MSX 2+

# HANDLEIDING
# BASIC 3.0

# KORTE HANDLEIDING MSX 2 PLUS

## ALGEMEEN

Met de schakelaar die op uw computer gemonteerd is kunt u schakelen tussen de MSX 2 en MSX 2+ stand, dit is nodig daar sommige MSX 2 software niet zal werken op de 2+ stand.

Uw MSX 2+ heeft een normaal Europeesch qwerty toetsenbord, de Japanse karakterset die normaal in een Japanse MSX 2+ aanwezig is hebben wij niet gehandhaafd, dit om het gebruik van de codetoets te laten zoals hij bij een Europeesche machine is.
Verder heeft uw MSX 2+ computer een Kanji rom aan boord waarmee het mogelijk is om grote letters te printen op uw scherm.
Dit doet u met call kanji.
Ook kan hiermee een stuk tekst op een eenvoudige wijze in een grafisch scherm geplaatst worden.
Bij de MSX 2 gebeurt dit door het commando 'open' te gebruiken, bij de 2+ kan het als volgt:

```
10 SCREEN..
20 CALL KANJI
30 PRINT "MSX"
```

Dit programma werkt op elk screen.
Call kanji kan gevolgdt worden door een cijfer, bijvoorbeeld call kanji 0, 1, 2 of 3, hiermee veranderd de grote van de letters die op uw scherm verschijnen.
Het uitschakelen van de z.g. kanji 'mode' gebeurt door call ank, waarna uw computer weer normale letters op het scherm print.

Functietoets F7 is aangepast, inplaats van CLOAD zien we nu LOAD.

Verder valt meteen op dat het hier om Basic versie 3.0 gaat in plaats van basic 2.1. Deze basic 3.0 is 80 K groot, zonder diskbasic, dus basic is 32 K groter geworden. De aanvulling bestaat hoofdzakelijk uit commando's die de nieuwe screens besturen, n.l. screen 10, 11 en 12; tevens is er een scroll commando, dat op elke screen werkt. Om de screens volledig te besturen zijn er ook een aantal registers bijgekomen (vdp's 25 t/m 28). Deze screens zijn vooral bedoeld om het nieuwe aantal kleuren te verwerken, want screen 12 telt bijvoorbeeld 19.268 kleuren.

## HET GROTE VERSCHIL

Het grote verschil tussen de MSX 2 en de MSX 2+ is dat de 2+ drie screens meer heeft (screen 10, 11 en 12) en het aantal kleuren is sterk toegenomen. Om dit verschil te laten zien volgt een tabel van alle kleuren. Screen 0 en 1 zijn tekstschermen, zodoende is de resolutie niet gegeven, maar het maximale aantal karakters dat op een regel kan. Tevens is bij deze screens (0 en 1) het aantal kleuren dat gebruikt kan worden toegenomen. De MSX 2 kan maar twee kleuren tegelijk aan, bij de MSX 2+ zijn dat er 16. Bij het aantal kleuren kan bijvoorbeeld staan: 16/512. Dit betekent dat er maximaal 16 verschillende kleuren op het beeldscherm kunnen en deze kleuren kunnen gekozen worden uit een kleurpalet vasn 512 kleuren.

| SCREEN | RESOLUTIE | AANTAL KLEUREN | AANTAL PAGE |
|---|---|---|---|
| 0 | 24 regels 80 karakters | 16/512 | |
| 1 | 24 regels 32 karakters | 16/512 | |
| 2 | 256*192 | 16/512 | |
| 3 | 64*48 | 16/512 | |
| 4 | 256*192 | 16/512 | |
| 5 | 256*212 | 16/512 | 4 |
| 6 | 512*212 | 4/512 | 4 |
| 7 | 512*212 | 16/512 | 2 |
| 8 | 256*212 | 256 | 2 |
| 10 | 256*212 | 12.499 max. 16 op een regel | 2 |
| 11 | 256*212 | 12.499 max. 256 op een regel | 2 |
| 12 | 256*212 | 19.268 max. 256 op een regel | 2 |

KLEURENOPBOUW van SCREEN 10, 11 en 12

Elke pixel bestaat uit de drie basiskleuren ROOD, GROEN en BLAUW. In screen 8 kan de intensiteit van elke basiskleur zo ingesteld worden dat we 256 kleuren krijgen. De kleur per pixel is onafhankelijk van de kleur die de voorgaande pixel heeft. Dit geldt voor de screens 0 t/m 8. In screen 10 t/m 12 is dit anders, de voorgaande pixel beinvloedt de kleur van de pixel nu wel. Om alle kleuren te krijgen in deze screens hebben we vier pixels naast elkaar nodig, die een kleur kunnen vormen. Dat wil zeggen dat niet elke kleurencombinatie een kleur vormt. Dit hangt af van de verschillende kleurencombinaties. Om pixels wat betreft kleur op elkaar af te stemmen zijn een paar formules nodig. Deze formules zijn voor elke pixel gelijk.

Stel we hebben vier pixels:

| 0 | 1 | 2 | 3 |
|---|---|---|---|

Elke pixel bestaat dus uit drie basiskleuren die per kleur in intensiteit verschillen. Om kleurencombinaties voor elke pixel te berekenen worden de volgende formules gebruikt:

```
------------------------------------------------------------
  R0=Y0+J                          R1=Y1+J
  G0=Y0+K                          G1=Y1+K
  B0=5/4*Y0-1/2*J-1/4*K            B1=5/4*Y1-1/2*J-1/4*K

  R2=Y2+J                          R3=Y3+J
  G2=Y2+K                          G3=Y3+K
  B2=5/4*Y2-1/2*J-1/4*K            B3=5/4*Y3-1/2*J-1/4*K
------------------------------------------------------------
```

De R staat voor rood, de G voor groen en de B voor blauw. Bij de getallen K en J kunnen we een, twee, drie of vier kleuren krijgen. Tevens kunnen we als we de juiste verhouding van J en K hebben, de kleuren laten overlopen. Dit is eigenlijk de kracht van de MSX 2+; de kleuren vloeiend over laten lopen. Voor Y nemen we vaak een vast getal. Om een kleur te gebruiken zullen we dus vier pixels op het beeldscherm moeten plaatsen. Dit kunnen we doen door het commando PSET. Bij dit commando is het belangrijk dat we weten dat er een logische operatie achter kan. Dit zijn de bewerkingen OR, AND en XOR. Deze operanten bepalen de kleur die het beeldscherm weergeeft. We geven het PSET commando kleur Z. De uiteindelijke kleur C wordt bepaald door de logische operatie met het gegeven paletnummer Z en het kleurennummer S van het opgegeven beeldschermpunt. De werking is als volgt:

```
--------------------------------
 XOR        C=NOT(Z)*S+Z*NOT(S)
 OR         C=Z+S
 AND        C=Z*S
 PSET       C=Z
 PRESET     C=NOT(Z)
--------------------------------
```

We zullen dit verduidelijken met een voorbeeld: I bepaalt de kleur:

```
----------------
 10 SCREEN 10
 20 FOR I=0 TO 7
 30 PSET (I,0),I
 40 NEXT I
 50 GOTO 50
----------------
```

Nu met een operant:

```
-------------------
 10 SCREEN 10
 20 FOR I=0 TO 7
 30 PSET (I,0),I,OR
 40 NEXT I
 50 GOTO 50
-------------------
```

Nu met een gecombineerde operant:

```
----------------------
 10 SCREEN 10
 20 FOR I=0 TO 7
 30 PSET (I,0),I OR 5
 40 NEXT I
 50 GOTO 50
----------------------
```

Let er wel op dat bij het laatste voorbeeld er geen komma staat tussen I en OR. Deze voorbeelden werken ook in screen 11 en 12. We zullen dit gedeelte afsluiten met een wat groter voorbeeld, hier kunnen we vele kleuren zien van screen 11 en 12. De CALL KANJI is alleen gebruikt om grotere letters op het beeldscherm te krijgen.

```
-------------------------------------------------
 100 'SCREEN11
 110 '
 120  KANJI0:WIDTH 64:DEFINT A-Z:YY=0
 130 CLEAR:DEFINT A-Z:YY=0
 140 ON STOP GOSUB 400:STOP ON
 150 SCREEN 12:COLOR &HF8,0,0:CALL CLS
 160 '
 170 FOR K=-32 TO 31:YP=112+K+K
 180 FOR J=-32 TO 31:XP=128+J*4
 190 PSET (XP,YP),K AND 7
 200 PSET (XP,YP+1),K AND 7
 210 PSET (XP+1,YP), (K AND 56)\8
 220 PSET (XP+1,YP+1), (K AND 56)\8
 230 PSET (XP+2,YP),J AND 7
 240 PSET (XP+2,YP+1),J AND 7
 250 PSET (XP+3,YP), (K AND 56)\8
 260 PSET (XP+3,YP+1), (K AND 56)\8
 270 NEXT:NEXT
 280 '
 290 SC=12:NS=1:SCREEN12
 300 FOR Y=1 TO 31:GOSUB 360:NEXT
 310 FOR Y=30 TO 0 STEP -1:GOSUB 360:NEXT
 320 SC=11:NS=1:SCREEN11
 330 FOR Y=2 TO30 STEP 2 :GOSUB 360:NEXT
 340 FOR Y=28TO0  STEP-2 :GOSUB 360:NEXT
 350 GOTO 280
 360 '
 370 IF NS THEN LOCATE 1,0:PRINT USING"SCREEN ##";
 SC:NS=0
 380 '
 390 LINE (0,48)-(255,175),(Y XOR YY)*8,BF,XOR:YY=
 Y:RETURN
 400 '
 410 SCREEN 0:COLOR 15,4,7:END
-------------------------------------------------
```

## BASIC COMMANDO'S

We zullen nu de basic commando's bespreken die nieuw of uitgebreid zijn ten opzichte van de MSX 2.

### BASE(N)

Met het Base commando kunnen we het startadres van de VDP tabellen vinden. Voor MSX 1 mag N de waarde hebben van 0 t/m 19. Voor MSX 2 mag N de waarde hebben van 0 t/m 44. Voor MSX 2+ mag N de waarde hebben van 0 t/m 44 en van 50 t/m 64.

### BLOAD".......",S

Met dit commando kunnen we een programma van disk direct in het videogeheugen plaatsen. Dit commando is uitgebreid ten opzichte van de MSX 2. Het werkt ook op de screens 10 t/m 12.

### BSAVE".......",S

Met dit commando kunnen we een programma dat in het videogeheugen zit direkt wegschrijven naar diskette. Dit commando is uitgebreid ten opzichte van de MSX 2. Het werkt ook op de screens 10 t/m 12.

### CIRCLE

Met dit commando kunnen we een cirkel tekenen op een willekeurig screen, behalve screen 0 en 1. De enige uitbreiding van dit commando is dat het ook werkt op de screens 10, 11 en 12.

### COLOR V.A.R.

V., A. en R staan respectievelijk voor de voorgrond, achtergrond en randkleuren. Dit commando is zodanig uitgebreid dat het ook werkt in de screens 10, 11 en 12.

### COLOR=(P.R.G.B.)

P.R.G. en B staan voor het paletnummer, rood groen en blauw. Dit commando kent een nieuwe combinatie kleurintensiteiten toe aan het paletnummer. Dit statement werkt niet in de screens 8, 10, 11 en 12.

### COLOR=RESTORE

Met dit commando worden de kleurintensiteiten weer op de waarde gezet die in het video-ram opgeslagen liggen. Dit commando heeft in screen 10, 11 en 12 dezelfde werking als in de andere screens. De waarde van de kleurintensiteit in de screens 11 en 12 wordt hetzelfde als in screen 8. De waarde van de kleurintensiteit in screen 10 krijgt de waarde van screen 7.

### COLOR SPRITE

Met dit commando krijgt een sprite een kleur. Dit commando werkt ook in de screens 10, 11 en 12. De werking blijft hetzelfde.

## COPY

Dit commando is aangepast voor screen 10, 11 en 12. Het wordt alleen gebruikt wanneer het commando copy betrekking heeft op het kopieeren van bestanden van en naar het videogeheugen.

## COPY SCREEN

Met dit commando kunnen we een van buiten komend beeld digitaliseren. Het commando is zodanig uitgebreid dat het ook op screen 10 en 12 werkt. Tevens kunnen we met dit commando een kleur meegeven. Er moet wel een digitaliseerder aanwezig zijn.

## DRAW

Met dit commando kunnen lijnen en punten getekend worden op het beeldscherm. Het commando werkt ook op screen 10, 11 en 12.

## KANJI

De kanjimodes stellen de gebruiker in staat tekst te plaatsen in elk gewenst scherm, ook de grafische. BASIC kent 4 Kanji modes, die zijn aan te roepen met _KANJI0, _KANJI1, _KANJI2 en _KANJI3. Houdt u er rekening mee dat er tussen _KANJI en het cijfer GEEN spatie moet, want dat resulteert in een 'Syntax error'. De kanjimodes hebben het volgende effect:

0 - Grote letters op het scherm, hierbij worden de letters getoond zoals op screen 1 het geval is, dus alle 8 pixels op een rijtje.

1 - Grote letters op het scherm, hierbij worden de letters getoond zoals op screen 0 het geval is, dus de laatste 2 pixels vallen weg.

2 - Idem als 0, maar dan met kleine letters.

3 - Idem als 1, maar dan met kleine letters.

Het is nu ook mogelijk tekst op screen 0 in de 15 paletkleuren te printen. De kleuren zijn gewoon met COLOR V,A,R in te stellen, waarbij de cijfers mogen lopen tot de toegestane getallen zoals ze in dat scherm gehanteerd worden.

Een schoon scherm in de kanjimodes is te verkrijgen door middel van _CLS. (GEEN CLS alleen dus!).

Door de speciale schermopbouw van screen 11 en 12 treedt hier een color-spill effect op als we op deze schermen printen. Screen 10 is welliswaar ook anders opgebouwd, maar door een andere behandeling van BASIC bij dit scherm, komt hier alles wel goed op het scherm.

Het printen in en Kanji-mode gaat gewoon met het PRINT commando en ook LOCATE mag gebruikt worden.

Het uitschakelen van de z.g. kanji 'mode' gebeurt door call ank, waarna uw computer weer normale letters op het scherm print.

## LINE

Met dit commando kunnen we lijnen en rechthoeken tekenen in alle screens, behalve in screen 0 en 1. Dit commando is uitgebreid voor de MSX 2+. Screen 10 werkt hetzelfde als screen 2 t/m 7. We kunnen hier 16 verschillende kleuren gebruiken, ook in screen 6. Screen 11 en 12 werken hetzelfde als screen 8.

## LOCATE

Met het locate-commando kunnen we de cursor op een willekeurige plaats zetten op het beeldscherm. MSX 2 kan dit commando alleen gebruiken in screen 0 en 1. Als Kanji van MSX 2+ geactiveerd wordt kan dit commando in elk willekeurig screen worden gebruikt.

## PAINT

Met dit commando kunnen we een figuur of het beeldscherm een kleur geven. Dit commando is dusdanig uitgebreid dat het ook werkt in screen 10 t/m 12. De kleurmogelijkheden zijn als volgt: Screen 2 t/m 7 en 10: er zijn 16 verschillende kleuren mogelijk. Screen 8, 11 en 12: er zijn 256 kleuren mogelijk.

## PRESET en PSET

Met deze commando's kunnen we in de grafische screens de coordinaten van een pixel opgeven en tevens de kleur van deze pixel bepalen. Deze commando's zijn aangepast aan screen 10 t/m 12.

## SCREEN

Met het screencommando bepalen we in welk screen we willen werken. Dit commando is natuurlijk uitgebreid voor screen 10 t/m 12.

## SET SCROLL A,B,C,D

Set scroll is een nieuw commando. Hiermee kunnen we het beeldscherm of een figuur op het beeldscherm willekeurig laten scrollen. De variabelen A, B, C en D hebben de volgende betekenis:

A Met deze variabele kan het beeld naar rechts en links scrollen. De waarde mag liggen tussen 0 en 512.

B Met deze variabele kunnen we het beeldscherm van boven naar beneden en van beneden naar boven laten scrollen. De waarde van deze variabele mag liggen tussen 0 en 256.

C Met deze variabele kunnen we het beeldscherm aan de linkerkant stil zetten. C mag de waarde 0 of 1 hebben.

D Met deze variabele geven we aan of het beeldscherm achter de aktieve pagina ook mee moeten scrollen. De waarde van D mag 0 of 1 zijn.

Met het commando SCROLL werkt in elk screen, al geeft het in screen 0 vreemde effecten. Nu een paar voorbeelden om een indruk te geven wat er met dit commando mogelijk is. We laten eerst een vierkantje scrollen van rechts naar links. Dit kunnen we doen met het volgende programma:

```
 10 SCREEN 5
 20 LINE (10,10)-(100,100),7,BF
 30 FOR I=0 TO 511
 40 SET SCROLL I,1,1,0
 50 NEXT
```

B bepaalt de plaats op het beeldscherm.

We kunnen het blokje ook de andere kant uit laten scrollen. Het enige wat we dan moeten doen is de FOR NEXT laten teruglopen.

```
 10 SCREEN 5
 20 LINE (10,10)-(100,100),7,BF
 30 FOR I=511 TO 0 STEP -1
 40 SET SCROLL I,1,1,0
 50 NEXT
```

Om het scrollen sneller te laten gaan kunnen we achter de FOR NEXT een grotere step plaatsen.

Als we het blokje omhoog willen laten scrollen, zetten we variabele A op 1 of een ander getal en stellen B gelijk aan 1. Om het blokje diagonaal te laten scrollen kunnen we A en B aan 1 gelijk stellen.

```
 10 SCREEN 5
 20 LINE (10,10)-(100,100),7,BF
 30 FOR I=0 TO 511
 40 SET SCROLL I,1,1,0
 50 NEXT
```

Om het blokje nu een willekeurige richting uit te laten scrollen kun je A en B laten lopen zoals je wilt.

Als we twee tekeningen of twee plaatjes achter elkaar willen laten scrollen, moeten we dit doen in screen 5 t/m 12. De lagere screens kunnen dit niet aan, omdat ze geen andere pagina tot hun beschikking hebben dan de geactiveerde pagina. Als we D op 1 stellen scrollen beeldschermpagina's 0 en 1 achter elkaar. We zullen dit laten zien aan de hand van een voorbeeld, de plaatjes scrollen alleen vertikaal.

```
 10 SCREEN12
 20 SET PAGE 0,1
 30 BLOAD"naam1.s12",S
 40 SETPAGE 1,0
 50 BLOAD"naam2.s12",S
 60 FOR I=0 TO 511
 70 SET SCROLL I,1,1,1
 80 NEXT
```

Op de regels 30 en 50 kan ook een tekenopdracht gezet worden. Met dit commando kunnen we zien hoe snel de videoprocessor is. We zullen dit aantonen door de scroll-opdracht steeds sneller te laten gaan.

```
-------------------------
10 SCREEN12
20 SET PAGE 0,1
30 BLOAD"naam1.s12",S
40 SETPAGE 1,0
50 BLOAD"naam2.s12",S
55 X=1
60 FOR I=0 TO 511 STEP X
70 SET SCROLL I,1,1,1
80 NEXT
90 X=X+1
100 GOTO 60
--------------------------
```

VDP

De VDP registers bevatten op de inhoud van het aangegeven besturingsregister van de Video Display Processor. De registers zijn:

0 t/m 8 (voor de MSX 1, MSX 2, MSX 2+)
-9 t/m -1, 9 t/m 24 en 33 t/m 47
(voor de MSX 2 en MSX 2+)
24 t/m 28 (voor de MSX 2+)

Dit was in grote lijnen het verschil van de MSX 2 en de MSX 2+.

KORTE HANDLEINGING MSX-BASIC-KUN (BASIC COMPILER)

MSX-BASIC-KUN is an incredible BASIC compiler. It will compile a BASIC program on memory in few seconds and execute it 15 to 100 times faster!!
It can compile most of the statements and functions of MSX-BASIC and can handle strings and floating numbers. Once you use it, you'd feel you'd never need to learn the Z-80 machine language. Real time games, C.G., demo programs can be written by the ease of BASIC for machine language speed.

*** USAGE ***

1. Settings & General knowledge

Yust type CALL BC and the compiler is active

Now you are in BASIC mode as usual, except that two commands are available.

CALL RUN
CALL TURBO ON/OFF

"CALL" can be written as "_" (underscore). I will use that from now on.

_RUN is the command to compile and execute the entire program on memory.
If it finds an error it will stop and yield the message.

_TURBO ON is the statement to define the beginning of the turbo block.
_TURBO OFF is to end the block.
The turbo block is the part of the program you want to execute fast.
When the entire program contains some uncompilable statements, you can define the block to be compiled using this set.

EXAMPLE

```
100 SCREEN 8:DEFINT A-Z
110 BLOAD"PICTURE",S
120 _TURBO ON
130 FOR X=0 TO 255
140 LINE(X,0)-(X,211),0
150 NEXT X
160 _TURBO OFF
170 GOTO 170
```

This program can not be "_RUN", because the "BLOAD" is one of the commands that can not be compiled. If you "RUN" this, the part lines 130 through 150 will be executed fast.

As '_RUN"FILE"' is not supported, you have to add _TURBO ON and _TURBO OFF at the beginning and the end if you want to RUN"FILE" and have the effect.

```
100 FOR I=0 TO 999
110 ..
  .
  .
890 ' END OF THE PRGRAM
```

So, this can be _RUN or add 10 _TURBO ON and 900 _TURBO OFF and RUN"FILE".

If you _RUN a program containning "_TURBO ON/OFF" it will be an error.

_TURBO ON/OFF can not be written in a multi-statement lines.

_TURBO ON/OFF can not be nested. But you may have many closed blocks in a program.

Variables and arrays are handled differently in and outside of the blocks.
Once you are out of the block, variables and arrays used in the block are lost. Only, the integer types can be defined as common.

```
100 DEFINT A-Z:DIM C(2),D(2)
110 A=1:B=2:C(0)=3:D(0)=4
120 _TURBO ON(A,C())
130 DIM D(2)
140 PRINT A,B,C(0),D(0)
150 A=5:B=6:C(0)=7:D(0)=8
160 _TURBO OFF
170 PRINT A,B,C(0),D(0)

RUN
 1 0 3 0
 5 2 7 4
Ok
```

Floating numbers used by the compiler is a special format 3-byte value.
It's accuracy is about 4.5 digits. Double precision is not available.

An array must be declared by a constant in the beginning.

This compiler works on the BASIC program on the RAM and creates the objects and variables on the left RAM. So there is a limit of the size of the source program about 10K. Big arrays, string variables (each uses 256 byte), CLEAR ???,&H???? will make the situation tighter as you can imagine. The compiled objects can not be saved as independent programs.

Interrupts availabe, such as KEY(1) ON, OFF etc. But it will decrease the efficiency of the executed object's size & speed.

Some statements may not work correctly.

```
100 GOTO 130
110 A=3/2
120 END
130 DEFINT A-Z
140 GOTO 110
```

If you RUN this, A is 1. If you _RUN this, A is 1.5. DEF??? will be effective when encountered during the execution in the case of interpreter, while it depends on the order of line number in the other case.
A little complicated string operation may cause easily a "String formula too complex" error. As this compiler has only one level of stack for it.
Break a long string formula into multiple small ones, if so.

If you _RUN an endless program, you can not stop it. Make a part to check keyboards.

```
100 GOTO 100     'Reset or power off to stop
```

```
100 IF INKEY$="" THEN 100
110 END
```
is better.

## 2. Difference from MSX-BASIC interpreter

List of statements, commands and functions that can not be compiled.

AUTO, BASE, BLOAD, BSAVE, CALL, CDBL, CINT, CLEAR, CLOAD, CLOAD?,CLOSE CONT, CSAVE, CSNG, CVD, CVI, CVS, DEFFN, DELETE, DRAW, DSKF, EOF, ERASE, ERL, ERR, ERROR, EQV, FIELD, FILES, FPOS, FRE, GET, IMP, INPUT#, KEY LIST, LFILES, LINEINPUT#, LIST, LLIST, LOAD, LOC, LOF, LPRINT USING, LSET, MAXFILES, MERGE, MOTOR, MKD$, MKI$, MKS$, NAME, NEW, ON ERROR GOTO, ON INTERVAL GOSUB(due to a bug), OPEN, PLAY, PRINT#, PRINT#USING, PRINT USING, PUT KANJI, RENUM, RESUME, RSET, SAVE, SPC, TAB, TRON, TROFF, WIDTH

List of those with limits.

CIRCLE Start, end angles and aspect ratio can't be specified.
COPY Only graphic COPY is available.
DEFDBL Same as DEFSNG.
DIM Must come first in the program or in the turbo block.
INPUT Can handle only one variable at a time.
KEY ON KEY GOSUB, KEY(n) ON/OFF only.
LOCATE x,y must be given in as a set. No cursor switch parameter.
NEXT Variable names after the NEXT can not be omitted.
ON ON STOP GOSUB, ON INTERVAL GOSUB not available.
PRINT Commas work in a different way. No wrapping for digits.
PUT PUT SPRITE only.
RUN Variables won't be initialized.
SCREEN Screen mode and sprite size only.
SET SET PAGE only.
STOP Same as END
USR Parameter type must be integer only.
VARPTR File number can not be given as the parameter.

Otherwise there is no significant difference.
In general, I/O commands & functions, and editing commands can not be compiled. Of course they are available in the direct mode, and outside of the turbo block. You can edit, debug and save a program in MSX-BASIC and execute it by _RUN.

If you want to use PRINT# to write characters on GRP:, use it outside

of turbo block. Otherwise study the sample, "PRINT.TRB".
If you want to use PLAY, use BGM compiler, and get the sound by USR(n).

## 3. New features added

3 special commands are available by starting a remark line with some specific characters.

### #I

Stands for INLINE. You can write a short machine-language routine.

```
100 DEFINT A-Z
110 K=1
120 '#I &H2A,K
130 '#I &HF3,&HCD,@150,&HFB
140 END
150 'SUB
160 RETURN
```

```
120 means   LD HL,(K)   ;K must be a simple variable of integer type.
130 means   DI
            CALL @150   ;Be careful, this line number won't be RENUMed.
            EI
```

### #C

Stands for CLIP. In the screen modes 5 through 8 (except for PAINT, and CIRCLE), this will set clipping on and off.

```
10 SCREEN 8
20 '#C-
30 LINE(0,0)-(255,255)  'Y CLIPPED
40 IF INKEY$="" THEN 40
50 '#C+
60 LINE(0,0)-(255,255)  'NOT CLIPPED
70 IF INKEY$="" THEN 70
```

### #N

Check if NEXT overflows.

```
10 FOR I%=0 TO &H7FFF
20 NEXT I%
```

This program will end up in an "Overflow error" in MSX-BASIC. And if _RUN, it will be an endless loop. If #N+ is specified, it will end normally. This code will decrease the efficiency of the object, too. Better not use this unless it's really necessary. To clear, specify #N-.

There are a lot more advice and hints, but please try and study samples in the disk. Most of them can be RUN or _RUN. Compare the execution speed.
Samples for this compiler have the extention ".TRB".