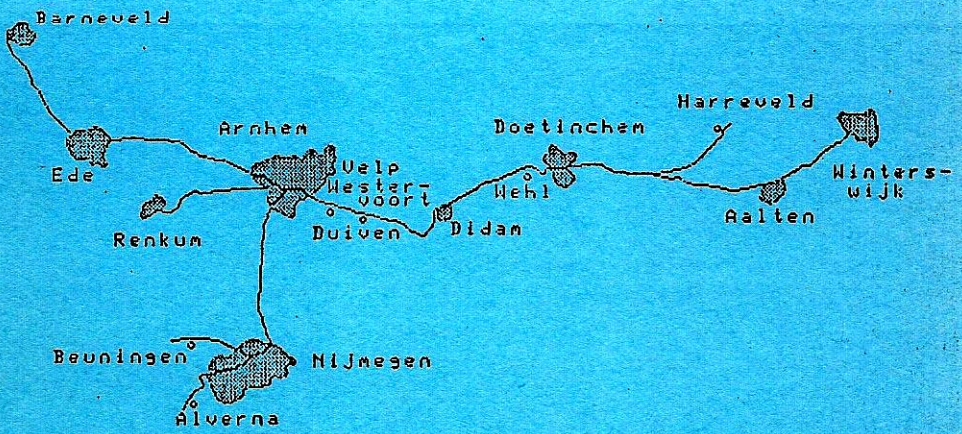
11

# Tempus Fugit

knipselkrant van de Acorn Computerclub
Regio Arnhem

EDAKTIE : G. Bouwman



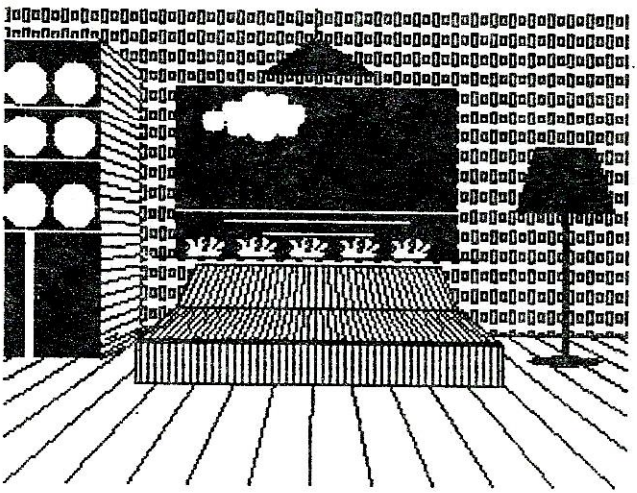

maart 1986

Gebruikte literatuur:


| | | |
|---|---|---|
| Acorn manual | | |
| Atom-ware 1 & 2 & 3 | van | HENK REINDERS |
| Junior computer | | Elektuur |
| Microcomputers A tot Z | | Immerzeel |
| Microcomputertechniek | | Cremers. |

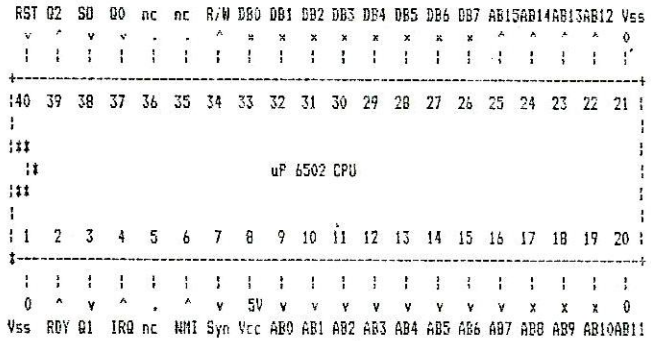

Samenstelling:


Andre Wessels Harreveld 31 december 1985.

# De CPU 6502 pinaansluitingen.

De pinaansluitingen van de CPU 6502 zijn in figuur 1 gegeven. De adresbussen zijn aangesloten op de pinnen 9 tot en met 20 en 22 tot en met 25, de databus op de pinnen 26 tot en met 33.
Vss (0V) is verbonden met de pinnen 1 en 21 en Vcc (+5V+/-5%, absoluut maximum is 7,0V) op pin 8.
Voor de kloksignalen zijn pin 3 (Q1), 37 (Q0) en 39 (Q2) gereserveerd (zie '^' uit- en 'v' ingangssignalen).
De ingang RDY (ReaDY) op pin 2 wordt toegepast indien EPROM's of EAROM's worden gebruikt. Die geheugens hebben een betrekkelijke lange toegangstijd en kunnen niet worden toegepast bij sistemen die op volle snelheid werken. Door de RDY-pin laag te maken tijdens de impuls van Q1 en R/W=1 (leesopdracht), wordt de fetchcyclus van de procesor vertraagd, totdat de data uit het geheugen beschikbaar is op de datalijnen.
Het Sync-signaal uit pin 7 wordt gebruikt om de fetchcyclus van de processor te signaleren. Deze pin is alleen hoog gedurende de operatiecode fetchcyclus (instruktiecyclus Von Neumann).
Pin 38 SO (Set Overflow) maakt het mogelijk de V-vlag in het CCR (Conditie Code Register) te beinvloeden.
De pinnen 5, 35 en 36 zijn niet aangesloten.




figuur 1.

# Pin NMI & RST & IRQ.

Met de ingang IRQ (Interrupt ReQuest) op pin 4 kan een lopend programma onderbroken worden voor een interrupt. Een interrupt wordt veroorzaakt als deze pin laag gemaakt wordt terwijl de I-vlag in het CCR '0' is. Dit interruptsignaal is 'maskeerbaar' en wordt toegestaan als I-vlag=1.
Op pin 6, NMI (Non Maskable Interrupt), kan een interrupt worden gegeven. Ook nu moet de pin laag gemaakt worden. Dit interrupt is echter niet maskeerbaar (dus NIET tegen te houden!) en wordt onafhankelijk van de stand van de I-vlag in het CCR uitgevoerd.

MEERDERE INTERRUPT-AANVRAGERS.

Meestal zijn er meerdere interrupt-aanvragers. In dit geval krijgt elke onderbreker een bepaalde prioriteit toegekend. Hoe hoger de prioriteit des te sneller men wordt behandeld. Een belangrijk verschil tussen IRQ en NMI is dat IRQ via en door het programma kan worden genegeerd (denk aan het bordje 'niet storen.'), een NMI wordt altijd doorgelaten (denk aan 'brandmelding!').

NESTEN VAN INTERRUPTS.

Wanneer het programmatechnisch mogelijk is of gewenst kunnen we ook interrupts toelaten in de ISR (Interrupt Service Routine). We moeten dan in deze ISR een CLI (CLear Interrupt) instruktie aanbrengen. Net als bij subroutines is het dan mogelijk om interrupts te nesten. We moeten dan wel zorgen dat de Stack voldoende diep is daar het gevaar van stackoverflow hierbij groot is.

## "Mag ik even STOREN?"

Wat gebeurt er nu in de ACORN ATOM?

*De NMI-vektor.*

De inhoud van de NMI-vektor (#FFFA & #FFFB) is het startadres van de NMI-routine op #FFC7, welke weer een vektoradres in RAM heeft op adres #0200 & #0201 (laag-hoog). Dit RAM-vektoradres wordt tijdens de RST-routine geladen met de inhoud #E87B (alleen als de DOSROM aanwezig is).
De NMI-pin is binnen de ACORN ATOM niet aangesloten doch is wel aanwezig op de bus PL6/7 pin 29. De disc maakt wel gebruik van de NMI-pin.
De NMI-routine heeft zijn startadres van de NMI-vektor en ziet er alsvolgt uit.

```
FFC7 PHA            zet accu inhoud op stack.'
FFC8 JMP(#0200)     spring naar het adres dat staat op
                    #0200 (#7B) & #0201 (#E8) dus#E87B.
                    (alleen bij disc)
```

Eerst wordt gekeken of het interrupt-request bit in het statusregister van de FDC (Floppy Disc Controller) gezet is. Gezet betekent dat er iets aan de hand is, of dat de opdracht voltooid is. In beide gevallen moet het resultaat-register bekeken worden.

```
E87B LDA#0A00       lees het statusregister van de FDC.
E87E AND @#04       test op interrupt request bit.
E880 BEQ#E88B       indien gezet,resultaat-register bekijken
E882 INC #F6        verhoog lage byte laad cq. save adres
E884 BNE#E88B       als geen overscrijding paginagrens naar
                    Zero-Page-routine
E886 INC #F7        verhoog hoge byte laad cq. save adres
E888 JMP#00F2       ga naar Zero-Page-routine.
```

Eerst registers en accu veilig zetten dan resultaat bekijken.

```
E88B TXA          X-register naar Accu
E88C PHA          X-register naar Stack
E88D TYA          Y-register naar Accu
E88E PHA          Y-register naar Stack
E88F CLD          clear Decimaal vlag
E890 JSR#E899     test resultaat-register
E893 PLA          haal Y-register
E894 TAY          zet terug
E895 PLA          haal X-register
E896 TAX          zet terug
E897 PLA          haal Accu van Stack
E898 RTI          einde interrupt.
Interrupt Request Routine.
E899 JMP(#00D5)       spring naar het adres wat staat op
                      de geheugenplaatsen #00D5 & #00D6.
```

*De RST-vektor.*

De inhoud van de RST-vektor (#FFFC & #FFFD) is het startadres van de RST-routine op #FF3F, welke weer een vektoradres in RAM heeft op adres #0202 & #0203 (laag-hoog). Dit RAM-vektoradres wordt tijdens de RST-routine geladen met de inhoud #C9D8 .
De RST-pin is binnen de ACORN ATOM aangesloten op de BREAK-toets, VIA 6522 en de ACIA 8255. Deze lijn wordt door de BREAK-toets laag gemaakt zodat de uP6502 gereset wordt. Door de RESET-routine op #FF3F gebeurt het volgende.

```
FF3F LDX @#17         laadt X-register met een index
FF41 LDA#FF9A,X       laadt een vectorbyte
FF44 STA#0204,X       breng deze naar zero-page
FF47 DEX              verlaag index
FF48 BPL#FF41         alle vektoradressen gehad?
FF4A TXS              X=#FF op Stack
FF4B TXA              Accu = #FF
FF4C INX              X=0
FF4D STX#EA           zet DOS/COS op nomon
FF4E STX#E1           zet cursor uit
FF51 STX#E7           zet lock uit
FF53 LDX @#33         laadt index
FF55 STA#02EB,X       zet array adressen op 0
FF58 DEX              verlaag index
FF59 BPL#FF55         alles op 0
FF5B LDA @#0A         laadt een linefeed
FF5D STA#FE           zet weg voor printerroutine (#FF04)
FF5F LDA @#8A         laadt poortinstelling
FF61 STA#B003         stel poorten ACIA in
FF64 LDA @#07         laadt poortinstelling
FF66 STA#B002         stel poort ACIA in
FF69 JSR#F7D1         schrijf ACORN ATOM op het scherm
FF6C-FF7B             tekst + kontrolekodes
```

```
FF7C LDA @#82          laadt text-pagina
FF7E STA #12           stel text-page-pounter in
FF80 CLI               reset interrupt-vlag
FF81 LDA @#55          laadt kontrole karakter
FF83 STA#2901          zet deze in het geheugen
FF86 CMP#2901          kontrole of het erin staat
FF89 BNE#FF91          staat er niet in,dus geen geheugen
FF8B ASL               vermenigvuldig kontrolekarakter
FF8C STA#2901          zet het er nog een keer in
FF8F CMP#2901          dubbele kontrole op geheugen
FF92 BNE#FF97          spring als er geen geheugen is
FF94 JMP#C2B2          textpointer op #2900 ga naar basic-
FF97 JMP#C2B6          textpointer op #8200 -interpreter
FF9A-FFB1              bevat vektoradressen van routine #FF3F.
```

Op #C9D8 start de BREAK-ROUTINE. Via de vektoren op #0202 & #0203 zijn we hier aangekomen. De statement-pointer wordt op #C9E8 gezet en met een jump naar de interpreteer-routine wordt de basic-regel die daar staat, uitgevoerd.

```
C9D8 PLA               haal processor statusvan Stack
C9D9 PLA               haal lage byte programmateller
C9DA STA#00            zet deze weg als error nummer
C9DC LDA#10            laadt lage byte error-handler
C9DE STA#05            zet deze naar statement pointer
C9E0 LDA#11            laadt hoge byte error-handler
C9E2 STA#06            zet deze naar statement pointer
C9E4 JMP#C2F2          interpreteer regel
```

```
C9E8-CA23 @=1;P.#6$7"ERROR "?0;@=8;IF?1!?2P." LINE"!1?#FFFF;E.
```

De volgende routine kijkt of een FP aanwezig is.

```
CA24 JSR#C424          kijk FP aanwezig
CA27 BCC#C6F2          niet aanwezig error 29
CA29 JMP(#D004)        ga door met het adres in de FP.
```

*De IRQ-vektor.*

De inhoud van de IRQ-vektor (#FFFE & #FFFF) is het startadres van de IRQ-routine op #FFB2, welke weer een vektoradres in RAM heeft op adres #0204 & #0205 (laag-hoog). Dit RAM-vektoradres wordt tijdens de RST-routine geladen met de inhoud #A000 .
De IRQ-pin is binnen de ACORN ATOM aangesloten op de VIA 6522 door LK2 net boven dit IC, PL8 via LK3 en op PL6/7 op pin 28.
Deze lijn kan door een van deze laag gemaakt worden zodat de uP6502 gaat zoeken via de IRQ-vektor naar de IRQ routine op #FFB2.

```
FFB2 STA#FF            zet de accu veilig
FFB4 PLA               haal processorstatus van Stack
FFB5 PHA               zet hem terug op Stack
FFB6 AND @#10          kijk of de Break-vlag is gezet
FFB8 BNE#FFC0          niet gezet dan naar interrupt-routine
FFBA LDA#FF            haal de accu inhoud terug
FFBC PHA               zet op Stack
FFBD JMP(#0204)        naar interrupt-routine
FFC0 LDA#FF            haal de accu inhoud terug
FFC2 PLP               haal processorstatus van Stack
FFC3 PHP               zet terug op Stack
FFC4 JMP(#0202)        ga naar foutmeldings-routine.
```

Op pin 40 vinden we de RST (ReSeT) ingang. Deze dient onder andere voor het resetten van de registers in de CPU. Bij het inschakelen zijn de inhouden van deze registers volkomen willekeurig. De processor moet nu echter in een wachtcyclus, zijn sisteem-programma doorlopen en wachten op gegevens voor het laden van het geheugen met een programma. Een wachtcyclus is bijvoorbeeld "de toetsenbordscan". Door de pin RST laag te maken en daarna weer hoog, wordt na zes klokcycli voor het initieren van de processor (registers in de beginstand plaatsen, b.v. de Stackpointer op #FF) de adresbus op #FFFC gezet. Op dit adres staat de laagstwaardige byte van het startadres voor het sisteemprogramma. Deze byte wordt in de programma-counter 'laag' geladen en de adreslijnen gaan op #FFFD. Op dit adres staat de hoogstwaardige byte van het startadres dat in de programma-counter 'hoog' geladen wordt. De inhoud van de programma-counter komt nu op de adreslijnen en de processor gaat het programma, beginnend op dat adres, afwerken. Zowel het startadres in #FFFC en #FFFD als het sisteemprogramma moeten in ROM aanwezig zijn.
De processor wordt dus met de RST-pin gestart. Het omlaag maken van de RST kan al geschieden tijdens het inkomen van de voedingsspanning. Is de voedingsspanning ingeschakeld dan moet na tenminste 2 klokcycli de RST-pin hoog worden. De processor start dan automatisch na het aanzetten van het apparaat.
Hier is dus op twee geheugenplaatsen een adres gegeven waar de eerste instruktie staat van een programmma. De inhoud van deze geheugenplaatsen worden een 'vektor' genoemd.
De uP 6502 CPU kent 3 vektoren:

De NMI vektor op #FFFA en #FFFB.
De RST vektor op #FFFC en #FFFD.
De IRQ vektor op #FFFE en #FFFF.

Bij een NMI-interrupt wordt de lopende instruktie in de CPU EERST voltooid, de programma-counter en het CCR worden op de STACK gezet en de inhoud van de geheugenplaatsen #FFFA en #FFFB worden in de processor geladen. Deze data vormen het adres voor het interrupt programma in de volgorde laag-hoog. Dit interruptprogramma wordt doorlopen. Aan het eind hiervan vinden we de instruktie RTI (ReTurn from Interrupt). De oorspronkelijke inhouden van de programma-counter en het CCR worden weer van de STACK gehaald zodat de processor op het adres verder kan gaan waar hij het lopende hoofdprogramma had verlaten.
Door het laag maken van de pin NMI volgt maar 1 keer een interrupt, ook al wordt deze pin verder omlaag gehouden. Om opnieuw een interrupt te kunnen veroorzaken moet de pin eerst hoog worden gemaakt en dan weer laag.
Dit is niet het geval als een interrupt wordt gegeven met de pin IRQ. Deze blijft een interrupt geven zolang de pin laag gehouden wordt. Daarom wordt nog voordat naar het IRQ-interruptprogramma wordt gesprongen, (startadres #FFFE en #FFFF in de volgorde laag-hoog) de interruptvlag I gezet (I=1). Bij de instruktie RTI wordt de I-vlag weer 0 zodat het zaak is VOOR die tijd de IRQ-pin hoog te maken. Anders vindt opnieuw een interrupt plaats.

Het statement DIAL en de daarbij behorende kies automaat

Om met het laatste tebeginnen, Na een aantal ontwerpjes voor een kiezer in atommieuws gezien te hebben, ben ik de kast maar eens ingedoken en heb m'n eigen ontwerp te voorschijn gehaald.
Dit ontwerp voldoet volgens mij nog het beste aan de eisen van de PTT,het kies gedeelte zelf is nl. een exacte kopie van een telefoontoestel.
Verder is er ook nog gedacht aan kiestoon detectie.

Het schema is grofweg in drie gedeeltes te verdelen nl.

1.het gedeelte boven de 5volt lijn.Dit is het telefoongedeelte van links naar rechts gezien komt men de volgende onderdelen tegen.
Relais A.;dit is het haakcontact.Op de (p)contacten wordt de telefoonlijn aangesloten,op de (nc) contacten wordt uw eigen telefoon aangesloten,op de (no) contacten komt de kiesunit.
Relais B.dit is het impulscontact en is normaal gesloten, tijdens het kiezen van een telefoon nummer gaat dit afwisselend open en dicht.parallel over dit contact staat een serieschakeling van C1 en R1,dit is een vonkenblus keten.
Relais C.;dit is het kortsluitcontact,dat er voor zorgt dat tijdens de kies impulsen de trafo 1 kortgesloten is; dit is nodig omdat tijdens het kiezen de telefooncentrale een kortsluiting of een open lijn moet zien.
Trafo 1.;deze vervangt de spreek/hoorspoel uit de telefoon.

Ter verduidelinking volgen nu een viertal tekeningetjes wat de tel.centr. electrisch gesproken op een tel.lijn kan 'zien'.

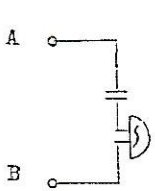


fig.1

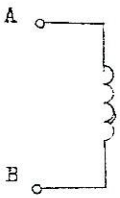


fig.2

fig.3

fig.4

figuur 1 is aanwezig als de haak van de telefoon er op ligt,
figuur 2 is aanwezig als u aan het spreken bent.
figuur 3 en figuur 4 zijn afwisselend aanwezig als u de kiesschijf teruglaat lopen als de schif weer in rust is
is fig.2 weer aan de orde.

2.over het tweede stuk valt weinig te vertellen.
dit is de aansturing van de relais'. Relais 1 wordt aangestuurd via een transistor vanwege z'n stroomverbruik(100mA) relais B. is vanwege de dubbele inverter normaal gesloten, relais C. is normaal geopend.

3.Het laatste gedeelte,dat elektrisch gezien achter TR.1 zit en in het schema rechts,zorgt voor de kiestoon detectie op de lijn.

Het kiesmodem wordt aangelöten op de VIA 6522 port B pen 4-7 dit kan men natuurlijk wijzigen maar dan zal ook de software aangepast moeten worden.

Het statement "DIAL".

Met DIAL is van uit een programma of direct vanaf het toetsenbord het statement aan te roepen.
Het tebellen telefoonnummer moet als een string worden ingevoerd.
Dus : DIAL"085-634842";of DIAL $A;of DIAL $AA(I).

Tijdens het kiezen test het prog. of het karakter een streepje is en/of een getal.
Als het kar. een streepje is wordt er middels een JSR naar de testroutine voor kiestoon gesprongen;als het prog. hier onverhoopt blijft hangen(geen tel.lijn aangesloten) is er middels de control uittekomen.
Als het kar.geen getal is komt het prog. terug met een error melding en wordt het kiezen afgebroken en de tel.lijn weer vrij gemaakt.

Dan volgt nu nog de onderdelenlijst van de kiezer.

| | |
|---|---|
| R1=560 | T1=2N1613 |
| R2=1K3 | T2=BC547B |
| R3=301K | T3=BC557B |
| R4=1M | |
| R5=100K | D1=LN4001 oid |
| R6=1K | D2=1N4001 oid |
| | D3=1N4001 oid |
| C1=1uF | |
| C2=10nF | IC1=7406 |
| C3=4uF,16V | IC2=74132 |

RL A=printrelais 5 volt 100mA 2xwissel
RL B= clare prme 15002
RL C=

TR1= lijntrafo 1:1 300 Ohm

R.J.Bronsveld
Reinaldstraat 18
6824-GN ARNHEM
085-634842

2.over het tweede stuk valt weinig te vertellen.
dit is de aansturing van de relais'. Relais 1 wordt aangestuurd via een transistor vanwege z'n stroomverbruik(100mA) relais B. is vanwege de dubbele inverter normaal gesloten, relais C. is normaal geopend.

3.Het laatste gedeelte,dat elektrisch gezien achter TR.1 zit en in het schema rechts,zorgt voor de kiestoon detectie op de lijn.

Het kiesmodem wordt aangelöten op de VIA 6522 port B pen 4-7 dit kan men natuurlijk wijzigen maar dan zal ook de software aangepast moeten worden.

Het statement "DIAL".

Met DIAL is van uit een programma of direct vanaf het toetsenbord het statement aan te roepen.
Het tebellen telefoonnummer moet als een string worden ingevoerd.
Dus : DIAL"085-634842";of DIAL $A;of DIAL $AA(I).

Tijdens het kiezen test het prog. of het karakter een streepje is en/of een getal.
Als het kar. een streepje is wordt er middels een JSR naar de testroutine voor kiestoon gesprongen;als het prog. hier onverhoopt blijft hangen(geen tel.lijn aangesloten) is er middels de ~~control~~ uittekomen.
Als het kar.geen getal is komt het prog. terug met een error melding en wordt het kiezen afgebroken en de tel.lijn weer vrij gemaakt.

Dan volgt nu nog de onderdelenlijst van de kiezer.

R1=560
R2=1K3
R3=301K
R4=1M
R5=100K
R6=1K

C1=1uF
C2=10nF
C3=4uF,16V

T1=2N1613
T2=BC547B
T3=BC557B

D1=LN4001 oid
D2=1N4001 oid
D3=1N4001 oid

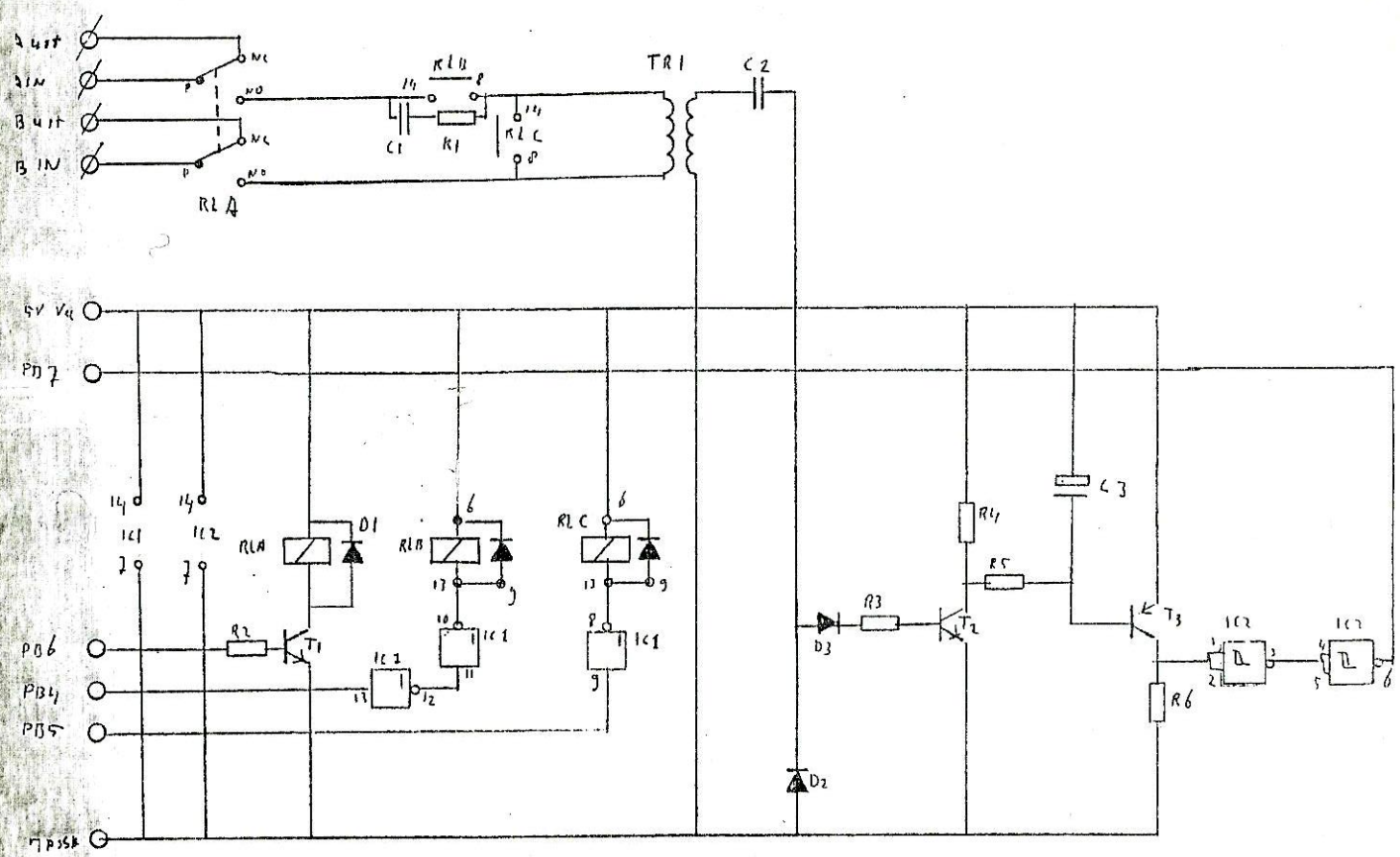
IC1=7406
IC2=74132

RL A=printrelais 5 volt 100mA 2xwissel
RL B= clare prme 15002
RL C=

TR1= lijntrafo 1:1 300 Ohm

R.J.Bronsveld
Reinaldstraat 18
6824-GN ARNHEM
085-634842

```
10 PROGRAM DIAL
20P.$15
30 PASS2;GOS.a
40 PASS1;GOS.a
50P.$14
60 END
70aASM-BEGIN OF DIAL ASSEMBLEY
80:BASE          =#400
85:PORT'B=#B001
90:BASIC         =#C55B
100:CR           =#0D
110:STRINGBUFFER =#140
120:ddrb         =#B802
130:drb          =#B800
140:YSAV         =#3F
150:OSWRCH       =#FFF4
160:WACHT        =#FB33
170:PR'TEXT      =#F7D1
180.CODE#400
190.OPTION#50
200
210.HARD
220 NOP;NOP;NOP
230 LDA drb\
240 AND@#BF\ ZET HAAKKONTACT
250 STA drb\ AF
260 JMP BASIC
270:START
280 JSR#CEFA
290 STY#34
300 STY#43
310 LDA#53
320 CMP@1
330 BNE KOPY
340 LDA#52
350 CMP@#40
360 BEQ KIES
370:KOPY
380 DEY
390:LOOP1
400 INY
410 LDA(#52),Y
420 STA STRINGBUFFER,Y
430 CMP@CR
440 BNELOOP1
450:KIES
460 LDA ddrb
470 ORA@#70
480 STA ddrb
490 LDA drb
500 ORA@#40
510 STA drb
520 JSR WACHT'KIESTOON
530 LDY@#FF
540 STY YSAV
550:LOOP2
560 INC YSAV
570 LDY YSAV
580 LDA STRINGBUFFER,Y
590 CMP@#2D\VERG.STREEPJE
600 BEQ BEL'OP
610 CMP@CR
620 BEQ EXIT
630 CMP@#30\GEEN'GETAL
640 BCC error
650 CMP@#40\GEEN'GETAL
660 BCS error
670:BEL'OP
680 CLC
690 ADC@#60   \PRINT ACCU
700 JSR OSWRCH\GEINVENTEERD
710 CMP@#8D\VERG.INVERS STREEPJE
720 BNE GO'ON\ZO NIET GA DOOR
730 JSR WACHT'KIESTOON\WEL STREEPJE WACHT OP
740 JMP LOOP2\HAAL VOLGENDE GETAL     KIESTOON
750:GO'ON
760 SBC@#90\HAAL ER#90VAN AF
770 BNE GEEN'NUL\INDIEN NUL
780 CLC\ TEL ER TIEN BY OP
790 ADC@#A\WANT NUL=TIEN BY DE Ptt
800:GEEN'NUL
810 TAY\ZET GETAL IN Y-REG. ALS LOOP TELLER
820 LDA drb
830 ORA@#20\SLUIT KORTSLUITKONTACT
840 STA drb
850 JSR DELAY+6
860:LOOP3
870 ORA@#10\OPEN IMPULSCONTACT
880 STA drb
890 JSR DELAY
900 AND@#EF\SLUIT IMPULSCONTACT
910 STA drb
920 JSR DELAY+3
930 DEY
940 BNE LOOP3\CYFER NOG NIET KLAAR
950 AND@#DF\OPEN KORTSLUIT CONTACT
960 STA drb
970 JMP LOOP2\HAAL VOLGENDE CIJFER
980:EXIT
990 JSR#CD54
1000 RTS
1010:error
1020 JSR PR'TEXT
1030.BYTE#07,#07,#0D,#0A
1040.ASCII'ALLEEN GETALLEN'
1050.BYTE#0D,#0A
1060 NOP
1070 JMP EXIT
1080:DELAY
1090 LDX@#4
1100 BIT#04A2
1110 BIT#1EA2
1120 JMP WACHT
1130:WACHT'KIESTOON
1131 BIT PORT'B
1132 BVS TEST'KIESTOON
1133 PLA
1134 PLA
1135 JMP EXIT
1136:TEST'KIESTOON
1140 BIT drb
1150 BPL WACHT'KIESTOON
1160 RTS
1170.IF (?#28FF&1);.CODE BASE;JSR START
1180.END OF SEMBLY
1190 RETURN
```


A out
A IN
B out
B IN
RLA
RLB
RLC
C1
R1
TR1
C2
5V Vcc
PD7
PB6
PB4
PB5
Tpsk
IC1
IC2
D1
D2
D3
R2
R3
R4
R5
R6
C3
T1
T2
T3

# De VIA 6522 onder INTERRUPT.

## *Versatile Interface Adapter.*

## DEEL 3. INTERRUPT.

We gaan nu met behulp van het I/O-moduul, Input en Output plegen op basis van interrupt. Hiervoor heb ik eerst een uitwerking gemaakt, hoe de ACORN ATOM het eigenlijk precies doet.
Zie "De CPU 6502 pinaansluitingen." en "Mag ik even STOREN".

### *Drie vormen van I/O.*

De 3 vormen van I/O die ons ter beschikking staan zijn:

*I* Geprogrammeerde I/O.
Het transport van data (I/O) is het gevolg van de aanwezigheid van I/O-instrukties op vastgestelde plaats(en) in het programma. De initiatiefnemer is het programma.
We hebben deze methode gebruikt in DEEL 2 'INPUT'.

*II* I/O op basis van Interrupt.
Het transport van data (I/O) vindt plaats op initiatief van het aangesloten besturingsproces en niet op dat van het uitvoerende programma.
Hier gaan we, in dit DEEL 3, verder op in.

*III* I/O op basis van Direct Memory Acces (DMA).
Dit betekent dat het aangesloten besturingsproces via een DMA-controller zijn gegevens direkt in het geheugen plaatst zonder tussenkomst van de microprocessor.
De DMA-controller is een soort slaaf uP, om het geheugen te vullen terwijl de uP zelf de meester blijft en zijn lopend programma kan blijven volgen zonder onderbreking.

### *De HARDWARE-interrupt.*

Voor de HARD-WARE interrupt maken we alleen gebruik van de IRQ (Interrupt ReQuest) pin 4 van de uP. Het voordeel van de IRQ-interrupt is, dat we software-matig kunnen bepalen of de interrupt wel of niet door de uP moet worden toegestaan. Dit hangt af van de toestand waarin het Interrupt masker bit (I-vlag) in het CCR (Conditie Code Register) zich bevindt. Indien het I-bit is geset , SEI-instruktie, (I=1), wordt de IRQ-interrupt niet toegestaan. Wanneer het I-bit is gereset, CLI-instruktie, (I=0) wordt de interrupt toegestaan, nadat de uP de lopende instruktie heeft afgemaakt. De PC (Program-Counter) en CCR worden op de Stack gezet, het I-bit wordt gezet en vervolgens wordt op adres #FFFE & #FFFF het startadres van de IRQ-routine opgehaald en via het vektoradres #0204 (laag) & #0205 (hoog) naar de ISR (Interrupt Service Routine) gesprongen.

Wanneer de uP op het eind van de ISR-routine de RTI-instruktie (Return from Interrupt) tegenkomt, dan herstelt hij de intene uP registers CCR en PC, en keert terug naar het hoofdprogramma.
Het hoofdprogramma wordt vervolgt op het eerstvolgende instruktie adres. We kunnen dit vergelijken met de RST-instruktie aan het eind van een subroutine, al wordt daar alleen het teruqkeeradres van de stack gehaald.

*De SOFTWARE-interrupt.*

Een interrupt kan ook SOFT-WARE tot stand komen. Hiertoe dient de instruktie:

BRK BReaK (forced interrupt).

Komt de processor tijdens het doorlopen van een programma de operatiekode voor een BRK-instruktie tegen (#00), dan worden de adresbuffers geladen met het adres #FFFE en 1 klokcyclus later met #FFFF. In deze geheugenplaatsen staat het startadres van de IRQ-routine. Eerst worden echter door de uP die operaties verricht zoals die bij elke interrupt worden verricht. Dus PC en CCR op stack. Ook wordt bij deze instruktie de B-vlag gezet. Dit is een bit in het CCR dat slechts dan 1 wordt als de processor de BRK-instruktie heeft ontvangen.

De BRK-instruktie is niet te maskeren met de interruptvlag. DIe instruktie zal ook als I=1 een interrupt genereren. De afhandeling van de BRK-instruktie binnen de ACORN ATOM is nu;

```
FFB2 STA#FF          zet accu veilig
FFB4 PLA             haal CCR van stack
FFB5 PHA             zet terug op stack
FFB6 AND@#10         kijk of de BREAK-vlag is gezet?
FFB8 BNE#FFC0        gezet? Dan naar BRK-routine
FFBA LDA#FF          haal accu inhoud terug
FFBC PHA             zet op stack
FFBD JMP(#0204)      naar ISR
FFC0 LDA#FF          haal accu inhoud terug,
FFC2 PLP             haal CCR van stack
FFC3 PHP             zet terug op stack
FFC4 JMP(#0202)      naar BRK-routine.
```

De SOFTWARE-interrupt BRK, wordt wel toegepast bij het DEBUGGEN van een programma. In het algemeen bevat een programma fouten die dan opgespoort moeten worden. Door in een programma een op-kode te vervangen door de BRK-instruktie wordt het betreffend programma tot die BRK-instruktie normaal doorlopen. Daarna wordt naar een interrupt programma gesprongen dat ons in staat stelt bepaalde belangrijke registers te kontroleren en zodoende vast te stellen of het programmadeel aan de verwachtingen heeft voldaan.

*PROGRAMMA:*

De volgende programma's zijn hierin opgenomen:

```
=  ITbasis 8200 8536  basis struktuur voor int. op #3800
=  ITbasal 8200 8570  basis struktuur overal te plaatsen
=  ITca1   8200 8529  interrupt door CA1
=  ITcb1&2 8200 8533  interrupt door CB1 en CB2
=  ITtoos  8200 85BD  interrupt door TIMER 1 one shot mode
=  ITtofr  8200 85BD  interrupt door TIMER 1 free run mode
=  ITtzos  8200 85BE  interrupt door TIMER 2 one shot mode
=  ITtzpc  8200 8633  interrupt door TIMER 2 pulse counting
=  ITsicb1 8200 8583  interrupt SHIFT REG. output clock CB1
=  ITsicb1 8200 857F  interrupt SHIFT REG. input clock CB1
```

Gebruikte literatuur:


```
Acorn Nieuws 3-4'84 blz.39   Freddy Ingels
Atom Nieuwe  4-7'85 blz.41   G.Dols
Datasheet  SY6522
Microcomputers A tot Z       M.Immerzeel.
Atom-ware  1&2&3             Henk Reinders.
```


Samenstelling:


Andre Wessels  Harreveld  januari 1986.

# PROGRAMMA'S OP INTERRUPT BASIS.

*Een algemene struktuur waarmee men goed kan werken.*

```
 10 PROGRAM BASIS INTERRUPT PROGRAMMA'S OP #3800
 20
 30 REM ANDRE WESSELS
 40
 50 DIM II(4)
 60 V=#8800;REM basisadres VIA6522
 70 P=#2800;P.$21;GOSUBm
 80 P=#2800;P.$ 6;GOSUBm
 90 LINK II0
100 END
110m
120[
130\initialiseer de VIA6522
140:II0 SEI                    \zet uP interrupt uit
150     LDA @#55;STA#0209 \reset printerroutine
160     LDA @#00;STA#0204 \lsb ISR
170     LDA @#38;STA#0205 \msb ISR
180     LDA @#FF;STA V+2  \DDRB output
190     LDA @#00;STA V+3  \DDRA input
200     LDA @#ff;STA V+#E \clear alle interrupts
210     LDA @#oo;STA V+#E \sta interrupt toe
220\initus timers- & schuifregisters
230:III CLI;RTS                \sta uP interrupt toe
240];P=#3800;[                 \startadres ISR
250\interrupt service routine
260:II2 LDA V+1;STA V          \schakelaarsop LED's
270     PLA;RTI                \accu van stack en terug
280]
290 RETURN
```

*Door regel 10 te veranderen, regel 85 toe toevoegen en regel 160, 170 en 240 weg te halen, krijgt men:*

```
 10 PROGRAM BASAL INTERRUPT PROGRAMMA'S OVERAL
 85 ?#204=II2%256;?#205=II2/256:REM IRQ vektor zetten
160
170
240
```

*Door schakelaar CA1 te bedienen, kan men onder interrupt de schakelaars uitlezen op de LED's, terwijl je een programma list.*

```
 10 PROGRAM ICA1 INTERRUPT
 20 REM CA1,NEERGAANDE FLANK
 30 REM ANDRE WESSELS
 40
 50 DIM II(4)
 60 V=#B800;REM basisadres VIA6522
 70 P=#2800;P.$21;GOSUBm
 80 P=#2800;P.$ 6;GOSUBm
 90 LINK IIO
100 END
110m
120[
130\initialiseer de VIA6522
140:IIO SEI                    \zet uP interrupt uit
150      LDA #0#55;STA#0208    \reset printerroutine
160      LDA #0#00;STA#0204    \lsb ISR
170      LDA #0#38;STA#0205    \msb ISR
180      LDA #0#FF;STA V+2     \DDRB output
190      LDA #0#00;STA V+3     \DDRA input
200      LDA #0#7D;STA V+#E    \clear alle interrupts behalve CA1
210      LDA #0#82;STA V+#E    \sta interrupt CA1 toe
220:II1 CLI;RTS                \sta uP interrupt toe
230];P=#3800;[                 \startadres ISR
240\interrupt service routine
250:II2 LDA V+1;STA V          \schakelaars op LED's
260      PLA;RTI               \accu van stack en terug
270]
280 RETURN
```

*Idem, maar nu door CB1 en CB2.*

```
 10 PROGRAM ICB1&2 INTERRUPT
 20 REM CB1&CB2,NEERGAANDE FLANK
 30 REM ANDRE WESSELS
 40
 50 DIM II(4)
 60 V=#B800;REM basisadres VIA6522
 70 P=#2800;P.$21;GOSUBm
 80 P=#2800;P.$ 6;GOSUBm
 90 LINK IIO
100 END
110m
120[
130\initialiseer de VIA6522
140:IIO SEI                    \zet uP interrupt uit
150      LDA #0#55;STA#0208    \reset printerroutine
160      LDA #0#00;STA#0204    \lsb ISR
170      LDA #0#38;STA#0205    \msb ISR
180      LDA #0#FF;STA V+2     \DDRB output
190      LDA #0#00;STA V+3     \DDRA input
200      LDA #0#67;STA V+#E    \clear alle interrupts op CB1&2 na
210      LDA #0#98;STA V+#E    \sta interrupt CB1&CB2 toe
220:II1 CLI;RTS                \sta uP interrupt toe
230];P=#3800;[                 \startadres ISR
240\interrupt service routine
250:II2 LDA V+1;STA V          \schakelaarsop LED's
260      PLA;RTI               \accu van stack en terug
270]
280 RETURN
```

*TIMER 1 geeft na elke uitlezing, na 65535uS een interrupt.*

```
  10 PROGRAM TOOS INTERRUPT T1
  20 REM TIMER 1 ONE SHOT MODE
  30 REM ANDRE WESSELS
  40
  50 DIM II(4)
  60 V=#8800;REM basisadres VIA6522
  70 P=#2800;P.$21;GOSUBm
  80 P=#2800;P.$ 6;GOSUBm
  90 LINK II0
 100 END
 110m
 120[
 130\initialiseer de VIA6522
 140:II0 SEI                    \zet uP interrupt uit
 150     LDA @#55;STA#0208 \reset printerroutine
 160     LDA @#00;STA#0204 \lsb ISR
 170     LDA @#38;STA#0205 \msb ISR
 180     LDA @#FF;STA V+2  \DDRB output
 190     LDA @#00;STA V+3  \DDRA input
 200     LDA @#00;STA V+#B \acr
 210     LDA @#C0;STA V+#E \sta interrupt T1 toe
 220     LDA @#FF;STA V+#4 \t1c-L
 230     LDA @#FF;STA V+#7 \t1l-H
 240               STA V+#5 \t1c-H
 250:II1 CLI;RTS           \sta uP interrupt toe
 260];P=#3800;[            \startadres ISR
 270\interrupt service routine
 280     LDA @#FF;STA V+#5 \t1c-H reset
 290:II2 LDA V+#1;STA V    \schakelaars op LED's
 300     LDA @#C0;STA V+#E \ier irq reset
 310     PLA;RTI           \accu van stack en terug
 320]
 330 RETURN
```

*TIMER 1 geeft elke 65535uS een interrupt (vaste tijd).*

```
  10 PROGRAM TOFR INTERRUPT T1
  20 REM TIMER 1 FREE RUN MODE
  30 REM ANDRE WESSELS
  40
  50 DIM II(4)
  60 V=#8800;REM basisadres VIA6522
  70 P=#2800;P.$21;GOSUBm
  80 P=#2800;P.$ 6;GOSUBm
  90 LINK II0
 100 END
 110m
 120[
 130\initialiseer de VIA6522
 140:II0 SEI                    \zet uP interrupt uit
 150     LDA @#55;STA#0208 \reset printerroutine
 160     LDA @#00;STA#0204 \lsb ISR
 170     LDA @#38;STA#0205 \msb ISR
 180     LDA @#FF;STA V+2  \DDRB output
 190     LDA @#00;STA V+3  \DDRA input
 200     LDA @#40;STA V+#B \acr
 210     LDA @#C0;STA V+#E \sta interrupt T1 toe
 220     LDA @#FF;STA V+#4 \t1c-L
 230     LDA @#FF;STA V+#7 \t1l-H
 240               STA V+#5 \t1c-H
 250:II1 CLI;RTS           \sta uP interrupt toe
 260];P=#3800;[            \startadres ISR
 270\interrupt service routine
 280     LDA V+4           \t1c-L reset
 290:II2 LDA V+#1;STA V    \schakelaars op LED's
 310     PLA;RTI           \accu van stack en terug
 320]
 330 RETURN
```

*TIMER 2 geeft na elke uitlezing. na 65535u3 een interrupt.*

```
 10 PROGRAM TZOS INTERRUPT T2
 20 REM TIMER 2 ONE SHOT MODE
 30 REM ANDRE WESSELS
 40
 50 DIM II(4)
 60 V=#B800;REM basisadres VIA6522
 70 P=#2800;P.$21;GOSUBm
 80 P=#2800;P.$ 6;GOSUBm
 90 LINK IIO
100 END
110m
120[
130\initialiseer de VIA6522
140:IIO SEI                    \zet uP interrupt uit
150     LDA @#55;STA#0208 \reset printerroutine
160     LDA @#00;STA#0204 \lsb ISR
170     LDA @#38;STA#0205 \msb ISR
180     LDA @#FF;STA V+2  \DDRB output
190     LDA @#00;STA V+3  \DDRA input
200     LDA @#00;STA V+#B \acr
210     LDA @#A0;STA V+#E \laat interrupt T2 toe
220     LDA @#FF;STA V+#8 \t2c-L
230     LDA @#FF;STA V+#9 \t2c-H
240:III CLI;RTS                \sta uP interrupt toe
250];P=#3800;[                 \startadres ISR
260\interrupt service routine
270     LDA V+#8               \t2c-L reset
280:II2 LDA V+#1;STA V         \schakelaars op LED's
290     LDA @#A0;STA V+#E \ier irq reset
300     LDA @#FF;STA V+#9 \t2c-H
310     PLA;RTI                \accu van stack en terug
320]
330 RETURN
```

*TIMER 2 genereert 'n interrupt door 't aftellen van PB6 pulsen.*

```
 10 PROGRAM TZPC INTERRUPT T2
 20 REM TIMER 2 PULSE COUNTING
 30 REM COUNTING OP PB6
 40 REM IN I/O-MODUUL CB1 VERBONDEN MET PB6 IVM.KONTAKTDENDER
 50 REM ANDRE WESSELS
 60
 70 DIM II(4)
 80 V=#B800;REM basisadres VIA6522
 90 P=#2800;P.$21;GOSUBm
100 P=#2800;P.$ 6;GOSUBm
110 LINK IIO
120 END
130m
140[
150\initialiseer de VIA6522
160:IIO SEI                    \zet uP interrupt uit
170     LDA @#55;STA#0208 \reset printerroutine
180     LDA @#00;STA#0204 \lsb ISR
190     LDA @#38;STA#0205 \msb ISR
200     LDA @#BF;STA V+2  \DDRB output PB6 input
210     LDA @#00;STA V+3  \DDRA input
220     LDA @#20;STA V+#B \acr
230     LDA @#A0;STA V+#E \laat interrupt T2 toe
240     LDA @#05;STA V+#8 \t2c-L 5*PULS op PB6
250     LDA @#00;STA V+#9 \t2c-H
260:III CLI;RTS                \sta uP interrupt toe
270];P=#3800;[                 \startadres ISR
280\interrupt service routine
290     LDA V+#8               \t2c-L reset
300:II2 LDA V+#1;STA V         \schakelaars op LED's
310     LDA @#A0;STA V+#E \ier irq reset
320     LDA @#00;STA V+#9 \t2c-H opnieuw
330     PLA;RTI                \accu van stack en terug
340]
350 RETURN
```

*Nu het Shift-Register laden, dan met CB1 via CB2 uitschuiven.*

```
 10 PROGRAM SOCB1 INTERRUPT SR
 20 REM DOOR EXTERNE KLOKPULS CB1,
 30 REM UITSTUREN OP CB2.
 40 REM ANDRE WESSELS
 50
 60 DIM II(4)
 70 V=#B800;REM basisadres VIA6522
 80 P=#2800;P.$21;GOSUBm
 90 P=#2800;P.$ 6;GOSUBm
100 LINK II0
110 END
120m
130[
140\initialiseer de VIA6522
150:II0 SEI                 \zet uP interrupt uit
160     LDA @#55;STA#0208   \reset printerroutine
170     LDA @#00;STA#0204   \lsb ISR
180     LDA @#38;STA#0205   \msb ISR
190     LDA @#FF;STA V+2    \DDRB output
200     LDA @#00;STA V+3    \DDRA input
210     LDA @#1C;STA V+#B   \acr
220     LDA @#84;STA V+#E   \laat interrupt SR toe
230     LDA @#AA;STA V+#A   \schuifreg. 1010 1010
240:II1 CLI;RTS             \sta uP interrupt toe
250];P=#3800;[              \startadres ISR
260\interrupt service routine
265     JSR#FD1A            \pieptoon
270:II2 LDA V+#1;STA V+#A   \schaklrs in sr zie CB2
280     PLA;RTI             \accu van stack en terug
290]
300 RETURN
```

*Nu met CB1 via CB2 inschuiven in het Shift-Register.*

```
 10 PROGRAM SICB1 INTERRUPT SR
 20 REM DOOR EXTERNE KLOKPULS CB1,
 30 REM INSTUREN OP CB2.
 40 REM ANDRE WESSELS
 50
 60 DIM II(4)
 70 V=#B800;REM basisadres VIA6522
 80 P=#2800;P.$21;GOSUBm
 90 P=#2800;P.$ 6;GOSUBm
100 LINK II0
110 END
120m
130[
140\initialiseer de VIA6522
150:II0 SEI                 \zet uP interrupt uit
160     LDA @#55;STA#0208   \reset printerroutine
170     LDA @#00;STA#0204   \lsb ISR
180     LDA @#38;STA#0205   \msb ISR
190     LDA @#FF;STA V+2    \DDRB output
200     LDA @#00;STA V+3    \DDRA input
210     LDA @#0C;STA V+#B   \acr
220     LDA @#84;STA V+#E   \laat interrupt SR toe
230     LDA @#FF;STA V+#A   \schuifreg. 1111 1111
240:II1 CLI;RTS             \sta uP interrupt toe
250];P=#3800;[              \startadres ISR
260\interrupt service routine
270     JSR#FD1A            \pieptoon
280:II2 LDA V+#A;STA V      \schuifreg. op LED's
290     PLA;RTI             \accu van stack en terug
300]
310 RETURN
```

Veel sukses met deze interrupt-programma's.

Andre.