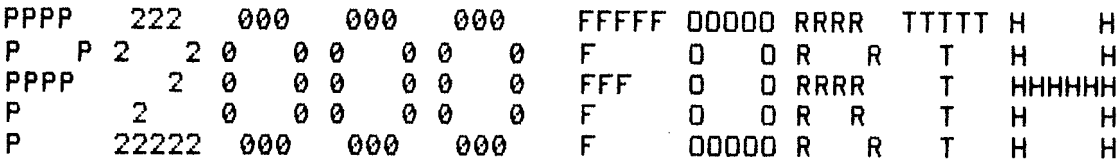

HANDLEIDING BIJ P2000 FORTH 3.0

een uitgave van GPC en NEBO
auteur: F.L. van der Markt
datum: 28-10-1986

Inhoudsopgave:

Inleiding

P2000 FORTH 3.0 is een module voor het eerste slot van de P2000, waarmee u in FORTH kunt programmeren. Het bevat een FORTH bestaande uit FIG-FORTH en enkele uitbreidingen cq. aanpassingen. Deze handleiding heeft tot doel de specifieke eigenschappen van P2000 FORTH 3.0 te beschrijven. Het is beslist geen introduktie in FORTH. Daarvoor bestaan al diverse boeken, zie de literatuurlijst. Vooral het boek "Forth, een taal voor programmeurs" [1] , is zeer geschikt als introduktie in de taal en stemt bovendien erg goed overeen met FORTH 3.0. Zo'n boek is noodzakelijk om zinvol met FORTH te kunnen werken, omdat het nodig is de werking van een aantal standaard definities te kennen.

P2000 FORTH 3.0 heeft een zog. soft-reset mogelijkheid. Dat is een groot voordeel t.o.v. de cassette-Forth die al langer bestond. Het betekent dat u niet het hele systeem opnieuw hoeft te laden als een programma vastgelopen is. Bovendien staat de Forth-code in rom, zodat ie door uw progr. niet per ongeluk gewijzigd kan worden.

Na het aanzetten van de spanning, en na het indrukken van de reset-knop kijkt het systeem of er een cassette aanwezig is. Indien aanwezig, wordt de opdracht: 1 LOAD uitgevoerd. Daarmee kunt u een applicatie automatisch starten.

P2000 FORTH 3.0 heeft een ingebouwde screen-editor waarmee een hele reeks screens tegelijk gewijzigd kan worden. (Zie het betreffende hoofdstuk).

Er is een speciale voorziening gemaakt om eigen applicaties in de eprom erbij te programmeren. Daarvoor is het eigenlijk alleen maar nodig de source te comprimeren. Door een aanpassing van het Fig-FORTH woord BLOCK kan zo'n applicatie dan gestart worden door het LOAD commando. Eventueel kan het BOOT-commando gepatched worden voor starten na power-up of reset.
(bv. HEX 4000 LOAD ipv. 1 LOAD)

Op het ogenblik werkt deze FORTH alleen met de cassette.
De cassette wordt ingedeeld in 40 zog. screens. In het geheugen staan een aantal buffers die ervoor zorgen dat u in feite een virtueel geheugen hebt dat een kant van de cassette omvat.
Door een fout in het FIG-Forth systeem kunnen er problemen optreden wanneer u een blok adresseert zonder dat de cassette aanwezig is, of bij een fout. Het betreffende screen wordt dan toch aanwezig verondersteld, met alle gevolgen vandien.
Geef maar eens .BUFS na een cassette-fout!
Indien er iets fout gaat met de cassette, is het in ieder geval verstandig om FLUSH (save en empty) of EMPTY-BUFFERS (indien er niets gewijzigd is) te geven. Dit maakt de buffers schoon, zodat volgende cassette-operaties beginnen met het screen in te lezen. Vooral de screen-editor heeft last van deze fout. Het is noodzakelijk dat een screen wat men wil editen, van de cassette gelezen wordt.

Ik wens u veel plezier met Forth en hoop dat deze krachtige taal spoedig een aantal vurige beoefenaars zal krijgen.

Extra woorden t.o.v. standaard FIG-FORTH

Notatie: (stack voor uitvoering --- stack na uitvoering)
Het meest rechtse getal is de top van de stack.
n,n1,n2 : 16-bits getallen (met teken)
un : 16-bits getal zonder teken
d,d1,d2 : 32-bits getal met teken
ud : 32-bits getal zonder teken
f : logische vlag, true of false

.BASE (---)
Dit woord drukt in decimale vorm af welk talstelsel aktief is
Voorbeeld: HEX .BASE resultaat: 16

.BUFS (---)
Dit woord drukt de beginadressen (in HEX) en de inhoud van de buffers af. Bovendien wordt aangegeven of de inhoud gewijzigd is.

.S (---)
Print de inhoud van de parameterstack op het scherm.

.VOC (---)
Print de naam van de CONTEXT vocabulary op het scherm

?KEY (--- f)
Geeft aan of er een toets is ingedrukt. In tegenstelling tot ?TERMINAL die alleen aangeeft of de STOP-toets is ingedrukt.Dit woord loopt via de vector V?KEY (zie systeemvariabelen).

(?KEY) (--- f)
Standaard routine waar V?KEY naar wijst. (Zie ?KEY).

(EMIT) (n ---)
Standaard routine waar VEMIT naar wijst. (Zie EMIT)

(KEY) (--- n)
Standaard routine waar VKEY naar wijst. (Zie KEY)

(PR) (n --- f)
Dit woord stuurt het karakter n zonder vertaling naar de printer.Het woord returnt met een vlag die aangeeft of de printer aan staat of niet. TRUE betekent dat de printer niet aan staat.

2DROP (d1 ---)
Double precision versie van DROP.Verwijdert de bovenste 2 enkele precisie getallen van de stack.

2OVER (d1 d2 --- d1 d2 d1
Double precision versie van OVER

2ROT (d1 d2 d3 --- d2 d3 d1)
Double precision versie van ROT

2SWAP (d1 d2 --- d2 d1)
Verwisseling van de bovenste twee double precision getallen van de stack.

ASCII (--- n)
De ASCII-waarde van het volgende teken in de input wordt op de stack gezet. Dit werkt zowel in directe stand als tijdens compileren.
Voorbeeld: ASCII G . resultaat: 72
: test KEY ASCII G = IF 7 EMIT THEN ;

BLOCK (n1 --- n2)
Dit woord geeft op de stack het beginadres n2 van screen n1. Voor FORTH 3.0 is de betekenis lichtelijk gewijzigd. Indien n1 kleiner is dan hexadecimaal 3000 geeft BLOCK het beginadres van screen n1. Indien n1 > 3000H geeft block gewoon de waarde n1 terug. Hiermee is het mogelijk om een eigen applicatie in rom te zetten als tekst, en uit te voeren als ware het een cassette. Op deze manier is bv. de screeneditor in de rom opgenomen. Hij kan geladen worden dmv. HEX 4000 LOAD.

BREAK (---)
Door dit woord op te nemen in een definitie kunt u op elk moment terugkeren naar FORTH. Op het moment van de break wordt geprint in welk woord het plaatsvindt en wat de inhoud van de stack is. Nu kunt u alle FORTH woorden gebruiken, ahw tijdens de afwerking van het programma. Als u geen fouten maakt, kunt u de uitvoer hervatten met de opdracht RESUME.
Voorbeeld: : TEST 1 2 3 BREAK . . . ;

CASE (n ---)
Begin van de controle struktuur CASE. De overige elementen van die struktuur worden ook hier behandeld.Het getal n op de stack is de zog. case-selector.
De gehele struktuur ziet er als volgt uit:

```
CASE
   3  OF ... ENDOF ( ... wordt uitgevoerd indien sel. = 3)
   6  OF ... ENDOF ( ... wordt uitgevoerd indien sel. = 6)
  17  OF ... ENDOF ( ... wordt uitgevoerd indien sel. = 17)
   ... ( wordt uitgevoerd in alle andere gevallen )
ENDCASE
```

De gedeelten aangeduid met ... kunnen willekeurige FORTH opdrachten zijn. CASE mag alleen voorkomen in een ":"-definitie. Het gedeelte na de laatste ENDOF kan eventueel leeg zijn. ENDCASE verwijdert de selector van de stack. ENDOF en OF laten de stack ongemoeid. Deze casestruktuur kan ook gecombineerd worden met een IF THEN indien bv. een reeks waarden getest moet worden.
Voorbeeld:

```
: Test KEY
     CASE
        65 OF ." A" ENDOF
        67 OF ." C" ENDOF
        ." Toets A of C "
     ENDCASE ;
```

CLEAR (n1 ---)
Wist de inhoud van screen n1. Bovendien wordt n1 geselekteerd voor editen. (Var. SCR)

CLS (
Wist het scherm.

COPY (n1 n2 ---)
Verandert het screennr. n1 in n2 waardoor dit na FLUSH op een andere plaats op de cassette komt. Dit is dus een screencopy.

DEPTH (--- n)
Geeft het aantal getallen op de stack

DUMP (un1,n2 ---)
Hiermee kunt u een stuk geheugen dumpen op het scherm in de vorm van hexadecimale codes. Bovendien wordt voor elk byte de ASCII representatie gegeven indien mogelijk. Un1 is het beginadres en n2 is het aantal te dumpen bytes.
Voorbeeld: HEX 6200 B0 DUMP

EDITOR

Er is een vocabulaire EDITOR ingebouwd met een zeer eenvoudige regeleditor. Daarnaast is er als applicatie ook een screeneditor aanwezig.(Zie screeneditor)
De eenvoudige regeleditor is te gebruiken na het intypen van EDITOR, om de vocabulaire te activeren. Er zijn de volgende commando's ingebouwd:

n LIST	List screen n, en activeert het voor editen.
LL	List het huidige screen.
LP	List het vorige screen.
LN	List het volgende screen.
n CLEAR	Maakt screen n schoon.
n1 n2 COPY	Veranderd in het geheugen screen n1 in n2.
FLUSH	Schrijft de gewijzigde screens naar de cassette.Na afloop zijn de buffers leeg.
SAVE	Schrijft het huidige screen tussentijds naar cassette, maar het blijft in de buffer staan.
n LINE	Geeft op de stack het beginadres en de lengte (64) van regel n in het huidige screen.
n PL tekst	plaatst de achter PL volgende tekst op regel n
n EL	wist regel n
n IL tekst	Voegt de tekst achter IL in op regel n. Alle regels eronder schuiven omlaag. Regel 15 verdwijnt hierbij.
n HL	Plaatst de regel n in een buffer (PAD)
n RL	Plaatst de regel in de buffer op regel n. HL en RL kunnen dus gebruikt worden op regels van het ene naar het andere screen over te brengen.
n1 n2 CL	Copieert regel n1 naar regel n2.
n DL	Verwijdert regel n en plaatst die in PAD Alle regels eronder schuiven omhoog. Regel 15 wordt hierbij leeg gemaakt.

Alle regelnummers liggen tussen 0 en 15.

EMIT (n ---)
Het karakter n wordt op het scherm gezet, of de funktie van de control-code n wordt uitgevoerd.EMIT kent de volgende control-codes:

1	Zet de cursor en de horizontale scroll aan
2	Zet de cursor en de horizontale scroll uit
5	Schakelt de printer erbij aan of weer uit. Hierbij wordt alle output naar het scherm ook naar de printer gestuurd. (per regel met vertaling)
7	geeft een piepje via de luidspreker.

```
  8   Wist het karakter links van de cursor en zet de cursor
      een plaats terug
 10   Linefeed plus carriage-return. Dus overgang naar het
      begin van een nieuwe regel.
 11   Inverse scroll. Het hele scherm zakt een regel omlaag,
      de bovenste regel wordt gewist en de cursor gaat naar de
      homepositie (linksboven).
 12   Wist het scherm en zet de cursor op home positie.
 13   Cursor naar begin van de regel.
 14   Cursor naar het einde van de regel.
 15   Wist de regel vanaf cursorpos. tot einde regel.
 16   Cursor een plaats naar links (met wraparound)
 17   Cursor een plaats omhoog (met wraparound)
 18   Cursor een plaats omlaag (met wraparound)
 19   Cursor een plaats naar rechts (met wraparound)
 30   Cursor naar homepositie (linksboven)
 31   Schakelen hoofdletters/kleine letters en weer terug.
```

Karaktercodes groter dan 128 worden als kleur weergegeven. Bv. 129 is alfanumeriek rood, 130 is alfanum. groen etc.
De snelheid van het printen naar het scherm is te vertragen door het indrukken van de hoofdlettertoets. (Variable SPEED).
Met shift 5 (rechter bordje) wordt de output stopgezet en met elke toets daarna weer hervat.
EMIT loopt via een vector (VEMIT). De karakters worden voor output vertaald naar andere codes dmv een vertaaltabel. (Variabelen STRT en STRP).

ENDCASE
Zie CASE.

ENDOF
Zie CASE

EXIT (---)
Het woord dat in uitvoering is wordt hierdoor onmiddelijk verlaten.

FORMAT-TAPE (---)
Deze opdracht wist, na bevestiging, de cassette die in de recorder zit en brengt hierop genummerde lege screens aan.
Let op!. Alle informatie die op de cassette stond, wordt hierbij gewist.

J (--- n)
Dit woord zet het derde getal van de returnstack op de parameter-stack. Bij twee geneste do-loops is dit de buitenste loopindex indien men het aanroept in de binnenste loop.
Voorbeeld:

```
: tabel 10 0 DO
            11 1 DO
                  I J *
            LOOP
        LOOP
```

KEY (--- n)
Deze opdracht wacht op een toets en zet daarna de ascii code ervan op de stack. Tijdens het wachten op input via EXPECT worden de volgende controltoetsen rechtstreeks uitgevoerd:

```
shift TAB  schakelt hoofdletters/kleine letters en terug.
INL        Zet de cursor aan
```

OPN Zet de cursor uit
enter Sluit de invoer af
rubout Wist het laatste karakter
pijl <-- Toont linkerhelft van het scherm (40 kar.)
pijl --> Toont rechterhelft van het scherm (40 kar.)

KEY loopt via een vector in ram (VKEY). De toetscode die de toetsen afgeven worden eerst vertaald naar ascii-codes dmv. een toetstabel. Daarna worden ze via een vertaaltabel naar de uiteindelijke asciiwaarde omgezet. Hiertoe dienen de systeem-variabelen KEYP, KTRP en KTRT .

LAYOUT (n1 n2 n3 n4 ---)
Met deze opdracht kan de schermlayout gewijzigd worden. De regel- en kolomtelling begint met 0,0 linksboven op het scherm. Het systeem start op met de waarden : 0,0,23,67.

n1 = beginregel
n2 = beginkolom
n3 = eindregel
n4 = eindkolom

De getallen n1 t/m n4 worden gecontroleerd en begrensd op de volgende waarden:

	minimum	maximum
aantal regels:	5	24
aantal kolommen:	5	80
begin-, eindregel:	0	23
begin-, eindkolom:	0	79

Voorbeeld: 10 8 18 35 LAYOUT

LPT (n1 n2 --- f)
Equivalent van TYPE maar de output gaat naar de printer. Vanaf adres n1 worden n2 karakters zonder vertaling naar de printer gestuurd. Het woord returnt met een vlag TRUE als de printer niet aan staat, anders met FALSE.

OF ---)
Zie CASE

P@ (n1 --- n2)
Input van poort n1 wordt als n2 op de stack gezet.

P! (n1 n2 ---)
Het byte n1 wordt naar poort n2 gestuurd.
Voorbeeld: 1 0 P! zet de 80-karakterkaart aan en
0 0 P! zet hem weer uit.

PICK (n --- n)
Dit woord kopieert het n-de getal van de stack naar de top van de stack. Bv. 2 PICK is hetzelfde als OVER, en 1 PICK is hetzelfde als DUP.

RESET (---)
Dit woord reset de tape-administratie. Indien u een nieuwe cassette plaatst moet! het ingetypt worden. Het heeft nl. tot gevolg dat nieuwe cassette-operaties beginnen met lezen op welk blok de cassette staat. Bovendien wordt na cassette-fouten automatisch een RESET uitgevoerd. Door het virtuele geheugenbeheer (cassette in blokken) is het noodzakelijk om aan te geven wanneer er een cassette gewisseld is. Als u dat niet doet is er een grote kans dat blokken op de verkeerde plaats op de cassette teruggeschreven worden.

REWIND (---)
Spoel de cassette terug naar het begin. Na afloop weet de P2000 dan dat de cassette op blok 0 staat. Systeemvariabele NXBL .

RESUME (---)
Hervat uitvoering woord na onderbreking dmv BREAK.

ROLL (n --- n)
Roteert met n-de getal van de stack naar boven. 3 ROLL komt dus overeen met ROT.

SAVE (---)
Bewaart het huidige screen, zoals aangegeven door de variabele SCR op de cassette zonder de buffer leeg te maken. Kan gebruikt worden om tussentijds backups naar cassette te maken. Na SAVE wordt de update-vlag van de buffer gereset.

SHL (n1 n2 --- n3)
Schuift het getal n1, n2 bits naar links en zet het resultaat als n3 op de stack. Voorbeeld: 3 2 SHL geeft 12 op de stack.

SHR (n1 n2 --- n3)
Schuift het getal n1, n2 bits naar rechts en zet het resultaat als n3 op de stack. Voorbeeld: 12 2 SHR geeft 3 op de stack.

STRING (n ---)
Definierend woord dat een string aanmaakt met maximale lengte n. Voorbeeld: 10 STRING JAN maakt een string aan van max. 10 tekens genaamd JAN. Daarna geeft JAN het startadres van de string en de lengte op de stack.
Voorbeeld: 10 STRING JAN
JAN EXPECT (input tekst in string JAN)
JAN TYPE (print inhoud van JAN op scherm)

TR/W (n1 n2 f ---)
Equivalent van R/W maar dan voor de cassette. De parameters zijn: n1 = startadres van het blok
n2 = screennr. (positionering cassette)
f = vlag, 0 voor wegschrijven en 1 voor lezen.

VECTOR (n ---)
Definierend woord dan een eendimensionaal array (vector) aanmaakt, met een opgegeven max. lengte n.
Voorbeeld: 20 VECTOR TABEL maakt een tabel van max. 20 getallen aan. Daarna geeft bv. 3 TABEL het adres van het 3e element. Indien de index groter is dan het maximum wordt het adres van het laatste element teruggegeven. Bv. 30 TABEL zou dus het adres van element 20 geven.

XY (n1 n2 ---)
Deze opdracht zet de cursor op de aangegeven waarde op het scherm.Hierbij is n1 de regel en n2 de kolom. De telling begint met 0,0 linksboven op het scherm. De regel en kolom worden gecontroleerd op toegelaten waarden, die eventueel dmv LAYOUT gewijzigd kunnen zijn.

YESNO (--- f)
Dit woord wacht op een toets. Als de toets overeenkomt met hoofdletter J of kleine j, dan wordt TRUE op de stack gezet, anders FALSE.

Screen-editor

In de module is op adres hex. 4000 een screen-editor opgenomen. Deze editor staat gewoon in FORTH-code en is dus niet gecompileerd. De tekst van de editor staat op de volgende pagina's (screen 1 t/m 7). Deze tekst is eerst gecomprimeerd dmv. het woord COMPRESS (zie eerste screen op volg. pag.), en daarna in rom geplaatst.

De editor moet eenmalig geactiveerd worden dmv. de opdracht: HEX 4000 LOAD. Daardoor wordt ie gecompileerd en in de dictionaire gezet. (dit duurt eventjes).
De editor is bedoeld om tegelijkertijd te werken op alle screens van een programma. Dit kunnen echter niet meer screens zijn dan #BUF-1, anders werkt het niet efficient.
Dus voor 16k max. 5, 32K max. 9, 48k en groter max. 15.
De screens vormen bij elkaar een soort file die in z'n geheel bewerkt kan worden.

U roept de editor aan met: n1 n2 EDIT
waarbij n1 het eerste screennr is en n2 het laatste van uw programma. n1 mag gelijk zijn aan n2.
Nu worden eerst de screens van cassette ingelezen.
Daarna ziet u op het scherm de eerste 16 regels (screen n1).
Bij elke eerste regel van het scherm, staat een * achter het regelnummer. (ivm doorcompileren met --> en commentaar in de eerste screenregels)

Bij het intypen van karakters worden ze automatisch ingevoegd in de tekst. De tekst voorbij kolom 64 verdwijnt dus.
Geef, indien u een 80-kar. kaart heeft eerst 1 0 P! , dat werkt veel overzichtelijker, omdat de hele tekst zichtbaar blijft.
Nu kunt u de volgende toetsen gebruiken:

rubout (boven enter	wis karakter links van cursor
shift rubout	wis karakter onder cursor
wis-regel	wis karakers onder en rechts van cursor
pijltjes-toetsen	verplaats cursor
shift pijl omlaag	ga 15 regels verder
na CODE	ga naar einde van de file
shift pijl omhoog	ga 15 regels terug
na CODE	ga naar begin van de file
shift pijl rechts	ga naar einde van de regel
shift pijl links	ga naar begin van de regel
X rechtsboven	voeg lege regel in, de laatste vervalt
: rechtsboven	verwijder de regel, hierbij wordt de laatste regel leeg
OPN en 1 t/m 6	zet de regel van de cursor in buffer 1 t/m 6 (onderin het scherm)
INL en 1 t/m 6	plaats buffer 1 t/m 6 op de cursorlijn
TAB	ga naar eerstvolgende TAB-positie, deze staan om de 4 kolommen.
shift TAB	omschakeling hoofdletters naar hoofdletters en kleine letters
STOP	stoppen met editen.

Na het editen moet men nog FLUSH ingeven indien de gewijzigde tekst naar de cassette geschreven moet worden.
De screen-editor uit het geheugen verwijderen, als hij niet meer nodig is, met de opdracht: FORTH FORGET TASK.

```
2 LIST
Scr # 2
  0                          ( COMPRESSIE VAN SCREENS   FLM 1-9-1986)
  1 DECIMAL           0 VARIABLE TOSC    0 VARIABLE FSC
  2 0 VARIABLE MEM    0 VARIABLE INK     0 VARIABLE MAXK
  3 : INC 1 INK  +! ;
  4 : ?KLAAR INK @ MAXK @ = ;
  5 : STORE DUP EMIT MEM @ 1018 /MOD TOSC @ + BLOCK + C!
  6       1 MEM +! UPDATE ; ( plaats voor --> )
  7 : INV TOSC ! OVER - 1+ 1024 * MAXK ! FSC ! 0 DUP INK ! MEM !
  8 : NKAR INK @ 1024 /MOD FSC @ + BLOCK + C@ ;
  9 : TO)   BEGIN INC NKAR 41 = UNTIL INC ;  ( comment)
 10 : TONSP BEGIN INC NKAR 32 = NOT ?KLAAR OR UNTIL ; ( spaties)
 11 : COMPRESS ( FirstScr LastScr ToScr  --- )  CR INV
 12     BEGIN NKAR DUP 40 = IF TO)
 13       ELSE DUP 32 = IF STORE TONSP ELSE STORE INC THEN THEN
 14     ?KLAAR UNTIL CR CR
 15    ." LENGTE = " MEM @ U. CR ;
 Ok
  Ok
  Ok
  Ok
1 LIST
Scr # 1
  0 ( SCREENEDITOR - FILES       FVDM 860811 )
  1 ( CR ." Screeneditor - 1 "  )
  2 : TASK ;  DECIMAL     16 CONSTANT L/S
  3 0 VARIABLE LC         0 VARIABLE CC
  4 0 VARIABLE L1         0 VARIABLE LMAX
  5 0 VARIABLE N1         C/L 1 - CONSTANT CCM
  6 : LA ( LINENR --- ADRES ) L/S /MOD N1 @ + BLOCK
  7     SWAP C/L * + ;
  8 : LL ( LINENR --- ) ( LIST LINE )
  9   DUP 3 .R DUP L/S MOD IF SPACE ELSE 42 EMIT THEN
 10     LA C/L TYPE ;
 11 : OLDPOS LC @ L1 @ - CC @ 4 + XY ;
 12 : SHOW 2 EMIT 30 EMIT L1 @ DUP 15 + DUP ROT
 13     DO I LL CR LOOP LL OLDPOS ;
 14
 15 ( -->)
 Ok
2 LIST
Scr # 2
  0 ( SCREENEDITOR - FILES       FVDM 860811 )
  1 ( CR ." Screeneditor - 2 " )   HEX
  2 : CURY 38 +ORIGIN C@ ;
  3 : CURAD LC @ LA CC @ + ;
  4 : UP LC @ L1 @ < IF 0B EMIT LC @ DUP LL L1 ! THEN ;
  5 : DOWN LC @ 0F - DUP L1 @ > IF 0A EMIT L1 ! LC @ LL
  6       ELSE DROP THEN ;
  7 : !C ( BYTE --- ) CURAD C! UPDATE ;
  8 : +L LC @ LMAX @ < IF 1 LC +! DOWN THEN ;
  9 : -L LC @ IF -1 LC +! UP THEN ;
 10 : +C CC @ CCM < IF 1 CC +! ELSE +L 0 CC ! THEN ;
 11 : -C CC @ IF -1 CC +! ELSE CCM CC ! -L THEN ;
 12 : RET 0 CC ! +L ;
 13 : TAB CC @ 4 / 1+ 4 * CCM MIN CC ! ;
 14 : EOL CCM CC ! ;
 15 ( -->)
 Ok
```

```
3 LIST
Scr # 3
  0 ( SCREENEDITOR - FILES       FVDM 860811 )
  1 ( CR ." Screeneditor - 3 " )
  2 : PADR ( REGEL --- SCREENADRES )
  3     11 + 50 * 5004 + ;
  4 : SCADR ( REGEL --- SCHERMADRES )
  5     CURY 50 * 5004 + ;
  6 DECIMAL
  7 : WHICH KEY 127 AND 48 - DUP DUP 0> SWAP 7 < AND ;
  8 : LCAD LC @ LA ;
  9 : REST CCM CC @ - ;
 10 : OPN WHICH IF PADR DUP LCAD SWAP C/L CMOVE
 11      SCADR SWAP C/L CMOVE ELSE DROP 7 EMIT THEN ;
 12 : INL WHICH IF PADR DUP LCAD C/L CMOVE UPDATE
 13      SCADR C/L CMOVE ELSE DROP 7 EMIT THEN ;
 14 : WIS CURAD REST BLANKS UPDATE 15 EMIT ;
 15 ( --> )
 Ok
4 LIST
Scr # 4
  0 ( SCREENEDITOR - FILES       FVDM 860811 )
  1 : LM LMAX @ 15 - ; ( CR ." Screeneditor - 4 " )
  2 : INSERT LMAX @ BEGIN DUP LC @ > WHILE
  3      DUP LA OVER 1- LA SWAP C/L CMOVE UPDATE 1- REPEAT
  4      DROP 0 CC ! WIS SHOW ;
  5 : DELETE LC @ BEGIN DUP LMAX @ < WHILE
  6      DUP LA OVER 1+ LA SWAP C/L CMOVE UPDATE 1+ REPEAT
  7      DROP LMAX @ LA C/L BLANKS UPDATE SHOW ;
  8 : CENTER LC @ 8 - 0 MAX LM MIN L1 ! SHOW ;
  9 : SBACK LC @ DUP 15 > IF 15 - LC ! CENTER ELSE
 10      DROP 0 LC ! 0 L1 ! SHOW THEN ;
 11 : SFORW LC @ DUP LM < IF 15 + LC ! CENTER
 12      ELSE DROP LMAX @ DUP LC ! 15 - L1 ! SHOW THEN ;
 13 : LLC 13 EMIT LC @ LL ;
 14 : TOETS OLDPOS 1 EMIT KEY 2 EMIT ;
 15 ( --> )
 Ok
5 LIST
Scr # 5
  0 ( SCREENEDITOR - FILES       FVDM 860811 )
  1 ( CR ." Screeneditor - 5 " )
  2 : INSK ( KAR --- ) CURAD DUP 1+ REST
  3     CMOVE !C LLC +C OLDPOS ;
  4 : DELK CURAD DUP 1+ SWAP REST CMOVE
  5   32 LCAD CCM + C! UPDATE LLC OLDPOS ;
  6 : HOME 0 L1 ! 0 LC ! 0 CC ! ;
  7 : BOTTOM LMAX @ DUP 15 - L1 ! LC ! 0 CC ! ;
  8 : GETSC 2DUP 1+ SWAP DO I BLOCK DROP LOOP ;
  9 : INIT GETSC OVER - 1+ L/S * 1- LMAX ! N1 !
 10    CLS 7 1 DO I 17 + 2 XY I 48 + EMIT LOOP
 11    16 0 XY 68 0 DO 45 EMIT LOOP
 12    0 0 15 68 LAYOUT 0 59 +ORIGIN C! HOME SHOW ;
 13 : ENDEDIT 32 59 +ORIGIN C! 0 0 23 67 LAYOUT 1 EMIT ;
 14 ( --> )
 15
 Ok
```

```
6 LIST
Scr # 6
  0 ( SCREENEDITOR - FILES       FVDM 860811 )
  1 ( CR ." Screeneditor - 6 " )
  2
  3 : CODE KEY CASE 11 OF BOTTOM SHOW ENDOF
  4      30 OF HOME SHOW ENDOF 7 EMIT ENDCASE ;
  5
  6 : TAB1 LC @ DUP IF 1- LA CC @ + CCM CC @ - 1 DO
  7         1+ DUP C@ 32 = NOT IF CC @ I + CC ! LEAVE THEN
  8         LOOP DROP ELSE DROP THEN ;
  9
 10
 11 ( --> )
 12
 13
 14
 15
 Ok
7 LIST
Scr # 7
  0 ( SCREENEDITOR - FILES       FVDM 860811 )
  1 ( CR ." Screeneditor - 7 " )
  2 : EDIT ( N1 N2 --- ) INIT BEGIN TOETS DUP DUP DUP
  3    31 > SWAP 127 < AND IF INSK ELSE
  4    CASE 16 OF -C ENDOF              17 OF -L ENDOF
  5         18 OF +L ENDOF              19 OF +C ENDOF
  6         13 OF 0 CC ! ENDOF          10 OF RET ENDOF
  7          8 OF -C DELK ENDOF        136 OF DELK DROP 0 ENDOF
  8          1 OF INL ENDOF             2 OF OPN ENDOF
  9          9 OF TAB ENDOF            14 OF EOL ENDOF
 10        170 OF INSERT DROP 0 ENDOF   27 OF CODE ENDOF
 11        175 OF DELETE DROP 0 ENDOF
 12         11 OF SFORW ENDOF           30 OF SBACK ENDOF
 13         15 OF WIS ENDOF             31 OF DUP EMIT ENDOF
 14    ENDCASE ENDIF
 15    03 = UNTIL ENDEDIT ;    ;S
 Ok
```

Systeemvariabelen:

In FORTH 3.0 worden de volgende systeemadressen gebruikt

HEX	naam	+ORIGIN	beschrijving
103C	DPUSH	----	Label 2 voor NEXT, push DE
103D	HPUSH	----	Label voor NEXT, push HL
103E	NEXT	----	Startadres van de inner-interpreter. NEXT springt eerst naar ram (RNEXT) en dan weer terug in de rom.
6200	ORIGIN	0	Start van de FORTH Origin tabel Jump naar COLD-start.
6204		4	Jump naar WARM-start.
6208		8	Fig.release en versie.
620C		12	Top NFA van FORTH dictionaire.
6210		16	Beginadres van uservariabelen.
6212		18	Beginadres van parameterstack.
6214		20	Beginadres van returnstack.
6216		22	Beginadres van inputbuffer.
6218		24	Initiele waarde van WIDTH.
621A		26	Initiele waarde van WARNING.
621C		28	Initiele waarde van FENCE.
621E		30	Initiele waarde van DP
6220		32	Initiele waarde van VOC-LINK.
6224		36	Initiele waarde van #BUF
6226		38	Initiele waarde van FIRST.
6228		40	Initiele waarde van LIMIT.
6230	TOPS	48	Beginadres van schermvenster.
6232	CURPOS	50	Laatste cursorpositie.
6234	XYST	52	Diverse statusvlaggen voor scherm.
6235	MAXC	53	Max. aantal kolommen.
6236	MAXL	54	Max. aantal regels.
6237	CURX	55	X-positie van cursor.
6238	CURY	56	Y-positie van cursor.
6239	KAR	57	Karakterbuffer voor EMIT.
623A	HOFF	58	Horizontale offset (EMIT).
623B	SPEED	59	Snelheid printen bij indrukken shift.
623C	RPP	60	Returnstack-pointer.
623E	UP	62	Pointer naar tabel van uservariabelen.
6240	SAVS	64	Bewaarplaats stackpointer.
6242	INHK	66	Deze variabele bepaalt het gedrag bij een P2000-reset. (Softreset)
6244	SPB	68	Bodem van de parameter-stack.
6246	KEYP	70	Pointer naar toets-tabel.
6248	NXBL	72	Next cassette-blok.
624A	RNEXT	74	Bevat sprong naar rom inner-interpreter. (Hook voor NEXT)
624D	BIP	77	Breekpunt voor IP, indien RNEXT gepatched is naar JP NEXT+1
624F	LATIP	79	Laatst gepasseerde IP, voor debugging.
6251	MONJ	81	Jump naar monitor, wordt geactiveerd als IP (reg. BC) gelijk wordt aan BIP. Is in deze versie niet ingevuld, omdat er geen debug-monitor aanwezig is.
6254	RUSE	84	Volgende, te gebruiken, buffer.
6256	RPREV	86	Laatst gebruikte buffer.

6258	USAB	88	Adres van user-abort routine. ABORT gebruikt deze variabele als vector. Standaard is hier NOOP ingevuld, maar elk FORTH-woord kan gebruikt worden.
	BOOTFL	90	Bootflag die aangeeft of er een boot van cassette (1 LOAD) moet gebeuren. Dit wordt gebruikt bij koude start en na indrukken van de reset-knop. Het booten gebeurt in het woord ABORT.
	CHECKB	92	Controle van returnstack-pointer. Wordt gebruikt in BREAK en RESUME.
625E	KTRP	94	Pointer naar toets-vertaaltabel.
6260	STRP	96	Pointer naar scherm-vertaaltabel.
6262	VEMIT	98	Pointer naar EMIT-routine. Wijst standaard naar (EMIT).
6264	VKEY	100	Pointer naar KEY-routine. Wijst standaard naar (KEY).
6266	VQKEY	102	Pointer naar ?KEY-routine. Wijst standaard naar (?KEY).
6268	VR/W	104	Pointer naar R/W-routine. Wijst standaard naar TR/W

Door de pointers VEMIT,VKEY,VQKEY en VR/W kan in principe alle I/O van het systeem gewijzigd worden. Bv. in een communicatie-programma zou VQKEY kunnen wijzen naar een routine, die zowel het toetsenbord aftast, als wel kijkt of er karakters binnen-komen.
De vector KTRP, geeft de mogelijkheid om naast de gewone toets-tabel (KEYP), enkele toetsen een andere code te laten afgeven.
Voor alle vertaaltabellen geldt dezelfde struktuur:
- aantal te vertalen codes
- oorspronkelijke code, nieuwe code
- oorspronkelijke code, nieuwe code
- etc. etc.

De tabel heeft default een lengte 17, plaats voor 8 vertalingen, maar kan op een andere plaats gezet worden door de pointer aan te passen.

Doordat de inner-interpreter eerst naar ram springt, is het altijd mogelijk iets te wijzigen of te bekijken tijdens de doorgang van NEXT. Ingebouwd is een mogelijkheid om de instruktie-pointer (IP = register BC) te vergelijken met een opgegeven waarde in BIP. Daartoe moet de jump op RNEXT veranderd worden naar JP NEXT+1. Als de vergelijking klopt wordt MONJ uitgevoerd. Hier moet dan een sprong naar een debugger of iets dergelijks gezet worden.
Indien men woorden in machine-code wil definieren, moet men er rekening mee houden dat register BC in geen geval gewijzigd mag worden. (het is de IP).
Voorbeeld FORTH en machinecode:

```
 : ADD + ;       ( n1 n2 --- n1+n2 )
  of
HEX CREATE ADD
        E1 C,   ( POP HL , HL = n1)
        D1 C,   ( POP DE , DE = n2 )
        19 C,   ( ADD HL,DE )
        C3 C, HPUSH ,   ( JP HPUSH, zet resultaat op stack en
                                    voert daarna NEXT uit)
     SMUDGE     ( maak nieuwe woord toegankelijk in dictionaire).
```

Geheugenindeling.

P2000 FORTH 3.0 gaat bij een koude start automatisch kijken hoeveel geheugen er aanwezig is. Aan de hand van het geheugen wordt bepaald hoeveel blokbuffers er gereserveerd moeten worden De buffers komen helemaal boven in het geheugen te staan. Achtereenvolgens ziet de geheugenmap er zo uit:

hex.	
1000-38FF	Forthcode in ROM
3900-3FFF	Vrije ruimte voor applicatie, als sourcetekst in rom.
4000-4AFF	Screeneditor in sourcevorm
4B00-4FFF	Nog vrij voor uitbreidingen.
5000-5780	Videoram
6000-61FF	Systeemgeheugen P2000 (monitor) 6070-61FF is niet gebruikt.
6200-622F	Origin-table van FORTH
6230-6269	Systeemvariabelen van FORTH 3.0
626A-627A	Keyboard vertaaltabel (KTRT)
627B-628B	Scherm vertaaltabel (STRT)
628C-629C	Printer vertaaltabel (PTRT)
629D-62FC	Messagetabel, bestaande uit de beginadressen van 48 boodschappen in de rom.
62FD	Begin van de terminal inputbuffer (TIB)
63C4	Einde van de returnstack.
63C5-63F4	Tabel van de user-variabelen.
63F5-6407	FORTH woord in ram.
640B	Start van de dictionaire.

De top van het geheugen is afhankelijk van het type P2000.

	16K	32K	48K en meer
Top van de stack:	87E7	B7D7	BFBF
Einde geheugen:	9FFF	DFFF	FFFF
Aantal buffers:	6	10	16
FIRST:	87E8	B7D8	BFC0
LIMIT:	A000	E000	0000

Hoewel dit niet ingebouwd is, is het relatief eenvoudig om het aantal buffers te wijzigen. (Variabelen #BUF,FIRST,LIMIT,SP0

In de ruimte HEX 3900-4FFF kan men eventueel eigen applicaties maken. Op het ogenblik is een gedeelte ervan gebruikt voor een screeneditor. Deze heeft men bij toepassingen meestal niet nodig, zodat er bijna 6K ruimte is. Deze applicaties kunnen gewoon uit FORTH bestaan. Om ruimte te winnen , heb ik een programma beschreven (COMPRESS) die uit een aantal FORTH-screens alle niet noodzakelijke tekst, zoals spaties en commentaar, verwijdert. Op die manier is de screeneditor, die bestaat uit 7 FORTH screens, teruggebracht tot minder dan 3kB. Om zo'n applicatie te starten geeft men het commando: Bv. HEX 4000 LOAD. Daarna komt de appl. gewoon in ram te staan, zodat er ook geen moeilijkheden zijn met variabelen en dergelijke. Eventueel kan men een patch aanbrengen in (ABORT) zodat de applicatie automatisch start na powerup of reset.

ALL
Alle definities in FORTH 3.0

!CSP	!	#S	#
#>	#BUF	'	(LINE)
	(?KEY)	(KEY)	(EMIT)
(ABORT	(NUMBER)	(.")	(;CODE)
(PR)	(FIND)	(DO)	(+LOOP)
(LOOP)	*/	*/MOD	*
+LOOP	+BUF	+-	+ORIGIN
+!	+	,	-->
-FIND	-TRAILING	-DUP	-
.VOC	.BUFS	.BASE	.S
.LINE	.	.R	."
/	/MOD	0	0>
0<	0=	0BRANCH	1-
1+	1	2-	2+
2	2!	2@	2ROT
2DUP	2SWAP	2DROP	2OVER
3	:	;CODE	
5	<#	<BUILDS	<
=	>	>R	?
?TERMINAL	?KEY	?STACK	?LOADING
?CSP	?EXEC	?PAIRS	?COMP
?ERROR	@	ASCII	AGAIN
ABS	ABORT	ALLOT	AND
BREAK	BEGIN	BLOCK	BUFFER
BLANKS	BASE	BLK	B/SCR
B/BUF	BL	BRANCH	CASE
CLS	CR	COLD	CREATE
COUNT	COMPILE	C,	CSP
CURRENT	CONTEXT	C/L	CONSTANT
CFA	C!	C@	CMOVE
DUMP	DO	D.	D.R
DR0	DABS	D+-	DEFINITIONS
DLITERAL	DOES>	DECIMAL	DPL
DP	DEPTH	DUP	DROP
DMINUS	D+	DIGIT	EDITOR
ENDCASE	ENDOF	ELSE	END
ENDIF	EMIT	EMPTY-BUFFERS	ERROR
ERASE	EXPECT	EXIT	ENCLOSE
EXECUTE	FORGET	FORMAT-TAPE	FLUSH
FENCE	FILL	FIRST	HOLD
HEX	HERE	HLD	IF
INDEX	IMMEDIATE	INTERPRET	ID.
IN	I	J	KEY
LAYOUT	LOOP	LIST	LOAD
LITERAL	LATEST	LPT	LFA
LIMIT	LEAVE	LIT	MESSAGE
M/MOD	MOD	M/	M*
MIN	MAX	MINUS	NUMBER
NOOP	NFA	NOT	OF
OFFSET	OUT	OVER	OR
PAD	PFA	P!	P@
PREV	PICK	QUIT	QUERY
RESUME	REPEAT	REWIND	ROLL
R/W	RESET	R#	R0
ROT	R	R>	RP!
RP@	STRING	SIGN	SPACES
S->D	SMUDGE	SPACE	STATE
SCR	S0	SHR	SHL

```
SWAP            SP!             SP@             THEN
TRIAD           TR/W            TYPE            TIB
TOGGLE          UNTIL           UPDATE          U.
USER            USE             U<              U/
U*              VLIST           VECTOR          VOCABULARY
VOC-LINK        VARIABLE        WHILE           WARM
WORD            WARNING         WIDTH           XOR
XY              YESNO           [COMPILE]       [
]
 Ok
  Ok
  Ok
  Ok
1 LIST
Scr # 1
  0 ( BREDE LIJST VAN DEFINITIES      FVDM  26-10-1986)
  1   0 VARIABLE NL
  2 : CRL CR 0 OUT ! 1 NL +! NL @ 60 > IF CR CR CR
  3       CR CR 0 NL ! 0 OUT ! THEN ;
  4 : PNAME ID. SPACE OUT @ 15 MOD -DUP IF 15 SWAP - SPACES THEN ;
  5 : ?NEW OUT @ 55 > IF CRL THEN ;
  6 : ?SAME ( letter NFA --- letter NFA 0/1)
  7    2DUP 1+ C@ 127 AND = ;
  8 : LIJST ' VLIST NFA BEGIN
  9      ?NEW ?SAME IF DUP PNAME THEN
 10      PFA LFA @ DUP 0= UNTIL DROP ;
 11 : ALL 0 NL ! CRL
 12       ." Alle definities in FORTH 3.0 " CRL CRL
 13          97 32 DO I LIJST DROP LOOP CR ;
 14  ;S
 15
 Ok
```

Literatuurlijst:
---

1. "FORTH een taal voor programmeurs."
   E.Floegel. (o met duitse umlaut)
   Kluwer Technische Boeken BV.
   Deventer-Antwerpen 1985.
   Uitstekende introduktie in FORTH en bovendien zeer goed
   overeenstemmend met FORTH 3.0

2. "FORTH Ok "
   drs. F.J.Meijer
   Wolfkamp , Amsterdam, 1982.
   Eenvoudige introduktie in FORTH

3. "Flitsend FORTH "
   Alan Winfield.
   Academic service, Den Haag, 1983.
   Gedegen boek over FORTH 79 standaard, dat
   iets afwijkt van de FIG-FORTH die in deze
   module gebruikt wordt.

4. "Discover FORTH"
   Thom Hogan.
   Osborne/ Mc Graw-Hill, Berkeley, 1982.
   Bespreekt zowel FIG-FORTH als FORTH-79, maar heeft
   helaas een afwijkende stack-notatie.

5. "Starting FORTH".
   Leo Brodie, Prentice Hall, Englewood Cliffs, 1981.
   Het boek over FORTH. In een humoristische trant, en
   met veel tekeningen. Bevat enkele zeer slimme toepassingen.
   Voorwoord van Ch. Moore, de uitvinder van FORTH.

6. "Het Vijgeblad."
   Periodieke uitgave van de HCC Forth Interesse Groep
   Contactadres: Hans Nieuwenhuijzen
                 Grunoplantsoen 10
                 3981 BT Bunnik (overgenomen uit HCC-blad)

   "FORTH Dimensions."
   Periodieke uitgave van de Amerikaanse FIG (Forth interest
   group). Verkrijgbaar via HCC Forth GG. (zie 6.)
   Bevat een schat aan toepassingen en ideeen , maar richt
   zich toch vooral op de gevorderde FORTH-gebruiker.

8. Fig-FORTH installation manual, glossary, model, editor.
   Uitgebreide documentatie van het Fig-FORTH systeem, voor
   wie er echt alles van wil weten.
   Ook zijn er listings voor diverse processors.
   Dit manual heeft ten grondslag gelegen aan deze FORTH 3.0
   Verkrijgbaar via HCC Forth GG. (zie 6.)

9. Omzetting van Fig-FORTH naar FORTH-79 en ook de FORTH-79
   standaard zijn verkrijgbaar bij de HCC Forth GG.

Alle definities in FORTH 3.0 CFA's

266D	!CSP	1A5C	!	2D73	#S
2D48	#	2D19	#>	2BC0	#BUFFERS
1B1F	#BUF	2CD4	'	3024	(LINE)
3011	(	2C9D	(?KEY)	2C82	(KEY)
2C47	(EMIT)	2292	(ABORT)	21BF	(NUMBER)
20A6	(.")	1FCD	(;CODE)	1C8B	(PR)
1602	(FIND)	1593	(DO)	1586	(+LOOP)
154E	(LOOP)	2607	*/	25F6	*/MOD
25B7	*	31EF	+LOOP	2744	+BUF
2536	+-	1AE8	+ORIGIN	1A04	+!
185D	+	1E5B	,	2FFB	-->
2266	-FIND	2073	-TRAILING	198D	-DUP
1869	-	3548	.VOC	3468	.BUFS
344E	.BASE	341C	.S	3042	.LINE
2DCD	.	2DBD	.R	20BF	."
25D6	/	25C6	/MOD	1A8B	0
1851	0>	183D	0<	1829	0=
1538	OBRANCH	1B61	1-	1B48	1+
1A97	1	1B6D	2-	1B54	2+
1AA3	2	1A78	2!	1A45	2@
19AE	2ROT	197D	2DUP	195A	2SWAP
193D	2DROP	1910	2OVER	1AAF	3
1CB8	:	1FE3	;CODE	1CE0	;
17BA	;S	2D0A	<#	1FFB	<BUILDS
1885	<	1877	=	376A	>BACKUP
1E7C	>	17E5	>R	2DD9	?
2CBC	?TERMINAL	2CA9	?KEY	23E8	?STACK
1F29	?LOADING	1F0C	?CSP	1EF7	?EXEC
1EE3	?PAIRS	1ECA	?COMP	1EB0	?ERROR
1A29	@	33AF	ASCII	3227	AGAIN
- 2EBE	AUTOREWIND	255A	ABS	24F6	ABORT
1E4F	ALLOT	1724	AND	379A	BACKUP>
3562	BREAK	3192	BEGIN	2E09	BLANKING
27D5	BLOCK	278D	BUFFER	2134	BLANKS
1FB9	BINARY	1E0D	BASE	1DB0	BLK
1ADC	B/SCR	1ACC	B/BUF	1ABC	BL
1523	BRANCH	38E6	CL	37F9	CLEAR
36E2	COPY	329D	CASE	2E39	CLSTIME
2DF4	CLS	2C70	CR	26D0	COLD
233C	CREATE	2032	COUNT	1F43	COMPILE
1E6C	C,	1E21	CSP	1DF6	CURRENT
1DE8	CONTEXT	1D4C	C/L	1D04	CONSTANT
1C58	CFA	1A6B	C!	1A37	C@
1684	CMOVE	38AD	DL	3682	DUMP
31C6	DO	2DAE	D.	2D8B	D.R
- 2AB2	DISK	28ED	DR/W	2569	DABS
2548	D+-	24B5	DEFINITIONS	23CD	DLITERAL
200B	DOES>	1FA4	DECIMAL	1E17	DPL
1D97	DP	19E8	DEPTH	196F	DUP
192F	DROP	18E3	DMINUS	18B1	D+
15D3	DIGIT	3848	EL	36D1	EDITOR
3306	ENDCASE	32DE	ENDOF	326C	ELSE
3219	END	31A4	ENDIF	2C53	EMIT
2774	EMPTY-BUFFERS	22A6	ERROR	2123	ERASE
1BAC	EXPECT	1811	EXIT	164E	ENCLOSE
1513	EXECUTE	63FD	FORTH	3126	FORGET
- 2F60	FORMAT-TAPE	282D	FLUSH	1D8E	FENCE
1C07	FILL	1AFA	FIRST	38FF	HL

```
2143  HOLD          1F8E  HEX           1E3F  HERE
1E34  HLD           3884  IL            3255  IF
3073  INDEX         2B43  INFO          2466  IMMEDIATE
2418  INTERPRET     22DB  ID.           1DB9  IN
15AF  I             15C0  J             2C8D  KEY
382B  LINE          37E5  LP            37D4  LN
37C5  LL            3353  LAYOUT        31D9  LOOP
30D0  LIST          2FC9  LOAD          23B0  LITERAL
1E9D  LATEST        1C9D  LPT           1C49  LFA
1B11  LIMIT         17D1  LEAVE         14FD  LIT
36F8  MCOPY         2639  MESSAGE       2619  M/MOD
25E6  MOD           2591  M/            2576  M*
2323  MIN           230D  MAX           18CD  MINUS
2B32  NOTIME        220A  NUMBER        1CF5  NOOP
1C75  NFA           175C  NOT           32B2  OF
1DDA  OFFSET        1DC3  OUT           1900  OVER
1736  OR            3857  PL            215B  PAD
1C60  PFA           1C37  P!            1C25  P@
1B3B  PREV          19CC  PICK          24C8  QUIT
20F1  QUERY         3910  RL            35BC  RESUME
323E  REPEAT      - 2EA1  REWIND        2E56  ROLL
284B  R/W         - 2730  RESET         1E2A  R#
1D5E  R0            199F  ROT           1822  R
17FB  R>            17A2  RP!           1794  RP@
380C  SAVE          33D4  STRING        2D32  SIGN
2CED  SPACES        2522  S->D          1F7C  SMUDGE
1E8C  SPACE         1E02  STATE         1DCD  SCR
1D55  S0            1B92  SHR           1B7B  SHL
194B  SWAP          177D  SP!           176E  SP@
31BB  THEN          30A3  TRIAD         2B1F  TIME
2AA1  TAPE        - 285E  TR/W          2045  TYPE
1D68  TIB           1A1B  TOGGLE        3205  UNTIL
3055  UPDATE        2DE6  U.            1D34  USER
1B2C  USE           189B  U<            16DC  U/
16AA  U*            3971  VLIST         33F3  VECTOR
247F  VOCABULARY    1DA6  VOC-LINK      1D22  VARIABLE
3928  WHERE         328E  WHILE         2680  WARM
216E  WORD          1D82  WARNING       1D74  WIDTH
1749  XOR           14D5  XY            2F36  YESNO
2394  [COMPILE]     1F59  [             1F67  ]

 Ok
```