Grafisch display

Monitor voor de KIM

Deel 4

M. Dohmen
R. Koekoek

In het voorgaande hebben we een aantal routines beschreven die nodig waren voor de monitor. De behandeling van een speciale routine is echter achterwege gelaten. Deze routine, de toetsophaalroutine, wordt nu in zijn geheel besproken.

Zoals u wellicht is opgevallen, kwamen in een aantal routines (CURSOR, WAIT en WIS LIJN) een drietal NOP's voor. We zeiden dat deze later zouden worden vervangen door een JSR-instructie. Nu zouden we deze NOP's kunnen vervangen door JSR KEYIN, welke begint op adres $E000 (deze is besproken in deel 1). We zouden ons er dan echter te gemakkelijk vanaf maken. Een sprong naar die toetsophaalroutine in programma's als CURSOR, WAIT en WIS LIJN zou erg onlogisch lijken. De toetsophaalroutine die we hebben gemaakt bestaat uit drie delen:

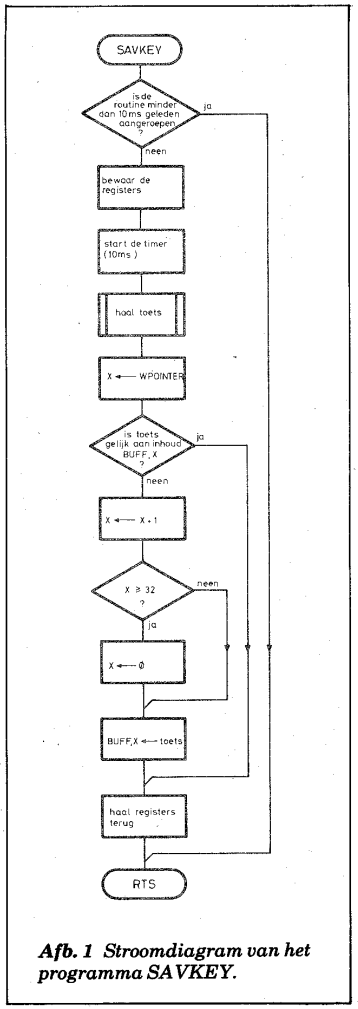
1. Haal toets op en bewaar hem.
2. Creëer repeat-functie.
3. Voer bewaarde toetsen uit.

Uit de praktijk is gebleken dat veel computers geen repeatfunctie kennen en dat dit een groot gemis is; denk aan de de toets „CRLF" of de „+"-toets van de KIM. Een tweede nadeel, dat gevonden is bij het uitproberen van diverse systemen, is dat, wanneer een toets wordt ingedrukt tijdens het uitvoeren van een programma (bijvoorbeeld TREK LIJN of WIS SCHERM), deze niet wordt geïnterpreteerd door de computer. Ook wanneer meerdere toetsen na elkaar worden ingedrukt, wordt in het uiterste geval alleen de eerste uitgevoerd. De hier beschreven toetsophaalroutine bezit een automatische repeat-functie. Verder is het mogelijk om tot een totaal van zestien toetsen op te slaan, terwijl een programma loopt, die na afloop ook weer na elkaar worden uitgevoerd. Op het eerste gezicht lijkt zestien toetsen veel, maar als u bedenkt dat het tekenen van een figuur op het beeldscherm relatief lang duurt, is het toch wel gewenst om tot zestien toetsen te kunnen „onthouden".

Toetsophaalroutine

In afb. 1 ziet u het stroomdiagram van dit programma. Er wordt gebruik gemaakt van de timer. Dit is gedaan om de routine wat sneller te laten verlopen. In de routine WAIT wordt deze routine bijvoorbeeld vijfmaal aangeroepen. Dit is vrij snel achter elkaar. Om te voorkomen dat voor niets de routine opnieuw moet worden uitgevoerd is er een timer die de tijd bepaalt tussen het aanroepen van de routine en het aanroepen van de routine de keer ervoor. Werd hij minder dan 10 ms geleden aangeroepen dan heeft het geen nut om naar het toetsenbord te kijken. Is het langer dan 10 ms geleden dan worden de registers op de stack gezet en wordt de timer opnieuw gestart. Vervolgens wordt naar het toetsenbord gekeken. Dit gebeurt twee keer om dender te onderdrukken. Nu we de toets hebben binnengehaald, moet deze nog worden bewaard. Dit kan slechts op één manier, namelijk met behulp van een FIFO-buffer. FIFO is de afkorting voor een Engelse term: First In First Out, wat zoveel betekent als: de waarde die als eerste de buffer in ging, moet er ook weer als eerste uit. Dit in tegenstelling tot de stack van een microproccessor, wat een LIFO-buffer is (Last In First Out). Voor de FIFO is nodig:

1. Een stukje geheugenruimte.
2. Een „wijzer" die aangeeft waar de laatst uitgelezen waarde staat (RPOINT).



Afb. 1 Stroomdiagram van het programma SAVKEY.

Afb. 2 Stroomdiagram van het programma RESKEY.
Lijst 1 Programma SAVKEY.
Lijst 2 Programma RESKEY.

3. Een „wijzer" die aangeeft waar de laatst ingelezen waarde staat (WPOINT).

In de routine SAVKEY hebben we met deze twee pointers nog niets te maken. We zetten gewoon de toets op de eerst volgende lege plaats, als het tenminste een nieuwe toets is. Want stel dat de toets nog was ingedrukt. De toets zou dan opnieuw op een plaats in de buffer worden gezet. Het gevolg zou zijn dat binnen 0,1 s de hele FIFO-buffer met één toetswaarde zou zijn gevuld. Maar we zitten met nog een probleem. Stel dat de toets tweemaal achter elkaar wordt ingedrukt. De vorige toets was dan gelijk aan de nieuwe, waardoor de laatste niet in de buffer zou worden gezet. Dit euvel kan op twee manieren worden opgelost:

1. In de toetsophaalroutine wordt gewacht totdat de toets weer is losgelaten. Een eerste nadeel is echter dat door het wachten andere programma's worden opgehouden. Een tweede nadeel is dat de repeat-functie niet gemakkelijk meer mogelijk is.
2. Er wordt aangegeven dat er een nieuwe toets is ingedrukt. (In Basic komt u iets soortgelijks tegen bij de CR-LF-toets om aan te geven dat er een nieuwe regel wordt ingevoerd.) Een nadeel is hier dat er na elke toets door middel van een andere toets moet worden aangegeven dat er een nieuwe toets ingedrukt gaat worden (denk aan de „+"-toets op de KIM die het volgende adres selecteert).

Toch hebben we gekozen voor methode 2, hoewel daar de nadelen groter lijken. Dit nadeel is echter uit de weg geruimd door een nieuwe toets te creëren, namelijk de „geen"-toets. Elke keer als een toets wordt ingedrukt volgt daarna automatisch de „geen"-toets. In de toetsophaalroutine wordt namelijk $FF in A gezet als er geen toets was ingedrukt. Door de „geen"-toets als een aparte toets te beschouwen blijft het geheel „real time", en is repeat-functie mogelijk. In plaats van 16 bytes wordt de FIFO-buffer nu wel 32 bytes lang, omdat na elke toets ook $FF moet worden opgeslagen om aan te geven dat de toets was losgelaten. Het hele programma is nu eenvoudig te begrijpen (lijst 1).

REPEAT of SAVKEY

Het tweede programma, dat we in dit deel bespreken is heel wat lastiger te begrijpen dat het eerste. Hier zijn namelijk twee programma's in elkaar verweven. Een vlag bepaalt welk van de twee programma's wordt doorlopen:

1. Creëer de repeat-functie.
2. Voer SAVKEY uit.

Deze vlag, aangeduid met TELLER, bevindt zich op adres $0081. Is de waarde $00 dan wordt SAVKEY uitgevoerd, hetgeen inhoudt dat de toets wordt opgehaald die het eerst in de buffer werd gezet. Is TELLER ongelijk aan nul dan wordt REPEAT uitgevoerd. Het hierin toegepaste principe is ongeveer gelijk aan dat van de Miniassembler voor de 6502 (RB, maart en april 1981), waarin een toets meerdere functies kreeg afhankelijk van de tijd dat deze was ingedrukt, wat door een teller werd bijgehouden. Op deze manier wordt hier bepaald hoe snel de repeat verloopt. Als TELLER gelijk is aan nul, is er geen repeat. Bij de nu volgende uitleg van het programma gaan we daar even van uit.

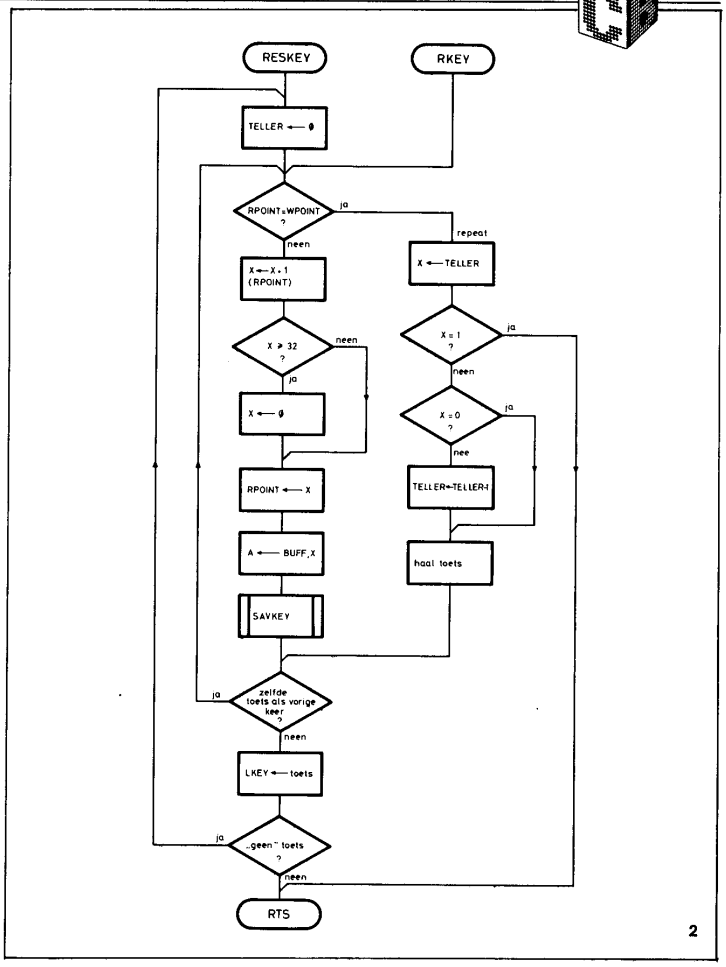
Lijst 1

```
                1150 ;   ROUTINE BEWAAR TOETS
                1160 ;
                1170 ;
E800- 2C 07 17  1180 SAVKEY     BIT TIMER    ;ZIJN DE 10 MS AL OM?
E803- 10 2E     1190            BPL RET      ;
E805- 48        1200            PHA  ;BEWAAR
E806- 8A        1210            TXA  ;REGISTERS
E807- 48        1220            PHA  ;OP DE
E808- 98        1230            TYA  ;STACK
E809- 48        1240            PHA  ;
E80A- A9 0A     1250            LDA #$0A     ;START TIMER OPNIEUW
E80C- 8D 07 17  1260            STA TIMER    ;(10.24 MS)
E80F- 20 20 02  1270 DEBON      JSR GETKEY   ;INDIRECTE SPRONG NAAR
                1280            ;TOETSOPHAALROUTINE VIA RAM
E812- 85 84     1290            STA *SKEY    ;OM DENDER TE
E814- 20 20 02  1300            JSR GETKEY   ;ONDERDRUKKEN
E817- C5 84     1310            CMP *SKEY    ;LOSGELATEN?
E819- D0 F4     1320            BNE DEBON    ;NEEN, OPNIEUW
E81B- A6 83     1330            LDX *WPOINT  ;IS HET AANGEWEZEN
E81D- DD 00 02  1340            CMP BUFF,X   ;KARAKTER UIT DE
E820- F0 0C     1350            BEQ OUT      ;BUFFER GELIJK?
E822- E8        1360            INX  ;NEEN, VOLGENDE PLAATS
E823- E0 20     1370            CPX #$20     ;BUFFER VOL?
E825- 90 02     1380            BCC BUFNU    ;NEEN, ZET TOETS OF 'GEEN TOETS'
E827- A2 00     1390            LDX #$00     ;RESET BUFFER
E829- 9D 00 02  1400 BUFNU      STA BUFF,X   ;IN DE BUFFER
E82C- 86 83     1410            STX *WPOINT  ;BEWAAR POINTER
E82E- 68        1420 OUT        PLA  ;HAAL
E82F- AA        1430            TAX  ;REGISTERS
E830- 68        1440            PLA  ;TERUG
E831- A8        1450            TAY  ;VAN DE
E832- 68        1460            PLA  ;STACK
E833- 60        1470 RET        RTS  ;KEER TERUG
                1480 ;
                1490 ;
                1500            .LS
                1504 ;
0220- 4C 00 E0  1505 GETKEY     JMP TOETS    ;INDIRECTE ADRESSERING VAN
                1506 ;                       TOETSOPHAALROUTINE VIA RAM
```

Lijst 2

```
                1530 ;   ROUTINE VOOR REPEAT-FUNCTIE
                1540 ;   OF OPSLAAN VAN EEN TOETS
                1550 ;
                1560 ;
                1570 ;
E87D- A0 00     1580 RESKEY     LDY #$00     ;GEEN REPEAT
E87F- 84 81     1590 RKEY       STY *TELLER  ;ZET TELLER
E881- A6 82     1600 SAMKEY     LDX *RPOINT  ;ZIJN DE POINTERS
E883- E4 83     1610            CPX *WPOINT  ;AAN ELKAAR GELIJK?
E885- F0 12     1620            BEQ REPT     ;ZO JA, REPEAT
E887- E8        1630            INX  ;NEEN, VOLGENDE BUFFERPLAATS
E888- E0 20     1640            CPX #$20     ;BUFFER VOL?
E88A- 90 02     1650            BCC NFBUF ;
E88C- A2 00     1660            LDX #$00     ;JA, RESET BUFFER
E88E- 86 82     1670 NFBUF      STX *RPOINT  ;ZET POINTER
E890- BD 00 02  1680            LDA BUFF,X   ;HAAL TOETS UIT BUFFER
E893- 20 00 E8  1690            JSR SAVKEY   ;NAAR BEWAARROUTINE
E896- 4C B2 E8  1700            JMP EXIT     ;TERUG
E899- A6 81     1710 REPT       LDX *TELLER  ;HAAL TELLER
E89B- CA        1720            DEX  ;WAS TELLER = 1?
E89C- F0 1C     1730            BEQ RTRN     ;ZO JA, TERUG
E89E- E8        1740            INX  ;WAS TELLER = 0?
E89F- F0 02     1750            BEQ BPREV    ;ZO JA, VOLGENDE
E8A1- C6 81     1760            DEC *TELLER  ;NEEN, REPEAT
E8A3- 20 20 02  1770 BPREV      JSR GETKEY   ;HAAL TOETS
E8A6- 85 84     1780            STA *SKEY    ;BEWAAR DE TOETS
E8A8- 20 23 02  1790            JSR SROUT    ;VOER MEM UIT
E8AB- 20 20 02  1800            JSR GETKEY   ;VOORKOM DENDEREN
E8AE- C5 84     1810            CMP *SKEY ;
E8B0- D0 F1     1820            BNE BPREV ;
E8B2- C5 80     1830 EXIT       CMP *LKEY    ;WAS HET DETELEFTE TOETS?
E8B4- F0 CB     1840            BEQ SAMKEY   ;ZO JA, OPNIEUW
E8B6- 85 80     1850            STA *LKEY    ;ZET TOETS OPST
E8B8- 30 C3     1860            BMI RESKEY   ;WAS HET $FF?
E8BA- 4C 26 02  1870 RTRN       JMP TPKEY    ;NEEN, EINDE
```

In afb. 2 ziet u het stroomdiagram. Eerst wordt gekeken of de twee FIFO-pointers (RPOINT en WPOINT) aan elkaar gelijk zijn. Is dat het geval dan was de laatste toets de vorige keer al opgehaald en is de FIFO-buffer dus leeg. Aangezien de repeat-functie alleen mogelijk is bij de laatst ingedrukte toets, wordt nu gekeken of deze aanstaat. Het X-register wordt geladen met de waarde in TELLER en vervolgens vergeleken met $01 (E89B: CA; F0 1C). Waarom er met $01 wordt vergeleken, wordt bij de behandeling van de repeat-functie verder uitgelegd.
Vervolgens wordt gekeken of X gelijk is aan $00. Zo ja, dan is er niets speciaals aan de hand en wordt een toets opgehaald en in de buffer opgeslagen. Is deze toets dezelfde als de vorige dan wordt terug gesprongen naar RKEY, omdat aan deze toets een repeat-functie kan zijn gekoppeld.
Is de toets niet gelijk aan de vorige dan wordt deze bewaard in LKEY (last key) en vervolgens vergeleken met „geen"-toets. Was er geen toets ingedrukt dan wordt teruggesprongen naar het begin om de volgende toets uit de FIFO-buffer op te halen. Was er wel een toets ingedrukt dan kan deze in een ander programma worden uitgevoerd.



2

Repeat-functie

Stel dat TELLER niet gelijk was aan $00, maar aan bijvoorbeeld $20 (dit kan voorkomen wanneer in plaats van naar RESKEY naar RKEY wordt gesprongen). Als we dan óók nog aannemen dat de FIFO-buffer leeg was, komen we bij het stukje programma, in het stroomdiagram aangegeven met REPEAT. Hier wordt X geladen met de waarde in TELLER (deze bevat $20). De repeat-functie staat dus aan, X is nog geen $01 en ook niet $00. Nu wordt TELLER met één verlaagd (TELLER–1→TELLER;=$1F) en wordt er naar A gesprongen. De toets was ingedrukt en blijft ingedrukt. Hier wordt de subroutine SAVKEY aangeroepen, maar omdat de toets ingedrukt bleef, wordt deze niet in de buffer opgeslagen. De toets is dus nog steeds gelijk aan de vorige. Er wordt weer teruggesprongen naar RKEY. Dit gaat zo nog $1E-maal door. U ziet het; afhankelijk van de waarde in TELLER wordt er een aantal keren een lus doorlopen die kijkt of de toets nog is ingedrukt. We gaan verder wanneer TELLER gelijk geworden is aan $01. RPOINT is nog steeds gelijk aan WPOINT. Nu wordt X geladen met $01. Zou TELLER nogmaals worden verlaagd dan zou dat betekenen dat de repeat-functie uit wordt gezet, terwijl dat niet de bedoeling is. Daarom wordt X vergeleken met $01. X is inderdaad $01 en er wordt dus naar adres $0226 gesprongen. Zoals u wellicht is opgevallen worden er in de routines RESKEY en SAVKEY sprongen uitgevoerd naar stukjes RAM ($0220 en verder). Dit heeft een reden. Op adres $0220 moet staan: 4C 00 E0. Heeft u echter een ander toetsenbord dan moet hier een sprong naar uw eigen toetsophaalroutine staan. Dit moet wel een routine zijn die ook terugkeert als er geen toets is ingedrukt en wel met $FF in A. Op adres $0223 moet een sprong staan naar een stukje vrije geheugenruimte in EPROM of RAM. In die geheugenruimte kunnen dan verschillende routines worden aangeroepen, zoals WAIT, CURSOR of zaken die u nodig heeft voor spelletjes enz. De computer springt dan telkens hier naar toe, wanneer hij niets te doen heeft. Op adres $0226 moet een sprong staan naar een stukje geheugen in EPROM of RAM, waar de toets die was ingedrukt wordt opgehaald. In het eenvoudigste geval wordt hier gezet: $0226 A5 80 60. Telkens als de toetsophaalroutine een toets heeft, wordt hier naar toe gesprongen.

Mogelijkheden

Er volgt nog een illustratie van wat met beide behandelde routines allemaal mogelijk wordt. Allereerst kan aan elke toets bij repeat een eigen herhalingsfrequentie worden gegeven. Ik kan mij bijvoorbeeld indenken dat een toets om een punt op het scherm te laten bewegen een hogere frequentie vereist dan de toets LF. Er volgen vier voorbeelden van het gebruik van deze toetsophaalroutine.

Gelijke herhalingsfrequenties voor elke toets

Hiervoor moet het volgende programma worden gebruikt, zie tabel 1. Het werkt als volgt. Wanneer de eerste keer een toets wordt ingedrukt, is TELLER gelijk aan $00, omdat de repeat-functie nog niet is gebruikt. Bij het doorlopen van bovenstaande routine wordt er $50 in TELLER gezet, zodat de herhalingsfrequentie laag is. Bij de tweede keer is TELLER gelijk aan $01, zodat $10 in TELLER wordt gezet.

Tabel 1

```
PRGVB1   LDYIM   $10      herhalingsfrequentie
         LDA     TELLER
         BNE     VERDER
         LDYIM   $50      tijdsduur tot het begin
                          van de repeat-functie
VERDER   JMP     RKEY
```

Tabel 2

```
REPOFF   LDYIM   $00
ENTRY    JMP     RKEY
         CMPIM   $01
         BNE     ETC
         JSR     TOETS1   voer functie uit
REP1     LDYIM   $10      hoge herhalingsfrequentie
         LDA     TELLER
         BNE     VERD2
         LDYIM   $50
VERD2    JMP     ENTRY
ETC      enzovoort
```

Tabel 3

```
ETC      CMPIM   $03
         BNE     ETC1
         JSR     TOETS3
         JMP     REPOFF
ETC1     enzovoort
```

Tabel 4

```
ETC1     CMPIM   $5F      „rub out"-toets?
         BNE     ETC2     geen „rub out"
         LDA     TELLER
         BNE     LANG     $00, dus geen repeat
         JSR     KORT5F   wis karakter
         LDYIM   $50      langzame repeat-functie
         JMP     ENTRY    blijft de toets ingedrukt?
LANG     JSR     LANG5F   wis hele regel
         LDYIM   $40      langzame repeat-functie
         JMP     ENTRY
ETC2     enzovoort
```

Tabel 5

```
LANG     JSR     LANG5F   wis hele regel
         JMP     REPOFF
         ETC2    enzovoort
```

Verschillende herhalingsfrequenties per toets

Hiervoor is het volgende programma geschikt, zie tabel 2.
Nadat de functie, verbonden aan toets 1, is uitgevoerd, wordt in dit programmavoorbeeld de repeat-functie opnieuw gestart met een waarde in Y, afhankelijk van de herhalingsfrequentie. Per toets is apart de herhalingsfrequentie en de tijd, voordat de repeat-functie start, in te stellen.

Toetsen zonder repeat-functie

Dit programmadeel sluit aan op het vorige voorbeeld en geeft de mogelijkheid toetsen zonder repeat-functie uit te voeren, zie tabel 3.

Tweede functie

Als laatste een zeer praktische toepassing, welke u al bent tegengekomen in de Mini-assembler voor de 6502, namelijk een tweede functie per toets. Als daarin een toets kort werd ingedrukt, werd de zoekmode aangezet (GO-toets) en als hij lang werd ingedrukt werden de spronginstructies omgezet. Een dergelijk programma is hier te verwezenlijken met dit verschil, dat beide functies worden uitgevoerd als de toets lang blijft ingedrukt en alleen de eerste functie als deze kort wordt ingedrukt. Hierbij is het ook nog mogelijk aan de toets de repeat-functie te koppelen. Als voorbeeld nemen we de functie „rub out". Bij kort indrukken wordt alleen het laatste karakter gewist. Bij constant indrukken de rest van de regel, waarna aan de „rub out"-toets een langzame repeat-functie ($40) wordt gekoppeld om de regels daarboven een voor een te wissen, zie tabel 4.
Wilt u bij de tweede functie van de „rub-out"-toets geen repeat-mogelijkheid dan wordt het programma, vanaf het label LANG, zoals het in tabel 5 is weergegeven.
U ziet, u kunt met deze twee routines alle kanten op. Er wordt in de monitor dan ook uitgebreid geprofiteerd van deze mogelijkheden.