file ibmkey.doc 29 oktober 1989
Een IBM-keyboard aansluiten aan Dos65.
Literatuur: De 6502 kenner
december 1989
door W.Schimmel

Het probleem van een IBM toetsenbord is de seriele uitgang.
In de genoemde literatuur is dat opgelost door de bits serieel in te lezen en m.b.v. software om te zetten naar een byte.
De oplossing van het betreffende artikel veroorzaakt echter nieuwe problemen:

a) Het ontbreken v.e. buffer is soms erg vervelend.
b) Aangezien er in Dos65 meerdere interrupts lopen gaat het inlezen v.d. bits niet altijd goed.
c) Het toetsenbord dat ik heb levert een signaal dat een beetje anders is dan in het genoemde artikel: De klokpuls die 560 us voorafgaat aan de data is er niet en daardoor is de cpu vaak te laat om de data nog goed in te lezen.

Daarom is nu gekozen voor een oplossing met hardware, maar dan zo eenvoudig mogelijk.

Het omzetten van het nummer v.d. toets naar een ascii-code blijft dus wel in het programma, het grote voordeel is de enorme flexibiliteit.
In de tabel die gebruikt wordt voor die omzetting is hier en daar een detail gewijzigd. Dat betreft dan hoofdzakelijk de toetsen in het numrieke deel aan de rechterkant:

toets opdruk	1	2	3	4	5	6	7	8	9	0
gewoon-ascii	19	18	1A	13	00	04	14	05	17	00
numlock-ascii	31	32	33	34	35	36	37	38	39	30

Het doel daarvan is: nu men hoeft de EDITOR niet te veranderen.
Nu blijven alle andere programma's gewoon werken, zonder enige aanpassing.
Nb: het programma voor de omzetting is iets veranderd zodat een toets waar code $00 bij staat, niet werkt. Er wordt dan dus geen toets doorgegeven aan Dos65.
De functietoetsen F1..F10 kunt u eventueel naar eigen wens invullen.

De hardware.
Die bestaat hoofdzakelijk uit een schuifregister, type 74LS164.
Daarbij is nog nodig een inverter 74LS00 en een monoflop 74LS123.
Deze schakeling zal ook werken met een toetsenbord dat wel een een startpuls afgeeft voordat de data-bits komen.

Opmerkingen:
-Er is ook nog geprobeerd om het schuifregister in de VIA te gebruiken maar dat werkte niet. Vermoedelijk stopt die na 8 bits en het toetsenbord verstuurt er negen. Ook schuift de via op de verkeerde flank v.d. klok.
-Op mijn toetsenbord ziet een toets "Sys Req" en die blijkt niets te doen, niet aangesloten misschien ?
-Het programma kan op twee manieren geassembleerd worden:

test=0 De definitieve versie.
test=1 Dan is het oude toetsenbord aan VIA1 en het nieuwe aan VIA2. Het programma schakelt dan direct terug als er een toets op het oude bord wordt aangeraakt. Deze versie is wel ca 80 bytes langer.

```
;file ibm keyboard test
;19890221 hs see DE 6502 KENNER  december 1987.
;19891024 hs aangepast aan parralel input ( dat vereist extra hardware ).
;19891027 hs maken eigen omzetting van toetsnummer naar ascii code.
;19891028 hs tot nu toe alles uitgeprobeert op via2 en het werkt.
;19891029 hs test/difinitief gemaakt met conditioneel assembleren.
;
        lib     doslib
        opt     rc02
;
initorg equ     $a000   initialisiatie routine
intorg  equ     $bd00   interrupt routine
;
test    equ     0       test=0= alleen ibm toetsenbord, aan VIA1. VIA2=vrij.
;test   equ     1       test=1= ibm toetsenbord aan VIA2 en normale aan VIA1.
;                               en de hardware heeft nog geen timer.
        if      test==0
via     equ     $E100   first via
        else
via     equ     $E110   second via
        endif
viapad  equ     via+1   poort a data
viadir  equ     via+3   datadirection a
viaacr  equ     via+11  aux.contr.register
viapcr  equ     via+12  per.contr.register
viaifr  equ     via+13  int.flag register
viaier  equ     via+14  int.enable reg.
;
keypnt  equ     $E7B7   lengte v.d. text in de inputbuffer
conint  equ     $CB2B   in dos65 staat hier LDA $E101
;
;het programma werkt niet volgens het oorspronkelijke
;artikel in DE 6502 KENNER (dus seriele input) maar
;er is extra hardware waardoor de serieele informatie
;van het toetsenbord wordt omgezet in  bytes paralel
;die dan direct kunnen worden ingelezen v.d. VIA ingang.
;
;ascii converter
numlock equ     $EF     binair: 1110 1111
caplock equ     $F7     binair: 1111 0111
shift   equ     $FD     binair: 1111 1101
control equ     $FE     binair: 1111 1110
alt     equ     control         dan doen alt en control precies hetzelfde
;alt    equ     $FB     binair: 1111 1011
nul     equ     $00     binair: 0000 0000
        ;constants declared in doslib.mac:
;BS     = $08       backspace
;TAB    = $09   ^I  horizontal tabulation
;LF     = $0A       line feed
;VT     = $0B       vertical tab (inverse line feed)
;CR     = $0D   ^M  carriage return
;escape = $1B       escape
;clrs   = $0C       clear screen (form feed)
;space  = $20       space
;delete = $7F       delete a character
        ;some key's used in the EDitor:
;  ^Q    = $1D   verlaat edit mode
;  ^S    = $13   links                       ^D      = $04   rechts
;  ^Z    = $18   beneden                     ^E      = $05   boven
;  ^F    = $06   woord vooruit               ^A      = $01   woord terug
```

```
; ^Y     = $19   einde regel                ^T     = $14   begin regel
; ^Z     = $1A   scherm vooruit             ^W     = $17   scherm terug
; ^B     = $02   delete woord               ^N     = $0E   delete rest v.d. regel
; ^O     = $0F   spring tab's terug         ^P     = $10   spring tab's
; ^U     = $15   delete regel in buffer     ^R     = $12   reprint oude regel
;
        org     $A000
inibm   php
        sei
        lda     #0
        sta     viadir  maak allemaal ingangen
        lda     viapcr  ca1 = negative active edge
        and     #$FE
        sta     viapcr
        if      test==0
        lda     viaacr  no timers and no shift reg., use latch
        ora     #$01
        sta     viaacr
        else
        lda     viaacr  no timers and no shift reg. and no latch
        and     #$FE
        sta     viaacr
        endif
        lda     #$82    CA1 interrupt
        sta     viaier
        lda     #0
        sta     tstnd   reset variabelen
        sta     keypnt
        lda     #$20     "JSR INTORG" maken in dos65
        sta     conint
        ldxa    #intorg
        stxa    conint+1
        if      test==1
        ldxa    #conint-1       EXTRA   enable via2 ca1 interrupt
        stxa    $E768           EXTRA
        ldxa    #oldkeyb-1      EXTRA i.v.m. oude keyboard
        stxa    $E75A           EXTRA  VIA1-CA1 is v.d. oude situatie
        endif
        plp
        rts             eind van initialisatie
;
        org     intorg
        lda     #$00
        sta     viaier          disable interrupts van CA1
        if      test==1
        ldx     #200    wacht 200x5= 1000 us, want zolang duurt een byte
2       nop             1us     1               de hardware schuift de byte
        nop             1us     2               in een schuifregister
        nop             1us     3               dat dan parralel uitgelezen
        dex             1us     4               kan worden.
        bne     2.b     1us     5
        endif
        lda     viapad          get byte and clear CA1 flag
        jsr     ascii
        bcc     tstuit
        pla                     fout
        pla
tstuit  ldx     #$82            goed
        stx     viaier          enable interrupts van CA1
        rts
```

```
;ASCII
ascii   php
        and     #$7F
        tay
        ldx     (tascii-1),y
        plp
        bpl     maak
        cpx     #$E0
        bcc     brkend          normale toetsen: doe niets
        cpx     #$f8
        bcc     lockkey
        txa                     shift of alt of cntrl : zet het bit uit
        and     tstnd
        sta     tstnd
        bra     brkend
lockkey txa                     numlock of caplock : inverteer "toggle" bit
        eor     #$FF
        eor     tstnd
        sta     tstnd
brkend  sec
        rts
maak    cpx     #nul
        beq     spclend         nul (=niet gedefinieerd ): doe niets
        cpx     #$E0
        bcc     gnctrl
        cpx     #$F8
        bcc     spclend         numlock of caplock    : doe niets
        txa                     shift of alt of cntrl : zet het bit aan
        eor     #$FF
        ora     tstnd
        sta     tstnd
spclend sec
        rts
gnctrl  cpy     #$47
        bcc     gnmrk
        lda     tstnd                   numeriek deel
        and     #^numlock!^shift        is de numlock functie aan?
        beq     mkend                           (d.i. numlock exor shift)
        cmp     #^numlock!^shift
        beq     mkend
        ldx     (tnmlock-$47),y
        bra     mkend
gnmrk   cpx     #'a'                    normale toetsen
        bcc     gnlttr
        cpx     #'z'+1
        bcs     gnlttr
        lda     #^caplock!^shift        het is een letter: caplock belangrijk
        bit     tstnd
        beq     mkend
        bra     keyshif
gnlttr  lda     #^shift                 het is geen letter: alleen shift
        bit     tstnd
        beq     mkend
keyshif ldx     (tshift-1),y
mkend   lda     #^control
        bit     tstnd
        beq     einde
        cpx     #$20            als contol ingedrukt is
        blt     einde           en als toetscode < $20
        txa
```

```
        and     #$1F            dan code := ascii AND $1F
        tax
einde   txa
        clc
        rts
;
tascii  fcc     escape,'123d567'
        fcc     '890-=',delete,TAB
        fcc     'qwertyuiop'
        fcc     '[]',CR,control
        fcc     'asdfghjkl'
        fcc     ';','\'',`,shift,'\\'
        fcc     'zxcvbnm,./'
        fcc     shift,'*',alt,space,caplock
                ;functie toetsen F1..F10:
        fcc     escape,nul,nul,nul,nul,nul,nul,nul,CR,CR
                ;toetsen in numerieke deel:
        fcc     numlock,nul,$14,$05,$17,'-'
        fcc     $13,nul,$04,'+',$19,$18,$1A,space,$07
tshift  fcc     escape,'!@#$%^&*()_+',delete,TAB
        fcc     'QWERTYUIOP'
        fcc     '{}',CR,nul
        fcc     'ASDFGHJKL'
        fcc     ':"~',nul,'|'
        fcc     'ZXCVBNM<>?'
        fcc     nul,'*',nul,space,nul
                ; shift+functie toetsen F1..F10:
        fcc     escape,nul,nul,nul,nul,nul,nul,nul,nul,nul
                ; control+numm.toets:
        fcc     nul,nul,$14,$05,$17,'-'
        fcc     $13,nul,$04,'+',$19,$18,$1A,space,$07
                ; numlock+numm.toets:
tnmlock fcc     '789-456+1230.'
tstnd   fcc     0                       ;initieele waarde nul=alles uit
;
        if      test==1         allen in testfase nodig:
oldkeyb lda     $E101           terugschakelen naar oude toetsenbord
        php
        sei
        lda     #0
        sta     viaier  disable via2 interrupts
        lda     #$AD
        sta     conint   "LDA $E101" terugzetten in dos65
        ldxa    #$E101
        stxa    conint+1
        ldxa    #conint-1
        stxa    $E75A           VIA1-CA1 =keyboard hersteld
        ldxa    #$F0F2-1
        stxa    $E768           ibm keyboard uitschakelen
        plp
        rts
        endif
;
        end     inibm
```

```
;file ibm keyboard
;19890221 hs see DE 6502 KENNER  december 1987.
;19891024 hs aangepast aan parralel input ( dat vereist extra hardware ).
;19891027 hs maken eigen omzetting van toetsnummer naar ascii code.
;19891028 hs tot nu toe alles uitgeprobeert op via2 en het werkt.
;19891029 hs test/difinitief gemaakt met conditioneel assembleren.
;19891224 hs alle overbodige ballast eruit gegooid
;             en probeer er functietoetsen in te bouwen
;
        lib     doslib
        opt     rc02
;
intorg  equ     $BD00   interrupt routine
;
via     equ     $E100   first via
viapad  equ     via+1   poort a data
viadir  equ     via+3   datadirection a
vi^cr   equ     via+11  aux.contr.register
viapcr  equ     via+12  per.contr.register
viaifr  equ     via+13  int.flag register
viaier  equ     via+14  int.enable reg.
;
keypnt  equ     $E7B7   Dos65 lengte v.d. text in de inputbuffer
keybuf  equ     $E7B8   Dos65 input buffer
conint  equ     $CB2B   in dos65 staat hier LDA $E101
;                       dat wordt nu JSR intorg
;
;het programma werkt niet volgens het oorspronkelijke
;artikel in DE 6502 KENNER (dus seriele input) maar
;er is extra hardware waardoor de serieele informatie
;van het toetsenbord wordt omgezet in  bytes paralel
;die dan direct kunnen worden ingelezen v.d. VIA ingang.
;
;ascii converter
numlock equ     $EF     binair: 1110 1111
caplock equ     $F7     binair: 1111 0111
sh^t    equ     $FD     binair: 1111 1101
control equ     $FE     binair: 1111 1110
alt     equ     control         dan doen alt en control precies hetzelfde
;alt    equ     $FB     binair: 1111 1011
fun     equ     $80     binair: 1000 0000       19891223, zie tabel  tfun
nul     equ     $00     binair: 0000 0000
        ;constants declared in doslib.mac:
;BS     = $08       backspace
;TAB    = $09   ^I  horizontal tabulation
;LF     = $0A       line feed
;VT     = $0B       vertical tab (inverse line feed)
;CR     = $0D   ^M  carriage return
;escape = $1B       escape
;clrs   = $0C       clear screen (form feed)
;space  = $20       space
;delete = $7F       delete a character
        ;some key's used in the EDitor:
; ^Q    = $1D   verlaat edit mode
; ^S    = $13   links                       ^D      = $0d   rechts
;       = $18   beneden                     ^E      = $05   boven
; ^F    = $06   woord vooruit               ^A      = $01   woord terug
; ^Y    = $19   einde regel                 ^T      = $1d   begin regel
; ^Z    = $1A   scherm vooruit              ^W      = $17   scherm terug
  ^^            ^^^^^^^^^^^^^             ^N      = $0E   delete rest v.d. regel
```

```
; ^O     = $0F    spring tab's terug        ^P      = $10    spring tab's
; ^U     = $15    delete regel in buffer    ^R      = $12    reprint oude regel
;
        ;Initialisiatie routine
        ;is alleen nodig om de interrupt vector te veranderen
        ;en om  VIA goed in te stellen.
        ;
        org     $A000
ibmkey  php
        sei
        lda     #0
        sta     viadir  maak allemaal ingangen
        lda     viapcr  ca1 = negative active edge
        and     #$FE
        sta     viapcr
        lda     viaacr  no timers and no shift reg., use latch
        ora     #$01
        sta     viaacr
        lda     #$82    CA1 interrupt
        sta     viaier
        lda     #0
        sta     tstnd   reset variabelen
        sta     keypnt
        lda     #$20     "JSR INTORG" maken in dos65
        sta     conint
        ldxa    #intorg
        stxa    conint+1
        plp
        rts             eind van initialisatie
;
        ;Interrupt routine
        ;Deze blijft in het geheugen en mag niet overschreven worden.
        ;
        org     intorg
        lda     #$00
        sta     viaier          disable interrupts van CA1
        lda     viapad          get byte and clear CA1 flag
        jsr     ascii
        bcs     fout
        cmp     #fun
        blt     tstuit          was gewone ascii of control code
        jsr     functie         was een funtietoets
fout    pla                     fout
        pla
tstuit  ldx     #$82            goed
        stx     viaier          enable interrupts van CA1
        rts
;ASCII
ascii   php
        and     #$7F
        tay
        ldx     (tascii-1),y
        plp
        bpl     maak
        cpx     #$E0
        bcc     brkend          normale toetsen: doe niets
        cpx     #$f8
        bcc     lockkey
        txa                     shift of alt of cntrl : zet het bit uit
        ora     #$80
```

```
         sta     tstnd
         bra     brkend
lockkey  txa                     numlock of caplock : inverteer "toggle" bit
         eor     #$FF
         eor     tstnd
         sta     tstnd
brkend   sec
         rts
maak     cpx     #nul
         beq     spclend         nul (=niet gedefinieerd ): doe niets
         cpx     #$E0
         bcc     gnctrl
         cpx     #$F8
         bcc     spclend         numlock of caplock    : doe niets
         txa                     shift of alt of cntrl : zet het bit aan
         eor     #$FF
         ora     tstnd
         sta     tstnd
spclend  sec
         rts
gnctrl   cpy     #$47
         bcc     gnmrk
         lda     tstnd                   numeriek deel
         and     #^numlock¦^shift        is de numlock functie aan?
         beq     mkend                           (d.i. numlock exor shift)
         cmp     #^numlock¦^shift
         beq     mkend
         ldx     (tnmlock-$47),y
         bra     mkend
gnmrk    cpx     #'a'                    normale toetsen
         bcc     gnlttr
         cpx     #'z'+1
         bcs     gnlttr
         lda     #^caplock¦^shift        het is een letter: caplock belangrijk
         bit     tstnd
         beq     mkend
         bra     keyshif
gnlttr   lda     #^shift                 het is geen letter: alleen shift
         bit     tstnd
         beq     mkend
keyshif  ldx     (tshift-1),y
mkend    lda     #^control
         bit     tstnd
         beq     einde
         cpx     #$20            als contol ingedrukt is
         blt     einde           en als toetscode < $20
         txa
         and     #$1F            dan code := ascii AND $1F
         tax
einde    txa                     nu is ACCU=ascii-code
         clc
         rts
;
tascii   fcc     escape,'123d567'
         fcc     '890-=',delete,TAB
         fcc     'qwertyuiop'
         fcc     '[]',CR,control
         fcc     'asdfghjkl'
         fcc     ';',"'",'`',shift,'\\'
         fcc     'zxcvbnm,./'
```

```
        fcc     shift,'*',alt,space,caplock
                ;functie toetsen F1..F10:
        fcc     escape,fun+2,fun+3,fun+4,nul,nul,nul,nul,CR,CR
                ;toetsen in numerieke deel:
        fcc     numlock,nul,$14,$05,$17,'-'
        fcc     $13,nul,$04,'+',$19,$18,$1A,space,$07
tshift  fcc     escape,'!@#$%^&*()_+',delete,TAB
        fcc     'QWERTYUIOP'
        fcc     '{}',CR,nul
        fcc     'ASDFGHJKL'
        fcc     ':"~',nul,'|'
        fcc     'ZXCVBNM<>?'
        fcc     nul,'*',nul,space,nul
                ; shift+functie toetsen F1..F10:
        fcc     escape,nul,nul,nul,nul,nul,nul,nul,nul,nul
                ; control+numm.toets:
        fcc     nul,nul,$14,$05,$17,'-'
        fcc     $13,nul,$04,'+',$19,$18,$1A,space,$07
                ; numlock+numm.toets:
tnmlock fcc     '789-456+1230.'
tstnd   fcc     0                       ;initieele waarde nul=alles uit
                ;
                ;19891223 hs. De functietoetsen plaatsen hun code
                ;direct in de inputbuffer.
                ;Het totale aantal characters v.d. funcietoetsen is max 128
                ;Aan het eind moet steeds een nul staan
                ;functietoetsen, F1..F10
        ;tfunx is een index tabel naar de functies
tfuny   fcc     F1-tfun,F2-tfun,F3-tfun,F4-tfun,F5-tfun,F6-tfun,F7-tfun
        fcc     F8-tfun,F9-tfun,FA-tfun
tfun
F1      fcc     0
F2      fcc     $1E,'S',0       toggle status regel
F3      fcc     $1E,'I',0       wijzig input devices
F4      fcc     $1E,'O',0       wijzig output devices
F5      fcc     0
F6      fcc     0
F7      fcc     0
F8      fcc     0
F9      fcc     0
FA      fcc     0

functie sec             fun<= Accu <$E0  en  Accu-fun is het functie nummer
        sbc     #fun
        tay
        lda     tfuny-1,y
        tay             Y point to first character
        ;copy the string to the input buffer
        ldx     keypnt  make x point in inputbuffer
4       cpx     #27     max key buffer
        bge     7.f
        lda     tfun,y
        beq     8.f
        sta     keybuf,x

        jsr     prch    debug

        iny
        beq     7.f
        inv
```

```
        bra     4.b
7       sec
        bra     9.f
8       stx     keypnt
        clc
9       rts
;
        end     ibmkey
```