VOOR DE BASIC PROGRAMMEUR: Command Drivers

De BASIC waarmee wij het als MSX-ers mee moeten doen is welliswaar zeer uitgebreid, maar toch.., voor een aantal toepassingen bestaat nog geen BASIC-commando.
Het MSX-systeem is echter zo doordacht dat men bij het ontwerp al rekening heeft gehouden met uitbreidingsmogelijkheden. Anders zou men op een MSX-1 nooit een diskdrive kunnen gebruiken. Een andere BASIC-uitbreiding is het zgn. CALL-statement dat voornamelijk door een uitbreidings-cartridge wordt aangemaakt. Ook nieuwe BASIC-commando's worden vaak met CALL aangeroepen (zoals bijvoorbeeld CALL FORMAT, CALL KANJI, CALL PCMREC enz.).

Het BASIC Commando: CMD

De BASIC-uitbreiding waar dit artikel over gaat is: CMD. De naam van hiervan is de afkorting van command. De algemene syntax (schrijfwijze) van CMD is als volgt:

```
CMD [argument][,parameter][,enz.]
```

Omdat het woord 'argument' tussen rechte haken staat, is zo'n argument optioneel. Echter, zo'n argument is wel degelijk van belang, omdat er zo onderscheid wordt aangebracht in verschillende commando's.
Een CMD is relatief eenvoudig te maken (eenvoudiger als een CALL-uitbreiding), maar U als BASIC programmeur kunt U beperken tot de toepassing van één of meerdere command drivers. Een command drivers is dus een klein machinetaal-programma dat zo'n CMD opdracht installeert, vertaalt en uitvoert. Er zijn enkele dingen waaraan de BASIC-programmeur zich aan moet houden, maar daarover straks meer.

De Command Drivers

In dit nummer van MSX Mozaik staan twee command drivers in de vorm van listings afgebeeld. Deze kunt U in Uw eigen BASIC-programma toevoegen. Echter, de meest eenvoudige manier is om ze te laden vanaf de bijgeleverde diskette. De verschillende command drivers zijn 'CMDBLINK.BIN' en 'CMDSRAM.BIN'.

Hoe & Waar?

Hoe wordt zo'n command driver geactiveerd? Dat kan op twee verschillende manieren: De eerste manier: door de machinecode in de vorm van DATA-regels in het BASIC programma op te nemen. Een BASIC-routine die met GOSUB kan worden aangeroepen POKEt de machinecode in het geheugen en activeert het. De tweede namier: door de command driver van schijf te laden met behulp van een BLOAD "filenaam",R commando.
Waar komt zo'n command driver eigenlijk in het geheugen? Een command driver neemt een stukje weg van het BASIC geheugen. Voor CMDBLINK is het volgende stukje geheugen gereserveerd: DA00 - DAFF. En voor CMDSRAM dit stukje geheugen: D900 - D9FF. Elke command driver komt 256 bytes lager in het geheugen te liggen, mits een command driver niet groter als 256 bytes is. We moeten BASIC nu duidelijk maken dat dit geheugen niet voor andere doeleinden mag worden gebruikt. Dat kan met behulp van een CLEAR-instructie. We reserveren het RAM geheugen vanaf adres D000 voor het gebruik van command drivers, zodat er dan genoeg ruimte over blijft voor een BASIC-programma en voor een aantal toekomstige command drivers: CLEAR x,&HD000 waarbij x het aantal bytes aangeeft dat gereserveerd moet worden voor stringvariabelen (bijv. 1000). Een ander belangrijk punt voor de BASIC-programmeur is dat een command driver maar één keer geinitialiseerd mag worden. Dit om het blijven hangen van de computer te voorkomen. Dus twee maal BLOAD"CMDBLINK.BIN",R of twee maal GOSUB x is verboden! Gelukkig kunnen alle command drivers in één keer worden gedeactiveerd. Dat gaat zo: POKE &HFE0D,201.
Samengevat moet een BASIC-programma als volgt worden opgebouwd:

```
10 CLEAR 1000,&HD000: 'Reserveren van ge-
heugen waarbij alleen het getal 1000 mag
worden veranderd (stringgeheugen).
20 POKE &HFE0D,201: 'Deactiveren oude CMD's
30 BLOAD"CMDnaam.BIN",R: 'CMD installeren
(meerdere kan ook)
40 ...(het echte programma, CMD commando's
toegestaaan!)
```

Het installeren van een CMD in regel 30 kan ook met behulp van een GOSUB-instructie naar de initialisatie van de command driver elders in het BASIC programma.

CMDBLINK.BIN

(voor MSX-2 of hoger) Een aardige mogelijkheid van de V9938, onze MSX-2 videochip, is dat deze in textmode 80*24 letters in een balk kan laten verschijnen. De letters in zo'n balk kunnen een andere voor- en achtergrondkleur bevatten als de letters die buiten de balk staan. Ook kan de V9938 en zijn opvolger de V9958 (de MSX2+ en MSX turbo R-videochip) deze balk automatisch in variabele snelheden laten knipperen. Echter, een BASIC-commando om dit alles aan te sturen is er tot op heden nooit geweest. Maar de slimme BASIC-programmeurs wisten er toch raad wel mee: alle informatie staat namelijk in een tabel in het video-RAM. Een bepaalde positie op het scherm kan van blinking worden voorzien door de informatie in deze tabel op bit-niveau te veranderen, wat niet echt gemakkelijk te noemen is. Bovendien zal die methode niet echt snel zijn, vandaar dit nieuwe BASIC commando. CMDBLINK.BIN activeert twee nieuwe commando's: `CMD CLRBLK` en `CMD BLINK x,y,aantal`
Het eerste commando wist de BLINK-tabel, zodat er geen schermposities meer blinken. Het andere commando zet vanaf de schermpositie x,y een aantal karakters in de blinking-mode, waarbij x loopt van 0 t/m 79, y van 0 t/m 23 en aantal van 1 t/m 255. Het nogmaals uitvoeren van dit commando resulteert in een verwijdering van de schermposities uit de BLINK-tabel. Het is wel nodig om eerst twee VDP registers in te stellen om enig resultaat te bereiken, namelijk: `VDP(13)=&Hxy` - Instellen van voorgrondkleur (x) en achtergrondkleur (y). Beide waarden lopen van 0 t/m F (hexadecimaal). `VDP(14)=&Hxy` - Instellen van de knippertijd in perioden van 1/5 seconde. X is de aan-tijd en met y wordt de uit-tijd ingesteld. Ook deze waarden lopen van 0 t/m F.
Dit voorbeeldprogramma toont het gebruikt van CMDBLINK:

```
10 CLEAR 1000,&HD000
20 POKE &HFE0D,201
30 BLOAD"CMDBLINK.BIN",R
40 SCREEN 0:WIDTH 80:'Werkt alleen in 80*24
tekstmode
50 CMD CLRBLK
60 VDP(13)=&H11: VDP(14)=&H11
70 FOR I=0 TO 22 STEP 2
80 CMD BLINK I*2,I,35
90 NEXT I
```

Als de command driver als DATA-regels achterin het BASIC-programma geplaatst wordt, dan moet regel 30 worden vervangen door: `30 GOSUB x` waarbij x het regelnummer voorstelt van de initialisatie-routine. De regels 10 en 20 zijn, zoals eerder besproken, nodig om conflicten te voorkomen.

CMDSRAM.BIN

(alleen voor MSX turbo R) Bezitters van een MSX turbo R weten vast wel dat er 16 kB SRAM extra in hun computer aanwezig is. Het handige van SRAM is dat de inhoud van dat geheugen altijd bewaard blijft, ook als de computer wordt uitgeschakeld. In het MSX turbo R artikel van MSX Mozaik nr 33 heb ik beloofd een programma te maken om op een eenvoudige manier het SRAM te benaderen. CMDSRAM.BIN voegt twee nieuwe BASIC-commando's toe, namelijk: `CMD RDSRAM adres, variabele` en `CMD WRSRAM adres,data`
Het eerste commando leest een byte uit het SRAM, waarbij het adres en de variabele waarin de data moet worden geplaatst opgegeven moeten worden. Het adres mag lopen van 0 t/m 16383 (decimaal) of van &H0000 t/m &H3FFF (hex). De data is altijd 8-bits breed, dus van 0 t/m 255 of van &H00 t/m &HFF. Het is wel noodzakelijk dat de variabele een INTEGER variabele is! Dus als we de inhoud van adres 0 in variabele A willen hebben, gaat dat als volgt:

```
DEFINT A: 'Definieer variabele A als een INTEGER variabele
(Een DEFINT opdracht kan het beste boven in een BASIC-
programma worden geplaatst).
CMD RDSRAM 0,A: 'Nu zal A de inhoud van SRAM adres 0
bevatten
```

Het andere nieuwe commando (WRSRAM) kan een adres van het SRAM voorzien van nieuwe data. Als in adres 1000 het getal 25 moet komen, dan gaat dat zo:

```
CMD WRSRAM 1000,25: 'Eenvoudiger kan toch niet?
```

Het is gelukkig ook mogelijk om in plaats van een constant getal, een variabele toe te passen, mits het een numerieke variabele is en de inhoud ervan binnen de hierboven aangegeven grenzen zit. Zo kan er een LOOP gemaakt worden die het SRAM in één keer wist:

```
10 CLEAR 1000,&HD000
20 POKE &HFE0D,201
30 BLOAD"CMDSRAM.BIN",R
40 FOR I=0 TO &H3FFF
50 CMD WRSRAM I,0
60 NEXT I
```

Een voorbeeldprogramma om tekst in het SRAM te plaatsen:

```
10 CLEAR 1000,&HD000
20 POKE &HFE0D,201
30 BLOAD"CMDSRAM.BIN",R
40 A$="Onverwoestbare tekst.":AD=0
50 FOR I=1 TO LEN(A$)
60 CMD WRSRAM AD,ASC(MID$(A$,I,1))
70 AD=AD+1:NEXT I
```

De variabele AD wijst naar het eerste adres van het SRAM en A$ bevat de te schrijven tekst. Met een dergelijk soort programma kan de tekst weer uit het SRAM gelezen worden:

```
10 CLEAR 1000,&HD000
20 POKE &HFE0D,201
30 BLOAD"CMDSRAM.BIN",R
40 DEFINT A ;A moet INTEGER zijn
50 T=21:AD=0 ;T=aantal, AD=startadres
60 FOR I=AD TO AD+T-1 ;lus
70 CMD RDSRAM I,A ;A <- (SRAM adres I)
80 PRINT CHR$(A); ;print ASCII karakter
90 NEXT I
```

De mogelijkheden zijn bijna onbeperkt, maar voor diegenen die liever niet zelf programmeren, staat een complete SRAM-editor op de bijgeleverde diskette (SRAM.BAS). Dit programma gebruikt trouwens beide in dit artikel besproken command drivers gelijktijdig!
Opmerking: Men moet niet vreemd opkijken als bepaalde SRAM-geheugenadressen opeens zijn veranderd. Dit komt doordat de MSX turbo R bepaalde dingen in het SRAM bijhoudt. Gelukkig zijn dit slechts enkele bytes. Echter, wanneer het ingebouwde Japanse software-pakket opgestart wordt door middel van het knopje naast de RESET knop (of door `CALL MENU` op de MSX2+ of `CALL HIRO` op de MSX turbo R), dan kan de inhoud van meerdere geheugen-bytes worden geruïneerd! Pas dus op met het opslaan van belangrijke gegevens!

CMDSRAM listing

```
BEGAD:      EQU   0DA00H
HCMD: EQU   0FE0DH      ;CMD hook
CHRGTR:     EQU   04666H      ;Haalt 1 byte uit BASIC text
GETBYT:     EQU   0521CH      ;Haalt 1-byte integere expressie op
FRMQNT:     EQU   0542FH      ;Haalt 2-byte integere expressie op
PTRGET:     EQU   05EA4H      ;Bereken opslagadres van variabele

      ORG   BEGAD-7
      DEFB  0FEH
      DEFW  START,EINDE,START

      ;initialiseren CMD routine

START:      LD    HL,HCMD     ;Hook CMD command
      LD    DE,OLDCMD
      LD    BC,5
      LDIR

      LD    HL,NEWCMD
      LD    (HCMD+1),HL
      LD    A,0C3H      ;Opcode voor JP
      LD    (HCMD),A

      RET         ;Naar BASIC

NEWCMD:     CALL  VOERUIT
OLDCMD:     DEFS  5     ;ook eventuele andre CMD's uitvoeren.
      RET

VOERUIT:    PUSH  HL
      LD    IX,CMDNAME  ;CMD naam is "WRSRAM"
      CALL  CHECK
      JP    Z,WRSRAM    ;schrijven naar SRAM
      POP   HL
      PUSH  HL
      LD    IX,CMDNAME2 ;CMD naam is "RDSRAM"
      CALL  CHECK
      JP    Z,RDSRAM    ;lezen van SRAM
      POP   HL
      RET         ;terug, geen goede CMD naam.

      ;vergelijkt naam met naam in BASIC text
CHECK:      DEC   HL
CHECKB:     LD    A,(IX+0)
      OR    A
      RET   Z     ;CMD naam komt overeen, z=1
      CALL  CHRGTR
      CP    (IX+0)
      INC   IX
      JR    Z,CHECKB
      RET         ;CMDNAME fout, z=0

CMDNAME:    DEFB  "WRSRAM",0
CMDNAME2:   DEFB  "RDSRAM",0
SRAMADRES:  DEFW  0     ;SRAM-adres voor lezen/schrijven
VARIADRES:  DEFW  0     ;wijst naar opslagadres variabele
RDWRFLAG:   DEFB  0     ;0=schrijven, 1=lezen
```

```
RDSRAM:     LD    A,1
      LD    (RDWRFLAG),A
      JR    SCAN

WRSRAM:     XOR   A
      LD    (RDWRFLAG),A

SCAN: INC   HL
      POP   AF
      POP   AF
      POP   AF
      CALL  FRMQNT      ;haal 2-byte integere expressie op
      LD    (SRAMADRES),DE    ;Adres waarin geschreven gaat worden
      DEC   HL
      CALL  CHRGTR      ;Haal byte uit BASIC-text
      CP    ","             ;test op komma
      RET   NZ    ;terug (error!) als geen komma
      INC   HL

      LD    A,(RDWRFLAG)
      OR    A
      JR    Z,GETDATA   ;spring als SRAM schrijf-opdracht
      PUSH  HL
      CALL  PTRGET      ;bereken opslag-adres van variabele
      LD    A,(0F663H)
      CP    2
      JR    Z,CORRECVAR
      POP   HL
      RET         ;Terug (error!) als geen integer variabele
CORRECVAR:  POP   AF
      LD    (VARIADRES),DE    ;opslagadres van variabele opslaan
      PUSH  HL
      JR    VERDER

GETDATA:    CALL  GETBYT      ;haal integere expressie, A=DATA
      PUSH  HL    ;HL wijst naar vervolg BASIC programma
      PUSH  AF

VERDER:     LD    HL,(SRAMADRES)
      LD    E,128
      BIT   5,H   ;Kies onderste of bovenste 8kB SRAM
      JR    Z,BLOK1
      INC   E
BLOK1:      PUSH  HL
      LD    HL,06800H
      LD    A,10001111B
      CALL  00014H      ;(6800)<-E in slot 3-3
      POP   HL

      LD    A,H
      AND   00011111B
      LD    H,A
      LD    DE,04000H
      ADD   HL,DE
      LD    A,(RDWRFLAG)
      OR    A
      LD    A,10001111B
      JR    Z,WRTDATA   ;Spring als SRAM schrijfopdracht
      CALL  0000CH
      LD    HL,(VARIADRES)
```

```
      LD    (HL),A
      INC   HL
      LD    (HL),0
      JR    STOPPEN
WRTDATA:    POP   DE    ;haal data
      LD    E,D
      CALL  00014H
STOPPEN:    LD    A,10001111B
      LD    E,12
      LD    HL,06800H
      CALL  00014H
      POP   HL    ;pointer wijst naar vervolg BASIC prog.
      RET         ;naar BASIC interpreter

EINDE:      NOP
```

CMDBLINK listing

```
BEGAD:      EQU   0D900H
HCMD: EQU   0FE0DH      ;CMD hook
CHRGTR:     EQU   04666H      ;Haalt 1 byte uit BASIC text
GETBYT:     EQU   0521CH      ;Haalt 1-byte integere expressie op

      ORG   BEGAD-7
      DEFB  0FEH
      DEFW  START,EINDE,START

      ;initialiseren CMD routine

START:      LD    HL,HCMD     ;Hook CMD command
      LD    DE,OLDCMD
      LD    BC,5
      LDIR

      LD    HL,NEWCMD
      LD    (HCMD+1),HL
      LD    A,0C3H      ;Opcode voor JP
      LD    (HCMD),A

      RET         ;Naar BASIC

NEWCMD:     CALL  VOERUIT
OLDCMD:     DEFS  5
      RET

VOERUIT:    PUSH  HL
      LD    IX,CMDNAME  ;CMD naam is "CLRBLK"
      CALL  CHECK
      JP    Z,CLEARBLK  ;blink-tabel wissen
      POP   HL
      PUSH  HL
      LD    IX,CMDNAME2 ;CMD naam is "BLINK"
      CALL  CHECK
      JP    Z,FILLBLK   ;blink-tabel bewerken
      POP   HL
      RET         ;terug, geen goede CMD naam.

      ;vergelijkt naam met naam in BASIC text
CHECK:      DEC   HL
CHECKB:     LD    A,(IX+0)
      OR    A
      RET   Z     ;CMD naam komt overeen, z=1
      CALL  CHRGTR
      CP    (IX+0)
      INC   IX
      JR    Z,CHECKB
      RET         ;CMDNAME fout, z=0

CMDNAME:    DEFB  "CLRBLK",0
CMDNAME2:   DEFB  "BLINK",0

      ;Wist de blinking tabel
CLEARBLK:   INC   HL
      POP   AF
      POP   AF
      POP   AF
      PUSH  HL
```

```
        LD    HL,00800H
        LD    BC,270
        XOR   A
        CALL  00056H      ;Fill VRAM
        POP   HL
        RET

FILLBLK:    INC   HL
        POP   AF
        POP   AF
        POP   AF
        CALL  GETBYT      ;Haal integere expressie
        LD    (KOLOMNR+1),A
        DEC   HL
        CALL  CHRGTR
        CP    ","             ;test op komma
        RET   NZ
        INC   HL
        CALL  GETBYT      ;haal integere expressie
        LD    (REGELNR+1),A
        DEC   HL
        CALL  CHRGTR
        CP    ","             ;test op komma
        RET   NZ
        INC   HL
        CALL  GETBYT      ;haal integere expressie
        LD    (LENGTE+1),A
        PUSH  HL    ;HL wijst naar vervolg BASIC programma

        ;regelnr. en kolomnr. omrekenen naar eerste VRAM adres

REGELNR:    LD    HL,0  ;L=Regelnr.
        ADD   HL,HL ;regel*2
        PUSH  HL
        POP   DE
        ADD   HL,HL ;regel*4
        ADD   HL,HL ;regel*8
        ADD   HL,DE ;regel*10
        LD    BC,00800H   ;offset van blink-tabel
        ADD   HL,BC

KOLOMNR:    LD    BC,0  ;C=kolomnr.
        LD    A,C
        AND   7
        SRL   C
        SRL   C
        SRL   C
        ADD   HL,BC
        LD    E,1
        INC   A
FILLBLKB:   RRC   E
        DEC   A
        JR    NZ,FILLBLKB
LENGTE:     LD    B,0
        LD    A,B
        OR    A
        JR    Z,STOP      ;een lengte van 0 is ongeldig
FILLBLKC:   CALL  0004AH
        XOR   E
        CALL  0004DH
        SRL   E
```

```
      JR    NZ,FILLBLKD
      LD    E,128
      INC   HL
FILLBLKD:   DJNZ  FILLBLKC
STOP: POP   HL
      RET
EINDE:      NOP
```