

COMAL\*.KGN beknopte handleiding  
 =====  
 COMAL\*.KIM GEBRUIKERS CLUB NED.

### 1. Gebruikte afkortingen

In deze handleiding wordt gebruik gemaakt van de navolgende afkortingen om data aan te duiden:

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

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

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

Overige afkortingen:

num ..... regelnummer  
 sto ..... stap tussen twee opeenvolgende regelnummers  
 sta ..... statement  
 blk ..... blok van meerdere statements  
 cr ..... carriage return met linefeed

Een indexcijfer heeft verschillende grootheden aan, dus svar1 is een andere svar dan svar2. Let op de combinatie van s(string) en var(variabele) tot s(string)var(variabele). Soms wordt een reeks variabelen aangegeven met var1.....varn waarbij het achtervoegsel n de betekenis heeft van 'laatste'.

### 2. Syntax regels

Elk commando moet op een aparte regel worden ingevoerd, achter een regelnummer. De volgorde van invoeren is niet van belang. COMAL zet zelf alle regels op volgorde van nummer. Een commando zonder regelnummer wordt, alleen indien mogelijk, direct uitgevoerd. In deze 'direct mode' heeft COMAL minder mogelijkheden dan Basic, omdat ook hier slechts een enkel commando per regel kan worden uitgevoerd. In tegenstelling tot Basic dwingt COMAL tot gestructureerd programmeren. Regelnummers hebben tijdens de programma executie geen enkele betekenis meer. Ze dienen slechts bij het invoeren of wijzigen van het programma. Regels kunnen worden gewijzigd in de listing, als uw systeem over editfaciliteiten beschikt. Door het invoeren van alleen een regelnummer kunt u een regel verwijderen. Door opnieuw invoeren kunt u een regel altijd wijzigen. Een regelnummer moet kleiner zijn dan 64000 of er volgt "SYNTAX ERROR".

### 3. Commando's

DELETE num1,num2

Wist de regels van en met num1 tot en met num2 uit het programma. num1 en num2 moeten altijd worden gespecificeerd. Gebruik van DELETE in een programma aag, doch is onjuist en stoot de programmaexecutie welke niet met CONT kan worden vervolgd.

RENUMBER num.sto

Hernummert alle regels. De eerste regel wordt regel num. Elk volgend nummer is sto hoger. Indien sto niet wordt opgegeven, wordt 10 genomen. Indien num ook niet is opgegeven wordt 100 genomen. Gebruik van RENUMBER in een programma aag, doch is onjuist en stoot de programmaexecutie welke niet met CONT kan worden vervolgd.

PRINT:exol:....:exon:

CLEAR

NEW

Drukt de waarden af van exol tot en met exon. 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 afdrucken 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.

Wist alle variabelen en executeert RESTORE. Alle nesting-niveaus gaan verloren!

Vernietigt het programma en wist alle variabelen.

#### 4. Operatoren

aexol + aexo2  
sexol + sexo2  
aexol - aexo2  
aexol \* aexo2  
aexol / aexo2  
aexol ^ aexo2  
aexol OR aexo2

Levert de som van beide expressies.

Voegt de strings sexol en sexo2 samen.

Levert het verschil aexol minus aexo2 op.

Levert het product van beide expressies.

Levert het quotient aexol gedeeld door aexo2.

Levert aexol tot de macht aexo2 op.

Levert 1 op indien een van beide expressies ongelijk nul is. Anders is het resultaat 0.

aexol AND aexo2

Levert 1 op indien beide expressies ongelijk 0 zijn. Anders is het resultaat 0.

aexol = aexo2

Levert 1 op indien beide expressies een gelijke waarde hebben. Anders is het resultaat 0.

aexol ( aexo2

Levert 1 op indien de waarde van aexol kleiner is dan de waarde van aexo2. Anders 0.

aexol ) aexo2

Levert 1 op indien de waarde van aexol groter is dan de waarde van aexo2. Anders 0.

aexol ( ) aexo2

Levert 1 op indien beide expressies een ongelijke waarde hebben. Anders 0.

#### 5. Functies

NOT(aexo)

Levert 1 op indien aexo gelijk nul is. Anders nul.

SGN(aexo)

Levert 1 of -1 op conform het algebraïsche teken van aexo.

INT(aexo)

Levert de integer waarde van aexo in real.

ABS(aexo)

Levert de absolute waarde van aexo.

USR(aexo)

Zet de waarde van aexo 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 flo accu worden gezet, waarna met RTS wordt teruggekeerd naar COMAL. Doet een 'carbage collect' en levert het aantal vrije geheugenlocaties op.

FRE(exo)

Levert het nummer van de laatst bedrukte kolom op.

POS(exo)

Levert de vierkantswortel van aexo.

SQR(aexo)

RND(aexo)

Indien aexo (0 wordt voor elke waarde een bijbehorend getal tussen 0 en 1 gegenereerd. Deze oneigenlijke RND functie kan dienen tijdens debuugen van programma's die gebruik maken van RND. Indien aexo > 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 oomieuw.

LOG(aexo)

Levert de natuurlijke logaritme van aexo. aexo()0 !

EXP(aexo)

Levert e tot de macht aexo. (e=2.71828183....)

COS(aexo)

Levert de cosinus van de hoek aexo (aexo in rad.)

SIN(aexo)

Levert de sinus van de hoek aexo (aexo in rad.)

TAN(aexo)

Levert de tangens van de hoek aexo (aexo in rad.) aexo()0i2 (pi=3.14159265)

ATN(aexo)

Levert de boog in rad. waarvan de tangens aexo is. aexo()1

PEEK(aexo)

Levert de inhoud van geheugenlocatie aexo. 0=(aexo=65535.

LEN(sexo)

Geeft de lengte van de string sexo.

STR\$(aexo)

Converteert de waarde van aexo tot string.

VAL(sexo)

Levert een getal gelijk aan de waarde van sexo, gelezen vanaf het begin van de string tot voor het eerste niet bruikbare karakter.

ASC(sexo)

Levert de ASCII code van het eerste karakter van sexo.

CHR\$(aexo)

Levert een string van een karakter dat de ascii code aexo heeft. 0=(aexo=255.

LEFT\$(sexo,aexo)

Levert een string bestaande uit de linker aexo karakters van sexo.

RIGHT\$(sexo,aexo)

Levert een string bestaande uit de rechter aexo karakters van sexo.

LABEL: sexo  
GOTO sexo  
ONERR GOTO sexo

Een LABEL statement wordt bij sequentiële executie genegeerd. Naar een label kan worden gesprongen met:  
Sprong naar LABEL:sexo. GOTO dient in principe slechts te worden gebruikt in combinatie met:  
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

RESUME

RUN sexo

//

STOP

WAIT aexo

LOAD, aexo

SAVE, aexo

DEF FN rvar1(rvar2)=fun(rvar2)

POKE aexo1, aexo2

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

Mag alleen gebruikt worden aan het einde van een foutafhandeling via ONERR. Het programma zal het statement waarin de fout ootrad opnieuw uitvoeren. Een prima toepassing van ONERR is detectie van invoerfouten. 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 zoeken van labels en procedures versnelt, doch de listing verniet. Correcte beëindiging 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 streken wordt door de interpreter bij executie overgeslagen. Men kan aldus verklarende tekst tussenvoegen. (REM in Basic)

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!

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

Leest programma nummer aexo van tape. 0(aexo/255. Indien men geen parameter specificiert of indien men deze geeft als 0 of 255 volgt de foutmelding ERROR zonder nadere aanduiding. Deze melding ontstaat ook bij een check sum error.

Schrijft het programma naar tape onder filenummer aexo. 0(aexo/255. Indien men geen parameter specificiert of indien men deze geeft als 0 of 255 volgt de foutmelding ERROR zonder nadere aanduiding.

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

Zet de waarde aexo2 in geheugenlocatie aexo1. Beide parameters worden automatisch tot integer geconverteerd. 0=(aexo1=65535 en 0=(aexo2=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 programmaaregel 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 invoeren! 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 geexecuteert geeft een verminkte listing. Zet in een dergelijk geval als eerste regel een LABEL: statement en start het programma met GOTO sexo waarbij tragere 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 'handt' 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,....,conm

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 inout-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 direct commando. Men krijgt dan de foutmelding ILLEGAL DIRECT ERROR. INPUT avar is niet toegestaan. Hiervoor dient INPUT svar gevolgd door avar:=val(sexo) te worden gebruikt. Indien het eerste karakter dat achter inout 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 inout 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(aexol,.....,aexon)

Reserveert ruimte voor een matrix met de naam var en met aexol+1 elementen in de eerste dimensie, tot aexon+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 programmaaregel. Indien de matrix var reeds bestond volgt de foutmelding REDIM'D ARRAY. Men kan naar een matrix element refereren met var(aexol,.....,aexon). Indien deze beschrijving niet overeenkomt met de door de DIM gedeclareerde beschrijving volgt de foutmelding BAD SUBSCRIPT ERROR. 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.

READ var

Staat toe alle data opnieuw te lezen. Een gedeeltelijke RESTORE is onmogelijk.

RESTORE

CALL aexo

Executeert een machinetaalroutine op adres aexo. aexo wordt automatisch tot integer geconverteerd. 0=(aexo = 65535). De machinetaalroutine dient te eindigen met een RTS om correct naar COMAL terug te keren. Asigneert de waarde van aexo aan var. var en aexo moeten tot hetzelfde datatype behoren, anders volgt de foutmelding TYPE MISMATCH ERROR. Bij uitzondering mogen meerdere assignments achter een regelnummer staan, gescheiden door punt-komma. Het sleutelwoord LET kent COMAL niet. Schreef het teken := uit als "wordt".

var :=aexo

MID\$(sexo, aexo1, aexo2)	Leverf 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 laatste gevalueerde expressie. Indien dat een string was volgt de foutmelding TYPE MISMATCH ERROR.

## 6. Control Structures.

IF-THEN-ELSE-ENDIF	<pre>IF aexo THEN   blk1 ELSE   blk2 ENDIF</pre> <p>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 waarna 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.</p>
WHILE-DO:-ENDWHILE	<pre>WHILE aexo DO:   blk ENDWHILE</pre> <p>Indien aexo ongelijk 0 is wordt verder gegaan op de regel na ENDDHILE. Anders wordt blk uitgevoerd waarna wordt teruggegaan naar WHILE, totdat een resultaat ongelijk 0 is verkregen. WHILE zonder ENDDHILE of omgekeerd geeft de foutmelding 'BAD FLOW OF CONTROL'.</p>
REPEAT-UNTIL	<pre>REPEAT   blk UNTIL aexo =</pre> <p>Voert eerst blk uit daarna wordt aexo gevalueerd en blk zonedig herhaald tot aexo=0. UNTIL zonder REPEAT geeft de foutmelding 'BAD FLOW OF CONTROL'.</p>
FOR-TO-STEP-ENDFOR	<pre>FOR rvar:=aexo1 TO aexo2 STEP aexo3   blk ENDFOR</pre> <p>Geefr rvar de waarde van aexo1 en voert blk uit. Daarna wordt rvar met aexo3 vermeerderd. Indien dit resultaat =aexo2 is wordt blk eenieuw 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 men in blk een FOR-loop oent met dezelfde rvar volat SYNTAX ERROR. ENDFOR zonder voorafgaande FOR geeft de foutmelding NEXT WITHOUT FOR ERROR.</p>
CASE-WHEN-OTHERWISE-ENDCASE	<pre>CASE var   WHEN exo1     blk1   WHEN exon     blkn   OTHERWISE     blk3 ENDCASE</pre>

Voor var mag geen integer of matrix element worden gebruikt. Het aantal WHEN-blokken is onbeperkt. Een voor een worden de expressies eval.... exon gevalueerd 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 een expressie wordt gevonden die voldoet, wordt het blok onder OTHERWISE uitgevoerd. Indien OTHERWISE ontbreekt en geen enkele expressie voldeed 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 sexo,var1,....,varn
  blk
ENDPROC
```

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

EXEC: sexo,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 weggelaten worden, 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 sequentiële executie wordt aangetroffen volgt SYNTAX ERROR. Indien ENDPROC wordt aangetroffen zonder aanroep door EXEC: volgt de foutmelding ENDPROC WITHOUT EXEC.

## 8. COMAL\*.KGM in Uw JUNIOR.

COMAL\*.KGM 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 videoshakeling met beeldschermgeheugen, waarin COMAL vervolgens strings ging ooslaan. U kunt in zulke gevallen het geheugengebruik van



COMAL betugelen 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 prompting 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-interoreter 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 \$80 (lo,hi). Type altijd CLEAR na een warme start. Anders bestaat de mogelijkheid dat U COMAL en programma kwijt raakt !

+++++

```
;-----  
;      Changes made in COMAL version 2.1 for DOS65  
;=====
```

Antoine Megens

May 1987

```
;-----  
;Directions:
```

```
;Load old COMAL with LOAD S:COMAL, then enter MONITOR.  
;Change the following addresses and save new COMAL with:  
;SAVE S:NCOMAL 2000,4DFF,3000. Then enter SETMODE -C  
;S:NCOMAL and test the changes with NCOMAL. The RUBOUT key  
;should work on screen now and when the following COMAL  
;program is executed, the file TEST.DAT should be closed  
;with a $00 byte instead of $1C (check this with DOS65  
;command DUMP TEST.DAT).
```

```
; 100 CREATE "TEST.DAT"  
; 110 OPEN #1;"TEST.DAT"  
; 120 PRINT #1;"Testing EOF change"  
; 130 CLOSE #1
```

```
; If this works you may rename the file NCOMAL to COMAL.  
;-----
```

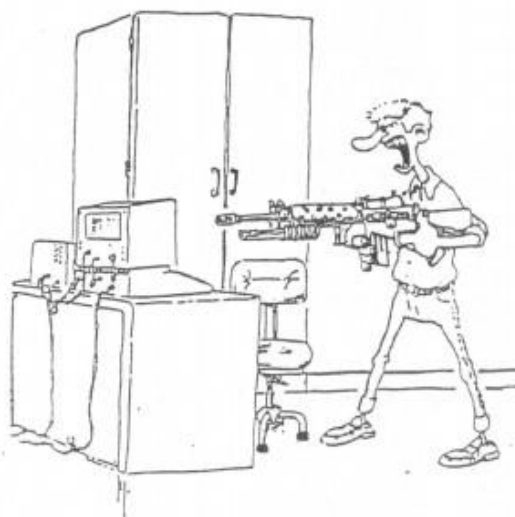
28D2	4C D0 4D	JMP	\$4DD0	;was JMP \$C023,
				;now check RUBOUT
				;first
4C94	A9 00	LDA	#\$00	;was LDA #\$1C
				; RUBOUT check routine (unused space in old COMAL)
4DD0	C5 2D	CMP	\$2D	;RUBOUT character?
4DD2	F0 03	BEQ	\$4DD7	;yes! else
4DD4	4C 23 C0	JMP	\$C023	;just print char.
4DD7	8A	TXA		;at zero position
				;of input?
4DD8	10 03	BPL	\$4DDD	;no, continue
4DDA	A5 2D	LDA	\$2D	;else exit with
				;RUBOUT char.
4DDC	60	RTS		;(do nothing)
4DDD	A9 08	LDA	#\$08	;print {BS}{space}
				;{BS}
4DDF	20 23 C0	JSR	\$C023	;to simulate
				;RUBOUT on screen
4DE2	A9 20	LDA	#\$20	
4DE4	20 23 C0	JSR	\$C023	
4DE7	A9 08	LDA	#\$08	
4DE9	20 23 C0	JSR	\$C023	
4DEC	A5 2D	LDA	\$2D	;exit with RUBOUT
				;char. in A
4DEE	60	RTS		



```

1000 // *****
1010 // * AMAZING MAZE V.1.COMAL *
1020 // *-----*
1030 // * A.Megens febr.87 *
1040 // * for MON/DOS65 systems *
1050 // *****
1060 //
1070 B:=26
1080 H:=10
1090 DIM D(B,H),B(3),M(2*B+1,2*H+1),V(2*B+1,2*H+1)
1100 C1:=1/(B+H)
1110 C2:=.7
1120 C3:=.8
1130 C4:=.5
1140 E:=B*H
1150 I:=INT(((RND(1)+.5)*B)/2)
1160 T:=1
1170 A$:="400140240124304134324132"
1180 GRAF$:=CHR$(27)+"F"
1190 TEXT$:=CHR$(27)+"G"
1200 FOR X:=0 TO B
1210   FOR Y:=0 TO H
1220     D(X,Y):=0
1230   ENDFOR
1240 ENDFOR
1250 X:=1
1260 Y:=0
1270 D(X,Y):=1
1280 X:=X-1
1290 B:=B-1
1300 H:=H-1
1310 A:=0
1320 P:=0
1330 // ***** MAIN LOOP MAZE GENERATOR *****
1340 REPEAT
1350   PRINT T,
1360   REPEAT
1370     IF D(X,Y)=0 OR (A+P)=0 THEN
1380       REPEAT
1390         X:=X+1
1400         IF X>B THEN
1410           X:=0
1420           Y:=Y+1
1430           IF Y>H THEN
1440             Y:=0
1450           ENDIF
1460         ENDIF
1470         UNTIL D(X,Y)<>0
1480       ENDIF
1490       A:=0
1500       P:=0
1510       IF X<B THEN
1520         IF D(X+1,Y)=0 THEN
1530           P:=1
1540         ENDIF
1550       ENDIF
1560       IF X>0 THEN
1570         IF D(X-1,Y)=0 THEN
1580           P:=P+2
1590         ENDIF
1600       ENDIF
1610       IF Y<H THEN
1620         IF D(X,Y+1)=0 THEN
1630           P:=P+4
1640         ENDIF
1650       ENDIF
1660       IF Y>0 THEN
1670         IF D(X,Y-1)=0 THEN
1680           A:=1
1690         ENDIF
1700       ENDIF
1710       IF P>0 THEN
1720         A:=A+1
1730         IF P>2 THEN
1740           A:=A+1
1750         ENDIF
1760       ENDIF
1770     UNTIL (A+P)<>0

```



*Waddaya mean, user error!?*

```

1780 FLAG:=0
1790 REPEAT
1800 REPEAT
1810 Q:=3*P+INT(RND(1)*A+1)
1820 UNTIL MID$(A$,Q,1)<>"0"
1830 C$:="FUN"+MID$(A$,Q,1)
1840 EXEC: C$,FLAG,X,Y,D(X,Y),C2
1850 UNTIL FLAG<>0
1860 PRINT
1870 T:=T+1
1880 IF RND(1)<C1 THEN
1890 X:=INT(RND(1)*B)
1900 Y:=INT(RND(1)*H)
1910 ENDIF
1920 UNTIL T>=E
1930 CLS
1940 D(B-I,H):=D(B-I,H)+4
1950 EXEC: "MAZE"
1960 END.
1970 //
1980 PROC "MAZE"
1990 PRINT GRAF$;
2000 Y:=0
2010 FOR X:=0 TO B
2020 IF X=I THEN
2030 PRINT "Z ";
2040 ELSE
2050 PRINT "ZXX";
2060 ENDIF
2070 ENDFOR
2080 PRINT "Z"
2090 FOR Y:=0 TO H
2100 FOR X:=0 TO B
2110 IN:=D(X,Y)
2120 EXEC: "BINARY",IN,B(1),B(2)
2130 IF B(1)=1 THEN
2140 PRINT " ";
2150 ELSE
2160 PRINT "Y ";
2170 ENDIF
2180 ENDFOR
2190 PRINT "Y"
2200 FOR X:=0 TO B
2210 IN:=D(X,Y)
2220 EXEC: "BINARY",IN,B(1),B(2)
2230 IF B(2)=1 THEN
2240 PRINT "Z ";
2250 ELSE
2260 PRINT "ZXX";
2270 ENDIF
2280 ENDFOR
2290 PRINT "Z"
2300 ENDFOR
2310 PRINT TEXT$
2320 ENDPROC
2330 //
2340 PROC "FUN1",FLAG,X,Y,D(X,Y),C2
2350 IF RND(1)<C2 THEN
2360 FLAG:=0
2370 ELSE
2380 X:=X+1
2390 PRINT "+X";
2400 D(X,Y):=2
2410 FLAG:=1
2420 ENDIF
2430 ENDPROC
2440 //
2450 PROC "FUN2",FLAG,X,Y,D(X,Y),C2
2460 IF RND(1)<(1-C2) THEN
2470 FLAG:=0
2480 ELSE
2490 D(X,Y):=D(X,Y)+2
2500 X:=X-1
2510 PRINT "-X";
2520 D(X,Y):=1
2530 FLAG:=1
2540 ENDIF
2550 ENDPROC
2560 //
2570 PROC "FUN3",FLAG,X,Y,D(X,Y),C2
2580 IF RND(1)<C3 THEN
2590 FLAG:=0
2600 ELSE
2610 D(X,Y):=D(X,Y)+4
2620 Y:=Y+1
2630 PRINT "+Y";
2640 D(X,Y):=1
2650 FLAG:=1
2660 ENDIF
2670 ENDPROC
2680 //
2690 PROC "FUN4",FLAG,X,Y,D(X,Y),C2
2700 Y:=Y-1
2710 PRINT "-Y";
2720 D(X,Y):=4
2730 FLAG:=1
2740 IF RND(1)<C4 THEN
2750 C2:=1-C2
2760 ENDIF
2770 ENDPROC
2780 //
2790 PROC "BINARY",IN,B(1),B(2)
2800 NUM:=IN
2810 FOR I:=3 TO 0 STEP -1
2820 IF NUM-2^I>=0 THEN
2830 B(I):=1
2840 NUM:=NUM-2^I
2850 ELSE
2860 B(I):=0
2870 ENDIF
2880 ENDFOR
2890 ENDPROC
2900 //

```

pag. 24

