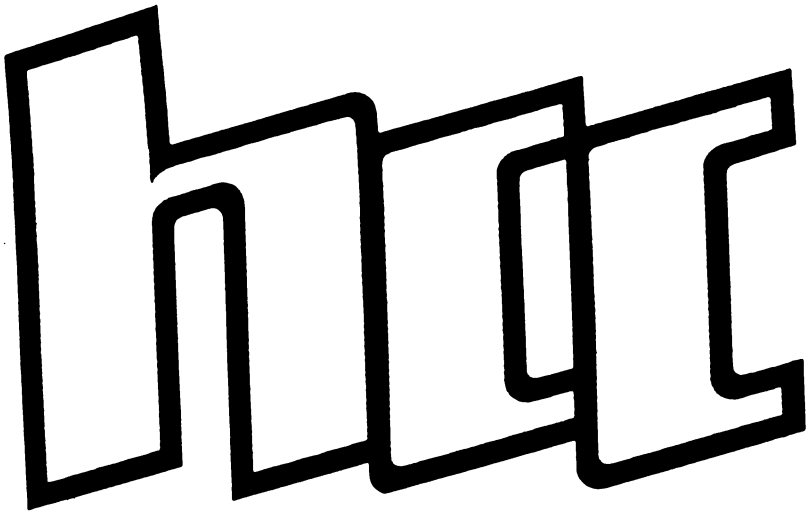


# OSI Gebruikersgroep

## deel 14

sept. 1985

Voor vragen en/of opmerkingen:
John Hermans
Meerkoetlaan 5
2636 ER Schipluiden
01738 - 8703

Een uitgave van Hobby Computer Club Nederland
Postbus 149 - 2250 AC Voorschoten

De 65C02 processor.

Een beetje geschiedenis.
Een van de meest gebruikte microprocessors in microcomputers is nog steeds de 6502. Deze processor werd oorspronkelijk ontwikkeld door de toenmalige MOS Technology Inc. (thans: Commodore Semiconductor Group Inc.), uit ontevredenheid met de 6800 processor van Motorola. Degene die met deze 6800 ontevreden was, is in PET/CBM kringen redelijk bekend: Chuck Peddle, die later de PET 2001 zou ontwerpen.
Ten opzichte van de 6800 kreeg de 6502 een aantal adresseermodes extra (waaronder indirecte modes), een tweede indexregister, en een accu(mulator) minder. Verder werd de interne architectuur vereenvoudigd, en de instructieafwikkeling versneld. Dit alles leverde een veelzijdige processor op, die goed geschikt bleek om hogere programmeertalen te verwerken via een interpreter. Het bekendste voorbeeld hiervan is de Microsoft BASIC interpreter, die vermoedelijk de meest verkochte interpreter ter wereld is.
De gelijkenis met de 6800 uit zich vooral in de hardware: de 6502 gebruikt dezelfde signalen als de 6800, al hebben ze soms een andere naam gekregen. Ook de pin-out van het IC lijkt enigszins op die van de 6800.
De 6502 heeft echter niet alleen extra's ten opzichte van de 6800. De instructieset werd namelijk kleiner, door een aantal instructies weg te laten die minder vaak gebruikt werden, en die vervangen kunnen worden door andere instructies die wel in de set voorkomen. Een voorbeeld hiervan is:

```
6800:  ADDA #$C9     ;tel $C9 zonder carry bij accu A op

6502:  CLC           ;maak de carry schoon
       ADC  #$C9     ;tel $C9 en de carry bij de accu op
```

In dit geval heeft de 6502 dus een instructie extra nodig. Dit wordt 'goedgemaakt' door het feit dat bij de 6502 de meeste instructies in minder clockcycles worden uitgevoerd dan bij de 6800.
Tot zover het ontstaan van de 6502.

De techniek schrijdt voort.
De 6502 is een IC dat in zogenoemde N-MOS technologie is opgebouwd. Dit wil zeggen, dat in het IC alleen N-channel MOSFETs voorkomen. Deze FETs worden gebruikt als schakel-transistor, of als drain-weerstand. Dit houdt in, dat in die weerstanden warmte gedissipeerd kan worden, zodat het energiegebruik van de chip behoorlijk kan zijn.
Dit klopt in de praktijk; een 6502 processor in vol bedrijf wordt redelijk warm.
In het begin van de jaren '70 werd een nieuw proces van IC-fabricage ontwikkeld: CMOS. Bij dit proces zitten op de chip zowel N- channel als P-channel FETs. Karakteristiek voor deze technologie is, dat de als weerstand gebruikte FETs ontbreken. Van een paartje FETs is er steeds 1 in geleiding: of de N-channel FET geleidt, en 'trekt' de uitgang naar de nul, of de P-channel FET 'trekt' de uitgang naar de voedingsspanning. Omdat steeds slechts 1 FET geleidt, loopt er

in het totale circuit alleen een (kleine) lekstroom. Het stroomverbruik van een CMOS IC in stabiele toestand is dan ook griezelig klein: een paar micro-amperes.
Dit wordt anders als de uitgang van 1 naar nul of andersom verandert: dan geleiden gedurende de overgang beide FETs even, waardoor er eventjes een grotere stroom gaat lopen. Dit is karakteristiek voor CMOS, en is er de oorzaak van, dat het stroomverbruik toeneemt met de schakelsnelheid (=clock frequentie).
In het begin van de jaren '70 kon men alleen kleine IC's integreren in CMOS. Naarmate de tijd verstreek, verschoof de grens van de maximaal integreerbare grootte, en thans is men aangeland op een zodanig niveau dat een statisch CMOS geheugen van 8 kilobyte (64K bit!) mogelijk is.
Zoals gezegd is 1 van de voordelen van CMOS het geringe stroomverbruik. Dit is vooral van belang in toepassingen waar een schakeling uit batterijen gevoed moet worden. Vandaar dat langzamerhand verschillende processors die in NMOS gemeengoed zijn, in CMOS technologie gaan verschijnen. Twee voorbeelden daarvan zijn de CMOS Z80 en de CMOS 6502. Deze laatste processor heeft als typenummer 65C02 gekregen.

Hardware verschillen tussen de 6502 en de 65C02.
Het eerder genoemde verschil in stroomverbruik is duidelijk te merken: bij een clockfrequentie van 1 MHz verbruikt de NMOS versie ca. 120mA, de CMOS versie is met ca. 4 mA tevreden. Dit uit zich vooral in de warmteontwikkeling: de 65C02 houdt het hoofd koel, hij wordt zelfs niet handwarm.
Er zijn nog meer verschillen. Sommige versies van de 65C02 bezitten een ingang om alle buslijnen in tri-state te zetten. Mensen die graag DMA plegen, roepen nu hoera. Om echt DMA mogelijk te maken, is er op deze chips ook nog een uitgang bijgekomen: Memory Lock. Deze uitgang geeft aan, dat de processor bezig is met een instructie die niet mag worden onderbroken, omdat het resultaat nog moet worden teruggeschreven in het geheugen. (Voorbeelden hiervan zijn ROR, ROL, ASL en LSR instructies).
Deze extra pennen zijn ondergebracht op aansluitingen die bij de NMOS 6502 'not connected' waren. De verdere aansluitingen zijn precies gelijk gebleven. Dit houdt in dat de 65C02 zonder meer in een voet bedoeld voor de 6502 geprikt mag worden.
U vraagt zich nu natuurlijk af: 'Ja, maar hoe zit dat dan met de software?'

CMOS software perikelen.

Om maar direct met de deur in huis te vallen: ALLE 6502 instructies zijn precies dezelfde gebleven. De 65C02 is dus volledig uitwisselbaar met de NMOS 6502. U kunt dus in een OSI de processor zonder meer vervangen door een CMOS exemplaar.
U zult echter denken: 'Wat schiet ik ermee op?', want een besparing in stroomgebruik van ca. 115 mA op een totaal van wellicht een paar Ampere is nauwelijks relevant. Dat klopt, maar er is echter meer.
Behalve de bestaande instructieset, heeft de 65C02 nog een aantal instructies en adresseermodes extra meegekregen. De extra's zijn de volgende:

1. Alle illegale opcodes leiden tot 1, 2 of 3 NOPs. Dit houdt in, dat de processor zich niet meer kan 'ophangen'. Berucht bij de 6502 zijn de opcodes die in hexadecimaal op 2 eindigen; deze laten de processor hangen, waarbij alleen een reset uitkomst biedt. De CMOS versie loopt dus gewoon verder.

2. Bij een reset wordt de D-vlag op nul gezet. In de resetroutine komt de CLD instructie dus te vervallen.

3. De vlaggen C, N en V waren bij de NMOS versie alleen geldig in hexadecimale mode. In decimale mode reageerden zij foutief. In de 65C02 zijn deze vlaggen ook geldig in decimale mode.

4. Nieuwe adresseermodes. Er zijn bij de 65C02 twee nieuwe adresseermodes bijgekomen. De eerste is indirect, en mag gebruikt worden bij de instructies LDA, STA, ORA, AND, EOR, ADC, SBC en CMP. De notatie is als volgt:

```
B2 C8   LDA ($C8)
```

Deze adresseermode vervangt:

```
6502: LDY #$00
      LDA ($C8),Y
en:
      LDX #$00
      LDA ($C8,X)
```

In beide gevallen is de 65C02 oplossing twee bytes en twee cycles korter. Daar in zeker de helft van de gevallen de offset (X of Y) toch nul is, kan dit een zinvolle besparing opleveren.
De tweede nieuwe addresseermode is 'Absolute Indirect Indexed' en komt bij 1 instructie voor, namelijk JMP. De opcode hiervan is $7C. Stel de volgende (veel voorkomende) constructie:

```
6502: TXA             ;haal commandonummer in A
      ASL             ;vermenigvuldig met twee
      TAX             ;en zet offset in X
      LDA ADRTAB,X    ;haal commando adr. lo
      STA JUMPADR     ;zet in geheugen
      LDA ADRTAB+1,X  ;haal hi-byte ook
      STA JUMPADR+1   ;maak JMP adres compleet
      JMP (JUMPADR)   ;en voer commando uit.
```

Met de 65C02 wordt dit:

```
65C02: TXA              ;haal commandonummer in A
       ASL              ;vermenigvuldig met twee
       TAX              ;en zet offset in X
       JMP (ADRTAB,X)   ;en voer commando uit.
```

Ook hier een belangrijke besparing in bytes en cycles.

5. Nieuwe instructies.
Behalve nieuwe addresseermodes, zijn er ook nieuwe

instructies bijgekomen, die al jaren op de verlanglijstjes van een groot aantal machinetaalprogrammeurs stonden. Deze instructies zijn:

5A. X en Y kunnen nu rechtstreeks op de stack gezet en van de stack gehaald worden. De mnemonics zijn: PLX, PLY, PHX en PHY.
Voorbeeld (verwissel X en Y):

```
6502:  PHA      ;redt de accu
       TYA      ;haal Y in A
       PHA      ;redt Y
       TXA      ;breng X naar A
       TAY      ;en A naar Y
       PLA      ;haal oude Y
       TAX      ;in X
       PLA      ;en haal A terug

65C02: PHY      ;redt Y
       PHX      ;en X
       PLY      ;haal oude X in Y
       PLX      ;en oude Y in X
```

Bij de 6502 moeten dergelijke operaties via A verlopen, waardoor A soms verloren gaat. De 65C02 heeft maar 4 bytes aan instructies nodig, en gebruikt noch de accu, noch geheugen.

5B. Nul maken van een geheugenlocatie kan nu rechtstreeks met STZ (=STore Zero). De toegestane adresseermodes zijn:

```
Absolute           : STZ $804F
Zero Page          : STZ $CD
Zero Page Indexed  : STZ $40,X
Absolute Indexed   : STZ $9D45,X
```

Dus geen LDA #$00 gevolgd door STA .... meer! Ook hier het voordeel dat geen der registers verloren gaat. STZ met accuadressering is zinloos: gewoon LDA #$00 bestond al.

5C. De accu kan nu rechtstreeks worden opgehoogd en afgelaagd met INA (INcrement Accu) en DEA (DEcrement Accu). Geknutsel met CLC/ADC #$01 en SEC/SBC #$01 en hun varianten behoort nu tot het verleden.

5D. Er kan met de 65C02 relatief en 'unconditional' gesprongen worden met: BRA (=BRanch Always). Dit biedt de mogelijkheid programma's reloceerbaar te maken.

5E. De 65C02 is uitgebreid met een tweetal instructies die rechtstreeks op bit-niveau werken; het uitzeven van bits met AND en OR instructies, of het verschuiven met shift-instructies komt dus te vervallen. De instructies zijn:

TRB : Test en Reset Bit(s). Zet het resultaat
van /A AND M in M, en zet de zerovlag Z
als het resultaat van A AND M nul is.

TSB : Test en Set Bit(s). Zet het resultaat van A OR M in M, en de Z-vlag als A AND M nul is.

Voorbeelden:

a. Zet bit 3 van $86 op nul.

```
6502 :  LDA $86
        AND #%11110111
        STA $86

65C02:  LDA #%00001000
        TRB $86
```

b. Zet de bits 5 en 7 van $50FD op 1 en spring naar PC+28 indien deze bits nul waren.

```
6502 :  LDA $50FD       ;haal waarde op
        PHA             ;redt de waarde
        ORA #%10100000  ;zet de gewenste bits
        STA $50FD       ;en zet de waarde weg
        PLA             ;haal de waarde terug
        AND #%01011111  ;zet vlaggen
        BEQ =+28        ;en spring indien nul

65C02:  LDA #%10100000  ;geef de bits aan
        TSB $50FD       ;zet deze bits
        BEQ =+28        ;spring indien b6 nul
```

Deze instructies kennen twee adresseermodes: Absolute en Zero page.
Merk verder op dat de te bewerken bits altijd als enen in de accu staan; bij de AND/OR methode moeten ze bij het resetten als nullen worden voorgesteld (zie eerste voorbeeld).
Indien zowel bits op nul als op een gezet moeten worden, maak het gebruik van de nieuwe instructies niet meer uit:

```
6502 : LDA $86          ;haal waarde op
       ORA #%00100010 ;zet de bits 1 en 5
       AND #%11101111 ;clear bit 4
       STA $86          ;en zet resultaat weg

65C02: LDA #%00100010 ;geef bits aan
       TSB $86          ;zet deze bits
       LDA #%00010000 ;geef bits aan
       TRB $86          ;en zet deze op nul
```

5F. De BIT-instructie is uitgebreid met drie extra adresseermodes:

```
BIT Immediate
BIT Zero Page,X
BIT Absolute,X
```

De nieuwe opcodes met hun mnemonics zijn (in hexadecimale volgorde):

| Opc. | Mnemonic | Opc. | Mnemonic |
|---|---|---|---|
| 04 | TSB Zero page | 74 | STZ Zero Page,X |
| 0C | TSB Absolute | 7A | PLY |
| 12 | ORA Indirect | 7C | JMP Indirect Indexed |
| 14 | TRB Zero Page | 80 | BRA Relative |
| 1A | INA | 89 | BIT Immediate |
| 1C | TRB Absolute | 92 | STA Indirect |
| 32 | AND Indirect | 9C | STZ Absolute |
| 34 | BIT Zero Page,X | 9E | STZ Absolute,X |
| 3A | DEA | B2 | LDA Indirect |
| 3C | BIT Absolute,X | D2 | CMP Indirect |
| 52 | EOR Indirect | DA | PHX |
| 5A | PHY | F2 | SBC Indirect |
| 64 | STZ Zero Page | FA | PLX |
| 72 | ADC Indirect | | |

Schrijfwijze van de nieuwe adresseermodes:

```
Indirect                    : ($HH)
Indirect Absolute Indexed   : ($HHHH,X)
```

Addertje onder het gras!
Er zijn twee fabrikanten van de 65C02. Deze fabrikanten zijn: Rockwell en GTE Microcircuits. Merk op dat MOS Technology (Lees: Commodore) heeft afgehaakt. Het addertje onder het gras is, dat de Rockwell 65C02 nog eens 32 instructies extra heeft boven de in dit artikel genoemde. Mocht hiervoor belangstelling zijn, dan zullen deze in een volgend artikel worden uitgelegd.

Samenvatting.
De nieuwe 65C02 processor is 'pin-to-pin compatible' met de NMOS 6502. Dit houdt in, dat de 65C02 de 6502 in alle toepassingen kan vervangen, zonder dat er wijzigingen in hard- of software nodig zijn. Er gelden hierbij de volgende voordelen:
1. Lager stroomverbruik
2. Processor kan zich niet meer ophangen.
3. Grotere en krachtiger instructieset.
Een nadeel is dat software geschreven voor de 65C02 een gewone 6502 kan laten crashen, daar de nieuwe instructies illegale opcodes voor de 6502 vormen.

Gemaakt door: Nico de Vries
Mari Andriessenrade 49
2907 MA Capelle aan den IJssel
Telefoon (010)-502239

# WIJZIGINGEN EN TIPS VOOR MOSI.

Bij de meeste MOSI computers staat het keyboard los van de computer en wordt deze aangesloten met een 15 aderig soepel snoer en een 15 polige DELTA konnektor. Door dit systeem is de databus van de computer rechtstreeks met deze kabel verbonden. Dit heeft diverse nadelen:

1) Storing van buitenaf die de keyboardkabel oppikt kan de werking van de processor beinvloeden en zelfs stoppen.
2) De bedradingscapaciteit wordt te groot, op 2 MHz blijft er van de signalen niets meer over.
3) De kabel moet zeer kort zijn om dit te voorkomen.

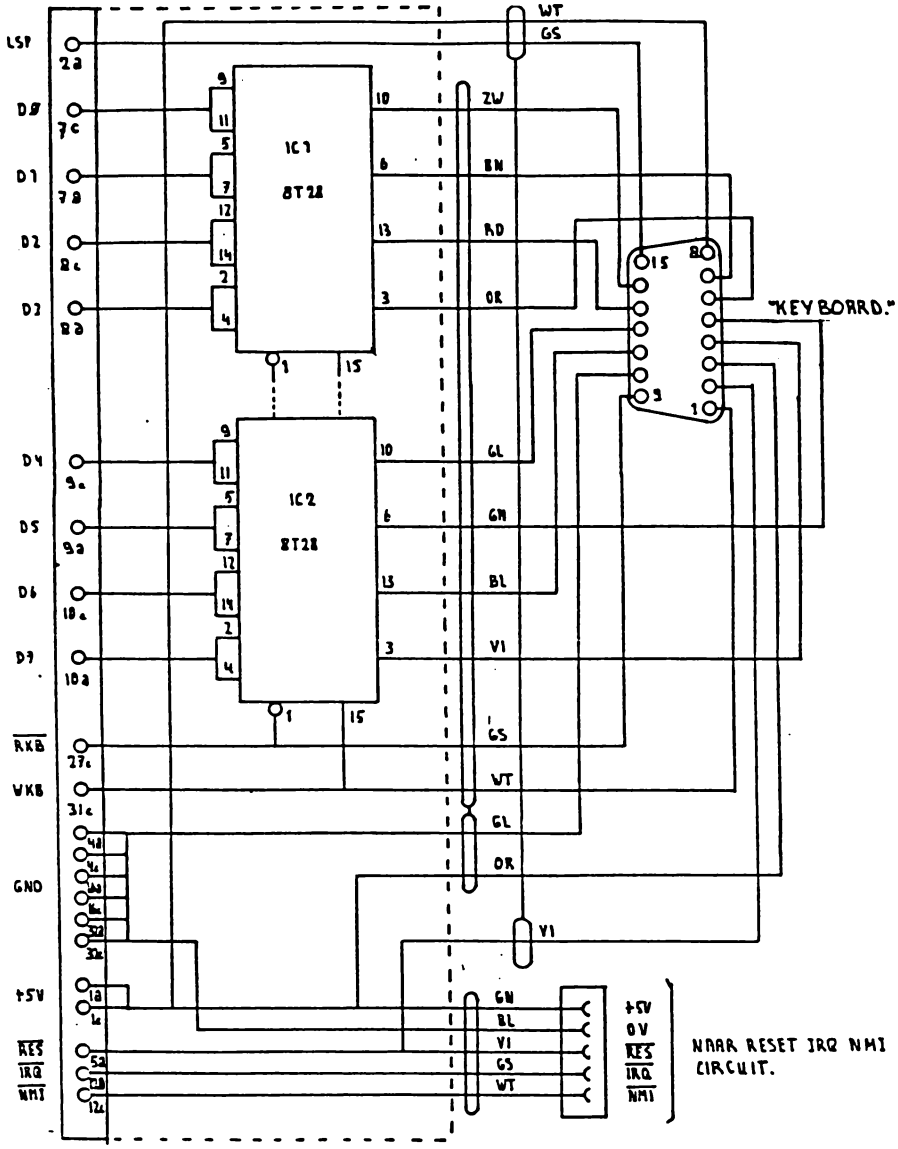
De oplossing hiervoor is een buffer te plaatsen om de keyboardsignalen van de databus te scheiden. Hiervoor kan een klein printje geplaatst worden met 2 maal 8T28 waarop dan tevens andere verbindingen met de bus gemaakt kunnen worden. Door deze buffer te plaatsen is de databus alleen tijdens lees en schrijfoperaties verbonden met de kabel. Eventuele storingen worden dan door de keyboardsoftware geheel onderdrukt. Een kabellengte van twee meter is nog goed bruikbaar. Gebruik echter geen afgeschermde kabel want hiervan geeft de bedradingscapaciteit toch nog problemen. De gangbare kabels hebben 16 aders, gebruik voor de 0 volt twee aders parallel. Ook is via dit printje het luidspreker signaal vanaf buspen 2a verbonden met keyboard konnektor pen 15. Een extra +5 volt is aangesloten op keyboardpen 8. Dit is gescheiden van de +5 volt voor de keyboardelectronica om storing hierop te voorkomen. In het keyboard kan nu de luidspreker gemonteerd worden wat prettiger klinkt als een piepje vanuit de verte. Op de CPU kaart dient hiervoor een verbinding gelegd te worden tussen pen 2a van de bus en de weerstand van 100 Ohm die aan pen 10 van IC 19 zit.

Op het extra printje zijn ook de NMI, IRQ en RESET signalen naar het front van de computer gebracht. Op het front kan het IRQ, NMI en RES. schakelingetje geplaatst worden. Met deze schakeling is het mogenlijk zelf interrupts te geven maar ook te zien als er een staat. Dit kan handig zijn als een programma blijft hangen op een foutief verwerkte hardware interrupt. (bron Databus april 1980). Het reset circuit verzorgd tevens de power-on reset. Een power-on reset is ook op het keyboard dekoderprintje van de MOSI aanwezig, deze hoeft niet te worden aangesloten. Gebruikt men deze wel dan moet op het nieuwe printje de onderdelen met een * worden weggelaten en de verbinding --▼-- worden aangebracht. Het zelf geven van interrupts is bijvoorbeeld handig bij de toolkit met de schermdump, deze kan met de NMI-toets

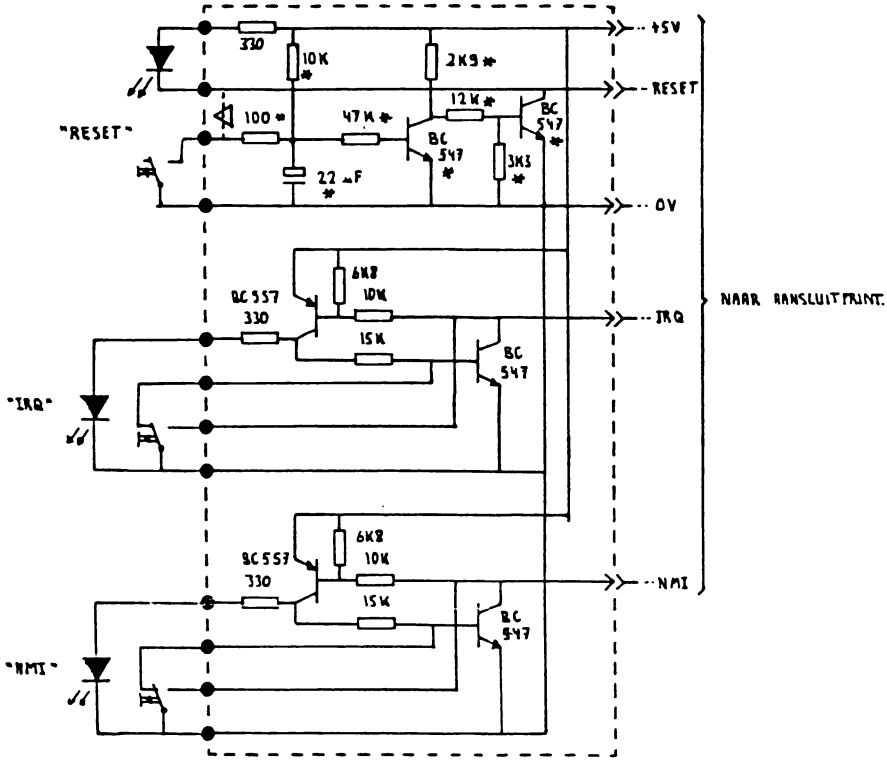
gestart worden. Ook is het mogenlijk om bij het debuggen van machinetaal programma's eerst de NMI-vector op de Extended monitor te zetten zodat als het programma hangt met een druk op de knop naar de ext. monitor teruggekeerd kan worden.

Chiel Broek
IJmuiden.
02550-32169

SEPTEMBER 1985




Aansluitprint met keyboardbuffer.




Schakeling NMI, IRQ en RESET.

OSI POEL

SEPTEMBER 1985

## Nog een real time klok voor OSI.

Deze klok is gebaseerd op het klok-IC MM-58167-A. Dit IC bevat naast een 8-byte klok ook 8 bytes RAM en een aantal controle registers. (Zie fig.1)
De overige eigenschappen zijn:

- 8 bytes klok in BCD-FORMAAT (hex-compatible),
- uitgebreide interrupt mogelijkheden,
- 32.768 Hz kristal,
- powerdown mode en powerdown interrupt.

De klok kan 8 verschillende interrupt signalen genereren, door het juiste bit op adres $11 hoog te maken (Zie fig.2).
Bij de "compare"-interrupt worden alleen die nibbles vergeleken, waarvan de bits 2 & 3 NIET BEIDE HOOG zijn. De zgn. ONGEBRUIKTE bits worden LAAG verondersteld. Welk interrupt gegenereerd was kan bekeken worden door adres $10 te lezen. Tevens wordt zo de interrupt GERESET.

Als pen 23 van de chip laag wordt gaat de klok in de zgn. powerdown mode. Het stroomverbruik gaat omlaag naar zo'n 20 uA, zodat twee Ni-Cd batterijtjes makkelijk maandenlang voor de voeding kunnen zorgen! Nu wordt de powerdown interrupt geactiveerd, als dat tenminste via adres $16 was geprogrammeerd. Deze interrupt kan bijv. worden gebruikt om de computer aan te zetten, of om processen te sturen, die onafhankelijk van het lichtnet werken.

Het zgn. GO-commando is niet nodig om de klok aan te zetten, maar een schrijfactie naar dit adres ($15) zet de registers $00 t/m $02 op 0 en verhoogt $03 (de minuten) met 1 als de seconden op 40 of hoger stonden. Dit is een eenvoudige manier om de klok precies gelijk te zetten (met bijv. de radio).

Door enen naar bits van adres $12 of $13 te schrijven worden de corresponderende bytes van de klok of van het RAM op nul gezet.

Het status-bit (D0 van adres $14) wordt wordt hoog als de klok gelezen wordt TERWIJL de tellers van de klok verhoogd worden. Dit kan dus een teken zijn dat de klok VERKEERD gelezen is. Het status-bit wordt GERESET door adres $14 te lezen.

Voor het schema zie figuur 3.
Omdat RD(genegeerd) en WR(genegeerd) NA CS(genegeerd) moeten komen is O2 (phi2) aan R/W gekoppeld.
Omdat de MM-58167 TRI-STATE data lijnen heeft en omdat de schakeling voor alle adreslijnen een FAN-IN van 1 heeft is buffering niet nodig en dus ook niet toegepast.
In het schema is een schakelingetje opgenomen dat er voor zorgt, dat de batterij wordt opgeladen als de computer aan staat.

Omdat ook de registers $00 t/m $07 zich als RAM gedragen is het misschien handig om (via een schakelaartje) een ENABLE signaal aan WR(genegeerd) te koppelen. (Zie fig.4) (Dan wel de adreslijnen bufferen met bijv. een 74LS241)
De klok staat op de adressen $C300 t/m $C31F.

Helaas heb ik nog geen tijd gehad om een goede print te ontwerpen, maar op een stuk gaatjes-print gaat het ook uitstekend.

Bijgaande programma's passen op de plaats van de oude #C klok uit de Toolkit. Er blijft ruimte over voor bijv. het type-commando !
Het programma voor de nieuwe #C.

```
                *=$E5AD
          SCREEN=$D12C
            KLOK=$C302

-------------------------------------------------------------------
NMI-Programma
-------------------------------------------------------------------
E5AD 48              PHA           ; red de registers
E5AE 8A              TXA           ;
E5AF 48              PHA           ;
E5B0 88              TYA           ;
E5B1 48              PHA           ;
E5B2 AD10C3          LDA $C310     ;reset interrupt
E5B5 A208            LDX #8        ;9 scherm posities
E5B7 A000            LDY #0        ;3 bytes
E5B9 B902C3  NEXT    LDA KLOK,Y    ;sec. min. uur.
E5BC 290F            AND #$0F      ;low nibble
E5BE 0930            ORA #$30      ;maak karakter
E5C0 9D2CD1          STA SCREEN,X  ;
E5C3 CA              DEX           ;
E5C4 B902C3          LDA KLOK,Y    ;high nibble
E5C7 4A              LSR A         ;
E5C8 4A              LSR A         ;
E5C9 4A              LSR A         ;
E5CA 4A              LSR A         ;
E5CB 0930            ORA #$30      ;maak karakter
E5CD 9Dc2D1          STA SCREEN,X  ;
E5D0 CA              DEX           ;
E5D1 A93A            LDA #':       ;':' naar scherm
E5D3 9D2CD1          STA SCREEN,X  ;
E5D6 CA              DEX           ;
E5D7 C8              INY           ;volgende ?
E5D8 C003            CPY #3        ;
E5DA D0DD            BNE NEXT      ;
E5DC A963            LDA #'c       ;en een 'c' ervoor
E5DE 8D2CD1          STA SCREEN    ;
E5E1 68              PLA           ;herstel registers
E5E2 A8              TAY           ;
E5E3 68              PLA           ;
E5E4 AA              TAX           ;
E5E5 68              PLA           ;
E5E6 40              RTI           ;end.
-------------------------------------------------------------------
TYPE-MACHINE
-------------------------------------------------------------------
E5E7 2000FD  TYPE    JSR $FD00     ;haal karakter
E5EA 20EEFF          JSR $FFEE     ;en weg.
E5ED C91B            CMP #$1B      ;<ESC> ?
E5EF D0F6            BNE $TYPE     ;
```

```
E5F1 4C2CFF          JMP $FF2C      ;warme start.
-------------------------------------------------------------------

                     *=$E7C5
-------------------------------------------------------------------
#C KLOK aanzetten.
-------------------------------------------------------------------
E7C5 A94C            LDA #$4C       ;"JMP $E5AD"
E7C7 A2AD            LDX #$AD       ;
E7C9 A0E5            LDY #$E5       ;
E7CB 8D3001          STA $0130      ;NMI-vector.
E7CE 8E3101          STX $0131      ;
E7D1 8C3201          STY $0132      ;
E7D4 A904            LDA #4         ;sec. puls.
E7D6 8D11C3          STA $C311      ;(klok interrupt register)
E7D9 60              RTS            ;end.
-------------------------------------------------------------------
#K KLOK uitzetten.
-------------------------------------------------------------------
E7DA A900            LDA #0         ;Geen puls meer op NMI.
E7DC 8D11C3          STA $C311      ;
E7DF Ad10C3          LDA $C310      ;reset en event. interrupt.
E7E2 60              RTS            ;end.
-------------------------------------------------------------------
```

Jan Kuc ter Heide  Lage der A 12/10
050-137422  9718 BJ Groningen

Fig. 1

**ADDRESS CODES AND FUNCTIONS**

| A4 | A3 | A2 | A1 | A0 | | Function |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | Counter—Ten Thousandths of Seconds |
| 0 | 0 | 0 | 0 | 1 | 1 | Counter—Hundredths and Tenths of Seconds |
| 0 | 0 | 0 | 1 | 0 | 2 | Counter—Seconds |
| 0 | 0 | 0 | 1 | 1 | 3 | Counter—Minutes |
| 0 | 0 | 1 | 0 | 0 | 4 | Counter—Hours |
| 0 | 0 | 1 | 0 | 1 | 5 | Counter—Day of Week |
| 0 | 0 | 1 | 1 | 0 | 6 | Counter—Day of Month |
| 0 | 0 | 1 | 1 | 1 | 7 | Counter—Month |
| 0 | 1 | 0 | 0 | 0 | 8 | RAM—Ten Thousandths of Seconds (only high nibble) |
| 0 | 1 | 0 | 0 | 1 | 9 | RAM—Hundredths and Tenths of Seconds |
| 0 | 1 | 0 | 1 | 0 | 10 | RAM—Seconds |
| 0 | 1 | 0 | 1 | 1 | 11 | RAM—Minutes |
| 0 | 1 | 1 | 0 | 0 | 12 | RAM—Hours |
| 0 | 1 | 1 | 0 | 1 | 13 | RAM—Day of Week ≤ 15 (only low nibble) |
| 0 | 1 | 1 | 1 | 0 | 14 | RAM—Day of Month |
| 0 | 1 | 1 | 1 | 1 | 15 | RAM—Months |
| 1 | 0 | 0 | 0 | 0 | 16 | Interrupt Status Register |
| 1 | 0 | 0 | 0 | 1 | 17 | Interrupt Control Register |
| 1 | 0 | 0 | 1 | 0 | 18 | Counters Reset |
| 1 | 0 | 0 | 1 | 1 | 19 | RAM Reset |
| 1 | 0 | 1 | 0 | 0 | 20 | Status Bit |
| 1 | 0 | 1 | 0 | 1 | 21 | GO Command |
| 1 | 0 | 1 | 1 | 0 | 22 | STANDBY INTERRUPT |
| 1 | 1 | 1 | 1 | 1 | 31 | Test Mode |

All others unused

## Functional Description (Continued)

TABLE I. REAL TIME COUNTER FORMAT

| Counter Addressed | | Units D0 | D1 | D2 | D3 | Max BCD Code | Tens D4 | D5 | D6 | D7 | Max BCD Code |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1/10,000 of Seconds | ($00_H$) | - | - | - | - | 0 | D4 | D5 | D6 | D7 | 9 |
| Hundredths and Tenths Sec | ($01_H$) | D0 | D1 | D2 | D3 | 9 | D4 | D5 | D6 | D7 | 9 |
| Seconds | ($02_H$) | D0 | D1 | D2 | D3 | 9 | D4 | D5 | D6 | - | 5 |
| Minutes | ($03_H$) | D0 | D1 | D2 | D3 | 9 | D4 | D5 | D6 | - | 5 |
| Hours | ($04_H$) | D0 | D1 | D2 | D3 | 9 | D4 | D5 | - | - | 2 |
| Day of the Week | ($05_H$) | D0 | D1 | D2 | - | 7 | - | - | - | - | 0 |
| Day of the Month | ($06_H$) | D0 | D1 | D2 | D3 | 9 | D4 | D5 | - | - | 3 |
| Month | ($07_H$) | D0 | D1 | D2 | D3 | 9 | D4 | - | - | - | 1 |

(-) Indicates unused bits

TRI-STATE® is a registered trademark of National Semiconductor Corp.



Fig. 2

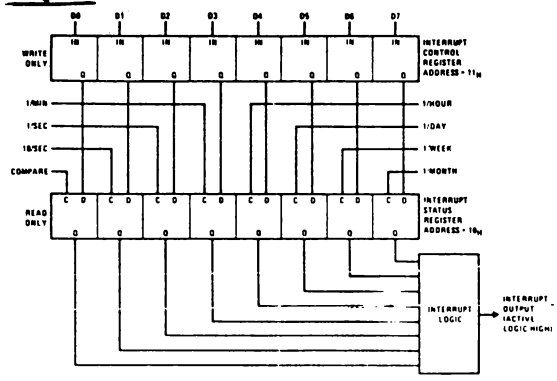



FIGURE 1. Interrupt Register Format

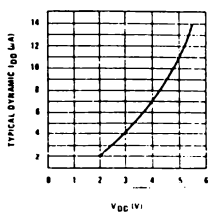



FIGURE 2. Typical Supply Current vs Supply Voltage

Fig 3

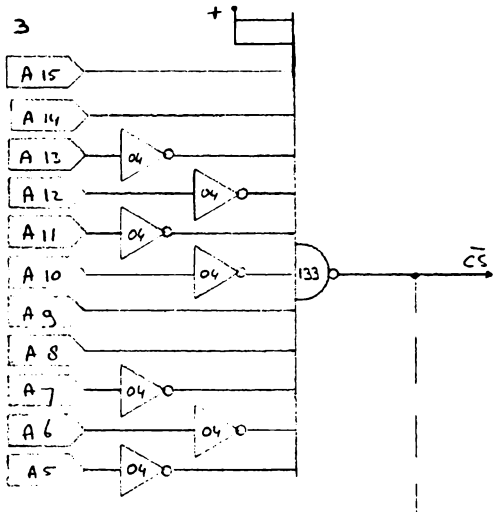
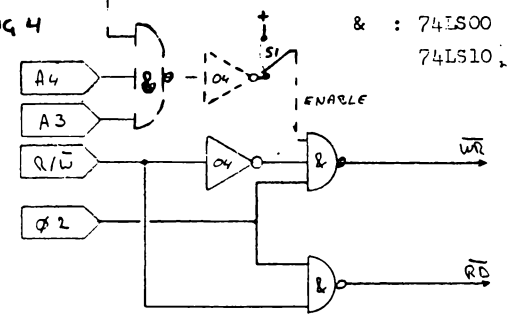
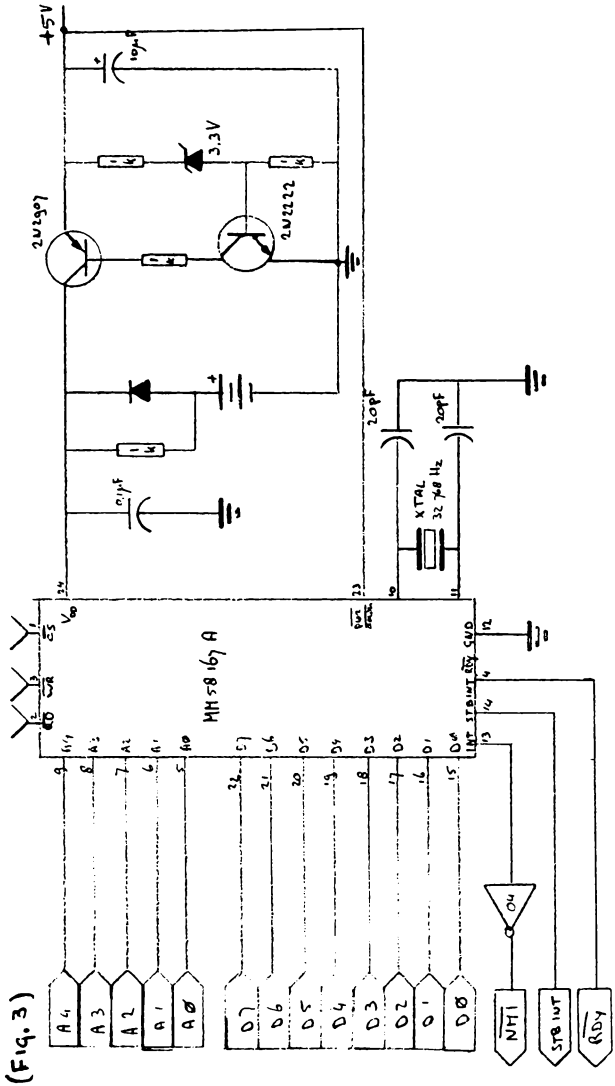



Fig 4




& : 74LS00 of
74LS10,
als het write-enable signaal wordt aangesloten.


+5V
10µF
3.3V
2N2907
2N2222
20pF
20pF
XTAL
32 768 Hz
0.1µF
MM58167A
A4
A3
A2
A1
A0
D7
D6
D5
D4
D3
D2
D1
D0
INT
STB INT
RDY

(Fig. 3)

```
LIST

 10 REM ****DATUM STRING DD$ MET KLKV15 ****
 20 DISK!"GO C40C: REM UPDATE CLOCK-BUFFER
 30 DA$="": FOR I=1TO6
 40 DA=PEEK(51157+I): DA$=DA$+CHR$(DA)
 50 NEXT I
 60 DA$=DA$+"1985"
 70 PRINT DA$,
 110 REM **** TYD-STRING TI$ MET KLKV15 ****
 120 DISK!"GO C40C: REM UPDATE CLOCK-BUFFER
 130 TI$="": FORI=1TO8
 140 TI=PEEK(51147+I): TI$=TI$+CHR$(TI)
 150 NEXT I
 160 PRINT TI$
 170 PRINT
 190 REM ****************************
 200 PRINT" JUMPTABLE OSI-KLOK KLKV15"
 210 PRINT" $C400 DEMO"
 220 PRINT" $C403 DATTYD ;DATUM EN TYD NAAR SCHERM"
 230 PRINT" $C406 DASCR  ;ALLEEN DATUM"
 240 PRINT" $C409 TISCR  ;ALLEEN TYD"
 250 PRINT" $C40C UPDATE ;NIEUWE TYD NAAR BUFFER"
 260 PRINT" $C40F SETKL  ;GELIJKZETTEN, LET OP S3 !!"
 270 PRINT" $C412 PWRUP  ;STARTEN KLOK (OOK IN SETKL)"

Ok
```

OSI POEL

SEPTEMBER 1985

STAPPENMOTORSTURING

Het bouwen van plotters en robots is op dit moment bijzonder populair, hiervoor worden meestal stappenmotoren gebruikt, de hier voorgestelde schakeling voor het sturen van stappenmotoren zal dan ook voor veel mensen (en motoren) een stap in de goede richting zijn. De meeste stapmotoren hebben twee spoelparen dus totaal vier spoelen deze worden veelal aangeduidt als spoel A,B,C en D , door nu de spoelen in de juiste volgorde te bekrachtigen zal de motor gaan draaien, normaal worden er steeds twee spoelen gelijktijdig bekrachtigd en wel in deze volgorde AB,BC,CD,DA de schakeling dient dan volgens fig.C bedraad te worden. We kunnen het aantal stappen ook verdubbelen, de volgorde word dan A,AB,B,BC,C,CD,D,DA (zie fig.A), omdat er tussentijds maar 1 spoel bekachtigd word zal het duidelijk zijn dat het koppel van de motor minder zal zijn. We kunnen de schakeling aansluiten op een userpoort van de computer hiervan gebruiken we twee uitgangen per motor, wat betreft het aansluiten van de schakeling hebben we twee mogelijkheden het teller i.c. aan de ingang van fig. A heeft twee klokingangen, afhankelijk van aan welke ingang we pulsen toevoeren draait de motor links of rechtsom, de niet gebruikte ingang dienen we hoog te houden. Het teller i.c. volgens fig. B heeft 1 klokingang, het links of rechtsom draaien word bereikt door pen 1 hoog of laag te maken. Het zal duidelijk zijn dat het type transistoren Tx afhankelijk is van de toegepaste motor, eveneens is Rx afhankelijk van de voedingsspanning en de stroom door de motor, het toevoegen van Cx geeft een verbetering van het koppel van de motor.
Voor mensen die meer over de werking van stappenmotoren willen weten kunnen dit vinden in Elektuur nr.258 en RB mei '83 Voor de schakeling vlg. fig. B c.q. fig.C is een print verkrijgbaar, op deze print kunnen twee van deze schakelingen plus een transistor schakeling voor het sturen van b.v. een penlift voor een plotter.

Ko van Ekeren

stappenmotor
Rx
Cx
Tx
1K
+5V
ø8
ø4
A
B
C
D
y/4ls138
y/4ls193
up
down
4
5
fig.A
dubbel aantal stappen
A
AB
B
BC
C
CD
D
DA
fig.C
normaal aantal stappen
AB
BC
CD
DA
fig.B
y/4ls164
up/down
richting
1
2
12
13
14
stappenmotorsturing
Ko v. Ekeren 2.85

# STAPPENMOTORSTURING

Als aanvulling op het voorgaande artikel over stappenmotorsturing is hier nogmaals fig.C afgedrukt. Omdat de aansluiting van i.c. 74LS00 op i.c. 74LS138 bij normaal aantal stappen anders moet zijn dan bij dubbel aantal stappen (fig. A). Dit kwam in het voorgaande schema niet tot uiting In het voorgaande artikel werd gesproken over stappenmotoren met vier spoelen de z.g.n. unipolaire motoren er zijn echter ook motoren met twee spoelen deze worden bipolaire motoren genoemd. Hoe we deze motoren kunnen aansturen laat fig. D zien.

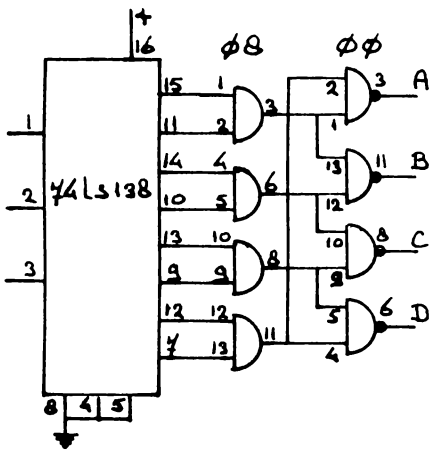



fig. C

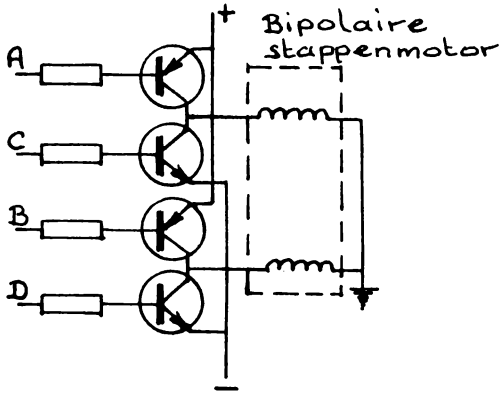



fig. D

TE64: een eenvoudige tekst-editor voor OSI.

De naam betekent: tekst-editor met 64 karakters per regel.
Deze tekst-editor bevindt zich op diskette op de screens 2 t/m 17. Screen 18 bevat een korte toelichting op het programma en screen 19 functionneert als klad-screen.
Ofschoon bij het laden van en het werken met TE64 geen foutmelding optreedt, zijn de screens 4 en 5 toch gevuld met de standaard-foutmeldingen.Deze screens zouden gevuld kunnen worden met een deel van het programma, waardoor dan screen 16 de functie van screen 18 en 17 die van 19 zou krijgen. Maar op de disk zijn 60 screens beschikbaar voor tekst. Dat zijn 60*16 regels van 64 karakters, zodat het wat 'overdoing' lijkt al dat werk te verrichten om 32 regels per diskette meer beschikbaar te hebben. De auteur van het programma zal u er echter niet van afhouden.
Hieronder volgt een beschrijving van het programma.
De tekst-editor kent twee toestanden (modes):
de 'command mode' en de 'edit mode'.
In de cmd mode
- kan worden gesprongen naar de 'edit mode';
- kan een screen op het scherm gebracht worden om gewijzigd of gevuld te worden;
- een screen, dat ingetikt of gewijzigd is en op het scherm te zien is ( het bevindt zich dan in video-ram en in het klad-screen ) kan naar een te kiezen screen geschreven worden;
- een of een aantal opeenvolgende screens kan naar de printer gestuurd worden, waarbij het aantal regels per pagina tevoren wordt opgegeven. Het aantal posities per regel is steeds 64;
- de bovenste of de onderste acht regels van het tekstveld (d.i. wat op het scherm te zien is) kunnen naar een tekst-buffer van acht regels gecopieerd worden en daarna weer naar de onderste of de bovenste helft van het tekstveld worden teruggehaald;
- en tot slot kan het programma worden verlaten.

Het is niet mogelijk de screens 2 tot en met 17 te lezen of naar de screens 2 tot en met 18 te schrijven. Dat laatste is gedaan om te voorkomen, dat een deel van het programma overschreven wordt. Het is dus ook niet nodig de programma-screens te kunnen lezen. Overigens kunt u altijd de EDITOR te hulp roepen. Denk er dan wel aan eerst een back up te maken van de screens waar u mee wilt gaan spelen.
In de 'edit mode' kunnen de 16 regels tussen de kaderlijnen worden gevuld met tekst. Boven het tekstveld staat in deze toestand aangegeven, welke speciale edit-functies ter beschikking staan. (De DELETE-toets voor wissend 'back space'n is ook beschikbaar.)
Dat zijn:
Ctrl-H: hiermee wordt het tekstveld naar het kladscreen geschreven en screen 18 met enige summiere aanwijzingen op het scherm gebracht, drukken op enige toets brengt het tekstveld terug;
Ctrl-S: het tekstveld wordt schoon gemaakt;
Ctrl-V: twee regels worden verwisseld, te beginnen met de positie van de cursor;
Ctrl-U: de cursor gaat naar dezelfde positie een regel hoger;
Ctrl-J: de cursor gaat naar dezelfde positie een regel lager;
Ctrl-L: de cursor gaat een positie naar links;
Ctrl-R: de cursor gaat een positie naar rechts;
Bij deze 4 ctrl-commando's wordt het karakter op de verlaten positie terug gezet.
Ctrl-I: er wordt op de plaats van de cursor plaats gemaakt voor een karakter;

Ctrl-T: er worden vanaf de plaats van de cursor 64 spaties tussen gevoegd;
Ctrl-D: het karakter onder de cursor wordt gewist; het deel van de regel rechts van de cursor gaat een positie naar links;
Ctrl-W: er wordt vanaf de cursor-positie een regel weggehaald;
Na het aanroepen van de 'edit mode' bevindt de cursor zich links boven in het tekstveld. Om met kleine letters te kunnen werken moet de SHIFT LOCK in de toestand OFF gebracht worden.
De 'edit mode' wordt verlaten door op de toets ESC te drukken. Wel moet dan de SHIFT LOCK in de toestand ON zijn !!!
Het woord PRINIT initialiseert de printer, d.w.z.
- de printer wordt in de default toestand gebracht;
- de papierlengte wordt op 12 inch ingesteld;
- er wordt een kantlijn gegeven van 8 posities;
- de printer zal unidirectional printen;
- er wordt geprint in ELITE-SIZE.

De codes, die in het woord worden gebruikt, zijn die voor de GEMINI-10X printer.

Het programma zelf.
In de listing van de screens staat maar weinig toelichting. Daarom hier enige aanvullende informatie over de werking van het programma. De gedefinieerde woorden worden in de volgorde,waarin ze in de screens optreden,bekeken. De namen van de woorden staan in hoofdletters.

CHECK kijkt of de toets,die is ingedrukt,al of niet van een cijfer is;
GETAL maakt van twee cijfers een getal en zet het op de stack. Is het geen getal van 2 cijfers, dan wordt het getal 19 op de stack gezet;
START is het startadres van het tekstveld;
EIND is het laatste (hoogste) adres van het tekstveld;
KARAKTER en KAR zijn hulpvariabelen bij het manipuleren met de de cursor:
LP bewaart het adres, waar de cursor zich bevindt:
CURPOS berekent het byteadres, waar de cursor staat en bergt dat adres op in de variabele LP;
<CMOVE is een woord, dat niet is opgenomen in de vocabulary van de gebruikte FORTH versie. Het copieert een aantal bytes van het ene startadres naar het andere startadres, en wordt gebruikt voor elkaar gedeeltelijk overlappende stukken geheugen
RECHTS verplaatst de cursor een positie naar rechts;
LINKS verplaatst de cursor een positie naar links;
KAR>POS zet het ASCII-teken, dat in de variabele KAR is opgeborgen, op de positie van de cursor;
POS>KAR bergt het ASCII-teken, dat op de cursorpositie staat, op in de variable KAR:
VOLGENDE brengt de cursor naar de volgende positie van de regel, waarbij niet naar een volgende regel wordt gesprongen, maar bij het verlaten van de regel verschijnt de prompt weer vooraan de regel ('wrap around');
VORIGE brengt de cursor naar de vorige positie van de regel maar verlaat de regel niet ('wrap around');
OMHOOG verlaagt het regelnummer, waarin de cursor zich bevindt;
OMLAAG verhoogt het regelnummer, waarin de cursor zich bevindt:
BOVEN brengt de cursor naar dezelfde positie op een regel hoger en keert eventueel onder in het tekstveld terug;
ONDER brengt de cursor naar dezelfde positie een regel lager en keert eventueel boven in het tekstveld terug;

C!PROMPT plaatst de prompt op de plaats waar de cursor zich bevindt;
^R (ctrl-R) verplaatst de cursor naar rechts en herplaats het karakter, dat zich onder de prompt bevond (zie ook RECHTS);
^L (ctrl-L) als ^R maar nu naar links;
^U (ctrl-U) als ^R maar nu naar boven;
^J (ctrl-J) als ^R maar nu naar beneden;
^I (ctrl-I) maakt ruimte voor een ASCII-karakter. Het karakter in de meest rechtse positie van de regel gaat veloren;
^D (ctrl-D) wist het teken onder de cursor en verplaats de rest van de regel een positie naar links;
^S (ctrl-S) vult het scherm met 'blanks';
^W (ctrl-W) laat 64 posities verdijnen geteld vanaf de positie van de cursor; de tekst daarachter wordt een regel omhoog geschoven; het laatste deel van het tekstveld wordt gevuld met 'blanks';
^T (ctrl-T) vanaf de positie van de cursor wordt de tekst een regel naar beneden geschoven en 64 posities vanaf de cursor worden gevuld met 'blanks';
^V (ctrl-V) vanaf de positie van de cursor verwisselen twee regels van plaats;
^H (ctrl-H) schrijft het tekstveld naar het kladscreen en brengt screen 18 (met enige aanwijzingen betreffende TE64) naar het tekstveld; een toetsindruk brengt het kladscreen weer naar het tekstveld terug;
^M (ctrl-M) is hetzelfde als <RETURN> of <ENTER>;
CONTROL contoleert of het screen-nummer hoger is dan 79;
>DISK schrijft het tekstveld naar het screen, waarvan het nummer op de stack staat;
DISK> brengt het screen, waarvan het nummer op de stack staat naar het tekstveld;
SCHOON en WEG wissen het cmd-menu, resp. het edit-menu;
EDIT-MENU print het edit-menu boven het tekstveld;
CMD-MENU print het cmd-menu onder het tekstveld;
SCREEN toont rechts in het veld van het edit-menu het nummer van het screen, dat naar het tekstveld is gebracht;
^KEY onderzoekt of een geldig controle-karakter is ingetoetst en voert in dat geval de desbetreffende opdracht uit;
->DISK verzorgt het geheel van handelingen bij het overbrengen van een screen naar het tekstveld;
DISK-> verzorgt het geheel van handelingen bij het overbrengen van het tekstveld naar diskette;
PRINIT initialiseert de printer;r
DISK>PRINTER brengt het opgegeven stel screens naar de printer; eerst wordt gevraagd hoeveel regels per pagina gebruikt moeten worden;
CHR-CTRL kijkt of een controle-karakter is ingetoetst; zo niet, dan wordt het karakter in het tekstveld geschreven;
STREEP brengt een afscheiding aan tussen menuveld en tekstveld;
TEKST verzorgt het geheel van handelingen in de 'edit mode';
BUFFEREN verzorgt het geheel van handelingen bij het overbrengen van acht regels tekst naar een buffer of terug;
TE64 het uiteindelijke programma-woord.

Schipluiden, voorjaar 1985 J.M.A. Hermans

```
SCR # 2
  0 ( INITIALISATIE EN PRINTER )
  1 26 EMIT ." TE64 (Texteditor-64 voor OSI)" CR  2 .
  2 548 CONSTANT ERGL ( EERSTE REGEL) 517 CONSTANT PRT
  3 549 CONSTANT LRGL ( LAATSTE REGEL)
  4 553 CONSTANT XCURSOR  554 CONSTANT YCURSOR
  5 : PRINIT 1 PRT C!
  6         64 27 EMIT EMIT   ( PRINTER IN DEFAULT )
  7         8 77 27 EMIT EMIT EMIT  ( LINKER KANTLIJNBREEDTE )
  8         1 85 27 EMIT EMIT EMIT  ( PRINT IN EEN RICHTING )
  9         12 0 67 27 EMIT EMIT EMIT EMIT  ( 12 INCH PAPIER )
 10         CR 0 PRT C! :
 11 --> TE64 is een eenvoudige text editor voor (M)OSI
 12           met HW-monitor geschreven door
 13           John M.A. Hermans
 14           Schioluiden  01738-8703
 15           voorjaar 1985

SCR # 3
  0 ( TE64 = TEXTEDITOR-64 1 )
  1 3 .
  2 : CHECK DUP 48 < SWAP DUP 57 > ROT OR :
  3 : GETAL 127 BASE !
  4         KEY CHECK
  5         IF DROP 19
  6         ELSE DUP EMIT KEY CHECK
  7           IF DROP DROP 19
  8           ELSE DUP EMIT 48 - SWAP 48 - 10 * +
  9           THEN
 10         THEN
 11         DECIMAL
 12         :
 13 6 LOAD
 14 CR ." TIK IN 'TE64 <RETURN>' OF 'TE64 <ENTER>'" CR CR
 15 :S

SCR # 18
  0  TE64  gebruiksaanwijzing
  1     - Er zijn twee 'MODE's  de 'command mode' en de 'edit mode'
  2       In de cmd mode staat onder het tekstveld het cmd-menu:
  3     - De edit mode wordt verlaten met het indrukken van de <ESC>-
  4       toets d.i. de toets: <-- .terwijl het lichtje rechts op het
  5       toetsenbord oplicht, dit is <SHIFT LOCK> ON.
  6     - Bij het verlaten van de edit mode  wordt  het  tekstscherm
  7       tussen de stippellijnen) automatisch  naar  screen  19  ge-
  8       schreven !!!
  9     - De screens 2 t/m 18  zijn  niet  toegankelijk voor het weg-
 10       schrijven van tekst, omdat daar het programma staat.
 11     - De tekst van dit scherm (HELP) staat op screen 18.
 12     - Met 'BUFFER' kunnen de bovenste 8 of de onderste 8 regels
 13       van het tekstscherm worden opgeborgen of teruggehaald.
 14
 15     VOOR VERVOLG DRUK OP EEN TOETS
```

```
SCR # 6
  0 ( TE64 = TEXTEDITOR-64 2)
  1 6 . 0 VARIABLE SCH 0 VARIABLE FF
  2 -12288 CONSTANT START 0 VARIABLE KARAKTER
  3 -10944 CONSTANT EIND 0 VARIABLE LP 0 VARIABLE KAR
  4 : CURPOS XCURSOR C@ YCURSOR C@ 64 * + START + LP ! :
  5 : <CMOVE ( OUDE STARTADR., NIEUWE IDEM, AANTAL BYTES --- )
  6         BEGIN -DUP
  7         WHILE 1 - >R OVER I + C@ OVER I + C! R>
  8         REPEAT DROP DROP :
  9 : RECHTS XCURSOR C@ 1+ DUP 64 =
 10         IF DROP 0 XCURSOR C! ELSE XCURSOR C! THEN :
 11 : LINKS XCURSOR C@ DUP 0=
 12         IF DROP 63 XCURSOR C!
 13         ELSE 1 - XCURSOR C!
 14         THEN :
 15 -->


SCR # 7
  0 ( TEXTEDITOR-64 3 ) 7 .
  1 : KAR>POS ( KAR NAAR CURSOR-POSITIE )
  2         KAR @ LP @ C! :
  3 : POS>KAR ( POSITIE NAAR KAR )
  4         LP @ C@ KAR ! :
  5 : VOLGENDE ( NAAR VOLGENDE POSITIE )
  6         RECHTS CURPOS :
  7 : VORIGE ( GA EEN POSITIE TERUG )
  8         LINKS CURPOS :
  9 : OMHOOG YCURSOR C@ DUP 5 =
 10         IF DROP 20 ELSE 1 - THEN YCURSOR C! :
 11 : OMLAAG YCURSOR C@ DUP 20 =
 12         IF DROP 5 ELSE 1+ THEN YCURSOR C! :
 13 : BOVEN ( NAAR POSITIE OP REGEL HOGER )
 14         OMHOOG CURPOS :
 15 : C!PROMPT ( PLAATS DE PROMPT ) 95 LP @ C! : -->


SCR # 8
  0 ( TEXTEDITOR-64 4 ) 8 .
  1 : ONDER ( NAAR POSITIE OP REGEL ERONDER ) OMLAAG CURPOS :
  2 : ^R KAR>POS VOLGENDE POS>KAR C!PROMPT :   ( NAAR RECHTS )
  3 : ^L KAR>POS VORIGE   POS>KAR C!PROMPT :   ( NAAR LINKS )
  4 : ^U KAR>POS BOVEN    POS>KAR C!PROMPT :   ( NAAR BOVEN )
  5 : ^J KAR>POS ONDER    POS>KAR C!PROMPT :   ( NAAR BENEDEN )
  6 : ^I CURPOS LP @ 64 MOD 63 =               ( MAAK PLAATS )
  7         IF
  8         ELSE LP @ DUP 1+ EIND OVER -
  9           64 MOD <CMOVE KAR @ LP @ 1+ C!
 10           32 KAR ! C!PROMPT
 11         THEN :
 12 : ^S ERGL C@ LRGL C@ 5 ERGL C! 21 LRGL C!   ( TEKSTVELD SCHOON)
 13         3 EMIT LRGL C! ERGL C! :
 14 : SCREEN SCH @  10 /MOD 48 + START 59 + C! 48 + START 60 + C!
 15    82 START 57 + C! 67 START 56 + C! 83 START 55 + C! : -->
```

```
SCR # 9
  0 ( TEXTEDITOR-64  5 ) 9 .
  1 ( ^W = HAAL 64 KAR. WEG: ^T = VOEG BLANCO REGEL TUSSEN )
  2 ( ^D = DELETE KAR.: ^V = WISSEL REGELS )
  3 : ^W    CURPOS KAR>POS LP @ DUP DUP 64 + ROT ROT
  4         EIND SWAP - CMOVE POS>KAR C!PROMPT :
  5 : ^T    CURPOS KAR>POS
  6         LP @ DUP 64 + OVER EIND SWAP - <CMOVE
  7         LP @ 64 32 FILL EIND 64 32 FILL
  8         POS>KAR C!PROMPT :
  9 : ^D CURPOS LP @ DUP 1+ SWAP EIND OVER -
 10         64 MOD DUP 0= IF DROP 63 THEN
 11     CMOVE POS>KAR C!PROMPT 32 YCURSOR C@ 64 * START + 63 + C! :
 12 : ^V CURPOS KAR>POS LP @ PAD 64 CMOVE
 13         LP @ DUP 64 + SWAP 64 CMOVE
 14         PAD LP @ 64 + 64 CMOVE POS>KAR C!PROMPT : -->
 15


SCR # 10
  0 ( TEXTEDITOR-64 6 ) 10 .
  1 : CONTROL DUP FF @ < SWAP DUP 79 > ROT OR :
  2 : >DISK 19 FF ! CONTROL ( SCR.NR --- )
  3         IF DROP
  4         ELSE EIND 1024 - SWAP DUP SCH !
  5           BUFFER 1024 CMOVE UPDATE ( FLUSH )
  6         THEN :
  7 : EDIT-MENU 0 ERGL C! 0 XCURSOR C! 0 YCURSOR C!
  8         ." CTRL-toetsen          ^U= OMHOOG      "
  9         ." ^I= RUIMTE MAKEN" CR
 10         ." ^H= HELP              ^J= OMLAAG      "
 11         ." ^T= VOEG REGEL IN" CR
 12         ." ^S= SCHERM SCHOON     ^L= LINKS       "
 13         ." ^D= WIS KARAKTER" CR
 14         ." ^V= VERWISSEL REGELS ^R= RECHTS      "
 15         ." ^W= WIS REGEL" CR : -->


SCR # 11
  0 ( TEXTEDITOR-64  7 ) 11 .
  1 : SCHOON 23 ERGL C! 3 EMIT :
  2 14202 VARIABLE BLOK
  3 : DISK> 18 FF ! CONTROL  ( SCR.NR. --- )
  4         IF DROP
  5         ELSE DUP SCH ! BLOCK DUP BLOK !
  6           EIND 1024 - 1024 CMOVE
  7         THEN :
  8 : ^H 19 >DISK 18 DISK> KEY DROP 19 DISK> :
  9 : ^M KAR>POS 0 XCURSOR C! YCURSOR C@ 1+ DUP 20 >
 10         IF DROP
 11         ELSE YCURSOR C!
 12         THEN
 13         CURPOS POS>KAR C!PROMPT :
 14 : WEG ERGL C@ LRGL C@ 0 ERGL C! 3 LRGL C!
 15         3 EMIT ( VENSTER SCHOON) LRGL C! ERGL C! SCREEN : -->
```

```
SCR # 12
  0 ( TEXTEDITOR-64  8 ) 12 .
  1 : ^KEY  ( IS HET EEN CTRL-TOETS ) DUP 18 =
  2         IF ^R  ELSE DUP 12 =
  3         IF ^L  ELSE DUP 21 =
  4         IF ^U  ELSE DUP 10 =
  5         IF ^J  ELSE DUP  4 =
  6         IF ^D  ELSE DUP  9 =
  7         IF ^I  ELSE DUP 13 =
  8         IF ^M  ELSE DUP 20 =
  9         IF ^T  ELSE DUP 22 =
 10         IF ^V  ELSE DUP  8 =
 11         IF ^H  ELSE DUP 19 =
 12         IF ^S  ELSE DUP 23 =
 13         IF ^W  THEN THEN THEN THEN THEN
 14         THEN THEN THEN THEN THEN THEN THEN DROP :
 15 -->


SCR # 13
  0 ( TEXTEDITOR-64 9 ) 13 .
  1 : ->DISK ( TEKSTVELD NAAR DISK )
  2         SCHOON ." NAAR SCREEN NR.: " GETAL
  3         >DISK 3 EMIT 0 ERGL C! SCREEN :
  4 : DISK-> ( VAN DISK NAAR TEKSTVELD )
  5         SCHOON ." VAN SCREEN NR.: " GETAL
  6         DISK> 3 EMIT 0 ERGL C! SCREEN :
  7 0 VARIABLE SCHERM 0 VARIABLE PAGINA 0 VARIABLE R/P
  8 : CMD-MENU SCHOON CR
  9 12 SPACES  ." Screen editor voor OSI" CR
 10 15 SPACES ." 1. SAVE" 10 SPACES ." 4. TEXT" CR
 11 15 SPACES ." 2. LIST" 10 SPACES ." 5. BUFFER" CR
 12 15 SPACES ." 3. PRINT" 9 SPACES ." 6. QUIT" CR CR :
 13 : REGELS ." AANTAL REGELS PER PAGINA: "
 14         GETAL CR CR R/P ! :
 15 -->


SCR # 14
  0 ( TEXTEDITOR-64 10 ) 14 .
  1
  2 : DISK>PRINTER PRINIT SCHOON ( DISK NAAR PRINTER )
  3         REGELS
  4         ." NAAR PRINTER VANAF SCREEN " GETAL CR CR
  5         ." TOT EN MET SCREEN " GETAL 1+ SWAP
  6         1 PRT C! 0 PAGINA !
  7         DO I BLOCK I SCH ! SCREEN
  8             DUP 1024 + SWAP
  9             DO CR I 64 -TRAILING TYPE
 10               PAGINA @ 1 + DUP PAGINA ! R/P @ =
 11               IF 0 PAGINA ! 12 EMIT
 12               THEN
 13             64 +LOOP
 14         LOOP CR 0 PRT C! :
 15 -->
```

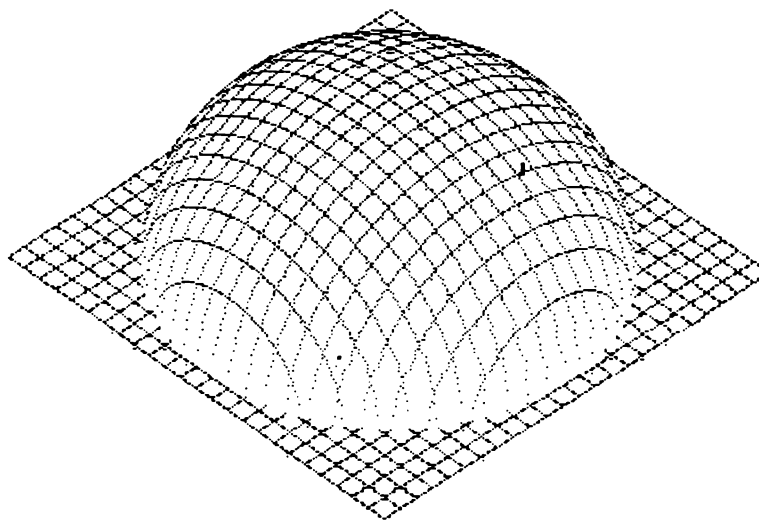
```
SCR # 15
  0 ( TEXTEDITOR-64  11 ) 15 .
  1 : CHR-CTRL ( WAT VOOR EEN KARAKTER? )
  2         KEY DUP KARAKTER ! DUP 32 <
  3         IF ^KEY
  4         ELSE LP @ 1+ C@ KAR ! EMIT CURPOS
  5           LP @ 64 MOD ABS 8 = IF 7 EMIT THEN
  6         THEN :
  7 : STREEP 8 0 DO ." --------+" LOOP :
  8 : TEKST CR EDIT-MENU STREEP
  9         BLOK @ C@ KAR ! 22 YCURSOR C! 0 XCURSOR C! STREEP
 10         ERGL C@ SCHOON ERGL C!
 11         0 XCURSOR C! 5 YCURSOR C! CURPOS C!PROMPT
 12         BEGIN
 13           CURPOS CHR-CTRL
 14           KARAKTER @ 27 = IF KAR>POS 1 ELSE 0 THEN
 15         UNTIL WEG 19 >DISK : -->


SCR # 16
  0 ( TEXTEDITOR-64 12 ) 16 .
  1 0 VARIABLE BUNKER 514 ALLOT BUNKER 512 32 FILL
  2 : BUFFEREN  23 ERGL C! 3 EMIT
  3 5 SPACES ." 1. EERSTE  8 SCHERMREGELS NAAR BUFFER " CR
  4 5 SPACES ." 2. LAATSTE 8 SCHERMREGELS NAAR BUFFER " CR
  5 5 SPACES ." 3. BUFFER NAAR EERSTE  8 SCHERMREGELS" CR
  6 5 SPACES ." 4. BUFFER NAAR LAATSTE 8 SCHERMREGELS" CR
  7 CR ."     UW KEUZE 1/2/3/4 " KEY DUP 49 =
  8         IF EIND 1024 - BUNKER 512 CMOVE ELSE DUP 50 =
  9         IF EIND 512  - BUNKER 512 CMOVE ELSE DUP 51 =
 10         IF BUNKER EIND 1024 - 512 CMOVE ELSE DUP 52 =
 11         IF BUNKER EIND 512 - 512 CMOVE
 12         THEN THEN THEN THEN DROP :
 13 -->
 14
 15


SCR # 17
  0 ( TEXTEDITOR-64 13 ) 17 .
  1 0 VARIABLE VLAG 0 VARIABLE FLAG CR : JA 1 FLAG ! :
  2 : TE64 26 EMIT          ( SCREEN EDITOR VOOR OSI )
  3         BEGIN 0 VLAG !
  4           CMD-MENU 12 SPACES ." UW KEUZE 1/2/3/4/5/6 "
  5           BEGIN 0 FLAG !
  6             KEY 48 - DUP 1 =
  7             IF JA ->DISK ELSE DUP 2 =
  8              IF JA DISK-> ELSE DUP 3 =
  9               IF JA DISK>PRINTER ELSE DUP 4 =
 10                IF JA TEKST ELSE DUP 5 =
 11                 IF JA BUFFEREN ELSE DUP 6 =
 12                  IF JA 1 VLAG ! 0 YCURSOR C! 26 EMIT
 13             THEN THEN THEN THEN THEN THEN DROP FLAG @
 14           UNTIL FLUSH 0 ERGL C! 0 XCURSOR C! 0 YCURSOR C! VLAG @
 15         UNTIL : CR  :S
```

```
10 REM RUIMTELIJKE GRAFIEK NAAR DE PRINTER
20 REM EEN PROGRAMMA VAN JAMES I. BARTHOLOMEW
30 REM COMPUTING TODAY FEBRUARI 1984
40 REM VOOR OSI C1P MET GEMINI 10X PRINTER BEWERKT
50 REM DOOR JOHN HERMANS
60 POKE132,30:POKE133,78:REM RESERVEREN VAN GRAFIEK-BUFFER
70 PRINT!(28):TRAP5000
80 PRINT"IK BEN BEZIG":PRINT:PP=.7071:AD=341
100 SB=20000:OG=SB+6308:EB=SB+12616:ZE=.7071:DR=341:ZS=6138
110 GOSUB2000
120 C=10
130 FORY=12TO-12.05STEP-.1
135 PRINTY;
140 IFC=10THENS=.1:C=0:GOTO160
150 S=1
160 FORX=-12TO12.05STEPS
165 R=SQR(X*X+Y*Y)
170 IFR=0THENZ=10:GOSUB600:GOTO310
275 Z=(ABS(X)+ABS(Y))/2:IFZ>6THENZ=0
280 IFZ<0THENZ=0
295 Z=(ABS(X)+ABS(Y))/2:IFZ>6THENZ=0
300 IFZ<0THENZ=0
305 GOSUB600
310 NEXTX
320 C=C+1
330 NEXTY
350 GOSUB400
360 END
395 REM UITVOER NAAR PRINTER
400 DISK!"IO ,03":CC=20000
403 REM PRINTER NAAR DEFAULT
405 PRINTCHR$(27):CHR$(64):PRINT:PRINT:PRINT
408 REM LINE FEED 8/72 INCHE
410 PRINTCHR$(27):CHR$(65):CHR$(8):PRINT
415 FORI=0TO36
418 REM PRINT NORMAL DENSITY GRAPHICS
420 PRINTTAB(10):CHR$(27):CHR$(75):CHR$(85):CHR$(1);
425 FORJ=0TO340
430 BB=CC+I*341+J
435 AA=PEEK(BB):IFAA=12THENAA=8
440 PRINTCHR$(AA);
450 NEXTJ
460 PRINT
470 NEXTI
480 PRINTCHR$(27):CHR$(64)
490 DISK!"IO .02"
500 RETURN
600 X1=(X+Y)*PP:Y1=(Y-X)*PP
610 ZD=Y1+Z:IZ=INT(ZD)
620 RZ=ZD-IZ:EP=INT(8*RZ):A=2^EP
630 XD=INT(10*X1):YD=-AD*IZ
640 B=OG+XD+YD
650 IFB>EBTHENRETURN
660 IFB<SBTHENB=OG+XD-ZS:A=0
662 GOTO680
665 FORM=OG+XD+ZSTOB+ADSTEP-AD
670 POKEM,0
675 NEXTM
```

```
680 T=PEEK(B):U=(T)OR(A):V=256-A:W=(U)AND(V)
685 W=WORPEEK(B):POKEB,W
690 RETURN
700 IFY>-5ANDY<0THENGOSUB800
710 RETURN
800 FORZ=3TO8STEP.1
805 A=1
810 GOSUB205
820 NEXTZ
830 RETURN
995 REM GRAFIEK NAAR DISK
1000 DISK!"SA 33.1=4E20/8"
1010 DISK!"SA 34.1=5620/8"
1020 DISK!"SA 35.1=5E20/8"
1030 DISK!"SA 36.1=6620/8"
1040 DISK!"SA 37.1=6E20/8"
1050 DISK!"SA 38.1=7620/8"
1060 DISK!"SA 39.1=7E20/2"
1070 RETURN
1995 REM GRAFIEK VAN DISK NAAR HOGE RAM
2000 DISK!"CA 4E20=33.1"
2010 DISK!"CA 5620=34.1"
2020 DISK!"CA 5E20=35,1"
2030 DISK!"CA 6620=36.1"
2040 DISK!"CA 6E20=37,1"
2050 DISK!"CA 7620=38.1"
2060 DISK!"CA 7E20=39,1"
2070 RETURN
```



```
SCR # 10
  0 ( HGR NAAR PRINTER )
  1 26 EMIT ." HGR.PRINT" CR
  2 : J R> R> R> R SWAP >R SWAP >R SWAP >R :
  3 HEX 4800 CONSTANT HGR-START DECIMAL 0 VARIABLE START
  4 61441 CONSTANT ACIAB 61440 CONSTANT ACIAA
  5 : HGR.PRINT 1 517 C! 27 EMIT 64 EMIT
  6       HGR-START START !
  7      27 EMIT 65 EMIT 7 EMIT CR ( LF 8/72 INCH )
  8      37 0 DO 10 SPACES 27 EMIT 75 EMIT 85 EMIT 1 EMIT
  9                          ( PRINT 341 BYTES )
 10        341 0 DO START @ J 341 * + I + C@
 11      ACIAB C! BEGIN ACIAA C@ 2 = UNTIL
 12              LOOP CR START @ 1 - START !
 13           LOOP CR CR
 14      27 EMIT 64 EMIT CR 0 517 C! :
 15 -->

SCR # 11
  0 ( HGR NAAR PRINTER 2 )
  1 ." HGR.PRINT 2" CR
  2 0 VARIABLE HULP
  3 HEX
  4 : SCHOON HGR-START DUP 3800 + SWAP
  5        DO HULP @ 1 + DUP HULP !
  6           11 = IF BF 0 HULP ! ELSE 0 THEN
  7           40 + I C!
  8        LOOP :
  9 : BLANK HGR-START 3800 0 FILL ;
 10
 11 DECIMAL
 12 -->
 13
 14
 15

SCR # 12
  0 ( HGR NAAR PRINTER 3 )
  1 ." HGR.PRINT 3" CR
  2 0 VARIABLE X 0 VARIABLE Y 0 VARIABLE Y1 0 VARIABLE Y2
  3 : PUNT?
  4       Y2 @ DUP 7 =
  5         IF 128 ELSE DUP 6 =
  6           IF 64 ELSE DUP 5 =
  7             IF 32 ELSE DUP 4 =
  8               IF 16 ELSE DUP 3 =
  9                 IF 8 ELSE DUP 2 =
 10                   IF 4 ELSE DUP 1 =
 11                     IF 2 ELSE 1
 12        THEN THEN THEN THEN THEN THEN THEN SWAP DROP :
 13 -->
 14
 15
```

```
SCR # 13
  0 ( HGR NAAR PRINTER 3 )
  1 ." HGR.PRINT 4" CR
  2 : PUNT Y @ 7 / Y1 !
  3        Y @ Y1 @ 7 * - Y2 !
  4        X @ 36 Y1 @ - 340 * +
  5        HGR-START + DUP C@ PUNT? OR SWAP C!
  6 ;
  7 : LIJN 300 0 DO I X ! I Y !
  8              PUNT X @ .
  9             LOOP ;
 10 ;S
 11
 12
 13
 14
 15
```

Het volgende programma GSOSRT is voor M-DMS geschreven
om zeer grote file's te kunnen sorteren. De file
wordt namenlijk niet geheel in het geheugen gezet,
maar alleen het veld waarop gesorteerd wordt. Er
wordt een pointertabel opgebouwd en aan de hand daarvan
worden de records op de disk verplaatst. Het is verstandig
om deze GSOSRT versie pas in te zetten als met de
gewone versie een OM error optreed. Het sorteren
op disk is namenlijk veel tijdrovender.

```
 1000 TRAP1620
 1010 REM  SORTEER ROUTINE VOOR M-DMS GROTE VERSIE
 1020 REM   24-12-1984  M.BROEK  IJMUIDEN
 1030 REM
 1040 BY=2048:VG$(1)="Oplopend":VG$(2)="Aflopend"
 1050 DEF FNA(X)=NB*X+1:PRINT!(18)!(24)
 1060 PRINT&(17,1)"** GROTE SORTEERROUTINE **":PRINT:PRINT
 1070 INPUT"Geef naam bestand";A$:T=LEN(A$):IFT>OTHENT=ASC(A$)
 1080 IFT<65ORT>90GOTO1070
 1090 FI$=A$:PRINT
 1100 FI$=FI$+"     ":FI$=LEFT$(FI$,5)+"0"
 1110 DISK OPEN,6,FI$:POKE12076,6:POKE12042,BY/64
 1120 INPUT#6,A$,NB,NF,PH,EN:IFEN=1GOTO1610
 1130 EN=EN-1:DISK GET,1:PRINT&(0,2)!(24)
 1140 PRINT:PRINT"Op welk veld sorteren:":PRINT
 1150 FORI=1TONF:INPUT#6,A$,B$:PRINTTAB(5)I">  ";A$:NEXT:PRINT
 1160 INPUT"Keuze";SE:IFSE<1ORSE>NFGOTO1160
 1170 PRINT:PRINT"Sorteervogorde :":PRINT
 1180 FORV=1TO2:PRINTV;"> ";VG$(V):NEXTV
 1190 PRINT:INPUT"Keuze";Y$
 1200 VG=VAL(Y$):IFY$=""THENVG=1
 1210 IFVG<1ORVG>2ORVG<>INT(VG)THEN1170
 1220 PRINT!(18)!(24):PRINT"Opgegeven info :":PRINT
 1230 PRINT"File type"TAB(21)": M-DMS MASTER"
 1240 PRINT"Aantal velden"TAB(21)":"NF
 1250 PRINT"Sorteren op veld"TAB(21)":"SE
 1260 PRINT"Records te sorteren"TAB(21)":"EN
 1270 PRINT"Sorteervolgorde"TAB(21)": "VG$(VG)
 1280 PRINT:INPUT"Klopt dit";Y$
 1290 IFLEFT$(Y$,1)="N"THENRUN
 1300 DIMC(EN),S$(EN),NP$(NF),OP$(NF)
 1310 PRINT:PRINT"Er wordt gesorteerd ---":PRINT
 1320 FORI=1TOEN:DISK GET,FNA(I)
 1330 FORJ=1TOSE:INPUT#6,S$(I):NEXTJ:NEXTI
 1340 REM SORTEERROUTINE
 1350 FORI=1TOEN:C(I)=I:NEXTI
 1360 FORI=2TOEN:X1=1:X2=I:P=C(I)
 1370 X3=X1+INT((X2-X1)/2):P1=C(X3):IFVG=2GOTO1410
 1380 IFS$(P)<S$(P1)THEN1440
 1390 IFS$(P)>S$(P1)THEN1460
 1400 GOTO1430
 1410 IFS$(P)>S$(P1)THEN1440
 1420 IFS$(P)<S$(P1)THEN1460
 1430 X2=X3:GOTO1480
 1440 IFX2=X3THEN1480
 1450 X2=X3:GOTO1370
 1460 IFX1=X3THEN1480
 1470 X1=X3:GOTO1370
```

```
1480 IFI=X2THEN1500
1490 FORX=ITOX2+1STEP-1:C(X)=C(X-1):NEXTX
1500 C(X2)=P:NEXTI
1510 REM VERPLAATS RECORDS
1520 FORI=1TOEN:X1=C(I):X2=I:PRINTCHR$(13);:PRINT"Record"I;
1530 IFC(X2)<>ITHENX2=X2+1:GOTO1530
1540 IFC(I)=ITHEN1600
1550 DISK GET,FNA(I):FORJ=1TONF:INPUT#6,OP$(J):NEXT
1560 DISK GET,FNA(X1):FORJ=1TONF:INPUT#6,NP$(J):NEXT
1570 DISK GET,FNA(X1):FORJ=1TONF:PRINT#6,OP$(J):NEXT:DISK PUT
1580 DISK GET,FNA(I):FORJ=1TONF:PRINT#6,NP$(J):NEXT:DISK PUT
1590 C(I)=I:C(X2)=X1
1600 NEXTI
1610 PRINT:PRINT"Sorteren gereed"
1620 PRINT:PRINT"Zorg dat de M-DMS schijf in drive A zit"
1630 INPUT"en geef <RETURN> voor doorgaan";Y$
1640 DISK!"SE A":RUN"BEXEC*
```

De volgende listing is een labelprinter voor M-DMS.
Er wordt gebruik gemaakt van de printerbuffer van
Hans de Jong uit OSI POEL blz CC 15 en verder. Verwijder
regel 1030 als deze niet gebruikt wordt. Als printer
is een Brother CE40 super gebruikt, de initialisatie
voor de printer moet dus aan de eigen printer aangepast
worden. Dit betrefd de regels 1690-1760 en 2360-2420.
Als printeroutput is device 4 in gebruik, wijzig
in regel 1040 eventueel DV=4 in DV=1. In de regels
1070-1090 staan de default waarden van de stickers.

```
1000 REM Labelprinter voor M-DMS.
1010 REM M.Broek IJmuiden
1020 REM V 1.0  05/06/85
1030 DISK!"CA 8000=12,6":DISK!"GO 8000":REM printerbuffer
1040 NF=30:DV=4:TRAP1920
1050 DIMFL$(NF),FL(NF),EN$(NF),T(150),P(150):DEFFNA(X)=NB*X+1
1060 PRINT!(18)!(24)&(17,1)"** LABELS DRUKKEN **"
1070 AF=45:REM Afstand tussen stickers
1080 HO= 6:REM Hoogte van de stickers
1090 AS= 2:REM Aantal stickers naast elkaar
1100 PRINT:PRINT"Hoeveel stickers zitten er naast elkaar <"AS;
1110 INPUT">";Y$:YL=LEN(Y$):IFYL<>0THENAS=VAL(Y$)
1120 IFAS<1ORAS>4THEN1100
1130 IFAS=1THENAF=132:GOTO1170
1140 PRINT:PRINT"Wat is de breedte van de stickers <"AF;
1150 INPUT">";Y$:YL=LEN(Y$):IFYL<>0THENAF=VAL(Y$)
1160 IFAF<50RAF>80THEN1140
1170 PRINT:PRINT"Wat is de hoogte van de stickers <"HO;
1180 INPUT">";Y$:YL=LEN(Y$):IFYL<>0THENHO=VAL(Y$)
1190 PRINT:INPUT"Hoeveel regels moeten er op en sticker";Y$
1200 YL=LEN(Y$):IFYL=0THEN1190
1210 RE=VAL(Y$):IF(HO-RE)<2THENPRINT"ONMOGENLIJK!":GOTO1190
1220 PRINT:PRINT"De layout is dus als volgt:":PRINT
1230 PRINT"Aantal stickers naast elkaar  :";AS
1240 PRINT"Afstand tussen de stickers    :";AF
1250 PRINT"De hoogte van de stickers     :";HO
1260 PRINT"Aantal regels op een sticker  :";RE
```

```
1270 PRINT:INPUT"Klopt dit < J >";Y$
1280 IFLEFT$(Y$,1)="N"THEN1060
1290 PRINT!(18)!(24):INPUT"Welk bestand kiest U";F$
1300 IFF$=""ANDFI$<>""THEN1330
1310 FI$=F$:L=LEN(FI$):IFL<10RL>5THEN1290
1320 FI$=FI$+"    ":FI$=LEFT$(FI$,5)+"0"
1330 PRINT!(18)!(24):DISK OPEN,6,FI$:POKE12076,6:POKE12042,32
1340 INPUT#6,A$,NB,NF,PH,EN:EN=EN-1
1350 IFEN=0THENPRINT"Dit bestand is leeg":GOTO1920
1360 DISK GET,1:FORI=1TONF:INPUT#6,FL$(I)
1370 INPUT#6,A$:FL(I)=ASC(A$)-45:IFFL(I)>18THENFL(I)=FL(I)-1
1380 NEXT
1390 PRINT!(18)!(24)
1400 GOSUB2040
1410 FORI=1TORE:GOSUB1980:NEXTI
1420 PRINT:PRINT"De indeling is dus:":PRINT
1430 FORI=1TORE:FORJ=1TORE(I):PRINTFL$(RI(I,J));" ";:NEXTJ
1440 PRINT:NEXTI
1450 ML=0:ER=0:FORI=1TORE:LE=0:FORJ=1TORE(I)
1460 LE=LE+FL(RI(I,J))+1:IFLE>MLTHENML=LE
1470 NEXTJ:IFLE<AF-3GOTO1490
1480 PRINT"Regel"I"is te lang!":ER=1
1490 NEXTI:IFER=1THEN1400
1500 PRINT:INPUT"Klopt dat < J >";Y$
1510 IFLEFT$(Y$,1)="N"GOTO1390
1520 PRINT!(18)!(24):PRINT"Wilt U een sorteervoorwaarde"
1530 INPUT"op een van de velden plaatsen < N >";Y$
1540 SC$="":SC=0:IFLEFT$(Y$,1)<>"J"GOTO1640
1550 GOSUB2040:PRINT:INPUT"Op welk veld";SF
1560 IFSF<10RSF>NFGOTO1550
1570 PRINT:PRINT"1> Kleiner dan"
1580 PRINT"2> Kleiner dan of gelijk aan":PRINT"3> Gelijk aan"
1590 PRINT"4> Groter dan of gelijk aan":PRINT"5> Groter dan"
1600 PRINT"6> Ongelijk aan":PRINT"7> Gelijk aan linkerdeel"
1610 PRINT:INPUT"Welke voorwaarde";SC
1620 IFSC<10RSC>7GOTO1570
1630 PRINT:INPUT"Vergelijkings woord";CS$:PRINT!(18)!(24)
1640 PRINT:PRINT"Record(s) zijn vol, beginnen met"
1650 INPUT"welk record < alles >";B$:IFB$=""THENB=1:E=EN:GOTO1680
1660 B=VAL(B$):IFB<10RB>ENGOTO1640
1670 INPUT"en eindigen met record";E:IFE<BORE>ENGOTO1640
1680 REM
1690 REM Initialiseer printer (BROTHER CE40 SUPER)
1700 DW=ML+((AS-1)*AF):IFDW=<78THEN1770
1710 IFDW=<90THEN1740
1720 IFDW>132THENPRINT"Breedte is groter dan 132!":GOTO1070
1730 PRINT#DV,CHR$(6)CHR$(19)CHR$(8)CHR$(17)CHR$(18);
1740 FORJ=1TO70:PRINT#DV," ";:NEXTJ
1750 FORJ=70TODW:PRINT"DV," ";CHR$(6);:NEXTJ
1760 GOSUB2410
1770 PRINT!(18)!(24):INPUT"Een proefdruk < N >";Y$
1780 IFLEFT$(Y$,1)<>"J"GOTO1840
1790 FORI=1TORE:FORJ=1TOAS
1800 PRINT#DV,TAB((J-1)*AF)I;:NEXTJ:PRINT#DV:NEXTI
1810 FORI=1TOHO-RE:PRINT#DV:NEXTI
1820 PRINT:INPUT"Nog een proefdruk < N >";Y$
1830 IFLEFT$(Y$,1)="J"GOTO1790
1840 PRINT:INPUT"Geef <RETURN> als printer klaar is";Y$
```

```
1850 RC=B:SN=1
1860 GOSUB2060:IFRC=0THEN1890
1870 GOSUB2280:IFSN<>ASTHENSN=SN+1:GOTO1860
1880 SN=1:GOSUB2320:GOTO1860
1890 IFPR$(1,1)=""THEN1910
1900 GOSUB2320
1910 GOSUB2360:GOTO1930
1920 GOSUB2360:GOTO1950
1930 PRINT:INPUT"nog een bestand < N >";Y$
1940 IFLEFT$(Y$,1)="J"THEN1060
1950 PRINT:PRINT"Zorg dat de M-DMS schijf in drive A zit"
1960 INPUT"en geef <RETURN> voor doorgaan";Y$
1970 DISK!"SE A":RUN"BEXEC*:END
1980 PRINT:PRINT"Geef velden in regel"I;
1990 INPUT"gescheiden door komma's";A$:L=LEN(A$):IFL=0GOTO1980
2000 A$=A$+",":L=L+1:K=1:M=1:N=1:O=1
2010 K=K+1:IFMID$(A$,K,1)<>","GOTO2010
2020 RI(I,O)=VAL(MID$(A$,N,K-N)):N=K+1:O=O+1:IFK<LTHEN2010
2030 RE(I)=O-1:RETURN
2040 PRINT"De velden zijn:":PRINT
2050 FORI=1TONF:PRINTI"> "FL$(I):NEXT:RETURN
2060 IFRC=E+1THENRC=0:RETURN
2070 DISK GET,FNA(RC):FORQ=1TONF:INPUT#6,EN$(Q):NEXTQ
2080 RC=RC+1:IFSC=0THENRETURN
2090 SC$=CS$:F$=EN$(SF):S=0:IFSC=7THEN2260
2100 B1$="":B2$=" ":IFVAL(SC$)<>0THENB1$=" ":B2$=""
2110 IFLEN(SC$)<LEN(F$)THENSC$=B1$+SC$+B2$
2120 IFLEN(SC$)>LEN(F$)THENF$=B1$+F$+B2$
2130 ONSCGOTO2220,2200,2180,2160,2140,2240
2140 IFSC$<F$THENRETURN
2150 GOTO2060
2160 IFSC$=<F$THENRETURN
2170 GOTO2060
2180 IFSC$=F$THENRETURN
2190 GOTO2060
2200 IFSC$=>F$THENRETURN
2210 GOTO2060
2220 IFSC$>F$THENRETURN
2230 GOTO2060
2240 IFSC$<>F$THENRETURN
2250 GOTO2060
2260 SL=LEN(SC$):IFLEFT$(F$,SL)=SC$THENRETURN
2270 GOTO2060
2280 FORI=1TORE:PR$(I,SN)="":FORJ=1TORE(I)
2290 IFEN$(RI(I,J))="-"GOTO2310
2300 PR$(I,SN)=PR$(I,SN)+EN$(RI(I,J))+" "
2310 NEXTJ:NEXTI:RETURN
2320 FORI=1TORE:FORJ=1TOAS
2330 PRINT#DV,TAB((J-1)*AF)PR$(I,J);:NEXTJ:PRINT#DV:NEXTI
2340 FORI=1TOHO-RE:PRINT#DV:NEXTI
2350 FORI=1TORE:FORJ=1TOAS:PR$(I,J)="":NEXTJ:NEXTI:RETURN
2360 IFDW=<90THENRETURN
2370 PRINT#DV,CHR$(19)CHR$(8)CHR$(17);
2380 FORJ=1TO12:PRINT#DV," ";:NEXTJ:PRINT#DV,CHR$(18);
2390 FORJ=1TO70:PRINT#DV," ";:NEXTJ
2400 FORJ=1TO9:PRINT#DV," ";CHR$(6);:NEXTJ
2410 PRINT#DV,CHR$(19)CHR$(18)CHR$(17)CHR$(19)CHR$(8)CHR$(17);
2420 RETURN
```

```
1 REM MERGINSTRUKTIES ØP 500-  (JOS BURGHOUTS)
5 PRINT!(28);:Ø0=0:Ø1=0:Ø2=32:Ø3=64
7 GØSUB50
8 REM
10 GØTØ 1000
20 GØTØ 1010
50 PØKE9394,32:PØKE9395,144:PØKE9396,53:PØKE9397,168
51 PØKE9398,169:PØKE9399,0:PØKE9400,76:PØKE9401,24
52 PØKE9402,18:RETURN
100 PRINT!(28);:RETURN
110 REM
111 Ø4=HØ:IF HØ>Ø3 THEN Ø4=Ø3
112 Ø5=VE:IF VE>Ø2 THEN Ø5=Ø2
113 PRINT:PRINT!(17,Ø4,Ø5);:RETURN
120 HØ=PEEK(13023):VE=PEEK(13024):RETURN
200 PØKE574,178:PØKE575,36:PI=USR(PI)
201 IFPI=ØTHENIN$="":RETURN
202 IN$=CHR$(PI):RETURN
210 PØKE574,40:PØKE575,37:IN=USR(1):IN$=CHR$(PEEK(9059)):RETURN
250 PØKE574,214:PØKE575,251:PI=USR(PI):RETURN
260 RV=RND(1):RETURN
270 FR=FRE(2):RETURN
300 SR$=STR$(SR)
301 Ø7=LEN(SR$):IFØ7=0 THEN RETURN
302 IF RIGHT$(SR$,1)<>" " THEN 304
303 SR$=LEFT$(SR$,Ø7-1):GØTØ 301
304 IF LEFT$(SR$,1)<>" " THEN RETURN
305 SR$=RIGHT$(SR$,Ø7-1):GØTØ 301
310 Ø4=SR:IF CN<>0 THEN 316
312 SR=INT(SR+.5):GØSUB300:GØTØ330
316 Ø5=SGN(SR):SR=ABS(SR):Ø8=INT(SR):Ø9=SR-Ø8
318 FØR Ø6=1 TØ CN:Ø9=Ø9*10:NEXT Ø6
320 Ø9=INT(Ø9+.5):SR=Ø9:GØSUB 300
322 Ø9$=RIGHT$("ØØØØØØØØØØØØØØØØØØØØ"+SR$,CN)
324 IF Ø8=0 AND Ø9=0 THEN Ø5=1
326 SR=Ø8:GØSUB300:IF Ø5=-1 THEN SR$="-"+SR$
328 SR$=SR$+"."+Ø9$
330 IF LEN(SR$)<=CT THEN 334
332 SR$=LEFT$("********************",CT):GØTØ 340
334 SR$=RIGHT$("                    "+SR$,CT)
340 SR=Ø4:RETURN
350 DISK!"I0 ,03":PRINTSR$;:DISK!"I0 ,02":RETURN
360 DISK!"I0 ,03":PRINT:DISK!"I0 ,02":RETURN
500 REM MERGEN; UIT ØSI-PØEL EE 101, HANS DE JØNG
510 REM DISK!"LØ BASICØ   ØF "LØ 38
520 REM DISK!"MEM EØØØ,EØØØ
530 REM LIST#5
540 REM PRINT#5,"EINDE"
550 REM DISK!"LØ PRØGRAMMA
560 REM DISK!"IØ 10
570 REM DISK!"PU NIEUWE-PRØGRAMMA
```

OSI FOEL

SEPTEMBER 1985

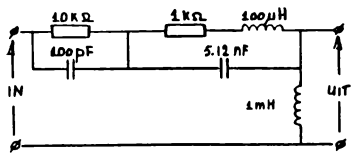
# NETWERK

Dit programma helpt u bij het ontwerp van analoge schakelingen. Het verzorgt een snel overzicht van fase- en frekwentiegedrag van een bepaalde schakeling in de ontwerpfase, zonder dat hiervoor eerst een proefschakeling gebouwd dient te worden.
Voor een inzicht in de werking en dus het gebruik van het programma, dient eerst uitgelegd hoe het gedrag van een analoge schakeling kan worden gedefinieerd:
Hiertoe wordt van elke combinatie van met elkaar verbonden weerstanden, condensatoren, op-amps, spoelen, transistors en/of andere elektronische componenten aangenomen dat ze zich binnen een 'black box' bevinden met alleen een tweetal ingangs- en een tweetal uitgangsklemmen.
Het gedrag van dit 'zwarte blok' voor wisselspanning laat zich nu door een tweetal faktoren definieren:

1) De amplitudeverhouding van in- en uitgangsspanning, ook wel versterking genoemd, en
2) De fasedraaiing tussen in- en uitgangssignaal.

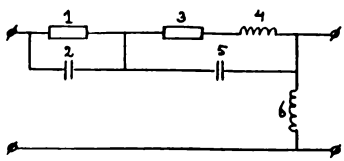
Door nu voor een aantal frekwenties deze faktoren te bepalen, kunnen we het gedrag van de 'black box', en daarmee van onze schakeling vastleggen; beide faktoren grafisch uitgezet tegen de frekwentie levert ons de fase- resp. amplituderesponsie.
Let wel, er wordt alleen beschouwd wat het wisselstroomgedrag van onze schakeling zal worden; alle gelijkspanningsinstellingen worden daarom achterwege gelaten.
Zo zal b.v. voor wisselspanning de voedingslijn zich gedragen als een zeer laagohmige belasting: in het wisselspanningsschema wordt een weerstand die naar een voedingslijn gaat, dan ook beschouwd als gaande naar de common. (aarde=laagohmig)

Een voorbeeld zal e.e.a. verduidelijken:
Gewenst is een inzicht in fase- en amplituderesponsie van het hieronder geschetste netwerkje (figuur 1), bestaande uit enige passieve componenten.



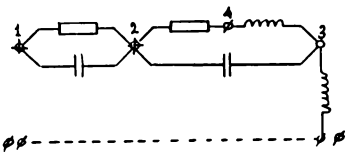

figuur 1: te analyseren schakeling

We beginnen met elk van de componenten een serienummer te geven (zie hiervoor figuur 2)




figuur 2: benummeren componenten

Hierna wordt elk van de afzonderlijke knooppunten (=punt, waar twee of meer componenten aan elkaar vastzitten, alsmede in-, uitgang en common) benummerd:
Let er wel op, dat de common het knooppuntnummer 0 krijgt!




figuur 3: benummeren knooppunten

Voor het programma NETWERK gestart kan worden, dient nog eerst het utility-programma ATNENB (disk 5 DOS 3.3) gerund te worden, omdat het NETWERK-programma gebruik maakt van de ATN-functie.
Hierna kan ons programma geladen worden.
Het invoeren van de benodigde gegevens volgt nu vanzelf:
Voor het gegeven voorbeeld voeren we in:

| | | | | | | |
|---|---|---|---|---|---|---|
| Component 1: | R | (van) | 1 | (naar) | 2, (waarde) | 10 Kilo-Ohm |
| Component 2: | C | | 1 | | 2, | .0001 microFarad |
| Component 3: | R | | 2 | | 4, | 1 Kilo-Ohm |
| Component 4: | L | | 4 | | 3, | .1 milliHenry |
| Component 5: | C | | 2 | | 3, | .00512 microFarad |
| Component 6: | L | | 3 | | 0, | 1 milliHenry |

Afsluiten door E(inde invoer) te geven, waarna het programma vraagt om opgave van ingangs-, resp. uitgangsknooppunt.

N.B.: Zoals in het voorbeeld aangegeven, dienen alle componenten qua waarde te worden teruggerekend naar de opgegeven normfactor.

Vervolgens vraagt het programma om opgave van het frekwentie-gebied, waarin de berekeningen moeten worden uitgevoerd.
Dit wordt opgegeven door beginfrekwentie, eindfrekwentie, en stapgrootte.
De laatste parameter kan op twee manieren worden gekozen:
wil men een lineaire verdeling van het frekwentiegebied, dient men een positieve waarde in te geven, gelijk aan de stapgrootte van de frekwentie;
wil men daarentegen een logarithmische verdeling, dan geeft men een negatief (-) bedrag op, waarvan de waarde gelijk is aan het aantal stappen (in frekwentie) tussen begin en eindwaarde.
Voor het netwerk, ingevoerd met bovenvermelde componenten, zijn een tweetal frekwentiedomeinen als voorbeeld gegeven:
Een met lineaire frekwentieverdeling, en een met logarithmische, beide met een twintig verschillende frekwentiewaarden.
(Uiteraard is het meest interessante frekwentiegebied gegeven)
De grafische representatie van de verkregen waardes is t.b.v. dit artikel met de hand getekend, het ligt echter in de bedoeling om ditzelfde plaatje met gewone OSI-graphics of zelfs met high-res graphics op het scherm te gaan presenteren;
Wie geeft me een seintje als hij het heeft gerealiseerd?

Ik wens een ieder bij het werken met dit programma veel succes, mochten er nog op- of aanmerkingen zijn, dan verneem ik dit gaarne van u.

Kees van Luijpen
Leiwater 73
2715 BB Zoetermeer
079-213245

SEPTEMBER 1985

Voorbeeld van berekening met het programma NETWERK van het frekwentiedomein 10 KHz tot 1 MHz, zowel lineair (in stappen van 50 KHz), als logarithmisch (in 20 stappen van minimum naar maximum frekwentie)

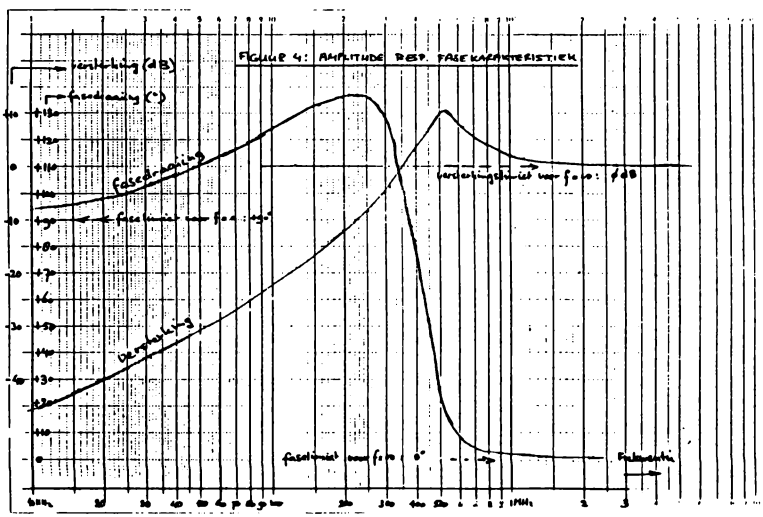
Op de vraag START,STOP,INCREMENT? te antwoorden:

in voorbeeld 1: 1E4,1E6,5E4 en in voorbeeld 2: 1E4,1E6,-20

| | | |
|---|---|---|
| 10000 | -44 | 94 |
| 60000 | -28 | 110 |
| 110000 | -21 | 121 |
| 160000 | -16 | 129 |
| 210000 | -11 | 134 |
| 260000 | -7.8 | 137 |
| 310000 | -3.8 | 136 |
| 360000 | .1 | 132 |
| 410000 | 4.2 | 123 |
| 460000 | 8.3 | 104 |
| 510000 | 10 | 71 |
| 560000 | 9.9 | 42 |
| 610000 | 8.2 | 25 |
| 660000 | 6.7 | 16 |
| 710000 | 5.6 | 11 |
| 760000 | 4.7 | 8.7 |
| 810000 | 4.0 | 6.7 |
| 860000 | 3.5 | 5.3 |
| 910000 | 3.1 | 4.2 |
| 960000 | 2.7 | 3.5 |

| | | |
|---|---|---|
| 10000 | -44 | 94 |
| 12589 | -44 | 95 |
| 15484 | -40 | 96 |
| 19952 | -38 | 98 |
| 25118 | -36 | 100 |
| 31622 | -34 | 102 |
| 39810 | -32 | 104 |
| 50118 | -29 | 107 |
| 63095 | -27 | 111 |
| 79432 | -25 | 115 |
| 100000 | -22 | 119 |
| 125892 | -19 | 124 |
| 158489 | -16 | 129 |
| 199526 | -12 | 133 |
| 251188 | -8.5 | 136 |
| 316227 | -3.4 | 136 |
| 398107 | 3.2 | 126 |
| 501187 | 10 | 77 |
| 630957 | 7.5 | 21 |
| 794328 | 4.2 | 7.2 |
| 1000000 | 2.5 | 3.0 |

Het bijbehorende plaatje is hieronder gegeven:



```
10 REM NETWERKANALYSE, UIT EDN 1/9/82
20 REM BEWERKT DOOR TH. KAPSENBERG, 18/11/1983
30 REM BEWERKT VOOR OSI (DOS 3.3) DOOR K. VAN LUIJPEN, 15/5/1985
35 GOSUB4000
40 PRINT:PRINT:PRINT"##################################################
45 PRINT"#                                                #
50 PRINT"#                  NETWERKANALYSE                #
55 PRINT"#                                                #
60 PRINT"##################################################
80 PRINT:K9=1:REM COMPONENTNUMMER
90 I=0:J=0:K=0:N=0:I1=0:D2=1
100 PRINT:PRINT:PRINT" MAXIMAAL 10 KNOOPPUNTEN
110 PRINT:PRINT" NUMMER ALLE KNOOPPUNTEN EN
120 PRINT:PRINT" COMPONENTEN OPVOLGEND VAN
130 PRINT:PRINT" 1 TOT 10
135 PRINT:PRINT" ONGEBRUIKTE PUNTEN AAN 0
140 PRINT:PRINT" IN EN UIT t.o.v. PUNT 0
150 DIMA(10,10),B(10,10),P(10,10),Q(10,10),B1(10,10),Q1(10,10)
160 GOSUB3130:REM CLEAR ARRAY
170 N=0
180 GOSUB4000
190 C$="   COMPONENTNUMMER:"
200 I=0:J=0:K=0:L=0:B1=0:V=0
210 PRINT"     ";C$;K9
212 POKE13024,4
216 PRINT:PRINT"    VOER COMPONENT IN MET LETTERCODE
218 PRINT:PRINT"   * WEERSTAND          : R
220 PRINT:PRINT"   * CONDENSATOR        : C
222 PRINT:PRINT"   * SPOEL              : L
224 PRINT:PRINT"   * OPAMP              : O
226 PRINT:PRINT"   * FET                : F
228 PRINT:PRINT"   * TRANSISTOR         : T
230 PRINT:PRINT"   * EINDE INVOER       : E
240 PRINT:PRINT"    UW KEUZE?";
242 POKE2797,20
244 INPUTT$
250 IFT$="R"THENN$="WEERSTAND":GOTO320
252 IFT$="C"THENN$="CONDENSATOR":GOTO360
254 IFT$="L"THENN$="SPOEL":GOTO400
256 IFT$="O"THENN$="OPAMP":GOTO530
258 IFT$="F"THENN$="FET":GOTO440
260 IFT$="T"THENN$="TRANSISTOR":GOTO480
262 IFT$="E"THEN610
264 GOTO244
310 END
320 GOSUB4000
322 PRINT: PRINT:PRINT" COMPONENT NUMMER ";K9;" IS EEN ";N$
324 PRINT:PRINT" EN LOOPT VAN KNOOPPUNTNUMMER: ";
326 INPUTI
328 PRINT:PRINT" NAAR KNOOPPUNTNUMMER: ";
330 INPUTJ
340 PRINT:PRINT" EN HEEFT DE WAARDE (IN KILO-OHM): ";
342 INPUTV
350 V=1/(1000*V):GOSUB1460:K9=K9+1:GOSUB4000:GOTO210
360 GOSUB4000
362 PRINT: PRINT:PRINT" COMPONENT NUMMER ";K9;" IS EEN ";N$
364 PRINT:PRINT" EN LOOPT VAN KNOOPPUNTNUMMER: ";
```

```
366 INPUTI
368 PRINT:PRINT" NAAR KNOOPPUNTNUMMER: ";
370 INPUTJ
372 PRINT:PRINT" EN HEEFT DE WAARDE (IN MICRO-FARAD): ";
374 INPUTV
390 V=V/1000000:GOSUB1570:K9=K9+1:GOSUB4000:GOTO210
400 GOSUB4000
402 PRINT: PRINT:PRINT" COMPONENT NUMMER ";K9;" IS EEN ";N$
404 PRINT:PRINT" EN LOOPT VAN KNOOPPUNTNUMMER: ";
406 INPUTI
408 PRINT:PRINT" NAAR KNOOPPUNTNUMMER: ";
410 INPUTJ
412 PRINT:PRINT" EN HEEFT DE WAARDE (IN MILLI-HENRY): ";
414 INPUTV
430 V=-(1000/V):GOSUB1640:K9=K9+1:GOSUB4000:GOTO210
440 GOSUB4000
442 PRINT: PRINT:PRINT" COMPONENT NUMMER ";K9;" IS EEN ";N$
444 PRINT:PRINT" DE GATE ZIT AAN KNOOPPUNTNUMMER:";
446 INPUTK
448 PRINT:PRINT" DE SOURCE ZIT AAN KNOOPPUNTNUMMER:";
450 INPUTJ
452 PRINT:PRINT" DE DRAIN ZIT AAN KNOOPPUNTNUMMER:";
454 INPUTI
456 PRINT:PRINT" DE VERSTERKING (MA/V) IS:";
458 INPUTV
470 V=V/1000:L=J:GOTO600
480 GOSUB4000
482 PRINT: PRINT:PRINT" COMPONENT NUMMER ";K9;" IS EEN ";N$
484 PRINT:PRINT" DE BASIS ZIT AAN KNOOPPUNTNUMMER:";
486 INPUTK
488 PRINT:PRINT" DE EMITTER ZIT AAN KNOOPPUNTNUMMER:";
490 INPUTJ
492 PRINT:PRINT" DE COLLECTOR ZIT AAN KNOOPPUNTNUMMER:";
494 INPUTI
496 PRINT:PRINT" DE VERSTERKING (BETA) BEDRAAGT:";
498 INPUTB1
500 PRINT:PRINT" DE BASIS-EMITTERWEERSTAND BEDRAAGT:";
502 INPUTV
510 L=I:I=K:V=1/V:GOSUB1460
520 I=L:L=J:GOTO590
530 GOSUB4000
532 PRINT: PRINT:PRINT" COMPONENT NUMMER ";K9;" IS EEN ";N$
534 PRINT:PRINT" DE +INPUT ZIT AAN KNOOPPUNTNUMMER:";
536 INPUTK
538 PRINT:PRINT" DE -INPUT ZIT AAN KNOOPPUNTNUMMER:";
540 INPUTL
542 PRINT:PRINT" DE +OUTPUT ZIT AAN KNOOPPUNTNUMMER:";
544 INPUTJ
546 PRINT:PRINT" DE -OUTPUT ZIT AAN KNOOPPUNTNUMMER:";
548 INPUTI
550 PRINT:PRINT" DE 'OPEN LOOP'VERSTERKING BEDRAAGT:";
552 INPUTB1
554 PRINT:PRINT" DE UITGANGSIMPEDANTIE (IN OHMS) BEDRAAGT:";
556 INPUTV
580 V=1/V:GOSUB1460
590 V=B1*V
600 GOSUB1710:K9=K9+1:GOSUB4000:GOTO210
```

```
610 GOSUB4000:PRINT:PRINT"   INPUT IS KNOOPPUNTNUMMER:";
612 INPUTE
614 PRINT:PRINT"  OUTPUT IS KNOOPPUNTNUMMER:";
616 INPUTF
620 FORI=0TON
630 FORJ=0TON
640 P(I,J)=A(I,J)
650 Q1(I,J)=B1(I,J)
660 Q(I,J)=B(I,J)
670 NEXT:NEXT
680 GOSUB4000
690 PRINT:PRINT:PRINT"     SCHAKELING HEEFT ";N;" KNOOPPUNTEN."
700 PRINT:PRINT"    KNOOPPUNT";E;"IS INPUT,"
702 PRINT:PRINT"    KNOOPPUNT";F;"IS OUTPUT."
800 FORX=0TO1000:NEXT
810 PRINT
820 PRINT
830 PRINT" START, STOP, INCREMENT? (+LIN, -LOG)
840 INPUTG,H,D
1000 FORX=0TO1000:NEXT
1001 GOSUB4000
1002 PRINT"   FREKWENTIE  VERSTERKING   FASE
1003 PRINT"       (Hz)         (dB)       (o)
1004 PRINT
1010 IFD<0THENF2=-D+1:GOTO1030
1020 F2=1+(H-G)/D
1030 IFD<0THEND=-((H/G)^(1/(-D)))
1040 F1=G
1110 FORI1=1TOF2
1120 W=2*3.1415*F1:D1=E:D2=F:GOSUB2340
1130 V=B1:U=D2
1140 IF(-1)^(E+F)>0THEN1160
1150 U=U-180
1160 D1=E:D2=E
1170 GOSUB2340:V=V/B1:U=U-D2
1180 IFU>180THENU=U-360
1190 IFU<-180THENU=U+360
1200 D0=8.68588964*LOG(V)
1220 PRINTTAB(10-LEN(STR$(INT(F1))))INT(F1);
1221 IFLEFT$(RIGHT$(STR$(D0),4),2)="E-"THENPRINTTAB(18)"0.00";:GOTO1227
1222 PRINTTAB(18)LEFT$(STR$(D0),4);
1227 PRINTTAB(28)LEFT$(STR$(U),4)
1250 IFD>0THEN1270
1260 X9=200*(I1-1)/(F2-1)+47:GOTO1280
1270 X9=(F1-G)*200/(H-G)+47
1280 GOTO1300
1300 IFD<0THENF1=-(F1*D):GOTO1320
1310 F1=F1+D
1320 NEXTI1
1330 PRINT:PRINT"ANDER FREKWENTIEGEBIED? (J/N)
1340 INPUTZ$
1350 IFZ$="J"THENGOSUB4000:GOTO830
1360 END
1460 IFI=0THEN1510
1470 A(I,I)=A(I,I)+V
1480 IFJ=0THEN1520
```

```
1490 A(I,J)=A(I,J)-V
1500 A(J,I)=A(J,I)-V
1510 A(J,J)=A(J,J)+V
1520 IFI<NTHEN1540
1530 N=I
1540 IFJ<NTHEN1560
1550 N=J
1560 RETURN
1570 IFI=0THEN1620
1580 B(I,I)=B(I,I)+V
1590 IFJ=0THEN1520
1600 B(I,J)=B(I,J)-V
1610 B(J,I)=B(J,I)-V
1620 B(J,J)=B(J,J)+V
1630 GOTO1520
1640 IFI=0THEN1690
1650 B1(I,I)=B1(I,I)+V
1660 IFJ=0THEN1520
1670 B1(I,J)=B1(I,J)-V
1680 B1(J,I)=B1(J,I)-V
1690 B1(J,J)=B1(J,J)+V
1700 GOTO1520
1710 IFI<>0ANDK<>0THENA(I,K)=A(I,K)+V
1720 IFJ<>0ANDL<>0THENA(J,L)=A(J,L)+V
1730 IFJ<>0ANDK<>0THENA(J,K)=A(J,K)-V
1740 IFI<>0ANDL<>0THENA(I,L)=A(I,L)-V
1750 IFK<NTHEN1770
1760 N=K
1770 IFL<NTHEN1790
1780 N=L
1790 GOTO1520
1800 REM BEREKENING DETERMINANT
1810 IFN>1THEN1830
1820 D1=A(N,N):D2=B(N,N):RETURN
1830 D1=1:D2=0:K=1
1840 L=K
1850 S=ABS(A(K,K))+ABS(B(K,K))
1860 FORI=KTON
1870 T=ABS(A(I,K))+ABS(B(I,K))
1880 IFS>=TTHEN1900
1890 L=I:S=T
1900 NEXTI
1910 IFL=KTHEN1990
1920 FORJ=1TON
1930 S=-A(K,J)
1940 A(K,J)=A(L,J)
1950 A(L,J)=S
1960 S1=-B(K,J)
1970 B(K,J)=B(L,J):B(L,J)=S1
1980 NEXTJ
1990 L=K+1
2000 FORI=LTON
2010 S1=A(K,K)*A(K,K)+B(K,K)*B(K,K)
2020 S=(A(I,K)*A(K,K)+B(I,K)*B(K,K))/S1
2030 B(I,K)=(A(K,K)*B(I,K)-A(I,K)*B(K,K))/S1
2040 A(I,K)=S:NEXTI
```

```
2050 J2=K-1
2060 IFJ2=0THEN2120
2070 FORJ=LTON
2080 FORI=1TOJ2
2090 A(K,J)=A(K,J)-A(K,I)*A(I,J)+B(K,I)*B(I,J)
2100 B(K,J)=B(K,J)-B(K,I)*A(I,J)-A(K,I)*B(I,J)
2110 NEXTI:NEXTJ
2120 J2=K:K=K+1
2130 FORI=KTON
2140 FORJ=1TOJ2
2150 A(I,K)=A(I,K)-A(I,J)*A(J,K)+B(I,J)*B(J,K)
2160 B(I,K)=B(I,K)-B(I,J)*A(J,K)-A(I,J)*B(J,K)
2170 NEXTJ:NEXTI
2180 IFK<>NTHEN1840
2190 L=1
2200 J2=INT(N/2)
2210 IFN=2*J2THEN2250
2220 L=0
2230 D1=A(N,N)
2240 D2=B(N,N)
2250 FORI=1TOJ2
2260 J=N-I+L
2270 S=A(I,I)*A(J,J)-B(I,I)*B(J,J)
2280 S1=A(I,I)*B(J,J)+A(J,J)*B(I,I)
2290 T=D1*S-D2*S1
2300 D2=D2*S+D1*S1
2310 D1=T
2320 NEXTI
2330 RETURN
2340 N1=N:N=N-1:I=0
2350 FORK=1TON
2360 IFK<>D1THEN2380
2370 I=1
2380 J=0
2390 FORL=1TON
2400 IFL<>D2THEN2420
2410 J=1
2420 A(K,L)=P(K+I,L+J)
2430 B(K,L)=W*Q(K+I,L+J)+Q1(K+I,L+J)/W
2440 NEXT:NEXT
2450 GOSUB1810
2460 N=N1
2470 B1=SQR(D1^2+D2^2)
2480 IFD1<>0THEN2520
2490 IFD2=0THEN2560
2500 IFD2>0THEND2=90:GOTO2560
2510 D2=-90:GOTO2560
2520 IFD1<0THENQ=180:GOTO2540
2530 Q=0
2540 IFD2<0THENQ=-Q
2550 D2=Q+ATN(D2/D1)*180/3.1415
2560 RETURN
3130 REM CLEAR ARRAY
3140 FORI=0TO10:FORJ=0TO10
3150 A(I,J)=0:B(I,J)=0:P(I,J)=0:Q(I,J)=0:B1(I,J)=0:Q1(I,J)=0
3160 NEXT:NEXT:RETURN
4000 POKE574,6:POKE575,254:X=USR(X):POKE13024,1:
     RETURN:REM SCREENCLEAR
```

FORTH WORKSHOP

BACK UPS

A. De systeemdisk, d.i. de disk met het kernel-programma kan worden gecopieerd met het programma COPIER onder OS65-D. Alleen de eerste vier tracks hoeven gecopieerd te worden. Als U screens 10 en hoger wil gebruiken voor programma's, dan eerst de disk als DATA-DISK initialiseren.
B. De programmadisks kunnen als volgt worden geinitialiseerd en vervolgens geformat.
   1. initialiseer de disk als data-disk m.b.v. BEXEC*
   2. start FORTH op en format de disk met:

```
: *FORMAT EMPTY-BUFFERS
          80 2 DO I BUFFER 1024 BLANKS
                UPDATE
                LOOP FLUSH ;
```

   (Dit werkt bij mij niet als aantal buffers is verminderd.)
   3. Copieer track zero met 'TRACK ZERO UTILITY' onder DOS Deze utility staat op de OS65-D MASTERDISK track 06,4.

FORTH

De naam: Charles H. Moore ontwikkelde deze taal eind zestiger jaren. Hij werkte toen voornamelijk op een IBM 1130. Dit is een derde generatie computer. Moore vond zijn taal een taal van de vierde generatie. En zo noemde Moore danook zijn taal: VIERDE Op de IBM 1130 mogen namen van variabelen, files, etc. uit niet meer dan vijf karakters bestaan. Vandaar FORTH i.p.v. FOURTH ook wel afgekort 4TH.

De taal: Voor de gebruiker van de computer en vooral de micro-computer zijn de programmeertalen, die tot nu toe beschikbaar zijn erg star. FORTH is zeer flexibel, uitbreidbaar en zelfs is aan de commando's een andere taak te geven. Het concept van FORTH is heel anders.

Wat is FORTH

Moore:
- hogere programmeertaal ) alles houdt zich aan
- operating system ) hetzelfde protocol
- assembleertaal ) daardoor sneller
- toolkit ) programmeren
- software ontwikkelfilosofie.

Arie Kattenburg van fig-nederland:

FORTH is een
- interaktieve
- modulair opgebouwde
- uitbreidbare
- programmeeromgeving.

Een compleet FORTH-systeem omvat 16 kbyte RAM. Hiervan vormt ca 7 kbyte de kern met 2 kbyte of meer machinecode. De woorden van deze hardware afhankelijke kern heten PRIMITIEVEN. De rest van de taal steunt op deze PRIMITIEVEN. Voor welke microprocessor de kern ook geschreven is, de taak van elk woord is voor elke machine dezelfde.

Ga je uit van een standaard FORTH-pakket en schrijf je een programma, dat alleen op jouw machine draait, dan heb je in feite een nieuwe taal geschreven met FORTH als achtergrond. Verwerk je geen machine-eigen mogelijkheden, dan loopt het FORTH-programma op elke machine, want FORTH blijft dialectvrij.
Er zijn drie standaarden:
fig-FORTH ) de verschillen zijn klein en hebben
FORTH-79 ) voornamelijk betrekking op de wijze van
FORTH-83 ) compileren.
Bij het werken met de FORTH-versie, waarmee ik op mijn OSI superboard II werk, wordt gebruik gemaakt van:

- bootstrap
- OUTCH $FFEE )
- INCH $FFEB ) aanroepadres
- CRLF $A86C ) ) BASIC-ROM
- XBLANK $A8E0 ) )
- er wordt gekeken naar $0200-$0230 de start is op $0400.

Snelheid: de tijd benodigd voor een software-opdracht:
MACHINETAAL : FORTH : BASIC ongeveer 1 : 10 : 200
Het schrijven van een machinetaal-routine vanuit FORTH is zeer eenvoudig:
Laad de screen-EDITOR en schrijf het machinetaal-programma in voor de ASSEMBLER begrijpelijke code.
Laad de ASSEMBLER en laad de screens met het programma
Een bezwaar is (voorlopig?), dat de ASSEMBLER dan ook altijd in het geheugen staat bij het runnen.

De EDITOR: dit is een programma, dat de programmeur in staat stelt een programma te ontwikkelen (invoeren, wijzigen). De tekst (source code) wordt min of meer automatisch weggeschreven naar disk.

De disks: deze zijn geformat in screens van 1024 bytes, 16 regels van 64 bytes. Op de disk voor OSI is plaats voor 78 screens, n.l. 2 - 79. Track 00 is niet te gebruiken voor screens.
Screen 4 en 5 (track 02) bevat de foutmeldingen.
Dat is ook de reden, dat de systeemdisk niet kan worden gebruikt voor het ontwikkelen van programma's en voor programma's, die een foutmelding oproepen.
Goedlopende programma's, die geen nieuwe taken geven aan al bestaande woorden kunnen op de systeemdisk vanaf track 05 (screen 10 e.v.) opgeborgen worden.

De ASSEMBLER: dit hulpprogramma verwerkt tijdens het compileren de machinecode-gedeelten van het programma.

De ASSEMBLER en EDITOR zijn geschreven in FORTH.

Programmeren in FORTH = FORTH uitbreiden, zelfs kunnen nieuwe structuren aan FORTH worden toegevoegd.

Compilatie: de invoertrits, hetzij uit de TERMINAL INPUT BUFFER, hetzij uit een BLOCK BUFFER, wordt vertaald van 'klare taal' in machinecode, meestal zijn dat adressen van reeds gedefinieerde woorden, die achter elkaar geplaatst worden. Door compilatie wordt FORTH uitgebreid: FORTH groeit in de richting van hoge RAM.

Uitvoering: als in de invoertrits geen nieuwe definitie wordt aangeboden, maar een getal, dan wordt dit op de PARAMETER STACK gezet. Betreft het een woord, dat in de dictionnaire voorkomt, dan wordt het uitgevoerd. B.v. berekeningen, een programma runnen, etc.

Het FORTH-systeem kan opgebouwd gedacht worden uit:
- BIBLIOTHEEK: de geschakelde reeks van FORTH-woorden, die bij compilatie worden afgezocht van hoge RAM naar lage RAM, totdat het opgegeven woord gevonden is. VLIST <RETURN> brengt de bibliotheek naar het scherm.
- KEYBOARD INTERPRETER: (toetsenbord-vertaler) alles, wat we intikken, gaat naar de TERMINAL INPUT BUFFER (TIB). Na indrukken van <RETURN> wordt dit vertaald of uitgevoerd.
  De SPATIE dient als delimiter, afscheider tussen woorden, getallen en strings.
  De taak van de KEYBOARD INTERPRETER:
  1. woorden worden opgezocht en uitgevoerd;
  2. getallen gaan op de PARAMETER STACK (getallenstapel)
  3. kent FORTH het niet, dan komt hij met een foutmelding, die hij van screen 4 of 5 haalt !!!
  Gaat alles goed, dan meldt FORTH zich na uitvoering terug met OK, en wacht op verdere commando's.
  (Probeer eens HEX 4C41 1372 ! DECIMAL op de OSI !!)

Eenvoudige definities kunnen met : en ; worden ingetikt, zonder gebruik te maken van de EDITOR. B.v.
: HALLO CR CR ." Wat kan FORTH voor U doen?" CR CR ;

De (PARAMETER) STACK: alle getallen worden eerst op STACK gezet, zowel in RUN TIME als in COMPILE TIME.
b.v. 10 20 30 <RETURN> naar de STACK
. . . <RETURN> 30 20 10 haalt ze weer van de STACK en drukt ze af op het scherm (of de printer).

FORTH WERKT IN REVERSED POLISH NOTATION. (POSTFIX-NOTATIE)
Het programma STACK geeft gelegenheid te bekijken, wat de verschillende stack-manipulatiewoorden doen.
STACK-MANIPULATIEWOORDEN:
+ - * / /MOD MOD . MAX MIN MINUS DUP DROP OVER SWAP ROT PICK ROLL PLUNK SPREAD SLIP (en R).

Na het opstarten van FORTH tikken we eerst in:
EMPTY-BUFFERS <RETURN>
SCR#58 in fig-FORTH INSTALLATION MANUAL geeft de definitie.

```
: EMPTY-BUFFERS  ( CLEAR BLOCK BUFFERS: DON'T WRITE TO DISK)
      FIRST LIMIT OVER - ERASE ;
```

FIRST = eerste byte, die gereserveerd is voor buffers $3778
LIMIT = eerste byte voorbij het gebruikte RAM $3F80
OVER = copieer het tweede getal van de STACK en plaats het boven op de STACK.

SCR#46 van idem:

```
: ERASE 0 FILL ;   ( FILL MEMORY WITH ZERO'S
                        BEGIN-2, QUAN-1 )
```

Literatuur over FORTH.

fig-FORTH 6502 ASSEMBLY SOURCE LISTING
release 1.1 FIG
FORTH-79 FIG
FORTH-79 STANDARD CONVERSION FIG
fig-FORTH INSTALLATON MANUAL
GLOSSARY MODEL EDITOR FIG
Alan Winfield Flitsend FORTH 1983
Leo Brodie Starting FORTH 1981
S.J.Wainwright BASIC & FORTH
in parallel 1984
Paul Gardner Learning FORTH
artikelen-serie in COMPUTING TODAY
november 1983 t/m mei 1984 (engels)
BYTE van augustus 1980
is gewijd aan FORTH

FORTH DIMENSIONS FIG
Vijgeblad, wordt uigegeven door HCC-FIG
OSI-POEL Rubrieken EC, EE en FD

Woordopbouw.

```
            head  ( NAME FIELD        NFA doet PFA --> NFA
                  ( LINK FIELD        LFA doet PFA --> LFA
            body  ( CODE FIELD        CFA doet PFA --> CFA
                  ( PARAMETER FIELD   PFA doet NFA --> PFA
```

Het parameter field van CONSTANT en VARIABLE is een cel.
De colon-definitie eindigt met EXIT (;).
Tijdens RUNTIME krijgt de INTERPRETER POINTER, voor het woord wordt uitgevoerd, waarvan het adres in het parameter field staat, het volgende adres. De IP wordt op de return stack gesaved.
EXIT haalt dat adres van de return stack en stopt het weer in de IP. Dan wordt het volgende adres in het parameter-field opgehaald, etc.
Uiteindelijk is alles op en komen we terecht in QUIT, dat er ongeveer uitziet, als volgt.

```
: QUIT BEGIN INTERPRET ." OK" CR 0 UNTIL ;
```

QUIT zorgt voor clear-ing van de return stack.

R> DROP doet hetzelfde als POP in BASIC,
maar je kunt het alleen in een colon-definitie gebruiken.

**Ledenadministratie**

☐ Hierbij geef ik mij op als lid van de HCC

Ik ben reeds lid
code ☐☐ lid nr ☐☐☐☐☐ 1e twee letters van uw achternaam ☐☐

Wilt u de aangekruiste wijziging aanbrengen. →

☐ **Aanmelding Nieuw lid**

☐ **Adres-wijziging**

☐ **Wijziging Afdeling**

☐ **Wijziging GG/WG**

Naam en voornaam:

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Straat en huisnummer:

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ ☐

Postcode ☐☐☐☐ ☐☐ Woonplaats ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ ☐

Telefoonnummer

☐☐☐ ☐☐☐☐☐☐☐

Ik vraag indeling bij afdeling / gebruikersgroep / werkgroep:

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ ☐

Ik bezit volgend systeem:

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ ☐

☐ Ik wens **niet** voor te komen op mailinglijsten.

Handtekening:

In envelop zenden naar:

**Hobby Computer Club**
**Leden Administratie**
**Prinsenhof 11**
**2641 RN Pijnacker**