I N H O U D S O P G A V E

Dick van den Berge (c) - H.Boschstraat 10 - Hazerswoude (dorp) - 01728-8239
31-03-87 EDUKATIEVE- & SYSTEEM SOFTWARE - Broederhof 11 - 5504 JC Veldhoven

Dit is een selectie uit de ROM routines van de P2000T ; hieronder vallen zowel de MONITOR-ROUTINES (&H 0000 - &H 1000), als ook de ROUTINES uit de BASIC INTERPRETER (&H 1000 - &H 5000).

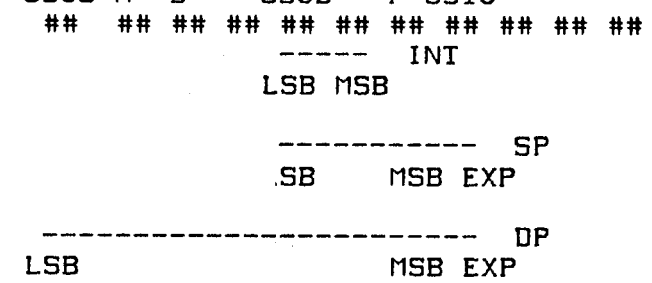
Notatie

De gebruikte adressen en getallen zijn Hexadecimaal (afgekort HEX).
Als in enkele voorbeelden decimale getallen gebruikt worden, is dit expliciet aangegeven. Intypen van de machinetaalroutines gaat het best met een MONITOR / DISASSEMBLER PROGRAM (Bv. Cassette A 212).

WERKGEBIEDEN, TABELLEN, BUFFERS

KB buffer	&H 6000 - 600B	Opslag ingetypte toetsen
INPUT (EDIT) buffer	6260 - 635F	voor EDITTEN van regels
PRINT BUFFER	651C - 6535	Verzamelen van grote getallen
Variabelen Type Tabel	640D - 6426	
STRING opslag Tabel	63BA - 63D9	
CASSETTE HULP-HULP HEADER	6103 - 6112	
CASSETTE HULP HEADER	6130 - 614F	
CASSETTE HEADER	6030 - 604F	

FAC = WRA1 = &H 6509 - 6510
WRA2 = 6513 - 651A



Verklaring : FAC = Floating Point Accumulator = WRA1
WRA1 = WORK AREA no.1
Hierin worden de resultaten van een berekening opgeslagen, alvorens ze aan BASIC worden doorgegeven.
WRA2 is identiek in opbouw met WRA1.

Om herkenbaar te zijn hebben deze rekenresultaten een vaste plaats en een vast formaat :

een INTEGER	beslaat 2 byte	(LSB voorop) en begint in &H	650D
een SINGLE PRECISION	4	(,, Exponent achter) vanaf	650D
een DOUBLE PRECISION	8	(,, , ,,)	6509

Leeg, ongebruikt gebied &H 6150 - 61FF geschikt voor eigen routines)
6360 - 63B3

	ROUTINE	HEX. CODE	FUNKTIE-OMSCHRIJVING	REGISTERS
1.1	CALL 121B	CD 1B 12	WIS BEELSCHERM = CHR$(12) en zet cursor linksboven	Gebruikt : A, B, DE, HL
	CALL 12D4	CD D4 12	WIS VENSTER = CHR$(22) vanaf cursorplaats tot onderaan	Gebruikt : alle reg. behalve C

1.3	LD HL, startadres
	LD A, regels
	CALL 0035
		21 yy xx
		3E nn
		CD 00 35
			WIS AANTAL REGELS
			xxyy = schermplaats &H
			nn = regels in &H
				Gebruikt
				A, B, HL

1.4	LD A, karakter CALL 1386	3E ch CD 86 13	Zet CHAR in A op scherm op huidige cursorplaats ch = karakter in ASCII-&H	Exit : AF, BC, DE, HL onveranderd

Cursor 1 plaats opgeschoven.

N.B. ch kan ook een stuurkarakter zijn, CHR$(1)-CHR$(31)

b.v. LD A, 0D zet cursor voor aan de regel
CALL 1386 (CR = carriage return)

LD A, 0A zet cursor 1 regel lager
CALL 1386 (LF = linefeed)

Andere mogelijkheden voor A zijn :

A = 01	cursor aan	A = 10	cursor 1 stap naar links
= 02	cursor uit	= 11	omhoog
= 07	piep	= 12	omlaag
= 08	wis char. links van cursor	= 13	naar rechts
= 0C	wis venster		

CALL 1A2A	CD 2A 1A	Doe CARRIAGE RETURN + LINEFEED	Exit : A = 0 BC, DE, HL onver- anderd
LD HL, mmnn LD (60B3), HL	21 nn mm 22 B3 60	Zet CURSOR in kolom &H nn + 1 en op regel &H mm + 1 N.B. cursor verschijnt pas na volgende PRINT-opdracht	
LD HL, TEKSTPTR CALL 3383	21 yy xx CD 83 33	Zet een TEKST op het scherm vanaf de cursorplaats	BC, DE onver- anderd

De TEKST moet in ASCII-code staan, het einde van de tekst wordt aangegeven door een 0
HL wijst naar het adres &H xxyy waar de eerste byte van de TEKST zich in het geheugen bevindt

CALL 1ADD	CD DD 1A	Zet de CURSOR AAN	(de opdracht wordt pas
CALL 1AE1	CD E1 1A	Zet de CURSOR UIT	uitgevoerd na de volgende PRINT-opdracht).
1.9 CALL 1085	CD 85 10	Haal CURSOR-ADRES in HL	Exit : HL = cursor schermadres
1.10 CALL 108D	CD 8D 10	Bereken SCHERMADRES van locatie, in kolom mm + 1 in regel nn + 1 (&H)	Entry : HL = nnmm Exit : HL = adres A, BC, DE onveranderd

Dick van den Berge (c) - H.Boschstraat 10 - Hazerswoude (dorp) - 01728-8239
31-03-87 EDUKATIEVE- & SYSTEEM SOFTWARE - Broederhof 11 - 5504 JC Veldhoven

	ROUTINE	HEX. CODE	FUNKTIE-OMSCHRIJVING	REGISTERS
	CALL 0029	CD 29 00	Opvragen van KB-STATUS	Exit : C = stoptoets, Z = geen toets BC, DE, HL onveranderd
2.2	CALL 0026	CD 26 00	Wacht op TOETSINDRUK	Exit : Toetscode in A
	CALL 193A	CD 3A 19	Wacht op TOETSINDRUK + omzetting in ASCII	Exit : In A ASCII- code BC, DE, HL onveranderd
2.4	CALL 1959	CD 59 19	OMZETTING TOETSCODE naar ASCII-code	Entry : A = toetscode Exit : A = ASCII-code Gebruikt HL
	CALL 2F2E	CD 2E 2F	Omzetting KLEINE letters naar HOOFDletters in ASCII	Entry : A kl. letter Exit : A hoofdletter
2.6	RST 38 LD A, (600D)	FF 3A 0D 60	SCAN KB, maar wacht niet op indruk	Exit : A = &H FF als geen toets ingedrukt als wel : A = toetscode

N.B. Deze routine is zo snel dat hij meestal de < RET > ziet van de START opdracht.

2.7 INPUT ROUTINE VOOR INTEGER GETALLEN

```
LD HL, 5280          21 80 52    Bufferadres op scherm
PUSH HL              E5
CALL 193A    nog     CD 3A 19    Wacht op een nieuw cijfer
LD (HL),A            77
INC HL               23
CP 0D                FE 0D       is < RET > ingedrukt ?
JR NZ nog            20 F7       neen, herhaal
POP HL               E1
CALL 28AC            CD AC 28    Omzetting ASCII naar HEX in DE
                                 Invoer : Getal intypen via KB
                                 Exit : Hexadecimale waarde in DE
```

N.B. Deze routine verzamelt een aantal cijfers die via het kleine of grote toetsenbord worden ingetypt. De cijfers worden gelezen door CALL 193A (zie 2.3) en opgeslagen in ASCII in een buffer die op het scherm staat, zodat de cijfers direkt leesbaar zijn. Het totale getal moet kleiner zijn dan 65529. Om te stoppen wordt de < RET >-toets ingetypt. Het tot dan toe verzamelde getal wordt dan door de routine CALL 28AC (zie 3.4) omgezet in een HEX-getal (een INTEGER) en in register DE gezet. Als een typefout gemaakt is, wordt alleen dat gedeelte van het getal omgezet, dat voor het eerste non-numerieke karakter staat

2.8	CALL 1900	CD 00 19	LINEINPUT via KB	INPUT : typ willekeurige tekst.
	N.B. De routine zet zelf eerst een ? op het scherm, daarachter wordt de tekst ingetypt Beeindig de tekst met < RET >, de tekst moet < 256			Exit de tekst staat in de buffer &H 6260
	CALL 1911 doet hetzelfde, maar zet geen ? op het scherm			

Dick van den Berge (c) - H.Boschstraat 10 - Hazerswoude (dorp) - 01728-8239
31-03-87 EDUKATIEVE- & SYSTEEM SOFTWARE - Broederhof 11 - 5504 JC Veldhoven.

2.9 V E R W I S S E L E N V A N T O E T S E N

Hiermee kunt u b.v. bereiken dat als u de toets "A" indrukt op het scherm de letter "B" verschijnt (Toepassing : als cryptografische versleuteling van uw toetsenbord).
Heel frustrerend is b.v. de <Return> toets in iets anders te veranderen, omdat dan geen enkele opdracht meer gegeven kan worden, wegens ontbreken van de <RET> toets.
De verwisseling vindt plaats via een reeds aanwezige, maar normaal lege correctietabel op 1F1D ; een pointer op &H 6094 verwijst naar deze tabel.
Zet nu op &H 6094/6095 een adres dat naar een eigen tabel wijst :

```
&H 6094  50
   6095  61    Startadres van eigen tabel = &H 6150

&H 6150  02    Aantal te verwisselen paren
   6151  22    toetscode van "A" = 34 = &H 22
   6152  42    ASCII van     "B" = 60 = &H 42
   6153  1D    toetscode van "B" = 29 = &H 1D
   6154  41    ASCII van     "A" = 65 = &H 41
```

Hiermee is voortaan toets "A" door "B" vervangen en "B" door "A".
De tabel bevat als eerste getal het aantal te verwisselen paren ; vervolgens komen er steeds paren : het eerste getal daarvan is de toetscode van de te vervangen letter, het tweede getal is de ASCII-waarde van de letter die ervoor in de plaats komt.
(Voor de ASCII's en Toetscodes zie Gebruiksaanwijzing BASIC NL pag. 141, 145,resp. 151).

2.10 H O O F D L E T T E R S N A A R K L E I N E L E T T E R S

In BASIC gebruikt u POKE &H 60B6,0 als u HOOFDLETTERS op het scherm wilt zien : dit werkt goed : of u nu een Hoofdletter intypt (via SHIFT<toets>) of een kleine letter (alleen <toets>), steeds ziet u Hoofdletters verschijnen.
Echter POKE &H 60B6,1 doet niet hetzelfde voor de kleine letters ; een ingetypte kleine letter (type <toets>) komt klein op het scherm, maar een ingetypte Hoofdletter (type in SHIFT<toets>) blijft gewoon een Hoofdletter op het scherm.

In machinetaal kunt u wel omzetting in beide richtingen uitvoeren

```
HOOFD naar kleine :
LD A, ASCII                    b.v. ASCII = &H 41 = 65 = "A"
OR 20                          41 OR 20   = &H 61 = 97 = "a"

klein naar HOOFD
LD A, ASCII                    b.v. ASCI  = &H 61 = 97 = "a"
AND 5F                         61 AND 5F  = &H 41 = 65 = "A"
```

De ASCII codes voor de kleine letters zijn steeds 32 (= &H 20) groter dan de overeenkomstige Hoofdletters, b.v.
"A" = &H 41 = 01 00 0001
"a" = &H 61 = 01 10 0001
d.w.z. in de Hoofdletters is bit 5 = 0,
in de kleine letters bit 5 = 1
De OR 20 en AND 5F opdrachten zetten dit 5e bit op de gewenste waarde

L A N G Z A M E K B - I N P U T R O U T N E

Deze routine laat zien dat zelfs als machinetaal gebruikt wordt, toch niet alles automatisch met flitsende snelheid verloopt.
Hiertoe zullen we het witte cursor blokje over het scherm bewegen door op de toetsen met de cursorpijltjes te drukken.
De "normale" routine die de toetscode van een ingedrukte toets ophaalt is CALL 0026 (zie 2.2) ; deze gebruiken we als volgt :

```
START  CALL 0029    CD 29 00   Check KB op "STOP" toets
       RET C        D8         als "STOP" dan terug naar MONITOR
       JR Z START   28 FA      blijft wachten op toets
       CALL 0026    CD 26 00   haal nu de toetscode in A
       CALL 1959    CD 59 19   omzetten toetscode naar ASCII
       CALL 1386    CD 86 13   verplaats cursor
       JR START     18 EF      herhaal
```

Vergeet nooit om zelf een "STOP" controle in te bouwen, want omdat u niet in BASIC werkt, doet de gewone "STOP" het niet.
De snelste cursorverplaatsing krijgt u hier door de toets ingedrukt te houden, echter de beweging blijft traag !
Dit komt omdat CALL 0026 de toets leest uit de KB-BUFFER :
nu gaat het lezenwel snel, maar daarna is de buffer LEEG en het vullen met een nieuwe toetswaarde gebeurt pas bij het volgend KB-interrupt, dwz. pas 20 msec later : zo komt het, dat u voor 40 toetsindrukken 40 * 20 ms = 0.8 sec moet wachten om de cursor een schermbreedte te laten doorlopen.

S N E L E K B - N P U T R O U T N E

Gelukkig kan het ook sneller, door niet op het vullen van de KB-BUFFER te wachten, maar door zelf alvast opdracht te geven de KB-lijnen te gaan scannen : dit gebeurt in de routine op adres &H 0038 en deze kan (met opzet) met een zeer korte opdracht RST 0038 = FF (zie 2.6) aangeroepen worden :

```
START   CALL 0029      CD 29 00   Check op "STOP"
        RET C          D8         als "STOP" dan terug naar MONITOR
        JRZ START      28 FA      blijft wachten op toets
        RST 0038       FF         doe KB scan
        LD A, (600D)   3A 0D 60   pak de toetscode uit &H 600D
        CP FF          FE FF      als er geen toets meer was
        JRZ START      28 FA      spring dan terug
        CALL 1959      CD 59 19   omzetten toetscode naar ASCII
        CALL 1386      CD 86 13   verplaats cursor
        JR START       18 EA      herhaal
```

RST 38 scant alle toetsenbordlijnen (zie 2.6) en zet de gevonden toetscode in &H 600D ; als er geen toets (meer) was, wordt teruggesprongen.
Deze routine verplaatst de cursor zo flitsend snel, dat u hem niet eens zult zien bewegen : inbouw van een vertraging (zie 7.1) is zeker nodig om ons oog wat tijd te gunnen.
Deze routine kan goed dienen als basis voor eigen gemaakte objecten (animatie sprites). De gewone lettertoetsen worden ook op het scherm gezet : u zult versteld staan van uw typesnelheid !

ROUTINE	HEX CODE	FUNKTIE-OMSCHRIJVING	REGISTERS
CALL 3E9A	CD 9A 3E	Omzetting HEX in HL naar DECIMAAL + Output naar scherm. Er wordt 16 geprint.	Entry : b.v. HL = 0010. Gebruikt alle registers

3.2 Omzetting HEX Integer naar ASCII — Exit : HL = einde buffer

LD HL,0010	21 10 00	willekeurige INT in HL	Resultaat in
LD (650D),HL	22 0D 65	move INT naar WRA1	ASCII in buffer
LD BC,0606	01 06 06	voorkom decimale punt	
LD HL,7000	21 00 70	HL = bufferadres voor resultaat	
CALL 426A	CD 6A 42	INT in WRA1 naar ASCII in buffer	

Floating Point Accumulator FAC = WRA1 = WORK AREA = &H 6509 - 6510
WRA2 = 6513 - 651A

B bepaalt de plaats van de scheidingskomma.
C bepaalt de plaats van de decimale punt.

N.B Als BC = 0300 dan resultaat 00,016, Als BC = 0001 dan resultaat .00016
0200 0,001,6 0002 0.0016
0100 ,000,16 0003 00.016

Omzetting HEX INT naar Single Precision (SP)

LD HL,0064	21 64 00	zet een willekeurige integer
LD (650D),HL	22 0D 65	in WRA1 (HL : 100 decimaal)
LD A,02	3E 02	zet type op INT
LD (63B6),A	32 B6 63	
CALL 38AB	CD AB 38	omzetting INT in WRA1 naar SP
		WRA1 bevat nu < 00 00 48 87 > = 100

Omzetting : ASCII naar HEX INTEGER

LD HL,7000	21 00 70	HL = pointer naar ASCII
CALL 28AC	CD AC 28	Omzetting : binair INTEGER in DE
LD (7020),DE	ED 53 20 70	save INT in buffer

Zet zelf de ASCII cijfercodes in de buffer vanaf &H 7000. HL moet een pointer zijn naar dit opslagadres. De omzetting stopt als het eerste non-numerieke char gevonden wordt. Het resultaat komt als een (hex) INTEGER in DE te staan.

Vb. : de ASCII string voor het getal 1000 (decimaal) moet omgezet worden ; zet dan achter 7000 : 31 30 30 30 30 00
Het resultaat is : DE = 03E8 , dit staat in de buffer achter 7020 vermeld als E8 03 .

Opm.: De MAXIMUM grootte van het ASCII-getal mag 65529 zijn. Grotere getallen worden niet geaccepteerd wegens een ingebouwd plafond in de ROM-routine (zie adres 28B6). Omdat deze routine ook gebruikt wordt om de ingetypte regelnummers te converteren, is dit tevens de reden dat geen regelnummers boven de 65529 kunnen voorkomen ! ("Normale" INTEGERS lopen tot 65536).

3.5 CALL 3EA9	CD A9 3E	Omzetting SP (of DP) naar ASCII	
		Omzetting van SP-getal in WRA1 in zijn ASCII equivalent	
		Het resultaat staat in de Printbuffer achter &H 651C.	
650D/0E/0F/10	CD CC 4C 7D	WRA1 bevat de SP voorstelling van de decimale breuk 0.1	
LD A,04	3E 04	zet type op SP	
LD (63B6),A	32 B6 63		
CALL 3EA9	CD A9 3E		

Voor omzetting van D(ouble) P(recision) moet WRA1 de DP voorstelling bevatten (6509-6510) en het type moet op 8 gezet worden (LD A,08).

BEREKEN SQR (X)

De SQR Routine kan alleen wortels trekken uit S(ingle) P(recision) getallen ; dit getal X moet aangeboden worden in WRA1.
Wilt u een wortel trekken uit een INTEGER, dan moet deze INTEGER omgezet worden in een SP, en in WRA1 gezet.

Vb.1. SQR (X) met X = 100.0 , als SP X = 00 00 48 87

```
LD A,04          3E 04       zet type op SP
LD (63B6),A      32 B6 63
LD HL, BUFFER    21 .. ..    HL = pointer naar adres waar SP staat
CALL 375D        CD 5D 37    move SP naar WRA1
CALL 4370        CD 70 43    SQR (SP) als SP in WRA1
                             Als resultaat bevat WRA1
                             00 00 20 84 = 10
```

Vb.2. SQR (X) met X = 100 als INTEGER

```
LD A,02          3E 02       Zet type op INT
LD (63B6),A      32 B6 63
LD HL,0064       21 64 00    input X als INTEGER ( X% = 100 = &H 64 )
LD (650D),HL     22 0D 65    zet X in WRA1
CALL 38AB        CD AB 38    omzetting INT naar SP in WRA1 ( zie 3.3 )
CALL 4370        CD 70 43    bereken SQR (SP)
CALL 3831        CD 31 38    omzetting SP naar INT in WRA1
                             Als resultaat bevat WRA1
                             ( 650D/E ) = A0 00 = 10 = SQR (100)
```

BEREKEN INT (X)

Deze funktie berekent het INTEGER gedeelte van het SP getal X
X wordt aangeboden in WRA1 ; het resultaat staat ook in WRA1.

```
LD A,04          3E 04       zet type op SP
LD (63B6),A      32 B6 63
LD HL, BUFFER    21 .. ..    HL = pointer naar SP getal
CALL 375D        CD 5D 37    verplaats SP naar WRA1
CALL 3937        CD 37 39    bereken INT (X) in WRA1
RET              C9

BUFFER           00 00 40 81 = SP voor 1.5
                 Resultaat : WRA1 bevat ( 650D ) : 01 = INT (1.5)
```

4.3 BEREKEN CINT (X)

Deze funktie zet een SP getal om in een INTEGER.
X wordt aangeboden als SP in WRA1, het resultaat komt ook in WRA1

```
LD A,04          3E 04       zet type op SP
LD (63B6),A      32 B6 63
LD HL, BUFFER    21 .. ..    verplaats SP naar
CALL 375D        CD 5D 37                WRA1
CALL 3831        CD 31 38    bereken CINT (X)
RET              C9

BUFFER           00 00 40 81 = SP voorstelling van 1.5
                 Resultaat staat in WRA1 ( 650D ) : 02 = CINT (1.5)
```

4.4 MACHTSVERHEFFEN X ^ Y

funktie verheft X tot de macht Y.

Vereiste invoer : X is een SP getal op de STACK
Y is een SP getal in WRA1
Het resultaat komt als SP, ook in WRA1.
INTEGER waarden voor X en Y worden eerst omgezet naar SP

```
LD HL, RETADRES   21 .. ..    zet een returnadres
PUSH HL           E5                  op STACK
LD A,02           3E 02       voer in X = 2 als INTEGER type
LD (63B6),A       32 B6 63
LD HL,0002        21 02 00    X = 2
LD (650D),HL      22 0D 65    zet X in WRA1
CALL 38AB         CD AB 38    omzetting van INT naar SP in WRA1
CALL 3750         CD 50 37    verplaats SP ( in WRA1 ) naar STACK
LD A,02           3E 02       voer in Y = 3 als INTEGER type
LD (63B6),A       32 B6 63
LD HL,0003        21 03 00    Y = 3
LD (650D),HL      22 0D 65    zet Y in WRA1
CALL 38AB         CD AB 38    omzetting van INT naar SP in WRA1
JP 437C           C3 7C 43    bereken X ^ Y

RETADRES          Het resultaat staat nu als SP in WRA1
                  2 ^ 3 = 00 00 00 84 = 8.
```

de aanroep van X ^ Y gebeurt met JP 437C (i.p.v. CALL), omdat de STACK in gebruik is voor de opslag van X. Om de terugkeer te verzekeren moet daarom een RETURNADRES op STACK gezet worden (van te voren : LD HL, RETADRES en PUSH HL). De X ^ Y routine keert dan terug naar RETADRES = 1e adres na bovenstaand aanroep programma.

4.5 BEREKEN SIN X)

Om de SIN te kunnen berekenen, moet de hoek X, uitgedrukt in radialen als SP getal in WRA1 gezet worden.
Doe deze berekening van te voren, en zet de gevonden SP waarde klaar op HOEKADR en vervolg dan met :

```
LD A,04           3E 04       zet type op SP
LD (63B6),A       32 B6 63
LD HL, HOEKADR    21 .. ..    HL = pointer naar hoekwaarde
CALL 375D         CD 5D 37    verplaats SP op adres HL naar WRA1
CALL 44DD         CD DD 44    bereken SIN (X)
                              Het resultaat staat nu als SP in WRA1.
                              Het SP getal kan m.b.v. Routine 3.5
                              in leesbare ASCII cijfers omgezet worden.
```

Het omzetten van de hoek in graden naar radialen gebeurt volgens :
1 graad = pi/180 rad.

De volgende tabel geeft enkele voorbeelden

```
 1 gr. = 0.0174533 = 35 FA 0E 7B
10 gr. = 0.174533  = C2 B8 32 7E
20 gr. = 0.349066  = C2 B8 32 7F
30 gr. = 0.523598  = 92 0A 06 80
40 gr. = 0.698131  = B8 B8 32 80
45 gr. = 0.785399  = DA 0F 49 80
```

BEREKEN RND (1)
Deze funktie berekent een (pseudo) random getal. Dit getal ligt tussen 0 en 1 en wordt als SP in WRA1 gezet.

```
BEGIN   XOR A          AF
        INC A          3C          maak A = 1 en zet vlag NZ
        CALL 446B      CD 6B 44    bereken RND (1)
                                   Het resultaat staat nu als SP in WRA1.
```

Met deze zelfde routine kunt u precies dezelfde RANDOM getallen maken, die ook in BASIC m.b.v. RND (1) geprint worden, als van te voren 3 tellers op nul gezet worden t.w. : &H 6214 = 0 ; 6215 = 0 ; 6216 = 0 en &H 6237 t/m 623A = 52 C7 4F 80
Dit laatste SP getal is een startwaarde die ook door BASIC ingelezen wordt vanaf ROM adres &H 23FE - 2401.
Na deze voorbereiding volgt de RND routine zoals boven bij BEGIN.
Omdat het vaker voor zal komen dat er (een of meerdere) RANDOM getallen nodig zijn met een INTEGER waarde, liggende tussen 0 en een eindwaarde, volgt er een tweede routine die deze mogelijkheid biedt.

BEREKEN INT (10 * RND (1))
Deze routine levert RND getallen, met een gehele waarde liggende tussen 0 en 10.
Het aantal en de eindwaarde (= 10) zijn vrij instelbaar.

```
        LD B,10            06 10       B = teller voor &H 10 RND -
START   PUSH BC            C5          store teller          getallen
        XOR A              AF       :
        INC A              3C       :
        CALL 446B          CD 6B 44 : maak RND, als SP < 1 in WRA1
        LD A,04            3E 04       zet type op SP
        LD (63B6),A        32 B6 63
        LD HL, SP.ADRES    21 .. ..    HL = pointer naar EINDWAARDE
        CALL 376E          CD 6E 37    verplaats EINDWAARDE naar BC/DE
        CALL 35C9          CD C9 35    vermenigvuldig BC/DE * RND
        CALL 3937          CD 37 39    omzetten SP naar INTEGER in WRA1
        LD HL,(650D)       2A 0D 65 :
        CALL 3E9A          CD 9A 3E : zet resultaat op het scherm
        POP BC             C1          pak teller
        DJNZ START         10 E0       herhaal
        RET                C9
    SP.ADRES           00 00 20 84   SP voorstelling van EINDWAARDE=10
```

Deze routine PRINT de gewenste RND getallen op het scherm ; voor de berekening is (o.a.) gebruikgemaakt van de bekende routines voor RND (zie 4.6), omzetten in INTEGER (zie 4.2) en PRINT op scherm (3.1)
Uit de reeds gegeven behandeling van INT (X), SQR (X) en SIN (X) blijkt een algemene wijze van aanroepen te gelden, die ook blijkt op te gaan voor de volgende nu te behandelen funkties (deze zullen daarom in het kort worden aangegeven). De algemene wijze van aanroep is :
1. zet het type klaar (= zet 04 in &H 63B6)
2. zet het argument als SP klaar in WRA1.
   (doe dat b.v. via LD HL, BUFPTR ; CALL 375D : move SP naar WRA1).
3. roep de funktie aan;het resultaat staat dan na afloop als SP in WRA1

Voor de volgende funkties volgt hier het aanroepadres :

4.8 ABS (X) = CALL 371E 4.11 EXP (X) = CALL 43C4 4.14 TAN (X) = CALL456A
4.9 ATN (X) = CALL 457F 4.12 LOG (X) = CALL 3583
4.10 COS (X) = CALL 44D7 4.13 FIX (X) = CALL 3924
4.15 X MOD Y

Vereiste invoer : X in DE als INTEGER van 2 byte
Y in HL als INTEGER van 2 byte
Het resultaat = DE MOD HL komt als integer in HL en ook in WRA1 te staan

```
LD DE, 000F   11 0F 00    bv. invoer X = 15
LD HL, 0004   21 04 00                 Y = 4
CALL 3AC2     CD C2 3A    resultaat HL = 15 MOD 4 = 0003
```

5.1 OPTELLEN van INTEGERS.

Deze routine telt 2 INTEGERS op die in HL, resp. DE staan ; hun som komt in HL te staan ; de inhoud van DE blijft onveranderd. Als er overflow ontstaat (som > 2 ^ 15), worden beide waarden eerst omgezet in SP en dan opgeteld. Het resultaat staat dan in WRA1. Vb. tel op HL = 8 en DE = 7.

```
LD A,02        3E 02       zet type op INTEGER
LD (63B6),A    32 B6 63
LD HL,0007     21 07 00    HL = 07
LD DE,0008     11 08 00    DE = 08
CALL 39E6      CD E6 39    tel op : HL = HL + DE
                           Als de type vlag = 04, was er overflow
```

5.2 AFTREKKEN van INTEGERS.

Deze routine trekt het INT in HL af van het INT in DE. Het verschil staat in HL. Vb. bereken 50-10 = &H 32 - &H 0A.

```
LD A,02        3E 02       zet type op INTEGER
LD (63B6),A    32 B6 63
LD HL,000A     21 07 00    HL = 0A
LD DE,0032     11 08 00    DE = 32
CALL 39DA      CD DA 39    trek af : DE - HL = 32 - 0A = 28
```

VERMENIGVULDIGEN van INTEGERS

Deze routine berekent het produkt HL * DE ; het resultaat staat in HL. Als er overflow optreedt, wordt er een SP vermenigvuldiging uitgevoerd met resultaat in WRA1. Vb. bereken 7 * 8

```
LD A,02        3E 02       zet type op INTEGER
LD (63B6),A    32 B6 63
LD HL,0007     21 07 00    HL = 07
LD DE,0008     11 08 00    DE = 08
CALL 3A06      CD 06 3A    Produkt in HL = &H 0038 = 56
```

DELEN van INTEGERS.

Deze routine deelt DE door HL : beide INTEGERS worden voor de deling omgezet in SP ; het quotient komt als SP in WRA1. Vb. bereken 100 : 10 = &H 0064 : &H 000A.

```
LD HL,000A     21 0A 00    HL = 0A
LD DE,0064     11 64 00    DE = 64
CALL 2E71      CD 71 2E    Deel DE/HL = 100 / 10 = 10
                       Het quotient staat in WRA1 nl.   00 00 20 84 = 10
```

INTEGER-DELING

Deze routine levert het INTEGER deel van de deling DE / HL, dwz. in BASIC notatie : INT (DE / HL). Vereiste invoer : DE en HL beide als 2 byte (Hex) INTEGER. Het resultaat komt als 2 byte INTEGER in HL en ook in WRA1 te staan Het blijkt dat de CALL 3A5D ook nog de "REST" van de deling kan opleveren : na deze CALL is nl. DE = 2 * "REST" ; de laatste 3 opdrachten delen DE door 2, zodat na afloop DE de REST bevat.

```
LD DE, 000F    11 0F 00     invoer DE = 15
LD HL, 0004                        HL =  4
CALL 3A5D                  int-deling HL = INT ( 15 /4 ) = 0003
XOR A                                 DE = 0006
RR D                       Halveer DE
RR E                                  DE = DE / 2 = 0003 = "REST"
```

N.B. de "REST" kan ook verkregen worden via de MOD fct : zie 4.15

N.B. verwar deze INTEGER-DELING niet met de DELING v INTEGERS : zie 5.4

Alle routines die de Interpreter gebruikt liggen vast in ROM ;
u kunt hier wel PEEKen maar niet POKEn, d.w.z. u kunt er niets
aan veranderen.
In de latere versie van BASIC NL is hierin verandering gebracht
door in het ROM sprongen aan te brengen naar het RAM-gebied.
Op dit RAM-adres staat dan vaak &H C9 (= Return); in principe
is er door deze heen- en weer terugsprong niets veranderd.
U kunt nu echter wel ingrijpen door in het RAM een extra, eigen,
routine tussen te voegen en pas daarna terug te springen.
Deze RAM-adressen waar naartoe gesprongen wordt, heten HOOKS ;
het veranderen van deze HOOKS om een eigen routine in te lassen,
heet " het ombuigen van de HOOK ".
De P-2000 heeft een veertien-tal HOOKS (hoe nuttig HOOKS kunnen
zijn, blijkt uit het feit dat de MSX er meer dan 100 heeft).
Elke HOOK bestaat uit 3 opeenvolgende bytes : de reden daarvan
is dat er dan een CALL nn opdracht precies inpast.
De 14 HOOKS liggen van &H 60C0 t/m &H 60EB .
Bij het opstarten worden deze adressen gevuld met beginwaarden
die u terug kunt vinden in de tabel &H 1804 t/m 18FF.
U ziet b.v. als inhoud van &H 60C0 : C3 86 13 = JP 1386
60C3 : C3 9A 13 = JP 139A enz.
Alvorens u een HOOK gaat veranderen, is een gedetaileerde kennis
nodig over het gebied WAAR de sprong VANDAAN kwam en WAAR hij
NAAR TOE moet.
De ingreep vindt n.l. plaats in een reeds lopend proces en
de verandering mag dit proces niet storen.
Meestal lukt dit wel door alle registers te redden en na afloop weer
terug te zetten. Zoals u in de volgende voorbeelden kunt zien,
geeft het verbuigen van HOOKS mogelijkheden, die anders niet
vervulbaar zijn, zoals :
Single Step Basic *
Funktietoetsen
Andere cursor
NO-LIST / Print FREE MEM *
Nieuwe BASIC commando's *
Cursor besturing *
Achteruit LISTEN
* Noot : Reeds gepubliceerd in P2000 gg Nieuwsbrief no. 10.

6.1 HOOKS IN DE P-2000

adres	Inhoud	Uit te voeren opdracht	Aangeroepen door
&H 60C0	JP 1386	zet char.in A op scherm	LIST, PRINT, EDIT
60C3	JP 139A	stuur 1 regel naar printer	LPRINT, LLIST, NEW
60C6	JP 1A12	LPRINT CHAR.in A	LPRINT
60C9	JP 1758	CLOAD	CLOAD
60CC	JP 16F0	CSAVE	CSAVE
60D0	00 00 00		na elk Basic-commando 'START'toets
60D3	00 00 00		BASIC READY WACHTLUS LIST
60D6	JP 0029	CHECK KB STATUS	zie 60D0 / 60D3
60D9	JP 1956	WACHT op KB TOETS	BASIC READY WACHTLUS
60DC	JP 247B	OUTPUT ERROR MESSAGE	CSAVE*ARRAY indien ERROR
60E0	JP 1A9B	ga in de EDIT stand	EDIT, INPUT, LINEINPUT
60E3	RET	geen	'STOP' en 'RETURN' toets
60E6	RET	geen	RST 10 JP 1013 JP 60E6
			N.B. de enige vrije RST
60E9	RET	geen	alle KB toetsen met
			ASCII >=12

CURSOR UITZETTEN

De routine 1.8 zet de cursor AAN (of UIT) door op adres &H 6013 een 1 (of 0) te plaatsen ; dit houdt in dat de cursor alleen maar uitgezet kan worden TIJDENS het uitvoeren van een BASIC-programma. Zogauw het programma klaar is, springt de computer naar de BASIC READY wachtlus, zet "O.K." op het scherm en ook de cursor is weer terug. Deze cursor die in de Directe Mode, of in de EDIT stand verschijnt, kunt u NIET laten verdwijnen (b.v. met POKE &H 6013,0), omdat deze cursor met opzet, vlak voor de entree in de wachtlus, nog even AANgezet wordt.
Omdat echter hierna nog het HOOKADRES &H 60D3 gepasseerd wordt, kan met de volgende machinetaalingreep, toch nog de cursor UITgezet worden

```
&H 60D3   CALL 6150   CD 50 61   Ombuiging van de HOOK

  6150    EXX         D9         Save registers
          CALL 1AE1   CD E1 1A   zet cursor UIT
          EXX         D9         herstel registers
          RET         C9         terug naar wachtlus
```

OPM het vullen van &H 60D3 t/m 60D5 MOET met het Monitor/Assembler programma gebeuren en NIET met POKE opdrachten vanuit BASIC. (de 1e POKE opdracht zou de lus verstoren die nog in de 2e en 3e POKE gebruikt moet worden).

CURSOR VERANDEREN

De volgende routine laat zien, dat door een ingreep op &H 60D3 niet alleen de cursor uitgezet kan worden, maar ook veranderd kan worden in een willekeurig ander CHARACTER.
Het principe is : zet de nieuwe cursor neer
wacht op een toetsindruk
doe nieuwe cursor weer uit
Een complicatie is, dat &H 60D3 ook gebruikt wordt door de LIST routine (o.a.) , deze mag niet gestoord worden, zodat we hiervoor een omleiding moeten inbouwen.

```
&H 60D3   JP 6150      C3 50 61   Ombuigen van de HOOK

  6150    EX(SP),HL    E3
          LD A,L       7D         check of &H 1947 op STACK staat
          CP 47        FE 47      omleiding voor LIST e.a. routines
          EX (SP),HL   E3
          JR Z    ---- 28 04
          JP 0029   -  C3 29 00
          RET       -  C9
          EXX     ---- D9         save registers
          CALL 1085    CD 85 10   haal cursoradres in HL
          LD A,41      3E 41      zet nieuwe cursor 41 = "A"
          LD (HL),A    77             op het scherm
          CALL 0029 -- CD 29 00   wacht op toetsindruk
          JR Z    ---- 28 FB
          LD A,80      3E 80      doe nieuwe cursor weer uit
          LD (HL),A    77
          EXX          D9         herstel registers
          POP BC       C1         clear STACK
          JP 1949      C3 49 19   hervat oude loop
```

Dick van den Berge (c) - H.Boschstraat 10 - Hazerswoude (dorp) - 01728-8239
31-03-87 EDUKATIEVE- & SYSTEEM SOFTWARE - Broederhof 11 - 5504 JC Veldhoven.

NIET LISTBAAR PROGRAMMA (bescherming van BASIC programma's).

```
&H 60D2   INC HL   23
   60D3   DEC HL   2B
```

Omdat de LIST opdracht &H 60D3 passeert, wordt door DEC HL de LISTING van een BASIC programma volledig onleesbaar.
Omdat RUN via &H 60D2 de compenserende INC HL passeert, blijft het BASIC programma normaal 'RUN'baar, terwijl het toch niet 'LIST'baar is.

SINGLE STEP voor BASIC programma's

Na de uitvoer van elke BASIC opdracht wordt &H 60D0 gepasseerd ; als we hier dus een (tijdelijk) oponthoud inbouwen, kunnen we de Basic opdrachten een voor een, na toestemming, laten uitvoeren.
Deze Single Step mogelijkheid is handig bij het zoeken naar BASIC programmafouten.
Voor een uitgebreidere beschrijving zie de P-2000 g.g.-Nieuwsbrief no.10 (nov.1984) pag.8.
Voor een eenvoudige Single Step is niets anders nodig dan :

```
&H 60D0   CALL 0026   CD 26 00
```

Het BASIC programma stopt nu na elke opdracht : door op een willekeurige toets te drukken, geeft u toestemming om door te gaan.

F U N K T I E T O E T S E N

De P-2000 kent al enkele Funktietoetsen b.v. de TOETSEN : CLEAR SCREEN, START, LIST, EDIT, ZOEK en FORMAT op het kleine toetsenbord.
Deze funktietoetsen reageren alleen als men in de Directe Mode is.
De STOP toets is een voorbeeld van een funktietoets, die reageert tijdens uitvoer van een (BASIC) programma.
Nieuwe funktietoetsen kunnen toegevoegd worden door ingreep via de HOOK &H 60E9, te gebruiken vanuit de Directe Mode, en door ingreep via HOOK &H 60D0, te gebruiken TIJDENS BASIC executie.
Voor beide gevallen geven we een voorbeeld.
Stel we willen beschikken over een nieuwe funktietoets, die tijdelijk de scherminhoud onzichtbaar maakt, zodanig dat dit later weer hersteld kan worden.

6.6 FUNKTIETOETS in Directe Mode

In de Directe Mode worden de toetsen onderscheiden in 2 groepen met resp. ASCII codes kleiner en groter of gelijk aan 128.
De 2e groep (>= 128) kan nu via HOOK &H 60E9 onderschept worden.

Onze nieuwe funktietoets moet dus van te voren een code >= 128 gekregen hebben : dit kan gebeuren via de correctietabel &H 6094 (zie routine 2.9 Verwisselen van toetsen).
Kies als nieuwe ASCII code een getal > 133, omdat de codes 128-133 al bezet zijn door resp. ZOEK, FORMAT, DEF, START, EDIT en LIST.
Op het kleine toetsenbord is de Toets SHIFT 2 nog vrij ; deze zullen we veranderen in ASCII code 136.
Het onzichtbaar maken van het scherm gebeurt via machinetaal omzetting van BASIC opdracht OUT 48,40.

vervolg
6.6

```
&H 6094   5D 61        Adres van nieuwe correctietabel

   60E9   JP 6150      C3 50 61

   6150   CP 88        FE 88       is het SHIFT 2
   6152   RET C        D8          zo neen : terug
   6153   LD A,28      3E 28       zo ja : zet scherm 2 ( rechter
   6155   OUT 30       D3 30               op de buis
   6157   CALL 0026    CD 26 00    wacht tot willekeurige toets
   615A   JP 1FC6      C3 C6 1F    terug naar BASIC        .

   615D   01                       Aantal paren te verwisselen
   615E   82                       toetscode SHIFT 2 toets
   615F   88                       gewenste ASCII code
```

6.7 FUNKTIETOETS tijdens BASIC programma executie

```
&H 60D0   CALL 6160    CD 60 61

   6160   CALL 0029    CD 29 00    check KB status
          RET Z        C8          ga terug als geen toets
          RET C        D8          ga terug als STOP toets
          CALL 193A    CD 3A 19    haal ASCII code van de toets
          CP 88        FE 88       is het SHIFT 2
          RET NZ       C0          nee : ga terug
          PUSH BC      C5
          PUSH DE      D5            save alle registers
          PUSH HL      E5
          PUSH AF      F5
          LD A,28      3E 28       zet 2e scherm
          OUT 30       D3 30          op de buis
          CALL 0026    CD 26 00    wacht tot willekeurige toets
          XOR A        AF          1e scherm
          OUT 30       D3 30       weer zichtbaar
          POP AF       F1
          POP HL       E1
          POP DE       D1            herstel alle registers
          POP BC       C1
          RET          C9          voortzetting van programma
```

Ook hier is via de correctietabel &H 6094 toetscode 82 in 88 veranderd.
U kunt nu een willekeurig, lopend BASIC programma, onderbreken door op SHIFT 2 te drukken : het beeld verdwijnt, totdat u een willekeurige toets indrukt ; het programma gaat dan door, waar het gebleven was.
U kunt meerdere en ook andere funktietoetsen benoemen door de correctietabel uit te breiden.
U kunt ook andere opdrachten laten uitvoeren (b.v. CLEAR SCREEN = CALL 121B enz.) door deze op te nemen tussen de PUSH en POP opdrachten
Als de funktietoets zowel in Directe Mode als tijdens het RUNNEN moet reageren, moet u beide routines achterelkaar opnemen.

Opm. Als het programma na een willekeurige toets niet opnieuw wil verdergaan, moet u voor de CALL 0026 opnemen :

```
XOR A          AF
LD (600C),A    32 0C 60
```

De machinetaal is n.l. zo snel dat de SHIFT 2 toets nog ingedrukt is als de test op een willekeurige toets al uitgevoerd wordt : Zet daarom eerst de KB bufferteller terug op nul.

Dick van den Berge (c) - H.Boschstraat 10 - Hazerswoude (dorp) - 01728-8239
31-03-87 EDUKATIEVE- & SYSTEEM SOFTWARE - Broederhof 11 - 5504 JC Veldhoven.

Omdat &H 60D3 in de (grote) BASIC wachtlus voorkomt, maar ook, als opvolger van &H 60D0, in de Executielus, moet het mogelijk zijn via een enkele HOOK (= 60D3) een funktietoets te maken, die werkt, zowel in de Directe Mode, als ook tijdens het RUNnen van een programma.
Hier volgt dan ook een 2e, verbeterde, versie van de "Verberg het scherm" funktietoets die de beide mogelijkheden van 6.6 en 6.7 in een routine combineeert.

&H 60D3 JP 6150

```
6150  CALL 0029       CD 29 00   check KB status
      RET Z           C8         ga terug als geen toets
      RET C           D8         ga terug alss STOP toets
      LD A, (6000)    3A 00 60   pak toetscode uit KB buffer
      CP 44           FE 44      is het toets 1/4
      RET NZ          C0         nee : terug, het is een andere toets
      XOR A           AF         set Z : er is nu geen toets ( meer )
      PUSH BC         C5         save alle registers
      PUSH DE         D5
      PUSH HL         E5
      PUSH AF         F5
      LD A, 28        3E 28      zet 2e scherm
      OUT 30          D3 30         op de buis
      XOR A           AF
      LD (600C), A    32 0C 60   reset KB bufferteller
      CALL 0026       CD 26 00   wacht op willekeurige toets
      XOR A           AF           herstel
      OUT 30          D3 30      1e scherm
      POP AF          F1
      POP HL          E1
      POP DE          D1
      POP BC          C1
      RET             C9         voortzetting wachtlusprogramma
```

U kunt elke toets tot funktietoets benoemen door zijn toetscode (zie Gebruiksaanwijzing BASIC NL p.151) in de plaats van CP 44 te zetten.

6.9 P I E P E N D T O E T S E N B O R D V I A 6 0 D 9

Vlak voor de routine, die van een (reeds geconstateerde) toets de toetscode gaat ophalen, is op 60D9 een HOOK geplaatst.
Als we via deze HOOK eerst nog even de geluidsroutine (de "BEEP" op CALL 0032) aanroepen, hoort u bij elke toetsindruk een piep-geluid :
&H 60D9 JP 6150 C3 50 61

```
6150  CALL 0032   CD 32 00   piep-routine
6153  JP 1956     C3 56 19   haal toetscode
```

In de P2000 gg Nieuwsbrief no.10, nov 1984, p.28 is dit zelfde effect bereikt via een hardware schakeling met een handvol onderdelen : hier ziet u duidelijk wat een omgebogen HOOK met 6 extra byte vermag.

Dick van den Berge (c) - H.Boschstraat 10 - Hazerswoude (dorp) - 01728-8239
31-03-87 EDUKATIEVE- & SYSTEEM SOFTWARE - Broederhof 11 - 5504 JC Veldhoven

We willen een digitale tijdsaanwijzing op een hoekje van het scherm maken ; omdat de klok moet doorlopen, zowel tijdens het RUNNEN van een (ander) programma, als tijdens het wachten, is 60D3 de aangewezen HOOK.
De tijd zelf ontlenen we aan de ingebouwde "klok" op &H 6010 / 6011 : dit vakje wordt elke 20 msec met 1 verhoogd, totdat 65535 bereikt is, want 65536 is weer 0, zodat telkens na ca. 21 minuten de klok terugspringt ; omdat bijhouden hoe vaak dit gebeurt en daarvoor corrigeren, veel rekenwerk geeft, is gekozen voor het "Km.teller" principe : accumuleer de 20 msec pulsen, totdat er 50 geweest zijn, reset de 20 ms teller, en verhoog tegelijk de klok met 1 seconde.

```
&H 60D3   CALL 6156       CD 56 61

   6156   EXX              D9          save alle registers
          LD A, (6010)     3A 10 60    pak huidige klokwaarde
          LD HL, (6154)                pak vorige klokwaarde
          CP L             BD          is de huidige al groter ?
          JR NZ verder     20 02       zoja, spring
   terug  EXX              D9          zonee, ga dan maar terug
          RET              C9
   verder LD (6154),A      32 54 61    bewaar de nieuwe huidige klok
          CP 32            FE 32       zijn er al 50 pulsen geweest ?
          JP NC terug      30 F7       nee, dan terug
          SUB A,32         D6 32       ja, verminder dan hun aantal met 50
          LD (6010),A      32 10 60    en bewaar als huidige klok,
          LD (6154),A      32 54 61    en als vorige klok.
          LD HL,6153       21 53 61    uren min  sec
                                       6150 6151 6152 6153 6154 KLOKBUFFER
          CALL 61AB        CD AB 61    UPDATE KLOKBUFFER : 1 sec erbij
          CP 60            FE 60       zijn er nu 60 sec ?
          JR C scherm      38 11       nee, spring
          CALL 61AB        CD AB 61    ja, UPDATE KLOKBUFFER : 1 min erbij
          CP 60            FE 60       zijn er nu 60 min ?
          JR C scherm      38 0A       nee, spring
          CALL 61AB        CD AB 61    ja, UPDATE BUFFER : 1 uur erbij
          CP 24            FE 24       zijn er 24 uur voorbij ?
          JR C scherm      38 03       nee, spring
          CALL 61AB        CD AB 61    ja, UPDATE  reset uren op 0
   scherm LD B,03          06 03       zet KLOKBUFFER op scherm
          LD HL, 6150      21 50 61    HL = bron adres
          LD DE, 5020      11 20 50    DE = doel adres
   6194   XOR A            AF          ga nu eerst 1 byte naar 2 ASCII omzetten
          RLD              ED 6F
          ADD 30           C6 30       maak ASCII
          LD (DE),A        12          zet 1e cijfer van UREN op scherm
          AND 0F           E6 0F
          RLD              ED 6F       roteer 2e cijfer in A
          ADD 30           C6 30       maak ASCII
          INC DE           13          schermadres + 1
          LD (DE),A        12          zet 2e cijfer van UREN op scherm
          RLD              ED 6F       roteer door : init de MIN
          INC DE           13          1 spatie overslaan
          INC DE           13          DE. schermadres van MIN
          INC HL           23          BUFFERptr. naar MIN
          DJNZ 6194        10 EB       herhaal : 2e keer naar MINUTEN
                                                 3e keer naar SEC.
          EXX              D9
    61AA  RET              C9          Exit klokroutine
```

Dick van den Berge (c) - H.Boschstraat 10 - Hazerswoude (dorp) - 01728-8239
31-03-87 EDUKATIEVE- & SYSTEEM SOFTWARE - Broederhof 11 - 5504 JC Veldhoven

```
&H 61AB   XOR A          AF       KLOKBUFFER BIJWERKEN :
          LD (HL),A      77       reset sec : = 0 ( resp.min : = 0 )
          DEC HL         2B       ptr.naar "min" adres
          LD A, (HL)     7E       pak de "minuten" ( resp.uren )
          INC A          3C       verhoog min : = min + 1 ( resp.uur :
                                                           = uur + 1 )
          DAA            27       noteer in decimale ( BCD ) cijfers
          LD (HL),A      77       1 byte bevat 2 BCD cijfers
   61B2   RET            C9
```

Deze klokroutine is wat langer omdat er meerdere dingen moeten gebeuren : &H 6010 wordt opgehoogd volgens een "stapjes" kromme ; de klokroutine is zo snel dat hij tijdens een stapje meerdere keren de tijd onderzoekt : toch mag hij dan maar een keer een 20 msec puls noteren ; daarna wordt de oude stand opgeslagen in &H 6154 en steeds vergeleken met de momentane waarde.
Anderzijds kan een BASIC opdracht zo lang duren, dat een bepaald tijdsstapje helemaal niet gezien wordt : in deze periode kan de 50e puls gepasseerd zijn, daarom wordt er 50 afgetrokken en het (eventuele) restant bewaard in &H 6010, ipv. een volledige reset op nul.
De gecumuleerde stand voor de uren, min en sec wordt bewaard in de klokbuffer &H 6150 - 6152 ; de tijd wordt hierin opgeslagen in decimale cijfers, dwz. "half acht 's-avonds" wordt genoteerd als 19 30 00.
Deze omzetting van binaire naar decimale cijfers wordt uitgevoerd door een enkele opdracht DAA ; zodoende kan de bufferinhoud (= de gewenste tijd) direct op het scherm worden gezet.
Een drievoudige aanroep vertaalt elk byte in 2 ASCII's en zet achtereenvolgens de uren, minuten en seconden rechtsboven aan het scherm.
U kunt de klok gelijkzetten door (m.b.v. MONITOR programma) de gewenste (decimale !) tijd in de buffer &H 6150 - 6152 te zetten.
Verlaat de MONITOR met QUIT en de klok loopt door, ook als u een ander (BASIC) programma gaat RUNNEN.
N.B. : Alleen tijdens CLOAD staat de klok stil omdat &H 6010 niet meer opgehoogd wordt ; als de klok niet precies gelijk loopt, komt dit omdat de pulsen niet precies om de 20 msec verschijnen : knutselaars kunnen dit verhelpen door 49 of 51 pulsen af te wachten en / of een geschikt DELAY in te bouwen.

7.1 DELAY

Omdat machinetaal zo snel is, is vaak een vertraging nodig.
Een aantal hiervan komt in de CASSETTE-routines voor :

	delay	
CALL 0ADA	69 msec	Er worden geen registers veranderd.
CALL 0ADF	101	
CALL 0AD0	120	
CALL 0AE4	261	
CALL 0AF2	500	

In deze routines wordt niets anders gedaan, dan de registers B en C met een STARTWAARDE te vullen en ze dan via een dubbele lus weer terug tot nul te laten tellen.
Een geschikte routine om zelf de STARTWAARDE te kunnen opgeven is de volgende :

```
        LD HL, nnmm     21 ..       STARTWAARDE in HL
herhaal LD B,L          45          pak teller uit HL
        DJNZ            10 FE       delay loop
        DEC HL          2B
        LD A,L          7D          Check of
        OR H            B4            HL = 0
        JR NZ           20 F8       zoniet, herhaal
```

enkele waarden van HL, levert dit de volgende vertraging

```
HL = 000A = 10          0.4 msec
     0064   10^2       27.4
     03E8   10^3      666.4        = 0.7 sec
     2710   10^4        6.8 sec
```

TERUG NAAR BASIC KOUDE START ; WARME START

Als u na het experimenteren met een (eigen) machinetaaloutine vreest, dat de STACK in de war is, zodat u zeer binnenkort een volledige CRASH te wachten staat, dan kunt u zich redden door naar BASIC terug te springen Hiervoor komen 2 sprongadressen in aanmerking :

7.2.1 a. het KOUDE START ADRES via JP 1F66.
Dit is het adres waar BASIC ook zelf naar toe springt, als hij voor het eerst wordt opgestart : vandaar de naam KOUDE START, er waren nl toch nog geen gegevens of variabelen in gebruik.
Het resultaat is daarom dan ook, dat alle BASIC programma's gewist worden, de Variabelen worden op nul gezet, maar ook de STACK wordt opnieuw in orde gemaakt en u eindigt met op het scherm de tekst : " Philips Cassette Basic ", "OK".

7.2.2 b. het WARME START ADRES via JP 1FC4 (ook 1FC6 is mogelijk)
U komt nu ook in BASIC terecht, maar uw vroegere gegevens worden nu ontzien : uw (oude) BASIC programma en de bijbehorende Variabelen worden NIET gewist ; bovendien is de STACK weer in orde gemaakt.
U eindigt in de BASIC wachtlus met "OK" op het scherm ; het BASIC programma is (nog) niet gestart.

OPM de zwarte RESET KNOP veroorzaakt een KOUDE START sprong ; het NMI knopje, zoals beschreven in de P2000 Niuwsbrief no.11 (feb 85, p.55) veroorzaakt een WARME START sprong : dit laatste kan dus hoogst nuttig zijn, als uw BASIC of machinetaal programma "vastzit" (= KB reageert niet meer) ; zonder uw programma te wissen, kunt u zich dan weer "bevrijden".

Een andere mogelijkheid is, dat u vanuit machinetaal wilt terugspringen naar BASIC, zodanig dat het BASIC PROGRAMMA meteen gestart wordt.
Naar keuze kunt u starten met de eerste BASIC REGEL, of met een van te voren opgegeven REGELNUMMER.
U springt terug naar de 1e BASIC REGEL door aanroep van het adres van de "RUN" routine, als volgt :

```
XOR A       AF           set FLAGS Z, NC
JP 28D4     C3 D4 28     Sprong naar "RUN  zonder regelnummer.
```

De "RUN" routine verwacht dat de Z en NC vlag gezet zijn : hiervoor zorgt de opdracht XOR A.
Om naar een bepaald regelnummer te springen, is de volgende routine nodig

```
XOR A          AF
INC A          3C          zet vlag op NZ
SCF            37          zet Carry vlag
LD HL,6150     21 50 61    HL : = pointer naar Regelnummer
JP 28D4        C3 D4 28
```

U springt naar hetzelfde "RUN"adres : dit keer verwacht de computer dat de NZ en C vlag gezet zijn. HL is een pointer, deze wijst op een bufferadres (&H 6150) en op dit adres moet van te voren het gewenste regelnummer ingevuld worden : N.B. dit regelnummer moet in ASCII code ingevuld worden.
B.v. als u RUN 10 wilt laten uitvoeren :

```
     moet op &H 6150   &H 31   gezet worden ( &H 31 = 49 = ASCII van 1 )
       en op    6151     30                 (    30   48              0 )
          op    6151     00                 ( afsluiten met echte nul   )
```

De keuze van het bufferadres &H 6150 is willekeurig, elk ander adres is ook goed ; het adres heeft NIETS te maken met de locatie in het BASIC programma waar het eigenlijke regelnummer staat ; het dient alleen als doorgeefluik om het regelnummer aan de RUN routine te overhandigen.