

MSX turbo R

De MSX Turbo R is een fantastische computer, daar zijn alle kenners het over eens. De computer is nu al meer dan een jaar in beperkte mate op de Nederlandse markt te koop. Toch wil de gemiddelde gebruiker zijn MSX2 nog niet wegdoen om een nieuwe te kopen. De eerste reden daarvoor is dat de MSX waar men mee werkt nog niet kapot is. De tweede reden om de overstap naar MSX Turbo R nog niet te maken is de onbekendheid met deze super-computer. En U weet; onbekend maakt onbemind.

Het is daarom dat Andre Ligthart, hij is ondermeer de schrijver van Squeek, de MSX Turbo R eens goed onder handen heeft genomen om uit te zoeken wat alle mogelijkheden zijn. Want naast de snellere processor bevat de MSX-turbo R vele andere extra's. In dit artikel worden software-technische tips en trucs gegeven voor iedereen die van plan is om zelf specifieke MSX-turbo R software te gaan schrijven of gewoon ge-intresseerden. Voor de MSX-turbo R is er in Nederland nog steeds geen DEMO gemaakt.

De volgende onderwerpen komen aan bod: Op de MSX Turbo R zitten twee extra LED's; hoe zijn deze aan te sturen. Uileg over de pause-toets, en of deze hard- of softwarematig is. De interne-software-selector, en hoe deze uit te lezen is. De drie verschillende modes: Z80-mode, R800/ROM-mode, R800/DRAM-mode. De Ren Sha-Turbo / Autofire en wat er mee gedaan kan worden. De twee extra JA & NEE-toetsen en hoe deze uit te lezen zijn. De PCM chips. Een misterieuze razendsnelle counter. 16kB S-RAM. En de onthulling: Is de VDP echt sneller geworden?

TWEE EXTRA LED's.

Naast de reeds bekende LED's "caps", "kana", "power" en die van de diskdrive, heeft de MSX-turbo R een pause-LED en een TURBO-LED. Deze kunnen worden aan gestuurd door middel van de I/O poort 0A7H. Het bit 0 komt overeen met de pause-LED en bit 7 met de TURBO-LED. De stand van de LED's kan niet worden uitgelezen. Voorbeeld:

```
OUT &HA7,&B00000001 ; stuurt PAUSE LED aan.
OUT &HA7,&B10000000 ; stuurt TURBO LED aan.
```

De pause-toets.

We zagen hierboven dat de pause-LED softwarematig kan worden aangestuurd. Ook het uitlezen van de pause-toets geschiedt softwarematig en wel door middel van het uitlezen van bit 0 van I/O poort 0A7H. De MSX turbo R controleert de stand van de pause-toets in zijn interruptroutine, 60 maal per seconde. Wanneer deze is ingedrukt, wordt de pause-LED aangezet en het geluid uitgezet. Nu wacht de MSX-turbo R totdat de pause-toets weer wordt ingedrukt, waarna het geluid wordt aangezet en de pause-LED weer uit gaat. Als we de interruptroutines uitschakelen, zal de computer niet meer op de pause-toets reageren, zodat het mogelijk is de pause-functie van de MSX-turbo R te omzeilen. Het volgende voorbeeldprogramma bewijst dit.

```
10 VDP(1)=VDP(1) XOR 32 ; VDP interrupt disable
20 PRINT INP(&HA7); ; Print inhoud poort 0A7H
30 GOTO 20
```

Tijdens het runnen wordt het beeld gevuld met nullen, maar na het bedienen van de pause-toets wordt bit 0 van I/O poort 0A7H hoog en zal het scherm met enen worden gevuld. De computer pauzeert echter niet, omdat in regel 10 de VDP wordt verteld dat deze geen interrupts meer moet genereren. Na het loslaten van de pause-toets verschijnen er nog steeds enen. Dit betekent, in tegenstelling tot alle andere toetsen, dat bij het bedienen van de pause-toets bit 0 van I/O poort 0A7H wordt geïnverteerd en niet alleen hoog wordt tijdens het bedienen. Aangezien de keyboard-scan routine via de interrupt wordt afgehandeld, is het niet mogelijk dit voorbeeldprogramma te onderbreken. Het is wel mogelijk de interrupt weer te enablen door de computer nogmaals regel 10 uit te laten voeren. Bedenk wel dat instructies zoals INKEY\$, INPUT\$(x), ON STOP GOSUB enz. niet werken wanneer de interrupt is uitgeschakeld.

DE INTERNE-SOFTWARE-SELECTOR.

Links van de reset-toets zit een schakelaar met een onbekende naam, ik noem hem de ISS. Bij het opstarten leest de computer de stand uit van deze schakelaar, staat deze in de stand 'links', dan wordt een





gigantisch groot softwarepakket opgestart. Het is misschien leuk om te weten hoe deze switch wordt uitgelezen, het gaat als volgt:

```
10 OUT &HE4,5 ;Geeft 0E5H een functie
20 A = INP(&HE5) AND 64 ;Leest bit 6 van 0E5H
30 IF A = 0 THEN PRINT "Rechts" ;Als laag, print 'Rechts'
40 IF A = 64 THEN PRINT "Links" ;Als hoog, print 'Links'
50 GOTO 20
```

In regel 10 wordt het getal vijf in I/O poort 0E4H geschreven, wat aangeeft dat I/O poort 0E5H moet openstaan voor het lezen van de ISS. Met deze kennis kan in een spel of demo eens iets anders geschakeld worden met de ISS.

Z80-MODE, R800-MODE, R800/DRAM-MODE
Zoals reeds wel bekend zal zijn, heeft de MSX-turbo R 3 modi om in te werken. In de Z80-mode zal de computer qua snelheid gelijk zijn aan elke andere MSX. Deze mode wordt geïnitieerd bij het opstarten als er een MSX-DOS 1.x diskette in de diskdrive zit of als tijdens het opstarten de '1' wordt ingedrukt totdat de BEEP klinkt. De R800/DRAM-mode wordt geïnitieerd als er een MSX-DOS 2.x diskette of juist helemaal geen disk in de drive zit. Deze mode houdt in dat de computer in zijn snelste stand staat, waarvoor een snellere processor wordt gebruikt (R800, Z80 compatible) en van een gedeelte van de meest gebruikte routines uit het ROM wordt in RAM gezet (DRAM staat voor Dynamische RAM). Deze laatste optie is van belang wanneer er veel van ROM-routines gebruik wordt gemaakt, aangezien RAM's sneller kunnen werken dan ROM's van de zelfde prijsklasse. De DRAM-optie gebruikt echter 64 kB RAM, zodat er nog 192 kB RAM voor eigen gebruik over blijft. Voor programma's die geen of weinig ROM-routines gebruiken of de volledige 256 kB RAM nodig hebben, kan tenslotte de R800-mode (zonder DRAM) worden geselecteerd, die in BASIC niet gebruikt wordt. Er zijn ROM routines die de mogelijkheid bieden om te schakelen naar een andere mode of om de actuele mode te lezen.

```
00180H - CHGCPU - Invoer: A = LxxxxMM
(x=0
L = 1 als LED mee moet veranderen
MM = 00 voor Z80 mode
MM = 01 voor R800 mode
MM = 10 voor R800/DRAM mode)
00183H - GETCPU - Uitvoer: A = xxxxxxMM
(betekenis MM idem aan CHGCPU.)
```

Als we met behulp van CHGCPU van de Z80-mode naar de R800-mode om willen schakelen, maar voor de gebruiker moet het lijken alsof de computer nog in de Z80-mode staat, dan moet bit 7 van de Accu (L) laag worden. Deze mogelijkheid wordt ook door BASIC gebruikt bij het opnemen / weergeven van geluid door middel van de PCM chip, omdat de MSX turbo R tijdens het opnemen / weergeven zijn volledige snelheid hard nodig heeft. Na het opnemen of weergeven zal de MSX turbo R weer netjes de mode inschakelen die in het begin gelezen werd en de gebruiker merkt er niets van! Met het volgende voorbeeldprogramma kunnen we de actuele mode van de MSX turbo R in BASIC uitlezen.

```
10 OUT &HE4,6 ;geeft 0E5H een functie
20 A = INP(&HE5) ;leest mode
30 IF A = 96 THEN PRINT "Z80 mode"
40 IF A = 64 THEN PRINT "R800 mode"
50 IF A = 0 THEN PRINT "R800/DRAM mode"
```

Ook hier wordt - net als bij de ISS - gelezen via I/O poort 0E5H, maar omdat nu het getal zes in I/O poort 0E4H wordt gezet staat I/O poort 0E5H open voor het lezen van de actuele processor-mode. Het omschakelen naar een andere processor-mode zou mogelijk moeten zijn door naar I/O poort 0E5H te schrijven. Dit is ook mogelijk, alleen in BASIC werkt het niet altijd, omdat bij het omschakelen de interne registers niet onveranderd blijven (ook de Stack Pointer niet). De BIOS-routine CHGCPU regelt dit allemaal door eerst alle registers op de stack te plaatsen, daarna SP op een vast RAM adres te POKEn en dan pas om te schakelen naar een andere mode. Nu wordt Stack Pointer weer gelezen en de overige registers worden van de stack gehaald.

De Ren Sha-Turbo / Het AUTOFIRE.

De MSX-turbo R is -boven de functietoetsen- voorzien van een naar links en rechts schuifbare potentiometer, waarvan links een LED is gemonteerd die in een bepaald tempo knippert. Het tempo wordt hoger wanneer de potentiometer meer naar rechts bewogen wordt. De LED is uit als de potmeter in zijn meest linkse positie staat. Wanneer de pause-toets wordt bediend, zal de knipperende LED doven, wat doet denken alsof de Ren Sha-Turbo of autofire LED -net als de pause-toets- ook softwarematig wordt aangestuurd. Dit is echter niet het geval, omdat de autofire-LED en de Pause-LED hardwarematig aan elkaar gekoppeld zijn zodat deze niet tegelijk kunnen branden. De blokpuls die de autofire-LED aangeeft wordt hardwarematig geORD met bit 0 van de 8e rij van de keyboard-matrix (daar zit de spatiebalk aan gekoppeld) en met bit 4 van register 14 van de PSG (TRIGGER-A van JOYSTICK 1 en 2). De Ren Sha-Turbo werkt dus niet op vuurknop B van de joysticks of muis. Aangezien de bits van de registers waarmee de spatiebalk of de vuurknop A worden uitgelezen 'laag-actief' zijn, wordt tijdens het indrukken van spatiebalk of de vuurknop A het actief zijn onderbroken op die momenten wanneer de autofire-LED brandt. Met dit vrij technische verhaal probeer ik duidelijk te maken dat we softwarematig niet veel meer met autofire kunnen als voor gebruik in actiespelen. Het is namelijk niet mogelijk de blokpuls via een register uit te lezen zonder een vuurknop of de spatiebalk te bedienen. Had dit wel mogelijk geweest, dan was de potentiometer voor meer doeleinden bruikbaar, omdat de computer de stand van de potmeter precies bij zou kunnen houden door middel van het meten van de lengte van één puls. Nu zou de computer eerst moeten vragen even de spatiebalk of vuurknop A te bedienen alvorens de lengte van de blokpuls te meten.

De JA en NEE-TOETSEN.

Aan beide kanten van spatiebalk zitten twee nieuwe toetsen die japanse namen hebben en worden toegepast in het ingebouwde japanse softwarepakket. De kleine toets -links van de spatiebalk- betekent NEE of "niet accepteren". De andere, wat grotere toets, betekent JA of "accepteren". De toetsen zijn toegevoegd aan de keyboard-matrix. De JA-toets komt overeen met bit 1 van rij 11 van de keyboard-matrix en de NEE-toets zit gekoppeld aan bit 3 van rij 11. De overige 6 bits van deze nieuwe rij 11 zijn ongebruikt. In het voorbeeld hieronder wordt getoond hoe men in assembleertaal de toetsen kan uitlezen:

```
ST: LD A,11 ;Rij 11
CALL SNSMAT ;SNSMAT = 00141H
BIT 1,A ;Test bit 1
JR Z,JA ;Naar 'JA' als laag
BIT 3,A ;Test bit 3
JR Z,NEE ;Naar 'NEE' als laag
```

DE PCM CHIP.

Met de PCM-chips kan zowel geluid afgespeeld als opgenomen worden. Het opnemen kan geschieden met behulp van de interne microfoon, maar het is ook

mogelijk extern een signaal te sturen zodat de ingebouwde microfoon automatisch wordt afgekoppeld. Voor het opnemen of afspelen is BASIC versie 4.0 voorzien van nieuwe opdrachten: PCMPPLAY voor het afspelen en PCMREC om op te nemen. De syntax van deze opdrachten is als volgt:

```
CALL PCMPPLAY(@<startadres>,<eindadres>,<kwaliteit>,[s])
```

```
CALL PCMREC(@<startadres>,<eindadres>,<kwaliteit>,[reactie],[bespaaroptie],[s])
```

```
startadres : 0-65535
eindadres : 0-65535
kwaliteit : 0-3 (0 voor hoogste samplefreq.)
VRAM : 's' voor opslag in videogeheugen.
reactie : 0-127 (reageert bij bepaald volume)
bespaaroptie: 1 invullen om geheugen te besparen.
```

Wanneer men PCMREC toepast, dan is het mogelijk de computer te laten wachten met opnemen totdat het volume een bepaalde waarde heeft bereikt, hiermee wordt "reactie" bedoeld. Als voor "reactie" een nul ingevuld wordt, dan zal de computer meteen reageren. De geheugenbespaaroptie houdt in dat wanneer het volume voor een bepaalde periode nul is, dit verkort wordt gecodeerd in de sample-data. Wanneer deze optie niet wordt gebruikt, dan zal het geheugen met een constant tempo worden gevuld. In de data zullen dan geen nullen voorkomen, omdat een nul gevolgt door een "teller" is gereserveerd voor de bespaaroptie. De teller na de nul geeft aan hoelang het volume nul was en zo zal bij "stilte" het geheugen minder snel worden gevuld. Een voorbeeldprogramma maakt dit duidelijk zichtbaar:

```
10 SCREEN 7
20 CALL PCMREC(@&H0000,&HD3FF,0,0,1,S)
```

Het is nu duidelijk zichtbaar dat het scherm (video geheugen) langzamer wordt gevuld wanneer er stilte heerst. Als in regel 20 de 1 in 0 veranderd wordt, dan zal het scherm wel met een constant tempo worden gevuld. Nu iets meer over de I/O met de PCM chips. Voor I/O worden twee poorten gebruikt, namelijk 0A5H en 0A4H, waarnaar de sample-data tijdens het afspelen wordt geschreven. Wanneer we I/O poort 0A4H gaan lezen blijkt deze poort te zijn gekoppeld aan een 2 bit counter die van 0 t/m 3 telt. Deze counter wordt gebruikt als timer door de ROM-routines. Na een sample genomen te hebben of data geschreven te hebben zal de counter worden geRESET en zal de computer wachten totdat de counter bij 0, 1, 2 of 3 is aangekomen, afhankelijk van de ingestelde kwaliteit wat ook van 0 t/m 3 loopt. Het volgende voorbeeldprogramma laat de counter zien:

```
10 PRINT INP(&HA4);:GOTO 10
```

Als dit voorbeeldprogramma in de turbo-mode wordt gerund, heeft BASIC ongeveer net zoveel tijd nodig om één waarde te printen, als dat de counter met één wordt verhoogd, zodat het effect duidelijk zichtbaar is. I/O poort 0A5H kan zowel beschreven als gelezen



worden. Bij het schrijven worden de bits 01/m 4 gebruikt en bij het lezen wordt naast deze 5 bits ook bit 7 gebruikt en wel om geluid op te nemen. U zult zich afvragen; gebeurt dit dan maar met één bit? Ja, wanneer geluid moet worden opgenomen, dan moet dit serieel gebeuren door middel van het uitlezen van bit 7 van I/O poort 0A5H. Voordat één zo'n bit uitgelezen kan worden, moet I/O poort 0A4H met een waarde worden beschreven die afhankelijk is van de al eerder van I/O poort 0A5H gelezen bit 7. Hieruit blijkt dat de MSX-turbo R zeer veel instructies moet doorlopen om één byte te samplen. In BASIC kunnen we bit 7 van I/O poort 0A5H lezen om te meten of het volume boven een bepaalde grens komt. Het volgende voorbeeldprogramma geeft aan of het stil genoeg is of niet.

```
10 SCREEN 0:COLOR 15,4:GRENS = 10
20 OUT &HA5,12
30 OUT &HA4,128+GRENS
40 T=0:FOR I=1 TO 100
50 IF INP(&HA5)>128 THEN T=T+1:COLOR ,1
   ELSE COLOR ,4
60 NEXT:LOCATE 0,0
70 IF T<5 THEN PRINT "Wat een stilte"
80 IF T>30 THEN PRINT "Wat een lawaai"
90 GOTO 30
```

In regel 20 worden de PCM-chips ingesteld om te lezen en op I/O poort 0A4H wordt het volume geschreven waarop de computer moet reageren. Er wordt 128 bij opgeteld, omdat 128 de nullijn is (2 complement gecodeerd). De "grens" kan naar eigen inzicht worden ingesteld. Er wordt 100 maal gelezen of bit 7 van I/O poort 0A5H hoog is. Als dit minder als 5 maal het geval was, dan registreert de computer "stilte" en als meer dan 30 keer bit 7 van I/O poort 0A5H hoog was, dan zal de melding "wat een lawaai" worden geprint. Normaal gesproken -in BASIC- zal I/O poort 0A5H het getal 3 bevatten, bit 1 van dit getal is dus hoog. Als deze laag is, wat bij 12 in het voorbeeldprogramma ook geldt, dan wordt er geen geluid naar buiten gebracht. Ook de PSG en de OPLL zijn dan niet meer hoorbaar.

OUT &HA5,1 ; geluid uit.

Het in één keer uitzetten van al de geluidchips die de MSX-turbo R heeft gaat op deze manier het beste. Wanneer een spel op pause wordt gezet, kan dit worden toegepast. Ook wanneer de pause-toets wordt bediend, zorgt de MSX-turbo R ervoor dat met deze instructie het geluid tijdelijk wordt uitgezet, zodat er geen "hangende" toon hoorbaar blijft. Door bit 3 van I/O poort 0A5H hoog te maken, wordt het geluid direct opgenomen en doorgestuurd naar de geluidsuitgang, indien bit 1 natuurlijk hoog is.

OUT &HA5,11 ; turbo R als versterker

Zet de monitor of TV niet te hard, want dan is het bekende rondzingen of "tuut-effect" hoorbaar (een fluittoon die steeds harder wordt). Het gebruik van een externe microfoon is noodzakelijk als de computer als versterker moet dienen. Wanneer na het invoeren van deze OUT-instructie een spel / demo wordt

geladen, is het mogelijk dwars door de FM/PSG muziek te praten.

EEN RAZENDSNELLE COUNTER

We zagen net al dat de PCM-routines gebruik maken van een 2 bits counter om in 4 verschillende snelheden te samplen. De processor is ook aangesloten op een 16 bit counter. Waar hij zit? De low-byte is te ver- krijgen door I/O poort 0E6H uit te lezen. De high- byte zit aan I/O poort 0E7H. Deze counter gaat zo snel, dat de Z80 zou willen dat hij het bij kon houden. De R800 redt dit wel. Door de hoge snelheid van de counter, kan de low-byte in BASIC als een soort willekeurigheidsgetal worden toegepast. Zelfs de high-byte telt bijna snel genoeg om dit in BASIC als random te beschouwen.

```
10 A=INP(&HE6) : 'Lees lowbyte van counter (8 bit)
20 B=A MOD 16 : 'B wordt laagste 4 bits
30 C=INT(A/16) : 'C wordt hoogste 4 bits
40 PRINT"Hoeveel is";B;"*";C;
50 INPUT D
60 IF D=B*C THEN PRINT "Goed" ELSE PRINT
   "Fout"
70 GOTO 10
```

De 16 kB SRAM

In de MSX-turbo R is 16kB SRAM aanwezig. Dit S-RAM is qua aansturing niet compatible met het S-RAM dat in de PAC, FM-PAC of data-cartridge zit. De 16kB S-RAM in de MSX turbo R wordt namelijk gebruikt door het ingebouwde softwarepakket. Er worden allerlei instellingen in opgeslagen, bijvoorbeeld hoe de aanwijzer bestuurd moet worden, met muis, keyboard of joystick. Nadat ik het S-RAM vol met tekst had gezet en daarna eens flink met het japanse programma had gespeeld, bleek het overgrote gedeelte van de 16kB S-RAM onveranderd. Het S-RAM is dus best te gebruiken voor andere toepassingen, je moet alleen even weten welke bytes door het japanse softwarepakket veranderd kunnen worden. Het S-RAM is alleen te adresseren via slot 3-3. In slot 3-3 kan namelijk elk willekeurig blok ROM met behulp van omschakeladressen in dit slot worden geschakeld. Elk blok ROM van 8kB heeft een nummer en ook de 16kB S-RAM is opgesplitst in 2 blokken van 8 kB S-RAM en hebben net als elk blok ROM ook een nummer. In totaal zijn er 192 blokken van 8kB aanwezig. De S-RAM kan in slot 3-3 geschakeld worden door het getal 128 of 129 te laden in één van de volgende adressen:

```
06000H - selecteer blok van 00000H-01FFFFH
06400H - selecteer blok van 02000H-03FFFFH
06800H - selecteer blok van 04000H-05FFFFH
06C00H - selecteer blok van 06000H-07FFFFH
07000H - selecteer blok van 08000H-09FFFFH
07400H - selecteer blok van 0C000H-0DFFFFH
07800H - selecteer blok van 0A000H-0BFFFFH
07C00H - selecteer blok van 0E000H-0FFFFFH
```

Een machinetaalprogrammeur kan met deze informatie uit de voeten, maar de BASIC programmeur kan hier niets mee. Voor de volgende MSX Mozaïk zal ik een klein machinetaal programma maken, die via BASIC naar het S-RAM kan schrijven en lezen door simpelweg het adres en de data op te geven.