II

1969

X X X WAMMES: een timesharing systeem voor de ELX8.

Gids voor gebruikers.

X X I : Bedoelingen

X X II : Uitvoering

X X III: Handleiding voor het gebruik

X X IV : Bijlagen

dok. nr.: WOO/1

X X X I: Bedoelingen.

In Maart 1968, toen we met het ontwerp van WAMMES begonnen, was ons bedrijfssysteem de MONOTOR, een uni-running-systeem (d.w.z. een programma kan ingelezen en gestart worden als het vorige beeindigd is.).
De belangrijkste tekortkomingen van unirunning, die aan de dag treden als de machine behoorlijk wordt bezet, zijn:

- er gaat veel tijd verloren, vooral bij programma-wisseling.
- direkt acces is niet mogelijk: niemand kan met een probleem direkt bij de computer aankloppen.
- zg. konversatie-programma's waarbij iemand via een telex de computer dirigeert (of door de computer gedirigeerd wordt) komen in het gedrang omdat ze teveel tijd konsumeren.

Hoewel deze konversatie programma's -in ons geval- niet de programma's zijn waar de machine om draait, zijn ze een onmisbare hulp geworden bij het ontwerpen, testen en verbeteren van programma's, het scheppen en onderhouden van programma-, tekst- en dokumentatie-archieven. Daarnaast moet er een ruime gelegenheid bestaan voor de programmerende gebruiker om zich "aan de machine" te bekwamen in nieuwere technieken, zoals listprocessing, string- en formulemanipulatie.
Ons doel was daarom een systeem te bouwen dat dezelfde programma's efficiënter zou kunnen verwerken en in beperkte mate in time-sharing zou voorzien (waarmee wij bedoelen dat een aantal gebruikers via een of meer telexen simultaan acces tot de machine hebben.)
We hopen vooral dat de machine met WAMMES een prettiger en beter hanteerbaar instrument is geworden. We hebben op elk vlak de beste medewerking gehad; aan de softwarekant vooral van Jan Derksen, die lange tijd heeft meegewerkt aan het ontwerp, en onze kollega's Anton Mars en Pim Biekmann die ons buitengewoon geholpen hebben.

Pieter van Engen en Rolf Meesters
31 maart 1969.

dok.nr.: WOO/2

X X X II: Uitvoering

Het ontwerp is sterk beïnvloed door triviale kondities zoals de funktie van de X8 op het instituut, bestaande konventies, beschikbare hardware, software, mensen en tijd.
De daaruit voortvloeiende konsekwenties zijn niet alle even triviaal en vereisen hier en daar enige toelichting.

X X II.1: Beschikbare hardware

X X II.2: Gekozen methoden

X X II.3: Iets over de struktuur van het systeem

X X II.4: Mogelijkheden

X X II.5: Ervaringen

X X X II.1: Beschikbare hardware

De EL X8 wordt gerekend tot de middelgrote computers.
Onze konfiguratie omvat op het ogenblik:

- 32 K (KILO=1024 woorden) kerngeheugen
  (27 bits + pariteitsbit, geheugencyclus-tijd 2.5 mms)
- een drum van 512 K (512 sporen, omwentelingstijd 40 ms)
- real time klok met wekker
- geheugenprotektie
- 4 magneetband units (120 KC, dichtheden 200, 556, 800 BPI)
- 1 regeldrukker (regelbreedte 144 characters, snelheid 5-10 regels/sec.)
- 1 bandlezer (1000 char./sec.)
- 1 bandponser (150 char./sec.)
- 4 teleksen (Siemens, 10 char./sec.)
- koppeling met een konfiguratie rond een DEC PDP8

Hierbij kan men opmerken dat er, voor in time-sharing noodzakelijke manoeuvres zoals het redden van programma's, het bufferen van in- en output, het inrichten van file- en gebruikersadministraties en voor een eventuele bibliotheek, weinig random-accesgeheugen beschikbaar is.
Daarom worden voor biblioteekwerk magneetbanden ingeschakeld (1 unit semi-permanent).

X X X II.2: Gekozen metoden

Time-sharing is niet nieuw meer. Er bestaat gelukkig een uitgebreide litteratuur over, die prakties alle relevante problemen bestrijkt. Veel van de daarin aangegeven metoden echter, zijn nogal afhankelijk van de gebruikte computer en eerder opgesomde omstandigheden. Daarom hebben we in een aantal gevallen een eigen interpretatie gevolgd. Het 'dozen'-systeem is destijds voor het WINDOW-SYSTEM ontworpen en omdat we er goede ervaringen mee hebben is het in WAMMES ook toegepast.

X X II.1: de positie van de gebruiker

X X II.2: gebruik van achtergrondgeheugen

X X II.3: geheugenverdeling

X X II.4: swap-metode

X X II.5: prioriteiten regeling

X X II.6: dynamiese werkruimte verdeling

X X X II.2.1: de positie van de gebruiker

In eerste instantie hebben we erg opgezien tegen het probleem: 'hoe moeten we de gebruiker zien'.
Voorop gesteld dient te worden dat wammes er is voor de mensen die ermee omgaan en niet omgekeerd. Dat houdt in dat een gebruiker makkelijk toegang moet hebben tot alles wat het siesteem te bieden heeft en tevens op eenvoudige wijze een eigen omgeving moet kunnen scheppen.
De vorm en de mate van bescherming van informatie, programma's, administraties raakt etiese problemen. Verregaande protektie en geheimhouding is zonder meer hinderlijk (en te duur) en, mede gezien de open sfeer op het instituut, ontoelaatbaar.
Dit heeft ons geholpen een tamelijk eenvoudige filosofie te vinden. Namelijk alleen de noodzakelijke maatregelen tegen vergissingen. De etiese problemen kunnen en mogen we niet door het siesteem laten bepalen.
Er zijn daarvoor alleen kontroles aanwezig tegen het overschrijven en weggooien van informatie. In prinsipe mag elke gebruiker alles 'lezen' (tapes, drum, administraties, enz.). Om dit te kunnen waar maken is wammes teleksgeoriënteerd. Een gebruiker wordt geïdentifiseerd met de teleks waaraan hij zit; waar hij dan alles mag doen zolang het zijn eigen zaken betreft.
Wanneer hij begint, noteert het siesteem zijn naam en de begintijd, waarna bij alle output en files die hij kreëert wordt genoteerd dat het van hem is (dit vergemakkelijkt b.v. het sorteren van regeldrukker output).
Er zijn nog enige ekstra mogelijkheden aangebracht waardoor de gebruiker zijn informatie tegen eigen fouten kan beschermen (o.a. bij de drum-files).
Wat betreft de lopende programma's is er de geheugenprotektie die ieder ander en het siesteem absoluut beveiligt tegen schrijffouten. Lezen mag een programma in zijn eigen gebied en in de administraties die in het eerste geheugenstuk staan. Een belangrijk punt is nog, dat we ernaar streven om zoveel mogelijk dokumentatie al in het siesteem zelf te stoppen (b.v. de verklaring van foutmeldingen).

X X X II.2.2: gebruik van achtergrondgeheugen

Het siesteem kent alleen 'lopende programma's' en 'files'. De files omvatten alle mogelijke soorten van informatie en worden alle op dezelfde manier behandeld. Omdat de drum niet groot genoeg is om alle files te bergen, hebben we een 2-traps-siesteem gekozen:
de files die op een bepaald moment snel toegankelijk moeten zijn, staan op de drum.
Een magneetband, de LOTUS fungeert als biblioteek.
Deze bevat a) een aantal standaardfiles
b) zoveel mogelijk files die men op die dag nodig heeft (ook een magneetband is te klein om alle programma- en dokumentatieteksten te bevatten)

De LOTUS is semipermanent aanwezig; d.w.z. de bezette unit kan door de operateur vrijgegeven worden. Een belangrijke fasiliteit is de momentane uitbreidbaarheid van de LOTUS. Elke gebruiker mag files toevoegen (de dumpen) die hij belangrijk vindt of tijdelijk wil redden. Omgekeerd kunnen files van de LOTUS op de drum gezet worden. Het dumpen, weggooien of halen van files wordt niet automaties door het siesteem gedaan. Het beheer ligt volledig in handen van de gebruiker(s). (Ook bij gebrek aan plaatsruimte op de drum).
In wammes zijn een aantal 'commands' opgenomen. Deze zijn niet vast in het kerngeheugen gebakken, maar uitgevoerd als losse programmaatjes welke op de drum staan (niet als files). Sommige files zijn reeds-vertaalde programma's. Met het 'command' START kan men zo'n programma aanmelden. Er wordt dan eerst in de drum- en daarna in de lotusadministratie gekeken of de betreffende file er is. Staat hij op de lotus, dan wordt het programma niet eerst op de drum gezet maar rechtstreeks in het kerngeheugen.
Een deel van de drum wordt gebruikt voor de buffering van in- en output van de langzame apparten. Een ander deel voor het swappen.

X X X II.2.3: geheugenverdeling

Het opdelen van programma's in segmenten zoals algemeen gebruikelijk is in sharing-siestemen was onuitvoerbaar omdat de X8 er niet voldoende voor is uitgerust. Adressen worden in de machine absoluut genoteerd (b.v. bij subroutinesprongen) zodat een eenmaal lopend programma plaatsgebonden is. (Men kan dit probleem bij andere talen oplossen door in de vertaling geen direkte adresbepalingen toe te passen en b.v. in alle subroutines de terugsprong op een afwijkende manier uit te voeren).
Bovendien is het voor paginering bestemde adresmodifikatie-register (D) niet tegen vullen door het programma geprotegeerd en wordt door vele in omloop zijnde programma's voor andere op zich zeer nuttige doeleinden gebruikt. Daarom wordt het benodigde geheugen bij aanmelding van een programma aaneensluitend en eenmalig bepaald. Wel was het mogelijk vertaalde programma's met relokatie-aanwijzingen op te slaan zodat ze bij het starten pas hun definitieve plaats krijgen. Hiervoor is een assembler-omgeving geschreven waarmee elk willekeurig programma (een enkele maal met een kleine wijziging) in die vorm gegoten kan worden. De metode is eenvoudig: op twee verschillende plaatsen assembleren en aan de hand van verschillen in de vertalingen relokatie aanwijzingen opstellen.

X X X II.2.4: swap-siesteem

Als de aanwezige programma's niet gelijktijdig in het kerngeheugen kunnen, moet er van tijd tot tijd één plaatsmaken voor een ander. Dit proses noemt men swappen. Hiervoor bestaan verschillende metoden, variërend van één waarbij telkens slechts 1 programma geheel in het kerngeheugen verblijft, tot een verfijnd paginerings-siesteem waarbij van de lopende programma's (of meer) niet alle pagina's gelijktijdig aanwezig hoeven te zijn. In de meeste siestemen met dezelfde handikap (plaatsgebonden programma's) is er telkens 1 programma aktief in het kerngeheugen aanwezig. Zonder ons al te veel komplikaties op de hals te halen konden we meer programma's tegelijk toelaten door alleen te streven naar een zo gelijkmatig mogelijke verdeling van het beschikbare kerngeheugen. En wel door voor een nieuw programma naar het 'dunst-bevolkte' gebied te zoeken. Op het ogenblik kunnen we volstaan met een maksimum van 2 programma's per pagina.
Overigens konden we een sterke vereenvoudiging bereiken door niet met een pagina als eenheid te werken maar met een kilo (2 pagina's).

X X X II.2.5: prioriteiten regeling

De verwachting was en is, dat bij unirunning de behoefte aan konversatie-programma's de daarvoor beschikbare tijd zal overtreffen, terwijl de meer rekenintensieve programma's niet de volledige machinetijd zouden opeisen (zie b.v. het verslag Machinegebruik EL X8 van Wil Carton). Slechts enkele (van de laatste programma's) vereisen in time-sharing-omgeving ijl-prioriteit (n.l. de rechtstreeks aan een eksperiment gekoppelde programma's) en een kombinatie van zulke ijl-taken zal waarschijnlijk een vrij zeldzame gebeurtenis worden.
In het normale geval (= afwezigheid van eenijl-taak) wordt nu de processortijd gelijkelijk over de lopende programma's verdeeld, ze krijgen om beurten de hoogste prioriteit. Tevens wordt er gestreefd naar een gemiddelde responsietijd van een aantal sekonden. Daarvoor wordt een lopend programma binnen een vastgestelde tijd onderbroken als een ander voortgezet zou kunnen worden maar daartoe een, door het eerste bezet, stuk kerngeheugen nodig heeft.
Een ijl-taak krijgt absolute prioriteit en wordt niet t.b.v. een ander programma uit het kerngeheugen verwijderd (dat kan wel op instruktie van de operateur).
Een eventuele twede ijl-taak komt in prioriteit na de eerste en voor alle andere programma's (taken), enz.
Als ze niet gelijktijdig in het kerngeheugen kunnen komt de twede dus pas aan bod als de eerste is afgelopen (of door de operateur onderbroken).

X X X II.2.6: Dynamische werkruimte verdeling.

De werkruimten van het systeem en van de programma's worden behandeld als geordende ketens van trajekten.
Aanvraag van een doos, d.i. een stuk werkruimte van willekeurige lengte, kan gedaan worden via een subroutine aanroep. In de werkruimteketen van e.g. vrije dozen wordt van voor naar achter gezocht naar een doos die groot genoeg is om er het gewenste stuk vanaf te knippen of de hele doos af te staan ("first-fit" methode) (zie schets 1.)
Bij vrijgeven van een gebruikte doos wordt deze op de juiste plaats in de keten gehangen en als hij grenst aan een voorganger en/of opvolger met deze verenigd tot een grotere vrije doos. Daarbij mag het teruggeven van gebruikte dozen in willekeurige volgorde gebeuren. De optimale methode van opvragen en vrijgeven van dozen, is de LIFO-methode: de laatst opgevraagde doos het eerst weer vrijgeven. Deze methode wordt zoveel mogelijk in het systeem toegepast en de gebruikers aanbevolen.
Opgemerkt moet nog worden dat aan voor- en achterkant van een doos een extra woord wordt geplaatst dat gekontroleerd wordt bij teruggave van de doos. Deze kontroles zijn van groot belang gebleken bij het testen van programma's.
Het systeem heeft een eigen werkruimte waaruit het dozen neemt voor zijn administraties. Elk programma heeft, in eigen gebied, een werkruimte waaruit zowel het systeem als het programma zelf, dozen kunnen opvragen.
Het systeem reserveert voor in- en output apparaten buffers in het gebied van het programma.

Schets 1.

werkruimte ketening.

situatie a.
na instelling van de werk-
ruimte (dus bij starten van
het systeem of het programma)

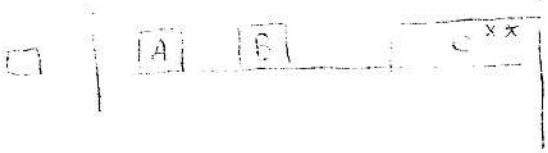
gehele werkruimte

situatie b.
willekeurig moment
(A,B,C en D stellen
vrije dozen voor)

situatie c.
er is een doos aangevraagd
en uit C genomen.
Als dezelfde doos direkt weer
zou worden teruggegeven,
ontstaat situatie b weer.

situatie d.
de gebruikte dozen tussen
C* en D zijn terug-
gegeven.

Als alle gebruikte dozen zijn
teruggegeven, is het weer
situatie a.



X X X II.3: iets over de struktuur van het siesteem

Een momentopname van wammes toont een aantal asynchrone prosessen, die zich in een van de volgende toestanden bevinden: aktief, onderbroken of geblokkeerd.
Aktief is het proses dat het CRO bestuurt. Er kan dus altijd maar één aktief proses zijn.
Nemen we een willekeurig proses, het VOORBEELD, dan zijn daarvoor de volgende overgangen mogelijk:

aktief → onderbroken;	oorzaak het doordringen van een ingreep.
aktief → geblokkeerd;	het VOORBEELD vraagt naar een, nog niet gekomen melding van een ander proses.
onderbroken → aktief;	het VOORBEELD krijgt het CRO toegewezen.
geblokkeerd → onderbroken;	een ander proses geeft de melding waarop het VOORBEELD stond te wachten.

De eerste drie overgangen hebben tot gevolg dat het sentrale gedeelte van WAMMES, de WACHTER, in aktie komt.
De verschillende prosessen worden onderscheiden in:

a) sentrale prosessen,(wammes machines genaamd 'WM'), die zorgen voor koördinatie van alle lopende prosessen.
b) kommunikatie prosessen (communication machines, CM), die voor de uitwisseling van informatie met de verschillende apparaten zorgen.
c) programma's (programmable machines, PM), alle niet voortdurend deelnemende prosessen, die toegevoegd of verwijderd kunnen worden.

X X X II.3.1: de WACHTER

Als een aktief proses door een ingreep onderbroken wordt of zichzelf blokkeert, komt de WACHTER in aktie. Deze redt de gegevens van het proses in een persoonlijke administratie (de LOG van dat proses), dooft eventuele ingrepen en deblokkeert de CM's die op deze ingrepen staan te wachten. Vervolgens wordt er een onderbroken proses geselekteerd en aktief gemaakt.
Is zo'n proses niet te vinden, dan krijgt DYST het CRO: de machine komt dan in een dynamiese stop, wachtend op een ingreep.
Bij de selektie van een onderbroken proses bestaat een zekere hiërarchie. De siesteemprosessen, WM's en CM's, staan genoteerd in de ROL in vaste volgorde. De WACHTER zoekt eerst in de ROL van boven naar beneden. Daarna pas wordt in een keten, KOOR, van programma's gekeken.

X X X II.3.2: enkele sentrale prosessen

De KLUTSER verzorgt de tijdverdeling over de aanwezige PM's. Met behulp van de KLOK vindt om een bepaalde tijd (de KLUTSTIJD die gekozen is op 0,5 sekonde) een inspektie plaats van de toestand. Er bestaan vier ketens waarin programma's zich kunnen bevinden:

KOOR	- potentieel aktieve PM's, die in het kerngeheugen staan.
RESERVE	- de in prinsipe werkeloze PM's.
STIL	- een tussenstadium van de overgang KOOR → RESERVE
BIJNABINNEN	- idem voor RESERVE → KOOR

Een programma mag een minimaal aantal sekonden, de VERBLIJFTIJD, achtereen in KOOR verblijven. Is die tijd voorbij dan noteert de KLUTSER de geheugenruimte van de PM als potentieel VRIJ. Als een opponent aanspraak maakt op die ruimte wordt de PM via STIL naar RESERVE verhuisd. STIL is nodig om alle (charon-)transporten van en naar het kerngeheugen af te wachten die de PM nog heeft lopen. Omgekeerd als een programma dat in RESERVE hangt in de onderbroken toestand komt, zal de KLUTSER de geheugenruimte die het nodig heeft bespreken. De PM gaat naar BIJNABINNEN. Zodra de besproken ruimte vrij (en stil) is verhuist hij naar KOOR.
De KLUTSER zorgt tevens voor het wisselen van de prioriteit in de keten KOOR, door de PM's in een andere volgorde te hangen.
Met de keuze van de VERBLIJFTIJD (momenteel 4 sekonden) hebben we gepoogd een maksimale responsietijd te kunnen garanderen, die niet hinderlijk is voor de gebruiker aan een teleks.
De VERHUIZER verzorgt het transporteren van PM kilo's van en naar de drum. Hij wordt geaktiveerd door de KLUTSER als er een programma in BIJNABINNEN hangt. De VERHUIZER kijkt welke pagina's de PM nodig heeft. Zijn deze vrij

(en stil) dan worden ze gered op de drum en de nieuwe inhouden in het geheugen geplaatst. Omdat er gekeken wordt vanuit de binnenkomer zullen alleen de noodzakelijke kilo's naar de drum verhuizen. De rest blijft waar ze is. Dat houdt in dat de kilo's van een PM, die in RESERVE hangt, deels in het geheugen en deels op de drum kunnen staan.
Een programma in KOOR is noodzakelijkerwijs altijd kompleet in het kerngeheugen.

X X X II.3.3: kommunikatie prosessen

Informatie uitwisseling tussen programma's en randapparaten verloopt altijd via CM's. Een CM regelt in het algemeen de koppeling van één apparaat aan één of meerdere prosessen. Een apparaat kan daarbij fysies per regel (teleksen), per transport (drum) of voor de duur van een gehele INFOSTROOM aan een PM gekoppeld worden.
Alleen voor tape-units geldt dat er slechts 1 programma op het apparaat aanspraak mag maken. Als een PM een tape-unit aanvraagt krijgt hij deze als er nog een vrij is, anders wordt hij genokt.
Alle andere apparaten kunnen door **alle** aanwezige PM's bespeeld worden. Dit (ook wel Multiplexing genaamd) wordt bereikt door in- en output op te sparen. Voor de teleksen in het kerngeheugen, voor de bandlezer, bandponser en regeldrukker op de drum.
De CMTAPU (bandponser) en de CMLIPR (regeldrukker) koppelen een stroom aan het apparaat als er een minimale hoeveelheid output verzameld is of als de stroom gesloten is. Dit minimum is gekozen op 30 meter resp. 4 formulieren.
Wanneer een stroom aan het apparaat gekoppeld is, blijft dat zo tot de stroom geleegd is. Is de stroom dan nog niet gesloten, dan wordt een andere stroom geselekteerd, als een bepaalde tijd verstreken is zonder dat de produsent een eenheid (1 formulier) afgeleverd heeft, òf als de beschikbare drumruimte een kritieke grens is gepasseerd. Bij selektie van een stroom wordt eerst gezocht naar een reeds gesloten stroom. Als die er niet is, naar de stroom met het grootste aantal klare eenheden.
De CMTARE behandelt aanvragen in volgorde van binnenkomst. Een papierband wordt geheel ingelezen en op de drum gezet tot een zeker maksimum. Heeft de CMTARE het hem toegestane maksimum aantal drumbuffers gevuld dan wordt hij geblokkeerd tot een PM een buffer geleegd heeft. Een PM mag pas om een nieuwe band vragen als een vorige geheel door hem verwerkt is.

dok. nr.: W00/10.7

X X X III: Handleiding voor het gebruik

De nu volgende aanwijzingen horen bij een huidige versie. Omdat het siesteem op de groei gebouwd is zullen er nog vele wijzigingen en aanvullingen volgen.
Inlichtingen over de laatste modifikaties en uitbreidingen van het repertoire zijn on-line verkrijgbaar (zie commands).

X X III.1: Hantering

X X III.2: Files

X X III.3: Commands

X X III.4: Talen

X X III.5: Editing

X X III.6: Foutmeldingen

X X III.7: Onderhoudsprosedures

X X III.8: Dokumentatie

X X X III.1: Hantering

Teleks o is gereserveerd voor meldingen van WAMMES aan de maschinekamer. Dat kunnen zijn: aanvragen voor magneet- en papierbanden en meldingen van het niet bedrijfsklaar zijn van apparaten. De breinbaas moet er voor zorgen dat alle apparaten ingeschakeld zijn; hij regelt de toewijzing van teleksen en tape-units, het inlezen van papierbanden, zorgt dat de goede LOTUS aanwezig is, beslist wanneer de lotus verwijderd mag worden en zorgt ervoor dat de techniese dienst gewaarschuwd wordt bij optredende storingen.
Over teleks o mogen ook programma's gestart worden maar dan bij voorkeur achtergrondprogramma's en alleen konversatieprogramma's als hierdoor niet andere gebruikers opgehouden worden (door het niet doorkomen van meldingen en aanvragen).

Enkele meldingen: (op + eindigen vragen, die met ++ beantwoord moeten worden als alles in orde is)

1) NBK <naam apparaat>+. Het apparaat staat niet aan of is defekt
2) EOM LIPR+. Papier van de regeldrukker is op
3) EOM TAPU+. Band is op (bandponser)
4) CHARONFOUT <app nr> : Machinestoring
5) <programmanaam>
   TELP<nr> BAND+ Aanvraag voor een papierband voor de bandlezer
6) LEESTAPE<unitnr.>
   <programmanaam>+ Aanvraag voor een magneetband op genoemde unit
7) SCHRIJFTAPE<unitnr.>
   <programmanaam>+ Aanvraag idem, voor <u>schrijven</u> (denk om de schrijfring)

Bij elke aanvraag voor een magneetband moet er, vóórdat de aanvraag gehonoreerd mag worden, gekeken worden of
a) de goede tape op de unit met aangegeven nummer staat en de unit op START
b) of niet 2 units dat nummer hebben en op START staan

Optuigen

Het siesteem kan van de trommel ingelezen worden m.b.v. het IP-bandje SYSTIN. Lukt dit niet (als de inhoud van de trommel bij onderhoud of testen overschreven is) dan kan het van de magneetband WAMMES ingelezen worden m.b.v. het IP-bandje TATRHOUT. Na inlezen vraagt het siesteem via de breinbaas teleks, datum en tijd, waarop een datum van 6 desimalen, een komma en een tijd van 4 desimalen gevolgd door 2 vlinders ingetypt moet worden.
v.b.: 690401,2000⊹⊹ (= 1 april '69 om 20.00 uur)
Als de goede LOTUS op de unit met nummer 3 staat kan het command LOTUS ++ gegeven worden.
(N.B.: een vlinder wordt hier en in de rest van het verhaal door een + voorgesteld, een spoorwegkruising door ≠)

Het siesteem vraagt dan LEESTAPES LOTUS ++ waarop de breinbaas met ++ antwoordt als alles in orde is.
Besluit hij in een later stadium dat de LOTUS verwijderd moet worden om een tape-unit vrij te maken, dan moet hij ervoor zorgen dat programma's en files, die andere gebruikers nodig hebben, naar de drum zijn overgeheveld.
Daarna typt hij LOTUS AF++
en pas na de melding AF kan hij de LOTUS-tape van de unit halen.

Werking van de teleksen

Elke gebruiker kan aan zijn teleks te werk gaan alsof hij een machine voor zich alleen heeft. Dat dit niet zo is kan hij merken als het siesteem hem fasiliteiten weigert, zoals geheugen- of drumruimte, files, tape-units of IKOB. Verder kan het gebeuren dat er over zijn teleks output verschijnt van een ander programma, omdat elke teleks in prinsipe voor elk programma bereikbaar is.
Aan een teleks kunnen een of meer programma's gestart worden. Stoppen of onderbreken van een programma d.m.v. een command kan alleen aan die teleks gebeuren, waarover het programma is aangemeld. Alle foutkodes, stop- en onderbrekingsberichten worden uitgetypt over de teleks

waarop het programma is aangemeld. De machine drukt altijd geprogrammeerde symbolen zwart af en ingetypte rood.

Een teleks kan zich in 4 toestanden bevinden:

a) rust
b) bezig met output
c) typist geeft antwoord op een vraag van de machine
d) typist geeft een instruktie aan het siesteem (command)

Als de output eindigt op + dan is dit een vraag; de machine geeft geen output meer over de teleks voordat de vraag beantwoord is. Een antwoord kan alleen beëindigd worden met ++ (uitzondering: + U zie uitgesteld antwoord)
Is de teleks in rust of bezig met output dan kan de deelnemer door een shiftsymbool (A.. of 1..) in te typen, te kennen geven dat hij een command wil geven.
Een eventuele outputregel wordt afgemaakt en de teleks drukt een sterretje af ten teken dat hij in de commandtoestand is gekomen. De teleks blijft in deze toestand tot ++ (korrekte afsluiting van een command) of + ≠ (de typist bedenkt zich en annuleert zijn command) wordt aangeslagen.
Als het command door het siesteem erkend wordt, wordt de twede + ook afgedrukt, anders een zwarte ≠.

Ponsen aan een teleks

Aan een teleks kan geponst worden zodra deze in commandtoestand is gebracht. Alle aangeslagen symbolen behalve TW (de toets <) worden teruggetypt, NR als TWNR.
Zolang er geen + aangeslagen wordt blijft de teleks in commandtoestand (verschijnt er dus geen output van een programma). Blanke band kan men produseren door de toets ster gevolgd door de herhaaltoets in te drukken.
Men moet het ponsen beëindigen door +≠ in te slaan.
De case aan het begin van een band kan men verkrijgen door achtereenvolgens aan te slaan:

≠ ⁂

X X X III.2: Files

Men kan het siesteem een hoeveelheid informatie met een naam laten opbergen zodat die later weer onder die naam bereikbaar is. Zo'n informatie-eenheid met een naam heet een file.

Files worden altijd eerst op de drum gekreëerd en kunnen vandaar op de lotus gedumpt worden. Men kan bovendien ekstra tapes gebruiken om spesiale files (b.v. array-files) te verzamelen: men kan op andere tapes dumpen en files van de lotus kopiëren.

Alle mogelijke informatie kan als file opgeborgen worden. Een bijzondere soort vormen de vertaalde of binaire programma's. Deze files kunnen n.l. gestart worden, of ze nu op de drum staan of op tape.

Voor files op de drum gelden de volgende regels:

1) elke file heeft een naam van 6 of minder alfanumerieke simbolen. Het eerste simbool moet een letter zijn.
2) de lengte van een file wordt uitgedrukt in woorden.
3) alle programma's kunnen alle files lezen.
4) files kunnen alleen beschreven worden door programma's van de gebruiker die de file op de drum gezet heeft.
5) een gebruiker kan een eigen file met SAVE tegen eigen programma's beschermen.
6) alle door één gebruiker gestarte programma's kunnen al zijn files beschrijven, behalve de files die met SAVE beveiligd zijn.
7) vanwege de kompatibiliteit met reeds bestaande programma's leveren reservering en opsporing het fysische begin-adres van de file op en kunnen ze via absolute adressering bereikt worden.
8) er is slechts een beperkt aantal files op de drum toegestaan (voorlopig 12).
9) men kan geen file kreëren met een reeds voorkomende naam.

Als aparte subroutine is uitgevoerd:

ZOEKFILE: levert absolute drumadres en lengte af.

De lotus wordt van tijd tot tijd gefilterd, d.w.z. er worden files van afgevoerd, naar andere tapes overgeheveld of op een rest tape gezet waar ze na een maand automaties uit verwijderd worden.

Voor files op lotus-tapes geldt:

1) voor naam en lengte hetzelfde als bij drum-files
2) elke gebruiker kan alle files van de lotus halen
3) er kan geen file op een lotus-tape gezet worden met een naam gelijk aan een reeds daarop staande file
4) bij dumpen van een file wordt de file achteraan op de tape toegevoegd (append)
5) de lotus kan door de breinbaas weggehaald zijn

Alle mogelijke manipulaties met files zijn als command én als subroutine uitgevoerd.
Enkele hiervan zijn:

CLAIM: reserveert drumruimte voor een nieuwe file en beschermt de file tegen overschrijven door andere gebruikers. CLAIM mislukt als de naam al op de drum voorkomt of als er teveel files zijn of als er niet genoeg ruimte is op de drum.

WIS: verwijdert een file uit de administratie en geeft de bezette drumruimte vrij. WIS mislukt als de file niet bekend is, van een andere gebruiker is of met SAVE beveiligd is.

DUMP: zet een drumfile achter op de lotus en wist de drumfile (zie WIS). Het dumpen mislukt als de naam al op de tape voorkomt of de tape vol is.

GET: haalt een file van de lotus, claimt een drumfile en zet de tapefile daarin. GET mislukt als de claim mislukt of als de naam onbekend is.

SAVE: hiermee kan een gebruiker een drumfile tegen overschrijven door eigen programma's beschermen en ook beschikbaar stellen aan een andere gebruiker.

FETCH: koppelt een met save beschermde file (van de gebruiker zelf of van een ander!) aan de teleks van de aanroeper.

X X X III.3: Commands

De nu volgende opsomming zal nog vaak uitgebreid worden. Daarom is er het command WAT waarmee men steeds de laatste lijst van mogelijkheden uitgetypt krijgt. De meeste commands (de niet-urgente) geven aanleiding tot het starten van een kleine PM (1K) die verder zelf de kommunikatie met de teleks en evt. andere apparaten ter hand neemt. Voor het starten van een command moet men eerst een shiftsimbool aanslaan en dan wachten tot er een sterretje teruggetypt wordt, daarna typt men het command.
Als een gebruiker begint moet hij/zij eerst het volgende command geven:

LOGIN <naam gebruiker>++
Als er nog iemand bezig was aan de betreffende teleks, wordt dat gemeld. De gebruiker kan dan aan de breinbaas vragen wat er gebeuren moet. Wie weggaat moet dit te kennen geven via LOGUIT++ nadat hij al z'n files heeft opgeruimd.

Aanroep en funktie van enkele commands:
Tussen () is fakultatief; haken niet meetypen.

WAT++
Typt een overzicht van alle mogelijke commands.

START <PM naam> (<aantal> K)++
Het bewuste programma wordt gezocht (eerst op de drum en daarna op de lotus) en gestart. Met het fakultatieve <aantal K> kan men het programma met een, van de tijdens assemblage opgegeven , afwijkende lengte (werkruimte) starten.

NOK <PM naam>++
De PM wordt verwijderd.

BREEK <PM naam>++
De PM wordt geblokkeerd en kan uit deze blokkade alleen verlost worden met NOK of CONT.

CONT <PM naam>++

Een met BREEK geblokkeerde PM wordt gedeblokkeerd.

+U

Kan als antwoord op een vraag ingetypt worden om het antwoord uit te stellen of te onderbreken.

HERHAAL++

De oudste vraag waarop het antwoord met +U uitgesteld was wordt herhaald, een antwoord dat met +U onderbroken was wordt tot en met het laatste ingetypte simbool opnieuw gedrukt waarna het antwoord vervolgd kan worden. Heeft men van een antwoord de laatste regel met +≠ geannuleerd en vervolgens met +U uitgesteld dan verschijnt na HERHAAL++ alleen een +.

FILES++

Typt een overzicht van alle aanwezige drumfiles, telkens <naam> <drumadres> <lengte> <teleksnr.>.

CLAIM <naam> <lengte>++

Reserveert een drumgebied voor een file. <lengte> mag het aantal benodigde woorden zijn of aantal gevolgd door een K (= aantal kilo's).

v.b.: 16 wil zeggen 16 woorden
16K " " 16 sporen of '40000' woorden

Als de claim lukt wordt teruggetypt:
<naam> <drumadres> <lengte>

WIS <naam>++

Wist een file van de gebruiker als deze tenminste niet met SAVE geprotegeerd was.

SAVE <naam>++

Beschermt een file tegen overschrijving door eigen programma's. De file kan hierna door iedereen met FETCH naar zich toe gehaald worden.

FETCH <naam>++

De betrokken drumfile wordt aan de gebruiker toegewezen als hij beschikbaar was (via SAVE).

DUMP <naam> (<evt. nieuwe naam><evt. nieuwe lengte>)++
De genoemde drumfile wordt achterop de lotus gezet (als de naam nog niet voorkwam) en daarna wordt de drumfile gewist.

GET <naam> (<evt. nieuwe naam><evt. nieuwe lengte>)++
De genoemde file wordt van de lotus gehaald en op de drum gezet (via claim).

STELP LIPR++
Stopt de regeldrukker-output die op dat moment verschijnt en gooit alle verdere regeldrukker-output van het betrokken programma weg. Kan alleen via de breinbaas teleks gedaan worden.

STELP TAPU++
Hetzelfde voor de bandponser.

WIE++
Typt de namen van de gebruikers aan de verschillende teleksen en de lopende programma's uit.

SHDR++
Start het programma voor schrijven van een (nieuw) headerblok op een magneetband. Kan alleen via de breinbaas teleks gestart worden.

TYPAD <adres> (<aantal>)++
Typt geheugen inhouden oktaal en desimaal uit (van het aktuele geheugen!).

PRIAD <adres> (<aantal>)++
Print geheugeninhouden als boven uit.

STAND++
Print een overzicht van alle levende programma's en siesteemadministraties.

REPRO++
Leest een papierband van flexowriter of telekskode in en print de inhoud op de regeldrukker af.

DUP++
Dupliseert een papierband van willekeurige kode.

dok. nr.: WOO/20

MEMO <nr>: <tekst van het bericht>++ voor het sturen van een bericht

<nr> geeft aan voor welke teleks het bericht bestemd is (0,1,2 of 3; 0= breinbaas).
De tekst van het bericht mag niet meer dan 64 simbolen bevatten (inklusief shiftsimbolen).

X X X III.4: Talen

Het siesteem en alle hulpprogramma's zijn in assemblerkode (Elan) geschreven. Het WINDOW-SYSTEM voor verwerking van eksperimentele metingen is Elan, met een samenhangende verzameling macro's uitgebreid. Verder zijn alle vertales voor de X8 in deze assemblerkode geschreven.
Daarom neemt Elan een sentrale plaats in. Als het siesteem wordt geassembleerd, wordt er tevens een assembler mee geassembleerd, waarmee dan alle assemblers en bruikbare vertalers weer in binaire programma's omgezet kunnen worden.

X X III.4.1: Elan

X X III.4.2: WINDOW-SYSTEM

X X III.4.3: Lisp

X X X III.4.1: Elan (Electrologica Language)

Dokumentatie: EL X8 handboek programmering
hoofdstukken 1 t/m 12 en
hoofdstuk 18 (BASIC TAPE)

EL X8 REF. MAN. hoofdstuk S4

Ekstra voorschriften:

1) niet toegestaan zijn de anonieme tellende en subroutinesprongen en besturingsopdrachten.
2) een programma moet op het 1e adres beginnen, de aanwijzing 'START' wordt genegeerd.
3) vaste werkruimte moet eksplisiet gereserveerd worden. Verboden is b.v. aan het eind van het programma:

```
STAPEL:
STAPEL  100: LYST:
        'END'
```

dit moet zijn:
```
STAPEL: 'SKIP'100
LYST: 'SKIP' benodigde lengte
      'END'
```

4) het siesteem heeft een voor alle programma's bruikbare routine verzameling. Namen van deze routines zijn de assembler bekend. Gebruikt een programma dezelfde namen dan moeten deze lokaal gemaakt worden. (zie III.4.1.3)
5) de parameterlijsten t.b.v. basic-tape en recordverwerking zijn op 2 punten afwijkend. (zie III.4.1.1.7)
6) er zijn een aantal ekstra assemblage mogelijkheden. (zie III.4.1.4)

X X III.4.1.1: behandeling van randapparaten

X X III.4.1.2: gebruik van werkruimte

X X III.4.1.3: beschikbare routines

X X III.4.1.4: assemblage mogelijkheden

X X III.4.1.5: tracer

dok. nr.: W00/23

X X X III.4.1.1: behandeling van randapparaten

X X III.4.1.1.1: teleksen

X X III.4.1.1.2: drum

X X III.4.1.1.3: bandlezer

X X III.4.1.1.4: bandponser

X X III.4.1.1.5: regeldrukker

X X III.4.1.1.6: klok en wekker

X X III.4.1.1.7: magneetbanden

X X III.4.1.1.8: IKOB

X X III.4.1.1.9: het definiëren van een eigen stroom

X X X III.4.1.1.1: teleksen

Elke PM kan elke teleks bereiken. De standaard infostroomnamen zijn: TELPo, TELP1, TELP2, TELP3 en COTEL. De infostroom COTEL heeft voor een PM altijd betrekking op die teleks waarover de PM gestart werd.
Een PM mag naast elkaar de stromen COTEL en de telpstroom van dezelfde teleks onafhankelijk aanroepen.
TELPo is de z.g. breinbaas-teleks waarover alle berichten betreffende het opzetten van tapes, het inzetten van nieuw papier of ponsband enz. verschijnen. Deze stroom kan ook gebruikt worden door de standaardroutines die een PM aanroept zodat hier de volgende beperking geldt:

regel: een PM mag niet een vraag stellen over TELPo en zonder eerst het antwoord binnen te halen andere in- of output-routines aanroepen.

Behalve de standaardaanroepen SUBC(TELPn)"nexta en
A=TELPn
SUBC(:EXAMN)"looka
is ook ter beschikking: A=TELPn
SUBC(:ULOOKA)

die met N-konditie terugkeert als er geen-
met Y- " en in A het simbool als er al een regel is ingetypt (dus niet het gehele antwoord.)

Men kan programmaties een reeds gestelde vraag weer intrekken door SUBC(:CONCL) aan te roepen voor deze stroom.

Een PM die met NEXTA of LOOKA om input vraagt die er nog niet is wordt geblokkeerd tot de eerste regel binnen is. Dit geldt niet voor de aanroep VLINDER(=SUBC(:ULOOKA)).

Antwoord op een machinevraag

Als antwoord op een geprogrammeerde vraag kunnen willekeurig veel regels ingetypt worden. De PM wordt gedeblokkeerd als de eerste regel af is en krijgt deze ook direkt binnen. De PM moet de eerste regel verwerkt hebben, voordat door de typist de 9^e regel ingetypt wordt. Is de PM niet op tijd dan wordt i.p.v. de TWNR van de 8^e regel een +W door de machine getikt, ten teken dat de typist moet wachten. Zodra het programma deze regel heeft afgewerkt herhaalt de machine de laatst ingetypte regel (en voegt een + toe) ten teken dat de typist weer verder kan gaan. (het maximum van 9 geldt voor alle teleksstromen van 1 PM gezamenlijk.)

Herroepen van een antwoord

Men kan de regel waar men mee bezig was herroepen door +≠ te typen. Als dit niet de enige regel van het antwoord is, is ook niet het gehele antwoord herroepen: de PM blijft dus input vragen tot ++ wordt ingetypt - en merkt niet dat er een regel herroepen en overnieuw gedaan is.

Herroepen van een vraag door de machine

Een PM kan een reeds gestelde vraag weer inslikken door de CLOSE van deze stroom aan te roepen. De vraag wordt dan wel uitgetikt als hij nog niet uitgetikt was en ook als het antwoord erop uitgesteld is maar wordt besloten met +C (CLOSE). Op deze mededeling wordt dus geen reaktie verlangd.

dok. nr.: WOO/25

Uitstellen van een antwoord

Door +U te typen kan men het antwoord op een machinevraag uitstellen. Na het command HERHAAL wordt de vraag dan weer opnieuw gesteld (waarop evt. het antwoord weer uitgesteld kan worden). Heeft men het antwoord op vragen van verschillende programma's uitgesteld dan reproduseren achtereenvolgende aanroepen van HERHAAL de vragen in volgorde van uitstellen.

Onderbreken van een antwoord

Je kunt je antwoord onderbreken om later op het punt waar je gebleven was verder te gaan door ook +U te typen. HERHAAL typt in dit geval de reeds ingetypte simbolen weer terug (in zwart). Als de antwoordregel eerst herroepen en daarna onderbroken wordt typt HERHAAL slechts een + uit (de regel bevat geen simbolen).

X X X **III.4.1.1.2: drum**

Elke PM kan de gehele drum lezen en alleen de files van de gebruiker beschrijven. Het (absolute) beginadres en de lengte van een file kunnen met ZOEKFILE (zie routines) opgevraagd worden.

Openen van de stroom: (Er is er altijd maar 1, deze gebruikt geen werkruimte)

```
SUBC(:PREPR)
 JUMP(1)
  :DRUM1
```

sluiten: A=DRUM1
SUBC(:CONCL) " er wordt gewacht op de afloop van alle nog hangende opdrachten.

Voor het geven van transportopdrachten zijn er 2 soorten aanroepen:

1) opdracht: A=:LYSTJE ; SUBC(DRUM1)
   afmelding: A=DRUM1 ; SUBC(:RDY)

   Men mag hierbij een willekeurig aantal opdrachten achter elkaar geven.
   De RDY-aanroep geeft echter bij terugkeer aan dat de oudste opdracht klaar is.

2) opdracht: A=:LYSTJE ; SUBC(:TROMMEL)
   afmelding: A=DRUM1 ; G=<kernadres> ; SUBC(:CLEAR)

   Ook hierbij mag men een willekeurig aantal opdrachten aanhangen. Bij de CLEAR wacht men echter op de afloop van een spesiale opdracht, bepaald door het kernadres in G (alle voorgaande opdrachten worden dan ook afgewacht).
   Men kan alleen een CLEAR doen van een TROMMEL-aanroep en alleen RDY van een DRUM1 -aanroep.
   Men mag beide soorten door elkaar gebruiken maar dit is niet aan te raden.

In beide soorten is de vorm van LYSTJE:<le drumadres>
<kernadres evt. + BIT 18>
<aantal woorden>

Bit 18 geeft aan dat het een schrijfopdracht is.

X X X III.4.1.1.3: bandlezer

Naam van de stroom: TARE1

Openen: SUBC(:PREPR)
JUMP(3)
:TARE1
:INSPT/:FLEXI/:ATELI/:DIRCT "omkodering
:labelnooduitgang

Andere omkoderingen zijn niet toegestaan. Bij onbekende omkodering wordt DIRCT gelezen.

sluiten: A=TARE1
SUBC(:CONCL)

aanroepen: SUBC(TARE1) " leest 1 simbool in A (NEXTA)

A=TARE1
SUBC(:EXAMN) " kijkt 1 simbool vooruit (in A) (LOOKA)

nooduitgang: als A pos. dan simbool met foute pariteit

als A neg. en BIT 25 = 0 --> einde band nexta
= 1 --> " " looka

als men hierbij GOTOR(MC[-1]) doet, komt men weer onder de aanroep van TARE1 of EXAMN terug.

X X X III.4.1.5: de tracer

Omdat het in multirun ontoelaatbaar is een programma te testen met behulp van de schakelaars, is een trace fasiliteit ingebouwd. Het geschiedt in prinsipe door het geven van een ingreep* bij iedere opdracht van het te tracen programma. De programmatekst verandert dus niet, de beïnvloeding vindt alleen plaats in tijd. Voor gevallen waar dit niet toelaatbaar is (langlopende PM's b.v.) bestaat de mogelijkheid om het zonder ingreep te doen.
De tracer zelf is niets anders dan een stukje supervisor programma, wat op verzoek toegevoegd wordt aan de PM, zonder konsekwenties voor het programma. Elke PM kan aan elke teleks getraced worden, onafhankelijk van andere lopende tracers. In de WACHTER wordt met de aanwezigheid van zo'n begeleider rekening gehouden.
Om te beginnen moet men in plaats van START <pm naam>, TRACE <pm naam> tikken. TRACE zoekt alleen op de drum. Als het programma al gestart is zal de mededeling <pm naam> NIET TE TRACEN verschijnen.
De tracer komt terug met GESTOPT '00000'+. Er is dan nog geen instruktie van de PM uitgevoerd. Iedere keer als de tracer stopt volgt deze vraag waarbij het getal het adres is waar gestopt is. Dit adres is relatief t.o.v. het beginadres. Ook alle adressen die men opgeeft worden zo geïnterpreteerd (uitzondering is ABS; zie verder).

* Het veroorzaken van een ingreep kan heel eenvoudig door het zetten van het IF bitje van een niet aanwezig apparaat. Jammer genoeg zit er een eigenaardigheid in de hardware, die het onmogelijk maakt het programma na een GOTOR opdracht op te vangen. Het blijkt dat ongeacht de te herstellen toestand de machine hierna gedurende minstens 1 opdracht doof is.

Als antwoord op de vraag GESTOPT zijn er de volgende mogelijkheden; tussen () is fakultatief.

<adres> (aantal)	men krijgt de inhoud te zien van het adres en de eventueel volgenden.
<adres> : inhoud	de : geeft aan dat men wil wijzigen.
.	het adres wordt gevuld met de gegeven inhoud.
<adres> : inhoud inhoud inhoud : :++	vanaf het gegeven adres vindt wijziging plaats zoveel keer als er inhouden gegeven worden.
ABS <adres> enz.	wil men absoluut kijken of wijzigen dan kan dit door ABS ervoor te gebruiken. Verder gelden dan dezelfde regels.

Het wijzigen en bekijken van adressen is geheel gebonden aan de regels van de geheugenprotektie!

A	laat de inhoud van het A register zien.
of B,D,F,G,S,T	idem
A : <inhoud>	wijzig de inhoud van het A register. Idem voor B,D,F,G,S
	het wijzigen van T slaat alleen maar op het adresdeel.
C of K	geeft de konditie (YES of NO)
C : YES of NO	wijzigt de konditie overeenkomstig
of K : YES of NO	

Al deze kommando's keren direkt terug zonder dat de PM zelf aan bod komt.
De volgende antwoorden hebben wel direkt tot gevolg dat het te tracen programma doorlopen gaat worden.

STOP	doe één opdracht, hierna volgt weer GESTOPT 'adres'
STOP <adres> (sikli)	komt terug met GESTOPT als het adres bereikt is door de PM. De daar staande opdracht is nog niet uitgevoerd. Geeft men een aantal sikli dan wordt pas teruggekeerd nadat het punt zoveel maal gepasseerd is.
RUN	de tracer geeft geen ingrepen meer; de PM wordt vrij gelaten. Men kan hem te pakken krijgen door TRACE <pm naam> te zeggen.
RUN <adres>	op het adres wordt een trace aanroep geplaatst, de oude inhoud wordt gered en het programma krijgt de vrijheid. Wordt de aanroep bereikt dan volgt de bekende mededeling. De oude inhoud is dan weer hersteld.
KLAAR	de tracer nokt de PM, dus ook zichzelf.

X X X III.4.1.4: assemblage mogelijkheden

1. het programma
2. het gebruik
3. het window-siesteem
4. vensterprogramma's

het programma

1. alle werkruimte die een programma nodig heeft moet eksplisiet gereserveerd worden, d.w.z. werkadressen, buffers en de stapel (zie talen).
2. het programma begint altijd op het 1e adres. De 'START' aanwijzing wordt genegeerd.
3. 'INLUD' : XX
   d.m.v. deze aanwijzing kan men tijdens assemblage een stuk programma doorlopen, wat begint op XX. De interlude moet eindigen met een GOTOR(LINK). Het stapelgebruik is onbeperkt. Eventuele drumtransporten moeten gedaan worden met SUBC(:TROMMEL). Men moet er wel voor zorgen dat in de interlude geen namen gebruikt worden die op het moment dat de 'INLUD' gedaan wordt, nog niet bekend zijn.
4. 'LINK' : XX
   hiermee kan men files aan elkaar linken tijdens assemblage. Op XX staat een string die opgevat wordt als de naam van een file. De assembler zal nu verder gaan vanaf het begin van deze nieuwe file. Men moet eraan denken genoeg 'END's' achter het laatste stuk te zetten, omdat bij elke LINK in prinsipe minstens één 'END' niet meegenomen wordt. Het belang van deze assemblage-instruktie zit vooral in de mogelijkheid hiermee een eigen profile te gebruiken. Dit is een verzameling van makro's en namen die men zelf handig vindt om te gebruiken. De profile wordt dan voorafgaande aan een programma geassembleerd, waarna in het programma alle gedefinieerde makro's gebruikt kunnen worden.

Het gebruik

Er zijn twee assemblers. AS die van drum assembleert en AST die vanaf tape werkt. Beide eisen z.g. MONT-files. De gebruiksaanwijzing is voor beide hetzelfde.
Vragen en antwoorden:

```
v          FILENAAM...++
a          <filenaam>

v          GEGEVENS...++
a  <beginadres>,<lengte>,<aanwijzing namen>,<aanwijzing
                                              labellijst>
```

beginadres : alleen te geven als het programma op een vaste plaats geassembleerd moet worden.

lengte : de lengte in kilo's die het programma plus werkruimte zal beslaan.

aanwijzing namen : = KORT, alleen te geven als men met korte namen wil assembleren.

aanwijzing labellijst : LIPR of TELP of niets. Geeft aan over welk apparaat men de labellijst plus foutmeldingen wil hebben. Bij niets komen alleen de fouten over de teleks.

Na assemblage volgt een printje:
BA '<adres>' LE '<lengte assemblaat>' LW '<lengte werkruimte>' <idem desimaal>
LW is de lengte van de werkruimte die nog over is.

```
v          NAAM...++
a          <naam voor binfile>,<lengte>
```

naam voor binfile : het assemblaat zal als binaire file op de drum gezet worden onder de gegeven naam.

lengte : alleen geven als men de lengte van het programma wil veranderen (vanwege de werkruimte b.v.).

v FILE OVERSCHRIJVEN...++

a <YES of NO>

Deze vraag verschijnt als er al een file bestaat die de gegeven naam heeft. Bij NO zal om een andere naam gevraagd worden.

Afhankelijk van het feit of het programma vast is of niet, zal een tweede assemblage siklus doorlopen worden. Hierbij worden geen vragen gesteld. Wel heeft AST na de eerste siklus de tape "geclosed", zodat een nieuwe open tape zal volgen. Bij beëindiging wordt uitgetikt:

<naam binfile> KLAAR Ten teken dat het programma gedumpt is onder deze naam en nu gestart kan worden. NOGO betekent dat er assemblagefouten in het programma zitten. Er wordt geen binfile afgeleverd.

In de labellijst vindt men de volgende lay-out:

regelnr. relatieve adres adresseringstype naam van de label

b.v. 5 '00241' 0 JAN

Het adresseringstype kan zijn 0 voor statiese, 1 voor M(B), 2 voor 'MT' en 3 voor de dynamiese adressering.

Lange labelnamen worden bij het afprinten afgeknipt op 14 simbolen met een punt erachter. Als men assembleert met korte namen, worden overeenkomstig de 1^e drie en de laatste drie simbolen gedrukt waartussen een punt.
De error uitprint heeft de betekenis:

regelnr. ERROR foutnummer relatieve adres evt. naam

De verklaring van het foutnummer is te verkrijgen met het command ERROR <foutnummer>.
Bij errors die niet slaan op een label is de naam de laatstbekendgemaakte label. Het kan zijn dat die er niet is, dan volgt ook geen naam.

Het window-siesteem

Voor de assemblage van het window-siesteem is een aparte

assembler, STORA, gemaakt. Deze staat vast op '62000', reserveert altijd vanaf '36000' en kent de aanwijzing 'STORE'.
Vragen en antwoorden:

v FILENAAM...++

a <filenaam>, <aanwijzing medium>

aanwijzing medium: TAPE of DRUM of niets.
Geeft aan waar de file zich bevindt. Niets betekent TAPE.

v LABELLIJST...+++

<aanwijzing labellijst>
aanwijzing labellijst: LIPR of TELP of niets.
De labellijst + de foutmeldingen zullen verschijnen over het gegeven apparaat.
Bij niets komen alleen de fouten over de teleks.

'STORE'
Dit heeft tot gevolg dat het resultaat van assemblage + de assembler zelf op de drum gedumpt worden als binfile onder de naam WINDAS. Er volgt een uitprint STOREFILE WINDAS en de assembler stopt. WINDAS kan nu gebruikt worden voor de assemblage van venster-programma's. 8K groot.
Het WINDOW-siesteem zelf heeft een file WINNIE gekreëerd die de ROUTZOEKER routines bevat. De labellijst bevat de absolute inzetadressen in plaats van de relatieve, zoals bij de gewone assembler.

Vensterprogramma's
Om een vensterprogramma te assembleren moet men WINDAS starten. Deze is 17 K groot en begint op '36000'. Het resultaat is een programma wat ook op '36000' begint en maksimaal 17 K groot is.

Vragen en antwoorden:

v FILENAAM...++

a <filenaam>,<aanwijzing medium>

aanwijzing medium: TAPE of DRUM of niets.
Geeft aan waar de file zich bevindt. Niets betekent DRUM.

v LABELLIJST...++

a <aanwijzing labellijst>

aanwijzing labellijst: LIPR of TELP of niets.
De labellijst verschijnt over het gedefinieerde apparaat.
Bij niets worden de fouten over de teleks gemeld.

v NAAM...++

a <naam voor binfile>,<lengte in kilo's>

lengte: alleen geven als men niet de volle 17 K wil hebben.

De labellijst bevat de absolute adressen van het programma en moet gekombineerd worden met die van WINDAS.
NOGO betekent dat er nog assemblagefouten in het programma zitten. Bij het starten van een vensterprogramma moet de file WINNIE op de drum staan.

X X X III.4.1.1.4: bandponser

Naam van de stroom: TAPU1

```
Openen: SUBC(:PREPR)
          JUMP(3)
           :TAPU1
       :FLEXO/:ATELO/:DIRCT    " omkoderingsmogelijkheden
            :BPEND             " standaard
```

Andere omkoderingen moeten voor de nexta-aanroep plaatsvinden.

```
nexta   : A=simboolwaarde
          SUBC(TAPU1)

sluiten : A=TAPU1
          SUBC(:CONCL)
```

Bij het openen van de stroom worden 2 dozen van lengte 128 uit de PM-werkruimte genomen, die bij sluiten weer teruggegeven worden.
Het openen en sluiten hebben beide het ponsen van een riedeltje blank tot gevolg.

X X X III.4.1.1.5: regeldrukker

naam van de stroom: **LIPR1**

```
openen:   SUBC(:PREPR)
          JUMP(3)
          :LIPR1
          :LIPCO    "omkodering en nooduitgang zijn
          :NEWLI     standaard uitgevoerd"
```

Men mag daarom ook gebruiken

```
          SUBC(:PREPR)
          JUMP(1)
          :LIPR1

nexta:    A=simbool
          SUBC(LIPR1)

sluiten:  A=LIPR1
          SUBC(:CONCL)
```

Bij het openen worden 2 dozen van 128 aangevraagd uit de pm ruimte. Bij het sluiten worden deze weer vrijgegeven. Het sluiten geeft een bladopvoer.
(voor omkodering, zie bijlage 1)

Bijlage 1 X8 kode → ASCII of LIPCO

X8 (desimaal)		simbool	ASCII (oktaal)	representatie op de regeldrukker
0	0	0	260	0
1	1	1	261	1
2	2	2	262	2
3	3	3	263	3
4	4	4	264	4
5	5	5	265	5
6	6	6	266	6
7	7	7	267	7
8	10	8	270	8
9	11	9	271	9
10	12	A	301	A
11	13	B	302	B
12	14	C	303	C
13	15	D	304	D
14	16	E	305	E
15	17	F	306	F
16	20	G	307	G
17	21	H	310	H
18	22	I	311	I
19	23	J	312	J
20	24	K	313	K
21	25	L	314	L
22	26	M	315	M
23	27	N	316	N
24	30	O	317	O
25	31	P	320	P
26	32	Q	321	Q
27	33	R	322	R
28	34	S	323	S
29	35	T	324	T
30	36	U	325	U
31	37	V	326	V
32	40	W	327	W
33	41	X	330	X
34	42	Y	331	Y
35	43	Z	332	Z
36	44	10	202	E
37	45	'	247	'
38	46	.	256	.
39	47	;	273	.
40	50	,	254	,
41	51	:	272	%
42	52	(	250	(
43	53	)	251	)
44	54	[	333	(
45	55	]	335	)

dok. nr.: WOO/40

Bijlage 1 (vervolg)

X8 (desimaal)		simbool	ASCII (oktaal)	representatie op de regeldrukker
46	56	+	253	+
47	57	-	255	-
48	60	x	252	*
49	61	/	257	/
50	62			%
51	63	↑	336	E (EXP)
52	64	=	275	=
53	65	<	274	(
54	66	>	276	)
55	67			#
56	70	←	337	(
57	71	@	300	◊
58	72	"	242	'
59	73	$	244	D
60	74	space	240	space
61	75	underlining	220	_
62	76	shift out	216	#
63	77	error	225	#
64	100	#	243	◊
65	101			◊ (PROPJE)
66	102	∨	231	%
67	103	^	232	&
68	104	¬	233	D
69	105	\	334	\
70	106			' (graad)
71	107	&	246	&
72	110	?	277	◊
73	111	!	241	◊
74	112	a	341	A
75	113	b	342	B
76	114	c	343	C
77	115	d	344	D
78	116	e	345	E
79	117	f	346	F
80	120	g	347	G
81	121	h	350	H
82	122	i	351	I
83	123	j	352	J
84	124	k	353	K
85	125	l	354	L
86	126	m	355	M
87	127	n	356	N
88	130	o	357	O
89	131	p	360	P
90	132	q	361	Q

Bijlage 1 (vervolg)

X8 (desimaal)		simbool	ASCII (oktaal)	representatie op de regeldrukker
91	133	r	362	R
92	134	s	363	S
93	135	t	364	T
94	136	u	365	U
95	137	v	366	V
96	140	w	367	W
97	141	x	370	X
98	142	y	371	Y
99	143	z	372	Z
100	144	-0		0
101	145	-1		1
102	146	-2		2
103	147	-3		3
104	150	-4		4
105	151	-5		5
106	152	-6		6
107	153	-7		7
108	154	-8		8
109	155	-9		9
110	156	%	245	%
111	157	blank	200	space
112	160	bell	207	#
113	161	altmode	375	#
114	162	back space	210	#
115	163	horizontal tab	211	space
116	164	line feed	212	NLCR
117	165	vertical tab	213	VERTICAL TAB
'166' = 118	166	form feed	214	form feed
119	167	carriage return	215	NLCR
120	170	unit separator	237	NLCR
121	171	record separator	236	NLCR
122	172	group separator	235	NLCR
123	173	file separator	234	#
124	174			#
125	175		374	#
126	176	shift in	217	#
127	177	delete/rubout	377	wordt weggegooid

Negatieve kodes en kodes groter dan 126 worden weggegooid.

X X X II.3.4: programma's

De hoofdeigenschap van een programma is, dat het niet resident is. Het kan geswapt of definitief verwijderd worden. Er zijn twee soorten programma's:

1) De commands (of handbedieningsopdrachten) die deel uitmaken van het siesteem.
   Ze hebben bepaalde privileges zoals de mogelijkheid om in besturingstoestand te komen. De besturingstoestand wil zeggen dat de geheugenprotektie dan opgeheven is.
   Ze worden met wammes mee geassembleerd.

2) De andere programma's.
   Het aanmelden van deze PM's geschiedt altijd met START <naam binfile>. Bij de binfile staat hoe groot het programma is, het startadres, en of het relokateerbaar is (dit zijn nagenoeg alle PM's).
   Start zoekt nu het kleinste gaatje in de geheugenbezetting waar de PM inpast. Daarbij is bovendien een voorkeur voor ruimte zonder opponenten. Dit om de overhead aan swaptijd zo klein mogelijk te houden.
   Kan de PM geplaatst worden dan wordt een LOG gereserveerd, waarin alle belangrijke gegevens over het programma genoteerd worden. Hierin worden ook de registers e.d. gered bij onderbreking van de PM.
   Als het programma een fout begaat (b.v. schrijven buiten zijn gebied) wordt het ogenblikkelijk genokt. Er wordt een foutkode uitgetikt en de ruimte en de LOG worden vrijgegeven. Alle apparaten en stromen worden netjes gesloten. Alleen eventuele drumfiles blijven bestaan. De binfile van een programma kan alleen maar via assemblage ontstaan.

X X X III.4.1.1.9: het definiëren van een eigen stroom

Elk programma heeft de beschikking over twee simboliese stromen: INPUT en OUTPUT.
Men kan met deze stromen een programma schrijven met variabele in-outputstroom, b.v. om als output apparaat bij testen regeldrukker te kiezen en dit dan later te wijzigen in bandponser.

Instellen: A=COTEL/TELPo/.../TELP3/TAPU1/LIPR1/eigen output
OUTPUT=A

resp: A=COTEL/TELPo/.../TELP3/TARE1/eigen input
INPUT=A

Bij het starten van een programma worden ze beide op COTEL gezet.

Openen:
SUBC(:PREPROUTPUT)
resp.: SUBC(:PREPRINPUT)

N.B.1: als OUTPUT=TAPU1 dan geldt als omkodering :FLEXO
2: " INPUT =TARE1 " " " " :INSPT
en einde band wordt aangegeven door de NEXTA en LOOKA met A=121 (vlinder) te laten terugkeren.

Nexta: SUBC(OUTPUT)
resp.: SUBC(INPUT)

Looka: SUBC(:LOOKIN)

Sluiten: SUBC(:CONCLOUTPUT)
resp.: SUBC(:CONCLINPUT)

Als OUTPUT en INPUT een teleksstroom voorstellen, verloopt alles presies zoals in III.4.1.1.1 beschreven.
Men kan ook een eigen stroom definiëren. Deze moet op z'n minst uit een nexta-routine bestaan (d.i. een routine die bij OUTPUT in A telkens een simbool aangeboden krijgt en

verwerkt en bij INPUT telkens in A een simbool moet afleveren). De stroom mag ook een prepr-, concl-, lookaroutine hebben.

Algemeenste geval: A=:LYSTJE

A + BIT 25 " stroom heeft een preprroutine

A + BIT 24 " " " " concl-routine

A + BIT 22 " invoerstroom heeft een lookaroutine

INPUT=A

(Het zelfde geldt voor OUTPUT, alleen is er daar geen lookaroutine.)

Het LYSTJE moet er dan als volgt uitzien:

```
          GOTO(:PREPRROUTINE)
          GOTO(:CONCLROUTINE)
LYSTJE:GOTO(:NEXTAROUTINE)
          GOTO(:LOOKAROUTINE)
```

Heeft een stroom geen preprroutine dan moet A + BIT 25 weggelaten worden, zo ook A + BIT 24, resp. A + BIT 22, als de stroom geen concl-, resp. lookaroutine heeft.

In het allerkarigste geval bestaat de stroom alleen uit een nextaroutine en kan dan ingesteld worden door:

```
A=:NEXTAROUTINE
OUTPUT=A
```

N.B! De nexta- en lookaroutine moeten COUNT,LINK, F en D ongewijzigd teruggeven.

Voor IN- en OUTPUT zijn een aantal stroomonafhankelijke routines geschreven die een programma mag aanroepen, zoals, DRUKDATUMTIJD, TEKST, READ, READNUMBER, GETAL, OKTAAL, INTEGER, SPATIES, YESORNO, etc. (zie routines).

Als een stroom geen prepr- of concl-routine heeft, hebben preprinput en conclinput geen effekt. Als er een LOOKA gedaan wordt van een stroom zonder looka, volgt een PM-fout.

X X X III.4.1.2: gebruik van werkruimte

Het geheugentrajekt van een programma bevat altijd een stuk werkruimte (zie dozen).

Hieruit nemen infostroomroutines (zie CM's) buffers en ook de PM kan hieruit trajekten opvragen:

aanroep S = aantal benodigde woorden
SUBC(:DOOS)

levert af: S onveranderd (lengte)
A = beginadres beschikbare trajekt
F onveranderd

teruggeven van een gebruikte doos:

S = lengte
A = beginadres
SUBC(:DOOSVRIJ) " wijzigt A en S

Meegeven van de lengte bij teruggave is nodig voor een kontrole of er niet buiten de doos geschreven is. Knoeit een PM in z'n werkruimte of ontstaat er een tekort, dan volgt een PM-fout.
Bij elke aanvraag wordt steeds de doos uit de eerste vrije doos die groot genoeg is gepakt. De manier om de werkruimte optimaal te gebruiken is stapelen volgens het LIFO-prinsipe (LAST IN FIRST OUT), d.w.z.: de laatst aangevraagde doos het eerst teruggeven. (Hieraan moet men dus bij het opzetten van het blokschema van een programma denken!)

X X X III.4.1.1.6: klok en wekker

De instelling van de klok vindt plaats bij LSNB. Er wordt dan gevraagd om de datum en de tijd. Het siesteem komt pas op gang als op deze vraag antwoord is gegeven.
De klok loopt altijd. Hij is uit te lezen met
A = TIJD of A = KLOK

de wekker
naam van de stroom : WEK
openen : SUBC(:PREPR)
JUMP(1)
:WEK
gebruikt geen werkruimte

aanroep A = positief (ongelijk Ø)
SUBC(WEK)

Het programma blokkeert zichzelf en wordt gewekt op het absolute tijdstip dat in A meegegeven is.

aanroep A = - aantal wekeenheden
SUBC(WEK)

1 wekeenheid is 10 millisekonden

De PM blokkeert zichzelf en wordt na de gegeven periode weer gewekt.
v.b. A = -100
SUBC(WEK)

1 sekonde na de aanroep gaat het programma onder de aanroep verder.
Hieruit volgt dat 1 min 6000 eenheden is en
1 uur 360.000

N.B.: Het adres DATUM is voor ieder programma te lezen.
De kodering is jaartal, maand, dag.
V.b.: 26 april 1969 is 690426.

60.00

X X X III.6: foutmeldingen

Er bestaan verschillende soorten foutmeldingen:

1) PMFOUT <foutnummer> <pm naam>
   De PM heeft een fout begaan en is daarom genokt. Verklaring van de fout verkrijgt men m.b.v. het command PMFOUT <foutnummer>
2) TAPE <unit nummer> FOUT <foutnummer> (evt. +)
   Er is een tapefout gekonstateerd. Als er een ijzeren kruis achter volgt dan is het een vraag. In het algemeen moet de tape dan vervangen worden of is de schrijfring niet aanwezig. Is de noodzakelijke handeling verricht, dan mag de vraag doorgeslagen worden. (De PM is dan niet genokt.)
   De verklaring van de fout krijgt men met TAPEFOUT <foutnummer>
3) ERROR <foutnummer>
   Dit treedt alleen op bij assemblage. Voor de verklaring geve men het command ERROR <foutnummer>.
4) GENOKT DOOR BREINBAAS <pm naam>
   De breinbaas heeft m.b.v. de sleutel de pm genokt. Verklaring bij de breinbaas.
5) FOUTADRES <pm naam>
   De pm heeft een foutadres ingreep veroorzaakt.
6) PARITEITSFOUT <pm naam>
   Er is een pariteitsfout gekonstateerd in het geheugen.
7) PROTEKTIEFOUT <pm naam>
   De pm heeft een aanslag gepleegd op de geheugenprotektie.
8) GENOKT <pm naam>
   Het programma is door de gebruiker via de teleks genokt.
9) EINDE <pm naam>
   De pm heeft zichzelf beëindigd door GOTO(:STOP)

Al deze boodschappen betekenen dat het programma verwijderd is (uitzondering onder punt 2). Eventuele output kan nog volgen.

X X X III.4.1.1.7: magneetbanden

Naast het basictape siesteem is ook een eenvoudige recordverwerking aangebracht. Beide siestemen werken onafhankelijk, maar mogen voor dezelfde file tussen openen en sluiten niet door elkaar gebruikt worden.

X X III.4.1.1.7.1: basic tape

X X III.4.1.1.7.2: eenvoudige recordverwerking

X X X III.4.1.1.7.1: basic tape

Verloopt geheel volgens hoofdstuk 18 EL X8 handboek, behoudens een wijziging in programlist en identificationlist. Bij basic tape moet het programma zelf buffers reserveren.

file description bestaat niet

een vermelding als 'FILE' 'BEGIN'1
INPUTTAPE,256,0
'END' die 70 geheugenplaatsen in beslag neemt.
kan men vervangen door:
INPUTTAPE:'SKIP'2

open file aanroep: SUBC(:OPNFIALG)
JUMP(3)
:INPUTTAPE " filenaam
:PROGRAMLIST
:IDENTIFICATIONLIST

programlist: 0/:MATA1/.../:MATA4 ": naam informatiestroom
0
:READT/:RITET/:DIRCT
<maximale recordlengte>

opmerkingen: men krijgt alleen de tape-unit, door het nummer van de informatiestroom aangegeven, (dus UNITO bij :MATA1, enz.) als deze unit vrij is, anders de eerste vrije. Met de kode DIRCT kan men een willekeurige tape lezen of beschrijven (is dus zeer gevaarlijk!), men moet dan zelf geheel voor lezen en/of schrijven van een headerblok zorgen.

identificationlist: 0
<aantal records per blok> "=1 voor " andere dan EL-kode
density

close file: G=:INPUTTAPE " :filenaam
SUBC(:ALGLOFE)/SUBC(:ALGCLOFI)
(eerste geeft REWIND, tweede REWIND UNLOAD)

andere aanroepen: (voor werking zie hoofdstuk 18)

```
G=:filenaam
S=:lijstje
SUBC(:READB)              " lees een blok

G=:filenaam
S=:lijstje
SUBC(:WRITEB)             " schrijf een blok

G=:filenaam
SUBC(:RITEMB)             " schrijf tapemark

S=aantal te spoelen blokken  (neg. bij terugspoelen)
G=:filenaam
SUBC(:SPACEB)

G=:filenaam
SUBC(:REWINB)             " rewind

G=:filenaam
SUBC(:REWUNB)             " rewind unload

G=:filenaam
S=:lijstje
SUBC(:CHECKB)             " wacht afmelding van een vo-
                            rige opdracht
```

daar zijn bijgekomen:

```
G=:filenaam
SUBC(:SCHRIJFHEADER)   " hiervoor moet de identificationlist
                       " uitgebreid zijn met
                       " datum, bewaarperiode, headernaam
                       " deze routine schrijft een header-
                         blok en tapemark

G=:filenaam
SUBC(:SLUITBLOK)       " schrijft tapemark, sluitblok, tape-
                                                       mark
                       " hierna moet de file nog door het
                       " programma zelf gesloten worden
```

X X X III.4.1.1.7.2: eenvoudige recordverwerking

Deze behandeling is het eenvoudigste in gebruik en is toereikend voor verreweg de meeste programma's. De werking van de verschillende routines is geheel overeenkomstig de EL-recordverwerking zonder p-wijzers, met slechts 1 record tegelijk in bewerking.

file description: bestaat niet
ook hier bij voorkeur zetten

filenaam : 'SKIP'2 in plaats van 'BEGIN''FILE' etc.

openen:
```
SUBC(:OPNFIALG)
  JUMP(3)
   :filenaam
   :programlist
   :identificationlist
```

programlist:
```
0/:MATA1/.../:MATA4
0
:READT/:RITET
 maks. recordlengte (in woorden)
 :label bijzondere toestand
```

identificationlist:
```
:ELTA7/:BINTA/:BCDTA
 maks. aantal records per blok
   density
   0/datum
   bewaarperiode
   '('headernaam')' " 10 simbolen!
```

Verwerking van het, altijd aanwezige, BCD-headerblok gebeurt standaard.

Kontroles: bij lezen op headernaam
" schrijven op schrijfring
en bewaarperiode

Bij konstatering van een fout: foutkode gevolgd door + waarop de breinbaas een goede tape op de unit kan zetten.

dok. nr.: WOO/53

sluiten: G=:filenaam
SUBC(:ALGCLOFE) " inklusief REWIND
of SUBC(:ALGCLOFI) " " REWIND UNLOAD

label bijzondere toestand wordt aangesprongen bij einde band. Hier kan een REWIND- of sluitroutine aangeroepen worden

nexrecord lezen: SUBC(:filenaam)
levert af: A=laatste woord record
S=lengte
G=beginadres -1

schrijven: S=lengte " van het gewenste vrije record
SUBC(:filenaam)
levert af: G=beginadres -1 van een nieuw, vrij record

ready record: G=:filenaam [3] " gehandhaafd voor de kompatibiliteit
SUBC(:READYALG)

transcribe record: A=beginadres input record -1
S=recordlengte
SUBC(:TRANSALG)
JUMP(2)
:filenaam input
:filenaam output

Recordverwerking pakt 2 buffers uit de PM-werkruimte ter lengte: {maks. recordlengte +3}

Basic tape gebruikt geen PM-werkruimte.

Bijlage 2

de LOTUS; enkele hulpprogramma's

MONT1	tekst-montage programma, zie verhaal simboliese biblioteek (aug. '67)
MONT2	idem
MONT3	idem
MONT21	MONT2 die 1 record per blok schrijft i.p.v. 7
MONT4	maakt van een mont-teksttape een gelayoute versie (tape → tape)
LAYOUT	idem voor papier naar papier en/of tekst (zie beschrijving)
MONTDI	montdirct; montage programma voor pap-archief beschrijving 690118 000 AM/SO/1
REPRAS	print een ASCII bandje over de regeldrukker
PFILE	print-file van drum
PFILID	print file identifikatieblokje van drum of tape
PTAPE	print tape
FILTER	montage programma voor dump-tapes
BIGBRO	big brother; testprogramma voor het F-register
TESTRD	test regeldrukker
TESTTL	test teleks (drukwerk)
TESTTA	test tape
BAKSOR	baksort; sorteerprogramma voor elektronisi
LHDR	leesheader; leest BCD-header van een tape, telt verder het aantal blokken
AS	assembler voor drum (mont-file)
AST	assembler voor tape (mont-file)
STORA	assembler voor window-siesteem
WINDAS	assembler voor vensterprogramma's
WINNIE	file met routzoekerroutines voor vensterprogramma's
TJOUR	print journaal van window tape
DUMPRO	file transport voor window files

SUBC(:20 EXFILE) F: FILENAAM A,S,F N, NIET BEKEND
Y, A: DRUMADRES
S: LENGTE

X X X II.4.1.3: beschikbare routines

AANROEP	PARAMETERS	WIJZIGT	LEVERT AF	RESULTAAT
SUBC(INPUT)	INPUT	A,S	A een simbool	voor eigen INPUT-stromen zijn de spesifikaties natuurlijk onbekend.
SUBC(:PREPRINPUT)	INPUT	A,S	-	opent gedefinieerde INPUT-stroom
SUBC(:CONCLINPUT)	INPUT	A,S	-	sluit gedefinieerde INPUT-stroom
SUBC(:LOOKIN)	INPUT	A,S		een LOOKA via INPUT
SUBC(:READ)	INPUT	A,S,F	NO	; of gelezen en geclosed
			YES S=negatief F=getal A=eindsimbool of	er is een getal gelezen
			YES S=Ø A=echte delimiter G=eindsimbool of	er is een delimiter gelezen
			YES S > Ø = aantal X8 woorden A=eindsimbool	er is een string gelezen die boven B geplaatst is. Met F=M(B) krijgt men de 1e 6 simbolen. S bevat het aantal X8 woorden van de string. De string is altijd aangevuld met 1 woord waar 3 * in staan
SUBC(:READNUMBER)	INPUT	A,S,F	NO	; of gelezen en geclosed
			YES F=getal A=eindsimbool	

AANROEP	PARAMETERS	WIJZIGT	LEVERT AF	RESULTAAT
SUBC(:READNAME)	INPUT	A,S,F	NO	; of gelezen en geclosed
			YES	string gelezen
			F=1e 6 simbolen	vanaf M B+1 is de string terug
			S=aantal X8 woorden	te vinden, aangevuld met 1 woord van 3 * (127)
			A=eindsimbool	
SUBC(OUTPUT)	OUTPUT A=een simbool	A,S	-	bij eigen OUTPUT-stroom zijn de spesifikaties natuurlijk anders
SUBC(:TEKST)	OUTPUT F=beginadres string S=aantal simbolen	A,S,F		uitvoer van een string
SUBC(:SPATIE)	OUTPUT	A,S		
SUBC(:SPATIES)	OUTPUT S=aantal	A,S COUNT		
SUBC(:PREPROUTPUT)	OUTPUT	A,S		openen van OUTPUT-stroom
SUBC(:CONCLOUTPUT)	OUTPUT	A,S		sluiten van OUTPUT-stroom
SUBC(:OKTAAL)	G=getal S=aantal oktalen OUTPUT	A,S COUNT		uitvoer van een oktaal getal
SUBC(:INTEGER)	G=getal OUTPUT	A,S		uitvoer van een integer getal

AANROEP	PARAMETERS	WIJZIGT	LEVERT AF	RESULTAAT
SUBC(:GETAL)	OUTPUT G=getal S=aantal posities	A,S,F COUNT	-	d.i. integer gelayout. De voorste nullen worden spaties
SUBC(:YESORNO)	OUTPUT, INPUT F=beginadresstring S=aantal simbolen	A,S,F	YES NO	print string af als vraag; verwacht daarna als antwoord YES of NO
SUBC(:JAOFNEE)	OUTPUT, INPUT	A,S,F	YES NO	print als vraag YESORNO; verwacht daarna als antwoord YES of NO
SUBC(:DRUKDATUMTYD)	OUTPUT	A,S,F COUNT		
SUBC(:DRUKTYD)	OUTPUT	A,S,F		
SUBC(:TRANSPORT)	A=haaladres G=brengadres S=aantal	A,S,F COUNT		transporteren van een trajekt
SUBC(:STELNUL)	A=beginadres S=aantal	A,S,F		stelt trajekt op +Ø
SUBC(:STELIN)	A=beginadres S=aantal G=getal	A,S,F		stelt trajekt op gewenste waarde
SUBC(:FLOTIN)	A=beginadres S=aantal F=getal	A,S,F		stelt trajekt op gewenste (floating) waarde
SUBC(:DOOS)	S=lengte	A,F	S=lengte A=beginadres	levert een doos af uit de werkruimte van de pm
SUBC(:DOOSVRY)	S=lengte A=beginadres	A,S,F		de doos wordt weer genoteerd als vrij doos (zie dynamiese werkruimte verdeling)