Techno talk
Joysticks en pads

Sandy Brand en Bas Vijfwinkel
MSX Computer & Club Magazine nummer 86 - november / december 1996

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

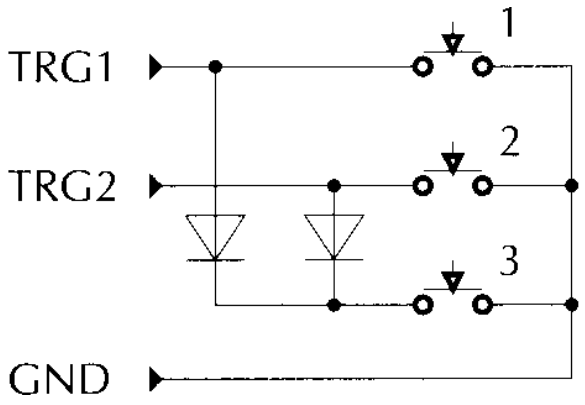
Na veel lol en wilde brainwaves is het aantal ideeën met betrekking tot de joystick zo toegenomen, dat we er maar eens een heel artikel aan wijden. Wij claimen geen originaliteit op de geplaatste schema's; hoewel we daar toch heel wat fouten uit hebben gehaald. Alle schema's zijn door ons gebouwd en wij hopen dat alle foutjes er nu uit zijn. Zo in de weer met de Joysticks benadrukken we nog eens de universaliteit van ons geliefde MSX.

Joystickpoort

De Joystickpoort zit als volgt in elkaar:

pen 1,2,3,4	richting in
pen 5	+5V
pen 6,7	vuurknop in
pen 8	puls uit
pen 9	aarde/GND

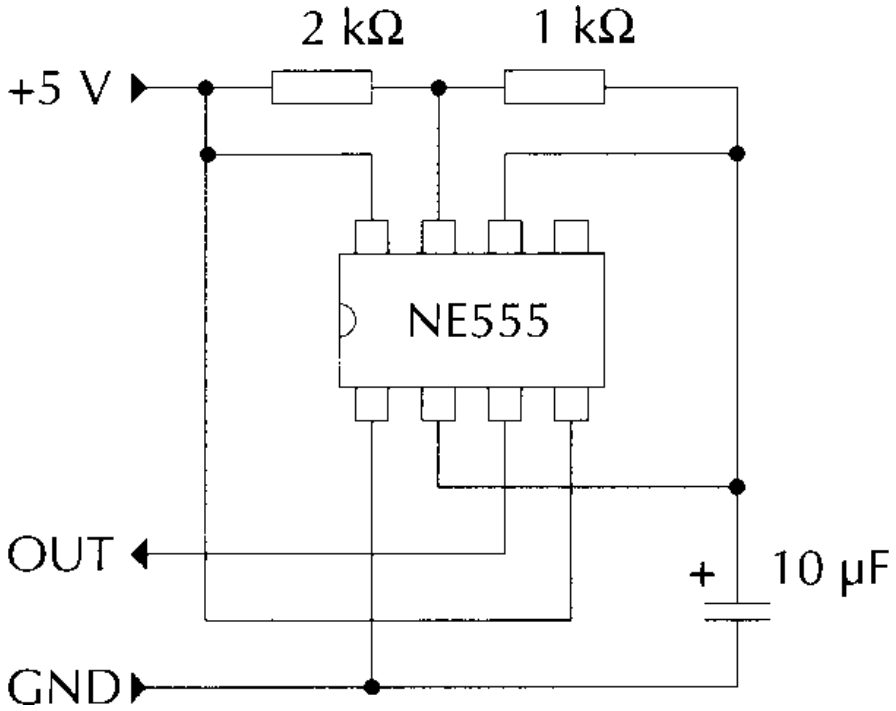
Het indrukken van een vuurknop of richtingsknop is eigenlijk niets meer dan dat pennetje verbinden met de aarde. Verder is er nog een tweede manier van registeren, maar daar komen we later op terug bij de PC joystick



Vuurknop

De meeste Joysticks en trackballs hebben twee vuurknoppen. Maar DotDesigner, bijvoorbeeld, gebruikt een derde vuurknop, die ondermeer op de Sony trackball zit. Hiermee is de kleur waarop de cursor staat tot werkkleur te kiezen. Na eens de

trackball te hebben opengemaakt, blijkt het een heel eenvoudige truc te zijn. De derde vuurknop is gewoon gelijk aan het indrukken van vuurknop l en 2 tegelijk. Dit is ook zelf met een schakelaar en twee diodes te vervaardigen. De ene kant van de drukschakelaar moet aan de aarde komen en de andere kant wordt via een diode aan pen 6 en via een diode aan pen 7 van de joystick ingang gezet. De diodes moeten dan geleiden naar de drukknop toe. Voor de duidelijkheid hebben we het maar in een schema gezet.



PCJOY.BAS

```
10 ' voorbeeld pc joystick
20 SCREEN 2: COLOR 15,4,7
30 C$=CHR$(26): D$=CHR$(255)
40 SPRITE$(0)=C$+C$+C$+D$+D$+C$+C$+C$
50 PUT SPRITE 0,(100,100),9
60 X=PDL(1): Y=PDL(3)
70 PUT SPRITE 0,(X,Y),9
80 IF STRIG(1)=-1 THEN PSET (X,Y),12
90 IF STRIG(3)=-1 THEN PSET (X,Y),5
100 GOTO 60
```

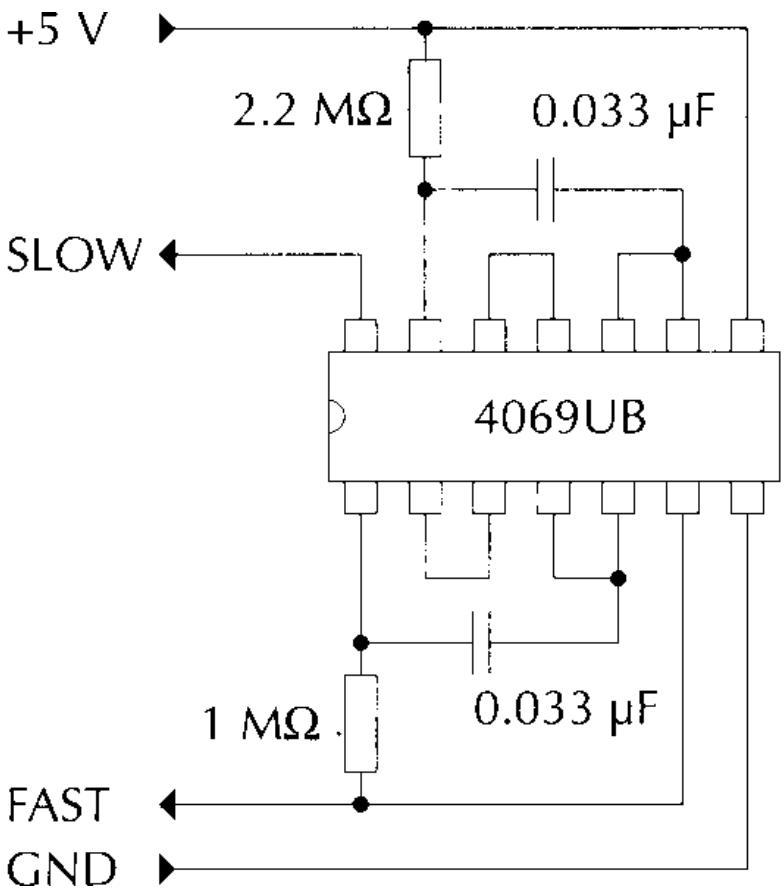
PCJOY2.BAS

```
10 ' voorbeeld 2 van gebruik pc joystick
20 CLS: KEY OFF
30 GOSUB 80 :' initialisatie
40 ' hoofdlus
50 GOSUB 120 :'in plaats van s=stick(1)
60 LOCATE 0,0: PRINT "richting joystick = ";S
70 GOTO 40
80 'initialisatie routine
90 PRINT "laat de joystick los aub en druk op de spatiebalk"
100 SX=PDL(1): SY=PDL(3)
```

```
110 A$=INKEY$: IF A$=" " THEN CLS: RETURN ELSE GOTO 110
120 'joystick hoofdroutine
130 's= richtingsvariable
140 S=0: X=PDL(1): Y=PDL(3)
150 X=PDL(1): Y=PDL(3)
160 IF X>SX+10 THEN S=3 ELSE IF X<SX-10 THEN S=7
170 IF Y>SY+10 THEN S=5 ELSE IF Y<SY-10 THEN S=1
180 RETURN
```

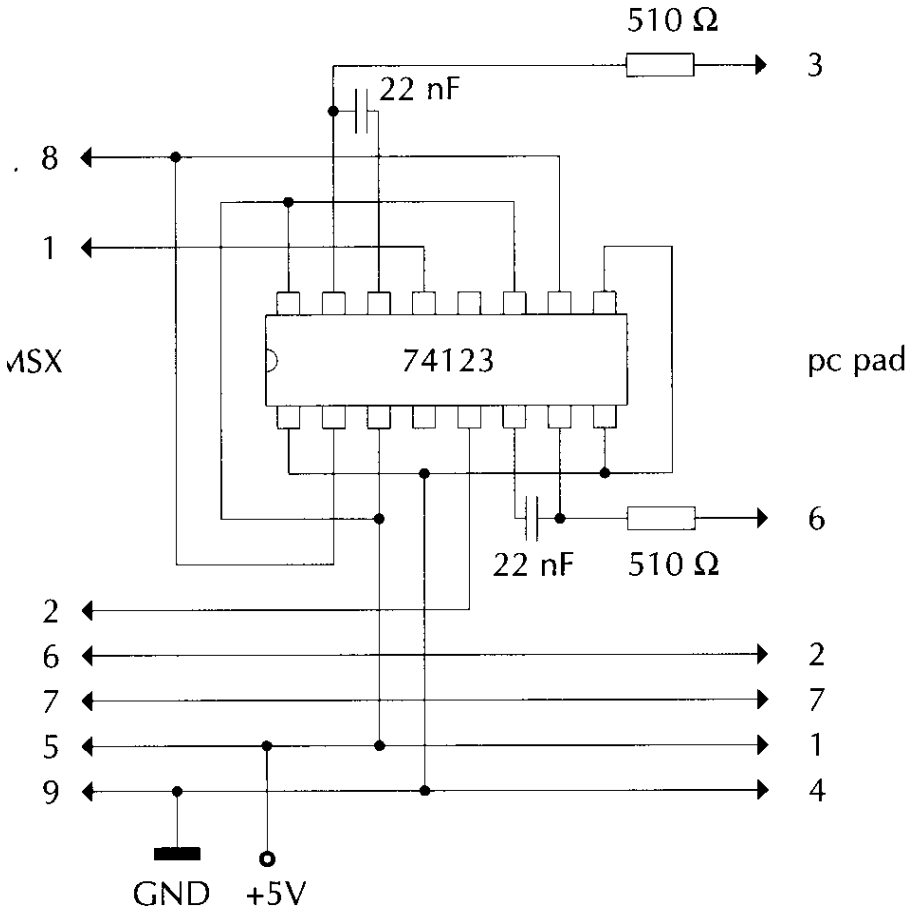
Autofire

Een geliefde optie voor spellen-freaks. Er bestaan vele varianten van, maar naast de versie die de NE555 timer gebruikt, wilden we ook nog een ander model laten zien. De NE555 moet bij iedere elec-tronica winkel te krijgen zijn, maar van de 4096UB zijn we niet helemaal zeker. Je moet echt de UB versie hebben, want bijvoorbeeld de B versie voldoet niet aan de specificaties. Het voordeel van de schakeling met de 4096UB is dat er twee losse autofi-res mee te maken zijn. Dat scheelt dus ook weer — wat zijn we milieu-bewust — een IC. Door de weerstand te vergroten gaat ie sneller vuren en maken we de weerstand kleiner, dan gaat hij langzamer vuren. In het schema wordt een 2.2 Mohm en een l Mohm weerstand gebruikt voor respectievelijk een snelle en een langzame autofire. Je zou ook kunnen overwegen om een variabele weerstand erin te zetten om zo een variabele autofire te maken. Hier kun je, denken we, wel zelf wat mee experimenteren.

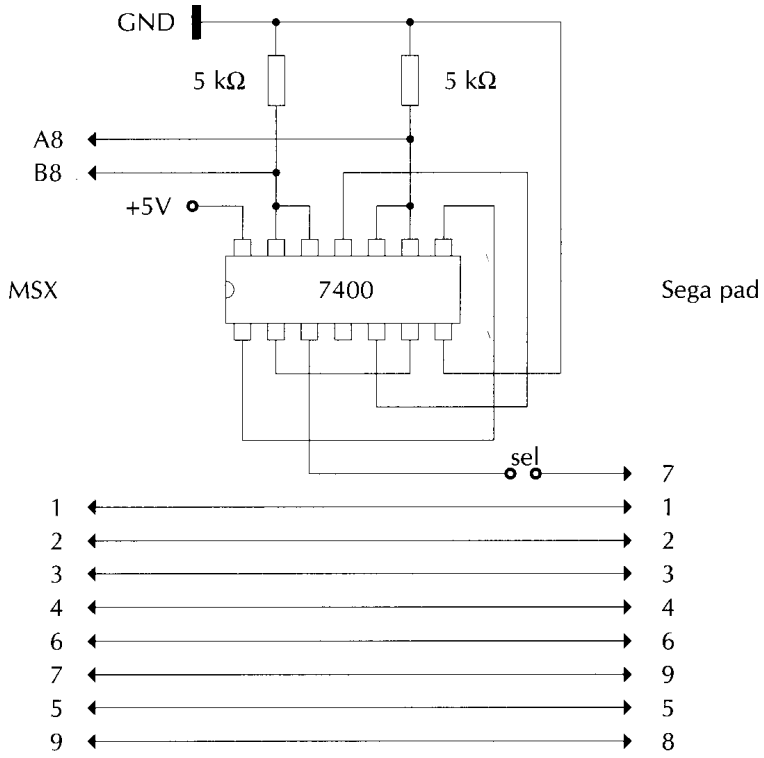


PC joystick

De PC joystick legt het bij mij altijd af tegen mijn vertrouwde arcade joystick, die zelfs een atoomoorlog zou moeten kunnen overleven. De zwabberende pc joystick is met behulp van één IC simpel op de MSX aan te sluiten. Je zou zelfs kunnen overwegen om zelf je joystick in elkaar te zetten, want meer dan twee variabele weerstanden zitten er niet in zo'n pc joystick. De loper van de variabele weerstand (150 k) moet dan aan pen 3 of 6 en een van beide zijden moet dan aan de aarde liggen. Het IC 74123 vormt de stand van de variabele weerstand om naar een puls die hiermee correspondeert. De MSX geeft op pen 8 een pulsje; daarna volgt deze puls die een bepaalde lengte heeft. De MSX gaat deze lengte meten en een hiermee corresponderend getal kan met de PDL instructie worden opgevraagd.



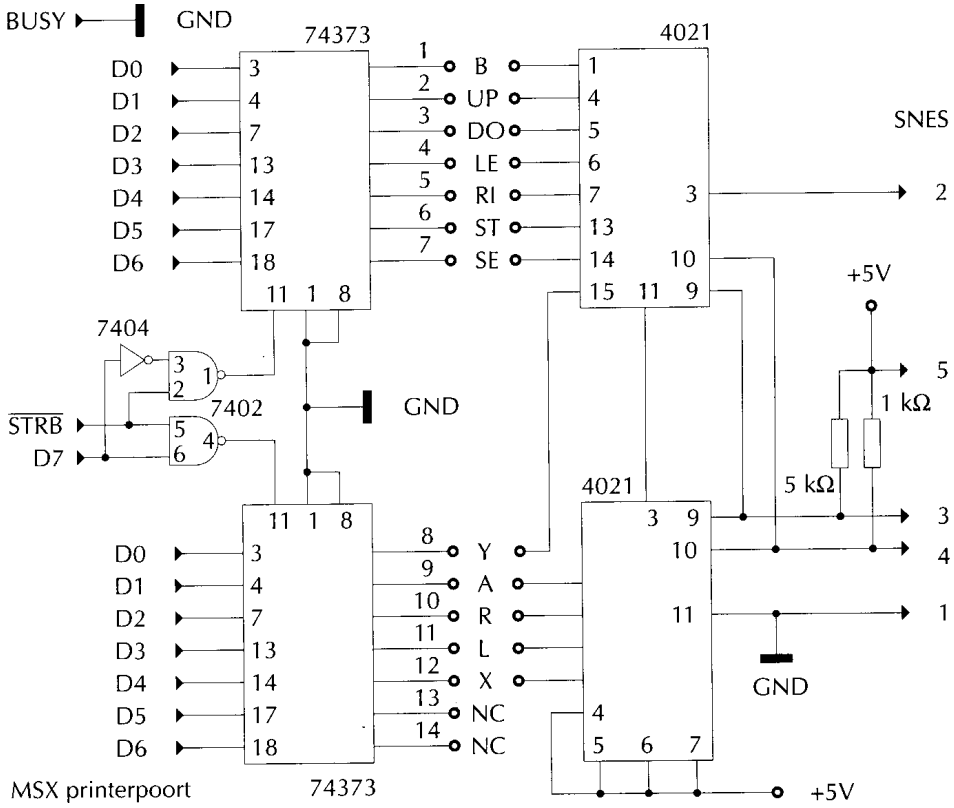
De potentiometer voor de x-beweging staat dus in het midden — bij veel Joysticks is dit ook bij te regelen met een draaiknopje — en als we nu bijvoorbeeld de pook naar rechts bewegen wordt de weerstand groter, wat als gevolg heeft dat de puls die het IC 74123 afgeeft relatief langer wordt. En dus is de waarde die terugkomt met PDL(l) ook relatief groter. We kunnen die waarde als directe x-waarde beschouwen en in ons programma gebruiken. Een eerste voorbeeld staat in PCJOY.BAS.
Maar we kunnen ook kijken of de waarde groter of kleiner wordt. Met het tweede voorbeeld willen we graag laten zien hoe je een PC joystick in je bestaande programma kunt gebruiken ter vervanging van een normale joystick. Dit tweede voorbeeld staat in PCJOY2.BAS.
Eerst moet je dus de neutrale stand bepalen en daarna ga je kijken of die verandert. In plaats van STICK te gebruiken en de waarde via de variabele S terug te krijgen, leest ons programma de PC joystick en zet die waarde om en zet hem in S. We hebben niet de tussenrichtingen — S=2, 4, 6 en 8 — erbij gedaan, maar we nemen aan dat je daar zelf wel iets voor kunt maken. Als we naar het schema kijken, dan zien we dat we per joystick-ingang slechts twee van de vier richtingspennen gebruiken dus we zouden met onze twee joystickingangen vrij eenvoudig vier joystick kunnen aansluiten. Als we dan het aantal vuurknoppen per PC joystick beperken tot één, dan kan ook iedereen nog een kogeltje afschieten. Afgezien van de kosten van de connectors en joystick, kost de rest nog geen ƒ 2,50 en deze zwabberpoken zijn voor een habbekrats op een pc beurs mee te nemen. Ook vrienden en kennissen willen nog wel eens dit soort joystickjes ergens hebben liggen.



Megadrive pad

De Megadrive pad is vrij eenvoudig op de MSX aan te sluiten. We kunnen alle knoppen uitlezen met een simpele schakeling. Als pen 7 hoog is, dan kunnen we de x- en y-rich-ting en de B- en C-button uitlezen. Is pen 7 laag, dan kunnen we de y-richting en de A en start button uitlezen. Het probleem is dat er op de joystickpoorten alleen twee pennetjes zitten die een pulsje geven, maar niet een signaal hoog of laag kunnen houden. Hiervoor hebben we het IC 7400 nodig. Met twee poortjes bouwen we een flipflop en met de twee overgebleven poortjes twee invertors. Als we dan nog twee pulldown weerstanden eraan hangen, dan hebben we een schakeling die van twee pulsjes een signaal maakt dat hoog of laag blijft staan. Met OUT &HA0,15:OUT &HA1,32 wordt pen 7 laag en kunnen we de start en A-button lezen met STRIG. Met OUT &HA0,15: OUT &HA1,16 wordt pen 7 hoog en kunnen we de rest uitlezen met STRIG en STICK. We kunnen hiervoor een klein programmaatje maken in basic.
In MEGADRV.BAS geven we een voorbeeld voor het uitlezen van de Megadrivepad. We moeten helaas wel constateren dat het niet 100% werkt. Dit euvel zal zich niet voordoen als we machinetaal zouden gebruiken. De basic inter-preter wil ook af en toe iets naar de PSG, waar ook de joysticks aan hangen, schrijven en daarmee verstoort hij eigenlijk de goede werking van ons programma. Nu kan het soms gebeuren dat wij de bits omhoog zetten en de interpreter deze weer omlaag gooit. In machinetaal heb je met DI het hele rijk alleen en zou het wel goed moeten werken. Er is ook een Megadrive pad met zes buttons, maar we hebben geen info kunnen vinden hoe die elkaar zit. Zonder elektronica is de pad ook te gebruiken, dan zou het je alleen twee connectors kosten. Dan moet pen 7 aan de 5 volt worden gelegd. De richting is zo uit te lezen, maar is niet meer te bepalen of de A/C button of Start/B button wordt ingedrukt. Eigenlijk een dubbel paar vuurknoppen.

	b7	b6	b5	b4	b3	b2	bl	b0	
data byte 1	1	select	start	right	left	up	down	B	
data byte 2	2	0	0	X	L	R	A	Y	



MEGADRV.BAS

```
10 CLS ' voorbeeld megadrive pad
20 OUT &HA0,15
30 OUT &HA1,&B00100000
40 S1=STRIG(1): S2=STRIG(3)
50 OUT &HA0,15
60 OUT &HA1,&B00010000
70 A=STICK(1)
80 S4=STRIG(1): S5=STRIG(3)
100 LOCATE 0,0
110 PRINT "waarde stick(0) :";A
120 PRINT "button a        :";S1
130 PRINT "button b        :";S4
140 PRINT "button c        :";S5
150 PRINT "start           :";S2
160 GOTO 20
```

Joystick-emulator

Naast het gebruiken van de joystick van een andere computer op de MSX, is het ook mogelijk om onze MSX als joystick op een andere computer te gebruiken. Op deze manier is de MSX als programmeerbare joystick te gebruiken. Dit heeft natuurlijk vooral nut bij spellen waarbij je specials kunt krijgen door een heel scala van vingeracrobatiek met alle gevolgen en frustraties vandien uit te voeren. Zouden we echter onze MSX deze code reeks kunnen laten uitvoeren wanneer wij bijvoorbeeld een bepaalde knop op onze computer indrukken, dan zou dat ons zware leven een stuk gemakkelijker maken. Dit schema is een joystick-emulator voor de Super Nintendo. Het originele schema was echter voor een Sega Saturn, maar aangezien die niet zo verspreid is als de Super Nintendo, hebben we het schema maar omgeknutseld voor de Super Nintendo. Maar terug naar ons schema.
Het geheel komt te hangen aan onze printerpoort en de voeding halen we uit de SuperNES joystick. Met behulp van het strobe-signaal en D7 wordt een soort selectieschakeling gemaakt, waarmee wordt bepaald in welk 7-bits geheugenblokje (74LS373) de data komt. Wat we dus nu eigenlijk hebben, is een 14-bits out-poort. Op de SNES gebruiken we 'maar' twaalf bits — of eigenlijk buttons — dus twee bits worden niet gebruikt. Je zou aan deze punten ook iets heel anders kunnen aansturen. Als je er veertien led's met bijbehorende weerstanden aanhangt, heb je een megabrede knight-riderbalk. Ook zou je met een transistor en een relais wat grovere spanningen kunnen aan- of uitzetten om bijvoorbeeld je koelkastdeur van 220 volt te voorzien als een bataljon kakkerlakken de aanval inzet op het resterende eten voor de maand. Ach, wat er allemaal niet mogelijk is met wat simpele elektronica. Het meeste komt toch aan op de software. Kan iemand zich nog de oproep herinneren van een of andere MSX'er die destijds hulp vroeg bij het debuggen van zijn cv-besturingssysteem, omdat hij steeds, letterlijk en figuurlijk weer in de kou kwam te staan?
Het volgende deel van de schakeling (4069) is in theorie overbodig, want die zit al in het joypadje van de Super Nintendo. Alle contactpunten van de schakelaars liggen met een kant aan aarde. Als je dus nu de juiste uitgangen van de beide 74373's aan de andere kant van de juiste schakelaars zou zetten, dan zou dat ook best werken. Maar dit wordt misschien al gauw een rommeltje, dus is het gemakkelijker om misschien maar de hele pad na te bouwen die eigenlijk maar uit twee IC's bestaat. De IC's schuiven met twee stuursignalen de twaalf bits serieel door één lijntje naar buiten.
De software stelt eigenlijk ook niet zoveel voor. We sturen om alle knoppen in de juiste stand te zetten twee bytes naar de printerpoort. Het beste is om het strobe-signaal zelf te genereren in plaats van LPRINT te gebruiken. Het strobe-signaal dient als het ware als een activeersignaal voor de schakeling om er iets mee te gaan doen. In de tabel hierboven staat hoe de data in elkaar steekt.
Als een button ingedrukt wordt, dan wordt de desbetreffende bit 0, anders is hij 1. Als we dus door willen geven dat we de startknop en de R-knop indrukken dan ziet dat er als volgt uit:

```
OUT &H91,&B11011111     '   data byte l
OUT &H90,1:
OUT &H90,0              '   strobe-signaal
OUT &H91,&B00011011     '   data byte  2
OUT &H90,1:
OUT&H90,0               '   strobe-signaal
```

We hebben een machinetaalprogramma gemaakt, omdat het in basic behoorlijk wat regels vergt om bijvoorbeeld het gelijktijdig indrukken van een X en een L om te zetten naar het dataformaat. De toetsen en de cursor komen overeen met die van de joypad, waarbij S=start en [SEL]=select.
De basic loader

```
10 BLOAD "scanl.bin"
20 DEFUSR=&HD000
30 A=USR(0):GOTO 30
```

Dit alleen voor degenen die niet met een assembler om kunnen gaan.

```
        ORG   &HD000

; S      = start
; SELECT = select
; B, A, X, Y, L, R = GELIJK
; CURSOR = RICHTING

BB:
        CALL  SCAN
        LD    A,H
        CALL  PRIDAT
        LD    A,L
        CALL  PRIDAT
        LD    (DATA),HL
        RET

DATA:   DEFW  0

; === Scan key-board ===
; OUPUT: H = #1 BYTE FOR OUTPUT
;        L = #2 BYTE FOR OUTPUT

SCAN:   ;SCAN key-board
        LD    HL,&HFFF3       ;reset first

        LD    A,(&HFBEC)      ;scan 'SELECT'
        ADD   A,A             ;skip
        ADD   A,A
        RL    H               ;out: H = 1111111E

        LD    A,(&HFBEA)      ;scan 'S'
        SRL   A
        RL    H               ;out: H = 111111ES

        LD    A,(&HFBED)      ;scan cursor
        ADD   A,A
        LD    B,A
        RL    H
SLA   B
        SLA   B
        SLA   B
        RL    H
        ADD   A,A
        RL    H
        ADD   A,A
        RL    H               ;out: H = 11ESRLDU

        LD    A,(&HFBE7)
        ADD   A,A
        RL    H               ;out: H = 1ESRLDUB

        LD    A,(&HFBEA)      ;scan 'X'
        ADD   A,A             ;skip 2 bits
        ADD   A,A
        ADD   A,A
        RL    L               ;out: L = 1111011X

        LD    A,(&HFBE9)      ;scan 'L'
        SRL   A               ;skip 1 bit
```

```
        SRL   A
        RL    L               ;out: L = 111011XL

        LD    A,(&HFBE9)      ;scan 'R'
        ADD   A,A
        RL    L               ;out: L = 11011XLR

        LD    A,(&HFBE7)      ;scan 'A'
        ADD   A,A             ;skip 1 bit
        ADD   A,A
        RL    L               ;out: L = 1011XLRA

        LD    A,(&HFBEA)      ;scan 'Y'
        ADD   A,A             ;skip 1 bit
        ADD   A,A
        RL    L               ;out: L = 011XLRAY

        RET
;===== PRInt DATa =======
PRIDAT:
        OUT   (&H91),A        ;output data
        XOR   A
        OUT   (&H90),A        ;strobe signal enable  (0)
        LD    A,1
        OUT   (&H90),A        ;strobe signal disable (1)

        RET
EE:
```

DATABIN.BAS

```
10 ' genereert SCAN1.BIN
20 FOR I=&HD000 TO &HD06C: READ A: POKE I,A: NEXT
30 BSAVE "SCAN1.BIN",&HD000,&HD06C: END
40 DATA &HCD,&H11,&HD0,&H7C,&HCD,&H63,&HD0,&H7D
50 DATA &HCD,&H63,&HD0,&H22,&H0F,&HD0,&HC9,&H00
60 DATA &H00,&H21,&HF3,&HFF,&H3A,&HEC,&HFB,&H87
70 DATA &H87,&HCB,&H14,&H3A,&HEA,&HFB,&HCB,&H3F
80 DATA &HCB,&H14,&H3A,&HED,&HFB,&H87,&H47,&HCB
90 DATA &H14,&HCB,&H20,&HCB,&H20,&HCB,&H20,&HCB
100 DATA &H14,&H87,&HCB,&H14,&H87,&HCB,&H14,&H3A
110 DATA &HE7,&HFB,&H87,&HCB,&H14,&H3A,&HEA,&HFB
120 DATA &H87,&H87,&H87,&HCB,&H15,&H3A,&HE9,&HFB
130 DATA &HCB,&H3F,&HCB,&H3F,&HCB,&H15,&H3A,&HE9
140 DATA &HFB,&H87,&HCB,&H15,&H3A,&HE7,&HFB,&H87
150 DATA &H87,&HCB,&H15,&H3A,&HEA,&HFB,&H87,&H87
160 DATA &HCB,&H15,&HC9,&HD3,&H91,&HAF,&HD3,&H90
170 DATA &H3E,&H1,&HD3,&H90,&HC9
```