stichting
mathematisch
centrum

MC

LR 1.1

APRIL 1971

samensteller: D. GRUNE

HANDLEIDING MILLI-SYSTEEM VOOR DE EL X8

2e boerhaavestraat 49 amsterdam

0. Algemeen.

0.1. Voorwoord.

0.1.1. Dit rapport bevat een handleiding bij het vervaardigen van ALGOL 60 programma's en invoergegevens voor het milli–ALGOL 60 systeem van de EL X8 computer van het Mathematisch Centrum.

0.1.2. De ontwerper en maker van dit systeem, prof. dr. F.E.J. Kruseman Aretz van het Natuurkundig Laboratorium der N.V. Philips te Waalre, zeggen wij hierbij dank voor de wijze waarop hij ons met raad en daad terzijde heeft gestaan bij de aanpassing aan onze configuratie. Ook zijn wij zeer dankbaar voor de bereidwilligheid van het Natuurkundig Laboratorium der N.V. Philips om dit bedrijfssysteem te onzer beschikking te stellen. Voorts gaat onze dank uit naar de MC–medewerkers P.Beertema, R.Brinkhuijsen, J.V.M. van der Grinten en C.Zuidema die meegeholpen hebben het systeem aan te passen aan de eisen van het MC en die ook voor verdere uitbreiding zorg zullen dragen.

Voor deze handleiding hebben wij geput uit MR81 "Het MC ALGOL 60 systeem van de X8. Voorlopige programmeurshandleiding" door F.E.J. Kruseman Aretz, uit het rapport "Het ALGOL 60 systeem van de X8" door W.P.J.Fontein van het Natuurkundig Laboratorium der N.V. Philips, en uit NR9, "Ponsconventies voor ALGOL 60 programma's voor de X8" door J.V.M. van der Grinten e.a.; verder geldt onze dank de velen die aan het verwezenlijken van deze handleiding hebben meegewerkt: adviseurs, correctors, mensen met ideeen, typistes en typografen, en in het bijzonder de samensteller D.Grune. Ook de X8 heeft zijn steentje bijgedragen.

0.1.3. Hen die vertrouwd zijn met het oude ALGOL 60 systeem en MR81 wordt aangeraden althans de hoofdstukken 4, 5 en 6 en paragraaf 3.3. door te nemen.

0.1.4. Deze handleiding is als losbladig systeem opgezet. De pagina's zijn gedateerd en per hoofdstuk genummerd. Aanvullingen en wijzigingen zullen regelmatig aan de balie van het MC beschikbaar zijn. Steeds bevat dan paragraaf 0.2. een bijgewerkte lijst van pagina's met datering.

R.P. van de Riet

0.2. Lijst van pagina's.

0.2. Lijst van pagina's, vervolg.

0.3. Inhoudsopgave.

4. Standaard procedures voor Input/Output
    4.1. Input procedures
        4.1.1. rehep
        4.1.2. resym
        4.1.3. resymbol
        4.1.4. read
    4.2. Uitvoer bandponser
        4.2.1. Besturingsprocedures
            4.2.1.1. outflexo
            4.2.1.2. outiso
        4.2.2. Non-numeriek
            4.2.2.1. puhep
            4.2.2.2. pusym
            4.2.2.3. puspace
            4.2.2.4. punlcr
            4.2.2.5. runout
            4.2.2.6. stopcode
            4.2.2.7. putext
        4.2.3. Numeriek
            4.2.3.1. absfixp
            4.2.3.2. fixp
            4.2.3.3. flop
            4.2.3.4. punch
    4.3. Uitvoer regeldrukker
        4.3.1. Besturingsprocedures
        4.3.2. Non-numeriek
            4.3.2.1. prsym
            4.3.2.2. space
            4.3.2.3. tab
            4.3.2.4. nlcr
            4.3.2.5. carriage
            4.3.2.6. new page
            4.3.2.7. printtext

1. Beschrijving van ALGOL 60 voor de X8.

1.1. Eisen te stellen aan ALGOL 60 programma's.

1.1.1. Het ALGOL 60-systeem voor de X8 accepteert programma's, geschreven in ALGOL 60 als gedefinieerd in het Revised Report on the Algorithmic Language ALGOL 60, onder de voorlopige beperking dat hierin:
a. geen own <type> array's gedeclareerd worden,
b. geen integer labels met een integer-waarde > 67108863 voorkomen,
c. het aantal gedeclareerde locale variabelen en locale labels binnen een procedure body zekere, zeer ruime grenzen niet overschrijdt,
d. het aantal entries in een switch declaration zekere, zeer ruime grenzen niet overschrijdt.

1.1.2. Een aantal (functie-) procedures mag, zonder dat zij in het programma gedeclareerd behoeven te worden, gebruikt worden. Naast de in 3.2.4. van het Revised Report opgenomen elementaire functies behoren hiertoe onder meer alle in- en uitvoer-procedures. Voor een volledige opsomming zie hoofdstuk 3. en 4..

1.1.3. Uitsluitend in commentaar (na comment en end) en binnen strings mogen, naast de basic symbols van ALGOL 60, ook de symbolen apostrof, aanhalingsteken en vraagteken (', " en ?), alsmede willekeurige onderstreepte of doorbalkte combinaties in beperkte mate voorkomen (voor restricties zie 6.1.4. t/m 6.1.7.).

1.1.4. Formele parameters van procedures behoeven, voor zover ze niet in de value part voorkomen, niet gespecificeerd te worden. Het is gewoonlijk voordelig, zo veel mogelijk parameters te specificeren, daar de extra informatie, verschaft door de specificatie, mede in de test op overeenkomst formele/actuele parameters wordt betrokken, en bovendien de efficientie van de uitvoering van het programma ten goede kan komen. Het is evenwel mogelijk, procedures te schrijven die aan de afwezigheid van een of meer specificaties hun speciale betekenis ontlenen.

1.1.5. In tegenstelling tot 4.3.5. van het Revised Report wordt de evaluatie van een switch designator die niet tot een entry uit de switch list leidt, als een ongeoorloofde situatie beschouwd.

1.2. Enige interpretaties van ALGOL 60.

1.2.1. De specificaties real en integer hebben de volgende betekenis:

1.2.1.1. als de bijbehorende formele parameter in de value part voorkomt, wordt, conform het Revised Report 4.7.3., in de procedure body een locale variabele gedeclareerd van het type volgens de specificatie, en aan die variabele wordt de waarde van de corresponderende actuele expressie geassigneerd, zonodig daarbij een real resultaat afrondend tot een integer waarde,

1.2.1.2. als de formele parameter niet in de value part voorkomt, worden beide specificaties geinterpreteerd als een mededeling "aritmetisch" en in hoofdzaak gebruikt voor de syntactische controle. Bij iedere aanroep van de procedure wordt overal in de body de formele parameter vervangen door de corresponderende actuele (zie Revised Report 4.7., in het bijzonder 4.7.3.2.), en het type van de actuele parameter dringt dus door tot in de body, zonder dat enige transferfunctie wordt ingelast, ongeacht de specificatie real of integer.

1.2.2. De specificatie real array of integer array wordt, als de formele parameter niet in de value part voorkomt, eveneens opgevat als een mededeling "aritmetisch array" met dezelfde consequenties als aangegeven onder 1.2.1.2. bij de specificatie real en integer.

1.2.3. De specificatie array, zonder type-vermelding, wordt, als de bijbehorende formele parameter in de value part voorkomt, geidentificeerd met de specificatie real array. Anders wordt de specificatie array beschouwd als een mededeling, dat de formele identifier de identifier van een array voorstelt.

1.2.4. Bij formele parameters die in de value part voorkomen, worden de specificaties real procedure, integer procedure en Boolean procedure geidentificeerd met respectievelijk real, integer en Boolean.

1.2.5. Een formele parameter, voorkomend in de value part, mag gespecificeerd worden als label. De corresponderende actuele parameter moet dan een designational expression zijn; deze wordt bij het binnenkomen van de procedure, evenals alle andere parameters uit de value part, geevalueerd.

1.2.6. Bij de uitwerking van expressies worden de operanden geevalueerd in de volgorde van de ALGOL-tekst, van links naar rechts. Bij het binnenkomen van een procedure worden de formele parameters uit de value part geevalueerd in de volgorde van de formele-parameterlijst; echter worden eerst de simpele value parameters, en daarna de value arrays afgehandeld.

2. De aritmetiek van het ALGOL-systeem

2.1. Een geheugenplaats of woord bestaat uit 27 binaire posities (bits).

2.2. Een variabele van het type boolean beslaat een woord in het geheugen, een boolean array een woord per 27 elementen.

2.3. Variabelen van type integer bezetten in het geheugen een enkel woord. Als gevolg hiervan is het waardebereik van integer-variabelen beperkt van - 67 108 863 (= - 2↑26 + 1) tot en met + 67 108 863 (= 2↑26 - 1), en mag, tijdens de uitvoering van het programma, geen waarde aan een integer-variabele toegekend worden, die (na eventuele afronding) in absolute waarde groter is dan 67 108 863. Dit geldt ook voor impliciete assignments aan integers (vergelijk Revised Report 3.1.4.2., 4.7.3.1., 5.2.4.1., en 5.4.4.).
Een uitzondering hierop vormt de integer procedure entier, waarvan het waardebereik niet in bovenstaande zin beperkt is (zie 3.1.9.).
Een integer array beslaat in het geheugen een woord per element.

2.4. Variabelen van type real bezetten in het geheugen twee woorden, en worden voorgesteld in floating-point representatie met een mantisse van 40 binaire posities (40 bits) en teken en een binaire exponent van 11 bits en teken. Hun waarde representeert in het algemeen een niet-geheel getal met een relatieve precisie van ongeveer 12 decimalen. Het in absolute waarde grootst mogelijke getal dat op deze wijze kan worden voorgesteld is $(2 \uparrow 40 - 1) \times 2 \uparrow (2 \uparrow 11 - 1)$, ongeveer $_{10}628$. Het in absolute waarde kleinst mogelijke doch van 0 verschillende getal is $1 \times 2 \uparrow (- 2 \uparrow 11 + 1)$, ongeveer $_{10}-616$. Gehele getallen, in absolute waarde kleiner dan 1 099 511 627 776 (= $2 \uparrow 40$) worden door een variabele van het type real exact voorgesteld.
Een real array beslaat in het geheugen twee woorden per element.

2.5. Voor de declaratie van een array zijn behalve het hierboven vermelde aantal geheugenplaatsen nog (aantal dimensies + 1) × 3 + 1 geheugenplaatsen nodig voor administratiedoeleinden.

2.6.1. De uitwerking van expressies geschiedt steeds in floating-point aritmetiek. De aritmetische operaties +, -, ×, en / geven het best mogelijke, in bovengenoemde floating-point representatie voorstelbare resultaat. Dit impliceert in het bijzonder dat de monotonie behouden blijft, bij voorbeeld als $a < b$ en $c \geq 0$, dan is $a \times c \leq b \times c$.

2.6.2. Als de operanden bij de operaties +, −, en × gehele getallen zijn, in absolute waarde kleiner dan 1 099 511 627 776, en het resultaat van deze operaties ook in absolute waarde deze grens niet overschrijdt, is dit resultaat exact.

2.6.3. Bij de aritmetische operatie : is het bereik van de operanden niet beperkt tot de integers. Als operanden zijn toegelaten de gehele getallen tussen −(2 ↑ 40 − 1) en +(2 ↑ 40 − 1), inclusief, en alle getallen die in absolute waarde groter zijn dan 2 ↑ 40 − 1 (dit zijn samen alle waarden van x waarvoor geldt entier(x)=x).

2.6.4. De aritmetische operatie ↑ levert een resultaat met een relatieve nauwkeurigheid van ongeveer 12 decimalen. Als de exponent een geheel getal is, in absolute waarde kleiner dan 32, wordt het resultaat opgebouwd door herhaalde vermenigvuldiging.

2.7. De waarde van expressies en variabelen van type real ligt in absolute waarde tussen ongeveer ₁₀628 en ongeveer ₁₀−616, of is exact 0.
Als bij aritmetische operaties een resultaat zou ontstaan, dat in absolute waarde groter dan ongeveer ₁₀628 zou zijn, wordt het grootst mogelijke, nog voorstelbare resultaat gevormd, ongeveer ₁₀628 met het correcte teken.
Een resultaat 0 ontstaat bij additie of subtractie alleen, als beide operanden in absolute waarde "bit voor bit" gelijk zijn, bij vermenigvuldigen slechts, als ten minste een der beide operanden 0 is en bij deling alleen als het deeltal 0 is terwijl de deler ≠ 0 is. In alle andere gevallen is het resultaat van een aritmetische operatie in absolute waarde tenminste ongeveer ₁₀−616.
Bij deling door 0 ontstaat, als het deeltal ≠ 0 is, het in absolute waarde grootst mogelijke resultaat. De waarde van 0/0 kan van geval tot geval verschillen.

2.8. De relaties =, ≠, >, <, ≥ en ≤ zijn exact. De waarde van de relatie a = b is dus slechts true, als de twee operanden a en b "bit voor bit" gelijk zijn. Evenzo leveren de relaties a > b en a < b zeker false, als de twee operanden a en b "bit voor bit" gelijk zijn.

3. Standaard-procedures.

De bibliotheek van het ALGOL 60-systeem voor de X8 bevat enige groepen procedures, die niet in het programma gedeclareerd behoeven te worden. De namen van deze procedures zijn opgebouwd uit kleine letters. In sommige gevallen mogen deze namen ook in hoofdletters gespeld worden; dit is dan in de beschrijving van de betreffende procedure vermeld. Het verdient aanbeveling de namen van de standaardprocedures steeds in kleine letters te schrijven.

3.1. Elementaire functies.

3.1.1. real procedure abs(x); value x; real x;
abs:= if x ≥ 0 then x else - x;

3.1.2. integer procedure sign(x); value x; real x;
sign:= if x = 0 then 0 else if x > 0 then 1 else - 1;

3.1.3. real procedure sqrt(x); value x; real x;
Als de vierkantswortel van x exact representeerbaar is in de floating-point aritmetiek van de X8, wordt dit exacte resultaat afgeleverd. In het bijzonder geldt sqrt(i × i) = i voor i = 0 (1) 1 048 575. Voor x < 0 wordt voor sqrt(x) een resultaat 0 gegeven.

3.1.4. real procedure sin(x); value x; real x;
Er is voldaan aan de eisen: sin(0) = 0 en abs(sin(x)) ≤ 1.

3.1.5. real procedure cos(x); value x; real x;
Er is voldaan aan de eisen: cos(0) = 1 en abs(cos(x)) ≤ 1.

3.1.6. real procedure arctan(x); value x; real x;
Deze levert de hoofdwaarde van arctan(x); het resultaat ligt tussen -pi/2 en +pi/2. Er is voldaan aan arctan(0) = 0.

3.1.7. real procedure ln(x); value x; real x;
Voor x < 0 wordt voor ln(x) het resultaat circa $-_{10}628$ afgeleverd. Er is voldaan aan ln(1) = 0.

3.1.8. real procedure exp(x); value x; real x;
Er is voldaan aan exp(0) = 1. Voor x < -1419 wordt circa $_{10}-616$ als resutaat van exp(x) afgeleverd, terwijl voor x > +1447 circa $_{10}628$ afgeleverd wordt.

3.1.9. integer procedure entier(x); value x; real x;
Bij wijze van uitzondering is het waardebereik van entier niet beperkt van - 67 108 863 tot en met + 67 108 863. Voor waarden van het argument die liggen tussen -(2 ↑ 40 - 1) en +(2 ↑ 40 - 1) levert entier het grootste gehele getal af dat niet groter is dan het argument, terwijl voor andere waarden van het argument entier het argument zelf aflevert.

3.2. Procedures voor het verwerken van reeksen, vectoren en matrices.

In het Milli-systeem is een aantal procedures opgenomen voor het verwerken van reeksen, vectoren en matrices. Met uitzondering van sum en inprod zijn deze procedures in extenso beschreven in:

T.J. Dekker,
ALGOL 60 procedures in numerical algebra, part 1.
Mathematical Centre Tracts 22(1968)

T.J. Dekker, W. Hoffman,
ALGOL 60 procedures in numerical algebra, part 2.
Mathematical Centre Tracts 23(1968)

De hieronder volgende beknopte beschrijving is bedoeld als geheugensteun voor de gebruiker; waar hierin hoofdstukken en secties vermeld zijn, verwijzen deze naar bovengenoemde publicaties.

3.2.1. Operaties op reeksen.

3.2.1.1.
```
real procedure sum(i,a,b,x); value b; integer i,a,b; real x;
begin   real s;
        s:= 0;
        for i:= a step 1 until b do s:= s + x;
        sum:= s
end sum;
```
De function designator sum levert de som af van de waarden van de met x corresponderende actuele expressie, uitgerekend voor waarden van i = a (1) b. Voor b < a levert sum de waarde 0 af. De real procedure sum is in ALGOL-vorm in het systeem opgenomen. In plaats van sum mag men ook de naam SUM gebruiken.

3.2.1.2.
```
real procedure inprod(i,a,b,x,y); value b;
integer i,a,b; real x,y;
begin   real s;
        s:= 0;
        for i:= a step 1 until b do s:= s + x × y;
        inprod:= s
end inprod;
```
De function designator inprod levert af het inwendig product van twee vectoren van getalwaarden, verkregen door de met x, respectievelijk y corresponderende actuele expressies te evalueren voor i = a (1) b. Voor b < a levert inprod de waarde 0 af. De real procedure inprod is in ALGOL-vorm in het systeem opgenomen. In plaats van inprod mag men ook de naam INPROD gebruiken.

3.2.2. Vector operaties (Hoofdstuk 20).

3.2.2.1. Inwendige producten (Sectie 200).

3.2.2.1.1. comment mca 2000;
real procedure vecvec(l,u,shift,a,b); value l,u,shift;
integer l,u,shift; array a,b;
vecvec:= het inwendig product van de vectoren gegeven in array a[l:u] en array b[shift + l : shift + u].

3.2.2.1.2. comment mca 2001;
real procedure matvec(l,u,i,a,b); value l,u,i;
integer l,u,i; array a,b;
matvec:= het inwendig product van de rijvector gegeven in array a[i:i,l:u] en de vector gegeven in array b[l:u].

3.2.2.1.3. comment mca 2002;
real procedure tamvec(l,u,i,a,b); value l,u,i;
integer l,u,i; array a,b;
tamvec:= het inwendig product van de kolomvector gegeven in array a[l:u,i:i] en de vector gegeven in array b[l:u].

3.2.2.1.4. comment mca 2003;
real procedure matmat(l,u,i,j,a,b); value l,u,i,j;
integer l,u,i,j; array a,b;
matmat:= het inwendig product van de rijvector gegeven in array a[i:i,l:u] en de kolomvector gegeven in array b[l:u,j:j].

3.2.2.1.5. comment mca 2004;
real procedure tammat(l,u,i,j,a,b); value l,u,i,j;
integer l,u,i,j; array a,b;
tammat:= het inwendig product van de kolomvectoren gegeven in array a[l:u,i:i] en array b[l:u,j:j].

3.2.2.1.6. comment mca 2005;
real procedure mattam(l,u,i,j,a,b); value l,u,i,j;
integer l,u,i,j; array a,b;
mattam:= het inwendig product van de rijvectoren gegeven in array a[i:i,l:u] en array b[j:j,l:u].

3.2.2.1.7. comment mca 2006;
real procedure seqvec(l,u,il,shift,a,b); value l,u,il,shift;
integer l,u,il,shift; array a,b;
seqvec:= het inwendig product van de vectoren gegeven in array a[il:il+ (u+l-1) × (u-l) : 2] en array b[shift + l : shift + u], waarbij de elementen van de eerste vector zijn a[il + (j + l-1) × (j-l) : 2] voor j = l, ..., u.

3.2.2.1.8. comment mca 2008;
real procedure scaprd1(la,sa,lb,sb,n,a,b);
value la,sa,lb,sb,n; integer la,sa,lb,sb,n; array a,b;
scaprd1:= het inwendig product van de vectoren gegeven in array a[min(la,la+ (n-1) × sa): max(la,la + (n-1) × sa)] en array b[min(lb,lb+(n-1) × sb): max(lb,lb+ (n-1) × sb)], waarbij de elementen van de vectoren zijn a[la + (j-1)×sa] en b[lb + (j-1)×sb] voor j = 1, ..., n.

3.2.2.2. Eliminatie (Sectie 201).

3.2.2.2.1. comment mca 2010;
procedure elmvec(l,u,shift,a,b,x); value l,u,shift,x;
integer I,u,shift; real x; array a,b;
x maal de vector gegeven in array b[shift + l:shift + u] wordt opgeteld bij de vector gegeven in array a[l:u].

3.2.2.2.2. comment mca 2011;
procedure elmveccol(l,u,i,a,b,x); value l,u,i,x; integer I,u,i;
real x; array a,b;
x maal de kolomvector gegeven in array b[l:u,i:i] wordt opgeteld bij de vector gegeven in array a[l:u].

3.2.2.2.3. comment mca 2012;
procedure elmcolvec(l,u,i,a,b,x); value l,u,i,x;
integer I,u,i; real x; array a,b;
x maal de vector gegeven in array b[l:u] wordt opgeteld bij de kolomvector gegeven in array a[l:u,i:i].

3.2.2.2.4. comment mca 2013;
procedure elmcol(l,u,i,j,a,b,x); value l,u,i,j,x;
integer I,u,i,j; real x; array a,b;
x maal de kolomvector gegeven in array b[l:u,j:j] wordt opgeteld bij de kolomvector gegeven in array a[l:u,i:i].

3.2.2.2.5. comment mca 2014;
procedure elmrow(l,u,i,j,a,b,x); value l,u,i,j,x;
integer I,u,i,j; real x; array a,b;
x maal de rijvector gegeven in array b[j:j,l:u] wordt opgeteld bij de rijvector gegeven in array a[i:i,l:u].

3.2.2.2.6. comment mca 2019;
integer procedure maxelmrow(l,u,i,j,a,b,x); value l,u,i,j;
integer I,u,i,j; real x; array a,b;
x maal de rijvector gegeven in array b[j:j,l:u] wordt opgeteld bij de rijvector gegeven in array a[i:i,l:u].
maxelmrow:= de waarde van de tweede index van een in absolute waarde maximaal element van de nieuwe rijvector in array a.
Als evenwel l > u dan maxelmrow:= l.

3.2.2.3. Verwisselingen (Sectie 202).

3.2.2.3.1. comment mca 2020;
procedure ichvec(l,u,shift,a); value l,u,shift;
integer I,u,shift; array a;
de elementen van de vector gegeven in array a[l:u] en array a[shift+l,shift+u] worden verwisseld.

3.2.2.3.2. comment mca 2021;
procedure ichcol(l,u,i,j,a); value l,u,i,j; integer l,u,i,j;
array a;
de elementen van de kolomvectoren gegeven in array a[l:u,i:i] en array a[l:u,j:j] worden verwisseld.

3.2.2.3.3. comment mca 2022;
procedure ichrow(l,u,i,j,a); value l,u,i,j; integer l,u,i,j;

array a;
de elementen van de rijvectoren gegeven in array a[i:i,1:u] en array a[j:j,1:u] worden verwisseld.

3.2.2.3.4. comment mca 2023;
procedure ichrowcol(1,u,i,j,a); value 1,u,i,j; integer 1,u,i,j;
array a;
de elementen van de rijvector gegeven in array a[i:i,1:u] en de kolomvector gegeven in array a[1:u,j:j] worden verwisseld.

3.2.2.3.5. comment mca 2024;
procedure ichseqvec(1,u,il,shift,a); value 1,u,il,shift;
integer 1,u,il,shift; array a;
de elementen van de vectoren gegeven in array a[il:il + (u+1–1) × (u–1) : 2] en array a[shift+1:shift+u] worden verwisseld, waarbij de elementen van de eerste vector zijn a[il+ (j+1–1) × (j–1) : 2] voor j = 1, ..., u.

3.2.2.3.6. comment mca 2025;
procedure ichseq(1,u,il,shift,a); value 1,u,il,shift;
integer 1,u,il,shift; array a;
de elementen van de vectoren gegeven in array a[il:il+ (u+1–1) × (u–1) : 2] en array a[shift+il:shift+il+ (u+1–1) × (u–1) : 2] worden verwisseld, waarbij de elementen van de vectoren zijn a[il+ (j+1–1) × (j–1) : 2] en a[shift+il+ (j+1–1) × (j–1) : 2] voor j = 1, ..., u.

3.2.2.4. Rotaties (Sectie 203).

3.2.2.4.1. comment mca 2031;
procedure rotcol(1,u,i,j,a,c,s); value 1,u,i,j,c,s;
integer 1,u,i,j; real c,s; array a;
de kolomvector x gegeven in array a[1:u,i:i] en de kolomvector y gegeven in array a[1:u,j:j] worden vervangen door de vectoren cx+sy en cy–sx.

3.2.2.4.2. comment mca 2032;
procedure rotrow(1,u,i,j,a,c,s); value 1,u,i,j,c,s;
integer 1,u,i,j; real c,s; array a;
de rijvector x gegeven in array a[i:i,1:u] en de rijvector y gegeven in array a[j:j,1:u] worden vervangen door de vectoren cx+sy en cy–sx.

3.2.3. Lineaire stelsels en matrix inversie (Hoofdstuk 21).

3.2.3.1. Driehoeksontbinding met rijverwisseling (Sectie 210).

3.2.3.1.1. comment mca 2100;
real procedure det(a,n,p); value n; integer n; array a;
integer array p;
de afmetingen van de arrays zijn a[1:n,1:n] en p[1:n].

3.2.3.1.2. comment mca 2101;

```
procedure sol(a,n,p,b); value n; integer n; array a,b;
integer array p;
de afmetingen van de arrays zijn a[1:n,1:n], p[1:n] en b[1:n].
```

3.2.3.1.3.
```
comment mca 2102;
real procedure detsol(a,n,b); value n; integer n; array a,b;
de afmetingen van de arrays zijn a[1:n,1:n] en b[1:n].
```

3.2.3.1.4.
```
comment mca 2103;
procedure inv(a,n,p); value n; integer n; array a;
integer array p;
de afmetingen van de arrays zijn a[1:n,1:n] en p[1:n].
```

3.2.3.1.5.
```
comment mca 2104;
real procedure detinv(a,n); value n; integer n; array a;
de afmetingen van het array zijn a[1:n,1:n].
```

3.2.3.2. Eliminatie met rij- en kolomverwisselingen (Sectie 211).

3.2.3.2.1.
```
comment mca 2110;
integer procedure rnkelm(a,n,m,aux,ri,ci); value n,m;
integer n,m; array a,aux; integer array ri,ci;
de afmetingen van de arrays zijn a[1:n,1:m], aux[0:3] en
ri,ci[1:n].
```

3.2.3.2.2.
```
comment mca 2111;
procedure solelm(a,n,ri,ci,b); value n; integer n; array a,b;
integer array ri,ci;
de afmetingen van de arrays zijn a[1:n,1:n], ri,ci[1:n] en
b[1:n].
```

3.2.3.2.3.
```
comment mca 2112;
integer procedure rnksolelm(a,n,aux,b); value n;
integer n; array a,aux,b;
de afmetingen van de arrays zijn a[1:n,1:n], aux[0:3] en
b[1:n].
```

3.2.3.2.4.
```
comment mca 2113;
procedure solhom(a,rank,m,k,ci,x); value rank,m,k;
integer rank,m,k; array a,x; integer array ci;
de afmetingen van de arrays zijn a[1:rank,1:m], ci[1:rank] en
x[1:m].
```

3.2.3.2.5.
```
comment mca 2114;
real procedure invelm(a,n,aux); value n; integer n;
array a,aux;
de afmetingen van de arrays zijn a[1:n,1:n] en aux[0:1].
```

3.2.3.3. Bandmatrices (Sectie 212).

3.2.3.3.1.
```
comment mca 2120;
real procedure detbnd(a,n,lw,rw,aux,m,p); value n,lw,rw;
integer n,lw,rw; integer array p; array a,aux,m;
de afmetingen van de arrays zijn
a[1: (lw+rw) × (n-1) + n],aux[0:2],m[1:lw × (n-2) + 1] en
p[1:n].
```

3.2.3.3.2. comment mca 2121;
procedure solbnd(a,n,lw,rw,n,p,b); value n,lw,rw;
integer n,lw,rw; integer array p; array a,b,m;
de afmetingen van de arrays zijn
a[1: (lw+rw) × (n–1) + n],m[1:lw × (n–2) + 1],p[1:n] en
b[1:n].

3.2.3.3.3. comment mca 2122;
real procedure detsolbnd(a,n,lw,rw,aux,b); value n,lw,rw;
integer n,lw,rw; array a,b,aux;
de afmetingen van de arrays zijn
a[1: (lw+rw) × (n–1) + n],aux[0:2] en b[1:n].

3.2.4. Positief definiete symmetrische lineaire stelsels en matrixinversie (Hoofdstuk 22).

3.2.4.1. Cholesky–ontbinding zonder verwisselingen (Sectie 220).

3.2.4.1.1. comment mca 2200;
real procedure detsym2(a,n); value n; integer n; array a;
de afmetingen van het array zijn a[1:n,1:n].

3.2.4.1.2. comment mca 2201;
procedure solsym2(a,n,b); value n; integer n; array a,b;
de afmetingen van de arrays zijn a[1:n,1:n] en b[1:n].

3.2.4.1.3. comment mca 2202;
real procedure detsolsym2(a,n,b); value n; integer n; array a,b;
de afmetingen van de arrays zijn a[1:n,1:n] en b[1:n].

3.2.4.1.4. comment mca 2203;
procedure invsym2(a,n); value n; integer n; array a;
de afmetingen van het array zijn a[1:n,1:n].

3.2.4.1.5. comment mca 2204;
real procedure detinvsym2(a,n); value n; integer n; array a;
de afmetingen van het array zijn a[1:n,1:n].

3.2.4.1.6. comment mca 2205;
real procedure detsym1(a,n); value n; integer n; array a;
de afmeting van het array is a[1: (n+1) × n : 2].

3.2.4.1.7. comment mca 2206;
procedure solsym1(a,n,b); value n; integer n; array a,b;
de afmetingen van de arrays zijn a[1: (n+1)×n:2] en b[1:n].

3.2.4.1.8. comment mca 2207;
real procedure detsolsym1(a,n,b); value n; integer n; array a,b;
de afmetingen van de arrays zijn a[1: (n+1)×n:2] en b[1:n].

3.2.4.1.9. comment mca 2208;
procedure invsym1(a,n); value n; integer n; array a;
de afmeting van het array is a[1: (n+1) × n : 2].

3.2.4.1.10. comment mca 2209;
real procedure detinvsym1(a,n); value n; integer n; array a;
de afmeting van het array is a[1: (n+1) × n : 2].

3.2.4.2. Cholesky-ontbinding met verwisselingen (Sectie 221).

3.2.4.2.1. comment mca 2210;
integer procedure rnksym20(a,n,p,aux); value n; integer n;
array a,aux; integer array p;
de afmetingen van de arrays zijn a[1:n,1:n], p[1:n] en aux[0:3].

3.2.4.2.2. comment mca 2211;
procedure solsym20(a,n,p,b); value n; integer n;
integer array p; array a,b;
de afmetingen van de arrays zijn a[1:n,1:n], p[1:n] en b[1:n].

3.2.4.2.3. comment mca 2212;
integer procedure rnksolsym20(a,n,b,aux); value n; integer n;
array a,b,aux;
de afmetingen van de arrays zijn a[1:n,1:n], b[1:n] en aux[0:3].

3.2.4.2.4. comment mca 2213;
procedure invsym20(a,n,p); value n; integer n;
integer array p; array a;
de afmetingen van de arrays zijn a[1:n,1:n] en p[1:n].

3.2.4.2.5. comment mca 2214;
integer procedure rnkinvsym20(a,n,aux); value n;
integer n; array a,aux;
de afmetingen van de arrays zijn a[1:n,1:n] en aux[0:3].

3.2.4.2.6. comment mca 221a;
integer procedure solsymhom20(a,n,aux); value n; integer n;
array a,aux;
de afmetingen van de arrays zijn a[1:n,1:n] en aux[0:3].

3.2.4.2.7. comment mca 2215;
integer procedure rnksym10(a,n,p,aux); value n;
integer n; integer array p; array a,aux;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2],p[1:n] en aux[0:3].

3.2.4.2.8. comment mca 2216;
procedure solsym10(a,n,p,b); value n; integer n;
array a,b; integer array p;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2],p[1:n] en b[1:n].

3.2.4.2.9. comment mca 2217;
integer procedure rnksolsym10(a,n,b,aux);
value n; integer n; array a,b,aux;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2],b[1:n] en aux[0:3].

3.2.4.2.10. comment mca 2218;
procedure invsym10(a,n,p); value n; integer n; array a;
integer array p;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2] en p[1:n].

3.2.4.2.11. comment mca 2219;
integer procedure rnkinvsym10(a,n,aux); value n; integer n;
array a,aux;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2] en aux[0:3].

3.2.4.3. Cholesky–ontbinding voor bandmatrices (Sectie 222).

3.2.4.3.1. comment mca 2220;
real procedure detsymbnd(a,n,w); value n,w; integer n,w; array a;
de afmetingen van het array zijn a[1: (n–1) × w + n].

3.2.4.3.2. comment mca 2221;
procedure solsymbnd(a,n,w,b); value n,w; integer n,w; array a,b;
de afmetingen van de arrays zijn a[1: (n–1) × w+n] en b[1:n].

3.2.4.3.3. comment mca 2222;
real procedure detsolsymbnd(a,n,w,b); value n,w; integer n,w;
array a,b;
de afmetingen van de arrays zijn a[1: (n–1) × w + n] en b[1:n].

3.2.4.4. Kleinste–kwadratenproblemen (Sectie 224).

3.2.4.4.1. comment mca 2240;
integer procedure lsqdec(a,n,m,aux,aid,ci); value n,m;
integer n,m; array a,aux,aid; integer array ci;
de afmetingen van de arrays zijn a[1:n,1:m], aux[0:2], aid[1:m] en ci[1:m] .

3.2.4.4.2. comment mca 2241;
procedure lsqsol(a,n,m,aid,ci,b); value n,m; integer n,m;
array a,aid,b; integer array ci;
de afmetingen van de arrays zijn a[1:n,1:m], aid[1:m], ci[1:m] en b[1:n].

3.2.4.4.3. comment mca 2242;
procedure lsqdglinv(a,m,aid,ci,diag); value m; integer m;
array a,aid,diag; integer array ci;
de afmetingen van de arrays zijn a[1:m,1:m], aid[1:m], ci[1:m] en diag[1:m].

3.2.4.4.4. comment mca 2243;
integer procedure lsqdecsol(a,n,m,aux,diag,b);
value n,m; integer n,m; array a,aux,diag,b;
de afmetingen van de arrays zijn a[1:n,1:m], aux[0:2], diag[1:m] en b[1:n].

3.2.5. Eigensystemen van reele symmetrische matrices (Hoofdstuk 23).

3.2.5.1. Transformatie van Householder (Sectie 230).

3.2.5.1.1. comment mca 2300;
procedure tfmsymtri2(a,n,d,b,bb,em); value n; integer n;
array a,d,b,bb,em;
de afmetingen van de arrays zijn a[1:n,1:n], d,b,bb[1:n],
em[0:1].

3.2.5.1.2. comment mca 2301;
procedure baksymtri2(a,n,n1,n2,vec); value n,n1,n2;
integer n,n1,n2; array a,vec;
de afmetingen van de arrays zijn a[1:n,1:n], vec[1:n,n1:n2].

3.2.5.1.3. comment mca 2302;
procedure tfmprevec(a,n); value n; integer n; array a;
de afmetingen van het array zijn a[1:n,1:n].

3.2.5.1.4. comment mca 2305;
procedure tfmsymtri1(a,n,d,b,bb,em); value n; integer n;
array a,d,b,bb,em;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2],
d,b,bb[1:n], em[0:1].

3.2.5.1.5. comment mca 2306;
procedure baksymtri1(a,n,n1,n2,vec); value n,n1,n2;
integer n,n1,n2; array a,vec;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2],
vec[1:n,n1:n2].

3.2.5.2. Lineaire interpolatie en inverse iteratie (Sectie 231).

3.2.5.2.1. comment mca 2310;
boolean procedure zeroin(x,y,fx,tolx); real x,y,fx,tolx;

3.2.5.2.2. comment mca 2311;
procedure valsymtri(d,bb,n,n1,n2,val,em); value n,n1,n2;
integer n,n1,n2; array d,bb,val,em;
de afmetingen van de arrays zijn d,bb[1:n], val[n1:n2],
em[0:3].

3.2.5.2.3. comment mca 2312;
procedure vecsymtri(d,b,n,n1,n2,val,vec,em); value n,n1,n2;
integer n,n1,n2; array d,b,val,vec,em;
de afmetingen van de arrays zijn d,b[1:n], val[1:n2],
vec[1:n,1:n2], em[0:9].

3.2.5.2.4. comment mca 2313;
procedure eigvalsym2(a,n,numval,val,em); value n,numval;
integer n,numval; array a,val,em;
de afmetingen van de arrays zijn a[1:n,1:n], val[1:numval],
em[0:3].

3.2.5.2.5. comment mca 2314;
procedure eigsym2(a,n,numval,val,vec,em); value n,numval;

integer n,numval; array a,val,vec,em;
de afmetingen van de arrays zijn a[1:n,1:n], val[1:numval], vec[1:n,1:numval], em[0:9].

3.2.5.2.6. comment mca 2318;
procedure eigvalsym1(a,n,numval,val,em); value n,numval;
integer n,numval; array a,val,em;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2], val[1:numval], em[0:3].

3.2.5.2.7. comment mca 2319;
procedure eigsym1(a,n,numval,val,vec,em); value n,numval;
integer n,numval; array a,val,vec,em;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2], val[1:numval], vec[1:n,1:numval], em[0:9].

3.2.5.3. QR–iteratie (Sectie 232).

3.2.5.3.1. comment mca 2320;
integer procedure qrivalsymtri(d,bb,n,em); value n; integer n;
array d,bb,em;
de afmetingen van de arrays zijn d,bb[1:n], em[0:5].

3.2.5.3.2. comment mca 2321;
integer procedure qrisymtri(a,n,d,b,bb,em); value n; integer n;
array a,d,b,bb,em;
de afmetingen van de arrays zijn a[1:n,1:n], d,b,bb[1:n], em[1:5].

3.2.5.3.3. comment mca 2322;
integer procedure qrivalsym2(a,n,val,em); value n; integer n;
array a,val,em;
de afmetingen van de arrays zijn a[1:n,1:n], val[1:n], em[0:5].

3.2.5.3.4. comment mca 2323;
integer procedure qrisym(a,n,val,em); value n; integer n;
array a,val,em;
de afmetingen van de arrays zijn a[1:n,1:n], val[1:n], em[0:5].

3.2.5.3.5. comment mca 2327;
integer procedure qrivalsym1(a,n,val,em); value n; integer n;
array a,val,em;
de afmetingen van de arrays zijn a[1: (n+1) × n : 2], val[1:n], em[0:5].

3.2.6. Eigensystemen van reele matrices (Hoofdstuk 24).

3.2.6.1. Transformatie van Wilkinson en equilibratie (Sectie 240).

3.2.6.1.1. comment mca 2400;
procedure tfmreahes(a,n,em,int); value n; integer n;
array a,em; integer array int;

de afmetingen van de arrays zijn a[1:n,1:n], em[0:1], int[1:n].

3.2.6.1.2. comment mca 2401;
procedure bakreahes1(a,n,int,v); value n; integer n; array a,v;
integer array int;
de afmetingen van de arrays zijn a[1:n,1:n], v[1:n], int[1:n].

3.2.6.1.3. comment mca 2402;
procedure bakreahes2(a,n,n1,n2,int,vec); value n,n1,n2;
integer n,n1,n2; array a,vec; integer array int;
de afmetingen van de arrays zijn a[1:n,1:n], vec[1:n,n1:n2], int[1:n].

3.2.6.1.4. comment mca 2405;
procedure eqilbr(a,n,em,d,int); value n; integer n;
array a,em,d; integer array int;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:0], d[1:n], int[1:n].

3.2.6.1.5. comment mca 2406;
procedure baklbr(n,n1,n2,d,int,vec); value n,n1,n2;
integer n,n1,n2; array d,vec; integer array int;
de afmetingen van de arrays zijn d[1:n], vec[1:n,n1:n2], int[1:n].

3.2.6.2. Enkele QR-iteratie (Sectie 241).

3.2.6.2.1. comment mca 2410;
integer procedure reavalqri(a,n,em,val); value n; integer n;
array a,em,val;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:5], val[1:n].

3.2.6.2.2. comment mca 2411;
procedure reaveches(a,n,lambda,em,v); value n,lambda;
integer n; real lambda; array a,em,v;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:9], v[1:n].

3.2.6.2.3. comment mca 2412;
integer procedure reaeigval(a,n,em,val); value n; integer n;
array a,em,val;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:5], val[1:n].

3.2.6.2.4. comment mca 2413;
procedure reascl(a,n,n1,n2); value n,n1,n2; integer n,n1,n2;
array a;
de afmetingen van het array zijn a[1:n,n1:n2].

3.2.6.2.5. comment mca 2414;
integer procedure reaeig1(a,n,em,vec); value n; integer n;
array a,em,val,vec;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:9], val[1:n], vec[1:n,1:n].

3.2.6.2.6. comment mca 2415;
integer procedure reaeig2(a,n,em,val); value n; integer n;
array a,em,val;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:9], val[1:n];
bovendien worden de eerste 4 × n ↑ 2 plaatsen van de trommel gebruikt.

3.2.6.2.7. comment mca 2416;
integer procedure reaqri(a,n,em,val,vec); value n; integer n;
array a,em,val,vec;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:5], val[1:n], vec[1:n,1:n].

3.2.6.2.8. comment mca 2417;
integer procedure reaeig3(a,n,em,val,vec); value n; integer n;
array a,em,val,vec;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:5], val[1:n], vec[1:n,1:n].

3.2.6.3. Dubbele QR-iteratie (Sectie 242).

3.2.6.3.1. comment mca 2420;
integer procedure comvalqri(a,n,em,re,im); value n; integer n;
array a,em,re,im;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:5], re,im[1:n].

3.2.6.3.2. comment mca 2421;
procedure comveches(a,n,lambda,mu,em,u,v); value n,lambda,mu;
integer n; real lambda,mu; array a,em,u,v;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:9], u,v[1:n].

3.2.6.3.3. comment mca 2422;
integer procedure comeigval(a,n,em,re,im); value n; integer n;
array a,em,re,im;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:5], re,im[1:n].

3.2.6.3.4. comment mca 2423;
procedure comscl(a,n,n1,n2,im); value n,n1,n2; integer n,n1,n2;
array a,im;
de afmetingen van de arrays zijn a[1:n,n1:n2], im[n1:n2].

3.2.6.3.5. comment mca 2424;
integer procedure comeig1(a,n,em,re,im,vec); value n; integer n;
array a,em,re,im,vec;
de afmetingen van de arrays zijn a[1:n,1:n], em[0:9], re,im[1:n], vec[1:n,1:n].

3.2.6.3.6. comment mca 2425;
integer procedure comeig2(a,n,em,re,im); value n; integer n;
array a,em,re,im;

de afmetingen van de arrays zijn a[1:n,1:n], em[0:9], re,im[1:n];
bovendien worden de eerste 4 × n ↑ 2 plaatsen van de trommel gebruikt.

3.3. Conversie procedures.

De volgende procedures dienen om numerieke gegevens om te zetten in rijen symbolen en rijen symbolen in numerieke gegevens; de symbolen worden hierbij weergegeven in hun interne representatie.

3.3.1. integer procedure stringsymbol(k, text); value k;
integer k; string text;
De function designator stringsymbol levert de resym–waarde van het k–de symbool uit de string text af, waarbij k begint te tellen bij 0. De componenten van samengestelde symbolen worden hierbij als afzonderlijke symbolen opgevat; dit geldt ook voor eventueel binnen in text voorkomende string–quotes.
Voor k < 0 ∨ k > (aantal symbolen uit de string) krijgt stringsymbol de waarde 255.
In plaats van stringsymbol mag ook de naam STRINGSYMBOL gebruikt worden.

3.3.2. real procedure read1(intexpr, intvar); integer intexpr, intvar;
De procedure read1 scant een symbolenrij die hij ontleent aan zijn parameters tot hij daaruit een getal en vervolgens een getalscheider heeft geisoleerd. De waarde van het getal wordt als functiewaarde afgeleverd, de interne representatie van de getalscheider wordt aan de parameter intvar toegekend.

De successieve symbolen uit de symbolenrij verkrijgt read1 door:
a) voor het eerste symbool de waarde van de parameter intvar te nemen,
b) voor elk volgend symbool de parameter intexpr (opnieuw) te evalueren.
De zo verkregen waarden interpreteert read1 telkens als de interne representatie van een symbool (zie tabel 6.1.).

Een rij symbolen vormt een getal wanneer ze voldoet aan de definitie van number in het Revised Report, waarbij bovendien als lay–out–symbolen spaties zijn toegestaan, en wel:
a) na het teken van het getal, of na het symbool "₁₀", of na het teken van de exponent: willekeurig veel spaties,
b) na een cijfer of na de decimale punt: een enkele spatie.
Een rij symbolen vormt de aanhef van een getal als ze ofwel zelf een number is ofwel tot een number kan worden uitgebreid door er een cijfer achter te zetten.
read1 zoekt in de symbolenrij het eerste cijfer op. read1 skipt daarbij in principe alle niet–cijfers en bovendien elke symbolenrij tussen twee apostrofs (dus ook eventuele cijfers tussen apostrofs). Echter worden alle symbolen die zodanig aan het eerste cijfer voorafgaan dat ze met dat cijfer de aanhef van een getal vormen bewaard en in het te isoleren getal

verdisconteerd (bijvoorbeeld "−₁₀+" of "+."). Hierna bewaart read1 alle symbolen die op het eerste cijfer volgen en met de eerder bewaarde symbolen de aanhef van een getal vormen. Het eerste symbool dat niet meer met de voorafgaande bewaarde symbolen de aanhef van een getal vormt wordt opgevat als getalscheider en aan de parameter intvar geassigneerd. Dit symbool is tevens het laatste door read1 gescande symbool.
Indien vervolgens de bewaarde symbolenrij, die in ieder geval de aanhef van een getal vormt, ook werkelijk een number voorstelt, wordt de waarde daarvan als functiewaarde van read1 afgeleverd. Anders wordt de executie van het programma afgebroken met een foutmelding met foutnummer 517. Dit laatste gebeurt bijvoorbeeld bij de symbolenrijen "abc+3.d" of "ab'j5:=3'cd3₁₀−.".
De door read1 afgeleverde waarde stelt het geïsoleerde getal voor met een relatieve precisie van ongeveer 12 decimalen en ligt in absolute waarde tussen de grenzen ₁₀−616 en ₁₀+628 of is exact 0. De waarde van integers, in absolute waarde kleiner dan 1 099 511 627 776 wordt door read1 exact afgeleverd.
Indien de evaluatie van intvar of intexpr een getal k oplevert, dat geen resym-waarde is, wordt k beschouwd als de interne representatie van een onbenoemd symbool dat niet in de aanhef van een getal voor kan komen. Als dit onbenoemde symbool aanleiding is de aanhef van het getal als beëindigd te beschouwen, wordt k geassigneerd aan intvar.
De actuele parameter die correspondeert met intvar mag een integer of real variabele zijn, of een element van een integer array of van een real array.

Ter illustratie het volgende programma:

```
begin   integer i,s; real x;
        integer procedure r;
        begin   integer k; r:= k:= resym; prsym(k) end;
        s:= ↑19;
        for i:= 1 while s ≠ 122 do
        begin   x:= read1(r,s); tab; fixt(3,1,x); nlcr end
end
```

met op het invoermedium de volgende symbolen:
.3.5.7₁₀1₁₀2+2−3a?−5.7?−3c

Over de regeldrukker worden nu de volgende symbolen afgedrukt:

.3.	+.3
5.	+.5
7₁₀1₁₀	+7.0
2+	+100.0
2−	+2.0
3a	−3.0
?−5.7?	−5.7

3.3.3. Boolean procedure fix(n,m,x,a); value n,m,x;
integer n,m; real x; integer array a;

Behoudens in het hieronder genoemde uitzonderingsgeval wordt door de procedure fix de waarde van x exact afgerond op de m-de decimaal en vervolgens geconverteerd tot een symbolenrij die die waarde in vaste-komma-representatie met n cijfers voor de decimale punt en m cijfers erna voorstelt.
Deze symbolenrij bestaat uit de volgende symbolen:

a) het teken van x en het gehele gedeelte van x in n+1 symbolen. In het gehele gedeelte van x worden non-significante nullen door spaties vervangen, uitgezonder de nul op de eenhedenpositie in het geval m = 0 (dwz. als het breukgedeelte ontbreekt). Het teken van x gaat direct vooraf aan het eerste cijfer, zodat eventuele spaties aan het teken voorafgaan.
b) als m > 0 volgt hierop een decimale punt en m cijfers.
c) op het laatste cijfer volgt 1 spatie.

In totaal bestaat de symbolenrij dus uit

if m = 0 then n + 2 else n + m + 3

symbolen.
Het i-de symbool van de zo verkregen symbolenrij wordt in interne representatie afgeleverd in het array-element a[i], voor i lopend van 1 tot en met if m = 0 then n + 2 else n + m + 3. Bijgevolg komt in a[1] de interne respresentatie van het voorste symbool (het teken van x of een spatie) en in a[if m = 0 then n + 2 else n+m+3] de interne respresentatie van het achterste symbool (steeds een spatie).
Als functiewaarde wordt true afgeleverd.
De met a corresponderende actuele parameter moet de identifier zijn van een een-dimensionaal array van type integer (en dus niet real).

Uitzonderingsgeval:
Indien blijkt dat de absolute waarde van x na afronding op de m-de decimaal $\geq 10 \uparrow n$, of indien niet voldaan is aan de relaties $n \geq 0$, $m \geq 0$ en $1 \leq n + m \leq 21$, wordt als functiewaarde false afgeleverd en voorts wordt in de elementen a[1] t/m a[21] dezelfde symbolenrij opgeslagen als bij een aanroep van flo(13,3,x,a) (zie 3.3.4.).

3.3.4. Boolean procedure flo(n,m,x,a); value n,m,x;
integer n,m; real x; integer array a;
Behoudens in het hieronder genoemde uitzonderingsgeval wordt door de procedure flo de waarde van x na afronding op n cijfers geconverteerd tot een symbolenrij die die waarde in floating-point-representatie voorstelt met een mantisse van n cijfers en een decimale exponent van m cijfers.
Deze symbolenrij bestaat uit de volgende symbolen:

a) het teken van x en de decimale punt.
b) de n cijfers van de mantisse.
c) het symbool "₁₀".
d) het teken van de decimale exponent.
e) de absolute waarde van die exponent in m cijfers, waarbij nonsignificante nullen vervangen worden door spaties behalve op de eenhedenpositie.
f) een spatie.

Voor x= 0 bestaat de mantisse uit n nullen en de exponent uit m − 1 spaties en een nul. Voor x ≠ 0 wordt de decimale exponent zo bepaald dat de mantisse in absolute waarde ≥ .1 en < 1 is. In totaal bestaat de symbolenrij steeds uit n + m + 5 symbolen.
Het i-de symbool van de zo verkregen symbolenrij wordt in interne representatie afgeleverd in het array-element a[i], voor i lopend van 1 tot en met n + m + 5. Bijgevolg komt in a[1] de interne representatie van het teken van x en in a[n + m + 5] steeds de interne representatie van een spatie.
Als functiewaarde wordt true afgeleverd.
De met a corresponderende actuele parameter moet de identifier zijn van een een-dimensionaal array van type integer (en dus niet real).

Uitzonderingsgeval:
Indien x ≠ 0 en de decimale exponent niet in m cijfers voorgesteld kan worden, of als niet voldaan is aan de relaties 1 ≤ n ≤ 13 en 1 ≤ m ≤ 3, wordt als functiewaarde false afgeleverd en voorts wordt in a[1] t/m a[21] dezelfde symbolenrij opgeslagen als voor het geval n = 13, m = 3.

3.4. Diverse procedures.

3.4.1. integer procedure even(n); value n; integer n;
even:= if n : 2 × 2 = n then 1 else −1;
De function designator even is ekwivalent met (−1) ↑ n.
In plaats van even mag ook de naam EVEN gebruikt worden.

3.4.2. integer procedure remainder(a,b); value a,b; integer a,b;
remainder:= if b = 0 then a else a − a : b × b;
De function designator remainder levert, tenzij b = 0, de rest af van de deling van a door b (de rest gedefinieerd als het in absolute waarde kleinste getal r met hetzelfde teken als a, waarvoor a = r (mod b) geldt).
In plaats van remainder mag ook de naam REMAINDER gebruikt worden.

3.4.3. real procedure random;
Opeenvolgende aanroepen van random geven min of meer homogeen verdeelde trekkingen uit het eenheidsinterval [0,1). Iets nauwkeuriger omschreven, iedere aanroep van random levert een waarde, ≥ 0 en < 1, gelijk aan het eerstvolgende getal uit een pseudo-random rij, gegenereerd volgens een proces van D.H.Lehmer (zie b.v. M.Greenberger, MTAC 15 (1961) 383). De periode van dit proces is 2 ↑ 26.
Indien aan de eerste aanroep van random in het programma geen aanroep van setrandom (zie 3.4.4.) is voorafgegaan, wordt als startwaarde van het proces de waarde .5 gebruikt.
In plaats van random mag ook de naam RANDOM gebruikt worden.

3.4.4. procedure setrandom(x); value x; real x;

De procedure setrandom definieert met behulp van x de startwaarde voor het proces, waarmee de function designator random zijn pseudo–random trekkingen genereert. Het argument x van setrandom moet voldoen aan $-(1 - 2 \uparrow (-27)) < x < +(1 - 2 \uparrow (-27))$. Als x buiten het toegestane gebied ligt, wordt het programma afgebroken met een foutmelding met foutnummer 510.
In plaats van setrandom mag ook de naam SETRANDOM gebruikt worden.

3.4.5. procedure exit;
De procedure exit beeindigt de executiefase van het programma, alsof de laatste end van het programma "gepasseerd" was.
In plaats van exit mag ook de naam EXIT gebruikt worden.

3.4.6. real procedure time;
De function designator time levert de tijd af, die verlopen is sinds het in behandeling nemen van het programma (dus vanaf het begin van de syntactische controle), in seconden (dus niet in milli–uren) met een nauwkeurigheid van 0.01 sec.

3.4.7. integer procedure available;
De function designator available levert het aantal af van de op het moment van aanroep beschikbare geheugenplaatsen. Ten aanzien van het begrip 'beschikbaar' moet het volgende opgemerkt worden. Weliswaar zijn er op het moment van aanroep inderdaad "available" geheugenplaatsen beschikbaar, maar dit wil niet zeggen dat men deze volledig voor array's kan gebruiken. Bij het vervolg van de executie van het programma kan namelijk een gedeelte van deze ruimte nodig zijn als werkruimte, ontoegankelijk voor het programma. Een ALGOL 60 programma heeft tijdens zijn uitvoering werkruimte nodig voor:

a) het opbergen van tussenresultaten bij het uitwerken van expressies.
b) het indiceren van array–elementen.
c) het aanroepen van standaardprocedures.
d) het bijhouden van administratie bij block–ingangen, procedure–aanroepen en bijbehorende parameter–overdrachten.
e) het reserveren van ruimte voor variabelen in procedure–bodies.
Voor de punten a, b en c zal in vele gevallen een bedrag van 50 geheugenplaatsen voldoende zijn. Voor punt d zal een bedrag van 20 geheugenplaatsen per binnengegane maar nog niet verlaten procedure, vermeerderd met 4 geheugenplaatsen per parameter, veelal voldoende zijn. Voor punt e zij verwezen naar hoofdstuk 2.. Bij het gebruik van recursieve procedures kunnen de punten d en e grote hoeveelheden geheugen opeisen.

3.4.8. integer procedure date;
Deze procedure levert de datum af. Nauwkeuriger gezegd, de waarde van de function designator date is:

$$(\text{dag} \times 100 + \text{maand}) \times 100 + \text{jaar} - 1900$$

Het is dezelfde integer als die welke uiterst links in het kopje boven aan elke regeldrukker-pagina wordt afgedrukt.

3.4.9. integer procedure real time;
Deze procedure levert de kloktijd af. Nauwkeuriger gezegd, de waarde van de function designator real time is:
(uren × 100 + minuten) × 100 + seconden

3.4.10. procedure process inquiry(users program);
procedure users program;
Voor een beschrijving van deze procedure wordt verwezen naar:

P. Beertema en G.J.R. Forch,
Handleiding bij het MC-enqueteprogramma, LR 2.1.

3.5. Bitmanipulatie procedures.

Deze procedures geven de programmeur de mogelijkheid te manipuleren met rijen van 27 bits. Een dergelijke bitrij (d[26],d[25],...,d[0]), waarbij d[i] = 0 of 1 voor $26 \geq i \geq 0$, wordt geidentificeerd met een integer d met de waarde $d = \mathrm{sum}(i, 0, 25, (d[i] - d[26]) \times 2 \uparrow i)$.
Zie voor een uitgebreide beschrijving LR1.2.

3.5.1. integer procedure and (a, b); value a, b; integer a, b;
Bitsgewijze logische $\wedge$.

3.5.2. integer procedure or (a, b); value a, b; integer a, b;
Bitsgewijze logische $\vee$.

3.5.3. integer procedure xor (a, b); value a, b; integer a, b;
Bitsgewijze 'exclusieve of'.

3.5.4. integer procedure bit (j, a); value j, a; integer j, a;
bit krijgt de waarde van het j-de bit van a (dwz. a[j]).

3.5.5. integer procedure bitstring (u, l, a); value u, l, a;
integer u, l, a;
bitstring krijgt de waarde die correspondeert met de bitrij (0, 0, ..., 0, a[u], a[u-1], ..., a[l+1], a[l]).

3.5.6. integer procedure set (c, u, l, a); value c, u, l, a;
integer c, u, l, a;
set krijgt de waarde die correspondeert met de bitrij (a[26],a[25],...,a[u+1],c[u-1],...,c[1],c[0],a[l-1],...,a[0]).

3.5.7. integer procedure clear shift (a, j); value a, j; integer a, j;
Als $j \geq 0$ krijgt clear shift de waarde die correspondeert met de bitrij (a[26-j], a[25-j], ..., a[1], a[0], 0, 0, ..., 0).
Als $j < 0$ krijgt clear shift de waarde die correspondeert met de bitrij (0, ..., 0, a[26], a[25], ..., a[|j|+1], a[|j|]).

3.5.8. integer procedure circ shift (a, j); value a, j; integer a, j;
Als j ≥ 0 krijgt circ shift de waarde die correspondeert met de bitrij (a[26–j],a[25–j], ..., a[1],a[0],a[26],a[25], ..., a[26–j+1]).
Als j < 0 krijgt circ shift de waarde die correspondeert met de bitrij (a[|j|–1], a[|j|–2], ..., a[0], a[26], ..., a[|j|+1], a[|j|]).

3.5.9. integer procedure norm shift (a, normed a); value a;
integer a, normed a;
Onder 'normeren' verstaan we naar links rondschuiven (dwz. a:= circ shift (a, 1) uitvoeren) totdat bit (26, a) ≠ bit (25, a). norm shift krijgt als waarde het aantal plaatsen waarover geschoven moet worden om dit te bereiken. Wanneer a[i] = a[26] voor 0 ≤ i ≤ 25, krijgt norm shift de waarde 26.
normed a is een output–parameter waarin de waarde afgeleverd wordt die a na normering zou hebben.

Hierna volgen nog drie procedures die betrekking hebben op real getallen. Een real getal in de EL X8 beslaat de ruimte van twee opeenvolgende integers, kop en staart genoemd.

3.5.10. integer procedure head of (x); value x; real x;
head of krijgt de waarde van de kop van x.

3.5.11. integer procedure tail of (x); value x; real x;
tail of krijgt de waarde van de staart van x.

3.5.12. real procedure compose (a, b); value a, b; integer a, b;
compose krijgt de waarde van het real getal waarvan a de kop en b de staart is.

3.6. Sorteerprocedures.

Deze procedures kunnen, wat betreft de werking, in 4 typen verdeeld worden:
a) Het type qsort: een rij elementen a[1],...,a[u] wordt gesorteerd in niet dalende volgorde.
b) Het type indqsort; de naar (een deelrij van) a[1 : u] verwijzende rij indices ind[li : ui] ($1 \leq ind[k] \leq u$ voor alle k met $li \leq k \leq ui$) wordt zo gewijzigd dat de elementen a[ind[li]],...,a[ind[ui]] in niet dalende volgorde staan. a[1 : u] blijft ongewijzigd.
c) Het type perm; de elementen a[1 : u] worden gepermuteerd in overeenstemming met de permutatie van de getallen 1 (1) u zoals die gegeven is in het array ind[1 : u].
d) Het type rank tie; de rij a[1 : u] blijft onveranderd, maar dezelfde permutatie van de indices 1 (1) u wordt gegenereerd als bij indqsort. Tevens wordt in rnk[1 : u] het (gemiddelde) rangnummer van de elementen van a afgeleverd. Dat wil zeggen: rnk[k]= (s(k) + t(k))/2 voor k is 1 (1) u waarbij s(k) is 1 + het aantal elementen dat kleiner dan a[k], en t(k) het aantal dat kleiner dan of gelijk a[k] is. Procedures van het type ranktie leveren in ties2 de som van de kwadraten van de lengten van de knopen en in ties3 de som van de derdemachten van die lengten af. Hierbij is een knoop gedefinieerd als bestaande uit een of meer elementen van gelijke waarde.

In het bovenstaande is steeds sprake van een rij elementen a[1 : u]. Deze rij kan behalve als vector ook op vier andere manieren gegeven worden. In totaal levert dit $5 \times 4 = 20$ procedures op. De vijf manieren om die rij te geven zijn:
1) Als vector(eendimensionaal array). De namen van de hierbij behorende procedures zijn samengesteld uit de prefix vec gevolgd door het type bewerking dus: vec qsort,vec indqsort,vec perm en vec ranktie.
2) Als rij van een matrix.De prefix is nu row dus: row qsort, row indqsort, row perm en row ranktie.
3) Als kolom van een matrix.De prefix is col.

Bij de volgende twee gevallen gaat het niet om het sorteren van een rij getallen, maar om het lexicografisch ordenen(per kolom of per rij) van een matrix.
Voorbeeld: Het per rij lexicografisch ordenen van de matrix

02	07	12	02		01	05	12	03
01	06	13	02		01	06	13	02
04	10	07	42	transformeert	02	07	12	01
02	08	14	01	deze in	02	07	12	02
01	05	12	03		02	08	14	01
02	07	12	01		04	10	07	42

4) Lexicografisch per rij. De prefix is r mat.
5) Lexicografisch per kolom. De prefix is c mat.

3.6.1. procedure vec qsort(a,lv,uv); value lv,uv;

integer lv,uv; array a;
a wordt niet dalend geordend.

3.6.2. procedure vec indqsort(a,ind,lvi,uvi); value lvi,uvi;
integer lvi,uvi; integer array ind; array a;
ind wordt zo gewijzigd dat a[ind[lvi]],...,a[ind[uvi]] in niet dalende volgorde staan. Voorafgaande aan de procedureaanroep moet ind gevuld zijn.

3.6.3. procedure vec perm(ind,lv,uv,a); value lv,uv;
integer lv,uv; integer array ind; array a;
De permutatie die in ind staat wordt op a toegepast.

3.6.4. procedure vec ranktie(a,lv,uv,ind,rnk,ties2,ties3);
value lv,uv; integer lv,uv; real ties2,ties3;
integer array ind; array a,rnk;
ind wordt zo gewijzigd dat a[ind[lv]],...,a[ind[uv]] in niet dalende volgorde staan.In rnk komen de rangnummers van de elementen van a te staan, ties2 wordtgelijk aan de som van de kwadraten van de lengten der knopen en ties3 wordt gelijk aan de som van de derdemachten van die lengten.

De volgende procedure valt buiten het schema uit 3.6. 3.6.5.
procedure vec2 qsort(a1,a2,lv,uv); value lv,uv;
integer lv,uv; array a1,a2;
De paren getallen a1[j],a2[j] (lv $\leq$ j $\leq$ uv) worden zo geordend dat de elementen van a1 in niet dalende volgorde staan.

3.6.6. procedure row qsort(b,r,lc,uc); value r,lc,uc;
integer r,lc,uc; array b;
De rij b[r,lc],...,b[r,uc] wordt niet dalend geordend.

3.6.7. procedure row indqsort(b,r,ind,lci,uci); value r,lci,uci;
integer r,lci,uci; integer array ind; array b;
ind wordt zo gewijzigd dat b[r,ind[lci]],...,b[r,ind[uci]] in niet dalende volgorde staan. Voorafgaande aan de aanroep van de procedure moet ind gevuld zijn.

3.6.8. procedure row perm(ind,r,lc,uc,b); value r,lc,uc;
integer r,lc,uc; integer array ind; array b;
De permutatie die in ind staat wordt op b[r,lc],...,b[r,uc] toegepast.

3.6.9. procedure row ranktie(b,r,lc,uc,ind,rnk,ties2,ties3);
value r,lc,uc; integer r,lc,uc; real ties2,ties3;
integer array ind; array b,rnk;
ind wordt zo gewijzigd dat b[r,ind[lc]],...,b[r,ind[uc]] in niet dalende volgorde staan. In rnk komen de rangnummers van b[r,lc],...,b[r,uc] te staan. ties2 wordt gelijk aan de som van de kwadraten van de lengten der knopen en ties3 wordt gelijk aan de som van de derdemachten van die lengten.

3.6.10. procedure col qsort(b,c,lr,ur); value c,lr,ur;
integer c,lr,ur; array b;
De kolom b[lr,c],...,b[ur,c] wordt niet dalend geordend.

3.6.11. procedure col indqsort(b,c,ind,lri,uri); value c,lri,uri;
integer c,lri,uri; integer array ind; array b;
ind wordt zo gewijzigd dat b[ind[lri],c],...,b[ind[uri].c] in niet dalende volgorde staan. Voorafgaande aan de aanroep van deze procedure moet ind gevuld zijn.

3.6.12. procedure col perm(ind,c,lr,ur,b); value c,lr,ur;
integer c,lr,ur; integer array ind; array b;
De permutatie die in ind staat wordt op b[lr,c],...,b[ur,c] toegepast.

3.6.13. procedure col ranktie(b,c,lr,ur,ind,rnk,ties2,ties3);
value c,lr,ur; integer c,lr,ur; real ties2,ties3;
integer array ind; array b,rnk;
ind wordt zo gewijzigd dat b[ind[lr],c],...,b[ind[ur],c] in niet dalende volgorde staan.In rnk komen de rangnummers van de elementen b[lr,c],...,b[ur,c]. ties2 wordt gelijk aan de som van de kwadraten van de lengten der knopen en ties3 wordt gelijk aan de som van de derdemachten van die lengten.

3.6.14. procedure r mat qsort(b,lr,ur,lc,uc); value lr,ur,lc,uc;
integer lr,ur,lc,uc; array b;
De rijen van de matrix b[lr:ur,lc: uc] worden in niet dalende lexicografische volgorde geplaatst.

3.6.15. procedure rmat indqsort(b,ind,lri,uri,lci,uci);
value lri,uri,lci,uci; integer lri,uri,lci,uci;
integer array ind; array b;
ind wordt zo gewijzigd dat de rijen van de submatrix

```
b[ind[lri],lci]   ...   b[ind[lri],uci]
       .                      .
       .                      .
       .                      .
b[ind[uri],lci]   ...   b[ind[uri],uci]
```

in niet dalende lexicografische volgorde staan. Voorafgaande aan de aanroep van de procedure moet ind gevuld zijn.

3.6.16. procedure r mat perm(ind,lr,ur,lc,uc,b); value lr,ur,lc,uc;
integer lr,ur,lc,uc; integer array ind; array b;
De volgorde van de rijen van b[lr:ur,lc:uc] wordt overeenkomstig de permutatie die in ind[lr: ur] staat gewijzigd.

3.6.17. procedure r mat ranktie(b,lr,ur,lc,uc,ind,rnk,ties2,ties3);
value lr,ur,lc,uc; integer lr,ur,lc,uc; real ties2,ties3;
integer array ind; array b,rnk;
ind wordt zo gewijzigd dat de rijen van de matrix

```
b[ind[lr],lc]   ...   b[ind[lr],uc]
       .                      .
       .                      .
       .                      .
b[ind[ur],lc]   ...   b[ind[ur],uc]
```

in niet dalende lexicografische volgorde staan. In rnk komen de rangnummers van de rijen van b[lr: ur,lc: uc]. ties2 wordt gelijk aan de som van de kwadraten van de lengten der knopen en ties3 wordt gelijk aan de som van de derdemachten van die lengten.

3.6.18. procedure c mat qsort(b,lr,ur,lc,uc); value lr,ur,lc,uc;
integer lr,ur,lc,uc; array b;
De kolommen van de matrix b[lr: ur,lc:uc] worden in niet dalende lexicografische volgorde geplaatst.

3.6.19. procedure c mat indqsort(b,ind,lri,uri,lci,uci);
value lri,uri,lci,uci; integer lri,uri,lci,uci;
integer array ind; array b;
ind wordt zo gewijzigd dat de kolommen van de submatrix

```
b[lri,ind[lci]]   ...   b[lri,ind[uci]]
      .                       .
      .                       .
      .                       .
b[uri,ind[lci]]   ...   b[uri,ind[uci]]
```

in niet dalende lexicografische volgorde staan. Voorafgaande aan de aanroep van de procedure moet ind gevuld zijn.

3.6.20. procedure c mat perm(ind,lr,ur,lc,uc,b); value lr,ur,lc,uc;
integer lr,ur,lc,uc; integer array ind; array b;
De volgorde van de kolommen van b[lr : ur,lc : uc] wordt overeenkomstig de permutatie die in ind[lc : uc] staat gewijzigd.

3.6.21. procedure c mat ranktie (b,lr,ur,lc,uc,ind,rnk,ties2,ties3);
value lr,ur,lc,uc; integer lr,ur,lc,uc; real ties2,ties3;
integer array ind; array b,rnk;
ind wordt zo gewijzigd dat de kolommen van de matrix

```
b[lr,ind[lc]]   ...   b[lr,ind[uc]]
      .                     .
      .                     .
      .                     .
b[ur,ind[lc]]   ...   b[ur,ind[uc]]
```

in niet dalende lexicografische volgorde staan. In rnk komen de rangnummers van de kolommen van b[lr : ur,lc : uc]. ties2 wordt gelijk aan de som van de kwadraten van de lengten der knopen en ties3 wordt gelijk aan de som van de derdemachten van die lengten.

3.7. Gebruikersprocedures.

Aan de gebruikers wordt de mogelijkheid geboden om veel gebruikte programma's in procedurevorm in de bibliotheek van het MILLI–systeem op te laten nemen.
Om hiervoor in aanmerking te komen dient een programma een redelijk grote omvang te hebben; bovendien moet te verwachten zijn dat het programma gedurende langere tijd gebruikt zal worden.
Op verzoek worden nadere inlichtingen verstrekt.
Gebruikersprocedures worden niet in deze handleiding opgenomen.

4. Standaardprocedures voor Input/Output.

De in dit hoofdstuk beschreven procedures behoeven niet te worden gedeclareerd.

4.1. Input procedures.

Bij invoer moet hetzelfde medium worden gebruikt als voor het programma, dwz. bij een ponsbandprogramma hoort ponsbandvoer en bij een kaartprogramma hoort kaartvoer. Bij de bandlezer is 8— gats band standaard. Indien nodig kan echter ook 5— of 7— gats band worden gebruikt. De snelheid van de lezer is ongeveer 1000 symbolen per seconde. Kaartinvoer geschiedt op 80— koloms ponskaarten met een snelheid van ongeveer 20 kaarten per seconde.

4.1.1. integer procedure rehep;

4.1.1.1. ponsbandinvoer.

De waarde van de function designator rehep is gelijk aan de getalwaarde van de eerstvolgende ponsing (pentade bij 5— gats band, heptade bij 7— gats band, octade bij 8— gats band) op de band. Deze getalwaarde is minstens 0 en hoogstens 31 respectievelijk 127 respectievelijk 255. Alle ponsingen worden door rehep gelijkelijk geaccepteerd en worden niet op pariteit gecontroleerd. Bij programma's geponst in flexowritercode heeft een aanroep van rehep geen invloed op de door resym en read bijgehouden laatste case—definitie.

4.1.1.2. kaartinvoer.

De rijen op een kaart zijn van boven naar beneden als volgt genummerd: 12,11,0,1,2,3,4,5,6,7,8,9. De waarde van rehep wordt nu verkregen door de eerstvolgende kaartkolom als een 12—bits binair getal op te vatten, waarbij de 12—ponsing het minst significante bit is. Duiden we het nummer van het bit met nb aan, dan vinden we dus de waarde van de bit—positie uit 2 ↑ (if nb > 9 then 12 − nb else nb +2). Alle 4096 mogelijke combinaties zijn toegelaten. De afgeleverde waarde is minstens 0 en hoogstens 2 ↑ 12 − 1 = 4095. Zo levert een 11—3—8 ponsing de waarde 2 ↑ 1 + 2 ↑ 5 + 2 ↑ 10 = 1058. Voor rehep heeft einde kaart geen speciale betekenis; na kolom 80 wordt verder gelezen met kolom 1 van de volgende kaart.
In plaats van rehep mag men ook de naam REHEP gebruiken.

4.1.2. integer procedure resym;

4.1.2.1. ponsbandprogramma's in flexowriter—code.

Alle ponsingen worden gecontroleerd op toelaatbaarheid als flexowriter—symbool. De waarde die een aanroep van resym aflevert is de interne representatie van het eerste flexowriter—symbool op de band, dat verschilt van:

- de ponsingen blank (0,0), erase (15,7), backspace (5,2) en stopcode (1,3) die door resym overal geskipt worden,
- de ponsingen lower case (15,2) en upper case (15,4),

met inachtneming van de laatste, door resym of read verwerkte case—definitie. Aan het begin van de uitvoering van een programma wordt de gemeenschappelijke case van resym en read door het systeem op lower case geinitialiseerd. De interne representatie van flexowriter—symbolen is gegeven in 6.2..
In plaats van resym mag ook de naam RESYM gebruikt worden.

4.1.2.2. ponsbandprogramma's in ISO—code.
Alle ponsingen worden gecontroleerd op toelaatbaarheid als ISO—symbool. Een aanroep van resym levert de interne representatie van het eerste ISO—symbool op de band dat in 6.3.2.1. is opgenomen. Hieraan voorafgaande ponsingen uit tabel 6.3.2.2. worden overal door resym geskipt; hiertoe behoren de zogenaamde control characters met uitzondering van CR, LF en HT.

4.1.2.3. kaartprogramma's.
Alle ponsingen worden gecontroleerd op toelaatbaarheid als IBM—EL—symbool. De waarde die een aanroep van resym aflevert is de interne representatie van het symbool in de eerstvolgende kaartkolom. Na kolom 80 levert de eerstvolgende aanroep van resym de waarde 119 (twnr bij wijze van nieuwe kaart) af; eerst de hierop volgende aanroep eist de aanwezigheid van een nieuwe kaart en levert dan de interne representatie van het symbool in kolom 1 af. De interne representatie van IBM—EL—symbolen is gegeven in 6.4..

4.1.3. integer procedure resymbol;
De procedure resymbol leest ponsingen van het invoermedium en tracht hieruit een symbool op te bouwen, volgens regels die analoog zijn aan die voor resym (zie 4.1.2.). Vormen de ponsingen een herkenbaar symbool, dan wordt zijn interne representatie afgeleverd. Bevat het invoermedium een ongedefinieerde ponsing (onbekend of van foute pariteit). dan levert resymbol de binaire waarde van de ponsing negatief af. Na het lezen van het laatste symbool van het invoermedium levert resymbol eenmaal de waarde —4096 af; bij een hierop volgende aanroep van resymbol wordt het programma beeindigd met foutmelding no. 998.

4.1.4. real procedure read;
De procedure read leest vanaf het invoermedium net zoveel symbolen tot hij daaruit een getal en vervolgens een getalscheider heeft geisoleerd. Dit impliceert, dat alle met read in te lezen getallen op het invoermedium moeten worden afgesloten door een getalscheider (dus ook het laatste). De waarde van het ingelezen getal wordt als functiewaarde van read afgeleverd. De geisoleerde getalscheider wordt door read in een own variabele van read bewaard en bij de volgende aanroep van read voor de te lezen symbolen ingelast (voor de eerste aanroep van read wordt geen symbool voor de te lezen symbolen ingelast). De zo bewaarde getalscheider kan het teken van het volgende getal zijn, of de decimale punt daarvan, of het symbool "$_{10}$" (voor getallen zonder mantisse).

De waarde van read stelt het ingelezen getal voor met een relatieve precisie van ongeveer 12 decimalen en ligt in absolute waarde tussen $_{10}-616$ en $_{10}+628$ of is exact 0. Integers op het invoermedium, in absolute waarde kleiner dan 1 099 511 627 776, worden exact ingevoerd. De relatie <number>=read is true als de decimale voorstelling van <number> in de ALGOL 60-tekst en van het door read van het invoermedium gelezen getal dezelfde is.

read maakt voor het lezen van symbolen vanaf het invoermedium gebruik van resym; daarom geldt hiervoor dezelfde controle op toegelaten symbolen als bij resym en worden dezelfde symbolen geskipt.

Een met read in te lezen getal moet een number zijn in de zin van het Revised Report. Daarbij zijn bovendien spaties als lay-out-symbolen toegestaan, en wel:

a) na het teken van het getal, of na het symbool "$_{10}$", of na het teken van de exponent: willekeurig veel spaties,
b) na een cijfer of na de decimale punt: een enkele spatie.

De onder 6.1.2. sub 6 genoemde interpretatie van e als $_{10}$ treedt hier niet op.

Hieronder wordt de wijze waarop read uit de ten behoeve daarvan in te lezen symbolen een getal isoleert gedetailleerd beschreven. Daarbij wordt onder een aanhef van een getal verstaan elke symbolenrij die ofwel zelf een number is ofwel tot een number kan worden uitgebreid door er een cijfer achter te zetten.

read zoekt in de symbolenrij, die begint met de in zijn own variabele bewaarde getalscheider en die verder bestaat uit met resym in te lezen symbolen, het eerste cijfer op. read skipt daarbij in principe alle niet-cijfers en bovendien elke symbolenrij tussen twee apostrofs (dus ook eventuele cijfers tussen apostrofs). Echter worden alle symbolen die zodanig aan het eerste cijfer voorafgaan dat ze met dat cijfer de aanhef van een getal vormen, bewaard en in het te isoleren getal verdisconteerd (bijvoorbeeld "$_{10}-$" of "+."). Hierna bewaart read alle symbolen die op het eerste cijfer volgen en met de eerder bewaarde symbolen de aanhef van een getal vormen. Het eerste symbool dat niet meer met de voorafgaande bewaarde symbolen de aanhef van een getal vormt wordt opgevat als getalscheider en aan de own variabele van read geassigneerd. Dit symbool is tevens het laatste door read vanaf het invoermedium gelezen symbool.

Indien vervolgens de bewaarde symbolenrij, die in ieder geval de aanhef van een getal vormt, ook werkelijk een number voorstelt, wordt de waarde daarvan als functiewaarde van read afgeleverd. Anders wordt de executie van het programma afgebroken met een foutmelding met foutnummer 517. Dit laatste gebeurt bijvoorbeeld als read de symbolenrijen "abc+3.d" of "ab'j5:=5'cd3$_{10}-$." te verwerken krijgt.

Aangezien in een getal na elk cijfer een spatie mag staan, fungeert, indien op een cijfer twee of meer spaties volgen, pas de tweede spatie als getalscheider. Overigens fungeren als getalscheider de volgende symbolen, indien ze direct op een cijfer volgen of daarvan door ten hoogste 1 spatie gescheiden staan:

a) alle symbolen die geen cijfer, punt of "₁₀" zijn,
b) "₁₀" na een cijfer van de exponent,
c) "." na een cijfer van de exponent of na een cijfer van het breukgedeelte (een getal kan slechts 1 decimale punt bevatten).

Men zij voorzichtig met het afwisselend gebruiken van rehep, resym en read. Na een aanroep van read geeft een aanroep van rehep de getalwaarde van de eerste nog niet gelezen ponsing, een aanroep van resym de interne representatie van het eerste nog niet gelezen symbool. Daarbij blijft de waarde van de own variabele van read onaangetast, en het is deze waarde, die, ongeacht tussengelaste aanroepen van resym of rehep, bij de volgende aanroep van read als eerste te onderzoeken symbool wordt gebruikt.

Formeel wordt read beschreven door de volgende proceduredeclaratie (vergelijk 3.3.2.):

```
real procedure read;
begin   own integer s;
        comment if eerste aanroep van read then s:= 119;
        read:= read1 (resym, s)
end read;
```

In plaats van read mag ook de naam READ gebruikt worden.

Voorbeelden:

invoer	afgeleverd
a[j3]:= — 3.14 ₁₀— 3 × 15s	+3, —.00314, +15
—a5b	+5
₁₀'tekst'8	+8
.6.8₁₀3.1	+.6, +800, +.1
.6.+8₁₀3.—1	+.6, +8000, —1
8.a3	foutmelding
8₁₀a3	foutmelding

4.2. Output procedures voor de bandponser.

De bandponser ponst 8-gats papierband met een snelheid van ongeveer 150 ponsingen per seconde. De hier genoemde procedures mogen in elk programma voorkomen; het laten uitvoeren van effectieve ponsopdrachten is echter slechts mogelijk in programma's met programmaletter 'b' en 'd' (zie hoofdstuk 5.).

4.2.1. Besturingsprocedures.

Indien aan de bandponser informatie wordt aangeboden in de vorm van resymwaarden (direct via pusym of indirect via absfixp, putext, ... etc.). wordt deze informatie geponst volgens een bepaalde code. Er staan twee codes ter beschikking, MC-flexowritercode (zie 6.2.) en ISO-code (zie 6.3.). Op het moment dat een programma in uitvoering genomen wordt, wordt een code als heersende code aangewezen, en wel bij programma's in ISO-code de ISO-code en bij programma's in enigerlei andere code de MC-flexowritercode. Tijdens de uitvoering van het programma kan dan door een aanroep van een besturingsprocedure een andere code als heersende code aangewezen worden, die dan geldt tot de volgende aanroep van een besturingsprocedure.

4.2.1.1. procedure outflexo;

De procedure outflexo wijst de MC-flexowritercode als heersende code aan, en initialiseert de bijbehorende case-definitie op "noch lower, noch upper" case. Deze procedure kan ook gebruikt worden om tijdens het ponsen in MC-flexowritercode de case-definitie op "noch lower, noch upper" case te zetten, en daardoor de uitvoer van een case-definierende ponsing te forceren.

4.2.1.2. procedure outiso;

De procedure outiso wijst de ISO-code als heersende code aan.

4.2.2. De volgende procedures dienen om niet-numerieke gegevens over de bandponser uit te voeren.

4.2.2.1. procedure puhep(n); value n; integer n;

De procedure puhep ponst de laatste 8 bits van de binaire representatie van n als 1 octade op de band. Voor $0 < n < 255$ wordt dus n zelf als octade geponst. Voor $n < 0$ is de binaire representatie uit die van abs(n) af te leiden door alle nullen door enen te vervangen en alle enen door nullen.

Ten behoeve van hogere ponsprocedures houdt puhep de laatst geponste case-definitie voor flexowriter-code bij. Door puhep(122) resp. puhep(124) wordt deze case-definitie op lower, resp. upper case gezet. Aan het begin van de uitvoering van een programma wordt de case geinitialiseerd op noch lower, noch upper case. In alle gevallen wordt een en slechts een octade geponst.
In plaats van puhep mag ook de naam PUHEP worden gebruikt.

4.2.2.2. procedure pusym(n); value n; integer n;

4.2.2.2.1. Indien de heersende code MC-flexowritercode is, geldt:

Voor de waarden van n die opgenomen zijn in de tabel in 6.5.2. wordt het in die tabel vermelde symbool (of, in sommige gevallen, de combinatie van 2 symbolen) geponst in flexowriter-code, waarbij dat symbool (of bij combinatie van symbolen, elk van die symbolen) zonodig wordt voorafgegaan door een case-definitie.
Dit laatste geschiedt als de flexowriter-case van het betreffende symbool niet overeenstemt met de door puhep bijgehouden laatst geponste case-definitie. Elke aanroep van pusym loopt via puhep en beinvloedt daarom de door puhep bijgehouden case-definitie.
pusym(93) is ekwivalent met puhep(16) en met puspace(1),
pusym(118) is ekwivalent met puhep(62),
pusym(119) en pusym(135) zijn ekwivalent met puhep(26) en met punlcr;
bij deze symbolen wordt nimmer een case-definitie ingelast.
pusym(134) is ekwivalent met een dummy statement.
Voor de niet in de tabel in 6.5.2. opgenomen waarden van n wordt een ? geponst, indien nodig voorafgegaan door een upper case ponsing.

4.2.2.2.2. Indien de heersende code ISO-code is, geldt:
Voor waarden van n die opgenomen zijn in de tabel in 6.5.2. wordt het in die tabel vermelde symbool geponst in ISO-code.
pusym(93) is ekwivalent met puhep(160) en puspace(1),
pusym(118) is ekwivalent met puhep(9),
pusym(119) is ekwivalent met begin puhep(141); puhep(10) end en met punlcr,
pusym(134) is ekwivalent met puhep(141),
pusym(135) is ekwivalent met puhep(10).
Voor de niet in tabel 6.5.2. opgenomen waarden van n wordt een ? geponst.

In plaats van pusym mag ook de naam PUSYM worden gebruikt.

4.2.2.3. procedure puspace(n); value n; integer n;
puspace ponst, als n positief is, n spaties op de band in de heersende code. puspace(n) is ekwivalent met, maar sneller dan:
begin integer i; for i:= 1 step 1 until n do pusym(93) end.
In plaats van puspace mag ook de naam PUSPACE worden gebruikt.

4.2.2.4. procedure punlcr;
Indien de heersende code MC-flexowritercode is, wordt een terug-wagen-nieuwe-regel symbool geponst. Indien de heersende code ISO-code is, wordt een CR-symbool gevolgd door een LF-symbool geponst. In beide gevallen is punlcr ekwivalent met pusym(119).
In plaats van punlcr mag ook de naam PUNLCR worden gebruikt.

4.2.2.5. procedure runout;
runout ponst een stuk blanke band (80 ponsingen blank, ongeveer 20 cm band). Indien de heersende code MC-flexowritercode is, wordt de door puhep bijgehouden case-definitie op "noch lower, noch upper" case gezet.
In plaats van runout mag ook de naam RUNOUT worden gebruikt.

4.2.2.6. procedure stopcode;
Indien de heersende code MC–flexowritercode is, ponst stopcode de flexowriter ponsing stopcode (1,3), gevolgd door een stuk blanke band, terwijl de case–definitie op "noch lower, noch upper" case wordt gezet; stopcode is dan ekwivalent met begin puhep(11); runout end.
Indien de heersende code ISO–code is, ponst stopcode de ISO–ponsing READER OFF (2,3), gevolgd door een stuk blank; stopcode is dan ekwivalent met begin puhep(147); runout end.
In plaats van stopcode mag ook de naam STOPCODE worden gebruikt.

4.2.2.7. procedure putext(s); string s;
De procedure putext is het ponsende analogon van printtext (zie 4.3.2.7.); de string wordt geponst in de heersende code. Echter reageert putext nooit op een eventuele overschrijding van de regelbreedte. Bij het ponsen in MC–flexowritercode last putext zo min mogelijk case–ponsingen in en maakt hierbij gebruik van de door puhep bijgehouden case–definitie.
In plaats van putext mag ook de naam PUTEXT worden gebruikt.

4.2.3. De volgende procedures dienen om numerieke gegevens via de bandponser uit te voeren.

4.2.3.1. procedure absfixp(n,m,x); value n,m,x; integer n,m; real x;
absfixp is het ponsende analogon van absfixt (zie 4.3.3.1.), met dien verstande dat het systeem nooit een punlcr inlast.
In plaats van absfixp mag ook de naam ABSFIXP worden gebruikt.
Opmerking: het ponsen gebeurt in de heersende code; als tijdens het ponsen in MC–flexowritercode de door puhep bijgehouden case–definitie van lower case verschilt, wordt vooraf een lower case ponsing ingelast en genoemde case–definitie op lower case gezet.

4.2.3.2. procedure fixp(n,m,x); value n,m,x; integer n,m; real x;
De procedure fixp is het ponsende analogon van fixt (zie 4.3.3.2.), met dien verstande dat het systeem nooit een punlcr inlast. Verder geldt de opmerking gegeven bij de beschrijving van absfixp (zie 4.2.3.1.) ook voor fixp.
In plaats van fixp mag ook de naam FIXP worden gebruikt.

4.2.3.3. procedure flop(n,m,x); value n,m,x; integer n,m; real x;
De procedure flop is het ponsende analogon van flot (zie 4.3.3.3.), met dien verstande dat het systeem nooit een punlcr inlast. Verder geldt de opmerking, gegeven bij de beschrijving van absfixp (zie 4.2.3.1.), ook voor flop.
In plaats van flop mag ook de naam FLOP worden gebruikt.

4.2.3.4. procedure punch(x); value x; real x;
De procedure punch is het ponsende analogon van print (zie 4.3.3.4.), met dien verstande dat het systeem nooit een nlcr inlast. Verder geldt de opmerking, gegeven bij de beschrijving van absfixp (zie 4.2.3.1.), ook voor punch.
In plaats van punch mag ook de naam PUNCH worden gebruikt.

4.3. Output procedures voor de regeldrukker.

De regeldrukker kan ongeveer 20 regels per seconde afdrukken. Een regel kan maximaal 144 symbolen bevatten; de posities waarop deze symbolen zich bevinden zijn genummerd van 0 t/m 143. Een pagina kan maximaal 60 regels bevatten; deze regels zijn genummerd van 1 t/m 60.

4.3.1. Er zijn geen besturingsprocedures voor de regeldrukker.

4.3.2. De volgende procedures dienen om niet-numerieke gegevens via de regeldrukker uit te voeren en om invloed uit te oefenen op de lay-out.

4.3.2.1. procedure prsym(n); value n; integer n;

Voor de waarden van n opgenomen in de tabel 6.5.2. wordt op de regeldrukker het in die tabel vermelde symbool afgedrukt; voor alle andere waarden van n wordt het symbool ? afgedrukt.
prsym(93) is ekwivalent met space(1),
prsym(118) is ekwivalent met tab,
prsym(119) is ekwivalent met nlcr,
prsym(134) is ekwivalent met carriage(0),
prsym(135) is ekwivalent met:

```
begin   integer save print pos;
        save print pos:= print pos; comment zie 4.3.4.2.;
        nlcr;   space(save print pos)
end line feed.
```

prsym (en via prsym elke andere outputprocedure voor de regeldrukker) houdt de positie op de regel en het regelnummer op de pagina bij. Als door een aanroep van prsym meer dan 144 symbolen op een regel dreigen te ontstaan, wordt door het systeem een nlcr ingelast; als door een aanroep van prsym meer dan 60 regels op een pagina dreigen te ontstaan, wordt overgegaan op een nieuwe pagina.
Aanroepen van prsym met n = 126 of n = 127 (onderstreping of doorbalking) verhogen de positie op de regel niet. Om een symbool te onderstrepen (doorbalken) geve men eerst de onderstreping (doorbalking), daarna het symbool.
In plaats van prsym mag men ook de naam PRSYM gebruiken.

4.3.2.2. procedure space(n); value n; integer n;

space verhoogt, als n positief is, de positie op de regel met n. Als daarbij het aantal symbolen op de regel de 144 dreigt te overschrijden, last het systeem zoveel regelopvoeren (daarbij telkens de positie op de regel verlagend met 144) in als nodig is om juiste regelbreedte te garanderen. space(n) is ekwivalent met, maar sneller dan:

```
begin integer i; for i:= 1 step 1 until n do prsym(93) end.
```

In plaats van space mag men ook de naam SPACE gebruiken.

4.3.2.3. procedure tab;

tab verhoogt in principe de positie op de regel met minstens 2 en ten hoogste 9, zodanig dat een 8—voud bereikt wordt. De denkbeeldige 'tabulator—stoppen' bevinden zich dus op de posities genummerd 8, 16, 24, ... , 136 en op de niet bestaande positie 144. Indien de positie op de regel verhoogd dreigt te worden tot 152, dan wordt een regelopvoer ingelast en de positie op de regel op 8 gesteld.
tab is ekwivalent met space(9 - remainder(print pos + 1, 8)).
In plaats van tab mag men ook de naam TAB gebruiken.

4.3.2.4. procedure nlcr;
nlcr geeft een regelopvoer en stelt de positie op de regel terug op 0. Als door het uitvoeren van nlcr het aantal regels op de pagina de 60 zou overschrijden, wordt in plaats van een regelopvoer een overgang naar een nieuwe pagina bewerkstelligd. In plaats van nlcr mag men ook de naam NLCR gebruiken.

4.3.2.5. procedure carriage(n); value n; integer n;
Voor n < —1 en ook voor n = 1 is carriage(n) ekwivalent met nlcr;
voor n = —1 is carriage(n) ekwivalent met new page.
Voor de andere waarden van n worden in principe n regelopvoeren gegeven en wordt de positie op de regel op 0 gesteld.
Als hierdoor het aantal regels op de pagina de 60 zou overschrijden, wordt in plaats van de n regel—opvoeren een overgang naar een nieuwe pagina bewerkstelligd.
De aanroep 'carriage(0)' heeft tot gevolg dat de positie op de regel tot 0 wordt teruggesteld, zonder dat een regel—opvoer wordt gegeven; de reeds bedrukte regel kan dus opnieuw bedrukt worden.
In plaats van carriage mag men ook de naam CARRIAGE gebruiken.

4.3.2.6. procedure new page;
new page bewerkstelligt de overgang naar een nieuwe pagina en stelt de positie op de regel op 0. Iedere nieuwe pagina wordt door het systeem voorzien van een standaardkopje (zie 5.4.).
In plaats van new page mag men ook de naam NEW PAGE gebruiken.

4.3.2.7. procedure printtext(s); string s;
De actuele parameter bij een aanroep van printtext mag uitsluitend een string of een formele identifier (in het geval van een "doorgegeven" formele parameter) zijn, mits deze laatste uiteindelijk met een string correspondeert.
De symbolen van de string, ontdaan van de buitenste string—quotes, worden, symbool na symbool, afgedrukt. Telkens als daarbij de regelbreedte van 144 posities overschreden dreigt te worden, last het systeem een nlcr in. Indien de string een onderstreepte doorbalkte hoofdletter bevat, wordt de overeenkomstige onderstreepte doorbalkte kleine letter afgedrukt. In plaats van printtext mag men ook de naam PRINTTEXT gebruiken.

4.3.3. De nu volgende procedures dienen om numerieke gegevens uit te voeren via de regeldrukker. Algemeen geldt dat het afgedrukte getal nooit over twee regels verdeeld wordt.

4.3.3.1. procedure absfixt(n,m,x); value n,m,x; integer n,m; real x;
De absolute waarde van x wordt door de procedure absfixt in het algemeen afgedrukt in vaste-komma-representatie met n cijfers voor de decimale punt, m cijfers erna, het geheel voorafgegaan en gevolgd door een spatie. Als m = 0, wordt het afdrukken van de decimale punt onderdrukt. In het gehele gedeelte van x worden non-significante nullen door spaties vervangen, uitgezonderd de nul op de eenhedenpositie in het geval m = 0 (dwz. als het breukgedeelte ontbreekt).
De waarde van x wordt exact op de laatste af te drukken decimaal afgerond. Als het resultaat in absolute waarde $\geq 10 \uparrow n$ is, of als niet voldaan is aan elk van de relaties $n > 0$, $m \geq 0$, $1 \leq n + m \leq 21$, dan wordt absfixt(n,m,x) door het systeem vervangen door flot(13,3,x).
Een aanroep van absfixt(n,m,x) verhoogt in principe de positie op de regel met if m = 0 then n + 2 else n + m + 3; de afdruk neemt dus minstens 3 en hoogstens 24 posities in.
Als hierdoor het aantal symbolen op de regel groter dan 144 zou worden, wordt voor de aanvang van het afdrukken door het systeem een nlcr ingelast. In plaats van absfixt mag men ook de naam ABSFIXT gebruiken.

4.3.3.2. procedure fixt(n,m,x); value n,m,x; integer n,m; real x;
De procedure fixt verschilt slechts van de procedure absfixt in dit opzicht, dat in plaats van de spatie die direct aan het eerste cijfer of aan de decimale punt voorafgaat, het teken van x (+ of –) wordt afgedrukt. In plaats van fixt mag men ook de naam FIXT gebruiken.

4.3.3.3. procedure flot(n,m,x); value n,m,x; integer n,m; real x;
De procedure flot drukt de waarde van x af in drijvende, decimale representatie. Na het teken van x en de decimale punt volgen een mantisse van n cijfers, het symbool "₁₀", het teken van de decimale exponent, de absolute waarde van die exponent in m cijfers (waarbij non-significante nullen, behalve op de eenhedenposities, vervangen worden door spaties), en tenslotte een spatie.
Voor x = 0 worden een mantisse 0 en een decimale exponent 0, beide met het goede aantal cijfers afgedrukt. Voor $x \neq 0$ wordt de decimale exponent zo bepaald, dat de mantisse in absolute waarde $\geq .1$ en < 1 is. Als de zo verkregen decimale exponent niet in m cijfers kan worden afgedrukt, of als niet voldaan is aan $1 < n < 13$, $1 \leq m \leq 3$, dan wordt flot(n,m,x) vervangen door flot(13,3,x).
De mantisse wordt exact op de laatste decimaal afgerond. Een aanroep van flot(n,m,x) verhoogt in principe de positie op de regel met n + m + 5; de afdruk neemt dus minstens 7 en hoogstens 21 posities in. Als hierdoor het aantal symbolen op de regel groter dan 144 zou worden, wordt vooraf door het systeem een nlcr ingelast.
In plaats van flot mag men ook de naam FLOT gebruiken.

4.3.3.4. procedure print(x); value x; real x;
Als de absolute waarde van x gelijk is aan een geheel getal, kleiner dan 1 099 511 627 776, wordt x afgedrukt volgens fixt(13,0,x), gevolgd door 6 extra spaties. Zo niet, dan volgens flot(13,3,x). In beide gevallen wordt de positie op de regel met 21 verhoogd; zonodig wordt echter vooraf door het systeem een nlcr ingelast.
In plaats van print mag men ook de naam PRINT gebruiken.

4.3.4. De volgende procedures dienen voor het verkrijgen van informatie aangaande de regeldrukker.

4.3.4.1. integer procedure line number;
De waarde van de function designator line number is het nummer van de regel "in opbouw" op de heersende pagina. Deze waarde is minstens 1 en hoogstens 60; na overgang op een nieuwe pagina levert line number de waarde 1 af.
In plaats van line number mag men ook de naam LINE NUMBER gebruiken.

4.3.4.2. integer procedure print pos;
De waarde van de function designator print pos is het nummer van de eerste vrije positie op de regel. Deze waarde ligt tussen 0 en 144, waarbij 0 betekent dat regel helemaal leeg is en 144 dat de regel helemaal vol is.

4.4. Procedures voor de magnetische trommel.
Van de ongeveer 500000 woorden op de magnetische trommel is een gedeelte voor de programmeur beschikbaar. De hier beschreven procedures mogen in alle programma's voorkomen; men kan ze echter uitsluitend aanroepen in programma's met programmaletter 'c' en 'd' (zie hoofdstuk 5).

4.4.1. procedure to drum(A,p); value p; array A; integer p;

4.4.2. procedure from drum(A,p); value p; array A; integer p;
Beide procedures nemen contact op met het trommelgeheugen, de eerste om de elementen van array A op de trommel op te bergen, de tweede om het array A vanaf de trommel te vullen. Het array A kan hierbij een real array, integer array of Boolean array zijn.
Voor het dumpen en weer ophalen van de arrayinhouden staat op de trommel een traject ter beschikking. Zij dumpmax het aantal woorden dat dit traject lang is, dan geldt voor programma's met programmaletter 'c': dumpmax = 40960 en voor 'd': dumpmax = 81920. De woorden van het trommeltraject zijn genummerd van 0 t/m dumpmax − 1. De parameter p geeft nu aan, welk van deze woorden met het eerste element van het array moet corresponderen. Steeds moeten $p \geq 0$ en $p + aw \leq dumpmax$ zijn, waarbij aw het aantal woorden voorstelt dat in beslag genomen wordt door de elementen van het array;

voor een integer array is dit gelijk aan het aantal elementen, voor een real array 2 × het aantal elementen en voor een Boolean array (aantal elementen + 26) : 27.
De tijdsduur van het transport van of naar de trommel kan geschat worden op 50 milliseconden per 1000 woorden met een minimum van 25 msec. Het is dus niet efficient om arrays van kleine omvang te dumpen. Wel wordt gedurende het transport van een array reeds verder gerekend, zolang het programma maar geen gebruik maakt van het array in kwestie, of het blok waarin dat array gedeclareerd is, verlaat.
In plaats van to drum mag men ook de naam TO DRUM gebruiken en in plaats van from drum ook FROM DRUM.

4.5. Output procedures voor de kaartponser.

De kaartponser ponst 80–koloms kaarten met een snelheid van 4 kaarten per seconde. De kolommen zijn genummerd van 1 t/m 80. Elke kolom bestaat uit 12 rijen, die van boven naar beneden als volgt genummerd zijn: 12,11,0,1,2,3,4,5,6,7,8,9.
De hier genoemde procedures mogen in elk programma voorkomen; het laten uitvoeren van effectieve ponsopdrachten is echter alleen toegestaan in programma's met programma–letter 'd' (zie hoofdstuk 5.).

4.5.1. Er zijn geen besturingsprocedures voor de kaartponser.

4.5.2. De volgende procedures dienen om niet–numerieke gegevens via de kaartponser uit te voeren.

4.5.2.1. procedure col(n); value n; integer n;

De procedure col ponst de laatste 12 bits van de binaire representatie van n in 1 kolom van de kaart (waarbij de 12–ponsing het minst signifikante bit is). Voor $0 \leq n \leq 4095$ wordt dus n zelf geponst. Voor $n < 0$ is de binaire representatie uit die van abs(n) af te leiden door alle enen door nullen te vervangen en omgekeerd.
col (en via col elke andere outputprocedure voor de kaartponser) houdt een kolomnummer op de kaart bij. Bij een aanroep van col wordt geponst in de door het kolomnummer aangegeven kolom en vervolgens wordt het kolomnummer met 1 verhoogd. Indien het kolomnummer groter is dan 80 last col voorafgaande aan het ponsen van de kolom een overgang op een nieuwe kaart in en stelt daarbij het kolomnummer op 1.

4.5.2.2. procedure csym(n); value n; integer n;

Voor n = 119 en n = 135 bewerkstelligt csym een overgang op een nieuwe kaart en stelt daarbij het kolomnummer op 1.
Voor n = 134 is csym ekwivalent met een dummy statement.
Voor n = 118 worden minstens 2 en ten hoogste 9 spaties geponst, zodanig dat na afloop het kolomnummer een 8–voud + 1 bedraagt. Slechts indien vooraf het kolomnummer groter dan 79 was, wordt eerst overgang op een nieuwe kaart bewerkstelligd.
Voor de overige waarden van n die opgenomen zijn in de tabel in 6.5.2. wordt het in die tabel vermelde symbool (of in enkele gevallen, de combinatie van 2 symbolen) geponst in 1 kolom (of bij combinaties in 2 kolommen).

Tevens wordt achteraf het kolommummer met 1 of bij combinaties met 2 verhoogd. Indien voorafgaande aan het ponsen het kolomnummer groter dan 80 was (of bij combinaties, groter dan 79), last csym eerst een overgang op een nieuwe kaart in en stelt daarbij het kolomnummer op 1. Voor de waarden van n die niet in tabel 6.5.2. voorkomen wordt het symbool ? (ponsing 0–7–8) geponst.

4.5.3. De volgende procedures dienen om numerieke gegevens via de kaartponser uit te voeren.

4.5.3.1. procedure absfixc(n,m,x); value n,m,x; integer n,m; real x;
absfixc is geheel het ponsende analogon van absfixt (4.3.3.1.). Een aanroep absfixc(n,m,x) verhoogt in principe het kolomnummer met if m = 0 then n + 2 else n + m + 3. Als hierdoor het kolomnummer groter dan 81 zou worden, wordt vooraf een overgang op een nieuwe kaart bewerkstelligd en daarbij het kolomnummer op 1 gesteld.

4.5.3.2. procedure fixc(n,m,x); value n,m,x; integer n,m; real x;
fixc is geheel het ponsende analogon van fixt (4.3.3.3.). Verder gelden de opmerkingen gegeven bij absfixc (4.5.3.1.) ook voor fixc.

4.5.3.3. procedure floc(n,m,x); value n,m,x; integer n,m; real x;
floc is geheel het ponsende analogon van flot (4.3.2.3.). Een aanroep floc(n,m,x) verhoogt in principe het kolomnummer met n + m + 5. Als hierdoor het kolomnummer groter dan 81 zou worden, wordt vooraf een overgang op een nieuwe kaart bewerkstelligd en daarbij het kolomnummer op 1 gesteld.

4.5.3.4. procedure cpunch(x); value x; real x;
cpunch is geheel het ponsende analogon van print (4.3.3.4.). Steeds wordt in principe het kolomnummer met 21 verhoogd. Als hierdoor het kolomnummer groter dan 81 zou worden, wordt vooraf een overgang op een nieuwe kaart bewerkstelligd en daarbij het kolomnummer op 1 gesteld.

4.5.4. De volgende procedure dient om informatie aangaande de kaartponser te verkrijgen.

4.5.4.1. integer procedure cpos;
De waarde van de function designator cpos is het door de procedure col en via col ook door alle andere outputprocedures bijgehouden kolomnummer. Deze waarde is tenminste 1 en ten hoogste 81 en is slechts 1 voorafgaande aan de eerste kaartponsopdracht of na een aanroep csym(119) of csym(135).

4.6. Output procedures voor de plotter.

De plotter kan per seconde maximaal 300 stapjes van elk 0.1 mm uitvoeren; de maximale breedte van een tekening is 2794 stapjes, de lengte in principe onbeperkt. Het op– en neer–bewegen van de pen kost 100 msec.

De positie van de pen kan door de programmeur uitgedrukt worden in plotterstapjes (plits) of in door de programmeur te definieren eenheden (dits), en is altijd relatief ten opzichte van een van te voren gedefinieerd kader (frame). Voorafgaand aan de eerste plotopdracht moet het kader gedefinieerd worden, b.v. door een aanroep van plotframe (zie 4.6.1.2.) waardoor tegelijkertijd het verband tussen plotterstapjes en dits vastgelegd wordt.
Het kader in plits uitgedrukt loopt van (0,0) links onder tot (maxx,maxy) rechts boven, en in dits uitgedrukt loopt het van (xmin,ymin) links onder tot (xmax,ymax) rechts boven; de systeemvariabelen maxx, maxy, xmin en ymin worden door de procedure plot bijgehouden,xmax,ymax,xlast en ylast door move.
Symbolen worden getekend op een parallellogram (symboolkader) met een breedte van 4 eenheden en een hoogte van 7 eenheden, dat loopt van (0,0) links onder tot (4,7) rechts boven.
De vorm van het symboolkader en zijn grootte kan door de procedure shape (zie 4.6.1.3.) ingesteld worden.
Waar in de procedures sprake is van hoeken worden deze uitgedrukt in graden. De hoek tussen a en b is positief als a linksom gedraaid moet worden om met b samen te vallen.
De plotterprocedures en hun toepassingen staan uitgebreid beschreven in LR1.4; hieronder volgt een beknopte beschrijving.
De hier genoemde procedures mogen in elk programma voorkomen; het laten uitvoeren van effectieve plotopdrachten is echter alleen toegestaan in programma's met programmaletter 'd' (zie hoofdstuk 5.).

4.6.1. Besturingsprocedures.

4.6.1.1. real procedure plot(x,y,ipen); value x,y, ipen; real x, y; integer ipen;

plot vormt de basis van de procedures voor de plotter; plot kan daartoe enige elementaire plothandelingen uitvoeren en houdt verder de administratie bij betreffende penstand en scaling (xlast, ylast, pen, maxx, maxy, scx, scy, xmin, ymin), waarbij de scaling factoren resp. scx = (xmax − xmin)/maxx en scy = (ymax − ymin)/maxy zijn. De precieze werking van plot is afhankelijk van de waarde van ipen, en wel als volgt.

Aanroepen die een beweging veroorzaken.
Voor 0 < abs(ipen) ≤ 4 wordt de pen van het punt (xlast,ylast) naar het punt (x,y) bewogen, waarbij voor positieve ipen x en y uitgedrukt zijn in dits en voor negatieve ipen in plits; deze beweging wordt voor abs(ipen)=1 uitgevoerd met pen neer, voor abs(ipen)=2 met pen op en voor abs(ipen)=3 met verticale penstand onveranderd, terwijl voor abs(ipen) = 4 een streepjeslijn wordt getekend.

Informatieve aanroepen.
ipen= 0: plot:= gebruikte plottertijd in eenheden van 1/300 sec.
ipen= 5: plot:= de waarde van x in dits, geconverteerd naar plits.

ipen= 6: plot:= de waarde van y in dits, geconverteerd naar plits.
ipen= 7: plot:= de waarde van x in plits, geconverteerd naar dits.
ipen= 8: plot:= de waarde van y in plits, geconverteerd naar dits.
ipen= 9: plot:= de waarde van xmin in dits.
ipen=10: plot:= de waarde van ymin in dits.
ipen=11: plot:= de waarde van scx.
ipen=12: plot:= de waarde van scy.
ipen=13: plot:= de waarde van maxx in plits.
ipen=14: plot:= de waarde van maxy in plits.
ipen=15: plot:= de waarde van xlast in plits.
ipen=16: plot:= de waarde van ylast in plits.
ipen=17: plot:= de waarde van xlast, geconverteerd naar dits.
ipen=18: plot:= de waarde van ylast, geconverteerd naar dits.
ipen=19: plot:= if pen op then +1 else −1.

Besturende aanroepen.
ipen= 21: xmin:= x; ymin:= y; pen omhoog.
ipen=−21: xmin:= x; ymin:= y.
ipen= 22: scx:= x; scy:= y.
ipen= 23: maxx:= x; maxy:= y; xlast:= ylast:= 0;
Het kader wordt gecentreerd in de y–richting.
pen naar punt (0,0).
ipen=−23: maxx:= x; maxy:= y.
ipen= 24: xlast:= x; ylast:= y.
ipen= 25: pen op:= x > 0.

4.6.1.2. procedure plotframe(xmin, ymin, xmax, ymax, maxx, maxy);
value xmin, ymin, xmax, ymax, maxx, maxy;
real xmin, ymin, xmax, ymax; integer maxx, maxy;
plotframe definieert het kader, dat na de aanroep loopt van (xmin,ymin) links onder tot (xmax,ymax) rechts boven, waarbij xmin, ymin, xmax en ymax in dits uitgedrukt zijn. De fysische lengte en breedte van het kader bedragen maxx/10 mm en maxy/10 mm. maxx mag niet groter zijn dan 15000, maxy niet groter dan 2794.

4.6.1.3. procedure shape(angle, height, italicity);
value angle, height, italicity; real angle, height, italicity;
Deze procedure definieert vorm en afmetingen van de navolgend te plotten symbolen, benevens de hoek, waaronder deze geplot moeten worden.
angle geeft de hoek aan tus en de x–as en de onderkant van het symboolkader.
height geeft de hoogte van het symboolkader in plits aan.
italicity geeft de hoek tussen de zijkant van het symboolkader en een loodlijn op de onderkant van het symboolkader aan.

4.6.1.4. procedure coord(x, y, dits); value x, y, dits;
real x, y; Boolean dits;
Deze procedure legt het referentiepunt (zie 4.6.2.2.) vast van het eerstvolgend te plotten symbool; tevens wordt hierdoor het begin van een regel gedefinieerd. De waarde van dits bepaalt of x en y gegeven zijn in dits (true) of in plits (false).

4.6.1.5. integer procedure scale(ti, i, n, nint, mode, min, max, dl);
value n, nint, mode; real ti, min, max, dl;
integer i, n, nint, mode;
Voor i=1(1)n wordt de expressie ti geevalueerd en bij de aldus gegeven rij van waarden wordt een passende schaalverdeling opgebouwd uit mooie getallen.
mode=0 betekent: verdeling in precies nint intervallen.
mode=1 betekent: verdeling in nint of minder intervallen.
mode=2 betekent: verdeling in ongeveer nint intervallen.
min wordt de waarde van het eerste punt op de as, max wordt de waarde van het laatste punt op de as terwijl dl de lengte van het interval wordt, dit alles in dits.
scale levert het aantal intervallen in de schaalverdeling af.

4.6.2. Non–numerieke uitvoerprocedures.

4.6.2.1. procedure move(alpha); value alpha; integer alpha;
Deze procedure laat de plotter een enkele beweging uitvoeren; het effect is slechts gedefinieerd voor de volgende waarden van alpha.
alpha= 0: geen beweging.
alpha= 1: een stapje in de +x–richting.
alpha= 2: een stapje in de –x–richting.
alpha= 4: een stapje in de +y–richting.
alpha= 8: een stapje in de –y–richting.
alpha=16: pen op.
alpha=32: pen neer.
alpha= 5: gelijktijdig een stapje in de +x en +y–richting.
alpha= 6: gelijktijdig een stapje in de –x en +y–richting.
alpha= 9: gelijktijdig een stapje in de +x en –y–richting.
alpha=10: gelijktijdig een stapje in de –x en –y–richting.

4.6.2.2. procedure plotsym(n); value n; integer n;
Deze procedure is het plottend analogon van prsym (zie 4.3.2.1.). Een lijst van beschikbare symbolen is opgenomen in tabel 6.5.2.; voor waarden van n welke niet in deze tabel voorkomen wordt een ? geplot.
De vorm van de symbolen wordt bepaald door de laatst voorafgaande aanroep van shape (zie 4.6.1.3.); de plaats van het symbool wordt bepaald door het referentiepunt (zie 4.1.6.4.). Na het plotten van het symbool rekent plotsym het referentiepunt voor het eerstvolgende symbool uit.
Voor n=0 t/m n=135 en voor n=145 en n=146 komt het referentiepunt overeen met de linkeronderhoek van het symboolkader. Voor de waarden 136 t/m 144 komt het overeen met het punt (2,3) van het symboolkader; deze symbolen zijn bedoeld als markeringen.
Voor een aanroep van plotsym moeten de procedures shape en coord elk minstens eenmaal aangeroepen zijn.

4.6.2.3. procedure plotsymbol(xc, yc); string xc, yc;

plotsymbol tekent een symbool waarvan de vorm door xc en yc gegeven wordt; de vorm, plaats en grootte van het symboolkader worden bepaald als bij plotsym (zie 4.6.2.2.). Het symbool wordt beschreven door een lijst van coordinaten op het symboolkader die in de gegeven volgorde door rechte lijnen verbonden worden. De x–coordinaten staan in de gegeven volgorde in xc als de cijfers van 0 t/m 4, terwijl de y–coordinaten in deze volgorde in yc staan als de cijfers van 0 t/m 7. Zo specificeert b.v. plotsymbol(‡11331‡,‡07700‡) de coordinatenlijst (1,0), (1,7), (3,7), (3,0) en (1,0) en tekent dus een smalle O.
In xc mogen ook de cijfers 5, 6 en 7 voorkomen met de volgende betekenis.
xc bevat 5:
Het kader wordt naar beneden verschoven over zoveel eenheden als het overeenkomstige cijfer in yc aangeeft.
xc bevat 6:
Het kader wordt naar boven verschoven over zoveel eenheden als het overeenkomstige cijfer in yc aangeeft.
xc bevat 7:
De eerstvolgende (en slechts de eerstvolgende) penbeweging wordt met pen omhoog uitgevoerd.
De genoemde verschuivingen van het symboolkader gelden totdat ze herroepen worden of tot het einde van de heersende aanroep van plotsymbol.
Voor een aanroep van plotsymbol moeten de procedures shape en coord elk minstens eenmaal aangeroepen zijn.

4.6.2.4. procedure plottext(s); string s;
Deze procedure plot de tekst gegeven in s. Om in de string een van de speciale symbolen, genoemd onder plotsym, mee te geven, dient men de waarde van dit symbool, direct voorafgegaan en gevolgd door onderstreepte accenten, in de string te plaatsen, b.v. plottext(‡abc'143'de‡). Hierbij hebben de waarden 200 en 300 een speciale betekenis: 200 heeft tot effect dat alle in de string erop volgende letters als hoofdletter worden getekend, 300 doet dit effect te niet.
Voor een aanroep van plottext moeten de procedures shape en coord elk minstens eenmaal aangeroepen zijn.

4.6.2.5. procedure paperfeed(nplits); value nplits; integer nplits;
De pen beweegt zich naar het punt (maxx+nplits,0); de pen is daarna op papier. De maximum waarde voor nplits is 3000. Een kader, gelijk aan dat van de voorafgaande tekening, kan nu eenvoudig gedefinieerd worden door de aanroep plot(0,0,24).

4.6.2.6. procedure plotaxis(x, y, angle, length, dl);
value x, y, angle, length, dl; real x, y, angle, length, dl;
Deze procedure tekent vanuit het punt (x,y) een lijn ter lengte abs(length) die een hoek angle met de x–as maakt. Langs deze lijn wordt een schaalverdeling aangebracht door middel van streepjes op afstanden dl.
Als length>0 is, zijn de waarden van x, y, length en dl opgegeven in dits; als length<0 is, worden deze waarden als plits opgevat.

De schaalverdelingsstreepjes worden alleen aangebracht als abs(dl)≤abs(length); als dl<0 worden ze aan beide zijden aangebracht, als dl>0 slechts aan een kant.

4.6.2.7. real procedure plotaxis2(min, max, dl, horizontal, other);
value min, max, dl, horizontal, other;
real min, max, dl, other; Boolean horizontal;
De procedure tekent, als horizontal true is, een lijn vanuit het punt (min,other) naar het punt (max,other) en als horizontal false is, van (other,min) naar (other,max). Langs deze lijn worden schaalverdelingsstreepjes aangebracht met een onderlinge afstand dl; bij elk schaalverdelingsstreepje wordt de overeenkomstige waarde getekend, dit alles voor zover de ruimte het toelaat.
De waarden min, max, dl en other moeten in dits worden opgegeven. De waarden min, max en dl kan men het best verkrijgen door een aanroep van scale (zie 4.6.1.5.); in het bijzonder moet (max – min)/dl een geheel getal zijn.
plotaxis2 levert een begincoordinaat in dits af voor eventuele tekst langs de betreffende as, en wel een y–coordinaat bij een x–as en een x–coordinaat bij een y–as.

4.6.2.8. real procedure logaxis(emin,emax,i,horizontal,other);
value emin,emax,i,horizontal,other; integer emin,emax,i;
boolean horizontal; real other;
De procedure tekent, als horizontal true is, een as vanuit het punt (emin,other) naar het punt (emax,other), en als horizontal false is, van (other,emin) naar (other,emax). Langs deze as worden volgens logaritmische schaal verdelingsstreepjes aangebracht. Begin– en eindpunt van de as stellen resp. de waarden $_{10}$emin en $_{10}$emax voor.
Eventuele annotatie bij de verdelingsstreepjes vindt als volgt plaats: tienmachten worden altijd getekend, bij de tussenliggende verdelingsstreepjes worden de getallen 2 t/m 9 getekend, indien de ruimte dit toelaat.
De waarde van i bestuurt het tekenen van de as:

1) i < 0 De as wordt niet getrokken, wel de verdelingsstreepjes.
2) i > 0 As en verdelingsstreepjes worden getrokken.
3) abs(i) = 1 Verdelingsstreepjes boven of rechts van de as; geen annotatie.
4) abs(i) = 2 Verdelingsstreepjes onder of links van de as; geen annotatie.
5) abs(i) = 3 Verdelingsstreepjes aan beide zijden van de as; geen annotatie.
6) abs(i) = 4 Verdelingsstreepjes boven of rechts van de as; met annotatie.
7) abs(i) = 5 Verdelingsstreepjes onder of links van de as; met annotatie.

De waarden van emin,emax en other moeten in dits worden opgegeven. logaxis levert een begincoordinaat in dits af voor eventuele tekst langs de betreffende as, en wel een y–coordinaat bij een x–as en een x–coordinaat bij een y–as.

4.6.2.9. real procedure plotcurve(x, y, i); value x, y, i;

real x, y; integer i;
Met behulp van deze procedure kan door een aantal punten een vloeiende kromme getrokken worden. De coordinaten van deze punten worden als parameters bij opeenvolgende aanroepen van plotcurve meegegeven; het precieze effect hangt van de waarde van i af.

i = 0
Eerste aanroep, de beginhelling van de kromme is bekend en wordt in x gegeven.

i = 1
Eerste aanroep, de beginhelling is onbekend.

i = -1
Door deze aanroep wordt voor het vervolg van de kromme x of y als onafhankelijke variabele gekozen: is x > 0 dan wordt x onafhankelijk en anders y. Zonder deze aanroep kiest de procedure zelf bij ieder punt opnieuw.

abs(i) = 2
Punt op de kromme, de coordinaten worden in x en y meegegeven, in dits indien i > 0, in plits indien i < 0.

i = 3
Beeindiging van de kromme, de eindhelling is niet bekend.

i = 4
Beeindiging van de kromme, de eindhelling is bekend en wordt in x meegegeven.

abs(i) = 5
van een getrokken kromme wordt overgegaan op een streepjeslijn, van een streepjeslijn wordt overgegaan op een getrokken kromme; zie verder het geval abs(i)=2.

In aanroepen met i=0 of i=1 heeft y een speciale functie:
als y<0 wordt de kromme als streepjeslijn geplot en als abs(y)>1 worden de opgegeven punten op de kromme gemarkeerd met het symbool dat in tabel 6.5.2. de waarde abs(y)+134 heeft.
Markering en streepjeslijn kunnen gecombineerd worden.
plotcurve levert de helling af van de kromme in het vorige punt.
Onder helling wordt verstaan: dy/dx.

4.6.2.10. procedure plotpicture(xi, yi, i, n, mark, deltamark, mode,
xmin, xmax, dx, maxx, xstring,
ymin, ymax, dy, maxy, ystring,
draw);
value n, mark, deltamark, mode,
xmin, xmax, dx, maxx,
ymin, ymax, dy, maxy;
real xi, yi, xmin, xmax, dx, ymin, ymax, dy;
integer i, n, mark, deltamark, mode, maxx, maxy;
string xstring, ystring;

Met deze procedure kunnen grafieken getekend worden, compleet met assenkruis, ruitjesverdeling en commentaar; meerdere grafieken kunnen op hetzelfde assenkruis getekend worden; bij dezelfde x–as kunnen meerdere y–assen optreden.

xi en yi zijn expressies die voor i=1(1)n de coordinaten van de punten uit de grafiek aangeven.

mark
voor abs(mark)≤1 worden de punten in de grafiek niet gemarkeerd; voor abs(mark)>1 worden de punten in de grafiek gemarkeerd met het symbool dat in tabel 6.5.2. de waarde abs(mark) + 134 heeft. Als mark < 0 is wordt een eventuele kromme door de punten als streepjeslijn getekend.

deltamark geeft aan dat om de abs(deltamark) punten een punt uit de grafiek gemarkeerd moet worden. Voor deltamark < 0 blijft het tekenen van een kromme door de punten achterwege.

mode= 1000 × kg + 100 × kd + 10 × kx + ky.

kg=0: nieuw assenkruis, zonder ruitjesverdeling.
kg=1: nieuw assenkruis, met ruitjesverdeling.
kg=2: oude assenkruis.

kd=0: geen hellingen gegeven.
kd=1: voor i=0 levert xi de beginhelling.
kd=2: voor i=0 levert yi de eindhelling.
kd=3: voor i=0 levert xi de beginhelling en yi de eindhelling.

ky=0: dy= intervallengte in plits.
maxy= lengte y–as in plits.
ky=1: dy= intervallengte in plits.
maxy= bovengrens lengte y–as in plits.
ky=2: dy= benaderde intervallengte in plits.
maxy= lengte y–as in plits.
ky=3: ymin= minimumwaarde van yi.
ymax= maximumwaarde van yi.
dy= intervallengte in plits.
maxy= lengte y–as in plits.
ky=4: ymin= minimumwaarde van yi.
ymax= maximumwaarde van yi.
dy= intervallengte in plits.
maxy= bovengrens lengte y–as in plits.
ky=5: ymin= minimumwaarde van yi.
ymax= maximumwaarde van yi.
dy= benaderde intervallengte in plits.
maxy= lengte y–as in plits.
ky=6: van ymin in stappen dy naar ymax.
maxy= lengte y–as in plits.
ky=7: geen nieuwe y–as.

Hetzelfde geldt mutatis mutandis voor kx en de x–as, waarbij echter nooit meer dan 1 x–as getekend kan worden: als kg=2 wordt kx=7 verondersteld.

xstring is de tekst die langs de x–as geschreven moet worden.
ystring is de tekst die langs de y–as geschreven moet worden.

draw is een procedure met drie parameters, overeenkomstig plotcurve (zie 4.6.2.9.), die door plotpicture aangeroepen wordt met de coordinaten (en eventuele hellingen) van de opeenvolgende punten in de grafiek als parameters.

Als actuele parameter kan meegegeven worden:
a) plotcurve, voor het plotten van een kromme,
b) plotline, voor het plotten van rechten,
c) plothist, voor het plotten van een histogram,
d) de naam van een door de gebruiker te schrijven procedure, welke dan qua parameters dezelfde eigenschappen moet hebben als plotcurve.
Voor i=0 of i=1 wordt draw aangeroepen met sign(mark) als tweede parameter.

4.6.2.11. procedure plotline(x, y, i); value x, y, i;
real x, y; integer i;
plotline is een hulpprocedure, te gebruiken als parameter in een aanroep van plotpicture (zie 4.6.2.10.); de punten van de grafiek worden verbonden door rechte lijnen. In de betreffende aanroep van plotpicture moet deltamark≥0 zijn.

4.6.2.12. procedure plothist(x, y, i); value x, y, i;
real x, y; integer i;
plothist is een hulpprocedure, te gebruiken als parameter in een aanroep van plotpicture (zie 4.6.2.10.); uitgaande van de gegeven punten wordt een histogram geplot, waarbij de gegeven punten corresponderen met de rechterbovenhoeken van de balkjes. In de betreffende aanroep van plotpicture moet mark=deltamark=0 zijn.

4.6.3. De volgende procedures dienen om numerieke gegevens over de plotter uit te voeren.

4.6.3.1. procedure absfixplot(n, m, x); value n, m, x;
integer n, m; real x;
Deze procedure is het plottend analogon van absfixt (zie 4.3.3.1.), waarbij evenwel nooit een nlcr wordt ingelast.
Voor een aanroep van absfixplot moeten de procedures shape en coord elk minstens eenmaal aangeroepen zijn.

4.6.3.2. procedure fixplot(n, m, x); value n, m, x;
integer n, m; real x;
Deze procedure is het plottend analogon van fixt (zie 4.3.3.2.), waarbij evenwel nooit een nlcr wordt ingelast.
Voor een aanroep van fixplot moeten de procedures shape en coord elk minstens eenmaal aangeroepen zijn.

4.6.3.3. procedure floplot(n, m, x); value n, m, x;
integer n, m; real x;

Deze procedure is het plottend analogon van flot (zie 4.3.3.3.), waarbij evenwel nooit een nlcr wordt ingelast.
Voor een aanroep van floplot moeten de procedures shape en coord elk minstens eenmaal aangeroepen zijn.

4.6.4. De volgende procedure dient voor het verkrijgen van informatie bij het plotten van symbolen.

4.6.4.1. integer procedure plpos(x, y); integer x, y;
Aan x en y worden de waarden toegekend van het referentiepunt van het eerstvolgend te plotten symbool (in plits), terwijl plpos de positie op de regel levert; deze laatste waarde is nul na een aanroep van coord, na plotsym(119) en na plotsym(134).

4.6.4.2. procedure plshape(an,he,it); real an,he,it;
Bij een aanroep van deze procedure worden aan an, he en it de heersende waarden toegekend van resp. angle, height en italicity (zie 4.6.1.3.).

4.7. Magneetbandprocedures.

4.7.1. Algemeen.

Magneetbanden dienen om tijdelijk grote hoeveelheden informatie op te slaan. Transport is mogelijk tussen het kerngeheugen en de magneetbanden en tussen de trommel en de magneetbanden.
Een magneetband, in het vervolg ook aan te duiden met tape, wordt gekenmerkt door een nummer, dat zowel op een label aan de buitenkant als intern op de magneetband staat.
De structuur van de tape is als volgt: Een headerblok, nul of meer logische blokken, die door een ALGOL programma op de tape geschreven zijn, en als afsluiting een markeringssymbool EOI (end of information). Enige meters voor het fysische einde van de band is een aanduiding EOT (end of tape) aangebracht, die het einde signaleert.
Op de magneetbandeenheid, verder unit te noemen, bevindt zich een lees–schrijfstation. Onder de positie van de band wordt in het vervolg verstaan de positie ten opzichte van dit lees–schrijfstation.
Men heeft de beschikking over drie units, aan te duiden met 0, 1 en 2, zodat kopieren en mengen van tapes mogelijk is. Het is niet mogelijk om met meer dan drie tapes tegelijk te werken. Wel kan men meer tapes na elkaar binnen een programma gebruiken (4.7.2.1.).
Naast eigen tapes kan men ook scratch tapes gebruiken. Zij kunnen dienen als informatieopslag voor de duur van een programma. Nadat de operateur een scratch tape van de unit heeft afgenomen is de erop geschreven informatie voor geen programma meer toegankelijk (zie verder 4.7.2.1.).
Een tape kan per maand via de administratie van het Mathematisch Centrum gehuurd worden (4.7.3.).

Bij de beveiliging is de mogelijkheid ingebouwd een eenmaal beschreven tape slechts te kunnen lezen zonder hem te kunnen beschrijven (4.7.4.).
De gebruiker zal een foutmelding ontvangen wanneer het tengevolge van machinestoringen niet duidelijk is of de tape op de juiste wijze is afgesloten (4.7.6.).
De magneetbandopdrachten worden asynchroon met het programma verwerkt, in die zin dat verder gerekend kan worden tijdens de uitvoering van een magneetbandopdracht (4.7.5.).
Tape procedures mogen in ieder programma voorkomen. Effectief gebruik van magneetbanden is echter alleen toegestaan in programma's met programmaletter 'd' (zie hoofdstuk 5.).

4.7.2. Werking van de procedures.

Voor alle procedures geldt dat de actuele parameter die correspondeert met de formele parameter unit slechts de waarde 0, 1 of 2 mag hebben. Verder zullen, indien verschillende procedures een formele parameter hebben met dezelfde identifier, de betekenis van die parameter en eventuele beperkingen opgelegd aan de ermee corresponderende actuele parameter slechts eenmaal beschreven worden.

4.7.2.1. Initialiseringsprocedures.

Voorafgaande aan het gebruik van een der overige tape procedures is een aanroep van een initialiseringsprocedure verplicht om de operateur de gelegenheid te geven de tape met opgegeven number resp. een scratch tape op de aangegeven unit op te hangen en te initialiseren. Indien op de aangegeven unit al een tape hangt, wordt deze teruggespoeld en weggenomen.
Na uitvoering van de procedure is de positie van de tape na het headerblok.

4.7.2.1.1. procedure own tape(header,number,unit); value number,unit;
string header; integer number,unit;

Bij deze procedure wordt de als actuele parameter meegegeven header vergeleken met de header op de tape (zie verder onder 4.7.2.2.8. en 4.7.4.).
Voor de parameter number geldt: $10 \leq$ number ≤ 999.
De string header mag slechts bestaan uit maximaal 10 letters en/of cijfers. Lay–out symbolen mogen wel voorkomen, maar worden niet meegerekend; bovendien wordt geen onderscheid gemaakt tussen hoofd– en kleine letters. Dit betekent dat bv. ⟨MC header⟩ en ⟨mc HeAder⟩ als actuele parameter voor header hetzelfde effect hebben, en alleen verschillen in leesbaarheid.

4.7.2.1.2. procedure scratch tape(unit); value unit; integer unit;

4.7.2.2. Schrijven, lezen en positioneren.

4.7.2.2.1. procedure to tape(array,unit); value unit;
array array; integer unit;

Deze procedure transporteert de inhoud van het opgegeven array naar de tape, indien schrijven is toegestaan (zie 4.7.4.). array kan een real array, integer array of boolean array zijn. to tape begint te schrijven bij EOI. Na afloop is de positie weer EOI. Zie, voor het geval dat tijdens het schrijven EOT wordt ontdekt 4.7.2.3.1.
In het vervolg zullen we het stuk informatie dat door een opdracht op de tape geschreven wordt, een logisch blok of ook kortweg blok noemen. De logische blokken worden oplopend genummerd vanaf 1. Verondersteld is dat men zelf bijhoudt welk nummer ieder blok op deze wijze meekrijgt; dit nummer wordt de key van het blok genoemd. Zie verder 4.7.5.

4.7.2.2.2. procedure drum to tape (startaddress,range,unit);
value startaddress,range,unit;
integer startaddress,range,unit;

Het verschil met de vorige procedure is alleen dat de betreffende informatie nu van het aangegeven trommeltraject komt. Dit trommeltraject behoort tot het gedeelte van de trommel dat de programmeur ter beschikking staat (zie 4.4.) en wordt geheel bepaald door de eerste twee parameters: startaddress geeft het eerste adres van het traject aan, en range het aantal trommelwoorden.
Steeds moet gelden:

$$0 \leq \text{startaddress} < \text{startaddress} + \text{range} \leq 81920.$$

4.7.2.2.3. procedure from tape(array,unit); value unit;
array array; integer unit;

Deze procedure leest een logisch blok van de tape in het opgegeven array. Het lezen begint bij het blok waarvoor de magneetband gepositioneerd is. Na afloop is de positie voor het volgende blok of EOI. Indien men een ander blok wil lezen dan dat waarbij de magneetband gepositioneerd is, moet men eerst de procedure seek aanroepen (4.7.2.2.5.).
Het array wordt vanaf het begin gevuld. Indien het logische blok minder X8-woorden bevat dan het array, blijven de overgebleven array-elementen onaangetast (zie ook 4.7.2.3.2.). Voor het aantal woorden dat in beslag genomen wordt door een array-element zie 4.4.2.
Het maakt geen verschil of het te lezen blok destijds met to tape of drum to tape is geschreven.
Indien bij aanroep van from tape de positie EOI is, volgt een foutmelding.

4.7.2.2.4. procedure drum from tape(startaddress,range,unit);
value startaddress,range,unit;
integer startaddress,range,unit;

Het verschil met de vorige procedure is alleen dat de betreffende informatie nu naar het aangegeven trommeltraject gaat.

4.7.2.2.5. procedure seek(key,unit); value key,unit; integer key,unit;

Deze procedure stelt de positie op het begin van het logische blok met de opgegeven key. Indien key max de key is van het laatste blok voor EOI, moet gelden: 1 ≤ key ≤ key max.

4.7.2.2.6. procedure seek end(unit); value unit; integer unit;

deze procedure zoekt EOI op. Voor een zinvol gebruik zie 4.7.5. en 4.7.2.3.4.

4.7.2.2.7. procedure delete from(key,unit); value key,unit;
integer key,unit;

Deze procedure verwijdert alle informatie vanaf het begin van het logische blok met de opgegeven key.
Bij aanroep moet gelden: 1 ≤ key ≤ key max. Na afloop is de positie EOI en geldt: key max = key - 1. Hierna kunnen nieuwe blokken op de tape geschreven worden.
delete from mag alleen gebruikt worden als schrijven is toegestaan (4.7.4.).

4.7.2.2.8. procedure new header(header,retention,unit);
value retention,unit; string header; integer retention,unit;

Een magneetband wordt gekarakteriseerd door een nummer en een header (zie procedure own tape 4.7.2.1.1.). Het nummer is ook aan de administratie en de operateur bekend. De header is in principe slechts de gebruiker bekend, en als zodanig a.h.w. de sleutel tot de band, met alle voor- en nadelen van dien.
Iedere band krijgt als standaard mee de header ⟨MC header⟩.
Met behulp van de procedure new header kan men een andere header op de band zetten. Omdat het header blok vooraan op de band staat, is het neveneffect dat alle eventuele overige informatie van de band verwijderd wordt (gelijk aan delete from(1,unit)).
retention wordt als datum geinterpreteerd, en wel als volgt:
jaar = retention (modulo 100) + 1900
maand = retention : 100 (modulo 100)
dag = retention : 10 000 (vergelijk procedure date, 3.4.8.).
Deze datum heeft de volgende betekenis: schrijven op de band zal, nadat deze van de unit genomen is, niet meer toegestaan zijn, totdat de door retention bepaalde datum verstreken is.
Er moet gelden: retention ≥ 0.
new header mag alleen gebruikt worden als schrijven is toegestaan (4.7.4.) en leidt bij scratch tapes tot een foutmelding.

4.7.2.3. Informatieprocedures.

De eerste twee procedures geven informatie over de laatste schrijf- resp. leesopdracht, de laatste geven de waarde van twee belangrijke variabelen.

4.7.2.3.1. boolean procedure end of tape(unit); value unit;
integer unit;

Deze procedure heeft betrekking op de laatste schrijfopdracht die met to tape of drum to tape gegeven is. Indien hierbij EOT ontdekt is, is het hele transport ongedaan gemaakt, en is key max niet opgehoogd. Aanroep van end of tape levert dan de waarde true af, in alle andere gevallen false.
Indien end of tape true aflevert kan men nogmaals trachten een blok op de tape te schrijven, bv. een kleiner blok. Een foutmelding wordt gegeven als na ontdekken van EOT to tape of drum to tape aangeroepen wordt, zonder dat intussen end of tape aangeroepen is.

4.7.2.3.2. integer procedure words from tape(unit); value unit; integer unit;

Deze procedure levert het aantal X8–woorden af dat bij de laatste aanroep van from tape (drum from tape) in het getransporteerde logische blok stond, en wel positief als dit blok precies in het array (trommeltraject) paste of kleiner was, en negatief als het groter was. Indien in het geheel geen leesopdracht uitgevoerd is, is de waarde ongedefinieerd.

4.7.2.3.3. integer procedure retention(unit); value unit; integer unit;

Deze procedure levert een integer af, waaruit de datum is af te leiden waarna men de magneetband mag beschrijven, zie 4.7.4. Deze waarde wordt niet beinvloed door het aanroepen van new header, zie 4.7.2.2.8. Bij een scratch tape is de waarde ongedefinieerd.

4.7.2.3.4. integer procedure max key(unit); value unit; integer unit;

Deze procedure levert de waarde af van key max, indien deze aan het systeem bekend is, d.w.z. als het systeem EOI is tegengekomen; dus in de volgende gevallen:
a) Er is een schrijfopdracht gedaan (to tape, drum to tape, delete from of new header).
b) De procedure seek end is aangeroepen.
c) De tape is een scratch tape.
In andere gevallen is de waarde van de procedure ongedefinieerd.

4.7.3. Aanvraag en huur van magneetbanden.

Bij de administratie kan men onder een opdrachtnummer een band aanvragen tegen maandhuur. Men krijgt dan het nummer van de tape, waarop men kan gaan werken. Dit nummer moet altijd gebruikt worden als actuele parameter voor number bij aanroep van own tape. De actuele parameter voor header bij deze procedure is ⟨MC header⟩, althans zolang men op deze tape geen andere header geschreven heeft m.b.v. new header. Tot die tijd is de waarde van retention nul.

4.7.4. Beveiliging.

Men kan een tape gebruiken als men het nummer kent en de header. De magneetbandhuurder kan dus anderen van zijn tape gebruik laten maken. Uitgezonderd het schrijven kunnen de in 4.7.2. beschreven handelingen zonder meer uitgevoerd worden.
Voor schrijven is bovendien vereist:
a) Het opdrachtnummer van het programma is hetzelfde als dat waaronder de tape is aangevraagd.
b) De datum, die de procedure retention aflevert, moet verstreken zijn.
Onder schrijven wordt verstaan alle handelingen die informatie op de tape toevoegen of verwijderen, dus gebruik van de procedures to tape, drum to tape, delete from en new header.

4.7.5. Efficient tapegebruik.

De verwerking van de magneetbandprocedures zal in het algemeen asynchroon met de verdere executie van het programma geschieden. Het is zeer wel mogelijk dat het programma verder rekent zonder dat de laatstaangeroepen tape procedure volledig afgehandeld is.
Wachten tot een tape procedure klaar is gebeurt in de volgende gevallen:
a) als de rest van het programma voltooid is.
b) bij aanroep van een volgende tape procedure.
c) als bij een arraytransport m.b.v. to tape of from tape het programma van het array in kwestie gebruik maakt of het blok verlaat waarin het array gedeklareerd is. Het array is a.h.w. tijdelijk op slot gezet. Iets dergelijks doet zich ook voor bij de procedures to drum en from drum, zie 4.4.
d) bij aanroep van to drum of from drum na aanroep van drum from tape en bij aanroep van to drum na aanroep van drum to tape, tenzij de betreffende trommeltrajecten disjunct zijn.
Opgemerkt kan worden dat, indien to drum of from drum volgt op drum to tape of drum from tape, wachtsituaties op analoge wijze optreden: het komt er op neer dat twee procedures in disjuncte trommeltrajecten altijd gelijktijdig kunnen werken, en in overlappende trajecten gelijktijdig kunnen lezen.

Ad b dient opgemerkt te worden dat als de bedoelde procedure een informatie procedure is, alleen gewacht zal worden indien de gevraagde informatie in de in behandeling zijnde procedure nog moet worden opgebouwd of gewijzigd.
Bij de inrichting van het programma kan men profijt trekken uit het asynchrone karakter van de transportprocedures door de wachttijden te minimaliseren.
Voorbeeld: Men heeft een programma waarin veel berekeningen gedaan worden. De resultaten daarvan schrijft men op een tape, achter de daar reeds in andere programma's opgebouwde informatie. Het verdient nu aanbeveling om vooraan in het programma own tape aan te roepen, en even later seek end, zodat alle voorbereidingen getroffen zijn op het moment dat de informatie naar de tape geschreven kan worden. Zelfs het positioneren naar EOI, waar to tape of drum to tape anders nog voor zou moeten zorgen (4.7.2.2.), is al gebeurd.

4.7.6. Foutmeldingen.

Nieuwe foutnummers zijn 550 t/m 571 en 985.
Vanwege de asynchrone verwerking van de magneetbandprocedures is het mogelijk dat een fout aanleiding geeft tot twee foutmeldingen. De eerste foutmelding geeft in dat geval foutnummer 550, het regelnummer waar de executie van het programma beeindigd is (dit kan bij de laatste end zijn) en het laatst ingelezen getal. De tweede foutmelding geeft dan het foutnummer dat verklaart waarom de verwerking van de laatst aangeroepen tape procedure is vastgelopen, het regelnummer van de aanroep en nogmaals het laatst ingelezen getal.
Foutnummer 550 kan nooit alleen voorkomen.
Men wordt aangeraden in zijn eigen belang bij het optreden van de foutnummers 564, 565 of 566 de systeemprogrammeur op de hoogte te stellen. Dit kan, onder bijvoeging van het programma, via de rekendienst. Men dient hierbij de waarde van key max (het aantal logische blokken) op te geven.
Na foutnummer 569 verdient het aanbeveling de tape te laten kopieren (eveneens via de rekendienst).

5. De verwerking van ALGOL 60-programma's.

5.1. Algemeen.

5.1.1. Voor registratiedoeleinden wordt een programma gekenmerkt door twee grootheden, te weten: opdrachtnummer en programmanummer. Het opdrachtnummer geeft aan onder welke opdracht het rekenwerk wordt uitgevoerd; het wordt door het MC vastgesteld. Het programmanummer is een unsigned integer, die dient om binnen een opdracht programma's van elkaar te onderscheiden; hij kan door de gebruiker vrij worden gekozen.

5.1.2.1. Wanneer men een programma door de machine wil laten verwerken, dient men de volgende gegevens — in de aangegeven volgorde — in geponste vorm aan te bieden aan de balie van het M.C.:
a.) de monitorkop (zie 5.2.), die het bedrijfssysteem van de X8 inlichtingen verschaft over aard en herkomst van het programma,
b) de ALGOL 60 — programmatekst (zie 5.3.1),
c) het voer (zie 5.3.2.); dit zijn gegevens die tijdens de uitvoering van het programma door de invoerprocedures worden gelezen. Indien geen invoerprocedures in het programma worden aangeroepen is geen voer nodig.
Naast deze geponste gegevens dient men op een geleidekaart een aantal gegevens over de programmeur en het opdrachtnummer in te vullen.

5.1.2.2. Programma's met programmaletter 'a' kan men ook in de X8 kamer aan de operateur overhandigen, eventueel zonder envelop. Men dient dan in de X8 kamer te blijven wachten om de resultaten onmiddellijk na verwerking weer in ontvangst te nemen. Voor onbestelbare resultaten draagt het MC geen verantwoordelijkheid.

5.1.3. Voor het invoermedium waarin de gegevens worden geponst en de daarbij te gebruiken codes heeft men de keuze uit drie mogelijkheden:
a.) ponsband in MC — flexowritercode (tabel 6.2.)
b.) ponsband in ISO — code (tabel 6.3)
c) ponskaarten in IBM — EL — code (tabel 6.4.).

Monitorkop, programma en voer moeten op hetzelfde invoermedium en in dezelfde code worden geponst. Bij gebruik van ponsband herkent het systeem de code (flexowriter of ISO) aan de eerste van blank verschillende ponsing van de eerste band; in geval deze ponsing een flexowriter terug-wagen-nieuwe-regel (3,2) of een flexowriter upper case (15,4) of een flexowriter lower case (15,2) is, wordt als code flexowritercode gekozen; in geval deze ponsing een ISO-ponsing LF (1,2) of een ISO-ponsing CR (1,5) is wordt als code ISO-code gekozen. Dit houdt in dat geen andere ponsing dan de bovengenoemde toegestaan zijn als eerste ponsing op de band; ook de ponsing erase (15,7) is hier verboden. (Voor de betekenis van de getallen tussen haakjes zie hoofdstuk 6.).

5.1.3.1. Voorschriften bij gebruik van ponsband:
De banden dienen in (aan de balie verkrijgbare) enveloppen te worden ingeleverd. Het is toegestaan de geponste gegevens over meer dan een band te verdelen. Alle banden moeten beginnen en eindigen met tenminste 25 cm. tape-feed. Aan het begin van elke band dient men, in leesbaar schrift, dus niet geponst, te vermelden: indien de band het programma of een gedeelte daarvan bevat: programma 1, programma 2, ... etc.; indien de band voer bevat: gegevens 1, gegevens 2, ... etc. Door de opgegeven nummering wordt de volgorde bepaald waarin de banden door de operateur aan het systeem worden aangeboden.
Het verdient aanbeveling om tussen monitorkop en programma een stuk tapefeed in te lassen.

5.1.3.2. Voorschriften bij gebruik ponskaarten:
Ponskaarten moeten voorzien zijn van een afgeschuinde "linkerbovenhoek". Alle kaarten van een pakket moeten bij voorkeur dezelfde orientatie hebben. De kaart waarvan de bedrukte zijde zichtbaar is wordt als eerste gelezen. Het pakket kaarten dient, bijeengehouden door elastiek, bij de balie te worden ingeleverd. Grote hoeveelheden kaarten kunnen in bakken (dozen) ingeleverd worden. Wordt het pakket over meerdere bakken verdeeld, dan dienen deze de vermeldingen bak 1, bak 2, ... etc. te dragen. Het verdient aanbeveling voor eigen gebruik de kaartpakketten aan de bovenkant te merken, b.v. met behulp van een viltstift.

5.2. De monitorkop.

5.2.1. De monitorkop dient om het bedrijfssysteem van de X8 inlichtingen te verstrekken over het te verwerken programma. Dit bedrijfssysteem verwerkt simultaan vier programmastromen, genaamd 'a', 'b', 'c' en 'd'. In elke stroom geschiedt de verwerking in volgorde van inlezen.

5.2.2. De programma's in de stromen 'a' en 'b' kunnen geen gebruik maken van de procedures voor de magnetische trommel uit hoofdstuk 4.4.. Programma's in de stromen 'a' en 'c' kennen slechts de regeldrukker als uitvoerorgaan, terwijl programma's in stroom 'b' naast de regeldrukker ook de bandponser mogen gebruiken. Verder verschillen de stromen 'a', 'b' en 'c' in toegelaten verwerkingsduur en in het maximaal aantal pagina's dat op de regeldrukker kan worden afgedrukt. Programma's in stroom 'd' mogen van alle uitvoerorganen en ook van de magnetische trommel en de magneetbandapparatuur gebruik maken.

5.2.3. De keuze van de stroom wordt door de programmeur vastgelegd door middel van de programmaletter in de monitorkop. Er wordt naar gestreefd om de resultaten van de verwerking van programma's uit stroom 'a' binnen ongeveer 5 minuten na inlevering aan de programmeur te retourneren. Wanneer men een programma op syntaxfouten wil controleren kan men het beste stroom 'a' kiezen, daar de tests op bijzondere uitvoer en gebruik van de magnetische trommel en de magneetbandapparatuur pas tijdens uitvoering van het programma geschieden.

5.2.4. In iedere stroom zijn standaardhoeveelheden beschikbaar voor verwerkingstijd en hoeveelheid uitvoer op elk uitvoerorgaan. Met behulp van "aanwijzingen" in de monitorkop kan de programmeur deze hoeveelheden wijzigen. Met deze aanwijzingen mag echter nooit meer worden gevraagd dan de voor elke stroom vastgelegde maxima.

De tabellen, vermeld in 9.2.1. en in 9.2.2., geven voor elk van de vier stromen de standaardhoeveelheden en de maxima aan. Uit deze tabellen blijkt dat de kaartponser en de plotter uitsluitend in stroom 'd' gebruikt mogen worden, terwijl de bandponser in de stromen 'b' en 'd' beschikbaar is en de magnetische trommel in de stromen 'c' en 'd'.

5.2.5. De syntax van de monitorkop.
Voor de letters die in de monitorkop voorkomen mogen zowel hoofdletters als kleine letters gebruikt worden.

5.2.5.1. De indeling van de monitorkop is als volgt:

<monitor kop> ::= <begin> <opdrachtnummer>.<programmanummer>,
<naam> <aanwijzingen> <einde>
<programmaletter> ::= a | b | c | d
<aanwijzingen> ::= <empty> |,<aanwijzing> <aanwijzingen>
<aanwijzing> ::= m | z | b<int> | k<int> | p<int> | r<int> |
t<int>
<int> ::= <unsigned integer>
<opdrachtnummer> ::= <unsigned integer> <controle letter>
<programmanummer> ::= <unsigned integer>

Voorts staat <naam> voor een aantal letters en spaties (dus geen punten), die dienen om de naam van de gebruiker aan te geven.

De ponsingen voor <begin> en <einde> zijn afhankelijk van de gebruikte code:

a) flexowritercode:
<begin> ::= < twnr > < lc > <programmaletter> |
< lc > < twnr > <programmaletter>
<einde> ::= < twnr >
Hier is <twnr> de ponsing terug-wagen-nieuwe-regel (3,2) en <lc> de ponsing lower case (15,2). De drie onder <begin> gegeven ponsingen moeten precies in deze vorm op de band staan, zonder dat de ponsing erase (15,7) eraan vooraf gaat en zonder dat de ponsingen tape-feed (0,0) en erase (15,7) ertussen voorkomen. Ook overtollige of herroepen case-definities zijn verboden.

b) ISO-code:
<begin> ::= < CR > < LF > <programmaletter> |
< LF > < CR > programmaletter>
<einde> ::= < CR > < LF > | < LF > < CR >
Hier is <CR> de ISO-ponsing (1,5) en <LF> de ISO-ponsing (1,2). De drie onder <begin> gegeven ponsingen moeten precies in deze vorm op de band staan, zonder dat de ponsing DEL (15,7) eraan vooraf gaat en zonder dat de ponsingen NUL (0,0) en DEL (15,7) ertussen voorkomen. Ook alle andere control characters zijn verboden.

c) IBM - EL - kaartcode:
<begin> ::= <programmaletter>
<einde> ::= <empty>
De onder <begin> gegeven programmaletter moet in kolom 1 geponst zijn. De ruimte op de kaart achter <einde> is niet beschikbaar voor programma-tekst.

Indien <begin> niet aan de hierboven gestelde eisen voldoet, wordt foutmelding no. 15 (programmaletter incorrect) gegeven en de verwerking van het programma afgebroken. Is de unsigned integer van het opdrachtnummer groter dan 99999 of past de controleletter niet bij de unsigned integer, dan wordt foutmelding no. 16 gegeven. Is het programmanummer groter dan 999 dan wordt foutmelding no. 18 gegeven.

5.2.5.2. Bij IBM - EL - kaartcode wordt bovendien geeist dat de monitorkop op een en slechts een kaart geponst is (alle 80 kolommen mogen worden gebruikt); het systeem beschouwt de eerste kaart van het pakket met een van blank verschillende ponsing in kolom 1 als de monitorkop-kaart. De programmaletter staat dus steeds in kolom 1 en blanco kaarten aan het begin van een pakket worden geskipt.

5.2.5.3. De verschillende aanwijzingen hebben de volgende effecten:
Geeft men de aanwijzing 'm' dan wordt tijdens de executie van het programma geen regeltelling bijgehouden. Dit geeft enige besparing in het gebruik van het machinegeheugen (ruwweg 2 instructies per regel ALGOL 60-tekst) en een verwaarloosbare tijdwinst; bij foutmeldingen tijdens executie wordt in dit geval het regelnummer niet opgegeven.

Het effect van de aanwijzing 'z' is, dat de programmatekst niet over de regeldrukker wordt afgedrukt.

Bij de aanwijzing 't' wordt de verwerking van het programma afgebroken na de tijd (in milli-uren) gegeven door de waarde van het getal dat volgt op de letter t. De tijd besteed aan het vertalen van het programma telt hierin mee.
Evenzo wordt de verwerking afgebroken wanneer de hoeveelheid uitvoer van de regeldrukker (aanwijzing 'r'), de bandponser (aanwijzing 'b'), de plotter (aanwijzing 'p') of de kaartponser (aanwijzing 'k') wordt overschreden. Bij het bepalen van het aantal pagina's regeldrukkeruitvoer tellen de bladzijden waarop het programma (met eventuele foutenlijst) afgedrukt wordt, wel mee, de door het systeem toegevoegde laatste bladzijde met gegevens echter niet.
Men is niet verplicht aanwijzingen te geven; laat men een aantal van de aanwijzingen 'b', 'k', 'p', 'r' of 't' achterwege dan staat voor de betreffende uitvoerorganen en/of de verwerkingstijd de standaardhoeveelheid ter beschikking.

5.2.5.4. Voorbeeld van een monitorkop (de ponsingen voor <begin> en <einde> zijn op papier niet zichtbaar, met uitzondering van de programmaletter):

d1600b.314,J v Neumann,t300,b100

Het hierbij behorende programma krijgt de beschikking over 300 milliuren rekentijd en mag 100000 symbolen doen ponsen op de bandponser; voor de andere uitvoerorganen is de standaardhoeveelheid beschikbaar.

5.3. Programma en voer.

5.3.1. Programma.

5.3.1.1. De programmatekst moet op hetzelfde medium en in dezelfde code worden geponst als de monitorkop.

5.3.1.2. Bij ponsbandprogramma's is het eerste symbool van het programma de ponsing die volgt op de laatste ponsing van <einde> in de monitorkop. In de programmatekst worden dezelfde symbolen geskipt als bij de procedure resym (zie 4.1.2.) het geval is; bovendien wordt in programma's in ISO-code het symbool CR (1,5) overal geskipt. Na de laatste end van de programmatekst dient nog een niet-onderstreept symbool te volgen (bij voorkeur een twnr in flexowritercode of een LF in ISO-code); dit symbool wordt dan geacht het laatste symbool van de tekst te zijn, en hoort niet bij eventueel voer.

5.3.1.3. Bij kaartprogramma's is het eerste symbool van het programma de ponsing in kolom 1 van de kaart die volgt op de monitorkop–kaart. Van de programma–tekst worden slechts de kolommen 1 t/m 72 gelezen; het verdient aanbeveling de kolommen 73 t/m 80 voor eigen gebruik van een identificatie en een nummering te voorzien. Bij overgang op een nieuwe kaart wordt door het systeem een symbool twnr ingelast; dit is o.a. van belang bij het ponsen van strings. Komt echter het symbool $ in een van de kolommen 1 t/m 72 voor, dan wordt de rest van de onderhavige kaart (inclusief de $) geskipt, en gaat het systeem over op de volgende kaart zonder een symbool twnr in te lassen.

5.3.2. Voer.

Het voer moet op hetzelfde medium worden geponst als de monitorkop en het programma. Ook de code dient dezelfde te zijn als die van monitorkop en programma, tenzij het voer gelezen wordt met behulp van rehep (zie 4.1.1.).
Bij gebruik van ponsband is de eerste ponsing van het voer de ponsing die direct volgt op het laatste symbool van de programmatekst (dus het symbool volgend op end hoort niet bij het voer). Bij gebruik van ponskaarten is het eerste symbool van het voer de ponsing in kolom 1 van de kaart die volgt op de kaart die de laatste 'END' van het programma bevat. Van voerkaarten worden alle kolommen gelezen, terwijl bij aanroepen van de procedure resym na kolom 80 een symbool twnr (resymwaarde 119) ingelast wordt. Voor de indeling van voerkaarten en voerbanden wordt verwezen naar de beschrijving van de invoerprocedures in hoofdstuk 4.1..

5.4. De verwerking door de machine.

5.4.1. Eerst wordt de monitorkop ingelezen en op correctheid onderzocht. Tijdens dit inlezen wordt de tekst van de monitorkop over de regeldrukker afgedrukt. Bij het vaststellen van een fout in de monitorkop wordt een foutmelding gegeven van de volgende gedaante:

er < foutnummer >

De interpretatie van foutnummers is gegeven in tabel 7.1..
Zodra een foutmelding over een fout in de monitorkop is gegeven wordt de verwerking afgebroken; de programmatekst wordt in dit geval dus niet onderzocht en evenmin afgedrukt, wel wordt de rest van de monitorkop nog afgedrukt.

5.4.2. Wanneer de monitorkop in orde is bevonden, wordt het programma ingelezen en in enige scans op syntactische fouten getoetst. Tijdens het inlezen wordt, wanneer in de monitorkop niet de aanwijzing 'z' voorkomt, de tekst afgedrukt over de regeldrukker, waarbij iedere regel wordt voorafgegaan door het regelnummer, gevolgd door drie spaties. Iedere pagina op de regeldrukker begint met een kopje, vermeldende: de datum van verwerking, het door het systeem toegekende serienummer, de programmaletter, het opdrachtnummer, het programmanummer, de gebruikersnaam en tenslotte een paginanummer. Geheel rechts in het kopje bevindt zich het scheurnummer dat alleen voor interne doeleinden gebruikt wordt. Voor tekst en foutmeldingen staan 60 regels per pagina ter beschikking.

5.4.3. Zo nodig wordt het afdrukken van een regel onderbroken voor het afdrukken van foutmeldingen voor fouten welke reeds tijdens de inleesfase gedetecteerd worden. Voor syntactische fouten, gevonden in de latere scans, wordt een foutmelding afgedrukt volgend op de programmatekst.

Iedere melding van een fout tegen de syntax heeft de volgende gedaante:
er <foutnummer> <regelnummer> <laatst gelezen symbool>
<waarde laatst gelezen constante>
<eerste 8 karakters van laatst verwerkte identifier>
De interpretatie van de foutnummers is gegeven in tabel 7.2.. Voor <laatst gelezen symbool> wordt het symbool zelf of zijn interne representatie (zie tabel 6.1.) afgedrukt. Wordt b.v. het getal 82 afgedrukt dan is for het laatst gelezen symbool. Het regelnummer verwijst naar de regel van de ALGOL 60-tekst, waarin de fout gedetecteerd is. Hoewel in de foutmelding slechts 8 karakters van de laatst verwerkte identifier gegeven worden, doen voor het ALGOL 60-systeem voor de X8 alle letters en cijfers van een identifier mee. Sommige fouten in de ALGOL 60-tekst kunnen tot andere foutmeldingen leiden dan voor de hand zou liggen. Dit hangt samen met de wijze, waarop het syntactisch onderzoek van een tekst, die tengevolge van een fout min of meer oninterpretabel is, verder wordt voortgezet. Voorbeeld: door een fout in een declaratie kan de dubbele punt in een array-declaratie als afsluiting van een label opgevat worden; dit veroorzaakt dan onverwachte foutmeldingen.
Opmerking:
Indien nog geen symbool of identifier gelezen is, wordt de desbetreffende plaats blank gelaten. Indien nog geen getal gelezen is, wordt -0 afgedrukt. Integer labels worden hierbij als getallen beschouwd.

5.4.4. Slechts als bij bovengenoemd syntactisch onderzoek geen fouten gevonden zijn, wordt het programma uitgevoerd. Output over de regeldrukker begint op een nieuwe pagina; output over de bandponser, de kaartponser en de plotter wordt voorafgegaan door een standaardbegin en afgesloten door een standaardslot.

5.4.4.1. Het standaardbegin voor de bandponser bevat:
– serienummer, volgnummer en opdrachtnummer (alle drie in "leesbare" vorm), gevolgd door een stuk blank
– een "leesbaar kleiner dan teken", gevolgd door een stuk blank
– de ponsing erase (31,7) gevolgd door een stuk blank.
Het genoemde volgnummer is normaal gelijk aan 1, maar wanneer tijdens het ponsen de voorraad ongeponste band in de bandponser dreigt op te raken wordt een stuk blank geponst; op de nieuw ingelegde band wordt weer het standaardbegin geponst waarbij het volgnummer met 1 wordt verhoogd.
Het standaardslot voor de bandponser luidt:
< 100 blanks > < twnr of crlf > < stuk blank >

5.4.4.2. Het standaardbegin voor de kaartponser bestaat uit 1 kaart die in "leesbare" vorm het serienummer en het opdrachtnummer bevat met een geponste rand "boven" en "onder" langs de kaart over de volle breedte van de kaart. Het standaardslot voor de kaartponser bestaat uit 1 kaart met slechts een geponste rand "boven" en "onder" langs de kaart (in alle kolommen een 12–9–ponsing).

5.4.4.3. Het standaardbegin voor de plotter bestaat uit de datum, het serienummer en het opdrachtnummer, welke geplot worden langs een scheurlijn. Het standaardslot voor de plotter bestaat uit 40 cm paper–feed.

5.4.5. Bij het detecteren van een ongeoorloofde situatie in de executiefase van een programma wordt de uitvoering direct afgebroken; als laatste handelingen wordt een foutmelding gegeven over de regeldrukker en zonodig een standaardslot uitgevoerd. De normale vorm voor een foutmelding tijdens de executie is:
er <foutnummer> <regelnummer>
<waarde laatste door read of read1 afgeleverde getal>
Voor de interpretatie van de foutnummers wordt verwezen naar 7.3.. Het regelnummer verwijst naar de regel van de ALGOL 60–tekst, waarin het laatst in executie genomen maar onvoltooid gebleven statement of de laatste in executie genomen maar nog niet voltooide array–declaratie begint. Bij programma's met aanwijzing 'm' in de monitorkop wordt het regelnummer weggelaten.
Opmerking:
Indien door read of read1 nog geen getal is afgeleverd, wordt op de betreffende plaats –0 afgedrukt.

5.4.6. De uitvoering van een programma wordt beeindigd:
a) door het "passeren" van de laatste end,
b) bij aanroep van de bibliotheek–procedure exit,
c) bij detectie van een fout,
d) door een operateurs–ingreep.

In dit laatste geval wordt een foutmelding gegeven met foutnummer 999.

5.4.7. Aan de regeldrukker–output van elk programma waarvan de verwerking niet reeds bij de monitorkop is beeindigd, wordt door het systeem een extra pagina toegevoegd waarop de volgende gegevens zijn vermeld:

5.4.7.1. Als het programma syntactisch incorrect is:
op regel 1:
de tijd, besteed aan de syntactische controle, gevolgd door de administratief in rekening gebrachte tijd, beide in milli–uren,
op regel 2:
de som van de (met rehep gelezen) ponsingen van de monitorkop plus de programmatekst,
op regel 3:
een schatting van het aantal woorden in het geheugen van de X8, voor het programma zelf benodigd na verbetering van de fouten.

5.4.7.2. Als het programma syntactisch correct is, en ook in uitvoering genomen is:
op regel 1:
de tijd, besteed aan de syntactische controle, gevolgd door de totale tijd, die de X8 aan het programma besteed heeft, gevolgd door de administratief in rekening gebrachte tijd, deze tijden in milli–uren,
op regel 2:
de som van de (met rehep gelezen) ponsingen van de monitorkop plus de programmatekst, gevolgd door de som van alle ponsingen die bij de controle en de uitvoering zijn verwerkt,
op regel 3:
het aantal woorden door het programma zelf in het geheugen van de X8 in beslag genomen, gevolgd door het aantal woorden dat tijdens de executie voor het programma en zijn werkruimte benodigd was,
op regel 4:
indien het programma resultaten over de bandponser heeft uitgevoerd: op positie 1 t/m 18 het aantal geponste octaden, gevolgd door de som van de geponste octaden na het standaardbegin (dus inclusief die van het standaardslot). Indien het programma resultaten over de kaartponser heeft uitgevoerd: op positie 19 t/m 36 het aantal geponste kaarten gevolgd door de som van de rehepwaarden van alle geponste kolommen. In beide tellingen worden standaardbegin en standaardslot voor de kaartponser meegeteld. Indien het programma plotopdrachten heeft uitgevoerd: op positie 37 t/m 54 de gebruikte tekentijd in milliuren gevolgd door het aantal plotbewegingen.

5.5. Beperkingen.

5.5.1. Voor vertaling en uitvoering van ALGOL 60–programma's staan zeker 40000 geheugenplaatsen ter beschikking. In de vertaalfase moet deze geheugenruimte gedeeld worden door de vertaler en zijn werkruimte, de tekst van het programma en de code van het objectprogramma in opbouw. Tijdens de uitvoering van het programma bevinden zich in deze ruimte het objectprogramma, de variabelen, de arrays en de blokadministratie bijgehouden door het ALGOL–systeem. Voor het objectprogramma en eventueel toegevoegde bibliotheekprocedures samen staan hoogstens 24000 geheugenplaatsen ter beschikking. Indien tijdens de vertaling van een programma de beschikbare geheugenruimte uitgeput is, wordt de vertaling afgebroken met een foutmelding met foutnummer 492, 493 of 494. Indien tijdens de uitvoering meer geheugen benodigd is dan beschikbaar is, wordt de uitvoering afgebroken met een foutmelding met foutnummer 609.

5.5.2. Aan de ingewikkeldheid van een programma worden door de vertaler bepaalde, overigens zeer ruime, beperkingen opgelegd. Slechts onder uitzonderlijke omstandigheden (b.v. gecompliceerde programma's gegenereerd door andere programma's) kan men te maken krijgen met deze begrenzingen, en dan nog redelijkerwijze alleen op de volgende punten:

5.5.2.1. Het aantal entries in een switch–list is begrensd. Als aanduiding van orde van grootte zij vermeld dat een switch in het buitenste blok, die alleen labels bevat, ruim 220 entries mag bevatten; dit aantal zal voor een switch in een binnenblok of voor een switch die ingewikkelde designational expressions bevat, lager zijn.

5.5.2.2. Het aantal parameters in een procedure–aanroep is begrensd. De begrenzing ligt bij ongeveer dezelfde getallen als voor een switch (zie 5.5.2.1.).

5.5.2.3. Aan de nesting van blokken zijn grenzen gesteld. Men zij er met nadruk op gewezen dat deze beperking alleen geldt tijdens de vertaling, en dan nog alleen voor blokken en niet voor compound statements; tijdens uitvoering kunnen (door middel van recursieve procedures) net zoveel blokken worden geopend als het beschikbare geheugen toelaat. Het aantal blokken dat de vertaler binnengegaan is en nog niet verlaten heeft mag maximaal ongeveer 15 bedragen; dit aantal zal uiteraard teruglopen indien de blokken zelf weer binnen ingewikkelde constructies voorkomen.

5.5.2.4. Het totale aantal blokken en proceduredeclaraties, genest en niet–genest, is begrensd tijdens vertaling. De begrenzing ligt, indien in het programma slechts parallele blokken voorkomen, die elk een (1) declaratie van een korte identifier bevatten, bij ongeveer 800 en zal in andere gevallen lager zijn.

Bij optreden van een van de situaties 5.5.2.1., 5.5.2.2. of 5.5.2.3. wordt de vertaling onmiddellijk beeindigd met een foutmelding met foutnummer 491.
Bij het bereiken van de grens uit 5.5.2.4. wordt een foutmelding met foutnummer 131, 132, 133 of 134 gegeven.

6. Codes.

6.1. Voor de representatie van ALGOL-symbolen binnen programma's zijn twee "stijlen" toegestaan, die vooral verschillen in de voorstelling van de zogenaamde word delimiters; in de onderstreep-stijl worden deze voorgesteld als onderstreepte letterwoorden, b.v. begin, terwijl ze in de apostrof-stijl voorgesteld worden door letterwoorden, voorafgegaan en gevolgd door een apostrof, b.v. 'begin'. Het systeem eist, dat een eenmaal gekozen stijl consequent wordt voortgezet; aan de hand van de eerste begin van het programma wordt uitgemaakt in welke stijl het programma genoteerd staat.
Voorkeursnotatie en alternatieven mogen door elkaar worden gebruikt, waarbij de voorkeursnotatie aanbeveling verdient omdat deze zo nauw mogelijk aansluit bij de voorstelling in de reference language.

6.1.1. Hieronder volgt de voorstelling van de basic-symbols in de twee representatiestijlen; de getallen gevolgd door een haakje-sluiten verwijzen naar opmerkingen in 6.1.2..

reference language symbool	interne repre-sen-tatie	onderstreep-stijl 2)4) voorkeurs-notatie	onderstreep-stijl 2)4) alterna-tieven	apostrof-stijl 2)5) voorkeurs-notatie	apostrof-stijl 2)5) alterna-tieven
0 t/m 9	0 t/m 9	0 t/m 9		0 t/m 9	
a t/m z	10 t/m 35	a t/m z	A t/m Z 1)	a t/m z	A t/m Z 1)
A t/m Z	37 t/m 62	A t/m Z		A t/m Z	
+	64	+		+	
-	65	-		-	
×	66	×	times	×	'times'
/	67	/		/	
÷	68	:	div	:	'div'
					'/' 3)
↑	69	↑	↑ 3)	↑	↑ 3)
			××		××
			power		'power'
=	70	=	equal	=	'equal'
			eq		'eq'
≠	71	≠	notequal	≠	'notequal'
			ne		'ne'
			nq		'nq'
<	72	<	less	<	'less'
			lt		'lt'
			ls		'ls'
≤	73	≤	notgreater	≤	'notgreater'
			le		'le'
			lq		'lq'

reference language symbool	interne representatie	onderstreep-stijl voorkeurs-notatie	onderstreep-stijl alterna-tieven	apostrof-stijl voorkeurs-notatie	apostrof-stijl alterna-tieven
>	74	>	greater	>	'greater'
			gt		'gt'
			gr		'gr'
≥	75	≥	notless	≥	'notless'
			ge		'ge'
			gq		'gq'
¬	76	¬	not	¬	'not'
≡	77	=	equiv	=	'equiv'
			eqv		'eqv'
⊃	78	⊐	↓	⊐	↓
			impl		'impl'
			imp		'imp'
∨	79	∨	or	∨	'or'
∧	80	∧ 4)	and	∧ 5)	'and'
goto	81	goto		'goto'	
for	82	for		'for'	
step	83	step		'step'	
until	84	until		'until'	
while	85	while		'while'	
do	86	do		'do'	
,	87	,		,	
.	88	.		.	
10	89	10	e 6)	10	e 6)
:	90	:	.. 3)	:	.. 3)
;	91	;	.; 3)	;	.; 3)

reference language symbol	interne representatie	onderstreep-stijl voorkeurs-notatie	onderstreep-stijl alternatieven	apostrof-stijl voorkeurs-notatie	apostrof-stijl alternatieven
:=	92	:= [3]	..= [3]	:= [3]	..= [3]
			.= [3]		.= [3]
␣	93	spatie		spatie	
if	94	if		'if'	
then	95	then		'then'	
else	96	else		'else'	
comment	97	comment	co	'comment'	'co'
(	98	(		(	
)	99	)		)	
[	100	[	(/ [3]	[	(/ [3]
]	101	]	/) [3]	]	/) [3]
‘	102	≮		'(' [3]	" [7]
					'' [8]
’	103	≯		')' [9]	" [7]
					'' [8]
begin	104	begin	bgn	'begin'	'bgn'
end	105	end		'end'	
own	106	own		'own'	
real	107	real		'real'	
integer	108	integer [2]	int	'integer' [2]	'int'
Boolean	109	boolean	bool	'boolean'	'bool'
string	110	string		'string'	
array	111	array	ar	'array'	'ar'
procedure	112	procedure	proc	'procedure'	'proc'
switch	113	switch		'switch'	

reference language symbool	interne representatie	onderstreep-stijl voorkeurs-notatie	onderstreep-stijl alternatieven	apostrof-stijl voorkeurs-notatie	apostrof-stijl alternatieven
label	114	label		'label'	
value	115	value	val	'value'	'val'
true	116	true		'true'	
false	117	false		'false'	
	118	tab		tab	
	119	twnr		twnr	
	120	'	10)	'	11)
	121	"	10)		
	122	?	10)	?	10)
	123	&	10)	&	10)
	125	#		#	
	126	_		_	
	127	|	10)	|	10)
	128	@	10)	@	10)
	129	!	10)	!	10)
	132	%	12)	%	12)
	133	$	13)	$	13)
	134	CR	13)	CR	13)
	135	LF	14)	LF	
	160	onbekende doorbalking	14)		
	161	onbekende onderstreping			
	162	onbekend onderstreept letterwoord		onbekend letterwoord tussen apostrofs	

6.1.2. Opmerkingen bij de tabel 6.1.1.:

1) Indien de gebruikte code slechts een kast letters toelaat stellen deze letters altijd de kleine letters voor.
2) De case van de letters doet er bij geen der woordsymbolen toe; zo is times ('times') gelijkwaardig met TIMES ('TIMES') en met Times ('Times').
3) Dit uit meer dan een karakter opgebouwde symbool moet als een ondeelbaar geheel worden beschouwd; dat wil zeggen dat spaties binnen het symbool niet toegestaan zijn. De opbouw tot een basic symbol geschiedt slechts buiten strings. Binnen strings stellen de samenstellende karakters uitsluitend zich zelf voor; dit geldt ook voor string-quotes (zie 3.3.1.).
4) In de onderstreep-stijl worden begin en einde van een woordsymbool bepaald door begin en einde van de onderstreping. Binnen een woordsymbool mogen onderstreepte spaties voorkomen, zo is not equal ekwivalent met notequal. De eerste letter van een woordsymbool mag niet worden voorafgegaan door onderstreepte spaties. Twee opeenvolgende woordsymbolen moeten door tenminste een (niet onderstreept) layout-symbool (spatie, tab of nieuwe regel) worden gescheiden, b.v. integer procedure.
5) In de apostrof-stijl zijn spaties binnen een woordsymbool toegestaan, zo is 'not equal' ekwivalent met 'notequal'. Wel moet na de openings-apostrof onmiddelijk een letter volgen.
6) De interpretatie van e als $_{10}$ vindt slechts inwendig in "unsigned numbers" plaats (dus a fortiori buiten strings en commentaar). Een "unsigned number" mag niet met e beginnen, het getal $_{10}8$ mag dus niet als e+8 of e8, wel als 1e8 of 1.0e+8 worden genoteerd. Het symbool E wordt nooit als $_{10}$ geinterpreteerd.
7) Buiten strings heeft het aanhalingsteken " de betekenis: quote; binnen strings betekent het: unquote.
8) Buiten strings heeft '' (twee apostrofs, niet gescheiden door enig ander symbool) de betekenis: quote; binnen strings betekent het: unquote.
9) Dit uit meer dan een karakter opgebouwde symbool moet als een geheel worden beschouwd; dit betekent dat spaties binnen het symbool niet toegestaan zijn. Opbouw tot het basic symbol "unquote" vindt slechts plaats binnen strings.
10) Uitsluitend binnen strings en commentaar toegestaan.
11) Zie 6.1.5..
12) In ALGOL 60 teksten op ponskaarten heeft een $ in een van de kolommen 1 t/m 72 de betekenis: skip de rest van de kaart en lees verder op de volgende kaart, zonder overgang op deze nieuwe kaart te interpreteren als terug-wagen-nieuwe-regel.
13) In ALGOL 60 teksten wordt het symbool LF geinterpreteerd als terug-wagen-nieuwe-regel en wordt CR overal geskipt.
14) Zie 6.1.4..

6.1.3. Onderstreping en doorbalking.

Woordsymbolen in onderstreep-stijl worden verkregen door alle elementen van het woord (letters en spaties) afzonderlijk te onderstrepen. Hierbij mag elk element meer dan eenmaal onderstreept worden.
Symbolen, die worden voorgesteld als een onderstreept ander symbool (:, <, >, =, ¬) worden verkregen door eerst te ponsen, onmiddelijk gevolgd door het te onderstrepen symbool (:, <, >, =, ¬). Ook hierbij is meervoudig onderstrepen toegestaan.
Symbolen, die worden voorgesteld als een doorbalkt ander symbool (↑, ≠, ≮, ≯) worden verkregen door eerst | te ponsen (eventueel meer dan eens), onmiddelijk gevolgd door het te doorbalken symbool (∧, =, <, >).
De opbouw tot basic symbol treedt binnen strings niet op.

6.1.4. Onbekende onderstrepingen en doorbalkingen.
Als een onderstreping of doorbalking samen met het er op volgende karakter een symbool geeft dat niet in de lijst in 6.1.1. voorkomt, ontstaat een onbekende onderstreping of doorbalking. Deze zijn toegestaan binnen commentaar en kunnen daar nooit tot het beeindigen van het commentaar leiden. De symbolen ; en ; werken dus niet als sluitsymbolen van het comment-symbool, zulks in tegenstelling tot ; en ., .

De opbouw tot onbekende onderstreping of doorbalking treedt niet op binnen strings.

6.1.5. Apostrofs binnnen strings en commentaar.

6.1.5.1. In onderstreep-stijl zijn apostrofs binnen strings en commentaar onbeperkt toegestaan.

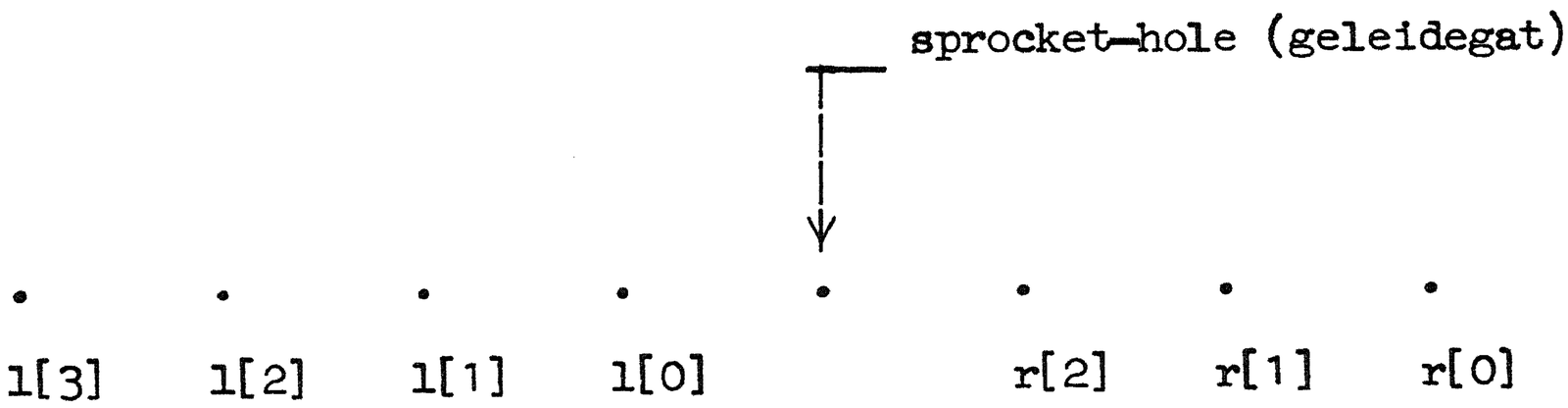
6.1.5.2. In apostrof-stijl wordt het gebruik van apostrofs binnen commentaar ontraden; de preciese regels zijn omslachtig, overtreding van de regels levert een foutmelding met foutnummer 108.
Een apostrof mag binnen een string niet onmiddellijk gevolgd worden door een tweede apostrof.
Een string die eindigt op een apostrof mag alleen door een aanhalingsteken (") afgesloten worden.

6.1.6. Aanhalingstekens (") binnen strings en commentaar.
Aanhalingstekens in apostrof-stijl binnen strings worden als unquotes opgevat; overal elders binnen strings en commentaar zijn ze onbeperkt toegestaan.

6.1.7. Vraagtekens binnen strings en commentaar.
Vraagtekens zijn in elke representatie-stijl overal binnen strings en commentaar onbeperkt toegestaan.

6.2. MC–flexowritercode.

6.2.1. De MC–flexowritercode is een 7–spoorscode die van de 128 mogelijke bandbeelden slechts de helft toelaat, namelijk die ponsingen waarin een oneven aantal gaatjes voorkomt (ponsingen met oneven pariteit). Bij het merendeel der ponsingen is de betekenis case–afhankelijk, dwz. afhankelijk van de laatst gegeven case–definitie (lower case dan wel upper case).
In de tabel wordt iedere ponsing aangegeven door een getallenpaar <l,r>, de bandcode genaamd, dat als volgt uit het bandbeeld wordt verkregen. We stellen de gaatjes links van de sprocket–hole voor door l[0:3] en de gaatjes rechts van de sprocket–hole door r[0:2]:



De waarde van l[i] en r[i] is 1 of 0 al naar gelang op de corresponderende plaats in de band een gat geponst is of niet.
We krijgen l en r nu uit:

l:= 8 × l[3] + 4 × l[2] + 2 × l[1] + l[0] en
r:= 4 × r[2] + 2 × r[1] + r[0].

Voor l en r geldt dus: $0 \leq l \leq 15$ en $0 \leq r \leq 7$.
Voorbeeld:

ponsing	bandcode
. . . 0 . . 0 0	1,3
0 0 0	12,1
0 0 0 0 . 0 0 0	15,7

In de hierna volgende tabel staat (waar nodig gesplitst naar upper case en lower case) vermeld:
- de bandcode
- het bijbehorende symbool
- de afgeleverde getalwaarde wanneer het symbool met behulp van resym wordt gelezen
- het karakter dat op de regeldrukker verschijnt wanneer een door resym gelezen symbool met behulp van prsym wordt afgedrukt.

6.2.2.1. Case - afhankelijke symbolen.

band-code	lower case symbool	resym	prsym	upper case symbool	resym	prsym
4,0	0	0	0	∧	80	∧
0,1	1	1	1	∨	79	∨
0,2	2	2	2	×	66	×
2,3	3	3	3	/	67	/
0,4	4	4	4	=	70	=
2,5	5	5	5	;	91	;
2,6	6	6	6	[	100	[
0,7	7	7	7	]	101	]
1,0	8	8	8	(	98	(
3,1	9	9	9	)	99	)
12,1	a	10	A	A	37	A
12,2	b	11	B	B	38	B
14,3	c	12	C	C	39	C
12,4	d	13	D	D	40	D
14,5	e	14	E	E	41	E
14,6	f	15	F	F	42	F
12,7	g	16	G	G	43	G
13,0	h	17	H	H	44	H
15,1	i	18	I	I	45	I
10,1	j	19	J	J	46	J
10,2	k	20	K	K	47	K
8,3	l	21	L	L	48	L
10,4	m	22	M	M	49	M
8,5	n	23	N	N	50	N
8,6	o	24	O	O	51	O
10,7	p	25	P	P	52	P

band-code	lower case symbool	resym	prsym	upper case symbool	resym	prsym
11,0	q	26	Q	Q	53	Q
9,1	r	27	R	R	54	R
6,2	s	28	S	S	55	S
4,3	t	29	T	T	56	T
6,4	u	30	U	U	57	U
4,5	v	31	V	V	58	V
4,6	w	32	W	W	59	W
6,7	x	33	X	X	60	X
7,0	y	34	Y	Y	61	Y
5,1	z	35	Z	Z	62	Z
14,0	+	64	+	"	121	"
8,0	-	65	-	¬	76	¬
6,1	<	72	<	>	74	>
11,3	,	87	,	?	122	?
13,3	.	88	.	:	90	:
7,3	⏨	89	⏨	'	120	'
1,6	_	126	_	|	127	|

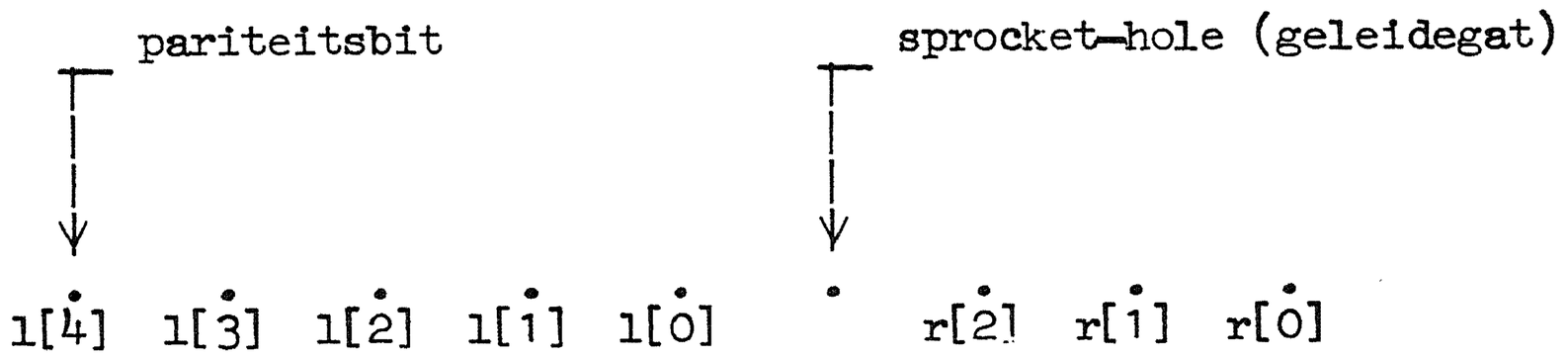
6.2.2.2. Case-onafhankelijke symbolen.

bandcode	symbool	resym	prsym
2,0	spatie	93	spatie
7,6	tab	118	tab
3,2	twnr	119	twnr
15,2	lower case	skip	
15,4	upper case	skip	
15,7	erase	skip	
1,3	stopcode	skip	
5,2	backspace	skip	
5,7	punch-off	skip	
0,0	tape feed	skip	

6.2.2.3. Tape feed is het enige toegelaten symbool van even pariteit.

6.3. ISO - code (ERC - MC versie)

6.3.1. De ISO-code is een 7-spoorscode (aangevuld met een achtste als pariteitsspoor). Alle 128 mogelijke gaatjescombinaties van de 7 informatiesporen hebben een betekenis, die niet van de voorgeschiedenis afhangt (geen case of shift). Het achtste spoor zorgt voor even pariteit.
In de tabel wordt iedere ponsing aangegeven door een getallenpaar <l,r>, de bandcode genaamd, dat als volgt uit het bandbeeld wordt verkregen. We stellen de gaatjes links van de sprocket-hole voor door l[0:4] en de gaatjes rechts van de sprocket-hole door r[0:2]:



De waarde van l[i] en r[i] is 1 of 0 al naar gelang op de corresponderende plaats in de band een gat geponst is of niet. We krijgen l en r nu uit:

$l := 8 \times l[3] + 4 \times l[2] + 2 \times l[1] + l[0]$ en
$r := 4 \times r[2] + 2 \times r[1] + r[0]$.

De pariteitsbit l[4] blijft dus buiten beschouwing, terwijl voor l en r geldt:

$$0 \leq l \leq 15 \text{ en } 0 \leq r \leq 7.$$

Ponsingen met $0 \leq l \leq 3$ worden control characters genoemd.

Voorbeelden:

ponsing	bandcode
. 0 0 0 . . 0 . .	14,4
0 . . 0 0 . 0 0 0	3,7 (de pariteitsbit wordt niet beschouwd)
0 0 0 0 0 . 0 0 0	15,7

In de hierna volgende tabel staat vermeld:

- de bandcode
- het bijbehorende symbool
  en voor die symbolen waaraan een interne representatie is toegevoegd:
- de afgeleverde getalwaarde wanneer het symbool met behulp van resym wordt gelezen
- het karakter dat op de regeldrukker verschijnt wanneer een door resym gelezen symbool met behulp van prsym wordt afgedrukt.

6.3.2.1. Symbolen waaraan een interne representatie is toegevoegd:

bandcode	symbool	resym	prsym
6,0	0	0	0
6,1	1	1	1
6,2	2	2	2
6,3	3	3	3
6,4	4	4	4
6,5	5	5	5
6,6	6	6	6
6,7	7	7	7
7,0	8	8	8
7,1	9	9	9
12,1	a	10	A
12,2	b	11	B
12,3	c	12	C
12,4	d	13	D
12,5	e	14	E
12,6	f	15	F
12,7	g	16	G
13,0	h	17	H
13,1	i	18	I
13,2	j	19	J
13,3	k	20	K
13,4	l	21	L
13,5	m	22	M
13,6	n	23	N
13,7	o	24	O
14,0	p	25	P

bandcode	symbool	resym	prsym
14,1	q	26	Q
14,2	r	27	R
14,3	s	28	S
14,4	t	29	T
14,5	u	30	U
14,6	v	31	V
14,7	w	32	W
15,0	x	33	X
15,1	y	34	Y
15,2	z	35	Z
8,1	A	37	𝔄
8,2	B	38	𝔅
8,3	C	39	ℭ
8,4	D	40	𝔇
8,5	E	41	𝔈
8,6	F	42	𝔉
8,7	G	43	𝔊
9,0	H	44	ℌ
9,1	I	45	ℑ
9,2	J	46	𝔍
9,3	K	47	𝔎
9,4	L	48	𝔏
9,5	M	49	𝔐
9,6	N	50	𝔑
9,7	O	51	𝔒
10,0	P	52	𝔓
10,1	Q	53	𝔔

bandcode	symbool	resym	prsym
10,2	R	54	R
10,3	S	55	S
10,4	T	56	T
10,5	U	57	U
10,6	V	58	V
10,7	W	59	W
11,0	X	60	X
11,1	Y	61	Y
11,2	Z	62	Z
5,3	+	64	+
5,5	—	65	—
5,2	×	66	×
5,7	/	67	/
11,6	↑	69	↑
7,5	=	70	=
7,4	<	72	<
7,6	>	74	>
15,6	⌝	76	⌝
12,0	↓	78	⌝
15,5	∨	79	∨
15,3	∧	80	∧
5,4	,	87	,
5,6	.	88	.
11,4	₁₀	89	₁₀
7,2	:	90	:
7,3	;	91	;
5,0	(	98	(

bandcode	symbool	resym	prsym
5,1	)	99	)
11,3	[	100	[
11,5	]	101	]
4,7	'	120	'
4,2	"	121	"
7,7	?	122	?
4,6	&	123	?
4,3	#	125	#
11,7	_	126	_
15,4	|	127	|
8,0	@	128	\
4,1	!	129	?
4,5	%	132	↑
4,4	$	133	$
4,0	SP	93	spatie
1,1	HT	118	tab
1,5	CR	134	cr
1,2	LF	135	lf

6.3.2.2. Symbolen waaraan geen interne representatie is toegevoegd en die door resym worden geskipt:

band-code	symbool	band-code	symbool
0,0	NUL	2,2	DC2
0,1	SOH	2,3	reader-off
0,2	STX	2,4	DC4
0,3	ETX	2,5	NAK
0,4	EOT	2,6	SYN
0,5	ENQ	2,7	ETB
0,6	ACK	3,0	CAN
0,7	BEL	3,1	EM
1,0	BS	3,2	SUB
1,3	VT	3,3	ESC
1,4	FF	3,4	FS
1,6	SO	3,5	GS
1,7	SI	3,6	RS
2,0	DLE	3,7	US
2,1	DC1	15,7	DEL

6.4. Ponskaartencode (IBM - EL versie).

6.4.1. Een ponskaart bestaat uit 80 kolommen, die elk 12 plaatsen bevatten. De plaatsen in een kolom zijn van boven naar beneden als volgt genummerd: 12, 11, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Iedere ponsing is zelfstandig en duidt een symbool aan; ook onderstreping en doorbalking beslaan een kolom. Van de 4096 mogelijke gaatjescombinaties hebben er in de IBM-EL versie slechts 64 een betekenis.
De 26 letters komen slechts in enkelvoud voor en worden geinterpreteerd als kleine letters.
Voor een ALGOL 60-tekst tellen uitsluitend de kolommen 1 t/m 72 mee; kolommen 73 t/m 80 kunnen worden gebruikt voor nummering of andere identificatie. Na kolom 72 wordt aan een nieuwe kaart begonnen; de tekst hiervan wordt geacht op een nieuwe regel te beginnen (dwz. de interne representatie van twnr wordt tussengevoegd). Indien echter in een van de kolommen 1 t/m 72 het symbool $ voorkomt wordt de rest van de kaart (inclusief de $) overgeslagen zonder twnr in te lassen.
Van voerkaarten worden alle kolommen (1 t/m 80) gelezen; bij gebruik van de procedure resym wordt na kolom 80 het symbool twnr (resym-waarde 119) ingelast. Het symbool $ wordt normaal gelezen en heeft de resymwaarde 133.
In de volgende tabel wordt steeds aangegeven:

- de kaartcode die aangeeft in welke positie(s) van een kolom een gat is geponst; zo betekent de kaartcode 11-2-8 dat er gaten geponst zijn in posities 11, 2 en 8
- het bijbehorende symbool
- de getalwaarde die wordt verkregen door de kolom met resym te lezen
- het symbool dat op de regeldrukker verschijnt wanneer een door resym gelezen kolom door middel van prsym wordt afgedrukt.

6.4.2.

kaartcode	symbool	resym	prsym
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
12–1	A	10	A
12–2	B	11	B
12–3	C	12	C
12–4	D	13	D
12–5	E	14	E
12–6	F	15	F
12–7	G	16	G
12–8	H	17	H
12–9	I	18	I
11–1	J	19	J
11–2	K	20	K
11–3	L	21	L
11–4	M	22	M
11–5	N	23	N
11–6	O	24	O
11–7	P	25	P
11–8	Q	26	Q
11–9	R	27	R

kaartcode	symbool	resym	prsym
0-2	S	28	S
0-3	T	29	T
0-4	U	30	U
0-5	V	31	V
0-6	W	32	W
0-7	X	33	X
0-8	Y	34	Y
0-9	Z	35	Z
12	^	80	^
12-2-8	]	101	]
12-3-8	.	88	.
12-4-8	<	72	<
12-5-8	(	98	(
12-6-8	+	64	+
12-7-8	|	127	|
11	-	65	-
11-2-8	[	100	[
11-3-8	$	133	$
11-4-8	×	66	×
11-5-8	)	99	)
11-6-8	;	91	;
11-7-8	¬	76	¬
0-1	/	67	/
0-2-8	¤	89	¤
0-3-8	,	87	,
0-4-8	%	132	↑
0-5-8	_	126	_
0-6-8	>	74	>

kaartcode	symbool	resym	prsym
0-7-8	?	122	?
2-8	:	90	:
3-8	#	125	#
4-8	V	79	V
5-8	'	120	'
6-8	=	70	=
7-8	"	121	"
blank	spatie	93	spatie

6.4.3. De 64 ponsingen van de IBM-EL-kaartcode kunnen verdeeld worden in 4 groepen van elk 16 symbolen. Deze vier groepen onderscheiden zich door de ponsingen op de plaatsen 12, 11 en 0; op deze drie plaatsen kan 0 of 1 ponsing voorkomen. De 16 symbolen binnen elke groep onderscheiden zich door de ponsingen op de plaatsen 1, 2, 3, 4, 5, 6, 7, 8 en 9. Op deze plaatsen kunnen 0, 1 of 2 ponsingen voorkomen; komen er 2 ponsingen voor, dan bevindt zich een ponsing op plaats 8. In tabel 6.4.4. worden de symbolen volgens bovenstaande classificatie weergegeven.

6.4.4.

	-	12	11	0
-	space	^	-	0
1	1	A	J	/
2	2	B	K	S
3	3	C	L	T
4	4	D	M	U
5	5	E	N	V
6	6	F	O	W
7	7	G	P	X
8	8	H	Q	Y
9	9	I	R	Z
2-8	:	]	[	¤
3-8	#	.	$	,
4-8	∨	<	×	%
5-8	'	(	)	_
6-8	=	+	;	>
7-8	"	|	¬	?

6.5. Effect van aanroepen van pusym, csym, prsym en plotsym.

6.5.1. In de kolom 'waarde' zijn de "interessante" waarden van de actuele parameter opgenomen.

In de kolom 'pusym flex-code' zijn de symbolen vermeld die, afgezien van ingelaste case-ponsingen, op de bandponser worden geponst bij een aanroep van pusym (zie 4.2.2.2.), wanneer de heersende code (4.2.1.) de MC-flexowriter-code is; deze bandcode staat beschreven in 6.2.. Voor niet opgenomen waarden wordt het symbool ? geponst, zonodig voorafgegaan door een upper-case ponsing.

In de kolom 'pusym ISO-code' zijn de symbolen vermeld die op de bandponser worden geponst bij een aanroep van pusym (zie 4.2.2.2.), wanneer de heersende code (4.2.1.) de ISO-code is; deze code staat beschreven in 6.3.. Voor niet opgenomen waarden wordt het symbool ? geponst.

In de kolom 'csym' zijn de symbolen vermeld die op de kaartponser worden geponst bij een aanroep van csym (zie 4.5.2.2.). Voor de kaartcode zie 6.4.. Voor niet opgenomen waarden wordt het symbool ? geponst.

In de kolom 'prsym' is aangegeven wat op de regeldrukker wordt afgebeeld bij een aanroep van prsym (zie 4.3.2.1.). Voor niet opgenomen waarden wordt het symbool ? afgedrukt.
Verder zij er op gewezen dat het tekenassortiment op de regeldrukker aanzienlijk uitgebreid kan worden door gebruik te maken van doorbalkingen en "overprintingen" (zie 4.3.2.5.). Als voorbeelden kunnen genoemd worden: psi, uit balk en u; phi, uit balk en 0; kruis, uit deelstreep en backslash (128); vierkant met kruis, uit n en z; klinkers met umlaut, met aanhalingsteken (121).

In de kolom 'plotsym' is aangegeven wat op de plotter wordt afgebeeld bij een aanroep van plotsym (zie 4.6.2.2.). Voor niet opgenomen waarden wordt het symbool ? afgebeeld. De in deze tabel getoonde lettergrootte en italiciteit zijn die welke ontstaan door een aanroep van shape(0,28,0) (zie 4.6.1.5.).

6.5.2.

waarde	pusym flex-code	pusym ISO-code	csym	prsym	plotsym
0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4
5	5	5	5	5	5
6	6	6	6	6	6
7	7	7	7	7	7
8	8	8	8	8	8
9	9	9	9	9	9
10	a	a	A	A	a
11	b	b	B	B	b
12	c	c	C	C	c
13	d	d	D	D	d
14	e	e	E	E	e
15	f	f	F	F	f
16	g	g	G	G	g
17	h	h	H	H	h
18	i	i	I	I	i
19	j	j	J	J	j
20	k	k	K	K	k
21	l	l	L	L	l
22	m	m	M	M	m
23	n	n	N	N	n
24	o	o	O	O	o
25	p	p	P	P	p
26	q	q	Q	Q	q

waarde	pusym flex–code	pusym ISO–code	csym	prsym	plotsym
27	r	r	R	R	r
28	s	s	S	S	s
29	t	t	T	T	t
30	u	u	U	U	u
31	v	v	V	V	v
32	w	w	W	W	w
33	x	x	X	X	x
34	y	y	Y	Y	y
35	z	z	Z	Z	z
37	A	A	A	A	A
38	B	B	B	B	B
39	C	C	C	C	C
40	D	D	D	D	D
41	E	E	E	E	E
42	F	F	F	F	F
43	G	G	G	G	G
44	H	H	H	H	H
45	I	I	I	I	I
46	J	J	J	J	J
47	K	K	K	K	K
48	L	L	L	L	L
49	M	M	M	M	M
50	N	N	N	N	N
51	O	O	O	O	O
52	P	P	P	P	P
53	Q	Q	Q	Q	Q
54	R	R	R	R	R

waarde	pusym flex-code	pusym ISO-code	csym	prsym	plotsym
55	S	S	S	S	S
56	T	T	T	T	T
57	U	U	U	U	U
58	V	V	V	V	V
59	W	W	W	W	W
60	X	X	X	X	X
61	Y	Y	Y	Y	Y
62	Z	Z	Z	Z	Z
64	+	+	+	+	+
65	−	−	−	−	−
66	×	×	×	×	*
67	/	/	/	/	/
68	_:	_:	_:	÷	÷
69	|∧	↑	|∧	↑	↑
70	=	=	=	=	=
71	|=	|=	|=	≠	≠
72	<	<	<	<	<
73	_<	_<	_<	≤	≤
74	>	>	>	>	>
75	_>	_>	_>	≥	≥
76	⌐	⌐	⌐	⌐	¬
77	_=	_=	_=	≡	≡
78	_⌐	↓	_⌐	⌐	⊐
79	∨	∨	∨	∨	∨
80	∧	∧	∧	∧	∧
87	,	,	,	,	,
88	.	.	.	.	.

waarde	pusym flex-code	pusym ISO-code	csym	prsym	plotsym
89	▯	▯	▯	▯	▪
90	:	:	:	:	:
91	;	;	;	;	;
93	spatie	spatie	spatie	spatie	spatie
98	(	(	(	(	(
99	)	)	)	)	)
100	[	[	[	[	[
101	]	]	]	]	]
102	|<	|<	|<	⊀	⊀
103	|>	|>	|>	⊁	⊁
118	tab	HT	tab	tab	tab
119	twnr	CR LF	nieuwe kaart	twnr	twnr
120	'	'	'	'	'
121	"	"	"	"	"
122	?	?	?	?	?
123	?	&	?	?	?
124	?	?	?	°	°
125	?	#	#	#	#
126	_	_	_	_	_
127	|	|	|	|	|
128	?	@	?	\	\
129	¶	!	?	?	?
132	¶	%	%	¶	%
133	|S	$	$	$	$
134	skip	CR	skip	CR	↵
135	twnr	LF	nieuwe kaart	LF	↓

waarde	pusym flex-code	pusym ISO-code	csym	prsym	plotsym
136	?	?	?	?	+
137	?	?	?	?	×
138	?	?	?	?	✖
139	?	?	?	?	✖
140	?	?	?	?	Y
141	?	?	?	?	⧗
142	?	?	?	?	⧗
143	?	?	?	?	◻
144	?	?	?	?	◆
145	?	?	?	?	*backspace*
146	?	?	?	?	π

7. Omschrijving van de betekenis van foutnummers.

7.1. Fouten in de monitorkop.

10 symbool van foute pariteit in de ponsband
11 ontoelaatbare ponsing
12 monitorkop is te lang
14 in ponsband ontbreekt de twnr vooraan
15 programmaletter incorrect
16 opdrachtnummer incorrect
17 punt na opdrachtnummer ontbreekt
18 programmanummer > 1000
21 na programmanummer ontbreekt de komma
22 na komma geen letter b,k,m,p,r of t waar vereist
23 na naam of aanwijzing geen komma of twnr
24 geen integer waar vereist
25 eerste karakter van naam is geen letter
26 meer ponsbanduitvoer gevraagd dan toegestaan
27 meer kaartuitvoer gevraagd dan toegestaan
28 meer plotteruitvoer gevraagd dan toegestaan
29 meer regeldrukkeruitvoer gevraagd dan toegestaan
30 meer rekentijd gevraagd dan toegestaan
99 verwerking afgebroken wegens gebrek aan verdere input

7.2. Fouten tegen de syntaxis.

100 in parameterscheider ontbreekt de colon
101 in parameterscheider ontbreekt de (
102 op de band staat een symbool van foute pariteit
103 op het invoermedium komt een onbekende ponsing voor
104 in declaratie wordt own niet gevolgd door <type>
105 in declaratie wordt own niet gevolgd door type- of arraydeclaratie
106 in declaratie wordt <type> gevolgd door switch
107 in specificatie wordt <type> gevolgd door label of switch
108 in apostrof-stijl volgt na apostrof en letter(s) geen sluit-accent
109 na $_{10}$ of . volgt geen getal
110 geen identifier waar vereist
111 programma begint met identifier, niet gevolgd door colon
112 programma begint met getal, niet gevolgd door colon
113 programma is geen compound statement of block
114 in procedure declaratie is de formele parameterlijst niet afgesloten met)
115 in procedure declaratie volgt op het <formal parameter part> geen semicolon
116 in value list staat een identifier die niet onder de formelen voorkomt
117 value list niet afgesloten door een semicolon
118 in specificatie staat een identifier die niet onder de formelen voorkomt
119 formele parameter wordt meer dan eens gespecificeerd
120 specificatie ontoelaatbaar voor parameter uit value list
121 specificatie niet door semicolon afgesloten
122 in typedeclaratie komt een reeds eerder in hetzelfde blok gedeclareerde identifier voor

123 in arraydeclaratie komt een reeds eerder in hetzelfde blok gedeclareerde identifier voor
124 in arraydeclaratie deugt de bound pair list niet
125 in arraydeclaratie ontbreekt de bound pair list
126 bij switchdeclaratie komt een reeds eerder in hetzelfde blok gedeclareerde identifier voor
127 bij proceduredeclaratie komt een reeds eerder in hetzelfde blok gedeclareerde identifier voor
128 declaratie niet door semicolon afgesloten
129 label in hetzelfde blok reeds eerder gedeclareerd of als label voor statement verschenen
130 numerieke label buiten de integercapaciteit of niet-integer
131/132/133/134 programma bevat teveel gedeclareerde namen en labels
199 invoer uitgeput voordat de afsluitende end gevonden is

200 structuurfout in declaratie
201 formele parameter uit value list niet gespecificeerd
202 te veel localen of labels in blok
203 in arraydeclaratie deugt de bound pair list niet
204 identifier onbekend

300 in aritmetische expressie is een ifclause niet besloten met then
301 in aritmetische expressie ontbreekt een else-deel
302 in aritmetische expressie ontbreekt een)
303 in aritmetische expressie begint een primary met ontoelaatbaar symbool
304 in aritmetische expressie staat een niet-aritmetische identifier
305 array- of switchidentifier wordt niet gevolgd door subscript list
306 subscript list niet afgesloten door]
307 in Boolean expressie is een ifclause niet besloten met then
308 in Boolean expressie ontbreekt een else-deel
309 in Boolean expressie ontbreekt een)
310 in Boolean expressie begint een Boolean primary met ontoelaatbaar symbool
311 in Boolean expressie wordt een aritmetisch gedeelte niet gevolgd door een relational operator
312 in Boolean expressie staat een niet-Boolean identifier
313 in aritmetische of Boolean expressie is een ifclause niet besloten met then
314 in aritmetische of Boolean expressie ontbreekt een else-deel
315 in aritmetische of Boolean expressie ontbreekt een)
316 aritmetische of Boolean expressie begint met ontoelaatbaar symbool
317 in aritmetische of Boolean expressie staat identifier van type string of van designational type
318 in stringexpressie is een ifclause niet besloten met then
319 in stringexpressie ontbreekt een else-deel
320 in stringexpressie ontbreekt een)
321 in stringexpressie komt ontoelaatbaar symbool voor
322 in stringexpressie staat een identifier, niet van type string
323 in designational expressie is een ifclause niet besloten door then
324 in designational expressie ontbreekt een else-deel
325 in designational expressie ontbreekt een)
326 in designational expressie komt een onbekende numerieke label voor
327 in designational expressie komt ontoelaatbaar symbool voor

328 in designational expressie staat een niet-designational identifier
329 Boolean- of stringexpressie in plaats van aritmetische of designational expressie
330 designational expressie in plaats van een <type>-expressie
331 in expressie is een ifclause niet besloten door then
332 in expressie ontbreekt een else-deel
333 in expressie ontbreekt een)
334 in expressie komt ontoelaatbaar symbool voor
335 statement begint met variabele, niet gevolgd door een colonequal
336 assignment aan formele type-procedure identifier
337 assignment aan functie-identifier buiten de declaratie
338 in left part list komt na een integer variabele een non-integer variabele voor
339 in left part list komt na een real variabele een non-real variabele voor
340 in left part list komt na een Boolean variabele een niet-logische variabele voor
341 in left part list komt na een stringvariabele een niet-stringvariabele voor
342/343 in left part list komt na een aritmetische variabele een niet-aritmetische variabele voor
344/345 in left part list komt een designational identifier voor
346 in expressie komt een non-type procedure identifier voor
347 statement begint met identifier op ontoelaatbare wijze
348 actuele parameter is een identifier maar geen array- of switch-identifier
349 actuele parameter is een identifier maar geen procedure identifier
350 actuele parameter is een identifier maar geen type-procedure identifier
351 actuele parameter is een array-, switch- of procedure identifier
352 actuele parameter is een expressie in plaats van een array-, switch- of procedure identifier
353 actuele parameter is geen variabele waaraan geassigneerd kan worden
354 actuele parameter is een expressie maar geen aritmetische
355 actuele parameter is een expressie maar geen logische
356 actuele parameter is een expressie maar geen stringexpressie
357 actuele parameter is een expressie maar geen designational
358 actuele parameter is een expressie maar designational
359 te veel actuele parameters
360 te weinig actuele parameters
361 actuele parameterlijst niet afgesloten met)
362 actuele parameterlijst ontbreekt
363 te veel of te weinig actuele parameters bij bibliotheek procedure
364 statement begint met een getal dat geen integer label is
365 in statement volgt na then een conditional statement
366 na for statement volgt een else-deel
367 statement niet correct
368 in statement is een ifclause niet besloten door then
369 for list niet met do besloten
370 van for statement is de controlled variable niet-aritmetisch
371 in for statement staat in plaats van een controlled variable een type-procedure identifier
372 in for statement volgt op de controlled variable geen colonequal
373 in step-until-element van for list ontbreekt de until
374 in for list volgt op aritmetische expressie geen step, while,

do of comma
375 for wordt niet gevolgd door een identifier
376 switch list bevat een niet-designational identifier
377 switch list bevat een getal dat niet als integer label bekend is
378/379 switch list niet correct
380 in switch declaratie volgt op de switch identifier geen colonequal
381 in switch declaratie volgt op switch geen identifier
382 in array declaratie ontbreekt de bound pair list
383 in bound pair list volgt op een lower bound geen colon
384 bound pair list niet afgesloten door een]
385 declaratie niet door semicolon afgesloten
388 statement begint met switch identifier
389 op label volgt geen colon
390 in type-procedure declaratie komt geen assignment aan de procedure identifier voor
391 identifier reeds eerder gedeclareerd of als label bekend geworden
392 goto statement leidt binnen een for statement
393 subscript list bevat te veel of te weinig subscript expressies
394 actuele parameter is een identifier van een array, switch of procedure met onjuist aantal subscripts of parameters
395 actuele parameter is een identifier van onjuist type
396/397/398/399/400/401/402 procedure body is geen statement
490 wegens de reeds eerder gemelde fouten wordt het syntactisch onderzoek niet verder voortgezet
491 het programma is te ingewikkeld
492 het programma is te lang voor het beschikbare geheugen
493 het objectprogramma is na toevoeging van bibliotheekprocedures te lang voor het beschikbare geheugen. In plaats van het laatst gelezen getal wordt de lengte van het totale objectprogramma afgedrukt
494 het objectprogramma is te lang voor het beschikbare geheugen

7.3. Fouten ontdekt tijdens de executiefase.

500 actuele parameter is geen variabele waaraan geassigneerd kan worden
501 in arraydeclaratie is een bovengrens kleiner dan de bijbehorende ondergrens
502 van een arrayelement valt de (laatste) index buiten de range
503 van een arrayelement valt een index (maar niet de laatste) buiten de range
504 van een switch designator valt de index buiten de range
505 bij een assignment aan een arrayelement van integer type ligt de te assigneren waarde buiten de integercapaciteit
506 bij een assignment aan een arrayelement van een formeel array blijkt de corresponderende actuele parameter de naam van een switch te zijn
507/508/509 de formele, geindiceerde controlled variable in een for statement blijkt niet-aritmetisch te zijn
510 de integercapaciteit blijkt, bij afronding tot een integer, overschreden te zijn
511 bij de operatie : is een operand een niet-geheel getal, in absolute waarde kleiner dan 1 099 511 627 776
515 in een getallenband vindt read of resym een symbool van foute pariteit
516 read of resym vindt een onbekende ponsing

517 read of read1 vindt een getalscheider binnen een getal
519 bij een aanroep van fix of flo is de vierde actuele parameter geen identifier van een een-dimensionaal integer array
520 bij een aanroep van to drum of from drum is de eerste actuele parameter geen array identifier
521 bij een aanroep van to drum of from drum is de tweede actuele parameter te klein of te groot
522 bij een aanroep van een van de procedures voor het verwerken van vectoren en matrices correspondeert met een formele, als array gespecificeerde parameter niet de naam van een real array of integer array
523 bij een aanroep van een van de procedures voor het verwerken van vectoren en matrices is de dimensie van een van de array identifiers incorrect
531 bij een aanroep van compose vormen de parameters geen correcte voorstelling van een real
532 bij een aanroep van bit, bitstring, set, clear shift of circ shift valt een parameter buiten het toegestane gebied
533 bij een aanroep van set past de binaire representatie van de eerste parameter niet in het gespecificeerde traject

550 programma afgebroken wegens fout in magneetbandgebruik (Deze foutmelding wordt door een andere gevolgd)
551 als eerste tapeprocedure niet own tape of scratch tape
552 key $\leq$ 0
553 key te groot
554 unit niet 0, 1 of 2
555 bij to tape of from tape is de tweede parameter geen array identifier
556 trommeltraject niet korrekt
557 aanroep van to tape of drum to tape, nadat de vorige schrijfopdracht op EOT is gestuit en zonder dat nadien end of tape aangeroepen is
558 bij aanroep van own tape is number < 10 of > 999
559 header niet korrekt
560 retention < 0
561 new header aangeroepen voor een scratch tape
562 bij aanroep van own tape komt de opgegeven header niet overeen met de header die op de tape staat
563 schrijven niet toegestaan
564 transportfout. Raadpleeg systeemprogrammeur
565 de inhoud van de tape is niet in overeenstemming met MC-eisen. Raadpleeg systeemprogrammeur
566 niet nader te specificeren fout. Raadpleeg systeemprogrammeur
567 magneetbandeenheid niet in orde
568 de operateur heeft bezwaar tegen het opgegeven tapenummer
569 band is onbruikbaar
570 leesopdracht voorbij EOI

600/601/602/603/604 het programma bevat niet-geimplementeerde stringoperaties
605 het programma declareert own arrays
606/607/608 het programma bevat niet-geimplementeerde stringoperaties
609 de geheugenruimte is uitgeput
610 het programma bevat niet-geimplementeerde stringoperaties
611 bij een aanroep van printtext, putext of stringsymbol is (een van) de actuele parameter(s) geen string

630 aanroep van plot met niet-toegestane waarde van ipen
631 voorbereiding voor plot niet compleet
632 kader te groot gedeclareerd
633 foutief gebruik van plotcurve
634 pen komt in x-richting buiten het gedeclareerde kader
635 pen komt in y-richting buiten het gedeclareerde kader
636 fout in string in plottext

7.4. Fouten door overschrijding van in- en uitvoerspecificaties.

985 magneetbandgebruik bij programma-letter 'a','b' of 'c'
988 meer plottijd dan gevraagd in monitorkop
989 plotopdracht bij programma-letter 'a','b' of 'c'
990 trommeltransport bij programma-letter 'a' of 'b'
991 meer kaartuitvoer dan gevraagd in monitorkop
992 kaartponsopdracht bij programma-letter 'a','b' of 'c'
993 meer ponsbanduitvoer dan gevraagd in monitorkop
994 bandponsopdracht bij programma-letter 'a' of 'c'
995 te veel informatie op een pagina door overmatig gebruik van carriage(0)
996 meer regeldrukkeruitvoer dan gevraagd in monitorkop
997 het programma is afgebroken wegens overschrijding van de in de monitorkop opgegeven tijdslimiet
998 het programma is afgebroken wegens het ontbreken van verdere input
999 het programma is door de operateur beeindigd

8. De tijdsduur van enige operaties.

8.1. Daar de tijdsduur van een operatie in het ALGOL 60-systeem voor de X8 als regel van de specifieke syntactische constructie afhangt, zijn de volgende cijfers zeer globaal. In het bijzonder zal het evalueren van formele identifiers soms relatief lang duren. De hier gegeven getallen maken echter een ruwe schatting van de benodigde tijd voor bepaalde programmaonderdelen mogelijk. Alle tijden zijn opgegeven in microseconden (1 microseconde = $_{10}$-6 sec).

8.2. De diadische arithmetische operaties:

a) integer operanden:

+ of -	8	of	20
:	90		

voor andere operatoren zie bij real operanden

b) real operanden:

+ of -	13	of	25
×	40	of	52
/	65	of	77
↑ 2	290		
↑ 3.14	1500		

Als twee waarden opgegeven worden, geldt de kleinste waarde voor het geval dat de tweede operand eenvoudig is (een constante of een simpele niet-formele identifier).

8.3. De logische operaties:

¬	5
∧, ∨, ≡, ⊃	21

8.4. Indiceren:

a) integer of real array	-25 + 85 × (aantal indices)
b) Boolean array	50 + 85 × (aantal indices)

8.5. Assignments:

a) aan een real	15
b) aan een integer	16
c) aan een Boolean	14
d) aan een formele parameter (called by name) terwijl de overeenkomstige actuele parameter een identifier is	40

8.6. For statements:

for i:= 1 step 1 until n do	80 per repetitie-slag

8.7. Blokingang en -verlating: 45

8.8. Array-declaratie: 150 + 100 per indexpositie

8.9. Procedure-ingang en -verlating:

165 +
135 per parameter by value +
90 per parameter by name waaraan niet geassigneerd wordt +
145 per parameter by name waaraan wel geassigneerd wordt

8.10. Standaard functies:

abs	13
arctan	840
available	18
cos	510
entier	70
even	165
exp	800
fix	1800 + 50 × n + 200 × m
flo	1750 + 200 × n + 75 × m
from drum	zie hoofdstuk 4.4.
ln	650
matmat	1400 + 85 per element
mattam	1400 + 85 per element
matvec	1200 + 85 per element
random	125
remainder	370
sign	16
sin	510
sqrt	350
stringsymbol	390
tammat	1400 + 85 per element
tamvec	1200 + 85 per element
time	105
to drum	zie hoofdstuk 4.4.
vecvec	1100 + 85 per element

De bedragen voor fix en flo moeten met ongeveer 2000 verhoogd worden voor het geval dat fix of flo false aflevert.

8.11. De in- en uitvoer geschiedt gebufferd, d.w.z. vrijwel altijd zal de in te voeren informatie zich reeds in het kerngeheugen bevinden en de uit te voeren informatie zal in eerste instantie naar een gedeelte van het kerngeheugen worden getransporteerd. Slechts in extreme gevallen zal men hinder ondervinden van de langzame in- en uitvoerapparaten.

9.1. Overzicht van de uitvoerprocedures.

		bandponser	regeldrukker	kaartponser
besturend		outflexo outiso		
binair		puhep(n)	--	col(n)
tekst	symbolen	pusym(n)	prsym(n)	csym(n)
	strings	putext(s)	printtext(s)	--
	spaties	puspace(n)	space(n)	--
	tabs	--	tab	--
	nieuwe regel	punlcr	nlcr	--
getallen	zonder layout-specificatie	punch(x)	print(x)	cpunch(x)
	volgens absfix	absfixp(n,m,x)	absfixt(n,m,x)	absfixc(n,m,x)
	volgens fixed point represent.	fixp(n,m,x)	fixt(n,m,x)	fixc(n,m,x)
	volgens floating point represent.	flop(n,m,x)	flot(n,m,x)	floc(n,m,x)
specifieke procedures		runout stopcode	newpage carriage(n)	
informatieve procedures			linenumber print pos	cpos

9.2.1. Tabel van standaardhoeveelheden.

functie	eenheid	programma-letter			
		a	b	c	d
rekentijd	1 milli-uur	30	60	200	30
regeldrukker	1 pagina	15	15	15	15
bandponser	1024 ponsingen	0	15	0	50
kaartponser	1024 kaarten	0	0	0	1
plottijd	1 milli-uur	0	0	0	250
magn. trommel	1024 woorden	0	0	40	80

9.2.2. Tabel van maximale hoeveelheden.

functie	eenheid	programma-letter			
		a	b	c	d
rekentijd	1 milli-uur	30	150	1000 (5000)	1000 (5000)
regeldrukker	1 pagina	15	100	500 (2500)	500 (2500)
bandponser	1024 ponsingen	0	75	0	250 (500)
kaartponser	1024 kaarten	0	0	0	5 (25)
plottijd	1 milli-uur	0	0	0	1000 (5000)
magn. trommel	1024 woorden	0	0	40	80

De tussen haakjes vermelde waarden gelden 's avonds. Bij gebruik hiervan dient men een speciale geleidekaart in te vullen.