LEOBAS 1.0

Een uitbreiding op de PC1500-PC2
BASIC van Sharp.

Geschreven door:
Ernst Mulder.

(c)1986

-- Inhoudsopgave --

-- Deze handleiding is gemaakt met AppleWorks 1.2 --

-- Wat doet LEOBAS 1.0 --

LEOBAS 1.0, een software-uitbreiding van Uw PC-1500 (Sharp), of Uw PC-2 (Radio Shack), heeft drie functies:

- LEOBAS 1.0 geeft U beschikking over enkele nieuwe BASIC-kommando´s. Met deze nieuwe kommando´s wordt Uw machine heel wat krachtiger dan voorheen.

- LEOBAS 1.0 geeft U beschikking over 128 nieuwe karakters in de karakterset, deze karakters zijn echter nog niet uit te printer met de bijbehorende plotter. Als U dat nodig vindt, kunt U bij mij een plotter-tabel krijgen, horende bij de LEOBAS 1.0 karakterset.

- LEOBAS 1.0 bevat een eigen toetsenbord-routine, welke het werken met de machine een stuk makkelijker maakt.

In de volgende hoofdstukken worden alle drie deze gebieden uitgebreid toegelicht, met vele duidelijke voorbeelden. Mochten er voor U nog zaken onduidelijk blijven, dan kunt U altijd kontakt met mij opnemen.
Ook als er fouten in deze handleiding mochten voorkomen, dan hoorde ik dat graag van U.

-- De nieuwe BASIC-kommando´s --

LEOBAS 1.0 bevat 36 nieuwe BASIC-kommando´s en functies. Om bekend te raken met deze hoeveelheid nieuwigheid, volgt hierna een alfabetisch overzicht van de nieuwe kommando´s en functies.

Het systeem in het overzicht is als volgt:

KOMMANDO **Afkortingen**

Uitleg over de werking van het kommando of van de functie. Na deze uitleg volgen dan indien noodzakelijk:

BV: Een of meer voorbeelden.

Wanneer er bij deze voorbeelden het volgende teken staat: ´=>´, dan betekent dit dat U nadat U het er-op-volgende hebt ingetikt, op ENTER moet drukken om de regel uit te laten voeren. Op de volgende regel staat dan meestal het bij het voorbeeld horende resultaat.

Als er als voorbeeld een stukje programma staat, dan moet U dit laten uitvoeren, om uit de resultaten de werking van het programma te begrijpen.

Het overzicht vindt U op de volgende pagina´s.

ANGLE **ANG. ANGL.**

Dit kommando geeft de waarde van de huidige hoekmaat (Namelijk de grootte van een gestrekte hoek.)

```
BV:  => DEGREE
     => ANGLE
        180
```

ASKEY **ASK. ASKE.**

Dit kommando geeft de ASCII-waarde van de op dat moment ingedrukte toets. (Dus 0 als er geen toets wordt ingedrukt)
Het kommando is een verkorte versie van ASC INKEY$.

```
BV:     10 BEEP 1:IF ASKEY END
        20 GOTO 10
```

AT ERROR

Wanneer U dit in Uw programma gebruikt en er vindt een ERROR in uw programma plaats, dan wordt het programma voortgezet achter dit kommando.

```
BV:     10 AT ERROR PRINT "Foute waarde!!!"
        20 INPUT A
        30 PRINT "LN(A)=";LN A
        40 GOTO 20
```

AWRITE **AW. AWR. AWRI. AWRIT.**

Met dit kommando kunt U de uitkomst van een berekening direct naar de inputbuffer brengen (Zo kunt U echte functietoetsen maken waarbij er met de uitkomst kan worden doorgerekend). Met dit kommando kunt U ook direct ´tokens´ naar de inputbuffer brengen om zo vuile truukjes uit te halen (Maar dat is natuurlijk alleen voor de freaks!).

```
BV:     10"C"AREAD X:AWRITE COS X
        (Zo hebt U een echte cosinus functietoets, DEF-C, want U
        kunt met de uitkomst ervan doorrekenen)

BV:  => AWRITE CHR$ 241+CHR$ 152
        (Druk daarna op ´pijl terug´)
        LET
        (Het was dus het ´token´ van LET!)

BV:     Een ander leuk handigheidje met AWRITE is de volgende:
        (In de PROgram mode:)
     => AWRITE"10DATA"+STR$ LN 2
        (Druk nu op ´pijl terug´ en daarna op ENTER, U hebt nu een
        programmaregel gemaakt met als dat als constante
        de waarde van LN(2), zonder dat U dit getal moest overtypen)
```

BIN$ **BI. BIN.**

Met deze functie kunt U getallen omzetten van het decimale naar het binaire getalstelsel. De uitkomst is in de vorm van een 8 bits ´binaire´ string.

```
BV:  => BIN$ 242
        11110010
```

BYTE$ **BY. BYT. BYTE.**

Hiermee kunt U getallen omzetten van het decimale stelsel naar het hexadecimale stelsel. De uitkomst krijgt U in de vorm van een ´byte´, een string van twee karakters lang.

```
BV:  => BYTE$ 123
        7B
```

DCR **DC.**

Met dit kommando kunt U de waarde van een of meerdere numerieke variabelen met de waarde 1 verlagen. Dit kan soms erg handig zijn in bepaalde loops.

Niet bestaande variabelen (Met uitzondering van arrays) worden gecreeerd.

```
BV:  => A=1,B=2,C=3
     => DCR A
     => DCR A,B,C
     => PRINT A;B;C
        -1 1 2
```

ERL

ERL geeft als uitkomst het regelnummer waarop de laatste error plaatsvond. Deze is 0 als de error is ge-clear-ed.

```
BV:     10 AT ERROR PRINT "Error in line ";ERL :RESUME
        20 A=1/0
```

ERN

Deze functie hoort bij ERL, en geeft als uitkomst het nummer van de laatst voorgekomen fout. Ook hier geldt: 0 betekent dat de error is ge-clear-ed.

```
BV:     10 AT ERROR PRINT "Error nummer ";ERN :RESUME
        20 A=1/0
```

EXIT **EX.**

Met dit kommando verlaat U een eventueel aanwezige omleiding van de toetsenbord-routine. De orginele toetsenbord-routine wordt weer gestart. Dit kan soms nodig zijn, omdat een eventuele omleiding in RAM kan lopen, dat U voor andere doeleinden wilt gaan gebruiken.

LEOBAS 1.0 gebruikt ook een eigen toetsenbord-routine, dus als U de RAM van &9800 tot &9FFF voor iets anders dan LEOBAS 1.0 wilt gaan gebruiken, dan moet U eerst EXIT-ten.

Bij een koude start van de machine onder LEOBAS 1.0, wordt automatisch de LEOBAS 1.0 toetsenbord-routine weer ingeschakeld.

FACT **FA. FAC.**

Deze functie geeft de faculteit van de integer waarde van de meegegeven parameter. Voor deze parameter moet gelden:
0 <= Getal <= 69

```
BV:  => FACT 69
        1.711224524E 98
```

FIX **FI.**

Op het eerste gezicht lijkt FIX dezelfde functie als de bestaande functie INT, maar bij negatieve getallen werkt FIX verschillend: Bij FIX worden letterlijk alle decimalen achter de punt weggegooid.

```
BV:  => INT -2.5
        -3
     => FIX -2.5
        -2
```

FN

Met de functie FN kunnen waarden (string of numeriek) van een ge-label-de BASIC-regel worden gehaald. Dit kan het best worden toegelicht aan de hand van enkele voorbeelden:

```
BV:     10 "Hoogte" 10.5                (Zo kunt U gestructureerd
        20 "Breedte" 123.45             programmeren)
        30 "Naam" "Rob"
         |        |
         |        |
        100 OPP=FN "Hoogte"*FN "Breedte"
        110 PRINT "De naam is ";FN "Naam"
```

```
BV:     10 "X" W OR Z
        20 "Y" Z AND Y
        30 "X XOR Y" (FN "X" AND NOT FN "Y") OR (NOT FN "X" AND
           FN "Y")

        Met FN "X XOR Y" krijgt U dus (W OR Z) XOR (Z AND Y)
```

FRAC **FR. FRA.**

Deze functie doet het omgekeerde van INT, ze geeft als uitkomst juist alleen de getallen achter de decimale punt.

```
BV:  => FRAC 2.56
        0.56
```

GET **GE.**

Met GET kunt U wachten tot er een toets wordt ingedrukt. Tijdens het wachten op een ingedrukte toets kunt U zelfs de machine uitschakelen, en ook de AUTO-POWER-OFF werkt. Als er een toets wordt ingedrukt ontvangt U deze, of de ASCII-waarde ervan, in de meegegeven variabele.

```
BV:     10 WAIT 0:PRINT "Press any key...":WAIT
        20 GET A$
        30 PRINT "You pressed the ";A$

BV:  => GET A
        (Druk dan op de ´1´)
     => A
        49
        (Dit is de ASCII-waarde van de ´1´)
```

HEXVAL **H. HE. HEX. HEXV. HEXVA.**

Deze functie geeft de hexadecimale waarde van dat gedeelte van een string, dat geldige hexadecimale karakters bevat, beginnend aan de linkerkant van de string. Als de string niet begint met een hexadecimaal getal, of leeg is, dan geeft HEXVAL de waarde 0.

```
BV:  => HEXVAL "4000 Een hex getal"
        16384
     => HEXVAL "HALLO"
        0
```

HOME **HO. HOM.**

Dit kommando doet niets anders dan CURSOR 0, en heb ik ook alleen maar gemaakt omdat ik nog een paar bytes over had. (HOME is wel sneller dan CURSOR 0)

ICR **IC.**

Met dit kommando kunt U de waarde van een of meerdere numerieke variabelen met de waarde 1 verhogen. Dit kan soms erg handig zijn in bepaalde loops.

Niet bestaande variabelen (Met uitzondering van arrays) worden gecreeerd.

```
BV:  => A=1,B=2,C=3
     => ICR A
     => ICR A,B,C
     => PRINT A;B;C
        3 3 4
```

ICURSOR **ICU. ICUR. ICURS. ICURSO.**

Met dit kommando kunt U zelf in een programma een soort ´cursor´ maken. ICURSOR inverteert het karakter op de aangewezen cursor-positie.

```
BV:     10 PAUSE "Press a key:?"
        20 ICURSOR 12:FOR I=1TO 20:A=ASKEY :IF A=0NEXT I:GOTO 20
        30 END
```

INV

Hiermee kunt U het gehele scherm, een kolom op het scherm, of dots op het scherm inverteren. De formats zijn als volgt:

```
- INV                    Inverteert het hele scherm.
- INV z                  Inverteert kolom z.
- INV x,y                Inverteert dot (x,y).
        Of combinaties:
- INV x1,y1,x2,y2,...    Inverteert verschillende dots.
- INV x1,y1,x2,y2,z      Inverteert verschillende dots en kolom z.
```

```
BV:     10 PAUSE "Een voorbeeld."
        20 INV
        30 FOR Y=0TO 6:FOR X=0TO 155
        40 INV X,Y
        50 NEXT X:NEXT Y
        60 FOR Z=0 TO 155
        70 INV Z
        80 NEXT Z
```

KEYWAIT **K. KE. KEY. KEYW. KEYWA. KEYWAI.**

Met dit kommando kunt U een programma laten stoppen totdat er op een toets wordt gedrukt, terwijl de AUTO-POWER-OFF functie in werking blijft. Tijdens een KEYWAIT kunt U de machine rustig uitschakeken.

```
BV:     10 WAIT 0:PRINT "Press any key..."
        20 KEYWAIT
```

PRES **PRE.**

Hiermee kunt U een of meerdere dots van het scherm ´uitzetten´.

```
BV:     10 PAUSE "Geen puntje op de i"
        20 PRES 110,0
        30 KEYWAIT
```

PSET **PS. PSE.**

Hiermee kunt U een of meerdere dots van het scherm ´aanzetten´.

```
BV:     10 PAUSE "Puntjes op de a"
        20 PSET 85,0,87,0
        30 KEYWAIT
```

RCP **RC.**

RCP geeft als uitkomst de reciproke waarde (1/x) van een getal ongelijk aan 0. Ik vond deze functie, evenals FACT op de PC 1401, en dacht dat het handig zou kunnen zijn om deze in het repertoire op te nemen.

```
BV:  => RCP 2
        0.5
```

RESUME **RESU. RESUM.**

Wanneer U alles tot nu toe hebt gelezen, dan bent U RESUME al eerder tegengekomen, in combinatie met AT ERROR, ERL en ERN.

Resume heft een met AT ERROR of ON ERROR GOTO onderschepte fout op, en hervat de uitvoer van een programma achter de plaats van de fout.

Als een fout niet door het programma onderschept was en een programma stopt met een error-melding, dan kan ook met RESUME hervat worden.

```
BV:     10 FOR A=0TO 10
        20 PRINT "LN(A)=";LN A
        30 NEXT A

      => RUN
         ERROR 39 IN LINE 20
         (Druk op CL en vervolg met:)
      => RESUME
         (En het programma gaat gewoon verder)
```

ROUND **RO. ROU. ROUN.**

Met ROUND kunt U getallen op gehele waarden afronden. Het getal wordt naar boven of beneden afgerond naar het eerste cijfer na de decimale punt. Enkele voorbeelden:

```
BV:    => ROUND 5.4999
          5
BV:    => ROUND 5.5
          6
BV:    => ROUND -4.5
          -4
BV:    => ROUND -4.50001
          -5
```

SCROLL **SC. SCR. SCRO. SCROL.**

Dit kommando zorgt voor het naar links scrollen van het scherm. Wat links verdwijnt, komt dus rechts weer terug. Dit zijn de formats:

- SCROLL — Scrollt 1 positie op maximum snelheid
- SCROLL x — Scrollt x posities met de snelheid opgegeven met een WAIT opdracht. (1 <= x <= 255)

```
BV:     10 WAIT 2:PRINT "Voorbeeld:"
        20 SCROLL 156
```

SHIFT **SH. SHI. SHIF.**

Dit kommando zorgt voor het naar links schuiven van het scherm. Wat links verdwijnt, komt dus niet meer terug. Dit zijn de formats:

- SHIFT — Schuift 1 positie op maximum snelheid
- SHIFT x — Schuift x posities met de snelheid opgegeven met een WAIT opdracht. (0 <= x <= 255)

```
BV:     10 WAIT 2:PRINT "Voorbeeld:"
        20 SHIFT 156
```

SPACE$ **SP. SPA. SPAC. SPACE.**

Met deze functie kunt U een string van spaties genereren.

```
BV:  => PRINT "5";SPACE$ 5;"Spaties"
        5     Spaties
```

SQU

Ook deze functie vond ik op de PC 1401, en aangezien ze nogal simpel te programmeren was (Een sprong in de ROM) heb ik SQU geimplementeerd. SQU geeft het kwadraat van een getal.

```
BV:  => SQU -5
        25
```

STRING$ **STRI. STRIN. STRING.**

Met deze functie kunt U strings van een bepaald karakter en een bepaalde lengte genereren. De format is:

- STRING$ (l,c) — Geeft een string met lengte l en van karakter c

```
BV:  => STRING$ (5,65)
        AAAAA
```

TEN **TE.**

Alweer een PC 1401 kommando. TEN geeft een macht van tien.

```
BV:  => TEN 4
        10000
     => TEN .5
        3.16227766
```

TME$ **TM. TME.**

TME$ doet werkelijk niets anders dan STR$ (TIME * TEN 4)

```
BV:  => TME$
        (Zou kunnen geven:)
        127235358
        (Het was toen ik dit probeerde namelijk 27-1-86 en 23:53:58)
```

VMJ **VM.**

Deze functie is speciaal bedoeld voor diegenen die veel in machinetaal werken. (Eigenlijk heb ik VMJ voor mijzelf gemaakt)
VMJ geeft het adres horende bij een bepaalde vector-jump.

```
BV:  => VMJ &E0
        52619
```

WORD$ **WO. WOR. WORD.**

WORD$ is de 16-bit vervanging van BYTE$, en geeft dus de 2-byte hexadecimale waarde van een getal.

```
BV:  => WORD$ VMJ &E0
        CD8B
```

ZERO **Z. ZE. ZER.**

Hiermee kunt U de inhoud van variabelen wissen. Strings worden leeg, en numerieke variabelen krijgen de waarde 0.
Niet bestaande variabelen (Met uitzondering van arrays) worden gecreeerd, en direct gewist.

```
BV:  => ZERO A,B,AA$,L$
```

Tabel 1: De getalgrenzen voor de functies.

BIN$ b	0 <= b <= 255
BYTE$ b	0 <= b <= 255
FACT r	0 <= int(r) <= 69
FIX r	-
FRAC r	-
HEXVAL h$	0 <= hexval(h$) <= 65535
RCP r	r <> 0
ROUND r	-
SPACE$ n	1 <= n <= 80
SQU r	abs(r) < 1E 50
STRING$ (n,x)	1 <= n <= 80 0 <= x <= 255
TEN r	-100 < r < 100
VMJ b	0 <= b <= 255
WORD$ w	0 <= w <= 65535

Tabel 2: De getalgrenzen voor enkele kommando´s

ICURSOR p	0 <= p <= 25
INV p	0 <= p <= 155
INV x,y	0 <= x <= 155
	0 <= y <= 6
PRES x,y	0 <= x <= 155
	0 <= y <= 6
PSET x,y	0 <= x <= 155
	0 <= y <= 6
SCROLL n	1 <= n <= 255
SHIFT n	1 <= n <= 255

-- De karakterset --

Bij de PC1500-PC2 bestaat er de mogelijkheid om extra karakters aan de bestaande karakterset toe te voegen. Bij LEOBAS 1.0 wordt van deze mogelijkheid gebruik gemaakt. Deze 128 nieuwe karakters bevinden zich in de karakterset in de reeks [128-255]. In dit gedeelte zijn verschillende gebieden te onderscheiden:

-[128-223] Deze karakters zijn zondermeer op te nemen in een BASIC-regel. U kunt ze daar in krijgen door bijvoorbeeld AWRITE te gebruiken.

```
BV: => AWRITE "10PRINT"+CHR$ 34+"H"+CHR$ 192+"l"+CHR$ 193+"ne"
       (Druk nu op ´pijl terug´ en daarna op ENTER, en op
       regel 10 staat de naam ´Helene´ zoals dat hoort)
```

-[224-255] Deze karakters zijn niet in BASIC-regels op te nemen zoals in het vorige voorbeeld, omdat ze door de machine als ´tokens´ worden herkend. U kunt ze wel gebruiken in strings.

```
BV: => A$=CHR$ 234
    => A$+A$
       (En er verschijnen twee omega´s op Uw scherm)
```

Wanneer U geen plotter-tabel in gebruik hebt, probeert U dan nooit een van de nieuwe karakters uit te printen met de plotter. In LEOBAS 1.0 was er geen ruimte om een plotter-tabel op te nemen, en daarom zou Uw plotter hopeloos op ´tilt´ raken.

Tussen de nieuwe karakters bevindt zich ook een door mij gedefinieerde lower-case. (In het bereik [161-186]) Ik heb dit gedaan, omdat ik de Sharp-lower-case niet mooi vond. Meer informatie over deze lower-case vindt U in het volgende hoofdstuk: ´Het toetsenbord´

-- Het toetsenbord --

Wanneer de toetsenbord-routine-omleiding van LEOBAS 1.0 aktief is reageert zij enigszins anders dan normaal. De voornaamste verschillen zijn:

- Er is AUTO-REPEAT aanwezig: Wanneer U een toets een bepaalde tijd ingedrukt houdt, dan gaat deze herhalen.
- In de RESERVE-mode kunt U met de ´op´- en ´neer´-pijlen door de functietoets-definities heen lopen. Er zijn nu drie functietoetsen meer beschikbaar dan normaal het geval was: 3 maal [F1..F7].
- Alle functie-toetsen bevinden zich ook onder de cijfers: Shift-[´1´..´7´] geeft [F1..F7]
- U kunt nu de karakters !,",#,$,% en & bereiken zonder de Shift-toets te gebruiken. Ze bevinden zich op de plaatsen van de functietoetsen. (Die U nog steeds kunt gebruiken door eerst op Shift te drukken)
- Voor de mensen die, zoals ik, van een ´schoon´ geheugen houden heb ik deze functie toegevoegd: Shift-CL wist al het ongebruikte BASIC-geheugen. Dit gebeurt door dit te vullen met CHR$ 95´s, ´underscore´, een herkenbaar ASCII-karakter. Gebruik deze functie dus nooit als U machinetaal-routines achter het BASIC-programma hebt staan!
- Als U de machine uit, en weer aan zet, reageert deze alsof er een AUTO-POWER-OFF plaatsvond, U raakt geen informatie kwijt. Wilt U toch een ´harde´-OFF geven, gebruik dan Shift-OFF.

LEOBAS 1.0 maakt gebruik van een andere lower-case dan normaal. In de nieuwe karakterset bevindt zich een nieuwe lower-case, die vanaf het toetsenbord bereikbaar is vanuit de SMALL-mode. Wanneer U echter lower-case karakters maakt door de Shift-toets te gebruiken, dan krijgt U de oude lower-case.

Hoe U deze optie moet uitschakelen kunt U vinden in het hoofdstuk ´Systeem informatie´.

Wanneer U gebruik maakt van de nieuwe lower-case in een BASIC-programma, dan moet U er wel rekening mee houden, dat het programma dan niet meer te LLIST-en is met de plotter! (Zonder plotter-tabel)

-- Over afkorten --

In de alfabetische lijst van nieuwe kommando´s en functies vindt U afkortingen van deze functies. Dit zijn de afkortingen die U kunt gebruiken als U alleen LEOBAS 1.0 gebruikt. Zodra er andere uitbreidingen aanwezig zijn, zoals de plotter, zouden bepaalde afkortingen niet meer kunnen gelden. **ROUND** kunt U normaliter bijvoorbeeld afkorten met RO. maar met de plotter aangesloten geeft RO. het plotter-kommando **ROTATE**

Een ander probleem dat kan optreden is het volgende: De machine kan een BASIC-regel van U verkeerd interpreteren. Voert U de volgende BASIC-regel in:

```
10 IF B>A THEN 20
```

Dan wordt deze door de machine vertaald naar:

```
10 IF B> AT HEN 20
```

Omdat de machine het kommando AT eerder zag dan THEN. In dit geval kunt U dit probleem oplossen door in te voeren:

```
10 IF A<B THEN 20
```

Deze regel wordt foutloos vertaald.
Soms kan het niet op deze manier opgelost worden. De enige mogelijkheid is dan de machine te forceren de juiste vertaling te doen. Vult U in de eerste regel inplaats van een ´A´ bijvoorbeeld een ´B´ in:

```
10 IF B>B THEN 20
```

En corrigeert U na invoeren van de regel de ´B´ naar een ´A´, dan krijgt U alsnog de regel waar U om vroeg:

```
10 IF B>A THEN 20
```

-- Systeem informatie --

In dit hoofdstuk kunnen belangstellenden gegevens vinden over LEOBAS 1.0 die men nodig heeft als men LEOBAS 1.0 aan eigen wensen en behoeften wil aanpassen. Voor meer informatie kunt U bij mij terecht.

-- Algemene informatie --

Beslagen geheugengebied	:	&9800 - &9FFF
Plaats van de toetsenbord-tabel	:	&9A80 - &9AFF
Plaats van de karakterset-tabel	:	&9B00 - &9D7F
Pointer naar een plotter-tabel	:	&89BE

-- Sophisticated informatie --

		Adres:	
Leosoft-lower-case ON-OFF	:	&99E5	ON: &60, OFF: &20
CA memory-fill character	:	&99CA	

Verder kan ik melden dat er een ´bug´ aanwezig is, maar aangezien deze niet ernstig is, en er geen ruimte aanwezig is om de ´bug´ te verwijderen (Wel in LEOBAS 2.0) heb ik de bug laten zitten.

BUG: TEN x geeft een ERROR 17, als x <= -100.

Voor alle reacties kunt U schrijven naar:

Ernst Mulder,
Europalaan 12
5691 EN Son.