Een ponsbandorganisatie voor de X8.

Motto:"Wie het kleine niet eert,
is het grote niet weerd."

Inleiding.

In dit rapport zal het hoofdaccent vallen op het lezen van ponsband. Waarom het ponsen minder grote vragen op hoeft te werpen, zal in de loop hiervan duidelijk worden.

De bandleesopdracht van de X8 is volledig code-onafhankelijk, elke configuratie van al of niet gaatjes dringt tot de machine door en in zekere zin "is er geen probleem", nl. in die zin, dat we de verwerking van deze banden kunnen programmeren zoals we willen. Omgekeerd geeft dit ons wel de plicht om uit de baaierd van mogelijkheden dan te kiezen, wat we willen. Een groot gedeelte van dit rapport houdt zich met deze keuze (en de motivering daarvan) bezig.

Het leidende principe is, dat het voor de gebruiker zo gemakkelijk mogelijk gemaakt moet worden. Dit betekent onder andere dat wanneer een gebruiker opeenvolgende getallen van een ponsband wil lezen, dat hij dan een intern passende identificatie van numerieke waarden in zijn proces wil opnemen, zonder dat hij zich bewust hoeft te zijn van de conventies, volgens welke deze getallen op de band staan. Deze situatie is geheel analoog aan zijn instelling tegenover variabelen van type integer of real: voor hem karakteriseren deze waarden, hoe is zijn zorg niet; de interne representatie hoeft niet eens uniek te zijn. Bij een dergelijke wijze van bandlezen dienen allerlei "irrelevantia", voordat de numerieke waarde tot het programma doordringt, dus onder tafel geraakt te zijn.

De complicatie hierbij is echter, dat wij niet a priori decreteren kunnen, wat voor de programmeur irrelevant is. Ponsbanden immers leiden een zelfstandig bestaan buiten de machine, zij kunnen komen van automatische meetapparatuur, zij kunnen XXXXX voor ponsbandgestuurde machinerie gemaakt moeten worden. Waar wij dus op aansturen is een organisatie, die enerzijds het gemak dient, maar die anderzijds desgewenst de programmeur de volledige controle geeft, die hij nodig heeft om ponsband als "general purpose medium" te kunnen gebruiken.

Naast de programmeur hebben we de operateur. Wat van hem verlangd wordt, vervult mij zo mogelijk met nog meer zorg. Immers: zolang een machine in uniprogrammering met een enkele bandlezer bedreven wordt, is het identificatieprobleem zo simpel, dat wij ons dit nauwelijks bewust zijn. Tegen de tijd, dat één of meer programma's via één of meer bandlezers informatie willen XXXXXXXXX opslorpen, rijst het probleem "Who is who?" in volle omvang. Het is duidelijk, dat de taak voor de operateur ondoenlijk wordt, tenzij deze identificatie in het overgrote deel der gevallen geen extra operateurshandelingen vereist.

Een volgende complicatie is het gemengde gebruik, dat men maken wil van het physische "einde band". "Einde band" is door de X8 detecteerbaar, de consequenties, die hieraan verbonden dienen te worden, kunnen van geval tot geval echter aanzienlijk verschillen.

Er zijn programma's, waarin einde band een essentieel onderdeel van de inkomende informatiestroom is, bv. een programma, dat een willekeurige band reproduceert. (Het is in dit licht jammer, dat de ponser niet beschikt over een door de X8 bestuurbaar afsnijertje!)

In andere organisaties wil men liever aan einde band geen betekenis toekennen. Als een programma vraagt om een heel lange band, die bv. uit twee stukken is opgebouwd, dan is het natuurlijk prettig, dat de EL 1000 zorgvuldig gemaakte plakjes slikt, maar het zou jammer zijn, als dit de operateur ook inderdaad tot plakken zou verplichten. Als de operateur om een of andere reden prefereert om zo'n band in twee stukken in te leggen, zou dit moeten mogen zonder dat daarbij een extra last op de schouders van de programmeur gelegd wordt. Evenmin als men de operateur de plicht tot plakken wil geven, evenmin wil men de operateur de plicht tot knippen -dwz. het verbod tot plakken- opleggen, althans in het merendeel der gevallen, waarin hierdoor geen onduidelijkheden kunnen ontstaan. (Het is duidelijk, dat men achteraan een band, waarop "einde band" een essentieel onderdeel van de informatie is, niet straffeloos een andere band mag aanplakken.)

Het idee van de standaardband en "de open bek".

We gaan er vanuit, dat het merendeel der ALGOL-programma's zo is, dat, zo zij om getallenmateriaal vragen,
a) zij niet hoeven te reageren op "einde band"
b) dit materiaal van te voren bijgeleverd kan worden.

Het (operationeel) normale geval zij, dat deze band met getallenmateriaal dan achter de programmatext aangeplakt wordt (c.q. meteen erachteraan geponst is). Wij noemen een dergelijke band met te verwerken materiaal "de eigen band" van dit programma.

Hier geeft het begin van de band -nl. de programmatext- aan hoe de rest -nl. het getallenmateriaal- verwerkt moet worden. Wij willen dit idee een stap uitbreiden en spreken af, dat vooraan de text een herkenbare standaardkreet, zeg "algol" voorkomt. Dit is een metakarakter, dat behelst, dat de eropvolgende ponsingen als de etxt van een ALGOL-programma geinterpreteerd moeten worden. Deze interpretatie (vertaling, gevolgd door executie) is een standaardverwerking; het metakarakter "algol" is op de band geintroduceerd, omdat ik wel ruimte zie voor uitbreidingen van het aantal standaardverwerkingen. Al deze standaardverwerkingen zullen via een bijbehorend metakarakter geactiveerd kunnen worden.

Een band, die zichzelf aan een dergelijke standaardverwerking doet onderwerpen noemen we een standaardband. Het doel, waarnaar wij streven is om het overgrote deel der operateurshandelingen te gieten in het keurslijf "verwerking van een standaardband".

Hiertoe voeren wij in het concept van de "bandlezer met de open bek". De bedoeling is, dat de operateur voor de verwerking van een standaardband niet meer hoeft te doen dan zo'n standaardband inleggen in een bandlezer en op de groene knop drukken. De toestand, waarin een bandlezer aldus op standaardwijze op een standaardband reageert noemen we "met open bek".

Het is de bedoeling, dat alle banden, eenmaal ingelegd, in principe op volle snelheid -dwz. ongeacht het tempo van feitelijke verwerking- ingelezen worden. Als de lezer sneller loopt dan de feitelijke verwerking, dan kan de rest op de trommel gedumpt worden. Zodra "einde band" gesignaleerd is, wordt de lezer vrijgegeven en blijft deze achter met de bek open.

N.B. We eisen niet, dat "einde band" gesignaleerd is, voordat de feitelijke verwerking begint. Als het dringen in het geheugen is, dan moet de coordinator kunnen kiezen en "lezerbezetting" boven "geheugenbezetting" kunnen prefereren.

Hieraan zitten al enige haken en ogen, omdat bandlezers niet uitsluitend voor de standaardverwerking van standaardbanden gebruikt zullen worden. (Er bestaat ook non-standaardverwerking van wat er overigens bedriegelijk als standaardband uitziet!)

De prijs, die we voor de gemakkelijke verwerking van standaardbanden moeten betalen bestaat uit enige extra hocus-pocus voor het non-standaardlezen. Men moet nl. een veilig sluisje maken voor het geval, dat de operateur te goeder trouw een standaardband inlegt in een lezer, juist op hetzelfde moment, dat de X8 beslist, deze lezer voor een ander doel te gaan gebruiken.

In ruwste aanleg kan een eenbandlezer zich in drie toestanden bevinden:
toestand 0: vrij
toestand 1: voor non-standaardwerk aangevraagd
toestand 2: bezet.

In toestand 0 hangt er een tamelijk symbolische startopdracht voor 1 enkele heptade, die alleen maar gegeven is om het inleggen van een nieuwe band en het indrukken van de groene knop tot de X8 te laten doordringen. (Ik neem aan, dat de heptade zelf wellicht genegeerd zal worden.)

In toestand 1 heeft de X8 besloten, dat de bandlezer voor non-standaardwerk gebruikt gaat worden; hiervan is via de verreschrijver melding gedaan. De bandlezer zal niet verder lopen, voordat de operateur op deze melding geantwoord heeft, dat er geen verkeerde band in de bandlezer ligt.

In toestand 2 is de bandlezer definitief ingeschakeld voor een welomlijnde (al of niet standaard) taak.

Vindt nu een ingreep plaats van een bandlezer, die zich in toestand 0 bevindt, dan wordt deze band als standaardband verder gelezen. In doofheid vindt, als reactie op de symbolische heptade, de overgang naar toestand 2 plaats (en er wordt genoteerd, dat we nu standaardbandlezen). De volgende ingrepen vinden per [illegible] definitie plaats in toestand 2.

Vindt een (eerste) ingreep plaats in toestand 1, dan wordt er tot nader order geen band meer gelezen. De operateur heeft dan nl. een standaardband ingelegd in de periode, dat de X8 de lezer al voor een ander doel bestemd heeft, maar de operateur nog niet geantwoord heeft, van deze beslissing kennisgenomen te hebben. De operateur merkt dit o.a. doordat van de band, waarvan hij verwacht, dat hij naarbinnen zal schieten, maar 1 heptade gelezen wordt.

Het operateursantwoord -behelzend, dat er geen ongevraagde band in de lezer ligt, bewerkstelligt de overgang van toestand 1 naar toestand 2, een volledige [illegible] specificatie van wat er dan wel moet gebeuren en tenslotte het vullen van het startmagazijn van de lezer in kwestie. Ligt de band al in, dan gaat hij lopen; de operateur mag het antwoord ook vast geven, als de lezer leeg is, waarna de band gaat lopen, zodra hij is ingelegd.

Het concept van de bandvariabele.

Als een band "synchroon met het physisch leesproces" verwerkt wordt, zoals bij ARMAC en X1, dan fungeert de ligging van de band in de lezer als geheugenelement. Tegen de tijd, dat we het bandbeeld in het geheugen hebben opgeslagen, moeten we dit "heptadewijzertje", dat het sequentieel aftasten van het bandbeeld bestiert, in het geheugen opnemen. Dit heptadewijzertje is een belangrijke constituent van de zg. bandvariabele.

Voorts bevat een bandvariabele het gegeven, of het physisch leesproces al beeindigd is of niet. Zo nee, dan bevat hij een verwijzing naar de physische lezer, de lezeradministratie bevat op zijn beurt een verwijzing naar de bandvariabele. De koppeling "bandvariabele - lezer", die bestaan blijft, totdat de X8 voor deze lezer het signaal E "einde band" gevonden heeft, wordt dus wederzijds geboekt. Zodra het signaal "einde band" tot de X8 doordringt, wordt deze koppeling verbroken. De bandvariabele heeft er dan geen weet meer van, via welke lezer deze band is binnengekomen, de lezer zelf gaat over in toestand 0 met de startopdracht voor de enkele heptade in het startmagazijn. (Dankzij het symbolisch afwerken van eventuele startopdrachten in het magazijn, zodra einde band gesignaleerd is, is het mogelijk om te zorgen, dat dit de enige startopdracht in het magazijn is.)

Een bandvariabele maakt altijd deel uit van een programma (uitgezonderd misschien bij de allereerste analyse van het metakarakter vooraf een standaardband). Tijdens vertaling is de bandvariabele een grootheid van de vertaler, aan het einde van de vertaling fungeert de dan heersende waarde van de bandvariabele als initialisering van de eigen-bandvariabele van het vertaalde programma. Dit overhevelen van bandvariabele van vertaler naar vertaald programma dient met enige zorg te geschieden: omdat de bandlezer het leesproces nog niet voltooid hoeft te hebben, bergt een bandvariabele dus een mogelijke synchronisatiebeperking in zich. Het is deze synchronisatiebeperking, die ook van vertaler naar vertaald programma overgegeven moet worden.

Gestroomlijnd lezen en de rol van metakarakters.

Wanneer wij meer dan 1 programma simultaan in de machine hebben zitten, dan hebben wij zoals bekend er voor te zorgen, dat het ene programma niet zo fout kan zijn, dat daardoor het andere programma verstoord wordt. Deze garantie impliceert beperkingen ten aanzien van de structuur der individuele programma's. Ik heb me steeds op het standpunt gesteld, dat elke gebruiker, die deze beperkingen niet accepteren kan omdat hij in machinecode programmeren wil, de machine dan maar in uniprogrammering bedrijven moet.

Een analoge situatie komen we tegen, wanneer we twee programma's, die verder niets met elkaar te maken hebben, achter elkaar op eenzelfde band willen invoeren. Het verwerkende systeem heeft dan de plicht om vast te stellen, waar de text van het tweede programma begint. De opsteller van het eerste programma mag dan niet de mogelijkheid hebben een zodanige fout te maken, dat het correct vaststellen van de plaats van deze caesuur niet meer een betrouwbare operatie is. Om veilig twee programma's achterelkaar op dezelfde band in te kunnen voeren, moet het eerste programma zekere spelregels in acht nemen, geheel analoog aan de boven gesignaleerde situatie. Uit de aard der zaak zullen wij ernaar streven, deze restricties zo min mogelijk knellend te maken.

Om het leven niet nodeloos te compliceren, veronderstel ik in het volgende dat een programma slechts op twee wijzen contact met de eigen band kan opnemen:
a) lees volgende ponsing
b) lees volgend getal.

In geval a) heeft de programmeur toegang tot ieder bit van de ingevoerde band en hij kan hierop naar eigen goeddunken reageren. Een programma, dat de ponsingen als zodanig van de eigen band leest is als het programma in machinecode; het is niet toegestaan om achter de eigen band van dit programma de text van een onafhankelijk volgend programma te plakken. (Wie rot zou willen doen, schrijft dan een

ALGOL-programma, waarvan het enige effect is, dat het volgende ALGOL-programma op de band wordt overgeslagen!)

Een programma, dat zijn contact met de eigen band echter beperkt tot "lees volgend getal" kan door het systeem onder de duim gehouden worden. Ik beschouw hier"lees volgend getal" als een nieuw primitivum en niet als een subroutine opgebouwd uit "lees volgende ponsing" in die zin, dat gebruik van "lees volgend getal" a fortiori gebruik van "lees volgende ponsing" zou impliceren. Hieraan verbinden wij twee consequenties.

Wij beschouwen "lees volgend getal" als een niet nader gespecificeerd proces, dat bij elke activering een volgende getalwaarde van de band destilleert (of alarm geeft). In het verwerkende programma is bv. niet te achterhalen, hoe deze getallen op de band gegeven zijn. Ze mogen via een flexowriter in decimalen geponst zijn, ze mogen wat mij betreft ook (door een vroeger ALGOL-programma) binair geponst zijn. Over het aantal mogelijkheden wil ik me nu niet uitlaten. In elk geval bevat deze band, wat we vroeger controlecombinaties, wat we nu misschien "syntactische structuur" noemen.

De tweede consequentie is, dat we deze syntactische structuur zo kunnen maken, dat ook de omvang van de informatiehoeveelheid ondubbelzinnig -en wel buiten de invloedssfeer van de programmeur- vastgesteld kan worden. Een programma, dat zich in zijn contact met de eigen band beperkt tot die vormen, waarin deze eindvaststelling aldus geschieden kan, leest zijn eigen band "gestroomlijnd".

Als de eigen band voor gestroomlijnd lezen bestemd is en op de flexowriter gemaakt is, dan bestaat deze syntactische structuur naast de interne structuur van de getallen uit conventies ter getalscheiding, waarschijnlijk door middel van welgedefinieerde separatoren. Ik stel me voor een dergelijke band te doen afsluiten door een soort "superseparator", zeg het metakarakter "enddata". Het gestroomlijnd lezen moet dan een zwaar controlerend proces zijn, dat o.a. onder geen beding het metakarakter "enddata" ongemerkt zou mogen laten voorbijgaan. Als het detectieproces voor het einde van de eigen band aldus tegen fouten van de programmeur is afgeschermd, mag de eigen band door een nieuwe, onafhankelijke band geolgd worden.

Een en ander suggereert, om de vertaler statisch de programma's in drie categorieen in te laten delen

0) programma's, waarin geen eigen band gelezen wordt
I) programma's, waarin de eigen band slechts gestroomlijnd gelezen wordt
II) programma's, waarin de eigen band (slechts) ongestroomlijnd gelezen wordt.

Alleen programma's uit de categorieen I en II staan een achteraangeplakt volgend programma toe (dwz. naar wij hopen bijna allemaal).

Opm.1. Het vaststellen in welke categorie een programma valt, dient statisch te geschieden. Immers, als we het dynamisch afwachten, of en zo ja, hoe een programma contact met de eigen band opneemt, dan zouden we bedrogen uitkomen, zodra een programma uit categorie III ten onrechte zijn activiteit beeindigt, nog voordat contact met de eigen band is opgenomen.

Opm.2. Naar aanleiding van de haakjes om "slechts" in de omschrijving van III. Misschien haal ik deze weg, en sta ik niet toe, dat een band deels gestroomlijnd, deels ongestroomlijnd gelezen wordt. Dit gemengd lezen hinkt nl. erg op twee gedachten. Ik voorzie, dat pogingen om onder alle omstandigheden de semantiek bij overgang te definieren aanleiding geven tot een rommeltje, dat geen gebruiker het vertrouwen geeft geen vergissingen gemaakt te hebben. Liever breidt ik de mogelijkheden van gestroomlijnd lezen wat uit.

Buffering.

Wanneer een band gelezen wordt, waarvan de feitelijke verwerkingssnelheid aanmerkelijk lager ligt dan die van het leesproces, dan zal, als er nog voldoende ruimte op de trommel is, de bandlezer moeten kunnen doorlopen en moet het bandbeeld in het geheugen opgebouwd worden.

Als we -even wat royaal rekenend- zeggen, dat we octaden lezen, waarvan we er 3 in een woord kunnen pakken, dan bergen we dus 1500 octaden in een pagina van 512 woorden. Een volledige rol -300 meter a 400 ponsingen per meter- bevat 120 000 ponsingen en vult dus 80 pagina's, 8 procent van de trommel. Dit is alleszins aceeptabel en we mogen verwachten, dat de lezer dus inderdaad haast altijd full-speed zal doorlopen. Door de snelheid van de schone schuifopdracht kan dit pakken dunkt me ook heel snel; van het idee om banden ongepakt op de trommel op te slaan, schrik ik nog wel een beetje. (Als de prijs van het pakken tegenvalt, kunnen we altijd nog overwegen, om als een band teveel ruimte in dreigt te nemen, pas dan er toe over te gaan om de rest te pakken.)

Hoe een en ander georganiseerd gaat worden is nog niet zo duidelijk. We hebben enige exploratie gepleegd van het volgende arrangement. Op grond van de overweging, dat een voorrakend leesproces impliceren gaat, dat het verwerkende programma het vervolg van het bandbeeld via de trommel gaat aanhalen,dachten we, dat het een eenvoudigere organisatie zouden krijgen, als we afspraken, dat het informatietransport van pakkingsprogramma naar verwerkende programma altijd via de trommel zou lopen. De verwachte simplificatie viel niet mee, integendeel; het zat nog niet eens helemaal rond en toen was het al bar ingewikkeld.

We zullen nu proberen om een arrangement uit te werken, waarbij assemblage- en verwerkingsproces communiceren via een soort cyclische buffer, waarin de elementen "paginadescriptoren" zijn en zullen kijken, of we dit kunnen spelen, onafhankelijk ervan of de pagina's zelf zich nog op de kernen bevinden. De bedoeling is, dat indien bandbeeldpagina's naar en van de trommel getransporteerd worden, ze zich in niets onderscheiden van andere pagina's, die aan de autonome trommeltransport onderworpen zijn. Misschien mede, omdat we dit nog niet hebben uitgewerkt, verwachten we hier veel van.

Ook bij output via ponsband willen we bufferen en we stellen ons voor, dat er normaliter pas een ponser voor een band gereserveerd wordt, als het volledige bandbeeld in het geheugen is opgebouwd. Hiermee hopen we drie dingen te bereiken.

Ten eerste zal hierdoor een programma een bandponser zo kort mogelijk aansluitend bezetten. Ten tweede is nu de lengte van de band bekend, voordat het ponsproces begint. Als de coordinator ingelicht wordt, wanneer er een nieuwe rol in een van de ponsers is ingelegd (en wanneer de operateur wat beschaiden met run out omgaat) dan kan de coordinator bijhouden, hoeveel papier er nog op de rol behoortte zitten. Er kan dus voor gezorgd worden, dat niet halverwege de band het papier plotseling op is. Ten derde kan men dit bandbeeld twee maal gebruiken, nl. eerst om te ponsen, en later om te controleren. Dit controleproces willen wij beschouwen als een van de standaardhandelingen, die geactiveerd kan worden, door een band met een passend metakarakter in een open bek te leggen. Dit extra metakarakter zal het ponsproces er ongevraagd vooraan toevoegen, als het zijn werk gedaan heeft, kan de operateur het afscheuren. Het succesvol beeindigen van het controleproces kan de geheugenruimte van het bandbeeld vrijgeven.