

SYMS hieraan wordt de SYM waarde van een te lezen symbool toegekend.

NUMBER deze boolean wordt <u>false</u> als er een oneven aantal malen een apostrofe gelezen wordt.

DECIMAL NUMBER deze boolean wordt <u>true</u> als een decimaal gedeelte in het te lezen getal aanwezig is.

De overige locale variabelen EXPONENT, UNSIGNED INTEGER, SIGN en B laten zich uit de tekst begrijpen.
N.B. de boolean B wordt in meerdere betekenissen gebruikt.

Voorkomende globale variabelen in NUM zijn:

APOSTROFE heeft de SYM waarde voor een apostrofe.

POS heeft de SYM waarde voor een plus.

NEG heeft de SYM waarde voor een min.

POINT heeft de SYM waarde voor een punt.
LOWTEN heeft de SYM waarde voor een laag tientje.
SPASE heeft de SYM waarde voor een spatie.

```
real procedure NUM(I);integer I;
begin procedure ALARM;
   begin NLCR;
      PRINTTEXT(⟨SYNTACTIC ERROR IN NUMBERTAPE⟩);NLCR;
      PRINTTEXT(⟨TOTAL OF NUMBERS READ⟩);
      PRINT(NUMBERS);
      go to L END
   end ALARM;
   real NUMBUF;
   if I=1 ∨ I=2 then
   begin integer DUMMY,EXPONENT,SYMS,SIGN;
      real UNSIGNED INTEGER;
      Boolean B,NUMBER,DECIMAL NUMBER;
      B:=NUMBER:=true;
      DECIMAL NUMBER:=false;
      NUMBUF:=0;
      for DUMMY:=0 while B do
      begin SYMS:=SYM(I);
         if SYMS=APOSTROFE then NUMBER:=¬NUMBER else
         B:=¬(((0≤SYMS ∧ SYMS≤9) ∨ SYMS=POS ∨ SYMS=NEG ∨ SYMS=POINT ∨
             SYMS=LOWTEN) ∧ NUMBER)
      end;
      SIGN:=if SYMS=NEG then -1 else 1;
      B:=(SYMS=POS ∨ SYMS=NEG);
      for DUMMY:=0 while B do
      begin SYMS:=SYM(I);B:=SYMS=SPASE end;
      if SYMS≠POINT ∧ SYMS≠LOWTEN ∧ (SYMS<0 ∨ SYMS>9) then ALARM else
      if SYMS≠POINT ∧ SYMS≠LOWTEN then
      begin UNSIGNED INTEGER:=SYMS;B:=true;
         for DUMMY:=0 while B do
         begin SYMS:=SYM(I);
            if SYMS=SPASE then SYMS:=SYM(I) else
```

```
      if SYMS<0 ∨ SYMS>9 then B:=false else
      UNSIGNED INTEGER:=10×UNSIGNED INTEGER + SYMS
   end;
   NUMBUF:=SIGN×UNSIGNED INTEGER;
   DECIMAL NUMBER:=true
end else
if SYMS=POINT then
begin SYMS:=SYM(I);
   if SYMS=SPASE then SYMS:=SYM(I) else
   if SYMS<0 ∨ SYMS>9 then ALARM else
   begin UNSIGNED INTEGER:=SYMS;EXPONENT:=1;B:=true;
      for DUMMY:=0 while B do
      begin SYMS:=SYM(I);
         if SYMS=SPASE then SYMS:=SYM(I) else
         if SYMS<0 ∨ SYMS>9 then B:=false else
         begin UNSIGNED INTEGER:=10×UNSIGNED INTEGER + SYMS;
            EXPONENT:=EXPONENT + 1
         end
      end;
      UNSIGNED INTEGER:=UNSIGNED INTEGER/10↑EXPONENT
   end;
   NUMBUF:=NUMBUF + SIGN×UNSIGNED INTEGER;
   DECIMAL NUMBER:=true
end else
if SYMS=LOWTEN then
begin integer SIGN;
   B:=true;
   for DUMMY:=0 while B do
   begin SYMS:=SYM(I);
      if SYMS≠SPASE ∧ (SYMS<0 ∨ SYMS>9) ∧ SYMS≠POS ∧ SYMS≠NEG
      then ALARM
      else B:=SYMS=SPASE
   end;
   SIGN:=if SYMS=NEG then -1 else 1;
   B:=(SYMS=POS ∨ SYMS=NEG);
   for DUMMY:=0 while B do
```

```
            begin SYMS:=SYM(I);B:=SYMS=SPASE end;
            if SYMS<0 V SYMS>9 then ALARM else
            begin UNSIGNED INTEGER:=SYMS;B:=true;
               for DUMMY:=0 while B do
               begin SYMS:=SYM(I);
                  if SYMS=SPASE then SYMS:=SYM(I);
                  if SYMS<0 V SYMS>9 then B:=false else
                  UNSIGNED INTEGER:=10×UNSIGNED INTEGER + SYMS
               end
            end;
            UNSIGNED INTEGER:=10↑(SIGN×UNSIGNED INTEGER);
            NUMBUF:=(if ¬DECIMAL NUMBER then SIGN else NUMBUF)×UNSIGNED INTEGER
         end
      end;
      BACKSYM(I,SYMS);
      NUMBERS:=NUMBERS + 1;
      NUM:=NUMBUF
   end NUM(I);
```

1.2.4.2 real procedure NUM1;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze door NUM geleverd wordt.

```
real procedure NUM1;NUM1:=NUM(1);
```

1.2.4.3 real procedure NUM2;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze door NUM geleverd wordt.

```
real procedure NUM2;NUM2:=NUM(2);
```

1.2.4.4 real procedure READ;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze uitsluitend via de eerste lezerstroom door NUM geleverd wordt.

```
real procedure READ;READ:=NUM(1);
```

1.2.4.5 real procedure read;

De waarde aan deze procedure toegekend is gelijk aan de waarde zoals deze uitsluitend via de eerste lezerstroom door NUM geleverd wordt.

```
real procedure read;read:=NUM(1);
```

1.2.5 procedure SKIPPATA;value I;integer I;

Met deze procedure wordt een resterend gedeelte van een band geskipt. Een nu volgende aanroep van REHEP of CHARF levert de waarde -1 af.

```
procedure SKIPPATA(I);value I;integer I;
begin if I=1 then for I:=I while BUF1[P1]≠-1 do P1:=P1 + 1 else
   if I=2 then for I:=I while BUF2[P2]≠-1 do P2:=P2 + 1 else
   begin PRINTTEXT(‹DYNAMIC ERROR IN STREAMNUMBER›);
      go to L END
   end
end SKIPPATA(I);
```

1.2.6 integer procedure SYMINSP(I); value I; integer I;

De waarde aan deze procedure toegekend is 0,1,2 of 3

0 Bij de laatste aanroep van CHARF voor SYMINSP is geen stopcode of einde band gelezen. Bij de aanroep van CHARF door SYMINSP blijkt dat geen stopcode of einde band gelezen wordt.

of

Bij de laatste aanroep van CHARF voor SYMINSP is wel een stopcode of einde band gelezen. Bij de aanroep van CHARF door SYMINSP blijkt dat een case definitie of een overgang op een nieuwe regel gelezen wordt.

1 Bij de laatste aanroep van CHARF voor SYMINSP is geen stopcode of einde band gelezen. Bij de aanroep van CHARF door SYMINSP blijkt dat er een stopcode of einde band gelezen wordt.

2 Bij de laatste aanroep van CHARF voor SYMINSP is wel een stopcode of einde band gelezen. Bij de aanroep van SYMINSP blijkt dat einde band gelezen wordt.

3 Bij de laatste aanroep van CHARF voor SYMINSP is een stopcode of einde band gelezen. Bij de aanroep van SYMINSP blijkt dat op de band een ponsing aangetroffen wordt die niet tot codekeuze kan leiden.

Dit is het geval wanneer na een stopcode of aan het begin van de band de case definitie of de overgang op een nieuwe regel wordt vergeten.

```
integer procedure SYMINSP(I);value I;integer I;
if I=1 ∨ I=2 then
begin integer PP,COUNTER,CHAS,X;
   Boolean OCOCTPAST,BOBOPAST,BCBCPAST,BSBSPAST,LOLOWERCASE,UPUPPERCASE,
   FLFLEXO,NENEGONE;
   if I=1 then PP:=P1 else PP:=P2;
   begin integer array BUBUF[PP:131071];
      if I=1 then
      begin OCOCTPAST:=OCTPAST1;BOBOPAST:=BOPAST1;BCBCPAST:=BCPAST1;
         BSBSPAST:=BSPAST1;LOLOWERCASE:=LOWERCASE1;UPUPPERCASE:=UPPERCASE1;
         FLFLEXO:=FLEXO1;NENEGONE:=NEGONE1;COUNTER:=P1-1;
      L3:  COUNTER:=COUNTER + 1;X:=BUF1[COUNTER];BUBUF[COUNTER]:=X;
         if X≠-1 then go to L3;
         CHAS:=CHARF(1);
         if FLEXO1 then SYMINSP:=0 else
         if FLFLEXO then SYMINSP:=1 else
         if BUF1[P1]=TAPE END then SYMINSP:=2 else SYMINSP:=3;
         P1:=PP;COUNTER:=PP-1;
      L4:  COUNTER:=COUNTER + 1;X:=BUBUF[COUNTER];BUF1[COUNTER]:=X;
         if X≠-1 then go to L4;
         NEGONE1:=NENEGONE;FLEXO1:=FLFLEXO;UPPERCASE1:=UPUPPERCASE;
         LOWERCASE1:=LOLOWERCASE;BSPAST1:=BSBSPAST;BCPAST1:=BCBCPAST;
         BOPAST1:=BOBOPAST;OCTPAST1:=OCOCTPAST;
      end else
      begin OCOCTPAST:=OCTPAST2;BOBOPAST:=BOPAST2;BCBCPAST:=BCPAST2;
         BSBSPAST:=BSPAST2;LOLOWERCASE:=LOWERCASE2;UPUPPERCASE:=UPPERCASE2;
         FLFLEXO:=FLEXO2;NENEGONE:=NEGONE2;COUNTER:=P2-1;
      L5:  COUNTER:=COUNTER + 1;X:=BUF2[COUNTER];BUBUF[COUNTER]:=X;
         if X≠-1 then go to L5;
         CHAS:=CHARF(2);
         if FLEXO2 then SYMINSP:=0 else
         if FLFLEXO then SYMINSP:=1 else
         if BUF2[P2]=TAPE END then SYMINSP:=2 else SYMINSP:=3;
         P2:=PP;COUNTER:=PP-1;
```

```
        L6:  COUNTER:=COUNTER + 1;X:=BUBUF[COUNTER];BUF2[COUNTER]:=X;
             if X≠-1 then go to L6;
             NEGONE2:=NENEGONE;FLEXO2:=FLFLEXO;UPPERCASE2:=UPUPPERCASE;
             LOWERCASE2:=LOLOWERCASE;BSPAST2:=BSBSPAST;BCPAST2:=BCBCPAST;
             BOPAST2:=BOBOPAST;OCTPAST2:=OCOCTPAST;
          end
       end
    end else
    begin PRINTTEXT(⟨DYNAMIC ERROR IN STREAMNUMBER⟩);
       go to L END
    end SYMINSP(I);
```

1.2.7 integer procedure NUMINSP(I); value I; integer I;

De waarde aan deze procedure toegekend is 0, 1 of 2.

0 Er komt op deze ponsband, afgezien van irrelevante basissymbolen, eerst een getalopener voor. Bij een aanroep van SYM, hetzij vanuit NUM, hetzij door de programmeur, zal deze getalopener afgeleverd worden.

1 Er komt eerst een marker (hiervoor is in het huidige THE-systeem het vraagteken "?" gekozen) die gebruikt wordt om het einde van de getallenrij aantegeven. De eerstvolgende aanroep van SYM levert het symbool af volgend op deze marker.

2 Er komt voor bandeinde noch een getalopener, noch een marker. Bij de eerstvolgende aanroep van SYM zal SYM tot bandaanvrage overgaan.
N.B. Markers en getalopeners, ingebed tussen apostrofes, worden niet als zodanig beschouwd.

```
integer procedure NUMINSP(I);value I;integer I;
if I=1 ∨ I=2 then
begin integer PP,COUNTER,CHAS,X;
   Boolean OCOCTPAST,BOBOPAST,BCBCPAST,BSBSPAST,LOLOWERCASE,UPUPPERCASE,
   FLFLEXO,NENEGONE,B,NUMBER;
   B:=NUMBER:=true;
   if I=1 then PP:=P1 else PP:=P2;
   begin integer array BUBUF[PP:131071];
      if I=1 then
      begin OCOCTPAST:=OCTPAST1;BOBOPAST:=BOPAST1;BCBCPAST:=BCPAST1;
```

```
   BSBSPAST:=BSPAST1;LOLOWERCASE:=LOWERCASE1;UPUPPERCASE:=UPPERCASE1;
   FLFLEXO:=FLEXO1;NENEGONE:=NEGONE1;COUNTER:=P1-1;
L7:  COUNTER:=COUNTER + 1;X:=BUF1[COUNTER];BUBUF[COUNTER]:=X;
   if X≠-1 then go to L7;
   for COUNTER:=0 while ¬B do
   begin CHAS:=CHARF(1);
      if CHAS=APOSTROFE then NUMBER:=¬NUMBER else
      B:=((0≤CHAS ∧ CHAS≤9) ∨ CHAS=POS ∨ CHAS=NEG ∨ CHAS=POINT ∨
         CHAS=LOWTEN ∨ CHAS=MARKER) ∧ NUMBER ∨ CHAS=TAPE END
   end;
   if CHAS=MARKER then NUMINSP:=1 else
   if BUF1[P1]≠TAPE END then NUMINSP:=0 else NUMINSP:=2;
   P1:=PP;COUNTER:=PP-1;
L8:  COUNTER:=COUNTER + 1;X:=BUBUF[COUNTER];BUF1[COUNTER]:=X;
   if X≠-1 then go to L8;
   NEGONE1:=NENEGONE;FLEXO1:=FLFLEXO;UPPERCASE1:=UPUPPERCASE;
   LOWERCASE1:=LOLOWERCASE;BSPAST1:=BSBSPAST;BCPAST1:=BCBCPAST;
   BOPAST1:=BOBOPAST;OCTPAST1:=OCOCTPAST;
end else
begin OCOCTPAST:=OCTPAST2;BOBOPAST:=BOPAST2;BCBCPAST:=BCPAST2;
   BSBSPAST:=BSPAST2;LOLOWERCASE:=LOWERCASE2;UPUPPERCASE:=UPPERCASE2;
   FLFLEXO:=FLEXO2;NENEGONE:=NEGONE2;COUNTER:=P2-1;
L9:  COUNTER:=COUNTER + 1;X:=BUF2[COUNTER];BUBUF[COUNTER]:=X;
   if X≠-1 then go to L9;
   for COUNTER:=0 while ¬B do
   begin CHAS:=CHARF(2);
      if CHAS=APOSTROFE then NUMBER:=¬NUMBER else
      B:=((0≤CHAS ∧ CHAS≤9) ∨ CHAS=POS ∨ CHAS=NEG ∨ CHAS=POINT ∨
         CHAS=LOWTEN ∨ CHAS=MARKER) ∧ NUMBER ∨ CHAS=TAPE END
   end;
   if CHAS=MARKER then NUMINSP:=1 else
   if BUF2[P2]≠TAPE END then NUMINSP:=0 else NUMINSP:=2;
   P2:=PP;COUNTER:=PP-1;
La10:  COUNTER:=COUNTER + 1;X:=BUBUF[COUNTER];BUF2[COUNTER]:=X;
   if X≠-1 then go to La10;
   NEGONE2:=NENEGONE;FLEXO2:=FLFLEXO;UPPERCASE2:=UPUPPERCASE;
```

```
            LOWERCASE2:=LOLOWERCASE;BSPAST2:=BSBSPAST;BCPAST2:=BCBCPAST;
            BOPAST2:=BOBOPAST;OCTPAST2:=OCOCTPAST
         end
      end
   end else
   begin PRINTTEXT(‹DYNAMIC ERROR IN STREAMNUMBER›);
      go to L END
   end NUMINSP(I);
```

1.2.8 procedure GLOBAL VARIABLES;

In de procedure worden aan de globale integer variabelen getalwaarden toegekend die overeenkomen met OCTET waarden en SYM waarden. Verder worden globale boolean variabelen geinitialiseerd.

```
procedure GLOBAL VARIABLES;
begin BLANK:=0;SPACE:=16;ERASE:=127;TAPE END:=-1;STOPCODE:=11;LOWCASE:=122;
   UPCASE:=124;UNDERSTROKE:=14;CAR RETURN:=26;POS:=64;NEG:=65;POINT:=88;
   LOWTEN:=89;SPASE:=93;APOSTROFE:=120;MARKER:=122;
   NUMBERS:=P1:=P2:=0;BUF1[P1]:=BUF2[P2]:=-1;
   OCTPAST1:=OCTPAST2:=BOPAST1:=BOPAST2:=BCPAST1:=BCPAST2:=BSPAST1:=
   BSPAST2:=LOWERCASE1:=LOWERCASE2:=UPPERCASE1:=UPPERCASE2:=FLEXO1:=
   FLEXO2:=NEGONE1:=NEGONE2:=false
end GLOBAL VARIABLES;
```

1.2.9 Het omvattende blok voor de systeeminvoerprocedures.

In dit omvattende blok zijn ondermeer de globale variabelen vermeld.

```
begin integer P1,P2,OCTBUF1,OCTBUF2,CHARFBUF1,CHARFBUF2,SYMBUF1,SYMBUF2,
   NUMBERS,BLANK,SPACE,ERASE,TAPE END,STOPCODE,LOWCASE,UPCASE,UNDERSTROKE,
   CAR RETURN,POS,NEG,POINT,LOWTEN,SPASE,APOSTROFE,MARKER;
   integer array BUF1[0:131071],BUF2[0:131071];
   Boolean OCTPAST1,OCTPAST2,BOPAST1,BOPAST2,BCPAST1,BCPAST2,BSPAST1,
   BSPAST2,LOWERCASE1,LOWERCASE2,UPPERCASE1,UPPERCASE2,FLEXO1,FLEXO2,
   NEGONE1,NEGONE2;

   < systeeminvoerprocedures >;

   GLOBAL VARIABLES;

   < gebruikersprogramma >.
L END
end;
```

HOOFDSTUK 2

Systeemuitvoerprocedures voor de regeldrukker bij de X-8 (THE)

2.1 Voorkomende regeldrukkerprocedures, die later nog besproken zullen worden, zijn:

```
procedure PRCHAR(N); value N; integer N;
procedure PRSYM(N); value N; integer N;
procedure NLCR;
procedure CARRIAGE(N); value N; integer N;
procedure NEWPAGE;
integer procedure LINENUMBER;
procedure SPACE(N); value N; integer N;
procedure TAB;
procedure FLOT(N,M,X); value N,M,X; integer N,M; real X;
procedure FIXT(N,M,X); value N,M,X; integer N,M; real X;
procedure ABSFIXT(N,M,X); value N,M,X; integer N,M; real X;
procedure PRINT(X); value X; real X;
procedure print(X); value X; real X;
procedure PRINTTEXT(S); string S;
```

Bij de beschrijving van de procedures voor de regeldrukker is voorondersteld dat de routine PRCHARX voor iedere CHARF waarde de juiste actie uitvoert. Dit komt neer op het afdrukken van één enkel karakter door de regeldrukker of het bewerkstelligen van een overgang op een nieuwe regel of als er al 60 overgangen op een nieuwe regel zijn geweest een overgang op een nieuwe pagina. Daarnaast zijn een tweetal routines noodzakelijk om een string te splitsen in zijn samenstellende delen. Deze routines zijn procedure SETSTRING(S); string S; en integer procedure STRINGSYM;
SETSTRING(S) is een routine die er zorg voor draagt dat volgende aanroepen van STRINGSYM de successievelijke SYM waarden van de string S opleveren. De string waarop STRINGSYM werkt wordt bepaald door de laatst meegegeven actuele parameter aan SETSTRING.
Op te merken is dat op een paginaregel van de regeldrukker in het oorspronkelijk aangeboden systeem B6700 120 posities beschikbaar waren genummerd van 0 t/m 119. Dit houdt in dat wanneer de programmeur een bedrukking voor positie 120 specificeert, de regeldrukker-

procedures een bedrukking op positie 0 van de volgende regel geven, dit in tegenstelling met het THE-systeem waar dit pas plaats vindt bij een bedrukking voor positie 144. Men dient dus niet meer dan 120 posities per regel voor verwerkte informatie te gebruiken in het B6700 systeem.

2.1.1 procedure PRCHAR(N); value N; integer N;

De procedure houdt rekening met het regelnummer, aangegeven door LINENUMBERX, met de positie op de regel, aangegeven door LINEPOSITION, en met het aantal toegestane positiebedrukkingen per regel, in deze beschrijving gesteld op 120, aangegeven door MAXPOSITION.

```
procedure PRCHAR(N); value N; integer N;
if N=119 then
begin if LINENUMBERX>59 then LINENUMBERX:=1
                        else LINENUMBERX:=LINENUMBERX + 1;
   LINEPOSITION:=0;
   PRCHARX(119)
end else
begin if LINEPOSITION=MAXPOSITION then PRCHAR(119);
   PRCHARX(N);
   LINEPOSITION:=LINEPOSITION + 1
end PRCHAR(N);
```

2.1.2 procedure PRSYM(N); value N; integer N;

N heeft een waarde die gezien wordt als de waarde van een basissymbool met SYM waarde N. Zie bijlage III. Dit basissymbool bestaat uit één of meerdere karakters. Op de regeldrukker zal het basissymbool worden afgedrukt tenzij N een niet toegestane waarde heeft. N<0,N=36,N=63,N>131. In het THE-systeem wordt dan # afgedrukt.

```
procedure PRSYM(N); value N; integer N;
begin procedure TEST; if LINEPOSITION > MAXPOSITION-10 then PRCHAR(119);
   switch P:=L,L68,L69,L,L71,L,L73,L,L75,L,L77,L78,L,L,L81,L82,L83,
            L84,L85,L86,L,L,L,L,L,L92,L,L94,L95,L96,L97,L,L,L,L,L02,
            L03,L04,L05,L06,L07,L08,L09,L10,L11,L12,L13,L14,L15,L16,
            L17,L18,L,L,L,L,L23,L24,L25,L26,L27,L28,L29,L30,L31;
   go to P[if N<68 V N>131 then 1 else N-66];
   L: PRCHAR(N); go to L32;
   L68: PRCHAR(218); go to L32;
   L69: PRCHAR(336); go to L32;
   L71: PRCHAR(326); go to L32;
   L73: PRCHAR(200); go to L32;
   L75: PRCHAR(202); go to L32;
   L77: PRCHAR(198); go to L32;
   L78: PRCHAR(204); go to L32;
   L81: TEST;PRCHAR(144);PRCHAR(152);PRCHAR(157);PRCHAR(152);go to L32;
   L82: TEST;PRCHAR(143);PRCHAR(152);PRCHAR(155);go to L32;
   L83: TEST;PRCHAR(156);PRCHAR(157);PRCHAR(142);PRCHAR(153);go to L32;
   L84: TEST;PRCHAR(158);PRCHAR(151);PRCHAR(157);PRCHAR(146);
        PRCHAR(149);go to L32;
   L85: TEST;PRCHAR(160);PRCHAR(145);PRCHAR(146);PRCHAR(149);
        PRCHAR(142);go to L32;
   L86: TEST;PRCHAR(141);PRCHAR(152);go to L32;
   L92: TEST;PRCHAR(140);PRCHAR(152);PRCHAR(150);PRCHAR(153);
        PRCHAR(149);PRCHAR(142);PRCHAR(161);go to L32;
   L94: TEST;PRCHAR(146);PRCHAR(143);go to L32;
   L95: TEST;PRCHAR(157);PRCHAR(145);PRCHAR(142);PRCHAR(151);go to L32;
   L96: TEST;PRCHAR(142);PRCHAR(149);PRCHAR(156);PRCHAR(142);go to L32;
   L97: TEST;PRCHAR(140);PRCHAR(152);PRCHAR(150);PRCHAR(150);
        PRCHAR(142);PRCHAR(151);PRCHAR(157);go to L32;
   L02: PRCHAR(328); go to L32;
   L03: PRCHAR(330); go to L32;
   L04: TEST;PRCHAR(139);PRCHAR(142);PRCHAR(144);PRCHAR(146);
        PRCHAR(151);go to L32;
   L05: TEST;PRCHAR(142);PRCHAR(151);PRCHAR(141);go to L32;
   L06: TEST;PRCHAR(152);PRCHAR(160);PRCHAR(151);go to L32;
   L07: TEST;PRCHAR(155);PRCHAR(142);PRCHAR(138);PRCHAR(149);go to L32;
```

```
    L08: TEST;PRCHAR(146);PRCHAR(151);PRCHAR(157);PRCHAR(142);
         PRCHAR(144);PRCHAR(142);PRCHAR(155);go to L32;
    L09: TEST;PRCHAR(139);PRCHAR(152);PRCHAR(152);PRCHAR(149);
         PRCHAR(142);PRCHAR(138);PRCHAR(151);go to L32;
    L10: TEST;PRCHAR(156);PRCHAR(157);PRCHAR(155);PRCHAR(146);
         PRCHAR(151);PRCHAR(144);go to L32;
    L11: TEST;PRCHAR(138);PRCHAR(155);PRCHAR(155);PRCHAR(138);
         PRCHAR(162);go to L32;
    L12: TEST;PRCHAR(153);PRCHAR(155);PRCHAR(152);PRCHAR(140);
         PRCHAR(142);PRCHAR(141);PRCHAR(158);PRCHAR(155);
         PRCHAR(142);go to L32;
    L13: TEST;PRCHAR(156);PRCHAR(160);PRCHAR(146);PRCHAR(157);
         PRCHAR(140);PRCHAR(145);go to L32;
    L14: TEST;PRCHAR(149);PRCHAR(138);PRCHAR(139);PRCHAR(142);
         PRCHAR(149);go to L32;
    L15: TEST;PRCHAR(159);PRCHAR(138);PRCHAR(149);PRCHAR(158);
         PRCHAR(142);go to L32;
    L16: TEST;PRCHAR(157);PRCHAR(155);PRCHAR(158);PRCHAR(142);go to L32;
    L17: TEST:PRCHAR(143);PRCHAR(138);PRCHAR(149);PRCHAR(156);
         PRCHAR(142);go to L32;
    L18: TEST;PRCHAR(153);PRCHAR(155);PRCHAR(152);PRCHAR(144);
         PRCHAR(142);PRCHAR(151);PRCHAR(141);go to L32;
    L23: TEST;PRCHAR(150);PRCHAR(157);PRCHAR(138);PRCHAR(153);
         PRCHAR(142);go to L32;
    L24: TEST;PRCHAR(149);PRCHAR(146);PRCHAR(139);PRCHAR(155);
         PRCHAR(138);PRCHAR(155);PRCHAR(162);go to L32;
    L25: TEST;PRCHAR(146);PRCHAR(151);PRCHAR(159);PRCHAR(156);
         PRCHAR(149);PRCHAR(138);PRCHAR(156);PRCHAR(145);go to L32;
    L26: PRCHAR(221); go to L32;
    L27: PRCHAR(349); go to L32;
    L28: TEST;PRCHAR(151);PRCHAR(149);PRCHAR(140);PRCHAR(155);go to L32;
    L29: PRCHAR(215); go to L32;
    L30: TEST;PRCHAR(138);PRCHAR(149);PRCHAR(144);PRCHAR(152);
         PRCHAR(149);go to L32;
    L31: TEST;PRCHAR(140);PRCHAR(152);PRCHAR(155);PRCHAR(155);
         PRCHAR(142);PRCHAR(140);PRCHAR(157);PRCHAR(146);PRCHAR(152);
         PRCHAR(151);
    L32:
end PRSYM(N);
```

2.1.3.1 procedure NLCR;

De procedure geeft een overgang op een nieuwe regel en stelt de positie op de regel terug op 0. Als door het uitvoeren van de procedure het aantal regels op de pagina de 60 overschrijdt, wordt in plaats van een overgang op een nieuwe regel een overgang naar een nieuwe pagina bewerkstelligt.

```
procedure NLCR; PRCHAR(119);
```

2.1.3.2 procedure CARRIAGE(N); value N; integer N;

Voor $0<N$ worden N regelopvoeren gegeven en wordt de positie op de regel terug op 0 gesteld. Voor $N\leq 0$ wordt de opdracht CARRIAGE(N) overgeslagen.

```
procedure CARRIAGE(N); value N; integer N;
if N>0 then for N:=N step -1 until 1 do PRCHAR(119);
```

2.1.3.3 procedure NEWPAGE;

De procedure bewerkstelligt de overgang naar een nieuwe pagina en stelt de positie op de regel terug op 0.

```
procedure NEWPAGE;
if LINENUMBERX≠1 ∨ LINEPOSITION≠0 then CARRIAGE(61-LINENUMBERX);
```

2.1.4 integer procedure LINENUMBER;

De waarde van de procedure is het nummer van de regel "in opbouw" op de heersende pagina. Deze waarde is tenminste 1 en ten hoogste 60, na een overgang op een nieuwe pagina levert LINENUMBER de waarde 1 af.

```
integer procedure LINENUMBER; LINENUMBER:=LINENUMBERX;
```

2.1.5.1 procedure SPACE(N); value N; integer N;

De procedure verhoogt de positie op de regel met N. Overschrijdt de positie op de regel de waarde van MAXPOSITION dan wordt er een regelopvoer ingelast. SPACE(-N) voor $N\geq 0$ is gelijkwaardig met SPACE(N).

```
procedure SPACE(N); value N; integer N;
for N:=abs(N) step -1 until 1 do PRCHAR(93);
```

2.1.5.2 procedure TAB;

De procedure TAB verhoogt de positie op de regel zodanig dat een 8-voud bereikt wordt. Overschrijdt de positie op de regel de waarde van MAXPOSITION dan wordt er een regelopvoer ingelast.

```
procedure TAB;
SPACE(8+((LINEPOSITION+1):8)×8-LINEPOSITION);
```

2.1.6 procedure FLOT(N,M,X); value N,M,X; integer N,M; real X;

De procedure FLOT voert de actuele waarde van de formele parameter X uit via de regeldrukker. Het getal wordt uitgevoerd in de vorm

teken punt ××× ××× laag tientje teken ××× spatie
aantal N aantal M
1≤N≤13 1≤M≤3

waarbij een kruisje staat voor een willekeurig cijfer en een sterretje voor een willekeurig cijfer of een spatie.
In het ontwikkelde algoritme wordt het getal 0 apart beschouwd. De absolute waarde van het uit te voeren getal wordt gebracht tussen 0.1 en 1 (maal een macht van 10). Daartoe wordt de grootte van het getal door middel van de 10-logarithme geschat. De constante die in de procedure voorkomt is de $^{e}\log 10$, in 13 cijfers nauwkeurig.
Juist vanwege deze logarithme is de 0 apart behandeld. Als de absolute waarde van het getal genormaliseerd is, wordt nagegaan of dit genormaliseerde getal inderdaad tussen 0.1 en 1 ligt. Door afrondingsfouten zou er net een keer te veel of te weinig met 10 vermenigvuldigd respectievelijk door 10 gedeeld kunnen zijn.
Als er een getal groter dan 10^{599} wordt aangeboden wordt +.1000000000000 +600 afgedrukt. Dit getal is gebaseerd op de arithmetiek van de X-8 (THE). Voor grotere getallen wordt tengevolge van afrondingsfouten de procedure erg onnauwkeurig. Opgemerkt dient te worden dat af te drukken getallen niet afgerond maar afgekapt worden.

```
procedure FLOT(N,M,X);value N,M,X;integer N,M; real X;
begin if N<1 ∨ N>13 ∨ M<1 ∨ M>3 then begin N:=13;M:=3 end;
   if LINEPOSITION+N+M+5>MAXPOSITION then NLCR;
   if abs(X)>((10↑10)-1) ∧ M=1 then begin N:=13;M:=3 end;
   if abs(X)>((10↑100)-1) ∧ M=2 then begin N:=13;M:=3 end;
   PRSYM(if X≥0 then 64 else 65);
   PRSYM(88);
   if X=0 then
   begin integer COUNTER;
      for COUNTER:=1 step 1 until N do PRSYM(0);
      PRSYM(89);
      PRSYM(64);
      for COUNTER:=2 step 1 until M do PRSYM(93);
      PRSYM(0);
      PRSYM(93);
   end else
   begin X:=abs(X);
      if X>10↑599 then
      begin integer COUNTER;
         PRSYM(1);
         for COUNTER:=2 step 1 until N do PRSYM(0);
         for COUNTER:=89,64,6,0,0,93 do PRSYM(COUNTER);
      end else
      begin integer COUNTER,DIGIT,EXPONENT,NUMBERM;
         real Y;
         Boolean NUMBER;
         Y:=ln(X)/2.302585092994;
         EXPONENT:=entier(Y)+1;
         Y:=X×10↑(-EXPONENT);
         if Y<0.1 then begin Y:=10×Y;EXPONENT:=EXPONENT-1 end else
         if Y≥1 then begin Y:=Y/10;EXPONENT:=EXPONENT+1 end;
         for COUNTER:=1 step 1 until N do
         begin Y:=Y×10;
            X:=entier(Y);
            PRSYM(X);
            Y:=Y-X
         end;
```

```
            PRSYM(89);
            PRSYM(if EXPONENT>0 then 64 else 65);
            NUMBER:=false;
            NUMBERM:=0;
            for COUNTER:=M step -1 until 0 do
            begin DIGIT:=abs(EXPONENT:(10↑COUNTER);
               if DIGIT≠0 then begin PRSYM(DIGIT);NUMBER:=true end else
               if NUMBERM=M then PRSYM(DIGIT) else
               if NUMBERM≠0 ∧ ¬NUMBER then PRSYM(93) else
               if NUMBERM≠0 then PRSYM(DIGIT);
               NUMBERM:=NUMBERM+1;
               EXPONENT:=EXPONENT-(EXPONENT:(10↑COUNTER))×(10↑COUNTER);
            end;
            PRSYM(93);
         end
      end
   end FLOT(N,M,X);
```

2.1.7.1 procedure FIXT(N,M,X); value N,M,X; integer N,M; real X;

De procedure FIXT voert de actuele waarde van de formele parameter X uit via de regeldrukker. Voorwaarden zijn:

abs(X)<10↑N; N≥0; M≥0; 1≤N+M≤21

Is aan een van deze voorwaarden niet voldaan dan wordt FLOT(13,3,X) uitgevoerd. Als M>0 wordt het getal uitgevoerd in de vorm:

teken ×××××××× punt ××××××× spatie
aantal N aantal M

Een kruisje staat voor een willekeurig cijfer en een sterretje voor een cijfer of spatie indien het cijfer een beginnul is. Het teken wordt dan ook opgeschoven. Als M=0 wordt het getal uitgevoerd in de vorm:

teken ×××××××× spatie
aantal N

Overschrijdt de positie op de regel de waarde van MAXPOSITION dan wordt een regelopvoer ingelast. In tegenstelling tot FLOT wordt bij FIXT wel afgerond en niet afgekapt.

De procedure FIXT maakt gebruik van de procedure SEMIFIXT. De boolean ABSF in de procedure SEMIFIXT wordt gebruikt om aan te geven of het teken vervangen moet worden door een spatie.

```
procedure FIXT(N,M,X);value N,M,X;integer N,M;real X;SEMIFIXT(N,M,X,false);

procedure SEMIFIXT(N,M,X,ABSF);value N,M,X;integer N,M;real X;Boolean ABSF;
if abs(X)+.5/10↑M>10↑N ∨ N<0 ∨ M<0 ∨ N+M<1 ∨ N+M>21 then FLOT(13,3,X) else
begin integer SIGNX,SYMSIGNX,COUNTER,DIGIT;
   real INTEGERPART,DECIMALPART,TENEXPONENT;
   Boolean SPAC;
   if LINEPOSITION+N+2+(if M=0 then 0 else M+1)>MAXPOSITION then NLCR;
   SIGNX:=sign(X);
   SYMSIGNX:=if ABSF then 93 else if SIGNX=-1 then 65 else 64;
   X:=SIGNX×X+.5/10↑M;
   INTEGERPART:=entier(X);
   if M≠0 then DECIMALPART:=(X-INTEGERPART)×10↑M;
   if N≠0 then
   begin if INTEGERPART=0 then
      begin SPACE(N-1);PRSYM(SIGNX);PRSYM(0) end else
      begin SPAC:=true;
         TENEXPONENT:=10↑N;
         for COUNTER:=1 step 1 until N do
         begin TENEXPONENT:=TENEXPONENT/10;
            DIGIT:=entier(INTEGERPART/TENEXPONENT);
            INTEGERPART:=INTEGERPART-DIGIT×TENEXPONENT;
            if DIGIT=0 ∧ SPAC then PRSYM(93) else
            begin if SPAC then
               begin SPAC:=false;PRSYM(SYMSIGNX) end;
               PRSYM(DIGIT)
            end
         end;
      end
   end else
   PRSYM(SIGNX);
   if M≠0 then
   begin PRSYM(88);
```

```
        for COUNTER:=1 step 1 until M do
        begin TENEXPONENT:=10↑(M-COUNTER);
           DIGIT:=entier(DECIMALPART/TENEXPONENT);
           DECIMALPART:=DECIMALPART-DIGIT×TENEXPONENT;
           PRSYM(DIGIT)
        end
     end;
     PRSYM(93)
  end SEMIFIXT(N,M,X,ABSF);
```

2.1.7.2 procedure ABSFIXT(N,M,X); value N,M,X; integer N,M; real X;

De procedure ABSFIXT voert de actuele waarde van de formele parameter X uit via de regeldrukker. Voorwaarden zijn:

abs(X)<10↑N; N≥0; M≥0; 1≤N+M≤21

Is aan één van deze voorwaarden niet voldaan dan wordt FLOT(13,3,X) uitgevoerd. Als M>0 wordt het getal uitgevoerd in de vorm:

spatie ×××××××× punt ××××××× spatie
aantal N aantal M

Een kruisje staat voor een willekeurig cijfer en een sterretje voor een cijfer of spatie indien het cijfer een beginnul is. Als M=0 wordt het getal uitgevoerd in de vorm:

spatie ××××××× spatie
aantal N

Overschrijdt de positie op de regel de waarde van MAXPOSITION dan wordt een regelopvoer ingelast.

```
procedure ABSFIXT(N,M,X);value N,M,X;integer N,M;real X;SEMIFIXT(N,M,X,true);
```

2.1.8.1 procedure PRINT(X); value X; real X;

Als de absolute waarde van X gelijk is aan een geheel getal kleiner 2^{40}, wordt X afgedrukt volgens FIXT(13,0,X) gevolgd door zes extra spaties. Zo niet, dan volgens FLOT(13,3,X). In beide gevallen wordt de positie op de regel verhoogd met 21, maar zonodig wordt door het systeem vooraf een overgang op een nieuwe regel ingelast.

```
procedure PRINT(X);value X;real X;
if X=entier(X) ∧ abs(X) < 1099511627776 then
begin if LINEPOSITION+21>MAXPOSITION then NLCR;
      FIXT(13,0,X);
      SPACE(6)
end else FLOT(13,3,X);
```

2.1.8.2 procedure print(X); value X; real X;

Deze procedure is gelijkwaardig met de procedure PRINT(X).

```
procedure print(X);value X;real X;PRINT(X);
```

2.1.9 procedure PRINTTEXT(S); string S;

De procedure PRINTTEXT voert een string uit via de regeldrukker. De symbolen van de string, ontdaan van de buitenste stringquotes, worden, symbool na symbool afgedrukt. Na het laatste symbool van S leveren volgende aanroepen van STRINGSYM het symbool "einde string" af, met waarde 510. Aanroepen van STRINGSYM slaan steeds op de string die bij de laatst uitgevoerde aanroep van SETSTRING is meegegeven.

```
procedure PRINTTEXT(S);string S;
begin integer SYMS;
      SETSTRING(S);
      for SYMS:=STRINGSYM while SYMS≠510 do PRSYM(SYMS)
end PRINTTEXT(S);
```

2.1.10 Het omvattende blok van de systeemuitvoerprocedures voor de regeldrukker.

```
begin integer LINEPOSITION,LINENUMBERX,MAXPOSITION;
   < systeemuitvoerprocedures voor de regeldrukker >
   LINEPOSITION:=0;LINENUMBERX:=1;MAXPOSITION:=120
end
```

HOOFDSTUK 3

Systeemuitvoerprocedures voor de bandponsers bij de X-8 (THE)

3 Voorkomende bandponsprocedures, die later nog besproken zullen worden, zijn:

```
procedure PUNOCT(I,N); value I,N; integer I,N;
procedure PUNOCT1(N); value N; integer N;
procedure PUNOCT2(N); value N; integer N;
procedure PUHEP(N); value N; integer N;
Boolean procedure PUNCHARF(I,N); value I,N; integer I,N;
Boolean procedure PUNCHARF1(N); value N; integer N;
Boolean procedure PUNCHARF2(N); value N; integer N;
Boolean procedure PUNSYM(I,N); value I,N; integer I,N;
Boolean procedure PUNSYM1(N); value N; integer N;
Boolean procedure PUNSYM2(N); value N; integer N;
Boolean procedure PUSYM(N); value N; integer N;
procedure PUNNLCR(I); value I; integer I;
procedure PUNNLCR1;
procedure PUNNLCR2;
procedure PUNLCR;
procedure PUNSPACE(I,N); value I,N; integer I,N;
procedure PUNSPACE1(N); value N; integer N;
procedure PUNSPACE2(N); value N; integer N;
procedure PUSPACE(N); value N; integer N;
procedure PUNRUNOUT(I,N); value I,N; integer I,N;
procedure PUNRUNOUT1(N); value N; integer N;
procedure PUNRUNOUT2(N); value N; integer N;
procedure RUNOUT;
procedure PUNTAB(I); value I; integer I;
procedure PUNTAB1;
procedure PUNTAB2;
procedure ENDPUNDOC(I); value I; integer I;
procedure ENDPUNDOC1;
procedure ENDPUNDOC2;
procedure PUNFLO(I,N,M,X); value I,N,M; integer I,N,M; real X;
```

```
procedure PUNFLO1(N,M,X); value N,M,X; integer N,M; real X;
procedure PUNFLO2(N,M,X); value N,M,X; integer N,M; real X;
procedure FLOP(N,M,X); value N,M,X; integer N,M; real X;
procedure PUNFIX(I,N,M,X); value I,N,M,X; integer I,N,M; real X;
procedure PUNFIX1(N,M,X); value N,M,X; integer N,M; real X;
procedure PUNFIX2(N,M,X); value N,M,X; integer N,M; real X;
procedure FIXP(N,M,X); value N,M,X; integer N,M; real X;
procedure PUNABSFIX(I,N,M,X); value I,N,M,X; integer I,N,M; real X;
procedure PUNABSFIX1(N,M,X); value N,M,X; integer N,M; real X;
procedure PUNABSFIX2(N,M,X); value N,M,X; integer N,M; real X;
procedure ABSFIXP(N,M,X); value N,M,X; integer N,M; real X;
procedure PUNUM(I,X); value I,X; integer I; real X;
procedure PUNUM1(X); value X; real X;
procedure PUNUM2(X); value X; real X;
procedure PUNCH(X); value X; real X;
procedure PUNTEXT(I,S); value I; integer I; string S;
procedure PUNTEXT1(S); string S;
procedure PUNTEXT2(S); string S;
procedure PUTEXT(S); string S;
```

3.1.1 Bij de beschrijving van de procedures voor de bandponsers is voorondersteld dat de routine PUNOCTX in staat is om één enkele bandponsing te ponsen en de ponsband één vertanding op te schuiven. De procedure PUNOCT maakt van de routine PUNOCTX gebruik. De procedure PUNOCT ponst voor $0 \leq N \wedge N \leq 255$ de waarde van de meegegeven parameter als octade. Voor N 256: de rest bij deling van N door 256. Voor N 0: 256 verminderd met de rest bij deling van abs(N) door 256. Een parameter I met de waarde 1 of 2 bepaalt over welke bandponser geponst wordt. Bij overgang van PUNSYM of PUNCHARF op PUNOCT (over dezelfde stroom) wordt een stopcode ingelast. PUNSYM en PUNCHARF worden later besproken. Het inlassen van de stopcode vindt plaats indien de globale boolean PUNOCTPASTi false is. PUNOCTPASTi wordt door PUNOCT true en door PUNCHARF false. Een beschrijving van PUNOCT en daarmee verwante procedures luidt:

```
procedure PUNOCT(I,N);value I,N;integer I,N;
begin if I=1 ∧ ¬PUNOCTPAST1 then
   begin PUNOCTX(1,11);PUNOCTPAST1:=true end else
   if I=2 ∧ ¬PUNOCTPAST2 then
   begin PUNOCTX(2,11);PUNOCTPAST2:=true end;
   if I=1 ∨ I=2 then
   PUNOCTX(I,if N≥0 then (if N≤255 then N else N-N:256x256)
                    else 256+N-N:256x256)
               else
   begin PRINTTEXT( DYNAMIC ERROR IN STREAMNUMBER );
      go to L END
   end
end PUNOCT(I,N);
```

3.1.2 `procedure PUNOCT1(N);value N;integer N;PUNOCT(1,N);`

3.1.3 `procedure PUNOCT2(N);value N;integer N;PUNOCT(2,N);`

3.1.4 `procedure PUHEP(N);value N;integer N;PUNOCT(1,N);`

3.2.1 `Boolean procedure PUNCHARF(I,N); value I,N; integer I,N;`

Met PUNCHARF kan men een zichtbare positiebedrukking specificeren door N te kiezen uit het waardenbereik van CHARF. Zie bijlage III. De waarde van de procedure is false als N behoort tot het waardenbereik van CHARF. Valt N buiten dat bereik, dan wordt niets geponst maar wordt de waarde van de procedure true. Spaties aan het einde van een regel worden niet geponst. Bij een overgang van PUNOCT op PUNCHARF of PUNSYM wordt een lowercase tussengevoegd. Wanneer de globale variabele PUNOCTPASTi= true dan wordt PUNOCTPASTi false door PUNCHARF en wordt de lowercase tussengevoegd.

```
Boolean procedure PUNCHARF(I,N);value I,N);integer I,N;
begin integer procedure OCTA;
   OCTA:=if X=1 ∨ X=2 ∨ X=4 ∨ X=7 ∨ X=8 then X else
   if X=29 ∨ X=37 ∨ X=38 ∨ X=35 then X+6 else
   if X=3 ∨ X=5 ∨ X=6 ∨ X=9 then X+16 else
   if X=48 ∨ X=50 ∨ X=51 ∨ X=54 then X+19 else
   if X=28 ∨ X=30 ∨ X=33 ∨ X=34 then X+22 else
   if X=46 ∨ X=47 ∨ X=49 ∨ X=52 ∨ X=53 then X+35 else
   if X=21 ∨ X=23 ∨ X=24 ∨ X=27 then X+46 else
   if X=37 ∨ X=38 ∨ X=40 ∨ X=43 then X+60 else
   if X=19 ∨ X=20 ∨ X=22 ∨ X=25 ∨ X=26 then X+62 else
   if X=39 ∨ X=41 ∨ X=42 then X+76 else
   if X=10 ∨ X=11 ∨ X=13 ∨ X=16 ∨ X=17 then X+87 else
   if X=12 ∨ X=14 ∨ X=15 ∨ X=18 then X+103 else
   if X=55 ∨ X=57 ∨ X=60 ∨ X=61 then X-5 else
   if X=56 ∨ X=58 ∨ X=59 ∨ X=62 then X-21 else
   if X=67 ∨ X=80 then X-48 else
   if X=79 ∨ X=100 then X-78 else
   if X=87 then X+4 else if X=88 then X+19 else
   if X=0 then X+32 else if X=64 then X+48 else
   if X=45 then X+66 else if X=90 then X+17 else
   if X=65 then X-1 else if X=121 then X-9 else
   if X=76 then X-12 else if X=72 then X-13 else
   if X=74 then X-25 else if X=89 then X-30 else
   if X=122 then X-31 else if X=120 then X-61 else
   if X=66 then X-64 else if X=70 then X-66 else
   if X=91 then X-70 else if X=99 then X-74 else
   if X=98 then X-90 else if X=101 then X-94;

   if I=1 ∨ I=2 then
   begin integer X,Z;
      if N≥384 then X:=N-384 else
      if N≥256 then X:=N-256 else
      if N≥128 then X:=N-128 else X:=N;
      PUNCHARF:=¬((0≤X ∧ X≤35) ∨ (37≤X ∧ X≤62) ∨ (64≤X ∧ X≤67) ∨ X=70
      ∨ X=74 ∨ X=76 ∨ X=79 ∨ X=80 ∨ (87≤X ∧ X≤91) ∨ X=93 ∨
      (98≤X ∧ X≤101) ∨ (119≤X ∧ X≤122);
```

```
if ¬PUNCHARF then
begin if I=1 then
   begin if PUNOCTPAST1 then
      begin PUNOCT(1,122);PUNOCTPAST1:=false;LOWER1:=true end;
      if X=93 then SPACER1:=SPACER1+1 else
      begin if X=119 then
         begin SPACER1:=0;PUNOCT(1,26);PUNCHPOSITION1:=0 end else
         begin if SPACER1≠0 then
            begin for Z:=SPACER1 step -1 until 1 do
               begin if PUNCHPOSITION1=MAXPOSITION then
                  begin PUNOCT(1,26);PUNCHPOSITION1:=0;PUNOCT(1,16) end
                                                      else
                  PUNCHPOSITION1:=PUNCHPOSITION1+1;
                  PUNOCT(1,16)
               end;
               SPACER1:=0
            end;
            if PUNCHPOSITION1>MAXPOSITION then
            begin PUNOCT(1,26);PUNCHPOSITION1:=0 end;
            if (0≤X ∧ X≤35) ∨ X=64 ∨ X=65 ∨ X=72 ∨ X=87 ∨ X=88 ∨ X=89 then
            begin if ¬LOWER1 then
               begin LOWER1:=true;
                  if N<128 then
                  begin PUNOCT(1,122);PUNOCT(1,OCTA) end else
                  if N<256 then
                  begin PUNOCT(1,122);PUNOCT(1,14);PUNOCT(1,OCTA) end else
                  if N<384 then
                  begin PUNOCT(1,14);PUNOCT(1,122);PUNOCT(1,OCTA) end else
                  begin PUNOCT(1,14);PUNOCT(1,122);PUNOCT(1,14);
                     PUNOCT(1,OCTA)
                  end
               end else
               if N<128 then PUNOCT(1,OCTA) else
               if N<256 then
               begin PUNOCT(1,14);PUNOCT(1,OCTA) end else
               if N<384 then
```

```
            begin PUNOCT(1,124);PUNOCT(1,14);PUNOCT(1,122);
               PUNOCT(1,OCTA)
            end else
            begin PUNOCT(1,124);PUNOCT(1,14);PUNOCT(1,122);
               PUNOCT(1,14);PUNOCT(1,OCTA)
            end
         end else
         if ¬LOWER1 then
         begin if N<128 then PUNOCT(1,OCTA) else
            if N<256 then
            begin PUNOCT(1,122);PUNOCT(1,14);PUNOCT(1,124);
               PUNOCT(1,OCTA)
            end else
            if N<384 then
            begin PUNOCT(1,14);PUNOCT(1,OCTA) end else
            begin PUNOCT(1,122);PUNOCT(1,14);PUNOCT(1,124);
               PUNOCT(1,14);PUNOCT(1,OCTA)
            end
         end else
         begin LOWER1:=false;
            if N<128 then
            begin PUNOCT(1,124);PUNOCT(1,OCTA) end else
            if N<256 then
            begin PUNOCT(1,14);PUNOCT(1,124);PUNOCT(1,OCTA) end else
            if N<384 then
            begin PUNOCT(1,124);PUNOCT(1,14);PUNOCT(1,OCTA) end else
            begin PUNOCT(1,14);PUNOCT(1,124);PUNOCT(1,14);
               PUNOCT(1,OCTA)
            end
         end;
         PUNCHPOSITION1:=PUNCHPOSITION1+1
      end
   end
end else
begin if PUNOCTPAST2 then
   begin PUNOCT(2,122);PUNOCTPAST2:=false;LOWER2:=true end;
```

```
if X=93 then SPACER2:=SPACER2+1 else
begin if X=119 then
   begin SPACER2:=0;PUNOCT(2,26);PUNCHPOSITION2:=0 end else
   begin if SPACER2≠0 then
      begin for Z:=SPACER2 step -1 until 1 do
         begin if PUNCHPOSITION2=MAXPOSITION then
            begin PUNOCT(1,26);PUNCHPOSITION2:=0;PUNOCT(2,16) end
                                                  else
            PUNCHPOSITION2:=PUNCHPOSITION2+1;
            PUNOCT(2,16)
         end;
         SPACER2:=0
      end;
      if PUNCHPOSITION2>MAXPOSITION then
      begin PUNOCT(2,26);PUNCHPOSITION2:=0 end;
      if (0<X ∧ X≤35) ∨ X=64 ∨ X=65 ∨ X=72 ∨ X=87 ∨ X=88 ∨ X=89 then
      begin if ¬LOWER2 then
         begin LOWER2:=true;
            if N<128 then
            begin PUNOCT(2,122);PUNOCT(2,OCTA) end else
            if N<256 then
            begin PUNOCT(2,122);PUNOCT(2,14);PUNOCT(2,OCTA) end else
            if N<384 then
            begin PUNOCT(2,14);PUNOCT(2,122);PUNOCT(2,OCTA) end else
            begin PUNOCT(2,14);PUNOCT(2,122);PUNOCT(2,14);
               PUNOCT(2,OCTA)
            end
         end else
         if N<128 then PUNOCT(2,OCTA) else
         if N<256 then
         begin PUNOCT(2,14);PUNOCT(2,OCTA) end else
         if N<384 then
         begin PUNOCT(2,124);PUNOCT(2,14);PUNOCT(2,122);
            PUNOCT(2,OCTA)
         end else
         begin PUNOCT(2,124);PUNOCT(2,14);PUNOCT(2,122);
            PUNOCT(2,14);PUNOCT(2,OCTA)
```

```
                end
              end else
              if NOT LOWER2 then
              begin if N<128 then PUNOCT(2,OCTA) else
                 if N<256 then
                 begin PUNOCT(2,122);PUNOCT(2,14);PUNOCT(2,124);
                    PUNOCT(2,OCTA)
                 end else
                 if N<384 then
                 begin PUNOCT(2,14);PUNOCT(2,OCTA) end else
                 begin PUNOCT(2,122);PUNOCT(2,14);PUNOCT(2,124);
                    PUNOCT(2,14);PUNOCT(2,OCTA)
                 end
               end else
               begin LOWER2:=false;
                 if N<128 then
                 begin PUNOCT(2,124);PUNOCT(2,OCTA) end else
                 if N<256 then
                 begin PUNOCT(2,14);PUNOCT(2,124);PUNOCT(2,OCTA) end else
                 if N<384 then
                 begin PUNOCT(2,124);PUNOCT(2,14);PUNOCT(2,OCTA) end else
                 begin PUNOCT(2,14);PUNOCT(2,124);PUNOCT(2,14);
                    PUNOCT(2,OCTA)
                 end
               end;
               PUNCHPOSITION2:=PUNCHPOSITION2+1
            end
         end
      end
   end
end else
begin PRINTTEXT(‹DYNAMIC ERROR IN STREAMNUMBER›);
   go to L_END
end
end PUNCHARF(I,N);
```

3.2.2 Boolean procedure PUNCHARF1(N);value N;integer N;PUNCHARF(1,N);

3.2.3 Boolean procedure PUNCHARF2(N);value N;integer N;PUNCHARF(2,N);

3.3.1 Boolean procedure PUNSYM(I,N); value I,N; integer I,N;

Met PUNSYM kan men een symbool specificeren door N te kiezen uit het waardenbereik van SYM.
Dit waardenbereik is $[N|(0\leq N \wedge N\leq 35) \vee (37\leq N \wedge N\leq 62) \vee (64\leq N \wedge N\leq 131)]$.
De waarde van de procedure is false als N behoort tot het waardenbereik van SYM. Valt N buiten dat bereik, dan wordt er niets geponst maar wordt de waarde van de procedure true. Wanneer PUNSYM begint aan het ponsen van een onderstreept basissymbool en het aantal nog op de regel beschikbare posities is kleiner dan 10, dan wordt voorafgaande aan het basissymbool een overgang op een nieuwe regel ingelast.

```
Boolean procedure PUNSYM(I,N);value I,N;integer I,N;
if I=1 ∨ I=2 then
begin PUNSYM:=¬(0≤N ∧ N≤35) ∨ (37≤N ∧ N≤62) ∨ (64≤N ∧ N≤131);
   if ¬PUNSYM then
   begin procedure POSITION;
      if I=1 then begin if PUNCHPOSITION1>MAXPOSITION-10 then PUNCHARF(1,119) end
             else
      if PUNCHPOSITION2>MAXPOSITION-10 then PUNCHARF(2,119);
      switch S:=W,W68,W69,W,W71,W,W73,W,W75,W,W77,W78,W,W,W81,W82,W83,W84,
                W85,W86,W,W,W,W,W,W92,W,W94,W95,W96,W97,W,W,W,W,W02,W03,
                W04,W05,W06,W07,W08,W09,W10,W11,W12,W13,W14,W15,W16,W17,
                W18,W,W,W,W,W23,W24,W25,W26,W27,W28,W29,W30,W31;
      go to S[if N<68 then 1 else N-66];
      W: PUNCHARF(I,N);go to W32;
      W68: PUNCHARF(I,218);go to W32;
      W69: PUNCHARF(I,336);go to W32;
      W71: PUNCHARF(I,326);go to W32;
      W73: PUNCHARF(I,200);go to W32;
      W75: PUNCHARF(I,202);go to W32;
      W77: PUNCHARF(I,198);go to W32;
      W78: PUNCHARF(I,204);go to W32;
```

```
W81: POSITION;PUNCHARF(I,144);PUNCHARF(I,152);PUNCHARF(I,157);
     PUNCHARF(I,152);go to W32;
W82: POSITION;PUNCHARF(I,143);PUNCHARF(I,152);PUNCHARF(I,155);
     go to W32;
W83: POSITION;PUNCHARF(I,156);PUNCHARF(I,157);PUNCHARF(I,142);
     PUNCHARF(I,153);go to W32;
W84: POSITION;PUNCHARF(I,158);PUNCHARF(I,151);PUNCHARF(I,157);
     PUNCHARF(I,146);PUNCHARF(I,149);go to W32;
W85: POSITION;PUNCHARF(I,160);PUNCHARF(I,145);PUNCHARF(I,146);
     PUNCHARF(I,149);PUNCHARF(I,142);go to W32;
W86: POSITION;PUNCHARF(I,141);PUNCHARF(I,152);go to W32;
W92: POSITION;PUNCHARF(I,140);PUNCHARF(I,152);PUNCHARF(I,150);
     PUNCHARF(I,153);PUNCHARF(I,149);PUNCHARF(I,142);
     PUNCHARF(I,161);go to W32;
W94: POSITION;PUNCHARF(I,146);PUNCHARF(I,143);go to W32;
W95: POSITION;PUNCHARF(I,157);PUNCHARF(I,145);PUNCHARF(I,142);
     PUNCHARF(I,151);go to W32;
W96: POSITION;PUNCHARF(I,142);PUNCHARF(I,149);PUNCHARF(I,156);
     PUNCHARF(I,142);go to W32;
W97: POSITION;PUNCHARF(I,140);PUNCHARF(I,152);PUNCHARF(I,150);
     PUNCHARF(I,150);PUNCHARF(I,142);PUNCHARF(I,151);
     PUNCHARF(I,157);go to W32;
W02: PUNCHARF(I,328);go to W32;
W03: PUNCHARF(I,330);go to W32;
W04: POSITION;PUNCHARF(I,139);PUNCHARF(I,142);PUNCHARF(I,144);
     PUNCHARF(I,146);PUNCHARF(I,151);go to W32;
W05: POSITION;PUNCHARF(I,142);PUNCHARF(I,151);PUNCHARF(I,141);
     go to W32;
W06: POSITION;PUNCHARF(I,152);PUNCHARF(I,160);PUNCHARF(I,151);
     go to W32;
W07: POSITION;PUNCHARF(I,155);PUNCHARF(I,142);PUNCHARF(I,138);
     PUNCHARF(I,149);go to W32;
W08: POSITION;PUNCHARF(I,146);PUNCHARF(I,151);PUNCHARF(I,157);
     PUNCHARF(I,142);PUNCHARF(I,144);PUNCHARF(I,142);
     PUNCHARF(I,155);go to W32;
W09: POSITION;PUNCHARF(I,139);PUNCHARF(I,152);PUNCHARF(I,152);
     PUNCHARF(I,149);PUNCHARF(I,142);PUNCHARF(I,138);
     PUNCHARF(I,151);go to W32;
```

```
W10: POSITION;PUNCHARF(I,156);PUNCHARF(I,157);PUNCHARF(I,155);
     PUNCHARF(I,146);PUNCHARF(I,151);PUNCHARF(I,144);
     go to W32;
W11: POSITION;PUNCHARF(I,138);PUNCHARF(I,155);PUNCHARF(I,155);
     PUNCHARF(I,138);PUNCHARF(I,162);go to W32;
W12: POSITION;PUNCHARF(I,153);PUNCHARF(I,155);PUNCHARF(I,152);
     PUNCHARF(I,140);PUNCHARF(I,142);PUNCHARF(I,141);
     PUNCHARF(I,158);PUNCHARF(I,155);PUNCHARF(I,142);
     go to W32;
W13: POSITION;PUNCHARF(I,156);PUNCHARF(I,160);PUNCHARF(I,146);
     PUNCHARF(I,157);PUNCHARF(I,140);PUNCHARF(I,145);
     go to W32;
W14: POSITION;PUNCHARF(I,149);PUNCHARF(I,138);PUNCHARF(I,139);
     PUNCHARF(I,142);PUNCHARF(I,149);go to W32;
W15: POSITION;PUNCHARF(I,159);PUNCHARF(I,138);PUNCHARF(I,149);
     PUNCHARF(I,158);PUNCHARF(I,142);go to W32;
W16: POSITION;PUNCHARF(I,157);PUNCHARF(I,155);PUNCHARF(I,158);
     PUNCHARF(I,142);go to W32;
W17: POSITION;PUNCHARF(I,143);PUNCHARF(I,138);PUNCHARF(I,149);
     PUNCHARF(I,156);PUNCHARF(I,142);go to W32;
W18: POSITION;PUNCHARF(I,153);PUNCHARF(I,155);PUNCHARF(I,152);
     PUNCHARF(I,144);PUNCHARF(I,142);PUNCHARF(I,151);
     PUNCHARF(I,141);go to W32;
W23: POSITION;PUNCHARF(I,150);PUNCHARF(I,157);PUNCHARF(I,138);
     PUNCHARF(I,153);PUNCHARF(I,142);go to W32;
W24: POSITION;PUNCHARF(I,149);PUNCHARF(I,146);PUNCHARF(I,139);
     PUNCHARF(I,155);PUNCHARF(I,138);PUNCHARF(I,155);
     PUNCHARF(I,162);go to W32;
W25: POSITION;PUNCHARF(I,146);PUNCHARF(I,151);PUNCHARF(I,159);
     PUNCHARF(I,156);PUNCHARF(I,149);PUNCHARF(I,138);
     PUNCHARF(I,156);PUNCHARF(I,145);go to W32;
W26: PUNCHARF(I,221);go to W32;
W27: PUNCHARF(I,349);go to W32;
W28: POSITION;PUNCHARF(I,151);PUNCHARF(I,149);PUNCHARF(I,140);
     PUNCHARF(I,155);go to W32;
W29: PUNCHARF(I,215);go to W32;
```

```
      W30: POSITION;PUNCHARF(I,138);PUNCHARF(I,149);PUNCHARF(I,144);
           PUNCHARF(I,152);PUNCHARF(I,149);go to W32;
      W31: POSITION;PUNCHARF(I,140);PUNCHARF(I,152);PUNCHARF(I,155);
           PUNCHARF(I,155);PUNCHARF(I,142);PUNCHARF(I,140);
           PUNCHARF(I,157);PUNCHARF(I,146);PUNCHARF(I,152);
           PUNCHARF(I,151);
      W32:
   end
end else
begin PRINTTEXT(‹DYNAMIC ERROR IN STREAMNUMBER›);
   go to L END
end PUNSYM(I,N);
```

3.3.2 Boolean procedure PUNSYM1(N);value N;integer N;PUNSYM(1,N);

3.3.3 Boolean procedure PUNSYM2(N);value N;integer N;PUNSYM(2,N);

3.3.4 Boolean procedure PUSYM(N);value N;integer N;PUNSYM(1,N);

3.4.1 procedure PUNNLCR(I); value I; integer I;

De procedure ponst de flexowriterponsing overgang op een nieuwe regel over ponserstrocm i.

procedure PUNNLCR(I);value I;integer I;PUNCHARF(I,119);

3.4.2 procedure PUNNLCR1;PUNCHARF(1,119);

3.4.3 procedure PUNNLCR2;PUNCHARF(2,119);

3.4.4 procedure PUNLCR;PUNCHARF(1,119);

3.5.1 procedure PUNSPACE(I,N); value I,N; integer I,N;

De procedure ponst N (flexowriter) spaties op de band over ponserstroom i.

```
procedure PUNSPACE(I,N);value I,N; integer I,N;
for N:=N step -1 until 1 do PUNOCT(I,16);
```

3.5.2
```
procedure PUNSPACE1(N);value N;integer N;
for N:=N step -1 until 1 do PUNOCT(1,16);
```

3.5.3
```
procedure PUNSPACE2(N);value N;integer N;
for N:=N step -1 until 1 do PUNOCT(2,16);
```

3.5.4
```
procedure PUSPACE(N);value N;integer N;
for N:=N step -1 until 1 do PUNOCT(1,16);
```

3.6.1 procedure PUNRUNOUT(I,N); value I,N; integer I,N;

De procedure ponst N octaden blanks over ponserstroom i.

```
procedure PUNRUNOUT(I,N);value I,N;integer I,N;
for N:=N step -1 until 1 do PUNOCT(I,0);
```

3.6.2 procedure PUNRUNOUT1(N);value N;integer N;PUNRUNOUT(1,N);

3.6.3 procedure PUNRUNOUT2(N);value N;integer N;PUNRUNOUT(2,N);

3.6.4 procedure RUNOUT;PUNRUNOUT(1,80);

3.7.1 procedure PUNTAB(I); value I; integer I;

De procedure PUNTAB verhoogt de positie op de regel met tenminste 2 en ten hoogste 9, zodanig, dat een 8-voud bereikt wordt. Overschrijdt de positie op de regel de waarde van MAXPOSITION dan wordt er een regelopvoer ingelast.

```
procedure PUNTAB(I);value I;integer I;
begin integer X;
   X:=if I=1 then (8+((PUNCHPOSITION1+1):8)×8-PUNCHPOSITION1)
             else (8+((PUNCHPOSITION2+1):8)×8-PUNCHPOSITION2);
   for X:=Xstep -1 until 1 do PUNCHARF(I,93)
end PUNTAB(I);
```

3.7.2 procedure PUNTAB1;PUNTAB(1);

3.7.3 procedure PUNTAB2;PUNTAB(2);

3.8.1 procedure ENDPUNDOC(I); value I; integer I;

De procedure bewerkstelligt afsluiting van het ponsdokument. Achter de door de programmeur gespecificeerde informatie wordt nog geponst 80 octaden blanks, 1 erase en 120 octaden blanks.

```
procedure ENDPUNDOC(I);value I;integer I;
begin integer N;
   for N:=80 step -1 until 1 do PUNOCT(I,0);
   PUNOCT(I,127);
   for N:=120 step -1 until 1 do PUNOCT(I,0)
end ENDPUNDOC(I);
```

3.8.2 procedure ENDPUNDOC1;ENDPUNDOC(1);

3.8.3 procedure ENDPUNDOC2;ENDPUNDOC(2);

3.9.1 procedure PUNFLO(I,N,M,X); value I,N,M; integer I,N,M; real X;

De procedure is het ponsende analogon van FLOT. Zie 2.1.6
Afhankelijk van de waarde van de parameter I wordt geponst over ponserstroom 1 of 2. Indien de waarde van I niet gelijk is aan 1 of 2 volgt een dynamische foutmelding.
In de procedure FLOT zijn er de volgende aanpassingen:

```
LINEPOSITION:=PUNCHPOSITIONi;
NLCR:=NLCR(I);
PRSYM(N):=PUNSYM(I,N);
```

3.9.2 procedure PUNFLO1(N,M,X);value N,M,X;integer N,M;real X;PUNFLO(1,N,M,X);

3.9.3 procedure PUNFLO2(N,M,X);value N,M,X;integer N,M;real X;PUNFLO(2,N,M,X);

3.9.4 procedure FLOP(N,M,X);value N,M,X;integer N,M;real X;PUNFLO(1,N,M,X);

3.10.1 procedure PUNFIX(I,N,M,X); value I,N,M,X; integer I,N,M; real X;

De procedure is het ponsende analogon van FIXT. Zie 2.1.7.1
Afhankelijk van de waarde van de parameter I wordt geponst over ponserstroom 1 of 2. Indien de waarde van I niet gelijk is aan 1 of 2 volgt een dynamische foutmelding.
Aanpassingen zijn:

```
FLOT(13,3,X):=PUNFLO(I,13,3,X);
LINEPOSITION:=PUNCHPOSITIONi;
PRCHAR(N):=PUNCHARF(I,N);
SPACE(N):=for N:=abs(N) step -1 until 1 do PUNCHARF(I,93);
PRSYM(N):=PUNSYM(I,N);
```

3.10.2 procedure PUNFIX1(N,M,X);value N,M,X;integer N,M;real X;
PUNFIX(1,N,M,X);

3.10.3 procedure PUNFIX2(N,M,X);value N,M,X;integer N,M;real X;
PUNFIX(2,N,M,X);

3.10.4 procedure FIXP(N,M,X);value N,M,X;integer N,M;real X;
PUNFIX(1,N,M,X);

3.11.1 procedure PUNABSFIX(I,N,M,X); value I,N,M,X; integer I,N,M; real X;

De procedure is het ponsende analogon van ABSFIXT. Zie 2.1.7.2
Aanpassingen zijn:

```
FLOT(13,3,X):=PUNFLO(I,13,3,X);
LINEPOSITION:=PUNCHPOSITIONi;
PRCHAR(N):=PUNCHARF(I,N);
SPACE(N):=for N:=abs(N) step -1 until 1 do PUNCHARF(I,93);
PRSYM(N):=PUNSYM(I,N);
```

3.11.2 procedure PUNABSFIX1(N,M,X);value N,M,X;integer N,M;real X;
PUNABSFIX(1,N,M,X);

3.11.3 procedure PUNABSFIX2(N,M,X);value N,M,X;integer N,M;real X;
PUNABSFIX(2,N,M,X);

3.11.4 procedure ABSFIXP(N,M,X);value N,M,X;integer N,M;real X;
PUNABSFIX(1,N,M,X);

3.12.1 procedure PUNUM(I,X);valueI,X; integer I; real X;

De procedure is het ponsende analogon van PRINT. Zie 2.1.8.1
Aanpassingen zijn:

```
PRCHAR(N):=PUNCHARF(I,N);
LINEPOSITION:=PUNCHPOSITIONi;
FIXT(13,0,X):=PUNFIX(I,13,0,X);
SPACE(N):=for N:=abs(N) step -1 until 1 do PUNCHARF(I,93);
FLOT(13,3,X):=PUNFLO(I,13,3,X);
```

3.12.2 procedure PUNUM1(X);value X;real X;PUNUM(1,X);

3.12.3 procedure PUNUM2(X);value X;real X;PUNUM(2,X);

3.12.4 procedure PUNCH(X);value X;real X;PUNUM(1,X);

3.13.1 procedure PUNTEXT(I,S); value I; integer I; string S;

De procedure is het ponsende analogon van PRINTTEXT. Zie 2.1.9
Aanpassing:

```
PRSYM(N):=PUNSYM(I,N);
```

3.13.2 procedure PUNTEXT1(S);string S;PUNTEXT(1,S);

3.13.3 procedure PUNTEXT2(S);string S;PUNTEXT(2,S);

3.13.4 procedure PUTEXT(S);string S;PUNTEXT(1,S);

3.14 Het omvattende blok van de systeemuitvoerprocedures voor de bandponsers.

```
begin integer PUNCHPOSITION1,PUNCHPOSITION2,SPACER1,SPACER2,MAXPOSITION;
      Boolean PUNOCTPAST1,PUNOCTPAST2,LOWER1,LOWER2;
      < systeemuitvoerprocedures bandponsers >
      PUNCHPOSITION1:=PUNCHPOSITION2:=SPACER1:=SPACER2:=0;
      MAXPOSITION:=120;
      PUNOCTPAST1:=PUNOCTPAST2:=true;
      L END
end
```

BIJLAGE I

Karakter representatie van Algol 60 symbolen

basic symbols	EL X-8	CD-6000	IBM-360 OS 48 char-set	Burroughs B6700	IBM 72 59 char-set additional representations
a t/m z	a t/m z				
A t/m Z	A t/m Z	A t/m Z	A t/m Z	A t/m Z	
0 t/m 9	0 t/m 9	0 t/m 9	0 t/m 9	0 t/m 9	
+ - × /	+ - × /	+ - × /	+ - × /	+ - × of TIMES /	
÷	÷	/ of 'DIV'	'/'	DIV	
↑	↑	'POWER'	'POWER' of ××	~~××~~	
<	<	'LESS'	'LESS'	< of LSS	<
≤	≤	'NOT GREATER'	'NOTGREATER'	≤ of LEQ	<=
=	=	= of 'EQUAL'	= of 'EQUAL'	= of EQL	
≥	≥	'NOT LESS'	'NOTLESS'	≥ of GEQ	>=
>	>	'GREATER'	'GREATER'	> of GTR	>
≠	≠	'NOT EQUAL'	'NOTEQUAL'	NEQ	¬=
≡	≡	'EQUIV'	'EQUIV'	EQV	
	⊐	'IMPL'	'IMPL'	IMP	
∨	∨ of or	'OR'	'OR'	OR	|
∧	∧ of and	'AND'	'AND'	AND	
¬	¬ of non of not	'NOT'	'NOT'	NOT	¬
,	,	,	,	,	
.	.	.	.	.	
␣	␣	'	'	c	
:	:	..	..	:	:
;	;	.,	.,	;	;
:=	:=	.= of ..=	.= of ..=	:=	:=

Karakter representatie van Algol 60 symbolen

basic symbols	EL X-8	CD-6000	IBM-360 OS 48 char-set	Burroughs B6700	IBM 72 59 char-set additional representations
(	(	(	(	(	
)	)	)	)	)	
[	[	(/	(/	[	
]	]	/)	/)	]	
	≮	'('	'('	"	
	≯	')'	')'	"	
go to	goto, go to, go to	'GO TO'	'GOTO'	GO of GO TO	
if	if	'IF'	'IF'	IF	
then	then	'THEN'	'THEN'	THEN	
else	else	'ELSE'	'ELSE'	ELSE	
for	for	'FOR'	'FOR'	FOR	
do	do	'DO'	'DO'	DO	
step	step	'STEP'	'STEP'	STEP	
until	until	'UNTIL'	'UNTIL'	UNTIL	
while	while	'WHILE'	'WHILE'	WHILE	
comment	comment	'COMMENT'	'COMMENT'	COMMENT	
begin	begin	'BEGIN'	'BEGIN'	BEGIN	
end	end	'END'	'END'	END	
own	own	'OWN'		OWN	
boolean	boolean, Boolean	'BOOLEAN'	'BOOLEAN'	BOOLEAN	
integer	integer	'INTEGER'	'INTEGER'	INTEGER	
real	real	'REAL'	'REAL'	REAL	
array	array	'ARRAY'	'ARRAY'	ARRAY	
switch	switch	'SWITCH'	'SWITCH'	SWITCH	
procedure	procedure	'PROCEDURE'	'PROCEDURE'	PROCEDURE	
string	string	'STRING'	'STRING'	STRING	
label	label	'LABEL'	'LABEL'	LABEL	
value	value	'VALUE'	'VALUE'	VALUE	

BIJLAGE II

Opmerkingen:
Bij de conversie van X-8 Algolteksten dient men speciale aandacht te verlenen aan die punten welke voorafgegaan worden door een doorbalkt vraagteken "?".
Vanwege het grote aantal uitbreidingen van de Burroughs B6700 ten opzichte van het Revised Report zijn alleen die uitbreidingen opgenomen die relevant zijn voor de uitwerking van een conversie van X-8 Algolteksten.

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	Beperkingen
<declarator>	2.3.9	↑ library ↑ complex ↑ mtape		
comment	2.3			in commentaar van welke vorm dan ook mag begin niet voorkomen end
types	2.5.4			alleen integers met een absolute waarde kleiner of gelijk aan 2 ↑ 26 - 1 zijn van type integer
<proper string>	2.6.1.1		↑ onderstreepte spatie ↑ doorbalkte spatie ↑ ? ↑ " ↑ ' ↑ flexowriter spatie ↑ ↥ ↑ nlcr	
type	2.8		↑ complex	

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	Beperkingen
standard functions	3.2.4		↑ re(z) type real ↑ im(z) type real ↑ com(z) type real ↑ mod(z) type real ↑ gcc(z) type real	
arithmetic primary	3.3.1.3	↑ (<assignment statement>)		
operators and types	3.3.4		↑ de samenstellende delen van simple arithmetic expression kunnen van het type complex zijn	
operators	3.3.4.1			zijn beide operanden integer dan is het resultaat toch real of complex
operations	3.3.4.2		↑ bij de deling mogen de operanden complex zijn, het resultaat is van type complex, de bewerking : is ook gedefinieerd als niet beide operanden integer zijn, het resultaat is van type real	
<label>	3.5.1.1	<unsigned integer>		
<type>	5.1.1.2	? complex		
<procedure body>	5.4.1.9	<code>		
specifications	5.4.5			* specificaties verplicht
subscripted variables	5.2.5			* voor own arrays mogen de upper en lower bounds slechts gegeven worden door arithmetic expressions waarin variables en function designators als primary niet voorkomen

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	CD-6000 Beperkingen
<letter>	2.1			kleine letters zijn niet gedefinieerd
<delimiter>	2.3.1	<code procedure body indicator> † <code procedure body indicator> ::= code		
types	2.5.4			integers met een absolute waarde kleiner of gelijk aan 2 ↑ 48 - 1 zijn van type integer, deze integers vermenigvuldigd met positieve machten van 2 kleiner of gelijk 1022 zijn ook van type integer
<proper string>	2.6.1.1			het karakter geassocieerd met de externe BCD-008
exponentiation	3.3.4.3		met a van type integer en i een variabele van type integer > 0 is het resultaat real	
<label>	3.5.1.1	<unsigned integer>		
<program>	4.1.1.11	<program> ::= <unlabeled block>|<un labeled compound>		
go to	4.3.5			een go to naar een niet gedefinieerde switch designator geeft een foutmelding
for statements	4.6.3			als de gecontroleerde variabele subscripted is dan heeft verandering van de subscript geen invloed op het array element waarmee getest wordt
go to	4.6.6		een go to statement buiten een for statement verwijzend naar een label binnen deze for statement geeft aanleiding tot een foutmelding	

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	CD-6000 Beperkingen
procedure statement	4.7.5			hoogstens 63 formele of actuele parameters niet meer dan 62 constanten als actuele parameter
<code>	4.7.8		* De procedure body in kode wordt weergegeven door code	
own arrays	5.2.5			* own array kent geen dynamische grenzen
<procedure body>	5.4.1.9	<code> ::= code <unsigned integer>		
procedure declaraties	5.4.3		een procedure declaratie mag ter compilatie worden aangeboden	
specificaties	5.4.5			* specificaties verplicht

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	IBM-360 OS Beperkingen
<letter>	2.1			kleine letters zijn niet gedefinieerd
<separator>	2.3.7	blank	in een "word" delimiter mogen blanks voorkomen	
<declarator>	2.3.9	own ↑ code		
identifiers	2.4			van een identifier zijn de eerste zes toegestane tekens significant
types	2.5.4			alleen integers met een absolute waarde kleiner of gelijk aan 2 31 - 1 zijn van type integer
exponentiation	3.3.4.3		* als a en i variabelen van type integer zijn is het resultaat van type real	
<label>	3.5.1.1	<unsigned integer>		
go to	4.3.5			een go to naar een niet gedefinieerde switch designator is niet gedefinieerd
for statements	4.6.3			wanneer de variabele array element is, wordt de test steeds uitgevoerd op het oorspronkelijk array element
procedure statement	4.7.5			hoogstens 15 actuele parameters
<code>	4.7.8		* de procedure body in kode wordt weergegeven door code	
<local or own type>	5.1.1.3	own		

Subject	R.R.	Syntax Toevoeging / Verwijdering	Uitbreidingen	IBM-360 OS Beperkingen
<procedure body>	5.4.1.9	<code> ::= code		
procedure declaraties	5.4.3		een procedure declaratie mag ter compilatie worden aangeboden	
specificaties	5.4.5			* specificaties verplicht

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	Burroughs B 6700 Beperkingen
<letter>	2.1			kleine letters zijn niet gedefinieerd
<operator>	2.3.2	<replacement operator> <replacement operator> ::= :=		
<sequential operator>	2.3.6	while		
<separator>	2.3.7	while :=		een <space> moet altijd scheiden: "multicharacter" <delimiter> <logical value> <identifier> <unsigned number>
<declarator>	2.3.9	double | label | forward		
<specificator>	2.3.10	string | label		
comment	2.3			in commentaar volgend op end mag until niet voorkomen
identifiers	2.4			identifiers zijn niet langer dan 63 toegestane tekens
types	2.5.4			alleen integers met een absolute waarde kleiner of gelijk aan 2 ↑ 39 − 1 zijn van type integer integers met een absolute waarde kleiner of gelijk aan 2 ↑ 78 − 1 zijn van type double
strings	2.6	zie 5.3 Burroughs B 6700 Extended Algol Language Information Manual		

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	Burroughs B6700 Beperkingen
<actual parameter>	3.2.1.2	<string>		
<parameter delimiter>	3.2.1.4	: string quote aan beide zijden van <letter string>		
<primary>	3.3.1.3	(<arithmetic assignment>)		
operators	3.3.4.1			indien beide operanden van type integer en de absolute waarde van het resultaat ≥ 2 ↑ 39 is het resultaat van type real
operations	3.3.4.2		de bewerking : is ook gedefinieerd wanneer niet beide operanden van type integer zijn	
exponentiation	3.3.4.3			als a en i van type integer en de absolute waarde van het resultaat > 2 ↑ 39 dan is het resultaat van type real als a van type integer en r van type real met r = 0 dan is het resultaat van type integer
<label>	3.5.1.1	<unsigned integer>		
<simple designational expression>	3.5.1.4	((<designational expression>) <simple designational expression> ::= <label>|<switch designator>| <designational expression>		
<designational expression>	3.5.1.5	<designational expression> ::= <if clause> <simple designational expression> else <designational expression>		

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	Burroughs B 6700 Beperkingen
compound statements and blocks	4.1.1	Syntax volkomen anders		† programma mag niet met een label beginnen
<left part list>	4.2.1.2			* omdat de left part list recursief gegeven wordt geeft 4.2.1.2 geen aanleiding tot wezenlijke verschillen
for statements	4.6			* wanneer er een label voor een for statement staat heet het geen iteration statement meer
<letter string>	4.7.1.2	zie 3.2.1.2		
<actual parameter list>	4.7.1.4	zie 3.2.1.4		
declarations	5	<label declaration> <forward reference declaration>		
type	5.1.1	<forward reference declaration> ::= <forward function declaration>| <forward procedure declaration>| <forward switch declaration> <forward function declaration> ::= <type> procedure <procedure heading> forward <forward procedure declaration> ::= procedure <procedure heading> forward <forward switch declaration> ::= switch <identifier> forward		

Subject	R.R.	Syntax Toevoeging Verwijdering	Uitbreidingen	Burroughs B 6700 Beperkingen
switch declarations	5.3.1	<label declaration> ::= label <identifier list> <identifier list> ::= <identifier>| <identifier list>, <identifier>		† labels moeten gedeclareerd worden
<specifier>	5.4.1.6	string array <type> array		
<specification part>	5.4.1.7	<array specification> ::= <array type> array <array specifier list> <array type> ::= <empty>|<type> <array specifier list> ::= <array specifier>|<array specifier list>, <array specifier> <array specifier> ::= <identifier list> [<lower bound list>] <lower bound list> ::= <specified lower bound>|<lower bound list>, <specified bound list>, <specified lower bound> <specified lower bound> ::= x|<integer>		
<procedure body>	5.4.1.9	<code>		† de procedure body is niet gelabeld
specifications	5.4.5			† verplichte dimensie specificaties bij formele array's

BIJLAGE III

TABEL VAN OCT WAARDEN

OCT waarde	MC-flexowritercode		
	lowercase	uppercase	case onafhankelijk
-1			tape end
0			blank
1	1	V	
2	2	×	
4	4	=	
7	7	]	
8	8	(	
11			stopcode
14		|	
16	-		space
19	3	/	
21	5	;	
22	6	[	
25	9	)	
26			carriage return
32	0	∧	
35	t	T	
37	v	V	
38	w	W	
41	z	Z	
49	<	>	
50	s	S	
52	u	U	
55	x	X	
56	y	Y	
59	¤	'	
64	¯	¬	
67	l	L	
69	n	N	
70	o	O	
73	r	R	
81	j	J	
82	k	K	
84	m	M	
87	p	P	
88	q	Q	
91	,	?	
97	a	A	
98	b	B	
100	d	D	
103	g	G	
104	h	H	
107	.	:	
112	+	"	
115	c	C	
117	e	E	
118	f	F	
121	i	I	
122			lowercase
124			uppercase
127			erase

BIJLAGE III

TABEL VAN CHARF WAARDEN

CHARF waarde	OCT waarde	flexowriterteken	opmerkingen
-2			foute pariteit of onbekende ponsing
-1			stopcode of tape end
0	32	0	
1	1	1	
2	2	2	
3	19	3	
4	4	4	
5	21	5	
6	22	6	
7	7	7	
8	8	8	
9	25	9	
10	97	a	
11	98	b	
12	115	c	
13	100	d	
14	117	e	
15	118	f	
16	103	g	
17	104	h	
18	121	i	
19	81	j	
20	82	k	
21	67	l	
22	84	m	
23	69	n	
24	70	o	
25	87	p	
26	88	q	
27	73	r	
28	50	s	
29	35	t	
30	52	u	
31	37	v	
32	38	w	
33	55	x	
34	56	y	
35	41	z	
37	97	A	
38	98	B	
39	115	C	
40	100	D	
41	117	E	
42	118	F	
43	103	G	
44	104	H	
45	121	I	
46	81	J	
47	82	K	
48	67	L	
49	84	M	
50	69	N	
51	70	O	
52	87	P	
53	88	Q	
54	73	R	
55	50	S	
56	35	T	
57	52	U	
58	37	V	
59	38	W	
60	55	X	
61	56	Y	
62	41	Z	
64	112	+	
65	64	-	
66	2	×	
67	19	/	
70	4	=	
72	49	<	
74	49	>	
76	64	¬	
79	1	∨	
80	32	∧	
87	91	,	
88	107	.	
89	59	¤	
90	107	:	
91	21	;	
93	16		space
98	8	(	
99	25	)	
100	22	[	
101	7	]	
119	26		new line carriage return
120	59	'	
121	112	"	
122	91	?	

Voor onderstreepte, doorbalkte en onderstreept-doorbalkte MC-flexowritertekens is de CHARF waarde gelijk aan de CHARF waarde van het basissymbool vermeerderd met respectievelijk 128, 256 of 384.

BIJLAGE III TABEL VAN SYMWAARDEN

SYM waarde	CHARF waarde	OCT waarde	flexowriter teken(s)	opmerkingen
-2				onbekende worddelimiter
-1	-2			foute pariteit of onbekende ponsing
0	0	32	0	
1	1	1	1	
2	2	2	2	
3	3	19	3	
4	4	4	4	
5	5	21	5	
6	6	22	6	
7	7	7	7	
8	8	8	8	
9	9	25	9	
10	10	97	a	
11	11	98	b	
12	12	115	c	
13	13	100	d	
14	14	117	e	
15	15	118	f	
16	16	103	g	
17	17	104	h	
18	18	121	i	
19	19	81	j	
20	20	82	k	
21	21	67	l	
22	22	84	m	
23	23	69	n	
24	24	70	o	
25	25	87	p	
26	26	88	q	
27	27	73	r	
28	28	50	s	
29	29	35	t	
30	30	52	u	
31	31	37	v	
32	32	38	w	
33	33	55	x	
34	34	56	y	
35	35	41	z	
37	37	97	A	
38	38	98	B	
39	39	115	C	
40	40	100	D	
41	41	117	E	
42	42	118	F	
43	43	103	G	
44	44	104	H	
45	45	121	I	

BIJLAGE III TABEL VAN SYMWAARDEN

SYM waarde	CHARF waarde	OCT waarde	flexowriter teken(s)	opmerkingen
46	46	81	J	
47	47	82	K	
48	48	67	L	
49	49	84	M	
50	50	69	N	
51	51	70	O	
52	52	87	P	
53	53	88	Q	
54	54	73	R	
55	55	50	S	
56	56	35	T	
57	57	52	U	
58	58	37	V	
59	59	38	W	
60	60	55	X	
61	61	56	Y	
62	62	41	Z	
64	64	112	+	
65	65	64	-	
66	66	2	×	
67	67	19	/	
68	218		÷	
69	336		↑	
70	70	4	=	
71	326		≠	
72	72	49	<	
73	200		≤	
74	74	49	>	
75	202		≥	
76	76	64	¬	
76			non	
76			not	
77	198		≡	
78	204		≢	
79	79	1	∨	
79			or	
80	80	32	∧	
80			and	

BIJLAGE III TABEL VAN SYMWAARDEN

SYM waarde	CHARF waarde	OCT waarde	flexowriter teken(s)	opmerkingen
81			goto	
81			go to	
81			go to	
82			for	
83			step	
84			until	
85			while	
86			do	
87	87	91	,	
88	88	107	.	
89	89	59	10	
90	90	107	:	
91	91	21	;	
92			complex	
93	93	16		spatie
94			if	
95			then	
96			else	
97			comment	
98	98	8	(	
99	99	25	)	
100	100	22	[	
101	101	7	]	
102	328		≤	
103	330		≥	
104			begin	
105			end	
106			own	
107			real	
108			integer	
109			Boolean	
109			boolean	
110			string	
111			array	

BIJLAGE III TABEL VAN SYMWAARDEN

SYM waarde	CHARF waarde	OCT waarde	flexowriter teken(s)	opmerkingen
112			procedure	
113			switch	
114			label	
115			value	
116			true	
117			false	
118			progend	
119	119	26		new line carriage return
120	120	59	'	
121	121	112	"	
122	122	91	?	
123			mtape	
124			library	
125			invslash	
126	221		←	
127	349		|	
128			nlcr	
129	215		↑	
130			algol	
131			correction	

BIJLAGE IV

LITERATUUR OVERZICHT

1 EL X-8
Technische Hogeschool Eindhoven
Rekencentrum Informatie nr. 30,32 d.d. 1-12-1971

2 CD-6000
Control Data Corporation
Algol generic reference manual
Publicatie nr. 60214900 d.d. 1-5-1968

3 IBM-360 OS
IBM System/360
Operating System Algol Language
File nr. S360-26
Form C28-6615-1 d.d. 15-4-1969

4 Burroughs B6700
Burroughs B6700 Extended Algol Language
Information Manual nr. 5000128 d.d. 1-7-1971

5 Revised Report
Revised Report on the Algorithmic Language Algol 60
International Federation for Information
Processing 1962.