ALWEER EEN TAAL OP DE KIM

Ongeveer 12 jaar geleden maakte ik kennis met een hogere computertaal, uitgebracht door DEC, op een PDP-8. Deze taal heette FØCAL en was bedoeld als een conversationeel hulpmiddel voor mensen, die met weinig inspanning op het gebied van programmeren, programmaatjes moesten schrijven voor wetenschappelijke berekeningen, analyse van data, statistiek en administratie.

Wie schetst mijn verbazing, toen enige weken geleden Anton MÜller bij mij kwam en zei: "Ik heb bij Aresco in Amerika voor $ 75 een FØCAL-compiler gekocht, Zullen we die eens proberen?

Uiteraard werd het programma in de KIM gedraaid en groot was de verrassing toen dit inderdaad een vrijwel volledige copie was van de FØCAL-versie op de PDP-8!!!!!!

Bij gebruik is het eerste, dat opvalt aan dit systeem de grote "gebruikersvriendelijkheid". Dit is vooral dankzij het commando MØDIFY, dat de gebruiker in staat stelt zonder veel moeite binnen een programmaregel verbeteringen en wijzigingen aan te brengen.

Iets dat niet zo verbazingwekkend meer is tegenwoordig, is dat FØCAL alle normale rekenkundige functies zoals: - optellen, aftrekken, vermenigvuldigen, delen, machtsverheffen, absolute waarde, geheel getal, random number en afronding aankan -, evenals formules met haakjes. (getallen 9 cijfers nauwkeurig) .

Eveneens vriendelijk is de mogelijkheid om een programma te tracen, d.w.z. ieder statement dat uitgevoerd wordt, wordt ook uitgeprint.

Verdere faciliteiten die FØCAL biedt, zijn:

- Stringhandling. Maximale grootte van een string is 250 characters.
- Arrays. Variabelen die uit een rijtje getallen bestaan in plaats van uit één getal.
- Volledige gebruikersbesturing van papierindeling bij output.
- Bij het uitvoeren van foute statements wordt het statement dat niet thuis te brengen is, uitgeprint met een pijltje onder de plaats, waar de fout mogelijkerwijs zit.
- Mogelijkheid om de echo, die normaal gesproken van het toetsenbord altijd naar de printer doorkomt op de KIM, te onderdrukken. Dit is werkelijk een zeer gave softwaretruc.
- Uitstekende documentatie. De documentatie bestaat uit een beschrijving van de taal en zijn faciliteiten plus een listing.

Vooral opmerkelijk is de flexibele opbouw van het geheel. Op diverse plaatsen is bijvoorbeeld ruimte opengelaten voor patches, die een gebruiker erin zou willen maken. De schrijver noemt dit: Space for hackers. Dit betekent, dat een gebruiker nieuwe statements kan bedenken en nieuwe functies.

De input-output dient wel een aparte waardering!!!!!

PROGRAMMEREN MET FOCAL

FOCAL is een programmeertaal die in veel opzichten hetzelfde is als BASIC, maar in een aantal gevallen is FOCAL krachtiger. Dit wordt geen volledige handleiding, statements als LET, PRINT INPUT, IF THEN, FOR NEXT enz in BASIC zijn in FOCAL ook terug te vinden. Daarom behandel ik alleen een paar interessante(re) statements.
Werken we in BASIC met regelnummers, in FOCAL werken we met groepen (max. 99). Elk verschillend onderdeel van een programma komt in een eigen groep. Binnen zo'n groep kan men 99 regels gebruiken. Zo betekent 10.30 groep 10 30ste regel.
FOCAL bezit 'gewone' statements en 'speciale' statements.
De gewone statements (IF, GOTO, FOR, RETURN, DO, SET enz) mogen tot hun eerste letter worden afgekort en werken (ongeveer) op dezelfde manier als in BASIC.
De speciale statements zijn FUNCTIES. Elke naam v/e FUNCTIE begint met een 'F' (bv. FCAL, FIDV enz). Het zijn deze FUNCTIES die interessant zijn.
FOCAL werkt (tenminste mijn uitvoering) met 8 getallen nauwkeurigheid, maar kent geen wetenschappelijke notatie.

```
33333.34          Bij het uitprinten van getallen kan men zelf
    1.00          bepalen hoeveel getallen voor en achter de
 -102.51          decimale punt moeten worden afgedrukt.Door
    0.01          deze mogelijkheid komt een reeks van getallen
22210.00          altijd mooi met de decimale punt onder elkaar.
```

In FOCAL is het mogelijk om op meerdere manieren een subroutine aan te roepen. Enkele voorbeelden:

a. dmv DO 3.1 wordt een regel (hier 3.1) als subr. uitgevoerd
b. dmv DO 3 wordt een groep (hier 3) als subr. uitgevoerd
c. dmv DO A$ wordt een string als subr. uitgevoerd
d. dmv FCAL wordt een machinetaal subr. uitgevoerd
e. dmv FSBR wordt een regel of groep als subr. uitgevoerd

FSBR heeft als eigenschap dat men een variabele mag meegeven naar een subr. met die waarde wordt alles berekend en de uitkomst komt weer in die variabele terug. Zeer handig als men met verschillende variabelen dezelfde berekeningen wilt uitvoeren.

HET GEBRUIK VAN STRINGS:

Normaal wordt een toetsenbord gebruikt als inputorgaan (input device) en het beeldscherm als outputorgaan (output device). Dit is in FOCAL softwarematig te veranderen zodat men bv. naar een printer, cassetterecorder, floppydisc enz. kan overschakelen. Het krachtige is echter dat men deze hardware zaken ook kan vervangen door software (nl. dmv. STRINGS).
Het is mogelijk om het toetsenbord te vervangen door een string alles wat in die string staat accepteert FOCAL als of het door de programmeur is ingetypt. Zo kan men kommando's, statements en zelfs gehele regels,die in een STRING staan,laten uitvoeren.
Ook is het mogelijk het outputorgaan (beeldscherm) te vervangen door een string (oid) nu worden alle terugmeldingen in die string (oid) opgeborgen.
Het vullen v/e string kan dmv:

a. een STring Input (FSTI(x,A$(y),z)) waarbij x het max. aantal karakters aangeeft, y de plaats waarin A$ wordt begonnen en z met welke karakter geëindigd kan worden (indien nog geen x karakters zijn ingevoerd).
b. A$(10)='Q nu wordt de 11ste plaats van A$ met een 'Q' gevult.
c. A$ als output device te verklaren en de data die in de string moet komen gewoon te laten uitprinten.

Ik zal een voorbeeld geven van c:
Dmv SET FODV(A$) wordt het output device (was beeldscherm) nu A$ TYPE (of T) doet hetzelfde als PRINT in BASIC en hiermee kun je nu A$ vullen (A$ vervangt immers het beeldscherm, waar het anders op zou komen). RESTORE OUTPUT (of R O) herstelt het outputorgaan. SET FIDV(A$) zal het inputorgaan (is meestal het toetsenbord) op A$ zetten. Dus alles wat in A$ staat lijkt voor FOCAL als of het gewoon is ingetypt. Een '!' bij een TYPE statement print een carriage return (='cr').
QUIT is hetzelfde als END in BASIC, er wordt gestopt met het programma.
Nu volgt er een programma dat eerst A$ met een regel FOCAL vult en daarna die regel bij het programma toevoegt.

```
1.1 SET FODV(A$)              maak A$ output voor het vullen van A$
1.2 TYPE "2.1 TYPE X*3",!,"R I",!
1.3 RESTORE OUTPUT            herstel het outputorgaan (evt. R O)
1.4 SET FIDV(A$)              vervang het toetsenbord door A$
1.5 QUIT                      stop en kijk naar het input device
```

wat er gedaan moet worden.
Dmv 1.1, 1.2 en 1.3 wordt A$ geladen met:

```
2.1 TYPE X*3
R I
```

Regel 1.4 zet het input device op A$ en 1.5 stopt het programma. Nu wordt er naar het input device (is nu dus A$) gekeken, daarin staat een regel FOCAL (+ een 'cr') deze wordt er dus bij geschreven. Nu wordt er weer verder gekeken in A$ en daarin staat R I (=RESTORE INPUT of wel herstel input) en het toetsenbord wordt inputorgaan. Er is dus nu regel 2.1 bij het programma gevoegd. Er bestaat dus een mogelijkheid om een FOCAL programma te maken dat zelf weer een FOCAL programma schrijft. Of een FOCAL programma dat een regel BASIC in een string opslaat en die string vervolgens gaat vertalen naar FOCAL en daarna die string (wat nu geheel FOCAL is) wegschrijft in het geheugen. Een soort BASIC naar FOCAL vertaler, geschreven in FOCAL, krijgt men dan.

HET WERKEN MET INTERRUPTS:

Het is, in FOCAL, mogelijk om mbv. een interrupt een FOCAL programma te beïnvloeden. Dmv. het 'setten' van bitjes en een statement in FOCAL kan men dan nog enkele interrupts prioriteit geven. Er wordt dan door FOCAL zelf gekeken of er een interrupt plaats vindt en wat er daarna gedaan moet worden waarbij de interrupt met het hoogste prioriteit als eerste wordt uitgevoerd. (zelf heb ik hiermee nog niet geëxperimenteerd).

J Janssen
Gerardsweg 30
65 25 RT NIJMEGEN
tel.:080-562082