

IN/OUT

in- en uitvoer via joystickpoort 2

Peter Zevenhoven
CUC Info

Scanned, ocr'ed and converted to PDF by HansO, 2001

Een van de vele mogelijkheden die een computer biedt, is het besturen van allerlei er aangeknoopte electronica en/of mechanica. Hiervoor is altijd minimaal een zo geheten I/O (Input/Output) poort nodig, die als schakel tussen de microproces sor en de buitenwereld fungeert.

De SV.328 heeft er een aantal, terwijl twee daarvan via de joystick poorten naar buiten uitgevoerd, en derhalve geheel vrij te gebruiken zijn. Per joystickpoort hebben we de beschikking over 7 ingangen, of 3 ingangen en 4 uitgangen. Aangezien er toepassingen te bedenken zijn die meer in /uitgangen verlangen, heb ik een programma geschreven dat deze poorten op een andere manier gebruikt. Als ingang, of uitgang, gebruiken we de joystickpoort op deze wijze niet direct, maar de in/uitgangen van enkele extra aan te sluiten speciale onderdelen, te weten: schuifregisters.

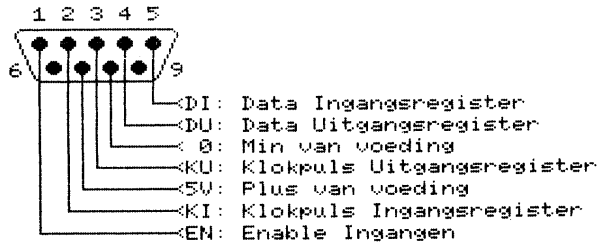
De gebruikte types hebben per IC 8 uitgangen of 8 ingangen en 3 besturingsaansluitingen die met de joystickpoort verbonden moeten worden." Het voordeel hiervan is, dat wanneer er bijvoorbeeld 40 uitgangen nodig zijn, er vijf uitgangsschuifregisters achter elkaar geknoopt kunnen worden, terwijl er nog steeds maar 3 verbindingen met de joystickpoort zijn.

Het programma stuurt de gegevens in serie naar de schuifregisters en aan de in/uitgangen van die registers staan die signalen parallel tot onze beschikking. De kracht van dit programma is, dat het vanzelf een aantal bits uit het geheugen naar de schuifregisters of bits van de schuifregisters naar het geheugen brengt. Dit gebeurt automatisch, 50 keer per seconde, zonder dat u daarvoor de opdracht geeft (via de VDP interrupt).

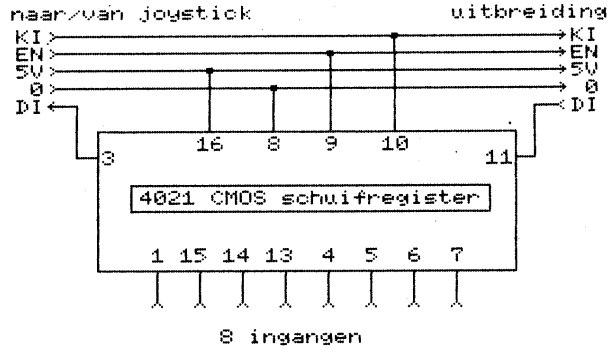
Nadat u de initialisatieprocedure gevolgd hebt (lees de REM regels van het input/output programma), kunt u in BASIC uw besturings procedures schrijven en wanneer u in het programma bijv. L=1 stelt, gaat er een lampje branden (zonder zelf OUT instructies te moeten geven). Of u drukt op een knop en op dat moment wordt de variabele S gelijk aan 1.

Aan de in- en uitgangen van de schuifregisters moet u zelf nog de nodige electronica hangen.

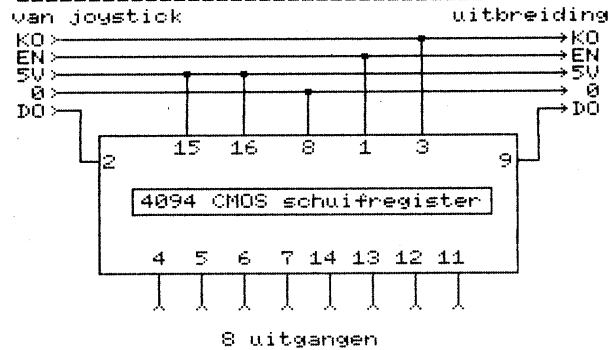
FIGUUR 1: joystick aansluiting



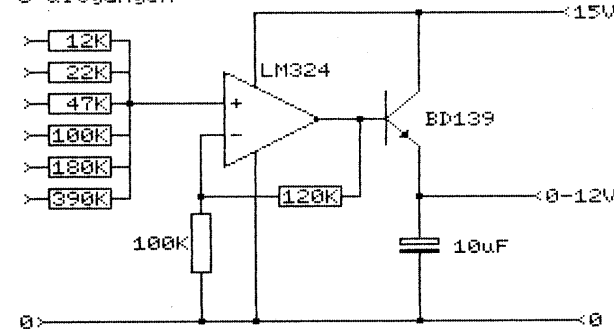
FIGUUR 2: ingangs schuifregister



FIGUUR 3: uitgangs schuifregister



FIGUUR 4: voorbeeld (motor) sturing
6 uitgangen



```

100 REM Bits van variabelen in of uit via joystick poort 2 -- (P.Zevenhoven)
105 '-----
451 110 IF PEEK(&HFEB8)=&HC9 THEN CLEAR 200, &HF330: ' Als er geen DISK BASIC is
115 :
142 120 DEFINT A-Z: ' Alleen integers kunnen worden in- of uitgeschoven
125 :
197 130 ON STOP GOSUB 50000: STOP ON: ' Eventueel alles uit na CTRL/STOP
375 135 ON ERROR GOTO 50000 : ' Idem als er een fout ontstaat
140 :
589 145 PA=&HF330: GOSUB 540: GOTO 1000: ' Plaats 1e gedeelte (INVA - USINT)
150 '-----
155 ' De subroutine INVA haalt bits (aantal in register B) uit het ingangs
160 ' schuifregister en schuift deze in het HL registerpaar. Het laatst
165 ' ingelezen bit komt in bit 0 van HL.
170 :
093 175 DATA 21,00,00: 'INVA ld hl,0 Wis de data
603 180 DATA 3E,CF : 'INBIT ld a,11001111b STROBE en CLOCK ingangen van
435 185 DATA D3,8C : ' out (8Ch),a schuifregisters laag
436 190 DATA DB,98 : ' in a,(98h) Lees het huidige bit
422 195 DATA 1F : ' rra
390 200 DATA 1F : ' rra Schuif het bit (2) in de Carry
405 205 DATA 1F : ' rra
522 210 DATA ED,6A : ' adc hl,hl Schuif bit in HL
627 215 DATA 3E,EF : ' ld a,11101111b Klokpuls (positieve flank) naar
409 220 DATA D3,8C : ' out (8Ch),a schuifregister
424 225 DATA 10,EF : ' djnz INBIT Volgende bit (als B>0)
423 230 DATA C9 : ' ret Einde INVA : waarde in HL
235 :
240 ' De subroutine OUTVA schuift bits (aantal in register B) uit registerpaar
245 ' HL en brengt deze naar het uitgangs schuifregister. Bit 0 van HL wordt er
250 ' het eerst uitgeschoven.
255 :
489 260 DATA CB,3C : 'OUTVA srl h Schuif HL een bit naar rechts
494 265 DATA CB,1D : ' rr l
448 270 DATA 3E,5F : ' ld a,01011111b STROBE uitgang en klokpuls laag
419 275 DATA 1F : ' rra Schuif bit in bit 7 van de ACCU
421 280 DATA D3,8C : ' out (8Ch),a Schrijf bit
207 285 DATA F6,40 : ' or 40h Klokpuls (positieve flank) naar
423 290 DATA D3,8C : ' out (8Ch),a schuifregister
197 295 DATA 10,F1 : ' djnz OUTVA Volgende bit (als B>0)
418 300 DATA C9 : ' ret Einde OUTVA
305 :
310 ' Na een interrupt of een USB aanroep, wordt naar USINT gesprongen. Deze
315 ' routine definieert de joystickpoort als uitgang, waarna een aantal keer
320 ' (totaal maximaal 50 keer) INVA of OUTVA aangeroepen wordt, afhankelijk
325 ' van de toepassing.
330 :
409 335 DATA F3 : 'USINT di Interrupt uit
208 340 DATA 3E,07 : ' ld a,7 Selecteer register 7 van de
297 345 DATA D3,8B : ' out (8Bh),a geluidsgenerator
334 350 DATA DB,90 : ' in a,(90h) Lees de huidige registerinhoud
202 355 DATA F6,40 : ' or 40h Maak bit 6 een 1 (poort A; uitgang)
418 360 DATA D3,8C : ' out (8Ch),a Schrijf nieuwe registerinhoud
395 365 DATA 3E,0E : ' ld a,14 Selecteer poort A (joystickpoort)
288 370 DATA D3,8B : ' out (8Bh),a van de geluidsgenerator
212 375 DATA *
380 '-----
385 ' CALLI wordt zoveel maal na USINT geplaatst als er variabelen in te lezen
390 ' zijn. De operands van de LD B en LD HL instructies worden aan betreffende
395 ' variabele aangepast.
400 :
318 405 DATA 06,?? : 'CALLI ld b,?? Laad B met het aantal bits
762 410 DATA CD,30,F3: ' call INVA Lees dat aantal uit het schuifreg.
873 415 DATA 22,??,??: ' ld (????),hl Geef nieuwe waarde aan de variabele
188 420 DATA *
425 :
430 ' CALLO wordt zoveel maal na USINT geplaatst als er variabelen uit te
435 ' schuiven zijn. Ook hier worden de operands van de LD B en LD HL
440 ' instructies aan betreffende variabele aangepast.
445 :
313 450 DATA 06,?? : 'CALLO ld b,?? Laad B met het aantal bits
016 455 DATA 2A,??,??: ' ld hl,(????) Haal huidige waarde van variabele
843 460 DATA CD,45,F3: ' call OUTVA Breng B bits naar schuifregister
211 465 DATA *
470 '-----
475 ' RETI besluit de rij van routines, de STROBE ingang der schuifregisters -->
480 ' wordt hoog en de BASIC interpreter mag weer doorgaan.

```

```

485 :
641 490 DATA 3E,FF : 'RETI ld a,0FFh      Maak alle uitgangen hoog (houd de
440 495 DATA D3,BC : '      out (8Ch),a  nieuwe data in de registers vast)
420 500 DATA C9 : '      ret          Terug naar BASIC (of interrupt).
200 505 DATA *
510 '-----
515 REM Nu volgen enkele subroutines om het gewenste machinetaalprogramma mee
520 ' samen te stellen
525 '-----
530 REM POKE data op adres (PA) en hoger tot een * in de data zit
535 :
249 540 READ A$: IF A*="*" THEN RETURN ELSE POKE PA,VAL("&H"+A$): PA=PA+1: GOTO 540
545 '-----
550 REM Voeg INVA toe zodat de variabele, waarvan het VARPTR adres in A en het
555 ' aantal benodigde bits in B staat, ingelezen kan worden.
560 :
619 565 RESTORE 405: C=2
570 :
355 575 GOSUB 540: POKE PA-7, B: ' Pas LD B aan
917 580 POKE PA-C, 255 AND A: POKE PA-C+1, (A + 65536!) / 256: ' Pas LD HL aan
454 585 RETURN
590 '-----
595 REM Voeg OUTVA toe zodat de variabele, waarvan het VARPTR adres in A en het
600 ' aantal benodigde bits in B staat, uitgeschoven kan worden.
605 :
980 610 RESTORE 450: C=5: GOTO 575
615 '-----
620 REM Voeg RETI toe en definieer machinetaal-startadres.
625 ' Met Z=USR(0) kan het datatransport gestart worden.
630 :
806 635 RESTORE 490: DEFUSR=&HF355: GOTO 540
640 '-----
645 REM Voeg RETI toe en laat het datatransport automatisch (50 keer per sec-
650 ' onde) plaatsvinden (gebruiker hoeft niet expliciet Z=USR(0) te doen).
655 :
823 660 GOSUB 635: ' Plaats RETI (+defusr)
665 :
670 ' Voer de 'gecompileerde' routine bij iedere VDP interrupt uit; zet de
675 ' interrupt HOOK op 0F355h.
680 :
956 685 POKE &HFE7B, &HF3: POKE &HFE7A, &H55: POKE &HFE79, &HC3
442 690 RETURN
695 '-----
1000 REM Nu dient de variabele-specificatie plaats te vinden. Voor iedere
1010 ' variabele dienen 4 instructies gegeven te worden:
1020 '
1030 ' - de variabele dient een waarde te krijgen (bijv. X=0);
1040 ' - de VARPTR (het adres van de variabele) moet aan A doorgegeven
1050 ' worden (bijv. A=VARPTR(X));
1060 ' - het aantal bits dat van die variabele in of uitgeschoven moet
1070 ' worden, hoort in B te staan (bijv. B=4);
1080 ' - afhankelijk van de transportrichting dient GOSUB 565 (om bits in
1090 ' te schuiven) of GOSUB 610 (om uit te schuiven) gegeven te worden.
1100 '
1110 ' Nadat voor iedere variabele een CALLI of CALLO toegevoegd is, moet
1120 ' de reeks worden besloten met GOSUB 630 (als het bit transport met
1130 ' Z=USR(0) gestart wordt) of met GOSUB 655 (wanneer het transport 50
1140 ' keer per seconde automatisch moet plaatsvinden).
1150 '
1160 ' Als voorbeeld: - variabelen S1 en S2 zijn in te lezen schakelaars;
1170 ' - M1 en M2 zijn motoren die gestuurd moeten worden.
1180 :
349 1190 S1=0: A=VARPTR(S1): B=1: GOSUB 565: ' Schakelaar 1 (invoer)
350 1200 S2=0: A=VARPTR(S2): B=1: GOSUB 565: ' Schakelaar 2
005 1210 M1=0: A=VARPTR(M1): B=6: GOSUB 610: ' Motor 1 (uitvoer)
034 1220 M2=0: A=VARPTR(M2): B=6: GOSUB 610: ' Motor 2
1230 :
587 1240 GOSUB 660: ' Schuifregisters worden vanaf nu om de 20ms gelezen/geschreven
1250 '-----
1260 ' Een klein voorbeeld van een motorbesturing: wanneer S1 ingedrukt is,
1270 ' moet M1 sneller draaien, na het indrukken van S2 langzamer.
1280 ' M1 is in regel 1210 bepaald als een 6 bits getal, dus de waarde van M1
1290 ' mag niet hoger worden dan 2^6-1 (63 dus).
1300 :
202 1310 IF S1 AND (M1<63) THEN M1=M1+1: 'Sneller
527 1320 IF S2 AND (M1>0) THEN M1=M1-1: 'Langzamer

```

```
1330 :
614 1340 GOTO 1310
1350 '-----
50000 ' Als in het programma een fout ontstaat of op CTRL/STOP gedrukt wordt,
50010 ' is het verstandig om, voordat het programma verlaten wordt, nog even
50020 ' alle OUTPUT variabelen op nul (o.i.d.) te zetten. Hierdoor wordt voor-
50030 ' komen dat allerlei motoren (treinen) op hol slaan.
50040 :
229 50050 M1=0: M2=0: Z=USR(0): ' Alle motoren uit
756 50060 POKE &HFE79, &HC9: ' Interrupt vector uit
50070 :
095 50080 ON ERROR GOTO 0: ' Eventuele foutmelding laten geven
265 50090 END
```