CLS = SCHERM SCHOON.

INTEGER. %

COMAL*.KGN beknoote handleidino
COMAL*.KIM GEBRUIKERS CLUB NED.

1. Gebruikte afkortinoen

In deze handleidino wordt oebruik oemaakt van de navoloende afkortinoen om data aan te duiden:

var ..... variabele
exp ..... exoressie
con ..... constante
fun ..... functie

Deze kunnen worden voorafoeoaan door een letter die het datatype omschrijft:

i ..... integer
r ..... real
s ..... strino
a ..... integer of real

Overioe afkortinoen:

num ..... reoelnummer
sto ..... stap tussen twee ooeenvoloende reoelnummers
sta ..... statement
blk ..... blok van meerdere statements
cr ..... carriage return met linefeed

Een indexcijfer oeeft verschillende orootheden aan. dus svar1 is een andere svar dan svar2. Let oo de combinatie van s(trino) en var(iabele) tot s(trino)var(iabele). Soms wordt een reeks variabelen aangegeven met var1....varn waarbij het achtervoegsel n de betekenis heeft van 'laatste'.

2. Syntax reoels

Elk commando moet oo een aparte reoel worden ingetvot. achter een reoelnummer. De volgorde van intvoen is niet van belano. COMAL zet zelf alle reoels oo volgorde van nummer. Een commando zonder reoelnummer wordt. alleen indien mogelijk. direkt uitgevoerd. In deze 'direct mode' heeft COMAL minder mogelijkheden dan Basic. omdat ook hier slechts een enkel commando per reoel kan worden uitgevoerd. In tegenstelling tot Basic dwingt COMAL tot gestructureerd oroorammeren. Reoelnummers hebben tijdens de oroorama executie geen enkele betekenis meer. Ze dienen slechts bij het invoeren of wijzigen van het oroorama. Reoels kunnen worden gewijzigd in de listino, als uw systeem over editfaciliteiten beschikt. Door het tvoen van alleen een reoelnummer kunt u een reoel verwijderen. Door oonieuw invoeren kunt u een reoel altijd wijzigen. Een reoelnummer moet kleiner zijn dan 64000 of er volgt "SYNTAX ERROR".

3. Commando's

DELETE num1,num2

Wist de reoels van en met num1 tot en met num2 uit het oroorama. num1 en num2 moeten altijd worden oegoecificeerd. Gebruik van DELETE in een oroorama mag, doch is onjuist en stoot de oroorammaexecutie welke niet met CONT kan worden vervolod.

RENUMBER num,sto

Hernummert alle reoels. De eerste reoel wordt reoel num. Elk voloend nummer is sto hoger. Indien sto niet wordt oogegeven, wordt 10 genomen. Indien num ook niet is oogegeven wordt 100 oenomen. Gebruik van RENUMBER in een oroorama mag, doch is onjuist en stoot de oroorammaexecutie welke niet met CONT kan worden vervolod.

PRINT:exp1:....:expn:	Drukt de waarden af van exp1 tot en met expn. Indien er geen misverstand kan bestaan over het begin van een expressie en het einde van de voorgaande, mag men de punt-komma weglaten. Indien men de laatste punt-komma weg laat wordt het afdrukken afgesloten met een crlf. Indien men de punt-komma vervangt door een komma, dan worden spaties afgedrukt tot voor de volgende kolom die een veelvoud is van 16.
CLEAR	Wist alle variabelen en executeert RESTORE. Alle nesting-niveaus gaan verloren!
NEW	Vernietigt het programma en wist alle variabelen.

4. Operatoren

aexp1 + aexp2	Levert de som van beide expressies.
sexp1 + sexp2	Voegt de strings sexp1 en sexp2 samen.
aexp1 - aexp2	Levert het verschil aexp1 minus aexp2 op.
aexp1 * aexp2	Levert het product van beide expressies.
aexp1 / aexp2	Levert het quotient aexp1 gedeeld door aexp2.
aexp1 ^ aexp2	Levert aexp1 tot de macht aexp2 op.
aexp1 OR aexp2	Levert 1 op indien een van beide expressies ongelijk nul is. Anders is het resultaat 0.
aexp1 AND aexp2	Levert 1 op indien beide expressies ongelijk 0 zijn. Anders is het resultaat 0.
aexp1 = aexp2	Levert 1 op indien beide expressies een gelijke waarde hebben. Anders is het resultaat 0.
aexp1 < aexp2	Levert 1 op indien de waarde van aexp1 kleiner is dan de waarde van aexp2. Anders 0.
aexp1 > aexp2	Levert 1 op indien de waarde van aexp1 groter is dan de waarde van aexp2. Anders 0.
aexp1 <> aexp2	Levert 1 op indien beide expressies een ongelijke waarde hebben. Anders 0.

5. Functies

NOT(aexp)	Levert 1 op indien aexp gelijk nul is. Anders nul.
SGN(aexp)	Levert 1 of -1 op conform het algebraische teken van aexp.
INT(aexp)	Levert de integer waarde van aexp in real.
ABS(aexp)	Levert de absolute waarde van aexp.
USR(aexp)	Zet de waarde van aexp in de floating-point accu en voert CALL 10 uit. Hier dient een JMP instructie te staan naar een zelf te schrijven machinetaal functie. Indien de JMP niet wordt gezet, wijst ze naar ILLEGAL QUANTITY ERROR. Het resultaat van de USR routine moet weer in de flp accu worden gezet, waarna met RTS wordt teruggekeerd naar COMAL.
FRE(exp)	Doet een 'carbage collect' en levert het aantal vrije geheugenlocaties op.
POS(exp)	Levert het nummer van de laatst bedrukte kolom op.
SQR(aexp)	Levert de vierkantswortel van aexp.
RND(aexp)	Indien aexp <0 wordt voor elke waarde een bijbehorend getal tussen 0 en 1 gegenereerd. Deze oneigenlijke RND functie kan dienen tijdens debuggen van programma's die gebruik maken van RND. Indien aexp >0 wordt een willekeurig getal tussen 0 en 1 gegenereerd. Deze RND functie voldoet in de meeste gevallen, doch is niet 100% echte 'random' RND(0) Levert het laatst gegenereerde RND-getal opnieuw.
LOG(aexp)	Levert de natuurlijke logarithme van aexp. aexp<>0 !
EXP(aexp)	Levert e tot de macht aexp. (e=2.71828183....)
COS(aexp)	Levert de cosinus van de hoek aexp (aexp in rad.)
SIN(aexp)	Levert de sinus van de hoek aexp (aexp in rad.)
TAN(aexp)	Levert de tangens van de hoek aexp (aexp in rad.) aexp<>pi/2 (pi=3.14159265)
ATN(aexp)	Levert de boog in rad. waarvan de tangens aexp is. aexp=<1
PEEK(aexp)	Levert de inhoud van geheugenlocatie aexp. 0=<aexp=<65535.
LEN(sexp)	Geeft de lengte van de string sexp.
STR$(aexp)	Converteert de waarde van aexp tot string.
VAL(sexp)	Levert een getal gelijk aan de waarde van sexp, gelezen vanaf het begin van de string tot voor het eerste niet bruikbare karakter.
ASC(sexp)	Levert de ASCII code van het eerste karakter van sexp.
CHR$(aexp)	Levert een string van een karakter dat de ascii code aexp heeft. 0=<aexp=<255.
LEFT$(sexp,aexp)	Levert een string bestaande uit de linker aexp karakters van sexp.
RIGHT$(sexp,aexp)	Levert een string bestaande uit de rechter aexp karakters van sexp.

LABEL: sexo

Een LABEL statement wordt bij sequentiele executie genegeerd. Naar een label kan worden gesprongen met:

GOTO sexo

Sprong naar LABEL:sexo. GOTO dient in principe slechts te worden gebruikt in combinatie met:

ONERR GOTO sexo

Indien na executie van ONERR een foutmelding zou moeten worden gegeven, wordt in plaats daarvan naar LABEL:sexo gesprongen. Hier dient een foutafhandelingsroutine te staan die de fout op de door de programmeur gewenste wijze afdoet. ONERR kan weer worden uitgeschakeld door POKE 216,0. In geheugenlocatie 222 is een codenummer te vinden dat aangeeft welke fout is opgetreden:

0	NEXT WITHOUT FOR
16	SYNTAX
22	ENDPROC WITHOUT EXEC
42	OUT OF DATA
53	ILLEGAL QUANTITY
69	OVERFLOW
77	OUT OF MEMORY
90	UNDEF' PROCEDURE
107	BAD SUBSCRIPT
120	REDIM'D ARRAY
133	DIVISION BY ZERO
163	TYPE MISMATCH
176	STRING TOO LONG
191	BAD FLOW OF CONTROL
224	UNDEF'D FUNCTION
225	BREAK INTERRUPT

In locaties 218 en 219 (lo,hi) staat het regelnummer van de fout. Met GOTO uit een foutafhandelingsroutine springen kan rare gevolgen hebben. Een foutafhandeling dient te eindigen met:

RESUME

Mag alleen gebruikt worden aan het einde van een foutafhandeling via ONERR. Het programma zal het statement waarin de fout optrad opnieuw uitvoeren. Een prima toepassing van ONERR is detectie van invoerfouten.

RUN sexo

Begint executie van het programma op de regel met het label sexo. Indien sexo niet wordt gegeven, wordt begonnen met de eerste programmaregel. RUN veroorzaakt een voorcompilatie die het opzoeken van labels en procedures versnelt, doch de listing vermindt. Correcte beeindiging van het programma, dus zonder foutmelding of BREAK herstelt de listing. Zie ook 'list'. RUN wist alle variabelen en executeert RESTORE.

//

Een regel die begint met twee schuine streepen wordt door de interpreter bij executie overgeslagen. Men kan aldus verklarende tekst tussenvoegen. (REM in Basic)

STOP

Veroorzaakt een break zonder de listing te herstellen. Het programma kan met CONT worden vervolgd. STOP dient uitsluitend voor het debuggen van programma's. Indien men na een BREAK een deel van het programma wil bekijken dient men eerst in direct mode END te typen. Als men daarna verder wil gaan met CONT moet men er rekening mee houden dat lange programma's aanzienlijk trager zijn geworden!

WAIT aexp

Vertraging. aexp=125 komt ongeveer overeen met 1 sec. Er is dus geen enkel verband met WAIT zoals Basic die kent!

LOAD,aexp

Leest programma nummer aexp van tape. 0<aexp<255. Indien men geen parameter specificeert of indien men deze opgeeft als 0 of 255 volgt de foutmelding ERROR zonder nadere aanduiding. Deze melding ontstaat ook bij een check sum error.

SAVE,aexp

Schrijft het programma naar tape onder filenummer aexp. 0<aexp<255. Indien men geen parameter specificeert of indien men deze opgeeft als 0 of 255 volgt de foutmelding ERROR zonder nadere aanduiding.

DEF FN rvar1(rvar2)=fun(rvar2)

Definieert een funktie met de naam FN rvar1, toegepast op een dummy variabele rvar2.

POKE aexp1,aexp2

Zet de waarde aexp2 in geheugenlocatie aexp1. Beide parameters worden automatisch tot integer geconverteerd. 0=<aexp1=<65535 en 0=<aexp2=<255.

LIST num1-num2	Geeft een afdruk van het programma of een deel daarvan. LIST -num geeft een afdruk van het programma tot en met regel num. LIST num- geeft een afdruk van programmaregel num tot het einde van het programma. LIST num drukt alleen regel num af. LIST num1-num2 drukt de regels num1 t/m num2 af. In de listing zijn de controlstructuren en 'nesting levels' te zien. doch vreemde effecten ontstaan indien de listing midden in een blok wordt begonnen. Indien een programma werd afgebroken door STOP. een break of een foutmelding, dient men geen wijziging aan te brengen of LIST te gebruiken. voordat men 'END.' heeft getypt. Anders ontstaat een merkwaardige listing terwijl een poging tot veranderen. een regelnummer op een vreemde plaats kan invoegen! Het is mogelijk LIST in een programma te gebruiken, alhoewel zulks onjuist is. Soms is het echter nodig om bepaalde trucks toe te passen tijdens programma ontwerp. LIST in een programma dat met RUN wordt geexecuteerd geeft een verminkte listing. Zet in een dergelijk geval als eerste regel een LABEL: statement en start het programma met GOTO sexp waarbij traagere executie op de koop toe moet worden genomen.
CONT	Vervolgt de programma executie na een STOP-statement of na een BREAK welke op de JUNIOR ontstaat door het drukken van een willekeurige toets. CONT werkt niet na een foutmelding of na wijziging van het programma. In dat geval ontstaat de foutmelding CAN'T CONTINUE ERROR. Indien CONT in een programma wordt gebruikt 'hangt' de interpreter, doch zonder schade. Onderbreek in dat geval de executie met een toetsdruk.
END.	Stopt de programma executie en ordent de listing. De punt achter END hoort er bij! Het is niet verplicht END. te gebruiken als laatste programma regel.
DATA con1,.....,conn	Staat toe data in het programma op te nemen. Het aantal constanten is slechts beperkt door de regellengte. String-constanten waarin de leestekens komma of punt-komma voorkomen. of waarin een sleutelwoord voorkomt, moeten tussen aanhalingstekens worden gezet.
INPUT svar	Wacht op invoer van een string-constante vanaf het input-device, meestal het toetsenbord en asigneert de ingevoerde string aan svar. INPUT accepteert elk karakter tot aan de eerstvolgende cr. INPUT mag niet worden gebruikt als direkt commando. Men krijgt dan de foutmelding ILLEGAL DIRECT ERROR. INPUT avar is niet toegestaan. Hiervoor dient INPUT svar gevolgd door avar:=val(sexp) te worden gebruikt. Indien het eerste karakter dat achter input wordt ingevoerd, ctrl-c (ASCII-3) is, wordt de programma executie na de cr met break afgebroken. De executie kan worden vervolgd met CONT en begint dan opnieuw met het input statement.
GET svar	Is identiek aan INPUT svar doch accepteert slechts een karakter zonder op cr te wachten. Ook hier geeft ctrl-c een break.
DIM var(aexp1,.....,aexpn)	Reserveert ruimte voor een matrix met de naam var en met aexp1+1 elementen in de eerste dimensie, tot aexpn+1 elementen in de laatste dimensie. Het aantal elementen mag maximaal 32767 per dimensie zijn. Het aantal dimensies is slechts begrensd door de lengte van de programmaregel. Indien de matrix var reeds bestond volgt de foutmelding REDIM'D ARRAY. Men kan naar een matrix element refereren met var(aexp1,.....,aexpn). Indien deze omschrijving niet overeenkomt met de door de DIM gedeclareerde omschrijving volgt de foutmelding BAD SUBSCRIPT ERROR.
READ var	Leest de volgende constante uit de data-statements en asigneert deze aan var. Indien geen ongelezen data meer beschikbaar is volgt de foutmelding OUT OF DATA ERROR.
RESTORE	Staat toe alle data opnieuw te lezen. Een gedeeltelijke RESTORE is onmogelijk.
CALL aexp	Executeert een machinetaalroutine op adres aexp. aexp wordt automatisch tot integer geconverteerd. 0=< aexp =<65535. De machinetaalroutine dient te eindigen met een RTS om correct naar COMAL terug te keren.
var :=exp	Asigneert de waarde van exp aan var. var en exp moeten tot hetzelfde datatype behoren, anders volgt de foutmelding TYPE MISMATCH ERROR. Bij uitzondering mogen meerdere asignments achter een regelnummer staan, gescheiden door punt-komma. Het sleutelwoord LET kent COMAL niet. Spreek het teken := uit als "wordt".

MID$(sexo,aexo1,aexo2) Levert een string bestaande uit aexo2 karakters van sexo. vanaf het aexo2-de karakter.

TAB(aexo) Mag alleen worden gebruikt in PRINT-statements. Print spaties tot voor kolom aexo. 0=<aexo=<255. (tabuleert) In COMAL werkt TAB ook correct naar de printer!

SPC(aexo) Print aexo spaties. 0=<aexo=<255. Mag alleen worden gebruikt in PRINT-statements.

RES(sexo),var1,.....,avarn Executeert procedure sexo en heeft als resultaat de waarde van avarn (de laatste variabele in de lijst mag geen string zijn). Indien de variabelenlijst ontbreekt, is het resultaat gelijk aan dat van de laatst geevalueerde expressie. Indien dat een string was volgt de foumelding TYPE MISMATCH ERROR.

6. Control Structuren.

IF-THEN-ELSE-ENDIF

```
IF aexo THEN
 blk1
ELSE
 blk2
ENDIF
```

Indien aexo ongelijk nul is, wordt blk1 uitgevoerd, waarna wordt verder gegaan op de regel na ENDIF. Indien aexo=0 wordt blk2 indien aanwezig, uitgevoerd waaarna wordt verder gegaan op de regel na ENDIF. Indien blk2 ontbreekt mag ELSE worden weggelaten. ELSE of ENDIF zonder voorafgaande IF geeft de foutmelding 'BAD FLOW OF CONTROL. IF zonder ELSE en zonder ENDIF geeft dezelfde foutmelding indien blk1 moet worden overgeslagen.

WHILE-DO:-ENDWHILE

```
WHILE aexo DO:
 blk
ENDWHILE
```

Indien aexo ongelijk 0 is wordt verder gegaan op de regel na ENDWHILE. Anders wordt blk uitgevoerd waarna wordt teruggegaan naar WHILE, totdat een resultaat ongelijk 0 is verkregen. WHILE zonder ENDWHILE of omgekeerd geeft de foutmelding 'BAD FLOW OF CONTROL'

REPEAT-UNTIL

```
REPEAT
 blk
UNTIL aexo =
```

Voert eerst blk uit daarna wordt aexo geevalueerd en blk zonodig herhaald tot aexo=0. UNTIL zonder REPEAT geeft de foutmelding 'BAD FLOW OF CONTROL'.

FOR-TO-STEP-ENDFOR

```
FOR rvar:=aexo1 TO aexo2 STEP aexo3
 blk
ENDFOR
```

Geeft rvar de waarde van aexo1 en voert blk uit. Daarna wordt rvar met aexo3 vermeerderd. Indien dit resultaat =<aexo2 is wordt blk opnieuw uitgevoerd tot rvar een waarde >aexo2 heeft bereikt. Voor rvar mag geen array element worden gebruikt. STEP aexo3 mag worden weggelaten. In dat geval wordt voor aexo3 de waarde 1 genomen. Indien een in blk een FOR-loop opent met dezelfde rvar volgt SYNTAX ERROR. ENDFOR zonder voorafgaande FOR geeft de foutmelding NEXT WITHOUT FOR ERROR.

CASE-WHEN-OTHERWISE-ENDCASE

```
CASE var
 WHEN exo1
  blk1
 WHEN exon
  blkn
 OTHERWISE
  blk3
ENDCASE
```

Voor var mag geen integer of matrix element worden gebruikt. Het aantal WHEN-blokken is onbeperkt. Een voor een worden de expressies exp1.... expn geevalueerd tot een expressie wordt gevonden die een resultaat gelijk aan var oplevert. In dat geval wordt het blok onder de WHEN uitgevoerd, waarna verder wordt gegaan op de regel na ENDCASE. Indien geen expressie wordt gevonden die voldoet, wordt het blok onder OTHERWISE uitgevoerd. Indien OTHERWISE ontbreekt en geen enkele expressie voldoet volgt een foutmelding. Indien een van de statements WHEN, OTHERWISE of ENDCASE wordt aangetroffen zonder voorafgaande CASE, volgt BAD FLOW OF CONTROL ERROR.

7. Procedures

PROC-ENDPROC

Een procedure is een subroutine met een naam van de vorm:

```
PROC sexp,var1,.....,varn
 blk
ENDPROC
```

sexp bevat de procedurenaam en is meestal een constante. Een procedure wordt aangeroepen met:

EXEC: sexp,var1,.....,varn

De lijsten van variabelen achter EXEC: en PROC dienen precies overeen te komen naar aantal en datatype. Men mag de lijst ook geheel weg laten mits men dat zowel achter PROC als EXEC: doet. EXEC: zoekt de naam van de procedure en indien deze niet wordt gevonden volgt de foutmelding UNDEF'PROCEDURE. Als de naam is gevonden, worden de waarden van alle variabelen in de lijst achter EXEC: toegekend aan de variabelen in de lijst achter PROC. Daarna wordt de procedure uitgevoerd. Na uitvoering worden de nieuwe waarden van de variabelen in de lijst achter PROC weer toegekend aan de variabelen in de lijst achter de EXEC: die de procedure deed uitvoeren. Vervolgens gaat het programma verder op de regel na de EXEC: die de procedure deed uitvoeren. Indien PROC bij sequentiele executie wordt aangetroffen volgt SYNTAX ERROR. Indien ENDPROC wordt aangetroffen zonder aanroep door EXEC: volgt de foutmelding ENDPROC WITHOUT EXEC.

8. COMAL*.KBN in Uw JUNIOR.

COMAL*.KBN wordt op tape geleverd voor een standaard Junior. De huidige versie houdt geen rekening met welk disk operating system dan ook, zodat ongetwijfeld conflicten ontstaan op page zero !

Een ander probleem vormen die systemen die hun monitor en/of I/O op een andere locatie hebben gezet. Het aantal routines in COMAL dat van de JUNIOR systeemsoftware gebruik maakt is echter zo gering dat aanpassingen gemakkelijk kunnen worden gemaakt. Hier volgen de I/O locaties:

```
INPCH op $2553 JSR $12AE
OUTCH op $47FA JSR $1334
CRLF  op $47F5 JSR $11E8
```

Verder komen een aantal referenties aan de monitor en i/o voor in de routines:

LOAD en SAVE van $28B0 t/m 28FC
BREAK en initialisering van $4248 t/m $4264

Bij het opstarten zoekt COMAL naar de hoogste RAM locatie. Een geval is reeds bekend van een JUNIOR-gebruiker die zijn systeem had voorzien van een videoschakeling met beeldschermgeheugen, waarin COMAL vervolgens strings ging opslaan. U kunt in zulke gevallen het geheugengebruik van

COMAL beteugelen door:

$4195 LDY #$00
$4197 LDA #h.adr (bv $80 voor $8000)

9. Oostarten, warme start en einde

Na laden van tape wordt COMAL gestart op adres $3000. Alles is ok indien de promoting verschijnt, bestaande uit een sluit-haakje). COMAL zet de break-vector van de JUNIOR naar een eigen breakroutine. Deze maakt het mogelijk een programma met een toetsdruk te onderbreken en het programma eventueel met CONT te vervolgen. Een en ander betekent dat U alleen uit de COMAL-interpreter kunt komen met een RESET. Als U vanuit de monitor geen van de navolgende locaties overschrijft, kunt U terug naar COMAL op adres $0. Uw programma gaat dan niet verloren:

$0000-$000D
$0067-$0080
$00AF-$00CC
$2000-$47FF

Om uw programma te behouden dient U ook het gebied $4800 tot einde programma ongemoeid te laten. Dit adres is te vinden op de locaties $AF en $B0 (lo.hi). Type altijd CLEAR na een warme start. Anders bestaat de mogelijkheid dat U COMAL en programma kwijt raakt !