MEGA 65

Alles over de Mega65

— Het grote Mega65-boek —



MUSEUM OF ELECTRONIC GAMES & ART

MEGA65-TEAM

Paul Gardner-Stephen, prof.
(highlander)
Oprichter
Software- en virtuele hardwarearchitect
Woordvoerder en hoofdwetenschapper

Martin Streit
(seriously)
Video- en fotoproductie
Fiscaal en organisatie
Sociale media

Dan Sanderson
(dddaaannn)
Media en documentatie
MEGA65.ROM

Edilbert Kirk, dr.
(Bit Shifter)
MEGA65.ROM
Handleiding en gereedschappen

Gábor Lénárt
(LGB)
Emulator

Farai Aschwanden
(Tayger)
Bestands-hosting en gereedschappen
Financieel adviseur

Falk Rehwagen
(bluewaysw)
GEOS

Robert Steffens
(kibo)
Netwerktechnologie
Jacht op bugs in onze cores

Detlef Hastik
(deft)
Mede-oprichter
Projectaansturing
Marketing en verkoop

Oliver Graf
(lydon)
Releasemanagement
VHDL- en platformverbeteringen

Antti Lukats
(antti-brain)
Host-hardwareontwerp en -productie

Dieter Penner
(doubleflash)
Host-hardwareondersteuning

Mirko H.
(sy2002)
Aanvullende platforms en advies

Gürçe Işıkyıldız
(gurce)
Gereedschappen en verbeteringen

Daniel England
(Mew Pokémon)
Aanvullende code en gereedschappen

Hernán Di Pietro
(indiocolifa)
Aanvullende emulatie en gereedschappen

Roman Standzikowski
(FeralChild)
Open ROMs

Anton Schneider-Michallek
(adtbm)
Presentatie en ondersteuning

Fouten en onvolkomenheden melden

Dit boek wordt voortdurend verbeterd door de MEGA65-gemeenschap. De versie van deze editie is:

```
commit c598652ac7f860872b8be62911f59ad5ad752169
date: Sat Oct 24 17:02:30 2026 +0200
```

We willen dat dit boek zo goed mogelijk wordt. Als u fouten ontdekt, iets mist of graag meer informatie wilt, meld dit dan via het issue-tracker van de MEGA65 User's Guide:

https://github.com/mega65/mega65-user-guide/issues

U kunt daar ook controleren of iemand anders al een vergelijkbaar probleem heeft gemeld, terwijl u wacht op een update van dit boek.

Ten slotte kunt u altijd de nieuwste versies van onze boekenserie downloaden via:

- https://mega65.org/mega65-book
- https://mega65.org/user-guide
- https://mega65.org/developer-guide
- https://mega65.org/basic65-ref
- https://mega65.org/chipset-ref
- https://mega65.org/docs

HET GROTE MEGA65-BOEK

Uitgegeven door
het MEGA Museum of Electronic Games & Art e.V., Duitsland.

IN AANBOUW

Copyright ©2019 - 2026 Paul Gardner-Stephen, het MEGA Museum of Electronic Games & Art e.V. en medewerkers.

Dit boek is beschikbaar onder de GNU Free Documentation License v1.3 of latere versie. Dit betekent dat u dit boek vrij mag wijzigen, vermenigvuldigen en herdistribueren, onder bepaalde voorwaarden. De volledige tekst van de GNU Free Documentation License v1.3 is te vinden op https://www.gnu.org/licenses/fdl-1.3.en.html. Een Nederlandse vertaling van de licentie is te vinden op https://meta.wikimedia.org/wiki/GNU_Free_Documentation_License/nl.

Deze copyrightlicentie omvat ook de toestemming om dit document geheel of gedeeltelijk te kopiëren en/of te verspreiden voor gebruik in onderwijsomgevingen. Wij willen het onderwijs van toekomstige generaties ondersteunen, dus als u vragen of bezwaren heeft, neem dan contact met ons op.

oktober 29, 2026

Inhoudsopgave

I VOORWOORD **1**

1 Over deze vertaling **1-1**

Voorwoord bij de vertaling 1-3

Praktische zaken 1-4

Hoe deze vertaling gemaakt is 1-5

2 Inleiding **2-1**

Welkom bij de MEGA65! 2-3

Overige boeken in deze reeks 2-4

Doe mee! 2-4

II KENNISMAKEN MET UW MEGA65 **2-5**

3 Ingebruikname **3-1**

Uitpakken en aansluiten van de MEGA65 3-3

Aansluitingen aan de achterzijde 3-4

Aansluitingen aan de zijkant 3-5

Beeldscherm en randapparatuur 3-6

Optionele aansluitingen 3-8

De batterij van de real-time klok installeren 3-8

De MEGA65 inschakelen 3-11

De Intro Disk 3-13

De cursor 3-15

4 Aan de slag **4-1**

Toetsenbord 4-3

Speciale toetsen 4-3

De schermeditor 4-8

Functionaliteit van de editor 4-11

Een venster maken 4-11

Aanvullende ASCII-tekens 4-12

Hoofdletters en kleine letters 4-13
Het Freezer-menu 4-13
Commodore 64-software uitvoeren 4-14
GO64-modus 4-15
De FPGA-core "C64 for MEGA65" 4-16

5 Configureren van uw MEGA65 **5-1**
Configureren van uw MEGA65 5-3
Configuratiehulpprogramma 5-3
Input 5-4
Chipset 5-4
Video 5-6
Audio 5-6
Network 5-7
Done 5-8
Kennismaken met SD-kaarten 5-9
Een nieuwe SD-kaart voorbereiden 5-9
De SD-kaart plaatsen 5-10
Het SD-kaarthulpprogramma 5-10
De gebundelde programmatuur verkrijgen 5-13

6 De MEGA65 upgraden **6-1**
Hoe u een MEGA65 kunt upgraden 6-5
Wat is een core? 6-5
De versies van de onderdelen bepalen 6-6
De nieuwste bestanden verkrijgen 6-7
Het corekeuzemenu 6-9
De core, ROM en systeembestanden upgraden 6-10
Alternatieve cores en ROM's installeren 6-13
Core-vlaggen instellen 6-14
Een coreslot wissen 6-15

De fabriekscore in slot 0 upgraden 6-16
Het opstartproces van de core begrijpen 6-18

7 Werken met schijven en schijfimages 7-1
Schijfstations 7-3
Unitnummers en drivenummers 7-3
Werken met virtuele schijfimages 7-4
Waar u schijfimagebestanden kunt verkrijgen 7-4
Schijfimages aankoppelen met de Freezer 7-4
Schijfimages aankoppelen vanuit BASIC 7-6
Een nieuw schijfimage aanmaken 7-6
SD-kaartbestanden beheren in submappen 7-6
Het ingebouwde 3.5″-diskettestation gebruiken 7-7
Het ingebouwde 3.5″-station aankoppelen met de Freezer 7-7
Het ingebouwde 3.5″-station aankoppelen vanuit BASIC 7-8
DD- en HD-schijven 7-8
Een schijf formatteren 7-9
Externe IEC-schijfstations gebruiken 7-10
Opstartbare schijven 7-10
Schijven automatisch opstarten 7-11
De SD-kaart benaderen vanuit BASIC 7-11
Gangbare schijfbewerkingen 7-12
DIR 7-13
DLOAD en RUN 7-13
DSAVE 7-13
BACKUP 7-14
COPY 7-14
RENAME 7-14
DELETE 7-14
Snelkoppelingen voor schijfcommando's 7-14

8 Bestanden overzetten **8-1**

Bestanden naar de MEGA65 overbrengen 8-3

Netwerken begrijpen 8-3

M65Connect verkrijgen 8-4

M65Connect voor Windows 8-4

M65Connect voor macOS 8-5

M65Connect voor Linux 8-5

Netwerkluisteren inschakelen 8-5

Bestanden overzetten 8-7

Bestanden overzetten met M65Connect 8-8

Het commandoregelhulpmiddel mega65_ftp 8-10

III EERSTE STAPPEN IN HET PROGRAMMEREN **8-13**

9 Hoe computers werken **9-1**

Computers zijn dom. Echt dom 9-3

Een eierdopjescomputer maken 9-3

10 Aan de slag met BASIC **10-1**

Uw eerste BASIC-programma's 10-3

Oefeningen om te proberen 10-12

Eerste stappen met tekst en getallen 10-13

Oefeningen om te proberen 10-29

Eenvoudige beslissingen nemen 10-30

Oefeningen 10-43

Willekeurige getallen en toeval 10-44

Oefeningen om te proberen 10-48

11 Tekstverwerking **11-1**

Tekens en tekenreeksen 11-3
Tekenreeksconstanten 11-5
Tekenreeksvariabelen 11-5
Tekenreeksfuncties 11-7
Eenvoudige opmaak 11-8
Nieuwe regels onderdrukken 11-8
Automatische tabstops 11-8
Tabstops en spatiëring 11-8
Voorbeeldprogramma's 11-9
Palindromen 11-9
Eenvoudige cijfers 11-9

12 C64-, C65- en MEGA65-modi **12-1**

Modi wisselen vanuit BASIC 12-3
Van MEGA65/C65 naar de C64-modus 12-3
Van C64 naar de MEGA65/C65-modus 12-4
De machinetaalmonitormodus openen 12-4
Het KEY-register 12-4
Extra C65-registers zichtbaar maken 12-5
Extra C65/MEGA65-registers uitschakelen 12-6
Extra MEGA65-registers inschakelen 12-6
Valkuilen 12-7
Geheugen benaderen vanuit BASIC 65 12-7
De MAP-instructie 12-8

IV GELUID EN GRAFIEK **12-9**

13 Grafiek **13-1**

V CPU EN GEHEUGEN 13-13

14 Kennismaken met de 45GS02-CPU 14-1

Werken met de monitor 14-5
CPU-instructies 14-5
CPU-registers 14-5
De Base-Page 14-6
Indirecte adressering 14-6
Het virtuele 32-bits register 14-6
De stapel 14-6
Interrupts 14-6
Instructies coderen 14-7
De 6502-familie 14-7

15 Programmeren met het geheugen 15-1

Geheugenbegrippen 15-4
Bits, bytes en nibbles 15-4
Hexadecimale notatie 15-4
Geheugen met willekeurige toegang 15-5
Alleen-lezen geheugen 15-6
I/O-registers 15-6
Adressen 15-7
16-bits adresvertaling 15-7
Banken en pagina's 15-8
Hoe adressen in het geheugen worden opgeslagen 15-8
De 28-bits adresruimte 15-11
De geheugenindeling van het chip-RAM 15-13
Hoe programma's het chip-RAM gebruiken 15-14
De geheugenindeling als contract 15-15
Geheugen gebruiken vanuit BASIC 15-17
Hermapping van adressen door BASIC 15-17

BANK en SYS 15-19
SYS TO 15-20
Geheugen gebruiken vanuit machinetaal 15-22
Het MAP-register 15-23
Het MAP-register bekijken 15-24
MAP en interrupts 15-25
28-bits adressen in machinetaal gebruiken 15-26
Videogeheugen van de VIC-IV 15-28
Schermgeheugen vinden 15-28
Tekengegevens vinden 15-29
Spritegeheugen vinden 15-30
Kleurengeheugen 15-31
Grote geheugenbewerkingen met Direct Memory Access 15-32
DMA-taken 15-32
DMA gebruiken vanuit BASIC 15-33
DMA gebruiken vanuit machinetaal 15-33
Een Enhanced-taaklijst in het geheugen aanroepen 15-34
Een Enhanced-taaklijst inline in de code aanroepen 15-35
Het formaat van de Enhanced-taaklijst 15-36
Gevorderde geheugenonderwerpen 15-40
De geheugenmapping testen met de Matrix Mode-debugger 15-40
Geheugenbanking in C64-stijl 15-40
Het bankingregister $D030 van de VIC-III 15-41
Cartridge-ROM 15-42
I/O-persoonlijkheden 15-43
ROM omzetten in RAM 15-44
MAP gebruiken om geheugen op hogere adressen te benaderen 15-45
Het MAP-register lezen 15-46
Al het chip-RAM beschikbaar maken 15-47
Waarom het SYS-commando slechts bepaalde adressen kan gebruiken 15-48

Hoe SYS TO werkt 15-49

VI HARDWARE 15-51

16 Nexys4-borden gebruiken als een MEGA65 16-1

Uw eigen MEGA65-compatibele computer bouwen 16-5
Werkende Nexys4-borden 16-6
Het Nexys4-bord 16-6
Het Nexys4DDR-bord 16-6
De Nexys A7 16-7
Stroom, jumpers, schakelaars en knoppen 16-8
Micro-USB-voeding 16-9
Externe voeding 16-9
Overige jumpers en schakelaars 16-10
Aansluitingen en randapparatuur 16-11
Communicatie met uw pc 16-11
Knoppen op het bord 16-12
Toetsenbord 16-12
Enkele toetstoewijzingen met een USB-toetsenbord 16-13
MicroSDHC-kaart voorbereiden 16-15
Voorbereidingsstappen 16-15
De bitstream laden vanuit QSPI 16-17
Voorbereidingsstappen 16-17
Widget-bord 16-17
PMOD-naar-joystick-adapter 16-20
Handige tips 16-21

VII PLATFORMONAFHANKELIJKE ONTWIKKELHULPMIDDELEN 16-23

17 Emulators 17-1

De Xmega65-emulator gebruiken 17-3

Het toetsenbord 17-3
Problemen met starten op Apple MacOS 17-4
Instellingen bijwerken 17-5
De Live ISO-image gebruiken 17-5
Een opstartbare USB-stick of DVD maken 17-5
Aan de slag 17-6
Andere functies van de Live ISO 17-7

18 Gegevensoverdracht en foutopsporing 18-1
m65-hulpprogramma 18-3
Schermfoto's met het m65-hulpprogramma 18-3
Een programma laden en uitvoeren op de MEGA65 18-4
De FPGA herconfigureren om een andere bitstream uit te voeren . . . 18-4
Toetsenbordinvoer op afstand 18-4
Unit-testing en logboekondersteuning 18-5
Unit-tests gebruiken met C 18-5
Unit-tests gebruiken met BASIC 65 18-6
BASIC 65-voorbeeld 18-7
M65Connect 18-8
mega65_ftp 18-9
TFTP-server 18-9
Een BASIC-tekstbestandlisting omzetten naar een PRG-bestand 18-10

19 Assemblers 19-1

20 C- en C-achtige compilers 20-1
MEGA65 libc 20-3

21 MEGA65 Standard C Library 21-1
Structuur en gebruik 21-3
conio.h 21-3
conionit 21-3

setscreenaddr 21-3
getscreenaddr 21-4
setcolramoffset 21-4
getcolramoffset 21-4
setcharsetaddr 21-4
getcharsetaddr 21-5
clrscr 21-5
getscreensize 21-5
setscreensize 21-5
set16bitcharmode 21-6
sethotregs 21-6
setextendedattrib 21-6
togglecase 21-6
togglecase 21-6
togglecase 21-7
bordercolor 21-7
bgcolor 21-7
textcolor 21-7
revers 21-7
highlight 21-8
blink 21-8
underline 21-8
altpal 21-8
clearattr 21-9
cellcolor 21-9
setpalbank 21-9
setpalbanka 21-9
getpalbank 21-10
getpalbanka 21-10
setmapedpal 21-10

getmapedpal 21-10
setpalentry 21-11
fillrect 21-11
box 21-11
hline 21-12
vline 21-12
gohome 21-13
gotoxy 21-13
gotox 21-13
gotoy 21-13
moveup 21-14
movedown 21-14
moveleft 21-14
moveright 21-14
wherex 21-15
wherey 21-15
pcputc 21-15
pcputsxy 21-15
cputcxy 21-15
pcputs 21-16
cputc 21-16
cputnc 21-16
cputhex 21-16
cputdec 21-17
cputs 21-17
cputsxy 21-17
cputcxy 21-18
cputncxy 21-18
cprintf 21-18
pcprintf 21-19

cgetc 21-19
kbhit 21-19
getkeymodstate 21-20
flushkeybuf 21-20
cinput 21-20
VIC_BASE 21-21

22 BASIC-tokenisatoren **22-1**

VIII BIJLAGEN 22-5

A Accessoires **A-1**

B Naslag BASIC65-commando's **B-1**
Commando's, functies en operatoren B-3
Alfabetisch overzicht BASIC-commando's B-17

C Schermcodes **C-1**
Schermcodes C-3

D PETSCII-codes **D-1**
PETSCII-codes D-3

E Toetsen van de schermeditor **E-1**
Toetsen van de schermeditor E-3
Control-codes E-3
Shiftcodes E-6
Escape-sequenties E-6

F Het toetsenbord van de MEGA65 **F-1**
De typgebeurteniswachtrij gebruiken F-4
ASCII-typgebeurtenissen interpreteren F-5
De keymap voor Unicode Basic-Latin F-10

Werking van het toetsenbord F-15
C65-toetsenbordmatrix F-15
Synthetische toetsgebeurtenissen F-17
Besturing van de toetsenbord-LED's F-18
Native toetsenbordmatrix F-20

G Systeempalet **G-1**
Systeempalet G-3

H Decimaal, binair en hexadecimaal **H-1**
Getallen H-3
Notaties en grondtallen H-4
Decimaal H-6
Binair H-7
Hexadecimaal H-9
Bewerkingen H-10
Tellen H-11
Rekenkunde H-12
Logische poorten H-14
Gehele en natuurlijke getallen H-16
Bitgewijze logische operatoren H-17
Getallen omrekenen H-19

I KERNAL-sprongtabel **I-1**
Gebruik van de KERNAL-sprongtabel I-3
Randvoorwaarden voor sprongtabelroutines I-3
Geschiedenis en compatibiliteit I-4
Snelnaslag van de MEGA65-KERNAL-sprongtabel I-5
Naslagwerk van de MEGA65-KERNAL-sprongtabel I-7
ACPTR I-8
ADDKEY I-9
BASIN I-10

BOOT_SYS I-12
BSOUT I-13
C64MODE I-14
CHKIN I-15
CINT I-16
CIOUT I-17
CKOUT I-18
CLALL I-19
CLOSE I-20
CLOSE_ALL I-22
CLRCH I-23
CMP_FAR I-24
CURSOR I-25
GETIN I-26
GETIO I-27
GETLFS I-28
IOBASE I-29
IOINIT I-30
JMPFAR I-31
JSRFAR I-32
KEY I-34
LDA_FAR I-35
LISTN I-36
LKUPLA I-37
LKUPSA I-38
LOAD I-39
MEMBOT I-41
MEMTOP I-42
MONEXIT I-43
MonitorCall I-44

OPEN I-45
PFKEY I-49
PHOENIX I-51
PLOT I-52
PRIMM I-54
RAMTAS I-55
RDTIM I-56
READSS I-58
RESET_RUN I-59
RESTOR I-60
SAVE I-61
SAVEFL I-64
ScanStopKey I-65
SCRORG I-66
SECND I-67
SETBNK I-68
SETLFS I-70
SETMSG I-71
SETNAM I-72
SETTIM I-73
SPIN_SPOUT I-74
STA_FAR I-75
STOP I-76
SWAPPER I-77
SYSFLAGS I-78
TALK I-79
TKSA I-80
UNLSN I-81
UNTLK I-82
VECTOR I-83

VERSIONQ I-89

J Systeemgeheugenindeling J-1

Inleiding J-3

Eigen geheugenindeling van de MEGA65 . . . . . J-4

De eerste zestien 64KB-banken J-4

Kleurengeheugen J-5

Extra RAM J-5

28-bits adresruimte J-5

$D000 - $DFFF I/O-persoonlijkheden J-7

Banking van het CPU-geheugen J-10

C64/C65-ROM-emulatie J-11

Indeling van de C65-compatibiliteits-ROM J-12

K 45GS02-microprocessor K-1

Inleiding K-3

Verschillen ten opzichte van de 6502 . . . . . K-3

Supervisor-/hypervisormodus K-3

Niet-bedoelde 6502-instructies K-4

Bugcompatibiliteit van Read-Modify-Write-instructies K-4

Variabele CPU-snelheid K-5

Langzame werking (1 MHz - 3.5 MHz) K-5

Instructietiming op volle snelheid (40 MHz) K-5

Direct Memory Access (DMA) K-6

Geheugen benaderen tussen 64 KB en 1 MB K-6

Geheugenbanking in C64-stijl K-6

VIC-III "ROM"-banking K-6

Display Address Translator van de VIC-III K-7

De MAP-instructie K-7

DMA-controller K-9

Platte geheugentoegang K-10

Toegang tot geheugen buiten het 1MB-punt K-10
De MAP-instructie gebruiken om >1MB te benaderen K-10
Platte geheugentoegang K-12
Virtueel 32-bit-register K-13
In het geheugen gemapte registers van de C64-CPU K-15
Nieuwe in het geheugen gemapte CPU-registers K-16
MEGA65 CPU-versnellingsregisters voor wiskunde K-18
MEGA65-hypervisormodus K-23
Reset K-23
De hypervisormodus in- en uitstappen K-24
Hypervisor-geheugenindeling K-25
Hypervisor-virtualisatieregisters K-27
Programmeren in de hypervisormodus K-31

L 45GS02 & 6502-instructiesets L-1

Inleiding L-3
Stapelbewerkingen L-3
Adresseermethoden L-4
Impliciet L-4
Accumulator L-4
Het pseudoregister Q L-4
Immediate-modus L-4
Immediate Word-modus L-5
Base-Page-modus L-5
Base-Page Quad-modus L-5
Base-Page X-Indexed-modus L-6
Base-Page Quad X-Indexed-modus L-6
Base-Page Y-Indexed-modus L-7
Absolute modus L-7
Absolute Quad-modus L-7
Absolute X-Indexed-modus L-8

Absolute Quad X-Indexed-modus L-8
Absolute Y-Indexed-modus L-8
Absolute Indirect-modus L-9
Absolute Indirect X-Indexed-modus L-9
Base-Page Indirect X-Indexed-modus L-9
Base-Page indirect Y-geïndexeerde modus L-10
Base-Page indirect Z-geïndexeerde modus L-10
Base-Page Quad indirect Z-geïndexeerde modus L-10
32-bits Base-Page-indirecte Z-geïndexeerde modus L-11
32-bits Base-Page-(Zero-Page)-indirecte quad Z-geïndexeerde modus L-11
Stapelrelatieve indirecte Y-geïndexeerde modus L-12
Relatieve adresseermethode L-12
Relatieve word-adresseermethode L-13
6502-instructieset L-14
Officiële en onbedoelde instructies L-14
Opcodetabel L-14
De instructieset van de 4510 . . . . . L-56
Instructietiming L-56
Opcodetabel L-56
Samengestelde instructies van de 45GS02 . . . . . L-113

M Ontwikkeling van systeemprogramma's M-1

Inleiding M-5
Flashmenu M-5
Format/FDISK-hulpprogramma M-6
Toetsenbordtest-hulpprogramma M-7
MEGA65-configuratiehulpprogramma M-7
Freezer-menu M-7
Hulpprogramma's van het Freezer-menu M-8
Hypervisor M-8
OpenROM M-8

N MEGA65 Hyppo-diensten **N-1**

Inleiding N-3

Hyppo-diensten: Beknopt overzicht N-7

Algemene diensten N-11

Station-/opslagservices N-15

Schijfimage-services N-44

Taak- en procesdiensten N-48

Systempartitieservices N-64

Freezerservices N-65

O Machinetaalmonitor **O-1**

Inleiding O-3

De machinetaalmonitor van de MEGA65 . . . . . O-3

Getallen O-4

De assembler O-4

Verschillen met de C65-monitor O-5

Tabel met MEGA65-monitorcommando's O-6

A : ASSEMBLE O-6

B : BITMAPS O-8

C : COMPARE O-9

D : DISASSEMBLE O-9

F : FILL O-10

G : GO O-11

H : HUNT O-11

J : JUMP O-11

L : LOAD O-12

M : MEMORY O-12

R : REGISTERS O-13

S : SAVE O-13

T : TRANSFER O-14

V : VERIFY O-14

X : EXIT O-14
. : ASSEMBLE O-15
> : MODIFY MEMORY O-15
; : MODIFY REGISTERS O-15
@ : DISK COMMAND O-15
De Matrix Mode/seriële monitor O-16
Tabel met Matrix Mode-monitorcommando's O-18
De monitor aanroepen O-18
<Enter> : Single-step O-18
: Hypervisor-trap in-/uitschakelen O-19
+ : Stel UART-baudratedeler seriële interface in O-19
@ : CPUMEMORY O-20
? of H : HELP O-20
B : BREAKPOINT O-20
D : DISASSEMBLE O-21
E : FLAGWATCH O-21
F : FILL O-21
G : SETPC O-22
I : INTERRUPTS O-22
J : DEBUGMON O-22
L : LOADMEMORY O-23
M : MEMORY O-23
R : REGISTERS O-23
S : SETMEMORY O-24
T : TRACE O-24
W : WATCHPOINT O-24
Z : CPUHISTORY O-25

P F018-compatibele controller voor Direct Memory Access (DMA) P-1

F018A/B-DMA-taken P-3

Indeling van de F018-DMA-taaklijst P-4

Structuur van de 11-byte F018-DMA-lijst P-5

Structuur van de 12-byte F018B-DMA-lijst P-5

Eenvoudige DMA-bewerkingen uitvoeren P-6

MEGA65 Enhanced DMA Jobs P-11

Textuurschaling en lijntekening P-14

Inline-DMA-lijsten P-16

Audio-DMA P-17

Adresbeheer voor samples P-17

Afspeelfrequentie en volume van samples P-18

Zuivere sinusgolf P-18

Regeling van het afspelen van samples P-19

F018 "DMAgic" DMA-controller P-19

Uitbreidingen van de MEGA65-DMA-controller P-20

Niet-geïmplementeerde functionaliteit P-24

Q VIC-IV Video Interface Controller Q-1

Kenmerken Q-5

Toegang tot VIC-III/IV-registers Q-6

VIC-II/III/IV detecteren Q-7

Video-uitvoerformaten, timing en compatibiliteit Q-9

Integrated Marvellous Digital Hookup™ (IMDH™) Digitale video-uitvoer Q-9

Aansluiten op eigenzinnige propriëtaire digitale videostandaarden Q-9

Frametiming Q-10

Fysieke en logische rasters Q-13

Bad lines Q-13

Geheugeninterface Q-14

Schermgeheugen lokaliseren en verplaatsen Q-14

Tekengeneratorgegevens verplaatsen Q-15
Het kleuren-/attribuutgeheugen verplaatsen Q-15
Spritepointers en sprite-afbeeldingen verplaatsen Q-16
Hete registers Q-17
Nieuwe modi Q-20
Waarom de nieuwe VIC-IV-modi teken- en bitmapmodi zijn en geen bitplanemodi Q-20
Weergeven van meer dan 256 unieke tekens via "Super-Uitgebreide Attribuutmodus" Q-21
Bitvelden indien GOTOX_flag is gewist: Q-24
Bitvelden indien GOTOX_flag is ingesteld: Q-25
Super-Uitgebreide Attribuutmodus gebruiken Q-27
Volkleur-tekstmodus (256 kleuren per teken) (FCM) Q-31
Nibble-kleuren-tekstmodus (16 kleuren per teken) (NCM) Q-31
Alphablending / Anti-aliasing Q-32
Tekens omdraaien Q-32
Variabele-breedte lettertypen Q-32
Raster Re-write Buffer Q-33
Sprites Q-34
VIC-II/III-spritebesturing Q-34
Uitgebreide sprite-afbeeldingssets Q-34
Variabele spritegrootte Q-35
Variabele sprite-resolutie Q-35
Sprite-paletbank Q-36
Vollekleuren-spritemodus Q-36
VIC-III errata-niveau Q-39
VIC-III errata-niveaus Q-39
VIC-II / C64-registers Q-40
VIC-III / C65-registers Q-43
VIC-IV / MEGA65-specifieke registers Q-45

R Sound Interface Device (SID) R-1

SID-registers R-3

S 6526 Complex Interface Adapter (CIA)-registers S-1

CIA 6526-registers S-3

CIA 6526-hypervisorregisters S-6

T 4551 UART, GPIO en utiliteitscontroller T-1

C65 6551 UART-registers T-3

Algemene I/O & diverse interface-registers van de 4551 . . . . . T-4

U 45E100 Fast Ethernet-controller U-1

Overzicht U-3

Verschillen met de RR-NET en vergelijkbare oplossingen U-3

Werking: het ontvangen van frames U-4

Toegang tot de Ethernet-framebuffers U-6

Werking: het verzenden van frames U-7

Geavanceerde functies U-7

Broadcast- en multicast-verkeer en promiscue modus U-7

Foutopsporings- en diagnosemogelijkheden U-8

In het geheugen gemapte registers U-8

Waarden van het COMMAND-register U-10

Voorbeeldprogramma's U-11

V 45IO27 multifunctionele I/O-controller V-1

Overzicht V-3

F011-compatibele schijfstationcontroller V-3

Ondersteuning voor meerdere schijfstations V-3

Bewerkingen met gebufferde sectoren V-4

Sectoren van een schijf lezen V-5

Automatische spoorcorrectie V-5

Sector-scheefstand en de Target Any-modus V-6

Schijfindeling en logische 1581-sectoren V-6
FD2000-diskettes V-7
HD-diskettes en diskettes met variabele dichtheid V-7
Track Information Blocks V-8
Schijven formatteren V-9
Schrijfvoorcompensatie V-10
Gebufferd sectorschrijven V-10
Floppy-track-DMA V-11
Floppy-track-DMA gebruiken V-11
De beperkingen van schijfstations begrijpen V-12
F011-registers van de schijfstationcontroller V-13
SD-kaartcontroller en F011-virtualisatiefuncties V-15
Toegang tot schijfimages op basis van de SD-kaart V-15
F011-virtualisatie V-17
Dual-bus SD-kaartcontroller V-18
Write-gate V-19
Vulmodus V-19
Selectie tussen meerdere SD-kaarten V-19
Commandotabel van de SD-controller V-19
Touchpadinterface V-22
Ondersteunende functies voor audio V-24
Overige audiofuncties V-26
Mixer-feedbackregisters V-26
8/16-bits stereo digitale audio V-26
Pulsbreedte versus pulsdichtheidsmodulatie V-27
Diverse I/O-functies V-28

W 4541 seriële buscontroller **W-1**

Overzicht W-3
Mogelijkheden van de 4541 . . . . . W-3
Ondersteuning van versnelde seriële protocolvarianten W-3

Processorontlasting via interruptondersteuning W-3
Onafhankelijkheid van processorsnelheid W-3
Coëxistentie via open-collectorlogica W-3
Werkingsprincipe W-4
Voorbeelden W-5
De status van het DOS-kanaal lezen W-5
Commandoreferentie W-7
Registertabel W-11
Seriële bustiming W-12
Een byte onder Attention verzenden W-12
JiffyDOS™-protocolaanvraag W-15
Verzenden van controller naar apparaat onder JiffyDOS™ W-17
Ontvangst van randapparaat naar controller onder JiffyDOS™ W-19
Omschakelen van spreker naar luisteraar W-21
Byte verzenden met einde-aanduiding (EOI) W-23
Byte ontvangen W-26
Optionele geïntegreerde datalogger W-26
Gegevens uit de datalogger halen W-28

X Registers van de real-time klok (RTC) X-1
Algemene RTC-registers (alle revisies) X-3
Uitgebreide RTC-registers (moederbord R4 en nieuwer) X-4
Registers op de Grove-RTC-vervangingsunit X-8

Y Speciale registers Y-1
UUID64 . . . . . Y-3

Z 5417E Spreadsheet Hardware Integrated Technology Engine Z-1
Overzicht Z-3
Architectuur Z-3
Niet-geïmplementeerde functionaliteit Z-4

AA Naslagtabellen **AA-1**

Opslageenheden AA-3

Getalstelselomrekening AA-4

AB Het flashen van de FPGA's en CPLD's in de MEGA65 **AB-1**

Aanbevolen specificaties voor de pc AB-3

Waarschuwing AB-4

Vivado installeren AB-4

De FTDI-stuurprogramma's installeren AB-20

Linux-drivers AB-20

Windows-drivers AB-20

De hoofd-FPGA flashen met Vivado AB-24

De CPLD in het toetsenbord van de MEGA65 flashen met Lattice Diamond . AB-37

Het flashen van de MAX10-FPGA op het MEGA65-moederbord met INTEL QUARTUS AB-44

AC Problemen oplossen **AC-1**

Hardware AC-3

Geen digitale video-uitvoer AC-3

MEGA65 R4, R5, R6 . . . . . AC-3

Geen lampjes bij het inschakelen AC-3

Vivado AC-4

RAM-vereisten AC-4

mega65_ftp AC-5

Ontbrekende bibliotheek AC-5

AD Modelspecifieke kenmerken **AD-1**

MEGA65-modellen detecteren AD-3

MEGA65-desktopcomputer, revisies 5 en 6 . . . . . AD-3

MEGA65-desktopcomputer, revisie 4 . . . . . AD-4

MEGA65-desktopcomputer, revisie 3 & revisie 3A AD-5

MEGA65-desktopcomputer, revisie 2 . . . . . AD-6

MEGAphone-handheld, revisies 1 en 2 . . . . . AD-7
Nexys4 DDR FPGA-bord AD-7

AE Elektronisch schema **AE-1**
MEGA65 R6 elektronisch schema AE-3
MEGA65 R3 elektronisch schema AE-27
MEGA65 R2 elektronisch schema AE-52
Nexys Widget Board elektronisch schema AE-75

AF Steuners & donateurs **AF-1**
Organisaties AF-3
Bijdragers AF-4
Steuners AF-5

Index **Index-3**

DEEL I

VOORWOORD

HOOFDSTUK 1

Over deze vertaling

- Voorwoord bij de vertaling
- Praktische zaken
- Hoe deze vertaling gemaakt is

VOORWOORD BIJ DE VERTALING

De Mega65 is een klassieke homecomputer zoals ze waren in het gouden decennium van de homecomputer, de jaren '80. Weliswaar heeft de Mega65 veel krachtiger hardware dan de computers van destijds, de filosofie achter de computer probeert de filosofie achter de klassieke homecomputers zo getrouw mogelijk na te leven. Eén van de elementen waaruit dat blijkt is de handleiding. Computers uit de jaren '80 werden geleverd met een informatieve handleiding. Een handleiding die begon bij het prille begin, maar de lezer niet voor dom hield, maar hem opleidde tot een volwaardige programmeur.

De handleiding van de Mega65 doet precies dat: De handleiding begint bij het begin, legt uit hoe de computer gebruikt moet worden, en begint daarna met de beginselen van het programmeren in BASIC. Er is evenwel nog een niet on-belangrijk detail: De computers van de jaren '80 hadden niet alleen een informatieve handleiding, zij hadden een informatieve handleiding in alle talen. Dat deed ertoe. Een tiener die in jaren '80 zijn eerste computer kreeg, was vaak onkundig om een volwaardige handleiding in het Engels te kunnen lezen. Juist doordat iedere bezitter van een homecomputer een handleiding had in zijn eigen taal, leerden ontzettend veel van hen te programmeren. De homecomputers uit onze jonge jaren brachten ons niet uitsluitend amusement, zij hebben ons gevormd tot kundige vakmensen, waardoor we de wereld konden bouwen zoals deze nu is.

De Mega65 is niet uitsluitend bedoeld voor degene die de jaren '80 actief heeft meegemaakt. De Mega65 is ook niet uitsluitend bedoeld voor computerexperts, nee, de Mega65 wil de fascinerende tijdsgeest van computers in de jaren '80 aan iedereen laten zien die in computers geïnteresseerd is. Dat maakt de taal van de handleiding een relevant onderwerp.

Het Engels is een taal met een enorme bibliotheek aan kennis en omdat de grammatica relatief eenvoudig is, spreken veel mensen het. Dat maakt het een logische taal om documentatie in te publiceren. Toch heeft deze taal ook nadelen. Juist doordat het Engels een eenvoudige grammatica heeft, bevatten zinnen minder informatie en kunnen begrippen zelfs dubbelzinnig opgevat worden. Dat maakt vertaling vanuit het Engels uitdagend: Wordt een tekst te letterlijk vertaald, dan ontstaat een onnatuurlijk "vertaalnederlands" dat heel anders is dan het Nederlands dat we dagelijks spreken. Vertaling vanuit het Engels is een heel creatief proces, waarbij de vertaler zaken als verkleinwoorden en positiewerkwoorden creatief moet toevoegen.

Er bestaat het risico dat de vertaler een dubbelzinnig begrip verkeerd interpreteert en daardoor de lezer wordt misleid. Een mooi voorbeeld is het begrip "serial pherpheral bus", dat door automatische vertaalhulpmiddelen aanvankelijk vertaald werd als"seriële perife bus". Deze problemen ontstaan doordat de Engelse grammatica geen onderscheid kan maken tussen een bijvoeglijk naamwoord en een zelfstandig naamwoord, waardoor de vertaler de mist in kan gaan. Als dit soort fouten niet worden opgemerkt, moet de lezer in dit soort situaties terugredeneren naar het Engels en concluderen dat "seriële periferierbus" bedoeld werd, of in nog beter Nederlands,

"seriële randapparatuurbus". Er ligt dus een belangrijke verantwoordelijkheid op de vertaler om goed werk af te leveren, anders mist de vertaling zijn doel.

Toch is dit wellicht het beste argument dat pleit om technische documenten te vertalen: Immers, als een vertaler dit soort interpretatiefouten kan maken, kan de lezer die fouten ook maken. De lezer zal in praktijk, als hij de materie beter begint te begrijpen, uiteindelijk vaak alsnog op de juiste betekenis van het woord komen. Desondanks maakt dit het Nederlands een zeer fijne taal om technische materie heel precies in uit te leggen. Het Nederlands is, bijvoorbeeld dankzij de krachtige woordconstuctiemechanismen, een taal waarin zinnen iets met heel hoge precisie kunnen formuleren, waardoor minder interpretatie aan de lezer overgelaten wordt. De tekst wordt daardoor toegankelijker en makkelijker begrijpbaar.

Met die insteek heb ik deze vertaling gemaakt. Deze vertaling is geen vertaling die woord-voor-woord het origineel volgt. In veel gevallen is dat niet eens zin-voor-zin. Bij het vertalen stond de boodschap van een tekstfragment te allen tijde voorop en de boodschap is opnieuw geformuleerd zoals dit in het Nederlands uitgelegd zou worden, creatief gebruik makend van de mogelijkheden die onze taal heeft.

Ik hoop dat met deze handleiding u nog meer plezierige en interessante momenten met uw Mega65 zult hebben.

– **Daniël Mantione**, oktober 2026.

PRAKTISCHE ZAKEN

Een bezitter van een MEGA65 is het volste respect waard, dus we zullen u in dit boek met "u" aanspreken.

Deze vertaling is geschreven volgens de Witte Spelling, waarbij de klassieke tussen-n-regel "n = meervoud" gebruikt wordt. Bij woorden waarbij zowel een n als een s voor het meervoud gebruikt kan worden, verkiezen we de n.

De Mega65-computer zelf spreekt geen Nederlands en gebruikt de Engelse getalnotatie met de punt (.) als decimaal scheidingsteken. In de handleiding halen we regelmatig fragmenten aan van programma's of andere tekst die ingetypt wordt aan. Het zou zeer verwarrend zijn als we voortdurend tussen de punt en komma als decimaal scheidingsteken zouden wisselen, daarom gebruiken we, in afwijking van de Nederlandse spelling, de punt als decimaal scheidingsteken. Om verwarring te voorkomen gebruiken we geen scheidingsteken voor duizendtallen.

Voor de verschillende soorten getallen gebruiken we de termen zoals u ze in de wiskundeles geleerd heeft:

1. Het begrip *unsigned integer* vertalen we met **natuurlijk getal**.
2. Het begrip *signed integer* vertalen we met **geheel getal**.
3. Het begrip *real number* vertalen we met **reëel getal**.

In delen van de documentatie die toegespitst zijn op beginners zullen we *integer* aanvankelijk vermijden en ook zullen we een *string* een **tekenreeks** noemen. In delen van de documentatie die toegespitst zijn op meer ervaren programmeurs hebben we wat minder moeite met deze termen, ze zijn immers goed ingeburgerd in het informaticajargon.

HOE DEZE VERTALING GEMAAKT IS

Het omvattende boek "Alles over de Mega65" beslaat inmiddels meer dan 1500 bladzijden. Laten we er geen doekjes om winden: Dat is niet allemaal met de hand vertaald. Deze handleiding is gemaakt met een combinatie van zowel handmatige, programmatische, als automatische vertaling. Delen met van de handleiding met veel afbeeldingen en codevoorbeelden zijn met de hand vertaald, waaronder de afbeeldingen en codevoorbeelden zelf.

Voor de naslaghoofdstukken, zoals bijvoorbeeld de BASIC65-naslag en registertabellen, geldt dat veel termen voor iedere behandeld commando terugkomen. Sommige zinnen komen ettelijke malen terug. Hiervoor zijn programma's geschreven, die de vertaalde hoofdstukken vanuit tabellen opbouwen, zodat elke term in het gehele boek consistent wordt gehanteerd.

Voor hoofdstukken met grote hoeveelheden "droge tekst" is gebruikt gemaakt van automatische vertaling door middel van kunstmatige intelligentie. Daartoe is eerst een stijlboek opgesteld en daarna is de tekst vertaald door een taalmodel. Daarna is met een tweede taalmodel gecontroleerd of de tekst aan kwaliteitseisen voldoet op het gebied van stijlboek, spelling en of de tekst als natuurlijk Nederlands leest. Alle automatisch vertaalde tekst is nog eens met de hand nagelopen, gecorrigeerd en geherformuleerd.

HOOFDSTUK 2

Inleiding

- Welkom bij de MEGA65!
- Overige boeken in deze reeks
- Doe mee!

WELKOM BIJ DE MEGA65!

Van harte gefeliciteerd met de aanschaf van één van de meest langverwachte computers uit de geschiedenis van het computergebruik! De MEGA65 is door de community ontworpen en is gebaseerd op de nooit uitgebrachte Commodore® 65[1]-computer; een computer die in 1989 werd ontworpen en in 1990 voor het publiek verkrijgbaar zou worden. Inmiddels zijn er decennia verstreken en wij hebben ons ingespannen om herinneringen op te roepen aan een vroegere tijd waarin computers eenvoudig en vriendelijk waren. Zij waren niet alleen eenvoudig te bedienen en te begrijpen, doch ook vriendelijk en toegankelijk voor nieuwe gebruikers.

Deze computers uit de jaren tachtig inspireerden veel van hun eigenaren tot de opwindende en dankbare technologiecarrières die zij vandaag de dag hebben. Stelt u zich eens de opwinding voor die deze computerpioniers beleefden, toen zij ontdekten dat zij hun nieuwe computer konden gebruiken om problemen op te lossen, een brief te schrijven, belastingaangifte te doen, nieuwe dingen uit te vinden, te ontdekken hoe het heelal werkt en misschien zelfs een spannend spelletje of twee te spelen! Wij willen datzelfde niveau van opwinding weer aanwakkeren (dat helaas in het moderne computergebruik niet meer te vinden is) en daarom hebben wij de **MEGA65** gemaakt.

Het MEGA65-team is van mening dat het bezitten van een computer vergelijkbaar is met het bezitten van een huis. U gebruikt een huis niet alleen maar; u verandert dingen, groot en klein, om er uw eigen persoonlijke leefruimte van te maken. Na verloop van tijd, wanneer u zich er eenmaal thuis voelt, kunt u besluiten uw huis te verbouwen of uit te breiden om het comfortabeler of praktischer te maken. Beschouw de MEGA65 als uw eigen "computerthuis".

Deze handleiding leert u meer dan alleen plaatjes aan de muur te hangen; hij laat u zien hoe u uw droomhuis bouwt. Terwijl u deze handleiding leest, leert u de MEGA65 bedienen, programma's schrijven, extra programmatuur toevoegen en de mogelijkheden van de hardware uitbreiden. Wat niet meteen duidelijk zal zijn, is dat u onderweg ook meer leert over de geschiedenis van het computergebruik, terwijl u de vele facetten van BASIC-versie 65 en de commando's van het besturingssysteem verkent.

Computergrafiek en muziek maken het computergebruik leuker, en wij hebben de MEGA65 ontworpen om leuk te zijn! In deze handleiding leert u programma's schrijven die gebruikmaken van de ingebouwde **grafische** en **geluids**mogelijkheden van de MEGA65. U hoeft echter geen programmeur te zijn om plezier met de MEGA65 te beleven. Omdat de MEGA65 een complete Commodore® 64™[2] bevat, kan hij ook duizenden bestaande spellen, hulpprogramma's en zakelijke programmapakketten uitvoeren, evenals nieuwe programma's die op dit moment door liefhebbers van de Commodore-computer worden geschreven. Het enthousiasme voor de MEGA65 zal groeien naarmate wij getuige zijn van de programmeerwonderen die onze MEGA65-community voortbrengt, terwijl zij (en u!) de krachtige mogelijkheden van deze moderne herschepping van een Commodore-computer ontdekken en beheersen. Samen

[1] Commodore is een handelsmerk van C= Holdings
[2] Commodore 64 is een handelsmerk van C= Holdings

met u kunnen wij een nieuwe "homebrew"-community opbouwen, barstensvol proram-matuur en projecten die de mogelijkheden van de MEGA65 oprekken tot ver voorbij wat iemand ooit voor mogelijk had gehouden.

Wij heten u welkom op deze reis! Dank u dat u deel uitmaakt van de MEGA65-community van gebruikers, programmeurs en enthousiastelingen!

OVERIGE BOEKEN IN DEZE REEKS

Dit boek is één van de boeken uit de MEGA65-documentatiereeks. De reeks omvat:

- **De Mega65-gebruikershandleiding**
  Biedt een inleiding tot de MEGA65 en een beknopt overzicht van de BASIC 65-commando's
- **De Mega65 naslaggids BASIC65**
  Uitgebreide documentatie van alle BASIC 65-commando's, functies en operators
- **De Mega65-chipset naslaggids**
  Gedetailleerde documentatie over de eigen chips van de MEGA65 en de C65
- **De Mega65 programmeergids**
  Informatie voor ontwikkelaars die programma's voor de MEGA65 willen schrijven
- **Alles over de Mega65**
  (Ook bekend als **Het grote Mega65-boek**)
  Alle delen in één enkel omvangrijk pdf-bestand voor eenvoudig zoeken. 1200 bladzijden en nog steeds groeiend!

DOE MEE!

Raak betrokken, leer meer over uw MEGA65 en sluit u online bij ons aan op:

- https://mega65.org/chat
- https://mega65.org/forum

DEEL II

KENNISMAKEN MET UW MEGA65

HOOFDSTUK 3

Ingebruikname

- Uitpakken en aansluiten van de MEGA65
- Aansluitingen aan de achterzijde
- Aansluitingen aan de zijkant
- Beeldscherm en randapparatuur
- Optionele aansluitingen
- De MEGA65 inschakelen
- De Intro Disk
- De cursor

UITPAKKEN EN AANSLUITEN VAN DE MEGA65

Het is tijd om uw MEGA65-homecomputer in gebruik te nemen! De doos bevat het volgende:

- MEGA65-computer
- Voeding (zwart kastje met aansluiting voor de netspanning)
- Dit boek, de MEGA65-gebruikershandleiding
- Uw persoonlijke registratiecode, op een papiertje (mogelijk in de gebruikershandleiding opgeborgen)

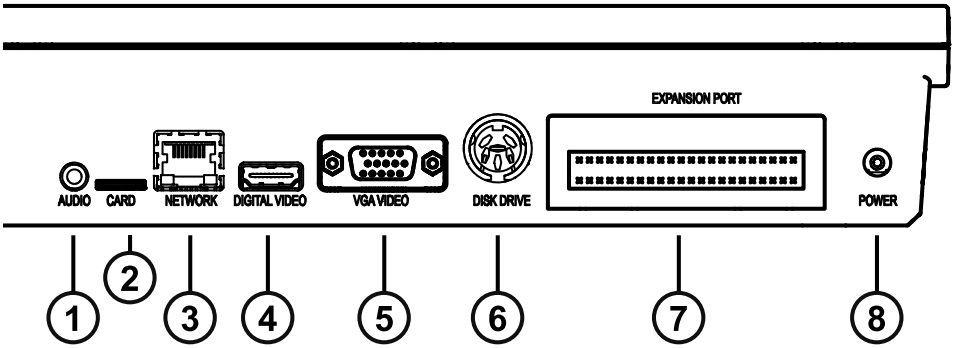
Om uw MEGA65-computer te kunnen gebruiken, heeft u bovendien het volgende nodig:

- Een televisie of computermonitor met een VGA- of digitale video-ingang, die een beeld van 480p (720x480) bij 60 Hz of 576p (720x576) bij 50 Hz kan weergeven
- Een geschikte videokabel voor uw beeldscherm, ofwel VGA, ofwel digitale video

Wellicht wilt u ook de volgende zaken gebruiken om het meeste uit uw MEGA65 te halen:

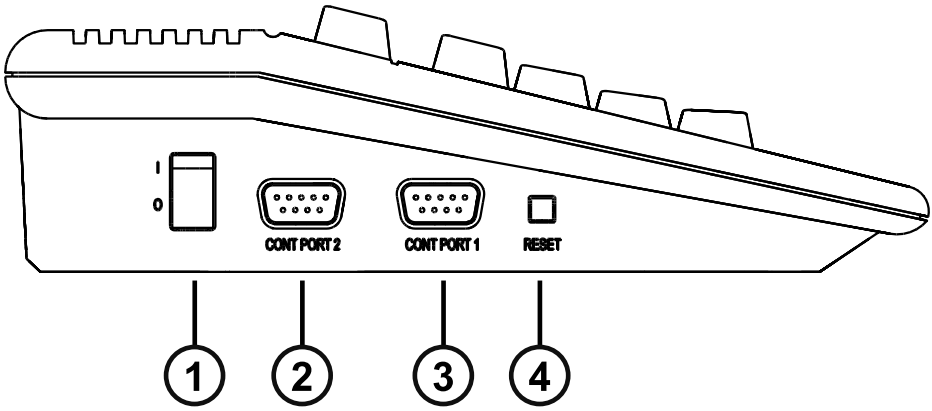
- Een beeldscherm met digitale video-aansluiting en ingebouwde audio, of actieve luidsprekers met een geschikte audiokabel met een 3.5 mm-audiojackstekker
- Een microSD-kaart, type SDHC, met een grootte tussen 4GB en 32GB
- Een RJ45-ethernetkabel en een netwerkrouter of -switch
- Een joystick of gamepad die compatibel is met Commodore-computers, met een negenpins-connector (DE-9)
- Een Commodore 1351-muis, een Amiga-muis of een moderne vervanging, zoals een mouSTer-USB-muisadapter

AANSLUITINGEN AAN DE ACHTERZIJDE



1	3.5 mm-audiojack
2	Externe microSD-kaartsleuf
3	Netwerk-LAN-poort
4	Digitale video-aansluiting (inclusief geluid)
5	VGA-videoaansluiting
6	Seriële IEC-bus-aansluiting voor schijfstations en printers
7	Cartridge-uitbreidingspoort
8	Voedingsaansluiting

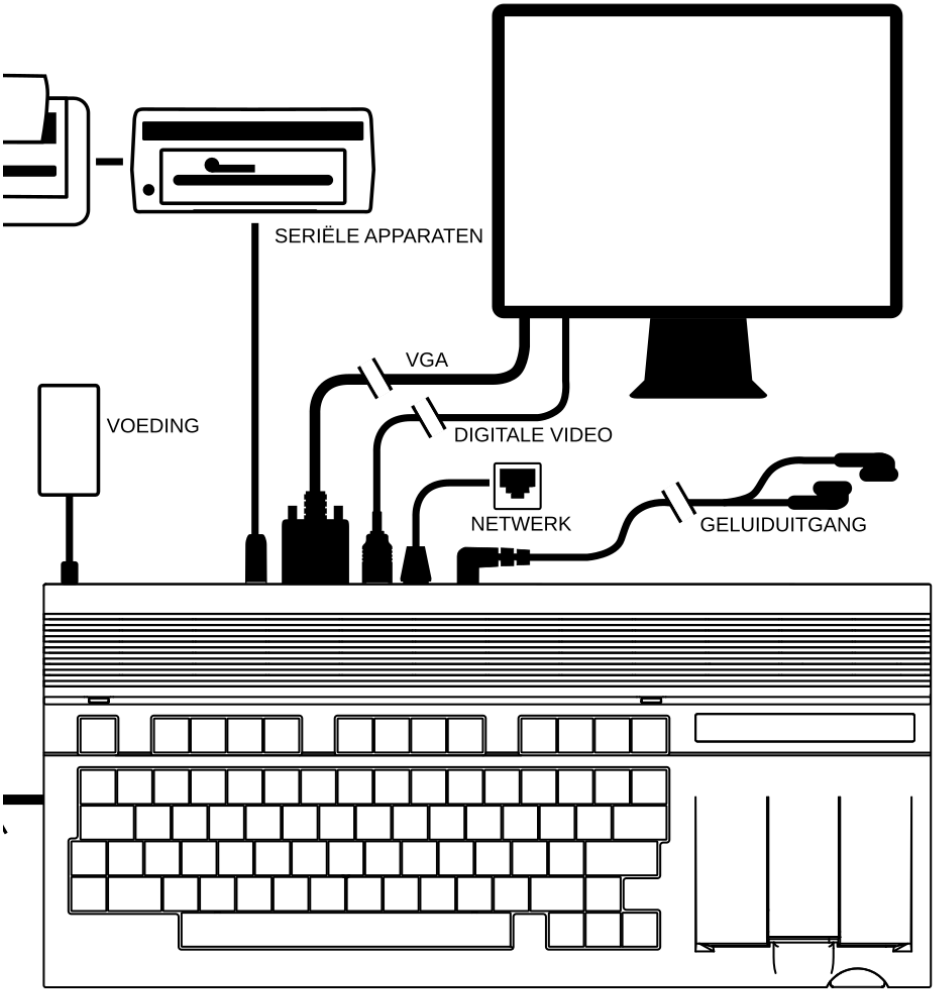
AANSLUITINGEN AAN DE ZIJKANT



1	Aan/uit-schakelaar
2	Controllerpoort 2
3	Controllerpoort 1
4	Resetknop

Op controllerpoorten 1 en 2 kunt u diverse randapparatuur aansluiten, zoals joysticks, peddels of muizen.

BEELDSCHERM EN RANDAPPARATUUR



U sluit uw MEGA65 als volgt op een beeldscherm aan:

1. Sluit de voeding aan op de voedingsaansluiting van de MEGA65.
2. Als u een VGA-monitor en een VGA-kabel hebt, sluit u het ene uiteinde aan op de VGA-poort van de MEGA65 en het andere uiteinde op uw VGA-monitor.

3. Als u een tv of monitor met een compatibele Digital Video-aansluiting hebt,[1] sluit u het ene uiteinde van uw kabel aan op de Digital Video-poort van de MEGA65 en het andere uiteinde op de Digital Video-poort van uw monitor. Als u een monitor met een DVI-aansluiting hebt, kunt u een Digital Video-naar-DVI-adapter gebruiken.

[1] Het type Digital Video-aansluiting heeft een herkenbare commerciële naam van vier letters, doch het MEGA65-project heeft geen licentiekosten betaald om naar dit type met deze naam te verwijzen. Deze *Gebruikershandleiding* verwijst hiernaar als de "Digital Video"-aansluiting.

OPTIONELE AANSLUITINGEN

1. De MEGA65 bevat een ingebouwd 3.5 inch-diskettestation. U kunt ook oudere Commodore®-diskettestations via de seriële IEC-bus aansluiten, zoals de Commodore 1541, 1571 of 1581. Sluit om deze stations te gebruiken het ene uiteinde van een IEC-kabel aan op het Commodore-diskettestation en het andere uiteinde op de schijfstationsaansluiting van de MEGA65. U kunt ook een SD2IEC-apparaat of een Pi1541-apparaat aansluiten. Bij de meeste apparaten kunt u extra diskettestations of met Commodore compatibele printers aan elkaar koppelen.
2. U kunt uw MEGA65 met een standaard ethernetkabel aansluiten op een ethernetnetwerk.
3. Om naar audio van uw MEGA65 te luisteren, kunt u een 3.5 mm-audiojackkabel aansluiten op een audioversterker of een luidsprekersysteem. Als uw systeem over RCA-aansluitingen beschikt, hebt u een 3.5 mm-audiojack-naar-twin-RCA-adapterkabel nodig. De MEGA65 heeft tevens een ingebouwde versterker, zodat u een hoofdtelefoon kunt gebruiken.
4. U kunt een microSD-kaart, type SDHC tussen 4GB en 32GB, in de externe microSD-kaartsleuf aan de achterzijde van de MEGA65 steken. Zie voor meer informatie over het gebruik van de microSD-kaartsleuf “Kennismaken met SD-kaarten” op bladzijde 5-9.
5. Aan de onderzijde van de MEGA65 geeft een klein klepje toegang tot de interne SD-kaart en twee aansluitingen voor toekomstige hardware-uitbreidingen.

De batterij van de real-time klok installeren

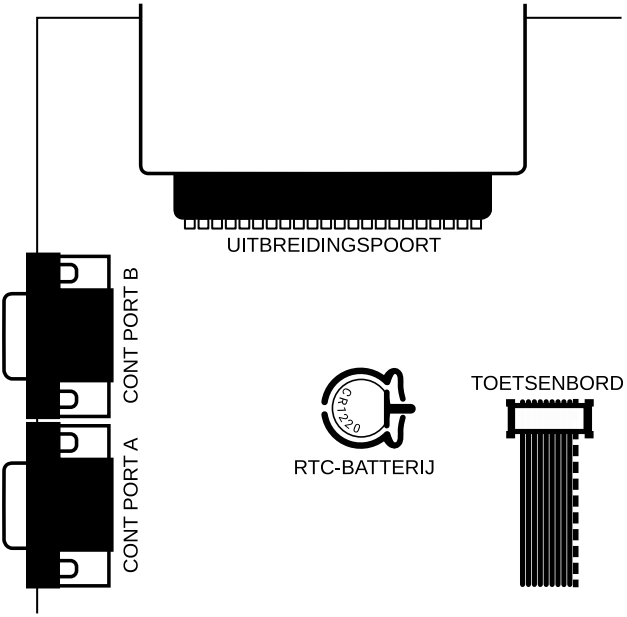
De MEGA65 bevat een real-time klok, welke wordt gebruikt om de tijd en datum op het opstartscherm weer te geven, om tijdstempels toe te voegen aan bestanden die de MEGA65 op uw SD-kaarten schrijft en om de **DT$** en **TI$**-BASIC-variabelen voor gebruik in programma’s te verschaffen. Deze klok gebruikt een CR2032-knoopcelbatterij om de tijd bij te houden wanneer de MEGA65 gedurende langere perioden van de voeding is losgekoppeld. De MEGA65 wordt niet met een batterij geleverd, om problemen met het internationaal verzenden van batterijen te vermijden.[2]

De RTC-batterij is optioneel. De MEGA65 kan de RTC zonder batterij draaiende houden, zelfs wanneer de computer gedurende meerdere dagen achter elkaar van de voeding is losgekoppeld. Met een batterij kan de computer de tijd gedurende veel langere perioden onthouden. U kunt de klok later altijd opnieuw instellen.

[2] Vroege modellen van de MEGA65 met de “R3A”-printplaatrevisie (gemaakt in het jaar 2022) gebruiken een batterij van het type CR1220 voor de real-time klok en de batterij is vereist om de RTC te laten functioneren. Revisie “R6” (gemaakt in 2024) gebruikt een batterij van het type CR2032 en de batterij is optioneel voor normaal gebruik.

Schakel de computer uit om de batterij te installeren. Gebruik een kruiskopschroevendraaier om de behuizing te openen, waardoor het moederbord zichtbaar wordt. De behuizing wordt bijeengehouden door drie schroeven, allemaal langs de onderkant van de voorzijde van de behuizing. Til, nadat de schroeven zijn verwijderd, voorzichtig de bovenste helft van de behuizing op. Let op de oriëntatie van de toetsenbordaansluiting en koppel deze vervolgens los.

De batterij bevindt zich tussen de controllerpoorten en de toetsenbordaansluiting.



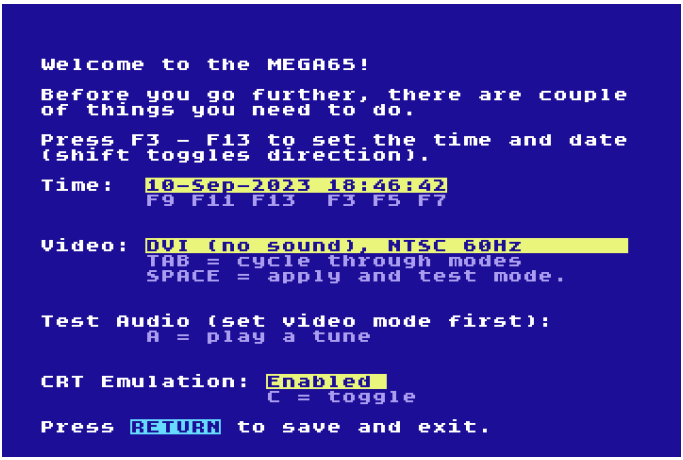
Indien u een bestaande batterij verwijdert, duwt u de ontgrendelingshendel van de batterij aan de onderkant (de platte zijde) van de batterijhouder weg van de batterij om deze te verwijderen. Plaats de nieuwe batterij met de positieve zijde (gemarkeerd met +) naar boven gericht en druk hem op zijn plaats.

Zodra u uw MEGA65 weer in elkaar hebt gezet, kunt u in het configuratiehulpprogramma de tijd instellen. Zie voor meer informatie over het instellen van de real-time klok het onderdeel over het configuratiehulpprogramma op bladzijde 5-3.

DE MEGA65 INSCHAKELEN

Schakel de MEGA65 in met de aan/uit-schakelaar aan de linkerkant van de computer.

Wanneer u uw MEGA65 voor het eerst inschakelt, toont de computer het scherm voor de initiële configuratie ("on-boarding"). Op dit scherm kunt u de tijd en de datum op de real-time klok instellen, de videomodus wijzigen en het geluid testen. Al deze instellingen kunt u later nog wijzigen.

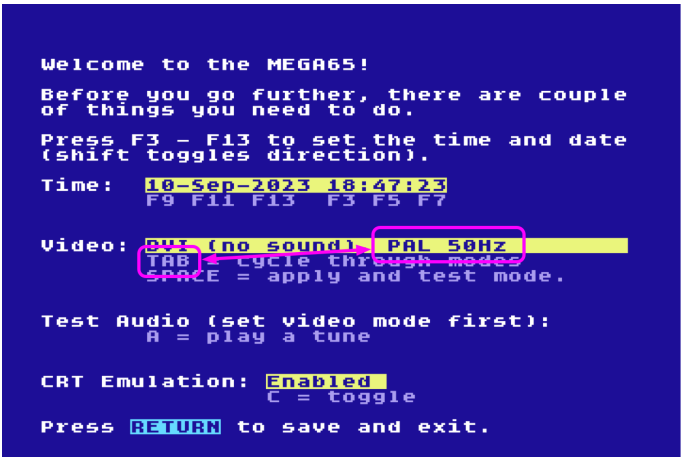


Voor videomodi kunt u kiezen tussen PAL-emulatie en NTSC-emulatie. Daarnaast kunt u aangeven of uw Digital Video-beeldscherm geluid ondersteunt. Indien u de VGA-videouitgang gebruikt, heeft de Digital Video-geluidsmodus geen effect.

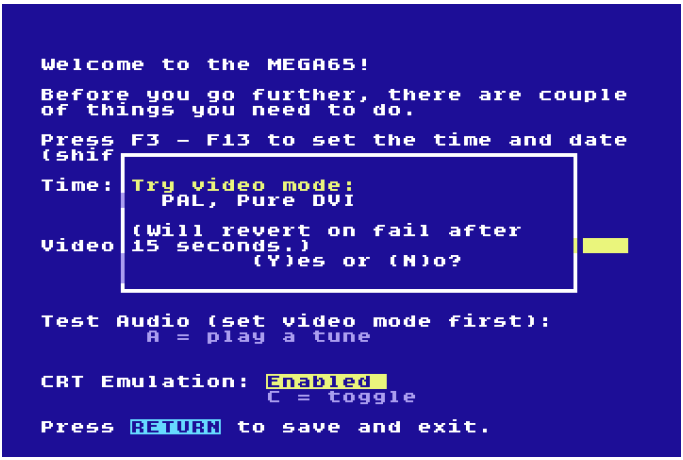
OPMERKING: Een DVI-beeldscherm dat geen geluid ondersteunt, werkt niet met de "enhanced"-geluidsmodus. Bij een dergelijk beeldscherm moet u een videomodus met "no sound" selecteren en een luidspreker op de 3.5 mm-audioaansluiting aansluiten.

PAL en NTSC zijn analoge videosignaalformaten die de resolutie en de verticale synchronisatiesnelheid van de videouitgang beïnvloeden, ook wanneer u een modern digitaal beeldscherm gebruikt. Uw beeldscherm ondersteunt mogelijk beide modi of slechts één van beide. U kunt dit scherm gebruiken om de modi met uw beeldscherm te testen.

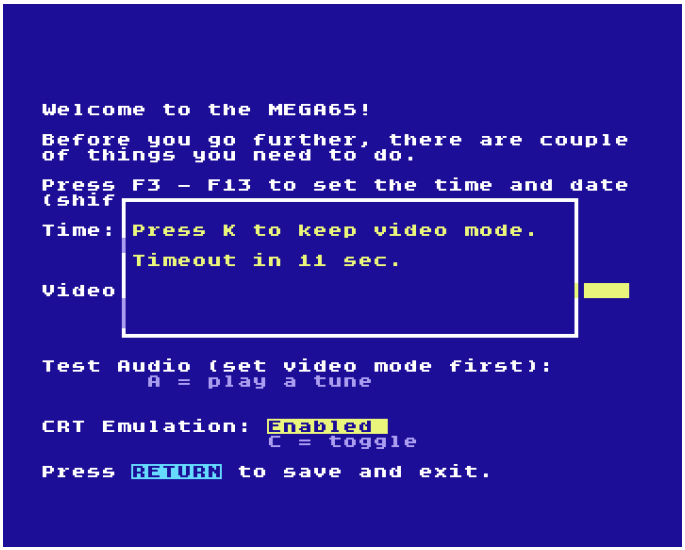
Selecteer en test uw videoconfiguratie. Druk bijvoorbeeld op TAB om naar de modus **PAL 50Hz** over te schakelen.



Druk op **SPACE** en vervolgens op **Y** om de nieuwe videomodus te testen.



Druk op **K** om de nieuwe videomodus te behouden.



Maak van deze gelegenheid gebruik om uw geluidsopstelling te testen. Druk op **A** om een geluid af te spelen.

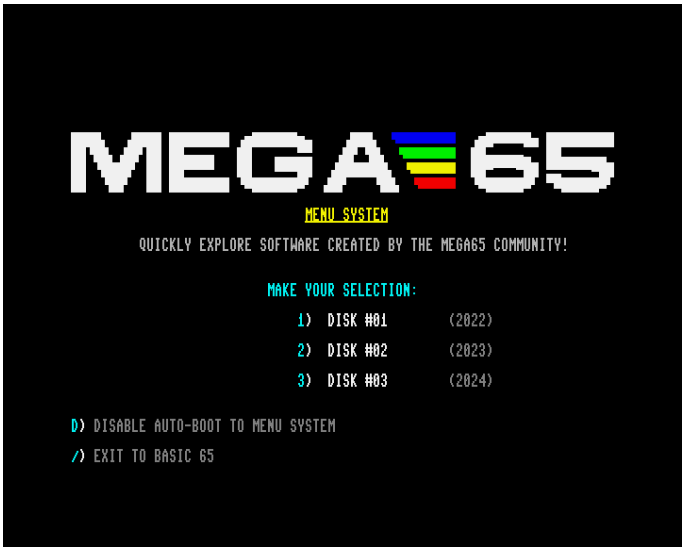
De optie "CRT Emulation" is een leuke keuze bij gebruik van een modern plat scherm. Deze optie voegt verticale tussenruimten toe tussen de pixels om de rasterlijnen van een beeldbuis na te bootsen. Probeer het om te zien of het u bevalt: druk op de toets **C** om de optie in en uit te schakelen.

Druk ten slotte op **RETURN** om de configuratie af te ronden.

Zie voor meer informatie over het configureren van uw MEGA65 hoofdstuk 5 op bladzijde 5-3.

DE INTRO DISK

Nadat u de initiële configuratie hebt afgerond, start uw MEGA65 het Intro Disk-menu. De Intro Disk is een verzameling software van de MEGA65-gemeenschap die enkele van de mogelijkheden van de computer demonstreert. Neem de tijd om door de menu's te bladeren en een aantal demo's uit te proberen. Druk na elke demo op de resetknop aan de linkerkant van de computer om terug te keren naar het Intro Disk-menu.



Standaard opent het Intro Disk-menu telkens wanneer u de computer inschakelt. Zodra u meer vertrouwd bent met de MEGA65, wilt u dit wellicht uitschakelen. Druk bij het Intro Disk-menu op **D** om de automatische opstartfunctie van de Intro Disk uit te schakelen.

Druk op **/** (de schuine streep) om het Intro Disk-menu te verlaten en toegang te krijgen tot BASIC 65. Indien de automatische opstartfunctie van de Intro Disk is uitgeschakeld, gaat de MEGA65 bij het inschakelen rechtstreeks naar BASIC 65.



DE CURSOR

Het knipperende blokje onder de READY-prompt wordt de *cursor* genoemd. De cursor geeft aan dat de computer klaar is om invoer te ontvangen. Wanneer u toetsen op het toetsenbord indrukt, verschijnen de bijbehorende tekens op het scherm. De tekens verschijnen op de huidige cursorpositie en de cursor schuift na elke toetsaanslag een positie op.

Hier kunt u commando's typen waarmee u bijvoorbeeld een programma kunt laden. U kunt ook beginnen met het intypen van programmacode!

HOOFDSTUK 4

Aan de slag

- Toetsenbord
- De schermeditor
- Functionaliteit van de editor
- Het Freezer-menu
- Commodore 64-software uitvoeren

TOETSENBORD

Nu alles is aangesloten, is het tijd om vertrouwd te raken met het toetsenbord van de MEGA65.

Wellicht valt u op dat het toetsenbord enigszins afwijkt van de toetsenborden van andere hedendaagse computers. De meeste toetsen staan op vertrouwde posities, maar er zijn enkele gespecialiseerde toetsen en sommige toetsen hebben speciale grafische symbolen op de voorkant.

In sommige beeldschermmodi zijn de grafische symbolen te typen, net als letters, cijfers en leestekens. De volledige verzameling tekens staat bekend als de *PETSCII-tekenset*.

Speciale toetsen

RETURN

Wanneer u op RETURN drukt, slaat de computer de informatie die u hebt getypt op in het geheugen. De computer voert vervolgens een opdracht uit, bewaart enkele gegevens of toont een foutmelding indien u een vergissing hebt gemaakt.

SHIFT

De twee SHIFT-toetsen bevinden zich links en rechts. Zij werken vrijwel hetzelfde als de Shift-toets van een gewoon toetsenbord en vervullen daarnaast nog enkele speciale functies.

In de modus met hoofdletters levert het ingedrukt houden van SHIFT en het indrukken van een toets met twee grafische symbolen op de voorkant het rechter symbool van die toets op. Zo toont SHIFT samen met J het teken ◹.

In de modus met kleine letters geeft het indrukken van SHIFT en een lettertoets de hoofdletter van die toets weer.

Om de Griekse letter π (pi) te typen, houdt u SHIFT ingedrukt en drukt u op ↑.

Het ingedrukt houden van SHIFT en het indrukken van een functietoets roept de functie op die op de voorkant van die toets staat vermeld. Zo activeert SHIFT samen met F1 de functie F2.

SHIFT LOCK

Naast SHIFT is er de SHIFT LOCK-toets. Druk op deze toets om de Shift-functie vast te zetten. Wanneer SHIFT LOCK verlicht is, toont elke toets die u indrukt het bijbehorende teken op het scherm, alsof u SHIFT ingedrukt houdt. Dit geldt ook voor speciale grafische tekens.

CTRL

CTRL is de Control-toets. Door CTRL ingedrukt te houden en een andere toets in te drukken, kunt u Controlcodes uitvoeren. Zo kunt u met CTRL ingedrukt en één van de cijfertoetsen (van 1 tot 8) de tekstkleur wijzigen. De kleur die op de bovenste rij van de voorkant van de cijfertoets is afgedrukt, wordt gebruikt. Het ingedrukt houden van CTRL en het indrukken van 9 of 0 schakelt de modus voor omgekeerde tekst in en uit.

Enkele voorbeelden hiervan vindt u op bladzijde 4-8 en alle Controlcodes staan vermeld op bladzijde E-3.

Wanneer een programma met **LIST** op het scherm wordt weergegeven, vertraagt het ingedrukt houden van CTRL de weergave van elke regel. Meer over het commando **LIST** leest u op bladzijde B-158.

Het ingedrukt houden van CTRL en het indrukken van * opent de Matrix Mode Debugger (raadpleeg **Het grote Mega65-boek** voor meer details).

RUN/STOP

Normaal gesproken stopt het indrukken van RUN STOP de uitvoering van een programma. Wanneer u SHIFT ingedrukt houdt en op RUN STOP drukt, voert u met **RUN** het eerste programma van de schijf uit.

Sommige programma's onderscheppen de RUN STOP-toets en kunnen op deze wijze niet worden gestopt.

U kunt uw MEGA65 naar de **Machinetaalmonitor** opstarten door RUN STOP ingedrukt te houden en de resetknop aan de linkerkant van de computer in te drukken (of door de toets ingedrukt te houden terwijl u de computer inschakelt).

RESTORE

Het computerscherm kan naar een schone toestand worden teruggebracht zonder het geheugen te wissen door RUN STOP ingedrukt te houden en RESTORE in te drukken.

Deze combinatie herstelt tevens de vectoren van het besturingssysteem en initialiseert de schermeditor opnieuw, hetgeen dit tot een handige combinatie maakt indien de computer een beetje in de war is geraakt.

Sommige programma's onderscheppen de toetsencombinatie RUN STOP + RESTORE en kunnen op deze wijze niet worden gereset.

U kunt de **Freezer** ook oproepen door RESTORE één seconde ingedrukt te houden en de toets vervolgens los te laten. U kunt meer over de Freezer lezen op bladzijde 4-13.

De cursor-toetsen

De cursor-toetsen staan rechtsonder op het toetsenbord. Met deze vier richtingtoetsen kunt u de cursor naar elke gewenste positie verplaatsen voor het bewerken van tekst op het scherm.

De cursor beweegt in de richting die op de toetsen is aangegeven: ← ↑ → ↓.

U hoeft een cursor-toets niet telkens opnieuw in te drukken. Indien u de cursor over een grote afstand moet verplaatsen, kunt u de toets ingedrukt houden. Wanneer u klaar bent, laat u de toets gewoon los.

De pijltjestoetsen

Deze toetsen verschillen van de cursor-toetsen. Het zijn ← (naast 1) en ↑ (naast RESTORE). Beide pijltjestoetsen worden gebruikt in diverse BASIC-functies en escape-sequenties.

Zo kan ← als snelkoppeling voor **SAVE** dienen en wordt ↑ gebruikt om een getal tot een macht te verheffen (hetgeen gelijkstaat aan het getal een opgegeven aantal malen met zichzelf te vermenigvuldigen).

U kunt meer lezen over de beschikbare escape-sequenties op bladzijde E-6.

Deze twee PETSCII-specifieke toetsen worden in de MEGA65-literatuur altijd met een witte achtergrond weergegeven.

Het is ook mogelijk de cursor met SHIFT en ↓ omhoog en met SHIFT en → naar links te verplaatsen. Dit is een erfenis van de Commodore 64, welke slechts twee cursor-toetsen had.

INSerT/DELete

De toets INST DEL, de meest rechtse toets van de bovenste rij, is de INSERT/DELETE-toets. Wanneer u INST DEL indrukt, wordt het teken links daarvan verwijderd en worden alle tekens rechts daarvan één positie naar links verschoven.

Om een teken in te voegen, houdt u SHIFT ingedrukt en drukt u op INST DEL. Het teken onder de cursor en de tekens rechts van de cursor worden naar rechts verschoven. Op de nieuw vrijgekomen plaats kunt u vervolgens een letter, een cijfer of elk ander teken typen.

CLeaR/HOME

De toets CLR HOME is de CLEAR/HOME-toets. Het indrukken van CLR HOME verplaatst de cursor naar de linkerbovenhoek van het scherm.

Door SHIFT ingedrukt te houden en op CLR HOME te drukken, wist u het volledige scherm *en* plaatst u de cursor in de linkerbovenhoek van het scherm.

Indien u per ongeluk CLR HOME indrukt, kunt u de cursor naar zijn vorige positie terugbrengen door ESC en vervolgens CLR HOME in te drukken.

De MEGA-toets

De of de MEGA-toets biedt een aantal verschillende functies en kan worden gebruikt om speciale hulpprogramma's op te starten.

Door SHIFT ingedrukt te houden en in te drukken, schakelt u tussen de kleine- en de hoofdlettermodus.

Door ingedrukt te houden en een willekeurige toets met twee grafische symbolen op de voorkant in te drukken, geeft de computer het meest linkse grafische symbool op het scherm weer. Zo geven en D het symbool ◪ weer.

Door ingedrukt te houden en een cijfertoets in te drukken, schakelt u over naar een secundaire kleur, d.w.z. de kleur die op de onderste rij aan de voorkant van de cijfertoets is afgedrukt.

Door ingedrukt te houden en TAB in te drukken, opent u de Matrix Mode Debugger (zie voor meer details **Het grote Mega65-boek**).

Indien u de MEGA65 inschakelt, of de resetknop aan de linkerkant indrukt terwijl u ingedrukt houdt, schakelt de MEGA65 over naar de GO64-modus.

NO SCROLL

Indien een programma met **LIST** naar het scherm wordt afgedrukt, onderbreekt u de schermuitvoer door NO SCROLL in te drukken. Druk een willekeurige toets in om de pauze op te heffen.

Deze functie is niet beschikbaar in de GO64-modus.

De functietoetsen

Er zijn zeven functietoetsen beschikbaar die door softwaretoepassingen gebruikt kunnen worden. F1 F3 F5 F7 F9 F11 en F13 kunnen worden gebruikt om speciale functies uit te voeren.

Houd SHIFT ingedrukt om F2 tot en met F14 te gebruiken, zoals op de voorkant van elke functietoets is aangegeven. In de GO64-modus zijn alleen de functietoetsen F1 tot en met F8 beschikbaar.

HELP

HELP kan door software worden gebruikt en fungeert tevens als F15.

ALT

Software kan specifieke functies uitvoeren door ALT ingedrukt te houden en tegelijkertijd andere toetsen in te drukken. Deze functie is niet beschikbaar in de GO64-modus.

Door ALT ingedrukt te houden terwijl u de MEGA65 inschakelt, opent u het hulpprogrammamenu. U kunt een SD-kaart formatteren, het MEGA65-configuratiehulpprogramma openen om de standaard videomodus te kiezen en andere instellingen te wijzigen, of uw toetsenbord testen.

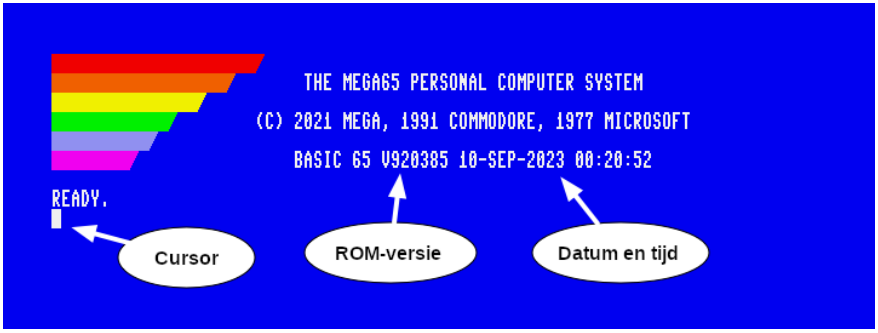
CAPS LOCK

CAPS LOCK werkt op dezelfde wijze als SHIFT LOCK in de C65- en MEGA65-modus, maar heeft alleen invloed op de lettertoetsen.

Indien de MEGA65 is ingesteld om op een lagere processorsnelheid te draaien, zoals in de GO64-modus, kunt u CAPS LOCK ingedrukt houden om de processor tijdelijk op volledige snelheid te laten draaien. Dit is in de GO64-modus nuttig om bijvoorbeeld het laden vanaf het interne schijfstation of de SD-kaart te versnellen, of om het uitpakproces na het uitvoeren van een programma aanzienlijk te versnellen. De MEGA65-modus draait standaard op maximale snelheid.

DE SCHERMEDITOR

Wanneer u uw MEGA65 inschakelt of reset, verschijnt het volgende scherm:[1]



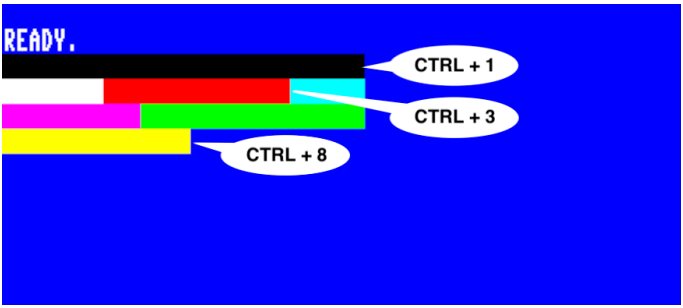
De kleurbalken linksboven in het scherm kunt u gebruiken als richtlijn om de kleuren van uw beeldscherm te kalibreren. Het scherm toont tevens de naam van het systeem, de copyrightmelding en de ROM-versie. De weergegeven datum en tijd zijn afkomstig van de interne RTC (Real-Time Clock) op het moment dat de computer werd ingeschakeld. U kunt de datum en tijd instellen in het configuratiehulpprogramma.

Ten slotte ziet u de READY-prompt en de knipperende cursor.

U kunt beginnen met typen op het toetsenbord en de tekens verschijnen op de cursorpositie. Na elke toetsaanslag schuift de cursor één positie verder.

U kunt ook omgekeerde tekst of kleurbalken maken door CTRL ingedrukt te houden en 9 of R in te drukken. Hiermee schakelt u de modus voor omgekeerde tekst in. Indien deze modus is ingeschakeld, kunt u de spatiebalk SPACE indrukken en ingedrukt houden. Terwijl u dat doet, wordt een witte balk over het scherm getekend. U kunt de huidige kleur wijzigen door CTRL ingedrukt te houden en een cijfertoets in te drukken (van 1 tot 8). Indien u bijvoorbeeld CTRL ingedrukt houdt en 1 indrukt, verandert de kleur in zwart. Wanneer u nu de spatiebalk SPACE ingedrukt houdt, wordt een zwarte balk getekend. Indien u de kleur blijft wijzigen en de spatiebalk SPACE indrukt, krijgt u een effect dat vergelijkbaar is met de volgende afbeelding:

[1] Hiervoor moet u het Intro Disk-menu hebben uitgeschakeld. Indien het Intro Disk-menu draait, drukt u op "/" (schuine streep) om naar dit scherm terug te keren.



U kunt de modus voor omgekeerde tekst uitschakelen door CTRL ingedrukt te houden en 0 in te drukken.

Nog acht kleuren kunt u selecteren door ■ ingedrukt te houden en een toets van 1 tot 8 in te drukken.

De kleur die op de onderste rij aan de voorkant van de cijfertoets is afgedrukt, wordt gebruikt. Indien u bijvoorbeeld ■ ingedrukt houdt terwijl u 4 indrukt, wordt donkergrijs gebruikt. Voor toegang tot nog 16 kleuren van het alternatieve/rainbow-palet verwijzen wij u naar de sneltoets CTRL + A, die beschreven staat op bladzijde E-3.

OPMERKING:

- **Zinnebeeldmodus**: Indien u " indrukt om een tekenreeks te beginnen en vervolgens probeert kleuren te wijzigen, tekst om te keren, de cursor te verplaatsen of de toets CLR HOME te gebruiken, verschijnen er in plaats van dat deze acties onmiddellijk plaatsvinden, grappige PETSCII-tekens. Dit komt door een voorziening van BASIC die *zinnebeeldmodus* heet en waarmee u dergelijke acties in een tekenreeks kunt coderen, zodat u ze op een later tijdstip kunt uitvoeren (bijvoorbeeld via een **PRINT**-instructie in uw programma's). Om de zinnebeeldmodus te beëindigen typt u eenvoudig een nieuwe " om het einde van uw tekenreeks aan te geven.
- **Invoegmodus**: Een vergelijkbare voorziening is de *invoegmodus*. Het aantal keren dat u SHIFT + INST DEL indrukt om enkele spaties in te voegen, is gelijk aan het aantal toetsaanslagen dat erop volgt en dat zich volgens dezelfde beginselen als de zinnebeeldmodus gedraagt.
- U kunt beide modi ook geforceerd beëindigen door ESC en daarna O in te drukken.

U kunt eenvoudig leuke afbeeldingen maken door deze kleuren en letters te gebruiken. Hieronder ziet u een voorbeeld van wat een vierdejaarsstudent tekende:



Wat gaat u tekenen?

Functies

Functies die gebruikmaken van CTRL worden **Controlcodes** genoemd. Functies die gebruikmaken van ▮▮ worden **Megacodes** genoemd. Er bestaan ook functies die via SHIFT worden aangeroepen; deze worden **Shiftcodes** genoemd.

Ten slotte maakt ESC het gebruik van **escape-sequenties** mogelijk.

Over al deze functies leest u meer op bladzijde E-3.

Escape-sequenties

Escape-sequenties worden iets anders uitgevoerd dan een controlcode of een shift-code. In plaats van de modificatietoets ingedrukt te houden, voert u een escape-sequentie uit door ESC in te drukken, deze los te laten en vervolgens de gewenste toets in te drukken.

Om bijvoorbeeld te schakelen tussen de 80-kolomsmodus en de 40-kolomsmodus drukt u ESC in, laat u deze los en drukt u vervolgens op X.

Er zijn nog meer tekstmodi beschikbaar. U kunt knipperende tekst maken door CTRL ingedrukt te houden en op O te drukken. Alle tekens die u intypt, zullen knipperen. Schakel de knippermodus uit door ESC en daarna O in te drukken.

FUNCTIONALITEIT VAN DE EDITOR

De schermeditor van de MEGA65 biedt verschillende manieren om de cursor snel over het scherm te verplaatsen, zodat u productiever kunt werken.

Druk bijvoorbeeld op CLR HOME om naar de startpositie op het scherm te gaan. Houd CTRL ingedrukt en druk enkele malen op W. Dit is de functie **Word Advance**, die uw cursor naar het volgende woord of afdrukbare teken verplaatst.

U kunt naar eigen inzicht tabposities op het scherm instellen. Druk op CLR HOME en vervolgens op → om de cursor naar de vierde kolom te verplaatsen. Houd CTRL ingedrukt en druk op X om een tab in te stellen. Verplaats de cursor nog 16 posities naar rechts en druk opnieuw op CTRL en X om een tweede tab in te stellen.

Druk op CLR HOME om terug te keren naar de startpositie. Houd CTRL ingedrukt en druk op I. Dit is de functie **Forward Tab**. Uw cursor springt naar de vierde positie. Druk opnieuw op CTRL en I. Uw cursor verplaatst zich naar positie 8. Standaard is elke achtste positie reeds als tabpositie ingesteld. De vierde en de twintigste positie zijn derhalve toegevoegd aan de bestaande tabposities. U kunt op CTRL en I blijven drukken om door te springen naar de zestiende en twintigste positie.

Een venster maken

U kunt een venster instellen op het werkscherm van de MEGA65. Verplaats uw cursor naar het begin van de tekst “BASIC 65”. Druk op ESC en daarna op T. Verplaats de cursor 10 regels omlaag en 15 posities naar rechts.

Druk op ESC en daarna op B. Alles wat u typt, blijft binnen dit venster.

Indien u bijvoorbeeld LIST typt om een programma weer te geven, blijft de listing beperkt tot het venstergedeelte dat u hebt aangegeven:



Druk tweemaal op CLR HOME om het venster te verlaten en terug te keren naar het volledige scherm.

Aanvullende ASCII-tekens

U hebt misschien opgemerkt dat zich op het toetsenbord van de MEGA65 een aantal ASCII-tekens bevindt die van oudsher geen deel uitmaken van de PETSCII-tekenset. Om deze tekens te gebruiken:

- Typ FONT A of FONT B.
- Druk op ◤ + SHIFT om over te schakelen naar kleine letters.

U kunt die extra ASCII-tekens nu typen door ◤ ingedrukt te houden en de toets in te drukken.

Naam	Teken	Toetscombinatie
Backtick	`	+ ←
Tilde	~	+ , (komma)
Verticale streep	|	+ . (punt)
Backslash	\	+ /
Linker accolade	{	+ :
Rechter accolade	}	+ ;
Underscore	_	+ =

De alternatieve lettertypen vervangen tekens uit de oorspronkelijke PETSCII-tekenset door deze symbolen. Om terug te keren naar de PETSCII-tekenset, typt u `FONT C`.

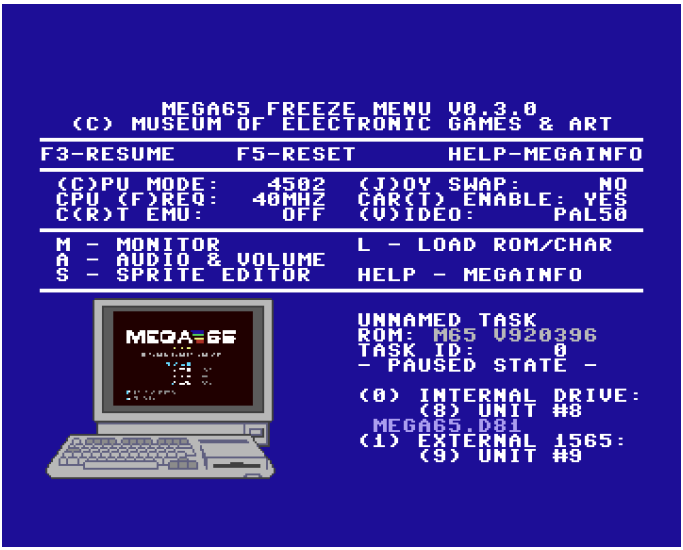
Hoofdletters en kleine letters

+ SHIFT schakelt voor het volledige scherm tussen tekst in hoofdletters en tekst in kleine letters. Dit werkt ook tijdens de uitvoering van een programma, zodat u het kunt aanpassen wanneer een programma zich in de verkeerde modus bevindt.

HET FREEZER-MENU

De MEGA65 brengt het grootste deel van zijn tijd door als een Commodore 65-computer, door een programma uit te voeren of te wachten op instructies in de BASIC-omgeving. Uw MEGA65 beschikt over extra voorzieningen die geen deel uitmaakten van het oorspronkelijke C65-ontwerp. U hebt toegang tot veel van deze voorzieningen via het Freezer-menu.

Om het Freezer-menu te openen, houdt u de toets RESTORE één seconde ingedrukt en laat u deze vervolgens los. De MEGA65 onderbreekt zijn bezigheden, laat de randkleur flikkeren en opent vervolgens het Freezer-menu. Het programma dat op dat moment liep, blijft in het geheugen staan en kan worden hervat door op de toets F3 te drukken. U kunt het draaiende programma ook afbreken en de MEGA65 resetten door op F5 te drukken.



Een functie om te onthouden tijdens het spelen van spellen is “(J)OY SWAP”. Deze functie laat de joystickpoorten van nummer wisselen. Hebt u een joystick in poort 2 en start u een spel dat een joystick in poort 1 verwacht, dan hoeft u de joystick niet los te koppelen en opnieuw aan te sluiten. Open in plaats daarvan het Freezer-menu, druk op **J** om de poortnummers om te wisselen en hervat vervolgens uw spel.

Het menu wordt “Freezer” genoemd omdat de toestand van de MEGA65 bevroren blijft terwijl u het menu gebruikt. Het Freezer-menu kan meerdere freeze-toestanden opslaan en u kunt tussen deze toestanden schakelen. Om de huidige toestand op te slaan, gaat u met de rechtercursortoets naar een ongebruikt *freeze-slot* en drukt u vervolgens op **F7**. Wanneer de rand ophoudt met knipperen, is de toestand opgeslagen. Om een toestand te herstellen, gaat u naar het freeze-slot en drukt u op **F3** om de werking te hervatten.

Het Freezer-menu bevat diverse ingebouwde opties en voorzieningen. Zie voor meer informatie over het MEGA65-informatiehulpprogramma (“MEGAINFO”) “De versies van de onderdelen bepalen” op bladzijde 6-6. Zie voor meer informatie over het aankoppelen van schijven en schijfimages hoofdstuk 7 op bladzijde 7-3.

COMMODORE 64-SOFTWARE UITVOEREN

De MEGA65 is in staat Commodore 64-software uit te voeren. Hiervoor zijn twee mogelijkheden: de ingebouwde GO64-modus en de FPGA-core *C64 for MEGA65*.

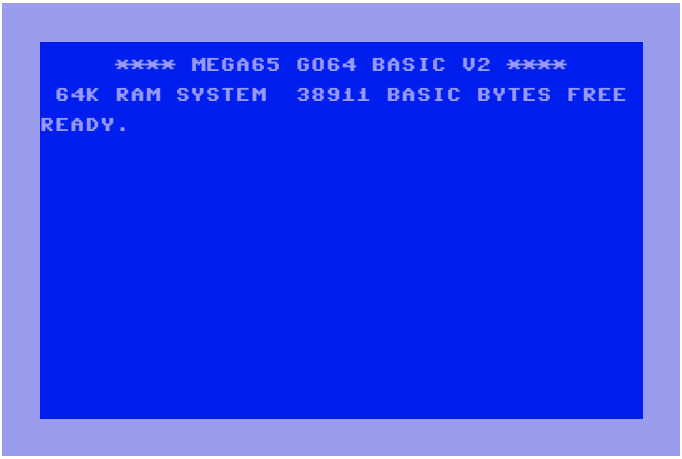
GO64-modus

De oorspronkelijke Commodore 65 was ontworpen om bepaalde Commodore 64-software te kunnen uitvoeren. De MEGA65 ondersteunt deze voorziening, die bekendstaat als "GO64-modus".

OPMERKING: Doordat Commodore deze voorziening op deze wijze heeft ontworpen, is niet alle C64-software compatibel met deze modus. Anders dan de vergelijkbare voorziening van de Commodore 128 gebruikt de Commodore 65 een andere CPU en het is bekend dat kleine verschillen compatibiliteitsproblemen met sommige softwaretitels veroorzaken.

Er zijn drie manieren om de MEGA65 in de GO64-modus te schakelen:

- Schakel de computer uit, houd de  ingedrukt en schakel hem weer in.
- Geef bij de READY-prompt van de MEGA65 dit commando in: GO64 en druk op Enter. Typ YES wanneer daarom wordt gevraagd.
- Schakel de computer uit, sluit een Commodore 64-cartridge aan op de expansiepoort en schakel de computer vervolgens weer in.



De GO64-modus is eigenlijk slechts een tijdelijke herconfiguratie van de MEGA65. Alle voorzieningen van de MEGA65 blijven aanwezig, waaronder het Freezer-menu voor het aankoppelen van D81-schijfimages.

Veel Commodore 64-software is op het internet te vinden in de vorm van D64-schijfimages. De MEGA65 ondersteunt via de SD-kaart en het Freezer-menu uitsluitend D81-schijfimages. U kunt een randapparaat, zoals de SD2IEC, op de IEC-poort van de MEGA65 aansluiten om D64-schijfimages te gebruiken. Zorg dat u een SD2IEC aanschaft met een onafhankelijke voeding en niet een exemplaar dat afhankelijk is van een tape-aansluiting van de Commodore 64.[2]

[2] Zie voor meer informatie over SD2IEC-apparaten: https://www.c64-wiki.com/wiki/SD2IEC

De FPGA-core “C64 for MEGA65”

De FPGA-core *C64 for MEGA65* van MJoergen en sy2002 herschept de oorspronkelijke Commodore 64-computer op de MEGA65-hardware met een hoge mate van nauwkeurigheid. Het doet dit door de MEGA65-core volledig te vervangen door een core die de chipset van de Commodore 64 implementeert, met inbegrip van de CPU. MEGA65-voorzieningen, zoals het Freezer-menu, zijn niet beschikbaar wanneer de C64-core draait. In plaats daarvan biedt de core een eigen menu voor het aankoppelen van D64-schijfimages en andere voorzieningen. Druk op de toets HELP terwijl de core draait om dit menu te openen.

Zie voor informatie over het installeren van FPGA-cores hoofdstuk 6 op bladzijde 6-5. Om de core *C64 for MEGA65* te downloaden en de belangrijke installatie-instructies te lezen, zie: https://github.com/MJoergen/C64MEGA65

HOOFDSTUK 5

Configureren van uw MEGA65

- Configureren van uw MEGA65
- Configuratiehulpprogramma
- Kennismaken met SD-kaarten
- Een nieuwe SD-kaart voorbereiden

CONFIGUREREN VAN UW MEGA65

Dit hoofdstuk beschrijft hoe u uw MEGA65 configureert.

Omdat de configuratiegegevens op de SD-kaart worden opgeslagen, beschrijft dit hoofdstuk ook hoe u een nieuwe SD-kaart voorbereidt. Uw MEGA65 wordt met een vooraf geïnstalleerde SD-kaart geleverd. Indien u uw MEGA65 met de vooraf geïnstalleerde SD-kaart configureert en later een nieuwe SD- of microSD-kaart installeert, moet u uw configuratie-instellingen opnieuw instellen.

Dit hoofdstuk laat u tevens kennismaken met de website MEGA65 Filehost, waarmee u games, apps, hulpprogramma's en systeemupdates voor uw computer kunt downloaden.

CONFIGURATIEHULPPROGRAMMA

U kunt uw MEGA65 configureren met het Configuratiehulpprogramma. Dit omvat de instellingen die werden weergegeven toen u de computer voor het eerst inschakelde en vele andere.

Om het Configuratiehulpprogramma te openen, schakelt u de MEGA65 uit, houdt u de toets ALT ingedrukt en schakelt u de MEGA65 weer in. Het hulpprogrammamenu verschijnt met verschillende opties. Druk op 1 om het Configuratiehulpprogramma te starten.

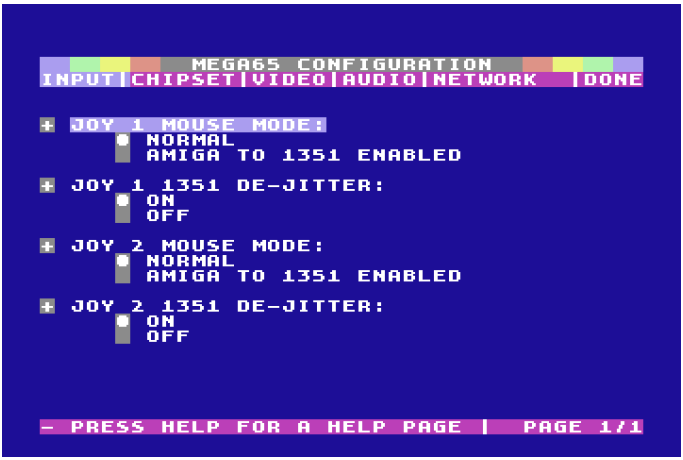


Het Configuratiehulpprogramma bevat verscheidene bladzijden met instellingen, die u kunt doorlopen met het toetsenbord of met een op poort 1 aangesloten muis. Gebruik ← en → om tussen de bladzijden te bladeren en ↑ en ↓ om onderdelen

op een bladzijde te selecteren. Druk op RETURN of SPACE om een instelling om te schakelen of een waarde te wijzigen.

Input

Op de bladzijde **Input** stelt u de muisinstellingen voor de twee randapparaatpoorten in.



De MEGA65 ondersteunt de Commodore 1351-muis, de Commodore Amiga-muis of moderne equivalenten, zoals een USB-muis die met een mouSTer-adapter is aangesloten. De poort moet op het juiste muistype worden ingesteld, waarbij **normal** naar de 1351-muis verwijst. Indien een Amiga-muis is aangesloten terwijl de poort in de modus **normal** staat, kan deze het gedrag van het toetsenbord verstoren.

Met de instelling **1351 De-jitter** stelt u de gevoeligheid in de Commodore 1351-muismodus af om jitter van het muispijltje te voorkomen. Het wordt aanbevolen deze op **on** te laten staan wanneer u in de modus **normal** een 1351-muis gebruikt.

Chipset

Op de bladzijde **Chipset** stelt u verschillende functies in, waaronder de real-time klok.



Om de real-time klok in te stellen, selecteert u het tijd- of datumveld, typt u de volledige waarde en drukt u vervolgens op RETURN. De klokinstelling wordt pas van kracht wanneer u op RETURN drukt; zonder RETURN wijzigt er niets. Merk op dat alle overige instellingen pas aan het einde worden opgeslagen. Alleen de RTC wordt onmiddellijk bijgewerkt.

Het veld **DMAGIC revision** bepaalt het gedrag van de DMA-controller. In de meeste gevallen wilt u de nieuwere instelling **F018B** gebruiken. De instelling **F018** dient voor achterwaartse compatibiliteit wanneer de C65-versies van de ROM worden gebruikt en is niet altijd vereist.

Het veld **F011 disk controller** bepaalt of de MEGA65 bij het inschakelen van de computer een opstartschijf zoekt op de SD-kaart of in het fysieke 3.5 inch-diskettestation. Wanneer deze op **SDCARD disk image,** is ingesteld, gebruikt de MEGA65 het virtuele D81-schijfimage dat in het veld **default disk image** is vermeld als opstartschijf. Wanneer u uw MEGA65 voor het eerst ontvangt, is dit ingesteld op de Intro Disk, genaamd `MEGA65.D81`. U kunt dit wijzigen naar een andere schijf. Om het automatisch koppelen uit te schakelen, wijzigt u de schijfnaam in een bestandsnaam die niet bestaat, of hernoemt u het bestand `MEGA65.D81` op de SD-kaart. (Indien u het veld leeg laat, wordt standaard `MEGA65.D81` gebruikt.)

Indien **F011 disk controller** op **3.5" floppy drive,** is ingesteld, pauzeert het opstartproces vlak vóór de READY-prompt om te controleren of er een opstartschijf in het station is geplaatst. Indien u geen fysieke opstartschijf gebruikt, kunt u deze op **SDCARD disk image** laten staan voor een sneller opstartproces.

Long FN support verwijst naar de ondersteuning van lange bestandsnamen in de bestandsbrowser van de MEGA65 voor SD-kaarten. Laat deze functie ingeschakeld, tenzij er problemen optreden bij het lezen van bestanden met bestandsnamen die langer zijn dan 11 tekens.

Video

Op de bladzijde **Video** stelt u de video-instellingen in. Dit zijn dezelfde instellingen als die van de initiële configuratie ("on-boarding"), waaronder de PAL- of NTSC-videomodus, het Digital Video-geluid en CRT-emulatie.



Met **Video mode** kiest u tussen de PAL-compatibiliteitsmodus en de NTSC-compatibiliteitsmodus. U kunt deze instelling ook tijdens het uitvoeren van programma's wijzigen met het Freezer-menu.

Met de instelling **Digital Video** schakelt u het gecombineerde video- en audiosignaal over de Digital Video-poort in of uit. Indien uw Digital Video-beeldscherm ingebouwde luidsprekers heeft, schakelt u deze instelling in (**Enhanced (with audio)**) om deze te gebruiken. Sommige DVI-beeldschermen zonder ingebouwde luidsprekers vereisen dat deze instelling wordt uitgeschakeld.

CRT emulation is een optionele instelling die het beeld meer laat lijken op dat van een klassiek beeldbuismonitor wanneer u een modern plat beeldscherm gebruikt.

Audio

Op de bladzijde **Audio** stelt u het geluidssysteem van de MEGA65 in. In de meeste gevallen kunt u deze instellingen op hun standaardwaarden laten staan.



Audio output kan worden ingesteld op volledige stereo of op het verzenden van een monauraal signaal naar beide luidsprekers. In de stereomodus kunt u de diverse audioapparaten van de MEGA65 met de audiomixer in het Freezer-menu naar links of rechts in het stereobeeld verplaatsen. De standaardinstellingen plaatsen de vier SID-chips enigszins naar links en rechts.

SID generation kiest tussen de audio-emulatie van de twee modellen SID-geluidschips: de oorspronkelijke 6581 die in sommige Commodore 64's wordt gebruikt en de nieuwere 8580 die in latere Commodore 64's en 128's wordt gebruikt. Sommige Commodore 64-spellen maakten gebruik van gebreken in de 6581 die in de 8580 zijn opgelost en klinken daardoor beter met de oudere generatie.

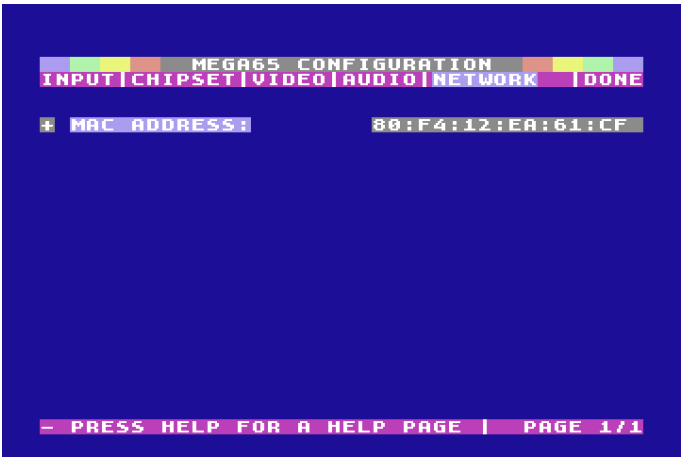
Swap stereo channels laat de linker- en rechterluidsprekers van elkaar verwisselen.

Audio amplifier regelt de ingebouwde versterker op de 3.5 mm-audioaansluiting. Stel deze in op **on** wanneer u een hoofdtelefoon of een ander apparaat gebruikt dat een versterkt signaal verwacht. Stel deze in op **off** voor een signaal op lijnniveau.

Network

Op de bladzijde **Network** kunt u het MAC-adres van de ethernetpoort aanpassen. De MEGA65 heeft geen door de hardware toegewezen MAC-adres. In plaats daarvan gebruikt de computer de hier ingevoerde waarde.

Indien het MAC-adres geheel uit nullen bestaat, drukt u op de toets **R** om een willekeurig adres te genereren. Met een MAC-adres dat geheel uit nullen bestaat, werken netwerkfuncties niet.



Done

Op de bladzijde **Done** kunt u het Configuratiehulpprogramma verlaten. Indien u wijzigingen hebt aangebracht die u wilt behouden, selecteert u **Save as defaults and exit.** U kunt de wijzigingen ook verwerpen, de fabrieksinstellingen herstellen of volledig opnieuw opstarten naar het scherm van de initiële configuratie.



Wanneer u het Configuratiehulpprogramma verlaat, wordt u gevraagd de computer uit en weer in te schakelen. Schakel de computer uit en vervolgens weer in.

KENNISMAKEN MET SD-KAARTEN

Uw MEGA65 is uitgerust met twee SD-kaartsleuven: een SD-kaartsleuf op vol formaat in de behuizing, bereikbaar via de onderzijde van de computer, en een microSD-kaartsleuf aan de achterzijde van de computer. De MEGA65 gebruikt de SD-kaart voor het opslaan van configuratie-instellingen, het laden van het besturingssysteem, het bijwerken van de firmware en het opslaan van uw programmatuur en gegevens als virtuele schijfimages.

De MEGA65 wordt geleverd met een SD-kaart op vol formaat in de interne SD-kaartsleuf, vooraf gevuld met de bestanden van het besturingssysteem en de meegeleverde programmatuur.[1] U kunt uw MEGA65 aansluiten en er direct mee beginnen, zonder een nieuwe SD-kaart in te stellen. U kunt deze SD-kaart op zijn plaats laten en doen alsof hij er niet is, alsof uw MEGA65 een computer uit de jaren negentig is met een verborgen mogelijkheid om niet-vluchtige gegevens op te slaan.

De MEGA65 gebruikt slechts één van de twee SD-kaartsleuven tegelijk. Indien er een microSD-kaart in de sleuf aan de achterzijde zit, wordt de interne SD-kaart genegeerd. Welke sleuf u gebruikt, hangt af van het beoogde gebruik van de computer. Naarmate u meer vertrouwd raakt met uw MEGA65, wilt u de SD-kaart wellicht tussen de MEGA65 en uw PC verplaatsen om bestanden te kopiëren en systeemupdates uit te voeren. Dit is gemakkelijker met de externe microSD-kaartsleuf.

U kunt uw MEGA65 ook met een ethernetkabel op uw PC of het plaatselijke netwerk aansluiten en met een hulpprogramma bestanden tussen de twee computers overbrengen. De bestandsoverdrachtfunctie heeft toegang tot de bestanden op de SD-kaart en gebruikt de sleuf die op dat moment actief is. Zie voor meer informatie over het overbrengen van bestanden hoofdstuk 8 op bladzijde 8-3.

EEN NIEUWE SD-KAART VOORBEREIDEN

U kunt de microSD-kaartsleuf aan de achterzijde van de MEGA65 gebruiken als permanente opslagruimte voor de configuratie- en systeembestanden van de computer. Een voorbereide kaart in deze sleuf heeft voorrang op de SD-kaart die in de computer is geïnstalleerd. Het is handig om een microSD-kaart geïnstalleerd te hebben indien u de kaart tussen uw MEGA65 en uw PC wilt verplaatsen.

De volgende aanwijzingen gelden voor geheugenkaarten in zowel de externe microSD-kaartsleuf als de interne SD-kaartsleuf op vol formaat.

De MEGA65 ondersteunt SD-kaarten van het type SDHC, met een grootte tussen 4GB en 32GB. Oudere kaarten kleiner dan 4GB en nieuwere SDXC-kaarten groter dan 32GB zullen naar verwachting niet werken.

[1] U kunt de oorspronkelijke inhoud van de SD-kaart opnieuw aanmaken met bestanden die u van het internet kunt downloaden. Desondanks wilt u wellicht een reservekopie van de inhoud van de SD-kaart op uw PC maken.

Een SD-kaart moet vóór gebruik door de MEGA65 worden voorbereid met behulp van het SD-kaarthulpprogramma. Het hulpprogramma maakt twee partities aan: een verborgen partitie voor configuratiegegevens en freezergegevens en een FAT32-compatibele partitie voor schijfimages en systeembestanden. U kunt toegang krijgen tot de FAT32-partitie door de SD-kaart op uw PC aan te sluiten.[2]

Een SD-kaart die door een andere computer is geformatteerd, werkt *niet* met de MEGA65, zelfs indien die computer alleen de FAT32-partitie wist. U *moet* de SD-kaart formatteren met het SD-kaarthulpprogramma van de MEGA65.

De SD-kaart plaatsen

Het formatteren van een SD-kaart wist de inhoud en deze handeling kan niet ongedaan worden gemaakt. Wij raden u aan de interne SD-kaart die bij de computer is geleverd niet te wissen.

Het SD-kaarthulpprogramma vraagt u welke van de kaarten die momenteel in de computer zijn geplaatst, geformatteerd moet worden. U kunt er als voorzorgsmaatregel voor kiezen de interne SD-kaart te verwijderen voordat u het SD-kaarthulpprogramma opent. U kunt hem later opnieuw plaatsen of hem uit de computer laten totdat u hem nodig hebt. Dit is tevens een goede gelegenheid om de gebundelde softwarebestanden (met bestandsnamen die eindigen op `.D81`) van de interne SD-kaart naar uw PC te kopiëren, zodat u ze later op de nieuwe SD-kaart kunt kopiëren.

Het hulpprogrammamenu is toegankelijk, zelfs indien er geen geldige SD-kaart aanwezig is. U kunt een nieuw systeem opstarten met uitsluitend een geschikte SD-kaart en het SD-kaarthulpprogramma.

Plaats de SD-kaart die u wilt voorbereiden, voordat u verdergaat.

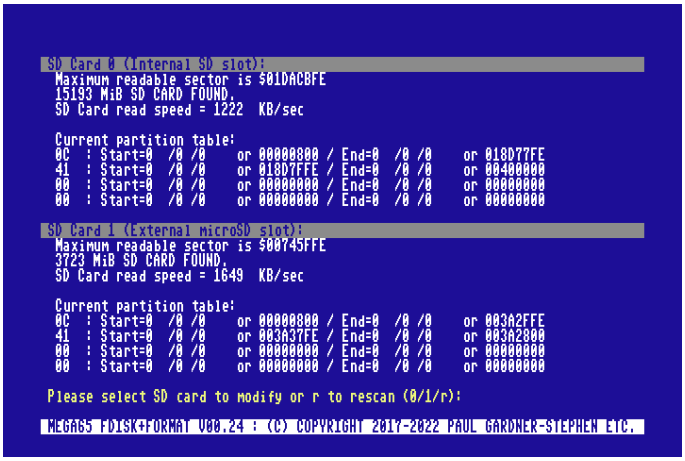
Het SD-kaarthulpprogramma

U opent het SD-kaarthulpprogramma vanuit het hulpprogrammamenu. Schakel de MEGA65 uit, houd de toets ALT ingedrukt en schakel hem weer in. Selecteer in het menu optie **2** om het SD-kaarthulpprogramma te starten (`SDCARD FDISK+FORMAT UTILITY`).

[2] Indien u een reservekopie van de volledige SD-kaart inclusief de verborgen partitie wilt maken, moet u een schijfhulpprogramma gebruiken dat volledige partities kopieert en niet alleen de bestanden op de FAT32-partitie.

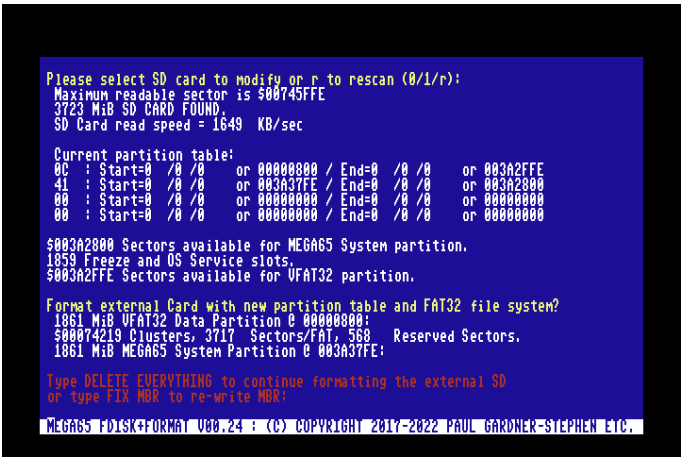


Het SD-kaarthulpprogramma opent en zoekt naar SD-kaarten die in de sleuven zijn geïnstalleerd. Indien u de SD-kaart die u wilt voorbereiden nog niet hebt geplaatst, doe dat nu en druk daarna op **R** om opnieuw te scannen.

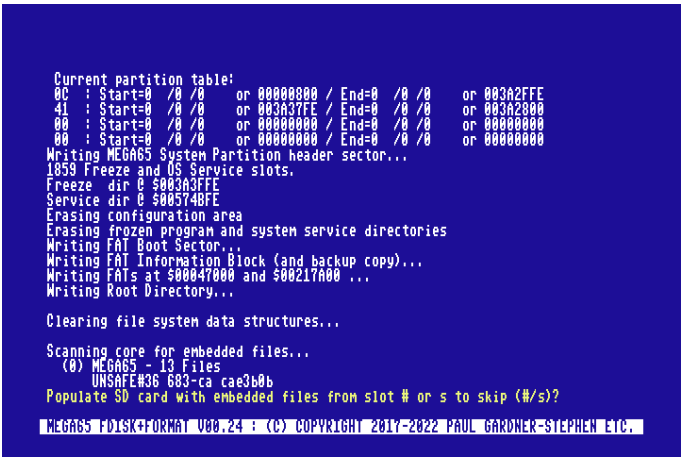


Selecteer de kaart die u wilt voorbereiden: **0** voor de interne SD-kaart, **1** voor de externe microSD-kaart. Indien er twee kaarten zijn geïnstalleerd, *let er goed op dat u de juiste kaartsleuf kiest*.

Het SD-kaarthulpprogramma vraagt bevestiging voordat de SD-kaart wordt gewist. Als allerlaatste voorzorgsmaatregel moet u de tekst `DELETE EVERYTHING` in hoofdletters typen en vervolgens op **RETURN** drukken om door te gaan. (Indien u dit proces wilt afbreken, is het veilig om de MEGA65 op dit moment uit te schakelen.)



Het hulpprogramma wist de SD-kaart en maakt de partities aan. Wanneer het klaar is, vraagt het of u de systeembestanden wilt installeren. De systeembestanden (met bestandsnamen die eindigen op .M65 en .ROM) zijn in de core ingebed en worden naar de FAT32-partitie gekopieerd voor gebruik. Indien u een bijgewerkte MEGA65-core in slot 1 hebt geïnstalleerd, selecteert u deze; selecteert u anders de in de fabriek geïnstalleerde MEGA65-core in slot 0. (Indien u uw MEGA65 zojuist hebt ontvangen, is slot 0 de enige optie.)



Start de computer opnieuw op wanneer daarom wordt gevraagd.[3]

[3] Indien u een core selecteert die MEGA65.ROM niet als één van de ingebedde bestanden bevat, vraagt het hulpprogramma u de SD-kaart naar uw PC te verplaatsen om dit bestand erop te kopiëren. Dit gebeurt alleen wanneer u een MEGA65-core gebruikt die afkomstig is van een andere bron dan een officieel MEGA65-releasespakket. Zie voor meer informatie over cores en het verkrijgen van MEGA65.ROM hoofdstuk 6 op bladzijde 6-5.

De gebundelde programmatuur verkrijgen

De systeembestanden die naar de vers geformatteerde SD-kaart worden gekopieerd, omvatten niet de gebundelde programmatuur die bij de oorspronkelijke SD-kaart in de interne kaartsleuf werd meegeleverd. U kunt deze bestanden met uw PC van de oorspronkelijke SD-kaart kopiëren en ze daarna op de nieuwe SD-kaart kopiëren.

De website MEGA65 Filehost biedt allerlei bestanden aan die u voor uw MEGA65 kunt downloaden. Dit omvat de nieuwste versies van de platformcomponenten, alternatieve cores en honderden spellen, demo's en toepassingen die door de MEGA65-gemeenschap zijn gemaakt. Dit omvat tevens de gebundelde programmatuur van de oorspronkelijke SD-kaart die bij de computer werd meegeleverd.

Indien u de gebundelde softwarebestanden niet meer hebt, kunt u ze verkrijgen via de website MEGA65 Filehost. Bezoek de volgende URL:

`https://files.mega65.org/?id=all-intros-public`

HOOFDSTUK 6

De MEGA65 upgraden

- Hoe u een MEGA65 kunt upgraden
- De versies van de onderdelen bepalen
- De nieuwste bestanden verkrijgen
- Het corekeuzemenu
- De core, ROM en systeembestanden upgraden
- Alternatieve cores en ROM's installeren
- Core-vlaggen instellen
- Een coreslot wissen
- De fabriekscore in slot 0 upgraden

- **Het opstartproces van de core begrijpen**

HOE U EEN MEGA65 KUNT UPGRADEN

Het MEGA65-platform bestaat uit drie belangrijke onderdelen:

1. De **MEGA65-core**, een beschrijving van de chipset die op de FPGA draait
2. De **ROM**, code die het besturingssysteem in Commodore-stijl (KERNAL) en BASIC definieert
3. **Systeemsoftware** voor functies zoals het Freezer-menu

U kunt deze onderdelen upgraden zodra er nieuwe versies verschijnen. U kunt ook één of meer van deze onderdelen afzonderlijk vervangen. Bij de core en de ROM kunt u zelfs meerdere versies tegelijk geïnstalleerd hebben en daartussen schakelen. In plaats van de nieuwste MEGA65-ROM kunt u bijvoorbeeld overschakelen naar de originele prototype-ROM van de Commodore 65. Of u kunt overschakelen naar een andere core die ervoor zorgt dat uw MEGA65-hardware zich geheel als een andere computer gedraagt, zoals een Commodore 64 of een ZX Spectrum.

De ROM en de systeemsoftware zijn bestanden die op de SD-kaart staan; het upgraden ervan is even eenvoudig als het vervangen van de bestanden. Om de core te upgraden installeert u via een bepaald proces een core-bestand in het core-flashgeheugen van de MEGA65. Dit hoofdstuk beschrijft dit proces.

Wat is een core?

De hardware-architectuur van de MEGA65 is gebaseerd op een veelzijdige chip die een "Field Programmable Gate Array" of FPGA wordt genoemd. Dit is een speciaal soort computerchip die geprogrammeerd kan worden om zich als andere chips voor te doen. Het doet dit door een enorme reeks logische poorten zodanig te configureren dat de poorten schakelingen nabootsen. FPGA's zijn geen emulatie, maar een elektronische nabootsing van andere chips. FPGA-code wordt soms aangeduid als *firmware*, een term die u wellicht herkent van moderne computers en andere apparaten.

Uw MEGA65 is in de fabriek geprogrammeerd om een chipset na te bootsen die is gebaseerd op de originele Commodore 65 en door het MEGA65-team is ontworpen. U kunt de FPGA van de MEGA65 opnieuw programmeren om te upgraden naar nieuwe versies van de MEGA65-chipset of om de chipset te vervangen door die van een heel andere computer!

Elke mogelijke chipset wordt een *core* genoemd. De MEGA65 kan maximaal acht cores opslaan en u kunt tussen deze cores schakelen via een menu dat opent wanneer u de computer inschakelt. U kunt dit menu ook gebruiken om een nieuwe core vanaf een bestand op de SD-kaart te laden, een proces dat *flashen* heet.

Leden van de MEGA65-gemeenschap hebben diverse nuttige en leuke alternatieve cores voor de FPGA-hardware gemaakt. *C64 voor MEGA65* van MJoergen en sy2002 bootst de originele Commodore 64-computer met een hoge mate van nauwkeurigheid na en is perfect voor het uitvoeren van Commodore 64-spellen, demo's en toepassingen. Andere cores bootsen de ZX Spectrum, de Game Boy en zelfs de hardware van

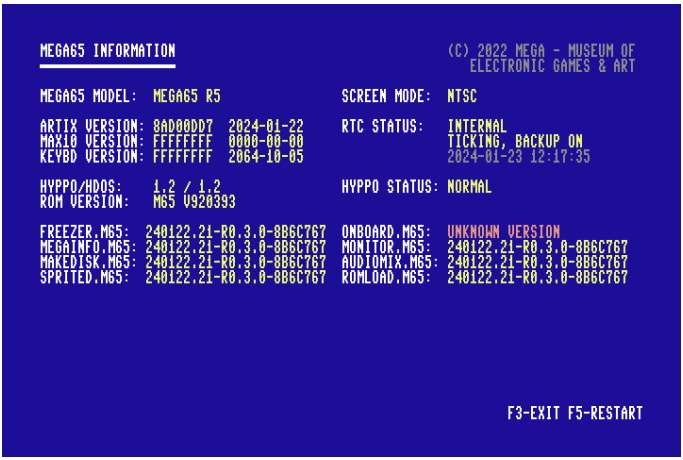
de originele Galaga-arcademachine na. De FPGA van de MEGA65 is krachtig genoeg om vrijwel alle 8-bit-homecomputers na te bootsen en waarschijnlijk ook enkele 16-bit-computers en spelcomputers, zoals de Commodore Amiga. Het hardware-ontwerp van de MEGA65, de indeling van het moederbord, de FPGA-core en overige informatie zijn allemaal gratis beschikbaar onder diverse openbronlicenties, zodat iedereen vrij is andere cores voor de MEGA65-hardware te maken.

DE VERSIES VAN DE ONDERDELEN BEPALEN

Alle onderdelen van het MEGA65-platform hebben een versieaanduiding. De MEGA65 kan de versieaanduidingen van al zijn onderdelen weergeven met het MEGA65-informatiehulpprogramma.

Om het MEGA65-informatiehulpprogramma te openen:

1. Schakel de MEGA65 in en laat hem opstarten tot BASIC.
2. Open de Freezer: houd RESTORE één seconde ingedrukt en laat hem daarna los.
3. Druk op HELP. Het MEGA65-informatiehulpprogramma opent.



Noteer deze versieaanduidingen:

Label en voorbeeld	Beschrijving
MEGA65-model `MEGA65 R6`	De revisie van de hardware. U moet dit weten wanneer u nieuwe core-bestanden downloadt.
Artix-versie `8AD00DD7 2024-01-22`	De MEGA65-core die op dit moment draait. Dit is een reeks van acht letters en cijfers en tevens een builddatum.
ROM-versie `M65 V920393`	De ROM die op dit moment draait. Voor MEGA65-ROM's is dit een oplopend nummer, waarbij grotere nummers nieuwere versies aanduiden.
Systeembestanden (.M65) `240122.21-R0.3.0-8B6C767`	Elk bestand van de systeemsoftware heeft een eigen versieaanduiding. Gewoonlijk hoeft u deze niet te kennen: u upgradet deze gelijktijdig met elke core. De aanduiding lijkt op de core-versie, doch komt niet altijd overeen met de core die op dat moment draait.

Druk op **F3** om terug te keren naar de Freezer en druk daarna nogmaals op **F3** om terug te keren naar BASIC.

Elke core heeft een afzonderlijke versie voor elke hardware-revisie. Vanaf het jaar 2024 gebruiken de productiemodellen van de MEGA65 twee verschillende moederbordrevisies, welke bekend staan als "R3" (meer specifiek "R3A") en "R6".[1]

De MEGA65-core is voor alle hardware-revisies beschikbaar. Indien u een alternatieve core installeert die niet beschikbaar is voor uw hardware-revisie, neem dan contact op met de auteur van de core.

DE NIEUWSTE BESTANDEN VERKRIJGEN

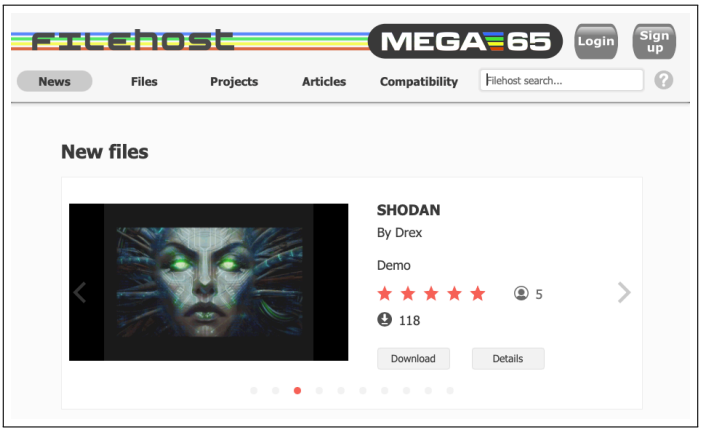
U kunt de nieuwste MEGA65-core, ROM en systeemsoftware downloaden van de website van de MEGA65-Filehost. Vanwege distributiebeperkingen voor de ROM-code van de Commodore 65 kunnen sommige bestanden alleen worden gedownload met een Filehost-account dat op naam van een MEGA65-eigenaar staat. Alle eigenaren van de MEGA65 hebben een licentie voor alle versies van deze ROM-code.[2]

Bezoek de volgende URL in uw webbrowser:

`https://files.mega65.org`

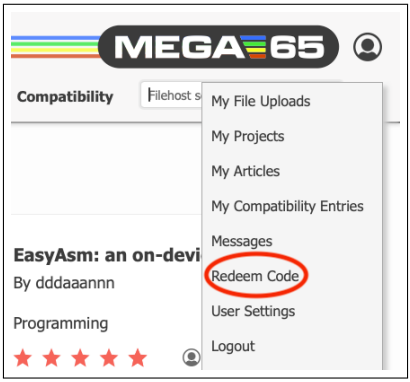
[1] Het MEGA65-"DevKit"-model dat in het jaar 2020 werd verkocht, is van revisie "R3". Het is tevens mogelijk de MEGA65-core op bepaalde FPGA-ontwikkelborden te laten draaien, met voor elk bord een afzonderlijke versie van het core-bestand.

[2] Er bestaat een procedure voor niet-eigenaren om de nieuwste MEGA65-ROM te verkrijgen, bijvoorbeeld om te gebruiken met de Xemu MEGA65 emulator. Deze procedure houdt in dat u C64 Forever Free Express Edition van Cloanto downloadt, het originele prototype-ROM-bestand van de Commodore 65 extraheert en vervolgens een hulpmiddel gebruikt om een patch toe te passen die u van Filehost kunt downloaden. Het volledige proces wordt beschreven in het volgende artikel: `https://mega65.org/rom-faq`



Om een Filehost-account met uw eigenaarcode te registreren:

1. Bezoek de Filehost-website. Klik op "Sign Up". Volg de aanwijzingen om een account aan te maken.
2. Zoek uw eigenaarcode op. Dit is een code die op een papiertje staat afgedrukt; het papiertje was bij uw MEGA65 inbegrepen (mogelijk is het in deze handleiding gestoken). De code ziet er ongeveer als volgt uit: `123-ABC-456`
3. Klik in de rechterbovenhoek van het Filehost-scherm op het gebruikerspictogram. Selecteer in het pop-upmenu "Redeem Code". Voer uw eigenaarcode in zoals gevraagd.



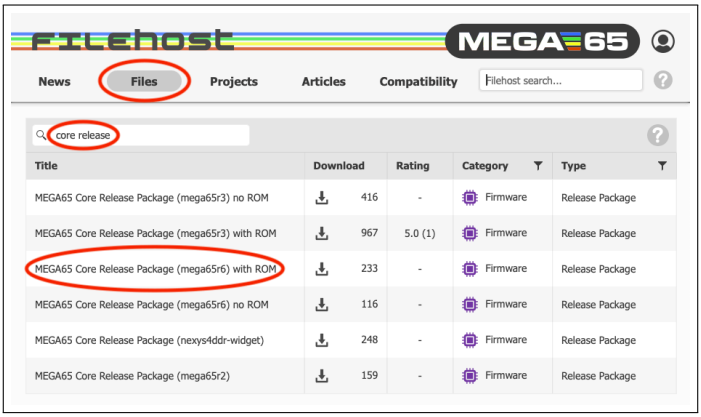
Om het nieuwste releasepakket te downloaden:

1. Klik op het tabblad "Files" van de Filehost-website.
2. Typ in het zoekvak aan de linkerkant: "core release". De lijst wordt bijgewerkt en toont vervolgens alleen bestanden met deze woorden in de titel.

3. Zoek de vermelding genaamd “MEGA65 Core Release Package (mega65r6) with ROM”, waarbij “mega65r6” overeenkomt met uw hardware-revisie. (Om uw hardware-revisie te bevestigen opent u het Freezer-menu en drukt u daarna op HELP.)

4. Klik op de vermelding. Bevestig dat dit releasepakket voor uw hardware-revisie is en klik daarna op “Download” om het bestand te downloaden.

OPMERKING: Er bestaat een vermelding van het releasepakket zonder de ROM. Deze vermelding is voor iedereen zichtbaar. Om er zeker van te zijn dat u een compatibele set bestanden gebruikt, kiest u het pakket waarop “with ROM” staat. Indien u geen vermelding ziet waarop “with ROM” staat, controleer dan of u bent aangemeld en of u een geldige eigenaarcode heeft ingewisseld.



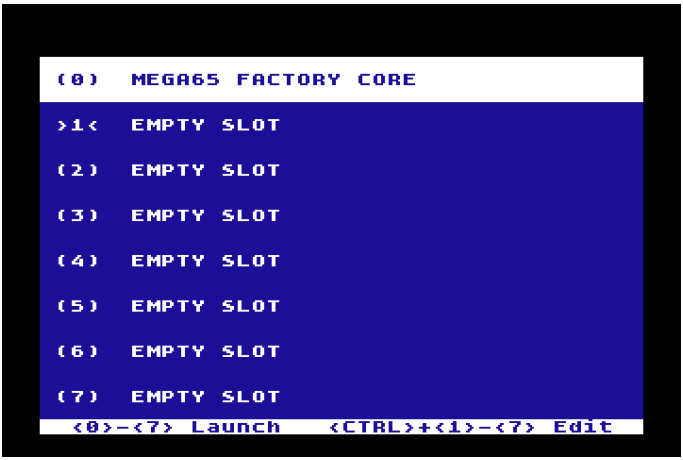
Pak het gedownloade `.7z`-archief uit. U zou een bestand moeten zien waarvan de naam eindigt op `.cor` en een map `sdcard-files` die een bestand bevat met de naam `MEGA65.ROM`.

HET COREKEUZEMENU

De MEGA65 bepaalt welke core bij het opstarten in de FPGA wordt geladen. U kunt dit proces onderbreken om te kiezen welke core wordt geladen.[3]

Om het corekeuzemenu te openen schakelt u de computer uit, houdt u de toets NO SCROLL ingedrukt en schakelt u de computer weer in. Het corekeuzemenu verschijnt, met de acht coreslots, genummerd van 0 tot en met 7.

[3] Strikt genomen start de MEGA65 de core in slot 0 op om het corekeuzemenu te laten draaien. Nadat u een keuze hebt gemaakt of de computer een standaardcore kiest, laadt de computer de gekozen core in de FPGA en vervolgt hij het opstartproces.



U kunt met de cursortoetsen en RETURN een core kiezen om op te starten, of u kunt eenvoudigweg de cijfertoets indrukken die overeenkomt met het slot. Het opstartproces wordt voortgezet met de nieuwe core. De MEGA65 blijft de nieuwe core gebruiken totdat u de computer uitschakelt. (Het indrukken van de resetknop wijzigt niet welke core wordt gebruikt.)

Wanneer u de computer inschakelt zonder het corekeuzemenu te openen, zoekt de MEGA65 naar een standaardcore. De computer controleert eerst of een core als standaardcore is gemarkeerd (een instelling die u kunt wijzigen). Indien geen enkele core zo is gemarkeerd, controleert de computer daarna of er een core in slot 1 staat.[4] Indien het slot leeg is, gebruikt de computer slot 0.

Uw computer is in de fabriek uitgerust met de MEGA65-core in slot 0. Wij raden u aan de in de fabriek geïnstalleerde core niet te upgraden, tenzij het MEGA65-team u dat adviseert. Installeer nieuwe versies van de MEGA65-core in plaats daarvan in slot 1.

DE CORE, ROM EN SYSTEEMBESTANDEN UPGRADEN

Vanuit het corekeuzemenu kunt u een core upgraden of een nieuwe core installeren. Tijdens dit proces leest de computer het bestand `.cor` van de SD-kaart.

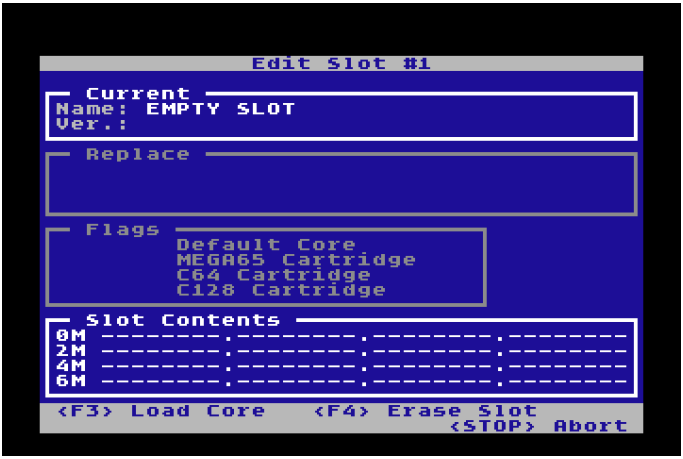
Om de MEGA65-core, ROM en systeembestanden te upgraden:

[4] Indien DIP-schakelaar #4 in de stand "on" staat, controleert de MEGA65 slot 2 in plaats van slot 1. De DIP-schakelaars bevinden zich in de behuizing, op het moederbord.

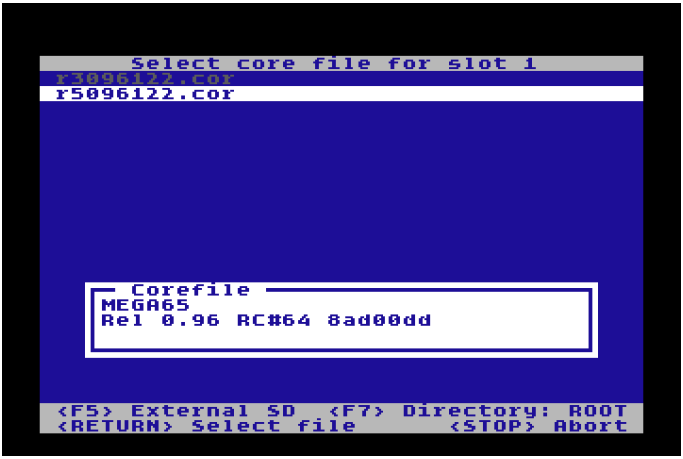
1. Verwijder de SD-kaart (of microSD-kaart) uit de MEGA65 en sluit deze met een SD-kaartlezer aan op uw PC.[5]
2. Kopieer het `.cor`-bestand dat u uit het `.7z`-archief hebt gehaald naar de SD-kaart.
3. Open op uw PC de map `sdcard-files` uit het `.7z`-archief en kopieer deze bestanden daarna naar de SD-kaart, waarbij u de bestaande bestanden vervangt. Plaats ze in de hoofdmap van het bestandssysteem van de SD-kaart, niet in een submap.
4. Werp de SD-kaart uit via het besturingssysteem van uw PC en plaats deze daarna terug in de MEGA65.
5. Open het corekeuzemenu: schakel de MEGA65 uit en houd daarna NO SCROLL ingedrukt terwijl u de computer weer inschakelt.
6. Houd CTRL ingedrukt en druk daarna op het cijfer van het slot dat u wilt upgraden. De sloteditor opent.
7. Druk op F3 om een core-bestand te laden. De bestandsselectie opent. Gebruik de cursortoetsen om het `.cor`-bestand te selecteren en druk daarna op RETURN.
8. Druk op F8 om het coreslot te flashen. Laat het flashen volledig verlopen. Dit duurt enkele minuten.
9. Wanneer het flashen is voltooid, drukt u op een willekeurige toets om terug te keren naar het corekeuzemenu.

De sloteditor ziet er als volgt uit:

[5] In plaats van de SD-kaart naar uw PC over te brengen kunt u de bestanden ook via een Ethernetverbinding overzetten. Zie hoofdstuk 8 op bladzijde 8-3.



Wanneer u een core-bestand laadt, vraagt de computer u het .cor-bestand te selecteren op een scherm dat op dit voorbeeld lijkt:

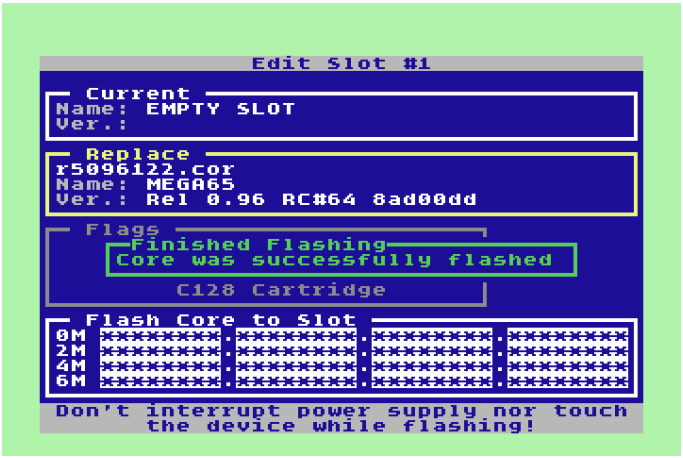


Nadat u de core hebt geselecteerd, toont de sloteditor welke wijziging deze in het slot wil aanbrengen. Wanneer u het flashen start, toont het beeldscherm de voortgang.

OPMERKING: Schakel uw computer *niet* uit en onderbreek de stroomtoevoer niet voordat deze stap is voltooid.



Wanneer de melding "Core was successfully flashed" wordt weergegeven, is het proces voltooid.



Het is nu veilig om uw computer uit te schakelen. Druk op een willekeurige toets om terug te keren naar het corekeuzemenu, of schakel de computer uit en daarna weer in om de standaardcore te starten.

ALTERNATIEVE CORES EN ROM'S INSTALLEREN

Het installeren van een alternatieve core, zoals de C64-core, verloopt volgens dezelfde stappen voor het flashen van de core naar een slot.

Het is raadzaam de slots 2 tot en met 7 voor alternatieve cores te gebruiken en slot 1 voor de nieuwste MEGA65-core te reserveren. Er staat u uiteraard niets in de weg om een alternatieve core in slot 1 te installeren, zodat de MEGA65 zich bij het inschakelen als een ander type computer gedraagt. De MEGA65-core kunt u altijd kiezen via het corekeuzemenu.

U kunt meer dan één versie van de MEGA65-ROM op de SD-kaart bewaren. Wanneer u de MEGA65-core opstart, kunt u één van deze ROM's selecteren door tijdens het opstarten een cijfertoets ingedrukt te houden.

Om alternatieve ROM's te installeren kopieert u ze naar de hoofdmap van de SD-kaart, met een bestandsnaam zoals `MEGA65x.ROM`, waarbij x een getal tussen 0 en 8 is. Om de alternatieve ROM op te starten houdt u de bijbehorende cijfertoets ingedrukt terwijl de MEGA65-core start. Indien u geen cijfer ingedrukt houdt, start de computer standaard `MEGA65.ROM` op.

Er zijn verschillende redenen waarom u alternatieve ROM's op uw SD-kaart zou willen bewaren:

- U helpt een nieuwe bètaversie van de ROM te testen en u wilt de bètaversie niet tot uw standaard-ROM maken.
- U wilt de originele prototype-ROM van de Commodore 65 proberen. De MEGA65-core behoudt de achterwaartse compatibiliteit met de C65-ROM waaraan Commodore werkte voordat het bedrijf het project annuleerde. Deze ROM is gebrekkig en onvolledig, maar vormt nog steeds een interessant historisch artefact.
- U wilt een alternatieve ROM proberen die door de MEGA65-gemeenschap is ontwikkeld. Eén van deze ROM's is de MEGA65 OpenROM, een project om een volledig nieuwe ROM te maken die onder een open-sourcelicentie wordt uitgebracht, zonder enig oorspronkelijk Commodore-materiaal.

Bij uw MEGA65-SD-kaart werden diverse alternatieve ROM's meegeleverd, die in de fabriek zijn geïnstalleerd. Probeer uw computer opnieuw op te starten terwijl u een cijfertoets ingedrukt houdt en kijk wat er gebeurt!

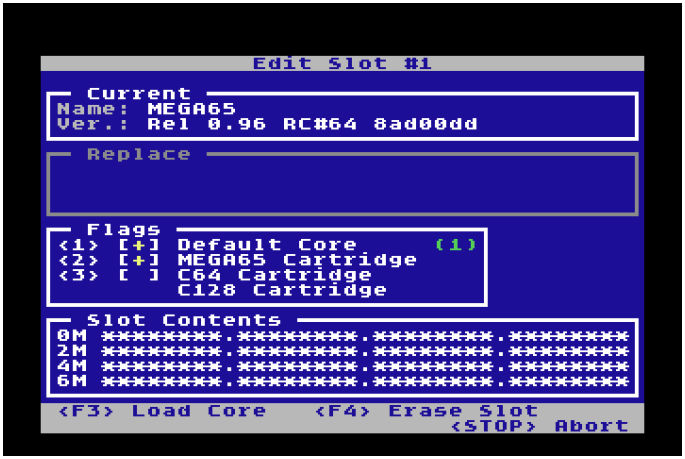
CORE-VLAGGEN INSTELLEN

Er zijn verschillende opties ("vlaggen") die u voor een core in de core-editor kunt kiezen.

Om vlaggen voor een core te wijzigen bewerkt u het coreslot. Druk op de cijfertoets die overeenkomt met de vlag om de waarde ervan om te schakelen. Sla de vlaggen in het slot op door het resultaat te flashen: druk op **F8**. U kunt de vlaggen vóór het flashen van nieuwe coregegevens instellen, of u kunt alleen de nieuwe vlaginstellingen flashen zonder de coregegevens te vervangen.

Om een core in te stellen als de standaardcore, welke wordt gebruikt wanneer de MEGA65 wordt ingeschakeld zonder NO SCROLL ingedrukt te houden, stelt u de vlag "Default core" in. Indien geen enkele core als standaardcore is ingesteld, wordt slot 1 als standaard gebruikt (of slot 2 indien DIP-schakelaar #4 in de stand "on" staat).

De vlaggen "cartridge" bepalen welke core wordt geselecteerd wanneer er een cartridge in de expansiepoort aanwezig is. Zo kunt u op basis van het type cartridge een andere standaardcore kiezen. U kunt bijvoorbeeld de MEGA65-core instellen om MEGA65-cartridges af te handelen en een andere core om C64-cartridges af te handelen. Standaard handelt de MEGA65-core C64-cartridges af met behulp van de "GO64"-modus. Wellicht geeft u er de voorkeur aan dit te wijzigen en de C64-core te gebruiken die u afzonderlijk installeert.



EEN CORESLOT WISSEN

Het flashen vervangt wat er in een coreslot staat door de nieuwe core. Indien er al een core in het slot is geïnstalleerd, overschrijft het flashen de bestaande core.

Indien u een core wilt verwijderen en het slot leeg wilt laten, bewerkt u het slot en drukt u op F4 om de vervanging in te stellen op "Erase slot". Druk daarna op F8 om het slot met lege gegevens te flashen.

DE FABRIEKSCORE IN SLOT 0 UPGRADEN

Het is mogelijk de in de fabriek geïnstalleerde MEGA65-core in slot 0 te upgraden. Dit hoeft u alleen in zeldzame gevallen te doen, bijvoorbeeld indien een nieuwere versie van de MEGA65-core wijzigingen of foutcorrecties voor het opstartproces bevat. Het is raadzaam slot 0 niet te upgraden, tenzij de aankondiging van de uitgave daartoe aanleiding geeft. De meeste upgrades van de MEGA65-core zijn in slot 1 volledig functioneel, zonder dat u slot 0 hoeft te upgraden.

Het is belangrijk dat ten minste één coreslot een werkende MEGA65-core bevat. Indien er tijdens het flashen iets misgaat, kan dit leiden tot een niet-werkende core in dat slot. Om ongelukken te helpen voorkomen verschilt de procedure voor het flashen van slot 0 enigszins van die van de andere slots en kan alleen een officiële MEGA65-core naar slot 0 worden geflasht.

Lees deze aanwijzingen zorgvuldig door voordat u met de procedure begint. Om de core in slot 0 te upgraden:

1. Installeer de nieuwste MEGA65-core in slot 1 met behulp van de hierboven beschreven procedure. De core moet zich in het standaardslot ongelijk aan nul bevinden om van eventuele problemen bij het upgraden van slot 0 te kunnen herstellen.
2. Start coreslot 1 op om te bevestigen dat het werkt.
3. Keer terug naar het corekeuzemenu: schakel de MEGA65 uit en houd daarna NO SCROLL ingedrukt terwijl u de computer weer inschakelt.
4. Houd ingedrukt en druk op de komma-toets om de editor voor slot 0 te openen.
5. Lees het informatiescherm en typ daarna het woord `CONFIRM` in hoofdletters en druk op RETURN.
6. Herhaal de rest van de flashprocedure om het core-bestand te selecteren en het slot te flashen.

OPMERKING: Indien u een MEGA65 van revisie R3A heeft en slot 0 nog niet eerder heeft geüpgraded, starten en de komma-toets de procedure niet: u heeft een oudere core in slot 0 die deze voorziening niet heeft. U kunt dit omzeilen door het corekeuzemenu opnieuw te starten met slot 1. Bereid u vanuit het corekeuzemenu voor om NO SCROLL ingedrukt te houden, druk op 1 om de core op te starten en druk daarna onmiddellijk op NO SCROLL en houd deze ingedrukt. Het corekeuzemenu wordt

daarna opnieuw geopend met behulp van slot 1. Druk op  en de komma-toets om de upgrade van slot 0 te voltooien.

Indien er tijdens het flashen van slot 0 iets misgaat, start uw MEGA65 mogelijk niet correct op. Schakel uw MEGA65 vóór al het andere in en wacht ongeveer een minuut. Na enige tijd zou de computer moeten merken dat er geen geldige core in slot 0 staat en vervolgens de core in slot 1 starten. U kunt tijdens dit proces 

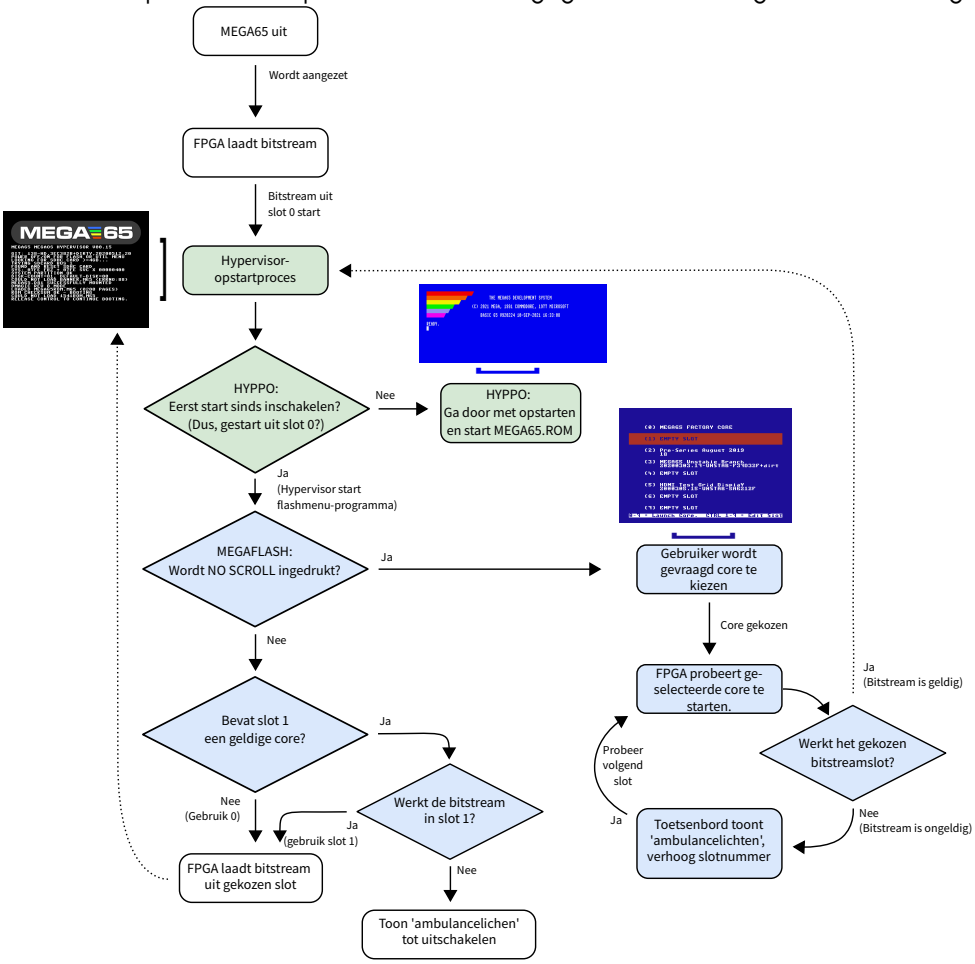
ingedrukt houden om het corekeuzemenu van slot 1 te openen en het flashen opnieuw te starten.

Indien de MEGA65 na enkele minuten nog geen enkele core kan opstarten, is de computer mogelijk vastgelopen. U kunt de computer wellicht herstellen met behulp van een apparaat dat een "JTAG-interface" heet en uw PC met het moederbord van de MEGA65 verbindt. Hiermee kunt u een bitstream rechtstreeks in de FPGA injecteren. Het onderdeel is goedkoop, doch niet altijd verkrijgbaar. Neem voor hulp contact op met het MEGA65-team via de Discord (https://mega65.org/chat).

Aan coreslot 0 kunnen geen vlaggen worden toegewezen, zoals een vlag om als standaardcore te dienen of om aan cartridgetypen te worden gekoppeld. Slot 0 wordt voor deze doeleinden gebruikt indien er geen andere core is geïnstalleerd. Het is raadzaam de nieuwste MEGA65-core in slot 1 te bewaren, naast het flashen van slot 0.

HET OPSTARTPROCES VAN DE CORE BEGRIJPEN

Deze sectie vat samen hoe de MEGA65 kiest met welke core hij bij het inschakelen opstart. Het proces wordt weergegeven in de volgende afbeelding:



Het opstartproces wordt beheerd door twee voorzieningen:

- De Hypervisor (ook bekend als HYPPO), welke op een niveau boven de KERNAL werkt. Eén van zijn taken is het beheren van delen van het opstartproces. Raadpleeg voor meer details over de Hypervisor Hoofdstuk/Bijlage K op bladzijde K-

23. In het diagram zijn de activiteiten die de Hypervisor uitvoert groen gemarkeerd.

- Het corekeuzemenuprogramma (ook bekend als "MegaFlash"), welke een lijst met beschikbare coreslots biedt om uit te kiezen. In het diagram zijn de activiteiten die MegaFlash uitvoert blauw gemarkeerd.

Wanneer de MEGA65 wordt ingeschakeld, doet de computer het volgende:

- De MEGA65 laadt de bitstream in slot 0 van het flashgeheugen. Indien dat de MEGA65-fabriekscore is, start de HYPPO-hypervisor van de MEGA65.
- Indien het de eerste opstart na het inschakelen is (wat betekent dat u vanuit slot 0 werkt), start HYPPO het Flash-menu-programma (ook wel MegaFlash genoemd) – maar houd er rekening mee dat het Flash-menu in deze modus mogelijk niets op het scherm toont om aan te geven dat het draait!
- Het Flash-menu controleert vervolgens of NO SCROLL ingedrukt wordt gehouden.
- Indien dat het geval is, toont het Flash-menu zijn scherm, zodat u een core kunt selecteren of opnieuw flashen.
- Indien NO SCROLL *niet* wordt ingedrukt gehouden, controleert het Flash-menu of Flash-slot 1 een geldige core bevat.
- Indien dat zo is, probeert het Flash-menu die core te laden.
- Indien dit lukt, herconfigureert het systeem zichzelf voor die core, waarna het gedrag van het systeem overeenkomt met die core.
- Indien dit mislukt, schakelt het toetsenbord over naar de "ambulancemodus", waarbij blauwe lampjes knipperen om aan te geven dat er eerste hulp nodig is. Merk op dat de resetknop in de ambulancemodus geen effect heeft: u moet de MEGA65 uitschakelen en weer inschakelen.

Indien u in het corekeuzemenu een andere core hebt geselecteerd, verloopt het proces op vergelijkbare wijze, met dit verschil dat de ambulancelampjes slechts gedurende een beperkte tijd branden: de FPGA zoekt namelijk automatisch door het flashgeheugen totdat hij een geldige core vindt. Indien hij het einde van het flashgeheugen bereikt, start hij de MEGA65-fabriekscore opnieuw vanuit slot 0.

HOOFDSTUK 7

Werken met schijven en schijfimages

- Schijfstations
- Werken met virtuele schijfimages
- Het ingebouwde 3.5″-diskettestation gebruiken
- Externe IEC-schijfstations gebruiken
- Opstartbare schijven
- De SD-kaart benaderen vanuit BASIC
- Gangbare schijfbewerkingen

SCHIJFSTATIONS

De MEGA65 beschikt over een ingebouwd 3.5"-diskettestation en ondersteunt externe schijfstations in Commodore-stijl via de seriële IEC-poort aan de achterzijde van de computer. De IEC-poort ondersteunt ook andere externe IEC-opslagapparaten, zoals de SD2IEC. Sommige IEC-opslagapparaten kunt u in een keten aansluiten en tegelijkertijd gebruiken.

De MEGA65 bevat ook een "virtueel" schijfstation dat D81- of D64-schijfimagebestanden op de SD-kaart kan aankoppelen. De meeste MEGA65-programmatuur die u van het internet downloadt, heeft de vorm van een D81-schijfimage. Programma's voor de Commodore 64 worden vaak verspreid in de vorm van een D64-schijfimage. U kunt rechtstreeks vanaf de MEGA65 een nieuw D81-schijfimage aanmaken en uw BASIC-programma's op de SD-kaart opslaan, zonder dat u extra hardware nodig hebt. U kunt ook bestanden kopiëren tussen fysieke floppy-schijven en schijfimages.

Het Intro Disk-menu dat u zag toen u de computer voor het eerst inschakelde, is een programma op een D81-schijfimage: een bestand met de naam `MEGA65.D81` op de SD-kaart. De MEGA65 is in eerste instantie geconfigureerd om dit schijfimage automatisch op te starten. U kunt dit wijzigen in het Configuratiehulpprogramma. (Zie ook hoofdstuk 5 op bladzijde 5-3.)

U kunt schijfstations en virtuele schijfimages vanuit het Freezer-menu beheren. Sommige van deze handelingen kunt u uitvoeren met BASIC-commando's zoals **MOUNT**.

Unitnummers en drivenummers

Elk schijfstation (fysiek of virtueel) wordt benaderd via een *unit*nummer. Bij de klassieke Commodore-computers verwijst het unitnummer naar een IEC-apparaat dat op de computer is aangesloten. Commodore reserveerde de unitnummers in het bereik 0 - 31 voor apparaten met uiteenlopende doeleinden, waarvan 8 - 11 voor schijfstations. Hebt u ooit een Commodore 64 gebruikt en `LOAD "*",8,1` ingetypt, dan verwijst de "8" naar het schijfstation dat als unit 8 is aangesloten. De schijfcommando's van BASIC 65 gebruiken standaardmatig unit 8 en aanvaarden een **U**-parameter om dit te wijzigen, bijvoorbeeld: `DLOAD "MYPROGRAM",U9`

Bij de MEGA65 kunt u een unitnummer toewijzen aan het virtuele schijfstation met een gekoppeld schijfimage, of aan het ingebouwde 3.5"-diskettestation. U moet een schijfimage of het ingebouwde 3.5"-diskettestation aan een unitnummer koppelen voordat het gebruikt kan worden. Elk bericht dat naar een unitnummer wordt gestuurd dat aan een virtuele schijf of het ingebouwde diskettestation is toegewezen, wordt door de MEGA65 afgehandeld. Alle overige berichten worden naar de seriële IEC-poort gestuurd.

Schijfcommando's aanvaarden ook een optionele parameter om een *drive*nummer op te geven. Deze is alleen nodig bij het aansluiten van een vintage dubbel diskettestation via de IEC-poort, zoals de Commodore 4040, 8050 of 8250. Elk schijfstation

kent drivenummer 0 toe aan het eerste station. Stations met twee schijven kennen drivenummer 1 toe aan het tweede station. Dubbele schijfstations zijn gewoonlijk uitgerust met een IEEE-488-interface en hebben een IEEE-488-naar-IEC-omzetter nodig om op de MEGA65 gebruikt te kunnen worden. De schijfcommando's van BASIC 65 gebruiken standaardmatig drive 0 en aanvaarden een **D**-parameter om dit te wijzigen.

WERKEN MET VIRTUELE SCHIJFIMAGES

De MEGA65 biedt twee "beheerde stations" die de via de IEC-poort aangesloten stations aanvullen. Aan het eerste beheerde station kunt u een D81- of D64-schijfimagebestand op de SD-kaart toewijzen, of u kunt het toewijzen aan het ingebouwde 3.5"-diskettestation. Aan het tweede beheerde station kunt u eveneens een D81- of D64-schijfimagebestand toewijzen, zodat er maximaal twee virtuele schijven tegelijkertijd gekoppeld kunnen worden.[1]

Het eerste beheerde station kan worden ingesteld op unit 8 of 10 en het tweede beheerde station op unit 9 of 11.

Waar u schijfimagebestanden kunt verkrijgen

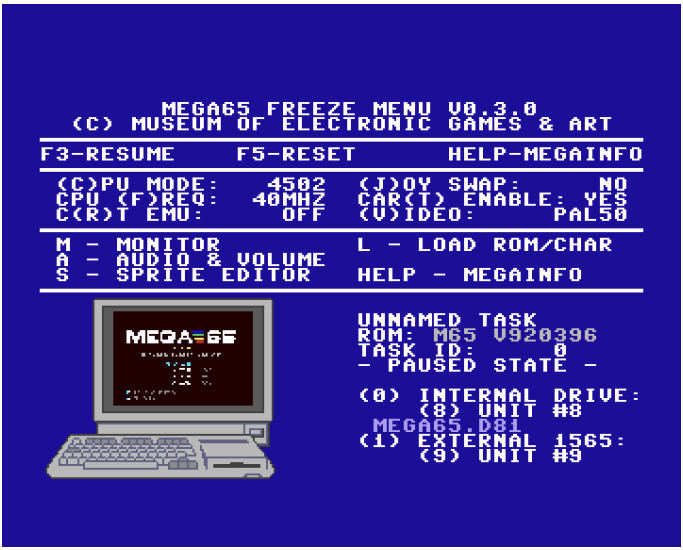
De website MEGA65 Filehost herbergt een bibliotheek met MEGA65-programmatuur die door de gemeenschap is gemaakt. U kunt door de programmatuur bladeren of erin zoeken, een titel downloaden en het schijfimage vervolgens met uw pc of met het Ethernet-bestandsoverdrachtshulpprogramma naar de SD-kaart kopiëren.

https://files.mega65.org/

Schijfimages aankoppelen met de Freezer

Open het Freezer-menu: houd RESTORE één seconde ingedrukt en laat de toets vervolgens los. Let op de huidige toestand van aangekoppelde stations rechtsonder op het scherm.

[1] Commodore was oorspronkelijk van plan een nieuw extern 3.5"-diskettestation uit te brengen, het "1565"-station, dat bij de Commodore 65 zou horen en op een speciale niet-IEC-poort zou worden aangesloten. Het MEGA65-project heeft de ambitie om zo'n station ooit te produceren; indien dat gebeurt, zou dit station aan het tweede beheerde station worden toegewezen.



Om een schijfimage aan unit 8 of 10 te koppelen, selecteert u het eerste beheerde station door op **0** te drukken. Om een schijfimage aan unit 9 of 11 te koppelen, selecteert u het tweede beheerde station door op **1** te drukken. Dit opent de bestandsbrowser van de SD-kaart.



Gebruik de cursortoetsen om een schijfimage te selecteren en druk vervolgens op **RETURN**. Het Freezer-scherm toont dat het geselecteerde schijfimage nu aan het beheerde station is gekoppeld.

Druk vanaf het hoofdscherm van de Freezer op **8** of **9** om te wisselen tussen de unitnummers van respectievelijk het eerste en het tweede beheerde station.

Schijfimages aankoppelen vanuit BASIC

Met het BASIC-commando **MOUNT** kunt u een schijfimage vanaf de SD-kaart aankoppelen zonder de Freezer te hoeven openen. U kunt dit commando bij de READY-prompt invoeren of het als onderdeel van een programma gebruiken.

Om een schijfimage aan unit 8 te koppelen, voert u **MOUNT** in met de volledige bestandsnaam tussen dubbele aanhalingstekens, inclusief het achtervoegsel .D81 of .D64:

```
MOUNT "MEGA65.D81"
```

Om een schijfimage aan unit 9 te koppelen, geeft u het **U**-argument op:

```
MOUNT "MEGA65.D81",U9
```

Een nieuw schijfimage aanmaken

U kunt vanuit de Freezer van de MEGA65 een nieuw leeg D81-schijfimage aanmaken.

1. Open de Freezer.
2. Druk op **0** om het eerste beheerde station te selecteren.
3. Selecteer boven in de bestandenlijst: - NEW D81 DD IMAGE -
4. Voer, wanneer u daarom gevraagd wordt, een naam voor de schijf in. (Laat het achtervoegsel .D81 weg; dit wordt automatisch toegevoegd.)

Het nieuwe schijfimage wordt op de SD-kaart aangemaakt en aan het eerste beheerde station gekoppeld. Het is geformatteerd en gereed voor gebruik.

SD-kaartbestanden beheren in submappen

Wanneer u enige tijd op Filehost games en toepassingen hebt gedownload, zult u uiteindelijk een grote verzameling schijfimages op uw SD-kaart hebben. Mogelijk wilt u deze bestanden in submappen ordenen. U kunt deze mappen aanmaken terwijl de SD-kaart op uw pc is aangesloten, of met het Ethernet-bestandsoverdrachtshulpprogramma.

De Freezer ondersteunt submappen in de bestandsbrowser. Elke submapnaam begint met een schuine streep (/). Selecteer een map om de bestanden ervan weer te geven. Om terug te keren naar de vorige map, selecteert u: /..

De MEGA65 houdt een "huidige werkmap" bij die als basismap dient voor BASIC-commando's zoals **MOUNT**. Om de huidige werkmap vanuit BASIC te wijzigen, gebruikt u het **CHDIR**-commando met het **U12**-argument:

```
CHDIR "DEMOS",U12

MOUNT "XANADU.D81"
```

OPMERKING: De ondersteuning voor submappen op de SD-kaart is nog in ontwikkeling. Indien een schijfimage in een submap is gekoppeld, wordt het ontkoppeld door elke handeling die de huidige werkmap wijzigt. Sommige voorzieningen die bestanden gebruiken, ondersteunen mogelijk geen bestanden in submappen. Het is momenteel niet mogelijk om vanuit de Freezer een nieuw schijfimage in een submap aan te maken. Wij hopen dit in een toekomstige update te verbeteren.

HET INGEBOUWDE 3.5"-DISKETTESTATION GEBRUIKEN

De MEGA65 beschikt over een ingebouwd 3.5"-diskettestation, vergelijkbaar met wat voor de Commodore 65 bedoeld was. U kunt fysieke floppy-schijven gebruiken om uw programma's en gegevens op te slaan. Sommige MEGA65-programmatuur kan op diskette worden aangeschaft.

Het ingebouwde 3.5"-diskettestation moet gekoppeld worden voordat het gebruikt kan worden. Het kan aan unit 8 of unit 10 worden gekoppeld, in het eerste beheerde station.

Het ingebouwde 3.5"-station aankoppelen met de Freezer

Open het Freezer-menu: houd RESTORE één seconde ingedrukt en laat de toets vervolgens los. Let op de huidige instellingen voor het aankoppelen van stations rechtsonder op het scherm.

Druk op 0 en gebruik vervolgens de pijltjestoets omlaag om - INTERNAL 3.5" - te selecteren. Druk op RETURN om de selectie te bevestigen. Het Freezer-menu toont dat het interne station aan het eerste beheerde schijfstation is gekoppeld.

Het UNIT # voor het eerste station kan 8 of 10 zijn. Druk op 8 om tussen deze opties te wisselen. De schijfcommando's van BASIC 65 gebruiken standaard unit 8, dus het is gebruikelijk unit 8 te gebruiken tenzij u tegelijkertijd met meerdere schijven werkt.

Het ingebouwde 3.5"-diskettestation kan alleen in het eerste beheerde station worden gekoppeld, met unitnummers 8 of 10. Het kan niet in het tweede beheerde station worden gekoppeld (unitnummers 9 of 11).

Het ingebouwde 3.5"-station aankoppelen vanuit BASIC

U kunt het ingebouwde 3.5"-diskettestation met het BASIC-commando **MOUNT** aan unit 8 koppelen. Dit commando werkt zowel vanaf de READY-prompt als vanuit een programma. Om het interne station aan unit 8 te koppelen, voert u het commando zonder argumenten in:

```
MOUNT
```

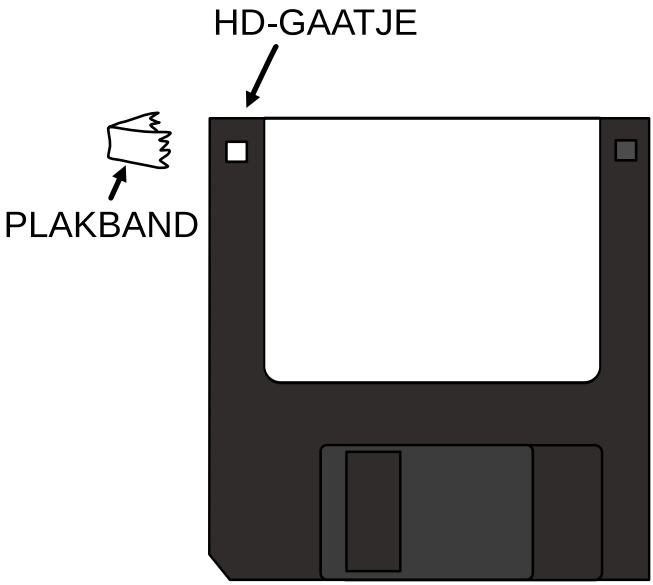
Het **MOUNT**-commando kan het interne station alleen aan unit 8 koppelen. Vanuit het Freezer-menu kunt u het alleen aan unit 10 koppelen.

DD- en HD-schijven

De schijfcontroller van de MEGA65 verwacht een diskette met Double Density (DD) in het ingebouwde 3.5"-diskettestation.[2] Floppy-schijven worden niet meer geproduceerd en de DD-variant is tegenwoordig moeilijk te vinden.

U kunt een diskette met High Density (HD) in het station gebruiken, mits u één belangrijke wijziging aanbrengt: u moet beide zijden van het gat in de linkerbovenhoek (van voren gezien) van de schijf bedekken met een klein stukje plakband. Hierdoor laat u het station geloven dat de schijf DD is en schakelt het over naar een modus die compatibel is met de schijfcontroller van de MEGA65. Een DD-schijf heeft op deze plaats geen gat.

[2] Het is mogelijk dat HD-schijven met volledige capaciteit in een toekomstige firmware-update worden ondersteund. De hardware van het station kan HD-schijven lezen.



OPMERKING: Zorg ervoor dat het plakband beide zijden van het gat bedekt.

Een schijf formatteren

Een diskette moet geformatteerd worden voordat deze gebruikt kan worden. Het ingebouwde 3.5"-diskettestation van de MEGA65 emuleert een Commodore 1581-station en kan schijven gebruiken die in een dergelijk station zijn geformatteerd. U kunt een schijf ook met de MEGA65 formatteren.

OPMERKING: Het formatteren van een schijf wist de inhoud ervan. Wees voorzichtig en doe dit uitsluitend wanneer u de gegevens op de schijf niet meer nodig hebt!

Voer de volgende stappen uit om een fysieke 3.5"-diskette met het interne station te formatteren:

1. Open het Freezer-menu.
2. Koppel het ingebouwde 3.5"-diskettestation aan het eerste beheerde station, unit 8.
3. *Controleer dat unit 8 het volgende aangeeft:* - INTERNAL 3.5" -
4. Hervat de computer: druk op **F3**.
5. Plaats de diskette die u wilt formatteren in het interne diskettestation.

6. Voer het BASIC-commando **FORMAT** in en geef de schijf een naam (`"MIJNDISK"`) en een ID van twee tekens (`XX`).

```
FORMAT "MIJNDISK",IXX
```

7. Voer, wanneer u daarom gevraagd wordt, `YES` in en druk op RETURN.

Het formatteren van de schijf duurt ongeveer een minuut. Het station maakt tijdens het formatteren zoemende en klikkende geluiden. Schakel de computer niet uit en werp de schijf niet uit voordat het formatteren is voltooid.

U kunt bevestigen dat het formatteren is geslaagd door het **DIR**-commando uit te voeren. U ziet dan een lege lijst van mappen met de naam en de ID die u hebt opgegeven. Uw schijf is nu gereed voor gebruik.

EXTERNE IEC-SCHIJFSTATIONS GEBRUIKEN

De MEGA65 werkt met externe schijfstations die op de seriële IEC-poort zijn aangesloten.

Externe stations hoeven niet te worden gekoppeld. Indien een unitnummer niet aan het ingebouwde 3.5"-diskettestation of aan een schijfimage is toegewezen, worden de schijfbewerkingen voor dat unitnummer naar de seriële IEC-poort verzonden. Het is aan het apparaat dat op de poort is aangesloten om zijn unitnummer te herkennen. Sommige IEC-apparaten hebben schakelaars waarmee u het unitnummer kunt instellen. Andere werken alleen met een specifiek nummer.

Indien u een extern station hebt dat een specifiek unitnummer verwacht, moet u ervoor zorgen dat de MEGA65 dat nummer niet aan een schijfimage of aan het interne station toewijst. Open het Freezer-menu en druk vervolgens op 8 of 9 om de unitnummertoewijzingen om te schakelen, zodat zij het benodigde unitnummer niet meer gebruiken.

De station- en unitnummertoewijzingen zijn tijdelijk en worden teruggezet op hun standaardwaarden wanneer de MEGA65 wordt uitgeschakeld. U zult de stationtoewijzingen opnieuw moeten instellen wanneer u de computer de volgende keer inschakelt.

OPSTARTBARE SCHIJVEN

Bij oudere Commodore-computers was het gebruikelijk dat softwaremakers de map op een diskette zo organiseerden dat het eerste bestand in de lijst het hoofdprogramma is. De gebruiker kon dan het commando `LOAD "*",8,1` invoeren om het hoofdprogramma te laden en `RUN` om het uit te voeren. Het sterretje is een jokerteken dat met elk bestand

overeenkomt, dus ook met het eerste bestand op de schijf, zonder dat de gebruiker de naam van het programma hoeft te typen.

Deze methode is nog steeds gebruikelijk en de MEGA65 biedt een snelle manier om dergelijke schijven op te starten: houd SHIFT ingedrukt en druk op RUN STOP. Dit voert het commando `RUN "*"` uit, hetwelk vergelijkbaar is met de bekende commandoreeks die het eerste programma op de schijf laadt en uitvoert.

Met de C65 introduceerde Commodore een nieuwe manier om schijven op te starten. In plaats van te vertrouwen op de volgorde van bestanden, kan een schijf een bestand bevatten met de naam `AUTOBOOT.C65`. Indien dit bestand bestaat en een programma is, laadt en voert het BASIC-commando **BOOT** dit bestand uit.

```
BOOT
```

Schijven automatisch opstarten

Zoals besproken in hoofdstuk 5 op bladzijde 5-3, kunt u het Configuratiehulpprogramma gebruiken om de MEGA65 in te stellen zodat hij tijdens het opstarten automatisch een virtueel schijfimage of het ingebouwde 3.5"-diskettestation koppelt.

Indien de gekoppelde schijf opstartbaar is — dat wil zeggen dat hij een programmabestand met de naam `AUTOBOOT.C65` bevat — laadt en voert de MEGA65 het opstartprogramma automatisch uit.

Zo werkt de Intro Disk. Het Intro Disk-menu is een programma met de naam `AUTOBOOT.C65` op het virtuele schijfimage `MEGA65.D81`, hetwelk vooraf is ingesteld om bij het opstarten van het systeem gekoppeld te worden. Wanneer u de Intro Disk via het menu uitschakelt, hernoemt hij `AUTOBOOT.C65` naar `MENU`, zodat de schijf niet langer als opstartbaar wordt beschouwd.

Zelf een opstartschijf inrichten is een handige manier om uw computer te configureren. U kunt een kort BASIC-programma schrijven dat het systeemlettertype wijzigt, de achtergrondkleur aanpast en **KEY**-macro's naar uw smaak instelt. Sla dit programma vervolgens op als `AUTOBOOT.C65` op een schijf die u hebt geconfigureerd om bij het opstarten van het systeem aan te koppelen. Dit programma wordt elke keer uitgevoerd wanneer u uw MEGA65 inschakelt.

DE SD-KAART BENADEREN VANUIT BASIC

Verscheidene BASIC 65-commando's kunnen rechtstreeks op de SD-kaart van de MEGA65 werken alsof deze een schijfstation is. In deze gevallen staat de SD-kaart bekend als unit 12.

OPMERKING: Unit 12 kan slechts voor enkele specifieke bewerkingen rechtstreeks worden benaderd. Hij kan de gehele SD-kaart niet als een CBDOS-schijf behandelen.

Om alle bestanden op de SD-kaart weer te geven, gebruikt u het **DIR**-commando met het argument `U12`:

```
DIR U12
```

U kunt de optionele **P**-vlag met dit commando gebruiken om de bestanden op de SD-kaart één pagina tegelijk weer te geven. Druk op Q om op de huidige pagina te stoppen, of op een willekeurige andere toets om naar de volgende pagina te gaan.

```
DIR U12,P
```

Om een PRG-bestand rechtstreeks van de SD-kaart te laden (dat zich niet in een schijfimage bevindt), gebruikt u het `U12`-argument bij het **DLOAD**-commando. In dit geval *moet* u het achtervoegsel `.PRG` van de bestandsnaam opnemen, hetgeen verschilt van het gebruik van PRG-bestanden op schijven of in schijfimages. **DSAVE** kan niet naar de SD-kaart opslaan met `U12` en meldt een fout als u het toch probeert; sla een programma daarom op in een schijfimage.

```
DLOAD "MIJNPROGJE.PRG",U12
```

Zoals eerder is besproken, ondersteunt de MEGA65 submappen op de SD-kaart en houdt een huidige werkmap bij voor schijfbewerkingen. Om de huidige werkmap te wijzigen naar een submap:

```
CHDIR "SUBDIR",U12
```

Om de huidige werkmap te wijzigen naar de bovenliggende map:

```
CHDIR "..",U12
```

Het **MOUNT**-commando kan een D81- of D64-schijfimage aan een unitnummer koppelen. Hoewel dit commando naar een bestand op de SD-kaart verwijst, maakt het geen gebruik van het `U12`-argument. In plaats daarvan stelt het commando met het **U**-argument het unitnummer van de aan te koppelen schijf in. Het **MOUNT**-commando gebruikt de door **CHDIR** ingestelde huidige werkmap om het bestand te vinden.

GANGBARE SCHIJFBEWERKINGEN

Hieronder vindt u enkele voorbeelden van gangbare schijfbewerkingen die u bij de `READY`-prompt kunt uitvoeren. Raadpleeg voor meer informatie de naslag over BASIC-commando's in bijlage B op bladzijde B-3.

De meeste commando's die bestandsnamen aanvaarden, aanvaarden ook een **U**-argument waarmee u aangeeft welke unit het bestand bevat. De standaardunit is 8.[3]

DIR

Om de map (de lijst van bestanden) van een schijf weer te geven, gebruikt u het **DIR**-commando.

```
DIR

DIR U9
```

In tegenstelling tot de methode van de Commodore 64, waarbij de schijfmap in het BASIC-geheugen wordt geladen, wijzigt het **DIR**-commando het BASIC-geheugen niet. U kunt **DIR** veilig gebruiken wanneer er een programma in het geheugen staat.

Om grotere mappen gemakkelijker te kunnen bekijken, geeft **DIR W** (van "wide") de map in kolommen weer en pauzeert na elke pagina.

DLOAD en RUN

Het **DLOAD**-commando laadt een programma van de schijf in het geheugen. Het **RUN**-commando voert het programma uit dat zich momenteel in het geheugen bevindt.

```
DLOAD "GAAFSPEL"
RUN
```

U kunt beide tot één commando combineren door de bestandsnaam rechtstreeks aan het **RUN**-commando te geven.

```
RUN "GAAFSPEL"
```

DSAVE

Het **DSAVE**-commando slaat het BASIC-programma dat zich momenteel in het geheugen bevindt op de schijf op.

```
DSAVE "MIJNSPEL"
```

Standaard overschrijft dit geen bestaand bestand met dezelfde naam. Om een bestaand bestand te laten overschrijven, plaatst u een apestaartje vóór de bestandsnaam, binnen de dubbele aanhalingstekens.

[3] Wanneer de computer voor het eerst wordt gestart, is de standaard schijfunit voor BASIC-commando's 8. U kunt deze wijzigen met het **SET DEF**-commando.

```
DSAVE "@MIJNSPEL"
```

Merk op dat opslaan met vervanging alleen wordt aanbevolen bij gebruik van schijfimages en het 3.5"-diskettestation. Oudere Commodore-stations bevatten fouten in deze functie die tot gegevensverlies kunnen leiden.

BACKUP

Het **BACKUP**-commando kopieert een volledige schijf van de ene unit naar de andere. Hierbij worden alle bestaande gegevens op de bestemmingsschijf gewist.

```
BACKUP U8 TO U9
```

U kunt **BACKUP** gebruiken om schijfimages van floppy-schijven te maken, schijfimages naar floppy-schijven te schrijven of alles van het ene schijfstation naar het andere te kopiëren.

COPY

Het **COPY**-commando maakt een kopie van een bestand. Indien bron en bestemming verschillende bestandsnamen op dezelfde unit zijn, dupliceert dit het bestand op de schijf. Wanneer u een bestand tussen units kopieert, behoudt het bestand zijn oorspronkelijke bestandsnaam. (Het nieuwe bestand kan met een afzonderlijk commando worden hernoemd.)

```
COPY "MIJNSPEL", U8 TO U9

COPY "MIJNSPEL" TO "MIJNSPEL-V1"
```

RENAME

Het **RENAME**-commando wijzigt de naam van een bestaand bestand.

```
RENAME "MIJNSPEL-V29" TO "MIJNSPEL-GEREED"
```

DELETE

Het **DELETE**-commando verwijdert een bestand.

```
DELETE "ROMMELBESTAND"
```

Snelkoppelingen voor schijfcommando's

BASIC 65 biedt enkele snelkoppelingen voor gangbare schijfcommando's die u bij de READY-prompt kunt gebruiken.

Snelkoppeling	Overeenkomend commando
/	**LOAD**
↑	**RUN**
←	**SAVE**
@	**DISK**
$	**DIR**

Deze snelkoppelingen zijn bedoeld om samen met een lijst van mappen programma's te starten zonder dat u bestandsnamen hoeft te typen. Bijvoorbeeld:

1. Geef de lijst van mappen van de schijf weer: typ **$** in en druk op RETURN.
2. Gebruik de cursortoetsen om de cursor te verplaatsen naar de regel met het programma dat u wilt uitvoeren.
3. Typ ↑ in en druk op RETURN.

Het geselecteerde programma wordt geladen en uitgevoerd. Merk op dat u geen overtollige tekens van de regel hoeft te verwijderen. De snelkoppeling negeert alles behalve de bestandsnaam tussen dubbele aanhalingstekens, zoals die door de lijst van mappen wordt weergegeven.

HOOFDSTUK 8

Bestanden overzetten

- Bestanden naar de MEGA65 overbrengen
- Netwerken begrijpen
- M65Connect verkrijgen
- Netwerkluisteren inschakelen
- Bestanden overzetten

BESTANDEN NAAR DE MEGA65 OVERBRENGEN

Het schrijven van uw eigen programma's voor de MEGA65 is erg leuk, maar uiteindelijk wilt u ook programma's uitvoeren die door anderen zijn geschreven. Wellicht wilt u ook een reservekopie van uw MEGA65-programma's op uw pc maken om ze veilig te bewaren.

De snelste en betrouwbaarste manier om bestanden tussen uw pc en uw MEGA65 over te zetten is met een Ethernet-kabel. U sluit het ene uiteinde van de kabel aan op de RJ45-aansluiting aan de achterzijde van de MEGA65. U kunt het andere uiteinde aansluiten op de router of switch van uw local area network (LAN), of rechtstreeks op uw pc. U gebruikt software op uw pc om bestandsoverdrachten in beide richtingen te starten: van de pc naar de MEGA65 of van de MEGA65 naar de pc.

Als alternatief kunt u D81-virtuele schijfimages met behulp van een willekeurige pc met een SD-kaartlezer naar uw door de MEGA65 geformatteerde SD-kaart kopiëren, zonder andere speciale hulpmiddelen of software. Uw pc herkent het gegevensgebied van de SD-kaart als een FAT32-partitie. Indien u deze methode gebruikt, dient u zich ervan bewust te zijn dat sommige besturingssystemen van pc's ongewenste neveneffecten kunnen hebben, zoals fragmentatie van bestanden op de SD-kaart of overbodige bestanden die door macOS Finder worden aangemaakt. Deze effecten zijn onschadelijk voor de gegevens, maar kunnen onderhoud vergen om de kaart bruikbaar te houden in de MEGA65. Indien de MEGA65 meldt dat een bestand gefragmenteerd is, kunt u op de gegevenspartitie een defragmentatiehulpmiddel voor pc-schijven gebruiken. Als alternatief kunt u alle bestanden van de SD-kaart naar de pc kopiëren, de SD-kaart in de MEGA65 opnieuw formatteren en de bestanden vervolgens vanaf de pc terugkopiëren.

Het is ook mogelijk om bestanden over te zetten via een JTAG- of UART seriële interface die op het moederbord is aangesloten. Dit is een geavanceerde techniek en wordt niet in deze gebruikershandleiding beschreven. JTAG- of UART Serial-hardware biedt toegang tot een debuginterface die voor sommige programmeurs nuttig kan zijn. JTAG is ook nuttig voor het ontwikkelen van FPGA-cores. Zie voor meer informatie *De Mega65 programmeergids*.

De meeste mensen geven de voorkeur aan de Ethernet-methode. Dit hoofdstuk beschrijft hoe u dat doet.

NETWERKEN BEGRIJPEN

De MEGA65 kan Ethernet gebruiken om verbinding te maken met andere computers in een netwerk, of om verbindingen van andere computers te aanvaarden. Met de juiste programmatuur kan hij via het internet verbinding maken met andere computers.

De Ethernet-hardware van de MEGA65 biedt het lokale netwerk een Media Access Control (MAC)-adres aan. In tegenstelling tot andere Ethernet-hardware wordt het MAC-adres van de MEGA65 niet in de fabriek toegewezen: het wordt ingesteld in het configuratiehulpprogramma. (Zie hoofdstuk 5 op bladzijde 5-3.)

Om bestanden over te zetten, geeft u de MEGA65 opdracht zich beschikbaar te stellen voor binnenkomende verbindingen. Vervolgens gebruikt u op uw pc de M65Connect-app (of een ander hulpprogramma, zoals `mega65_ftp`) om een verbinding tot stand te brengen. Het besturingssysteem van uw pc kan om toestemming vragen om het hulpprogramma toegang tot het netwerk te verlenen wanneer u het voor de eerste keer uitvoert. Het hulpmiddel gebruikt UDP-poort 4510 om de eerste verbinding met de MEGA65 tot stand te brengen en gebruikt voor de bestandsoverdrachtssessie een zelf toegewezen IPv6-adres dat op basis van het MAC-adres van de MEGA65 is aangemaakt. Hiervoor moet IPv6 op de netwerkinterface van de pc zijn ingeschakeld, wat in de meeste gevallen de standaardinstelling is.

Als alternatief voor de aansluiting op uw lokale netwerk kunt u de MEGA65, indien uw pc over een Ethernet-aansluiting beschikt, rechtstreeks met een Ethernet-kabel op uw pc aansluiten. Zo vormt u een klein lokaal netwerk zonder toegang tot het internet. De werkwijze voor het overzetten van bestanden is bij een rechtstreekse verbinding dezelfde als bij een verbinding via het lokale netwerk.

M65CONNECT VERKRIJGEN

M65Connect is een toepassing voor Windows, Mac of Linux die bestandsoverdrachten en andere nuttige functies voor MEGA65-gebruikers mogelijk maakt. De toepassing beschikt over een interface met vensters en bevat tevens commandoregelhulpmiddelen die nuttig zijn bij het programmeren.

Om M65Connect te verkrijgen:

1. Bezoek de MEGA65 Filehost-website in een browser: `https://files.mega65.org`
2. Typ in het zoekveld rechtsboven: “M65Connect”
3. Selecteer de versie van M65Connect voor het besturingssysteem van uw pc.
4. Klik op de knop “Download”.
5. Gebruik uw pc om het gedownloade archiefbestand uit te pakken.

M65Connect voor Windows

De Windows-versie van M65Connect staat in de map “M65Connect”: **M65Connect.exe**. Zoals bij de meeste openbronprogrammatuur het geval is, kan Microsoft Defender weigeren de programmatuur uit te voeren en een dialoogvenster weergeven. Indien dit gebeurt, klikt u op “More info” en vervolgens op de knop “Run anyway” die verschijnt.

De commandoregelhulpmiddelen staan in een submap met de naam “M65Connect Resources”, zoals: `M65Connect Resources\mega65_ftp.exe`

M65Connect voor macOS

De macOS-versie van M65Connect is een Mac-toepassingsbundel: **M65Connect.app**. Zoals bij de meeste openbronprogrammatuur het geval is, beschouwt macOS het niet als door de ontwikkelaar ondertekend en zal macOS weigeren het uit te voeren. U moet het kenmerk “quarantine” verwijderen om de toepassing te kunnen uitvoeren.

In de meeste versies van macOS kunt u het quarantainekenmerk het eenvoudigst verwijderen met een Terminal-opdracht:

1. Verplaats de M65Connect-app naar uw map Applications.
2. Open de Terminal-app, die bij macOS is inbegrepen. U vindt deze in de map Applications, in een submap met de naam Utilities.
3. Voer deze opdracht in: `xattr -cr /Applications/M65Connect.app`

U kunt de M65Connect-app nu met een dubbelklik uitvoeren.

De commandoregelhulpmiddelen bevinden zich in de map van de toepassingsbundel, zoals: `/Applications/M65Connect.app/Contents/Resources/mega65_ftp.osx`

M65Connect voor Linux

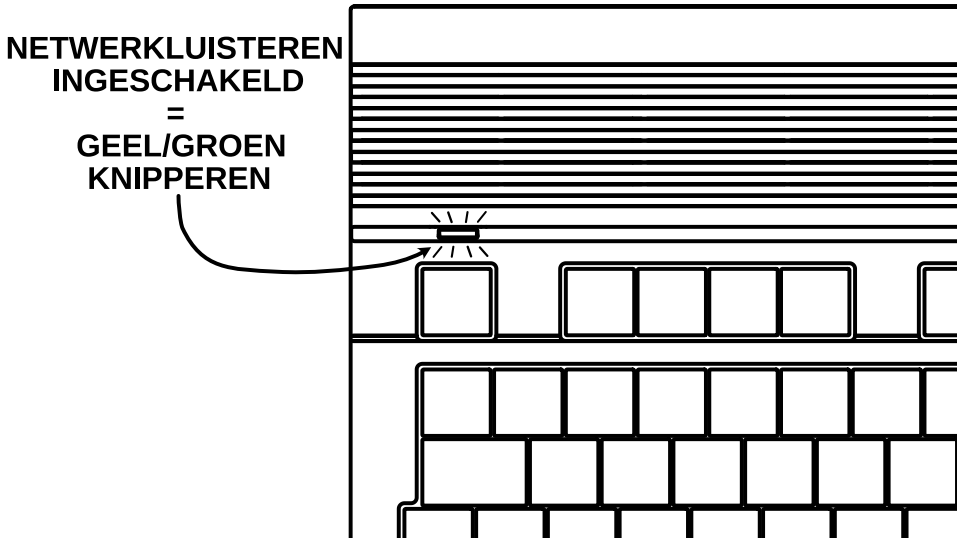
De Linuxversie van M65Connect bevindt zich in de map “M65Connect”: **M65Connect**. Dubbelklik erop om het uit te voeren.

De commandoregelhulpmiddelen staan in een submap met de naam “M65Connect Resources”, zoals: `M65Connect Resources/mega65_ftp`

NETWERKLUISTEREN INSCHAKELEN

Standaard negeert de MEGA65 alle pogingen van andere computers om er via het netwerk verbinding mee te maken. Programmatuur welke op de MEGA65 draait, kan luisteren naar netwerkverbindingen, doch de MEGA65 doet dit niet uit zichzelf.

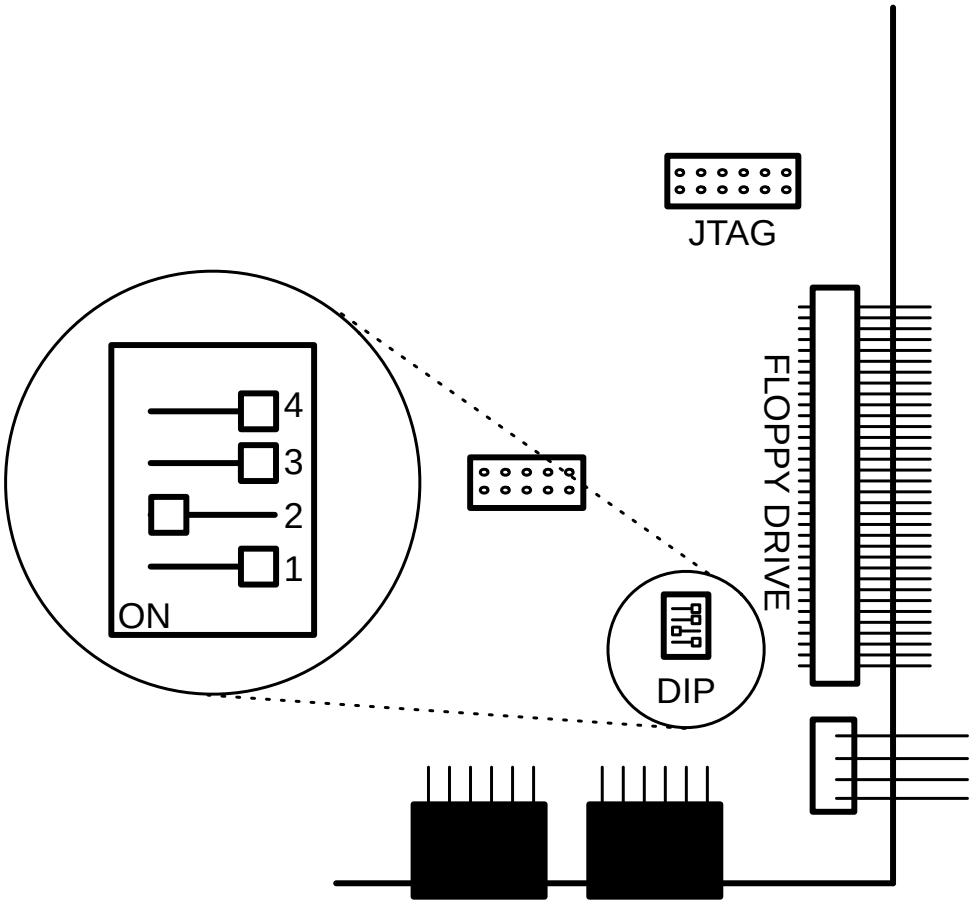
Om bestanden met M65Connect over te zetten, dient u de MEGA65 te laten luisteren naar binnenkomende verbindingspogingen van M65Connect. Druk op SHIFT + £ om een netwerkluistersessie in te schakelen. Het stroomlampje knippert afwisselend geel en groen wanneer netwerkluisteren actief is.



Druk opnieuw op SHIFT + £ om netwerkluisteren uit te schakelen, of reset de computer.

Indien het stroomlampje na het indrukken van SHIFT + £ niet begint te knipperen, moet u mogelijk DIP-schakelaar #2 op het moederbord instellen. Bij MEGA65-modellen die in 2024 of later zijn gefabriceerd, hoort deze schakelaar in de fabriek ingesteld te zijn. Bij oudere MEGA65-modellen staat deze schakelaar standaard uit.

Om de DIP-schakelaar in te stellen, opent u de behuizing zoals beschreven in hoofdstuk 3 op bladzijde 3-3. Zoek de DIP-schakelaars op het moederbord en zet DIP-schakelaar #2 in de positie “on”. Zoek naar markeringen op de schakelaars om schakelaar #2 en de richting “on” te herkennen. (De oriëntatie van uw DIP-schakelaars kan van dit diagram afwijken.)



U kunt DIP #2 gerust in deze positie laten staan voor normaal gebruik.

BESTANDEN OVERZETTEN

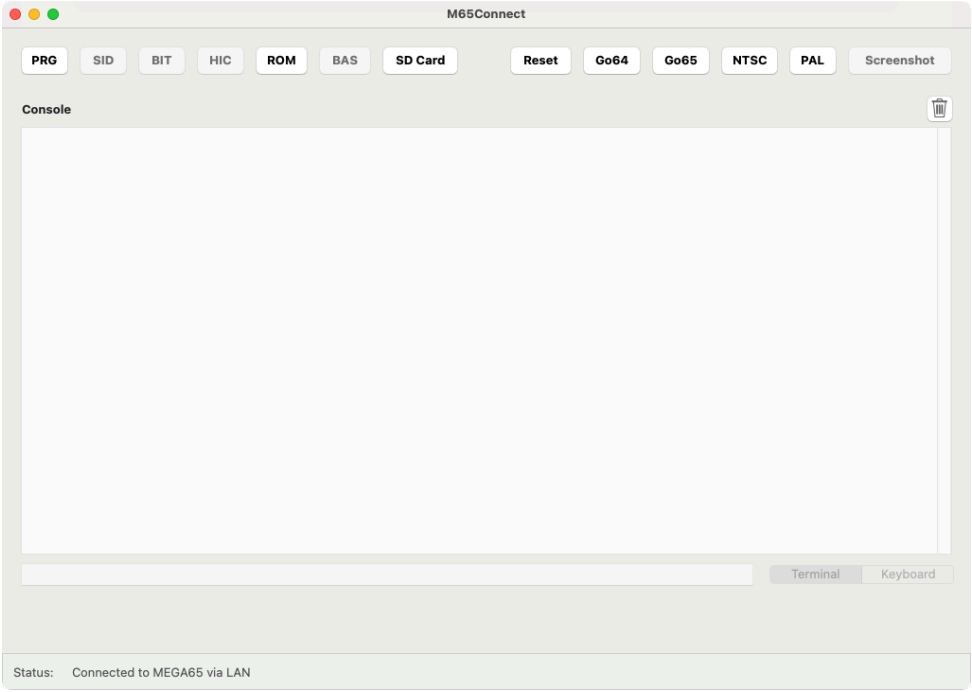
Om bestanden over te zetten, start u een bestandsoverdrachtssessie met de toepassing M65Connect of het commandoregelhulpmiddel `mega65_ftp`. Dit hulpmiddel maakt verbinding met de MEGA65 en uploadt een bestandsoverdrachtclient voor gebruik tijdens de sessie. Wanneer u de sessie beëindigt, reset de MEGA65.

Het starten van een bestandsoverdrachtssessie reset de MEGA65. Sla eventuele programma's of gegevens op voordat u verdergaat.

OPMERKING: Indien u het geheugen wist door de computer te resetten, vergeet dan niet om netwerkluisteren opnieuw in te schakelen: druk op SHIFT + £ en zorg dat het stroomlampje knippert.

Bestanden overzetten met M65Connect

M65Connect detecteert automatisch of de MEGA65 naar verbindingen luistert. Open M65Connect en schakel vervolgens de netwerkluistersessie op de MEGA65 in. M65Connect meldt een status van "Connected to MEGA65 via LAN". M65Connect schakelt daarna in het venster diverse knoppen in, waaronder de **PRG**- en **SD Card**-knoppen.



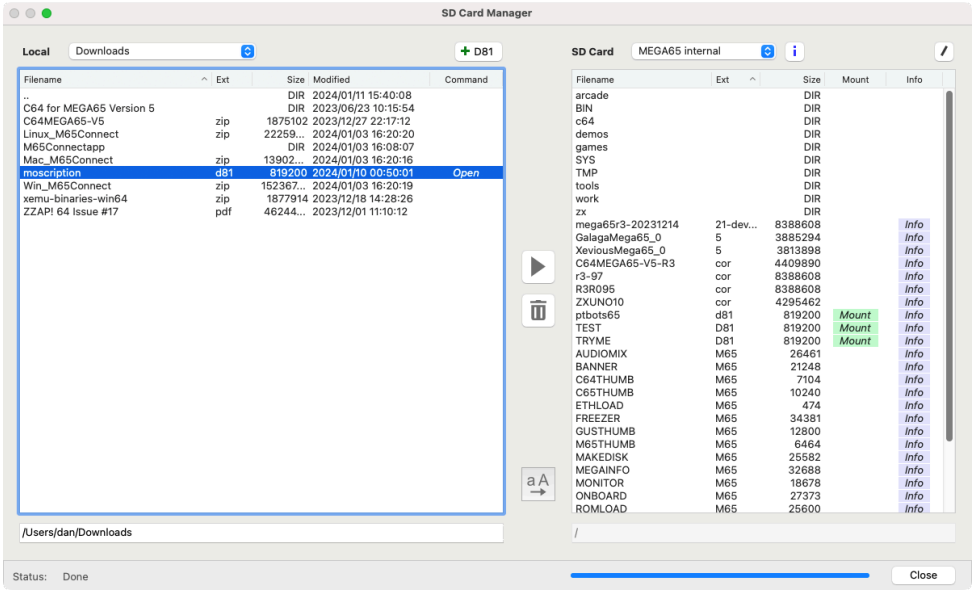
Indien M65Connect een status van "Not connected to MEGA65" meldt, controleer dan het volgende:

- De MEGA65 en de pc zijn op hetzelfde netwerk aangesloten of rechtstreeks op elkaar via een netwerkkabel.
- De MEGA65 bevindt zich in de netwerkluistermodus met een knipperend stroomlampje.

- Open in M65Connect het menu Settings en selecteer Connections. Het veld "LAN Port" hoort een IPv6-adres te bevatten. Indien dat niet het geval is, wacht dan enkele seconden of klik op de knop "Autodetect LAN Port".

Klik op de knop **SD Card** om een bestandsoverdrachtssessie te starten. Het SD Card Manager-venster gaat open.

OPMERKING: Het starten van een bestandsoverdrachtssessie reset de MEGA65 om het bestandsoverdrachtshulpprogramma te laden. Zorg ervoor dat u eventuele gegevens op de MEGA65 hebt opgeslagen voordat u de sessie start.



Gebruik het linkerdeelvenster om door de bestanden op uw pc te navigeren en het rechterdeelvenster om door de bestanden op de SD-kaart van de MEGA65 te navigeren. Selecteer het bestand en klik daarna op de pijlknop om het over te zetten. De knop geeft de richting aan waarin het bestand wordt overgezet.

U kunt M65Connect ook gebruiken om D81-schijfimages aan te maken en bestanden van en naar D81-schijfimages te kopiëren. Zoek een D81-schijfimage op uw pc of klik op de knop **+ D81** om er een aan te maken. Klik daarna in de bestandsverkenner op de opdracht "Open" om het schijfimage in het linkerdeelvenster te openen. Zet bestanden met de pijlknop over van en naar het schijfimage. Klik op de knop **X** in de rechterbovenhoek om terug te keren naar de bestandsverkenner.

Klik op de knop **Close** om de bestandsoverdrachtssessie te beëindigen en het SD Card Manager-venster te sluiten. Dit reset de MEGA65.

Het commandoregelhulpmiddel mega65_ftp

Het commandoregelhulpmiddel `mega65_ftp` start een bestandsoverdrachtssessie met de MEGA65. Het kan interactief in de terminal draaien en meerdere bestandsoverdrachtopdrachten aanvaarden, of niet-interactief draaien met deze opdrachten als argumenten.

Om een interactieve bestandsoverdrachtssessie te starten, voert u de opdracht `mega65_ftp` uit en geeft u daarbij het argument `-e` mee om aan te geven dat u een Ethernet-verbinding wilt gebruiken.

OPMERKING: Het starten van een bestandsoverdrachtssessie reset de MEGA65 om het bestandsoverdrachtshulpprogramma te laden. Zorg ervoor dat u eventuele gegevens op de MEGA65 hebt opgeslagen voordat u de sessie start.

```
% mega65_ftp -e
```

Het hulpprogramma uploadt de bestandsoverdrachtclient en u ziet de client op de MEGA65 draaien. Indien er niets gebeurt, drukt u op Ctrl-C (op de pc) om de overdracht af te breken en controleert u daarna nogmaals of de MEGA65 verbonden is en of netwerkluisteren is ingeschakeld.

Zodra de verbinding tot stand is gekomen, ziet de prompt voor bestandsoverdrachtopdrachten er ongeveer zo uit:

```
MEGA65 SD-Card:/>
```

Om de sessie te beëindigen, gebruikt u de opdracht `exit`. Het hulpprogramma sluit af, u keert terug naar de shellprompt en de MEGA65 reset.

```
MEGA65 SD-Card:/> exit
%
```

Hieronder volgen enkele nuttige opdrachten die u tijdens de bestandsoverdrachtssessie kunt gebruiken. Gebruik de opdracht `help` om een volledige lijst van de beschikbare opdrachten te zien.

Opdracht	**Omschrijving**
`put` *bestandsnaam*	Verstuurt een bestand van de pc naar de MEGA65.
`get` *bestandsnaam*	Haalt een bestand op van de MEGA65 en plaatst het op de pc.
`dir`	Toont een lijst van de mappen van de SD-kaart van de MEGA65.
`ldir`	Toont een lijst van de huidige lokale werkmap.
`mkdir` *bestandsnaam*	Maakt een submap aan op de SD-kaart van de MEGA65.
`cd` *bestandsnaam*	Wijzigt de huidige werkmap op de SD-kaart van de MEGA65.
`lcd` *bestandsnaam*	Wijzigt de huidige lokale werkmap.
`help`	Toont een lijst met beschikbare opdrachten.
`exit`	Beëindigt de bestandsoverdrachtssessie.

Om opdrachten van `mega65_ftp` aan te roepen zonder een interactieve prompt te starten, geeft u het argument `-c` één keer per opdracht mee:

```
% mega65_ftp -e -c 'put mydisk.d81' -c 'exit'
```

Het hulpprogramma start een sessie, voert de opdrachten uit en stopt vervolgens. Zorg ervoor dat u de opdracht `exit` als laatste opdracht invoert om de MEGA65 te resetten, of reset de MEGA65 handmatig nadat de bestandsoverdracht is voltooid.

DEEL III

EERSTE STAPPEN IN HET PROGRAMMEREN

HOOFDSTUK 9

Hoe computers werken

- **Computers zijn dom. Echt dom**

Wist u dat veel computerexperts en programmeurs al leerden omgaan met computers toen ze nog kleine kinderen waren? Thuiscomputers werden pas in het begin van de jaren tachtig gemeengoed. Ze waren toen zo nieuw dat mensen vaak zelf programma's schreven om te doen wat zij wilden doen, omdat er nog geen software bestond die dat werk voor hen deed.

Het was ook heel gewoon dat mensen met allerlei kantoorbanen leerden programmeren op de computers die zij voor hun werk gebruikten. Zo leerden bijvoorbeeld de mensen die de salarisadministratie van een bedrijf deden, de computer te programmeren zodat die het salaris van iedereen berekende!

Sindsdien is er veel veranderd. De meeste mensen kiezen tegenwoordig bestaande programma's of toepassingen om te doen wat zij nodig hebben en denken dat programmeren een gespecialiseerde vaardigheid is die slechts sommige mensen kunnen leren. Dat is echter niet waar. Natuurlijk is iedereen, net als in elk ander vakgebied, in sommige dingen beter dan in andere, of het nu om sport, breien, wiskunde of schrijven gaat. Bijna iedereen is in staat genoeg te leren om zich in het leven te kunnen redden.

Wij hebben de MEGA65 gemaakt, omdat wij ervan overtuigd zijn dat juist u kunt leren programmeren, zodat computers nuttiger voor u worden, en dat u, net als bij het aanleren van elke nieuwe vaardigheid, de voldoening, het plezier en de nieuwe avonturen kunt ervaren die dat met zich meebrengt!

COMPUTERS ZIJN DOM. ECHT DOM

Hoe kan dit nu waar zijn? Computers kunnen zoveel verschillende dingen doen, vaak duizenden malen sneller dan een mens dat kan. Hoe kunnen wij dan beweren dat computers dom zijn? Het antwoord is dat geen enkele computer iets kan doen wat een persoon hem niet heeft opgedragen. Zelfs aan de nieuwste kunstmatige-intelligentiesystemen heeft men geleerd hoe zij moeten leren (of hoe zij moeten leren, hoe zij moeten leren). Om te begrijpen waarom dit zo is, helpt het om te begrijpen hoe computers werkelijk werken.

Een eierdopjescomputer maken

Het hart van een computer is de Central Processing Unit (CPU). Veel moderne computers hebben meer dan één processor, maar laten we het om te beginnen eenvoudig houden. De processor heeft een reeks eenvoudige instructies die hij begrijpt, zoals "haal het ding uit eierdopje #21", "stop dit ding in eierdopje #403", "tel deze dingen bij elkaar op" of "voer de volgende instructie uit, maar alleen als het ding in eierdopje #712 het getal 3 is."

Wat bedoelen we eigenlijk met al die "dingen" en "eierdopjes"? Laten we beginnen met na te denken over hoe we ons als computer kunnen voordoen, met niets meer dan een lege eierdoos, wat kleine papiertjes en een potlood of pen. Schrijf eerst in elk van de kleine eierdopjes van de eierdoos een getal, te beginnen bij één. Schrijf daarna het getal nul op een klein papiertje en leg dat papiertje in het eerste eierdopje.

Doe hetzelfde voor de andere eierdopjes. U heeft nu een eierdoos met genummerde eierdopjes, waarin elk eierdopje een papiertje bevat waarop het getal nul staat. Nu hoeven we alleen nog een paar eenvoudige regels te bedenken die beschrijven hoe onze eierdopjescomputer werkt:

- Ten eerste mag elk eierdopje precies één papiertje tegelijk bevatten. Nooit meer. Nooit minder. Zo weten we zeker dat er, wanneer we de vraag stellen "wat zit er in eierdopje zoveel", één eenduidig antwoord is. Zo werkt ook het computergeheugen: elke geheugenlocatie kan slechts één ding tegelijk bevatten.
- Ten tweede heeft de computer een manier nodig om te weten wat hierna moet gebeuren. Op de meeste computers heet dit de programmateller (PC), niet te verwarren met de PC als afkorting van **p**ersonal **c**omputer. De programmateller is niets meer dan het nummer van de volgende geheugenlocatie (of in ons geval, het eierdopje) die de computer bekijkt wanneer hij beslist wat hierna moet gebeuren. Het is handig om nog een extra papiertje bij de hand te hebben waarop u onderweg het nummer van de PC kunt schrijven.
- Ten derde hebben we een lijst nodig van de dingen die de eierdopjescomputer zal doen, op basis van het getal dat in het door de PC aangegeven eierdopje staat.

Laten we nu bedenken wat de computer allemaal kan doen, op basis van het getal in het door de PC aangegeven eierdopje. We houden het eenvoudig en beperken ons tot het volgende:

Getal in het eierdopje	**Handeling**
0	i) Tel één op bij de PC en doe verder niets.
1	i) Tel één op bij de PC. ii) Stel de PC in op het getal dat in dat eierdopje staat.
2	i) Tel één op bij de PC. ii) Tel het getal in het door de PC aangegeven eierdopje op bij het getal in het eierdopje dat wordt aangegeven door het getal in het daaropvolgende eierdopje. iii) Zet het antwoord in het eierdopje dat wordt aangegeven door het daaropvolgende eierdopje. iv) Tel ten slotte nog twee op bij de PC, om de eierdopjes die we gebruikt hebben over te slaan.

Maakt u zich geen zorgen als dit nu wat verwarrend klinkt, vooral die laatste – die nemen we heel binnenkort uitgebreid door! De beste manier om dit uit te leggen is enkele voorbeelden door te werken.

HOOFDSTUK 10

Aan de slag met BASIC

- Uw eerste BASIC-programma's
- Eerste stappen met tekst en getallen
- Eenvoudige beslissingen nemen
- Willekeurige getallen en toeval

Op de MEGA65 is het mogelijk om in veel talen te programmeren, maar de meeste mensen beginnen met BASIC. Dat is begrijpelijk, want BASIC is de afkorting van Beginner's All-purpose Symbolic Instruction Code: de taal is gemaakt om mensen zoals u de eerste stappen in de wereld van het programmeren te laten zetten!

Nog enkele korte woorden voordat we van start gaan: BASIC is een programmeertaal en heeft, net als een gesproken taal, conventies, grammatica en woordenschat. Gelukkig is BASIC veel sneller en gemakkelijker te leren dan onze ingewikkelde menselijke talen. Als u echter goed oplet, zult u een aantal van die structuren herkennen en dat kan u op weg helpen in de wereld van het programmeren.

Indien u Hoofdstuk/Bijlage 4 op bladzijde 4-3 nog niet gelezen heeft, is het verstandig dat alsnog te doen, waardoor u zelfverzekerder met de MEGA65-computer kunt omgaan.

Het is ook goed om te weten dat u, als u de MEGA65 echt in de war heeft gebracht, altijd terug kunt keren naar de READY.-prompt: druk gewoon op de resetknop aan de linkerkant van het toetsenbord, of, als dat niet helpt, schakel de computer met de aan/uit-schakelaar aan de linkerkant van het toetsenbord uit en weer in. U hoeft zich geen zorgen te maken over het netjes afsluiten van de computer en soortgelijke idioterie. Het enige dat u moet onthouden, is dat niet-opgeslagen werk verloren gaat wanneer u de computer uitschakelt en weer inschakelt of op de resetknop drukt.

Tot slot: als u niet alle beschrijvingen en toelichtingen bij een voorbeeld begrijpt, geen zorgen! Wij hebben zoveel mogelijk informatie opgenomen, zodat die beschikbaar is wanneer u vragen heeft, problemen tegenkomt of gewoon meer wilt ontdekken. Blader gerust door naar de voorbeelden en probeer zelf van alles uit; u kunt later terugkeren en het nog eens nalezen wanneer u iets te weten wilt komen of een probleem wilt oplossen. En als u geen antwoord op uw probleem vindt, stuur ons dan een bericht! Er zijn ondersteuningsfora voor de MEGA65 op `https://mega65.net` en u kunt problemen met deze handleiding melden op:

`https://github.com/mega65/mega65-user-guide`

Wij hopen dat u net zoveel plezier beleeft aan het leren programmeren van de MEGA65 als wij hebben gehad bij het maken ervan!

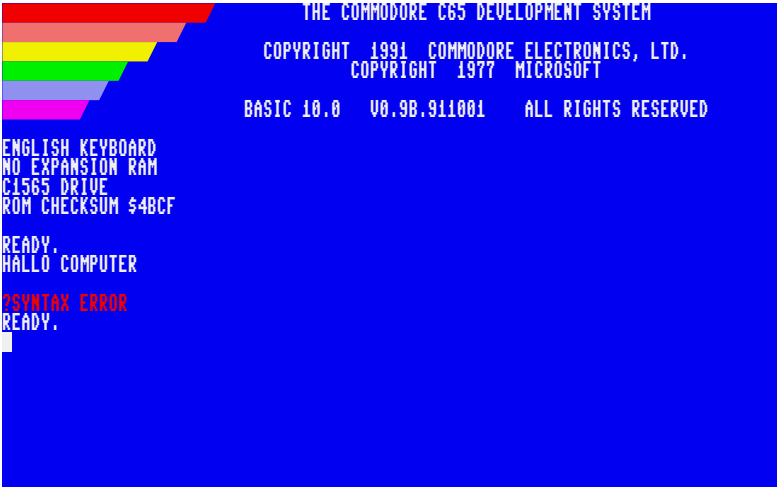
UW EERSTE BASIC-PROGRAMMA'S

De MEGA65 is ontworpen om te programmeren! Wanneer u hem inschakelt, heeft de computer een paar seconden nodig om zijn zaken op orde te brengen en daarna toont hij u direct een "READY."-prompt en een knipperend blokje dat de cursor wordt genoemd. Zolang de cursor knippert, vertelt hij u dat de computer wacht op invoer. De "READY."-melding vertelt u dat de programmeertaal BASIC draait en gereed is om te beginnen met programmeren. U hoeft niet eens een programma te laden - u kunt gewoon van start gaan.

Typ het volgende in en kijk wat er gebeurt:

```
HALLO COMPUTER
```

Typ daarvoor gewoon de letters zoals u ze hierboven ziet. De computer staat al in de hoofdlettermodus, dus u hoeft SHIFT of CAPS LOCK niet ingedrukt te houden. Druk op RETURN nadat u "HALLO COMPUTER" heeft getypt. De computer geeft u daarmee te kennen dat hij de door u getypte invoerregel moet aanvaarden. Indien u dat doet, ziet u ongeveer de volgende melding:



Als u een melding zag die ongeveer lijkt op **SYNTAX ERROR**, dan gefeliciteerd: u bent erin geslaagd om met de computer te communiceren! Foutmeldingen klinken veel erger dan ze zijn. De MEGA65 gebruikt ze, in het bijzonder de syntaxisfout, om u te laten weten wanneer hij moeite heeft om te begrijpen wat u heeft getypt of wat u in een programma heeft gezet. U hoeft er niet bang voor te zijn; ervaren programmeurs krijgen er voortdurend mee te maken.

In dit geval raakte de computer in de war, omdat hij het woord "hallo" of het woord "computer" niet begrijpt. Dat wil zeggen: hij wist niet wat u wilde dat hij deed. In dat opzicht zijn computers tamelijk dom: ze kennen slechts enkele woorden en zijn niet bijzonder vindingrijk in de manier waarop ze die interpreteren.

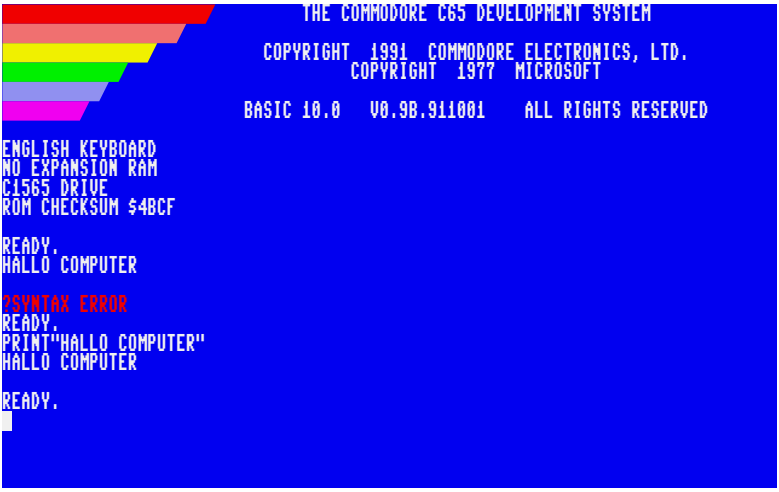
Laten we het dus nog eens proberen op een manier die de computer wel begrijpt. Typ het volgende in. Dat kunt u direct doen; het maakt niet uit dat de syntaxisfoutmelding nog op het scherm staat. Tegen de tijd dat hij u weer **READY.** toonde, was de computer die alweer vergeten.

```
PRINT "HALLO COMPUTER"
```

Zorg er opnieuw voor dat u tijdens het intypen geen shift of shift-lock gebruikt. De symbolen rondom het woord **HALLO COMPUTER** zijn dubbele aanhalingstekens. Als u het toetsenbord van een PC gewend bent, zult u wellicht merken dat de toets voor het dubbele aanhalingsteken op een heel andere plaats zit dan u gewend bent: dubbele aanhalingstekens typt u op de MEGA65 door SHIFT ingedrukt te houden en vervolgens 2 in te drukken. Vergeet niet op RETURN te drukken wanneer u klaar bent, zodat de computer weet dat u wilt dat hij iets met uw invoer doet.

Als u tijdens het typen een fout maakt, kunt u INST DEL gebruiken om de fout uit te wissen en te herstellen. U kunt ook de cursor-toetsen gebruiken om op de regel heen en weer te gaan terwijl u de regel bewerkt die u aan het typen bent, maar er schuilt een trucje in wanneer u al een dubbel aanhalingsteken getypt heeft: als u de cursor-toetsen probeert te gebruiken, wordt er een grappig omgekeerd teken weergegeven in plaats van dat de cursor beweegt. Dat komt doordat de computer denkt dat u de verplaatsing van de cursor in de tekst zelf wilt vastleggen, wat erg nuttig en leuk kan zijn en waarover u meer kunt lezen in Hoofdstuk/Bijlage 4 op bladzijde 4-3. Voor nu geldt echter: maakt u een fout, druk dan gewoon op RETURN en typ de verprutste regel opnieuw.

Hopelijk ziet u nu zoiets als dit:

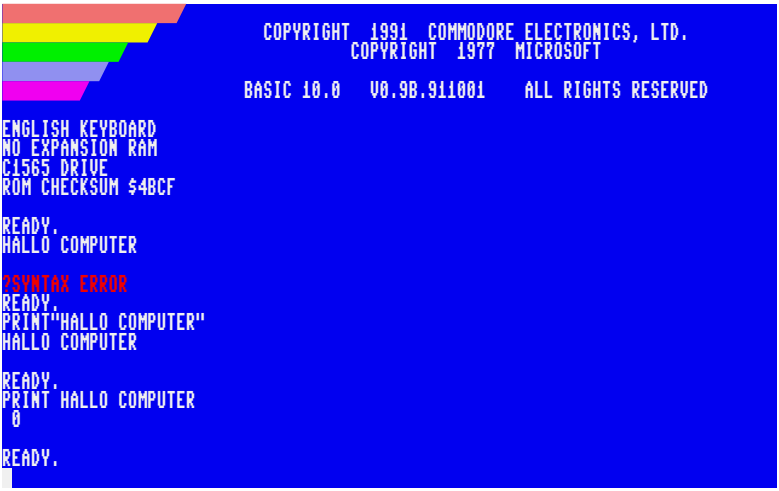


Deze keer zou er geen nieuwe **SYNTAX ERROR**-melding moeten verschijnen. Mocht er echter een of andere foutmelding zijn verschenen, probeer het commando dan gewoon opnieuw in te typen, nadat u goed gekeken heeft waar de fout zou kunnen zitten.

In plaats van een fout zouden we **HALLO COMPUTER** herhaald moeten zien onder de regel die u getypt heeft. De reden dat dit gebeurt, is dat de computer het woord **PRINT** wel degelijk begrijpt. Hij weet dat alles wat er achter het woord **PRINT** staat op het scherm weergegeven moet worden. We moesten **HALLO COMPUTER** tussen dubbele aanhalingstekens plaatsen om de computer te vertellen dat we willen dat het letterlijk wordt weergegeven.

Hadden we de dubbele aanhalingstekens niet geplaatst, dan zou de computer hebben gedacht dat **HALLO COMPUTER** de naam was van een opgeslagen stukje gegevens. Aangezien we op zo'n plaats geen enkel stukje gegevens hebben opgeslagen, staat er voor de computer een nul en zal de computer dus het getal nul weergeven. Indien de computer een nul of een ander getal weergeeft terwijl u een of andere tekst verwachtte, kan dit de oorzaak zijn.

U kunt het desgewenst zelf proberen en u zou dan zoiets als het volgende moeten zien:



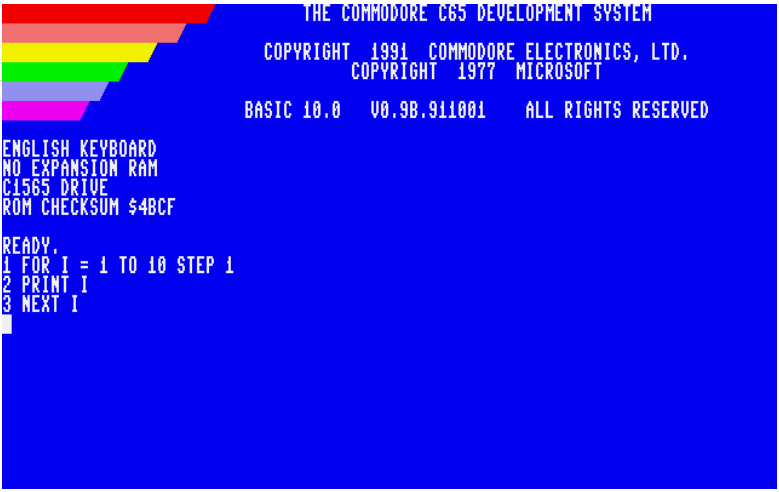
In de bovenstaande voorbeelden typten we commando's rechtstreeks in en de computer voerde ze onmiddellijk uit nadat u op RETURN drukte. Het op deze manier typen van commando's wordt derhalve vaak *directe modus* genoemd (soms ook wel "immediate mode" genoemd).

We kunnen de computer echter ook een lijst van commando's laten onthouden die één voor één uitgevoerd moeten worden. Dat doen we met de nogal fantasieloos benoemde *niet-directe modus*. Om de niet-directe modus te gebruiken, zetten we gewoon een getal tussen 0 en 63999 aan het begin van het commando. De computer onthoudt dat commando vervolgens. Anders dan bij een commando in de directe modus, geeft de computer niet opnieuw **READY.** weer. In plaats daarvan verschijnt de cursor gewoon weer op de volgende regel, klaar om meer commando's in te typen.

Laten we dat eens proberen met een eenvoudig programmaatje. Typ de volgende drie invoerregels in:

```
1 FOR I = 1 TO 10 STEP 1
2 PRINT I
3 NEXT I
```

Indien u dit gedaan heeft, zou het scherm er ongeveer zo uit moeten zien:



Als dat niet zo is, kunt u het opnieuw proberen. Vergeet niet dat u, wanneer u merkt dat de computer helemaal in de war raakt, gewoon de resetknop kunt indrukken of de aan/uit-schakelaar aan de linkerkant van de computer uit en weer aan kunt zetten om de computer opnieuw op te starten. Dat duurt slechts een paar seconden en doet de MEGA65 geen enkel kwaad.

We hebben de computer gevraagd drie commando's te onthouden, namelijk **`FOR I = 1 TO 10 STEP 1`**, **`PRINT I`** en **`NEXT I`**. We hebben de computer ook gevraagd in welke volgorde we ze willen uitvoeren: de computer begint met het commando met het laagste nummer en voert vervolgens elk commando met het eerstvolgende hogere nummer uit, totdat hij het einde van de lijst bereikt. Het is dus een beetje als een herinneringslijst voor de computer. Dit is wat we een programma noemen, een beetje zoals het programma van een concert of een theatervoorstelling: het vertelt ons wat er gaat komen en in welke volgorde. Laten we de computer dus vragen dit programma uit te voeren.

Laten we eerst proberen te raden wat er zal gebeuren. Laten we beginnen met het middelste commando, **`PRINT I`**. We hebben het **`PRINT`**-commando al gezien en we weten dat het de computer vertelt dat hij dingen op het scherm moet weergeven. Het ding dat hij zal proberen weer te geven is **`I`**. Net als eerder, omdat er geen dubbele aanhalingstekens rond **`I`** staan, zal hij proberen een opgeslagen stukje gegevens weer te geven. Het stukje gegevens dat hij probeert weer te geven, is het stukje dat hoort bij **`I`**.

Indien we een stukje gegevens als dit een naam geven, noemen we dat een *variabele*. Ze worden variabelen genoemd omdat ze kunnen variëren. Dat wil zeggen dat we het stukje gegevens dat hoort bij de variabele genaamd I kunnen vervangen door een ander stukje gegevens. Het oude stukje wordt daardoor vergeten. Als we dus een

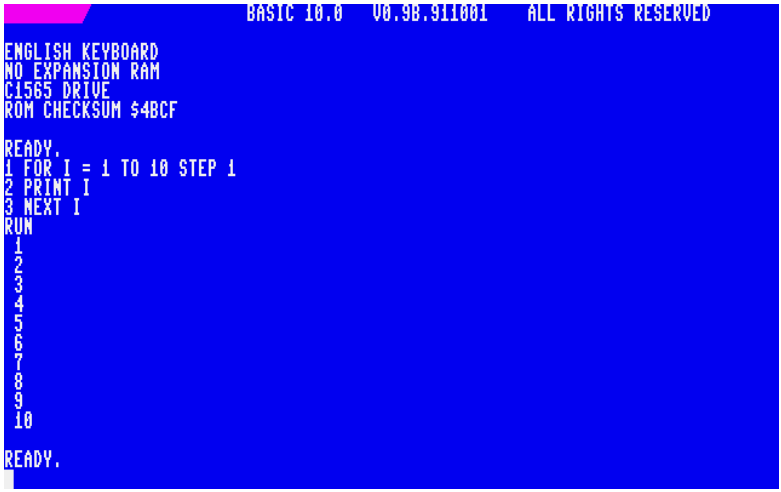
commando als **LET I = 3** zouden geven, zou dit alles wat er in de variabele **I** was opgeslagen vervangen door het getal 3.

Laten we teruggaan naar ons programma: we weten inmiddels dat het 2e commando zal proberen het stukje gegevens weer te geven dat in de variabele **I** is opgeslagen. Laten we dus naar het eerste commando kijken: **FOR I = 1 TO 10 STEP 1**. Ook al hebben we het **FOR**-commando niet eerder gezien, we kunnen wel een beetje raden hoe het werkt. Het lijkt erop dat het iets in de variabele **I** zal zetten. Dat iets lijkt te maken te hebben met het bereik van de getallen 1 tot en met 10 en met een stap of interval van 1. Wat denkt u dat het zal doen?

Als u geraden heeft dat het de waarden 1, 2, 3, 4, 5, 6, 7, 8, 9 en daarna 10 in de variabele **I** zet, dan verdient u een schouderklopje, want dat is precies wat het doet. Het helpt ons ook het 3e commando te begrijpen, **NEXT I**: dat commando zegt tegen de computer dat hij de volgende waarde in de variabele **I** moet zetten. En hier volgt een beetje magie: wanneer de computer dat doet, gaat hij terug in de lijst met commando's en vervolgt hij vanaf het commando na het **FOR**-commando.

Laten we dit eens samenvoegen: wanneer de computer het eerste commando uitvoert, ontdekt hij dat hij tien verschillende waarden in de variabele **I** moet zetten. Hij begint met het eerste getal erin te zetten, in dit geval het getal 1. De computer gaat vervolgens verder met het tweede commando, dat hem vertelt het stukje gegevens weer te geven dat op dat moment in de variabele **I** is opgeslagen. Dat zal het getal 1 zijn, aangezien dat het laatste was dat de computer erin moest zetten. Daarna gaat de computer door naar het derde commando, dat hem vertelt dat het tijd is om de volgende waarde in de variabele **I** te zetten. De computer gooit dus het getal 1 weg dat zich op dat moment in de variabele **I** bevindt en zet het getal 2 erin, aangezien dat het volgende getal in de lijst is. Vervolgens gaat hij verder vanaf het 2e commando, waardoor de computer de inhoud van de variabele **I** opnieuw weergeeft. Alleen is deze keer het getal 2 het laatst in **I** opgeslagen, dus geeft de computer het getal 2 weer. Dit proces herhaalt zich totdat de computer alle tien waarden heeft weergegeven die het **FOR**-commando aangaf.

Om dit in actie te zien, moeten we de computer het programma van commando's laten uitvoeren dat we hebben ingetypt. We doen dat met het **RUN**-commando. Omdat we willen dat het programma onmiddellijk wordt uitgevoerd, gebruiken we de directe modus. Typ dus gewoon het woord **RUN** en druk op RETURN. U zou dan een beeld moeten zien dat er ongeveer als volgt uitziet:



U zult hier misschien een paar dingen opmerken:

Ten eerste: de computer vertelt ons weer dat hij **READY.** is, zodra hij klaar is met het uitvoeren van het programma. Dit maakt het voor ons alleen maar makkelijker om te weten wanneer we weer commando's aan de computer kunnen geven.

Ten tweede: toen de computer de onderkant van het scherm bereikte, schoof hij het beeld automatisch omhoog om ruimte te maken. Dat is heel normaal. Wat belangrijk is om te onthouden, is dat de computer alles vergeet wat boven uit het scherm schuift. De enige uitzondering is wanneer u de computer een commando heeft laten onthouden door er een nummer voor te zetten. Ons programma is dus voorlopig veilig. Dat kunnen we zien door het **RUN**-commando nog een paar keer te typen: de programmalijst zal dan boven uit het scherm geschoven zijn, maar we kunnen het programma nog steeds uitvoeren, omdat de computer het heeft onthouden. Probeer het maar! Heeft het gewerkt?

Wilt u het programma met onthouden commando's zien, dan kunt u het **LIST**-commando gebruiken. Dit commando laat de computer het onthouden programma op het scherm weergeven, zoals in het voorbeeld hier. Wilt u een van de commando's in het programma vervangen, dan kunt u een nieuwe regel typen met hetzelfde nummer als de regel die u wilt wijzigen.



Om de resultaten bijvoorbeeld allemaal op één regel af te drukken, kunnen we de tweede regel van het programma wijzigen in **PRINT I;** door de volgende regel in te typen en op RETURN te drukken:

```
2 PRINT I;
```

U kunt controleren of de wijziging is onthouden door het **LIST**-commando opnieuw uit te voeren, zoals we hier zien. Vervolgens kunt u het **RUN**-commando gebruiken om het gewijzigde programma uit te voeren, als volgt:

```
 7
 8
 9
 10

READY.
LIST

1 FOR I = 1 TO 10 STEP 1
2 PRINT I
3 NEXT I

READY.
2 PRINT I;
LIST

1 FOR I = 1 TO 10 STEP 1
2 PRINT I;
3 NEXT I

READY.
RUN
 1  2  3  4  5  6  7  8  9  10
READY.
```

Het is heel eenvoudig om uw programma's op deze manier te wijzigen. Naarmate u meer vertrouwd raakt met de werkwijze, zijn er nog twee handige trucs:

Ten eerste kunt u het **LIST**-commando het nummer van een commando geven, of van een regel, zoals commando's ook wel worden aangeduid en het zal alleen die regel van het programma weergeven. U kunt ook een bereik opgeven, gescheiden door een minteken, om alleen een deel van het programma weer te geven, bijv. **LIST 1 - 2** om de eerste twee regels van ons programma te tonen.

Ten tweede kunt u de cursor-toetsen gebruiken om de cursor naar een regel te verplaatsen die al is onthouden en op het scherm wordt weergegeven. Wijzigt u wat u op het scherm ziet en drukt u vervolgens op RETURN terwijl de cursor op die regel staat, dan leest de BASIC-interpreter de gewijzigde regel in en vervangt hij de oude versie ervan. Het is belangrijk om op te merken dat u, wanneer u meerdere regels van het programma tegelijk wijzigt, op elke gewijzigde regel RETURN moet drukken. Het is goed gebruik om te controleren of het programma correct is gewijzigd. Gebruik hiervoor opnieuw het **LIST**-commando.

Oefeningen om te proberen

1. Kunt u het tot een hoger of lager getal laten tellen?

Op dit moment telt het van 1 tot 10. Kunt u het veranderen zodat het tot 20 telt? Of zodat het van 3 tot 17 telt? Of van 14.5 tot 21.5? Wat denkt u nodig te hebben om de telvolgorde om te keren?

Aanwijzing: U zult het **FOR**-commando moeten aanpassen.

2. Kunt u de telstap veranderen?

Op dit moment telt het met stapjes van één, d.w.z. elk getal is één meer dan het vorige. Kunt u het veranderen zodat het met stapjes van twee telt? Of met halve stapjes, zodat het 1, 1.5, 2, 2.5, 3, ...telt?

Aanwijzing: U zult de **STEP**-clausule van het **FOR**-commando moeten aanpassen.

3. Kunt u één van de tafels van vermenigvuldiging laten weergeven?

Op dit moment geeft het de antwoorden van de tafel van 1 weer, omdat het met stapjes van één telt. Kunt u het met stapjes van drie laten tellen en de tafel van 3 weergeven?

Aanwijzing: U zult het **FOR**-commando moeten aanpassen.

4. Kunt u de tafels van vermenigvuldiging van 1 × 1 tot 10 × 10 laten weergeven?

Aanwijzing: U kunt een puntkomma (;) achter het **PRINT**-commando zetten, zodat er meer dan één waarde per regel op het scherm past.
Aanwijzing: Het **PRINT**-commando zonder argument springt gewoon naar het begin van de volgende regel.
Aanwijzing: Mogelijk heeft u meerdere **FOR**-lussen nodig, de één binnen de andere.

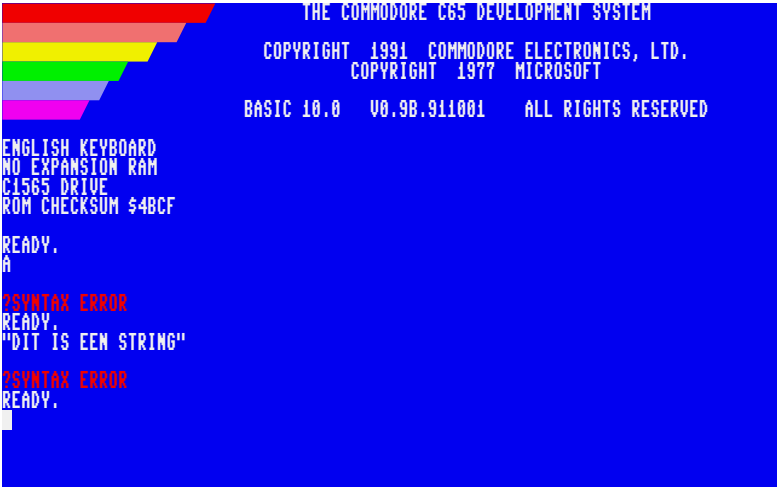
EERSTE STAPPEN MET TEKST EN GETALLEN

In het vorige gedeelte gebruikten we zowel getallen als tekst. Tekst in een computer ontstaat door afzonderlijke letters en andere symbolen aaneen te rijgen. Zo'n aaneenschakeling noemen we *strings*. Ook de afzonderlijke letters en symbolen hebben een naam: we noemen ze *tekens*. Het Engelse woord character komt uit de drukkerswereld, waar hij verwijst naar elk symbool dat op een bladzijde gedrukt kan worden. Voor computers heeft het vrijwel dezelfde betekenis en de verzameling tekens die een computer kan weergeven noemen we, nogal fantasieloos, een *tekenset*..

Indien de MEGA65 een vorm van invoer verwacht, zoekt hij gewoonlijk naar één van vier dingen:

1. *een sleutelwoord* zoals **PRINT** of **STEP**, woorden die een speciale betekenis voor de computer hebben;
2. *een variabelenaam* zoals **I** of **A$**, welke de computer vervolgens gebruikt om een stukje informatie op te slaan of op te halen;
3. *een getal* zoals **42** of **-30.3137**; of
4. *een string* zoals **"HALLO COMPUTER"** of **"23 KILOMETER"**.

Soms kunt u kiezen welk soort invoer u geeft, terwijl die keuze andere keren beperkter is. Wat de computer aanvaardt, hangt af van wat u op dat moment doet. In het vorige gedeelte ontdekten we bijvoorbeeld dat u de computer een sleutelwoord of een getal kunt geven wanneer hij **READY** weergeeft. Denkt u dat de computer alle vier de soorten aanvaardt wanneer hij **READY.** weergeeft? We weten al dat u sleutelwoorden en getallen kunt invoeren, maar hoe zit het met variabelenamen en strings? Laten we een variabelenaam intypen, bijvoorbeeld **N**, RETURN indrukken en kijken wat er gebeurt. Daarna proberen we het met een string, bijvoorbeeld **"DIT IS EEN STRING"**.



Elke keer krijgt u een syntaxisfout, welke u vertelt dat de computer de invoer die u hem gegeven heeft niet begrijpt. Laten we beginnen met de omstandigheid waarin u de variabele typte: als u de computer alleen maar de naam van een opgeslagen stukje gegevens geeft, heeft hij er geen flauw benul van wat u wilt dat hij doet. Hetzelfde geldt wanneer u de computer een stukje gegevens geeft, zoals een string, zonder hem te vertellen wat hij ermee moet doen.

Zoals we in het vorige gedeelte echter hebben gezien, kunnen we de computer met het **PRINT**-commando vertellen dat we het stukje gegevens dat in een variabele is opgeslagen, willen zien. In plaats daarvan zouden we dus **PRINT N** kunnen intypen; de computer weet dan wat hij moet doen en geeft het stukje gegevens weer dat in de variabele **N** is opgeslagen.

Het gebruik van het **PRINT**-commando is zelfs zo algemeen dat programmeurs het vervelend vonden om steeds het **PRINT**-commando te moeten intypen. Ze bedachten daarom een sneltoets: typt u een vraagteken, d.w.z. een **?**, dan weet de computer dat u **PRINT** bedoelt. Typt u bijvoorbeeld **? N**, dan gebeurt hetzelfde als wanneer u **PRINT N** typt. U moet na elk commando uiteraard RETURN indrukken om de computer te vertellen dat u wilt dat hij verwerkt wat u getypt heeft. Vanaf nu gaan we ervan uit dat u daaraan denkt zonder dat we u eraan hoeven te herinneren.

De **?**-sneltoets werkt ook wanneer u de computer een commando als onderdeel van een programma laat onthouden. Typt u dus **1 ? N** en daarna **LIST**, dan ziet u **1 PRINT N**, zoals te zien is in de volgende schermafdruk:

```
READY.
A

?SYNTAX ERROR
READY.
"DIT IS EEN STRING"

?SYNTAX ERROR
READY.
PRINT A
 0

READY.
? A
 0

READY.
1 ? A
LIST

1 PRINT A

READY.
```

Zoals we in het vorige gedeelte zagen, is er nog geen waarde in de variabele **N** opgeslagen; wanneer de computer dus kijkt wat erin staat, vindt hij niets. Omdat **N** een *numerieke variabele* is, betekent "niets" hier nul. Was het een *stringvariabele*, dan had hij letterlijk niets gevonden. Dat kunnen we uitproberen, maar eerst moeten we uitleggen hoe we de computer vertellen dat we het over een stringvariabele hebben. Dat doen we door een dollarteken, d.w.z. een **$**, achter de naam van de variabele te zetten. Zetten we dus een **$** achter de variabelenaam **N**, dan verwijst dat naar een stringvariabele genaamd **N$**.

Met het inmiddels vertrouwde **PRINT**-commando (of de **?**-sneltoets) kunnen we met deze variabelen experimenteren en bekijken wat erin staat. We hebben echter een eenvoudige manier nodig om er waarden in te zetten. Gelukkig zijn we niet de eersten die waarden in variabelen willen zetten; daarom bestaat er een **LET**-commando. Met het **LET**-commando zet u een waarde in een variabele. Zo kunnen we de computer bijvoorbeeld vertellen:

```
LET N = 5.3
```

Hiermee vertelt u de computer dat hij de waarde 5.3 in de variabele **N** moet zetten. Met het **PRINT**-commando kunnen we vervolgens controleren of dat gelukt is. Op dezelfde manier kunnen we een waarde in de variabele **N$** zetten, bijvoorbeeld met een regel als:

```
LET N$ = "DE KONING DER AARDAPPELMENSEN"
```

Als we deze uitproberen, zien we ongeveer het volgende:

```
? N
 0

READY.
1 ? N
LIST

1 PRINT N

READY.
LET N = 5.3

READY.
? N
 5.3

READY.
LET N$ = "DE KONING DER AARDAPPELMENSEN"

READY.
? N$
DE KONING DER AARDAPPELMENSEN

READY.
```

Eerder vermeldden we al dat **N** een numerieke variabele is en **N$** een stringvariabele. Dat betekent dat we alleen getallen in **N** kunnen zetten en alleen strings in **N$**. Proberen we het verkeerde soort gegevens in een variabele te zetten, dan laat de computer ons weten dat het soort gegevens niet past op de plaats waar we het willen zetten. Hij geeft ons dan een **TYPE MISMATCH ERROR**, zoals hier:

```
READY.
LET N = 5.3

READY.
? N
 5.3

READY.
LET N$ = "DE KONING DER AARDAPPELMENSEN"

READY.
? N$
DE KONING DER AARDAPPELMENSEN

READY.
LET N = "HEER FIEBEL"

?TYPE MISMATCH ERROR
READY.
LET N$ = 42

?TYPE MISMATCH ERROR
READY.
```

Dit brengt ons bij een vrij belangrijk punt: **N** en **N$** zijn afzonderlijke variabelen, ook al lijken hun namen op elkaar. Dit geldt voor alle mogelijke variabelenamen: eindigt de variabelenaam op een **$**-teken, dan is het een stringvariabele die geheel losstaat van de gelijknamige numerieke variabele. Om een beetje jargon te gebruiken: elke variabele van een bepaald *type* heeft zijn eigen afzonderlijke *naamruimten*.

(Er zijn overigens nog vier andere variabelenaamruimten die we nog niet hebben besproken: integer-variabelen, herkenbaar aan een **%**-teken achter de naam, bijv. **N%** en arrays van numerieke variabelen, stringvariabelen of integer-variabelen. Maakt u zich daar voorlopig nog geen zorgen over; daar komen we later nog op terug.)

Tot nu toe hebben we variabelen alleen waarden gegeven in de directe modus, of met behulp van constructies als **FOR**-lussen. We hebben echter nog niet gezien hoe we gegevens van de gebruiker kunnen krijgen terwijl een programma draait. Een manier om dat te doen is met het **INPUT**-commando.

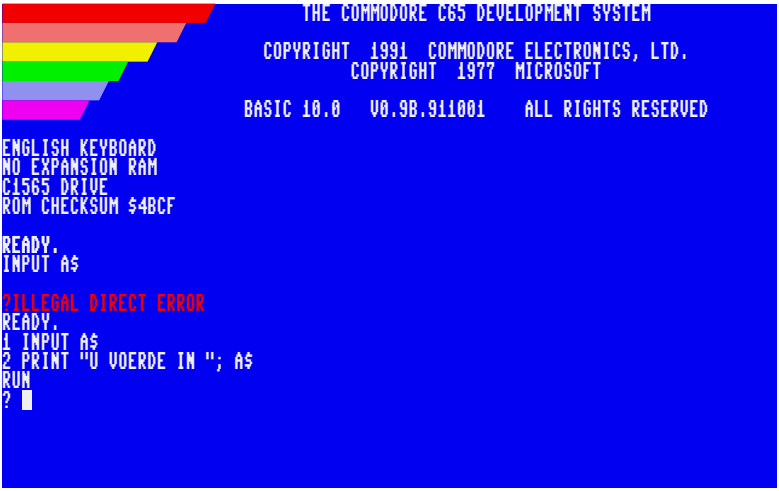
INPUT is heel eenvoudig te gebruiken: we hoeven alleen maar aan te geven in welke variabele de invoer moet komen. Om de computer bijvoorbeeld de gebruiker te laten vragen iets in te typen dat in de variabele **A$** moet komen, kunnen we zoiets gebruiken als **INPUT A$**. Het enige nadeel van het **INPUT**-commando is dat u het niet in de directe modus kunt gebruiken. Probeert u het toch, dan meldt de computer **ILLEGAL DIRECT ERROR**. Probeer het maar; u zult ongeveer het volgende zien:



Dat betekent dat het **INPUT**-commando alleen als onderdeel van een programma gebruikt kan worden. In plaats daarvan kunnen we dus iets doen als volgt:

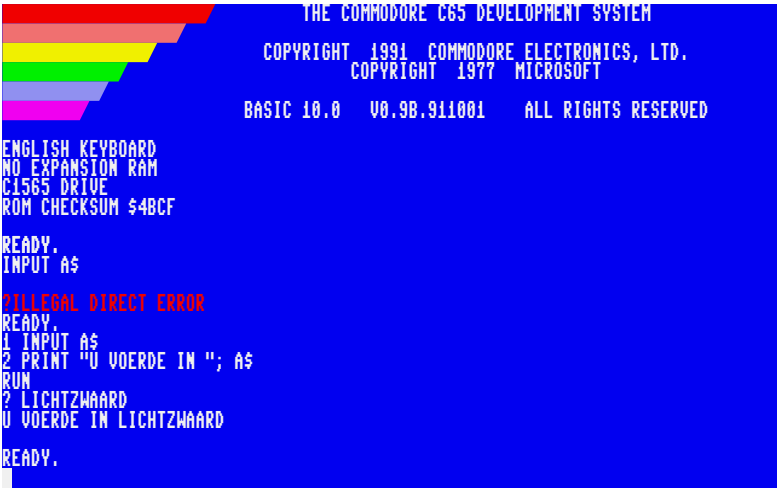
```
1 INPUT A$
2 PRINT "U VOERDE IN "; A$
RUN
```

Wat denkt u dat dit zal doen? De eerste regel vraagt de computer om invoer die in de variabele **A$** gezet moet worden en de tweede regel geeft de string **"U VOERDE IN"** weer, gevolgd door wat het **INPUT**-commando van de gebruiker heeft gelezen. Laten we het uitproberen:



Had u dit verwacht? Wat doet dat vraagteken daar? Het **?** hier is de wijze waarop de computer u laat weten dat een programma op invoer van u wacht. Dat betekent dat de computer hetzelfde teken, **?**, gebruikt om twee verschillende dingen aan te duiden: typt u het als onderdeel van een programma of in de directe modus, dan is het een verkorte vorm van het **PRINT**-commando. Dat is zo wanneer u het intypt. Toont de computer het evenwel aan u, dan heeft het een andere betekenis: namelijk dat de computer wacht tot u iets intypt. Er is ook nog een derde wijze waarop het **?**-teken gebruikt wordt. Heeft u gemerkt hoe? Het geeft het begin van een foutmelding aan. Een syntaxisfout wordt bijvoorbeeld aangegeven met **?SYNTAX ERROR**. Indien een teken of iets anders in verschillende omstandigheden of contexten verschillende betekenissen heeft, zeggen we dat het *contextafhankelijk* is.

Laten we terugkeren naar ons voorbeeld: als u nu iets intypt en op RETURN drukt om de computer te laten weten dat u klaar bent, gaat het programma verder, zoals hier:



Natuurlijk wisten we niet echt wat we moesten intypen, want het programma gaf de gebruiker geen enkele aanwijzing over wat de programmeur wilde dat hij deed. We zouden dus moeten proberen wat instructies te geven. Als we de gebruiker bijvoorbeeld zijn naam wilden laten typen, konden we een boodschap weergeven met het verzoek zijn naam te typen, zoals hier:

```
1 PRINT "HOE HEET U"
2 INPUT A$
3 PRINT "HALLO "; A$
```

Als we dit programma nu uitvoeren, krijgt de gebruiker een aanwijzing over wat we van hem verwachten en de hele ervaring zal voor hem een stuk duidelijker worden:

```
ROM CHECKSUM $4BCF

READY.
INPUT A$

?ILLEGAL DIRECT ERROR
READY.
1 INPUT A$
2 PRINT "U VOERDE IN "; A$
RUN
? LICHTZWAARD
U VOERDE IN LICHTZWAARD

READY.
1 PRINT "HOE HEET U"
2 INPUT A$
3 PRINT "HALLO "; A$
RUN
HOE HEET U
? LENNART
HALLO LENNART

READY.
```

Wanneer we het programma uitvoeren, zien we eerst de boodschap **HOE HEET U** uit regel 1. De computer geeft de dubbele aanhalingstekens niet weer, omdat die de computer alleen maar vertelden dat het stukje gegevens ertussen een string is. De string zelf is alleen het deel dat ertussen staat.

Daarna zien we opnieuw het **?**-teken en de knipperende cursor, welke ons vertelt dat de computer op invoer van ons wacht. De rest van het programma is *geblokkeerd* en kan niet verdergaan totdat we het stukje gegevens hebben getypt. Zodra we de invoer hebben getypt, slaat de computer die op in de variabele **A$** en kan hij verder. Wanneer hij dus bij regel 3 van het programma komt, heeft hij alles wat hij nodig heeft en geeft hij zowel de **HALLO**-boodschap weer als de gegevens die in de variabele **A$** zijn opgeslagen.

Merk op dat het woord **LENNART** nergens in het programma voorkomt. Het bestaat alleen in de variabele. Dit vermogen om gegevens te verwerken die geen deel van het programma uitmaken, is een van de dingen die computerprogramma's zo krachtig en geschikt maken voor zoveel doeleinden. We hoeven alleen maar de invoer te wijzigen en we kunnen dan andere uitvoer krijgen.

Ons programma kunnen we bijvoorbeeld keer op keer uitvoeren en elke keer andere invoer geven, waarna het programma zijn uitvoer aanpast aan wat we typen. Vrij handig, nietwaar? Laat de rest van de bemanning het maar eens uitproberen:



We kunnen zien dat het programma elke keer de boodschap weergeeft, aangepast aan de invoer die u getypt heeft... Tot we bij **REMWAALS, HEER** komen. Zoals altijd veroorzaakt meneer Remwaals problemen. In dit geval kon hij het niet laten om zijn aanspreekvorm achter zijn naam te zetten.

De computer heeft ons een soort foutmelding gegeven, **?EXTRA IGNORED**. De melding is niet in het rood geschreven en het woord **ERROR** ontbreekt op het eind. Dat betekent dat het een waarschuwing is in plaats van een fout. Omdat het slechts een waarschuwing is, gaat het programma gewoon verder. Er is echter wel iets gebeurd: de computer heeft de aanspreekvorm **HEER** van meneer Remwaals genegeerd, dat wil zeggen dat hij de extra invoer heeft genegeerd. Dat komt doordat het **INPUT**-commando niet daadwerkelijk een hele regel invoer leest. Het leest veeleer *één stukje gegevens*. Het **INPUT**-commando neemt aan dat een stukje gegevens eindigt aan het eind van een invoerregel of wanneer het een komma (**,**) of dubbele punt (**:**) tegenkomt.

Wilt u een van die symbolen gebruiken, dan moet u het hele stukje gegevens tussen dubbele aanhalingstekens zetten. Als meneer Remwaals deze handleiding had gelezen in plaats van de computer op de bonnefooi te gebruiken, had hij geweten dat hij **"REMWAALS, HEER"** moest typen, compleet met de dubbele aanhalingstekens, om het programma goed te laten draaien. Het is belangrijk dat de aanhalingstekens om het hele stukje gegevens staan, want anders denkt de computer dat het eerste aanhalingsteken het begin van een nieuw stukje gegevens aangeeft. Hieronder ziet u het verschil dat dit maakt:

```
LIST

1 PRINT "HOE HEET U"
2 INPUT A$
3 PRINT "HALLO "; A$

READY.
RUN
HOE HEET U
? "HEER REMWAALS"
HALLO HEER REMWAALS

READY.
RUN
HOE HEET U
? REMWAALS", "HEER.
?EXTRA IGNORED
HALLO REMWAALS"

READY.
```

Al kan dit soms behoorlijk vervelend zijn, het heeft een doel: met het **INPUT**-commando kunt u meer dan één stukje gegevens lezen. Dat doet u door meer dan één variabele achter het **INPUT**-commando te zetten, telkens gescheiden door een komma. Het **INPUT**-commando verwacht dan meerdere stukjes gegevens. Zo zouden we bijvoorbeeld kunnen vragen om iemands naam en leeftijd, met een programma als dit:

```
1 PRINT "HOE HEET U EN WAT IS UW LEEFTIJD"
2 INPUT A$, A
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A; " JAAR OUD."
```

Voeren we dit programma uit, dan kunnen we de twee stukjes gegevens op één regel typen wanneer de computer ons de **?**-prompt toont, bijvoorbeeld **LENNART, 3000000**. Let op de komma die de twee stukjes gegevens, **LENNART** en **3000000**, van elkaar scheidt. Het is ook de moeite waard om op te merken dat we in het getal 3000000 geen scheidingstekens voor duizendtallen hebben gezet. Hadden we dat wel gedaan, dan zou de computer denken dat we drie afzonderlijke stukjes gegevens bedoelden, **3**, **000** en **000** en dat is niet wat we bedoelden. Laten we dus eens bekijken hoe het eruitziet wanneer we **LENNART, 3000000** als invoer aan het programma geven:

```
READY.
RUN
HOE HEET U
? REMWAALS", "HEER
?EXTRA IGNORED
HALLO REMWAALS"

READY.
LIST

1 PRINT "HOE HEET U EN WAT IS UW LEEFTIJD"
2 INPUT A$, A
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."
READY.

RUN
HOE HEET U EN WAT IS UW LEEFTIJD
? LENNART, 3000000
HALLO LENNART
U BENT 3000000  JAAR OUD.

READY.
```

In dit geval leest het **INPUT**-commando de twee stukjes gegevens en zet het eerste in de variabele **A$** en het tweede in de variabele **A**. Wanneer het programma bij regel 3 komt, geeft het **HALLO** weer, gevolgd door het eerste stukje gegevens. Wanneer het vervolgens bij regel 4 komt, geeft het de string **U BENT** weer, gevolgd door de inhoud van de variabele **A**, het getal 3000000 en ten slotte de string **JAAR OUD**.

Het is ook mogelijk om telkens maar één stukje gegevens tegelijk te geven. In dat geval vraagt het **INPUT**-commando met een prompt van twee vraagtekens om het tweede gegeven, d.w.z. **??**. Zodra het het tweede gegeven heeft. (Staan er meer dan twee variabelen op het **INPUT**-commando, dan toont de computer nog steeds dezelfde **??**-prompt, in plaats van steeds meer vraagtekens af te drukken.)

Als we dit met ons programma proberen, zien we deze **?**- en **??**-prompts en hoe het eerste stukje gegevens in **A$** terechtkomt, omdat dit de eerste variabele in het **INPUT**-commando is. Het tweede gegeven komt in **A** terecht, omdat **A** de tweede variabele na het **INPUT**-commando is. Zo ziet het eruit wanneer we deze invoer aan ons programma geven:

```
READY.
LIST

1 PRINT "HOE HEET U EN WAT IS UW LEEFTIJD"
2 INPUT A$, A
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."
READY.

RUN
HOE HEET U EN WAT IS UW LEEFTIJD
? LENNART, 3000000
HALLO LENNART
U BENT 3000000  JAAR OUD.

READY.
RUN
HOE HEET U EN WAT IS UW LEEFTIJD
? LENNART
?? 3000000
HALLO LENNART
U BENT 3000000  JAAR OUD.

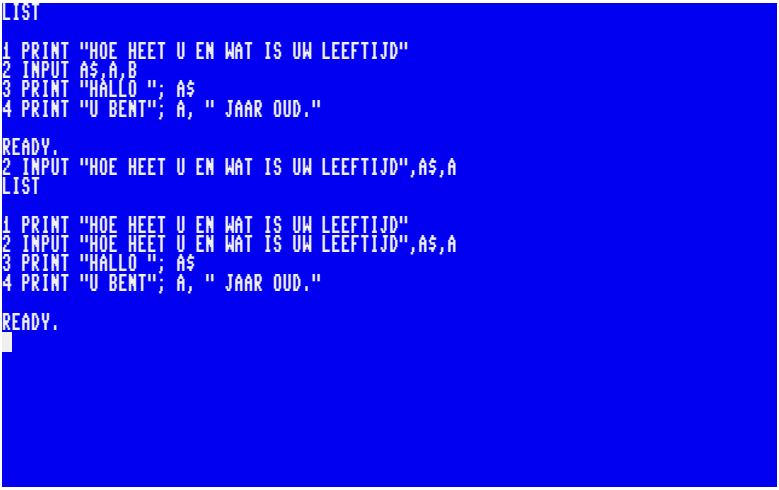
READY.
```

Tot nu toe hebben we de gebruiker om gegevens gevraagd door met een **PRINT**-commando de boodschap weer te geven en vervolgens met een **INPUT**-commando aan de computer kenbaar te maken in welke variabelen we de gegevens willen opnemen. Net als bij het **PRINT**-commando gebeurt dit echter zo vaak dat er een verkorte vorm voor bestaat. Het heeft bovendien het voordeel dat het er tijdens het uitvoeren netter uitziet en het programma iets korter maakt. De verkorte vorm bestaat eruit de te tonen boodschap na het **INPUT**-commando te plaatsen, maar vóór de eerste variabele.

We kunnen ons programma aanpassen om deze werkwijze te gebruiken. Eerst kunnen we regel 3 zodanig wijzigen dat de prompt na het **INPUT**-commando komt. Dat kunnen we op twee manieren doen. De eerste is eenvoudig een nieuwe regel 3 in te typen. De computer vervangt dan automatisch de oude regel 3 door de nieuwe.

Zoals we inmiddels al een paar keer hebben vermeld, zijn programmeurs luie wezens en dus is er een verkorte vorm: als u de regel die u wilt wijzigen op het scherm kunt zien, kunt u met de cursortoetsen naar die regel gaan, hem op het scherm bewerken en vervolgens op RETURN drukken om de computer de nieuwe versie van de regel te laten aanvaarden.

Hoe dan ook, u kunt controleren of de wijzigingen zijn geslaagd door het **LIST**-commando op een willekeurige lege regel van het scherm in te typen. Dit toont de herziene versie van het programma. Bijvoorbeeld:



Toch hebben we nog een klein probleempje: regel 1 geeft de boodschap **HOE HEET U EN WAT IS UW LEEFTIJD** weer, waarna regel 2 hem nog eens weergeeft! We willen dat de boodschap slechts één keer verschijnt. We willen daarom regel 1 zodanig wijzigen dat hij dit niet meer doet. Omdat er op regel 1 geen ander commando staat dat we willen behouden, kan die regel gewoon leeg worden. We kunnen dus zoiets intypen:

```
1
```

We kunnen bevestigen dat de inhoud van de regel is verwijderd door het **LIST**-commando opnieuw uit te voeren, zoals hier:

```
1 PRINT "HOE HEET U EN WAT IS UW LEEFTIJD"
2 INPUT A$,A,B
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."

READY.
2 INPUT "HOE HEET U EN WAT IS UW LEEFTIJD",A$,A
LIST

1 PRINT "HOE HEET U EN WAT IS UW LEEFTIJD"
2 INPUT "HOE HEET U EN WAT IS UW LEEFTIJD",A$,A
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."

READY.
1
LIST

2 INPUT "HOE HEET U EN WAT IS UW LEEFTIJD",A$,A
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."

READY.
```

Heeft u iets interessants opgemerkt? Toen we de computer opdroegen regel 1 van het programma leeg te maken, heeft hij hem volledig verwijderd! Dat komt doordat de computer van mening is dat een lege regel nergens voor dient. Hij zorgt er bovendien voor dat uw programma's niet vol lege regels komen te staan wanneer u veel wijzigingen in uw programma's aanbrengt.

Het is ook mogelijk om met DELETE een reeks regels te verwijderen. U kunt bijvoorbeeld (maar doe dit nu nog niet) regels 3-4 verwijderen met:

```
DELETE 3-4
```

U kunt meer lezen over het DELETE-commando in de Naslag BASIC 65-commando's.

Nu we dat uit de weg hebben, kunt u ons programma nu uitvoeren en kijken wat er gebeurt. Typ zoals gebruikelijk gewoon het **RUN**-commando in en druk op RETURN. U ziet dan zoiets:

```
1 PRINT "HOE HEET U EN WAT IS UW LEEFTIJD"
2 INPUT A$,A,B
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."

READY.
2 INPUT "HOE HEET U EN WAT IS UW LEEFTIJD",A$,A
LIST

1 PRINT "HOE HEET U EN WAT IS UW LEEFTIJD"
2 INPUT "HOE HEET U EN WAT IS UW LEEFTIJD",A$,A
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."

READY.
1
LIST

2 INPUT "HOE HEET U EN WAT IS UW LEEFTIJD",A$,A
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."

READY.
RUN
HOE HEET U EN WAT IS UW LEEFTIJD
```

We zien daar onze prompt **HOE HEET U EN WAT IS UW LEEFTIJD**, maar nu verschijnt de cursor zonder enig vraagteken (**?**). Dat komt doordat we een komma (**,**) achter de boodschap in het **INPUT**-commando hebben gezet. Om het vraagteken te krijgen moeten we in plaats daarvan een puntkomma (**;**) achter de boodschap zetten, zoals dit:

```
INPUT "HOE HEET U EN WAT IS UW LEEFTIJD"; A$, A
```

Als u het programma nu uitvoert, zou u moeten zien wat we zoeken:

```
LIST

2 INPUT "HOE HEET U EN WAT IS UW LEEFTIJD";A$,A
3 PRINT "HALLO "; A$
4 PRINT "U BENT"; A, " JAAR OUD."

READY.
RUN
HOE HEET U EN WAT IS UW LEEFTIJD? LENNART,3000000
HELLO LENNART
U BENT 3000000  JAAR OUD.

READY.
```

Oefeningen om te proberen

1. Kunt u het programma iemand naar zijn of haar naam laten vragen en vervolgens naar de lievelingskleur?

Op dit moment vraagt het programma naar de naam en de leeftijd. Kunt u het zo veranderen dat het in plaats van de leeftijd de lievelingskleur weergeeft?

Aanwijzing: Wat voor soort gegevens is de leeftijd? Is het een numerieke waarde of een string? Is dat hetzelfde soort gegevens als de naam van een kleur?

2. Kunt u een programma schrijven dat iemand naar zijn of haar naam vraagt, de begroeting weergeeft en vervolgens naar de leeftijd vraagt en dat antwoord weergeeft?

Op dit moment verwacht het programma beide stukjes gegevens tegelijkertijd. Dit betekent dat het programma geen bericht over het eerste stukje gegevens kan weergeven voordat het beide stukjes heeft ontvangen. Verander het programma zodat u in plaats daarvan een interactie krijgt zoals de volgende:

```
HOE HEET U? DIEPE GEDACHTE
HALLO DIEPE GEDACHTE
WAT IS HET ANTWOORD? 42
VOLGENS U WAS HET ANTWOORD 42
```

Aanwijzing: U zult meer regels nodig hebben in uw programma, zodat u meer dan één **INPUT**- en **PRINT**-commando kunt gebruiken.

3. Kunt u een programma schrijven dat enkele vragen stelt en vervolgens de lijst met gegeven antwoorden weergeeft?

Bedenk enkele vragen die u iemand zou willen stellen en schrijf vervolgens een programma dat die vragen stelt, de antwoorden onthoudt en ze met een passende melding weergeeft. Het uitvoeren van uw programma zou er bijvoorbeeld als volgt uit kunnen zien:

```
HOE HEET U? FRODO
HOE OUD BENT U? 33
WAT IS UW LIEVELINGSETEN? ALLES!
BEDANKT VOOR DE ANTWOORDEN.
U HEET FRODO
U BENT 33 JAAR OUD
UW LIEVELINGSETEN IS ALLES!
```

Aanwijzing: U zult meer regels in uw programma nodig hebben om de verschillende **INPUT**- en **PRINT**-commando's op te nemen.

Aanwijzing: U zult goed moeten nadenken over welke variabelenamen u zult gebruiken.

EENVOUDIGE BESLISSINGEN NEMEN

In het vorige gedeelte hebben we geleerd hoe we tekst en numerieke gegevens invoeren en weergeven. De programma's hebben echter steeds gewoon de instructieregels in volgorde gevolgd, zonder dat ze op basis van de ingevoerde gegevens konden beslissen wat er moest gebeuren.

In dit gedeelte zien we hoe we eenvoudige beslissingen kunnen nemen met behulp van de **IF**- en **THEN**-commando's. Het **IF**-commando controleert of iets waar of onwaar is en als dat het geval is, zorgt het ervoor dat de computer het commando uitvoert dat na het **THEN**-commando staat.

De computer beslist of iets waar of onwaar is door de aangeleverde gegevens te bewerken met behulp van één van meerdere symbolen. Deze symbolen worden daarom *operatoren* genoemd. Omdat ze twee dingen met elkaar vergelijken, hangen ze bovendien af van de relatie tussen die dingen. Om die reden worden ze *relationele* operatoren genoemd.Daartoe behoren de volgende:

- Gelijkheid (=). Zo zou `3 = 3` waar zijn, terwijl `3 = 2` onwaar zou zijn.
- Kleiner dan (<). Zo zou `1 < 3` waar zijn, terwijl zowel `3 < 3` als `3 < 1` onwaar zou zijn.
- Groter dan (>). Zo zou `3 > 1` waar zijn, terwijl zowel `3 > 3` als `1 > 3` onwaar zou zijn.

Omdat het vaak voorkomt dat u wilt nagaan of iets gelijk is aan of groter dan iets anders, of gelijk is aan of kleiner dan iets anders, bestaan hiervoor verkorte schrijfwijzen. Wilt u daarnaast testen of iets niet gelijk is aan iets anders, dan is daar ook een relationele operator voor:

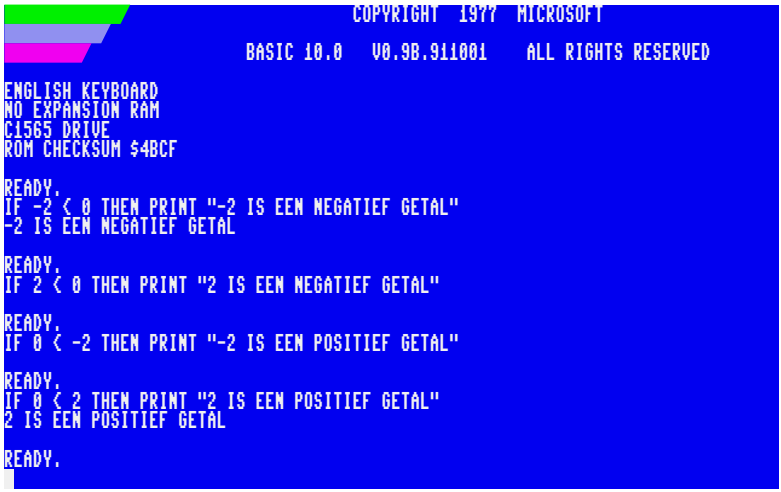
- Ongelijkheid, in het Engels gewoonlijk *not equal* genoemd (**<>**). Dit wijkt af van het wiskundige teken voor niet-gelijk, $\neq$, omdat de tekenset van de MEGA65 geen teken bevat dat daarop lijkt. De programmeurs die BASIC voor de MEGA65 schreven, gebruikten daarom de tekens voor groter dan en kleiner dan samen om aan te geven dat iets kleiner dan of groter dan is, dat wil zeggen: niet gelijk aan. Zo zou **1 <> 3** waar zijn, terwijl **3 <> 3** onwaar zou zijn.
- Kleiner dan of gelijk aan (**<=**). Zo zouden **1 < 3** en **3 <= 3** waar zijn, terwijl **4 < 3** onwaar zou zijn.
- Groter dan of gelijk aan (**>=**). Zo zouden **3 >= 1** en **3 >= 3** waar zijn, terwijl **1 >= 3** onwaar zou zijn.

Een handig trucje wanneer u zich niet meer kunt herinneren hoe de tekens **(**<) en **(**>) wijzen: de kant met de meeste lijnuiteinden is de kant waar het grootste getal moet staan. Zo heeft het teken **(**<) links één punt, maar aan de rechterkant twee lijnuiteinden. Wil een uitspraak met **(**<) waar zijn, dan moet het getal aan de linkerkant kleiner zijn dan het getal aan de rechterkant. Dit trucje werkt zelfs voor het gelijkteken, **(**=), want dat heeft aan beide kanten evenveel uiteinden; u kunt dus onthouden dat de getallen aan beide kanten gelijk moeten zijn. Het werkt ook wanneer u twee tekens samen gebruikt, zoals **(**>=): het is waar wanneer de voorwaarde voor één van de tekens waar is. In dit geval heeft het teken **(**>) dus meer uiteinden aan de linkerkant dan aan de rechterkant, zodat het waar is wanneer het getal aan de linkerkant groter is dan het getal aan de rechterkant. Aangezien het teken **(**=) aan elke kant twee uiteinden heeft, is het ook waar wanneer de twee getallen gelijk zijn.

Met deze relationele operatoren kunnen we een regel schrijven die iets doet, maar alleen wanneer iets waar of onwaar is. U kunt dit eens uitproberen met een paar voorbeelden:

```
IF -2 < 0 THEN PRINT "-2 IS EEN NEGATIEF GETAL"
IF 2 < 0 THEN PRINT "2 IS A EEN NEGATIEF GETAL"
IF 0 < -2 THEN PRINT "-2 IS EEN POSITIEF GETAL"
IF 0 < 2 THEN PRINT "2 IS EEN POSITIEF GETAL"
```

Deze commando's werken prima in de directe modus, dus u kunt ze zo direct in de computer intypen om te zien wat ze doen. Dat kan handig zijn om te testen of u de logica goed heeft wanneer u een **IF** – **THEN**-commando opstelt. Typt u deze commando's in, dan ziet u ongeveer het volgende:



We kunnen zien dat alleen de **PRINT**-commando's werden uitgevoerd die op een **IF**-commando volgden waarvan de waarde waar was. De computer negeerde de rest zwijgend. We kunnen deze commando's echter ook in een programma opnemen. Laten we dus een programmaatje maken dat om twee getallen vraagt en zegt of ze gelijk zijn, of dat het ene groter of kleiner is dan het andere. Denk, voordat u naar het programma kijkt, eerst na over hoe u het zou aanpakken en kijk of u er zelf uit kunt komen. De aanwijzing die wij u geven, is dat het **IF**-commando ook de naam van een variabele aanvaardt, niet alleen getallen. U kunt dus bijvoorbeeld iets doen als **IF A > B THEN PRINT "IETS"**. Het programma staat op de volgende bladzijde, zodat u niet kunt spieken voordat u erover nagedacht heeft!

Heeft u het geprobeerd? Er zijn veel verschillende manieren waarop het gedaan kan worden, maar dit is wat wij bedacht hebben:

```
1 INPUT "WAT IS HET EERSTE GETAL"; A
2 INPUT "WAT IS HET TWEEDE GETAL"; B
3 IF A = B THEN PRINT "DE GETALLEN ZIJN GELIJK"
4 IF A > B THEN PRINT "HET EERSTE GETAL IS GROTER"
5 IF B > A THEN PRINT "HET TWEEDE GETAL IS GROTER"
```

Vervolgens kunnen we het programma zo vaak uitvoeren als we willen en de computer kan ons vertellen welke van de twee getallen die we hem geven het grootste is, of dat ze gelijk zijn:

```
1 INPUT "WAT IS HET EERSTE GETAL";A
2 INPUT "WAT IS HET TWEEDE GETAL";B
3 IF A = B THEN PRINT "DE GETALLEN ZIJN GELIJK"
4 IF A > B THEN PRINT "HET EERSTE GETAL IS GROTER"
5 IF B > A THEN PRINT "HET TWEEDE GETAL IS GROTER"

RUN
WAT IS HET EERSTE GETAL? 2
WAT IS HET TWEEDE GETAL? 2
DE GETALLEN ZIJN GELIJK

READY.
RUN
WAT IS HET EERSTE GETAL? 3
WAT IS HET TWEEDE GETAL? 4
HET TWEEDE GETAL IS GROTER

READY.
RUN
WAT IS HET EERSTE GETAL? 10
WAT IS HET TWEEDE GETAL? 2
HET EERSTE GETAL IS GROTER

READY.
```

Let u erop dat we in dit programma geen vaste getallen in het **IF**-commando gebruikten, maar in plaats daarvan namen van variabelen. Dit is een van de krachtigste eigenschappen van het programmeren, samen met de mogelijkheid om op basis van gegevens beslissingen te nemen. Doordat de programmeur gegevens bij naam kan aanspreken, ongeacht hun huidige waarde of hoe ze daar gekomen zijn, kan hij zeer flexibele programma's maken.

Laten we eens nadenken over een iets interessanter voorbeeld: een spelletje "raad het getal". Daarvoor hebben we een getal nodig dat iemand moet raden en vervolgens moeten we de gokken aannemen en aangeven of de gok juist was of niet. Is de gok onjuist, dan moeten we de gebruiker vertellen of het juiste getal hoger of lager is.

De meeste ingrediënten voor zo'n programma hebben we al geleerd: we kunnen **LET** gebruiken om een variabele op het geheime getal te zetten, **INPUT** om de gebruiker naar zijn gok te vragen en vervolgens **IF**, **THEN** en **PRINT** om de gebruiker te vertellen of zijn gok juist was of niet. Laten we dus iets maken. Ook nu geldt: als u wilt, kunt u een paar minuten stilstaan, nadenken en experimenteren om te zien of u zo'n programma zelf kunt maken.

Zo hebben wij het gedaan. U hoeft zich echter geen zorgen te maken als u het op een heel andere manier heeft gedaan: er zijn vaak veel manieren om een programma te schrijven dat een bepaalde taak uitvoert.

```
1 SN=23
2 PRINT "RAAD EEN GETAL TUSSEN 1 EN 100"
3 INPUT "WAT IS UW GOK"; G
4 IF G<SN THEN PRINT "MIJN GETAL IS GROTER"
5 IF G>SN THEN PRINT "MIJN GETAL IS KLEINER"
6 IF G=SN THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"
```

De eerste regel zet ons geheime getal in de variabele **SN**. De tweede regel toont een bericht dat de gebruiker vertelt wat de bedoeling is. De derde regel vraagt de gebruiker naar zijn gok en zet die in de variabele **G**. De vierde, vijfde en zesde regel controleren vervolgens of de gok juist is of niet en zo niet, welk bericht er getoond moet worden. Dat gebeurt met het **IF**-commando en een passende relationele operator bij elke beslissing. Dit werkt goed, maar slechts tot op zekere hoogte. Bijvoorbeeld:

```
LIST

1 S$=23
2 PRINT "RAAD HET GETAL TUSSEN 1 EN 100"
3 INPUT "WAT IS UW GOK";G
4 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
5 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
6 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"

READY.
RUN
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 10
MIJN GETAL IS GROTER

READY.
RUN
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 23
GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!

READY.
```

We kunnen zien dat het programma het bericht toont, om een gok vraagt en gepast reageert. Willen we echter nog eens raden, dan moeten we voor elke extra gok opnieuw het **RUN**-commando gebruiken. Dat is vanuit het perspectief van de gebruiker niet zo mooi. Toch zal dat waarschijnlijk niet lang een probleem zijn, want de gebruiker kan het geheime getal in de programmalijst op het scherm zien!

Deze problemen willen we dus graag oplossen. Laten we beginnen met het verbergen van de programmalijst. Eerder noemden we al dat wanneer het scherm scrollt, alles wat bovenaan het scherm stond verdwijnt. We kunnen er dus simpelweg voor zorgen dat het scherm voldoende scrollt, zodat een zichtbare programmalijst niet langer zichtbaar is. Dat kunnen we doen met **PRINT** en een **FOR**-lus. Het scherm telt 25 regels, dus we kunnen iets doen als:

```
FOR I = 1 to 25
PRINT
NEXT I
```

Dit kan evenwel op een betere manier aangepakt worden. Als u SHIFT ingedrukt houdt en vervolgens CLR HOME indrukt, wordt het scherm gewist. Dat is veel eenvoudiger en handiger. Hoe kunnen we zoiets echter in ons programma voor elkaar krijgen? Dat blijkt heel eenvoudig: u kunt het typen terwijl u een string invoert! Het toetsenbord werkt namelijk anders, afhankelijk van of u zich in de *zinnebeeldmodus* bevindt.

De zinnebeeldmodus is slechts een fraaie benaming voor wat er gebeurt wanneer u een aanhalingsteken in de computer typt: niets, totdat u nog een aanhalingsteken typt of op RETURN drukt. U zult zich herinneren dat we het eerder hadden over het probleem van vreemde symbolen die verschijnen wanneer u de cursor-toetsen gebruikt. Toen wilden we u niet afleiden, maar dat is een verschijnsel van de zinnebeeldmodus: in de zinnebeeldmodus tonen veel speciale toetsen een *zinnebeeld* dat hen vertegenwoordigt, in plaats van hun normale werking te vervullen. Drukt u in de zinnebeeldmodus bijvoorbeeld de cursor-toets naar-links in, dan verschijnt het zinnebeeld ▌. Drukt u de cursor-toets naar-rechts in, dan verschijnt een ▐; bij omhoog een ▀, bij omlaag een ▄ en bij CLR HOME een ▚. Houdt u SHIFT ingedrukt en drukt u CLR HOME in, dan verschijnt een ▞.

Alternatieve termen:

- De Nederlandstalige handleiding van de Commodore 64 uit 1982 spreekt over een codeteken in plaats van een zinnebeeld. De handleiding vermijdt het benoemen van de modus.
- In Engelstalige Mega65-documentatie wordt de zinnebeeldmodus de *quote mode* genoemd, naar het aanhalingsteken dat hem start.

Laten we dit dus gebruiken om de tweede regel het scherm te laten wissen wanneer hij de melding **RAAD HET GETAL TUSSEN 1 EN 100** weergeeft. De eerste keer dat u het probeert, is het wat verwarrend, maar zodra u de slag te pakken heeft, is het vrij eenvoudig. Wat we uiteindelijk willen, is een regel die er als volgt uitziet:

```
2 PRINT "■RAAD HET GETAL TUSSEN 1 EN 100"
```

Begin hiervoor met het intypen van **2 PRINT "**. Houd vervolgens de SHIFT-toets ingedrukt en druk op CLR HOME. Uw regel zou er nu als volgt uit moeten zien: **2 PRINT"■**. Is dat het geval, dan is het gelukt! U kunt de regel nu op de normale manier verder typen. Indien u dat gedaan heeft, kunt u zoals gebruikelijk het **LIST**-commando gebruiken om na te gaan of u het programma met succes heeft gewijzigd. U zou uw gewijzigde regel met het ■-zinnebeeld moeten zien.

```
LIST

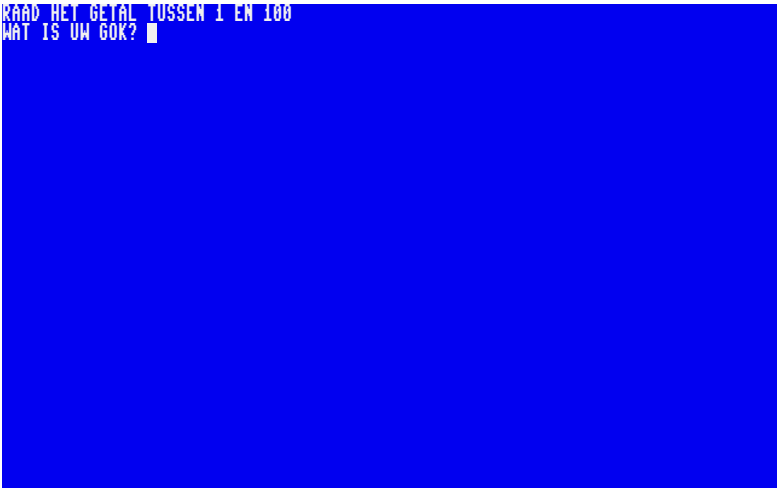
1 S$=23
2 PRINT "RAAD HET GETAL TUSSEN 1 EN 100"
3 INPUT "WAT IS UW GOK";G
4 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
5 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
6 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"

READY.
2 PRINT"■RAAD HET GETAL TUSSEN 1 EN 100"
LIST

1 S$=23
2 PRINT"■RAAD HET GETAL TUSSEN 1 EN 100"
3 INPUT "WAT IS UW GOK";G
4 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
5 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
6 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"

READY.
```

Voert u het programma nu uit door **RUN** in te typen en zoals gebruikelijk op RETURN te drukken, dan geeft de 2e regel de computer de opdracht eerst het scherm te wissen en daarna de rest van de melding weer te geven, als volgt:



Dit verbergt de programmalijst voor de gebruiker, zodat hij niet onmiddellijk kan zien wat ons geheime getal is. We kunnen onze gok intypen, net als voorheen, maar net als voorheen keert de computer na één gok terug naar de **READY.**-prompt. We zouden het prettig vinden als mensen meer dan één gok konden wagen, zonder dat zij hoeven te weten dat zij het programma opnieuw moeten uitvoeren.

Er zijn een paar manieren waarop we dit kunnen aanpakken. We hebben het **FOR** - **NEXT**-patroon al gezien. Met dat patroon kunnen we het programma de gebruiker een zeker aantal gokken laten geven. Als we het **NEXT**-commando door een andere programmaregel zouden laten volgen, konden we de gebruiker zelfs vertellen wanneer hij te veel gokken heeft gewaagd. Laten we dus eens naar ons programma kijken en bedenken hoe we dat zouden kunnen doen. Hier is onze huidige programmalijst opnieuw:

```
1 SN=23
2 PRINT "RAAD EEN GETAL TUSSEN 1 EN 100"
3 INPUT "WAT IS UW GOK"; G
4 IF G<SN THEN PRINT "MIJN GETAL IS GROTER"
5 IF G>SN THEN PRINT "MIJN GETAL IS KLEINER"
6 IF G=SN THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"
```

Als we willen dat de gebruiker meerdere gokken kan wagen, moeten regels 2 tot en met 6 meerdere keren worden uitgevoerd. Dat maakt het er voor ons wat lastiger op, want we moeten een regel tussen regel 1 en 2 invoegen. Tenzij u echter een wiskundig genie bent, zijn er geen gehele getallen tussen 1 en 2 en de MEGA65 begrijpt regelnummers zoals 1.5 niet.

Gelukkig beschikt de MEGA65 over het **RENUMBER** -commando. Dit commando kan alleen in de directe modus worden ingetypt. Bij uitvoering verandert het de regelnummers in het programma, zonder hun volgorde te wijzigen. De nieuwe nummers zijn gewoonlijk veelvouden van 10, zodat u tussenin veel vrije nummers overhoudt om extra regels toe te voegen. Als we het bijvoorbeeld op ons programma toepassen, dan hernummert het de regels tot 10, 20, ..., 60. Dat dit is gebeurd, kunnen we controleren met het **LIST**-commando:

```
READY.
LIST

1 S$=23
2 PRINT "RAAD HET GETAL TUSSEN 1 EN 100"
3 INPUT"WAT IS UW GOK";G
4 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
5 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
6 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"

READY.
RENUMBER

READY.
LIST

10 S$=23
20 PRINT "RAAD HET GETAL TUSSEN 1 EN 100"
30 INPUT"WAT IS UW GOK";G
40 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
50 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
60 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"

READY.
```

Nu is ons leven een stuk eenvoudiger: we kunnen voor ons **FOR**-commando elk getal tussen 10 en 20 kiezen. Het is een gebruikelijke keuze om het middelste getal te gebruiken, zodat u, als u later nog andere zaken wilt toevoegen, daarvoor de ruimte heeft. Laten we dus een **FOR**-commando toevoegen dat de gebruiker 10 kansen geeft om het getal te raden. We kunnen hiervoor elke variabelenaam gebruiken die we willen, behalve **G** en **SN**, want die gebruiken we al. Het zou erg verwarrend zijn als we die door elkaar haalden! Laten we dus een regel als deze toevoegen:

```
15 FOR I = 1 TO 10 STEP 1
```

Nu hebben we na regel 60 een bijpassend **NEXT I** nodig. Laten we het handige patroon van het optellen van 10 aanhouden om het volgende regelnummer te bepalen en het als regel 70 toevoegen:

```
70 NEXT I
```

Die regels kunnen we intypen, waarna we met het **LIST**-commando controleren of het resultaat klopt:

```
LIST

10 S$=23
20 PRINT "RAAD HET GETAL TUSSEN 1 EN 100"
30 INPUT "WAT IS UW GOK";G
40 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
50 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
60 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"

READY.
15 FOR I = 1 TO 10 STEP 1
70 NEXT I
LIST

10 S$=23
15 FOR I = 1 TO 10 STEP 1
20 PRINT "RAAD HET GETAL TUSSEN 1 EN 100"
30 INPUT"WAT IS UW GOK";G
40 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
50 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
60 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"
70 NEXT I

READY.
```

Dat ziet er al heel aardig uit. Er zijn echter nog een paar kleine probleempjes. Kunt u bedenken welke? Wat gebeurt er nu nadat de gebruiker een gok waagt? En wat gebeurt er als zijn gokken op zijn?

Als u heeft bedacht dat het wagen van een gok het scherm onmiddellijk wist, dan mag u uzelf gerust op de schouder kloppen! De gebruiker heeft nauwelijks tijd om de melding te lezen. Erger nog: als hij het getal juist raadt, komt hij daar niet eens achter en het programma gaat gewoon door. We zouden graag zien dat het programma stopt of eindigt zodra de gebruiker een juiste gok maakt.

Dit kunnen we doen met het **STOP**- of het **END**-commando. Deze twee commando's lijken vrij veel op elkaar. Het belangrijkste verschil is dat de computer u vertelt waar het programma gestopt is wanneer u het met **STOP** stopt en dat u de kans krijgt om het programma voort te zetten met het **CONT**-commando. Het **END**-commando vertelt de computer daarentegen dat het programma het einde bereikt heeft en dat de computer terug moet keren naar **READY**. Voor ons programma is het **END**-commando zinvoller, want nadat de gebruiker het getal geraden heeft, is er geen enkele reden meer om verder te gaan.

Nu hebben we een manier nodig om de computer twee verschillende dingen te laten doen wanneer de gebruiker een juiste gok maakt. We zouden gewoon een extra **IF**-commando na regel 60 kunnen toevoegen dat de felicitatiemelding weergeeft, bijv. **65 IF G=SN THEN END**.

We kunnen echter nog iets eleganter te werk gaan. Er is namelijk een manier om meerdere commando's op één enkele regel te zetten. Als u zich herinnert hoe we het **INPUT**-commando hebben leren gebruiken, weet u misschien nog dat er twee verschillende tekens zijn waarmee stukjes gegevens van elkaar worden gescheiden: **,**

en :. Het tweede teken, :, heet een dubbele punt en kan ook gebruikt worden om BASIC-commando's op één regel van elkaar te scheiden. Willen we regel 60 dus zo wijzigen dat het **PRINT**-commando de felicitatiemelding weergeeft en daarna het programma beëindigt met **END**, dan kunnen we gewoon : **END** aan het einde van de regel toevoegen. De regel zou er dan als volgt uit moeten zien:

```
60 IF G=SN THEN PRINT "GEFELICITEERD! UW HEEFT MIJN GETAL GERADEN!": END
```

Dat probleem is daarmee opgelost. Het zou echter ook fijn zijn om het scherm niet telkens na een gok te wissen, zodat de gebruiker kan zien wat zijn laatste gok was en of die groter of kleiner was dan het getal. Daartoe kunnen we het wissen van het scherm uit regel 20 halen en een nieuw PRINT-commando aan een lager regelnummer toevoegen, zodat het scherm één keer aan het begin van het programma gewist wordt, voordat de gebruiker met gokken kan beginnen.

We zouden het bijvoorbeeld in regel 5 kunnen zetten, zodat het de allereerste handeling van het programma is. Zoals we eerder opmerkten, doen de regelnummers zelf er niet toe. Het enige dat telt, is te onthouden dat de computer bij het laagste regelnummer begint en de regels op volgorde uitvoert. Laten we die wijzigingen aan ons programma dan maar doorvoeren:

```
20 PRINT "RAAD EEN GETAL TUSSEN 1 EN 100"
5 PRINT "J"
```

Als u die regels intypt en het programma opnieuw met **LIST** weergeeft, zou u ongeveer het volgende moeten zien:

```
40 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
50 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
60 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"
70 NEXT I

READY.

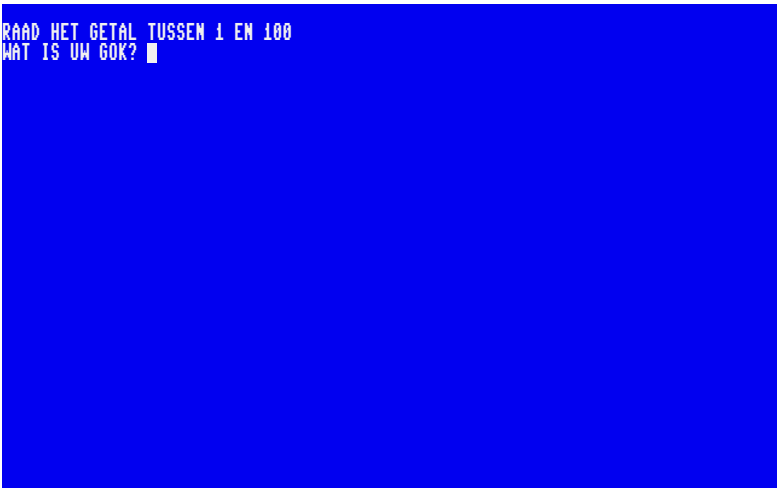


20 PRINT "□RAAD HET GETAL TUSSEN 1 EN 100"
5 PRINT "□"
LIST

5 PRINT "□"
10 S$=23
15 FOR I = 1 TO 10 STEP 1
20 PRINT "□RAAD HET GETAL TUSSEN 1 EN 100"
30 INPUT"WAT IS UW GOK";G
40 IF G<S$ THEN PRINT "MIJN GETAL IS GROTER"
50 IF G>S$ THEN PRINT "MIJN GETAL IS KLEINER"
60 IF G=S$ THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!"
70 NEXT I

READY.
```

We kunnen het programma nu met **RUN** uitvoeren en kijken of het werkt. Laten we het proberen!



Het scherm wordt nog steeds gewist en dat is goed. Ziet u echter al één klein verschil? Boven de eerste melding staat een lege regel. Dat komt doordat ons **PRINT**-commando in regel 5 na het weergeven van het teken dat het scherm wist naar de volgende rij op het scherm gaat. Dat kunnen we verhelpen door een **;** (puntkomma) aan het einde van het **PRINT**-commando te plaatsen. Dat vertelt het **PRINT**-commando dat het na alles te hebben weergegeven niet naar het begin van de volgende rij op het scherm moet gaan. Veranderen we regel 5 dus in **5 PRINT "□";**, dan verdwijnt de lege ruimte boven aan het scherm.

Laten we terugkeren naar ons programma: we kunnen nu gokken en het programma vertelt ons of elke gok meer of minder is dan het juiste getal. Na tien gokken vraagt het programma niet meer om een nieuwe gok en keert het terug naar de **READY.**-prompt, als volgt:

```
MIJN GETAL IS KLEINER
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 60
MIJN GETAL IS KLEINER
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 50
MIJN GETAL IS KLEINER
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 40
MIJN GETAL IS KLEINER
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 30
MIJN GETAL IS KLEINER
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 20
MIJN GETAL IS KLEINER
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 28
MIJN GETAL IS KLEINER
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 27
MIJN GETAL IS KLEINER

READY.
```

Het zou fijn zijn de gebruiker te vertellen wanneer zijn gokken op zijn. We moeten deze melding na het **NEXT**-commando toevoegen. Het is ook netjes de gebruiker te vertellen wat het geheime getal was, in plaats van hem in onzekerheid te laten. Laten we de regel dus aan het einde van ons programma toevoegen, als regel 80:

```
80 PRINT "HELAAS! UW KANSEN ZIJN OP. MIJN GETAL WAS"; SN
```

Raadt de gebruiker het getal nu niet, dan krijgt hij een nuttige melding, als volgt:

```
RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 97

RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 96

RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 95

RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 94

RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 93

RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 92

RAAD HET GETAL TUSSEN 1 EN 100
WAT IS UW GOK? 91
MIJN GETAL IS KLEINER
HELAAS! UW KANSEN ZIJN OP. MIJN GETAL WAS 23

READY.
```

Oefeningen

1. Kunt u het programma aan het begin naar het geheime getal laten vragen?

Op dit moment zet het programma het geheime getal elke keer op 23. Om het spel interessanter te maken zou het mooi zijn de eerste gebruiker naar het geheime getal te vragen en daarna de rest van het spel te starten, zodat iemand anders kan proberen het getal te raden.

Aanwijzing: U zult de regel die de **SN**-variabele instelt zo moeten wijzigen dat de waarde bij de eerste gebruiker kan worden opgevraagd. Het **INPUT**-commando kan daarbij van pas komen.

2. Kunt u het programma naar de naam van de gebruiker laten vragen en het vervolgens gepersonaliseerde antwoorden laten geven?

Op dit moment toont het programma zeer eenvoudige meldingen. Het zou mooi zijn om de gebruiker naar zijn of haar naam te vragen en die naam vervolgens te gebruiken om gepersonaliseerde meldingen te maken, zoals **HELAAS DIRK, MAAR DAT GETAL IS TE KLEIN**.

Aanwijzing: U zult vroeg in het programma een regel moeten toevoegen om de gebruiker naar zijn of haar naam te vragen.

Aanwijzing: Mogelijk wilt u nog eens bekijken hoe we het **PRINT***-commando hebben gebruikt, onder meer met* **;** *om meer dan één waarde op een regel weer te geven.*

3. Kunt u het uiterlijk van de meldingen verbeteren met kleuren en een betere spatiëring?

Het programma is eigenlijk niet bepaald fraai geworden. Het zou mooi zijn om kleuren te gebruiken.

Aanwijzing: Mogelijk wilt u meer **PRINT***-commando's toevoegen om de spatiëring en de opmaak van de meldingen te verbeteren.*

Aanwijzing: Mogelijk wilt u de kleurcodes gebruiken in de meldingen die u met **PRINT** *weergeeft.*

Aanwijzing: Mogelijk wilt u ook de commando's **FOREGROUND**, **BACKGROUND** *en* **BORDER** *gebruiken om de kleur van de tekst, de schermachtergrond en de rand in te stellen.*

4. Kunt u het programma laten aangeven of een gok "warmer" of "kouder" is dan de vorige gok?

Op dit moment vertelt het programma u alleen of de gok hoger of lager is dan het geheime getal. Het zou mooi zijn als het programma u bij elke gok kon vertellen of de gok dichterbij komt of juist verder weg raakt: wanneer de gok dichterbij komt, moet het de gebruiker laten weten dat hij "warmer" wordt en "kouder" wanneer hij verder weg raakt.

Dit is heel wat ingewikkelder dan de voorgaande oefeningen en u zult een aantal nieuwe aangelenheden moeten uitdenken.

Aanwijzing: U zult de vorige gok in een aparte variabele moeten onthouden en die vervolgens met de laatste vergelijken: is hij dichterbij of verder weg. Mogelijk heeft u **IF***-commando's nodig waarin op het eerste* **IF** *nog een ander volgt, of moet u leren de operator* **AND** *te gebruiken.*

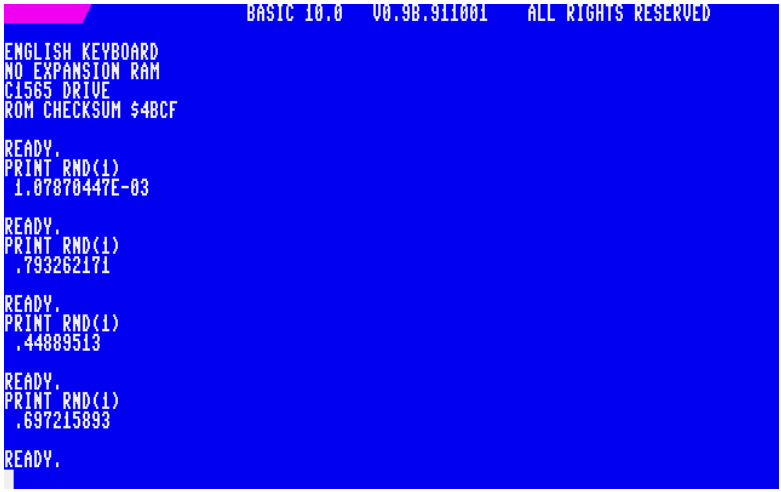
WILLEKEURIGE GETALLEN EN TOEVAL

Op het spelletje "raad het getal" komen we zo terug, maar laten we eerst een uitstapje maken. Een uitstapje door een doolhof. Laten we hopen dat we er vóór het einde van de les weer uitkomen! Laten we een eenvoudige manier bekijken om een doolhof te maken. Dit programma is reeds lange tijd bekend. Het werkt door willekeurig te kiezen of het een ⧹- of een ⧸-teken weergeeft. Deze tekens krijgt u door SHIFT ingedrukt te houden en op de toets N of M te tikken. U kunt de tekens op de voorkant van die toetsen zien. Hoewel ze op de toetsen binnen een kader staan afgebeeld, verschijnt dat kader niet; alleen de diagonale lijn is zichtbaar. Het blijkt dat het herhaald lukraak afdrukken van één van deze twee tekens een aardig doolhof oplevert.

Laten we het eens proberen. Om dit te kunnen doen, hebben we een manier nodig om willekeur te genereren. De MEGA65 heeft daarvoor de functie **RND(1)**. Deze functie werkt als een variabele, maar telkens wanneer u haar gebruikt, levert zij een ander resultaat op. Laten we bekijken hoe dat in zijn werk gaat. Typ het volgende in:

```
PRINT RND(1)
```

Elke keer dat u dit intypt, levert het een ander antwoord op, zoals u hier kunt zien:



We kunnen zien dat dit ons verschillende resultaten oplevert: **1.07870447E-03**,**.793262171**, **.44889513**, **.697215893**. Elk van deze getallen ligt tussen 0 en 1, ook het eerste. Het eerste getal is in *wetenschappelijke notatie* geschreven. De **E-03** betekent dat de waarde $1.07870447 \times 10^{-3}$ = 0.000107870447 is. Dat wil zeggen dat de **E-03** aangeeft dat de decimale punt drie plaatsen naar links verschuift. Staat er een **+** achter de **E**, dan betekent dat dat de decimale punt naar rechts verschuift. **1.23456E+3** staat bijvoorbeeld voor het getal 1234.56.

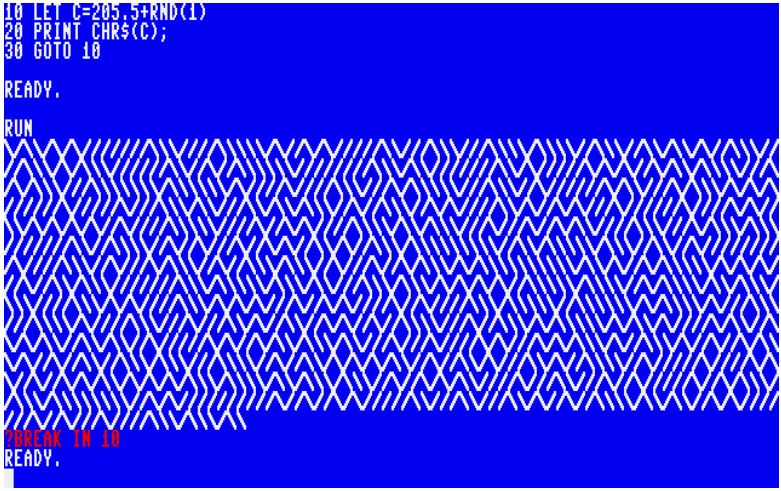
Goed, we hebben u een doolhof beloofd, dus geven we u er maar één. Met deze **RND(1)** kunnen we tussen deze twee tekens kiezen. Het eerste heeft een tekencode van 205 en het tweede heel handig 206. Dat betekent dat we, wanneer we het resultaat van **RND(1)** bij 205.5 optellen, een getal tussen 205.5 en 206.5 krijgen. De helft van de tijd is dat $205.iets$ en de andere helft van de tijd $206.iets$. Zo kunnen we het ene of het andere teken weergeven met de functie **CHR$()**, welke het teken teruggeeft dat hoort bij het getal dat we tussen de haakjes zetten. Met die functie kunnen we iets doen als:

```
LET C = 205.5+RND(1)
PRINT CHR$(C);
```

Dit geeft elke keer het ene of het andere teken weer. Hiermee konden we het doolhof al afdrukken door dit keer op keer te herhalen en er een lus van te maken. Daarvoor zouden we **FOR** en **NEXT** kunnen gebruiken. In dit geval willen we echter dat het programma voor altijd blijft doorgaan, dat wil zeggen: elke keer dat het programma aan het einde komt, moet het weer bij het begin beginnen. De makers van BASIC waren niet erg creatief, want het commando dat dit doet heet **GOTO**. U zet daarachter het nummer van de regel die vervolgens uitgevoerd moet worden, bijv. **GOTO 1**. Hiermee kunnen we ons kleine doolhofprogramma zo schrijven dat het onafgebroken blijft draaien:

```
10 LET C = 205.5+RND(1)
20 PRINT CHR$(C);
30 GOTO 10
```

Voert u dit programma uit met **RUN**, dan begint de computer eindeloos een doolhof te tekenen, dat eruitziet zoals op de onderstaande schermafbeelding. U kunt het programma op elk moment stoppen door RUN STOP in te drukken en u kunt het onderbreken door NO SCROLL in te drukken en hervatten door opnieuw NO SCROLL in te drukken. Drukt u RUN STOP in, dan vertelt de computer u waar hij op dat moment gebleven was. In het geval van de onderstaande schermafbeelding was hij bezig met regel 10:



Dat werkt prima en tekent een zeer beroemd doolhof [1]. Toch kunnen we het programma kleiner maken. Het resultaat van de berekening van welk symbool er weergegeven moet worden, hoeft niet op een aparte regel te staan. We kunnen de berekening rechtstreeks tussen de haakjes van de functie **CHR$()** zetten:

```
10 PRINT CHR$(205.5+RND(1));
20 GOTO 10
```

En we kunnen gebruikmaken van wat we over het :-symbool (de dubbele punt) hebben geleerd en het **GOTO**-commando op dezelfde regel zetten als het **PRINT**-commando:

```
10 PRINT CHR$(205.5+RND(1));: GOTO 10
```

Ziet u hoe er vaak vele manieren zijn om met een programma hetzelfde effect te bereiken? Dat is heel normaal. Bij complexe programma's zijn er heel veel manieren om hetzelfde effect te bereiken. Dit is één van de terreinen in het computerprogrammeren waarop u zeer creatief kunt zijn.

Laten we terugkeren naar het onderwerp willekeur. Het is goed dat we deze willekeurige getallen tussen 0 en 1 kunnen gebruiken om een doolhof te tekenen, maar het is wat lastig om mensen te vragen een heel lang decimaal getal in te typen. Willen we een getal in het bereik 1 tot 100, dan kunnen we wat we van **RND(1)** krijgen met 100 vermenigvuldigen. Doen we dat, dan wordt het al wat beter, maar we krijgen nog steeds getallen als **55.0304651**, **30.3140154**, **60.2505497** en **.759229916**.

Dat komt al in de buurt, maar we willen die gebroken delen echt kwijt. We willen namelijk hele getallen, ofwel *integers*. BASIC heeft daarvoor de functie **INT()**, welke net zo werkt als de functie **RND(1)**: welk getal u ook tussen de haakjes zet, de functie geeft alleen het gehele deel daarvan terug. **INT(2.18787)** geeft bijvoorbeeld de waarde **2** terug. Zoals we zojuist zeiden, hakt de functie het gebroken deel eraf, dat wil zeggen dat hij altijd naar beneden afrondt. Zelfs **INT(2.9999999)** levert dus nog steeds **2** op in plaats van **3**. Dat betekent dat we, als we het resultaat van **RND(1)** met 100 vermenigvuldigen, een getal in het bereik 0 - 99 krijgen in plaats van 1 - 100. Dat is gelukkig eenvoudig op te lossen: we tellen gewoon 1 bij het resultaat op. Om dus een integer te genereren, dat wil zeggen een geheel getal tussen 1 en 100, beide inbegrepen, kunnen we iets doen als:

```
PRINT INT(RND(1)*100) + 1
```

Dat ziet er veel beter uit. Laten we ons "raad het getal"-programma nogmaals intypen. Deze keer vervangen we echter de plek waar we ons geheime getal op 23 zetten door een willekeurig geheel getal tussen 1 en 100. Kijk nog niet naar de oplossing. Denk eerst na over hoe u met het bovenstaande **SN** op een willekeurig geheel getal tussen 1 en 100 kunt zetten. Heeft u uw eigen gok klaar, bekijk dan wat wij hieronder hebben bedacht. Wellicht heeft u een ander programma gemaakt dat hetzelfde kan doen. Dat is ook prima!

```
10 SN=INT(RND(1)*100)+1
20 PRINT "␣"
30 FOR I = 1 TO 10 STEP 1
40 PRINT "RAAD HET GETAL TUSSEN 1 EN 100"
50 INPUT "WAT IS UW GOK"; G
60 IF G<SN THEN PRINT "MIJN GETAL IS GROTER"
70 IF G>SN THEN PRINT "MIJN GETAL IS KLEINER"
80 IF G=SN THEN PRINT "GEFELICITEERD! U HEEFT MIJN GETAL GERADEN!": END
90 NEXT I
100 PRINT "HELAAS, UW KANSEN ZIJN OP"
```

Nu hoeven we niet langer te vrezen dat iemand het getal raadt en we hoeven ook niemand anders het getal voor ons te laten kiezen. Daardoor is het programma veel leuker om te spelen. Kunt u het verslaan?

Oefeningen om te proberen

1. Kunt u het doolhofprogramma verschillende doolhoven laten tekenen?

Het doolhofprogramma toont momenteel evenveel ☑ als ☒. Kunt u het programma zo veranderen dat het twee keer zoveel van het ene teken afdrukt als van het andere? Hoe ziet het doolhof er dan uit?

Aanwijzing: Wij hebben **205.5** *gebruikt, zodat we na het optellen van een willekeurig getal tussen 0 en 1 de helft van de tijd op* $205.iets$ *en de andere helft van de tijd op* $206.iets$ *uitkomen. Verlaagt u* **205.5** *richting* **205**, *of verhoogt u het richting* **206**, *dan verandert u de relatieve verhouding waarin elk teken verschijnt.*

2. Kunt u het "raad het getal"-programma aanpassen zodat het een getal tussen 1 en 10 kiest?

Op dit moment kiest het programma een getal tussen 1 en 100. Pas het programma aan zodat het een getal uit een ander bereik kiest. Vergeet niet de melding aan te passen die de computer aan de gebruiker toont. Heeft hij dan nog steeds 10 gokken nodig? Verander het maximum aantal gokken dat hij krijgt voordat hij verliest in een geschikter aantal.

Aanwijzing: U zult de regel moeten aanpassen die **SN** *instelt, evenals de* **PRINT***-melding die de gebruiker uitleg geeft.*

3. Zet de scherm-, rand- en tekstkleur op willekeurige kleuren

Pas het doolhofprogramma of het "raad het getal"-programma aan om willekeurige kleuren te gebruiken. Hoe kunt u ervoor zorgen dat de tekst altijd zichtbaar blijft?

Aanwijzing: Gebruik de commando's **FOREGROUND**, **BACKGROUND** *en* **BORDER** *om de kleuren in te stellen. Gebruik kleurnummers tussen 0 en 15, inclusief. U kunt aan het einde van deze commando's een berekening plaatsen in plaats van een eenvoudig getal.*

Aanwijzing: Om te voorkomen dat u de tekstkleur op dezelfde kleur als de achtergrond instelt, kunt u berekenen welke achtergrondkleur u wilt gebruiken en die in de ene variabele bewaren. Bereken vervolgens de tekstkleur die u wilt gebruiken en bewaar die in een andere variabele. Hebben de twee variabelen hetzelfde getal, dan moet u een van beide veranderen.

4. Laat het "raad het getal"-programma bij het starten willekeurig kiezen tussen twee verschillende welkomstmeldingen

Het "raad het getal"-programma toont momenteel telkens wanneer het start dezelfde melding. Pas het aan zodat het elke keer een van twee mogelijke meldingen toont.

Aanwijzing: Gebruik **RND(1)** *om een willekeurig getal te verkrijgen. Is dat getal kleiner dan een bepaalde drempelwaarde, toon dan de eerste melding; zo niet, toon dan de tweede melding.*

Aanwijzing: Het kan gemakkelijker zijn om het willekeurige getal in een variabele te bewaren, zodat u twee **IF***-instructies kunt gebruiken om te bepalen of u elke melding toont.*

Aanwijzing: Gebruikt u **<** *(kleiner dan) als relationele operator in een van de* **IF***-instructies, dan moet u in de andere de tegenovergestelde gebruiken. Het tegenovergestelde van kleiner dan is groter dan of gelijk aan.*

HOOFDSTUK 11

Tekstverwerking

- Tekens en tekenreeksen
- Tekenreeksconstanten
- Tekenreeksvariabelen
- Tekenreeksfuncties
- Eenvoudige opmaak
- Voorbeeldprogramma's

TEKENS EN TEKENREEKSEN

Veel computerprogramma's moeten tekstuele informatie kunnen vastleggen in de vorm van afdrukbare letters, cijfers en symbolen. De behoefte aan tekst ontstaat bij tekstverwerkingsprogramma's en woordspellen. Ook bij natuurlijke taalverwerking en bij tekstgebaseerde avonturenspellen is tekst nodig, omdat beide de invoer moeten begrijpen. Het begrijpen van tekstinvoer heet *parsen*. Kortom, tekstverwerking wordt overal toegepast. Om dergelijke informatie in te voeren, weer te geven en te bewerken, moeten wij twee kernbegrippen introduceren: tekens en tekenreeksen.

Tekens kunnen afdrukbaar of niet-afdrukbaar zijn. Een teken vertegenwoordigt meestal één enkel, primitief element van afdrukbare tekst, dat met de **PRINT**-opdracht op het scherm kan worden weergegeven. Het meest voor de hand liggend en natuurlijk is het om een teken op te vatten als een letter uit een alfabet. Een teken kan bijvoorbeeld een van de hoofdletters 'A' tot 'Z' zijn, of een van de kleine letters 'a' tot 'z'. Tekens kunnen echter ook veelgebruikte symbolen vertegenwoordigen, zoals leestekens of valutasymbolen. Zij kunnen zelfs de decimale cijfers '0' tot '9' vertegenwoordigen. Het is goed om te beseffen dat het hier om de tekstuele weergave van de cijfers 0 tot 9 als afdrukbare symbolen gaat en niet om hun numerieke tegenhangers. Daarnaast biedt de MEGA65 een uitgebreide reeks speciale symbolen die samen kunnen worden gebruikt voor spellen, voor het tekenen van sierlijke randen of kunst. Naast het weergeven van informatie kunnen zulke symbolen ook eenvoudige, doch boeiende visuele patronen vormen. Voor het gemak staan deze speciale symbolen op de voorzijde van de toetsen van de MEGA65.

Een teken kan ook niet-afdrukbaar zijn. Het gebruik van dergelijke tekens (in een **PRINT**-opdracht) kan bepaald gedrag activeren of bepaalde modi actief maken, zoals het omzetten van alle tekst op het scherm naar kleine letters of het instellen van de voorgrondkleur op oranje. Andere niet-afdrukbare tekens kunnen een regelterugloop vertegenwoordigen of het scherm wissen.

Voor een volledig overzicht van de beschikbare tekens verwijzen wij u naar Hoofdstuk/Bijlage D op bladzijde D-3. De tabel vermeldt de tekens die bij een bepaald codenummer horen. Het codenummer moet als argument aan de **CHR$**-functie worden meegegeven; in combinatie met de **PRINT**-opdracht geeft dat de bijbehorende tekens op het scherm weer.

Hier ziet u een voorbeeld van het weergeven van het uitroepteken met behulp van een tekencode:

```
PRINT CHR$(33)
!
```

Merk op dat het '!'-teken inderdaad op het scherm zichtbaar is, omdat het een afdrukbaar teken is.

Hier ziet u een voorbeeld van het veranderen van de voorgrondkleur naar wit met behulp van tekencodes:

```
PRINT CHR$(5)
```

Hoewel er geen teken wordt weergegeven, krijgen alle afdrukbare tekens die daarna op het scherm verschijnen de kleur wit.

Soms is het nuttig om de omzetting omgekeerd uit te voeren: van een teken naar het bijbehorende codenummer. Hiertoe moet één enkel teken, tussen aanhalingstekens, als argument aan de **ASC**-functie worden meegegeven; in combinatie met de **PRINT**-opdracht geeft dat het bijbehorende codenummer decimaal op het scherm weer.

Hier ziet u een voorbeeld van het bepalen van het codenummer van het uitroepteken.

```
PRINT ASC("!")
 33
```

En hier ziet u een voorbeeld van het bepalen van het codenummer van het uitroepteken en het opslaan ervan in een integer-variabele:

```
A% = ASC("!")
```

Hoewel we afzonderlijke tekens herhaaldelijk kunnen weergeven met **CHR$**, zou het omslachtig zijn om dit steeds op deze manier te doen.

Het begrip tekenreeks is nodig, omdat het het idee van een aaneengesloten blok tekst vertegenwoordigt. Een tekenreeks kan dus meerdere afdrukbare en/of meerdere niet-afdrukbare tekens bevatten, in elke gewenste combinatie. Een tekenreeks kan ook leeg zijn en helemaal geen tekens bevatten. Om een tekenreeks te schrijven, zetten we het teken*s* tussen aanhalingstekens. De reeks "HALLO WERELD!" is bijvoorbeeld een tekenreeksconstante.

```
PRINT "HALLO WERELD!"
HALLO WERELD!
```

Alle tekenreeksen hebben een eigenschap met de naam lengte, welke aangeeft hoeveel afdrukbare en niet-afdrukbare tekens er in die tekenreeks aanwezig zijn. De lengte kan variëren van 0 (de lege tekenreeks) tot 255. Een poging om een tekenreeks te maken die langer is dan 255 tekens, leidt tot een **?STRING TOO LONG ERROR**.

```
PRINT LEN("HALLO WERLED!")
 12
```

```
PRINT LEN("")
 0
```

Het is mogelijk om variabelen te maken die specifiek voor tekenreeksen bedoeld zijn. Alle dergelijke tekenreeksvariabelen hebben namen die met een letter beginnen, eventueel een tweede, alfanumeriek teken bevatten en op een $ eindigen. Nadat zij een waarde hebben gekregen, kunnen zij met **PRINT** worden gebruikt.

```
AB$ = "HALLO WERELD!": PRINT AB$
HALLO WORLD!
```

```
A1$ = "HALLO WERELD!": PRINT LEN(A1$)
 12
```

TEKENREEKSCONSTANTEN

Tekenreeksconstanten kunnen met één of meer andere tekenreeksconstanten worden samengevoegd tot een samengestelde tekenreeks. Dit proces heet *samenvoegen*. Om twee of meer tekenreeksconstanten samen te voegen, koppelt u ze met de +-operator aan elkaar.

Hier volgen enkele voorbeelden:

```
PRINT ("TWEEDE" + "HANDS")
TWEEDEHANDS
```

```
PRINT ("AUTOBAND" + "VENTIEL" + "DOPJE")
AUTOBANDVENTIELDOPJE
```

Soms zijn leestekens of spaties nodig om de uiteindelijke uitvoer er correct opgemaakt te laten uitzien, zoals in het volgende voorbeeld.

```
PRINT ("FRUIT: " + "APPEL, " + "PEER EN " + "FRAMBOOS.")
FRUIT: APPEL, PEER EN FRAMBOOS.
```

TEKENREEKSVARIABELEN

Samenvoegen komt vaker voor bij tekenreeksvariabelen in combinatie met tekenreeksconstanten. In een tekstgebaseerd avonturenspel wilt u bijvoorbeeld enkele uitgangen vermelden, zoals noord of zuid. Omdat deze uitgangen afhankelijk zijn van de plek waar u zich op dat moment bevindt, is het zinvol om variabelen voor de uitgangen zelf te gebruiken en om de uitvoer met constanten, zoals komma's, spaties en punten, passend op te maken.

```
A$ = "PINDA": B$ = "KAAS": PRINT (A$ + B$ + "POT")
PINDAKAASPOT
```

Het is ook mogelijk om tekenreeksen als parameters van **DATA**-opdrachten te gebruiken, zodat zij later met het **READ**-opdrachten kunnen worden ingelezen. Het volgende voorbeeld laat tevens zien dat arrays ook tekenreeksen kunnen bevatten.

```
10 DIM A$(6)
20 PRINT "KLEUREN VAN DE REGENBOOG: ";
30 FOR I = 0 TO 5
40 :  READ A$(I): PRINT (A$(I) + ", ");
50 NEXT I
60 READ A$(I): PRINT ("AND " + A$(I) + ",")
70 DATA "ROOD", "ORANJE", "GEEL", "GROEN", "BLAUW", "INDIGO", "VIOLET"
```

Het komt vaak voor dat tekenreeksgegevens of gegevens van één enkel teken rechtstreeks uit gebruikersinvoer afkomstig zijn. Wanneer de gebruiker tekst intoetst, zal die tekst vaak geparst moeten worden of terug naar het scherm moeten worden weergegeven. In het algemeen zijn er drie hoofdmanieren om dat te doen: via de **GET**-opdracht, via de **GETKEY**-opdracht of via de **INPUT**-opdracht.

Alle drie de opdrachten gedragen zich verschillend. Het is daarom belangrijk om ze met elkaar te vergelijken en tegen elkaar af te zetten, zodat u begrijpt hoe elk ervan werkt.

De **GET**-opdrachten is handig om de huidige toetsaanslag in een variabele op te slaan. Het programma wacht niet op een toetsaanslag: het gaat onmiddellijk door met het uitvoeren van de volgende opdracht. Om die reden is het soms belangrijk om de **GET**-opdracht binnen een of andere lus te plaatsen—de lus wordt pas verlaten wanneer een geldige toetsaanslag is gedetecteerd. Als de variabele die aan **GET** wordt meegegeven een tekenreeksvariabele is en er geen toetsaanslag wordt gedetecteerd, dan krijgt die tekenreeksvariabele de waarde van een lege tekenreeks.

```
10 GET A$: REM WACHT NIET OP EEN TOETSDRUK--LEES EEN TOETSDRUK IN DE VARIABELE
20 PRINT A$: IF (A$ = "J" OR A$ = "N") THEN END
30 GOTO 10
```

De **GETKEY**-opdracht is ook nuttig om de huidige toetsaanslag in een variabele op te slaan. In tegenstelling tot de **GET**-opdracht wacht de **GETKEY**-opdracht bij de uitvoering wel degelijk op één toetsaanslag voordat het doorgaat met het uitvoeren van de volgende opdracht.

```
10 GETKEY A$: WACHT OP EEN TOETSDRUK--PAUZEER EN LEES 'M IN DE VARIABELE
20 PRINT A$: IF (A$ = "J" OR A$ = "N") THEN END
30 GOTO 10
```

Waar **GET** en **GETKEY** goed geschikt zijn voor het lezen van losse tekens, is de **INPUT**-opdracht nuttig voor het inlezen van complete tekenreeksen—dat wil zeggen, nul of meer tekens per keer.

Wanneer de **INPUT**-opdracht met een komma en een variabele wordt gebruikt, verschijnt de prompttekst op de gewone manier, met een cursor waarmee de gebruiker tekst kan intypen. Wanneer de **INPUT**-opdracht met een puntkomma en een variabele wordt gebruikt, verschijnt de prompttekst met een toegevoegd vraagteken en met een cursor waarmee de gebruiker tekst kan intypen.

```
10 INPUT "VOER UW NAAM IN", A$: REM GEEN VRAAG
20 PRINT ("HALLO " + A$)
```

```
10 INPUT "HOE HEET U"; A$: REM VRAAG
20 PRINT ("HALLO " + A$)
```

In beide gevallen rondt het indrukken van RETURN de tekstinvoer af—de ingevoerde tekst wordt in de opgegeven variabele opgeslagen. Merk op dat wanneer de tekenreeksvariabele al een tekenreeks bevat en RETURN wordt ingedrukt zonder dat er nieuwe gegevens worden ingetypt, de oude tekenreekswaarde die zich momenteel in de variabele bevindt, behouden blijft.

TEKENREEKSFUNCTIES

Er zijn drie veelgebruikte functies voor het bewerken van tekenreeksen: **MID$**, **LEFT$** en **RIGHT$**. Deze zijn bij uitstek geschikt voor het isoleren van deelreeksen, waaronder afzonderlijke tekens.

Het volgende programma vraagt om een invoertekenreeks en geeft daarna alle linkerdeelreeksen weer.

```
10 INPUT "VOER EEN WOORD IN:", A$
20 PRINT "ALLE LINKER DEELTEKENREEKSEN ZIJN:"
30 FOR I = 0 TO LEN(A$)
40 :  PRINT LEFT$(A$, I)
50 NEXT I
```

Het volgende programma vraagt om een invoertekenreeks en geeft daarna alle rechterdeelreeksen weer.

```
10 INPUT "VOER EEN WOORD IN:", A$
20 PRINT "ALLE RECHTER DEELTEKENREEKSEN ZIJN:"
30 FOR I = 0 TO LEN(A$)
40 :  PRINT RIGHT$(A$, I)
50 NEXT I
```

Het volgende programma vraagt om een invoertekenreeks die bestaat uit een voornaam, gevolgd door een spatie en daarna een achternaam. Het geeft vervolgens de beginletters van beide namen weer.

```
10 INPUT "VOOR EEN VOORNAAM, EEN SPATIE EN EEN ACHTERNAAM IN:", A$
20 N = -1
30 FOR I = 1 TO LEN(A$)
40 :  IF (MID$(A$, I, 1) = " ") THEN N = I: GOTO 60
50 NEXT I
60 IF (N = -1) THEN GOTO 10
70 PRINT "INITIALEN ZIJN: "; MID$(A$, 1, 1)+"."+MID$(A$, N + 1, 1)+"."
```

EENVOUDIGE OPMAAK

Nieuwe regels onderdrukken

Wanneer de **PRINT**-opdracht in een programma wordt gebruikt, is het standaardgedrag dat de tekenreeks wordt weergegeven en dat de cursor daarna naar de volgende regel gaat. Om te voorkomen dat de cursor automatisch naar de volgende regel gaat, plaatst u eenvoudigweg een ; (puntkomma) achter het einde van de tekenreeks. Vergelijk de regels 10, 20 en 30 in het volgende programma.

```
10 PRINT "DIT IS EEN ENKELE REGEL TEKST": REM ER WORDT EEN NIEUWE REGEL BEGONNEN
20 PRINT "THE SECOND LINE"; : REM EEN NIEUWE REGEL WORDT ONDERDRUKT
30 PRINT " USES A SEMICOLON" : REM EEN REGEL WORDT AAN DE VORIGE VASTGEMAAKT
```

Automatische tabstops

Soms is het handig om de **PRINT**-opdracht te gebruiken om informatie netjes in kolommen weer te geven. Dit kunt u doen door een , (komma) achter het einde van de tekenreeks te plaatsen. Bekijk het volgende voorbeeldprogramma.

```
10 PRINT "TEKST 1", "TEKST 2", "TEKST 3", "TEKST 4"
```

Merk op dat de tabstops telkens 10 tekens uit elkaar liggen. TEKST 1 begint dus in kolom 0, TEKST 2 begint in kolom 10, TEKST 3 begint in kolom 20 en TEKST 4 begint in kolom 30.

Tabstops en spatiëring

Bij het weergeven van tekst op het scherm is het vaak nodig om tekst op te maken met behulp van spaties en tabs. Twee commando's zijn hierbij behulpzaam: **SPC** en **TAB**.

Het commando **SPC(5)** schuift bijvoorbeeld vijf tekens naar rechts. Alle tekens die zich tussen de huidige cursorpositie en de positie vijf tekens verder naar rechts bevinden, blijven onveranderd staan.

Het commando **TAB(20)** gaat bijvoorbeeld naar kolom 20 door de huidige cursorpositie van twintig af te trekken en vervolgens dat aantal tekens naar rechts te schuiven.

Ligt de beginpositie van de cursor rechts van kolom 20, dan doet het commando niets. Dit commando is vaak bruikbaar om tekst netjes in kolommen uit te lijnen.

VOORBEELDPROGRAMMA'S

We sluiten af met enkele voorbeelden.

Palindromen

Een *palindroom* is een woord, zin of getal dat voorwaarts en achterwaarts gelezen hetzelfde is. Enkele voorbeelden zijn: CIVIC, LEVEL, RADAR, MADAM en 1234321. Het volgende programma draait de ingevoerde tekst om en bepaalt vervolgens of de oorspronkelijke zin gelijk is aan de omgekeerde zin.

```
10 REM *** PALINDROMEN ***
20 INPUT "VOER WAT TEKST IN: ", A$
30 B$ = ""
40 FOR I = 1 TO LEN(A$)
50 :  B$ = MID$(A$, I, 1) + B$
60 NEXT I
70 IF (A$ = B$) THEN PRINT (A$ + " IS EEN PALINDROOM"): ELSE PRINT (A$ + " IS GEEN PALINDROOM")
80 GOTO 20
```

Eenvoudige cijfers

Nu bekijken we drie eenvoudige voorbeelden van het versleutelen en ontsleutelen van Engelstalige tekstberichten. Dit proces van versleutelen en ontsleutelen is het studiegebied van de *cryptografie* en wordt toegepast om informatie beveiligd te houden, zodat anderen deze niet kunnen lezen, behalve wie het voorrecht heeft de methode en de geheime sleutel van het cijfer te kennen.

Het proces waarbij een gegeven bericht wordt versleuteld, heet *encryptie*. De gewone, leesbare, onversleutelde tekst heet *klare tekst*. Het versleutelen van klare tekst levert een door elkaar gehaald bericht op; deze versleutelde tekst heet *cijfertekst*. Het proces waarbij de cijfertekst wordt ontsleuteld, heet *decryptie*. Het ontsleutelen van de cijfertekst levert weer een leesbaar bericht op — de klare tekst.

Stel dat wij een stuk klare tekst versleutelen en de resulterende cijfertekst vervolgens naar een vriend sturen. Indien die vriend de methode en de geheime sleutel kent die voor het versleutelen van het bericht zijn gebruikt, kan deze de cijfertekst ontsleutelen en zo ons oorspronkelijke bericht in klare tekst herstellen en lezen.

Als iemand anders probeert de cijfertekst te lezen met de verkeerde methode en/of de verkeerde geheime sleutel, is de resulterende tekst onbegrijpelijk.

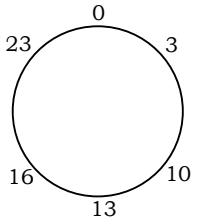
De cryptografische systemen die we hier beschrijven, zijn zeer eenvoudig. Uiteraard zou men ze tegenwoordig niet meer moeten gebruiken, omdat cryptanalysetechnieken ze gemakkelijk kunnen kraken. Ze illustreren echter wel enkele basistechnieken en laten zien hoe we een voorbeeldprogramma kunnen opbouwen.

We bekijken drie cijfers: het ROT13-cijfer, het Caesar-cijfer en het Atbasj-cijfer. Deze maken deel uit van een groep cijfers die bekend staat als *affiene cijfers*.

Wiskundig gezien is het nuttig om de letters van het Nederlandse alfabet als genummerd te beschouwen. A is 0, B is 1 enzovoort, waarbij Z gelijk is aan 25.

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Waarde	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Een belangrijk wiskundig onderdeel van een cryptografisch systeem is *modulair rekenen*, soms informeel "rekenen met een klok" genoemd, omdat de getallen bij nul beginnen en oplopen tot een bovengrens, waarna ze weer naar nul teruggaan, net als bij een cirkel. In ons geval, aangezien het Nederlandse alfabet 26 letters telt, rekenen we modulo 26—onze letters zijn genummerd van 0 tot 25.



Om een gegeven getal modulo 26 te herleiden, kunnen we de volgende functie gebruiken:

$$f(x) = x - \left\lfloor \frac{x}{26} \right\rfloor \times 26$$

Hieruit blijkt dat we, om de waarde van een getal x modulo 26 te verkrijgen, eerst x door 26 delen en naar beneden afronden, waarmee we het aantal keren bepalen dat we de cirkel rond zijn gegaan. Daarna vermenigvuldigen we dit resultaat weer met 26 en trekken we het van x af. Het eindresultaat is de rest die overblijft en zal altijd een waarde tussen 0 en 25 zijn.

Als voorbeeld is het getal 28 in modulo 26 gelijk aan 2:

$$f(28) = 28 - \left\lfloor \frac{28}{26} \right\rfloor \times 26 = 28 - 1 \times 26 = 2$$

Het programma aan het einde van dit hoofdstuk gebruikt deze formule door op regel 30 een overeenkomstige functie te definiëren:

```
DEF FN F(X)=X-INT(X/26)*26
```

ROT13: Om elke letter uit de klare tekst te versleutelen, schuiven we 13 plaatsen naar voren. De letter A uit de klare tekst wordt zo de letter N uit de cijfertekst, B wordt O, waarbij de laatste letters "terugspringen" naar het begin van het alfabet. De letter Z uit de klare tekst wordt daardoor de letter M uit de cijfertekst. Hiermee is de versleuteling beschreven. Het ontsleutelen van de cijfertekst werkt op precies dezelfde manier: we herhalen het proces eenvoudigweg door opnieuw 13 plaatsen naar voren te schuiven, waardoor de cirkel rond is en we weer uitkomen waar we begonnen. Zo wordt de letter N uit de cijfertekst weer de letter A uit de klare tekst.

We kunnen dit visueel zien als een afbeelding in tabelvorm:

Klare tekst	A	B	C	D	E	F	G	H	I	J	K	L	M
ROT13-cijfertekst	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Klare tekst	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
ROT13-cijfertekst	A	B	C	D	E	F	G	H	I	J	K	L	M

Om met ROT13 te versleutelen, zoekt u de letter van de klare tekst in de bovenste rij en gaat u omlaag naar de onderste rij om de bijbehorende letter van de cijfertekst te vinden. Om met ROT13 te ontsleutelen, zoekt u de letter van de cijfertekst in de onderste rij en gaat u omhoog naar de bovenste rij om de bijbehorende letter van de klare tekst te vinden.

Beschouwen we het ROT13-cijfer vanuit wiskundig oogpunt, dan zien we dat we voor zowel versleutelen als ontsleutelen eenvoudigweg 13 optellen bij de numerieke waarde van een letter uit de klare tekst of de cijfertekst en deze vervolgens modulo 26 herleiden. Dit levert een nieuw getal tussen 0 en 25 op dat overeenkomt met de versleutelde of ontsleutelde letter. De functie E_{ROT13} is de versleutelingsfunctie. Deze aanvaardt de waarde van een letter x uit de klare tekst als argument en geeft de waarde van de bijbehorende letter uit de cijfertekst als resultaat terug. De functie D_{ROT13} is de ontsleutelingsfunctie. Deze aanvaardt de waarde van een letter x uit de cijfertekst als argument en geeft de waarde van de bijbehorende letter uit de klare tekst als resultaat terug.

$$E_{ROT13}(x) = (x + 13) \bmod 26$$

$$D_{ROT13}(x) = (x + 13) \bmod 26$$

Merk op dat de definities van zowel de versleutelings- als de ontsleutelingsfunctie in dit geval precies hetzelfde zijn.

Atbasj: Atbasj is een eeuwenoude techniek om het Hebreeuwse alfabet van 22 letters te versleutelen, maar we kunnen dezelfde logica toepassen om het Nederlandse

alfabet van 26 letters te versleutelen. Om een letter met Atbasj te versleutelen, moeten we het Nederlandse alfabet achterstevoren opschrijven. Zo wordt de letter A uit de klare tekst bij het versleutelen de letter Z uit de cijfertekst, B wordt Y, C wordt X enzovoort. Het ontsleutelen van de cijfertekst werkt op dezelfde manier: de letter A uit de cijfertekst wordt de letter Z uit de klare tekst, B wordt Y, C wordt X enzovoort.

We kunnen dit visueel zien als een afbeelding in tabelvorm:

Klare tekst	A	B	C	D	E	F	G	H	I	J	K	L	M
Atbasj-cijfertekst	Z	Y	X	W	V	U	T	S	R	Q	P	O	N

Klare tekst	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Atbasj-cijfertekst	M	L	K	J	I	H	G	F	E	D	C	B	A

Om met Atbasj te versleutelen, zoekt u de letter van de klare tekst in de bovenste rij en gaat u omlaag naar de onderste rij om de bijbehorende letter van de cijfertekst te vinden. Om met Atbasj te ontsleutelen, zoekt u de letter van de cijfertekst in de onderste rij en gaat u omhoog naar de bovenste rij om de bijbehorende letter van de klare tekst te vinden.

Beschouwen we het Atbasj-cijfer vanuit wiskundig oogpunt, dan zien we dat we voor zowel versleutelen als ontsleutelen de numerieke waarde van de klare tekst of de cijfertekst met 25 moeten vermenigvuldigen, er daarna 25 bij moeten optellen en het geheel vervolgens modulo 26 moeten herleiden. Dit levert een nieuw getal tussen 0 en 25 op dat overeenkomt met de versleutelde of ontsleutelde letter. De functie E_{Atbasj} is de versleutelingsfunctie. Deze aanvaardt de waarde van een letter x uit de klare tekst als argument en geeft de waarde van de bijbehorende letter uit de cijfertekst als resultaat terug. De functie D_{Atbasj} is de ontsleutelingsfunctie. Deze aanvaardt de waarde van een letter x uit de cijfertekst als argument en geeft de waarde van de bijbehorende letter uit de klare tekst als resultaat terug.

$$E_{Atbasj}(x) = (25 \times x + 25) \bmod 26$$

$$D_{Atbasj}(x) = (25 \times x + 25) \bmod 26$$

Merk op dat de definities van zowel de versleutelingsfunctie als de ontsleutelingsfunctie in dit geval exact hetzelfde zijn.

Caesar: Het Caesar-cijfer is eveneens een eeuwenoude techniek om berichten te versleutelen en te ontsleutelen. Om een letter te versleutelen met het Caesar-cijfer, schuiven we drie plaatsen naar voren. De letter A uit de klare tekst wordt zo de letter D in de cijfertekst, B wordt E, C wordt F enzovoort. Het ontsleutelen van de cijfertekst werkt precies andersom. In plaats van drie plaatsen naar voren te schuiven, schuiven we nu drie plaatsen naar achteren. De letter A uit de cijfertekst wordt zo de letter X in de klare tekst, B wordt Y, C wordt Z enzovoort.

We kunnen dit visueel zien als een afbeelding in tabelvorm:

Klare tekst	A	B	C	D	E	F	G	H	I	J	K	L	M
Caesar-cijfertekst	D	E	F	G	H	I	J	K	L	M	N	O	P

Klare tekst	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Caesar-cijfertekst	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Om met het Caesar-cijfer te versleutelen, zoekt u de letter uit de klare tekst in de bovenste rij op en gaat u naar de onderste rij om de bijbehorende letter uit de cijfertekst te vinden. Om met het Caesar-cijfer te ontsleutelen, zoekt u de letter uit de cijfertekst in de onderste rij op en gaat u naar de bovenste rij om de bijbehorende letter uit de klare tekst te vinden.

Bezien we het Caesar-cijfer vanuit wiskundig oogpunt, dan zien we dat we om te versleutelen 3 moeten optellen bij de numerieke waarde van de letter uit de klare tekst en het resultaat vervolgens modulo 26 moeten reduceren. Dit levert een nieuw getal tussen 0 en 25 op, dat overeenkomt met de versleutelde letter. Om te ontsleutelen, moeten wij 3 van de numerieke waarde van de letter uit de cijfertekst aftrekken en het resultaat modulo 26 reduceren. Dit levert een nieuw getal tussen 0 en 25 op, dat overeenkomt met de ontsleutelde letter.

De functie E_{Caesar} is de versleutelingsfunctie. Zij aanvaardt als argument de waarde x van een letter uit de klare tekst en geeft de waarde van de bijbehorende letter uit de cijfertekst als resultaat terug. De functie D_{Caesar} is de ontsleutelingsfunctie. Zij aanvaardt als argument de waarde x van een letter uit de cijfertekst en geeft de waarde van de bijbehorende letter uit de klare tekst als resultaat terug.

$$E_{Caesar}(x) = (x + 3) \bmod 26$$

$$D_{Caesar}(x) = (x - 3) \bmod 26$$

Merk op dat de definities van zowel de versleutelingsfunctie als de ontsleutelingsfunctie in dit geval verschillend zijn.

Alle drie de bovenstaande methoden kunnen we generaliseren door te stellen dat zij de volgende versleutelings- en ontsleutelingsfuncties gebruiken:

$$E(x) = (A_1 x + B_1) \bmod 26$$

$$D(x) = (A_2 x + B_2) \bmod 26$$

Hierbij zijn A_1, A_2, B_1 en B_2 constanten, die tezamen de *versleutelingssleutel* vormen van een affien cijfer.

Het uitvoeren van het volgende programma toont een tekstmenu. De gebruiker kan kiezen om een tekenreeks te versleutelen of te ontsleutelen, of om het programma af te sluiten. U kunt hiermee oefenen door een zin in klare tekst in te typen, deze te versleutelen en daarna de versleutelde zin weer te ontsleutelen, zodat de oorspronkelijke klare tekst terugkomt.

Een goede voorbeeldtekenreeks om een cijfer mee te testen is:

OP BRUTE WIJZE VING DE SCHOOLJUF DE QUASI-KALME LYNX

Deze tekenreeks is 52 tekens lang en bevat acht spaties, één koppelteken en 43 letters. Elke letter van het alfabet komt ten minste één keer in deze tekenreeks voor, zodat versleutelen en ontsleutelen hiermee controleert dat elke letter op de verwachte wijze wordt omgezet.

Versleuteling van de bovenstaande tekenreeks met het ROT13-cijfer levert het volgende op:

BC OEHGR JVWMR IVAT QR FPUBBYWHS QR DHNFV-XNYZR YLAK

Versleuteling van de bovenstaande tekenreeks met het Atbasj-cijfer levert het volgende op:

LK YIFGV DRQAV ERMT WV HXSLLOQFU WV JFZHR-PZONV OBMC

Versleuteling van de bovenstaande tekenreeks met het Caesar-cijfer levert het volgende op:

RS EUXWH ZLMCH YLQJ GH VFKRROMXI GH TXDVL-NDOPH OBQA

```
10 REM *** CRYPTOGRAFIE ***
20 POKE 0,65: PRINT CHR$(142): PRINT CHR$(147)
30 DEF FN F(X)=X-INT(X/26)*26
40 C$="": P$=""
50 PRINT "SELECT AN OPTION (E, D OR Q):": PRINT
60 PRINT "{SPACE*3}[V] KLARE TEKST VERCIJFEREN": PRINT
70 PRINT "{SPACE*3}[O] CIJFERTEKST ONTCIJFEREN": PRINT
80 PRINT "{SPACE*3}[E] EINDE": PRINT
90 GET S$
100 IF (S$="E") THEN END
110 IF (S$="V") THEN GOSUB 150: GOTO 40
120 IF (S$="O") THEN GOSUB 270: GOTO 40
130 GOTO 90
140 REM VERCIJFEREN
150 INPUT "VOER DE BOODSCHAP IN KLARE TEKST IN: ", P$
160 IF P$="" THEN GOTO 150
170 M$=P$: GOSUB 390
180 IF (V=0) THEN GOSUB 460: GOTO 150
190 A=1:B=3
200 FOR I=1 TO LEN(P$)
210 :   L$ = MID$(P$,I,1)
220 :   IF (L$=" ") THEN C$=C$+" ": ELSE C$=C$+CHR$(65+(FN F(A*(ASC(L$)-65)+B)))
230 NEXT I
240 PRINT: PRINT "{REVERSE ON}DE CIJFERTEKST IS:{REVERSE OFF}", C$: PRINT
250 RETURN
260 REM ONTCIJFEREN
270 INPUT "VOER DE VERCIJFERDE BOODSCHAP IN: ", C$
280 IF C$="" THEN GOTO 270
290 M$=C$: GOSUB 390
300 IF (V=0) THEN GOSUB 460: GOTO 270
310 A=1: B=-3
320 FOR I=1 TO LEN(C$)
330 :   L$ = MID$(C$,I,1)
340 :   IF (L$=" ") THEN P$=P$+" ": ELSE P$=P$+CHR$(65+(FN F(A*(ASC(L$)-65)+B)))
350 NEXT I
360 PRINT: PRINT "{REVERSE ON}ONTCIJFERDE KLARE TEKST:{REVERSE OFF}", P$: PRINT
370 RETURN
380 REM VALIDATE
390 V = 1
400 FOR I=1 TO LEN(M$)
410 :   L$ = MID$(M$,I,1)
420 :   IF NOT (((L$ )= "A") AND (L$ (= "Z")) OR (L$=" ")) THEN V = 0
430 NEXT I
440 RETURN
450 REM FOUTMELDING
460 PRINT: PRINT "GEBRUIK UITSLUITEND LETTERS EN SPATIES": PRINT
470 RETURN
```

Indien u het ROT13-cijfer wilt gebruiken, moet u de volgende regels wijzigen:

```
190 A=1: B=13
310 A=1: B=13
```

Indien u het Atbasj-cijfer wilt gebruiken, moet u de volgende regels wijzigen:

```
190 A=25: B=25
310 A=25: B=25
```

Indien u het Caesar-cijfer wilt gebruiken, moet u de volgende regels wijzigen:

```
190 A=1: B=3
310 A=1: B=-3
```

De programmalijst, zoals die nu staat, gebruikt standaard het Caesar-cijfer.

HOOFDSTUK 12

C64-, C65- en MEGA65-modi

- Modi wisselen vanuit BASIC
- Het KEY-register
- Geheugen benaderen vanuit BASIC 65
- De MAP-instructie

De MEGA65 heeft, net als de C65 en de C128, meerdere bedrijfsmodi. Tussen de MEGA65 en beide eerdere computers bestaan evenwel belangrijke verschillen.

Standaard start het bestand MEGA65.ROM op in de MEGA65-modus (waaronder BASIC 65) en biedt het een methode om via het GO 64-commando naar de C64-modus te schakelen. Het is evenwel ook mogelijk om een originele C65-ROM (versie 91XXXX.BIN) die naar MEGA65.ROM is hernoemd te gebruiken, waardoor de MEGA65 in de C65-modus met BASIC 10 opstart. Ook deze ROM biedt dezelfde functionaliteit om naar de C64-modus te schakelen.

Afhankelijk van uw keuze voor de opstart-ROM hebt u derhalve:

Opstartmodus	ROM-versie	BASIC	C64-modus
MEGA65	92XXXX	BASIC 65	GO 64
C65	91XXXX	BASIC 10	GO 64

Voor lezers die bekend zijn met de C128 is het belangrijkste verschil dat alle nieuwe voorzieningen van de MEGA65 vanuit elke modus toegankelijk zijn en dat u zelfs tussen de verschillende modi heen en weer kunt schakelen. Het is ook mogelijk om hybride modi te maken die voorzieningen uit de verschillende modi combineren - alles wat u nodig hebt, is de **MAP**-instructie en het adres van het KEY-register, namelijk **53295** ($D02F).

Dit hoofdstuk behandelt de verschillende modi, de **MAP**-instructie en het KEY-register. Met het KEY-register kunt u de bedrijfsmodus van de MEGA65 wijzigen. Daarnaast legt dit hoofdstuk uit hoe u met BASIC-commando's van de ene modus naar de andere schakelt.

MODI WISSELEN VANUIT BASIC

De MEGA65 werkt in de C64-modus (met BASIC 2), in de C65-modus (met BASIC 10) bij ROM-versies 91XXXX, of in de MEGA65-modus (met BASIC 65) bij ROM-versies 92XXXX.

Verschillende MEGA65-voorzieningen kunt u echter vanuit alle modi benaderen en alle MEGA65-voorzieningen staan ter beschikking van programma's die in assembleertaal / machinetaal zijn geschreven. Meer informatie over het schrijven van zulke programma's vindt u in de verschillende bijlagen.

Van MEGA65/C65 naar de C64-modus

Om van MEGA65/C65 naar de C64-modus te schakelen, gebruikt u het vertrouwde GO 64-commando dat identiek is aan het overschakelen naar de C64-modus op een C128:

```
GO 64
ARE YOU SURE? Y
```

Let op dat alle programma's in het geheugen verloren gaan bij het wisselen van modus. Dit is hetzelfde als bij de C128. U kunt ook  ingedrukt houden terwijl u op de resetknop drukt of de MEGA65 inschakelt. Ook dit is hetzelfde als bij de C128.

Van C64 naar de MEGA65/C65-modus

Om van C64 naar de MEGA65/C65-modus te schakelen, gebruikt u het volgende commando. **Let op: dit commando vraagt niet om bevestiging**!

```
SYS 58552
```

U kunt ook terugkeren naar de MEGA65/C65-modus door op de resetknop aan de linkerkant van de MEGA65 te drukken of door de MEGA65 uit en weer in te schakelen.

Een andere mogelijkheid is om  een halve tot één seconde ingedrukt te houden en vervolgens in het Freezer-menu op te drukken. Dit simuleert het indrukken van de resetknop.

Let op dat alle programma's in het geheugen verloren gaan bij het wisselen van modus. Dit is hetzelfde als bij de C128.

De machinetaalmonitormodus openen

De machinetaalmonitor opent u door vanuit BASIC 65/10 het commando **MONITOR** in te typen, of door  ingedrukt te houden en vervolgens op de resetknop aan de linkerkant van de MEGA65 te drukken.

HET KEY-REGISTER

De MEGA65 heeft een VIC-IV als videocontroller in plaats van de VIC-II van de C64 of de VIC-III van de C65. Net zoals de VIC-III extra registers heeft ten opzichte van de VIC-II, heeft de VIC-IV er nog meer. Als deze registers voortdurend zichtbaar zouden zijn, bestaat de kans dat software die voor de C64 en de VIC-II is gemaakt, onbedoeld gebruikmaakt van deze nieuwe registers, wat tot onverwacht gedrag leidt. De makers van de C65 hebben derhalve een manier bedacht om de extra registers van de VIC-III voor oude C64-programma's te verbergen. Het in- en uitschakelen (of verbergen en weer zichtbaar maken) van de extra registers gebeurt via het KEY-register. Voor meer informatie over welke registers in elk van de I/O-modi van de VIC-II, VIC-III en VIC-IV zijn uit- en ingeschakeld, raadpleegt u Hoofdstuk/Bijlage J op bladzijde J-7.

Het KEY-register, dat zich op adres **53295** bevindt, is een ongebruikt register van de VIC-II, waarin u met **POKE** kunt schrijven en waaruit u met **PEEK** kunt lezen, net als bij andere registers. Het KEY-register heeft echter een bijzondere functie: schrijft u er twee bepaalde waarden vlak na elkaar naar toe, dan kunt u de VIC-IV opdracht geven om de registers van de VIC-III of VIC-IV niet langer voor de rest van de MEGA65 verborgen te houden.

Extra C65-registers zichtbaar maken

Om bijvoorbeeld de nieuwe registers van de VIC-III in de C64-modus in te schakelen, moet u de waarden 165 en 150 met **POKE** in het KEY-register schrijven. De eenvoudigste manier hiervoor is om de MEGA65 uit en weer in te schakelen, GO 64 te typen en met Y te antwoorden om in de C64-modus te komen.

OPMERKING: Voert u deze POKE's uit terwijl de MEGA65 zich in de C65-modus bevindt, dan kan het voorkomen dat de MEGA65 niet correct functioneert.

Bent u eenmaal in de C64-modus, probeer dan de volgende commando's in te typen:

```
POKE 53295,165: POKE 53295,150
```

Zodra u deze commando's intypt, geeft de MEGA65 de **READY.**-prompt terug en lijkt er verder niets te gebeuren. Dat is te verwachten, want de MEGA65 heeft slechts de nieuwe registers van de VIC-III ingeschakeld (en enkele andere functies uit de C65-modus). De C64-BASIC en de KERNAL blijven normaal functioneren en het lijkt wel of er niets is veranderd... Er *is* echter wel degelijk iets veranderd.

Zo kunt u bijvoorbeeld iets doen wat de C64 met zijn VIC-II niet kan: de ene kleur geleidelijk in de andere laten overgaan. De VIC-III heeft registers waarmee u de rood-, groen- en blauwcomponenten van de kleuren kunt wijzigen. Nu de registers van de VIC-III zijn ingeschakeld, kunt u de kleur van de achtergrond geleidelijk van blauw naar paars laten overgaan, door de roodcomponent van de kleur die op de C64 normaal blauw is te vergroten. De registers voor de waarde van de roodcomponent liggen op 53504 - 53759 ($D100 - $D1FF). Blauw is kleur 6, dus u moet register 53510 (53504 + 6, of $D106) wijzigen. Een voorbeeldlisting in BASIC met een **FOR**-lus om de kleur te wijzigen is:

```
FOR I = 0 TO 15 STEP 0.2 : POKE 53510,I : NEXT
```

Eenmaal het programma is ingetypd, typt u **RUN** op een nieuwe regel. De achtergrond van het scherm vervaagt daarna van blauw naar paars. Wilt u het effect sneller laten verlopen, vergroot dan de 0.2 naar een groter getal, zoals 0.5. Langzamer gaat het met een kleiner getal, zoals 0.02. U kunt de roodcomponent ook wijzigen door met **POKE** een ander getal naar 53504 - 53759 ($D100 - $D1FF) te schrijven, de groencomponent op 53760 - 54015 ($D200 - $D2FF), of de blauwcomponent op 54016 - 54271 ($D300 - $D3FF). Om bijvoorbeeld de rand en de tekst (beide zijn

normaal “lichtblauw”) van blauw naar groen te laten overgaan, kunt u het volgende proberen:

```
POKE 53518,0 : FOR I = 0 TO 15 STEP 0.1 : POKE 53774,I : POKE 54030,15-I : NEXT
```

Extra C65/MEGA65-registers uitschakelen

U kunt de registers van de VIC-III ook weer uitschakelen door met **POKE** een willekeurig getal in het KEY-register te schrijven, bijv.:

```
POKE 53295,0
```

Voert u de bovenstaande voorbeelden opnieuw uit met **RUN**, dan veranderen de kleuren niet, omdat de registers zijn uitgeschakeld. In plaats daarvan verandert het schrijven naar die adressen enkele registers van de VIC-II, omdat deze op een C64 meerdere keren voorkomen. Voor het bovenstaande voorbeeld valt dat gelukkig mee: de gebruikte registers hebben geen voor de hand liggende neveneffecten. Dat komt doordat de registers die in de voorbeelden hierboven worden gewijzigd, op een standaard VIC-II worden gebruikt om de spriteposities te wijzigen. Omdat er geen sprites op het scherm staan, ziet u niets veranderen.

Extra MEGA65-registers inschakelen

De MEGA65 heeft *nog meer* registers dan de C65. Om deze in de C64-modus in te schakelen, moet u nog twee waarden met **POKE** in het KEY-register schrijven:

```
POKE 53295,71: POKE 53295,83
```

Ook nu merkt u niet onmiddellijk een verschil, net als bij het inschakelen van de registers van de VIC-III. De MEGA65 kan er nu echter niet alleen de registers van de VIC-II en VIC-III mee aanspreken, maar ook die van de VIC-IV. Wilt u dat zelf controleren, dan kunt u de voorbeelden van eerder in dit hoofdstuk proberen en ziet u dat de registers van de VIC-III weer toegankelijk zijn. Bovendien kunt u er nu ook dingen mee doen die specifiek zijn voor de MEGA65. Wilt u bijvoorbeeld het begin van de bovenrand hoger op het scherm laten beginnen, dan kunt u zoiets als het volgende proberen:

```
POKE 53320,60
```

U kunt ook voor de aardigheid de schermranden animeren, door ze dichter bij elkaar en verder uit elkaar te laten bewegen:

```
FOR I = 255 TO 0 STEP -1 : POKE 53320,I : POKE 53322, 255 - I : NEXT
```

Het bovenstaande voorbeeld laat de lus achteruit tellen (van 255 naar 0), zodat u niet met slechts een piepklein strookje zichtbare tekst komt te zitten. U kunt de lus desgewenst ook vooruit laten tellen. Blijft u steken met slechts een strookje van het scherm, dan kunt u RUN STOP en RESTORE indrukken. U vraagt zich wellicht af: waarom werken RUN STOP en RESTORE, terwijl het hier registers van de VIC-IV betreft die de BASIC en de KERNAL in de C64-modus niet kennen? De verklaring is dat de VIC-IV een functie heeft die "hete registers" heet, waarbij bepaalde registers van de C64 en C65 ertoe leiden dat sommige registers van de MEGA65 naar de standaardwaarden van de C64- of C65-modus worden teruggezet. In dit specifieke geval is het de KERNAL die het VIC-II-scherm opnieuw instelt via 53265 ($D011); dit register stelt in de C64/C65-modus de grootte van de verticale rand bij en is dus een "heet register" voor de registers met de verticale randpositie van de MEGA65.

Kijk of u het scherm in plaats daarvan kunt laten schudden door de registers TEXTXPOS en TEXTYPOS van de VIC-IV te wijzigen. De **POKE**-codes voor die registers, en voor nog veel meer andere interessante nieuwe registers, vindt u door te bladeren in Hoofdstuk/Bijlage Q op bladzijde Q-5.

Valkuilen

In alle modi zet de DOS van het interne 3.5"-diskettestation (ook wanneer u D81-schijfimages van een SD-kaart gebruikt) het KEY-register bij elke toegang terug naar de VIC-II-modus. Voert u derhalve acties uit zoals het opvragen van de stationstatus, of het laden of opslaan van een bestand met **LOAD** of **SAVE**, dan wordt het KEY-register gereset en worden uitsluitend de registers van de VIC-II ingeschakeld. U kunt de registers van de C65 of de MEGA65 uiteraard weer inschakelen door de juiste waarden opnieuw met **POKE** naar het KEY-register te schrijven.

GEHEUGEN BENADEREN VANUIT BASIC 65

BASIC 65 bevat krachtige commando's voor geheugenbanking en Direct Memory Access (DMA), waarmee u geheugengebieden buiten de 128KB RAM van de C65 kunt lezen, vullen, kopiëren en beschrijven. De MEGA65 heeft 384KB hoofdgeheugen, verdeeld over 6 banken van elk 64KB. Dat zijn:

- BANK 0 en BANK 1 - vormen samen de normale 128KB RAM van de C65.
- BANK 2 en BANK 3 - zijn doorgaans schrijfbeveiligd en bevatten het ROM-image van de C65.
- BANK 4 en BANK 5 - worden gebruikt voor alle grafische routines van BASIC 65 voor bitplanegrafiek met hoge resolutie. BASIC 10 gebruikt de banken 4 en 5 niet.

Met de commando's **BANK**, **PEEK** en **POKE** kunt u dit geheugengebied eenvoudig benaderen, bijvoorbeeld:

```
BANK 4: POKE0,123: REM PLAATS 123 IN LOCATIE $40000
BANK 4: PRINT PEEK(0): REM TOON INHOUD VAN LOCATIE $40000
```

U kunt de huidige inhoud van het scherm en het kleurengeheugen (colour-RAM) echter ook met het **DMA**-commando naar BANK 4 kopiëren:

```
DMA 0, 2000, 2048, 0, 0, 4 : REM SCHERMTEKST NAAR BANK 4
DMA 0, 2000, DEC("F800"), 1, 2000, 4 : REM KOPIEER KLEURENRAM NAAR BANK 4
```

Vervolgens kunt u iets anders op het scherm zetten en dit met het volgende commando terugkopiëren:

```
DMA 0, 2000, 0, 4, 2048, 0 : REM SCHERMTEKST VANUIT BANK 4
DMA 0, 2000, 2000, 4, DEC("F800"), 1 : REM KOPIEER KLEURENRAM VANUIT BANK 4
```

DE MAP-INSTRUCTIE

De bovenstaande methoden kunt u vanuit BASIC gebruiken. De **MAP**-instructie is daarentegen een assembleertaalinstructie, waarmee u het geheugen van de MEGA65 opnieuw kunt indelen. De C65-ROM en BASIC 65 gebruiken deze instructie om te beheren welk geheugen op elk gegeven moment bruikbaar is. Voor een nadere toelichting op de **MAP**-instructie verwijzen wij u naar de betreffende sectie van Hoofdstuk/Bijlage K op bladzijde K-7.

DEEL IV

GELUID EN GRAFIEK

HOOFDSTUK 13

Grafiek

Laten we nu eens plezier beleven aan grafiek! In dit deel van het boek willen wij de grafiekmodi van de MEGA65 bekijken door voorbeeldcode in machinetaal en BASIC stap voor stap te doorlopen, zodat u de verschillende mogelijkheden van de MEGA65 op het gebied van grafiek leert kennen. Allereerst is het belangrijk te weten dat de MEGA65 drie verschillende basisgrafiekmodi ondersteunt:

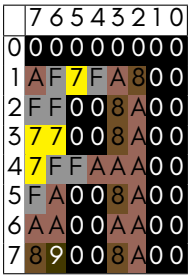
- Bitmapgrafiek
- Grafiek gebaseerd op tekensets
- Bitplanes

BITMAPGRAFIEK

In bitmapgrafiek wordt elke pixel van een afbeelding afzonderlijk opgeslagen. De manier waarop de pixels in het geheugen worden bewaard, verschilt van systeem tot systeem en hangt in de meeste gevallen af van de prestaties van de hardware. Als het geheugen onbeperkt zou zijn, zou de eenvoudigste manier om een pixel te onthouden zijn om zijn RGB-waarden in drie aparte bytes op te slaan. Voorbeeld: 0xFFFFFF voor wit zou betekenen dat er drie waarden opgeslagen moeten worden: $FF, $FF en $FF. Eerlijk gezegd is dit te eenvoudig en niet echt efficiënt. Laten wij dus een andere manier bedenken. Waarom leggen wij niet een kleurentabel (of kleurenpalet) vast, slaan de RGB-waarden één keer op en verwijzen daarna alleen nog naar de kleur via de index ervan in de tabel? Dat bespaart ons veel geheugen! Stel dat wij een kleurrijke 8x8-bitmap maken, die op het scherm een "A" moet voorstellen. Kleurrijk wil hier zeggen dat wij hem in wat bruintinten willen hebben. De kleurentabel daarvoor kan er dan als volgt uitzien:

Index	Kleur
0	Zwart
9	Bruin
8	Oranje
A	Lichtrood
F	Lichtgrijs
7	Geel

De kleurwaarden zijn overigens precies dezelfde als de kleurwaarden uit het standaard C64-kleurenpalet. Daarna ontwerpen we de "A". Elke pixel verwijst naar een waarde uit de bovenstaande kleurentabel.



Hoeveel geheugen gebruikt dit kleine afbeeldingetje eigenlijk?

Als we een eendimensionale array maken, krijgen we een array met 64 elementen, want onze afbeelding bestaat uit 8x8 pixels = 64 kleurwaarden die opgeslagen moeten worden. Vertaald naar het geheugen van de MEGA65 betekent dit dat we 64 bytes in het geheugen moeten bewaren. Een afbeelding die het volledige scherm vult, bestaat evenwel uit veel meer pixels. Op de C64 en natuurlijk ook op de MEGA65 zijn 320x200 pixels nodig om een afbeelding te maken die het hele scherm vult. Als we dat naar onze array doorrekenen, komen we uit op in totaal 64000 elementen. Omgerekend naar het geheugen van de MEGA65 zijn dat 64000 bytes, oftewel bijna 64 kilobyte aan gegevens! Als we daarbij ook nog bedenken dat de goede oude C64 slechts over 64K RAM beschikte, dan wordt de dramatiek duidelijk! Dat is simpelweg veel te veel gegevens! We hebben strategieën nodig om onze bitmapgegevens te beperken. Op de C64 waren er twee typen bitmapgrafieken, elk met zijn eigen mechanismen om het geheugengebruik tot een minimum te beperken.

- Hires
- Multicolour (MCM)

Hires

Allereerst delen we een bitmap op in blokken van elk 8x8 pixels. Om de volledige resolutie van 320x200 pixels te bereiken, vormen 40 van zulke blokken naast elkaar één regel. Bouwen we vervolgens 25 van die regels op, dan komen we uit op een afbeelding die het volledige scherm vult.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1																																									
2																																									
3																																									
4																																									
5																																									
6																																									
7																																									
8																																									
9																																									
10																																									
11																																									
12																																									
13																																									
14																																									
15																																									
16																																									
17																																									
18																																									
19																																									
20																																									
21																																									
22																																									
23																																									
24																																									
25																																									

De opdeling in blokken is zinvol, want daardoor kunnen wij de gegevens aanzienlijk verkleinen. Elke regel van een 8x8-blok bestaat uit 8 bits, dus waarom zouden wij de kleurindexen niet loslaten en simpelweg afspreken dat een gezette pixel een "1" in de regel voorstelt en een niet-gezette pixel overeenkomt met "0".

	7	6	5	4	3	2	1	0	dec	hex
0	0	0	0	0	0	0	0	0	0	00
1	0	1	1	1	1	0	0	0	120	78
2	1	1	0	0	1	1	0	0	204	CC
3	1	1	0	0	1	1	0	0	204	CC
4	1	1	1	1	1	1	0	0	252	FC
5	1	1	0	0	1	1	0	0	204	CC
6	1	1	0	0	1	1	0	0	204	CC
7	1	1	0	0	1	1	0	0	204	CC

Zoals hierboven te zien is, kunt u de bits van elke regel nu eenvoudig omrekenen naar de bijbehorende hexadecimale waarde en houdt u uiteindelijk 8 bytes aan gegevens over. Dit is de centrale gedachte achter hires-grafiek en met dit concept bespaart u veel geheugen. In dit voorbeeld zijn het 8 bytes, tegenover het vorige voorbeeld,

waarin 64 bytes nodig waren om alle kleurverwijzingen te beheren. Rekenen we het eens door voor een afbeelding die het volledige scherm vult: we hebben 40x25 blokken, dat zijn er in totaal 1000. Elk blok is 8 bytes, dus dat levert "slechts" 8K op voor een afbeelding op volledig scherm.

Dit is een briljante oplossing, maar nu hebben wij een probleem: wij zijn de kleuren kwijt!

Als u naar de eerste afbeelding kijkt, ziet u dat die erg kleurrijk was. Hebben we die vele kleuren echter wel nodig? Dit is het tweede belangrijke concept in hires-bitmapgrafiek: minder kleuren betekent minder geheugen. De hires-modus beperkt u in feite tot **twee kleuren** per 8x8-pixelblok. Deze informatie wordt bewaard in het schermgeheugen (Screen-RAM) en niet, zoals u misschien zou aannemen, in het kleurengeheugen (Colour-RAM). Ook het idee erachter is opnieuw erg vernuftig: in het schermgeheugen kunt u waarden tussen $00 en $FF bewaren, terwijl het in het kleurengeheugen alleen zin heeft om waarden tussen $0 en $0F op te slaan, om één specifieke kleur uit het C64-kleurenpalet aan te duiden, dat uit 16 kleuren bestaat ($0-$F). Op de MEGA65 zijn zelfs nog meer kleuren mogelijk en u kunt de standaardkleuren aanpassen, maar dat wordt later uitgelegd.

Terug naar het schermgeheugen. Hier bewaart u **twee kleuren** voor elk 8x8-blok. Als u de byte splitst in zijn hoge en lage nibble, heeft u de mogelijkheid om er een achtergrond- en een voorgrondkleur in onder te brengen! De hexadecimale waarde $0a bijvoorbeeld kunt u lezen als $0 (hoge nibble) en $a (lage nibble). Op die manier krijgen wij een zwarte achtergrond en lichtrood voor de pixels in het blok. Uiteindelijk betekent dit dat een hires-bitmap voor het volledige scherm niet 8 maar 9 kilobyte gebruikt: 8K voor de ruwe bitmapgegevens, zoals eerder genoemd, en nog eens 1K voor de kleuren in het schermgeheugen.

Eenvoudige hires-bitmaps programmeren

Laten we gaan programmeren! We doen dat eerst in BASIC, voordat we overschakelen naar machinetaal. Om bitmapgrafieken op het scherm te brengen, moeten we de VIC-IV-chip configureren. De VIC moet een paar dingen weten voordat de chip de bitmapgegevens kan vinden en lezen. U moet het volgende doen:

- Zorg dat u de VIC-registers heeft geactiveerd die u nodig heeft (denk eraan: de MEGA65 biedt verschillende versies van de VIC-chip)
- Stel de resolutie in
- Bepaal een locatie voor de bitmapgegevens
- En schakel ten slotte de bitmapmodus in

Al deze instellingen kunt u in verschillende registers van uw MEGA65 configureren. Een overzicht van de bestaande registers van de grafische chip (dat is uiteraard de VIC) vindt u in Hoofdstuk/Bijlage Q op bladzijde Q-5.

Uw MEGA65 bevat één grafische chip, de **VIC-IV**, die voortbouwt op zijn twee voorgangers en aan die voorgangers nieuwe functies in de vorm van extra registers toevoegt. Deze twee voorgangers zijn de **VIC-II**, zoals die in de oorspronkelijke Commodore 64 werd gebruikt, en de **VIC-III**, ook wel "Bill" genoemd, zoals die voor de Commodore 65 was gepland.

Het is belangrijk te begrijpen dat deze verschillende VIC-versies op elkaar voortbouwen; beschouw ze als modi die u aan of uit kunt zetten. VIC-III breidt de bestaande registers van de VIC-II uit, terwijl de VIC-IV u alle bestaande registers van VIC-II en VIC-III biedt en daarop voortbouwt door aan eerder ongebruikte registers nieuwe functionaliteit toe te voegen. Dat betekent dat in Hoofdstuk/Bijlage Q op bladzijde Q-5 niet alleen de registers van de VIC-IV belangrijk zijn om te kennen, maar ook die van VIC-II en VIC-III, afhankelijk van welke VIC-modus u wel of niet heeft geactiveerd. Houd hiermee rekening.

Zorg dat u de VIC-registers heeft geactiveerd die u nodig heeft - wat betekent dat? In grote lijnen gebruikt u het nieuwe register $D02F hex of 53295 decimaal om tussen de VIC-modi te schakelen (d.w.z. om specifieke registers toe te voegen of te verwijderen). Voor het schakelen gebruikt u een zogeheten klopsequentie, door achtereenvolgens twee specifieke waarden te poken:

VIC	Eerste hex/dec	Tweede hex/dec
II	$00/0	$00/0
III	$A5/165	$96/150
IV	$47/71	$53/83

Terzijde: de twee waarden waarmee u de VIC-IV activeert, zijn de ASCII-tekens "G" en "S". Dat zijn de initialen van de maker van de VIC-IV: universitair hoofddocent Paul Gardner-Stephen. Een leuk paaseitje, nietwaar?

Het is belangrijk te begrijpen dat de computer, wanneer u uw MEGA65 inschakelt, in de C64-modus start en vervolgens zoekt naar een C65-compatibele ROM. Wordt een C64-cartridge gedetecteerd of drukt u de MEGA-toets in, dan blijft het systeem in de C64-modus. In dat geval heeft u alleen de C64-registers tot uw beschikking. Vindt het systeem een C65-ROM, dan activeert het de beschikbare registers. Bij de MEGA65-ROM betekent dit dat u alle VIC-registers (II, III en IV) kunt benaderen. Geeft u het GO64-commando, dan beperkt het systeem u tot de VIC-II-registers - maar denk eraan - door de klopsequentie zoals hierboven beschreven uit te voeren, kunt u die registers weer toevoegen! Het maakt dus niet uit in welke "modus" u zich bevindt. Alle mogelijkheden zijn overal beschikbaar. De MEGA65 is werkelijk een flexibele machine!

In dit hoofdstuk gaan we ervan uit dat u in de C65-modus werkt en daarbij de standaard-ROM van de MEGA65 met BASIC65 gebruikt.

De volgende stap is het **instellen van de resolutie**. Volgens de registers in Hoofdstuk/Bijlage Q op bladzijde Q-5 kunt u dat doen met $D031 hex, of 53297 decimaal.

Dit register is afkomstig van VIC-III. De C65 beschikt over een beeldscherm met 80 kolommen, de C64 had er slechts 40. Om 80 kolommen te realiseren, moet de grafische chip met hogere resoluties overweg kunnen. Wanneer u de computer inschakelt, werkt u in de 80-kolomsmodus met 25 regels, hetgeen neerkomt op een resolutie van 640x200. Om klassieke bitmaps te tonen, moet u terug naar 320x200 (dat zijn de 40 kolommen en 25 regels, zoals al in de eerdere tabel te zien was). De C64 kon 640x200 niet weergeven, dus had de VIC-II geen register nodig om dat te regelen. De C65 beheerst daarentegen beide, zodat u met $D031 ertussen kunt schakelen. Manipuleert u het register bitgewijs, dan kunt u nog meer instellen: de resolutie (databit 7, H640), de C65-fastmodus (databit 6, FAST) of het in- en uitschakelen van de nieuwe bitplane-modus als alternatief voor klassieke bitmap- of tekengebaseerde grafiek (databit 4, BPM). Sommige registers zijn wel gepland, maar nooit geïmplementeerd. Bijvoorbeeld een H1280-modus op databit 2, welke de C65 in staat moest stellen op te schalen naar 1280x200 of 1280x400. Zie Hoofdstuk/Bijlage Q op bladzijde Q-5 voor alle fijne details.

De volgende vraag is: hoe kunnen we specifieke bits in een register aansturen? Schakel uw computer in en typ het volgende commando in

```
PRINT PEEK($D031)
 224
```

U krijgt dan een decimale waarde terug, zoals 224. De decimale waarde helpt u niet verder: u moet het getal omzetten naar zijn binaire voorstelling.

Bit 7 H640	Bit 6 FAST	Bit 5 ATTR	Bit 4 BPM	Bit 3 V400	Bit 2 H1280	Bit 1 MONO	Bit 0 INT	Decimale waarde
1	1	1	0	0	0	0	0	224
0	1	1	0	0	0	0	0	96

De eerste rij in de tabel toont het binaire getal voor decimaal 224; dit zou het geval moeten zijn wanneer u uw MEGA65 inschakelt. Databit 7, de H640-modus, is zoals verwacht geactiveerd. In de tweede rij ziet u de waarde die u moet gebruiken wanneer u wilt terugschakelen naar 320x200: databit 7 wordt op 0 gezet, zodat de H640-modus is uitgeschakeld.

```
POKE $D031,96
```

Wacht echter even: de waarde rechtstreeks poken is niet altijd de juiste aanpak. Als u precies weet wat u doet - of beter: welke toestand elke bit van het register moet hebben - kunt u op deze manier verdergaan. Het wegschrijven van de waarde 96 betekent dat u alle databits in één keer instelt. Dat is niet altijd wat we willen. In sommige gevallen willen we slechts één specifieke bit manipuleren, bijvoorbeeld databit 7 voor de resolutie, terwijl de overige bits in hun bestaande toestand blijven. Dat bereikt u met logische bewerkingen, met behulp van de AND- of OR-operator.

Bewerking	Resultaat
0 AND 0	0
1 AND 0	0
0 AND 1	0
1 AND 1	1

Bewerking	Resultaat
0 OR 0	0
1 OR 0	1
0 OR 1	1
1 OR 1	1

De logische tabellen hierboven zijn belangrijk, omdat zij laten zien hoe de bits worden gewijzigd als u er een operator op toepast. Als u nadenkt over het idee van logische bewerkingen in het voorbeeld dat volgt, ziet u dat u AND kunt gebruiken om een specifieke bit op nul te zetten (te wissen, via een masker), en OR om de bit op 1 te zetten. Eerlijk gezegd is deze methode niet alleen bruikbaar om één enkele bit in te stellen, maar ook om een willekeurig aantal bits te wijzigen; om het eenvoudig te houden wijzigen we er slechts één. Terug naar de "slechte" poke van 96 die we hierboven zagen: die kunt u beter als volgt schrijven:

```
POKE $D031, PEEK($D031) AND 127 :REM RESET BIT 7 (d.w.z. zet het op nul)
POKE $D031, PEEK($D031) OR 128  :REM ZET BIT 7 (d.w.z. zet het op   n)
```

Om deze bewerkingen te begrijpen, moet u opnieuw elk decimaal getal omzetten naar zijn binaire vorm. Daarna passen we de AND / OR-bewerkingen bit voor bit toe, volgens de regels uit de voorgaande logische tabellen.

	Binair	Decimaal
	11100000	224
AND	01111111	127
Resultaat	01100000	96

	Binair	Decimaal
	01100000	96
OR	10000000	128
Resultaat	11100000	224

Dit kunt u overigens testen met enkele eenvoudige PRINT-commando's:

```
PRINT 224 AND 127
  96

PRINT 96 OR 128
  224
```

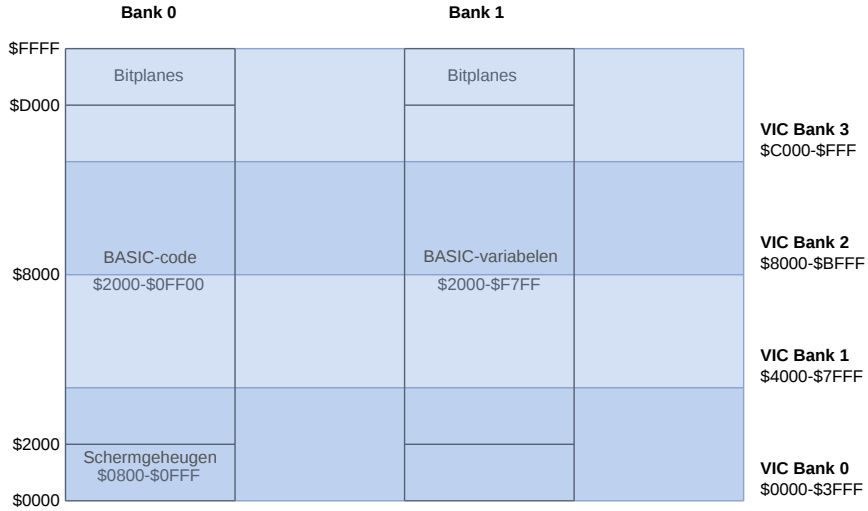
Gewapend met deze kennis kunnen we de VIC nu zodanig configureren dat deze bitmapgrafiek weergeeft. Daartoe moeten we een aantal registers fijnafstellen, maar voordat we daarmee kunnen beginnen, is het nodig om iets meer te begrijpen van de grafische chip, hoe deze is opgebouwd en vooral hoe hij het geheugen benadert.

Kanttekening:

Een alternatieve aanpak, beschikbaar in BASIC 65 voor het zetten/wissen van bits, is het gebruik van de nieuwe **SETBIT**- en **CLRBIT**-commando's. Voor wie in assembleertaal programmeert, zijn bovendien de nieuwe instructies **SMBx** (zet bit x) en **RMBx** (wis bit x) beschikbaar.

De organisatie van de VIC-IV en de basis van geheugentoegang

In tegenstelling tot de CPU van uw MEGA65, welke in één keer 64KB kan adresseren, kan de VIC dat helaas niet. Hij kan slechts 16KB overzien, dus u moet precies definiëren welke 16KB hij mag zien. Hiertoe wordt de 64KB van Bank 0 (dit is de geheugenbank waarin de VIC huist en zijn gegevens vandaan haalt) verdeeld in blokken van elk 16KB. Deze 16KB-blokken worden "videobanken" of "VIC-banken" genoemd.



Los van de VIC is het geheugenbeheer van de Commodore 65 of MEGA65 iets complexer dan dat van het vorige model. Dat komt doordat er in de machine meer geheugen beschikbaar is: bij de Commodore 65 waren 128KB gepland (mogelijk uit te breiden tot 1MB), terwijl de MEGA65 wordt geleverd met 384KB chip-RAM + 8MB zolder-RAM (mogelijk uit te breiden tot 256MB). Zoals eerder vermeld, kan de processor niet het hele geheugenbereik adresseren. Er is slechts 64KB zichtbaar. Om die reden is het geheugen verdeeld in blokken van elk 64KB en deze blokken worden "banken" genoemd.

Het systeem kan tot 16 banken hebben en bepaalde banken hebben daarnaast ook specifieke taken. Bank 0 en 1 zijn RAM, wat betekent dat u er zowel kunt lezen als schrijven, terwijl bank 2 en 3 ROM zijn en niet zo eenvoudig te overschrijven.

Bank 0 (gezien vanuit het besturingssysteem BASIC 65 dat start wanneer u de computer inschakelt) heet de systeembank. Deze bevat onder meer de volgende gebieden:

- Het schermgeheugen, welke het geheugenbereik $0800-$2000 beslaat (standaard)
- In het geheugen gemapte VIC-registers (indien deze zijn ingemapt, wat standaard het geval is)

Overigens staat de implementatie van BASIC65 zelf in ROM (bank 3), terwijl het BASIC van de C64 eveneens in ROM staat (bank 2).

Wilt u meer weten over geheugenbeheer in het algemeen, kijk dan bij Hoofdstuk/-Bijlage J op bladzijde J-3. In BASIC is de situatie helaas nog complexer. Direct na het inschakelen van de computer bevindt u zich in Bank 128, een speciale bank met een speciale geheugenconfiguratie. U kunt het commando BANK gebruiken om tussen banken te wisselen. Een probleem daarbij is dat u gemakkelijk het overzicht verliest over wat er in welke bank staat of in welke bank u op dat moment werkt. Het is eigenlijk iets eenvoudiger wanneer u in machinetaal werkt. Voor meer informatie over het BANK-commando en over banken in het algemeen raadpleegt u Hoofdstuk/Bijlage B op bladzijde B-26. Houd er rekening mee dat u eerst iets over banken moet weten voordat u op veilige wijze PEEK, POKE of SYS kunt gebruiken. Anders vraagt u zich misschien af waarom uw programma zich niet zo gedraagt als u verwachtte. Wij komen hier later op terug. Terug naar de grafiek.

Laten we beginnen met het schrijven van een eenvoudig BASIC-programmaatje om hires-bitmapgrafiek weer te geven. Indien u dit hoofdstuk tot op dit punt zorgvuldig heeft gevolgd, zult u geen moeite hebben met het begrijpen van de volgende coderegels:

```
10 POKE $D020,0 : POKE $D021,0          :REM MAAK SCHERM ZWART
20 POKE $D031, PEEK($D031) AND 127      :REM STEL RESOLUTIE IN
30 POKE $DD02, 3                        :REM MAAK VIRTUALE CIA-2 LEES- EN SCHRIJFBAA
40 POKE $$DD00, PEEK($DD00) AND 252 OR 2:REM KIES VIDEOBANK 1
```

Zoals u op de afbeelding op de vorige bladzijde heeft gezien, verdeelt de VIC-chip het geheugen in 4 videobanken van elk 16KB. Het is belangrijk te begrijpen dat u geen register van de VIC gebruikt om de videobank binnen (geheugen-)bank 0 te definiëren; in plaats daarvan gebruikt u de databits 0 en 1 van $DD00 of decimaal 56576. Dit was ooit een register van de CIA-2-chip in de Commodore 64, maar in uw MEGA65 is het geïmplementeerd als virtueel register. De "voorbereiding" die nodig is om de videobank in te stellen, gebeurt in regel 30, terwijl in de daaropvolgende regel de videobank zelf wordt ingesteld. Overigens geldt: als u de decimale waarde 3 (regel 30) omzet naar binair formaat, is dat 00000011 en dat bewijst dat de databits 0 en 1 op 1 staan, zodat lezen en schrijven is geactiveerd.

Bits	VIC-bank	Geheugengebied	BASIC
00	3	$C000-$FFFF	... AND 252 OR 0 :REM AND 11111100 OR 000000**00**
01	2	$8000-$BFFF	... AND 252 OR 1 :REM AND 11111100 OR 000000**01**
10	1	$4000-$7FFF	... AND 252 OR 2 :REM AND 11111100 OR 000000**10**
11	0	$0000-$3FFF	... AND 252 OR 3 :REM AND 11111100 OR 000000**11**

De tabel toont welke bits moeten worden ingesteld om een videobank te selecteren.

DEEL V

CPU EN GEHEUGEN

HOOFDSTUK 14

Kennismaken met de 45GS02-CPU

- Werken met de monitor
- CPU-instructies
- CPU-registers
- De Base-Page
- Indirecte adressering
- Het virtuele 32-bits register
- De stapel
- Interrupts

- Instructies coderen
- De 6502-familie

Aan dit hoofdstuk wordt nog gewerkt.

- Hoe werkt een CPU
- Kenmerken van de 45GS02

WERKEN MET DE MONITOR

- wat is een machinetaalmonitor
- MONITOR-commando
- registers bekijken
- geheugen bekijken
- code disassembleren
- code assembleren
- een subroutine uitvoeren
- onderbreken en hervatten

CPU-INSTRUCTIES

- Instructies
- Adresseermethoden
- Registers en statusvlaggen
- Geheugen lezen en schrijven
- Berekeningen uitvoeren
- Vertakkingen en subroutines
- MAP en EOM

CPU-REGISTERS

- Accumulatorregister
- X, Y, Z
- Programmateller
- Stackpointer

- Base-Page (B)
- Statusvlaggen; het statusregister
- MAP

DE BASE-PAGE

- Doel: snel tijdelijk geheugen; pointers
- Zero-Page
- Het B-register
- Base-Page-adresseermethoden
- Andere Base-Page-bewerkingen
- KERNAL-overwegingen

INDIRECTE ADRESSERING

HET VIRTUELE 32-BITS REGISTER

DE STAPEL

- Waarden op de stapel plaatsen en van de stapel halen
- Subroutine-aanroepen
- 8-bits versus 16-bits stackpointer
- Stackrelatieve adresseermethode

INTERRUPTS

- Typen interrupts
- Interruptvectoren
- Interface voor interrupthandlers
- Een voorbeeld zonder de KERNAL
- De KERNAL-IRQ uitbreiden

- Opmerking over het MAP-register

INSTRUCTIES CODEREN

- Opcodes en operanden
- 32-bits indirecte adresseermethode
- Instructies voor het virtuele 32-bits register

DE 6502-FAMILIE

- 6502 / 6510
- 65C02
- 65CE02 (“4502”) / 4510
- 45GS02

HOOFDSTUK 15

Programmeren met het geheugen

- Geheugenbegrippen
- De 28-bits adresruimte
- De geheugenindeling van het chip-RAM
- Geheugen gebruiken vanuit BASIC
- Geheugen gebruiken vanuit machinetaal
- Videogeheugen van de VIC-IV
- Grote geheugenbewerkingen met Direct Memory Access
- Gevorderde geheugenonderwerpen

Als u programma's in assembleertaal schrijft, is het van cruciaal belang dat u begrijpt hoe het geheugen werkt. Vrijwel elke bewerking heeft op de een of andere manier met het geheugen te maken. Gegevensverwerking vindt in het geheugen plaats en programma's worden in het geheugen opgeslagen en vanuit het geheugen uitgevoerd. Bij een hardware-registerarchitectuur in Commodore-stijl loopt alle in- en uitvoer van de computer via het manipuleren van waarden in het geheugensysteem.

Als u programma's in MEGA65 BASIC schrijft, hoeft u zich nauwelijks met het geheugen bezig te houden. Het BASIC-systeem en de meeste van zijn commando's vertegenwoordigen gegevens als getallen, arrays en strings die in variabelen met een naam zijn opgeslagen. U hoeft nooit te weten hoe die gegevens als enen en nullen worden weergegeven, of welke adressen bij welke gegevens horen. Toch kan inzicht in de werking van het geheugen uw BASIC-programma's naar een hoger niveau tillen. U kunt nieuwe manieren bedenken om de gegevens van uw programma te beheren en u kunt rechtstreeks contact leggen met de MEGA65-hardware. Sommige geavanceerde mogelijkheden van de MEGA65 zijn uitsluitend via geheugenregisters toegankelijk.

Net als verschillende andere systemen in de MEGA65 is het geheugensysteem geavanceerder dan u wellicht verwacht, met meerdere bedrijfsmodi. Wanneer u uw MEGA65 voor het eerst inschakelt, begint deze met een geheugenconfiguratie die bedoeld is om de BASIC 65-omgeving te ondersteunen. Een programma kan het geheugen opnieuw configureren om de mogelijkheden van de MEGA65 volledig te benutten, of om de geheugenomgeving compatibel te maken met een Commodore 64 (zoals **GO64** doet) of een Commodore 65.

In dit hoofdstuk leert u alles over het geheugensysteem van de MEGA65, met de nadruk op het schrijven van programma's voor de MEGA65 in BASIC en assembleertaal. Volledige beschrijvingen van de geheugenvoorzieningen, inclusief de functies voor achterwaartse compatibiliteit met de C64 en C65, vindt u in de bijlagen.

GEHEUGENBEGRIPPEN

Een computer voert taken uit door gegevens te manipuleren die in zijn *geheugen* zijn opgeslagen. Het computergeheugen bestaat uit vele duizenden cellen, waarvan elke cel één *byte* aan gegevens bevat die 256 mogelijke waarden kan vertegenwoordigen. Elke cel heeft een numeriek *adres*. Het programma dat het geheugen gebruikt, bepaalt wat voor soort gegevens op elk adres staat. Het programma zelf staat als bytes in het geheugen, als machinetaalinstructies die de CPU uitvoert.

Bits, bytes en nibbles

Een *bit* is één enkel digitaal signaal met twee mogelijke waarden: uit of aan. Vaak worden bits met andere bits gecombineerd om meer mogelijke waarden te kunnen vertegenwoordigen. Een byte bestaat uit 8 bits en heeft 256 mogelijke waarden.

Bits en bytes worden vaak gebruikt om getallen weer te geven. Hun waarden worden daarbij vaak beschreven als positieve gehele getallen, ook wanneer zij andere soorten gegevens vertegenwoordigen. Als we het over de bits van een byte hebben, staan die van rechts naar links in oplopende plaatswaarde. Bit 0 is de meest rechtse bit, ook bekend als de "minst significante" bit. Bit 7 is de meest linkse bit, ook bekend als de "meest significante" bit.[1]

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
0	1	1	0	1	0	1	1	= 107
128	64	32	16	8	4	2	1	

Een *nibble* is 4 bits. In de computerarchitectuur verwijst de term "nibble" naar één van de twee helften van een byte. De *hoge nibble* bestaat uit bits 7 – 4 en de *lage nibble* uit bits 3 – 0.

Hexadecimale notatie

Hexadecimaal is een talstelsel waarin elk cijfer zestien mogelijke waarden kan aannemen, geschreven met de cijfers 0 – 9 gevolgd door de letters A – F. Het is bijzonder handig voor het beschrijven van waarden in het computergeheugen, omdat elk hexadecimaal cijfer een nibble (4 bits) aan gegevens vertegenwoordigt. De 8-bits binaire waarde `%01101011` kunt u lezen als de twee nibbles `0110 1011`, ofwel $6B in hexadecimale notatie.

Geheugenadressen worden doorgaans in hexadecimale notatie weergegeven. Het computergeheugen is gewoonlijk zo ingedeeld dat de groottes als ronde hexadecimale getallen uit te drukken zijn, zoals $C000.

$0000	$0001	$0002	···	$0FFE	$0FFF	$1000	$1001	···

[1] Zie Hoofdstuk/Bijlage H op bladzijde H-3 voor een volledige uitleg van de binaire en hexadecimale notatie.

Ronde hexadecimale getallen komen overeen met veelvoorkomende geheugengroottes:

```
   $0001 1 byte
   $0010 16 bytes
   $0100 256 bytes
   $0400 1 kilobyte (1KB = 1024 bytes)
   $1000 4 kilobytes (4KB = 4096 bytes)
   $4000 16KB
  $10000 64KB
$100000 1 megabyte (1MB = 1024KB)
```

De adressering begint bij 0, dus gebieden van deze gebruikelijke groottes beginnen aan de rechterkant doorgaans met "0"-cijfers en stoppen vlak vóór het begin van het volgende gebied. Het bereik $2000 - $2FFF is bijvoorbeeld 4KB groot.

In BASIC kunt u getallen hexadecimaal invoeren door het dollarteken als voorvoegsel te gebruiken ($). Om bijvoorbeeld de decimale waarde van een hexadecimaal getal in BASIC weer te geven:

```
PRINT $C000
```

Om een getal in hexadecimale notatie in BASIC weer te geven, gebruikt u de functie **HEX$()**:

```
PRINT HEX$(127)
```

Als u in assembleertaal programmeert of een machinetaalmonitor gebruikt, worden adressen en waarden standaard in hexadecimale notatie weergegeven.

Dit boek gebruikt soms een punt (.) om groepen van vier hexadecimale cijfers van elkaar te scheiden, zodat ze gemakkelijker te lezen zijn: FFF.FFFF. Als u een lang adres in een programma of commando invoert, tikt u de punt niet mee. Om FFF.FFFF in BASIC in te voeren, tikt u: $FFFFFFF

Elk hexadecimaal cijfer vertegenwoordigt een nibble van een waarde. De waarde FFF.FFFF kan met zeven nibbles (28 bits) worden weergegeven.

Geheugen met willekeurige toegang

Random Access Memory (RAM) is geheugen dat gegevens tijdelijk bevat. Een programma kan deze gegevens lezen en ernaar schrijven. Wanneer de computer wordt uitgeschakeld, gaan deze gegevens verloren. Een programma dat deze gegevens wil bewaren, moet ze opslaan op een apparaat voor langetermijnopslag, zoals een diskette (of de SD-kaart van de MEGA65), voordat de computer wordt uitgeschakeld.

Probeer dit. Typ het volgende commando in bij de READY.-prompt:

```
POKE $1800,127
```

Dit **POKE**-commando slaat de bytewaarde 127 op geheugenadres $1800 op. Typ nu dit commando in:

```
PRINT PEEK($1800)
```

De **PEEK()**-functie levert de bytewaarde op die op geheugenadres $1800 is opgeslagen, in dit geval 127. Het **PRINT**-commando geeft deze waarde weer.

Schakel uw MEGA65 uit, schakel hem weer in en voer `PRINT PEEK($1800)` opnieuw uit. De waarde in het RAM-geheugen is teruggezet.

Alleen-lezen geheugen

Read-Only Memory (ROM) bevat gegevens die in de fabriek permanent zijn weggeschreven. Net als andere vintage-microcomputers heeft de Commodore 65 een ROM-chip met de programmacode van het besturingssysteem: BASIC, de KERNAL, routines voor het benaderen van randapparaten en voor het uitvoeren van gangbare berekeningen. ROM-gegevens kunnen niet rechtstreeks door een programma worden overschreven en blijven in de computer staan wanneer de computer wordt uitgeschakeld.

De MEGA65 bevat geen traditionele ROM-chip. In plaats daarvan laadt de MEGA65-Hypervisor (het besturingssysteem dat de voorzieningen van de MEGA65 beheert) de programmagegevens van de "ROM" uit een bestand op de SD-kaart naar een gebied in RAM. De Hypervisor beveiligt dat RAM standaard als "alleen lezen", waarmee de ROM-chip van een Commodore 65 wordt nagebootst. U kunt het ROM-bestand op de SD-kaart vervangen om te upgraden naar nieuwere versies van de MEGA65-ROM.

Probeer dit. Typ deze commando's in en probeer te voorspellen wat het tweede commando zal weergeven. Wijzig 63 in een ander getal en probeer het daarna opnieuw.

```
POKE $2100,63
PRINT PEEK($2100)
```

Het adres $2100 weigert een nieuwe waarde te aanvaarden. Een volledige uitleg waarom dit zo is, vergt verschillende begrippen die in dit hoofdstuk worden beschreven. Voorlopig volstaat de opmerking dat het geheugen op adres $2100 zo is ingesteld dat het zich als ROM gedraagt wanneer het op deze manier wordt benaderd.

I/O-registers

Sommige adressen verwijzen naar speciale geheugencellen die de hardware van de computer gebruikt. Deze staan bekend als *I/O-registers*. U kunt met de hardware communiceren door gegevens naar deze adressen te schrijven en eruit te lezen. Op deze manier geven BASIC-commando's grafische beelden weer, spelen ze geluiden en lezen ze joysticks uit: ze lezen waarden uit de I/O-registers en zetten er nieuwe in. ("I/O" verwijst naar de gegevens die als *invoer* naar en *uitvoer* vanuit deze apparaten gaan.)

Probeer dit commando uit. Vervang 7 door een ander getal tussen 0 en 31.

```
POKE $D020,7
```

De VIC-videochip gebruikt het adres $D020 om de kleur van de schermrand op te slaan. Het **BORDER**-commando stelt dit register in om hetzelfde effect te bereiken.

```
BORDER 7
```

Adressen

De voorbeelden tot nu toe gebruikten adressen die uit vier hexadecimale cijfers bestaan, zoals $1800. Elk hexadecimaal cijfer stelt vier bits voor (één nibble), dus een hexadecimaal getal van vier cijfers stelt een 16-bits adres voor. Een dergelijk adres ligt in het bereik $0000 - $FFFF, met 65536 mogelijke adressen. De CPU van de Commodore 64 gebruikt 16-bits adressen om maximaal 64 kilobyte (64KB) geheugen te benaderen. Met andere woorden: de Commodore 64 heeft een *adresruimte* van 16 bits, in het bereik $0000 - $FFFF.

De MEGA65 heeft een adresruimte van 28 bits, in het bereik 000.0000 - FFF.FFFF. Zulke adressen kunnen als 32-bits waarde worden opgeslagen, waarbij de hoogste nibble op 0 wordt gezet. In hexadecimale notatie zou dat een getal van acht cijfers zijn waarvan het meest linkse cijfer 0 is. (Dit cijfer wordt in de notatie doorgaans weggelaten.)

Het BASIC-commando **POKE** en de functie **PEEK()** kunnen 28-bits adressen gebruiken. Bijvoorbeeld, om eerst een waarde in het RAM op adres 800.0000 op te slaan en deze daarna weer op te halen:

```
POKE $8000000,63
PRINT PEEK($8000000)
```

Dit voorbeeld werkt op de MEGA65 en de DevKit. Het Nexys-bord heeft op dat adres geen RAM en negeert pogingen om er gegevens naartoe te schrijven.

16-bits adresvertaling

Het geheugensysteem van de MEGA65 maakt het programma's mogelijk om met 16-bits adressen naar geheugen te verwijzen. Om het volledige 28-bits adres te bepalen, vertaalt het geheugensysteem het adres volgens regels die het programma zelf kan instellen. Deze vertaling staat ook bekend als *hermapping* of, in sommige contexten, als *banking*.

16-bits adresvertaling biedt verschillende voordelen. De 45GS02-CPU verwerkt 16-bits adressen sneller dan 32-bits adressen. 16-bits adressen nemen bij het opslaan minder ruimte in het geheugen in. De meeste taken van een programma werken binnen één enkel 64KB-gebied van het geheugen, dus het is vaak handig voor een programma om de geheugenconfiguratie aan het begin van een taak in te stellen en dat gebied vervolgens met 16-bits adressen te benaderen.

Adresvertaling is een veelvoorkomende voorziening in de meeste microcomputers. De Commodore 64 gebruikt geheugenbanking om programma's met uitsluitend 16-bits adressen 64KB RAM *en* daarnaast ROM en I/O-registers te laten benaderen. De Commodore 65 was zo ontworpen dat het geheugensysteem via 16-bits adresvertaling op de geheugenindeling van de Commodore 64 kon worden ingesteld en bovendien banking in C64-stijl van ROM en I/O-registers ondersteunde.

De MEGA65 biedt meerdere mechanismen voor adresvertaling. De configuratie van het geheugensysteem is één van de complexere onderwerpen binnen de MEGA65-programmering. Dit hoofdstuk behandelt dit onderwerp in detail.

Banken en pagina's

De adressen van de MEGA65 zijn onder te verdelen in gebieden van 64KB, bekend als *banken*. Wanneer een adres in hexadecimale notatie is gegeven, is de bijbehorende bank gemakkelijk te bepalen: de meest linkse cijfers van het adres vormen het banknummer en de vier meest rechtse cijfers wijzen een plaats binnen die bank aan.



Bijvoorbeeld: het adres 3.FDA8 ($003FDA8) is een adres in bank $3. Het adres 0.1800 ($0001800) is een adres in bank $0. Het adres 880.12AB ($88012AB) ligt in bank $880.[2]

Het is belangrijk te onthouden dat, wanneer u een 16-bits adres ziet zoals $2100, de bank waarnaar dit werkelijk verwijst niet noodzakelijk bank 0 is. Het volledige adres hangt af van de configuratie van het geheugensysteem en van de manier waarop het geheugen wordt benaderd.

Een bank is verder onder te verdelen in gebieden van 256 bytes, bekend als *pagina's*. Zoals bij banken is ook de pagina gemakkelijk uit een hexadecimaal adres af te leiden: de pagina wordt gevormd door het vierde en derde cijfer van rechts. Pagina $1C bestaat bijvoorbeeld uit $1C00 - $1CFF. Het adres $1CB2 ligt in pagina $1C.

Hoe adressen in het geheugen worden opgeslagen

Computers verwerken geheugenadressen vaak als gegevens, bijvoorbeeld door adressen op te slaan in RAM of in registers. Een 16-bits adres wordt opgeslagen als twee bytes en een 28-bits adres als vier bytes (waarbij de meest linkse nibble op nul wordt gezet).

Om de bytes van een adres te bepalen, deelt u de hexadecimale cijfers in paren, beginnend bij het *meest rechtse* paar. Elk paar cijfers vormt een bytewaarde. Het adres $853FDA8 kan bijvoorbeeld worden weergegeven door de bytes $A8, $FD, $53 en $08.

[2] In enkele gevallen is dit beter te begrijpen als "megabyte $88, bank $0, offset $12AB".

Deze volgorde van de bytes van rechts naar links staat bekend als *little-endian-volgorde*. De MEGA65 slaat waarden van meerdere bytes volgens conventie op in little-endian-volgorde. De 45GS02-CPU, de I/O-registers en andere routines verwachten dat adressen op deze manier zijn opgeslagen.[3]

Om bijvoorbeeld het 28-bits adres 853.FDA8 op te slaan, heeft u vier bytes geheugen nodig. Om dit op te slaan op geheugenplaatsen 0.1800 - 0.1803, begint u met het opslaan van de meest rechtse byte ($A8) op plaats 0.1800. De rest gaat als volgt:



Wanneer een 28-bits adres als vier bytes wordt opgeslagen, blijven de bovenste vier bits ongebruikt. In enkele gevallen gebeurt het dat een 28-bits adres zodanig wordt opgeslagen dat die bits iets anders bevatten. U moet er bij het schrijven van zulke adressen zorg voor dragen dat de bovenste bits de juiste waarden behouden. Dat gebeurt doorgaans met booleaanse logica, waarbij de bovenste bits worden "gemaskeerd".

Het volgende BASIC-voorbeeld stelt de variabele B in op $A9 en werkt die daarna zodanig bij dat de bovenste nibble behouden blijft en de onderste nibble op $3 wordt gezet. Probeer, voordat u het uitvoert, te raden wat het zal tonen.

```
10 B = $A9
20 B = B AND $F0 OR $03
30 PRINT HEX$(B)
```

De Commodore 65 gebruikt een 20-bits adresruimte en gebruikt daarom soms slechts vijf nibbles (vijf hexadecimale cijfers) om een adres te coderen. In enkele gevallen breidt de MEGA65 dit uit naar 28 bits, door naast een "bank-nibble" en twee bytes voor het onderste deel van het adres nog een aparte "megabyte-byte" op te slaan. Later in dit hoofdstuk wordt een voorbeeld hiervan uitgelegd.

BASIC 65 bevat commando's om 16-bits waarden als bytes te lezen en te schrijven, zonder dat u zelf expliciet naar de little-endian-volgorde hoeft te converteren. **WPOKE addr,val** stelt **addr** in op de lage byte van **val** en **addr+1** op de hoge byte van **val**. Evenzo leest **WPEEK(addr)** de bytes op **addr** en **addr+1** en levert als resultaat het 16-bits getal op.

```
WPOKE $1900,$FFD2
```

BASIC heeft geen specifieke commando's voor het lezen of schrijven van 32-bits waarden, maar u kunt de 16-bits commando's gebruiken om telkens twee bytes tegelijk te verwerken. Let erop dat u voor elke helft de little-endian-volgorde aanhoudt:

[3] Er bestaan computers die de tegenovergestelde volgordeconventie hanteren, bekend als *big-endian* bytevolgorde. De meeste microcomputers gebruiken de little-endian-conventie.

```
WPOKE $1802, $0853
WPOKE $1800, $FDA8
```

DE 28-BITS ADRESRUIMTE

De 28-bits adressen vormen een *adresruimte* in het bereik 000.0000 - FFF.FFFF. Slechts een deel van de adressen is toegewezen aan geheugencellen. Andere adressen zijn toegewezen aan I/O-registers en andere bijzondere apparaten. De overige adressen zijn niet toegewezen; die zijn gereserveerd voor toekomstig gebruik door uitbreidingshardware en toekomstige versies van de computer.

De MEGA65 en de DevKit hebben drie grote gebieden met toegewezen adresruimte:

- **chip-RAM**, 384KB, in het bereik 000.0000 - 005.FFFF
- **zolder-RAM**, 8MB, in het bereik 800.0000 - 87F.FFFF
- **I/O-registers en speciaal RAM**, in het bereik F00.0000 - FFF.FFFF

De overige gebieden van de adresruimte zijn gereserveerd voor toekomstige uitbreidingen en toekomstige modellen. Nexys-borden hebben geen zolder-RAM. Zie Hoofdstuk/Bijlage J op bladzijde J-3 voor details over de gereserveerde gebieden.

Chip-RAM is de belangrijkste werkruimte van de MEGA65. Het bevat programmacode, de ROM-code van de MEGA65 en ruimte voor variabelen en andere gegevens. Het heet "chip-RAM" omdat de CPU het met volledige snelheid kan benaderen. BASIC- en KERNAL-functies maken gebruik van delen van het chip-RAM en ook uw BASIC- en machinetaalprogramma's zullen er uitgebreid gebruik van maken.

Zolder-RAM biedt extra geheugen voor algemeen gebruik, met enkele beperkingen. Doorgaans gebruikt een programma het zolder-RAM via de Direct Memory Access (DMA)-voorziening van de MEGA65 om gegevens tussen het zolder-RAM en het chip-RAM te kopiëren (later beschreven). Het zolder-RAM kan niet rechtstreeks door de VIC-chip worden gebruikt voor grafische gegevens, evenmin kan het rechtstreeks worden gebruikt voor het afspelen van audiosamples ("audio DMA"). De CPU kan code die in het zolder-RAM staat wel uitvoeren, maar die code draait langzamer dan code in het chip-RAM. In het algemeen is het benaderen van zolder-RAM ongeveer tien keer zo langzaam als het benaderen van chip-RAM.

OPMERKING: Het Freezer-hulpprogramma van de MEGA65 slaat het zolder-RAM niet op en herstelt het ook niet. In theorie kan een gebruiker een programma starten, de Freezer activeren, een freeze-toestand opslaan en op een later tijdstip een nieuwe sessie starten vanuit die freeze-toestand, met een gewist zolder-RAM als gevolg. In de praktijk zou het onredelijk zijn om programma's zo te schrijven dat zij ervan uitgaan dat het zolder-RAM elk moment kan verdwijnen. De beste tussenoplossing is gebruikers erop te wijzen wanneer een programma het zolder-RAM gebruikt en het gebruik van freeze-toestanden onder bepaalde omstandigheden af te raden. (Het louter activeren van de Freezer verandert het zolder-RAM niet. Alleen het herstellen van een freeze-toestand nadat de computer is uit- en weer ingeschakeld, kan problemen veroorzaken.)

De bovenste I/O-registerruimte is de plaats waar de apparaatregisters en het geheugen voor de Hypervisor-functies permanent thuishoren. De meeste programma's

benaderen deze voorzieningen indirect, via de geheugenconfiguratievoorzieningen die later in dit hoofdstuk worden beschreven, met name het kleurengeheugen (colour-RAM) van de VIC, de character-ROM en de I/O-registers van de VIC en de SID. Zie Hoofdstuk/Bijlage J op bladzijde J-3 voor een gedetailleerde lijst van de bovenste I/O-gebieden.

DE GEHEUGENINDELING VAN HET CHIP-RAM

De volgende tabel vat samen hoe KERNAL en BASIC van de MEGA65 het chip-RAM gebruiken.

Start	Einde	Omschrijving
0.0000	0.0000	Gegevensrichting van de CPU-I/O-poort
0.0001	0.0001	CPU-I/O-poortgegevens
0.0002	0.15FF	KERNAL-variabelen en -gegevens
0.1600	0.1EFF	Vrij voor gebruik door programma's
0.1F00	0.1FFF	Basispagina voor de bitmapgrafiek van BASIC
0.2000	0.F6FF	BASIC: programmatekst
0.F700	0.FEFF	BASIC: scalare variabelen
0.FF00	0.FFFF	Gereserveerd
1.0000	1.1FFF	DOS-buffers en -variabelen
1.2000	1.F6FF	BASIC: arrays en strings
1.F700	1.F7FF	Gereserveerd
1.F800	1.FFFF	Venster op het kleurengeheugen
2.0000	2.FFFF	ROM
3.0000	3.FFFF	ROM
4.0000	4.FFFF	BASIC-bitmapgrafiek, hulpprogramma's
5.0000	5.FFFF	BASIC-bitmapgrafiek, hulpprogramma's

De ROM-banken zijn in gebieden ingedeeld. De meeste programma's hoeven deze adressen niet rechtstreeks te benaderen.

Start	Einde	Omschrijving
2.0000	2.3FFF	DOS
2.4000	2.8FFF	Gereserveerd
2.9000	2.9FFF	Tekenset A
2.A000	2.BFFF	C64 BASIC
2.C000	2.CFFF	Interface
2.D000	2.DFFF	C64-tekenset (C)
2.E000	2.FFFF	C64 BASIC en KERNAL
3.0000	3.1FFF	Monitor
3.2000	3.7FFF	C65 BASIC
3.8000	3.BFFF	C65-BASIC-grafische routines
3.C000	3.CFFF	Gereserveerd

Vervolgd ...

...Vervolgd

Start	Einde	Omschrijving
3.D000	3.DFFF	Tekenset B
3.E000	3.FFFF	C65 KERNAL

Hoe programma's het chip-RAM gebruiken

De twee meest voorkomende soorten MEGA65-programma's zijn programma's die geheel in BASIC zijn geschreven en programma's die geheel in machinetaal zijn geschreven (via assembleertaal of een gecompileerde taal zoals C of Rust).

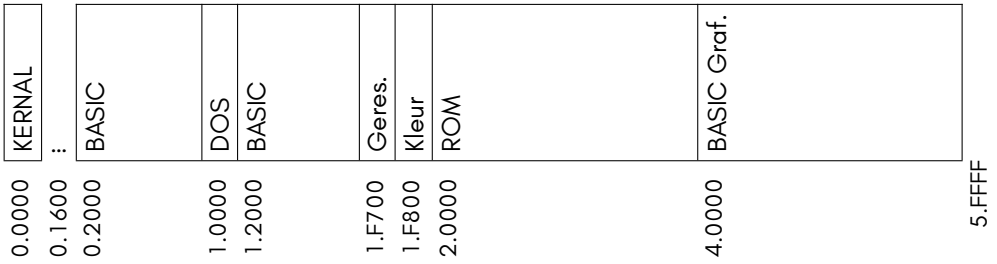
Een typisch BASIC 65-programma benadert het geheugen nooit rechtstreeks.[4] Het programma gebruikt BASIC-commando's en taalonderdelen om variabelen, arrays, strings, geluid, sprites en bitmapgrafiek te bewerken. Het BASIC-systeem gebruikt het volledige chip-RAM om deze voorzieningen te ondersteunen en beheert de geheugenconfiguratie om namens het programma registers te benaderen. Dit vereist dat de ROM, welke door de Hypervisor geladen en beschermd wordt, op zijn plaats blijft.

Een BASIC-programma dat het bitmapgrafieksysteem niet gebruikt, kan de geheugengebieden die aan die voorziening zijn toegewezen, voor andere doelen gebruiken. Dit omvat de gebied 0.1F00 - 0.1FFF en de volledige banken 4 en 5. Indien u van plan bent bitmapgrafiek te gebruiken, maar toch wat extra chip-RAM voor een ander doel nodig heeft, dan kunt u met het commando **MEM** blokken in bank 4 en 5 reserveren. Het grafieksysteem houdt de gebieden die met **MEM** zijn gereserveerd automatisch vrij, ten koste van minder geheugen voor meer schermen, een hogere resolutie of een grotere kleurdiepte. Voor een beschrijving van het commando **MEM** zie deelparagraaf B.

Sommige BASIC-hulpmiddelen in de directe modus, zoals **RENUMBER**, maken tijdelijk gebruik van geheugen in bank 4 en 5. Een programma dat zijn geheugen bij de start initialiseert, heeft daar geen last van. Interactieve taken bij de READY.-prompt die dat geheugen onaangetast nodig hebben, kunnen erdoor wel verstoord worden.

Een gangbare techniek om vanuit een BASIC-programma geheugen te benutten, is gebruikmaken van het ongebruikte geheugen voor "programmatekst" na het einde van de programmagegevens van BASIC in 0.2000 - 0.F6FF. Indien uw programma dit nodig heeft, houd er dan rekening mee dat sommige grafische commando's van BASIC 65 het geheugen direct na het einde van het programma als tijdelijke opslag gebruiken. Afhankelijk van de lengte van uw programma kunt u doorgaans veilig een hoog adresbereik kiezen voor gebruik door het programma, zoals 0.C000-0.F6FF.

[4] Commodore 64 BASIC-programmeurs zijn gewend om met **POKE** en **PEEK** het geheugen en de I/O-registers te benaderen voor de meeste functies, zoals beeldweergave, geluid en invoerapparaten. BASIC 65 biedt voor deze functies speciale commando's.



Een typisch programma in machinetaal gebruikt het BASIC-systeem helemaal niet en beheert in plaats daarvan zijn eigen geheugen. Het kan een klein BASIC-programmaatje gebruiken om het programma op te starten, maar keert nadat het programma is gestart nooit meer terug naar het systeem. Zo'n programma mag elke regio van het chip-RAM gebruiken die BASIC anders in beslag zou nemen.

Indien het programma de KERNAL aanroept, bijvoorbeeld voor schijffuncties, dan moet het de gebieden van het chip-RAM vermijden die voor gebruik door de KERNAL zijn gereserveerd. Met de rest kan het doen wat het wil.



Als het programma de KERNAL nooit hoeft aan te roepen, kan het de ROM volledig ontgrendelen en verwijderen en zo elke byte van de 384KB chip-RAM voor eigen doeleinden opeisen. Met een techniek die verderop wordt besproken, kunt u, als u niet alle 32KB van het kleurengeheugen van de MEGA65 nodig heeft, het "venster op het kleurengeheugen" op 1.F800 herbestemmen als gewoon chip-RAM.

De geheugenindeling als contract

Het MEGA65-project heeft de ROM-code overgenomen uit het onafgemaakte ontwerp van Commodore voor de C65 en heeft tijd en moeite gestoken in het afmaken van onvoltooide voorzieningen en het verhelpen van fouten. Dit project loopt nog door en het MEGA65-team verwacht in de toekomst nieuwe versies van de ROM uit te geven.

Het schrijven van programma's met de MEGA65-ROM verschilt daardoor wezenlijk van het schrijven van programma's met de ROM van de Commodore 64, welke in silicium is geëtst en niet langer wordt doorontwikkeld. Het is belangrijk om code te schrijven die zich houdt aan het gedocumenteerde gedrag dat in deze handleidingen

staat beschreven—en niet te vertrouwen op ongedocumenteerd gedrag dat in een toekomstige versie zou kunnen veranderen.

Het is een doel van het project om de geheugenindeling van het chip-RAM in toekomstige versies van de MEGA65-ROM consistent te houden met de tabel hierboven. Om de compatibiliteit met toekomstige versies te waarborgen, doen programma's er verstandig aan "gereserveerde" gebieden niet als vrije ruimte te behandelen. Indien een nieuwe modus of voorziening wordt ingevoerd die de geheugenindeling wijzigt, zal het programma die voorziening expliciet moeten aanvragen, zodat oudere programma's die geen gebruikmaken van de nieuwe voorziening op de oorspronkelijke geheugenindeling kunnen rekenen.

Binnen elk blok van de geheugenindeling is slechts van een deel van de adressen gegarandeerd dat het tussen ROM-versies gelijk blijft. Een machinetaalprogramma kan bijvoorbeeld KERNAL-routines aanroepen via adressen in een "jump table" (sprongtabel), een reeks vaste adressen in de ROM die gegarandeerd ook in toekomstige versies gelijk blijft.[5]

Behalve waar expliciet gedocumenteerd, is het niet de bedoeling dat programma's KERNAL-variabelen rechtstreeks lezen of wijzigen. Sommige geheugenlocaties, zoals de locatie van het schermgeheugen, kunnen uit registers worden afgeleid (en via registers worden aangepast). (Zie Hoofdstuk/Bijlage Q op bladzijde Q-5.)

Het gebied van chip-RAM in 0.1600 - 0.1EFF wordt gegarandeerd niet gebruikt door de KERNAL en BASIC van de MEGA65. Het gebied 0.1F00 - 0.1FFF is eveneens beschikbaar indien u het bitmapgrafieksysteem van BASIC niet gebruikt. In de standaardgeheugenconfiguratie is dit gebied als RAM toegankelijk, zodat het geschikt is voor het opslaan van machinetaal en gegevens die vanuit een BASIC-programma benaderd kunnen worden.

[5] De jump table wordt in dit boek nog niet beschreven. De C65 gebruikte dezelfde jump table als de C128, met enkele kleine wijzigingen en tot nu toe heeft de MEGA65-ROM deze niet uitgebreid. Indien een ROM-revisie deze tabel uitbreidt, blijven de bestaande vermeldingen onveranderd. Zie de MEGA65 Wiki voor een lijst: https://mega65.atlassian.net/

GEHEUGEN GEBRUIKEN VANUIT BASIC

BASIC 65 bevat verschillende commando's voor het benaderen en bewerken van geheugen en voor het aanroepen van machinetaalsubroutines:

- **POKE adres, waarde** : slaat een waarde op een adres weg
- **WPOKE adres, waarde** : slaat een 16-bits waarde weg beginnend bij een adres
- **PEEK(adres)** : leest een waarde van een adres
- **WPEEK(adres)** : leest een 16-bits waarde beginnend bij een adres
- **BLOAD bestandsnaam [,argumenten...]** : laadt gegevens van een bestand in het geheugen
- **BSAVE bestandsnaam, P begin TO P eind [,argumenten...]** : slaat een geheugengebied op in een bestand
- **SYS adres [,registers...]** : roept een machinetaalsubroutine aan die op een adres is opgeslagen
- **EDMA ...** : transformeert of kopieert grote geheugengebieden snel met behulp van hardwareversnelling

Ook de commando's **BOOT** en **WAIT** werken op een of andere manier op het geheugen. Zie de BASIC 65-commandoreferentie voor meer informatie over deze commando's.

Hermapping van adressen door BASIC

U is het wellicht opgevallen dat de adressen uit de BASIC-commando's **POKE** die aan het begin van dit hoofdstuk zijn beschreven, allemaal verwijzen naar gebieden van chip-RAM die in de geheugenindeling staan aangeduid als "Vrij voor gebruik door programma's" of "BASIC: programmatekst". Toch lijken ze zich anders te gedragen:

- `POKE $1800,127` gedraagt zich als RAM: het stelt het geheugen op het adres in en het geheugen kan worden gelezen met `PRINT PEEK($1800)`.
- `POKE $2010,63` gedraagt zich als ROM: een poging om het geheugen in te stellen blijft zonder gevolg en het lezen van het geheugen met `PRINT PEEK($2010)` geeft de oorspronkelijke ROM-waarde terug.
- `POKE $D020,7` gedraagt zich als een I/O-register: het verandert onmiddellijk de kleur van de rand. Deze waarde kan weliswaar worden gelezen met `PRINT PEEK($D020)`, maar als geheugen is die niet bruikbaar, omdat de randkleur met de waarde meeverandert.

De commando's en functies van BASIC 65 behandelen adressen in het bereik $0000 - $FFFF volgens bijzondere hermappingregels, ook wel *banking* genoemd. In de standaardmodus hermapt BASIC 65 deze adressen als volgt:

16-bits adresblok	Gemapt adresblok	Type
$0000 - $1FFF	0.0000 - 0.1FFF	chip-RAM
$2000 - $3FFF	3.2000 - 3.3FFF	ROM
$4000 - $5FFF	3.4000 - 3.5FFF	ROM
$6000 - $7FFF	3.6000 - 3.7FFF	ROM
$8000 - $9FFF	3.8000 - 3.9FFF	ROM
$A000 - $BFFF	3.A000 - 3.BFFF	ROM
$C000 - $CFFF	2.C000 - 2.CFFF	ROM
$D000 - $DFFF	FFD.3000 - FFD.3FFF	MEGA65-I/O
$E000 - $FFFF	3.E000 - 3.FFFF	ROM

Benadert u bijvoorbeeld de I/O-registers via $D000 - $DFFF, dan benadert u in feite de I/O-registers in het bovenste gebied van de adresruimte. Deze twee BASIC-commando's zijn gelijkwaardig:

```
POKE $D020,7

POKE $FFD2020,7
```

Het BASIC-systeem zelf gebruikt chip-RAM rechtstreeks, volgens de geheugenindeling van het chip-RAM. BASIC slaat programmatekst bijvoorbeeld op vanaf 0.2000 in het chip-RAM, hoewel de BASIC-functie `PEEK($2000)` in de standaardmappingmodus vanaf 3.2000 leest.

De hermapping van adressen door BASIC 65 geldt alleen voor adressen in het bereik $0000 - $FFFF. Gebruikt u een adres van $10000 of hoger met **POKE** of **PEEK()**, dan benadert het commando dat chip-RAM-adres rechtstreeks. **SYS** ondersteunt ook lange adressen, zij het met beperkingen (later besproken).

U kunt de adreshermapping van BASIC sturen met het commando **BANK**. Dit commando accepteert een banknummer als argument, in het bereik 0 - 5. Adressen in het bereik $0000 - $FFFF die u daarna gebruikt, worden geïnterpreteerd als offsets binnen die bank. Om bijvoorbeeld de waarde 63 naar adres 0.2010 in het chip-RAM te schrijven:

```
BANK 0
POKE $2010,63
```

Het **BANK**-mechanisme kan alleen met de banken 0 - 5 worden gebruikt. Het zolder-RAM en de bovenste I/O-registers zijn ermee niet te bereiken.

Om de adreshermapping van BASIC weer in te schakelen, voert u **BANK** uit met het argument 128. Dit getal verwijst niet naar een bank in het geheugen. In plaats daarvan houdt het in dat het commando **BANK** de adresvertaling weer volgens de tabel hierboven activeert.

```
BANK 128
POKE $D020,9
```

BANK beïnvloedt elk BASIC-commando en elke BASIC-functie die een adres als parameter accepteert. Dit omvat **PEEK**, **POKE**, **WAIT**, **BOOT**, **BSAVE** en **BLOAD**. **SYS** gaat ook op een bijzondere manier met **BANK** om.

BANK en SYS

Het commando **SYS** voert een machinetaalsubroutine op een opgegeven adres uit. Het behandelt het adres anders dan andere BASIC-commando's en kent enkele beperkingen. Omdat **SYS** de besturing overdraagt aan een eigen routine, moet het extra voorzorgsmaatregelen nemen om te voorkomen dat het systeem de toegang verliest tot het geheugen dat BASIC nodig heeft.

Concreet gebruikt **SYS** de **BANK**-instelling alleen voor adressen in het bereik $2000 – $7FFF. Andere adressen gaan altijd naar het RAM van bank 0, naar KERNAL ROM of naar de I/O-registers, als volgt:

Adresbereik	**SYS-bestemming**
$0000 – $1FFF	RAM in bank 0
$2000 – $7FFF	Adres volgens de **BANK**-instelling
$8000 – $BFFF	RAM in bank 0
$C000 – $CFFF	Character ROM, of bank 0 (ROMC-bit van $D030)
$D000 – $DFFF	I/O-registers, of bank 0 (bankingregister $01 in C64-stijl)
$E000 – $FFFF	KERNAL ROM in bank 3

SYS negeert bit 7 van de **BANK**-instelling. `BANK 128` is gelijkwaardig aan `BANK 0` in het geval van **SYS**. Dit verschilt van **PEEK**, **POKE** en dergelijke, welke op $2000 – $BFFF met `BANK 128` ROM-code te zien krijgen.

```
BANK 128
SYS $1800      : rem Roept de subroutine op 0.1800 aan
SYS $7000      : rem Roept de subroutine op 0.7000 aan
SYS $FFD2,65   : rem Roept de KERNAL-ROM-routine aan
               : rem (met een argument)

BANK 1
SYS $1800      : rem Roept de subroutine in bank 0 op 0.1800 aan
SYS $7000      : rem Roept de subroutine in bank 1 op 1.7000 aan
SYS $FFD2,65   : rem Roept de KERNAL-ROM-routine op 3.FFD2 aan
               : rem (met een argument)
```

SYS kan ook een adres groter dan $FFFF aanvaarden dat naar een locatie in de banken 1 – 5 verwijst. Deze adressen hebben echter dezelfde beperkingen als bij het gebruik van **BANK**. Alleen de offsets $2000 – $7FFF binnen de bank verwijzen werkelijk naar het geheugen van die bank, zoals adres $47000 in bank 4. Voor andere offsets gedraagt het commando **SYS** zich volgens de tabel (RAM in bank 0, ROM of I/O-registers).

Deze beperking levert het tegenintuïtieve gedrag op dat een lang adres voor **SYS** niet noodzakelijk naar het geheugen op dat adres verwijst en dat meerdere adressen naar dezelfde locatie in het geheugen kunnen verwijzen. `SYS $11800` en `SYS $41800` gedragen zich beide als `SYS $1800`, hetwelk een subroutine op adres 0.1800 in bank 0 aanroept.

```
SYS $11800     : rem Roept de subroutine in bank 0 op 0.1800 aan
SYS $41800     : rem Roept de subroutine in bank 0 op 0.1800 aan
SYS $51800     : rem Roept de subroutine in bank 0 op 0.1800 aan

SYS $17000     : rem Roept de subroutine in bank 1 op 1.7000 aan
SYS $47000     : rem Roept de subroutine in bank 4 op 4.7000 aan
SYS $57000     : rem Roept de subroutine in bank 5 op 5.7000 aan

SYS $1FFD2,65 : rem Roept de KERNAL-ROM-routine op 3.FFD2 aan
               : rem (met een argument)
```

Het commando **SYS** kan niet elk adres benaderen. ROM en I/O-registers verbergen de adresoffsets $C000 – $FFFF in elke bank. De adresoffsets $0000 – $1FFF en $8000 – $BFFF bereiken uitsluitend bank 0 en blijven in de banken 1 – 5 verborgen.

SYS is beperkt tot de banken 0 – 5. Het kan het zolder-RAM en de hogere geheugenadressen niet benaderen, zelfs niet met lange adressen.

Ondanks deze beperkingen is **SYS** tamelijk krachtig. Het komt vaak voor dat een machinetaalprogramma begint met een kort BASIC-programmaatje, zodat de gebruiker na het laden **RUN** kan typen. Dit programmaatje roept **SYS** aan om de machinetaal te starten die in het geheugen staat na het einde van de BASIC-tekst.

```
10 SYS $200F
```

In vroege versies van de MEGA65-ROM moest u **BANK** eerst op 0 instellen voordat u **SYS** op deze manier aanriep. Dat blijft sowieso een goede gewoonte, zodat het programma kan starten, zelfs als de gebruiker de **BANK**-instelling heeft gewijzigd:

```
10 BANK 0:SYS $2014
```

U kunt **SYS** ook gebruiken om een BASIC-programma uit te breiden met subroutines in machinetaal. Een geschikte locatie voor deze subroutines is het bereik $1600 – $1EFF van bank 0. **SYS** kan deze locatie benaderen, ongeacht de huidige **BANK**-instelling.

SYS TO

Als aanvulling op de beperkingen van de geheugentoegang van het **SYS**-commando kent BASIC 65 een variant, **SYS TO**, met extra mogelijkheden. Deze variant verschilt op de volgende punten van **SYS**:

- Zowel **SYS** als **SYS TO** verwachten een adres als eerste argument. **SYS** aanvaardt daaropvolgende positionele argumenten voor het instellen van registers. **SYS TO** aanvaardt benoemde argumenten voor het instellen van registers en voor het benaderen van andere voorzieningen.

- Een programma kan **SYS TO** gebruiken om een pagina-offset toe te passen op adressen $2000 tot $7FFF, zodat een subroutineadres in dat bereik kan verwijzen naar een fysiek adres buiten dat bereik, in een bank.
- Een programma kan **SYS TO** gebruiken om routines aan te roepen terwijl BASIC actief blijft, bijvoorbeeld om **PLAY**-muziek of **MOVSPR**-spritebeweging soepel te laten doorlopen terwijl de machinetaal wordt uitgevoerd.

Om een pagina-offset in te stellen, geeft u het argument 0 op aan **SYS TO**. De waarde bestaat uit drie nibbles, die met 256 worden vermenigvuldigd (twee nibbles naar links geschoven) om de offset te bepalen. Een offset van $520 heeft bijvoorbeeld tot gevolg dat bij elk adres in het bereik $2000 tot $7FFF een offset van $52000 wordt opgeteld.

```
REM : MAP $2000-$7FFF NAAR $54000-$59FFF,
REM : DAARNA AANROEP $6100 + $52000 = $58100

SYS TO $6100,O($520)
```

Om een routine aan te roepen terwijl BASIC actief blijft, geeft u het argument I op met de waarde 1. U kunt deze modus gebruiken of een pagina-offset instellen, maar niet beide. Indien u deze modus inschakelt, wisselt de KERNAL-IRQ-handler tussen de MAP van uw programma en de MAP van KERNAL om de BASIC-functies op basis van IRQ's uit te voeren.

```
SYS TO $1600,I1
```

GEHEUGEN GEBRUIKEN VANUIT MACHINETAAL

Machinetaalprogramma's benaderen geheugen en registers, verrichten berekeningen en beslissen welke code daarna moet worden uitgevoerd. Meer doen ze niet! Een programma verricht invoer/uitvoerbewerkingen, zoals het weergeven van een afbeelding, het afspelen van een geluid of het uitlezen van een joystick, louter door met geheugen en registers te werken.

Elke 45GS02-machinetaalinstructie die met geheugen werkt, ondersteunt één of meer *adresseermethoden*, waarin staat hoe de instructie het geheugen benadert. Eén van deze adresseermethoden werkt met volledige 28-bits adressen en leest het volledige adres uit vier opeenvolgende bytes in het geheugen, in little-endian-volgorde. Deze methode is bruikbaar, maar in veel gevallen niet de efficiëntste manier om het geheugen te benaderen: de CPU moet tijd besteden aan het lezen van het 28-bits adres en aan het vinden van de geheugenplaats voor de bewerking.

Adresseermethoden die met 16-bits adressen werken, zijn sneller en ruimtezuiniger dan 28-bits indirecte adressering. Bovendien gebruikt de CPU 16-bits adressen om te verwijzen naar de plaatsen van machinetaalinstructies: de CPU slaat het adres van de volgende instructie (de *programmateller*) op als een 16-bits waarde en de instructies `jmp` en `jsr` gebruiken 16-bits adressen.

Om het mogelijk te maken elke geheugenplaats met een 16-bits adres te benaderen, houdt de 45GS02-CPU een *geheugenindeling* bij, welke 16-bits adressen aan 28-bits adressen koppelt. Uw programma kan de geheugenindeling wijzigen met behulp van een register met een speciale functie, bekend als *het MAP-register*. Het instellen van het MAP-register vereist het uitvoeren van een reeks speciale CPU-instructies.

Bij de MEGA65 beïnvloeden drie andere mechanismen eveneens de adresvertaling van 16-bits geheugenadressen:

- ROM- en I/O-banking van de Commodore 64, bestuurd door de CPU-registers op de adressen $0000 en $0001
- ROM-banking van de Commodore 65, met behulp van een register op adres $D030
- Cartridgegeheugen: indien er een cartridge in de uitbreidingspoort is geplaatst, maken aanvullende regels de datalijnen van de cartridge beschikbaar op specifieke 16-bits adresbanken

Machinetaalprogramma's die voor de MEGA65 zijn geschreven, kunnen voor de geheugentoegang volledig op het MAP-register vertrouwen en kunnen de geheugenconfiguratieregisters van de C64 en C65 op hun standaardinstellingen laten. Wanneer de betrokken gebieden van 16-bits adressen voor andere doeleinden nodig zijn, heeft MAP voorrang op de andere mechanismen.

Deze sectie beschrijft de geheugentoegang met 45GS02-adresseermethoden en het MAP-register. De overige mechanismen voor het mappen van geheugen komen verderop in dit hoofdstuk kort aan de orde. Zie Hoofdstuk/Bijlage L op bladzijde L-4 voor een volledige beschrijving van de adresseermethoden en Hoofdstuk/Bijlage K op bladzijde K-3 voor een gedetailleerde uitleg van de bankingmechanismen.

Het MAP-register

Het MAP-register koppelt 16-bits adressen aan 28-bits adressen met behulp van een *offset*, een waarde die bij het 16-bits adres wordt opgeteld om het uiteindelijke adres te berekenen.

Stel bijvoorbeeld dat uw programma met een 16-bits adresseermethode geheugen op adres $4855F moet benaderen. Een mogelijke geheugenindeling is dan om aan het adresgebied $2000 - $3FFF een offset van $45200 toe te wijzen. Met deze configuratie benadert het 16-bits adres $335F het werkelijke adres $4855F in bank 4.

$335F	+	$45200	=	$4855F
16-bits adres		offset		werkelijk adres

Het MAP-register hermapt telkens blokken van 8KB ($2000) uit de 16-bits adresruimte. Het MAP-register kan twee offsets opslaan: één voor blokken in het bereik $0000 - $7FFF ("MAPLO") en één voor blokken in het bereik $8000 - $FFFF ("MAPHI"). Ook slaat het voor elk blok van 8KB een selectievlag op, welke aangeeft of de offset op dat blok moet worden toegepast.

Elk adresblok van 8KB is ofwel "ongemapt" of gebruikt de MAP-offset voor zijn helft (MAPLO of MAPHI). Alle gemapte adresblokken in dezelfde helft moeten dezelfde offset delen.

MAPLO:	$6000 - $7FFF 0	$4000 - $5FFF 0	$2000 - $3FFF **1**	$0000 - $1FFF 0	= $2
MAPHI:	$E000 - $FFFF **1**	$C000 - $DFFF 0	$A000 - $BFFF **1**	$8000 - $9FFF **1**	= $B

De selectievlaggen en de offset voor MAPLO en MAPHI worden tezamen gecodeerd als vier bytes: twee voor MAPLO en twee voor MAPHI. De eerste nibble bevat de selectievlaggen voor de vier blokken van het gebied. Let op de volgorde van de vlagbits: de hogere bits horen bij de hogere adresblokken.

Om bijvoorbeeld uitsluitend het blok $2000 - $3FFF te selecteren, stelt u de MAPLO-selectievlag in op het binaire getal "0 0 1 0", dat gelijk is aan het hexadecimale cijfer $2.

De overige drie nibbles vormen de bovenste drie hexadecimale cijfers van de offset. Om de offset $45200 voor de geselecteerde MAPLO-blokken te gebruiken, stelt u de overige bits van MAPLO in op $452. Voor dit voorbeeld is de volledige MAPLO-waarde $2452.

MAPLO		MAPHI	
X	A	Z	Y
$$s_{lo}$$ $$o_1$$	$$o_2$$ $$o_3$$	$$s_{hi}$$ $$o_1$$	$$o_2$$ $$o_3$$
$24	$52	$B3	$00

De offset moet een veelvoud van 256 bytes ($100) zijn. De 4510-CPU uit de Commodore 65, waarop de 45GS02 van de MEGA65 is gebaseerd, ondersteunt offsets tot $FFF00. Hierdoor kan elk 8KB-blok overal in het chip-RAM op een grens van 256 bytes worden hermapd, inclusief mogelijke toekomstige RAM-uitbreidingen tot 1MB. (De 45GS02 biedt ook een manier om grotere offsets in te stellen; die wordt verderop besproken.)

Om het MAP-register in te stellen, laadt u de gecodeerde bytes voor MAPLO en MAPHI in de vier CPU-registers X, A, Z en Y en roept u daarna de instructies **MAP** en **EOM** ("end of MAP") aan. Let op de volgorde waarin de registers hun waarden toegewezen krijgen. Om MAPLO in te stellen, laadt u X met de selectievlaggen en de bovenste nibble van de offset en laadt u A met de onderste nibbles van de offset. Om MAPHI in te stellen, laadt u Z en Y overeenkomstig.

```
LDA #$52    ; MAPLO = selecteer $2 offset $45200
LDX #$24
LDY #$00    ; MAPHI = selecteer $B offset $30000
LDZ #$B3
MAP
EOM
```

Sommige assemblers kennen de EOM-instructie niet. U kunt in plaats van EOM de NOP-instructie gebruiken, welke dezelfde codering gebruikt. Assemblers die 65CE02-instructies ondersteunen, gebruiken voor MAP de AUG-instructie.

De MAP-instelling blijft actief totdat deze wordt gewijzigd. Houd er rekening mee dat KERNAL-routines afhankelijk zijn van de standaard-MAP-instelling. Herstel daarom de oorspronkelijke geheugenindeling voordat u de KERNAL aanroept.

Als het optellen van de offset bij het 16-bits adres een waarde oplevert die groter is dan het grootste 20-bits adres ($FFFFF), kapt de optelling de overloopbit af, waardoor het 20-bits adres "omslaat". Dit maakt het mogelijk om een negatieve offset toe te passen, opgegeven als tweecomplement. Om bijvoorbeeld het 16-bits adres $8000 te mappen op het 20-bits adres $02000, stelt u de offset in op $FA000 (negatief $06000).

Het MAP-register bekijken

U kunt de waarde van het MAP-register bekijken met de Matrix Mode-debugger, een seriële console die de Hypervisor van de MEGA65 aanbiedt. Om de Matrix Mode-debugger op de MEGA65 te openen, houdt u  ingedrukt en drukt u op . (Druk nogmaals op  +  om af te sluiten.) Met de juiste hardware kunt u deze console ook bereiken via een seriële UART- of JTAG-verbinding. Gebruik in Matrix Mode

het commando **R**, of druk gewoon op RETURN, om alle registers te bekijken, waaronder MAPHI en MAPLO.[6]

Om dit in werking te zien, doet u het volgende:

1. Start de MEGA65-monitor met het commando **MONITOR**.
2. Assembleer het volgende machinetaalprogramma op adres $1800 door de eerste regel als volgt in te typen: `A1800 LDA #$52` De monitor berekent voor elke regel de daaropvolgende adressen. Druk na de laatste regel op RETURN zonder er een instructie achter te typen.

```
LDA #$52
LDX #$24
LDY #$00
LDZ #$B3
MAP
EOM
JMP $180A
```

3. Roep deze routine aan met de "jump"-commando van de monitor: `J1800` De MEGA65 komt in een oneindige lus terecht en keert niet uit de aanroep terug, zodat deze schijnbaar vastloopt.
4. Open de Matrix Mode-debugger: houd ingedrukt en druk op TAB.
5. Druk op RETURN om een nieuwe set registers te tonen. Let op de waarden van MAPH en MAPL.

OPMERKING: De MAP-instructie wijzigt de waarde van het MAP-register onmiddellijk. Zorg ervoor dat u de geheugenindeling niet wijzigt voor het geheugen waarin de code staat die de geheugenindeling wijzigt! Als de geheugenindeling verandert voor het adres van de code zelf, bereikt de uitvoering de EOM-instructie nooit.

MAP en interrupts

Wanneer een CPU-interrupt optreedt, onderbreekt de CPU de uitvoering van het programma en roept daarna een interrupt-handlerroutine aan. De adressen van de interrupt-handlerroutines (de interrupt-*vectoren*) staan op de 16-bits adressen $FFFA - $FFFF en zijn zelf eveneens 16-bits adressen.

De geheugenindeling beïnvloedt zowel de plaats van de vectoren als de adressen van de handlers. Om uw code de kans te geven de geheugenindeling te herstellen zonder onderbroken te worden, schakelt de MAP-instructie interrupts uit (vergelijkbaar met SEI) en schakelt de EOM-instructie interrupts weer in (vergelijkbaar met CLI). Dit geldt

[6] Zie Hoofdstuk/Bijlage O op bladzijde O-16 voor meer informatie over de Matrix Mode-debugger.

ook wanneer interrupts al zijn uitgeschakeld op het moment dat MAP wordt aangeroepen: EOM schakelt interrupts weer in. Indien nodig kan uw programma tussen MAP en EOM nog extra instructies uitvoeren.

De standaardgeheugenindeling houdt de KERNAL-ROM van de MEGA65 gemapt op $E000 - $FFFF. Dit omvat de KERNAL-interrupt-handlerroutines en hun vectoradressen. Als uw programma de KERNAL-interrupt-handlerroutines actief houdt, moet het ervoor zorgen dat $E000 - $FFFF gemapt is op 3.E000 - 3.FFFF (met een MAPHI-offset van $30000), zolang interrupts zijn ingeschakeld.

28-bits adressen in machinetaal gebruiken

De 45GS02-CPU bevat een adresseermethode waarmee 28-bits adressen rechtstreeks benaderd kunnen worden, zonder 16-bits adresvertaling. Hoewel deze modus technisch gezien meer CPU-cycli vergt om uit te voeren, weegt het gemak die kosten vaak op bij het benaderen van losse adressen in het bovenste geheugen of van geheugen via pointers, in vergelijking met het gebruik van het MAP-register. Deze modus omzeilt bovendien banking en hermapping, waardoor een programma een MAP-registerwaarde kan behouden die mogelijk door andere code wordt gebruikt.

De Base-Page Quad Indirect Z-Indexed-adresseermethode benadert een 28-bits adres dat als vier bytes op de Base-Page is opgeslagen. Deze methode is een uitbreiding van de Base-Page Indirect Z-Indexed-modus, welke een 16-bits adres benadert dat als twee bytes is opgeslagen.

In een assembler die 45GS02-uitbreidingen ondersteunt, zoals de ACME-assembler, gebruikt u voor de Quad Indirect-modus vierkante haken rond het eerste Base-Page-adres waarop het adres is opgeslagen:

```
; Selecteer Base-Page $1600.
; (Dit behoudt de compatibiliteit met de KERNAL, die
; Base-Page $0000 reserveert.)
LDA #$16
TAB

; Bewaar adres 810.4AB0 op Base-Page-adres $40-$43.
; (Vergeet niet de little-endian-volgorde te gebruiken.)
LDA #$B0
STA $40
LDA #$4A
STA $41
LDA #$10
STA $42
LDA #$08  ; (Het meest linkse cijfer is nul.)
STA $43

; Laad de accumulator met de byte op 810.4AB0 (met
; een Z-index van nul).
LDZ #$0
```

```
LDA [$40],Z
```

Deze modus wordt gecodeerd als een gewone Base-Page Indirect-modus, voorafgegaan door een NOP-instructie. Als uw assembler de notatie met vierkante haken niet ondersteunt, kunt u het op deze manier schrijven, met ronde haakjes in plaats van vierkante haken:

```
LDZ #$0
NOP
LDA ($40),Z
```

Zie Hoofdstuk/Bijlage K op bladzijde K-3 voor meer informatie over adresseermethoden.

Dit is in essentie de manier waarop het BANK-systeem van BASIC werkt. Indien BANK op 0 - 5 is ingesteld, of een programma een adres benadert dat groter is dan $FFFF, gebruiken commando's zoals POKE en PEEK de Base-Page Quad indirect Z-geïndexeerde adresseermethode, waarbij het volledige adres wordt weggeschreven naar een locatie die in de Base-Page van de KERNAL is gereserveerd. Dit omzeilt de adresvertaling zonder dat de geheugenconfiguratie gewijzigd hoeft te worden. (Het commando SYS doet iets complexers en daardoor kent het ongewone beperkingen.)

VIDEOGEHEUGEN VAN DE VIC-IV

De videocontroller van de VIC-IV leest geheugen om te bepalen wat er op het scherm moet worden weergegeven, zoals tekens, bitmapgrafiek, sprites en kleurgegevens. Een programma wijzigt de weergave door deze geheugengebieden bij te werken. De geheugenlocaties die voor deze doeleinden worden gebruikt, kan het programma met behulp van I/O-registers aanpassen.

De VIC-IV kan uitsluitend chip-RAM benaderen. Technisch gezien kan de VIC-IV de eerste 16MB van de adresruimte benaderen. De bestaande modellen van de MEGA65 bieden 384KB chip-RAM, met ruimte voor toekomstige uitbreiding.

Deze sectie beschrijft enkele VIC-IV-voorzieningen die met het geheugensysteem te maken hebben. De volgende beschrijvingen gelden niet voor de VIC-II-compatibiliteitsmodus. Zie Hoofdstuk/Bijlage Q op bladzijde Q-5 voor een volledige beschrijving van de VIC-IV.

Schermgeheugen vinden

De VIC-IV leest de inhoud van het scherm (tekens of pixels) uit een aaneengesloten gebied van het geheugen. Registers bepalen de beginlocatie van dit geheugen. De inhoud en de grootte van het schermgeheugen hangen af van de weergavemodus.

Het startadres van het schermgeheugen neemt 28 bits in beslag, hoewel er slechts 20 bits nodig zijn om een chip-RAM-adres op te slaan. Het 28-bits adres wordt verspreid over de vier bytes $D060 - $D063. De hoogste nibble van het adres staat in de lagere nibble van $D063, welke voor adressen tot 005.FFFF altijd nul is.

Om het schermgeheugen bijvoorbeeld te laten beginnen op 4.26F0 in bank 4 van het chip-RAM:

SCRNPTRLSB $D060	SCRNPTRMSB $D061	SCRNPTRBNK $D062	SCRNPTRMB $D063 (laagste 4 bits)
$F0	$26	$04	$0

```
LDA #$F0
STA $D060
LDA #$26
STA $D061
LDA #$04
STA $D062

; Voor de volledigheid: wis bits 0-3 van SCRNPTRMB. De
; MEGA65 heeft geen door de VIC benaderbaar geheugen boven
; 005.FFFF, dus dit is niet nodig indien de vorige
; waarde naar een geldig adres wees.
LDA $D063
AND #$F0    ; wis bits 0-3
STA $D063
```

Indien u hiermee experimenteert in de MEGA65-monitor, houd er dan rekening mee dat de BASIC-editor de nieuwe schermlocatie niet automatisch gebruikt nadat u deze heeft gewijzigd. U kunt blind typen om het geheugen in te stellen en zo zien dat de nieuwe locatie in gebruik is. Druk op RUN STOP + RESTORE om terug te zetten.

Indien uw programma naar het schermgeheugen schrijft zonder dit register te initialiseren, lees dan de locatie van het schermgeheugen uit het register. Ga niet uit van de oorspronkelijke waarde. De 80x50-weergavemodus, welke de gebruiker kan selecteren door op ESC 5 te drukken, gebruikt een andere locatie voor het schermgeheugen dan de standaard 80x25-weergavemodus.

Tekengegevens vinden

Bij gebruik van een tekensweergavemodus interpreteert de VIC de schermgegevens als schermcodes van tekens. De pixelgegevens voor elk teken worden gelezen uit een tekenset die in het geheugen staat. Registers bepalen het startadres van de geselecteerde tekenset.

De grootte en de indeling van de tekengegevens hangen af van de tekensweergavemodus. De standaardtekstmodus leest tekengegevens als acht bytes per teken, één regel monochrome pixels per byte, voor in totaal 512 mogelijke schermcodes (4KB).

Het startadres van de tekenset wordt als drie bytes opgeslagen in $D068 - $D06A. Om bijvoorbeeld de tekenset op 3.D000 te selecteren:

CHARPTRLSB $D068	CHARPTRMSB $D069	CHARPTRBNK $D06A
$00	$D0	$03

De ROM-gegevens bevatten drie monochrome tekensets (gebruikt door het BASIC-commando **FONT**):

- 2.9000: Tekenset A, de C64-tekenset met kleine ASCII-letters
- 3.D000: Tekenset B, een gestileerde set die op een ASCII-terminal lijkt
- 2.D000: Tekenset C, de C64-PETSCII-tekenset (de standaard)

Uw programma kan elk van de ingebouwde tekensets naar eigen wens aanpassen door het 4KB-gebied naar RAM te kopiëren en de CHARPTR-registers dienovereenkomstig in te stellen.

De tekengenerator van de VIC-IV kent nog een andere manier om tekengliefen intern op te slaan. Stelt u CHARPTR in op de speciale waarde $001000, dan gebruikt de VIC-IV 4KB aan intern tekengeheugen waarin de standaard-PETSCII-tekenset is opgeslagen. Zo kunt u alle drie de ROM-regio's voor tekensets een andere bestemming geven zonder de toegang tot de PETSCII-tekenset te verliezen.

Spritegeheugen vinden

De VIC-IV ondersteunt acht hardwaresprites. Naar de afbeeldingsgegevens van deze sprites wordt verwezen door een reeks spritepointers. Anders dan de VIC-II van de Commodore 64 ondersteunt de VIC-IV het verplaatsen van de spritepointers en de afbeeldingsgegevens naar een willekeurige plaats in chip-RAM. De VIC-IV is standaard voor een deel compatibel met de VIC-II, waardoor extra registerbits nodig zijn om de verplaatsing volledig in te schakelen.

Bij de VIC-IV kunnen ook de spritepointers zelf overal in het chip-RAM staan. De locatie van de spritepointers wordt opgeslagen in de registers op $D06C - $D06E. Dit adres moet uitgelijnd zijn op een grens van 16 bytes: wanneer u het adres in hexadecimale notatie uitschrijft, moet het meest rechtse cijfer een 0 zijn. Het adres $2C60 is bijvoorbeeld uitgelijnd op een grens van 16 bytes.

Bit 7 van $D06E vertelt de VIC-IV dat de spritepointers 16 bits (twee bytes) breed zijn. Indien de bit niet is gezet, gaat de VIC-IV uit van waarden van 8 bits (één byte) voor de pointers, net zoals de VIC-II.

De waarde van een spritepointer is het adres van de beeldgegevens van de sprite, gedeeld door 64. Deze ongewone eigenschap stelde de VIC-II oorspronkelijk in staat om spritegegevens binnen een bereik van 16KB te lokaliseren aan de hand van een waarde van één byte. De VIC-IV houdt deze eigenschap vanwege de compatibiliteit in stand en staat bovendien toe dat deze waarde 16 bits breed is. Eén gevolg van dit ontwerp is dat het adres van een sprite-afbeelding moet zijn uitgelijnd op een gebied van 64 bytes binnen een pagina: de twee meest rechtse hexadecimale cijfers van een spriteadres moeten $00, $40, $80 of $C0 zijn.

Om bijvoorbeeld de spritepointers te verplaatsen naar 0.3200 en 16-bits pointerwaarden in te schakelen:

SPRPTRADRLSB $D06C	SPRPTRADRMSB $D06D	SPRPTRBNK $D06E (0-6)	SPRPTR16 $D06E (7)
$00	$32	$00	+$80

```
LDA #$00
STA $D06C
LDA #$32
STA $D06D
LDA #$80
STA $D06E
```

Om de eerste sprite beeldgegevens op adres 4.3100 te laten gebruiken, stelt u de 16-bits pointer in op 4.3100 / 64 = $10C4 (in little-endian-volgorde):

```
LDA #$C4
STA $3200
LDA #$10
STA $3201
```

Kleurengeheugen

De VIC-IV stelt een volledig schermbeeld samen uit één set teken- of pixelgegevens in het schermgeheugen en een aparte set kleurgegevens in het kleurengeheugen. In de 80x25-tekstmodus bevatten de eerste 2000 bytes van het schermgeheugen bijvoorbeeld de schermcodes van de tekens en zijn de eerste 2000 bytes van het kleurengeheugen de kleuren van die tekens.

De VIC-IV heeft 32KB geheugen dat speciaal voor kleurgegevens is gereserveerd en dat alle mogelijke schermmodi kan ondersteunen. U kunt dit geheugen rechtstreeks benaderen via de hogere geheugenadressen FF8.0000 - FF8.7FFF.

Voor het gemak zijn de eerste 2KB van het kleurengeheugen ook zichtbaar op de adressen 1.F800 - 1.FFFF. Dit adresvenster in bank 1 laat zich niet via de geheugenconfiguratie uitschakelen: het benadert altijd de eerste 2KB van het kleurengeheugen.

Vanwege achterwaartse compatibiliteit is de eerste 1KB van dit venster, samen met de I/O-registers, ook opgenomen in de banking op $D800 - $DBFF. Bit 0 van het bankingregister van de VIC-III op $D030 (signaalnaam "CRAM2K") breidt dit uit tot de volledige 2KB, als $D800 - $DFFF.

Het startadres van het kleurengeheugen van de VIC-IV is met een register in te stellen, als offset binnen de gebied FF8.0000 - FF8.7FFF van het kleurengeheugen. Indien u niet alle 32KB van het kleurengeheugen voor uw schermweergave nodig heeft, kunt u het startadres verschuiven en het begin van het kleurengeheugen voor andere doelen gebruiken.

Om bijvoorbeeld het begin van het kleurengeheugen met 2KB te verschuiven, stelt u het register in op $0800 (in little-endian-volgorde):

COLPTRLSB $D064	COLPTRMSB $D065
$00	$08

Het verschuiven van het startadres verschuift *noch* het venster op het kleurengeheugen op 1.F800, *noch* de banking van het kleurengeheugen op $D800. Verschuift u het begin van het kleurengeheugen met 2KB, dan gebruiken zowel het venster als de I/O-bank de 2KB-gebied die voor de VIC-IV onzichtbaar blijft. Zo kunt u het venster op het kleurengeheugen herbestemmen als RAM voor algemeen gebruik.

Dit BASIC-programmaatje stelt de kleurengeheugen-offset in op 2KB ($0800) en laat vervolgens zien dat de waarde op $FF80000 ook zichtbaar is op $1F800 en $D800. Ten slotte verandert het de kleur van het eerste teken door $FF80800 in te stellen.

```
10 POKE $D064,0:POKE $D065,8
20 PRINT CHR$(5);CHR$(147);"HALLO DAAR."
30 POKE $FF80000,$BB
40 PRINT HEX$(PEEK($FF80000))
50 PRINT HEX$(PEEK($1F800))
60 PRINT HEX$(PEEK($D800))
70 POKE $FF80800,4
```

GROTE GEHEUGENBEWERKINGEN MET DIRECT MEMORY ACCESS

In vroege microcomputers voert de CPU alle geheugenbewerkingen uit. Grote geheugenbewerkingen, zoals het kopiëren van een blok geheugen van de ene plaats naar de andere, zijn tijdrovend en leggen de interactie met het programma veelal even stil. Dit beperkt programma's tot kleine geheugenbewerkingen tijdens interactieve taken, zoals spellen en sommige toepassingen.

De Commodore 65 beschikt over specifieke hardware voor het kopiëren en verwerken van grote hoeveelheden geheugen, een systeem dat bekend staat als *Direct Memory Access* (DMA). De DMA-hardware voert geheugentaken uit, bekend als *taken*. De MEGA65 breidt deze mogelijkheden uit met hogere snelheden en voorzieningen die gespecialiseerd zijn in grafische en audiogegevens.

De DMA van de MEGA65 verschilt op één belangrijk punt van de DMA van de C65 (en van de DMA in andere microcomputers): bij de MEGA65 zit de DMA in de CPU ingebouwd en DMA-taken pauzeren de uitvoering van CPU-instructies. Dat lijkt het doel van specifieke DMA-hardware tegen te werken, ware het niet dat de MEGA65 de DMA-taken in vergelijking met de C65 zeer snel uitvoert, dankzij de hogere maximale CPU-snelheid. Het kopiëren van een grote grafische afbeelding naar het scherm gaat snel genoeg om een vast onderdeel te vormen van de code in de hoofdlus van een spel.

Dit vereenvoudigt ook het programmeermodel: een programma kan een DMA-taak behandelen als een bijzondere soort CPU-instructie, zonder extra logica om te wachten tot de taken klaar zijn. Er bestaat zelfs een methode om de DMA aan te roepen waarbij u de takenlijst voor de DMA rechtstreeks in uw code opneemt, zonder daarvoor extra geheugen te hoeven reserveren. In het algemeen geldt: als de DMA de bewerking ondersteunt die u wilt uitvoeren en de bewerking meer dan acht bytes geheugen omvat, dan is een DMA-taak sneller dan gelijkwaardige CPU-instructies.

Deze sectie introduceert DMA in het kort, als onderdeel van dit overzicht van het geheugen. Voor een volledige beschrijving van de DMA-controller en de uitgebreide voorzieningen van de MEGA65, waaronder grafische en audio-DMA, zie Hoofdstuk/-Bijlage P op bladzijde P-3.

DMA-taken

Om een DMA-taak uit te voeren, beschrijft u de taak in het geheugen en roept u daarna de DMA aan door het adres van de taakbeschrijving naar de registers te schrijven. De DMA voert de taak uit zodra u de laatste adreswaarde van de taak naar het register schrijft. De CPU onderbreekt zijn werk om de taak uit te voeren en gaat daarna verder met de volgende programmainstructie.

Conceptueel bestaat een taakbeschrijving uit de volgende velden:

1. Een opdracht: kopiëren of vullen

2. Een aantal te verwerken bytes, tot 64KB
3. Bij kopiëren: het bronstartadres
4. Bij vullen: een vulbyte
5. Het bestemmingsstartadres
6. De instelling van de wijze waarop de bron- en bestemmingsgebieden moeten worden doorlopen ("adressering")

Meerdere taken kunnen in één enkele aanroep van de DMA worden beschreven, in de vorm van een "taaklijst". De taken worden uitgevoerd in de volgorde waarin ze in de lijst voorkomen.

De MEGA65 breidt het formaat van het C65-DMA-taakrecord uit met een lijst van taakoptietokens. Veel van de uitgebreide voorzieningen van de MEGA65, zoals het kopiëren van afbeeldingsgegevens met transparantie, maken gebruik van taakoptietokens.

DMA gebruiken vanuit BASIC

In BASIC 65 voert het **EDMA**-commando één DMA-taak uit, met beperkte opties. Het **DMA**-commando van de Commodore 65 wordt eveneens ondersteund, maar EDMA verdient de voorkeur voor alle nieuwe programma's die niet op een C65 hoeven te draaien.

Het **EDMA**-commando aanvaardt vier door komma's gescheiden argumenten: de opdracht (0=kopiëren, 3=vullen), het aantal bytes, ofwel een vulbyte (bij vullen) ofwel een bronstartadres (bij kopiëren) en het bestemmingsstartadres.

```
10 SC=WPEEK($D060)+PEEK($D062)*$10000
20 FOR X=32 TO 40
30 EDMA 3,80*25,X,SC
40 GETKEY A$
50 NEXT
```

De overige DMA-voorzieningen zijn vanuit het **EDMA**-commando niet eenvoudig te benaderen. Om geavanceerdere DMA-taken uit te voeren, kan een BASIC-programma de taakgegevens met **POKE** in het geheugen zetten en de taak via de registers aanroepen, zoals hieronder beschreven.

DMA gebruiken vanuit machinetaal

Er zijn verschillende manieren om DMA-taken te beschrijven en te starten, waaronder manieren om programma's te schrijven die compatibel zijn met de Commodore 65. Deze sectie beschrijft twee methoden: het aanroepen van een Enhanced-DMA-taaklijst van de MEGA65 vanuit een gegevensstructuur die op een geheugenadres staat en het aanroepen van een Enhanced-taaklijst die inline in de machinetaal staat. Voor een volledige beschrijving van de opties, inclusief de lijst met de mogelijke MEGA65-taakoptietokens, zie Hoofdstuk/Bijlage P op bladzijde P-3.

Een Enhanced-taaklijst in het geheugen aanroepen

Een Enhanced-DMA-taaklijst kan overal in het geheugen staan en kan zowel met een fysiek 28-bits adres worden benaderd als met een 16-bits adres dat via MAP opnieuw is toegewezen.

Om een Enhanced-DMA-taaklijst aan te roepen vanuit een record dat binnen de eerste megabyte van het geheugen staat, schrijft u het adres naar de volgende registers, in deze volgorde:

1. $D702: de *bank*-adresbyte, bits 16-22; de hoogste bit wordt genegeerd
2. $D701: de *pagina*-adresbyte, bits 8-15
3. $D705: de laagste adresbyte, bits 0-7

Door de laagste adresbyte naar $D705 te schrijven, start de uitvoering van de taaklijst, waarbij het adres als een fysiek (niet-gemapt) adres wordt geïnterpreteerd.

Om bijvoorbeeld een Enhanced-DMA-taaklijst aan te roepen die op adres $42A80 staat:

```
LDA #$04
STA $D702
LDA #$2A
STA $D701
LDA #$80
STA $D705    ; De DMA-taaklijst begint met uitvoeren...
```

De MEGA65 heeft de mogelijkheid om een Enhanced-DMA-taaklijst op een willekeurig 28-bits adres op te slaan. Om een Enhanced-DMA-taaklijst aan te roepen vanuit een record dat op een 28-bits adres staat (zoals een adres in het zolder-RAM), schrijft u het adres naar de volgende registers, in deze volgorde:

1. $D702: de *bank*-adresbyte, bits 16-22; de hoogste bit wordt genegeerd
2. $D704: de *megabyte*-adresbyte, bits 20-27; overlapt de bank
3. $D701: de *pagina*-adresbyte, bits 8-15
4. $D705: de laagste adresbyte, bits 0-7

In dit geval liggen de registeradressen *niet* naast elkaar. Vóór de MEGA65 deze voorziening kreeg, bestond er geen "megabyte"-adresbyte en stelden programma's de "bank"-adresbyte in zonder de "megabyte"-adresbyte in te stellen. Om compatibel te blijven met deze programma's, zet het schrijven van de adresbank naar $D702 de adresmegabyte automatisch weer op nul. Om alle vier de adresbytes te gebruiken, moet u de bank schrijven vóórdat u de megabyte schrijft.

Om de DMA het adres van de taaklijst te laten interpreteren vanuit het gezichtspunt van de CPU op de 16-bits adresruimte, met inachtneming van het MAP-register,

schrijft u het laatste lage byte in plaats daarvan naar $D706.[7] De adressen in de taaklijst zelf zijn nooit gemapte adressen: DMA-taken werken altijd op fysieke adressen. Het gebruik van $D706 verandert alleen de manier waarop het adres van de taaklijst wordt geïnterpreteerd, namelijk als gemapt 16-bits adres.

Het volgende voorbeeld roept een Enhanced-DMA-taaklijst aan die is opgeslagen onder het label `dmajob`. Voor de volledigheid gebruikt het voorbeeld alle vier de registers, zodat de taaklijst ook naar een hogere megabyte van het geheugen geassembleerd zou kunnen zijn. In de praktijk wordt de meeste programmacode naar de eerste 384KB van het geheugen geassembleerd, zodat het megabyteregister $D704 kan worden weggelaten.

```
LDA #^dmajob
STA $D702
LDA #(dmajob >>> 20)
STA $D704
LDA #>dmajob
STA $D701
LDA #<dmajob
STA $D705  ; De DMA voert direct uit

; Het programma gaat verder...
RTS

dmajob:
!byte $0B, $00       ; tokenlijst
!byte $03, $00, $08  ; vul ($03) $0800 bytes
!byte $BB, $00, $00  ; met waarde $BB
!byte $00, $30, $04  ; beginnend op 4.3000
!byte $00, $00, $00  ; geen rare trucs
```

Het `^dmajob` hierboven (een dakje gevolgd door een symboolnaam) is de syntaxis van de ACME-assembler voor de bankbyte van het 24-bits adres van het symbool `dmajob`. Pas dit aan naar de syntaxis van uw assembler, of codeer deze waarde vast als u weet in welke bank het `dmajob`-geheugen komt te liggen.

Een Enhanced-taaklijst inline in de code aanroepen

In plaats van eerst een taaklijst in het geheugen op te zetten, kunt u de taaklijst ook inline in uw machinetaal opnemen. Dit is een handige manier om een DMA-taak te schrijven alsof het een CPU-instructie met vooraf vastgestelde waarden is. Gebruik daartoe de instructie `STA $D707` (met een willekeurige accumulatorwaarde), gevolgd door assembler-directieven waarmee de gegevensstructuur van de Enhanced-taaklijst wordt geassembleerd. De 45GS02-CPU weet dan dat hij de besturing aan de DMA

[7] De MEGA65 ondersteunt een C65-compatibel DMA-taaklijstformaat, dat een programma kan aanvragen door het lage adresbyte naar $D700 te schrijven in plaats van naar $D705 / $D706. U gebruikt dit uitsluitend als u wilt dat uw programma draait op compatibele C65-prototypehardware. Let op: sommige C65-prototypen hebben een onvolledige DMA-implementatie.

moet geven en de programmateller voorbij het einde van de DMA-taakgegevens moet zetten.

```
STA $D707  ; De DMA voert direct uit
!byte $0B, $00       ; tokenlijst
!byte $03, $00, $08  ; vul ($03) $0800 bytes
!byte $BB, $00, $00  ; met waarde $BB
!byte $00, $30, $04  ; beginnend op 4.3000
!byte $00, $00, $00  ; geen rare trucs

; Het programma gaat verder...
RTS
```

Het formaat van de Enhanced-taaklijst

Deze sectie presenteert een vereenvoudigde versie van het formaat van de Enhanced-taaklijst. Voor een volledige beschrijving, inclusief de velden die de 45GS02-DMA nog niet ondersteunt en de volledige lijst met taakoptietokens, zie Hoofdstuk/Bijlage P op bladzijde P-3.

De *taaklijst* is een binair formaat waarin de waarden in bytes zijn verpakt. De lijst bestaat uit één of meer taakrecords, waarbij de "chain"-bit voor elke taak is gezet, behalve voor de laatste taak in de lijst.

Een Enhanced-*taakrecord* bestaat uit een lijst van één of meer taakoptietokens die altijd eindigt met het token "end of job option list" ($00), gevolgd door een gegevensstructuur van 12 bytes. Elk taakoptietoken is een bytewaarde en bij sommige tokens hoort nog een argumentwaarde van één byte. Tenzij u een programma schrijft dat bedoeld is voor een Commodore 65, gebruikt uw tokenlijst doorgaans het token $0B (zonder argumenten) om het taakrecordformaat van 12 bytes te kiezen.

De structuur van 12 bytes ziet er als volgt uit:

Bytes	Inhoud
$00	Opdracht **1 – 0**: 00 voor kopiëren, 11 voor vullen **2**: 0 indien dit de laatste taak in de lijst is, 1 indien de lijst doorloopt **3**: Interrupts toelaten
$01 – $02	Aantal (16 bits)
$03 – $05	Bronadres of vulbyte Vulbyte in $03 Bronadres (20 bits) in $03 – $05 Hoge bits van $05: **6**: Richting: 0 om vanaf de start op te tellen, 1 om vanaf de start af te tellen **7**: I/O: 1 om tijdens de DMA de I/O-registers op $D000 – $DFFF te benaderen
$06 – $08	Bestemmingsadres (20 bits) Gebruikt hetzelfde formaat als het bronadres
$09	Adresseeropties **1 – 0**: Adresseermethode van de bron **3 – 2**: Adresseermethode van de bestemming Adresseermethoden: 00: Lineair: schuift voor alle bytes vooruit in de aangegeven richting 10: Vasthouden: schuift niet op, maar herhaalt alleen het startadres
$0A – $0B	Gereserveerd, altijd ingesteld op $00 $00

Bron- en bestemmingsadressen worden in het taakrecord opgeslagen als 20-bits adressen, in little-endian-volgorde. Om de volledige 28-bits adresruimte te kunnen benaderen, gebruikt u de taakoptietokens waarmee de hoogste byte van het startadres ($80 $xx) en het bestemmingsadres ($81 $xx) wordt ingesteld. Het token geldt voor alle taken in de taaklijst.

Bij het uitvoeren van een keten van meerdere Enhanced-taken heeft elk taakrecord zijn eigen taaktokenlijst. Taaktokens van eerder in de keten gelden voor alle latere taken in dezelfde keten. Om de tokens voor een volgende taak ongewijzigd te laten, gebruikt u één enkele $00 als taaktokenlijst.

Voorbeeld: het geheugen met de BASIC-programmatekst in zijn geheel kopiëren naar het zolder-RAM.

```
dmajob:
!byte $80, $00        ; Hoogste byte van de bron: $00
!byte $81, $80        ; Hoogste byte van de bestemming: $80
!byte $0B             ; Gebruik het taakrecordformaat van 12 bytes
!byte $00             ; Einde van de tokenlijst
!byte $00, $00, $d7   ; kopieer ($00) $d700 bytes
!byte $00, $20, $00   ; vanaf (00)0.2000
!byte $00, $00, $00   ; naar (80)0.0000
!byte $00, $00, $00   ;
```

Voorbeeld: een keten van drie taken, waarin elke taak een vulbewerking uitvoert.

```
dmajob:
!byte $0B, $00        ; tokenlijst
!byte $07, $00, $01   ; vul ($03) $0100 bytes
                      ; ga verder met de taakketen (+$04)
!byte $BB, $00, $00   ; met waarde $BB
!byte $00, $30, $04   ; beginnend op 4.3000
!byte $00, $00, $00   ;

!byte $00             ; tokenlijst (ongewijzigd)
!byte $07, $00, $01   ; vul ($03) $0100 bytes
                      ; ga verder met de taakketen (+$04)
!byte $CC, $00, $00   ; met waarde $CC
!byte $00, $31, $04   ; beginnend op 4.3100
!byte $00, $00, $00   ;

!byte $00             ; tokenlijst (ongewijzigd)
!byte $03, $00, $01   ; vul ($03) $0100 bytes
                      ; einde van de taakketen
!byte $DD, $00, $00   ; met waarde $DD
!byte $00, $32, $04   ; beginnend op 4.3200
!byte $00, $00, $00   ;
```

DMA-taken negeren alle mapping- en banking-instellingen. U kunt aangeven dat een bron- of bestemmingsadres het gebied $D000 - $DFFF als I/O-registers moet behandelen, door bit 23 van het adresrecord in te stellen.

U kunt een DMA-taak voorstellen als een proces met twee cursors: één voor de bron (bij kopiëren) en één voor de bestemming. De DMA schuift elke cursor volgens de taakbeschrijving vooruit, gedurende evenveel stappen als het aantal aangeeft en kopieert of vult bij elke stap de bestemming.

Adresseermethoden bieden één manier om invloed uit te oefenen op de beweging van de cursors:

- De *lineaire* adresseermethode schuift de cursor bij elke stap één adres op.
- De *vasthoudende* adresseermethode schuift de cursor niet op. Deze blijft bij elke stap op het startadres staan.
- De *doorlooprichting* geeft aan of er bij elke stap tijdens lineaire adressering wordt opgeteld of afgeteld.

Voorbeeld: een kopieerbewerking waarbij de broncursor wordt vastgehouden en de bestemmingscursor lineair opschuift, lijkt op het uitvoeren van een vulbewerking met de byte op het bronadres.

```
dmajob:
!byte $0B, $00        ; tokenlijst
!byte $00, $80, $00   ; kopieer ($00) $0080 bytes
!byte $00, $19, $00   ; vanaf 0.1900
!byte $40, $31, $04   ; naar 4.3140
```

```
!byte $02, $00, $00  ; houd de bron vast
```

De uitgebreide DMA van de MEGA65 ondersteunt nog veel geavanceerdere doorloop-patronen, die met taakoptietokens worden aangestuurd. Met de DMA kunt u lijnen en patronen in het scherm- en kleurengeheugen tekenen.

Commodore heeft de implementatie van de DMA van de Commodore 65 beschreven, maar deze nooit voltooid. Omwille van de historie reserveert de MEGA65 bits in het DMA-taakrecord voor deze onvoltooide voorzieningen, ook al kan een implementatie nooit werkelijk compatibel zijn met het prototype van de Commodore 65.

GEVORDERDE GEHEUGENONDERWERPEN

De geheugenmapping testen met de Matrix Mode-debugger

In de Matrix Mode-debugger,[8] geeft het `m`-commando de gegevens weer op een opgegeven geheugenlocatie. Om bijvoorbeeld de inhoud van het RAM vanaf adres 0.2000 weer te geven:

```
m2000
```

De debugger leest *altijd* vanuit 32-bits adressen en houdt geen rekening met MAP of andere voorzieningen voor geheugenbanking, ook niet wanneer er een adres van vier hexadecimale cijfers wordt opgegeven. Om het gemakkelijker te maken te zien hoe de CPU de 16-bits adressen ziet, plaatst de MEGA65 het gezichtspunt van de CPU op de hogere adressen 777.0000 - 777.FFFF. Om bijvoorbeeld weer te geven wat de CPU ziet op het 16-bits adres $2000:

```
m7772000
```

De bank 777 houdt rekening met het MAP-register, evenals met de mechanismen voor het mappen van geheugen die in de volgende secties worden beschreven.

Geheugenbanking in C64-stijl

De 16-bits adressen $0000 en $0001 verwijzen naar registers. Deze registers functioneren precies hetzelfde als de CPU-registers die op deze adressen in een Commodore 64 te vinden zijn. Zij regelen het mappen van de C64-ROM in de 16-bits adresruimte, precies zoals dat op een C64 gebeurt.

In een MEGA65 dienen deze registers twee doelen: ze nemen de I/O-registers op $D000 - $DFFF op in de 16-bits adresruimte, zodat deze het chip-RAM overlappen en ze verschaffen in de C64-modus (**GO64**) toegang tot de C64-versies van BASIC en KERNAL.

Om in de MEGA65-modus de I/O-registers uit de 16-bits adresruimte te halen, zodat de 16-bits adressen $D000 - $DFFF het onderliggende chip-RAM op 0.D000 - 0.DFFF benaderen, schrijft u $84 naar locatie $0001. Om de I/O-registers te herstellen, schrijft u $85 naar locatie $0001.

OPMERKING: De registers $0000 en $0001 zijn uitsluitend schrijfbaar. Ze werken niet met de machinetaalinstructies TRB en TSB.

---

[8] Ter herinnering: om de Matrix Mode-debugger te activeren, houdt u  ingedrukt en drukt u op . Zie Hoofdstuk/Bijlage O op bladzijde O-16 voor meer informatie over de Matrix Mode-debugger.

Dit BASIC-voorbeeld demonstreert hoe u locatie $D020 eerst als I/O-register (de randkleur) benadert en vervolgens als RAM-adres. Probeer te raden wat het zal tonen voordat u het uitvoert.

```
10 POKE $D020,3
20 POKE 1,$84
30 POKE $D020,4
40 X=PEEK($D020)
50 POKE 1,$85
60 PRINT X
70 PRINT PEEK($D020)
```

Banking in C64-stijl geldt alleen voor adresblokken van 8KB die niet door het MAP-register zijn geselecteerd ("ongemapte" blokken). Als de banking in C64-stijl bijvoorbeeld de I/O-registers op $D000 - $DFFF heeft liggen en de regio $C000 - $DFFF met een offset van 0 is gemapt, dan benadert het programma het RAM op die locatie en *niet* de I/O-registers. De regio $C000 - $DFFF moet ongemapt zijn, wil de CPU toegang tot de I/O-registers hebben.

Op vergelijkbare wijze kunt u het MAP-register gebruiken om het RAM op de adressen 0.0000 en 0.0001 te benaderen, dat gewoonlijk door de bankingregisters in C64-stijl wordt verborgen. Als het MAP-register het adresblok $0000 - $1FFF voor mapping selecteert en een offset van 0 toepast, benaderen de 16-bits adressen $0000 en $0001 het RAM op 0.0000 en 0.0001 en niet de bankingregisters in C64-stijl. De bankingregisters in C64-stijl zijn uitsluitend te benaderen wanneer de regio $0000 - $1FFF ongemapt is.

Het is gebruikelijk dat een MEGA65-programma de I/O-registers op $D000 in de banking laat en het MAP-register gebruikt wanneer het het onderliggende RAM moet benaderen. Zo'n programma hoeft het bankingregister in C64-stijl niet te wijzigen.

Het bankingregister $D030 van de VIC-III

De Commodore 65 biedt een bankingmechanisme om aanvullende ROM-gegevens met 16-bits adressen toegankelijk te maken. Deze banking wordt bestuurd door het I/O-register $D030.[9]

De bits 3, 4, 5 en 7 mappen gebieden van ROM-gegevens in de 16-bits adresruimte volgens deze tabel. (Ter herinnering: bits worden van rechts naar links genummerd vanaf 0, zodat bit 7 de hoogste bit is.)

$D030-bit	Signaal-naam	20-bits adres	16-bits adres
3	ROM8	$28000 - $29FFF	$8000 - $9FFF

Vervolgd ...

[9] Om het bankingregister van de VIC-III op $D030 te benaderen, moet de computer de VIC-III/IV-"I/O-persoonlijkheid" gebruiken, die verderop wordt beschreven. Dit is de standaard in de MEGA65-modus.

...Vervolgd

$D030-bit	Signaal-naam	20-bits adres	16-bits adres
4	ROMA	$2A000 - $2BFFF	$A000 - $BFFF
5	ROMC	$2C000 - $2CFFF	$C000 - $CFFF
7	ROME	$2E000 - $2FFFF	$E000 - $FFFF

Als $D030 bijvoorbeeld is ingesteld op $64 = %01100100, dan is bankingbit 5 gezet en zijn de bits 3, 4 en 7 gewist. Dit betekent dat ROMC is ingeschakeld en dat de 16-bits adressen $C000 - $CFFF 2.C000 - 2.CFFF benaderen. Dit is de standaard-toestand van $D030 wanneer u de computer inschakelt.

Net als bij banking in C64-stijl geldt de $D030-banking alleen voor adresblokken van 8KB die het MAP-register niet voor het toepassen van een offset heeft geselecteerd. Voor ongemapte regio's geldt: indien voor een regio of de ROM-banking in C64-stijl of de $D030-ROM-banking is ingeschakeld, verwijzen die adressen naar de ROM in bank 2. Elk van deze ROM-bankingmechanismen gebruikt de ROM in bank 2, dus er is geen conflict wanneer beide tegelijk zijn ingeschakeld.

Houd er rekening mee dat $D030 geen enkele MEGA65-KERNAL-ROM uit bank 3 inmapt. De KERNAL gebruikt zelf het MAP-register om zijn eigen code te benaderen en gebruikt de $D030-banking uitsluitend om ROM-code op 2.C000 te benaderen met ROMC ingeschakeld. Zoals bij het bankingregister in C64-stijl kan een typisch MEGA65-programma de bankingbits van $D030 ongemoeid laten en MAP gebruiken om indien nodig RAM of ROM op deze adressen te benaderen.

Cartridge-ROM

Indien er een Commodore 64-cartridge in de uitbreidingspoort is geplaatst, leidt het geheugensysteem de toegang tot delen van de 16-bits adresruimte om naar de lijnen van de poort, welke doorgaans verbonden zijn met de ROM-chips in de cartridge. De cartridge bestuurt twee extra lijnen waarmee zij via de CPU-registers op de adressen $0000 en $0001, bekend als EXROM en GAME, verschillende geheugenconfiguraties kan opvragen. Deze configuraties zijn identiek aan die van een Commodore 64. Zie voor meer informatie een naslagwerk over de Commodore 64.

De Commodore 64 herkent dat er een C64-cartridge is geïnstalleerd wanneer één of beide lijnen EXROM en GAME laag (0) zijn. Een typische cartridge in C64-stijl maakt op basis van de waarden van EXROM en GAME één of twee gebieden van 8KB voor het systeem toegankelijk. Als beide lijnen hoog zijn, wordt er geen cartridge-ROM op 16-bits adressen gemapt.

De cartridge-hardware is verbonden met de volledige 16-bits adresbus en kan in beginsel voor alle 65536 adressen waarden teruggeven. Op de MEGA65 is de volledige

64KB aan cartridgeadressen beschikbaar op de daartoe bestemde bovenste adressen 400.0000 - 400.FFFF.[10]

De adresvertaling door het MAP-register heeft voorrang op de geheugenconfiguratie van de C64-cartridge. De mapping van cartridge-ROM overschrijft de geheugenmapping in $D030-stijl.

I/O-persoonlijkheden

De I/O-registers vormen de belangrijkste manier waarop de KERNAL, de BASIC-interpreter of een machinetaalprogramma met de MEGA65-hardware omgaat. Deze registers gedragen zich als geheugenlocaties waarmee u waarden in de hardware instelt, zoals voor het weergeven van grafische beelden of het voortbrengen van geluid, of waarden uit de hardware leest, zoals gebruikersinvoer op het toetsenbord of de joystick. In BASIC geschreven programma's benaderen deze registers doorgaans indirect via commando's, hoewel zij dit ook rechtstreeks via geheugentoegang kunnen doen.

De MEGA65 start in een configuratie waarmee de meeste van zijn voorzieningen via registers op 16-bits adressen in $D000 - $DFFF beschikbaar zijn. Voor deze registers bestaan er vier verschillende configuraties, bekend als *I/O-persoonlijkheden:*

- **MEGA65**. Schakelt alle voorzieningen van de MEGA65 VIC-IV en de Commodore 65 VIC-III in. Dit is de standaard.
- **Commodore 65**. De VIC-III zonder de VIC-IV-uitbreidingen. Voor C65-compatibiliteit.
- **Commodore 64**. De VIC-II. Wordt gebruikt door de C64-modus (**GO64**).
- **MEGA65 Ethernet**. Een persoonlijkheid voor een speciaal doel, voor het benaderen van de Ethernet-poort van de MEGA65.

De registers van alle vier de persoonlijkheden zijn beschikbaar in de 28-bits adresruimte, in elk 4KB-blok van het bereik FFD.0000 - FFD.3FFF. Wanneer een bepaalde persoonlijkheid is ingesteld, wordt $D000 - $DFFF gemapt op het overeenkomstige blok van bovenste adressen. De standaard-MEGA65-persoonlijkheid mapt deze adressen bijvoorbeeld op FFD.3000 - FFD.3FFF. Met 28-bits adressen kunt u alle vier de persoonlijkheden tegelijk benaderen, hoewel u rekening moet houden met wisselwerking tussen deze persoonlijkheden (zoals "hete registers" van de VIC).

Let op dat het bankingregister $D030 niet beschikbaar is in de C64-I/O-persoonlijkheid. Als uw programma in de C64-modus start of op een andere manier naar die I/O-persoonlijkheid omschakelt, moet het eerst omschakelen naar de C65- of MEGA65-persoonlijkheid voordat het dit register benadert.

Een typisch voor de MEGA65 geschreven programma gebruikt de MEGA65-persoonlijkheid en gaat ervan uit dat deze al ingesteld is wanneer het programma

[10] De MEGA65 ondersteunt een protocol voor MEGA65-cartridges die automatisch in de MEGA65-modus opstarten. Dit wordt in een latere versie van deze handleiding gedocumenteerd.

start. Zie Hoofdstuk/Bijlage J op bladzijde J-3 voor informatie over hoe u kunt wisselen van gebruikte persoonlijkheid, bijvoorbeeld om bij het starten van uw programma uit voorzorg de gewenste persoonlijkheid in te stellen.

ROM omzetten in RAM

De MEGA65 bewaart de code voor de KERNAL en de BASIC-interpreter in de banken 2 en 3. Om deze code te beschermen tegen overschrijving door andere programma's, dwingt de MEGA65 af dat deze banken alleen-lezen zijn, waarmee wordt gesimuleerd dat er ROM-chips in Commodore-stijl op die adressen zijn aangesloten. De MEGA65-Hypervisor laadt de ROM-gegevens tijdens het opstarten en beheert de schrijfbeveiliging.

Een programma kan de schrijfbeveiliging van de banken 2 en 3 opheffen om die gebieden als chip-RAM te gebruiken. De meeste programma's hebben niet de hele ROM in het geheugen nodig. Een programma dat in de C65-modus draait, heeft bijvoorbeeld naar alle waarschijnlijkheid geen C64-BASIC-ROM-code nodig. De geheugenindeling van het chip-RAM eerder in dit hoofdstuk beschrijft welke ROM-gebieden voor welk doel worden gebruikt.

Een zorgvuldig geschreven programma dat de ingebouwde routines geheel niet nodig heeft, kan het volledige gebied van 128KB een andere bestemming geven. Zorg er bij het overschrijven van de KERNAL voor dat u interrupts uitschakelt en de vectoren van de interrupt-handlers vervangt.

Om de schrijfbeveiliging van de banken 2 en 3 uit te schakelen, roept u de Hypervisor-trap ROM Write-enable aan (`hyppo_rom_writeenable`), als volgt:

```
LDA #$02
STA $D641
CLV
```

Om deze weer in te schakelen gebruikt u op dezelfde manier de trap ROM Write-protect (`hyppo_rom_writeprotect`) met `LDA #$00`. De instructie na de `STA` moet een instructie met één cyclus zijn, doorgaans `CLV`: de CPU voert die bij het terugkeren uit de Hypervisor al dan niet uit. `NOP` wordt niet aanbevolen, omdat deze op de 45GS02 als prefix voor de volgende instructie kan werken.

Om naar RAM in de banken 2 en 3 te schrijven, moet u het MAP-register, 32-bits instructies of DMA gebruiken. Wanneer banking in C64-stijl of $D030-banking actief is (en de betreffende 16-bits adressen niet-gemapt zijn), benadert het lezen van die adressen bank 2, terwijl het schrijven naar die adressen in RAM van bank 0 terechtkomt. Dit lijkt op wat er gebeurt wanneer u op een Commodore 64 naar een adres met ROM-banking schrijft.

Om na te gaan of het gebied schrijfbeveiligd is, kunt u eenvoudig proberen er een waarde naar te schrijven. De MEGA65 start op met ingeschakelde schrijfbeveiliging.

Het register $D640 is alleen beschikbaar in de MEGA65-I/O-persoonlijkheid.

Houd er rekening mee dat dit *geen* bankingmechanisme is, zoals dat op een Commodore 64 wel het geval is. Als uw programma de oorspronkelijke ROM-code nodig heeft nadat u die heeft overschreven (bijvoorbeeld om deze te herstellen), kopieert u eerst met DMA de oorspronkelijke ROM naar het zolder-RAM.

MAP gebruiken om geheugen op hogere adressen te benaderen

Zoals eerder besproken, kan het MAP-register elk blok van 8KB hermapen naar een andere locatie in de eerste 1MB van de adresruimte. Dit is handig voor het benaderen van de 384KB chip-RAM, inclusief eventuele toekomstige uitbreidingen. De MAP-instructie komt oorspronkelijk van de 4510-CPU van de Commodore 65, welke ontworpen was voor een adresruimte van 1MB.

De 45GS02-CPU van de MEGA65 breidt het MAP-register uit met de mogelijkheid om elk blok van 256 bytes in de 28-bits adresruimte te benaderen. Dit gebeurt in de vorm van een extra hoge byte (twee nibbles) voor elk van MAPLO en MAPHI. Deze hoge byte dient als bits 20-27 van het uiteindelijke adres.

$335F	+	$45200	=	$4855F
16-bits adres		offset		gemapt adres

$80	→	$804855F
hoge byte		uiteindelijk adres

Merk op dat de hoge MAP-byte de bits 20-27 van het eindadres instelt. Levert de optelling van de offset bij het 16-bits adres een waarde op die groter is dan 20 bits, dan "rolt" de optelling over binnen het 20-bits gedeelte van het adres en heeft zij geen invloed op de hoge byte van het eindadres.

U stelt een hoge MAP-byte in door de instructie **MAP** twee keer aan te roepen voordat u **EOM** aanroept. De eerste aanroep stelt de hoge byte in en de tweede aanroep stelt, zoals eerder, de selectievlaggen en de resterende 12 bits van de offset in. De computer schakelt de interrupts automatisch uit tussen de eerste **MAP**-aanroep en de **EOM**-aanroep.

Om de hoge byte in te stellen, laadt u de hoge byte in het A-register voor MAPLO of in het Y-register voor MAPHI. Laadt daarna de speciale waarde $0F in het X-register voor MAPLO of in het Z-register voor MAPHI. Roep **MAP** aan om de hoge byte in te stellen. Laadt vervolgens de selectievlaggen en de overige offsetwaarden in, roep **MAP** aan en roep tot slot **EOM** aan.

Om bijvoorbeeld $A000 - $BFFF in het zolder-RAM te mappen naar $8000000 - $8001FFF, gebruikt u een offset van $F6000 en een hoge byte van $80. Stelt u MAPHI in op deze offset met de selectievlag $2, dan past de computer de offset op het gewenste blok toe.

```
LDA #$00
LDX #$00
LDY #$80  ; MAPHI hoge byte van $80
```

```
LDZ #$0F  ; (Bijzondere waarde om de hoge byte in te stellen)
MAP

LDA #$00
LDX #$00
LDY #$60  ; MAPHI-offset: $F60
LDZ #$2F  ; MAPHI-selectievlag: $2 = A000 - BFFF
MAP
EOM
```

Let op: wijzig de hoge byte van het 32KB-adresgebied waarin de code staat die het mappen uitvoert, niet! De instelling van de hoge byte wordt actief bij de eerste MAP-instructie en als de code per ongeluk wordt weggemapt, kan de programmateller de overige instructies niet meer bereiken, zelfs niet als de uiteindelijke mapping compatibel is.

OPMERKING: De hoge MAP-bytes behouden hun waarde totdat ze opnieuw worden bijgewerkt met deze procedure van twee MAP-instructies. Door de procedure van één MAP-instructie worden ze *niet* op nul gezet. De hoge MAP-bytes hebben bij een herstart van de CPU de beginwaarde nul en de KERNAL zet deze bytes niet op een waarde ongelijk aan nul, zodat een programma er in de meeste gevallen vanuit kan gaan dat ze aan het begin van het programma nul zijn. Het volgende voorbeeld zet alle MAP-offsetbytes op nul.

```
LDA #$00
LDX #$0F
LDY #$00
LDZ #$0F
MAP         ; MAPHI- en MAPLO-megabyte-offsets zijn nu nul
            ; Schakel interrupts uit

LDA #$00
LDX #$00
LDY #$00
LDZ #$00
MAP         ; MAPHI- en MAPLO-bankoffsets zijn nu nul
EOM         ; Einde MAP, schakel interrupts in
```

In de meeste gevallen is het voor programma's eenvoudiger om het geheugen op hogere adressen te benaderen via 28-bits adresseermethoden of via DMA. Deze uitgebreide MAP-voorziening is opgenomen voor de volledigheid.

Het MAP-register lezen

De 45GS02-CPU kan het MAP-register instellen met een combinatie van CPU-instructies, maar er is geen manier om het register op dezelfde wijze te lezen als andere CPU-registers. Een programma kan hiervoor echter een Hypervisor-trap gebruiken. De hypervisor-trap hyppo_get_mapping schrijft de registerwaarden van MAPHI en MAPLO, inclusief de megabyte-offsets, naar zes bytes op een opgegeven plaats in het geheugen.

De geheugenplaats waarnaar de hypervisor-trap deze waarden schrijft, moet op een paginagrens tussen 0.0000 en 0.7E00 liggen. U stelt het Y-register in op de meest significante byte van het beginadres wanneer u de trap aanroept. Om bijvoorbeeld het MAP-register naar het RAM op de adressen 0.6200 - 0.6205 te kopiëren:

```
LDY #$62

LDA #$74
STA $D640
CLV
```

Al het chip-RAM beschikbaar maken

Met de technieken die in dit hoofdstuk worden beschreven, kunt u het geheugensysteem zodanig configureren dat al het chip-RAM beschikbaar komt op de adressen 0.0000 - 5.FFFF. Een zorgvuldig geschreven programma kan de volledige machine overnemen, I/O-registers op hogere adressen benaderen en al het chip- en zolder-RAM gebruiken.

Een overzicht van de procedure:

1. Schakel de interrupts uit.
2. Wis alle bits van $D030.
3. Zet het MAP-register volledig op nul.
4. Wis de bits 0, 1 en 2 van $0001 om de I/O-registers uit het bereik $D000 - $DFFF te verwijderen.
5. Verschuif het begin van het kleurengeheugen met ten minste $0800 (2KB) vooruit, zodat het venster op het kleurengeheugen als RAM voor algemeen gebruik kan dienen.
6. Hef de ROM-schrijfbeveiliging voor de banken 2 en 3 op met de trap ROM Write-enable (`LDA #$02 : STA $D641 : CLV`).
7. Laad aangepaste interrupt-handlercode in bank 0, stel de interruptvectoren in en schakel de interrupts daarna weer in.

Om de eerste 2KB als geheugen voor algemeen gebruik te kunnen benutten, mag het programma alleen weergavemodi gebruiken die ten hoogste 30KB aan kleurgegevens vereisen. Een programma dat alle 32KB van het kleurengeheugen gebruikt, moet de eerste 2KB zichtbaar laten in het venster op het kleurengeheugen op 1.F800 - 1.FFFF. (Het is niet vereist om dit geheugen in te mappen op $D800 - $DFFF.)

Het programma moet MAP gebruiken om het RAM op 0.0000 en 0.0001 te benaderen, door MAP in te schakelen voor $0000 - $1FFF met een offset van 0.

Waarom het SYS-commando slechts bepaalde adressen kan gebruiken

Eerder in dit hoofdstuk is uitgelegd dat het BASIC-commando **SYS** tegenintuïtieve beperkingen kent op de adressen die het kan benaderen. Dat kunnen we nu volledig verklaren, met behulp van de geavanceerdere begrippen uit dit hoofdstuk.

Voor BASIC-functies en -commando's zoals **PEEK** en **POKE** gebruikt de KERNAL de voorziening voor 32-bits indirecte adressering van de 45GS02 om grote adressen te benaderen. Bij een adres van $10000 of hoger laadt het commando het volledige adres in een variabele op de Base-Page en roept het vervolgens de 32-bits indirecte adressering aan om het geheugen te benaderen. Bij een adres van $0000 - $FFFF met een **BANK**-instelling van 0 - 5 gebruikt het commando de **BANK**-instelling als het bovenste deel van het adres. De 32-bits indirecte adressering negeert het MAP-register en de voorzieningen voor geheugenbanking en kan elk adres in de computer benaderen.

Is het adres $0000 - $FFFF en is de **BANK**-instelling 128, dan gebruikt BASIC de 16-bits adresseermethode van de CPU, welke de MAP-registerinstelling van de KERNAL respecteert, evenals de D030-instelling en de instellingen voor geheugenbanking in C64-stijl. Zo benaderen die commando's de ROM en de I/O-registers met BANK 128.

SYS draagt de CPU op om code op een plaats in het geheugen uit te voeren. De programmateller van de CPU is een 16-bits register en kan niet, zoals de 32-bits indirecte adressering dat kan, los van het MAP-register en andere voorzieningen voor geheugenmapping grotere adressen benaderen. Als onderdeel van de KERNAL moet de KERNAL-ROM in het geheugen beschikbaar blijven op de 16-bits adressen $E000 - $FFFF, zodat interrupts afgehandeld kunnen worden. Ook moeten de KERNAL-variabelen benaderbaar blijven op de adressen $0000 - $1FFF, zodat de interrupthandlercode ze kan gebruiken. Code die door **SYS** wordt aangeroepen, kan zelf beslissen of de KERNAL functioneel blijft, maar **SYS** houdt de KERNAL actief voor het gebruikelijke geval waarin de machinetaal de besturing wil teruggeven aan een actief BASIC-programma of aan de READY.-prompt.

De KERNAL gebruikt het MAP-register om de ROM in bank 3 te benaderen. Om de ROM op $E000 - $FFFF te houden, stelt hij de MAPH-offset in op 3 en markeert dat geheugengebied als "gemapt". Er is slechts één offset die MAP voor alle 8KB-gebieden in $8000 - $FFFF kan gebruiken, zodat bij de aanvang van een **SYS**-aanroep de overige gebieden in dat bereik ofwel ongemapt moeten zijn en naar bank 0 verwijzen (of de D030/$01-banking gebruiken), ofwel naar bank 3 gemapt moeten zijn. De meest bruikbare keuze voor **SYS** is om de overige hogere gebieden ongemapt te laten.

Evenzo moet het gebied, om te zorgen dat adressen in het bereik $0000 - $1FFF blijven verwijzen naar het KERNAL-variabelengeheugen in bank 0, ofwel ongemapt zijn ofwel met een offset van nul gemapt zijn. Het ligt voor de hand dat **SYS** dat gebied ongemapt laat, zodat het voor de adressen $2000 - $7FFF andere banken kan benaderen en dat is dan ook wat het doet. **SYS** gebruikt het MAP-register om voor een adres waarvan het onderste woord $2000 - $7FFF is, hogere banken te benaderen, zowel bij gebruik van **BANK** als wanneer een groot adres wordt opgegeven.

Het MAP-register is veel flexibeler dan dit: het maakt het mogelijk om bijna elk adres te bereiken via de 16-bits adressen $2000 - $7FFF, met behulp van een breed scala aan mogelijke offsets. De machinetaal die door **SYS** wordt aangeroepen, moet dan wel zodanig geschreven zijn dat zij weet welke offset er wordt gebruikt. De programmateller zou daarbij relatief zijn ten opzichte van de offset en een JMP-instructie in de code zou een adres moeten gebruiken dat op dezelfde offset is gebaseerd. Het is wel mogelijk om machinetaal te schrijven met uitsluitend relatieve spronginstructies (zoals BRA), maar historisch gezien heeft het commando **SYS** dat nooit vereist. In plaats daarvan beperkt **SYS** zich tot offsets van 64KB, om het programmeermodel eenvoudig te houden.

Hoe SYS TO werkt

Het commando **SYS TO** kan een pagina-offset toepassen op een adres in het bereik $2000 - $7FFF, om zo meer geheugenplaatsen te benaderen. Nu we weten dat MAP hierbij betrokken is, is dit eenvoudig uit te leggen: het argument 0 stelt de MAP-offset in voor de onderste helft van de 16-bits adresruimte, waarbij $2000 - $7FFF als te mappen gebied is geselecteerd. Met de pagina-offset kunt u code overal in de eerste 384KB van het geheugen benaderen, zolang de routine zodanig is geschreven dat deze rekening houdt met deze MAP-instelling.

```
REM : MAP $2000-$7FFF NAAR $54000-$59FFF,
REM : DAARNA AANROEP $6100 + $52000 = $58100

SYS TO $6100,0($520)
```

Standaard schakelen **SYS** en **SYS TO** BASIC uit terwijl de machinetaalsubroutine wordt uitgevoerd. Dit omvat het wijzigen van MAP, zodat de KERNAL-ROM niet langer op $2000 - $7FFF en $8000 - $BFFF is gemapt, alsmede het instellen van een interne vlag voor de KERNAL-IRQ-handler, zodat deze geen BASIC-IRQ-voorzieningen meer bedient, zoals **PLAY** en **MOVSPR**. Wanneer de machinetaal terugkeert, herstelt **SYS** de MAP-instelling van KERNAL en schakelt de BASIC-IRQ-voorzieningen weer in.

Het tijdelijk uitschakelen van de BASIC-IRQ-voorzieningen kan voor een BASIC-programma dat van deze voorzieningen afhankelijk is een kortstondige onderbreking veroorzaken, zoals muziekweergave die hapert. Om dit te voorkomen, biedt het commando **SYS TO** een bedrijfsmodus waarin BASIC tijdens de subroutine actief blijft. Het argument I1 schakelt deze modus in. Wanneer deze modus is ingeschakeld, geeft **SYS TO** de KERNAL-IRQ-handler opdracht om voor het uitvoeren van de BASIC-functies over te schakelen naar de MAP van KERNAL en daarna terug te schakelen naar de MAP van uw programma.

```
SYS TO $1600,I1
```

SYS TO kan niet tegelijkertijd een pagina-offset gebruiken en BASIC actief houden, omdat deze voorzieningen het MAP-register op verschillende manieren gebruiken.

DEEL VI

HARDWARE

HOOFDSTUK 16

Nexys4-borden gebruiken als een MEGA65

- Uw eigen MEGA65-compatibele computer bouwen
- Werkende Nexys4-borden
- Stroom, jumpers, schakelaars en knoppen
- Toetsenbord
- MicroSDHC-kaart voorbereiden
- De bitstream laden vanuit QSPI
- Widget-bord
- PMOD-naar-joystick-adapter

- **Handige tips**

UW EIGEN MEGA65-COMPATIBELE COMPUTER BOUWEN

U kunt uw eigen MEGA65-compatibele computer bouwen met een Nexys4DDR-bord (ook wel Nexys A7) of een ouder Nexys4-bord (zonder DDR). Deze bijlage beschrijft het proces om een Nexys4DDR (Nexys A7)-bord hiervoor in te stellen (het nieuwere, bij voorkeur te gebruiken bord). Het oudere non-DDR Nexys4-bord wordt ook ondersteund en de instructies zijn hetzelfde, met uitzondering van het feit dat u een bitstream moet gebruiken die voor dat bord ontworpen is. Het gebruik van een Nexys4DDR-bitstream op een non-DDR Nexys4-bord of andersom kan onherstelbare schade aan uw bord veroorzaken. Zorg er daarom voor dat u de juiste bitstream voor uw bord heeft.

DISCLAIMER: M.E.G.A kan geen enkele verantwoordelijkheid aanvaarden voor schade die aan uw Nexys4DDR/NexysA7/Nexys4-borden kan ontstaan.

WERKENDE NEXYS4-BORDEN

Er zijn momenteel drie Nexys FPGA-borden die als MEGA65 ingesteld kunnen worden:

Het Nexys4-bord

Niet meer geproduceerd, maar nog wel verkrijgbaar via sommige websites met oude voorraad.



Documentatie:

- https://reference.digilentinc.com/reference/programmable-logic/nexys-4/reference-manual
- https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4/nexys4_rm.pdf

Het Nexys4DDR-bord

Niet meer geproduceerd, maar nog wel verkrijgbaar via sommige websites met oude voorraad.



Documentatie:

- https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual
- https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4-ddr/nexys4ddr_rm.pdf

De Nexys A7

Dit is het bovenstaande Nexys4 DDR-bord, opnieuw uitgebracht onder een andere naam:

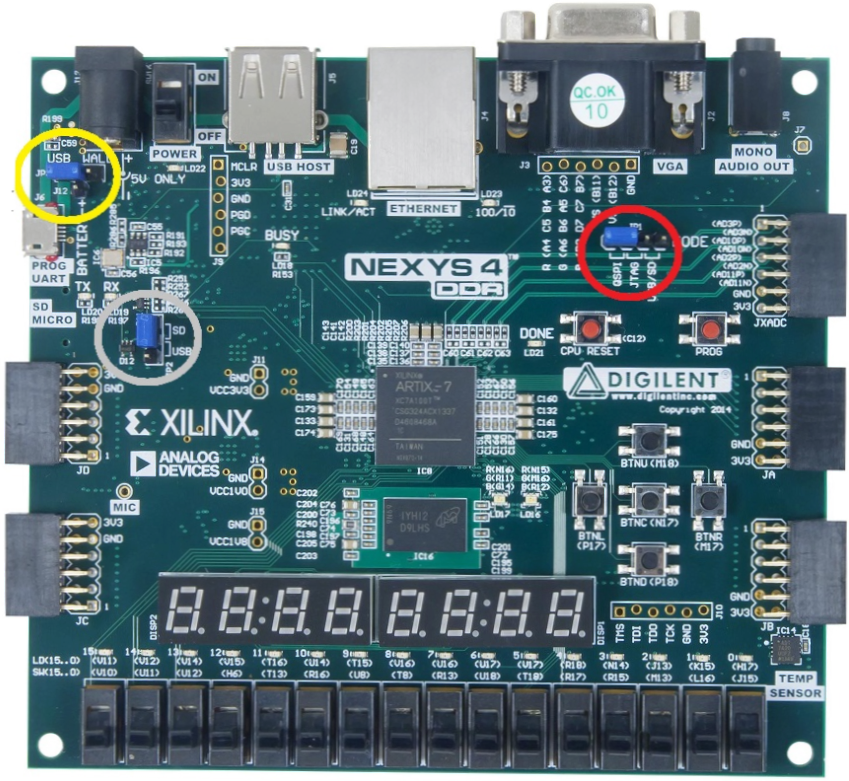


Documentatie:

- https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual
- https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-a7/nexys-a7_rm.pdf

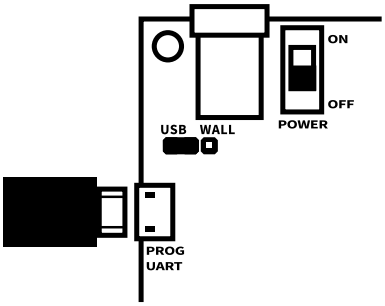
STROOM, JUMPERS, SCHAKELAARS EN KNOPPEN

Deze foto van bovenaf toont de belangrijkste jumperposities op het Nexys4-bord:



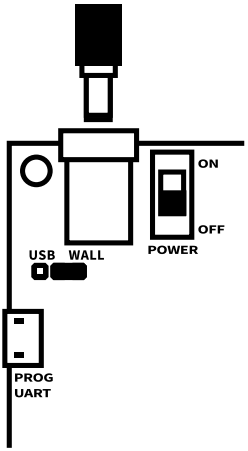
De Nexys4-borden kunnen op twee manieren van stroom worden voorzien: via een externe voeding of via een standaard USB-poort.

Micro-USB-voeding



Sluit uw micro-USB-kabel aan op een USB-poort van een USB-lader of pc om stroom te leveren. Sluit het andere uiteinde aan op de micro-USB-connector van de Nexys4. Plaats de JP3-jumper op pennen 1 en 2 om USB-voeding te selecteren. Gebruik de schakelaar om de Nexys4 in te schakelen.

Externe voeding



De MEGA65-core kan veel stroom verbruiken en een standaard USB-poort kan in potentie te weinig vermogen leveren voor het Nexys4-bord. Het schrijven naar de SD-kaart kan met name vastlopen of vreemd gedrag vertonen. U wordt dan aangeraden een voeding van 5V te gebruiken.

Digilent verkoopt een voeding voor het Nexys4-bord en wij raden u aan deze te gebruiken om het risico op schade aan uw Nexys4-bord te vermijden. De gekozen voeding moet een center-positieve stekker met een binnendiameter van 2.1 mm hebben en 4.5VDC tot 5.5VDC leveren met een stroomsterkte van ten minste 1 ampère.

Sluit de voedingskabel aan op de voedingsaansluiting van de Nexys4. Plaats de JP3-jumper op pennen 2 en 3 om netvoeding te selecteren. Gebruik de schakelaar om de Nexys4 in te schakelen.

Overige jumpers en schakelaars

Voor uw eerste installatie raden wij u aan de volgende jumpers op uw Nexys4-bord in te stellen:

- JP1 - USB/SD
- JP2 - SD

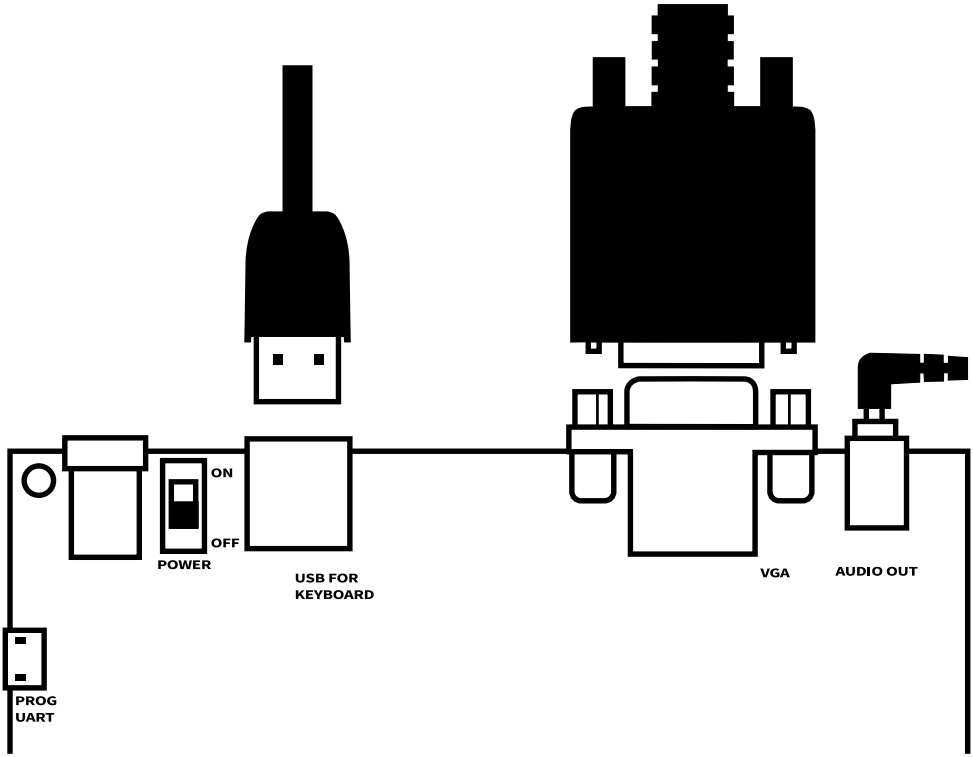


Dit zorgt ervoor dat de bitstream-bestanden bij het opstarten vanaf uw SD-kaart geladen worden.

Op een later moment kunt u ervoor kiezen de bitstream vanaf de ingebouwde QSPI-flash te laden. In dat geval kunt u de instelling van de JP1-jumper aanpassen naar de QSPI-positie.

Alle 16 schakelaars aan de onderkant van het bord moeten op de uit-positie staan.

Aansluitingen en randapparatuur



Aan de USB-poort kan een USB-toetsenbord worden aangesloten. Alleen een toetsenbord zonder USB-hub werkt met het Nexys4-bord. Over het algemeen werken zeer goedkope toetsenborden, terwijl duurdere toetsenborden vaak een USB-hub bevatten en niet werken. Het zou kunnen voorkomen dat u meerdere toetsenborden moet proberen voordat u er een vindt die werkt.

Aan de VGA-poort kan een VGA-beeldscherm worden aangesloten.

De mono-audio-uitgang kan worden aangesloten op de line-in van een versterker.

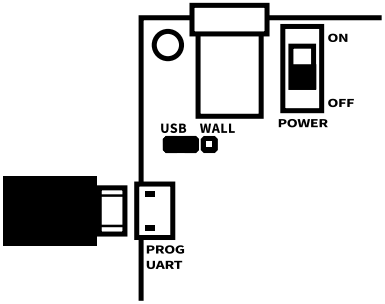
Communicatie met uw pc

Er kunnen situaties zijn waarin u vanaf uw pc met het Nexys4-bord wilt communiceren, bijvoorbeeld om de volgende activiteiten uit te voeren:

- Het QSPI-flashgeheugen flashen via Vivado
- Bitstream-bestanden rechtstreeks vanaf uw pc uploaden (via het m65-hulpprogramma)

- Hulpprogramma's gebruiken zoals M65Connect, m65, mega65_ftp, m65dbg, enz.

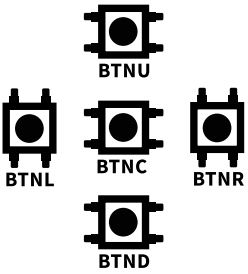
In zulke gevallen moet u de micro-USB-kabel op uw pc aansluiten.



Knoppen op het bord



De "CPU RESET"-toets reset de MEGA65 wanneer u deze indrukt, terwijl de "PROG"-toets de FPGA zelf de MEGA65-core laat herladen. Het belangrijkste verschil tussen de twee is dat CPU RESET sneller is en de inhoud van het geheugen niet wist, terwijl de FPGA-knop langzamer is en de inhoud van het geheugen wel reset.



Twee van de vijf toetsen in de kruisopstelling kunnen eveneens worden gebruikt: BTND gedraagt zich alsof u RESTORE heeft ingedrukt, terwijl BTNC een interrupt activeert, alsof de IRQ-lijn naar massa is getrokken.

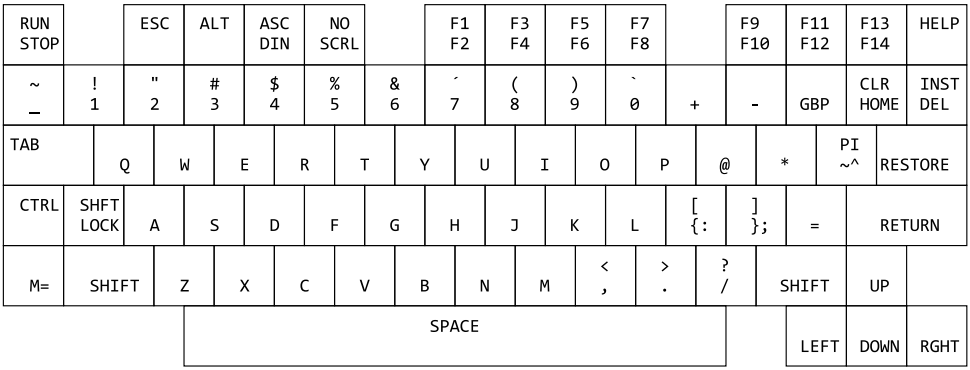
TOETSENBORD

De toetsenbordindeling is positioneel en niet logisch. Dit betekent dat toetsen die zich in dezelfde positie bevinden als de toetsen op een C65-toetsenbord een vergelijkbare

functie hebben. Deze relatie gaat ervan uit dat uw USB-toetsenbord een Amerikaanse toetsenbordindeling gebruikt.

Om u te helpen bij het vinden van de toetstoewijzingen van de diverse MEGA65-toetsen, beschikt de MEGA65 over een ingebouwde virtuele toetsenbordtest. Deze is op twee manieren te bereiken.

De eenvoudigste manier is om ALT ingedrukt te houden terwijl u de Nexys4 inschakelt, of de Nexys4 reset met de "PROG"-toets. Het configuratiemenu wordt weergegeven en door op 3 te drukken, verschijnt het virtuele toetsenbord op een zwarte achtergrond.



Wanneer u een toets op het USB-toetsenbord indrukt, wordt de betreffende toets op het virtuele toetsenbord gemarkeerd weergegeven om u te helpen bij het herkennen van de toetstoewijzing.

De andere manier om het virtuele toetsenbord te bereiken is vanuit de MEGA65 zelf. Houd ingedrukt en druk op TAB om de Matrix Mode Debugger te openen. Voer vervolgens het volgende in:

```
s ffd3615 ff
```

Dit opent een halfdoorzichtig virtueel toetsenbord aan de bovenkant van het scherm. Als alternatief:

```
s ffd3615 ff ff
```

Dit opent een halfdoorzichtig virtueel toetsenbord in het midden van het scherm.

Houd ingedrukt en druk op TAB om de Matrix Mode Debugger te verlaten en terug te keren naar de MEGA65.

Enkele toetstoewijzingen met een USB-toetsenbord

RESTORE is toegewezen aan de Page Up-toets.

RUN STOP is toegewezen aan ESC.

MICROSDHC-KAART VOORBEREIDEN

De MEGA65 vereist een SDHC-kaart met een capaciteit van 4 GB tot 64 GB. Sommige SDXC-kaarten kunnen werken, maar dit wordt niet officieel ondersteund.

De voorbereidingsstappen voor de SD-kaart van het Nexys4-bord komen grotendeels overeen met de stappen die nodig zijn voor echte MEGA65-hardware. Het is dan ook de moeite waard om een kijkje te nemen in Hoofdstuk/Bijlage 5 op bladzijde 5-3 als u ooit meer details nodig heeft.

In dit onderdeel bespreken we de specifieke stappen uitgebreider en beperken we ons tot een beknopte behandeling van de gemeenschappelijke stappen.

Een verschil tussen het Nexys-bord en de echte MEGA65-hardware is dat laatstgenoemde standaard een bitstream/core bevat waarmee u uw SD-kaart in de specifieke indeling kunt formateren die door de MEGA65 wordt vereist.

Voor eigenaren van het Nexys4-bord is er echter geen standaard bitstream; raadpleeg daarom het **grote Mega65-boek** voor meer informatie over waar u de juiste bestanden “nexys4.bit of “nexys4ddr-widget.bit voor uw apparaat kunt downloaden.

Voorbereidingsstappen

De stappen zijn:

- Formateer de SD-kaart op een handige computer met het FAT32-bestandssysteem. De MEGA65- en Nexys4-borden ondersteunen geen andere bestandssystemen, met name niet het exFAT-bestandssysteem.
- Kopieer uw bitstream-bestand (met een bestandsnaam die eindigt op “.bit”) naar de SD-kaart.
- Plaats de SD-kaart in de SD-kaartsleuf aan de onderkant van het Nexys4-bord.
- Schakel het Nexys4-bord in.
- Open het hulpprogrammamenu door ALT ingedrukt te houden op het USB-toetsenbord dat op het Nexys4-bord is aangesloten wanneer het MEGA65-startscherm wordt weergegeven.
- Open het FDISK/FORMAT-hulpprogramma door op 2 te drukken wanneer de optie verschijnt op het MEGA65-opstartscherm.
- Volg de aanwijzingen in het FDISK/FORMAT-programma om de SD-kaart opnieuw te formatteren voor gebruik door de MEGA65.

  Het FDISK-hulpprogramma maakt twee partities op uw SD-kaart aan en formatteert deze.

  - De ene is type $41 = MEGA65-systeempartitie, waar de opslagslots, configuratiegegevens en andere bestanden staan.
    (Deze partitie is onzichtbaar op bijvoorbeeld Windows-pc's).
  - De andere partitie met type $0C = VFAT32, waar KERNAL, ondersteuningsbestanden, spellen enzovoort later naartoe worden gekopieerd.
    (Deze partitie is zichtbaar op bijvoorbeeld Windows-pc's).

- Zodra het formatteren is voltooid, schakelt u het Nexys4-bord uit en verwijdert u de microSDHC-kaart uit het Nexys-bord en plaatst u deze terug in uw pc
- Kopieer deze keer de volgende items naar de SD-kaart:
  - Het bitstream-bestand
  - De uitgepakte bestanden uit het "**SD essentials.rar**"-of het "**SD essentialsNoROM.rar**"-bestand dat u van de MEGA65-filehost hebt gedownload.
  - Als u uw eigen voorkeurs-ROM-bestand hebt (bijvoorbeeld "**911001.BIN**"), kopieer dit ook naar de SD-kaart en hernoem het naar "**MEGA65.ROM**" (hoofdletters zijn verplicht).
  - Eventuele .D81-schijfimage-bestanden waarvan u gebruik wilt maken.
    * Let op: wanneer een bestand met de naam MEGA65.D81 aan de SD-kaart wordt toegevoegd, wordt dit automatisch gemonteerd bij het opstarten.
    * Zorg ervoor dat alle .D81-bestanden namen hebben die voldoen aan de oude DOS 8.3-tekenlimiet en in hoofdletters zijn geschreven. Deze beperking wordt in een toekomstige versie verwijderd.
- Verwijder de SD-kaart en plaats deze terug in uw Nexys4-bord.
- Schakel het Nexys4-bord weer in. De MEGA65 zou binnen 15 seconden moeten opstarten.
- "Bij de eerste opstart bevindt u zich op het onboarding-scherm, waarvan meer details te vinden zijn in Hoofdstuk/Bijlage 5 op bladzijde 5-3 .

Gefeliciteerd. Uw MEGA65 is ingesteld en gereed voor gebruik.

Houd er rekening mee dat bij de bovenstaande methode het bitstream-bestand naar de SD-kaart wordt gekopieerd, waardoor de Nexys-FPGA de bitstream bij elke opstartcyclus laadt; het opstarten duurt dan ongeveer 13 seconden. De bitstream kunt u ook met Vivado-software naar het QSPI-flashgeheugen flitsen; de opstarttijd verkort u daarmee tot 0.3 seconden.

Voor meer gedetailleerde informatie over het voorbereiden en configureren van uw MEGA65 verwijzen wij u naar Hoofdstuk/Bijlage 5 op bladzijde 5-3 hoofdstuk.

DE BITSTREAM LADEN VANUIT QSPI

Hoewel het laden van de bitstream vanaf de SD-kaart de aanbevolen (en veelbewandelde) route is die dit document heeft gekozen, hebben de laatste tijd meer Nexys4-gebruikers de alternatieve weg verkend om de bitstream vanuit het QSPI-flashgeheugen te laden. Enkele mogelijke redenen hiervoor zijn:

- Snellere opstarttijden (0,3 seconden versus 13 seconden)
- Sommige gebruikers waren geïnteresseerd in de mogelijkheid om meerdere cores op hun QSPI te flitsen (via stappen die beschreven staan in de Hoofdstuk/-Bijlage 6 op bladzijde 6-5 Hoofdstuk)
- Sommige gebruikers hebben last gehad van hardnekkige problemen met de route via de SD-kaart, zoals:
  - Het systeem kan niet opnieuw opstarten vanaf het onboarding-scherm
  - Het systeem kan niet opnieuw opstarten vanuit het freeze-menu na het wisselen tussen PAL/NTSC

Indien dit in de toekomst een populairdere route blijkt te zijn, kunnen wij onze documentatie hierop aanpassen. Hier volgen enkele stappen in het kort.

Voorbereidingsstappen

Als u deze route wilt proberen, moet u de JP1-jumperinstelling aanpassen om QSPI te gebruiken en vervolgens de stappen volgen in het hoofdstuk Het flashen van de FPGA's en CPLD's in de MEGA65 met betrekking tot Vivado installeren en De hoofd-FPGA flashen met Vivado.

Houd er rekening mee dat de installatie van Vivado een langdurig proces is (zowel wat betreft de downloadtijd als de installatietijd).

Nadat u Slot0 van uw QSPI-chip hebt geflitst via Vivado, kunt u de stappen volgen die beschreven staan in Hoofdstuk/Bijlage 5 op bladzijde 5-3 voor de aangepaste SD-kaartformattering, het installeren van ROM- en ondersteuningsbestanden en de onboarding.

WIDGET-BORD

Als u een Nexys-bord bezit (alle modellen), vindt u het wellicht interessant om een widget-bord aan uw Nexys-apparaat toe te voegen, zodat u verbinding kunt maken met:

- een originele C64- of C65-toetsenbord
- 2 DB-9-joysticks

- Paddles of een 1351-muis
- Cartridges (werken nog niet)

Het widget-bord wordt via de PMOD-connectoren met uw Nexys-bord verbonden.

Momenteel is het alleen verkrijgbaar als onbewerkte printplaat (zonder componenten), hier aan te schaffen:

- https://oshpark.com/shared_projects/Y37xg9N7

De firmware, de printplaatdiagrammen, het schema en de stuklijst zijn in het volgende GitHub-project te vinden:

- https://github.com/sy2002/DM65PIC

U moet de onderdelen uit de stuklijst apart aanschaffen en de componenten zelf op het bord monteren.

Dit forum64.de-thread kan voor u van belang zijn wanneer u de ervaringen van anderen die dit proces hebben doorlopen wilt lezen of wanneer u zelf vragen heeft:

- https://www.forum64.de/index.php?thread/90465-mega-65-ports-add-on-card-s

De printplaatbestanden (c65Keyb.brd) en schematische bestanden (c65Keyb.sch) bevinden zich in de map "eagle/" van het DM65PIC-project en kunnen worden bekeken met het gratis programma AutoDesk Eagle, hier beschikbaar:

- https://www.autodesk.com/products/eagle/free-download

Hier volgen enkele foto's van het widget-bord in gebruik:






MEGA=65

Voor het gemak zijn de printplaat- en schema's van het widget-bord eveneens opgenomen in Hoofdstuk/Bijlage AE op bladzijde AE-75.

Enkele aanvullende achtergrondnotities over het bord:

- Het widget-bord is oorspronkelijk bedoeld om in een C65-behuizing te passen
- De uitbreidingspoort is niet bruikbaar en er moet een stuk worden weggezaagd voor de C64-behuizing
- Het zou redelijk zijn het bord aan te passen voor gebruik in een C64-behuizing (wat zeker het meest voorkomende gebruik zal zijn)

PMOD-NAAR-JOYSTICK-ADAPTER

Als alternatief (en goedkoper) voor wie alleen een DB9-joystick-poort via de PMOD-connectoren wil toevoegen, heeft een community-lid, TheChief, een manier bedacht om dat te doen. Meer informatie is te vinden in het volgende Discord-thread:

- https://discord.com/channels/719326990221574164/903079038015389716/929369283119685643



Joystick-actie	PMOD-JA-aansluitingen	DB9-aansluitingen
Vuur	jahi(9)	pin 6
Omhoog	jalo(1)	pin 1
Links	jalo(2)	pin 3
Omlaag	jahi(7)	pin 2
Rechts	jahi(8)	pin 4

HANDIGE TIPS

Hier volgen enkele handige tips om vertrouwd te raken met de MEGA65:

- Druk op & houd  (of de Commodore-toets als u een Commodore 64- of 65-toetsenbord gebruikt) ingedrukt tijdens het opstarten om in C64-modus op te starten in plaats van C65-modus.
- Druk op & houd  ingedrukt tijdens het opstarten om de machine-taalmonitor te openen in plaats van BASIC te starten.
- Druk op  en houd deze ongeveer een halve tot een seconde ingedrukt om het MEGA65 Freeze-menu te openen. Dit menu biedt handige hulpmiddelen om de processorsnelheid te wijzigen, te wisselen tussen PAL & NTSC-videomodus, audio-instellingen aan te passen, freeze-states te beheren en D81-schijfimages te selecteren. U kunt bovendien het geheugen van het bevroren programma inspecteren en bewerken, en nog veel meer. Dit menu is in vele opzichten het hart van de MEGA65 en is het ontdekken zeker waard.
- Typ `POKE0,65` in C64-modus om de processor op volle snelheid (40 MHz) te zetten. Sommige software gedraagt zich in deze modus onjuist, terwijl andere software juist sneller werkt.
- Typ `POKE0,64` in C64-modus om de processor op 1 MHz te zetten.
- Typ `SYS58552` in C64-modus om naar C65-modus te schakelen.
- Typ `GO64` in C65-modus en bevestig door `Y` in te drukken om naar C64-modus te schakelen, hetzelfde als op een C128.
- De C65-ROM maakt apparaat 8 tot standaard, zodat u het **,8** aan het einde van LOAD- en SAVE-commando's normaal gesproken kunt weglaten.
- Het indrukken van  +  in de C64- of C65-modus probeert van schijf op te starten.

Veel plezier! Veel plezier! Er is jarenlang met liefde aan de MEGA65 gewerkt voor uw plezier. Wij hopen dat u er net zoveel plezier aan beleeft als wij bij het maken hebben gehad!

Het MEGA Museum of Electronic Games & Art verwelkomt uw feedback, suggesties en bijdragen aan dit open-source-project voor het bewaren van digitaal erfgoed.

DEEL VII

PLATFORMONAFHANKELIJKE ONTWIKKELHULPMIDDELEN

HOOFDSTUK 17

Emulators

- De Xmega65-emulator gebruiken
- De Live ISO-image gebruiken

Op het moment van schrijven is er slechts één emulator voor de MEGA65, **xmega65**; de Xemu-emulatorsuite van LGB. De LGB-ontwikkelaars werken hard om bij te blijven met de ontwikkeling van de MEGA65; niettemin is sommige MEGA65-emulatie mogelijk niet nauwkeurig, maar zou deze voldoende moeten zijn voor softwareontwikkeling op de MEGA65.

Test software tijdens de ontwikkeling regelmatig op echte hardware, zoals een MEGA65 of een FPGA-board dat een MEGA65-core kan uitvoeren.

Download de broncode van de MEGA65-emulator van https://github.com/lgblgblgb/xemu.

Download voorgecompileerde versies van https://github.lgb.hu/xemu/. Installers zijn beschikbaar voor macOS, Windows en Linux.

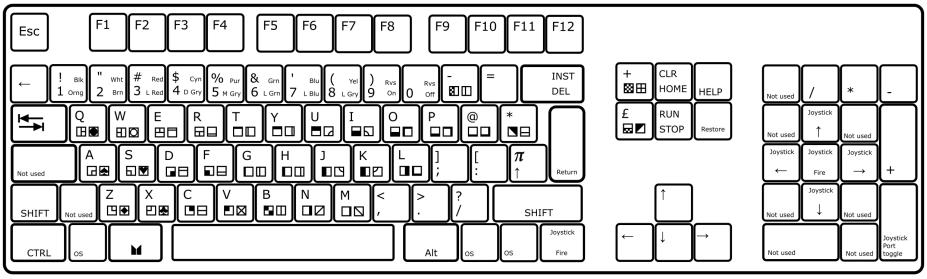
Een live ISO-image met de emulator, documentatie en andere hulpprogramma's is beschikbaar via Forum64.de op https://www.forum64.de/index.php?thread/104698-xemu-live-system-iso-file/&postID=1549927#post1549936. Indien u deze wilt gebruiken, vindt u de instructies op bladzijde 17-5.

DE XMEGA65-EMULATOR GEBRUIKEN

Het toetsenbord

De MEGA65-toetsenbordindeling is gebaseerd op de Commodore 65, wat afwijkt van een modern pc-toetsenbord. In de meeste gevallen herkent de Xmega65-emulator pc-toetsen als hun equivalente MEGA65-toetsen, zoals letters, cijfers en veelgebruikte modificatietoetsen zoals Shift. Xmega65 gaat uit van een Amerikaanse (VS-)indeling.

Sommige toetsen komen alleen op het MEGA65-toetsenbord voor en hebben geen pc-equivalenten. Xmega65 gebruikt de pc-toetsen Insert, Delete, Home, End, Page Up en Page Down voor +, £, CLR HOME, RUN STOP, HELP en RESTORE, respectievelijk.



Op de meeste pc-toetsenborden staan RUN STOP en RESTORE naast elkaar, zodat deze toetscombinatie eenvoudig is in te voeren.

Deze afbeelding toont de PETSCII-glyphen zoals ze op een MEGA65-toetsenbord verschijnen, met de -glyphe links en de Shift-glyphe rechts. Merk op dat op een pc-toetsenbord de toets voor rechts van de Shift staat.

Xemu biedt twee joysticks (een per keer) op het numerieke toetsenbord. Enter op het numerieke toetsenbord wordt gebruikt om bij herhaaldelijk drukken poorten 1 en 2 in en uit te schakelen.

Wees u ervan bewust dat de emulator standaard F9 F10 F11 als eigen sneltoetsen heeft gedefinieerd. Als u deze toetsen wilt gebruiken met de geëmulieerde MEGA65, moet u de toetsenbordindeling aanpassen. De bestanden staan in de map met de Xemu-instellingen zoals uitgelegd op bladzijde 17-5. Dit doet u door **keymap-default.cfg** te kopiëren naar **keymap.cfg** en vervolgens de volgende regels te verwijderen:

```
XEMU-EXIT F9
XEMU-FULLSCREEN F11
```

en te wijzigen in:

```
F9 Unknown
F11 Unknown
```

naar:

```
F9 F9
F11 F11
```

Nadat deze toetsen opnieuw zijn toegewezen, zijn de functies voor volledig scherm en afsluiten nog steeds beschikbaar via het menu. Klik met rechts (of Ctrl-klik) ergens in het venster om dit te openen.

Problemen met starten op Apple MacOS

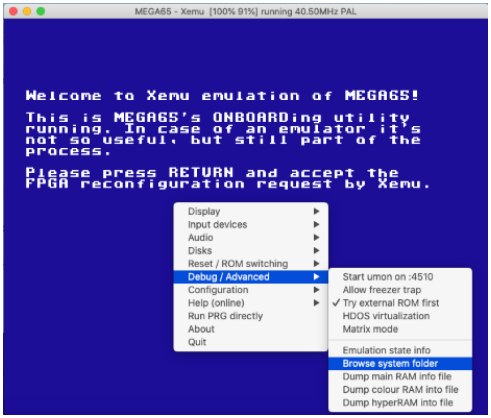
Als het bericht **macOS cannot verify that this app is free from malware** verschijnt, gebruik dan de volgende procedure wanneer u de app voor de eerste keer opent:

- Open Finder op uw Mac.
- Ctrl-klik (of rechtsklik) op het Xmega65-app-pictogram.
- Selecteer in het contextmenu **Open**.

- Het bericht verschijnt opnieuw, maar deze keer is er een knop **Open**. Klik op **Open**. U hoeft deze procedure slechts de eerste keer uit te voeren. Vanaf nu kunt u de app normaal openen.

Instellingen bijwerken

U kunt het Xemu-instellingsbestand bewerken om de toetsenbordindeling aan te passen of een reservekopie van het SD-kaart-imagebestand te maken. Open vanuit Xemu het menu om deze bestanden te openen: selecteer menu, selecteer **Debug / Advanced** en vervolgens **Browse system folder**. Bewerk de bestanden **keymap-default.cfg** en **mega65-default.cfg** met een teksteditor. Voor specifieke instructies over het wijzigen van **keymap-default.cfg** verwijzen we naar bladzijde 17-3.



De locatie van de Xemu-systeemmap verschilt per besturingssysteem:

- Windows (kan in de adresbalk van de verkenner worden gekopieerd): **%appdata%\xemu-lgb\mega65**
- Linux: **~/.xemu-lgb/**
- Macintosh: **~/Library/Application\ Support/xemu-lgb/mega65/**

DE LIVE ISO-IMAGE GEBRUIKEN

De Live ISO-image is het product van een vrijwillige gemeenschap; niet van het MEGA65-team. We nemen deze op voor uw gemak.

Een opstartbare USB-stick of DVD maken

Er zijn veel manieren om een live ISO-image te maken. De methode die u kiest hangt af van uw besturingssysteem en of u deze op een USB-station wilt installeren of op

een DVD wilt branden. Het branden op een DVD is eenvoudig, ervan uitgaande dat u een computer met een DVD-brander hebt. Als u een sneller opstartbaar USB-station wilt maken, probeert u een van de onderstaande methoden:

Als u Windows gebruikt, overweeg dan een hulpprogramma zoals http://www.isotousb.com/.

Op Linux kunt u de instructies op https://fossbytes.com/create-bootable-usb-media-from-iso-ubuntu/ volgen.

Voor Apple Macs zijn de instructies op https://ubuntu.com/tutorials/create-a-usb-stick-on-macos#1-overview beschikbaar.

Vergelijkbare instructies zijn beschikbaar voor andere populaire computers, zoals Amiga's (https://forum.hyperion-entertainment.com/viewtopic.php?t=3857) of Sun UltraSPARC-werkstations (https://forums.servethehome.com/index.php?threads/how-to-create-a-bootable-solaris-11-usb.1998/).

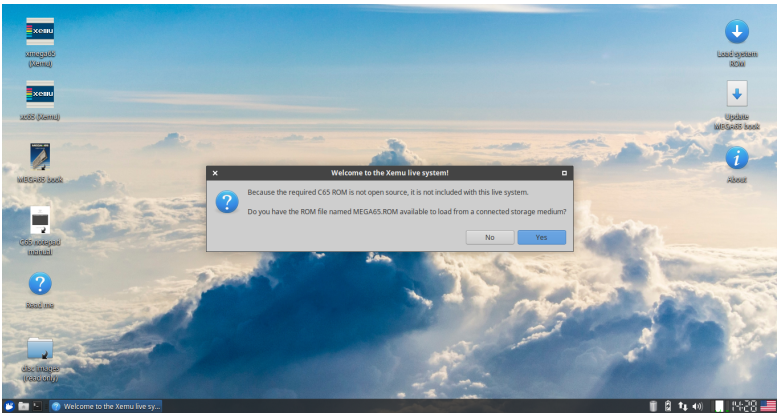
Tot slot is de populaire, eenvoudig te gebruiken en gratis cross-platform belanaEtcher beschikbaar op https://www.balena.io/etcher/.

Aan de slag

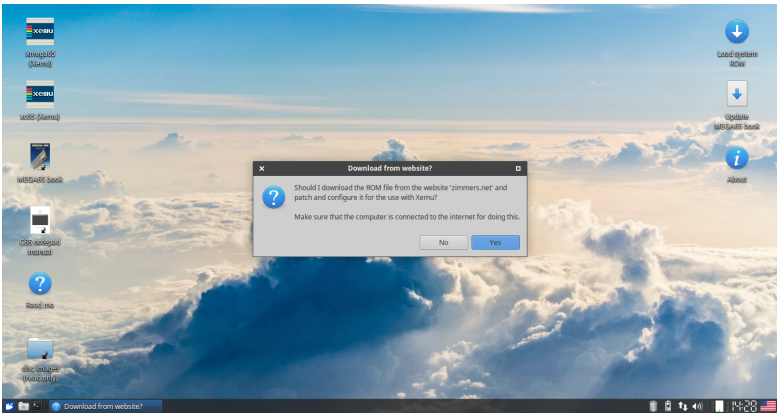
Om potentiële auteursrechtelijke kwesties te vermijden, bevat de opstartbare ISO-image geen propriëtaire ROM's voor de MEGA65, zoals oude C65-ROM's. Het bevat wel een vervangende open-source-ROM van ons OpenROMs-project. Deze ROM start op in een BASIC 2-omgeving waarmee u veel C64-programma's kunt laden en uitvoeren, zoals te zien is in de onderstaande afbeelding:



Als u een C65-ROM met BASIC 10 wilt gebruiken, download het juiste ROM-bestand en plaats het op een andere USB-stick met de naam **MEGA65.ROM**. Bij het opstarten zal de MEGA65 vragen of een ROM is gedownload; zoals te zien is in de onderstaande afbeelding:



Als de Live ISO geen ROM kan vinden, zal het u vragen om een ROM te downloaden; zoals hieronder te zien is:



Andere functies van de Live ISO

Zoals de bovenstaande schermafbeeldingen laten zien, biedt de Live ISO diverse handige bureaubladkortingen. Aan de linkerzijde staan koppelingen om de MEGA65-emulator en de C65-emulator te starten, zodat u kunt testen of programma's op beide platforms werken. Zoals eerder vermeld zijn beide emulators in ontwikkeling en mogelijk niet 100% compatibel.

Een andere koppeling geeft toegang tot het grote Mega65-boek. Deze omvattende band van ongeveer 800 pagina's bevat de officiële MEGA65-documentatie. Het grootste deel van deze handleiding voor ontwikkelaars is ook opgenomen in het grote Mega65-boek.

Deze ISO bevat ook documentatie voor de C65 Notepad; een programma voor de C65 en MEGA65 geschreven door Snoopy (de ontwikkelaar van de Live ISO-image). Een "lees mij"-bestand bevat verdere informatie over de Live ISO.

Tot slot zijn er aan de rechterzijde koppelingen om een C65-ROM te downloaden en het grote Mega65-boek bij te werken naar de nieuwste versie. Hiermee voorkomt u dat u steeds een nieuwe opstartbare image moet maken wanneer er een frequente update van het grote Mega65-boek wordt uitgebracht.

De volledige inhoud van de Live ISO-image is toegankelijk via de bestandsverkenner.

HOOFDSTUK 18

Gegevensoverdracht en foutopsporing

- m65-hulpprogramma
- M65Connect
- mega65_ftp
- TFTP-server
- Een BASIC-tekstbestandlisting omzetten naar een PRG-bestand

De sleutel tot effectieve cross-platform-ontwikkeling is het hebben van snelle en eenvoudige middelen om programmatuur op de MEGA65 te testen. Dit geldt zeker nu de MEGA65-emulator nog in ontwikkeling is. Zelfs wanneer de MEGA65-emulator voltooid is, zal het onwaarschijnlijk zijn dat deze volledige compatibiliteit op volledige snelheid kan bieden, omdat de MEGA65 veel veeleisender is om te emuleren dan de C64.

Er zijn verschillende hulpprogramma's die kunnen worden gebruikt voor gegevensoverdracht en foute opsporing. Deze maken doorgaans gebruik van ofwel de seriële monitorinterface van de MEGA65, of via de fast-ethernetadapter van de MEGA65. De seriële monitorinterface is beschikbaar via de UART-lijnen op de JB1- connector.

Als u geen toegang hebt tot de seriële monitorinterface, worden er hulpprogramma's ontwikkeld voor de fast-ethernetpoort die een deel, maar niet alles, van de mogelijkheden van de seriële monitor bieden. Deze worden gedocumenteerd zodra ze beschikbaar zijn. Het restant van dit hoofdstuk richt zich op methoden die de seriële monitorinterface gebruiken.

U kunt een 3.3V UART-adapter op de juiste lijnen aansluiten, of gemakkelijker, een TE-0790-03 JTAG-debugmodule op deze connector plaatsen. Dit geeft u een USB-verbinding die kan worden gebruikt voor het overzetten van software, op afstand fouten opsporen en geheugen inspecteren, alsmede het activeren of flashen van bitstreams. Met deze verbinding zijn de volgende hulpprogramma's beschikbaar:

M65-HULPPROGRAMMA

Het repository https://github.com/mega65/mega65-tools bevat een aantal hulpprogramma's, utilities en voorbeeldprogramma's. Deze hulpprogramma's zijn voornamelijk voor Linux maar kunnen worden gebruikt op Windows met Cygwin. Een van die hulpprogramma's is het **m65**-hulpprogramma. Dit is eigenlijk een zakmes verzameling van utilities in één. Veelgebruikte functies zijn onder meer:

Schermfoto's met het m65-hulpprogramma

Om een schermfoto van de MEGA65 te maken:

```
m65 -S
```

Dit maakt een bestand aan met de naam **mega65-screen-000000.png**, of als dit bestand al bestaat, wordt het eerste niet-gebruikte nummer gebruikt in plaats van **000000**.

Merk op dat deze schermfotofunctie werkt door **m65** de functionaliteit van de VIC-IV te laten emuleren. Hoewel het uitstekende digitale schermfoto's oplevert, komt het mogelijk niet exact overeen met het werkelijke scherm van de MEGA65. Op het moment van schrijven rendert het geen sprites of bitplanelen, alleen tekst- en bitmapgebaseerde videomodi.

Een programma laden en uitvoeren op de MEGA65

Om een programma op de MEGA65 te laden en uit te voeren, kunt u een commando als het volgende gebruiken:

```
m65 -F -4 -r foo.prg
```

De optie **-F** draagt **m65** op de MEGA65 te herstarten voordat het programma wordt geladen.

De optie **-4** draagt **m65** op de MEGA65 naar C64-modus te schakelen voordat het programma wordt geladen. Als dit wordt weggelaten, zal het proberen het programma in C65-modus te laden.

De optie **-r** draagt **m65** op het programma onmiddellijk na het laden uit te voeren.

Merk op dat dit commando werkt met het normale BASIC LOAD-commando en daarom beperkt is tot het laden van programma's in het onderste 64 KB van het geheugen.

De FPGA herconfigureren om een andere bitstream uit te voeren

Om een andere MEGA65-bitstream uit te proberen, kan een commando als het volgende worden gebruikt:

```
m65 -b bitstream.bit
```

Dit zal de genoemde bitstream naar de FPGA zenden. Omdat de FPGA door deze actie wordt herconfigureerd, zal het momenteel uitgevoerde programma niet alleen worden gestopt, maar ook het hoofdgeheugen zal worden gewist. Voor modellen van de MEGA65 die zijn uitgerust met 8 MB of 16 MB uitbreidingsgeheugen, worden deze uitbreidingsgeheugens geïmplementeerd in externe chips en dus zullen de inhoud daarvan niet worden gewist.

Voor niet-MEGA65-bitstreams (zoals zxunomega65 en gbc4mega65), gebruik in plaats daarvan het argument '-q':

```
m65 -q bitstream.bit
```

Toetsenbordinvoer op afstand

De toetsenbordinterface van de MEGA65 ondersteunt het overzetten van synthetische toetsgebeurtenissen via de registers $D615 - $D617. Het **m65**-hulpprogramma gebruikt dit om typen op afstand op de MEGA65 mogelijk te maken op een manier die transparant is voor software. Er zijn drie manieren om dit te gebruiken:

```
m65 -t ditiswattekst
```

Deze vorm typt de opgegeven tekst, in dit geval *ditiswattekst*, maar simuleert niet het indrukken van RETURN. Als u het indrukken van RETURN wilt simuleren, gebruikt u **-T** in plaats van **-t**, bijvoorbeeld:

```
m65 -T list
```

Dit zou het LIST-commando typen en uitvoeren.

Tot slot is het mogelijk om algemene toetsenbordbesturing op afstand te starten via:

```
m65 -t -
```

In deze modus wordt elke toets, die op het toetsenbord van de computer waarop **m65** draait, doorgegeven aan de MEGA65. Merk op dat niet alle speciale toetsen worden ondersteund en dat er enige vertraging is, dus het gebruik van de toetsherhalingsfunctie kan onverwachte resultaten opleveren. Voor besturing op afstand in het algemeen is het echter een zeer nuttige voorziening.

Unit-testing en logboekondersteuning

Het **m65**-hulpprogramma ondersteunt unit-testing direct op MEGA65-hardware. Wanneer de *unit testing-modus* actief is, wacht **m65** tot de MEGA65 bepaalde bytereeksen via de seriële interface stuurt die de huidige status (gestart, geslaagd, mislukt) signaleren van een gegeven test. Daarnaast is het mogelijk om logboekberichten te verzenden vanaf de MEGA65 naar de hostcomputer.

De unittestingmodus wordt betreden door **m65** aan te roepen met de vlag **-u**. Om een BASIC-programma op afstand in C65-modus uit te voeren en tegelijk **m65** in de *unittestingmodus* te zetten, kan het volgende commando worden gebruikt:

```
    m65 -Fur attic-ram.prg -w tests.log
```

De opties **-F** en **-r** vertellen **m65** om de MEGA65 te herstarten voordat het programma "attic-ram.prg" wordt geladen en automatisch uitgevoerd. De optie **-u** vertelt **m65** vervolgens om in de *unittestingmodus* te gaan in plaats van te stoppen na het starten van het programma. De optionele optie **-w** zorgt ervoor dat **m65** de testresultaten aan het bestand "test.log" toevoegt (het bestand aanmaakt als het nog niet bestaat).

Merk op dat **m65** automatisch de *unittestingmodus* verlaat als er gedurende meer dan 10 seconden geen teststatussignalen zijn ontvangen.

Er is ondersteuning voor het verzenden van unittestingsignalen naar de hostcomputer vanuit C- en BASIC 65-programma's:

Unit-tests gebruiken met C

De MEGA65 libc bevat ondersteuning voor unit-testing via functies gedefinieerd in **tests.h** en **tests.c**.

Om het begin van een test te signaleren, includeert u **tests.h** en gebruikt u:

```
unit_test_setup("testName",issueNumber);
```

waarbij *"testName"* een leesbare naam van de test is (bijvoorbeeld "VIC-II") en *issueNumber* een verwijzing naar het bijbehorende bug-issue (bijvoorbeeld het issue-nummer van GitHub).

Na het starten van een test kunnen geslaagde tests worden gesignaleerd met de functie unit_test_ok():

```
unit_test_ok();
```

Een mislukte test wordt gesignaleerd met unit_test_fail():

```
unit_test_fail("fail message");
```

Elke keer dat de functies unit_test_ok() of unit_test_fail() worden aangeroepen, wordt het *sub-issue* van de test (gerapporteerd op de hostcomputer) verhoogd. Dit maakt het eenvoudiger om meerdere tests in één bestand te combineren en te identificeren.

U kunt willekeurige logboekberichten verzenden via unit_test_log():

```
unit_test_log("hello world from mega65!");
```

...en ten slotte, wanneer alles klaar is, wordt het einde van unit-testing gesignaleerd door het gebruik van:

```
unit_test_done();
```

Unit-tests gebruiken met BASIC 65

b65support.bin is een machinetaalmodule die ondersteuning biedt voor unit-testing vanuit BASIC 65. Deze staat in het mega65-tools- repository op https://github.com/MEGA65/mega65-tools en kan worden gebouwd met:

```
make bin65/b65support.bin
```

Het bestand **b65support.bin** is dan beschikbaar in de map **bin65**.

Deze module werkt door de USR-vector om te leiden om de functies uit te voeren die nodig zijn om te communiceren met de test-host.

In een geautomatiseerd testsenario wilt u mogelijk het binaire bestand **b65support.bin** overzetten naar het MEGA65-geheugen door middel van **m65**:

```
m65 -@ mega65-tools/bin65/b65support.bin@15fe
```

Het is natuurlijk ook mogelijk om **b65support.bin** rechtstreeks vanaf de MEGA65 te laden door de **M65UTILS.D81**-image van de Freezer te koppelen en het volgende commando te geven:

```
BLOAD "B65SUPPORT.BIN"
```

Na het laden wordt **b65support.bin** geïnitialiseerd met:

```
SYS $1600
```

Na initialisatie biedt **b65support.bin** de volgende functies:

```
A=USR((issueNum))
```

bereidt een nieuwe test voor met nummer <issueNum> en herstelt het subissue-nummer naar 0.

```
A=USR("=(testName)")
```

stelt de testnaam in en stuurt het testsignaal; bijvoorbeeld: **A=USR("=VIC-III")** stelt de testnaam in op 'VIC-III' en signaleert de hostcomputer dat de test is gestart.

```
A=USR("/(logMessage)")
```

stuurt een logboekbericht naar de hostcomputer.

```
A=USR("P")
```

stuurt het signaal 'geslaagd' naar de hostcomputer en verhoogt het subissue-nummer.

```
A=USR("F")
```

stuurt het signaal 'test mislukt' naar de hostcomputer en verhoogt het subissue-nummer.

```
A=USR("D")
```

stuurt het signaal 'test voltooid' naar de hostcomputer.

Alle aanroepen retourneren het huidige subissue-nummer of **?ILLEGAL QUANTITY ERROR** in geval van een ongeldig commando.

BASIC 65-voorbeeld

Het volgende is een volledig BASIC 65-voorbeeld dat laat zien hoe de unittesting-functies van **m65** worden gebruikt:

```
100 rem attic ram cache test
110 poke $bfffff2,$e0          : rem schakel cache van zolder-ram in
120 sys $1600                  : rem initialiseer testmodule
130 a=usr(379)                 : rem zet issue-nummer
140 a=usr("=attic-ram-cache")  : rem zet testnaam
150 bank128:poke0,65           : rem uit voorzorg
160 b0=$8000000 : b1=$8000100 : rem zolden-ram-gebieden om te testen
170 for r=0 to $ff
180   poke b0+r,0              : rem vul gebied 1 met 0
190   poke b1+r,$ff            : rem vul gebied 2 met $ff
200 next r
210 for t=1 to 10              : rem 10 pogingen
220   poke b0,32               : rem schrijf naar b0
230   for x=0 to $ff:t1=b1+x
240     a=peek(b0)             : rem lees uit b0
250     b=peek(t1):b=peek(t1)  : rem lees tweemaal uit t1
260     ifb()255 thenf=t:t=11:x=256   : rem dit mag niet gebeuren
270   next x
280 next t
290 if f=0 then begin
300   print "geen fouten gevonden na";t;"pogingen."
310   a=usr("p")               : rem signal 'test passed' to host
320 bend : else begin
330   a=usr("f")               : rem signal 'test failed' to host
340   print "hyper-ram-fout gedetecteerd na";f;"pogingen."
350   print "peek($";hex$(t1);") [t1] is";b;"maar moet 255 zijn"
360 bend
370 a=usr("d")                 : rem test gereed
```

M65CONNECT

Dit is een cross-platform grafisch hulpprogramma dat beschikbaar is voor Windows, Linux en MacOSX, waarmee toegang wordt geboden tot de meeste functies van het **m65**-hulpprogramma, zonder dat een commandoregel nodig is, of zonder dat u het hulpprogramma voor uw favoriete besturingssysteem hoeft te compileren.

Het repository voor M65Connect is: https://github.com/MEGA65/m65connect

De nieuwste binaire versie is beschikbaar op https://files.mega65.org.

Met de MEGA65 of Nexys FPGA uitgeschakeld, sluit een USB-kabel aan van uw computer op de MEGA65 of Nexys FPGA-board. Voer het *M65Connect*-programma uit en volg de aanwijzingen om verbinding te maken. Het programma helpt u bij het identificeren van welke USB Serial Port moet worden gebruikt.

Met dit hulpprogramma kunt u eenvoudig PRG-programma's en diverse andere bestanden overdragen. M65Connect kan de overdracht, het schakelen naar C64-modus en de uitvoering van programma's afhandelen.

MEGA65_FTP

Het **mega65_ftp**-hulpprogramma van het repository https://github.com/mega65/mega65-tools is enigszins misleidend genoemd: hoewel het een File Transfer Program is, gebruikt het niet het File Transfer Protocol (FTP). Het gebruikt in plaats daarvan de seriële monitorinterface om de MEGA65 op afstand te besturen en heeft directe toegang tot de SD-kaart om het kopiëren van bestanden tussen de MEGA65 en de hostcomputer mogelijk te maken.

Merk op dat het de staat van de MEGA65 niet perfect herstelt bij het afsluiten, en dus alleen moet worden gebruikt wanneer de MEGA65 bij de READY.-prompt staat, zodat eventuele draaiende software niet in de war raakt. Met name moet u het gebruik vermijden wanneer een gevoelig programma draait, zoals het Freeze-menu, het MEGA65-configuratiehulpprogramma of het MEGA65 Format/FDISK-hulpprogramma. (Dit probleem kan met weinig inspanning worden opgelost, als iemand de tijd en interesse heeft om het te fixen).

Bij uitvoering biedt het een FTP-achtige interface die de commando's **get**, **put**, **rename** en **dir** ondersteunt. Merk op dat u bij het plaatsen van een bestand moet zorgen dat het een naam heeft die volledig in hoofdletters staat en een DOS-compatibele 8.3-teken bestandsnaam heeft. Dit komt door beperkingen in zowel **mega65_ftp** als de VFAT32-bestandssysteemcode van de Hypervisor van de MEGA65. Opnieuw kunnen deze problemen worden opgelost met een bescheiden hoeveelheid inspanning van de kant van een gemotiveerd lid van de gemeenschap.

Tot slot is het programma **mega65_ftp** *zeer* langzaam bij het pushen van nieuwe bestanden naar de MEGA65, met doorgaans snelheden van rond de 5 KB/sec. Dit komt deels doordat de seriële interface in staat is om gegevens over te dragen met slechts 40 KB/sec (bij 4.000.000 bits per seconde), en deels doordat het besturingsproces op afstand resulteert in veel heen-en-weerverkeer waarbij hulpoutines op de MEGA65 worden uitgevoerd om sectoren op de SD-kaart te lezen, schrijven en verifiëren. Het zou heel haalbaar zijn om dit te verbeteren tot bijna 40 KB/sec, en mogelijk sneller met een combinatie van gegevenscompressie, deduplicatie van identieke sectoren (met name bij het uploaden van schijf- images) en andere technieken. Opnieuw zou dit een zeer welkome bijdrage zijn die iemand uit de gemeenschap zou kunnen leveren ten algemene nutte.

TFTP-SERVER

Er is een echte TFTP-server voor de MEGA65 in ontwikkeling die snelle TFTP-overdrachten via de 100 Mbit-ethernet ondersteunt en die kan worden gebruikt om

zeer snel bestanden van de MEGA65 te lezen. Snelheden van bijna 1 MB/sec zijn mogelijk, afhankelijk van de prestaties van de SD-kaart. In plaats van DHCP te gebruiken, zal dit hulpprogramma reageren op *elk* IP-adres dat eindigt op .65. Het gebruikt altijd het MAC-adres 40:40:40:40:40:40. Echte DHCP- ondersteuning en het gebruik van het geconfigureerde ethernet-MAC-adres van de MEGA65 worden mogelijk in de toekomst toegevoegd.

Belangrijker nog, de ondersteuning voor het schrijven van bestanden naar de SD-kaart is nog niet volledig en wordt geblokkeerd door de behoefte aan de implementatie van de nodige functies in de Hypervisor van de MEGA65 voor het aanmaken en vergrootten van bestanden. Een bijzondere uitdaging is het mogelijk maken van het aanmaken van bestanden met aaneengesloten clusters zoals vereist voor D81-schijfimages: als een D81-bestand gefragmenteerd is, kan het niet worden gekoppeld, omdat het koppelingsmechanisme een pointer naar het aaneengesloten blok van de SD-kaart vereist dat de schijfimage bevat. In de tussentijd kan **`mega65_ftp`** als alternatief worden gebruikt.

EEN BASIC-TEKSTBESTANDLISTING OMZETTEN NAAR EEN PRG-BESTAND

Als u een ongetokeniseerd BASIC-programma in platte-tekstformaat hebt dat afkomstig is van bijvoorbeeld een internetbericht, en u wilt het op de MEGA65 proberen zonder het in te typen, dan is het mogelijk het om te zetten naar een PRG.

C64List is een Windows-gebaseerd commandoregelhulpprogramma waarmee u de conversie kunt uitvoeren. Zodra u een *.PRG*-bestand hebt, kunt u een hulpprogramma zoals M65Connect gebruiken om het naar de MEGA65 of Nexys FPGA te uploaden.

C64List is beschikbaar voor download op `http://www.commodoreserver.com/Downloads.asp`

Zorg ervoor dat u een programmalijst in een bestand op uw lokale computer hebt opgeslagen (bijvoorbeeld *program.txt*) gecodeerd als ANSI of UTF8.

Gebruik C64List om het bestand om te zetten naar een PRG-bestand:

```
C64List program.txt -prg
```

Nu kunt u uw nieuw geconverteerde programma uploaden naar de MEGA65 met M65Connect of een van de eerder beschreven hulpprogramma's.

Het is vermeldenswaard dat deze methode niet 100% effectief zal zijn bij listings met speciale PETSCII-tekens. Programma's met PETSCII vereisen enige bewerking op de MEGA65 zelf voordat ze op schijf worden opgeslagen.

HOOFDSTUK 19

Assemblers

Onderstaande tabel geeft een overzicht van assemblers waarvan bekend is dat ze met de MEGA65 werken. Voor algemeen gebruik raden we **ACME** aan, omdat het goede ondersteuning biedt voor de 45GS02-instructieset, open source is en bovendien in C is geschreven. Dat laatste betekent dat de assembler op termijn naar de MEGA65 geporteerd zou kunnen worden om er native te draaien.

Naam	45GS02	Bron	Verwijzing
ACME	ja	C	https://sourceforge.net/projects/acme-crossass
KickAss	ja	Java	gitlab.com/jespergravgaard/kickassembler65ce02
LLVM-MOS	ja	C++	https://llvm-mos.org/wiki/Welcome
Ophis	ja	Python	https://github.com/michaelcmartin/Ophis
BSA	ja	C	https://github.com/Edilbert/BSA
CA65	nee[1]	C	https://github.com/mega65/cc65

De **BSA**-assembler wordt momenteel gebruikt om de **MEGA65.ROM** te bouwen. Het grootste deel van de broncode is geschreven in de syntaxis van de oude **BSO**-assembler (Boston Systems Office), die in de jaren 1989-1991 werd gebruikt door softwareontwikkelaars aan de C65. De **BSA**-assembler heeft een compatibiliteitsmodus waarmee deze oude broncodes met weinig of geen aanpassingen kunnen worden geassembleerd. De **BSA**-assembler bevat momenteel alleen een beschrijving van de commando's in de C-broncode van de assembler.

Het **LLVM-MOS**-project wordt geleverd met een GNU-compatibele macro-assembler en -disassembler die alle mnemonieken van de 45GS02 ondersteunt. Het kan als zelfstandig hulpprogramma worden gebruikt met het commando **llvm-mc**.

[1] Onze fork van CA65 (onderdeel van CC65) herkent de CPU van de MEGA65 correct, maar heeft geen expliciete ondersteuning voor de mogelijkheden van de processor

HOOFDSTUK 20

C- en C-achtige compilers

- MEGA65 libc

Kort antwoord: CC65 en KickC werken beide op de MEGA65.

Zowel CC65 als KickC werken op de MEGA65, maar beide hebben standaard alleen een C64-geheugenmodel en gebruiken uitsluitend 6502-opcodes. Het zou geweldig zijn als iemand een C65-geheugenconfiguratie voor CC65 zou ontwikkelen; dat zou niet al te moeilijk moeten zijn.

CC65 ondersteunt overlays, wat krachtig kan worden ingezet met het extra geheugen van de MEGA65 om programma's groter dan 64 KB mogelijk te maken. Daarvoor is echter een geschikte loader nodig, die nog niet bestaat.

Op dezelfde manier zou het aanpassen van de codegenerator van CC65 om 45GS02-mogelijkheden te gebruiken niet bijzonder moeilijk zijn en zou het helpen om de anders vreselijk trage en omvangrijke code die CC65 produceert te overwinnen. Het toevoegen van volledige ondersteuning voor de 45GS02-CPU-mogelijkheden in CA65 (of nog beter: het laten produceren van ACME-compatibele assembleeruitvoer door CC65) zou van overweldigend voordeel zijn en niet bijzonder moeilijk te realiseren. Dit zouden allemaal geweldige taken zijn om mee bezig te zijn terwijl u wacht op uw MEGA65 DevKit!

Een voorbeeldsjabloon voor een C-programma dat met CC65 kan worden gecompileerd en op de MEGA65 kan worden uitgevoerd, staat in het repository `https://github.com/MEGA65/hello-world`. Dit repository kan zelfs CC65 downloaden en compileren als u dat nog niet op uw systeem hebt geïnstalleerd. Het repository werkt op Linux en Mac en op Windows via het Windows Subsystem for Linux (WSL).

Het LLVM-MOS-project (`https://llvm-mos.org`) heeft een MEGA65-doelplatform dat is getest met C99, C++11 en Rust. De LLVM MOS-compiler heeft ondersteuning voor 45GS02-instructies (`-mcpu=mos45gs02`) en wordt geleverd met een assembler en disassembler. Voorgecompileerde binaire bestanden van de SDK zijn beschikbaar voor Linux, Windows en macOS.

MEGA65 LIBC

Er wordt een C-bibliotheek ontwikkeld voor de MEGA65, die al een aantal nuttige functies bevat. Deze bibliotheek is beschikbaar via `http://github.com/mega65/mega65-libc`. De procedures, functies en definities die het biedt, zijn gedocumenteerd in een apart hoofdstuk.

De MEGA65 libc is momenteel alleen beschikbaar voor LLVM en CC65, hoewel we een KickC-port hiervan zeer op prijs zouden stellen.

HOOFDSTUK 21

MEGA65 Standard C Library

- Structuur en gebruik
- conio.h

Er wordt een C-bibliotheek ontwikkeld voor de MEGA65, die al een aantal nuttige functies bevat. Deze bibliotheek is beschikbaar via http://github.com/mega65/mega65-libc. De procedures, functies en definities die zij biedt, worden gedocumenteerd in een apart hoofdstuk.

STRUCTUUR EN GEBRUIK

De MEGA65-libc wordt bewust alleen als broncode aangeboden, met functiegroepen in aparte bestanden en afzonderlijke headerbestanden voor inclusie. Het idee is dat u alleen de headerbestanden opneemt die u nodig heeft, en alleen de benodigde bronbestanden toevoegt aan de lijst van bronbestanden van het programma dat u compileert. Daarmee voorkomt u dat de compiler functies in uw gecompileerde programma opneemt die nooit worden gebruikt, waardoor kostbaar geheugen verspild wordt.

Let op dat sommige bronbestanden van de bibliotheek in C geschreven zijn en dus bestanden met de extensie `.c` betreffen, terwijl andere in assemblerertaal geschreven zijn, hetzij vanwege efficiëntie, hetzij uit noodzaak, en de extensie `.s` hebben.

Gangbaar gebruik is de broncode van mega65-libc uit te checken in een naburige map of binnen de bronmap van uw eigen project. In het laatste geval kan dit met behulp van de git submodule-faciliteit.

De volgende secties documenteren elk van de headerbestanden en de bijbehorende functies die zij bieden.

CONIO.H

conionit

Beschrijving: Initialiseert de interne status van de bibliotheek

Syntaxis: `void conioinit(void)`

Opmerkingen:
U moet deze functie aanroepen voordat u een conio-bibliotheekfunctie gebruikt.

setscreenaddr

Beschrijving: Stelt het startadres van de screen RAM in

Syntaxis: `void setscreenaddr(long addr);`

Parameters: **addr:** Het adres dat als begin van de screen RAM ingesteld moet worden

Opmerkingen:
Er wordt geen grenscontrole uitgevoerd op het geselecteerde adres

getscreenaddr

Beschrijving: Retourneert het startadres van de screen RAM

Syntaxis: `long getscreenaddr(void);`

Resultaat: Het huidige startadres van de screen RAM.

setcolramoffset

Beschrijving: Stelt de offsetwaarde van het begin van de color RAM in

Syntaxis: `void setcolramoffset(long offset);`

Parameters: **addr:** De offset vanaf het begin van de color RAM ($FF80000)

Opmerkingen:
Er wordt geen grenscontrole uitgevoerd op het resulterende adres. Overschrijd de beschikbare grootte van de color RAM niet

getcolramoffset

Beschrijving: Retourneert de offsetwaarde van het begin van de color RAM

Syntaxis: `long getscreenaddr(void);`

Resultaat: De huidige offsetwaarde van het begin van de color RAM.

setcharsetaddr

Beschrijving: Stelt het startadres van de tekenset in

Syntaxis: `void setcharsetaddr(long addr);`

Parameters: **addr:** Het adres dat als begin van de tekenset ingesteld moet worden

Opmerkingen:
Er wordt geen grenscontrole uitgevoerd op het geselecteerde adres

getcharsetaddr

Beschrijving: Retourneert het huidige startadres van de tekenset

Syntaxis: `long getscreenaddr(void);`

Resultaat: Het huidige startadres van de tekenset.

clrscr

Beschrijving: Maakt het tekstscherm leeg.

Syntaxis: `void clrscr(void)`

Opmerkingen:
De color RAM wordt met de huidige tekstkleur leeggemaakt

getscreensize

Beschrijving: Retourneert de afmetingen van het tekstscherm

Syntaxis: `void getscreensize(unsigned char* width, unsigned char* height)`

Parameters: **width:** Aanwijzer naar de locatie waar de breedte wordt teruggegeven

height: Aanwijzer naar de locatie waar de hoogte wordt teruggegeven

setscreensize

Beschrijving: Stelt de afmetingen van het tekstscherm in

Syntaxis: `void setscreensize(unsigned char width, unsigned char height)`

Parameters: **width:** De breedte in kolommen (40 of 80)

height: De hoogte in rijen (25 of 50)

Opmerkingen:
Momenteel worden alleen 40/80 en 25/50 aanvaard. Andere waarden worden genegeerd.

set16bitcharmode

Beschrijving: Schakelt de 16-bits tekenmodus in of uit

Syntaxis: `void set16bitcharmode(unsigned char f)`

Parameters: **f:** Stel true in om de 16-bits tekenmodus in te schakelen

Opmerkingen:
Dit leidt tot een reset van de videoparameters indien HOTREG is ingeschakeld. Zie de functie sethotregs.

sethotregs

Beschrijving: Schakelt het hot-registergedrag van de VIC-IV-chip in of uit.

Syntaxis: `void set16bitcharmode(unsigned char f)`

Parameters: **f:** Stel true in om het hotreg-gedrag in te schakelen

Opmerkingen:
Indien deze modus is ingeschakeld, wordt een reset van de videomodus geactiveerd bij toegang tot $D011, $D016, $D018, $D031 of de VIC-II-bankbits van $DD00.

setextendedattrib

Beschrijving: Schakelt de uitgebreide attributenmodus van de VIC-III in of uit ter ondersteuning van knipperen, onderstrepen, vet en markeren.

Syntaxis: `void setextendedattrib(unsigned char f)`

Parameters: **f:** Stel true in om de uitgebreide attributenmodus in te schakelen

togglecase

Beschrijving: Stelt de tekenset met kleine letters in

Syntaxis: `void setlowercase(void)`

togglecase

Beschrijving: Stelt de tekenset met hoofdletters in

Syntaxis: `void setuppercase(void)`

togglecase

Beschrijving: Wisselt de huidige tekenset tussen kleine en hoofdletters

Syntaxis: `void togglecase(void)`

bordercolor

Beschrijving: Stelt de huidige randkleur in

Syntaxis: `void bordercolor(unsigned char c)`

Parameters: **c:** De in te stellen kleur

bgcolor

Beschrijving: Stelt de huidige schermkleur (achtergrondkleur) in

Syntaxis: `void bgcolor(unsigned char c)`

Parameters: **c:** De in te stellen kleur

textcolor

Beschrijving: Stelt de huidige tekstkleur in

Syntaxis: `void textcolor(unsigned char c)`

Parameters: **c:** De in te stellen kleur

Opmerkingen:
Deze functie behoudt de attributen in de bovenste 4 bits indien uitgebreide attributen zijn ingeschakeld. Zie setextendedattrib.

revers

Beschrijving: Schakelt het reverse-attribuut in

Syntaxis: `void revers(unsigned char c)`

Parameters: **enable:** 0 om uit te schakelen, 1 om in te schakelen

Opmerkingen:
De uitgebreide attributenmodus moet actief zijn. Zie setextendedattrib.

highlight

Beschrijving: Schakelt het highlight-attribuut in

Syntaxis: `void highlight(unsigned char c)`

Parameters: **enable:** 0 om uit te schakelen, 1 om in te schakelen

Opmerkingen:
De uitgebreide attributenmodus moet actief zijn. Zie setextendedattrib.

blink

Beschrijving: Schakelt het knipperattribuut in

Syntaxis: `void blink(unsigned char c)`

Parameters: **enable:** 0 om uit te schakelen, 1 om in te schakelen

Opmerkingen:
De uitgebreide attributenmodus moet actief zijn. Zie setextendedattrib.

underline

Beschrijving: Schakelt het onderstrepingsattribuut in

Syntaxis: `void underline(unsigned char c)`

Parameters: **enable:** 0 om uit te schakelen, 1 om in te schakelen

Opmerkingen:
De uitgebreide attributenmodus moet actief zijn. Zie setextendedattrib.

altpal

Beschrijving: Schakelt het attribuut van het alternatieve palet in

Syntaxis: `void altpal(unsigned char c)`

Parameters: **enable:** 0 om uit te schakelen, 1 om in te schakelen

Opmerkingen:
De uitgebreide attributenmodus moet actief zijn. Zie setextendedattrib.

clearattr

Beschrijving: Wist alle tekstattributen

Syntaxis: `void clearattr())`

Opmerkingen:
De uitgebreide attributenmodus moet actief zijn. Zie setextendedattrib.

cellcolor

Beschrijving: Stelt de kleur van een tekencel in

Syntaxis: `void cellcolor(unsigned char x, unsigned char y, unsigned char c)`

Parameters: **x:** De X-coördinaat van de cel

y: De Y-coördinaat van de cel

c: De in te stellen kleur

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

setpalbank

Beschrijving: Stelt de huidige tekst-/bitmappaletbank in (BTPALSEL).

Syntaxis: `void setpalbank(unsigned char bank)`

Parameters: **bank:** De in te stellen paletbank. Geldige waarden zijn 0, 1, 2 of 3.

Opmerkingen:
Gebruik setpalbanka om een alternatief tekst-/bitmappalet in te stellen

setpalbanka

Beschrijving: Stelt de alternatieve tekst-/bitmappaletbank in.

Syntaxis: `void setpalbanka(unsigned char bank)`

Parameters: **bank:** De in te stellen paletbank. Geldige waarden zijn 0, 1, 2 of 3.

Opmerkingen:
Gebruik setpalbank om het hoofdtekst-/bitmappalet in te stellen

getpalbank

Beschrijving: Haalt de geselecteerde tekst-/bitmappaletbank op.

Syntaxis: `unsigned char getpalbank(void)`

Opmerkingen:
Gebruik getpalbanka om het geselecteerde alternatieve tekst-/bitmappalet op te halen

Resultaat: De huidige geselecteerde hoofdtekst-/bitmappaletbank.

getpalbanka

Beschrijving: Haalt de geselecteerde alternatieve tekst-/bitmappaletbank op.

Syntaxis: `unsigned char getpalbanka(void)`

Opmerkingen:
Gebruik getpalbank om het geselecteerde hoofdtekst-/bitmappalet op te halen

Resultaat: De huidige geselecteerde alternatieve tekst-/bitmappaletbank.

setmapedpal

Beschrijving: Stelt de ingemapte paletbank in op $D100-$D3FF.

Syntaxis: `void setmapedpal(unsigned char bank)`

Parameters: **bank:** De in te mappen paletbank. Geldige waarden zijn 0, 1, 2 of 3.

getmapedpal

Beschrijving: Haalt de ingemapte paletbank op $D100-$D3FF.

Syntaxis: `unsigned char getmapedpal(void)`

setpalentry

Beschrijving: Stelt een paletkleur in voor het ingemapte palet

Syntaxis: `void setpalentry(unsigned char c, unsigned char r, unsigned char g, unsigned char b)`

Parameters: **c:** De index van de paletkleur (0-255)

r: De rode kleurcomponent

g: De groene kleurcomponent

b: De blauwe kleurcomponent

Opmerkingen:
Gebruik setmapedmal om het te wijzigen palet in te mappen

fillrect

Beschrijving: Vult een rechthoekig gebied met een teken en een kleurwaarde

Syntaxis: `void fillrect(const RECT *rc, unsigned char ch, unsigned char col)`

Parameters: **rc:** Een RECT-structuur die de coördinaten van de rechthoek opgeeft

ch: Een tekencode om de rechthoek mee te vullen

col: De kleur om mee te vullen

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

box

Beschrijving: Tekent een kader met grafische tekens

Syntaxis: `void box(const RECT *rc, unsigned char color, unsigned char style, unsigned char clear, unsigned char shadow)`

Parameters: **rc:** Een RECT-structuur die de coördinaten van het kader opgeeft

color: De kleur die voor de grafische tekens gebruikt wordt

style: De stijl voor de randen van het kader. Kan worden ingesteld op BOX_STYLE_NONE, BOX_STYLE_ROUNDED, BOX_STYLE_INNER, BOX_STYLE_OUTER, BOX_STYLE_MID

clear: Zet op 1 om het binnenvlak van het kader te wissen met de geselecteerde kleur

shadow: Zet op 1 om een slagschaduw te tekenen

Opmerkingen:

Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

hline

Beschrijving: Tekent een horizontale lijn.

Syntaxis: `void hline(unsigned char x, unsigned char y, unsigned char len, unsigned char style)`

Parameters:

x: De X-coördinaat van het begin van de lijn

y: De Y-coördinaat van het begin van de lijn

len: De lengte van de lijn

style: De stijl voor de lijn. Zie de HLINE_-constanten voor de beschikbare stijlen.

Opmerkingen:

Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

vline

Beschrijving: Tekent een verticale lijn.

Syntaxis: `void vline(unsigned char x, unsigned char y, unsigned char len, unsigned char style)`

Parameters:

x: De X-coördinaat van het begin van de lijn

y: De Y-coördinaat van het begin van de lijn

len: De lengte van de lijn

style: De stijl voor de lijn. Zie de VLINE_-constanten voor de beschikbare stijlen.

Opmerkingen:

Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

gohome

Beschrijving: Stelt de huidige positie in op de thuispositie (coördinaat 0,0)

Syntaxis: `void gohome(void)`

gotoxy

Beschrijving: Stelt de huidige positie in op de coördinaten X,Y

Syntaxis: `void gotoxy(unsigned char x, unsigned char y)`

Parameters: **x:** De nieuwe X-coördinaat

y: De nieuwe Y-coördinaat

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

gotox

Beschrijving: Stelt de X-coördinaat van de huidige positie in

Syntaxis: `void gotox(unsigned char x)`

Parameters: **x:** De nieuwe X-coördinaat

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

gotoy

Beschrijving: Stelt de Y-coördinaat van de huidige positie in

Syntaxis: `void gotoy(unsigned char y)`

Parameters: **y:** De nieuwe Y-coördinaat

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

moveup

Beschrijving: Verplaatst de huidige positie naar boven

Syntaxis: `void moveup(unsigned char count)`

Parameters: **count:** Het aantal te verplaatsen posities

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

movedown

Beschrijving: Verplaatst de huidige positie naar beneden

Syntaxis: `void movedown(unsigned char count)`

Parameters: **count:** Het aantal te verplaatsen posities

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

moveleft

Beschrijving: Verplaatst de huidige positie naar links

Syntaxis: `void moveleft(unsigned char count)`

Parameters: **count:** Het aantal te verplaatsen posities

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

moveright

Beschrijving: Verplaatst de huidige positie naar rechts

Syntaxis: `void moveright(unsigned char count)`

Parameters: **count:** Het aantal te verplaatsen posities

Opmerkingen:
Er worden geen grenscontroles op de schermgrenzen uitgevoerd; het gedrag buiten het scherm is ongedefinieerd

wherex

Beschrijving: Retourneert de X-coördinaat van de huidige positie

Syntaxis: `unsigned char wherex(void)`

Resultaat: De X-coördinaat van de huidige positie

wherey

Beschrijving: Retourneert de Y-coördinaat van de huidige positie

Syntaxis: `unsigned char wherey(void)`

Resultaat: De Y-coördinaat van de huidige positie

pcputc

Beschrijving: Geeft één PETSCII-teken weer op het scherm op de huidige positie

Syntaxis: `void cputc(unsigned char c)`

Parameters: **c:** Het weer te geven PETSCII-teken

pcputsxy

Beschrijving: Geeft een PETSCII-string weer op de coördinaten X,Y

Syntaxis: `void pcputsxy (unsigned char x, unsigned char y, const unsigned char* s)`

Parameters: **x:** De X-coördinaat waar de string wordt weergegeven

y: De Y-coördinaat waar de string wordt weergegeven

s: De weer te geven PETSCII-string

Opmerkingen:

Er wordt geen pointercontrole uitgevoerd. Indien s null of ongeldig is, is het gedrag ongedefinieerd

cputcxy

Beschrijving: Geeft één PETSCII-teken weer op de coördinaten X,Y

Syntaxis: `void pcputcxy (unsigned char x, unsigned char y, unsigned char c)`

Parameters: **x:** De X-coördinaat waar het teken wordt weergegeven

y: De Y-coördinaat waar het teken wordt weergegeven

c: Het weer te geven PETSCII-teken

pcputs

Beschrijving: Geeft een PETSCII-string weer op de huidige positie

Syntaxis: `void pcputs(const unsigned char* s)`

Parameters: **s:** De weer te geven string

Opmerkingen: Er wordt geen pointercontrole uitgevoerd. Indien s null of ongeldig is, is het gedrag ongedefinieerd

cputc

Beschrijving: Geeft één schermcodeteken weer op het scherm op de huidige positie

Syntaxis: `void cputc(unsigned char c)`

Parameters: **c:** De schermcode van het weer te geven teken

cputnc

Beschrijving: Geeft N exemplaren van een teken weer op de huidige positie

Syntaxis: `void cputnc(unsigned char count, unsigned char c)`

Parameters: **c:** De schermcode van de weer te geven tekens

count: Het aantal weer te geven tekens

cputhex

Beschrijving: Geeft een hexadecimaal opgemaakt getal weer op de huidige positie

Syntaxis: `void cputhex(long n, unsigned char prec)`

Parameters: **n:** Het te schrijven getal

prec: De precisie van het hexadecimale getal, in cijfers. Voorloopnullen worden dienovereenkomstig weergegeven

Opmerkingen:

Het $-teken wordt automatisch aan het begin van de string toegevoegd

cputdec

Beschrijving: Geeft een decimaal getal weer op de huidige positie

Syntaxis: `void cputdec(long n, unsigned char padding, unsigned char leadingZ)`

Parameters: **n:** Het te schrijven getal

padding: De opvullingsruimte die vóór het getal moet worden toegevoegd

leadingZ: De voorloopnullen die moeten worden weergegeven

cputs

Beschrijving: Geeft schermcodes weer op de huidige positie

Syntaxis: `void cputs(const unsigned char* s)`

Parameters: **s:** Een array van schermcodes die moeten worden weergegeven

Opmerkingen:

Deze functie werkt uitsluitend met schermcodes. Voor het weergeven van gewone ASCII-/PETSCII-strings gebruikt u de macro "pcputs". Er wordt geen pointercontrole uitgevoerd. Indien s null of ongeldig is, is het gedrag ongedefinieerd.

cputsxy

Beschrijving: Geeft meerdere schermcodes weer op de X,Y-coördinaten

Syntaxis: `void cputsxy (unsigned char x, unsigned char y, const unsigned char* s)`

Parameters: **x:** De X-coördinaat waar de string wordt weergegeven

y: De Y-coördinaat waar de string wordt weergegeven

s: Een array van schermcodes die moeten worden weergegeven

Opmerkingen:

Deze functie werkt uitsluitend met schermcodes. Voor het weergeven van gewone ASCII-/PETSCII-strings gebruikt u de macro "pcputsxy". Er wordt geen pointercontrole uitgevoerd. Indien s null of ongeldig is, is het gedrag ongedefinieerd.

cputcxy

Beschrijving: Geeft één teken weer op de X,Y-coördinaten

Syntaxis:
```
void cputcxy (unsigned char x, unsigned char y, unsigned char c)
```

Parameters:

x: De X-coördinaat waar het teken wordt weergegeven

y: De Y-coördinaat waar het teken wordt weergegeven

c: De schermcode van het weer te geven teken

cputncxy

Beschrijving: Geeft N exemplaren van een teken weer op de X,Y-coördinaten

Syntaxis:
```
void cputncxy (unsigned char x, unsigned char y, unsigned char count, unsigned char c)
```

Parameters:

x: De X-coördinaat waar het teken wordt weergegeven

y: De Y-coördinaat waar het teken wordt weergegeven

count: Het aantal weer te geven tekens

c: De schermcode van de weer te geven tekens

cprintf

Beschrijving: Drukt opgemaakte uitvoer af.
Escape-strings kunnen worden gebruikt om attributen te wijzigen, de cursor te verplaatsen, enz., vergelijkbaar met PRINT in CBM BASIC.

Syntaxis:
```
unsigned char cprintf (const unsigned char* format, ...)
```

Parameters: **format:** De weer te geven string. De beschikbare escape-codes zijn:

Cursorpositionering

\t Ga naar de volgende tabpositie (veelvoud van 8)
\r Carriage Return
\n Nieuwe regel

{clr}	Scherm wissen	{home}	Cursor naar homepositie (linksbove
{d}	Cursor omlaag	{u}	Cursor omhoog
{r}	Cursor naar rechts	{l}	Cursor naar links

Attributen

{rvson}	Omgekeerd-attribuut AAN	{rvsoff}	Omgekeerd-attribu
{blon}	Knipperattribuut AAN	{bloff}	Knipperattribuut UI
{ulon}	Onderstrepingsattribuut AAN	{uloff}	Onderstrepingsattri

Kleuren (standaardpalet)

{blk}	{wht}	{red}	{cyan}
{pur}	{grn}	{blu}	{yel}
{ora}	{brn}	{pink}	{gray1}
{gray2}	{lblu}	{lgrn}	{gray3}

Opmerkingen:

Deze functie werkt uitsluitend met schermcodes! Gebruik de "pcprintf"-macro om gewone ASCII/PETSCII-tekenreeksen weer te geven. Momenteel wordt geen vervanging van argumenten uitgevoerd bij de variabele argumenten.

pcprintf

Beschrijving: Drukt opgemaakte PETSCII-tekenreeksuitvoer af.

Syntaxis: `see cprintf`

cgetc

Beschrijving: Wacht totdat een teken in de toetsenbordbuffer staat en retourneert dit

Syntaxis: `unsigned char cgetc (void);`

Resultaat: Het laatste teken in de toetsenbordbuffer

Opmerkingen:

Geretourneerde waarden zijn ASCII-tekencodes

kbhit

Beschrijving: Retourneert het teken in de toetsenbordbuffer

Syntaxis: `unsigned char kbhit (void);`

Resultaat: De tekencode in de toetsenbordbuffer, 0 indien de buffer leeg is.

Opmerkingen:
Geretourneerde waarden zijn ASCII-tekencodes

getkeymodstate

Beschrijving: Retourneert de toetsmodificatiestatus.

Syntaxis: `unsigned char getkeymodstate(void)`

Resultaat: Een byte met de toetsmodificatiestatusbits, waarbij de bits:

Bit	Betekenis	Constante
0	Rechts SHIFT-status	KEYMOD_RSHIFT
1	Links SHIFT-status	KEYMOD_LSHIFT
2	CTRL-status	KEYMOD_CTRL
3	MEGA-status	KEYMOD_MEGA
4	ALT-status	KEYMOD_ALT
5	NOSCRL-status	KEYMOD_NOSCRL
6	CAPSLOCK-status	KEYMOD_CAPSLOCK
7	Gereserveerd	-

flushkeybuf

Beschrijving: Leegt de toetsenbordbuffer

Syntaxis: `void flushkeybuf(void)`

cinput

Beschrijving: Ontvangt invoer van het toetsenbord en drukt binnenkomende tekens af op de huidige positie.

Syntaxis: `unsigned char cinput(char* buffer, unsigned char buflen, unsigned char flags)`

Parameters: **buffer:** Doeltekenbuffer die vooraf door de aanroeper is toegewezen

buflen: Lengte van de doelbuffer in tekens, inclusief het afsluitende nulteken

flags: Vlaggen voor invoer: (standaard worden alle afdrukbare tekens aanvaard) CINPUT_ACCEPT_NUMERIC
Aanvaardt numerieke tekens.

CINPUT_ACCEPT_LETTER
Aanvaardt letters.

`CINPUT_ACCEPT_SYM`
Aanvaardt symbolen.

`CINPUT_ACCEPT_ALL`
Aanvaardt alles. Is gelijk aan `CINPUT_ACCEPT_NUMERIC |CINPUT_ACCEPT_LETTER |CINPUT_ACCEPT_SYM`

`CINPUT_ACCEPT_ALPHA`
Aanvaardt alfanumerieke tekens. Is gelijk aan `CINPUT_ACCEPT_NUMERIC |CINPUT_ACCEPT_LETTER`

`CINPUT_NO_AUTOTRANSLATE`
Schakelt de functie uit waarmee cinput automatisch hoofdletters weergeeft wanneer de standaard kleinelettertekenset is geselecteerd en u letters intoetst zonder de SHIFT-toets; zonder deze functie geeft de computer grafische tekens weer in plaats van alfabetische tekens.

Resultaat: Aantal succesvol gelezen tekens in de buffer

VIC_BASE

VIC_BASE is een preprocessor-macro die het basisadres van de VIC-IV-chip opgeeft, namelijk $D000.

IS_H640 is een preprocessor-macro welke 0 retourneert wanneer de huidige VIC-III/IV-videomodus is ingesteld op 320 pixels breed (40-kolomsmodus) en een waarde ongelijk aan nul wanneer deze is ingesteld op 640 pixels breed (80-kolomsmodus).

HOOFDSTUK 22

BASIC-tokenisatoren

Er bestaan verschillende tokenisatoren voor C64-BASIC, zoals https://github.com/catseye/hatoucan, https://www.c64-wiki.com/wiki/C64list of het hulpprogramma **petcat** dat deel uitmaakt van VICE. Als u Ubuntu Linux gebruikt, kunt u **petcat** installeren met het volgende commando:

```
sudo apt-get install vice
```

We raden **petcat** aan, omdat het zowel C64 BASIC 2 als C65 BASIC 10 ondersteunt.

Sommige IDE's bieden BASIC 65-tokenisatorens, zoals:

Eleven

https://files.mega65.org?id=8b189d0b-ea1e-45a7-a4de-87bcb0b11696

C64 Studio

https://www.georg-rottensteiner.de/files/C64StudioRelease.zip

CBM prg Studio

https://www.ajordison.co.uk

DEEL VIII

BIJLAGEN

BIJLAGE A

Accessoires

BIJLAGE B

Naslag BASIC65-commando's

- Commando's, functies en operatoren
- Alfabetisch overzicht BASIC-commando's

COMMANDO'S, FUNCTIES EN OPERATOREN

In deze bijlage worden de commando's, functies en andere aanroepbare elementen van BASIC 65 beschreven. BASIC 65 is een uitgebreide versie van BASIC 10. Een aantal van deze elementen kan één of meer argumenten aannemen: dit zijn gegevens die u als onderdeel van het commando of de functieaanroep opgeeft om te beschrijven wat u wilt bereiken. Sommige vereisen bovendien het gebruik van speciale trefwoorden.

Hieronder volgt een voorbeeld van hoe commando's, functies en operatoren (welke alle ook wel trefwoorden worden genoemd) in dit bijlage worden beschreven.

KEY getal**,** tekenreeks

Hier is **KEY** een *trefwoord*. Trefwoorden zijn speciale woorden die BASIC begrijpt. In deze handleiding worden trefwoorden altijd in **VETTE HOOFDLETTERS** geschreven, zodat u ze eenvoudig kunt herkennen.

De "getal" en "tekenreeks" (in niet-vette tekst) zijn voorbeelden van *argumenten*. Deze vervangt u door waarden of algebraïsche expressies (*expressies*) die de gegevens weergeven die het gedrag van het commando bepalen.

Leestekens en andere letters in vette tekst geven andere tekens aan die precies zo moeten worden getypt als ze verschijnen. In dit voorbeeld moet er een komma staan tussen het argument voor het getal en het argument voor de tekenreeks.

Hier volgt een voorbeeld van het gebruik van het **KEY**-commando op basis van dit patroon:

```
KEY 6, "LIST" + CHR$(13)
```

Wanneer u vierkante haakjes om argumenten ziet staan, geeft dit aan dat de argumenten optioneel zijn. U moet de vierkante haakjes niet typen. Bekijk de volgende beschrijving van het **CIRCLE**-commando, dat optionele argumenten aanvaardt:

CIRCLE xc**,** yc**,** straal [**,** vlaggen **,** begin**,** einde]

De volgende voorbeelden van het **CIRCLE**-commando zijn beide geldig. Ze hebben verschillend gedrag op basis van hun verschillende argumenten.

```
CIRCLE 100, 150, 30

CIRCLE 100, 150, 30, 0, 45, 135
```

Deze rangschikking van trefwoorden, symbolen en argumenten wordt *syntaxis* genoemd. Indien u iets weglaat of iets op de verkeerde plek zet, zal de computer het commando niet begrijpen en een *syntaxisfout* melden.

U hoeft zich geen zorgen te maken wanneer u een fout van de MEGA65 krijgt. Op die manier geeft de MEGA65 u te kennen dat er iets niet in orde is, zodat u het probleem eenvoudiger kunt vinden en verhelpen. Indien u bijvoorbeeld de komma in het **KEY**-commando weglaat of vervangt door een punt, zal de MEGA65 reageren met een Syntaxisfout, zoals hieronder te zien is:

```
READY.
KEY 8"VIS"

?SYNTAX ERROR
READY.
KEY 8."VIS"

?SYNTAX ERROR
READY.
```

Expressies kunnen een numerieke waarde zijn zoals `23.7`, een tekenreekswaarde zoals `"HALLO"`, of een complexere berekening die waarden, functies en operatoren combineert om een getal of tekenreeks weer te geven: `"LIST"+CHR$(13)`

Het is belangrijk om het juiste type expressie te gebruiken bij het schrijven van programma's. Indien u per ongeluk het verkeerde type gebruikt, zal de MEGA65 een Type Mismatch-fout weergeven om aan te geven dat het type expressie dat u heeft opgegeven niet overeenkomt met wat werd verwacht. Het volgende commando geeft bijvoorbeeld een Type Mismatch-fout, omdat `"POTATO"` een tekenreeksexpressie is en er een numerieke expressie wordt verwacht:

```
KEY "AARDAPPEL", "SOEP"
```

Commando's zijn instructies die u direct vanaf het `READY.`-prompt kunt gebruiken, of vanuit een programma. Bijvoorbeeld:

```
READY.
PRINT "HALLO DAAR"
HALLO DAAR

READY.
10 PRINT "HALLO DAAR"
RUN
HALLO DAAR
```

U kunt een reeks instructies op één regel plaatsen door ze te scheiden met dubbele punten. Bijvoorbeeld:

```
PRINT "HALLO" : PRINT "HOE GAAT HET?" : PRINT "LEKKER WEERTJE?"
HALLO
HOE GAAT HET?
LEKKER WEERTJE?
```

Commando's in directe modus

Sommige commando's werken alleen in **directe modus** (soms "onmiddellijke modus" genoemd). Dit betekent dat het commando geen onderdeel van een BASIC-programma kan zijn, maar wel direct op het scherm kan worden ingevoerd. Het **RENUMBER**-commando werkt bijvoorbeeld alleen in directe modus, omdat het tot doel heeft de regels van een BASIC-programma te hernummeren.

In de twee **PRINT**-voorbeelden hierboven werd de eerste in directe modus ingevoerd, terwijl de tweede deel uitmaakte van een programma. Het **PRINT**-commando werkt zowel in directe modus als in een programma.

Syntaxisbeschrijvingen van commando's

De volgende tabel beschrijft de overige symbolen die voorkomen in de syntaxisbeschrijvingen van commando's.

Symbool	Betekenis
[ ]	Optioneel
...	De syntaxis tussen vierkante haakjes kan nul of meer keer worden herhaald
< | >	Kies een van beiden
[|]	Neem optioneel één van de keuzen op
{ , }	Ten minste één van de argumenten is verplicht. De komma's links van het laatste opgenomen argument zijn verplicht. Optionele komma's achteraan moeten worden weggelaten. Zie **CURSOR** voor een voorbeeld.
[{ , }]	Vergelijkbaar met { , } maar alle argumenten kunnen worden weggelaten

Lettertypen

Voorbeelden van tekst die op het scherm verschijnt, hetzij door u getypt hetzij door de MEGA65 afgedrukt, worden weergegeven in het schermlettertype: `"LIST"+CHR$(13)`

BASIC 65-constanten

Waarden die direct in een expressie of programma worden getypt, worden *constanten* genoemd. Deze waarden zijn "constant" omdat ze niet veranderen op basis van andere aspecten van de programmastatus.

Hierna volgen de typen constanten die in een BASIC 65-expressie kunnen voorkomen.

Type	Voorbeeld	Voorbeeld
Decimaal geheel getal	32000	-55
Decimaal vastpuntgetal	3.14	-7654.321
Decimaal drijvend-kommagetal	1.5E03	7.7E-02
Hexadecimaal	$D020	$FF
Binair	%11010010	%101
Tekenreeks	"X"	"TEXT"

BASIC 65-variabelen

Een programma bewerkt gegevens door waarden in het geheugen van de computer op te slaan, opgeslagen waarden aan te roepen en deze bij te werken op basis van logica. In BASIC heten geheugenelementen die waarden opslaan *variabelen*. Elke variabele heeft een naam; er bestaan aparte reeksen van variabelenamen voor elk type waarde.

De variabele AA kan bijvoorbeeld een numerieke waarde opslaan. De variabele AA$ kan een tekenreekswaarde opslaan. Commodore BASIC beschouwt deze als twee aparte variabelen, hoewel beide namen beginnen met AA.

Een manier om een waarde aan een variabele toe te wijzen, is met de toewijzingsoperator (=). Voorbeeld:

```
AA = 1.95

AA$ = "HALLO, "
```

Variabelenamen moeten beginnen met een letter en mogen alleen letters en getallen bevatten. De naam mag elke lengte hebben, doch Commodore BASIC herkent alleen de eerste twee letters. SPEED en SP worden als dezelfde variabele beschouwd.

Variabelenamen mogen geen van de BASIC-trefwoorden bevatten. Het gebruik van lange namen is daardoor moeilijk: het is gemakkelijk om per ongeluk een trefwoord te gebruiken. ENFORCEMENT is bijvoorbeeld geen geldige variabelenaam, omdat **FOR** een trefwoord is. Het is gebruikelijk om korte variabelenamen te kiezen om dit soort problemen te voorkomen.

Een variabele kan worden gebruikt in een expressie samen met constanten, variabelen, functies en operatoren. Op het moment van uitvoering wordt de variabele vervangen door de waarde die deze dan bevat.

```
10 INPUT "HOE HEET JE"; NA$
20 MSG$ = "HALLO, " + NA$ + "!"
30 PRINT MSG$
```

In tegenstelling tot sommige programmeertalen hoeven BASIC-variabelen niet te worden gedeclareerd voor gebruik. Een variabele heeft een standaardwaarde van nul voor numerieke variabelen, of de lege tekenreeks ("") voor tekenreeksvariabelen.

Een variabele die een enkele waarde opslaat, wordt ook wel een *scalaire variabele* genoemd. De scalaire variabelentypen en hun waardebereiken zijn als volgt.

Type	Naamsymbool	Bereik	Voorbeeld
Byte	&	0 .. 255	BY& = 23
Integer	%	-32768 .. 32767	I% = 5
Real	geen	-1E37 .. 1E37	XY = 1/3
String	$	lengte = 0 .. 255	AB$ = "TEXT"

Een variabele waarvan de naam bestaat uit één letter gevolgd door het typesymbool (of geen symbool voor reële getallen) is een *snelle variabele*. BASIC 65 slaat de variabele zodanig op dat de waarde sneller toegankelijk is dan bij variabelen met langere namen. Verder gedraagt deze zich zoals elke andere variabele. Dit geldt ook voor functies die met **DEF FN** worden gedefinieerd.

BASIC 65-arrays

Naast scalaire variabelen kent Commodore BASIC ook een type variabele dat meerdere waarden kan opslaan: een *array*.

Het volgende voorbeeld slaat drie tekenreekswaarden op in een array en gebruikt een **FOR**-lus om voor elk element een **PRINT**-melding weer te geven:

```
10 DIM NA$(3)
20 NA$(0) = "DEFT"
30 NA$(1) = "GARDNERS"
40 NA$(2) = "LYDON"
50 FOR I = 0 TO 2
60 PRINT "HALLO, "; NA$(I); "!"
70 NEXT I
```

Elke waarde in een array wordt aangeroepen met de naam van de arrayvariabele en een geheel getal als index. Verwijzingen naar array-elementen gebruiken de indexwaarde, zoals in AA(7) dat verwijst naar het element van de array AA() met index 7. De index begint bij nul: het eerste element heeft index 0. De index kan een numerieke expressie zijn, wat een krachtige manier is om met meerdere gegevenselementen te werken.

Alle waarden in een array moeten van hetzelfde type zijn. Het type wordt aangeduid in de naam van de variabele, net als bij scalaire variabelen. AA() is een array van reële getallen, AA$() is een array van tekenreeksen.

Arrayvariabelenamen worden als apart beschouwd van scalaire variabelenamen. De scalaire variabele AA heeft geen relatie met de arrayvariabele AA().

BASIC moet de maximale grootte van de array kennen voordat deze voor het eerst wordt gebruikt, zodat het geheugen voor de volledige array kan worden toegewezen. Een programma kan de grootte van een array declareren met het **DIM**-trefwoord, waarmee de "dimensies" van de array worden opgegeven. Indien een arrayvariabele wordt gebruikt zonder expliciete declaratie, wijst BASIC een eendimensionale array met 10 elementen toe, en kan de array niet opnieuw gedimensioneerd worden (tenzij u alle variabelen met **CLR** wist).

Een array kan meerdere dimensies hebben, elk met een eigen index gescheiden door een komma. De array moet worden gedeclareerd met de maximale waarde voor elke dimensie. Houd er rekening mee dat BASIC 65 geheugen toewijst voor de gehele array, zodat grote arrays beperkt kunnen worden door het beschikbare geheugen.

```
DIM BO$(3, 3)
BO$(1, 1) = "X"
BO$(0, 0) = "O"
BO$(0, 2) = "X"
BO$(1, 0) = "O"
```

Schermtekst- en kleurenarrays

Een BASIC 65-programma kan tekst op het scherm weergeven op verschillende manieren. Het **PRINT**-commando toont een tekenreeks op de huidige cursorpositie, wat vooral handig is voor terminal-achtige uitvoer. Het **CURSOR**-commando verplaatst de cursor naar een opgegeven positie. Een programma kan deze commando's samen gebruiken om afbeeldingen of gebruikersinterfaces te tekenen.

Een programma kan individuele tekens op het scherm benaderen met behulp van de speciale ingebouwde arrays **T@&()** en **C@&()**. Deze arrays zijn tweedimensionaal met indexen die overeenkomen met de kolom en rij van elk teken op het scherm, beginnend bij (0,0) in de linkerbovenhoek.

T@&(kolom**,** rij**)** is de schermcode van het teken. Schermcodes verschillen van PETSCII-codes. Zie bijlage C op bladzijde C-3 voor een lijst met schermcodes.

C@&(kolom**,** rij**)** is de kleurcode van het teken. Dit is een itemnummer van het systeempalet. Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard systeempalet. De hogere bits stellen ook tekstattributen in, zoals knipperen.

Net als bij gewone arrays kunnen de waarden van scherm- en kleurarrays worden toegewezen of worden gebruikt in expressies om hun huidige waarden op te halen.

```
10 FOR X = 10 TO 30
20 T@&(X, 2) = 1
30 C@&(X, 2) = INT(RND(1) * 16)
40 NEXT X
50 PRINT "KLEUR OP POSITIE 15: "; C@&(15, 2)
```

De afmetingen van deze arrays hangen af van de huidige tekstschermmodus. In 80 × 25 tekstmodus varieert de kolom van 0 – 79 en de rij van 0 – 24. De MEGA65 ondersteunt ook 80 × 50 en 40 × 25 tekstmodi.

BASIC 65-operatoren

Een *operator* is een symbool of trefwoord dat een functie uitvoert in een expressie. De operator werkt op een of twee deelexpressies, die *operanden* worden genoemd. De operator en zijn operanden *evalueren tot* het resultaat van de bewerking.

De * (asterisk) operator voert bijvoorbeeld een vermenigvuldiging uit van twee numerieke operanden. De operator en zijn operanden evalueren tot het resultaat van de vermenigvuldiging.

```
A = 6
PRINT A * 7
```

De +-operator (plus) heeft een verschillende betekenis afhankelijk van het type van de operanden. Bij getaloperanden voegt deze de waarden samen. Bij tekenreeksoperanden voegt deze de twee tekenreeksen samen.

```
A = 64
PRINT A + 1

A$ = "MEGA"
PRINT A$ + "65"
```

De --operator (min) aanvaardt één of twee operanden. Bij één getaloperand keert deze de waarde om. Bij twee getaloperanden trekt deze de tweede waarde af van de eerste.

```
A = 64
PRINT -A

PRINT A - 16
```

Het gelijkheidssymbool (=) wordt gebruikt als zowel een toewijzingsopdracht als een relationele operator. Als toewijzing is het =-symbool een opdracht die de waarde van een variabele bijwerkt. De linkerzijde moet een variabele of array-elementreferentie zijn en het type moet overeenkomen met het type van de expressie aan de rechterzijde. De toewijzing is geen operator: deze maakt geen deel uit van een expressie.

```
AA = 7
NA$ = "DEFT"
```

Als relationele operator gedraagt het =-symbool zich als een expressie. Het evalueert de expressie aan weerszijden van de operator en test vervolgens of de waarden gelijk zijn. Als ze gelijk zijn, evalueert de gelijkheidsoperator tot -1, de weergave van BASIC van "true". Als ze niet gelijk zijn, evalueert de operator tot 0, oftewel "false". De gelijkheidsexpressie kan worden gebruikt met een **IF**-opdracht om de programmastroom te sturen, of als onderdeel van een numerieke expressie. Beide expressies moeten van hetzelfde type zijn.

```
100 IF X = 99 THEN 130
110 X = X + 1
120 GOTO 100
130 PRINT "GEREED."
```

BASIC 65 onderscheidt toewijzing en gelijkheid op basis van de context. Bekijk de volgende twee regels:

```
A = B = 10
```

BASIC 65 verwacht een opdracht en herkent een variabelenaam gevolgd door het symbool =. Het concludeert dat dit een opdracht is die een waarde toekent aan de numerieke variabele A. Vervolgens verwacht het een numerieke expressie rechts van het symbool =.

De operatoren **NOT**, **AND**, **OR** en **XOR** kunnen zowel als logische operatoren als bitgewijze operatoren worden gebruikt. Bij gebruik als logische operatoren werken deze op de booleaanse waarden van hun operanden. Bij gebruik als bitgewijze operatoren werken deze op de afzonderlijke bits van hun operanden.

```
IF X = 99 OR Y < 5 THEN 130

IF Y > 10 AND Y < 20 THEN 150
```

Een booleaanse operator aanvaardt twee getaloperanden en voert een berekening uit op de bits van de operanden. Het resultaat is een getalwaarde.

```
A = 17
PRINT A AND 20
```

In tegenstelling tot andere gevallen waarin operatoren zich verschillend gedragen afhankelijk van hoe ze worden gebruikt, hoeft BASIC 65 niet te bepalen of deze operatoren zich gedragen als logische operatoren of booleaanse operatoren. Omdat "true" en "false" worden weergegeven met zorgvuldig gekozen getallen, gedragen de logische operatoren zich hetzelfde of hun operanden nu voorwaardelijke expressies of getallen zijn. Een "true"-voorwaardelijke expressie is het getal -1, dat intern een binair getal is met alle bits gezet. De logische expressie "true and false" komt overeen

met de binaire booleaanse expressie %....0000 & %....1111. In dit geval evalueert de **AND**-operator tot 0, wat "false" is.

Voorwaardelijke expressies die getallen opleveren kunnen bij sommige slimme programmeertechnieken worden gebruikt. Bijvoorbeeld:

```
A = A - (B > 7)
```

Deze opdracht verhoogt de waarde in A met 1 als de waarde in B groter is dan 7. Anders laat hij deze ongewijzigd. Als de sub-expressie B > 7 waar is, evalueert deze tot -1. A - (-1) is equivalent aan A + 1. Als de sub-expressie onwaar is, evalueert deze tot 0, en A - 0 is equivalent aan A.

Wanneer meerdere operatoren in een enkele expressie worden gebruikt, wordt de volgorde waarin ze worden geëvalueerd gespecificeerd door *precedence*. In de opdracht A * A - B * B worden bijvoorbeeld eerst beide vermenigvuldigingen uitgevoerd en dan de aftrekking. Net als in de algebra kunt u haakjes gebruiken om de volgorde van uitvoering te wijzigen. In de expressie A * (A - B) * B wordt de aftrekking eerst uitgevoerd.

De volledige reeks operatoren en hun volgorde van voorrang worden samengevat in de hiernavolgende secties.

Toegewezen instructie

Symbool	Beschrijving	Voorbeelden
=	Toewijzing	A = 42, A$ = "HALLO", A = B < 42

Unaire wiskundige operatoren

Naam	Symbool	Beschrijving	Voorbeeld
Plus	+	Positief teken	A = +42
Min	-	Negatief teken	B = -42

Binaire wiskundige operatoren

Naam	Symbool	Beschrijving	Voorbeeld
Plus	+	Optellen	A = B + 42
Min	-	Aftrekken	B = A - 42
Ster	*	Vermenigvuldigen	C = A * B
Schuine streep	/	Delen	D = B / 13
Omhoog-pijl	↑	Machtsverheffen	E = 2 ↑ 10
Links verschuiven	<<	Links verschuiven	A = B << 2
Rechts verschuiven	>>	Rechts verschuiven	A = B >> 1

OPMERKING: Het teken ↑ dat voor de verheffing tot een macht wordt gebruikt, wordt ingetoetst met [↑], naast [RESTORE].

Relationele operatoren

Symbool	Beschrijving	Voorbeeld
)	Groter dan	A) 42
)=	Groter dan of gelijk aan	B)= 42
(	Kleiner dan	A (42
(=	Kleiner dan of gelijk aan	B (= 42
=	Gelijk	A = 42
()	Niet gelijk	B () 42

Logische operatoren

Trefwoord	Beschrijving	Voorbeeld
AND	En	A) 42 AND A (84
OR	Of	A) 42 OR A = 0
XOR	Exclusief-of	A) 42 XOR B) 42
NOT	Ontkenning	C = NOT A) B

Booleaanse operatoren

Trefwoord	Beschrijving	Voorbeeld
AND	En	A = B AND $FF
OR	Of	A = B OR $80
XOR	Exclusief-of	A = B XOR 1
NOT	Ontkenning	A = NOT 22

Tekenreekshandeling

Naam	Symbool	Beschrijving	Operandtype	Voorbeeld
Plus	+	Voegt tekenreeksen samen	Tekenreeks	A$ = B$ + ".PRG"

Operatorrangorde

Rangorde	Operatoren
Hoog	↑
	+ - (unair wiskundig)
	* /
	+ - (binair wiskundig)
	(( )) (rekenkundig verschuiven)
	((=))= = ()
	NOT
	AND
Laag	OR XOR

Trefwoorden en tokens deel 1

*	AC	COLLISION	FE17	EXP	BD
+	AA	COLOR	E7	FAST	FE25
-	AB	CONCAT	FE13	FGOSUB	FE48
/	AD	CONT	9A	FGOTO	FE47
<	B3	COPY	F4	FILTER	FE03
<<	FE52	COS	BE	FIND	FE2B
=	B2	CURSOR	FE41	FN	A5
>	B1	CUT	E4	FONT	FE46
>>	FE53	DATA	83	FOR	81
ABS	B6	DCLEAR	FE15	FOREGROUND	FE39
AND	AF	DCLOSE	FE0F	FORMAT	FE37
APPEND	FE0E	DEC	D1	FRE	B8
ASC	C6	DECBIN	CE11	FREAD#	FE1C
ATN	C1	DEF	96	FREEZER	FE4A
AUTO	DC	DELETE	F7	FWRITE#	FE1E
BACKGROUND	FE3B	DIM	86	GCOPY	FE32
BACKUP	F6	DIR	EE	GET	A1
BANK	FE02	DISK	FE40	GO	CB
BEGIN	FE18	DLOAD	F0	GOSUB	8D
BEND	FE19	DMA	FE1F	GOTO	89
BLOAD	FE11	DMODE	FE35	GRAPHIC	DE
BOOT	FE1B	DO	EB	HEADER	F1
BORDER	FE3C	DOPEN	FE0D	HELP	EA
BOX	E1	DOT	FE4C	HEX$	D2
BSAVE	FE10	DPAT	FE36	HIGHLIGHT	FE3D
BUMP	CE03	DSAVE	EF	IF	8B
BVERIFY	FE28	DVERIFY	FE14	IMPORT	DD
CATALOG	FE0C	ECTORY	FE29	INFO	FE4D
CHANGE	FE2C	EDIT	FE45	INPUT	85
CHAR	E0	EDMA	FE21	INPUT#	84
CHDIR	FE4B	ELLIPSE	FE30	INSTR	D4
CHR$	C7	ELSE	D5	INT	B5
CIRCLE	E2	END	80	JOY	CF
CLOSE	A0	ENVELOPE	FE0A	KEY	F9
CLR	9C	ERASE	FE2A	LEFT$	C8
CMD	9D	ERR$	D3	LEN	C3
COLLECT	F3	EXIT	ED	LET	88

Trefwoorden en tokens deel 2

LINE	E5	PRINT	99	SPEED	FE26
LIST	9B	PRINT#	98	SPRCOLOR	FE08
LOAD	93	RCOLOR	CD	SPRITE	FE07
LOADIFF	FE43	RCURSOR	FE42	SPRSAV	FE16
LOCK	FE50	READ	87	SQR	BA
LOG	BC	RECORD	FE12	STEP	A9
LOG10	CE08	REM	8F	STOP	90
LOOP	EC	RENAME	F5	STRBIN$	CE12
LPEN	CE04	RENUMBER	F8	STR$	C4
MEM	FE23	RESTORE	8C	SYS	9E
MERGE	E6	RESUME	D6	TAB(	A3
MID$	CA	RETURN	8E	TAN	C0
MKDIR	FE51	RGRAPHIC	CC	TEMPO	FE05
MOD	CE0B	RIGHT$	C9	THEN	A7
MONITOR	FA	RMOUSE	FE3F	TO	A4
MOUNT	FE49	RND	BB	TRAP	D7
MOUSE	FE3E	RPALETTE	CE0D	TROFF	D9
MOVSPR	FE06	RPEN	D0	TRON	D8
NEW	A2	RPLAY	CE0F	TYPE	FE27
NEXT	82	RREG	FE09	UNLOCK	FE4F
NOT	A8	RSPCOLOR	CE07	UNTIL	FC
OFF	FE24	RSPEED	CE0E	USING	FB
ON	91	RSPPOS	CE05	USR	B7
OPEN	9F	RSPRITE	CE06	VAL	C5
OR	B0	RUN	8A	VERIFY	95
PAINT	DF	RWINDOW	CE09	VIEWPORT	FE31
PALETTE	FE34	SAVE	94	VOL	DB
PASTE	E3	SAVEIFF	FE44	VSYNC	FE54
PEEK	C2	SCNCLR	E8	WAIT	92
PEN	FE33	SCRATCH	F2	WHILE	FD
PIXEL	CE0C	SCREEN	FE2E	WINDOW	FE1A
PLAY	FE04	SET	FE2D	WPEEK	CE10
POINTER	CE0A	SGN	B4	WPOKE	FE1D
POKE	97	SIN	BF	XOR	E9
POLYGON	FE2F	SLEEP	FE0B	^	AE
POS	B9	SOUND	DA		
POT	CE02	SPC(	A6		

Tokens en trefwoorden deel 1

80 END	A5 FN	CA MID$
81 FOR	A6 SPC(	CB GO
82 NEXT	A7 THEN	CC RGRAPHIC
83 DATA	A8 NOT	CD RCOLOR
84 INPUT#	A9 STEP	CE02 POT
85 INPUT	AA +	CE03 BUMP
86 DIM	AB -	CE04 LPEN
87 READ	AC *	CE05 RSPPOS
88 LET	AD /	CE06 RSPRITE
89 GOTO	AE ^	CE07 RSPCOLOR
8A RUN	AF AND	CE08 LOG10
8B IF	B0 OR	CE09 RWINDOW
8C RESTORE	B1 >	CE0A POINTER
8D GOSUB	B2 =	CE0B MOD
8E RETURN	B3 <	CE0C PIXEL
8F REM	B4 SGN	CE0D RPALETTE
90 STOP	B5 INT	CE0E RSPEED
91 ON	B6 ABS	CE0F RPLAY
92 WAIT	B7 USR	CE10 WPEEK
93 LOAD	B8 FRE	CE11 DECBIN
94 SAVE	B9 POS	CE12 STRBIN$
95 VERIFY	BA SQR	CF JOY
96 DEF	BB RND	D0 RPEN
97 POKE	BC LOG	D1 DEC
98 PRINT#	BD EXP	D2 HEX$
99 PRINT	BE COS	D3 ERR$
9A CONT	BF SIN	D4 INSTR
9B LIST	C0 TAN	D5 ELSE
9C CLR	C1 ATN	D6 RESUME
9D CMD	C2 PEEK	D7 TRAP
9E SYS	C3 LEN	D8 TRON
9F OPEN	C4 STR$	D9 TROFF
A0 CLOSE	C5 VAL	DA SOUND
A1 GET	C6 ASC	DB VOL
A2 NEW	C7 CHR$	DC AUTO
A3 TAB(	C8 LEFT$	DD IMPORT
A4 TO	C9 RIGHT$	DE GRAPHIC

Tokens en trefwoorden deel 2

DF PAINT	FE08 SPRCOLOR	FE2F POLYGON
E0 CHAR	FE09 RREG	FE30 ELLIPSE
E1 BOX	FE0A ENVELOPE	FE31 VIEWPORT
E2 CIRCLE	FE0B SLEEP	FE32 GCOPY
E3 PASTE	FE0C CATALOG	FE33 PEN
E4 CUT	FE0D DOPEN	FE34 PALETTE
E5 LINE	FE0E APPEND	FE35 DMODE
E6 MERGE	FE0F DCLOSE	FE36 DPAT
E7 COLOR	FE10 BSAVE	FE37 FORMAT
E8 SCNCLR	FE11 BLOAD	FE39 FOREGROUND
E9 XOR	FE12 RECORD	FE3B BACKGROUND
EA HELP	FE13 CONCAT	FE3C BORDER
EB DO	FE14 DVERIFY	FE3D HIGHLIGHT
EC LOOP	FE15 DCLEAR	FE3E MOUSE
ED EXIT	FE16 SPRSAV	FE3F RMOUSE
EE DIR	FE17 COLLISION	FE40 DISK
EF DSAVE	FE18 BEGIN	FE41 CURSOR
F0 DLOAD	FE19 BEND	FE42 RCURSOR
F1 HEADER	FE1A WINDOW	FE43 LOADIFF
F2 SCRATCH	FE1B BOOT	FE44 SAVEIFF
F3 COLLECT	FE1C FREAD#	FE45 EDIT
F4 COPY	FE1D WPOKE	FE46 FONT
F5 RENAME	FE1E FWRITE#	FE47 FGOTO
F6 BACKUP	FE1F DMA	FE48 FGOSUB
F7 DELETE	FE21 EDMA	FE49 MOUNT
F8 RENUMBER	FE23 MEM	FE4A FREEZER
F9 KEY	FE24 OFF	FE4B CHDIR
FA MONITOR	FE25 FAST	FE4C DOT
FB USING	FE26 SPEED	FE4D INFO
FC UNTIL	FE27 TYPE	FE4F UNLOCK
FD WHILE	FE28 BVERIFY	FE50 LOCK
FE02 BANK	FE29 ECTORY	FE51 MKDIR
FE03 FILTER	FE2A ERASE	FE52 <<
FE04 PLAY	FE2B FIND	FE53 >>
FE05 TEMPO	FE2C CHANGE	FE54 VSYNC
FE06 MOVSPR	FE2D SET	
FE07 SPRITE	FE2E SCREEN	

ALFABETISCH OVERZICHT BASIC-COMMANDO'S

ABS

Token: $B6

Formaat: **ABS**(x)

Retourneert:

De absolute waarde van het numerieke argument **x**.

x numeriek geheel- of reëel getal als expressie-argument.

Opmerkingen:

Het resultaat is van het type reëel.

Voorbeelden:

Gebruik van **ABS**

```
PRINT ABS(-123)
 123

PRINT ABS(4.5)
 4.5

PRINT ABS(-4.5)
 4.5
```

AND

Token: $AF

Formaat: operand **AND** operand

Gebruik: Voert een bitgewijze logische AND-bewerking uit op twee 16-bit-waarden.

Gehele operanden worden ongewijzigd gebruikt. Reële operanden worden omgezet naar een geheel getal van 16 bit in tweecomplementformaat (waarbij de precisie verloren gaat). Logische operanden worden omgezet naar geheel getal van 16 bit met $FFFF (-1) voor TRUE en $0000 (0) voor FALSE.

Expressie	Resultaat
0 AND 0	0
0 AND 1	0
1 AND 0	0
1 AND 1	1

Opmerkingen:

Het resultaat is van het type geheel getal. Indien het resultaat in een logische context wordt gebruikt, wordt de waarde 0 als ONWAAR beschouwd en alle andere niet-nul-waarden als WAAR.

Voorbeelden:

Gebruik van **AND**

```
PRINT 1 AND 3
 1

PRINT 128 AND 64
 0
```

Gebruik van **AND** in een logische context (**IF**)

```
IF (C >= 0 AND C < 256) THEN PRINT "BYTE VALUE"
```

APPEND

Token: $FE $0E

Formaat: **APPEND#** kanaal**,** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Opent een bestaand bestand van het type **SEQ** of **USR** voor schrijven en plaatst de schrijfcursor aan het einde van het bestand.

kanaalnummer waarbij:

- 1 <= kanaal <= 127: regelafsluiting is CR.
- 128 <= kanaal <= 255: regelafsluiting is CR LF.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

APPEND# werkt op dezelfde manier als **DOPEN#... ,W**, met dien verstande dat het bestand al moet bestaan.

De inhoud van het bestand behouden en alle afgedrukte tekst wordt aan het einde toegevoegd.

Het proberen om **APPEND#** op een niet-bestaand bestand uit te voeren, geeft een DOS-foutmelding.

Voorbeelden:

Gebruik van **APPEND#** (bestaande bestanden openen)

```
APPEND#5, "DATA", U9

APPEND#130, (DD$), U(UN%)

APPEND#3, "USER FILE,U"

APPEND#2, "DATA BASE"
```

ASC

Token: $C6

Formaat: **ASC**(tekenreeks)

Retourneert:

De PETSCII-code van het eerste teken van de tekenreeks als argument, als getal.

Opmerkingen:

ASC retourneert nul voor lege tekenreeksen. Dit verschilt van BASIC 2, dat een Illegal Quantity-fout gaf voor lege tekenreeksen.

De inverse functie van **ASC** is **CHR$**. Zie de functie **CHR$** op bladzijde B-48 voor meer informatie.

De naam was blijkbaar gekozen als een geheugensteuntje voor “ASCII”, maar de geretourneerde waarde is een PETSCII-code.

Voorbeelden:

Gebruik van **ASC**

```
PRINT ASC("MEGA")
 77

PRINT ASC("")
 0
```

ATN

Token: $C1

Formaat: **ATN**(numerieke expressie)

Retourneert:

De arctangens van het argument.

Het resultaat ligt in het bereik ($-\pi/2$ tot $\pi/2$).

Opmerkingen:

Een vermenigvuldiging van het resultaat met $180/\pi$ zet de waarde om naar de eenheid "graden". **ATN** is de inverse functie van **TAN**.

Voorbeelden:

Gebruik van **ATN**

```
PRINT ATN(0.5)
 0.46364761

PRINT ATN(0.5) * 180 / π
 26.565051
```

AUTO

Token: $DC

Formaat: **AUTO** [stap]

Gebruik: Schakelt automatische regelnummering in of uit tijdens het invoeren van een BASIC-programma. Nadat een nieuwe programmaregel aan de BASIC-editor is aangeboden met RETURN, genereert de functie **AUTO** een nieuw BASIC-regelnummer voor de invoer van de volgende regel. Het nieuwe nummer wordt berekend door **stap** op te tellen bij het huidige regelnummer.

stap regelnummerstijging.

Het typen van **AUTO** zonder argument schakelt de functie uit.

Voorbeelden:

Gebruik van **AUTO**

```
AUTO 10 : REM GEBRUIK AUTO MET STAPPEN VAN 10

AUTO    : REM ZET AUTO AF
```

BACKGROUND

Token: $FE $3B

Formaat: **BACKGROUND** kleur

Gebruik: Stelt de achtergrondkleur van het scherm in.

kleur paletindex, in het bereik 0 - 255

Alle kleuren binnen dit bereik zijn aanpasbaar via het commando **PALETTE**.

Bij opstarten heeft de MEGA65 slechts de eerste 32 kleuren geconfigureerd. Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard systeempalet.

Voorbeeld: Gebruik van **BACKGROUND**

```
BACKGROUND 3 : REM KIES CYAAN ALS ACHTERGRONDKLEUR
```

BACKUP

Token: $F6

Formaat: **BACKUP U** bron **TO U** doel
BACKUP D bron **TO D** doel [**,U** eenheid]

Gebruik: Kopieert een schijf naar een andere.

De eerste vorm van **BACKUP**, waarbij eenheden voor bron en doel worden opgegeven, kan alleen worden gebruikt voor de stations die op de interne FDC (Floppy Disk Controller) zijn aangesloten. Eenheden 8 en 9 zijn gereserveerd voor deze controller. Dit kunnen de interne diskette (eenheid 8) en een andere diskette (eenheid 9) zijn die op dezelfde lintkabel is aangesloten, of gemonteerde D81-schijfimages. **BACKUP** kan worden gebruikt om van diskette naar diskette, diskette naar image, image naar diskette en image naar image te kopiëren, afhankelijk van de image-mounts en de aanwezigheid van een tweede fysieke diskette.

De tweede vorm van **BACKUP**, waarbij stations voor bron en doel worden opgegeven, is bedoeld voor dual-drive-eenheden die op de IEC-bus zijn aangesloten. Bijvoorbeeld: CBM 4040, 8050, 8250 via een IEEE-488-naar-IEC-adapter. In dit geval wordt de back-up intern door het schijfstation uitgevoerd.

bron station of schijf# van bronstation.

doel station of schijf# van doelschijf.

Opmerkingen:

De doelschijf wordt geformatteerd en er wordt een identieke kopie van de bronschijf geschreven.

BACKUP kan niet worden gebruikt om van interne apparaten naar IEC-apparaten te kopiëren of omgekeerd.

Voorbeelden:

Gebruik van **BACKUP**

```
BACKUP U8 TO U9      : REM BACKUP STATION OP UNIT 8 NAAR UNIT 9

BACKUP U9 TO U8      : REM BACKUP STATION OP UNIT 9 NAAR UNIT 8

BACKUP D0 TO D1, U10 : REM BACKUP OP EEN DUBBELDRIVE VERBONDEN VIA IEC
```

BANK

Token: $FE $02

Formaat: **BANK** bank getal

Gebruik: Selecteert de geheugenconfiguratie voor BASIC-commando's die 16-bit-adressen gebruiken. Dit zijn **LOAD**, **LOADIFF**, **PEEK**, **POKE**, **SAVE**, **SYS** en **WAIT**. Zie de geheugenindeling van het systeem in Hoofdstuk/Bijlage J op bladzijde J-3 voor meer informatie.

Opmerkingen:

Een waarde > 127 selecteert geheugentoegewezen I/O.

De standaardwaarde voor het banknummer bij het opstarten van het systeem is 128.

Deze configuratie heeft RAM van $0000 tot $1FFF, de BASIC- en KERNAL-ROM en I/O van $2000 tot $FFFF.

Voorbeeld: Gebruik van **BANK**

```
BANK 1 : REM KIES GEHEUGENCONFIGURATIE 1
```

BEGIN

Token: $FE $18

Formaat: **BEGIN** ... **BEND**

Gebruik: Het begin van een samengestelde instructie die na **THEN** of **ELSE** moet worden uitgevoerd. Hiermee wordt de beperking van één enkele regel van de standaard **IF** ... **THEN** ... **ELSE**-constructie overwonnen.

Opmerkingen:
Gebruik **GOTO** of **GOSUB** niet om in een samengestelde instructie te springen, omdat dit tot onverwachte resultaten kan leiden.

Voorbeeld: Gebruik van **BEGIN** ... **BEND**

```
10 GET A$
20 IF A$ >= "A" AND A$ <= "Z" THEN BEGIN : REM NEGEER ALLES BEHALVE (A-Z)
30 PW$ = PW$ + A$
40 IF LEN(PW$) > 7 THEN 90
50 BEND
60 IF A$ <> CHR$(13) GOTO 10
90 PRINT "PW="; PW$
```

BEND

Token: $FE $19

Formaat: **BEGIN** ... **BEND**

Gebruik: Het einde van een samengestelde instructie die na **THEN** of **ELSE** moet worden uitgevoerd. Hiermee wordt de beperking van één enkele regel van de standaard **IF** ... **THEN** ... **ELSE**-constructie overwonnen.

Opmerkingen:
Het onderstaande voorbeeld toont een eigenaardigheid in de implementatie van de samengestelde instructie. Indien de voorwaarde als **FALSE** wordt beoordeeld, wordt de uitvoering niet direct na **BEND** zoals het zou moeten, maar aan het begin van de volgende regel.

Voorbeeld: Gebruik van **BEGIN** ... **BEND**

```
5 REM DEMO'T DE BESCHREVEN EIGENAARDIGHEID
10 IF Z > 1 THEN BEGIN : A$ = "EEN"
20 B$ = "TWEE"
30 PRINT A$; " "; B$; : BEND : PRINT " RAAR!"
40 REM PROGRAMMA LOOPT HIER VERDER VOOR Z <= 1
```

BLOAD

Token: $FE $11

Formaat: **BLOAD** bestandsnaam [**,B** bank] [**,P** adres] [**,R**] [**,D** station] [**,U** eenheid]

Gebruik: Laadt een bestand van het type **PRG** in RAM op adres **P**. ("Binair laden.")

BLOAD heeft twee modi: De platte geheugenmodus kan worden gebruikt om een programma te laden op elk adres in het 28-bit (256 MB) adresbereik waar RAM aanwezig is. Dit omvat de standaard RAM-banken 0 tot 5, alsmede de 8 MB "zolder-RAM" op adres $8000000 (800.0000).

Deze modus wordt geactiveerd door een adres bij parameter **P** op te geven dat groter is dan $FFFF. De bankparameter wordt in deze modus genegeerd.

Om compatibiliteit met oudere BASIC-versies te waarborgen, accepteert **BLOAD** ook de syntaxis met een 16-bit-adres op P en een banknummer op B. De zolder-RAM valt buiten het bereik van deze compatibiliteitsmodus.

De optionele parameter **R** ("ruwe modus") interpreteert of gebruikt de eerste twee bytes van het bestand niet als laadadres, hetgeen anders het standaardgedrag is. In de ruwe modus wordt elke byte als gegevens gelezen.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

bank specificeert de te gebruiken RAM-bank. Indien niet opgegeven, wordt de huidige bank, zoals ingesteld met het laatste **BANK**-statement, gebruikt.

adres overschrijft het laadadres dat in de eerste twee bytes van het **PRG**-bestand is opgeslagen.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

BLOAD kan geen bankgrenzen overschrijden.

BLOAD gebruikt het laadadres uit het bestand indien geen **P**-parameter is opgegeven.

Het is mogelijk om **BLOAD** te dwingen een bestand van het type **SEQ** te laden met behulp van de bestandsnaamextensie: `",S"`. Zorg er ook voor

dat de ruwe modus is ingeschakeld (**,R**) om te voorkomen dat de eerste twee bytes als een **PRG**-laadadres worden behandeld.

Voorbeelden:

Gebruik van **BLOAD**

```
BLOAD "ML DATA", B0, U9

BLOAD "SPRITES"

BLOAD "ML ROUTINES", B1, P32768

BLOAD (FI$), B(BA%), P(PA), U(UN%)

BLOAD "CHUNK", P($8000000)

BLOAD "USER DATA,S", P($1600), R
```

BOOT

Token: $FE $1B

Formaat: **BOOT** bestandsnaam [**,B** bank][**,P** adres] [**,D** station] [**,U** eenheid]
BOOT SYS
BOOT

Gebruik: Laadt en voert een programma of bootsector van een schijf uit.

BOOT bestandsnaam laadt een bestand van het type **PRG** in het RAM op adres **P** en bank **B** en begint de code op het laadadres uit te voeren.

BOOT SYS laadt de bootsector (512 bytes in totaal) van sector 0, spoor 1 en apparaat 8 naar adres $0400 in bank 0 en voert daarna een JSR `$0400` uit (Jump To Subroutine).

BOOT zonder parameters probeert een bestand met de naam `AUTOBOOT.C65` van het standaardapparaat te laden en uit te voeren. Dit is een verkorte versie van: `RUN "AUTOBOOT.C65"`

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

bank geeft de te gebruiken RAM-bank op. Indien niet opgegeven, wordt de huidige bank gebruikt zoals die met het laatste **BANK**-commando is ingesteld.

adres overschrijft het laadadres dat in de eerste twee bytes van het **PRG**-bestand is opgeslagen.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Voorbeelden:
Gebruik van **BOOT**

```
BOOT SYS

BOOT (FI$), B(BA%), P(PA), U(UN%)

BOOT
```

BORDER

Token: $FE $3C

Formaat: **BORDER** kleur

Gebruik: Stelt de randkleur van het scherm in.

kleur paletindex in het bereik 0 - 255.

Alle kleuren in dit bereik zijn aanpasbaar met het **PALETTE**-commando. Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard systeempalet.

Voorbeeld: Gebruik van **BORDER**

```
BORDER 4 : REM KIES DE RANDKLEUR PAARS
```

BOX

Token: $E1

Formaat: **BOX** x0,y0, x2,y2 [, massief]
BOX x0,y0, x1,y1, x2,y2, x3,y3 [, massief]

Gebruik: Bitmapgrafiek: tekent een kader.

De eerste vorm van **BOX** met twee coördinaatparen en een optionele parameter **massief** tekent een eenvoudige rechthoek, waarbij de coördinaatparen twee diagonaal tegenoverliggende hoeken aangeven.

De tweede vorm met vier coördinaatparen geeft een pad van vier punten op, die met lijnen worden verbonden. Het pad wordt gesloten door de laatste coördinaat met de eerste te verbinden.

De vierhoek wordt getekend met de huidige tekencontext die met **SCREEN**, **PALETTE** en **PEN** is ingesteld. De vierhoek wordt gevuld indien de parameter **massief** niet 0 is.

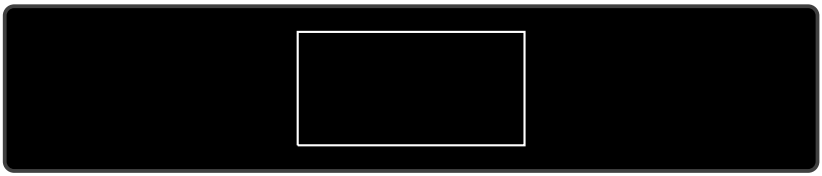
Opmerkingen:

BOX kan met vier coördinaatparen worden gebruikt om elke vorm te tekenen die met vier punten kan worden gedefinieerd, niet alleen rechthoeken. Bijvoorbeeld ruiten, vliegers, trapezoïden en parallellogrammen. Het is ook mogelijk strikachtige vormen te tekenen.

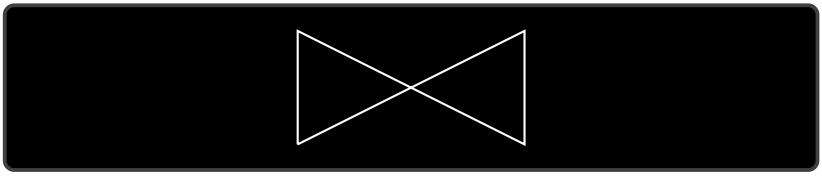
Voorbeelden:

Gebruik van **BOX**

```
BOX 0, 0,  160, 0,  160, 80,  0, 80
```



```
BOX 0, 0,  160, 80,  160, 0,  0, 80
```



```
BOX 20, 0,  140, 0, 160, 80,  0, 80
```



BSAVE

Token: $FE $10

Formaat: **BSAVE** bestandsnaam**, P** begin **TO P** einde [**,B** bank] [**,R**] [**,D** station] [**,U** eenheid]

Gebruik: Slaat een geheugenbereik op als een bestand van het type **PRG**. ("Binair opslaan.")

BSAVE heeft twee modi:

De vlakke geheugenmodus kan worden gebruikt om een geheugenblok op te slaan in het 28-bits (256 MB) adresbereik waarin het RAM is geïnstalleerd. Dit omvat de standaard RAM-banken 0 tot 5, alsmede de 8 MB "zolder-RAM" op adres $8000000 (800.0000).

Deze modus wordt geactiveerd door adressen voor de begin- en eindparameter **P** op te geven die groter zijn dan $FFFF. De bankparameter wordt in deze modus genegeerd. Met deze vlakke geheugenmodus kunnen bereiken groter dan 64 KB worden opgeslagen.

Om compatibiliteitsredenen met oudere BASIC-versies aanvaardt **BSAVE** ook de syntaxis met 16-bits adressen bij **P** en een banknummer bij **B**. De zolder-RAM valt buiten het bereik van deze compatibiliteitsmodus. Deze modus kan geen bankgrenzen overschrijden, dus het begin- en eindadres moeten zich in dezelfde bank bevinden.

De optionele parameter **R** ("ruwe modus") voorkomt dat het 16-bits startadres als de eerste twee bytes wordt geschreven. De adresheader van twee bytes is nodig voor PRG-bestanden die op het opgeslagen adres worden geladen, maar is doorgaans ongewenst voor SEQ-bestanden. **BSAVE** schrijft standaard het startadres als de eerste twee bytes van het bestand.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

Indien het eerste teken van de bestandsnaam het `@`-teken (`@`) is, wordt dit geïnterpreteerd als een "opslaan en vervangen"-operatie. Het wordt afgeraden deze optie op 1541- en 1571-stations te gebruiken, omdat deze een "save and replace bug" in hun DOS bevatten.

begin adres waar het opslaan begint. Dit wordt tevens het laadadres dat in de eerste twee bytes van het **PRG**-bestand wordt opgeslagen.

einde adres waar het opslaan eindigt. **einde-1** is het laatste adres dat voor het opslaan wordt gebruikt.

bank te gebruiken RAM-bank. Indien niet opgegeven, wordt de huidige bank gebruikt zoals die met het laatste **BANK**-commando is ingesteld.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

De lengte van het bestand is **einde - begin + 2**.

Indien het getal na een argumentletter geen decimaal getal is (zoals een hexadecimale constante of een variabelenaam), moet het tussen haakjes worden geplaatst.

Bij het opslaan van een bestand van het type **PRG** vormen de eerste twee bytes doorgaans het 16-bits laadadres. Dit is de standaardinstelling. Indien het startadres zich in een andere bank dan nul bevindt (het startadres is groter dan $FFFF), worden alleen de twee laagste bytes van het adres geschreven.

Om een bestand van het type **SEQ** op te slaan, gebruikt u het bestandsnaamsuffix ",S". Voor bestanden van dit type wordt het 16-bits laadadres doorgaans weggelaten. Geef hiervoor het argument **,R** om "ruwe modus" te activeren.

Voorbeelden:

Gebruik van **BSAVE**

```
BSAVE "ML DATA", P 32768 TO P 33792, B0, U9

BSAVE "SPRITES", P 1536 TO P 2058

BSAVE "ML ROUTINES", B1, P($9000) TO P($A000)

BSAVE (FI$), B(BA%), P(PA) TO P(PE), U(UN%)

BSAVE "USER DATA,S", P($1600) TO P($16FF), R
```

BUMP

Token: $CE $03

Formaat: **BUMP**(type)

Retourneert:

Een bitveld van sprites die op dit moment botsen met andere sprites of schermgegevens.

type het type botsing, waarbij 1:sprite-sprite en 2:sprite-schermgegevens.

Elke bit die in de retourwaarde is gezet, geeft aan dat de sprite die overeenkomt met die bitpositie betrokken is geweest bij een botsing sinds het laatste aanroepen van **BUMP**. Het aanroepen van **BUMP** reset de botsingsmasker, zodat u altijd een overzicht krijgt van de botsingen sinds het laatste aanroepen van **BUMP**.

Sprite	Retour-neert	Masker
0	1	0000 0001
1	2	0000 0010
2	4	0000 0100
3	8	0000 1000
4	16	0001 0000
5	32	0010 0000
6	64	0100 0000
7	128	1000 0000

Opmerkingen:

Het is mogelijk meerdere botsingen te detecteren, maar u moet de sprite-coördinaten evalueren om te bepalen welke sprites met elkaar gebotst hebben.

Voorbeeld: Gebruik van **BUMP**

```
10 S% = BUMP(1) : REM BOTSINGEN SPRITE (-) SPRITE
20 IF (S% AND 6) = 6 THEN PRINT "SPRITE 1 & 2 BOTSEN"
30 REM ---
40 S% = BUMP(2) : REM BOTSINGEN SPRITE (-) SCHERMGEGEVENS
50 IF (S% <> 0) THEN PRINT "EEN SPRITE RAAKT SCHERMGEGEVENS"
```

BVERIFY

Token: $FE $28

Formaat: **BVERIFY** bestandsnaam [**,P** adres] [**,B** bank] [**,D** station] [**,U** eenheid]

Gebruik: Vergelijkt een geheugenbereik met een bestand van het type **PRG**. ("Binair verifiëren.")

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

bank geeft de te gebruiken RAM-bank op. Indien niet opgegeven, wordt de huidige bank gebruikt zoals die met het laatste **BANK**-commando is ingesteld.

adres adres waar de vergelijking begint. Indien de parameter **P** wordt weggelaten, wordt het laadadres gebruikt dat in de eerste twee bytes van het **PRG**-bestand is opgeslagen.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

BVERIFY kan alleen op gelijkheid testen. In de directe modus geeft **BVERIFY** ofwel het bericht OK ofwel VERIFY ERROR met het adres van het eerste niet-identieke byte. In de programmamodus stopt een Verify-fout de uitvoer of gaat naar de **TRAP**-foutafhandeling, indien deze actief is.

Voorbeelden:

Gebruik van **BVERIFY**

```
BVERIFY "ML DATA", P 32768, B0, U9

BVERIFY "SPRITES", P 1536

BVERIFY "ML ROUTINES", B1, P(DEC("9000"))

BVERIFY (FI$), B(BA%), P(PA), U(UN%)
```

C@&

Token: N.v.t.

Formaat: **C@&**(kolom**,** rij**)**

Gebruik: Leest of stelt de kleurcode van een teken op de gegeven schermcoördinaten in.

Dit gedraagt zich als een arrayvariabele. Door een waarde toe te wijzen, wordt de kleurcode voor het gegeven teken onmiddellijk gewijzigd.

Opmerkingen:

C@& is een gereserveerde systeemvariabele.

Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard systeempalet. Bovenste bits stellen ook tekstattributen in, zoals knipperen.

De coördinaten beginnen bij (0, 0) voor de linkerbovenhoek. Indien de gegeven coördinaten buiten de huidige schermmodus vallen, geeft het openen van de array een Bad Subscript-fout.

Voorbeeld: Zie **T@&**.

CATALOG

Token: $FE $0C

Formaat: **CATALOG** [bestandspatroon] [**,W**] [**,R**] [**,D** station] [**,U** eenheid]
$ [bestandspatroon] [**,W**] [**,R**] [**,D** station] [**,U** eenheid]

Gebruik: Drukt een bestandscatalogus/map van de opgegeven schijf af.

De parameter **W** (Breed) geeft de map in drie kolommen op het scherm weer en stopt nadat het scherm is gevuld met een pagina (63 mapitems). Door op een willekeurige toets te drukken, wordt de volgende pagina weergegeven.

De parameter **R** (Herstelbaar) toont alleen de bestanden die als verwijderd zijn gemarkeerd maar nog steeds herstelbaar zijn, elk gemarkeerd met een '–'. Andere bestanden worden niet getoond.

bestandspatroon is een aangehaalde tekenreeks, bijvoorbeeld: `"DA*"` of een tekenreeksexpressie tussen haakjes, bijv. `(DI$)`

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

CATALOG is een synoniem van **DIRECTORY** en **DIR** en geeft dezelfde lijst.

Het patroon **bestandspatroon** kan worden gebruikt om de lijst te filteren. De tekens ***** en **?** kunnen als jokerteken worden gebruikt. Door **,T=** aan de patroonreeks toe te voegen, waarbij **T** een bestandstype aangeeft van **P**, **S**, **U**, **R** of **C** (voor **P**RG, **S**EQ, **U**SR, **R**EL, **C**BM), wordt de uitvoer op dat bestandstype gefilterd.

Het afkortingssymbool **$** kan alleen in de directe modus worden gebruikt.

Voorbeelden:

Gebruik van **CATALOG**

```
CATALOG
0 "BLACK SMURF     " BS 2A
508  "STORY PHOBOS"    SEQ
27   "C8096"           PRG
25   "C128"            PRG
104 BLOCKS FREE.
```

```
CATALOG "*,T=S"
0 "BLACK SMURF     " BS 2A
508  "STORY PHOBOS"    SEQ
104 BLOCKS FREE.
```

Gebruik van de brede parameter (**W**)

```
CATALOG W
0 "BASIC EXAMPLES  "
1 "BEGIN"          P    1 "FREAD"          P    2 "PAINT.COR"      P
1 "BEND"           P    1 "FRE"            P    3 "PALETTE.COR"    P
1 "BUMP"           P    2 "GET#"           P    1 "PEEK"           P
1 "CHAR"           P    1 "GETKEY"         P    3 "PEN"            P
1 "CHR$"           P    1 "GET"            P    1 "PLAY"           P
4 "CIRCLE"         P    2 "GOSUB"          P    2 "POINTER"        P
1 "CLOSE"          P    2 "GOTO.COR"       P    1 "POKE"           P
1 "CLR"            P    2 "GRAPHIC"        P    1 "POS"            P
2 "COLLISION"      P    1 "HELP"           P    1 "POT"            P
1 "CURSOR"         P    1 "IF"             P    1 "PRINT#"         P
0 "DATA BASE"      R    2 "INPUT#"         P    1 "PRINT"          P
1 "DATA"           P    2 "INPUT"          P    1 "RCOLOR.COR"     P
1 "DEF FN"         P    2 "JOY"            P    1 "READ"           P
1 "DIM"            P    1 "LINE INPUT#"    P    1 "RECORD"         P
1 "DO"             P    3 "LINE"           P    1 "REM"            P
5 "ELLIPSE"        P    1 "LOOP"           P    1 "RESTORE"        P
1 "ELSE"           P    1 "MID$"           P    1 "RESUME"         P
1 "EL"             P    1 "MOD"            P    1 "RETURN"         P
1 "ENVELOPE"       P    1 "MOVSPR"         P    1 "REVERS"         S
2 "EXIT"           P    1 "NEXT"           P    3 "RGRAPHIC"       P
1 "FOR"            P    2 "ON"             P    1 "RMOUSE"         P
```

CHANGE

Token: $FE $2C

Formaat: **CHANGE /**zoekstring**/ TO /**vervangstring**/** [**,** regel bereik]
CHANGE "zoekstring**" TO "**vervangstring**"** [**,** regel bereik]

Gebruik: Bewerkt het BASIC-programma dat zich momenteel in het geheugen bevindt om alle voorkomens van één tekenreeks door een andere te vervangen.

Een optioneel **regelbereik** beperkt de zoekactie tot dit bereik, anders wordt het volledige BASIC-programma doorzocht. Bij elk voorkomen van de **zoekstring** wordt de regel weergegeven en wordt de gebruiker om een actie gevraagd:

- **Y** RETURN voert de vervanging uit en zoekt de volgende tekenreeks.
- **N** RETURN voert **geen** vervanging uit en zoekt de volgende tekenreeks.
- ***** RETURN vervangt de huidige en alle volgende overeenkomsten.
- RETURN beëindigt het commando en vervangt de huidige overeenkomst niet.

Opmerkingen:

Vrijwel elk teken dat geen onderdeel van de tekenreeks is, inclusief letters en leestekens, kan in plaats van **/** worden gebruikt.

Het gebruik van het dubbele aanhalingsteken zoekt echter naar tekenreeksen die niet getokeniseerd zijn en dus geen onderdeel van een trefwoord vormen. Het voorbeeld `CHANGE "LOOP" TO "OOPS"` zal het BASIC-trefwoord `LOOP` niet vinden, omdat het trefwoord als token en niet als tekst wordt opgeslagen. `CHANGE /LOOP/ TO /OOPS/` vindt en vervangt het wel (wat tot syntaxisfouten kan leiden).

Vanwege een beperking van de BASIC-parser kan **CHANGE** de trefwoorden **REM** en **DATA** niet herkennen. Zie **FIND**.

Kan alleen in de directe modus worden gebruikt.

Voorbeelden:

Gebruik van **CHANGE**

```
CHANGE "XX$" TO "UU$", 2000-2700

CHANGE /IN/ TO /OUT/

CHANGE &IN& TO &OUT&
```

CHAR

Token: $E0

Formaat: **CHAR** kolom**,** rij**,** hoogte**,** breedte**,** richting**,** tekenreeks [**,** adres van tekenset]

Gebruik: Bitmapgraphics: geeft tekst weer op een grafisch scherm.

kolom (in eenheden van tekenposities) begint de positie van de uitvoer horizontaal. Omdat één kolomeenheid 8 pixels breed is, heeft een schermbreedte van 320 een kolombereik van 0 - 39, terwijl een schermbreedte van 640 een kolombereik van 0 - 79 heeft.

rij (in pixel-eenheden) begint de positie van de uitvoer verticaal. In tegenstelling tot de kolomparameter is de eenheid pixels (geen tekenposities), waarbij de bovenste rij de waarde 0 heeft.

hoogte factor die op de verticale grootte van de tekens wordt toegepast, waarbij 1 de normale grootte (8 pixels) is, 2 de dubbele grootte (16 pixels), enzovoort.

breedte factor die op de horizontale grootte van de tekens wordt toegepast, waarbij 1 de normale grootte (8 pixels) is, 2 de dubbele grootte (16 pixels), enzovoort.

richting bepaalt de afdrukrichting:

- 1: omhoog
- 2: rechts
- 4: omlaag
- 8: links

Het optionele **adres van tekenset** kan worden gebruikt om een tekenset te selecteren. Het standaardadres van de tekenset is $29800, het kleine-letter+hoofdlettergedeelte van lettertype A (PETSCII met enkele ASCII-vervangingen).

U kunt elk adres gebruiken waar een tekenset is opgeslagen, inclusief een eigen tekenset die u in het geheugen heeft geladen. Er zijn drie ingebouwde tekensets (zie ook **FONT**), alsmede de tekensetbuffer van de VIC-IV:

- $29000: lettertype A (ASCII)
- $3D000: lettertype B (vetgedrukt)
- $2D000: lettertype C (CBM)
- $FF7E000: tekensetbuffer van de VIC-IV

Om **CHAR** te gebruiken met eigen tekens die met **CHARDEF** zijn gedefinieerd, gebruikt u het adres van de tekensetbuffer van de VIC-IV, $FF7E000.

Een tekenset met de PETSCII-indeling heeft twee subsets: de set met hoofdletters en grafische tekens en de set met kleine letters en hoofdletters. Met **CHAR** kunt u een subset selecteren op basis van het adres: $xx000 voor hoofdletters en grafische tekens en $xx800 voor kleine letters en hoofdletters. Gebruik bijvoorbeeld tekensetadres $2D800 om de kleine letters van lettertype C te selecteren.

tekenreeks constante of expressie die wordt afgedrukt. Deze tekenreeks mag optioneel een of meer van de volgende besturingstekens bevatten:

Expressie	Sneltoets	Zinnebeeld	Beschrijving
CHR$(2)	CTRL+B	B	Lege cel
CHR$(6)	CTRL+F	F	Teken omdraaien
CHR$(9)	CTRL+I	I	EN met scherm
CHR$(15)	CTRL+O	O	OF met scherm
CHR$(24)	CTRL+X	X	XOR met scherm
CHR$(18)	RVSON	R	Omgekeerd
CHR$(146)	RVSOFF	■	Omgekeerd uit
CHR$(147)	CLR	▢	Venster wissen
CHR$(21)	CTRL+U	U	Onderstrepen
CHR$(25)+"-"	CTRL+Y + "-"	Y–	Links draaien
CHR$(25)+"+"	CTRL+Y + "+"	Y+	Rechts draaien
CHR$(26)	CTRL+Z	Z	Spiegelen
CHR$(157)	Cursor links	▌	Links verplaatsen
CHR$(29)	Cursor rechts	▐	Rechts verplaatsen
CHR$(145)	Cursor op	▢	Omhoog verplaatsen
CHR$(17)	Cursor neer	Q	Omlaag verplaatsen

Opmerkingen:

Merk op dat de beginpositie van de tekenreeks verschillende eenheden heeft in de horizontale en verticale richting. Horizontaal is in kolommen en verticaal in pixels.

Raadpleeg de **CHR$**-functie op bladzijde B-48 voor meer informatie.

Voorbeeld: Gebruik van **CHAR**

```
10 SCREEN 640, 400, 2
20 CHAR 28, 180, 4, 4, 2, "MEGA65", $29000
30 GETKEY A$
40 SCREEN CLOSE
```

CHARDEF

Token: $E0 $96

Formaat: **CHARDEF** index, bitmatrix

Gebruik: Wijzigt de weergave van een teken.

index schermcode van het teken dat moet worden gewijzigd (@:0, A:1, B:2, enz.). Zie bijlage C op bladzijde C-3 voor een lijst van schermcodes. Schermcodes 0 - 255 behoren tot de set van hoofdletters en grafische tekens. Schermcodes 256 - 511 behoren tot de set van kleine letters en hoofdletters.

bitmatrix reeks van 8 byte-waarden die de rasterweergave van het teken definiëren, van de bovenste rij naar de onderste rij. Wanneer meer dan 8 waarden als argumenten worden gebruikt, worden waarden 9 - 16 gebruikt voor het teken index+1, 17 - 24 voor index+2, enzovoort.

Opmerkingen:

De wijzigingen in de bitmap van het teken worden toegepast op de tekensetbuffer van de VIC, die zich in het RAM bevindt op het adres FF7.E000.

Alle wijzigingen zijn tijdelijk. De tekensetbuffer van de VIC wordt overschreven door het commando **FONT** en wordt door een reset hersteld naar PETSCII-lettertype C.

Voorbeelden:

Gebruik van **CHARDEF**

```
CHARDEF 1, $FF, $81, $81, $81, $81, $81, $81, $FF : REM VERANDER 'A' IN RECHTHOEK

CHARDEF 9, $18, $18, $18, $18, $18, $18, $18, $00 : REM MAAK 'I' CURSIEF
```

CHDIR

Token: $FE $4B

Formaat: **CHDIR** mapnaam [**,U** eenheid]

Gebruik: Wijzigt de huidige werkmap.

mapnaam de naam van een map. Ofwel een aangehaalde tekenreeks zoals "EENOFANDEREMAP", of een tekenreeksexpressie in haakjes zoals: (DR$)

Afhankelijk van het type **eenheid** wordt **CHDIR** op verschillende bestandssystemen toegepast.

UNIT 12 is gereserveerd voor de SD-kaart (FAT-bestandssysteem). Dit commando kan worden gebruikt om door submappen te navigeren en schijfimages die daar zijn opgeslagen te koppelen. **CHDIR "..",U12** schakelt over naar de bovenliggende map op **UNIT 12**.

Voor andere eenheden die door CBDOS worden beheerd (doorgaans 8 en 9) wordt **CHDIR** gebruikt om naar submappen op een diskette of schijfimage van het type **D81** te wisselen of deze te verlaten. Bestaande submappen worden weergegeven als bestandstype **CBM** in de bovenliggende map en worden aangemaakt met het commando **MKDIR**. **CHDIR "/",U eenheid** schakelt over naar de hoofdmap.

Voorbeelden:

Gebruik van **CHDIR**

```
CHDIR "AVONTUUR", U12 : REM GA AVONTUUR OP SD-KAART BINNEN
CHDIR "..", U12       : REM GA TERUG NAAR BOVENLIGGENDE DIRECTORY
CHDIR "RACEBAAN", U12 : REM GA SUBDIRECTORY RACEBAAN BINNEN
0 "MEGA65          "   1D
800  "MEGA65-SPELLEN"   CBM
800  "MEGA65-TOOLTJES"  CBM
600  "BASICPROGJES"     CBM
960  BLOCKS FREE.

CHDIR "MEGA65 GAMES", U8 : REM GA SUBDIRECTORY OP DISKETTE BINNEN
CHDIR "/", U8            : REM GA TERUG NAAR DE HOOFDDIRECTORY
```

CHR$

Token: $C7

Formaat: **CHR$(**numerieke expressie**)**

Retourneert:
Een tekenreeks met één teken van de opgegeven PETSCII-waarde.

Opmerkingen:
Het bereik van het argument loopt van 0 - 255, zodat deze functie ook kan worden gebruikt om besturingstekens in te voegen in tekenreeksen. Zelfs het NULL-teken met code 0 is toegestaan. **CHR$** is de omgekeerde functie van **ASC**. De volledige tabel van tekens (en hun PETSCII-codes) staat op bladzijde D-3.

Voorbeeld: Gebruik van **CHR$**

```
10 QUOTE$  = CHR$(34)
20 ESCAPE$ = CHR$(27)
30 PRINT QUOTE$; "MEGA65"; QUOTE$ : REM PRINT "MEGA65"
40 PRINT ESCAPE$; "Q";            : REM WIS TOT EINDE REGEL
```

CIRCLE

Token: $E2

Formaat: **CIRCLE** xc, yc, straal [, vlaggen, begin, einde]

Gebruik: Bitmapgraphics: tekent een cirkel. Dit is een speciaal geval van **ELLIPSE**, waarbij dezelfde waarde voor de horizontale en verticale straal wordt gebruikt.

xc de x-coördinaat van het middelpunt in pixels.

yc de y-coördinaat van het middelpunt in pixels.

straal de straal van de cirkel in pixels.

vlaggen bepaalt het vullen, de bogen en de positie van de nulradiaal. De standaardinstelling (nul) is niet vullen, de poten van een boog weergeven en de nulradiaal wijst naar 3 uur.

Bit	Naam	Waarde	Actie indien ingesteld
0	vulling	1	Vul ellips of boog met de huidige penkleur
1	poten	2	Onderdruk het tekenen v/d poten van een boog
2	combs	4	Laat de nulradiaal naar 12 uur wijzen

De eenheden voor de begin- en eindhoek zijn graden in het bereik van 0 tot 360. De nulradiaal begint bij 3 uur en beweegt met de klok mee. Het instellen van bit 2 van vlaggen (waarde 4) verplaatst de nulradiaal naar de 12-uurpositie.

begin de beginhoek van de boog.

einde de eindhoek van de boog.

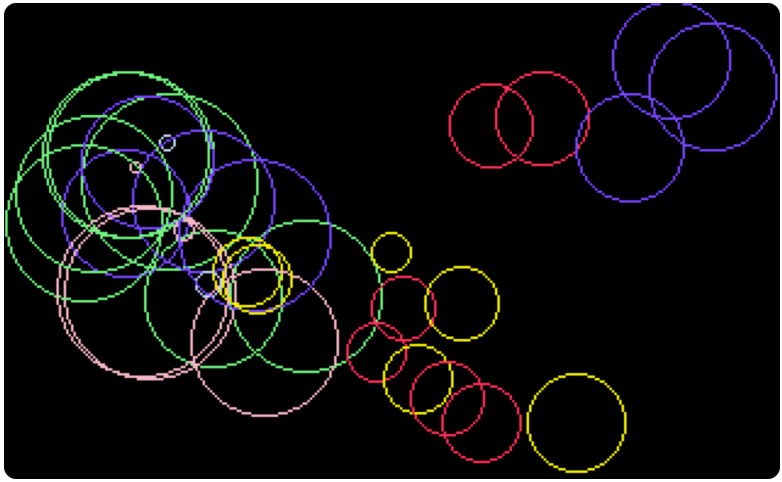
Opmerkingen:

CIRCLE wordt gebruikt om cirkels te tekenen op schermen met een verhouding van 1:1 (bijvoorbeeld: 320 x 200 of 640 x 400). Bij andere resoluties (zoals 640 x 200) wordt de vorm een ellips.

Voorbeeld: Gebruik van **CIRCLE**

```
100 REM CIRKEL (NAAR F.BOWEN)
110 BORDER 0                                          : REM ZWART
120 SCREEN 320, 200, 4                                : REM EENVOUDIGE SCHERMOPZET
130 PALETTE 0, 0,          0,          0,          0 : REM ZWART
140 PALETTE 0, 1, RND(.) * 16, RND(.) * 16,         15 : REM LUKRAKE KLEUREN
150 PALETTE 0, 2, RND(.) * 16,         15, RND(.) * 16
160 PALETTE 0, 3,         15, RND(.) * 16, RND(.) * 16
170 PALETTE 0, 4, RND(.) * 16, RND(.) * 16,         15
180 PALETTE 0, 5, RND(.) * 16,         15, RND(.) * 16
190 PALETTE 0, 6,         15, RND(.) * 16, RND(.) * 16
200 SCNCLR 0                                          : REM WIS SCHERM
210 FOR I = 0 TO 32                                   : REM CIRKELLUS
220 PEN 0, RND(.) * 6 + 1                             : REM LUKRAKE PEN
230 R  = RND(.) * 36 + 1                              : REM STRAAL
240 XC = R + RND(.) * 320 : IF (XC + R) ) 319 THEN 240  : REM X-CENTRUM
250 YC = R + RND(.) * 200 : IF (YC + R) ) 199 THEN 250  : REM Y-CENTRUM
260 CIRCLE XC, YC, R, .                               : REM TEKEN
270 NEXT
280 GETKEY A$                                         : REM WACHT OP TOETS
290 SCREEN CLOSE : BORDER 6
```

Het voorbeeldprogramma gebruikt de willekeurige-getallenfunctie **RND** voor de kleur, grootte en positie van de cirkel. Het toont daarom bij elke uitvoering een ander beeld:



CLOSE

Token: $A0

Formaat: **CLOSE** kanaal

Gebruik: Sluit een invoer- of uitvoerkanaal.

kanaal nummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

Opmerkingen:

Het sluiten van bestanden die voor het einde van het programma zijn geopend, is zeer belangrijk, vooral voor uitvoerbestanden. **CLOSE** maakt uitvoerbuffers leeg en werkt de informatie van de map op schijven bij.

Het niet uitvoeren van **CLOSE** kan bestanden en schijven beschadigen.

BASIC sluit kanalen en bestanden *niet* automatisch wanneer een programma stopt.

Voorbeeld: Gebruik van **CLOSE**

```
10 OPEN 2, 8, 2, "TEST,S,W"
20 PRINT#2, "TEKENREEKSJE TER TEST"
30 CLOSE 2              : REM CLOSE VERZUIMEN BETEKENT EEN ZOMBIEBESTAND
```

CLR

Token: $9C

Formaat: **CLR**
CLR variabele

Gebruik: Wist het variabelengeheugen van BASIC.

Na het uitvoeren van **CLR** worden alle variabelen en matrices niet gedeclareerd. De aanwijzers van de run-time-stack en de tabel met geopende kanalen worden eveneens gereset.

Wanneer een **variabele** wordt opgegeven, wordt deze gewist (op nul gezet). **variabele** kan een numerieke variabele of een tekenreeksvariabele zijn, maar geen matrix.

Opmerkingen:

CLR mag niet worden gebruikt in lussen of subroutines, omdat dit het retouradres vernietigt.

Na **CLR** zijn alle variabelen onbekend en worden ze geïnitialiseerd bij het volgende gebruik.

RUN voert **CLR** automatisch uit.

Voorbeeld: Gebruik van **CLR**

```
10 A = 5 : P$ = "MEGA65"
20 CLR
30 PRINT A; P$

RUN
 0
```

CLRBIT

Token: $9C $FE $4E

Formaat: **CLRBIT** adres**,** bitnummer

Gebruik: Wist (zet terug) een enkele bit op het **adres**.

Wanneer het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank die met **BANK** is ingesteld, gebruikt.

Adressen groter dan of gelijk aan $10000 (65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

bitnummer waarde van 0 - 7.

Opmerkingen:

CLRBIT is een verkorte versie van het gebruik van een bitsgewijze **AND** om een bit te wissen, doch u kunt slechts één bit tegelijk wissen. Zie **SETBIT** om een bit in te stellen.

Voorbeeld: Gebruik van **CLRBIT**

```
10 BANK 128       : REM KIES SYSTEEMMAPPING
20 CLRBIT $D011, 4 : REM SCHAKEL SCHERM UIT
30 CLRBIT $D016, 3 : REM SCHAKEL NAAR 38- OF 76-KOLOMSMODE
```

CMD

Token: $9D

Formaat: **CMD** kanaal [**,** tekenreeks]

Gebruik: Verlegt de standaarduitvoer van het scherm naar een kanaal.

Hiermee kunt u programmalijsten en mappen naar andere uitvoerkanalen afdrukken. Het is ook mogelijk deze uitvoer om te leiden naar een bestand op schijf of een modem.

kanaal nummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

De optionele **tekenreeks** wordt naar het kanaal gestuurd voordat de verlegging begint en kan bijvoorbeeld worden gebruikt voor escapesequenties voor de instelling van een printer of modem.

Opmerkingen:

De **CMD**-modus wordt gestopt met **PRINT#** of door het kanaal te sluiten met **CLOSE**. Het wordt aanbevolen om **PRINT#** te gebruiken voordat u sluit, zodat de uitvoerbuffer gegarandeerd wordt geleegd.

Voorbeeld: Gebruik van **CMD**

```
OPEN 1, 4 : REM LIST OP EEN PRINTER

CMD 1

LIST

PRINT#1

CLOSE 1
```

COLLECT

Token: $F3

Formaat: **COLLECT** [**,D** station] [**,U** eenheid]

Gebruik: Herstelt de Block Availability Map (BAM) van een schijf, waarbij bestanden die niet goed zijn gesloten (splat-bestanden) worden verwijderd en ongebruikte blokken als vrij worden gemarkeerd.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Hoewel dit commando nuttig is om een schijf van splat-bestanden te reinigen, is het gevaarlijk voor schijven met opstartblokken of willekeurig toegankelijke bestanden. Deze blokken zijn niet gekoppeld aan standaard schijfbestanden en worden daarom als vrij gemarkeerd, waardoor ze door verdere schrijfbewerkingen kunnen worden overschreven.

Voorbeelden:

Gebruik van **COLLECT**

```
COLLECT

COLLECT U9

COLLECT D0, U9
```

COLLISION

Token: $FE $17

Formaat: **COLLISION** type [**,** regelnummer]

Gebruik: Schakelt een gebruikergestuurde interrupt-handler voor sprite-collisions in of uit.

Indien een handler is ingeschakeld, onderbreekt een sprite-collision van het opgegeven **type** het BASIC-programma en voert een **GOSUB** uit naar **regelnummer**. Deze handler moet de besturing teruggeven met **RETURN**.

type collisiontype voor deze interrupt-handler:

Type	Beschrijving
1	Sprite - Sprite Collision
2	Sprite - Data - Collision
3	Light Pen

regelnummer regelnummer van een subroutine die de sprite-collision afhandelt en eindigt met **RETURN**.

Een aanroep zonder de parameter regelnummer schakelt de handler uit.

Opmerkingen:

Het is mogelijk de interrupt-handler voor alle typen in te schakelen, maar er kan er slechts één tegelijk worden uitgevoerd. Een interrupt-handler kan niet worden onderbroken door een andere interrupt-handler.

Functies zoals **BUMP**, **LPEN** en **RSPPOS** kunnen worden gebruikt om de betrokken sprites en hun posities te evalueren.

COLLISION was niet voltooid in BASIC 10, maar is wel beschikbaar in BASIC 65.

Voorbeeld: Gebruik van **COLLISION**

```
10 COLLISION 1, 70                                : REM INSCHAKELEN
20 SPRITE 1, 1 : MOVSPR 1, 120,   0 : MOVSPR 1, 0#5
30 SPRITE 2, 1 : MOVSPR 2, 120, 100 : MOVSPR 2, 180#5
40 FOR I = 1 TO 50000 : NEXT
50 COLLISION 1                                    : REM UITSCHAKELEN
60 END
70 REM SPRITE <-> SPRITE INTERRUPTHANDLER
80 PRINT "BUMP RETURNS"; BUMP(1)
90 RETURN                                         : REM KEER TERUG UIT INTERRUPT
```

COLOR

Token: $E7

Formaat: **COLOR** kleur

Gebruik: Stelt de voorgrondtekstkleur in voor opvolgende **PRINT**-commando's.

kleur paletindexnummer in het bereik 0 - 31.

Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard systeempalet.

Opmerkingen:

Dit is een andere naam voor **FOREGROUND**.

Voorbeeld: Gebruik van **COLOR**

```
COLOR 2 : PRINT "DIT IS ROOD"

COLOR 3 : PRINT "DIT IS CYAAN"
```

CONCAT

Token: $FE $13

Formaat: **CONCAT** toevoegbestand [**,D** station] **TO** doelbestand [**,D** station] [**,U** eenheid]

Gebruik: Voegt de inhoud van het **toevoegbestand** samen met het **doelbestand**. Na uitvoering bevat het **doelbestand** de inhoud van beide bestanden, terwijl het **toevoegbestand** ongewijzigd blijft.

toevoegbestand is een aangehaalde tekenreeks, bijvoorbeeld: "DATA" of een tekenreeksexpressie tussen haakjes, bijvoorbeeld: (FI$)

doelbestand is een aangehaalde tekenreeks, bijvoorbeeld: "SAFE" of een tekenreeksexpressie tussen haakjes, bijvoorbeeld: (FS$)

Als het schijfstation twee stations heeft, is het mogelijk **CONCAT** toe te passen op bestanden die op verschillende schijven zijn opgeslagen. In dat geval moet het station# voor beide bestanden worden opgegeven. Dit is ook noodzakelijk indien beide bestanden op station#1 zijn opgeslagen.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

CONCAT wordt uitgevoerd door het DOS van het schijfstation.

Beide bestanden moeten bestaan en patronen (jokertekens) zijn niet toegestaan.

Alleen bestanden van het type **SEQ** kunnen worden samengevoegd.

Voorbeelden:

Gebruik van **CONCAT**

```
CONCAT "NIEUWE DATA" TO "ARCHIEF", U9

CONCAT "ADRES", D0 TO "ADRESBOEK", D1
```

CONT

Token: $9A

Formaat: **CONT**

Gebruik: Hervat de programma-uitvoering na een onderbreking of stop als gevolg van een **END**- of **STOP**-instructie, of na het indrukken van RUN STOP.

Dit is een nuttig hulpmiddel bij het foutzoeken. Het BASIC-programma kan worden gestopt, waarna variabelen kunnen worden bekeken en zelfs gewijzigd.

De **CONT**-instructie hervat de uitvoering.

Opmerkingen:

CONT kan niet worden gebruikt indien het programma is gestopt als gevolg van een fout. Daarnaast belemmert elke bewerking van het programma de voortzetting. Het stoppen en hervatten kan de schermuitvoer verstoren en kan ook de invoer/uitvoer beïnvloeden.

Voorbeeld: Gebruik van **CONT**

```
10 I = I + 1 : GOTO 10

RUN

BREAK IN 10
READY.
PRINT I
 947

CONT
```

COPY

Token: $F4

Formaat: **COPY** bron [**,D** station] [**,U** eenheid] **TO** [doel] [**,D** station] [**,U** eenheid]

Gebruik: Kopieert een bestand naar een ander bestand, of een of meer bestanden van de ene schijf naar de andere.

bron een aangehaalde tekenreeks, bijv. "DATA" of een tekenreeksexpressie tussen haakjes, bijv. (FI$)

doel een aangehaalde tekenreeks, bijv. "BACKUP" of een tekenreeksexpressie tussen haakjes, bijv. (F$$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Indien geen of één **eenheid**nummer is opgegeven, of indien de eenheidnummers voor en na het **TO**-token gelijk zijn, wordt **COPY** op het schijfstation zelf uitgevoerd en bevinden de bron- en doelbestanden zich op dezelfde schijf.

Indien de bron-eenheid (voor **TO**) verschilt van de doeleenheid (na **TO**), voert **COPY** een door de CPU gestuurde routine uit die de bronbestanden in een RAM-buffer leest en naar de doeleenheid schrijft. In dat geval kan de doelbestandsnaam niet worden gekozen; deze is gelijk aan de bronbestandsnaam. Met de uitgebreide eenheid-naar-eenheid kopieermodus kunnen enkele bestanden, bestanden volgens een patroon of alle bestanden van een schijf worden gekopieerd. Elke combinatie van eenheden is toegestaan: interne diskette, D81-schijfimages, IEC-diskettestations zoals de 1541, 1571, 1581, of CMD-diskettenstations en hardeschijfstations.

Opmerkingen:

De bestandstypen **PRG**, **SEQ** en **USR** kunnen worden gekopieerd. Indien bron en doel zich op dezelfde schijf bevinden, moet de doelbestandsnaam verschillen van de bronbestandsnaam.

COPY kan geen **DEL**-bestanden kopiëren; deze worden doorgaans gebruikt als titels of scheidingstekens in directory's. Deze voldoen niet aan de Commodore DOS-regels en kunnen niet worden benaderd met standaard **OPEN**-routines.

REL-bestanden kunnen niet van eenheid naar eenheid worden gekopieerd.

Voorbeelden:

Gebruik van **COPY**

```
COPY U8 TO U9           : REM KOPIEER ALLE BESTANDEN
COPY "CODES" TO "BACKUP" : REM KOPIEER ENKEL BESTAND
COPY "*.TXT", U8 TO U9   : REM KOPIEER VOLGENS PATROON
COPY "M*", U9 TO U11     : REM KOPIEER VOLGENS PATROON
```

COS

Token: $BE

Formaat: **COS**(expressie in radialen)
COSD(expressie in graden)

Retourneert:

De cosinus van een hoek.

De **COS**-functie verwacht een argument in radialen.

De **COSD**-variant verwacht een argument in graden.

Het resultaat ligt in het bereik (-1.0 tot +1.0).

Voorbeelden:

Gebruik van **COS**

```
PRINT COS(0.7)
 0.76484219

X = 60 : PRINT COS(X * π / 180)
 0.5

PRINT COSD(60)
 0.5
```

CURSOR

Token: $FE $41

Formaat: **CURSOR** <**ON** | **OFF**> [{**,** kolom**,** rij**,** stijl}]
CURSOR kolom**,** rij

Gebruik: Verplaatst de tekstcursor naar de opgegeven positie op het huidige tekstscherm.

ON of **OFF** toont of verbergt de cursor. Wanneer de cursor op **ON** staat, verschijnt deze op de cursorpositie tijdens **GETKEY**.

kolom en **rij** specificeren de nieuwe positie.

stijl stelt een vaste (1) of knipperende (0) cursor in.

Opmerkingen:

Wanneer de cursor "aan" staat, kan het verplaatsen van de cursor met **PRINT** weergaveartefacten veroorzaken waarbij het cursor-aan-teken zichtbaar blijft wanneer een PETSCII-code de cursor verplaatst. In het gebruikelijke geval waarbij **PRINT** wordt gebruikt om door de gebruiker getypte tekens weer te geven, is de aanbevolen techniek om de cursor uit te schakelen voordat er wordt afgedrukt:

```
10 CURSOR ON
20 GETKEY A$
30 CURSOR OFF
40 PRINT A$
50 GOTO 10
```

Voorbeeld: Gebruik van **CURSOR**

```
 10 SCNCLR
 20 CURSOR 1, 2
 30 PRINT "A"; : SLEEP 1
 40 PRINT "B"; : SLEEP 1
 50 PRINT "C"; : SLEEP 1
 60 CURSOR 20, 10
 70 PRINT "D"; : SLEEP 1
 80 CURSOR ,5          : REM ZET DE CURSOR OP RIJ 5 ZONDER VAN KOLOM TE VERANDEREN
 90 PRINT "E"; : SLEEP 1
100 CURSOR 0           : REM ZET DE CURSOR AAN HET BEGIN VAN DE RIJ
110 PRINT "F"; : SLEEP 1
```

CUT

Token: $E4

Formaat: **CUT** x, y, breedte, hoogte

Gebruik: Bitmapgrafiek: kopieert de inhoud van het opgegeven rechthoek met linkerbovenpositie **x, y** en de **breedte** en **hoogte** naar een buffer en vult het gebied vervolgens met de kleur van de geselecteerde pen.

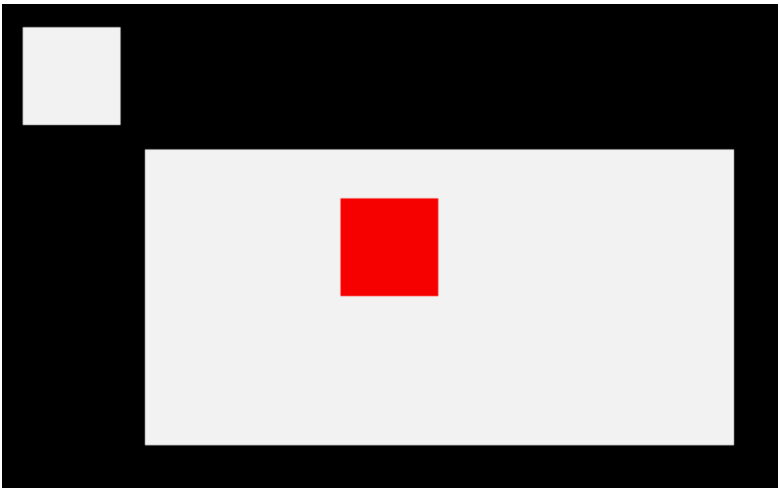
Het uitgesneden gebied kan op elke positie worden ingevoegd met het commando **PASTE**.

Opmerkingen:

De grootte van de rechthoek is beperkt tot de omvang van 1 KB van de buffer. Het geheugengebruik voor een uitgesneden gebied is breedte * hoogte * aantal bitplanes / 8. Dit mag niet 1024 bytes of meer bedragen. Voor een scherm met 4 bitplanes heeft een gebied van bijv. 45 x 45 1012.5 bytes nodig.

Voorbeeld: Gebruik van **CUT**

```
10 SCREEN 320, 200, 2
20 BOX 60, 60, 300, 180, 1 : REM TEKEN EEN WIT VAK
30 PEN 2                   : REM KIES RODE PEN
40 CUT 140, 80, 40, 40     : REM SNIJ EEN OPPERVLAK VAN 40 * 40 UIT
50 PASTE 10, 10            : REM PLAK HET OP NIEUWE POSITIE TERUG
60 GETKEY A$               : REM WACHT OP TOETSDRUK
70 SCREEN CLOSE
```



DATA

Token: $83

Formaat: **DATA** [constante [**,** constante ...]]

Gebruik: Definieert constanten die door **READ**-instructies in een programma kunnen worden gelezen.

Getallen en tekenreeksen zijn toegestaan, maar expressies niet. Items worden gescheiden door komma's. Tekenreeksen die komma's, dubbele punten of spaties bevatten moeten tussen aanhalingstekens worden geplaatst.

RUN initialiseert de gegevenspointer op het eerste item van de eerste **DATA**-instructie en schuift deze voor elk gelezen item op. Het is de verantwoordelijkheid van de programmeur dat het type van de constante en de variabele in de **READ**-instructie overeenkomen.

RESTORE kan worden gebruikt om de gegevenspointer op een specifieke regel te zetten voor volgende leesacties.

Opmerkingen:

Het is een goede programmeerpraktijk om grote hoeveelheden **DATA**-instructies aan het einde van het programma te plaatsen, zodat ze de zoekactie naar regelnummers na **GOTO** en andere instructies met regelnummerdoelen niet vertragen.

Lege items zonder constante tussen komma's zijn toegestaan en worden geretourneerd als nul voor numerieke constanten en als lege tekenreeks voor tekenreeksexpressies.

Voorbeeld: Gebruik van **DATA**

```
10 READ NA$, VE
20 READ N% : FOR I=2 TO N% : READ GL(I) : NEXT I
30 PRINT "PROGRAMMA: "; NA$; "  VERSIE:"; VE
40 PRINT "N-PUNTS GAUSS-LEGENDRE-FACTOREN E1:"
50 FOR I = 2 TO N% : PRINT I; GL(I) : NEXT I
60 END
80 DATA "MEGA65", 1.1
90 DATA 5, 0.5120, 0.3573, 0.2760, 0.2252

RUN
PROGRAMMA: MEGA65  VERSIE: 1.1
N-PUNTS GAUSS-LEGENDRE-FACTOREN E1:
 2  0.512
 3  0.3573
 4  0.276
 5  0.2252
```

DCLEAR

Token: $FE $15

Formaat: **DCLEAR** [**,D** station] [**,U** eenheid]

Gebruik: Verstuurt een initialiseercommando naar het opgegeven eenheid en het station.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Het DOS van het schijfstation sluit alle geopende bestanden, wist alle kanalen, maakt buffers vrij en herschrijft de BAM. Alle geopende kanalen op de computer worden ook gesloten.

Voorbeelden:

Gebruik van **DCLEAR**

```
DCLEAR

DCLEAR U9

DCLEAR D0, U9
```

DCLOSE

Token: $FE $0F

Formaat: **DCLOSE** [**U** eenheid]
DCLOSE# kanaal

Gebruik: Sluit een enkel bestand of alle bestanden voor het opgegeven station.

kanaal kanaalnummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

DCLOSE wordt gebruikt met ofwel een kanaalparameter ofwel een eenheidnummer, maar nooit met beide.

Opmerkingen:

Het is van belang alle geopende bestanden te sluiten voordat het programma eindigt. Anders worden buffers niet vrijgemaakt en kunnen open bestanden die zijn geschreven onvolledig zijn (ook wel "splat-bestanden" genoemd) en niet meer bruikbaar zijn.

Voorbeelden:

Gebruik van **DCLOSE**

```
DCLOSE#2  : REM SLUIT BESTAND GEASSOCIEED AAN KANAAL 2

DCLOSE U9 : REM SLUIT ALLEN OPEN BESTANDEN OP UNIT 9
```

DEC

Token: $D1

Formaat: **DEC**(tekenreeksexpressie)

Retourneert:

De decimale waarde van een hexadecimale tekenreeks.

Het bereik van het argument is "0" tot "FFFFFFFF".

Opmerkingen:

Toegestane cijfers in de hoofdletter-/graphicsmodus zijn 0 - 9 en A - F (`0123456789ABCDEF`) en in de kleine-letter-/hoofdlettermodus 0 - 9 en a - f (`0123456789abcdef`).

DEC negeert alles na en inclusief het eerste niet-toegestane cijfer of het 8e teken. Een tekenreeks die alleen niet-toegestane cijfers bevat geeft 0 terug.

Om een getal naar een hexadecimale tekenreeks om te zetten, zie **HEX$**.

Om een letterlijke waarde als hexadecimaal getal op te geven, gebruikt u **$** gevolgd door de cijfers: $A12F

Voorbeelden:

Gebruik van **DEC**

```
PRINT DEC("D000")
53248

POKE DEC("600"), 255
```

DECBIN

Token: $CE $11

Formaat: **DECBIN**(tekenreeksexpressie)

Retourneert:

De decimale waarde van een binaire tekenreeks.

Het bereik van het argument is “0” tot “11111111 ... 11111111” (32 enen).

Opmerkingen:

DECBIN negeert alles na het eerste niet-binaire cijfer of het 32e teken. Een tekenreeks die alleen niet-toegestane cijfers bevat geeft 0 terug.

Om een getal naar een binaire tekenreeks om te zetten, zie **STRBIN$**.

Om een letterlijke waarde als binair getal op te geven, gebruikt u **%** gevolgd door de cijfers: %10011101

Voorbeeld: Gebruik van **DECBIN**

```
PRINT DECBIN("1101000000000000")
53248
```

DEF FN

Token: $96 (DEF) $A5 (FN)

Formaat: **DEF FN** naam(reëel-getal-variabele) = [expressie]

Gebruik: Definieert een gebruikersfunctie van één instructie met een argument van het type reëel getal, welke een reëel-getal-waarde retourneert wanneer deze wordt geëvalueerd.

De definitie moet worden uitgevoerd voordat de functie in expressies kan worden gebruikt. Het argument is een dummy-variabele, welke wordt vervangen door het argument wanneer de functie wordt aangeroepen.

Opmerkingen:

Het functie-argument is geen reëel-getal-variabele en overschrijft geen variabele met die naam. Het vertegenwoordigt uitsluitend de argumentwaarde binnen de functiedefinitie.

Merk op dat **DEF FN** in dit geval eigenlijk bestaat uit twee aparte tokens: **DEF** en **FN**.

Voorbeeld: Gebruik van **DEF FN**

```
10 PD = π / 180
20 DEF FN CD(X) = COS(X * PD) : REM COS VOOR GRADEN
30 DEF FN SD(X) = SIN(X * PD) : REM SIN VOOR GRADEN
40 FOR D = 0 TO 360 STEP 90
50 PRINT USING "####"; D;
60 PRINT USING " ##.##"; FNCD(D);
70 PRINT USING " ##.##"; FNSD(D)
80 NEXT D

RUN
  0  1.00  0.00
 90  0.00  1.00
180 -1.00  0.00
270  0.00 -1.00
360  1.00  0.00
```

DELETE

Token: $F7

Formaat: **DELETE** [regelbereik]
DELETE bestandsnaam [**,D** station] [**,U** eenheid] [**,R**]

Gebruik: De eerste vorm verwijdert een bereik van regels uit het BASIC-programma. De tweede vorm verwijdert een of meer bestanden van een schijf.

regel bereik bestaat uit de eerste en laatste te verwijderen regel, of een enkel regelnummer. Indien het eerste nummer wordt weggelaten, wordt de eerste BASIC-regel aangenomen. Het tweede nummer in de bereikaanduiding standaard op de laatste BASIC-regel.

bestandsnaam is een tekenreeks tussen aanhalingstekens, bijvoorbeeld: "SAFE" of een tekenreeksexpressie tussen haakjes, bijvoorbeeld: (F$$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

De **R**-vlag wordt gebruikt om een eerder verwijderd bestand te herstellen. Dit werkt alleen indien er geen schrijfbewerkingen tussen verwijdering en herstel hebben plaatsgevonden, welke de inhoud van het bestand hadden kunnen wijzigen.

Opmerkingen:
DELETE is een synoniem van **SCRATCH** en **ERASE**.

Voorbeelden:
Gebruik van **DELETE**

```
DELETE 100       : REM VERWIJDER REGEL 100

DELETE 240-350   : REM VERWIJDER REGELS VAN 240 TOT 350

DELETE 500-      : REM VERWIJDER VAN 500 TOT EIND

DELETE -70       : REM VERWIJDER VAN START TOT 70

DELETE "DRM", U9 : REM VERWIJDER BESTAND DRM OP UNIT 9

DELETE "*=SEQ"   : REM VERWIJDER ALLE SEQUENTIELE BESTANDEN

DELETE "R*=PRG"  : REM VERWIJDER PROGRAMMABESTANDEN BEGINNEND MET 'R'
```

DIM

Token: $86

Formaat: **DIM** naam(limieten) [**,** naam(limieten) ...]

Gebruik: Geeft de vorm, grenzen en het type van een BASIC-array aan.

Als declaratie-instructie moet deze slechts eenmaal worden uitgevoerd en vóór elk gebruik van de gedeclareerde arrays. Een array kan één of meer dimensies hebben. Eendimensionale arrays worden vaak vectoren genoemd, terwijl twee of meer dimensies een matrix definiëren. De onderste grens van een dimensie is altijd nul, terwijl de bovengrens bepaald is zoals gedefinieerd. De regels voor variabelenamen gelden ook voor arraynamen. U kunt byte-arrays, integer-arrays, reëel-getal-arrays en string-arrays aanmaken. Het is toegestaan om dezelfde identifier te gebruiken voor scalarvariabelen en arrayvariabelen. Het linker haakje na de naam onderscheidt arraynamen.

Opmerkingen:

Byte-arrays verbruiken één byte per element, integer-arrays twee bytes, reëel-getal-arrays vijf bytes en string-arrays drie bytes voor de stringdescriptor plus de lengte van de string zelf.

Indien een array-identifier wordt gebruikt zonder voorafgaande declaratie, wordt impliciet een eendimensionale array met een limiet van 10 gedeclareerd.

Voorbeeld: Gebruik van **DIM**

```
10 DIM A%(8)     : REM ARRAY VAN 9 ELEMENTEN
20 DIM XX(2, 3)  : REM ARRAY VAN 3X4 = 12 ELEMENTEN
30 FOR I = 0 TO 8 : A%(I) = PEEK(256 + I) : PRINT A%(I); : NEXT : PRINT
40 FOR I = 0 TO 2 : FOR J = 0 TO 3 : READ XX(I, J) : PRINT XX(I, J); : NEXT J, I
50 END
60 DATA 1, -2, 3, -4, 5, -6, 7, -8, 9, -10, 11, -12

RUN
 45  52  50  0  0  0  0  0  0
 1 -2  3 -4  5 -6  7 -8  9 -10  11 -12
```

DIR

Token: $EE (DIR) $FE $29 (ECTORY)

Formaat: **DIR** [bestandspatroon] [**,W**] [**,P**] [**,R**] [**,D** station] [**,U** eenheid]
DIRECTORY [bestandspatroon] [**,W**] [**,P**] [**,R**] [**,D** station] [**,U** eenheid]
$ [bestandspatroon] [**,W**] [**,R**] [**,D** station] [**,U** eenheid]
DIR [bestandspatroon] **,U12** [**,P**]

Gebruik: Geeft de bestandsmap/catalogus van de opgegeven schijf weer.

De parameter **W** (Breed) geeft de directory in drie kolommen op het scherm weer en stopt nadat het scherm is gevuld met een pagina (63 directory-items). Door op een willekeurige toets te drukken, wordt de volgende pagina weergegeven.

De parameter **P** (Paginering) geeft de map in één kolom weer en onderbreekt na elke volle schermpagina. Druk op de Q-toets om de lijstopgave op de huidige pagina te onderbreken. Druk op een andere toets om de volgende pagina weer te geven.

De parameter **R** (Herstelbaar) toont alleen de bestanden die als verwijderd zijn gemarkeerd maar nog steeds herstelbaar zijn, elk gemarkeerd met een '-'. Andere bestanden worden niet getoond.

bestandspatroon is een tekenreeks tussen aanhalingstekens ("D*") of een tekenreeksexpressie tussen haakjes: (D$)

Het argument **U12** geeft de inhoud van de SD-kaart weer. Dit kan worden gecombineerd met het argument **P** voor een gepagineerde weergave. Het ondersteunt een bestandspatroon. Andere argumenten worden niet ondersteund.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

DIR is een synoniem van **CATALOG** en **DIRECTORY**, en produceert dezelfde lijstopgave.

De **bestandspatroon** kan worden gebruikt om de lijstopgave te filteren. Binnen het patroon:

- * komt overeen met een willekeurig aantal tekens van elk type.
- # komt overeen met één cijfer.
- $ komt overeen met één letter.

- ? komt overeen met één teken van elk type.

Door **,T=** toe te voegen aan de patroontekenreeks, waarbij **T** een bestandstype opgeeft van **P**, **S**, **U**, **R** of **C** (voor **P**RG, **S**EQ, **U**SR, **R**EL, **C**BM), wordt de uitvoer gefilterd op dat bestandstype.

Het afkortingssymbool **$** kan alleen in de directe modus worden gebruikt.

Voorbeeld: Gebruik van **DIR**

```
DIR

DIR "MEG*",U12
DIR "ME????.D81",U12
DIR "ME$$65.D81",U12
DIR "MEGA##.D81",U12
DIR "M*.D81",U12
0 "BLACK SMURF     " BS 2A
508  "STORY PHOBOS"    SEQ
27   "C8096"           PRG
25   "C128"            PRG
104 BLOCKS FREE.
```

DISK

Token: $FE $40

Formaat: **DISK** commando [**,U** eenheid]
@ commando [**,U** eenheid]

Gebruik: Verstuurt een commandotekenreeks naar het opgegeven schijfstation.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

commando een tekenreeksexpressie.

Opmerkingen:

De commandotekenreeks wordt door het schijfstation geïnterpreteerd en moet compatibel zijn met de gebruikte DOS-versie. Raadpleeg de handleiding van het schijfstation voor mogelijke commando's.

Het gebruik van **DISK** zonder commandotekenreeks geeft de schijfstatus weer.

De afkorting **@** kan alleen in de directe modus worden gebruikt.

Voorbeelden:

Gebruik van **DISK**

```
DISK "I0"   : REM INITIALISEER DISK IN DRIVE 0

DISK "U0>9" : REM ZET DEVICENUMMER OP 9
```

DLOAD

Token: $F0

Formaat: **DLOAD** bestandsnaam [**,D** station] [**,U** eenheid]
DLOAD "$[patroon=type]" [**,D** station] [**,U** eenheid]
DLOAD "$$[patroon=type]" [**,D** station] [**,U** eenheid]

Gebruik: De eerste vorm laadt een bestand van het type PRG in het geheugen dat gereserveerd is voor BASIC-programma's.

De tweede vorm laadt een map in het geheugen, welke vervolgens met **LIST** kan worden bekeken. De structuur lijkt op een BASIC-programma, doch in plaats van regelnummers worden bestandsgroottes weergegeven.

De derde vorm lijkt op de tweede, maar de bestanden krijgen opeenvolgende regelnummers. Deze lijstopgave kan als een BASIC-programma worden gescrold met de toetsen **F9** of **F11**, bewerkt, weergegeven, opgeslagen of afgedrukt.

Een filter kan worden toegepast door een patroon of een patroon met een type op te geven. Het asterisk * komt overeen met de rest van de naam, terwijl het ? met elk willekeurig teken overeenkomt. De typespecificatie kan een van de tekens (P, S, U, R, C) zijn, respectievelijk programma-, sequentieel, gebruiker-, relatief- of CBM-bestand.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:
Het laadadres dat in de eerste twee bytes van het PRG-bestand is opgeslagen, wordt genegeerd. Het programma wordt altijd in BASIC-geheugen geladen. Hierdoor kunnen BASIC-programma's worden geladen die op andere computers met andere geheugenconfiguraties zijn opgeslagen. Na het laden wordt het programma opnieuw gekoppeld en is het klaar om te worden uitgevoerd (**RUN**) of bewerkt.

Het is mogelijk om **DLOAD** in een lopend programma te gebruiken. Dit wordt "overlayen" of "ketenen" genoemd. Indien u dit doet, vervangt het nieuw geladen programma het huidige en start de uitvoering automatisch op de eerste regel van het nieuwe programma. Variabelen, arrays

en strings van de huidige uitvoering worden behouden en kunnen ook door het nieuw geladen programma worden gebruikt.

Elke **DLOAD**, van een programma of een directorylijst, vervangt een programma dat zich momenteel in het geheugen bevindt.

Voorbeelden:

Gebruik van **DLOAD**

```
DLOAD "APOCALYPSE"

DLOAD "MEGA TOOLS", U9

DLOAD (FI$), U(UN%)

DLOAD "$"      : REM LAAD HELE DIRECTORY - MET BESTANDSGROOTTES

DLOAD "$$"     : REM LAAD HELE DIRECTORY - SCROLLBAAR

DLOAD "$$X*=P" : REM DIRECTORY MET PRG-BESTANDEN BEGINNEND MET 'X'
```

DMA

Token: $FE $1F

Formaat: **DMA** commando [**,** lengte**,** bronadres**,** bronbank**,** doeladres**,** doelbank[**,** sub]]

Gebruik: **DMA** ("Direct Memory Access") is verouderd en is vervangen door **EDMA**.

Het **commando** bestuurt de functie met behulp van de twee onderste bits: 0: kopiëren, 1: mengen, 2: verwisselen, 3: vullen.

lengte aantal bytes (in het bereik 0 tot 65535). OPMERKING: Het opgeven van lengte 0 wordt geïnterpreteerd als een lengte van 65536 (exact 64KB).

bronadres (16-bits) van het leesgebied of het vulbyte.

bronbank nummer van de bron (genegeerd in de vulmodus).

doeladres van het schrijfgebied (16-bits).

doelbank nummer van het doel.

sub het te gebruiken subcommando.

Opmerkingen:

DMA heeft toegang tot het onderste 1MB-adresbereik, georganiseerd in 16 banken van 64KB. Om deze beperking te omzeilen, gebruikt u **EDMA**, welke toegang heeft tot het volledige 256MB-adresbereik.

OPMERKING: Alleen kopiëren en vullen zijn geïmplementeerd in de MEGA65-DMAcontroller ten tijde van het schrijven. Andere DMAgic-commandobits kunnen ook worden gezet, bijvoorbeeld om het kopiëren van gegevens in de omgekeerde richting toe te laten of het bron- of doeladres vast te houden.

Voorbeelden:

Gebruik van **DMA**

```
DMA 0, 80*25, 2048, 0, 0, 4  : REM BEWAAR SCHERM OP $0000 IN BANK 4

DMA 3, 80*25, 32, 0, 2048, 0 : REM MAAK SCHERM LEEG

DMA 0, 80*25, 0, 4, 2048, 0  : REM HERSTEL SCHERM VANAF $0000 IN BANK 4
```

DMODE

Token: $FE $35

Formaat: **DMODE** jam**,** complement**,** sjabloon**,** stijl**,** dikte

Gebruik: Bitmapgraphics: stelt de parameters van de "weergavemodus" van de grafische context in, welke door tekencommando's worden gebruikt.

Indien **jam** 1 is, worden kleurwaarden gecombineerd wanneer een pixel op dezelfde locatie als een andere pixel wordt getekend. De standaardwaarde is 0: de pixel overschrijven.

Indien **complement** 1 is, wordt een nieuwe kleur geproduceerd gebaseerd op de XOR van de bitplaten wanneer een pixel op dezelfde locatie als een andere pixel wordt getekend. De standaardwaarde is 0: de pixel overschrijven.

Indien **stijl** 1 is, worden bij het tekenen het sjabloon van **DPAT** gebruikt voor lijnen en vulling. De standaardwaarde is 0: ononderbroken lijnen en vulling.

Opmerkingen:

De parameters **sjabloon** en **dikte** zijn niet geïmplementeerd en hebben geen effect. Deze argumenten zijn behouden uit het ontwerpdocument van de C65 voor mogelijke toekomstige implementatie. Eveneens suggereert de C65-documentatie een derde **stijl**-waarde welke eveneens niet is geïmplementeerd.

De C65-documentatie suggereert dat **jam**-modus een aparte penkleur gebruikt bij het tekenen over de achtergrond, ingesteld met **PEN 1,C**. Dit lijkt niet te zijn geïmplementeerd.

Zie ook **DPAT**.

Voorbeeld: Gebruik van **DMODE**

```
 10 SCREEN 320, 200, 5
 20 PEN 2
 30 BOX 50, 50, 100, 100, 1
 40 PEN 6
 50 DMODE 1, 0, 0, 0, 0
 60 BOX 40, 80, 170, 84, 1
 70 DMODE 0, 1, 0, 0, 0
 80 BOX 40, 90, 170, 94, 1
 90 GETKEY A$
100 SCREEN CLOSE
```

DO

Token: $EB

Formaat: **DO** ... **LOOP**
DO [<**UNTIL** | **WHILE**> logische expressie] ... opdrachten [**EXIT**]
LOOP [<**UNTIL** | **WHILE**> logische expressie]

Gebruik: **DO** definieert het begin van een BASIC-lus.

Het gebruik van **DO** ... **LOOP** zonder modificatoren creëert een oneindige lus, welke alleen met de instructie **EXIT** kan worden afgesloten. De lus kan worden beheerd door **UNTIL** of **WHILE** toe te voegen na het **DO** of het **LOOP**.

Opmerkingen:

DO-lussen mogen genest worden. Een instructie **EXIT** sluit alleen de huidige lus af.

Voorbeelden:

Gebruik van **DO**

```
10 PW$ = "" : DO
20 GET A$ : PW$ = PW$ + A$
30 LOOP UNTIL LEN(PW$) > 7 OR A$ = CHR$(13)

10 DO                          : REM WACHT OP GEBRUIKERSBESLISSING
20 GET A$
30 LOOP UNTIL A$ = "Y" OR A$ = "N"

10 DO WHILE ABS(EPS) > 0.001
20 GOSUB 2000                  : REM ITERATIESUBROUTINE
30 LOOP

10 I% = 0                      : REM INTEGERLUS 1-100
20 DO : I% = I %+ 1
30 LOOP WHILE I% < 101
```

DOPEN

Token: $FE $0D

Formaat: **DOPEN#** kanaal**,** bestandsnaam>**,L** [rec-lengte]] [**,W**] [**,D** station] [**,U** eenheid]

Gebruik: Opent een bestand voor lezen of schrijven.

kanaal getal, waarbij:

- 1 <= kanaal <= 127: regelafsluiting is CR.
- 128 <= kanaal <= 255: regelafsluiting is CR LF.

L geeft aan dat het bestand een relatief bestand is, welke wordt geopend voor lezen/schrijven en willekeurige toegang.

rec-lengte is de recordlengte en is verplicht voor het aanmaken van relatieve bestanden. Bij bestaande relatieve bestanden wordt **rec-lengte** als veiligheidscontrole gebruikt indien opgegeven.

W opent een bestand voor schrijftoegang. Het bestand mag niet bestaan.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

DOPEN# kan worden gebruikt om alle bestandstypen te openen. Het sequentiële bestandstype **SEQ** is standaard. Het relatieve bestandstype **REL** wordt gekozen met de parameter **L**. Andere bestandstypen moeten in de bestandsnaam worden opgegeven, bijv. door ",P" toe te voegen aan de bestandsnaam voor **PRG**-bestanden of ",U" voor **USR**-bestanden.

Indien het eerste teken van de bestandsnaam een "@" (apestaartje) is, wordt dit geïnterpreteerd als een "opslaan en vervangen"-operatie. Het gebruik van deze optie op 1541- en 1571-stations wordt niet aanbevolen, omdat deze een "opslaan en vervangen-fout" in hun DOS bevatten.

Voorbeelden:

Gebruik van **DOPEN**

```
DOPEN#5, "DATA", U9

DOPEN#130, (DD$), U(UN%)

DOPEN#3, "USER FILE,U"

DOPEN#2, "DATA BASE", L240

DOPEN#4, "MYPROG,P"
```

DOT

Token: $FE $4C

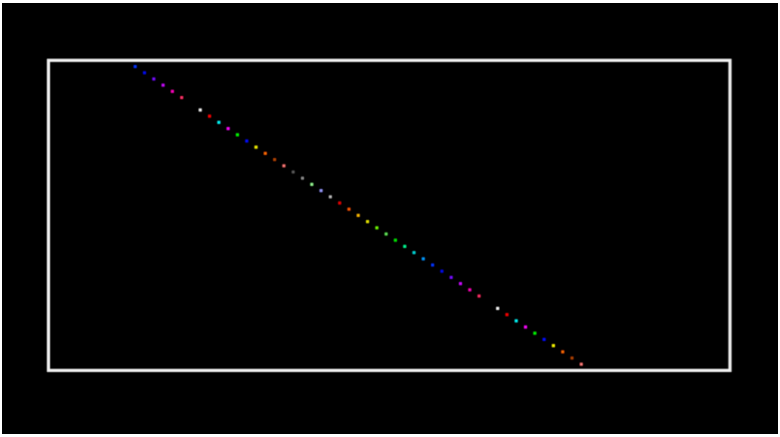
Formaat: **DOT** x, y [,kleur]

Gebruik: Bitmapgraphics: tekent een pixel op schermcoördinaten (x, y).

De optionele derde parameter bepaalt de te gebruiken kleur. Indien niet opgegeven, wordt de huidige penkleur gebruikt.

Voorbeeld: Gebruik van **DOT**

```
10 SCREEN 320, 200, 5
20 BOX 50, 50, 270, 150
30 VIEWPORT 50, 50, 220, 100
40 FOR I = 0 TO 127
50 DOT I + I + I, I + I, I
60 NEXT
70 GETKEY A
80 SCREEN CLOSE
```



DPAT

Token: $FE $36

Formaat: **DPAT** type [**,** getal**,** patroon ...]

Gebruik: Bitmapgraphics: stelt het tekenpatroon van de grafische context in voor tekencommando's.

Er zijn vier voorgedefinieerde patronen die geselecteerd kunnen worden door het typenummer (1, 2, 3 of 4) als enkele parameter op te geven.

- 1: Dunne stip.
- 2: dikke stip.
- 3: onderbroken.
- 4: stip-streep.

Een typenummer van nul geeft een door de gebruiker gedefinieerd patroon aan. Dit patroon kan ingesteld worden met een bitreeks van 8, 16, 24 of 32 bits. Het aantal gebruikte patroonbytes wordt als tweede parameter opgegeven. Dit bepaalt hoeveel patroonbytes (1, 2, 3 of 4) volgen.

- Type: 0 - 4.
- Aantal: aantal volgende patroonbytes (1 - 4).
- Patroon: patroonbytes.

Opmerkingen:

Om het tekenpatroon te gebruiken, stelt u met **DMODE** de **stijl**-modus in op 1.

Het effect van een tekenpatroon is afhankelijk van het commando dat tekent. **LINE** en de omtrekken van **BOX** en **POLYGON** tekenen het patroon in de richting van de lijn. Een gevulde rechthoek getekend met **BOX** geeft het patroon horizontaal weer in de vulling. Andere gevulde figuren, zoals **CIRCLE**, **ELLIPSE** en niet-rechthoekige **BOX**, kunnen het patroon op onvoorspelbare wijze toepassen op basis van hun vullingsalgoritme.

Zie ook **DMODE**.

Voorbeeld: Gebruik van **DPAT**

```
10 SCREEN 320, 200, 5
20 DMODE 0, 0, 0, 1, 0
30 PEN 2
40 DPAT 1
50 BOX 100, 100, 149, 149, 1
60 PEN 3
70 DPAT 0, 4, %00110011, %00111100, %00001111, %11110000
80 BOX 150, 150, 199, 199, 1
90 GETKEY A$
100 SCREEN CLOSE
```

DS

Token: N.v.t.

Formaat: **DS**

Gebruik: De status van de laatste schijfbewerking.

Dit is een vluchtige variabele. Elk gebruik activeert het uitlezen van de schijfstatus van het momenteel actieve schijfstation.

DS is gekoppeld aan de tekenreeksvariabele **DS$**, welke tegelijkertijd wordt bijgewerkt.

Het uitlezen van de schijfstatus van een schijfstation wist automatisch elke foutstatus op dat station. De variabelen **DS** en **DS$** behouden hun waarden tot de volgende schijfbewerking.

Opmerkingen:

DS is een gereserveerde systeemvariabele.

Voorbeeld: Gebruik van **DS**

```
100 DOPEN#1, "DATA"
110 IF DS <> 0 THEN PRINT "COULD NOT OPEN FILE DATA" : STOP
```

DS$

Token: N.v.t.

Formaat: **DS$**

Gebruik: De status van de laatste schijfbewerking in de tekstvorm "<code>,<bericht>,<spoor>,<sector>".

DS$ is gekoppeld aan de numerieke variabele **DS**. DS$ wordt ingesteld op `00,OK,00,00` indien er geen fout is opgetreden; anders wordt deze ingesteld op een DOS-foutmelding (vermeld in de handleidingen van de schijfstations).

Opmerkingen:

DS$ is een gereserveerde systeemvariabele.

De spoorwaarde wordt ook gebruikt voor het rapporteren van hoeveelheden, zoals het aantal verwijderde bestanden na een **DELETE**.

Voorbeeld: Gebruik van **DS$**

```
100 DOPEN#1, "DATA"
110 IF DS <> 0 THEN PRINT DS$ : STOP
```

DSAVE

Token: $EF

Formaat: **DSAVE** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Slaat het BASIC-programma in het geheugen op als een bestand van het type **PRG**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

De maximale lengte van de bestandsnaam is 16 tekens.

DSAVE geeft een fout indien er al een bestand met de opgegeven naam bestaat. Om een bestaand bestand te vervangen, plaatst u een apestaartje (@) voor de bestandsnaam, tussen de aanhalingstekens.

OPMERKING: Opslaan met vervanging wordt niet aanbevolen op 1541- en 1571-stations, vanwege een bug in oudere versies van het besturingssysteem van het schijfstation welke gegevensverlies kan veroorzaken.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

DVERIFY kan na **DSAVE** gebruikt worden om te controleren of het opgeslagen programma op de schijf identiek is aan het programma in het geheugen.

Voorbeelden:

Gebruik van **DSAVE**

```
DSAVE "AVONTUUR"

DSAVE "CMYPROG"

DSAVE "ZORK-I", U9

DSAVE "KERKERGEWELF", D1, U10
```

DT$

Token: N.v.t.

Formaat: **DT$**

Gebruik: De huidige datum, als tekenreeks.

De datumwaarde wordt bijgewerkt vanuit de RTC (Real-Time Clock). De tekenreeks **DT$** heeft het formaat `DD-MMM-JJJJ`, bijvoorbeeld `04-APR-2021`.

Opmerkingen:

DT$ is een gereserveerde systeemvariabele. Voor meer informatie over het instellen van de Real-Time Clock, raadpleeg Hoofdstuk/Bijlage 5 op bladzijde 5-3.

Voorbeeld: Gebruik van **DT$**

```
100 PRINT "VANDAAG IS HET: "; DT$
```

DVERIFY

Token: $FE $14

Formaat: **DVERIFY** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Controleert of het BASIC-programma in het geheugen gelijk is aan een bestand van het type **PRG**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

DVERIFY kan alleen op gelijkheid testen. Het geeft geen informatie over het aantal of de positie van bytes met verschillende waarden. **DVERIFY** sluit af met het bericht OK of met een Verify-fout. Indien een Verify-fout optreedt, is het weergegeven adres altijd $2001.

Voorbeelden:

Gebruik van **DVERIFY**

```
DVERIFY "AVONTUUR"

DVERIFY "ZORK-I", U9

DVERIFY "KERKERGEWELF", D1, U10
```

EDIT

Token: $FE $45 (EDIT) $91 (ON) $FE $24 (OFF)

Formaat: **EDIT** <**ON** | **OFF**>

Gebruik: Schakelt de tekstbewerkingsmodus van de schermeditor in of uit.

EDIT ON schakelt de tekstbewerkingsmodus in. In deze modus kunt u bestanden van het type SEQ als tekstbestanden aanmaken, bewerken, opslaan en laden met dezelfde regeleditor die u gebruikt om BASIC-programma's te schrijven. In deze modus geldt het volgende:

- De prompt wordt weergegeven als OK in plaats van READY.
- De editor voert geen tokenisering/parsing uit. Alle tekst die na een regelnummer wordt ingevoerd, blijft zuivere tekst; BASIC-trefwoorden zoals **FOR** en **GOTO** worden niet omgezet in BASIC-tokens, zoals in de programmode het geval is.
- De regelnummers worden uitsluitend gebruikt voor tekstorganisatie, sortering, verwijdering, weergave etc.
- Wanneer de tekst met **DSAVE** naar een bestand wordt opgeslagen, wordt een sequentieel bestand (type **SEQ**) geschreven, geen programma- (**PRG**-) bestand. Regelnummers worden *niet* naar het bestand geschreven.
- **DLOAD** in de tekstmodus kan uitsluitend sequentiële bestanden laden. Regelnummers worden automatisch gegenereerd voor bewerkingsdoeleinden.
- De tekstmodus is alleen van toepassing op regels met een regelnummer. Regels zonder regelnummer worden, zoals gebruikelijk, als BASIC-commando's uitgevoerd.

EDIT OFF schakelt de tekstbewerkingsmodus uit en keert terug naar de BASIC-programmabewerkingsmodus. De MEGA65 start in de BASIC-programmabewerkingsmodus.

Sequentiële bestanden die met de tekseditor zijn aangemaakt, kunnen op het scherm weergegeven worden (zonder ze te laden) met behulp van **TYPE**.

Voorbeelden:

Gebruik van **EDIT**

```
ready.
edit on

ok.
100 Dit is een eenvoudige teksteditor.
dsave "voorbeeld"

ok.
new

ok.
catalog

0 "demoleeg        " 00 3d
1 "voorbeeld         seq
3159 blocks free

ok.
type "voorbeeld"
Dit is een eenvoudige teksteditor.

ok.
dload "voorbeeld"

loading

ok.
list

100 Dit is een eenvoudige teksteditor.

ok.
edit off

ready.
```

EDMA

Token: $FE $21

Formaat: **EDMA** commando**,** lengte**,** bron**,** doel

Gebruik: Kopieert of werkt grote hoeveelheden geheugen snel.

EDMA ("Extended Direct Memory Access") is de snelste methode om geheugengebieden te manipuleren met behulp van de DMA-controller. Raadpleeg Hoofdstuk/Bijlage P op bladzijde P-3 voor meer informatie over EDMA.

commando 0: kopiëren, of 3: vullen.

Omdat de commandobits hetzelfde register delen met andere bits, kunt u bijvoorbeeld bit 5 gebruiken om de lusomkering te activeren. Dit werkt ook bij overlappende geheugenregio's voor bron en doel. Zie het onderstaande voorbeeld.

lengte aantal bytes (in het bereik 0 tot 65535). OPMERKING: Het opgeven van lengte 0 wordt geïnterpreteerd als een lengte van 65536 (exact 64KB).

bron 28-bits adres van het leesgebied of het vulbyte.

doel 28-bits adres van het schrijfgebied.

Opmerkingen:

EDMA heeft toegang tot het volledige 256MB-adresbereik, met maximaal 28 bits voor de adressen van bron en doel.

Voorbeelden:

Gebruik van **EDMA**

```
EDMA 0, $800, $F700, $8000000 : REM KOPIEER SCALAIRE VARIABELEN NAAR ZOLDER-RAM

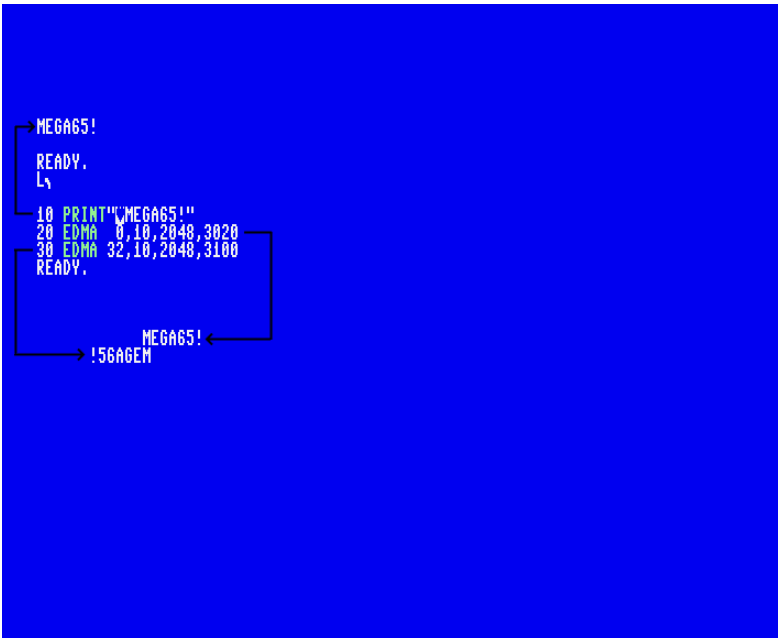
EDMA 3, 80*25, 32, 2048       : REM MAAK HET SCHERM LEEG

EDMA 0, 80*25, 2048, $8000800 : REM KOPIEER SCHERM NAAR ZOLDER-RAM
```

Door 32 (bit 5) aan de commandoparameter toe te voegen, kan de DMA-bewerking in omgekeerde volgorde worden uitgevoerd:

```
10 PRINT "■MEGA65!"
20 EDMA  0, 10, 2048, 3020 : REM 2048 IS BEGIN SCHERMGEHEUGEN
30 EDMA 32, 10, 2048, 3100 : REM 3020 EN 3100 ZIJN ONDERSTE DEEL SCHERM
```

Lijst en uitvoer van het omgekeerde voorbeeld:



EL

Token: N.v.t.

Formaat: **EL**

Gebruik: Het regelnummer waar de meest recente BASIC-fout is opgetreden, of de waarde -1 indien er geen fout is opgetreden.

Opmerkingen:

EL is een gereserveerde systeemvariabele.

Deze variabele wordt doorgaans gebruikt in een **TRAP**-routine, waarbij het foutregelnummer uit **EL** wordt gelezen.

Voorbeeld: Gebruik van **EL**

```
 10 TRAP 100
 20 PRINT SQR(-1)                        : REM OPZETTELIJKE FOUT
 30 PRINT "BIJ REGEL 30"                 : REM PROGRAMMALOOP GAAT HIER VERDER
 40 END
100 IF ER > 0 THEN PRINT ERR$(ER); " ERROR"
110 PRINT " OP REGEL"; EL
120 RESUME NEXT                          : REM GA VERDER NA FOUT
```

ELLIPSE

Token: $FE $30

Formaat: **ELLIPSE** xc**,** yc**,** xr**,** yr[**,** vlaggen **,** begin**,** einde]

Gebruik: Bitmapgraphics: tekent een ellips.

xc de x-coördinaat van het midden in pixels.

yc de y-coördinaat van het midden in pixels.

xr de x-straal van de ellips in pixels.

yr de y-straal van de ellips in pixels.

vlaggen bepaalt de vulling, bogen en de oriëntatie van de nulradiaal (combs-vlag, genoemd naar **retroCombs**). Standaardinstelling (nul) is: niet vullen, poten tekenen, starten met tekenen om 3 uur.

Bit	Naam	Waarde	Actie bij ingesteld
0	vulling	1	Vul ellips of boog met de huidige penkleur
1	poten	2	Onderdruk het tekenen v/d poten van een boog
2	combs	4	Laat de nulradiaal naar 12 uur wijzen

De eenheid van de begin- en eindhoek is graden in het bereik van 0 tot 360. De nulradiaal begint om 3 uur en loopt met de klok mee. De combs-vlag verschuift de nulradiaal en de startpositie naar de 12-uurpositie.

begin beginhoek om een elliptische boog te tekenen.

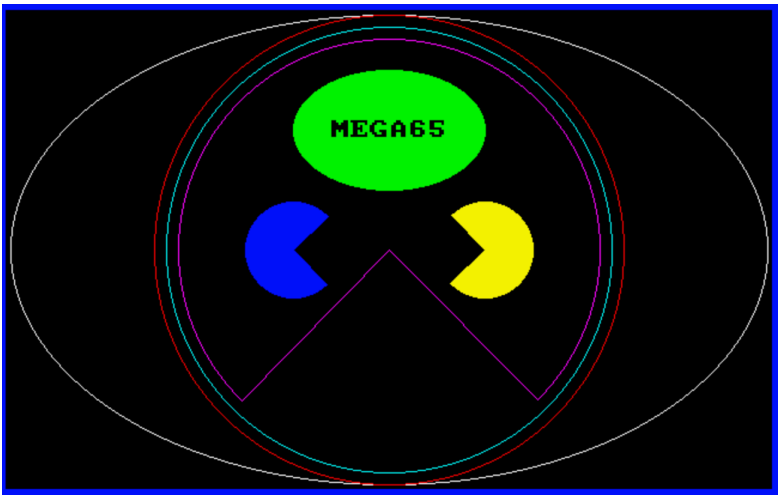
einde eindhoek om een elliptische boog te tekenen.

Opmerkingen:

ELLIPSE wordt gebruikt om ellipsen op schermen met verschillende resoluties te tekenen. Indien een volledige ellips getekend moet worden, moeten begin en einde of weggelaten worden of beide op nul worden gezet (niet 0 en 360). Het tekenen en vullen van volledige ellipsen is aanzienlijk sneller dan het gebruik van elliptische bogen.

Voorbeeld: Gebruik van **ELLIPSE**

```
100 S% = 2 : D% = 3 : W% = 320 * S% : H% = 200 * S%        : REM SCHERMINSTELLINGEN
110 CX% = W% / 2 : CY% = H% / 2                            : REM MIDDELPUNT EN STRAAL
120 RX% = W% / 2 : RY% = H% / 2
130 SCREEN W%, H%, D%                                      : REM OPEN SCHERM
140 ELLIPSE CX%, CY%, CX% - 4, CY% - 4
150 PEN 2 : CIRCLE CX%, CY%, RY% - 4, 2
160 PEN 3 : CIRCLE CX%, CY%, RY% - 14, 2
170 PEN 4 : CIRCLE CX%, CY%, RY% - 24, 0, 135, 45
180 PEN 5 : ELLIPSE CX%, CY% / 2, RX% / 4, RY% / 4, 1
190 PEN 6 : CIRCLE 120 * S%, CY%, 40, 1, 45, 315
200 PEN 7 : CIRCLE 200 * S%, CY%, 40, 1, 225, 135
210 PEN 0 : CHAR 34, CY% / 2 - 8, 2, 2, 2, "MEGA65", $3D000
220 GETKEY A$                                              : REM WACHT OP TOETSDRUK
230 SCREEN CLOSE                                           : REM SLUIT GRAFISCH SCHERM
```



ELSE

Token: $D5

Formaat: **IF** expressie **THEN** waar-clausule [**ELSE** onwaar-clausule]

Gebruik: **ELSE** is een optioneel onderdeel van een **IF**-statement.

expressie logische of numerieke expressie. Een numerieke expressie wordt als **FALSE** beschouwd indien de waarde nul is en als **TRUE** voor elke niet-nul waarde.

waar-clausule een of meer instructies die direct na **THEN** op dezelfde regel beginnen. Een regelnummer na **THEN** voert in plaats daarvan een **GOTO** uit naar die regel.

onwaar-clausule een of meer instructies die direct na **ELSE** op dezelfde regel beginnen. Een regelnummer na **ELSE** voert in plaats daarvan een **GOTO** uit naar die regel.

Opmerkingen:

Voor **ELSE** moet een dubbele punt staan. Na **ELSE** mag geen dubbele punt of einde-van-regel staan.

De standaard **IF** ... **THEN** ... **ELSE**-structuur is beperkt tot één regel, maar de **waar-clausule** en **onwaar-clausule** kunnen worden uitgesmeerd over meerdere regels met behulp van een samengestelde instructie tussen **BEGIN** ... **BEND**.

Wanneer de **waar-clausule** geen **BEGIN** ... **BEND** gebruikt, moet **ELSE** op dezelfde regel staan als **IF**.

Voorbeelden:

Gebruik van **ELSE**

```
100 REM ELSE
110 ROOD$ = CHR$(28) : ZWART$ = CHR$(144) : WIT$ = CHR$(5)
120 INPUT "GEEF EEN GETAL"; V
130 IF V < 0 THEN PRINT ROOD$; : ELSE PRINT ZWART$;
140 PRINT V                                    : REM PRINT NEGATIEVE GETALLEN IN ROOD
150 PRINT WIT$
160 INPUT "PROGRAMMA STOPPEN (J/N)"; A$
170 IF A$ = "J" THEN END
180 IF A$ = "N" THEN 120 : ELSE 160
```

Gebruik van **ELSE** met **BEGIN** ... **BEND**

```
100 A=0 : GOSUB 200
110 A=1 : GOSUB 200
120 END
200 IF A=0 THEN BEGIN
210 PRINT "HALLO"
220 BEND : ELSE BEGIN
230 PRINT "TOT ZIENS"
240 BEND
250 RETURN
```

END

Token: $80

Formaat: **END**

Gebruik: Beëindigt de uitvoering van het BASIC-programma.

De READY-prompt verschijnt en de computer schakelt over naar de directe modus om op toetsenbordinvoer te wachten.

Opmerkingen:

END wist *geen* kanalen en sluit *geen* bestanden.

Variabele-definities blijven na **END** geldig.

Het programma kan worden voortgezet met het **CONT**-statement. Na het uitvoeren van de laatste regel van een programma wordt **END** automatisch uitgevoerd.

Voorbeeld: Gebruik van **END**

```
10 IF V < 0 THEN END : REM NEGATIEF GETAL STOPT PROGRAMMA
20 PRINT V
```

ENVELOPE

Token: $FE $0A

Formaat: **ENVELOPE** n [{, aanzet, terugval, aanhouden, wegsterven, golfvorm, pw}]

Gebruik: Stelt de parameters in voor de synthese van een muziekinstrument voor gebruik met **PLAY**.

n envelopeslot (0 - 9).

aanzet aanzetsnelheid (0 - 15).

terugval terugvalsnelheid (0 - 15).

aanhouden aanhoudniveau (0 - 15).

wegsterven wegsterfsnelheid (0 - 15).

golfvorm 0: driehoek, 1: zaagtand, 2: vierkant/puls, 3: ruis of 4: ringmodulatie.

pw pulsbreedte (0 - 4095) voor de golfvorm.

Er zijn 10 slots voor het opslaan van instrumentparameters, vooraf ingesteld met de volgende standaardwaarden:

n	A	D	S	R	WF	PW	Instrument
0	0	9	0	0	2	1536	piano
1	12	0	12	0	1		accordeon
2	0	0	15	0	0		calliope
3	0	5	5	0	3		drum
4	9	4	4	0	0		fluit
5	0	9	2	1	1		gitaar
6	0	9	0	0	2	512	klavecimbel
7	0	9	9	0	2	2048	orgel
8	8	9	4	1	2	512	trompet
9	0	9	0	0	0		xylofoon

Voorbeeld: Gebruik van **ENVELOPE**

```
10 ENVELOPE 9, 10, 5, 10, 5, 2, 4000
20 VOL 9, 9
30 TEMPO 30
40 PLAY "T9O4Q CDEFGAB U3T8 CDEFGAB L", "T5O3Q H CGEQG T7 HCGEQG L"
```

ER

Token: N.v.t.

Formaat: **ER**

Gebruik: Het nummer van de meest recente BASIC-fout die heeft plaatsgevonden, of -1 wanneer er geen fout was opgetreden.

Opmerkingen:

ER is een gereserveerde systeemvariabele.

Deze variabele wordt doorgaans in een **TRAP**-routine gebruikt, waarbij het foutnummer aan **ER** wordt ontleend.

De mogelijke waarden van **ER** zijn als volgt:

ER	Betekenis	Vertaling
0	OK	Oké
1	TOO MANY FILES	Teveel bestanden
2	FILE OPEN	Bestand open
3	FILE NOT OPEN	Bestand niet open
4	FILE NOT FOUND	Bestand niet gevonden
5	DEVICE NOT PRESENT	Apparaat niet aanwezig
6	NOT INPUT FILE	Geen invoerbestand
7	NOT OUTPUT FILE	Geen uitvoerbestand
8	MISSING FILE NAME	Ontbrekende bestandsnaam
9	ILLEGAL DEVICE NUMBER	Ongeldig apparaatnummer
10	NEXT WITHOUT FOR	NEXT zonder FOR
11	SYNTAX	Syntaxis(fout)
12	RETURN WITHOUT GOSUB	RETURN zonder GOSUB
13	OUT OF DATA	data is op
14	ILLEGAL QUANTITY	ongeldige hoeveelheid
15	OVERFLOW	overloop(fout)
16	OUT OF MEMORY	geheugen is op
17	UNDEFINED STATEMENT	onbekende opdracht
18	BAD SUBSCRIPT	ongeldige index
19	REDIMENSIONED ARRAY	array al gedimensioneerd
20	DIVISION BY ZERO	deling door nul
21	ILLEGAL DIRECT	ongeldig direct
22	TYPE MISMATCH	strijdig type
23	STRING TOO LONG	tekenreeks te lang
24	FILE DATA	bestandsdata
25	FORMULA TOO COMPLEX	formule te complex
26	CAN'T CONTINUE	kan niet doorgaan
27	UNDEFINED FUNCTION	ongedefinieerde functie
28	VERIFY	verificatie(fout)
29	LOAD	laad(fout)
30	BREAK	afbreek(fout)

voortgezet ...

ER	Betekenis	Vertaling
31	CAN'T RESUME	kan niet doorgaan
32	LOOP NOT FOUND	lus niet gevonden
33	LOOP WITHOUT DO	LOOP zonder DO
34	DIRECT MODE ONLY	uitsluitend directe modus
35	SCREEN NOT OPEN	scherm niet open
36	BAD DISK	slechte disk
37	BEND NOT FOUND	BEND niet gevonden
38	LINE NUMBER TOO LARGE	regelnummer te hoog
39	UNRESOLVED REFERENCE	onopgeloste verwijzing
40	UNIMPLEMENTED COMMAND	ongeïmplementeerd commando
41	FILE READ	bestandlees(fout)
42	EDIT MODE	edit-modus
43	RESERVED NAME	gereserveerde naam
44	CBDOS DRIVES ONLY	uitsluitend CBDOS-drives
45	OK	Oké

Voorbeeld: Gebruik van **ER**

```
10 TRAP 100
20 PRINT SQR(-1)        : REM OPZETTELIJKE FOUT
30 PRINT "BIJ REGEL 30"  : REM PROGRAMMALOOP GAAT HIER VERDER
40 END
100 IF ER > 0 THEN PRINT ERR$(ER); " ERROR";
110 PRINT " OP REGEL"; EL
120 RESUME NEXT         : REM GA VERDER NA FOUT
```

ERASE

Token: $FE $2A

Formaat: **ERASE** bestandsnaam [**,D** station] [**,U** eenheid] [**,R**]

Gebruik: Verwijdert een schijfbestand.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

R herstelt een eerder verwijderd bestand. Dit werkt alleen wanneer er geen schrijfbewerkingen hebben plaatsgevonden tussen het verwijderen en het herstellen, de inhoud van de schijf mogelijk hebben gewijzigd.

Opmerkingen:

ERASE is een synoniem van **SCRATCH** en **DELETE**.

In de directe modus wordt het succes en het aantal verwijderde bestanden afgedrukt. De voorlaatste waarde uit het bericht bevat het aantal succesvol verwijderde bestanden.

Voorbeelden:

Gebruik van **ERASE**

```
ERASE "DRM", U9          : REM WIS HET BESTAND "DRM" OP UNIT 9
01, FILES SCRATCHED,01,00

ERASE "OUD*"             : REM WIS ALLE BESTANDEN BEGINEND MET "OUD"
01, FILES SCRATCHED,04,00

ERASE "R*=PRG"           : REM WIS ALLE PROGRAMMABESTANDEN BEGINNEND MET "R"
01, FILES SCRATCHED,09,00
```

ERR$

Token: $D3

Formaat: **ERR$**(getal)

Retourneert:

De tekenreeks met de beschrijving van een gegeven BASIC-foutnummer.

nummer BASIC-foutnummer (bereik 1 - 44).

Deze functie wordt doorgaans gebruikt in een **TRAP**-routine, waarbij het foutnummer wordt gelezen uit de gereserveerde variabele **ER**.

Opmerkingen:

Een BASIC-foutnummer buiten het bereik (1 - 44) geeft een `OK`.

Voorbeelden:

Gebruik van **ERR$**

```
10 TRAP 100
20 PRINT SQR(-1)        : REM OPZETTELIJKE FOUT
30 PRINT "BIJ REGEL 30"  : REM PROGRAMMALOOP GAAT HIER VERDER
40 END
100 IF ER > 0 THEN PRINT ERR$(ER); " ERROR";
110 PRINT " OP REGEL"; EL
120 RESUME NEXT         : REM GA VERDER NA FOUT
```

Alle foutnummers en hun beschrijvingen weergeven

```
100 SCNCLR
110 FOR N = 0 TO 255
120 PRINT USING "###"; N; ": "; ERR$(N)
130 IF MOD(N, 16) <> 15 THEN 160
140 PRINT "[ MEER ]"
150 GETKEY A$
160 NEXT N
170 PRINT "DONE"
```

EXIT

Token: $ED

Formaat: **EXIT**

Gebruik: Verlaat de huidige **DO** ... **LOOP** en vervolgt de uitvoering bij het eerste instructie na **LOOP**.

Opmerkingen:
In geneste lussen verlaat **EXIT** alleen de huidige lus en vervolgt de uitvoering in de buitenste lus (indien aanwezig).

Voorbeeld: Gebruik van **EXIT**

```
1 REM EXIT
10 OPEN 2, 8, 0, "$"          : REM OPEN CATALOGUS
15 IF DS THEN PRINT DS$ : STOP : REM KAN NIET LEZEN
20 GET#2, D$, D$              : REM LAADADRES WEGGOOIEN
25 DO                         : REM REGELLUS
30   GET#2, D$, D$            : REM REGELPOINTER WEGGOOIEN
35   IF ST THEN EXIT          : REM END-OF-FILE
40   GET#2, LO, HI            : REM BESTANDSGROOTTEBYTES
45   S = LO + 256 * HI        : REM BESTANDSGROOTTE
50   LINE INPUT#2, F$         : REM BESTANDSNAAM
55   PRINT S; F$              : REM PRINT BESTANDSINGANG
60 LOOP
65 CLOSE 2
```

EXP

Token: $BD

Formaat: **EXP**(numerieke expressie)

Retourneert:

De waarde van de wiskundige constante het getal van Euler (2.71828183) verheven tot de macht van het argument.

Opmerkingen:

Een argument groter dan 88 veroorzaakt een Overflow-fout.

Voorbeelden:

Gebruik van **EXP**

```
PRINT EXP(1)
 2.7182818

PRINT EXP(0)
 1

PRINT EXP(LOG(2))
 2
```

FAST

Token: $FE $25

Formaat: **FAST** [snelheid]

Gebruik: Stelt de kloksnelheid van de CPU in op 1 MHz, 3.5 MHz of 40 MHz.

snelheid kloksnelheid van de CPU, waarbij:

- 1: Stelt de CPU in op 1 MHz.
- 3: Stelt de CPU in op 3.5 MHz.
- Iets anders dan **1** of **3** stelt de CPU in op 40 MHz.

Opmerkingen:

Hoewel het mogelijk is **FAST** aan te roepen met elk reëel getal, wordt het fractionele deel (de decimale punt en alle cijfers daarna) genegeerd.

FAST is een synoniem van **SPEED**.

FAST heeft geen effect indien eerder `POKE 0, 65` is gebruikt om de CPU op 40 MHz in te stellen.

Voorbeelden:

Gebruik van **FAST**

```
FAST     : REM ZET SNELHEID MAXIMAAL (40 MHZ)

FAST 1   : REM ZET SNELHEID OP 1 MHZ

FAST 3   : REM ZET SNELHEID OP 3.5 MHZ

FAST 3.5 : REM ZET SNELHEID OP 3.5 MHZ
```

FGOSUB

Token: $FE $48

Formaat: **FGOSUB** numerieke expressie

Gebruik: Evalueert de gegeven numerieke expressie en voert vervolgens **GOSUB** uit op het subroutine bij het resulterende regelnummer.

Opmerkingen:

Wees voorzichtig bij het gebruik van **RENUMBER** om de regelnummers van uw programma te wijzigen; zorg dat eventuele **FGOSUB**-instructies nog steeds de beoogde nummers gebruiken. Indien het regelnummer niet bestaat, geeft FGOSUB een Undefined Statement-fout.

Voorbeeld: Gebruik van **FGOSUB**

```
 10 INPUT "WELKE SUBROUTINE UITVOEREN 100, 200, 300"; LI
 20 FGOSUB LI                    : REM HOPELIJK BESTAAT DIT REGEL#
 30 GOTO 10                      : REM HERHAAL
100 PRINT "OP REGEL 100" : RETURN
200 PRINT "OP REGEL 200" : RETURN
300 PRINT "OP REGEL 300" : RETURN
```

FGOTO

Token: $FE $47

Formaat: **FGOTO** numerieke expressie

Gebruik: Evalueert de gegeven numerieke expressie en gaat vervolgens met **GOTO** naar het resulterende regelnummer.

Opmerkingen:

Wees voorzichtig bij het gebruik van **RENUMBER** om de regelnummers van uw programma te wijzigen; zorg dat eventuele **FGOTO**-instructies nog steeds de beoogde nummers gebruiken. Indien het regelnummer niet bestaat, treedt een Undefined Statement-fout op.

Voorbeeld: Gebruik van **FGOTO**

```
 10 INPUT "WELK REGEL# UITVOEREN 100, 200, 300"; LI
 20 FGOTO LI           : REM HOPELIJK BESTAAT DIT REGEL#
 30 END
100 PRINT "OP REGEL 100" : END
200 PRINT "OP REGEL 200" : END
300 PRINT "OP REGEL 300" : END
```

FILTER

Token: $FE $03

Formaat: **FILTER** sid [{**,** frequentie**,** lp**,** bp**,** hp**,** res}]

Gebruik: Stelt de parameters van een SID-geluidsfilter in.

sid 1: rechter SID, 2: linker SID.

frequentie grensfrequentie van het filter (0 - 2047).

lp laagdoorlaatfilter (0: uit, 1: aan).

bp banddoorlaatfilter (0: uit, 1: aan).

hp hoogdoorlaatfilter (0: uit, 1: aan).

res resonantie (0 - 15).

Opmerkingen:

Ontbrekende parameters behouden hun huidige waarde. Het effectieve filter is de som van alle filterinstellingen. Hiermee worden bandstop- en notch-effecten mogelijk gemaakt.

Voorbeeld: Gebruik van **FILTER**

```
 10 PLAY "T7X103P9C"
 15 SLEEP 0.02
 20 PRINT "ZWAAI MET LAAGDOORLAATFILTER" : L = 1 : B = 0 : H = 0 : GOSUB 100
 30 PRINT "ZWAAI MET BANDDOORLAATFILTER" : L = 0 : B = 1 : H = 0 : GOSUB 100
 40 PRINT "ZWAAI MET HOOGDOORLAATFILTER" : L = 0 : B = 0 : H = 1 : GOSUB 100
 50 GOTO 20
100 REM *** ZWAAI ***
110 FOR F = 50 TO 1950 STEP 50
120 IF F >= 1000 THEN FF = 2000 - F : ELSE FF = F
130 FILTER 1, FF, L, B, H, 15
140 PLAY "X1"
150 SLEEP 0.02
160 NEXT F
170 RETURN
```

FIND

Token: $FE $2B

Formaat: **FIND /**tekenreeks**/** [**,** regelbereik]
FIND "tekenreeks**"** [**,** regelbereik]

Gebruik: Doorzoekt het BASIC-programma dat zich momenteel in het geheugen bevindt op alle voorkomens van een tekenreeks.

Het zoekt in een opgegeven regelbereik (indien gespecificeerd); anders wordt het volledige BASIC-programma doorzocht.

Bij elk voorkomen van de tekenreeks wordt de regel weergegeven met de tekenreeks gemarkeerd.

NO SCROLL kan worden gebruikt om de uitvoer te pauzeren.

Opmerkingen:

Bijna elk teken dat geen deel uitmaakt van de tekenreeks, waaronder letters en leestekens, kan in plaats van de "/" (schuine streep) worden gebruikt.

Het gebruik van dubbele aanhalingstekens (**"**) als scheidingsteken heeft een speciaal effect: de zoektekst wordt niet getokeniseerd. **FIND "FOR"** zoekt naar de drie letters "F," "O," en "R," niet naar het BASIC-sleutelwoord **FOR**. Daardoor kan het het woord **FOR** vinden in stringconstanten of **REM**-instructies, maar niet in programmacode.

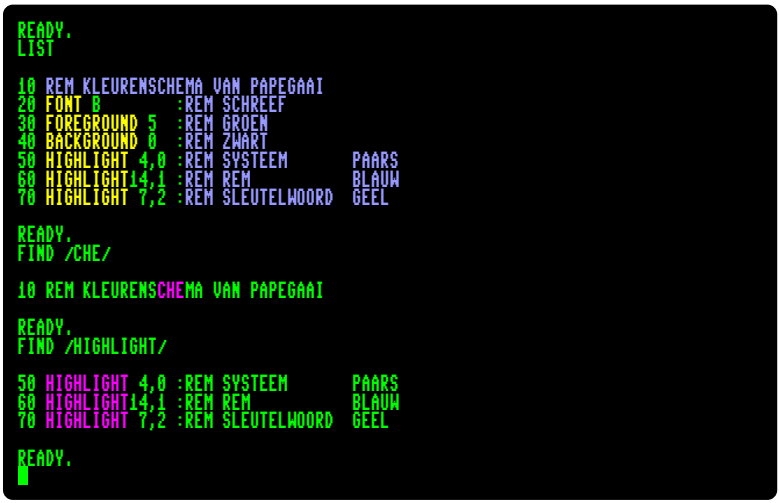
Anderzijds vindt **FIND /FOR/** alle voorkomens van het BASIC-sleutelwoord, maar niet de tekst "FOR" in strings.

Gedeeltelijke sleutelwoorden kunnen niet worden gezocht. **FIND /LOO/** vindt bijvoorbeeld het sleutelwoord **LOOP** niet.

Vanwege de wijze waarop BASIC wordt verwerkt, is het zoeken van de sleutelwoorden **REM** en **DATA** alleen mogelijk met de dubbelepunt als scheidingsteken: **FIND :REM TODO:** Dit werkt niet met het **CHANGE**-commando.

FIND is een editorcommando dat alleen in de directe modus kan worden gebruikt.

Voorbeeld: Gebruik van **FIND**


READY.
LIST
10 REM KLEURENSCHEMA VAN PAPEGAAI
20 FONT B :REM SCHREEF
30 FOREGROUND 5 :REM GROEN
40 BACKGROUND 0 :REM ZWART
50 HIGHLIGHT 4,0 :REM SYSTEEM PAARS
60 HIGHLIGHT 4,1 :REM REM BLAUW
70 HIGHLIGHT 7,2 :REM SLEUTELWOORD GEEL
READY.
FIND /CHE/
10 REM KLEURENSCHEMA VAN PAPEGAAI
READY.
FIND /HIGHLIGHT/
50 HIGHLIGHT 4,0 :REM SYSTEEM PAARS
60 HIGHLIGHT 4,1 :REM REM BLAUW
70 HIGHLIGHT 7,2 :REM SLEUTELWOORD GEEL
READY.

FN

Token: $A5

Formaat: **FN** naam(numerieke expressie)

Gebruik: **FN**-functies zijn door de gebruiker gedefinieerde functies die een numerieke expressie als argument aanvaarden en een reële waarde retourneren. Ze moeten eerst met **DEF FN** worden gedefinieerd voordat ze kunnen worden gebruikt.

naam naam van de functie zoals eerder gedefinieerd met **DEF FN**.

Voorbeeld: Gebruik van **FN**

```
10 PD = π / 180
20 DEF FN CD(X) = COS(X * PD)     : REM COS VOOR GRADEN
30 DEF FN SD(X) = SIN(X * PD)     : REM SIN VOOR GRADEN
40 FOR D = 0 TO 360 STEP 90
50 PRINT USING "####"; D;
60 PRINT USING " ##.##"; FNCD(D);
70 PRINT USING " ##.##"; FNSD(D)
80 NEXT D

RUN
  0  1.00  0.00
 90  0.00  1.00
180 -1.00  0.00
270  0.00 -1.00
360  1.00  0.00
```

FONT

Token: $FE $46

Formaat: **FONT** <**A** | **B** | **C**>

Gebruik: Wisselt alle tekens naar het opgegeven ingebouwde lettertype.

FONT A is het PETSCII-lettertype waarbij verschillende kleine letters zijn vervangen door ASCII-leestekens.

FONT B is een alternatieve weergave van **FONT A**.

FONT C is het PETSCII-lettertype. Dit is de standaardinstelling wanneer de MEGA65 voor het eerst wordt ingeschakeld.

Hiermee worden alle wijzigingen die met het **CHARDEF**-commando zijn aangebracht, ongedaan gemaakt.

De ASCII-symbolen van lettertypen **A** en **B** worden ingevoerd door de toetsen in de onderstaande tabel in te drukken, waarvan er enkele ook het ingedrukt houden van de  -toets vereisen. De codes voor hoofd- letters en kleine letters zijn omgedraaid ten opzichte van ASCII.

Code	Toets	PETSCII	ASCII
$5C	Pond	£	\ (schuine streep)
$5E	Pijl omhoog (naast RESTORE)	↑	^ (accentcircumflex)
$5F	Pijl links (naast 1)	←	_ (laagstreep)
$7B	MEGA + dubbele punt	┼	{ (openende accolade)
$7C	MEGA + punt	┬	| (verticale streep)
$7D	MEGA + puntkomma	┤	} (sluitende accolade)
$7E	MEGA + komma	├	~ (tilde)

Opmerkingen:

De extra ASCII-symbolen die door **FONT A** en **B** worden aangeboden, zijn alleen beschikbaar bij gebruik van de kleine-letterset.

FONT D, **FONT E** en **FONT F** komen overeen met respectievelijk **A**, **B** en **C**, waarbij het lettertype automatisch naar de kleine-lettermodus wordt geschakeld.

Voorbeelden:

Gebruik van **FONT**

```
FONT A : REM ASCII - ACTIVEER (!)←↑
FONT B : REM VERGELIJKBAAR MET A, MET SCHREEF
FONT C : REM COMMODORE-FONT (STANDAARD)
```

FOR

Token: $81

Formaat: **FOR** index = begin **TO** einde [**STEP** stap] ... **NEXT** [index]

Gebruik: **FOR**-instructies starten een BASIC-lus met een indexvariabele.

index kan bij elke iteratie met een constante waarde worden verhoogd of verlaagd. Standaard wordt de variabele met 1 verhoogd. De index-variabele moet een reële variabele zijn.

begin wordt gebruikt om de index te initialiseren.

einde wordt aan het einde van een iteratie gecontroleerd en bepaalt of nog een iteratie wordt uitgevoerd of dat de lus wordt afgesloten.

stap definieert de wijziging die aan het einde van een iteratie op de indexvariabele wordt toegepast. Positieve stapwaarden verhogen deze, terwijl negatieve waarden deze verlagen. Standaard is de waarde 1.0 indien niet opgegeven.

Opmerkingen:

Bij positieve verhogingen moet **einde** groter of gelijk zijn aan **begin**, terwijl bij negatieve verhogingen **einde** kleiner of gelijk moet zijn aan **begin**.

Het is een slechte programmeerpraktijk om de waarde van de **index**-variabele binnen de lus te wijzigen of met **GOTO** in of uit een lus te springen.

Voorbeelden:

Gebruik van **FOR**

```
10 FOR D = 0 TO 360 STEP 30
20 R = D * π / 180
30 PRINT D; R; SIN(R); COS(R); TAN(R)
40 NEXT D

10 DIM M(20, 20)
20 FOR I = 0 TO 20
30 FOR J = I TO 20
40 M(I, J) = I + 100 * J
50 NEXT J, I
```

FOREGROUND

Token: $FE $39

Formaat: **FOREGROUND** kleur

Gebruik: Stelt de voorgrondtekstkleur in voor de volgende **PRINT**-commando's.

kleur paletindex in het bereik 0 - 31.

Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard systeempalet.

Opmerkingen:

Dit is een andere naam voor **COLOR**.

Voorbeeld: Gebruik van **FOREGROUND**



FORMAT

Token: $FE $37

Formaat: **FORMAT** schijfnaam [**,I** id] [**,D** station] [**,U** eenheid]

Gebruik: Formatteert een schijf. OPMERKING: *Hierbij worden alle gegevens op de schijf gewist.*

I schijf-ID. De maximale lengte is 2 tekens.

schijfnaam is een tekenreeks tussen aanhalingstekens, bijv. `"DATA"` of een tekenreeksexpressie tussen haakjes, bijv. `(DN$)`. De maximale lengte is 16 tekens.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

FORMAT is een andere naam voor het commando **HEADER**.

Voor nieuwe diskettes die nog niet in MEGA65 (1581)-formaat zijn geformatteerd, moet het schijf-ID worden opgegeven met de parameter **I**. Hiermee wordt het formaatcommando overgeschakeld naar een laagformattering, waarbij sector-ID's worden geschreven en alle inhoud wordt gewist. Dit kost enige tijd, omdat elk blok op de diskette wordt geschreven.

Indien de parameter **I** wordt weggelaten, wordt een snelle formattering uitgevoerd. Dit is alleen mogelijk als de schijf al is geformatteerd als MEGA65- of 1581-diskette. Een snelle formattering schrijft de nieuwe schijfnaam, wist de eerste directorysector en de bloktoewijzingstabel, en markeert alle blokken als vrij. Het schijf-ID wordt niet gewijzigd. Omdat de directory wordt gewist, kunnen bestanden na een snelle formattering niet meer worden hersteld met **ERASE R**. U kunt meer lezen over **ERASE** op bladzijde B-106.

Voorbeelden:

Gebruik van **FORMAT**

```
FORMAT "AVONTUUR", IDK   : REM SCHIJF VOLLEDIG FORMATTEREN MET NAAM "ADVENTURE" EN ID "DK"

FORMAT "ZORK-I", U9      : REM SCHIJF SNEL FORMATTEREN IN UNIT 9 MET NAAM "ZORK-I"

FORMAT "GEWELF", D1, U10 : REM SCHIJF SNEL FORMATTEREN IN DRIVE 1 UNIT 10 MET NAAM "GEWELF"
```

FRE

Token: $B8

Formaat: **FRE**(bank)

Retourneert:

Informatie over het geheugengebruik of andere systeemkenmerken.

FRE(0) retourneert het aantal vrije bytes in bank 0, die wordt gebruikt voor de BASIC-programmabron.

FRE(1) retourneert het aantal vrije bytes in bank 1, de bank voor BASIC-variabelen, arrays en strings. **FRE(1)** activeert tevens "garbage collection", een proces dat strings in gebruik bovenaan de bank verzamelt en daarmee het stringgeheugen defragmenteert.

FRE(4) en **FRE(5)** retourneren respectievelijk het geheugengebruik van banken 4 en 5 als een 8-bits bitveld. Elke bit vertegenwoordigt 8KB geheugen in de bank. Bit 0 van **FRE(4)** bijvoorbeeld vertegenwoordigt het geheugengebied 4.0000 - 4.1FFF. Indien een bit is ingesteld, is het gereserveerd, hetzij door **MEM** hetzij door het systeem. Zie **MEM**.

FRE(-1) retourneert de ROM-versie, een zescijferig getal in het formaat 92XXXX.

Voorbeeld: Gebruik van **FRE**

```
10 PM = FRE(0)
20 VM = FRE(1)
30 RV = FRE(-1)
40 PRINT PM; " BYTES BESCHIKBAAR VOOR PROGRAMMA"
50 PRINT VM; " BYTES BESCHIKBAAR VOOR VARIABELEN"
60 PRINT RV; " ROM-VERSIE"
```

FREAD#

Token: $FE $1C

Formaat: **FREAD#** kanaal**,** pointer**,** grootte

Gebruik: Leest **grootte** bytes van **kanaal** in het geheugen, beginnend op het 32-bits adres **pointer**.

kanaal nummer, dat werd opgegeven bij een eerdere aanroep van commando's zoals **DOPEN** of **OPEN**.

FREAD# kan worden gebruikt om gegevens direct van schijf in een variabele te lezen. Het wordt aanbevolen de **POINTER**-instructie te gebruiken voor het pointer-argument en de grootteparameter te berekenen door het aantal elementen te vermenigvuldigen met de itemgrootte.

Type	Itemgrootte
Byte-array	1
Integer-array	2
Real-array	5

Opmerkingen:

Houd er rekening mee dat de **POINTER**-functie met een stringargument *niet* het stringadres retourneert, maar de stringdescriptor. Het wordt afgeraden **FREAD#** te gebruiken voor strings of stringarrays, tenzij u volledig op de hoogte bent van de interne werking van de stringopslag.

Om in een array te lezen, moet u altijd een array-index opgeven zodat **POINTER** het adres van een element retourneert. Het startadres van array `XY()` is `POINTER(XY(0))`. `POINTER(XY)` retourneert het adres van de scalaire variabele `XY`.

Voorbeeld: Gebruik van **FREAD#**

```
100 N = 23
110 DIM B&(N), C&(N)
120 DOPEN#2, "TEKST"
130 FREAD#2, POINTER(B&(0)), N
140 DCLOSE#2
150 FOR I = 0 TO N - 1 : PRINT CHR$(B&(I)); : NEXT
160 FOR I = 0 TO N - 1 : C&(I) = B&(N - 1 - I) : NEXT
170 DOPEN#2, "REVERS,W"
180 FWRITE#2, POINTER(C&(0)), N
190 DCLOSE#2
```

FREEZER

Token: $FE $4A

Formaat: **FREEZER**

Gebruik: Opent het Freezer-menu.

Opmerkingen:
Het invoeren van het commando **FREEZER** is een alternatief voor het indrukken en loslaten van de RESTORE-toets.

Voorbeeld: Gebruik van **FREEZER**

```
FREEZER : REM FREEZERMENU AANROEPEN
```

FWRITE#

Token: $FE $1E

Formaat: **FWRITE#** kanaal**,** pointer**,** grootte

Gebruik: Schrijft **grootte** bytes naar **kanaal** vanuit het geheugen, beginnend op het 32-bits adres **pointer**.

kanaal kanaalnummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

FWRITE# kan worden gebruikt om de waarde van een variabele naar een bestand te schrijven.

Het wordt aanbevolen de **POINTER**-instructie te gebruiken voor het pointer-argument en de grootteparameter te berekenen door het aantal elementen te vermenigvuldigen met de itemgrootte.

Zie de tabel met itemgroottes voor **FREAD#** op bladzijde B-122.

Opmerkingen:

Houd er rekening mee dat de **POINTER**-functie met een stringargument *niet* het stringadres retourneert, maar de stringdescriptor. Het wordt afgeraden **FWRITE#** te gebruiken voor strings of stringarrays, tenzij u volledig op de hoogte bent van de interne werking van de stringopslag.

Om een array te schrijven, moet u altijd een array-index opgeven zodat **POINTER** het adres van een element retourneert. Het startadres van array `XY()` is `POINTER(XY(0))`. `POINTER(XY)` retourneert het adres van de scalaire variabele `XY`.

Voorbeeld: Gebruik van **FWRITE#**

```
100 N = 23
110 DIM B&(N), C&(N)
120 DOPEN#2, "TEKST"
130 FREAD#2, POINTER(B&(0)), N
140 DCLOSE#2
150 FOR I = 0 TO N - 1 : PRINT CHR$(B&(I)); : NEXT
160 FOR I = 0 TO N - 1 : C&(I) = B&(N - 1 - I) : NEXT
170 DOPEN#2, "REVERS,W"
180 FWRITE#2, POINTER(C&(0)), N
190 DCLOSE#2
```

GCOPY

Token: $FE $32

Formaat: **GCOPY** x, y, breedte, hoogte

Gebruik: Bitmapgrafieken: kopieert de inhoud van de opgegeven rechthoek met de bovenliggende positie **x, y** en de **breedte** en **hoogte** naar een buffer.

Het gekopieerde gebied kan op elke positie worden geplakt met het commando **PASTE**.

Opmerkingen:

De grootte van de rechthoek is begrensd door de omvang van 1KB van de buffer. De geheugenvereiste voor een gebied is breedte * hoogte * aantal bitplanes / 8. Dit mag niet groter zijn dan of gelijk aan 1024 bytes. Voor een scherm met 4 bitplanes bijvoorbeeld heeft een gebied van 45 x 45 1012,5 bytes nodig.

Voorbeeld: Gebruik van **GCOPY**

```
10 SCREEN 320, 200, 2
20 BOX 60, 60, 300, 180, 1 : REM TEKEN EEN WIT VAK
30 GCOPY 140, 80, 40, 40   : REM KOPIEER EEN GEBIED VAN 40 * 40
40 PASTE 10, 10            : REM PLAK HET OP EEN NIEUWE POSITIE
50 GETKEY A$               : REM WACHT OP TOETSDRUK
60 SCREEN CLOSE
```

GET

Token: $A1

Formaat: **GET** variabele

Gebruik: Haalt het volgende teken, of de byte-waarde van het volgende teken, uit de toetsenbordwachtrij.

Indien de **variabele** die wordt ingesteld een stringvariabele is en de wachtrij leeg is, wordt een lege string toegewezen; anders wordt een string van één teken aangemaakt en toegewezen.

Indien de **variabele** een numerieke variabele is, wordt de byte-waarde van de toets toegewezen; anders wordt nul toegewezen indien de wachtrij leeg is. **GET** wacht niet op toetsinvoer en is daarom bruikbaar om op regelmatige intervallen of in lussen op toetsaanslagen te controleren.

Opmerkingen:

GETKEY werkt vergelijkbaar, maar wacht tot er een toets is ingedrukt.

Voorbeeld: Gebruik van **GET**

```
10 DO : GET A$ : LOOP UNTIL A$ <> ""
20 IF A$ = "W" THEN PRINT "NOORD"  : REM GA NOORD
30 IF A$ = "A" THEN PRINT "WEST"   : REM GA WEST
40 IF A$ = "S" THEN PRINT "OOST"   : REM GA OOST
50 IF A$ = "Z" THEN PRINT "ZUID"   : REM GA ZUID
60 IF A$ = CHR$(13) THEN END       : REM ERUIT
70 GOTO 10
```

GET#

Token: $A1 '#'

Formaat: **GET#** kanaal**,** variabele [**,** variabele ...]

Gebruik: Leest een enkele byte van het kanaalargument en wijst tekenreeksen van één teken toe aan stringvariabelen, of een 8-bits binair getal aan numerieke variabelen.

Dit is bruikbaar om tekens (of bytes) een voor een uit een invoerstroom te lezen.

kanaal nummer, dat werd opgegeven bij een eerdere aanroep van commando's zoals **DOPEN** of **OPEN**.

Opmerkingen:

Alle waarden van 0 tot en met 255 zijn geldig, zodat **GET#** ook kan worden gebruikt om binaire gegevens te lezen.

Voorbeeld: Gebruik van **GET#**

```
10 OPEN 2, 8, 0, "$"          : REM OPEN CATALOGUS
15 IF DS THEN PRINT DS$ : STOP : REM KAN NIET LEZEN
20 GET#2, D$, D$              : REM LAADADRES WEGGOOIEN
25 DO                         : REM REGELLUS
30   GET#2, D$, D$            : REM REGELPOINTER WEGGOOIEN
35   IF ST THEN EXIT          : REM END-OF-FILE
40   GET#2, LO, HI            : REM BESTANDSGROOTTEBYTES
45   S=LO+256*HI              : REM BESTANDSGROOTTE
50   LINE INPUT#2, F$         : REM BESTANDSNAAM
55   PRINT S;F$               : REM PRINT BESTANDSINGANG
60 LOOP
65 CLOSE 2
```

GETKEY

Token: $A1 (GET) $F9 (KEY)

Formaat: **GETKEY** variabele

Gebruik: Haalt het volgende teken, of de byte-waarde van het volgende teken, uit de toetsenbordwachtrij. Indien de wachtrij leeg is, wacht het programma tot er een toets is ingedrukt.

Nadat er een toets is ingedrukt, wordt de **variabele** ingesteld en gaat de programma-uitvoering verder. Bij gebruik met een stringvariabele wordt een string van één teken aangemaakt en toegewezen. Indien de variabele een numerieke variabele is, wordt de byte-waarde toegewezen.

Voorbeeld: Gebruik van **GETKEY**

```
10 GETKEY A$                    : REM WACHTEN LEES TEKEN
20 IF A$ = "W" THEN PRINT "NOORD" : REM GA NOORD
30 IF A$ = "A" THEN PRINT "WEST"  : REM GA WEST
40 IF A$ = "S" THEN PRINT "OOST"  : REM GA OOST
50 IF A$ = "Z" THEN PRINT "ZUID"  : REM GA ZUID
60 IF A$ = CHR$(13) THEN END      : REM EINDE
70 GOTO 10
```

GO64

Token: $CB (GO) $36 $34 ("64")

Formaat: **GO64**

Gebruik: Schakelt de MEGA65 over naar de C64-compatibele modus.

Indien u zich in de directe modus bevindt, wordt een beveiligingsprompt `ARE YOU SURE?` weergegeven, waarop u met Y moet antwoorden om door te gaan.

U kunt terugkeren naar de MEGA65-modus met het commando `SYS 58552`.

Opmerkingen:

Indien u met `SYS 58552` terugkeert naar de MEGA65, vindt de overschakeling onmiddellijk plaats. Er wordt geen bevestiging gevraagd.

Voorbeeld: Gebruik van **GO64**

```
READY.
GO64
ARE YOU SURE? Y
```

Terugkeren naar MEGA65

```
READY.
SYS 58552
```

GOSUB

Token: $8D

Formaat: **GOSUB** regelnummer

Gebruik: **GOSUB** (GOto SUBroutine) zet de programma-uitvoering voort op de opgegeven BASIC-regel, waarbij de huidige BASIC-programmateller en regelnummer op de run-tijdstack worden bewaard.

Hiermee kan de uitvoering worden hervat na het **GOSUB**-statement, zodra een **RETURN**-statement in de aangeroepen subroutine is uitgevoerd.

Aanroepen van subroutines via **GOSUB** mogen genest worden, maar de subroutines moeten altijd eindigen met **RETURN**, anders kan een stack overflow optreden.

Opmerkingen:

In tegenstelling tot andere programmeertalen ondersteunt BASIC geen argumenten of lokale variabelen voor subroutines. Programma's kunnen worden geoptimaliseerd door subroutines te groeperen aan het begin van de programmabron. De **GOSUB**-aanroepen hebben dan lage regelnummers met minder cijfers om te verwerken. De subroutines worden ook sneller gevonden, omdat het zoeken naar subroutines vaak begint aan het begin van het programma.

Voorbeeld: Gebruik van **GOSUB**

```
 10 GOTO 100                                  : REM NAAR HOOFDPROGRAMMA
 20 REM *** SUBROUTINE DISKSTATUSCONTROLE ***
 30 DD = DS : IF DD THEN PRINT "DISKFOUT"; DS$
 40 RETURN
 50 REM *** SUBROUTINEPROMPT J/N ***
 60 DO : INPUT "DOORGAAN (J/N)"; A$
 70 LOOP UNTIL A$ = "J" OR A$ = "N"
 80 RETURN
 90 REM *** HOOFDPROGRAMMA ***
100 DOPEN#2, "BIG DATA"
110 GOSUB 30 : IF DD THEN DCLOSE#2 : GOSUB 60   : REM VRAAG
120 IF A$ = "N" THEN END
130 GOTO 100                                  : REM PROBEER NOGMAALS
```

GOTO

Token: $89 (GOTO) of $CB $A4 (GO TO)

Formaat: **GOTO** regel
GO TO regel

Gebruik: Zet de programma-uitvoering voort op de opgegeven BASIC-regel.

Opmerkingen:

Als het doel**regel**nummer groter is dan het huidige regelnummer, begint de zoekactie vanaf de huidige regel en gaat door naar hogere regelnummers. Als het doel**regel**nummer lager is, begint de zoekactie bij het eerste **regel**nummer van het programma. Het is mogelijk de uitvoeringssnelheid van het programma te optimaliseren door veelgebruikte doelen aan het begin te groeperen (met lagere regelnummers).

GOTO (als één woord geschreven) wordt sneller uitgevoerd dan **GO TO**.

Voorbeeld: Gebruik van **GOTO**

```
 10 GOTO 100                                  : REM NAAR HOOFDPROGRAMMA
 20 REM *** SUBROUTINE DISKSTATUSCONTROLE ***
 30 DD = DS : IF DD THEN PRINT "DISKFOUT"; DS$
 40 RETURN
 50 REM *** SUBROUTINEPROMPT J/N ***
 60 DO : INPUT "DOORGAAN (J/N)"; A$
 70 LOOP UNTIL A$ = "J" OR A$ = "N"
 80 RETURN
 90 REM *** HOOFDPROGRAMMA ***
100 DOPEN#2, "BIG DATA"
110 GOSUB 30 : IF DD THEN DCLOSE#2 : GOSUB 60   : REM VRAAG
120 IF A$ = "N" THEN END
130 GOTO 100                                  : REM PROBEER NOGMAALS
```

GRAPHIC

Token: $DE

Formaat: **GRAPHIC CLR**

Gebruik: Bitmapgrafieken: initialiseert het BASIC-bitmapgrafiekensysteem. Het wist het grafisch geheugen en scherm, en stelt alle parameters van de grafische context in op hun standaardwaarden.

Nadat het grafisch systeem is gewist, kunnen commando's zoals **LINE**, **PALETTE**, **PEN**, **SCNCLR** en **SCREEN** worden gebruikt om parameters van het grafisch systeem in te stellen.

Voorbeeld: Gebruik van **GRAPHIC**

```
110 GRAPHIC CLR           : REM INITIALISEER
120 SCREEN DEF 1, 1, 1, 2 : REM 640 X 400 X 2
130 SCREEN OPEN 1         : REM OPEN HET
140 SCREEN SET 1, 1       : REM TOON HET
150 PALETTE 1, 0, 0,  0, 0 : REM ZWART
160 PALETTE 1, 1, 0, 15, 0 : REM GROEN
170 SCNCLR 0              : REM VUL SCHERM MET ZWART
180 PEN 0, 1              : REM KIES PEN
190 LINE 50, 50, 590, 350 : REM TEKEN LIJN
200 GETKEY A$             : REM WACHT OP TOETSDRUK
210 SCREEN CLOSE 1        : REM SLUIT SCHERM EN HERSTEL PALET
```

HASBIT

Token: $CE $13

Formaat: **HASBIT**(adres, bitnummer)

Retourneert:

Waar (-1) indien de waarde op het adres de opgegeven bit heeft ingesteld, anders onwaar (0).

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

bitnummer waarde in het bereik van 0 - 7.

Een bankwaarde > 127 wordt gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enz.

Voorbeelden:

Gebruik van **HASBIT**

```
BANK 128

SETBIT $D031,6

PRINT HASBIT($D031,6)
-1

CLRBIT $D031,6

PRINT HASBIT($D031,6)
0
```

HEADER

Token: $F1

Formaat: **HEADER** schijfnaam [**,I** id] [**,D** station] [**,U** eenheid]

Gebruik: Formatteert een schijf. OPMERKING: *Hierbij worden alle gegevens op de schijf gewist.*

I schijf-ID. Maximale lengte is 2 tekens.

schijfnaam is een tekenreeks tussen aanhalingstekens, bijv. "DATA" of een tekenreeksexpressie tussen haakjes, bijv. (DN$). De maximale lengte is 16 tekens.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

HEADER is een andere naam voor het commando **FORMAT**.

Voor nieuwe diskettes die nog niet in MEGA65 (1581)-formaat zijn geformatteerd, moet het schijf-ID worden opgegeven met de parameter **I**. Hiermee wordt het formaatcommando overgeschakeld naar een laagformattering, waarbij sector-ID's worden geschreven en alle inhoud wordt gewist. Dit kost enige tijd, omdat elk blok op de diskette wordt geschreven.

Indien de parameter **I** wordt weggelaten, wordt een snelle formattering uitgevoerd. Dit is alleen mogelijk als de schijf al is geformatteerd als MEGA65- of 1581-diskette. Een snelle formattering schrijft de nieuwe schijfnaam, wist de eerste directorysector en de bloktoewijzingstabel, en markeert alle blokken als vrij. Het schijf-ID wordt niet gewijzigd. Omdat de directory wordt gewist, kunnen bestanden na een snelle formattering niet meer worden hersteld met **ERASE R**. U kunt meer lezen over **ERASE** op bladzijde B-106.

Voorbeelden:

Gebruik van **HEADER**

```
HEADER "AVONTUUR", IDK      : REM EEN DISKETTE VOLLEDIG FORMATTEREN MET NAAM "AVON-
TUUR" EN ID "DK"

HEADER "ZORK-I", U9         : REM EEN DISKETTE IN UNIT 9 SNEL FORMATTEREN MET NAAM "ZORK-I"

HEADER "KERKERGEWELF", D1, U10 : REM EEN DISKETTE SNEL FORMATTE-
REN IN DRIVE 1 VAN UNIT 10 MET NAAM "KERKERGEWELF"
```

HELP

Token: $EA

Formaat: **HELP**

Gebruik: Geeft informatie over de plaats waar een fout is opgetreden in een BASIC-programma.

Wanneer het BASIC-programma door een fout wordt onderbroken, kan **HELP** gebruikt worden om meer informatie te verkrijgen. De vertaalde regel wordt weergegeven, waarbij de foutieve instructie gemarkeerd of onderstreept wordt getoond.

Opmerkingen:

Geeft BASIC-fouten weer. Voor fouten met betrekking tot schijf-I/O moet de schijfstatusvariabele **DS** of de schijfstatustekenreeks **DS$** worden gebruikt.

Voorbeeld: Gebruik van **HELP**

```
10 A = 1.E20
20 B = A + A : C = EXP(A) : PRINT A, B, C
RUN

?OVERFLOW ERROR IN 20
READY.
HELP

20 B = A + A : C=EXP(A) : PRINT A, B, C
```

HEX$

Token: $D2

Formaat: **HEX$**(numerieke expressie)

Retourneert:

Een vier- of achtteken hexadecimale weergave van het argument.

Het argument moet in het bereik van 0 - 4294967295 liggen, wat overeenkomt met de hex-getallen $00000000-$FFFFFFFF. Indien het argument tussen 0 - 65535 ligt, wordt een tekenreeks van vier tekens geretourneerd. Voor alle waarden boven 65535 wordt in plaats daarvan een tekenreeks van acht tekens geretourneerd.

Opmerkingen:

Indien reële getallen als argumenten worden gebruikt, wordt het decimale deel genegeerd. Met andere woorden, reële getallen worden niet afgerond.

Om een hexadecimale tekenreeks om te zetten in een getal, zie **DEC**.

Voorbeelden:

Gebruik van **HEX$**

```
PRINT HEX$(10), HEX$(100), HEX$(1000.9)
000A     0064     03E8

PRINT HEX$(4294967295)
FFFFFFFF
```

HIGHLIGHT

Token: $FE $3D

Formaat: **HIGHLIGHT** kleur [, modus]

Gebruik: Stelt de kleuren in die voor de code-markering worden gebruikt.

Er kunnen verschillende kleuren worden ingesteld voor systeemmeldingen, **REM**-instructies en BASIC-trefwoorden.

kleur een van de eerste 16 kleuren in het huidige palet. Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard systeempalet.

modus geeft aan waarvoor de kleur wordt gebruikt.

- 0: Systeemmeldingen (de standaardmodus).
- 1: **REM**-instructies.
- 2: BASIC-trefwoorden.

Opmerkingen:

De kleur voor systeemmeldingen wordt gebruikt bij het weergeven van foutmeldingen en in de uitvoer van **CHANGE**, **FIND** en **HELP**. De kleuren voor **REM**-instructies en BASIC-trefwoorden worden door **LIST** gebruikt.

Voorbeeld: Gebruik van **HIGHLIGHT**

```
LIST

10 REM *** DIT IS HALLO WERELD ***
20 PRINT "HALLO WERELD"

READY.
HIGHLIGHT 0,2

READY.
LIST

10 REM *** DIT IS HALLO WERELD ***
20 PRINT "HALLO WERELD"

READY.
```

IF

Token: $8B

Formaat: **IF** expressie **THEN** waar-clausule [**ELSE** onwaar-clausule]

Gebruik: Voert instructies voorwaardelijk uit.

expressie logische of numerieke expressie. Een numerieke expressie wordt als **FALSE** beschouwd indien de waarde nul is en als **TRUE** voor elke waarde ongelijk aan nul.

waar-clausule een of meer instructies die direct na **THEN** op dezelfde regel beginnen. Een regelnummer na **THEN** voert in plaats daarvan een **GOTO** uit naar die regel.

onwaar-clausule een of meer instructies die direct na **ELSE** op dezelfde regel beginnen. Een regelnummer na **ELSE** voert in plaats daarvan een **GOTO** uit naar die regel.

Opmerkingen:

De standaardstructuur **IF** ... **THEN** ... **ELSE** is beperkt tot één regel. De **waar-clausule** en **onwaar-clausule** kunnen echter worden uitgesmeerd over meerdere regels met behulp van een samengestelde instructie tussen **BEGIN** ... **BEND**.

Een **ELSE**-clause kan een andere **IF**-instructie bevatten. Indien de interne **IF**-instructie meerdere regels beslaat met **BEGIN** ... **BEND**, moet de gehele interne **IF**-instructie ook tussen **BEGIN** ... **BEND** worden geplaatst.

Vanwege een ongebruikelijke beperking bij **BEGIN** ... **BEND**: indien een **THEN**-clause **BEGIN** ... **BEND** gebruikt en er een **ELSE**-clause aanwezig is, moet het trefwoord **ELSE** op dezelfde regel staan als de vorige **BEND**-instructie. Zie het onderstaande voorbeeld.

Voorbeeld: Gebruik van **IF**

```
10 ROOD$ = CHR$(28) : ZWART$ = CHR$(144) : WIT$ = CHR$(5)
20 INPUT "GEEF EEN GETAL"; V
30 IF V < 0 THEN PRINT ROOD$; : ELSE PRINT ZWART$;
40 PRINT V                                  : REM PRINT NEGATIEVE GETALLEN IN HET ROOD
50 PRINT WIT$
60 INPUT "PROGRAMMA STOPPEN (J/N)"; A$
70 IF A$ = "J" THEN END
80 IF A$ = "N" THEN 20 : ELSE 60
```

```
 10 X=1
 20 IF X=0 THEN BEGIN
 30   PRINT "NUL"
 40 BEND:ELSE BEGIN
 50   IF X=1 THEN BEGIN
 60     PRINT "EEN"
 70   BEND:ELSE BEGIN
 80     IF X=2 THEN BEGIN
 90       PRINT "TWEE"
100     BEND:ELSE PRINT "ANDERS"
110   BEND
120 BEND
```

IMPORT

Token: $DD

Formaat: **IMPORT** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Laadt BASIC-code in tekstformaat van een bestand van het type **SEQ** in het geheugen dat voor BASIC-programma's is gereserveerd.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Het programma wordt in het BASIC-geheugen geladen en van tekst omgezet naar de getokeniseerde vorm van **PRG**-bestanden. Hiermee kunnen BASIC-programma's worden geladen die als platte tekst als programmalijst zijn opgeslagen.

Na het laden wordt het programma opnieuw gelinkt en is het gereed om te worden **RUN** of bewerkt. Het is mogelijk **IMPORT** te gebruiken om een programmatekstbestand van de schijf samen te voegen met een programma dat zich al in het geheugen bevindt. Elke regel die uit het bestand wordt gelezen, wordt op dezelfde manier verwerkt alsof de gebruiker deze via de schermeditor zou hebben ingevoerd.

Er is geen **EXPORT**-tegenhanger, omdat deze functie al beschikbaar is. De volgorde `DOPEN#1,"LISTING",W:CMD 1:LIST:DCLOSE#1` converteert het programma in het geheugen naar tekst en schrijft het naar het bestand dat in de **DOPEN**-instructie is opgegeven.

IMPORT kan alleen in de directe modus worden gebruikt.

Voorbeelden:

Gebruik van **IMPORT**

```
IMPORT "APOCALYPS"

IMPORT "MEGA-TOOLTJES", U9

IMPORT (FI$), U(UN%)
```

INFO

Token: $FE $4D

Formaat: **INFO**

Gebruik: Geeft informatie over de uitvoeromgeving.

Opmerkingen:

Het commando **INFO** geeft informatie over de BASIC-uitvoeromgeving, waaronder:

- De videomodus (PAL, NTSC).
- De versie van de ROM.
- De CPU-snelheid.
- De huidige **MEM**-instelling.
- Het gebruikte geheugen en het beschikbare geheugen voor programmatekst en variabelen.

Voorbeeld: Gebruik van **INFO**

```
READY.
INFO

INFO: PAL     DATA BYTES  USED /  FREE
---------------------------------------
ROM-V   920408   PROGRAM:    7 / 55030
SPEED   40 MHZ   SCALARS:    0 /  1472
BANK4 --------   STRINGS:    0 / 54980
BANK5 --------   ARRAYS :    0 / 54980
```

INPUT

Token: $85

Formaat: **INPUT** [prompt <**,** | **;**>] variabele [**,** variabele ...]

Gebruik: Vraagt de gebruiker om toetsenbordinvoer door een optionele prompttekst en vraagteken op het scherm weer te geven.

prompt optionele tekenreeksexpressie die als prompt wordt afgedrukt.

Indien het scheidingsteken tussen **prompt** en **variabelenlijst** een komma (**,**) is, wordt de cursor direct na de prompt geplaatst. Indien het scheidingsteken een puntkomma (**;**) is, worden in plaats daarvan een vraagteken en een spatie aan de prompt toegevoegd.

variabelenlijst een of meer variabelen die de invoer ontvangen.

De invoer wordt verwerkt nadat de gebruiker RETURN indrukt.

Opmerkingen:

De gebruiker moet ervoor zorgen dat het juiste type invoer wordt ingevoerd, zodat deze overeenkomt met de typen van de **variabelenlijst**. Het aantal invoerelementen moet eveneens overeenkomen met het aantal variabelen. Een overschot aan invoerelementen wordt genegeerd, terwijl te weinig invoerelementen een nieuwe vraag om invoer uitlokt met de prompt ??.

Het invoeren van niet-numerieke tekens voor variabelen van het type geheel getal of reëel getal levert een Type Mismatch-fout op.

Strings voor stringvariabelen moeten tussen dubbele aanhalingstekens (**"**) staan indien ze spaties of komma's bevatten.

Veel programma's die een veilige invoerroutine nodig hebben, gebruiken **LINE INPUT** en een aangepaste parser om programmafouten door verkeerde invoer van de gebruiker te voorkomen.

Voorbeeld: Gebruik van **INPUT**

```
 10 DIM N$(100), A%(100), S$(100)
 20 DO
 30 INPUT "NAAM, LEEFTIJD, GESLACHT";NA$, AG%, SE$
 40 IF NA$ = "" THEN 30
 50 IF NA$ = "EIND" THEN EXIT
 60 IF AG% < 18 OR AG% > 100 THEN PRINT "LEEFTIJD?" : GOTO 30
 70 IF SE$ <> "M" AND SE$ <> "V" THEN PRINT "GESLACHT?" : GOTO 30
 80 REM CHECK OK, OPSLAAN IN ARRAY
 90 N$(N) = NA$ : A%(N) = AG% : S$(N) = SE$ : N = N + 1
100 LOOP UNTIL N = 100
110 PRINT "N;" NAMEN ONTVANGEN"
```

INPUT#

Token: $84

Formaat: **INPUT#** kanaal**,** variabele [**,** variabele ...]

Gebruik: Leest een record van een invoerapparaat, bijv. een schijfbestand, en wijst de gegevens toe aan de variabelen in de lijst.

kanaal nummer, dat werd opgegeven bij een eerdere aanroep van commando's zoals **DOPEN** of **OPEN**.

variabelenlijst een of meer variabelen die de invoer ontvangen.

Het invoerrecord moet worden afgesloten met een RETURN-teken en mag niet langer zijn dan de invoerbuffer (160 tekens).

Opmerkingen:

Het type en het aantal gegevens in een record moeten overeenkomen met de variabelenlijst. Het lezen van niet-numerieke tekens voor variabelen van het type geheel getal of reëel getal levert een File Data-fout op. Strings voor stringvariabelen moeten tussen dubbele aanhalingstekens staan indien ze spaties of komma's bevatten.

LINE INPUT# kan worden gebruikt om een volledig record in één stringvariabele te lezen.

Volgordelijke bestanden die met **INPUT#** kunnen worden gelezen, kunnen door programma's met **PRINT#** of met de editor worden aangemaakt.

Voorbeeld: Gebruik van **INPUT#**

```
READY.
EDIT ON                                    : REM BESTAND AANMAKEN

OK.
10 "CHUCK PEDDLE", 1937, "ONTWERPER VAN DE 6502"
20 "JACK TRAMIEL", 1928, "OPRICHTER VAN CBM"
30 "BILL MENSCH", 1945, "HARDWARE"

DSAVE "CBM-PERSONEN"

OK.
EDIT OFF

READY.
```

Lees bestand

```
 10 DIM N$(100), B%(100), S$(100)
 20 DOPEN#2, "CBM-PERSONEN"       : REM OPEN SEQ-BESTAND
 25 IF DS THEN PRINT DS$ : STOP   : REM OPENFOUT
 30 FOR I = 0 TO 100
 40 INPUT#2, N$(I), B%(I), S$(I)
 50 IF ST AND 64 THEN 80          : REM END OF FILE
 60 IF DS THEN PRINT DS$ : GOTO 80 : REM DISKFOUT
 70 NEXT I
 80 DCLOSE#2
110 PRINT "I + 1; "RECORDS GELEZEN"
120 FOR J = 0 TO I : PRINT N$(J) : NEXT J

RUN
3 RECORDS GELEZEN
CHUCK PEDDLE
JACK TRAMIEL
BILL MENSCH

TYPE "CBM-PERSONEN"
"CHUCK PEDDLE", 1937, "ONTWERPER VAN DE 6502"
"JACK TRAMIEL", 1928, "OPRICHTER VAN CBM"
"BILL MENSCH", 1945, "HARDWARE"
```

INSTR

Token: $D4

Formaat: **INSTR**(hooiberg**,** naald [**,** begin])

Gebruik: Zoekt de positie van de tekenreeksexpressie **naald** in de tekenreeksexpressie **hooiberg** en retourneert de index van het eerste voorkomen, of nul indien er geen overeenkomst is.

De tekenreeksexpressie **hooiberg** wordt doorzocht op het voorkomen van de tekenreeksexpressie **naald**.

Een verbeterde versie van tekenreekszoeken met patroonherkenning wordt gebruikt indien het eerste teken van de zoekreeks een '£' (pondteken) is. Het pondteken maakt geen deel uit van de zoekopdracht, maar maakt het mogelijk het '.' (punt) als jokerteken te gebruiken, dat overeenkomt met elk willekeurig teken. Het tweede speciale patroonteken is het '*' (asterisk) teken. De asterisk in de zoekreeks geeft aan dat het voorgaande teken nooit, eenmaal of herhaaldelijk mag voorkomen om als overeenkomst te worden beschouwd.

Het optionele argument **begin** is een integerexpressie die de startpositie voor de zoekopdracht in **hooiberg** bepaalt. Indien het ontbreekt, wordt standaard één gebruikt.

Opmerkingen:

Indien een van de tekenreeksen leeg is of er geen overeenkomst is, retourneert de functie nul.

Voorbeelden:

Gebruik van **INSTR**

```
I = INSTR("ABCDEF", "CD")    : REM I = 3
I = INSTR("ABCDEF", "XY")    : REM I = 0
I = INSTR("RAIIIN", "£A*IN") : REM I = 5
I = INSTR("ABCDEF", "£C.E")  : REM I = 3
I = INSTR(A$ + B$, C$)
```

INT

Token: $B5

Formaat: **INT(**numerieke expressie**)**

Retourneert:

Het gehele getal van een getal.

Dit wordt berekend als de "floor" van het getal, waarbij het decimale deel naar negatieve oneindigheid wordt afgerond. `INT(3.1) = 3`; `INT(-3.1) = -4`.

Deze functie is *niet* beperkt tot het gebruikelijke 16-bit integerbereik (-32768 tot 32767), omdat reële rekenkunde wordt gebruikt. Het toegestane bereik wordt daarom bepaald door de grootte van de reële mantissa, die 32 bits breed is (-2147483648 tot 2147483647).

Opmerkingen:

Het is niet nodig de functie **INT** te gebruiken om reële waarden toe te wijzen aan integer- of bytevariabelen, omdat deze conversie impliciet wordt uitgevoerd.

Voorbeelden:

Gebruik van **INT**

```
X = INT(1.9)      : REM X = 1
X = INT(-3.1)     : REM X = -4
X = INT(100000.5) : REM X = 100000
N% = INT(100000.5) : REM ?ILLEGAL QUANTITY ERROR
```

JOY

Token: $CF

Formaat: **JOY**(poort)

Retourneert:

De stand van de joystick voor de geselecteerde controller**poort** (1 of 2).

Een **poort**-waarde van 3 retourneert de samengevoegde stand van beide joystickpoorten, zodat een enkele joystick op willekeurige poort gebruikt kan worden.

De stok kan in acht richtingen bewogen worden, welke met de klok mee genummerd worden vanaf de bovenste stand.

	Links	Midden	Rechts
Omhoog	8	1	2
Midden	7	0	3
Omlaag	6	5	4

JOY() retourneert de stokrichting, plus een getal voor elke ingedrukte knop:

Knop	Waarde	Functie
1	128	A (trekker)
2	256	B
3	512	C of Start
4	1024	Start
5	2048	Select

Opmerkingen:

De retourwaarde van **poort** 3, wanneer twee joysticks zijn aangesloten en tegelijkertijd bewogen worden, is niet gedefinieerd.

Het overgrote deel van de Commodore- en Atari-joysticks heeft één vuurknop. De C64GS-controller en Atari CX78-gamepad hebben elk twee actieknoppen. Sommige moderne Commodore-compatibele gamecontrollers en adapters ondersteunen drie knoppen en kunnen de derde knop toewijzen aan een actieknop of een knop met het label "Start".

Het `5plusbuttonsJoystick`-protocol van `crystalct` specificeert een manier voor Commodore-programmatuur om tot vijf knoppen uit te lezen. Knoppen 4 en 5 hergebruiken de signalen van de richtingstok, waardoor de richtingstok mogelijk niet werkt terwijl deze knoppen worden ingedrukt. Een typische vijfknopscontroller wijst deze knoppen toe aan "Start" en "Select".

Het drieknoppenprotocol gebruikt de paddle-signalen van de joystickpoort voor knoppen 2 en 3. **JOY()** kan onverwachte waarden retourneren

wanneer paddles zijn aangesloten. Gebruik **POT()** om paddle-controllers uit te lezen.

De waarden voor elk van de vijf knoppen zijn afzonderlijke bitwaarden. Een programma kan de **AND**-operator gebruiken om elk bit afzonderlijk te testen, los van de richtingstok en de andere knoppen. De expressie `JOY(2) AND 128` is bijvoorbeeld waar wanneer knop 1 van de joystick in poort 2 wordt ingedrukt, ongeacht de stand van de andere knoppen.

Voorbeeld: Gebruik van **JOY**

```
 10 N = JOY(1)
 20 IF N AND 128 THEN PRINT "VUUR!"
 30 REM              N    NO   O    ZO   Z    ZW   W    NW
 40 ON N AND 15 GOSUB 100, 200, 300, 400, 500, 600, 700, 800
 50 GOTO 10
100 PRINT "GA NOORD"     : RETURN
200 PRINT "GA NOORDOOST" : RETURN
300 PRINT "GA OOST"      : RETURN
400 PRINT "GA ZUIDOOST"  : RETURN
500 PRINT "GA ZUID"      : RETURN
600 PRINT "GA ZUIDWEST"  : RETURN
700 PRINT "GA WEST"      : RETURN
800 PRINT "GA NOORDWEST" : RETURN
```

KEY

Token: $F9

Formaat: **KEY**
KEY <**ON** | **OFF**>
KEY <**LOAD** | **SAVE**> bestandsnaam
KEY getal**,** tekenreeks
KEY RESTORE

Gebruik: Beheert de functietoetsmacro's in de BASIC-editor.

Elke functietoets kan een tekenreeks toegewezen krijgen die wordt ingetypt wanneer de toets wordt ingedrukt. De functietoetsen hebben standaardtoewijzingen bij het opstarten, welke met het **KEY**-commando gewijzigd kunnen worden.

KEY : toont de huidige toewijzingen.

KEY ON : schakelt de functietoetsteksten in. De toetsen verzenden de toegewezen tekenreeksen wanneer ze worden ingedrukt.

KEY OFF : schakelt de functietoetsteksten uit. De toetsen verzenden hun tekencode wanneer ze worden ingedrukt.

KEY LOAD bestandsnaam : laadt toetstoewijzingen uit een bestand.

KEY SAVE bestandsnaam : slaat toetstoewijzingen op in een bestand.

KEY getal**,** tekenreeks : wijst de tekenreeks toe aan de toets met het opgegeven nummer.

nummer kan elke waarde binnen dit bereik zijn:

- 1 - 14: Komt overeen met toetsen van **F1** tot en met **F14**
- 15: Komt overeen met HELP
- 16: Komt overeen met SHIFT RUN STOP

KEY RESTORE : herstelt alle functietoetsmacro's naar hun standaardtoewijzingen.

Standaardtoewijzingen:

```
KEY
KEY 1,CHR$(27)+"X"
KEY 2,CHR$(27)+"C"
KEY 3,"DIR"+CHR$(13)
KEY 4,"DIR "+CHR$(34)+"*=PRG"+CHR$(34)+CHR$(13)
KEY 5,"LU"
KEY 6,"KEY6"+CHR$(141)
KEY 7,"L"
KEY 8,"MONITOR"+CHR$(13)
KEY 9,"P"
KEY 10,"KEY10"+CHR$(141)
KEY 11,"U"
KEY 12,"KEY12"+CHR$(141)
KEY 13,CHR$(27)+"O"
KEY 14,"U"+CHR$(27)+"O"
KEY 15,"HELP"+CHR$(13)
KEY 16,"RUN "+CHR$(34)+"*"+CHR$(34)+CHR$(13)
```

Opmerkingen:

De som van de lengtes van alle toegewezen tekenreeksen mag maximaal 240 tekens bedragen. Speciale tekens zoals RETURN of QUOTE worden ingevoerd met hun codes via de **CHR$**-functie. Zie **CHR$** op bladzijde B-48 voor meer informatie.

Voorbeelden:

Gebruik van **KEY**

```
KEY ON                    : REM FUNCTIETOETSEN AAN
KEY OFF                   : REM FUNCTIOTOETSEN UIT
KEY                       : REM TOON TOEWIJZING
KEY 2, "PRINT π" + CHR$(14) : REM WIJS PRINT PI AAN F2 TOE
KEY SAVE "MIJN TOETSSET"  : REM BEWAAR HUIDIGE DEFINITIES IN BESTAND
KEY LOAD "SET ELF"        : REM LAAD DEFINITIES UIT BESTAND
KEY RESTORE
```

LEFT$

Token: $C8

Formaat: **LEFT$**(tekenreeks**,** n**)**

Retourneert:

Een tekenreeks met de eerste **n** tekens van het argument **tekenreeks**.

Indien de lengte van **tekenreeks** gelijk aan of kleiner dan **n** is, zal de resulterende tekenreeks identiek zijn aan het argument.

tekenreeks de tekenreeksexpressie.

n numerieke expressie (0 - 255).

Opmerkingen:

Lege tekenreeksen en lengte nul zijn toegestane waarden.

Voorbeeld: Gebruik van **LEFT$**

```
PRINT LEFT$("MEGA65", 4)
MEGA
```

LEN

Token: $C3

Formaat: **LEN**(tekenreeks)

Retourneert:
De lengte van een tekenreeks.

tekenreeks de tekenreeksexpressie.

Opmerkingen:
BASIC-tekenreeksen kunnen elk teken bevatten, inclusief het nulteken. De lengte van een tekenreeks wordt intern opgeslagen in een tekenreeks-descriptor.

Voorbeeld: Gebruik van **LEN**

```
PRINT LEN("MEGA65" + CHR$(13))
 7
```

LET

Token: $88

Formaat: [**LET**] variabele = expressie

Gebruik: Wijst waarden (of resultaten van expressies) toe aan variabelen.

Opmerkingen:

Het **LET**-statement is overbodig en niet vereist. Toewijzing aan variabelen kan zonder **LET** gedaan worden, maar het is in BASIC 65 behouden gebleven voor achterwaartse compatibiliteit.

Voorbeelden:

Gebruik van **LET**

```
LET A = 5 : REM LANGER EN TRAGER

A = 5     : REM KORTER EN SNELLER
```

LINE

Token: $E5

Formaat: **LINE** x-beg**,** y-beg[**,** x-volgend1**,** y-volgend1 ...]

Gebruik: Bitmapgrafiek: tekent een lijn of een reeks lijnen.

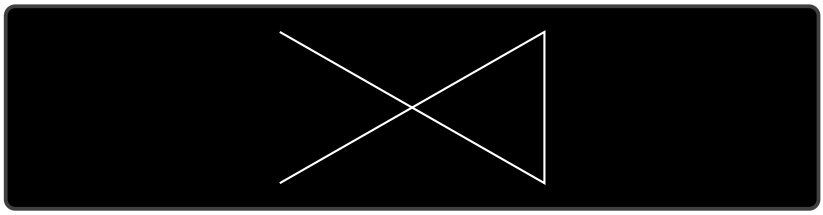
Indien slechts één coördinatenpaar wordt opgegeven, tekent **LINE** een punt.

Indien meer dan één paar wordt gedefinieerd, wordt een lijn getekend op het huidige grafische scherm vanaf de coördinaat (**x-beg**, **y-beg**) naar de volgende coördinatenpaar(en).

Alle momenteel gedefinieerde modes en waarden van de grafische context worden gebruikt.

Voorbeeld: Gebruik van **LINE**

```
10 SCREEN 320, 200, 2                    : REM SCHERM 320 X 200 X 2
20 PEN 1                                 : REM TEKENPEN KLEUR 1 (WIT)
30 LINE 25, 25, 295, 175, 295, 25, 25, 175 : REM TEKEN LIJNEN
40 GETKEY A$                             : REM WACHT OP TOETSDRUK
50 SCREEN CLOSE                          : REM SLUIT SCHERM EN HERSTEL PALET
```



LINE INPUT

Token: $E5 $85

Formaat: **LINE INPUT** [prompt <**,** | **;**>] tekenreeksvariabele [**,** tekenreeksvariabele ...]

Gebruik: Vraagt de gebruiker om invoer vanaf het toetsenbord, waarbij een optionele prompttekenreeks en vraagteken op het scherm worden afgedrukt.

prompt optionele tekenreeksexpressie die als prompt wordt afgedrukt.

Indien het scheidingsteken tussen **prompt** en de eerste **tekenreeksvariabele** een ',' (komma) is, wordt de cursor direct achter de prompt geplaatst. Indien het scheidingsteken een ';' (puntkomma) is, worden een vraagteken en een spatie aan de prompt toegevoegd.

tekenreeksvariabele een of meer tekenreeksvariabelen die elk één regel invoer accepteren.

Opmerkingen:

Dit verschilt van **INPUT** in de wijze waarop de invoer wordt verwerkt. **LINE INPUT** accepteert elk teken dat op een regel wordt ingetypt als een enkele tekenreekswaarde. Alleen de RETURN-toets produceert geen teken.

Indien de variabelenlijst meer dan één variabele bevat, zal **LINE INPUT** de gehele eerste regel gebruiken voor de eerste variabele en het ??-prompt tonen voor elke volgende variabele.

LINE INPUT werkt alleen met tekenreeksvariabelen. Indien een niet-tekenreeksvariabele wordt gebruikt, resulteert **LINE INPUT** in een Type Mismatch-fout nadat de gegevens zijn ingevoerd.

Voorbeeld: Gebruik van **LINE INPUT**

```
10 LINE INPUT "GEEF EEN ZIN IN: ", PH$
20 PRINT "DE ZIN DIE U INVOERDE:"; CHR$(13); "  "; PH$

RUN
VOER EEN ZIN IN: JIJ ZEGT "AARDAPPEL," IK ZEG "AARDAPPEL."
DE ZIN DIE U INVOERDE:
  JIJ ZEI "AARDAPPEL," IK ZEI "AARDAPPEL."
```

LINE INPUT#

Token: $E5 $84

Formaat: **LINE INPUT#** kanaal**,** variabele [**,** variabele ...]

Gebruik: Leest één record per variabele van een invoerapparaat (zoals een schijfstation) en wijst de gelezen gegevens toe aan de variabele. De records moeten afgesloten worden door een **RETURN**-teken, welke niet naar de tekenreeksvariabele gekopieerd wordt. Een lege regel die alleen uit het **RETURN**-teken bestaat, resulteert derhalve in een lege tekenreeks.

kanaal nummer, dat werd opgegeven bij een eerdere aanroep van commando's zoals **DOPEN** of **OPEN**.

variabelenlijst een of meer variabelen die de invoer ontvangen.

Opmerkingen:

In de variabelenlijst kunnen alleen tekenreeksvariabelen of tekenreeksarray-elementen worden gebruikt. In tegenstelling tot andere INPUT-commando's interpreteert of verwijdert **LINE INPUT#** geen aanhalingstekens in de invoer. Deze worden net als alle andere tekens als gegevens geaccepteerd. Records mogen niet langer zijn dan de invoerbuffer, welke 160 tekens bedraagt.

Voorbeeld: Gebruik van **LINE INPUT#**

```
 10 DIM N$(100)
 20 DOPEN#2, "DATA"
 30 FOR I = 0 TO 100
 40 LINE INPUT#2, N$(I)
 50 IF ST = 64 THEN 80           : REM EINDE BESTAND
 60 IF DS THEN PRINT DS$ : GOTO 80 : REM DISKFOUT
 70 NEXT I
 80 DCLOSE#2
110 PRINT "I; " RECORDS GELEZEN"
```

LIST

Token: $9B

Formaat: **LIST** [**P**] [regelbereik]
LIST [**P**] bestandsnaam [**,U** eenheid]

Gebruik: Toont een bereik van regels uit het BASIC-programma in het geheugen.

Bij een enkel regelnummer toont **LIST** die regel.

Bij een bereik van regelnummers toont **LIST** alle regels in dat bereik. Een bereik kan twee getallen zijn gescheiden door een '–' (koppelteken), of het begin of einde van het bereik kan weggelaten worden om het begin of einde van het programma impliciet te maken. Zie de voorbeelden hieronder.

Indien een bestandsnaam wordt opgegeven, wordt het bereik van regels direct uit een BASIC-programmabestand getoond.

Opmerkingen:

De optionele parameter **P** schakelt paginamodus in. Na het tonen van een schermvol regels stopt de lijst en wordt het [MORE]-prompt onderaan het scherm weergegeven. Door **Q** in te drukken wordt de paginamodus afgesloten; elke andere toets gaat verder naar de volgende pagina.

De uitvoer van **LIST** kan naar andere apparaten worden omgeleid via **CMD**.

Een andere manier om een programmalijst uit het geheugen op het scherm weer te geven, is het gebruik van de toetsen F9 en F11, of CTRL P en CTRL V, om een BASIC-lijst op het scherm omhoog of omlaag te scrollen.

Voorbeelden:

Gebruik van **LIST**

```
LIST 100      : REM LIST REGEL 100

LIST 240-350  : REM LIST ALLE REGELS VAN 240 TOT 350

LIST 500-     : REM LIST VAN 500 TOT EIND

LIST -70      : REM LIST VANAF BEGIN TOT 70

LIST "DEMO"   : REM LIST BESTAND "DEMO"

LIST P        : REM LIST PROGRAMMA GEPAGINEERD

LIST P "MURX" : REM LIST BESTAND "MURX" GEPAGINEERD
```

LOAD

Token: $93

Formaat: **LOAD** bestandsnaam [**,** eenheid [**,** vlag]]
LOAD "$[patroon=type]" [**, eenheid**]
LOAD "$$[patroon=type]" [**, eenheid**]
/ bestandsnaam [**,** eenheid [**,** vlag]]

Gebruik: De eerste vorm laadt een bestand van het type **PRG** in het geheugen dat gereserveerd is voor BASIC-programma's.

De tweede vorm laadt een map in het geheugen, welke vervolgens met **LIST** kan worden bekeken. De structuur lijkt op een BASIC-programma, doch in plaats van regelnummers worden bestandsgroottes weergegeven.

De derde vorm lijkt op de tweede, maar de bestanden krijgen opeenvolgende regelnummers. Deze lijstopgave kan als een BASIC-programma worden gescrold met de toetsen **F9** of **F11**, bewerkt, weergegeven, opgeslagen of afgedrukt.

Er kan een filter worden toegepast door een patroon of een patroon en een type op te geven. Het asterisk-teken (*) komt overeen met de rest van de bestandsnaam, terwijl het (?) overeenkomt met willekeurig één teken. De typespecificatie kan één van de tekens (P, S, U, R) zijn, respectievelijk programma-, sequentieel, gebruiker- of relatief bestand.

Een veelgebruikte toepassing van het afkortingsteken **/** is het snel laden van **PRG**-bestanden. Hiervoor:

1. Druk een schijfmap af met **DIR** of **CATALOG**.
2. Verplaats de cursor naar de gewenste regel.
3. Typ '/' in de eerste kolom van de regel en druk op RETURN.

Na het indrukken van RETURN wordt het bestand op de regel met het voorafgaande '/' geladen. Tekens voor en achter de bestandsnaam '"' (dubbele aanhalingstekens) worden genegeerd. Dit geldt uitsluitend voor **PRG**-bestanden.

bestandsnaam is een tekenreeks tussen aanhalingstekens, bijv. `"PROG"`, of een tekenreeksexpressie `("N$)`

Het nummer van de **eenheid** is optioneel. Indien dit niet wordt opgegeven, wordt de standaard schijfeenheid gebruikt.

Indien de **vlag** een ongelijkwaardige waarde heeft, wordt het bestand geladen op het adres dat uit de eerste twee bytes van het bestand

wordt gelezen. Anders wordt het geladen naar het begin van het BASIC-geheugen en wordt het laadadres in het bestand genegeerd.

Opmerkingen:

LOAD laadt bestanden van het type **PRG** in bank 0 van het RAM-geheugen, dat ook voor de broncode van BASIC-programma's wordt gebruikt.

LOAD "*" kan worden gebruikt om het eerste **PRG**-bestand van de opgegeven **eenheid** te laden.

LOAD "$" kan worden gebruikt om de bestandslijst van de opgegeven **eenheid** te laden. Bij gebruik van **LOAD "$"** kan **LIST** worden gebruikt om de lijst op het scherm weer te geven. Hierbij wordt eventueel reeds in het geheugen aanwezig programma overschreven.

LOAD is opgenomen in BASIC 65 voor achterwaartse compatibiliteit met BASIC V2.

Het afkortingsteken **/** kan uitsluitend in de directe modus worden gebruikt.

Standaard gebruikt de C64 **eenheid** 1, welke is toegewezen aan datasette-bandrecorders op de cassettepoort. De MEGA65 gebruikt standaard **eenheid** 8, welke is toegewezen aan het interne schijfstation. Hierdoor is het niet nodig om ,8 toe te voegen aan **LOAD**-commando's die hiervan gebruikmaken.

Voorbeelden:

Gebruik van **LOAD**

```
LOAD "APOCALYPS"    : REM LAAD EEN BESTAND GEHETEN APOCALYPS IN BASICGEHEUGEN

LOAD "MEGA TOOLS",9 : REM LAAD EEN BESTAND "MEGA TOOLS" VANAF UNIT 9 IN BASICGEHEUGEN

LOAD "*",8,1        : LAAD EERSTE BESTAND OP UNIT 8 IN RAM ZOALS IN BESTAND AANGEGEVEN

LOAD "$"            : REM LAAD HELE DIRECTORY - MET BESTANDSGROOTTES

LOAD "$$"           : REM LAAD HELE DIRECTORY - SCROLLBAAR

LOAD "$$X*=P"       : REM DIRECTORY, MET PRG-BESTANDEN BEGINNEND OP 'X'
```

LOADIFF

Token: $FE $43

Formaat: **LOADIFF** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Bitmapgraphics: laadt een IFF-bestand in het grafisch geheugen.

IFF (Interchange File Format) wordt ondersteund door veel verschillende toepassingen en besturingssystemen. **LOADIFF** gaat ervan uit dat de bestanden bitplane-graphics bevatten die overeenkomen met het actieve grafische scherm qua resolutie en kleurdiepte.

Ondersteunde resoluties zijn:

Breedte	Hoogte	Bitplanes	Kleuren	Geheugen
320	200	max. 8	max. 256	max. 64KB
640	200	max. 8	max. 256	max. 128KB
320	400	max. 8	max. 256	max. 128KB
640	400	max. 4	max. 16	max. 128KB

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Er zijn tools beschikbaar om populaire afbeeldingsformaten naar IFF te converteren. Deze tools zijn beschikbaar op diverse besturingssystemen, zoals AMIGA OS, macOS, Linux en Windows. Bijvoorbeeld **ImageMagick** is een gratis grafisch pakket dat een tool genaamd **convert** bevat, welke in combinatie met de **ppmtoilbm**-tool uit het **Netbpm**-pakket kan worden gebruikt om IFF-bestanden aan te maken.

Om `convert` en `ppmtoilbm` te gebruiken voor het converteren van een JPG-bestand naar een IFF-bestand op Linux:

```
convert <myImage.jpg> <myImage.ppm>

ppmtoilbm -aga <myImage.ppm> > <myImage.iff>
```

Voorbeeld: Gebruik van **LOADIFF**

```
100 BANK 128 : SCNCLR
110 REM TOON PLAATJES IN 320 X 200 X 7 RESOLUTIE
120 GRAPHIC CLR : SCREEN DEF 0, 0, 0, 7 : SCREEN OPEN 0 : SCREEN SET 0, 0
130 FOR I = 1 TO 7 : READF$
140 LOADIFF(F$+".IFF") : SLEEP 4 : NEXT
150 DATA ALIEN, BEAKER, JOKER, PICARD, PULP, TROOPER, RIPLEY
160 SCREEN CLOSE 0
170 PALETTE RESTORE
```

LOCK

Token: $FE $50

Formaat: **LOCK** <bestandsnaam|patroon> [**,D** station] [**,U** eenheid]

Gebruik: Vergrendelt een bestand op de schijf, waardoor het niet kan worden bijgewerkt of verwijderd.

Het opgegeven bestand of de set bestanden die aan het patroon voldoen, wordt vergrendeld en kan niet worden verwijderd met de commando's **DELETE**, **ERASE** of **SCRATCH**.

Het commando **UNLOCK** verwijdert de vergrendeling.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

In de directe modus wordt het aantal vergrendelde bestanden weergegeven. Het op-een-na laatste getal in het bericht bevat het aantal vergrendelde bestanden.

Voorbeelden:

Gebruik van **LOCK**

```
LOCK "DRM", U9        : REM VERGRENDEL BESTAND DRM OP UNIT 9
03,FILES LOCKED,01,00

LOCK "BS*"            : REM VERGENDEL BESTANDEN BEGINNEND OP "BS"
03,FILES LOCKED,04,00
```

LOG

Token: $BC

Formaat: **LOG(**numerieke expressie**)**

Retourneert:

De natuurlijke logaritme van een getal.

De natuurlijke logaritme gebruikt het getal van Euler (2.71828183) als grondtal. Op een wetenschappelijke rekenmachine wordt deze functie doorgaans aangeduid als "LN".

Opmerkingen:

Voor het gemak bevat BASIC 65 afzonderlijke functies voor logaritmen met grondtal 10 en grondtal 2: respectievelijk **LOG10()** en **LOG2()**.

Om logaritmen met andere grondtallen te berekenen, gebruik dan de volgende logaritmische identiteit:

$\log_b(n) =$ `LOG(n) / LOG(b)`

Voorbeelden:

Gebruik van **LOG**

```
PRINT LOG(1)
 0

PRINT LOG(0)
?ILLEGAL QUANTITY ERROR

PRINT LOG(4)
 1.3862944

PRINT LOG(100) / LOG(10)
 2
```

LOG10

Token: $CE $08

Formaat: **LOG10**(numerieke expressie)

Retourneert:

De decimale logaritme van de numerieke expressie.

De decimale logaritme gebruikt 10 als grondtal.

Voorbeelden:

Gebruik van **LOG10**

```
PRINT LOG10(1)
 0

PRINT LOG10(0)
?ILLEGAL QUANTITY ERROR

PRINT LOG10(5)
 0.69897001

PRINT LOG10(100); LOG10(10); LOG10(1); LOG10(0.1); LOG10(0.01)
 2  1  0 -1 -2
```

LOG2

Token: $CE $16

Formaat: **LOG2**(numerieke expressie)

Retourneert:

De binaire logaritme van de numerieke expressie.

De binaire logaritme gebruikt 2 als grondtal.

Voorbeelden:

Gebruik van **LOG2**

```
PRINT LOG2(1)
 0

PRINT LOG2(0)
?ILLEGAL QUANTITY ERROR

PRINT LOG2(5)
 2.3219281

PRINT LOG2(4); LOG2(2); LOG2(1); LOG2(0.5); LOG2(0.25)
 2  1  0 -1 -2
```

LOOP

Token: $EC

Formaat: **DO** ... **LOOP**
DO [<**UNTIL** | **WHILE**> logische expressie] ... opdrachten [**EXIT**]
LOOP [<**UNTIL** | **WHILE**> logische expressie]

Gebruik: **DO** ... **LOOP** definieert het bereik van een BASIC-lus. Het gebruik van **DO** ... **LOOP** zonder modifiers creëert een oneindige lus, welke alleen kan worden afgebroken met de **EXIT**-instructie. De lus kan worden beheerd door **UNTIL** of **WHILE** toe te voegen na **DO** of **LOOP**.

Opmerkingen:
DO-lussen mogen genest worden. Een instructie **EXIT** sluit alleen de huidige lus af.

Voorbeelden:
Gebruik van **DO** ... **LOOP**

```
10 PW$ = "" : DO
20 GET A$ : PW$ = PW$ + A$
30 LOOP UNTIL LEN(PW$) > 7 OR A$ = CHR$(13)

10 DO : REM WACHT OP GEBRUIKERSBESLISSING
20 GET A$
30 LOOP UNTIL A$ = "J" OR A$ = "N"

10 DO WHILE ABS(EPS) > 0.001
20 GOSUB 2000               : REM ITERATIESUBROUTINE
30 LOOP

10 I% = 0                   : REM INTEGERLUS 1-100
20 DO : I% = I% + 1
30 LOOP WHILE I% < 101
```

LPEN

Token: $CE $04

Formaat: **LPEN**(coördinaat)

Retourneert:

De status van een lightpen-periferie.

Deze functie vereist een CRT-beeldbuis (of televisie) en een lightpen. Ze werkt niet met een LCD- of LED-scherm. De lightpen moet op poort 1 worden aangesloten.

LPEN(0) retourneert de X-positie van de lightpen, in het bereik 60 - 320.

LPEN(1) retourneert de Y-positie van de lightpen, in het bereik 50 - 250.

Opmerkingen:

De X-resolutie bedraagt twee pixels, waardoor **LPEN(0)** uitsluitend even getallen retourneert.

Er is een felle achtergrondkleur nodig om de lightpen te activeren. De **COLLISION**-instructie kan worden gebruikt om een interrupt-handler in te schakelen.

Voorbeeld: Gebruik van **LPEN**

```
PRINT LPEN(0), LPEN(1) : REM PRINT LICHTPENCOORDINATEN
 100       200
```

MEM

Token: $FE $23

Formaat: **MEM** masker4**,** masker5

Gebruik: Reserveert geheugen in bank 4 of 5, zodat het bitmapgrafieksysteem dit niet zal gebruiken.

masker4 en **masker5** zijn byte-waarden, geïnterpreteerd als een bitveld. Elke bit die op 1 staat, vertegenwoordigt een segment van 8 KB geheugen dat is gereserveerd voor programma-gebruik. Het eerste argument beschrijft bank 4 en het tweede argument beschrijft bank 5.

bit	geheugensegment
0	$0000 - $1FFF
1	$2000 - $3FFF
2	$4000 - $5FFF
3	$6000 - $7FFF
4	$8000 - $9FFF
5	$A000 - $BFFF
6	$C000 - $DFFF
7	$E000 - $FFFF

Opmerkingen:

Na het reserveren van geheugen met **MEM** zal de grafische bibliotheek de gereserveerde gebieden niet gebruiken, zodat deze voor andere doeleinden kunnen worden aangewend. Toegang tot bank 4 en 5 is mogelijk met de commando's **PEEK, WPEEK, POKE, WPOKE** en **EDMA**.

Indien een grafisch scherm niet kan worden geopend omdat het resterende geheugen ontoereikend is, geeft het **SCREEN**-commando een "Out of Memory"-fout.

Enkele commando's in de directe modus, zoals **RENUMBER**, maken gebruik van geheugen in bank 4 en 5 en respecteren de **MEM**-reserveringen niet. **MEM**-reserveringen worden uitsluitend gegarandeerd tijdens de uitvoering van programma's.

Wanneer de 80 × 50-tekstmodus is ingeschakeld, wordt segment 0 van bank 4 automatisch gereserveerd en gebruikt voor schermgegevens. Het gebruikt altijd segment 0, ook indien dit voorheen met **MEM** of een grafisch scherm was gereserveerd. Indien uw programma gebruikmaakt van de 80 × 50-tekstmodus, gebruik segment 0 van bank 4 dan niet voor andere doeleinden.

Om geheugen dat door **MEM** is gereserveerd weer vrij te geven, roep **MEM** opnieuw aan met alle bits op 0. Om geheugen dat door **SCREEN** is gereserveerd vrij te geven, sluit het scherm. Zie **SCREEN CLOSE**. Om

geheugen dat door de 80 × 50-tekstmodus is gereserveerd vrij te geven, schakel over naar een van de andere tekstmodi.

FRE(*4*) en **FRE(*5*)** retourneren bitvelden die het geheugen weergeven dat respectievelijk in bank 4 en 5 is gereserveerd door **MEM**, **SCREEN** en de 80 × 50-tekstmodus tezamen. Zie **FRE()**.

Voorbeeld: Gebruik van **MEM**

```
10 MEM 1, 3              : REM RESERVEER $40000 - $41FFF EN $50000 - $53FFF
20 SCREEN 320, 200       : REM SCREEN ZAL GEEN GERESERVEERDE SEGMENTEN GEBRUIKEN
40 EDMA 3, $2000, 0, $4000 : REM VUL SEGMENT MET NULLEN
```

MERGE

Token: $E6

Formaat: **MERGE** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Laadt een BASIC-programmabestand vanaf de schijf en voegt dit toe aan het programma in het geheugen.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Het laadadres dat in de eerste twee bytes van het bestand is opgeslagen, wordt genegeerd. Het geladen programma vervangt niet het programma in het geheugen (wat **DLOAD** wel doet), maar wordt eraan toegevoegd. Na het laden wordt het programma opnieuw gelinkt en is het gereed om te worden uitgevoerd of bewerkt.

Het is de verantwoordelijkheid van de gebruiker om te waarborgen dat er geen conflicten met regelnummers bestaan tussen het programma in het geheugen en het samengevoegde programma. Het eerste regelnummer van het samengevoegde programma moet groter zijn dan het laatste regelnummer van het programma in het geheugen.

MERGE kan uitsluitend in de directe modus worden gebruikt.

Voorbeeld: Gebruik van **MERGE**

```
DLOAD "HOOFDPROGRAMMA"
MERGE "BIBLIOTHEEK"
```

MID$

Token: $CA

Formaat: **MID$**(tekenreeks**,** index**,** n)
MID$(tekenreeksvariabele**,** index**,** n**)** = tekenreeksexpressie

Gebruik: Als functie: geeft een deeltekenreeks van een string terug. Als instructie: vervangt een deeltekenreeks van een tekenreeksvariabele door een andere string.

tekenreeks de tekenreeksexpressie.

index de startindex (1 - 255).

n de lengte van de deeltekenreeks (0 - 255).

Opmerkingen:
Lege tekenreeksen en lengtes van nul zijn toegestane waarden.

Voorbeeld: Gebruik van **MID$**

```
10 A$ = "MEGA65"
20 PRINT MID$(A$, 3, 4)
30 MID$(A$, 4, 1) = "0"
40 PRINT A$

RUN
GA65
MEG065
```

MKDIR

Token: $FE $51

Formaat: **MKDIR** mapnaam **,L** grootte [**,U** eenheid]

Gebruik: Maakt een submap aan op een diskette of D81-schijfimage.

mapnaam de naam van een map. Ofwel een aangehaalde tekenreeks zoals "EENOFANDEREMAP", of een tekenreeksexpressie in haakjes zoals: (DR$)

MKDIR kan alleen worden gebruikt op stations die door CBDOS worden beheerd. Dit betreft het interne diskettestation en SD-kaartafbeeldingen van het type D81. Het commando kan niet worden gebruikt op externe stations die op de seriële IEC-bus zijn aangesloten.

De parameter **grootte** geeft het aantal sporen op dat voor de submap worden gereserveerd, waarbij één spoor bestaat uit 40 sectoren van 256 bytes. Het eerste spoor van het gereserveerde bereik wordt als mappenspoor voor de submap gebruikt.

De minimale grootte is 3 sporen, de maximale 38 sporen. Er moet een aaneengesloten gebied van lege sporen op de diskette (of D81-afbeelding) aanwezig zijn dat groot genoeg is voor het aanmaken van de submap. Indien een dergelijk gebied niet beschikbaar is, wordt een Disk Full-fout gemeld.

Er kunnen meerdere submappen worden aangemaakt, mits er voldoende lege sporen beschikbaar zijn.

Na het succesvol aanmaken van de submap vindt er automatisch een **CHDIR** naar deze submap plaats.

CHDIR "/" wisselt terug naar de hoofdmap.

Voorbeeld: Gebruik van **MKDIR**

```
MKDIR "SUBDIR", L5 : REM MAAK SUBDIRECTORY MET 5 TRACKS
DIR
0 "SUBDIR          "   1D
160 BLOCKS FREE.
```

MOD

Token: $CE $0B

Formaat: **MOD**(teller**,** deler)

Retourneert:
De rest van een delingsbewerking.

Opmerkingen:
In andere programmeertalen zoals C is deze functie geïmplementeerd als operator (%). In BASIC is zij als functie geïmplementeerd.

Voorbeeld: Gebruik van **MOD**

```
FOR I = 0 TO 8 : PRINT MOD(I, 4); : NEXT I
 0  1  2  3  0  1  2  3  0
```

MONITOR

Token: $FA

Formaat: **MONITOR**

Gebruik: Roept de KERNAL-machinetaalmonitor aan.

Opmerkingen:

Het gebruik van de **MONITOR** vereist kennis van de CSG4510 / 6502 / 6510-processors, de assembleertaal die zij gebruiken, en hun architecturen. Meer informatie over de **MONITOR** is beschikbaar in Hoofdstuk/-Bijlage O op bladzijde O-3.

Om de monitor te verlaten, druk op **X**.

Hulptekst kan worden weergegeven met **?** of **H**.

Voorbeeld: Gebruik van **MONITOR**

```
MONITOR
```

```
BS MONITOR COMMANDS:ABCDEFGHJMR!UXG.,);?$+&%'LSV
    PC   SR AC XR YR ZR BP  SP  NVEBDIZC
; 00FFA2 00 00 00 00 00 00 01F8 --------
ASSEMBLE     - A ADDRESS MNEMONIC OPERAND
BIMAPS       - B [FROM]
COMPARE      - C FROM TO WITH
DISASSEMBLE  - D [FROM [TO]]
FILL         - F FROM TO FILLBYTE
GO           - G [ADDRESS]
HUNT         - H FROM TO (STRING OR BYTES)
JSR          - J ADDRESS
LOAD         - L FILENAME [UNIT [ADDRESS]]
MEMORY       - M [FROM [TO]]
REGISTERS    - R
SAVE         - S FILENAME UNIT FROM TO
TRANSFER     - T FROM TO TARGET
VERIFY       - V FILENAME [UNIT [ADDRESS]]
EXIT         - X
.<DOT>       - . ADDRESS MNEMONIC OPERAND
)<GREATER>   - ) ADDRESS BYTE SEQUENCE
;<SEMICOLON> - ; REGISTER CONTENTS
@DOS         - @ [DOS COMMAND]
?HELP        - ?
```

MOUNT

Token: $FE $49

Formaat: **MOUNT** bestandsnaam [**,U** eenheid]
MOUNT [**U** eenheid]
MOUNT OFF [U eenheid]

Gebruik: Mount een schijfimagebestand van het type D81 of D64 vanaf de SD-kaart op station 8 (standaard) of station 9 (**U9**).

MOUNT zonder argumenten wijst het interne fysieke diskettestation toe aan station 9 en heft de toewijzing van station 9 op. Indien **eenheid** wordt opgegeven, wordt alleen dat station gewijzigd.

MOUNT OFF heft de toewijzing van zowel station 8 als station 9 op. Dit is equivalent aan de "no disk"-instelling in het Freezer-menu. Indien **eenheid** wordt opgegeven, wordt alleen dat station gewijzigd.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals `"DATA"`, of een tekenreeksexpressie in haakjes zoals: `(FI$)`

eenheid is ofwel 8 voor het eerste virtuele station, of 9 voor het tweede virtuele station.

Opmerkingen:

MOUNT kan zowel in de directe modus als in een programma worden gebruikt.

Station 8 en station 9 verwijzen respectievelijk naar de virtuele stations 0 en 1. U kunt de stationnummers voor deze stations wijzigen met het **SET DISK**-commando. Het **MOUNT**-commando verwacht altijd **U8** of **U9** als argument om naar deze stations te verwijzen, zelfs indien de werkelijke stationnummers door **SET DISK** of in het Freezer-menu zijn gewijzigd.

MOUNT zonder argumenten (of met het argument **U9**) wijst station 9 eigenlijk toe aan een hypothetische Commodore 1565 externe diskettestation. Een standaard MEGA65 heeft geen 1565-poort en dergelijke stations zijn er nog niet, waardoor station 9 effectief niet wordt gemount.

OPMERKING: Het **MOUNT**-commando onthoudt de meest recente schijfimage-bestandsnaam voor elk station en probeert de schijfimages opnieuw te mounten na het indrukken van de resetknop. **MOUNT** zonder bestandsnaam, of **MOUNT OFF**, doet de meest recente schijfimage-bestandsnamen vergeten. Schijfimages die via de Freezer zijn gemount, worden niet op deze manier onthouden. Deze functionaliteit wordt herzien en kan zich in een latere versie anders gedragen.

Voorbeelden:

Gebruik van **MOUNT**

```
MOUNT "APOCALYPS.D81"  : REM MOUNT IMAGE OP UNIT 8

MOUNT "BASIC.D81", U9  : REM MOUNT IMAGE OP UNIT 9

MOUNT (FI$), U(UN%)    : REM MOUNT MET VARIABELE ARGUMENTEN

MOUNT                  : REM INTERNE DRIVE OP UNIT 8, U9 NIET TOEGEWEZEN
MOUNT U8               : REM INTERNE DRIVE OP UNIT 8, U9 ONVERANDERD

MOUNT OFF              : REM BEIDE SCHIJVEN OP "GEEN DISK"
MOUNT OFF U9           : REM ENKEL UNIT 9 OP "GEEN DISK"
```

MOUSE

Token: $FE $3E

Formaat: **MOUSE ON** [{**,** poort**,** sprite**,** activeringspunt**,** pos}]
MOUSE OFF

Gebruik: Schakelt de muisdriver in en verbindt de muis op de opgegeven poort met de muispijltjessprite.

poort muispoort 1 of 2 (standaard 2), of "3" zodat een muis in beide poorten het sprite kan verplaatsen.

sprite nummer van de muispijltjessprite (standaard 0).

activeringspunt locatie van het "activeringspunt" dat de positie en het klikdoelwit bepaalt (x, y) (standaard 0, 0).

positie beginspositie van de muis (x, y). Indien niet opgegeven, wordt de laatst bekende positie van het sprite gebruikt.

MOUSE OFF schakelt de muisdriver uit en verbergt het bijbehorende sprite.

Opmerkingen:

Het "activeringspunt" van de muis bepaalt welk deel van de muisafbeelding als klikdoelwit wordt beschouwd, zoals de bovenkant van een pijl of het midden van een richtkruis. Het activeringspunt blijft altijd binnen de schermrand. Het standaard **activeringspunt** is (0, 0), wat de linkerbovenhoek van het sprite vertegenwoordigt.

Bij het opstarten van het systeem wordt sprite 0 geïnitialiseerd als een afbeelding van een muispijltje, met het activeringspunt op (0, 0).

Gebruik **RMOUSE** om de locatie en de knopstatus van de muis te testen. Dit retourneert de coördinaten van de linkerbovenhoek van het sprite, niet de coördinaten van het activeringspunt. Om de coördinaten van het activeringspunt te verkrijgen, telt u de locatie van het activeringspunt op bij de sprite-coördinaten.

positie kan een absolute coördinaat zijn, of een relatieve coördinaat ten opzichte van de huidige muispositie, vergelijkbaar met **MOVSPR**.

U kunt een van de hoge-resolutie-sprite-modi van de MEGA65 gebruiken voor de muispijltjessprite. De modi moeten worden ingesteld voordat de muis met **MOUSE** wordt geactiveerd.

Zie ook **SET FAST MOUSE**.

Voorbeelden:

Gebruik van **MOUSE**

```
REM LAAD DATA IN SPRITE #0 ALVORENS HET TE GEBRUIKEN
MOUSE ON, 1          : REM MUIS AAN MET SPRITE #0
MOUSE OFF            : REM MUIS UIT

MOUSE ON, 1, 0, 2, 4  : REM ZET DE HOTSPOT TOP (2,4)
RMOUSE X, Y, B        : REM HAAL MUISSPRITE-COORDINATEN OP
X = X + 2 : Y = Y + 4 : REM BEREKEN COORDINATEN VAN DE HOTSPOT

REM ZET BEGINPOSITIE OP (300,75)
MOUSE ON, 1, 0, 0, 0, 300,75

MOUSE ON, 3    : REM MUIS VERBONDEN MET EENDER WELKE POORT BEWEEGT PIJLTJE

SETBIT $D076,0 : REM VERTICALE HIRES VOOR SPRITE 0 AAN
SETBIT $D054,4 : REM HORIZONTAL HIRES VOOR ALLE SPRITES AAN
MOUSE ON, 1    : REM MUISPIJLTJE IS HIRES OP BEIDE ASSEN
```

MOVSPR

Token: $FE $06

Formaat: **MOVSPR** nummer**,** positie
MOVSPR nummer**,** begin-positie **TO** eind-positie**,** snelheid

Gebruik: Verplaatst een sprite naar een positie op het scherm.

Elke **positie**-parameter bestaat uit twee 16-bits waarden, welke een absolute coördinaat, een relatieve coördinaat, een hoek of een snelheid specificeren. Het waardetype wordt bepaald door een prefix:

- **+waarde** relatieve coördinaat: positieve offset.
- **-waarde** relatieve coördinaat: negatieve offset.
- **#waarde** snelheid.

Indien geen prefix wordt opgegeven, wordt de absolute coördinaat of hoek gebruikt.

De positie-parameter kan derhalve worden gebruikt om:

- het sprite op een absolute positie op het scherm te plaatsen.
- een verplaatsing ten opzichte van de huidige positie op te geven.
- een relatieve beweging vanaf een opgegeven positie te activeren.
- beweging met een hoek en snelheid vanaf de huidige positie te beschrijven.

MOVSPR nummer, positie wordt gebruikt om de sprite onmiddellijk op de positie te plaatsen of, in het geval van een hoek#snelheid-argument, de verdere beweging te beschrijven.

In de tweede vorm plaatst het sprite op de beginpositie, definieert de bestemmingspositie en de snelheid van de beweging.

Het sprite wordt op de beginpositie geplaatst en beweegt in een rechte lijn naar de bestemming op de opgegeven snelheid. De coördinaten moeten absoluut of relatief zijn. De beweging wordt beheerd door de BASIC-interrupt handler en vindt parallel aan de programma-uitvoering plaats.

nummer sprite-nummer (0 - 7).

positie coördinaten: x,y | xrel,y | x,yrel | xrel,yrel | hoek#snelheid.

x absolute schermcoördinaat in pixels.

y absolute schermcoördinaat in pixels.

xrel relatieve schermcoördinaat in pixels.

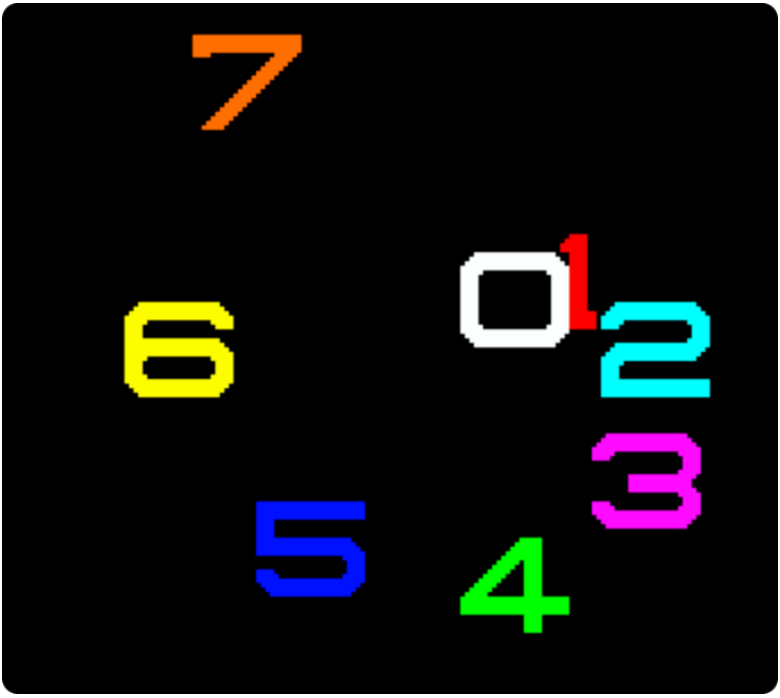
yrel relatieve schermcoördinaat in pixels.

hoek kompasrichting voor de spritebeweging [graden]. 0: omhoog, 90: rechts, 180: omlaag, 270: links, 45: rechtsboven, enzovoort.

snelheid snelheid van de beweging, geconfigureerd als een zwevend getal in het bereik 0.0 – 127.0, in pixels per frame. PAL heeft 50 frames per seconde, NTSC heeft 60 frames per seconde. Een snelheidswaarde van 1.0 verplaatst het sprite met 50 pixels per seconde in de PAL-modus.

Voorbeeld: Gebruik van **MOVSPR**

```
100 CLR : SCNCLR : SPRITECLR
110 BLOAD "DEMOSPRITES1", B0, P1536
130 FOR I = 0 TO 7 : C = I + 1 : SP = 0.07 * (I + 1)
140 MOV SPR I, 160, 120
145 MOV SPR I, 45*I#SP
150 SPRITE I, 1, C,, 0, 0
160 NEXT
170 SLEEP 3
180 FOR I = 0 TO 7 : MOVSPR I, 0#0 : NEXT
```



NEW

Token: $A2

Formaat: **NEW**
NEW RESTORE

Gebruik: Wist het BASIC-programma in het geheugen en reset alle BASIC-parameters naar hun standaardwaarden.

Omdat **NEW** parameters en pointers reset, maar het adresbereik van een BASIC-programma dat in het geheugen aanwezig was niet overschrijft, is het mogelijk het programma te herstellen. Indien er na **NEW** geen **LOAD**-operaties of bewerkingen hebben plaatsgevonden, kan het programma worden hersteld met **NEW RESTORE**.

Voorbeelden:

Gebruik van **NEW**

```
NEW         : REM RESET BASIC EN WIS PROGRAMMA

NEW RESTORE : REM PROBEERD GE'NEW'D PROGRAMMA TE HERSTELLEN
```

NEXT

Token: $82

Formaat: **FOR** index = begin **TO** einde [**STEP** stap] ... **NEXT** [index]

Gebruik: Markeert het einde van de lus die bij de opgegeven indexvariabele hoort. Een lus die met **FOR** wordt gedefinieerd, moet eindigen met **NEXT**.

De **index**-variabele kan bij elke iteratie met een constante waarde **stap** worden verhoogd of verlaagd. Standaard wordt de variabele met 1 verhoogd. De indexvariabele moet een reële variabele zijn.

begin de waarde waarmee de index wordt geïnitialiseerd.

einde wordt aan het einde van een iteratie gecontroleerd en bepaalt of nog een iteratie wordt uitgevoerd of dat de lus wordt afgesloten.

stap definieert de verandering die aan het einde van elke iteratie op de indexvariabele wordt toegepast. Positieve stapwaarden verhogen de variabele, negatieve waarden verlagen deze. Standaard is 1,0 indien niet opgegeven.

Opmerkingen:

De **index**-variabele na **NEXT** is optioneel. Indien deze wordt weggelaten, wordt de variabele van de huidige lus aangenomen. Meerdere opeenvolgende **NEXT**-instructies kunnen worden gecombineerd door de indexen als een door komma's gescheiden lijst op te geven. De instructies "NEXT I : NEXT J : NEXT K" en "NEXT I, J, K" zijn equivalent.

Voorbeelden:

Gebruik van **NEXT**

```
10 FOR D = 0 TO 360 STEP 30
20 R = D * π / 180
30 PRINT D;R;SIN(R);COS(R);TAN(R)
40 NEXT D

10 DIM M(20, 20)
20 FOR I = 0 TO 20
30 FOR J = I TO 20
40 M(I, J) = I + 100 * J
50 NEXT J, I
```

NOT

Token: $A8

Formaat: **NOT** operand

Gebruik: Voert een bitgewijze logische NOT-bewerking uit op een 16-bits waarde.

Gehele operanden worden als zodanig gebruikt, terwijl reële operanden worden omgezet naar een geheel getal van 16 bit in het tweecomplementformaat (waarbij precisie verloren gaat).

Logische operanden worden omgezet naar een geheel getal van 16 bit, waarbij $FFFF (decimaal -1) voor TRUE en $0000 (decimaal 0) voor FALSE wordt gebruikt.

Expressie	Resultaat
NOT 0	-1
NOT -1	0

Opmerkingen:

Het resultaat is van het type geheel getal.

Voorbeelden:

Gebruik van **NOT**

```
PRINT NOT 3
-4

PRINT NOT -65
64

C = 257
OK = C ( 256 AND C )= 0
IF (NOT OK) THEN PRINT "GEEN BYTEWAARDE"
```

OFF

Token: $FE $24

Formaat: trefwoord **OFF**

Gebruik: **OFF** is een secundair trefwoord dat in combinatie met primaire trefwoorden wordt gebruikt, zoals **KEY** en **MOUSE**.

Opmerkingen:

OFF kan niet op zichzelf worden gebruikt.

Voorbeelden:

Gebruik van **OFF**

```
KEY OFF   : REM FUNCTIESNELTOETSEN UIT

MOUSE OFF : REM MUISPIJLTJE UIT
```

ON

Token: $91

Formaat: **ON** expressie **GOSUB** regelnummer [**,** regelnummer ...]
ON expressie **GOTO** regelnummer [**,** regelnummer ...]
trefwoord **ON**

Gebruik: Voert een **GOSUB** of **GOTO** uit naar een regelnummer dat door een getalsexpressie wordt geselecteerd.

Afhankelijk van het resultaat van de expressie wordt het doel voor **GOSUB** en **GOTO** gekozen uit de tabel met regelnummers aan het einde van de instructie.

Wanneer **ON** als secundair trefwoord wordt gebruikt, wordt het gecombineerd met primaire trefwoorden zoals **KEY** en **MOUSE**.

expressie positieve numerieke waarde. Reële getallen worden omgezet naar een geheel getal (met verlies van nauwkeurigheid) in tweecomplementformaat. Logische operanden worden omgezet naar 16-bits gehele getallen, waarbij $FFFF (decimaal -1) voor WAAR en $0000 (decimaal 0) voor ONWAAR wordt gebruikt.

Opmerkingen:

Negatieve waarden voor **expressie** onderbreken het programma met een Illegal Quantity-fout.

De **regelnummerlijst** geeft de doelen op voor waarden van 1, 2, 3 enzovoort.

Een expressieresultaat van nul, of een resultaat dat groter is dan het aantal doelregels, heeft geen effect en het programma gaat verder met de volgende instructie.

Voorbeeld: Gebruik van **ON**

```
 20 KEY   ON : REM FUNCTIESNELTOETSEN AAN
 30 MOUSE ON : REM MUISDRIVER AAN
 40 N = JOY(1) : IF N AND 128 THEN PRINT "VUUR! ";
 60 REM               N    NO   O    ZO   Z    ZW   W    NW
 70 ON N AND 15 GOSUB 100, 200, 300, 400, 500, 600, 700, 800
 80 GOTO 40
100 PRINT "GA NOORD"     : RETURN
200 PRINT "GA NOORDOOST" : RETURN
300 PRINT "GA OOST"      : RETURN
400 PRINT "GA ZUIDOOST"  : RETURN
500 PRINT "GA ZUID"      : RETURN
600 PRINT "GA ZUIDWEST"  : RETURN
700 PRINT "GA WEST"      : RETURN
800 PRINT "GA NOORDWEST" : RETURN
```

OPEN

Token: $9F

Formaat: **OPEN** kanaal**,** eerste adres [**,** secundair adres [**,** bestandsnaam]]

Gebruik: Opent een invoer/uitvoerkanaal voor een apparaat.

kanaal getal, waarbij:

- 1 <= kanaal <= 127: regelafsluiting is CR.
- 128 <= kanaal <= 255: regelafsluiting is CR LF.

eerste adres apparaatnummer. Voor IEC-apparaten is het eenheidsnummer het primaire adres. De volgende primaire adreswaarden zijn mogelijk:

Eenheid	Apparaat
0	Toetsenbord
1	Standaard van het systeem
2	RS232-seriële verbinding
3	Scherm
4 - 7	IEC-printer en plotter
8 - 31	IEC-schijfstations

Het **secundaire adres** heeft enkele gereserveerde waarden voor IEC-schijfeenheden: 0: laden, 1: opslaan, 15: commandokanaal. De waarden 2 - 14 kunnen voor schijfbestanden worden gebruikt.

bestandsnaam is een aangehaalde tekenreeks, bijv. `"DATA"` of een tekenreeksexpressie. De syntaxis verschilt van **DOPEN#**, omdat de **bestandsnaam** voor **OPEN** alle bestandsattributen bevat, bijvoorbeeld: `"0:DATA,S,W"`

Opmerkingen:

Voor IEC-schijfeenheden wordt het gebruik van **DOPEN#** aanbevolen.

Indien het eerste teken van de bestandsnaam een "@" (apestaartje) is, wordt dit geïnterpreteerd als een "opslaan en vervangen"-operatie. Het gebruik van deze optie op 1541- en 1571-stations wordt niet aanbevolen, omdat deze een "opslaan en vervangen-fout" in hun DOS bevatten.

Voorbeeld: Gebruik van **OPEN**

```
OPEN 4, 4                    : REM OPEN PRINTER
CMD 4                        : REM LEID STANDAARDUITVOER NAAR 4 OM
LIST                         : REM PRINT LISTING OP PRINTERDEVICE 4
OPEN 3, 8, 3, "0:USRBESTAND,U"
OPEN 2, 9, 2, "0:DATA,S,W"
```

OR

Token: $B0

Formaat: operand **OR** operand

Gebruik: Voert een bitgewijze logische OR-bewerking uit op twee 16-bits waarden.

Gehele operanden worden ongewijzigd gebruikt. Reële operanden worden omgezet naar een geheel getal van 16 bit (met verlies van nauwkeurigheid) in tweecomplementformaat. Logische operanden worden omgezet naar 16-bits gehele getallen, waarbij $FFFF (decimaal -1) voor WAAR en $0000 (decimaal 0) voor ONWAAR wordt gebruikt.

Expressie	Resultaat
0 OR 0	0
0 OR 1	1
1 OR 0	1
1 OR 1	1

Opmerkingen:

Het resultaat is van het type geheel getal. Indien het resultaat in een logische context wordt gebruikt, wordt de waarde 0 als ONWAAR beschouwd en alle andere niet-nul-waarden als WAAR.

Voorbeelden:

Gebruik van **OR**

```
PRINT 1 OR 3
 3

PRINT 128 OR 64
 192

IF (C<0 OR C>255) THEN PRINT "GEEN BYTEWAARDE"
```

PAINT

Token: $DF

Formaat: **PAINT** x, y, modus [, gebiedsrandkleur]

Gebruik: Bitmapgrafiek: voert een vulbewerking uit op een afgesloten grafisch gebied met de huidige tekenkleur.

x, y coördinaatpaar, dat zich binnen het te vullen gebied moet bevinden.

modus specificeert de tekenmodus:

- 0: De kleur van pixel (x,y) bepaalt de kleur die door de tekenkleur wordt vervangen.
- 1: De **gebiedsrandkleur** bepaalt het gebied dat met de tekenkleur wordt gevuld.
- 2: Vult het gebied dat verbonden is met pixel (x, y).

randkleur van het gebied bepaalt de kleurindex voor modus 1.

Voorbeeld: Gebruik van **PAINT**

```
10 SCREEN 320, 200, 2      : REM OPEN SCHERM
20 PALETTE 0, 1, 10, 15, 10 : REM KLEUR 1 OP LICHTGROEN
30 PEN 1                   : REM ZET TEKENPEN (PEN 0) OP LICHTGROEN (1)
40 LINE 160, 0, 240, 100   : REM 1E REGEL
50 LINE 240, 100, 80, 100  : REM 2E REGEL
60 LINE 80, 100, 160, 0    : REM 3E REGEL
70 PAINT 160, 10           : REM VUL DRIEHOEK MET PENKLEUR
80 GETKEY A$               : REM WACHT OP TOETSDRUK
90 SCREEN CLOSE            : REM GRAFISCH SCHERM UIT
```

PALETTE

Token: $FE $34

Formaat: **PALETTE** scherm, kleur, rood, groen, blauw
PALETTE COLOR kleur, rood, groen, blauw
PALETTE RESTORE

Gebruik: **PALETTE** kan worden gebruikt om een item van het systeemkleurenpalet of het palet van een scherm te wijzigen.

PALETTE RESTORE herstelt het systeempalet naar de standaardwaarden.

scherm schermnummer (0 - 3).

COLOR trefwoord voor het wijzigen van het systeempalet.

kleur index naar paletitem (0 - 255). PALETTE kan kleuren definiëren buiten de standaard systeempaletitems 0 - 31.

rood intensiteit (0 - 15).

groen intensiteit (0 - 15).

blauw intensiteit (0 - 15).

Voorbeelden:

Gebruik van **PALETTE**

```
10 REM VERANDER SYSTEEMKLEURINDEX
20 REM --- INDEX 9 (BRUIN) TO (DONKERBLAUW)
30 PALETTE COLOR 9, 0, 0, 7
```

```
 10 GRAPHIC CLR           : REM INITIALISEER
 20 SCREEN DEF 1, 0, 0, 2 : REM 320 X 200
 30 SCREEN OPEN 1         : REM OPEN
 40 SCREEN SET 1, 1       : REM MAAK SCHERM ACTIEF
 50 SCREEN CLR 0          : REM MAAK SCHERM ZWART
 60 PALETTE 1, 0,  0, 0, 0 : REM 0 = ZWART
 70 PALETTE 1, 1, 15, 0, 0 : REM 1 = ROOD
 80 PALETTE 1, 2,  0, 0,15 : REM 2 = BLAUW
 90 PALETTE 1, 3,  0,15, 0 : REM 3 = GROEN
100 PEN 2                 : REM ZET TEKENPEN (PEN 0) OP BLAUW (2)
110 LINE 160, 0, 240, 100  : REM 1E REGEL
120 LINE 240, 100, 80, 100 : REM 2E REGEL
130 LINE 80, 100, 160, 0   : REM 3E REGEL
140 PAINT 160, 10          : REM VUL DRIEHOEK MET HUIDIGE PENKLEUR
150 GETKEY K$              : REM WACHT OP TOETSDRUL
160 SCREEN CLOSE 1         : REM GRAFISCH SCHERM UIT
```

PASTE

Token: $E3

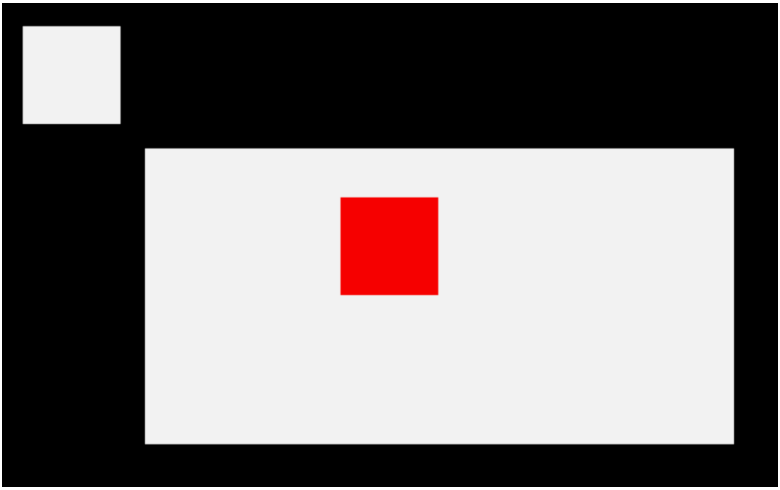
Formaat: **PASTE** x, y

Gebruik: Bitmapgrafiek: plakt de inhoud van de **CUT**- of **GCOPY**-buffer op het scherm. De argumenten **x, y** geven de linkerbovenhoek van de plakpositie op het scherm aan, als pixelcoördinaten.

Opmerkingen:
De grootte van de rechthoekje wordt beperkt door de 1KB-grootte van de buffer. Het geheugenvereiste voor een gebied is breedte * hoogte * aantal bitplanes / 8. Dit mag niet 1024 bytes of meer bedragen. Bij een scherm met 4 bitplanes heeft een gebied van 50 x 50 bijvoorbeeld 1.250 bytes nodig.

Voorbeeld: Gebruik van **PASTE**

```
10 SCREEN 320, 200, 2
20 BOX 60, 60, 300, 180, 1 : REM TEKEN EEN WIT VAK
30 PEN 2                   : REM KIES RODE PEN
40 CUT 140, 80, 40, 40     : REM SNIJ EEN GEBIED VAN 40 * 40 UIT
50 PASTE 10,10             : REM PLAK HET OP NIEUWE POSITIE
60 GETKEY A$               : REM WACHT OP TOETSDRUK
70 SCREEN CLOSE
```



PEEK

Token: $C2

Formaat: **PEEK**(adres)

Retourneert:

De byte-waarde die in het geheugen op het opgegeven **adres** is opgeslagen, als een natuurlijk getal van 8 bit.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

Opmerkingen:

Banken 0 - 127 geven toegang tot RAM- of ROM-banken. Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enzovoort.

Voorbeeld: Gebruik van **PEEK**

```
10 BANK 128                          : REM KIES SYSTEEMBANK
20 L = PEEK($02F8)                   : REM DOELADRES USR-JUMP LAAG
30 H = PEEK($02F9)                   : REM DOELADRES USR-JUMP HOOG
40 T = L + 256 * H                   : REM 16-BIT JUMPADRES
50 PRINT "ADRES USR-FUNCTIEAANROEP";T
```

PEN

Token: $FE $33

Formaat: **PEN** kleur

Gebruik: Bitmapgrafiek: stelt de kleur (paletitem) van de grafische pen in voor het huidige scherm.

kleur paletindex, uit het palet van het huidige scherm.

Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard systeempalet.

Opmerkingen:

De ontwerpdocumentatie van de C65 beschrijft een versie van **PEN** met twee argumenten, waarmee het paletitem voor een van de twee pennen wordt ingesteld: de tekenpenkleur (pen 0) en een secundaire kleur die door de **jam**-tekenmodus wordt gebruikt (pen 1). Deze functie van **jam** lijkt niet te zijn geïmplementeerd. Zie **DMODE**.

Voorbeeld: Gebruik van **PEN**

```
 10 GRAPHIC CLR             : REM INITIALISEER
 20 SCREEN DEF 1, 0, 0, 2   : REM 320 X 200
 30 SCREEN OPEN 1           : REM OPEN
 40 SCREEN SET 1, 1         : REM MAAK SCHERM ACTIEF
 50 PALETTE 1, 0,  0,  0,  0 : REM 0 = ZWART
 60 PALETTE 1, 1, 15,  0,  0 : REM 1 = ROOD
 70 PALETTE 1, 2,  0,  0, 15 : REM 2 = BLAUW
 80 PALETTE 1, 3,  0, 15,  0 : REM 3 = GROEN
 90 PEN 1                   : REM ZET TEKENPEN (PEN 0) OP ROOD (1)
100 LINE 160, 0, 240, 100   : REM TEKEN RODE LIJN
110 PEN 2                   : REM ZET TEKENPEN (PEN 0) OP BLAUW (2)
120 LINE 240, 100, 80, 100  : REM TEKEN BLAUWE LIJN
130 PEN 3                   : REM ZET TEKENPEN (PEN 0) OP GROEN (3)
140 LINE 80, 100, 160, 0    : REM TEKEN GROENE LIJN
150 GETKEY K$               : REM WACHT OP TOETSDRUK
160 SCREEN CLOSE 1          : REM GRAFISCH SCHERM UIT
```

PIXEL

Token: $CE $0C

Formaat: **PIXEL**(x, y)

Gebruik: Bitmapgrafiek: de kleur van een pixel op de opgegeven positie.

x absoluut schermcoördinaat.

y absoluut schermcoördinaat.

Retourneert:
De kleur op positie (x, y). **PIXEL** geeft een Scherm niet geopend-fout terug indien er geen scherm is geopend.

Voorbeeld: Gebruik van **PIXEL**

```
10 SCREEN 320, 200, 4          : REM BOUW SCHERM OP MET 10.000 WILLEKEURIGE PIXELS
20 FOR I = 1 TO 10000 : DOT RND(1) * 320, RND(1) * 200, RND(1) * 16 : NEXT
30 MOUSE ON, 1, 0
40 RMOUSE X, Y, B              : REM LEES MUIS X, Y
50 MOVSPR 0, X, Y              : REM BEWEEG PIJLTJE
60 BORDER PIXEL(X, Y)          : REM ZET RANDKLEUR OP KLEUR OP POSITIE
70 GET A$ : IF A$ = "" THEN 40 : REM EEN TOETSDRUK BEEINDIGT PROGRAMMA
80 BORDER 6 : MOUSE OFF
90 SCREEN CLOSE
```

PLAY

Token: $FE $04

Formaat: **PLAY** [{tekenreeks1, tekenreeks2, tekenreeks3, tekenreeks4, tekenreeks5, tekenreeks6}]

Gebruik: Start het afspelen van een reeks muzieknoten of stopt een momenteel spelende reeks.

PLAY zonder argumenten doet alle stems zwijgen en reset alle variabelen van het muzieksysteem (zoals **TEMPO**).

PLAY aanvaardt maximaal zes kommagescheiden tekenreeksen, waarbij elke tekenreeks de reeks noten en directieven beschrijft die op een specifieke stem van de twee beschikbare SID-chips worden afgespeeld, waardoor tot 6-kanaals polyfonie mogelijk is.

PLAY gebruikt SID1 (voor stems 1 tot 3) en SID3 (voor stems 4 tot 6) van de 4 SID-chips van het systeem. Standaard zijn SID1 en SID2 enigszins rechtsgeoriënteerd en SID3 en SID4 licht linksgeoriënteerd in de stereomix.

```
PLAY "CEG"
PLAY "C", "E", "G"
```

Binnen een **PLAY**-tekenreeks is een muzieknoot een teken (A, B, C, D, E, F of G) dat eventueel door een wijzigingsteken kan worden voorafgegaan.

Mogelijke wijzigingstekens zijn:

Teken	Effect
#	Kruis
$	Mol
.	Punt
W	Hele noot
H	Halve noot
Q	Kwartnoot
I	Achtste noot
S	Zestiende noot
R	Rust

Let op: het wijzigingsteken (**.**) staat vóór de nootnaam, niet erna zoals in traditionele bladmuziek.

Directieven bestaan uit een letter gevolgd door een cijfer. Directieven blijven van kracht voor alle volgende noten, totdat de parameter door een andere directief wordt gewijzigd.

Teken	Directief	Bereik
O	Octaaf	0 – 6
T	Instrumentomhulling	0 – 9
V	Volume	0 – 9
X	Filter	0 – 1
M	Modulatie	0 – 9
P	Portamento	0 – 9
L	Herhaling	N.v.t.

Een octaaf is een bereik noten van C tot B. Het standaard octaaf is 4, het "middelste" octaaf.

Instrumentomhullingen beschrijven de aard van het geluid. Zie **ENVELOPE** voor een lijst met standaard omhullingsstijlen en informatie over het aanpassen van de tien omhullingen.

De modulatiedirectief voegt een toonhoogtegebaseerde vibrato aan uw noot toe, met de sterkte die u opgeeft (1 – 9). Een waarde van 0 schakelt het uit.

De portamentodirectief schuift op de snelheid die u opgeeft (1 – 9) tussen opeenvolgende noten. Een waarde van 0 schakelt het uit. Noten worden aan het eind van elke noot nog steeds uitgeschakeld (gate-off) zolang portamento is ingeschakeld.

Indien een tekenreeks eindigt met de directief **L**, wordt het patroon na voltooiing teruggelusd naar het begin van de tekenreeks.

U kunt een tekenreeks voor een bepaalde stem overslaan, zodat een al spelend patroon op die stem doorgaat, door lege argumenten te gebruiken:

```
PLAY "O4EDCDEEERL",,, "O2CGEGCGEGL"
```

Een voorbeeld met stem 2 en stem 5

```
PLAY , "O5T2IGAGFEDCEGO6.QCL",,, "O3T2.QG.B O4ICO3GE.QCL"
```

RPLAY(stem) test of er muziek wordt afgespeeld op de opgegeven stem en retourneert 1 indien dit het geval is, of 0 indien niet.

Een beperking waar u rekening mee moet houden, is dat BASIC-strings een maximale lengte van 255 bytes hebben. Indien uw melodie deze lengte overschrijdt, kunt u de melodie opsplitsen in meerdere tekenreeksen en vervolgens **RPLAY(stem)** gebruiken om te bepalen wanneer de eerste tekenreeks is geëindigd, waarna de volgende tekenreeks kan worden afgespeeld.

Instrumentomhullingen kunnen worden aangepast met het **ENVELOPE**-statement. De standaardinstellingen voor de omhullingen staan op bladzijde B-103.

Retourneert:

Het **PLAY**-statement maakt gebruik van een interruptgestuurde routine die de tekenreeks begint te verwerken en de melodie af te spelen. De programma-uitvoering gaat door met het volgende statement en blokkeert niet totdat de melodie is geëindigd. Dit verschilt van de Commodore 128, die de programma-uitvoering tijdens het afspelen onderbreekt.

De 6 stemkanalen die door het **PLAY**-commando worden gebruikt (op SID1 en SID3) zijn onafhankelijk van de 6 kanalen die door het **SOUND**-commando worden gebruikt (op SID2 en SID4). Geluidseffecten onderbreken de muziek niet en omgekeerd.

Voorbeelden:

Gebruik van **PLAY**

```
 5 REM *** VOORBEELD VAN EEN EENVOUDIGE ZWAAI ***
10 ENVELOPE 9, 10, 5, 10, 5, 0, 300
20 VOL 8, 8
30 TEMPO 30
40 PLAY "O5T9HCIDCDEHCG IGAGFEFDEWCL", "O2T0QCGEGCGEG DBGB CGEGL"
```

```
 5 REM *** VOORBEELD VAN MODULATIE + PORTAMENTO ***
10 TEMPO 20
20 M$ = "M5 T2O5P0QD P5FP0RP5QG .AI#AQA HGQE.C IDQE HFQD .DI#CQD HEQ#CQO4HA"
30 M$ = M$ + "O5QDHFQG.AI#AQA HGQE.C IDQEFED#CO4BO5HC DO4AFD P0R L"
40 B$ = "T0QRO2H.D.F.CO1.A.#A.G.A QAIO2AGFE H.D.F.CO1.A.#A.AO2 .D DL"
50 PLAY M$, B$
```

POINTER

Token: $CE $0A

Formaat: **POINTER**(variabele)

Retourneert:

Het huidige adres van een variabele of een array-element als een 32-bits pointer.

Voor stringvariabelen is dit het adres van de stringdescriptor, niet van de string zelf. De stringdescriptor bestaat uit drie bytes: lengte, lage byte van het stringadres, hoge byte van het stringadres. Het stringadres is een offset in bank 1.

Voor scalaire variabelen van een numeriek type is dit het adres van de waarde. Het formaat hangt af van het type. Een bytevariabele (`A&`) beslaat één byte in het tweecomplement-formaat voor gehele getallen. Een integervariabele (`A%`) beslaat twee bytes, waarbij de meest significante byte als eerste komt. Een reële variabele (`A`) beslaat vijf bytes in het compacte zwevende-kommagetalformaat.

Om het adres van een array te verkrijgen, gebruikt u **POINTER** met het eerste element van de array (index `0` in elke dimensie). Array-elementen worden opeenvolgend opgeslagen in het formaat van het scalaire record, waarbij de meest linkse index de kortste stap gebruikt. Een array die is gedefinieerd als `DIM A%(3, 2)` begint op adres `POINTER(A%(0, 0))`, heeft tweebiterecords en is geordend als:

(0, 0) (1, 0) (2, 0) (3, 0) (0, 1) (1, 1) (2, 1) (3, 1) ...

Opmerkingen:

De adreswaarden vanarraysen hun elementen zijn constant tijdens de uitvoering van het programma. De adressen van strings (niet hun descriptors) kunnen echter te allen tijde wijzigen door “garbage collection”.

Voorbeeld: Gebruik van **POINTER**

```
10 BANK 0                          : REM SCALAIRE VARS ZITTEN IN BANK 0
20 H$ = "HALLO"                    : REM WIJS TEKENREEKS TOE AAN H$
30 P = POINTER(H$)                 : REM VRAAG DESCRIPTORADRES OP
40 PRINT "DESCRIPTOR OP: $"; HEX$(P)
50 L = PEEK(P) : SP = WPEEK(P+1)   : REM LENGTE & STRINGPOINTER
60 PRINT "LENGTE = "; L            : REM PRINT LENGTE
70 BANK 1                          : REM STRINGS ZITTEN IN BANK 1
80 FOR I% = 0 TO L - 1 : PRINT PEEK(SP + I%); : NEXT : PRINT
90 FOR I% = 0 TO L - 1 : PRINT CHR$(PEEK(SP + I%)); : NEXT : PRINT

RUN
DESCRIPTOR AT: $FD75
LENGTH =  5
 72  69  76  76  79
HELLO
```

POKE

Token: $97

Formaat: **POKE** adres**,** waarde [**,** waarde ...]

Retourneert:

Schrijft één of meer bytes in het geheugen of geheugengestuurde I/O, beginnend bij **adres**.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

Indien **waarde** in het bereik van 0 - 255 valt, wordt deze in het geheugen gepoked; anders wordt de lage byte van de waarde gebruikt. Een commando als `POKE AD,V AND 255` kan daarom worden geschreven als `POKE AD,V`, omdat **POKE** sowieso de lage byte gebruikt.

Opmerkingen:

Het adres wordt voor elke databyte verhoogd, waardoor een geheugenbereik met één enkele **POKE** kan worden geschreven.

Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enz.

Voorbeeld: Gebruik van **POKE**

```
10 BANK 128          : REM KIES SYSTEEMBANK
20 POKE $02F8, 0, 24 : REM ZET USR-VECTOR OP $1800
```

POLYGON

Token: $FE $2F

Formaat: **POLYGON** x, y, xrad, yrad, zijden [{, tekenzijden, overspannen, hoek, massief}]

Gebruik: Bitmapgrafiek: tekent een regelmatige n-hoek. De veelhoek wordt getekend met de huidige tekensituatie die is ingesteld met **SCREEN**, **PALETTE** en **PEN**.

x,y centrumcoördinaten.

xrad,yrad straal in x- en y-richting.

zijden aantal zijden van de veelhoek.

tekenzijden aantal te tekenen zijden.

overspannen tekent een lijn van het centrum naar het beginpunt indien ingesteld (1).

hoek beginhoek.

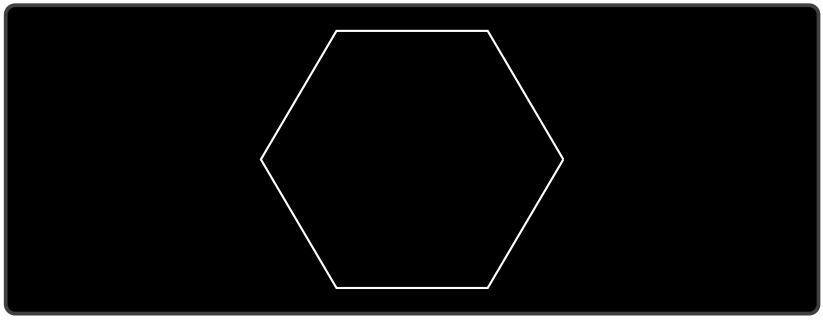
massief vult (1) of omlijnt (0) de veelhoek.

Opmerkingen:

Een regelmatige veelhoek is zowel isogoon als isotoxaal, wat betekent dat alle zijden en hoeken gelijk zijn.

Voorbeeld: Gebruik van **POLYGON**

```
100 SCREEN 320, 200, 1          : REM OPEN SCHERM VAN 320 x 200
110 POLYGON 160, 100, 40, 40, 6 : REM TEKEN HONINGGRAAT
120 GETKEY A$                   : REM WACHT OP TOETSDRUK
130 SCREEN CLOSE                : REM GRAFISCH SCHERM UIT
```



POS

Token: $B9

Formaat: **POS**(dummy)

Retourneert:
De huidige cursorkolom ten opzichte van het momenteel gebruikte venster.

dummy numerieke waarde, die wordt genegeerd.

Opmerkingen:
POS geeft de kolompositie van de schermcursor. Het werkt niet voor omgeleide uitvoer.

Voorbeeld: Gebruik van **POS**

```
10 IF POS(0) > 72 THEN PRINT : REM VOEG RETURN IN
```

POT

Token: $CE $02

Formaat: **POT**(paddle)

Retourneert:

De positie van een paddle-invoerapparaat.

paddle paddle-nummer (1 - 4).

De lage byte van de retourwaarde is de paddle-waarde, waarbij 0 de met de klok mee is en 255 de antimet de klok mee.

Een waarde groter dan 255 geeft aan dat de vuurknop ook wordt ingedrukt.

Opmerkingen:

Analoge paddles zijn onnauwkeurig. Het bereik kan kleiner zijn dan 0 - 255 en de waarden die door **POT** worden geretourneerd, kunnen enigszins fluctueren.

Paddles die zijn gemaakt voor Atari-spelconsoles geven andere waarden terug dan paddles voor Commodore-computers. Commodore-paddles leveren de meest nauwkeurige waarden in het bereik 0 - 255 bij een volledige draai van de paddle. Atari-paddles gaan doorgaans slechts voor een deel van een draai van 0 naar 255, wat resulteert in een snellere beweging en een "dode zone" voor de rest van de draai.

Voorbeeld: Gebruik van **POT**

```
10 X = POT(1)    : REM LEES PADDLE #1
20 B = X ) 255   : REM WAAR (-1) INDIEN VUURKNOP INGEDRUKT
30 V = X AND 255 : REM WAARDE PADDLE #1
```

PRINT

Token: $99

Formaat: **PRINT** argumenten

Gebruik: Drukt een reeks waarden af die zijn opgemaakt naar de huidige uitvoerstroom, doorgaans het scherm.

Waarden worden opgemaakt op basis van hun type. Voor meer controle over de opmaak, zie **PRINT USING**.

De volgende expressies en tekens kunnen in de argumentenlijst voorkomen:

- **numeriek**: De afdruk begint met een spatie voor positieve en nulwaarden, of een minteken voor negatieve waarden. Integere waarden worden afgedrukt met het benodigde aantal cijfers. Reële waarden worden afgedrukt in vastekommavorm (doorgaans 9 cijfers) of wetenschappelijke notatie indien de waarde buiten het bereik van 0.01 tot 999999999 valt.
- **tekenreeks**: De tekenreeks kan bestaan uit afdrukbare tekens en controlcodes. Afdrukbare tekens worden op de cursorpositie afgedrukt. Controlcodes worden uitgevoerd.
- Puntkomma (**;**) scheidt argumenten van de lijst. Er worden geen tekens afgedrukt. Een puntkomma aan het einde van de argumentenlijst onderdrukt het automatische terugloopteken (carriage return).
- Komma (**,**) verplaatst de cursor naar de volgende tabpositie.

Opmerkingen:

De functies **SPC** en **TAB** kunnen in de argumentenlijst worden gebruikt voor positionering.

CMD kan worden gebruikt om afgedrukte tekens om te leiden naar een ander apparaat dan het scherm.

Voorbeeld: Gebruik van **PRINT**

```
10 FOR I = 1 TO 10
20 PRINT I, I * I, SQR(I)
30 NEXT
```

PRINT#

Token: $98

Formaat: **PRINT#** kanaal**,** argumenten

Gebruik: Drukt een reeks waarden af die zijn opgemaakt naar het apparaat dat is toegewezen aan **kanaal**.

Waarden worden opgemaakt op basis van hun type. Voor meer controle over de opmaak, zie **PRINT USING**.

kanaal kanaalnummer dat bij een eerdere aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN** is opgegeven.

De volgende argumententypen worden verwerkt:

- **numeriek** De afdruk begint met een spatie voor positieve en nulwaarden, of een minteken voor negatieve waarden. Integere waarden worden afgedrukt met het benodigde aantal cijfers. Reële waarden worden afgedrukt in vastekommavorm (doorgaans 9 cijfers) of wetenschappelijke notatie indien de waarde buiten het bereik van 0.01 tot 999999999 valt.
- **tekenreeks** Kan bestaan uit afdrukbare tekens en controlcodes. Afdrukbare tekens worden op de cursorpositie afgedrukt, terwijl controlcodes worden uitgevoerd.
- Puntkomma (**;**) scheidt argumenten van de lijst. Er worden geen tekens afgedrukt. Een puntkomma aan het einde van de argumentenlijst onderdrukt het automatische terugloopteken (carriage return).
- Komma (**,**) verplaatst de cursor naar de volgende tabpositie.

Opmerkingen:

De functies **SPC** en **TAB** zijn niet geschikt voor apparaten anders dan het scherm.

Voorbeeld: Gebruik van **PRINT#**

```
10 DOPEN#2, "TABLE", W, U8
20 FOR I = 1 TO 10
30 PRINT#2, I, I * I, SQR(I)
40 NEXT
50 DCLOSE#2
```

U kunt controleren of het bestand **'TABLE'** is geschreven door `DIR "TA*"` in te typen, en vervolgens de inhoud van het bestand bekijken door `TYPE "TABLE"` in te typen.

PRINT USING

Token: $98 $FB of $99 $FB

Formaat: **PRINT**[**#** kanaal**,**] **USING** indeling**;** argument

Gebruik: Geeft een reeks waarden weer die met behulp van een patroon zijn opgemaakt, naar de huidige uitvoerstroom (doorgaans het scherm) of een uitvoerkanaal.

Het argument kan een string of een numerieke waarde zijn. De opmaak van de uitvoer wordt bepaald door de **opmaak**-tekenreeks.

Het **kanaal**nummer, dat eerder is opgegeven bij een aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN**. Indien geen kanaal wordt opgegeven, gaat de uitvoer naar het scherm.

indeling is een stringvariabele of een stringconstante die de regels voor het opmaken definieert. Bij het gebruik van een getal als **argument** kan de opmaak zowel in CBM-stijl plaatsvinden (met een patroon als "###.##") als in C-stijl, met een specificatie voor **breedte**.**precisie** zoals "%3D %7.2F %4X".

Het te formatteren **argument**-nummer. Indien het argument niet in het format past, bijvoorbeeld bij het afdrukken van een variabele met vier cijfers in een reeks van drie hashes "###", wordt in plaats daarvan een asterisk (*) gebruikt.

Opmerkingen:

De formatstring wordt slechts op één argument toegepast, doch het is mogelijk om er meer toe te voegen met opeenvolgende **USING indeling;argument**-reeksen.

Het **argument** kan afdrukbare tekens en controlcodes bevatten. Afdrukbare tekens worden op de cursorpositie weergegeven, terwijl controlcodes worden uitgevoerd.

Het aantal #-tekens bepaalt de breedte van de uitvoer. Indien het eerste teken van de formatstring een gelijkteken (**=**) is, wordt het argument gecentreerd. Indien het eerste teken een groter-dan-teken (**>**) is, wordt het argument rechts uitgelijnd.

Bij een **F**-patroon in C-stijl telt de **breedte** de cijfers mee, maar niet het decimaalteken, zodat `%6.4F` zeven tekens afdrukt en `%7.2F` acht tekens. Dit verschilt van C, waar de breedte het decimaalteken wel meetelt.

Bij een patroon in C-stijl kan met een voorloop-nul worden gewerkt in plaats van spaties, door de **breedte** van een '0' te laten voorafgaan, zoals: `%07.2F` Voorloopnullen worden verwijderd indien de waarde negatief is. Velden na het decimaalteken en hexadecimale patronen worden standaard links met nullen aangevuld.

Voorbeelden:

Gebruik van **PRINT USING**

```
PRINT USING "##.##"; π, USING " [%6.4F] "; SQR(2)
 3.14 [ 1.4142]

PRINT USING " ( # # # ) ";12*31
 ( 3 7 2 )

PRINT USING "####"; "ABCDE"
ABC

PRINT USING ")####"; "ABCDE"
CDE

PRINT USING "ADDRESS:$%4X";65000
ADDRESS:$FDE8

A$ = "####,####,####.#" : PRINT USING A$; 1E7 / 3
  3,333,333.3
```

RCOLOR

Token: $CD

Formaat: **RCOLOR**(kleurbron)

Retourneert:

De huidige kleurindex voor de geselecteerde kleurbron.

De kleurbronnen zijn:

- 0: Achtergrondkleur (VIC $D021).
- 1: Tekstkleur ($F1).
- 2: Markeerkleur ($02D8).
- 3: Randkleur (VIC $D020).

Voorbeeld: Gebruik van **RCOLOR**

```
10 C = RCOLOR(3) : REM KLEURINDEX VAN RANDKLEUR
```

RCURSOR

Token: $FE $42

Formaat: **RCURSOR** {kolvar, rijvar}

Gebruik: Leest de huidige cursorkolom en -rij in variabelen in.

Opmerkingen:
De rij- en kolomwaarden beginnen bij nul, waarbij de meest linker kolom nul is en de bovenste rij nul.

Voorbeeld: Gebruik van **RCURSOR**

```
100 CURSOR ON, 20, 10
110 PRINT "[HERE]";
120 RCURSOR X, Y
130 PRINT " COL:"; X; " ROW:"; Y

RUN


                [HERE] COL: 26  ROW: 10
```

RDISK

Token: $CE $15

Formaat: **RDISK**(eigenschapsnummer)

Retourneert:

De waarde van een eigenschap van de meest recente schijfoperatie.

De eigenschappen van schijfoperaties zijn:

- 0: Het aantal bytes dat is gelezen of geschreven.
- 1: Het apparaatnummer.
- 2: Het stationnummer van een multi-station IEC-apparaat, indien van toepassing.

Voorbeeld: Gebruik van **RDISK**

```
DLOAD "MYPROGRAM",U9

10 DU=RDISK(1)
15 REM : LAAD BRONNENBESTAND VANAF ZELFDE STATION ALS PROGRAMMA
20 BLOAD "MIJNBRONNEN",U(DU)
```

READ

Token: $87

Formaat: **READ** variabele [**,** variabele ...]

Gebruik: Leest waarden uit **DATA**-regels in variabelen in.

De **variabelenlijst** bevat willekeurige geldige variabelen.

Alle soorten constanten (gehele getallen, reële getallen en strings) kunnen worden gelezen, maar geen expressies. Elementen worden door komma's gescheiden. Strings die komma's, dubbele punten of spaties bevatten, moeten tussen aanhalingstekens worden geplaatst.

RUN initialiseert de gegevenspointer op het eerste element van de eerste **DATA**-regel en schuift deze bij elk gelezen element op. Het is de verantwoordelijkheid van de programmeur dat het type van de constante en de variabele in de **READ**-regel overeenkomen. Lege elementen zonder constante tussen komma's zijn toegestaan en worden als nul geïnterpreteerd voor numerieke variabelen en als een lege string voor stringvariabelen.

RESTORE kan worden gebruikt om de gegevenspointer op een specifieke regel te zetten voor volgende leesacties.

Opmerkingen:

Het is goede programmeringspraktijk om grote hoeveelheden **DATA**-regels aan het einde van het programma te plaatsen, zodat ze de zoekactie naar regelnummers na **GOTO** en andere instructies met regelnummerdoelen niet vertragen.

Voorbeeld: Gebruik van **READ**

```
10 READ NA$, VE
20 READ N% : FOR I = 2 TO N% : READ GL(I) : NEXT I
30 PRINT "PROGRAM:"; NA$; "   VERSION:"; VE
40 PRINT "N-POINT GAUSS-LEGENDRE FACTORS E:"
50 FOR I = 2 TO N% : PRINT I; GL(I) : NEXT I
30 STOP
80 DATA "MEGA65", 1.1
90 DATA 5, 0.5120, 0.3573, 0.2760, 0.2252
```

RECORD

Token: $FE $12

Formaat: **RECORD#** kanaal**,** record [**,** byte]

Gebruik: Positioneert de lees-/schrijfpointer van een relatief bestand.

Het **kanaal**nummer, dat eerder is opgegeven bij een aanroep van commando's zoals **DOPEN** of **OPEN**.

Het doel**record** (1 - 65535).

De **byte**-positie in het record.

RECORD kan alleen worden gebruikt voor bestanden van het type **REL**, welke relatieve bestanden zijn die directe toegang mogelijk maken.

RECORD positioneert de bestandspointer op het opgegeven recordnummer. Indien dit recordnummer niet bestaat en er voldoende ruimte op de schijf is waar **RECORD** naartoe schrijft, wordt het bestand uitgebreid tot het gevraagde aantal records door lege records toe te voegen. Wanneer dit gebeurt, geeft de schijfstatus het bericht "RECORD NOT PRESENT" weer, maar dit is geen fout!

Na een aanroep van **INPUT#** of **PRINT#** schuift de bestandspointer door naar de volgende recordpositie.

Opmerkingen:

De Commodore-schijfstations hebben een bug in hun DOS, waardoor gegevens verloren kunnen gaan bij het gebruik van relatieve bestanden. Een aanbevolen tijdelijke oplossing is om het commando **RECORD** tweemaal aan te roepen, voor en na de I/O-bewerking.

Voorbeeld: Gebruik van **RECORD**

```
100 DOPEN#2, "DATA BASE", L240        : REM OPENEN OF AANMAKEN
110 FOR I% = 1 TO 20                  : REM SCHRIJFLUS
120 PRINT#2, "RECORD #"; I%           : REM SCHRIJF RECORD
130 NEXT I%                           : REM HERHAAL
140 DCLOSE#2                          : REM SLUIT BESTAND
150                                   : REM NU TESTEN
160 DOPEN#2, "DATA BASE", L240        : REM HEROPEN
170 FOR I% = 20 TO 2 STEP -2          : REM LEES BESTAND ACHTERWAARDS
180 RECORD#2, I%                      : REM POSITIONEER OP RECORD
190 INPUT#2, A$                       : REM LEES RECORD
200 PRINT A$; : IF I% AND 2 THEN PRINT
210 NEXT I%                           : REM HERHAAL
220 DCLOSE#2                          : REM SLUIT BESTAND

RUN
RECORD # 20 RECORD # 18
RECORD # 16 RECORD # 14
RECORD # 12 RECORD # 10
RECORD # 8 RECORD # 6
RECORD # 4 RECORD # 2
```

REM

Token: $8F

Formaat: **REM**

Gebruik: **REM** (afkorting van remark) negeert alle volgende tekens op een regel BASIC-code als een code-opmerking.

Voorbeeld: Gebruik van **REM**

```
10 REM *** PROGRAMMATITEL ***
20 N = 1000     : REM AANTAL ELEMENTEN
30 DIM NA$(N)
```

RENAME

Token: $F5

Formaat: **RENAME** oud **TO** nieuw [**,D** station] [**,U** eenheid]

Gebruik: Hernoemt een schijfbestand.

Het **oude** bestand is ofwel een string tussen aanhalingstekens, bijv. `"DATA"` of een stringexpressie tussen haakjes, bijv. `(FI$)`.

Het **nieuwe** bestand is ofwel een string tussen aanhalingstekens, bijv. `"BACKUP"` of een stringexpressie tussen haakjes, bijv. `(FS$)`.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

RENAME wordt uitgevoerd in de DOS van het schijfstation. Het kan alle reguliere bestandstypen hernoemen: **PRG**, **REL**, **SEQ**, **USR** en **CBM**. Het oude bestand moet bestaan en het nieuwe bestand mag niet bestaan. Er kunnen slechts enkele bestanden worden hernoemd; jokertekens zoals '*' en '?' zijn niet toegestaan. Het bestandstype kan niet worden gewijzigd.

Voorbeeld: Gebruik van **RENAME**

```
RENAME "CODES" TO "BACKUP" : REM HERNOEM ENKEL BESTAND OP STANDAARD UNIT
```

RENUMBER

Token: $F8

Formaat: **RENUMBER** [**C**] [{nieuw**,** stapgrootte**,** bereik}]

Gebruik: Herenummereert regels van een BASIC-programma.

Het **nieuwe** beginregelnummer van het regelbereik dat hernummerd moet worden. De standaardwaarde is 10.

De **stapgrootte** die moet worden gebruikt. De standaardwaarde is 10.

Het **bereik** van regels dat hernummerd moet worden. De standaardwaarden lopen van de eerste tot de laatste regel.

Indien de vlag **C** is opgegeven, verwijdert **RENUMBER** spaties die vóór regelnummers staan. Dit vergroot de kans dat de gewijzigde regelnummers de regel niet te lang maken voor weergave op het scherm, in zeldzame gevallen waarin de regel al dicht bij de maximale logische regellengte ligt. Dit heeft geen invloed op de werking van de code.

RENUMBER wijzigt alle regelnummers in het gekozen bereik en past ook alle verwijzingen aan in instructies die **GOSUB**, **GOTO**, **RESTORE**, **RUN**, **TRAP**, **COLLISION** enz. gebruiken.

RENUMBER kan alleen in de directe modus worden uitgevoerd. Indien een probleem wordt gedetecteerd, zoals geheugenoverloop, onopgeloste verwijzingen of regelnummeroverstroming (meer dan 64000 regels), wordt de uitvoering gestopt met een foutmelding en blijft het programma ongewijzigd.

RENUMBER kan met 0 - 3 parameters worden aangeroepen. Niet-opgegeven parameters gebruiken hun standaardwaarden.

Opmerkingen:

RENUMBER kan voor grote programma's enkele minuten nodig hebben om uit te voeren.

RENUMBER kan alleen in de directe modus worden gebruikt.

Dit commando maakt tijdelijk geheugen in banken 4 en 5 aan en kan eventueel opgeslagen gegevens overschrijven.

Voorbeelden:

Gebruik van **RENUMBER**

```
RENUMBER                : REM NUMMERS WORDEN 10, 20, 30, ...
RENUMBER 100, 5         : REM NUMMERS WORDEN 100, 105, 110, 115, ...
RENUMBER C 601, 1, 500  : REM OPTIMALISATIE, HERNUMMER BEGINNEND BIJ 500 TOT 601, 602, ...
RENUMBER 100, 5, 120-180 : REM HERNUMMER REGELS 120-180 NAAR 100, 105, ...

10 GOTO  20
20 GOTO 10

RENUMBER 100, 10        : REM BEHOUD SPATIES
100 GOTO  110
110 GOTO 100

RENUMBER C 100,10       : REM OPTIMALISATIE
100 GOTO110
110 GOTO100
```

RESTORE

Token: $8C

Formaat: **RESTORE** [regel]

Gebruik: Zet de interne pointer voor **READ** uit **DATA**-regels.

De nieuwe **regel** voor **RESTORE**. De standaardwaarde is de eerste programmaregel.

Opmerkingen:

De nieuwe pointerdoel**regel** hoeft geen **DATA**-regels te bevatten. Elke **READ** schuift de pointer automatisch door naar de volgende **DATA**-regel.

Voorbeeld: Gebruik van **RESTORE**

```
10 DATA 3, 1, 4, 1, 5, 9, 2, 6
20 DATA "MEGA65"
30 DATA 2, 7, 1, 8, 2, 8, 9, 5
40 FOR I = 1 TO 8 : READ P : PRINT P : NEXT
50 RESTORE 30
60 FOR I = 1 TO 8 : READ P : PRINT P : NEXT
70 RESTORE 20
80 READ A$ : PRINT A$
```

RESUME

Token: $D6

Formaat: **RESUME** [regel | **NEXT**]

Gebruik: Hervat de normale programma-uitvoering in een **TRAP**-routine na het afhandelen van een fout.

RESUME zonder parameters probeert de instructie die de fout veroorzaakte opnieuw uit te voeren. De **TRAP**-routine heeft de plaats waar de fout optrad inmiddels moeten onderzoeken en herstellen.

Het **regel**nummer waarop de programma-uitvoering wordt hervat.

NEXT hervat de uitvoering na de instructie die de fout veroorzaakte. Dit kan de volgende instructie op dezelfde regel zijn, gescheiden door een dubbele punt (:), of de instructie op de volgende regel.

Opmerkingen:

RESUME kan niet in de directe modus worden gebruikt.

Voorbeeld: Gebruik van **RESUME**

```
 10 TRAP 100
 20 FOR I = 1 TO 100
 30 PRINT EXP(I)
 40 NEXT
 50 PRINT "GESTOPT BIJ I ="; I
 60 END
100 PRINT ERR$(ER) : RESUME 50
```

RETURN

Token: $8E

Formaat: **RETURN**

Gebruik: Geeft de besturing terug vanuit een subroutine die is aangeroepen met **GOSUB** of een gebeurtenisafhandelaar die is gedeclareerd met **COLLISION**.

De uitvoering gaat verder bij de instructie na de **GOSUB**-aanroep.

In het geval van de **COLLISION**-afhandelaar gaat de uitvoering verder bij de instructie waarvandaan de afhandelaar werd aangeroepen.

Voorbeeld: Gebruik van **RETURN**

```
 10 SCNCLR            : REM MAAK SCHERM LEEG
 20 FOR I = 1 TO 20   : REM BEGIN EEN LUS
 30 GOSUB 100         : REM ROEP SUBROUTINE AAN
 40 NEXT I            : REM HERHAAL
 50 END               : REM EINDE PROGRAMMA
100 CURSOR ON, I, I, 0 : REM ACTIVEER EN POSITIONEER CURSOR
110 PRINT "X";        : REM PRINT X
120 SLEEP 0.5         : REM WACHT 0.5 SECONDEN
130 CURSOR OFF        : REM KNIPPERENDE CURSOR AFZETTEN
140 RETURN            : REM TERUG NAAR DE PLAATS VAN AANROEP
```

RGRAPHIC

Token: $CC

Formaat: **RGRAPHIC**(scherm, parameter)

Retourneert:
Bitmap-graphics: de status van een gegeven grafische scherminstelling.

Parameter	Beschrijving
0	Open (1), Gesloten (0), of Ongeldig (>1)
1	Breedte (0=320, 1=640)
2	Hoogte (0=200, 1=400)
3	Diepte (1 - 8 bitplanes)
4	Gebruikte bitplanes (bitmasker)
5	Gebruikte blokken bank 4 (bitmasker)
6	Gebruikte blokken bank 5 (bitmasker)
7	Tekenvenster # (0 - 3)
8	Weergavevenster # (0 - 3)
9	Tekentstanden (bitmasker)
10	Patroontype (bitmasker)

Voorbeeld: Gebruik van **RGRAPHIC**

```
 10 GRAPHIC CLR                                : REM INITIALISEER
 20 SCREEN DEF 0, 1, 0, 4                      : REM SCHERM 0: 640 X 200 X 4
 30 SCREEN OPEN 0                              : REM OPEN
 40 SCREEN SET 0, 0                            : REM TEKEN OP 0 EN TOON 0
 50 SCNCLR 0                                   : REM SCHERM LEEGMAKEN
 60 PEN 0, 1                                   : REM KIES KLEUR
 70 LINE 0, 0, 639, 199                        : REM TEKEN LIJN
 80 FOR I = 0 TO 10 : A(I) = RGRAPHIC(0, I) : NEXT
 90 SCREEN CLOSE 0
100 FOR I = 0 TO 6 : PRINT I; A(I) : NEXT      : REM PRINT INFO

RUN
 0  1
 1  1
 2  0
 3  4
 4  15
 5  15
 6  15
```

RIGHT$

Token: $C9

Formaat: **RIGHT$**(tekenreeks**,** n**)**

Retourneert:

Een tekenreeks met de laatste **n** tekens van **tekenreeks**.

Indien de lengte van **tekenreeks** kleiner of gelijk is aan **n**, is de resulterende tekenreeks identiek aan het argument.

tekenreeks tekenreeksexpressie.

n numerieke expressie (0 - 255).

Opmerkingen:

Lege tekenreeksen en lengtes van nul zijn toegestane waarden.

Voorbeeld: Gebruik van **RIGHT$**

```
PRINT RIGHT$("MEGA65", 2)
65
```

RMOUSE

Token: $FE $3F

Formaat: **RMOUSE** x-variabele**,** y-variabele**,** knopvariabele

Gebruik: Leest de muispositie en de knopstatus.

x-variabele numerieke variabele waarin de x-positie wordt opgeslagen.

y-variabele numerieke variabele waarin de y-positie wordt opgeslagen.

knopvariabele numerieke variabele die de knopstatus ontvangt. De linker knop zet bit 7, de rechter knop zet bit 0.

De coördinaten zijn compatibel met sprite-coördinaten. Dit is beperkt tot het zichtbare scherm binnen de rand, met de coördinaten (24, 50) linksboven.

Waarde	Status
0	Geen knop
1	Rechter knop
128	Linker knop
129	Beide knoppen

Opmerkingen:

RMOUSE plaatst -1 in alle variabelen indien de muis niet is aangesloten of is uitgeschakeld.

Voorbeeld: Gebruik van **RMOUSE**

```
10 MOUSE ON, 1, 1                              : REM MUIS IN POORT 1 MET SPRITE 1
20 RMOUSE XP, YP, BU                           : REM LEES MUISSTATUS
30 IF XP < 0 THEN PRINT "GEEN MUIS IN POORT 1" : STOP
40 PRINT "MUIS:"; XP; YP; BU
50 MOUSE OFF                                   : REM MUIS UIT
```

RND

Token: $BB

Formaat: **RND**(type)

Retourneert:

Een pseudowillekeurig getal.

Dit wordt een “pseudowillekeurig” getal genoemd, omdat computers geen werkelijk willekeurige getallen kunnen genereren. Pseudowillekeurige getallen worden rekenkundig afgeleid van een ander getal, een “seed” genoemd, dat reproduceerbare reeksen genereert. **type** bepaalt welke seedmethode wordt gebruikt:

- type = 0: Gebruik de systeemklok.
- type < 0: Gebruik de waarde van **type** als seed.
- type > 0: Leid een nieuw willekeurig getal af van het vorige.

Opmerkingen:

Willekeurige getallenreeksen met een seed produceren dezelfde reeks voor identieke seeds.

Het algoritme wordt tijdens het opstarten geïnitialiseerd met een seed vanuit de Real-Time Clock en andere factoren, waardoor **RND**(1) onwaarschijnlijk dezelfde reeks twee keer retourneert. Dit verschilt van de Commodore 64, die altijd dezelfde initiële seed gebruikte. Indien **RND** ooit met een negatieve waarde wordt aangeroepen, wordt die waarde als nieuwe seed gebruikt en worden de reeksen die door **RND**(1) worden gegenereerd voorspelbaar. Dit is met name nuttig bij het uitvoeren van bepaalde tests. Gebruik **RND**(0) om opnieuw te seeden met een onvoorspelbare waarde.

Elke aanroep van **RND**(0) genereert een nieuwe seed op basis van de systeemklok en andere factoren. Het herhaald aanroepen van **RND**(0) produceert doorgaans een betere verdeling van waarden dan op een Commodore 64, vanwege de precisie van de seedbronnen.

Voorbeeld: Gebruik van **RND**

```
10 DEF FN DI(X) = INT(RND(0) * 6) + 1 : REM DUBBELSTEENFUNCTIE
20 FOR I = 1 TO 10                    : REM 10 KEER TONEN
30 PRINT I; FN DI(0)                  : REM PRINT GEDOBBELDE WAARDE
40 NEXT
```

RPALETTE

Token: $CE $0D

Formaat: **RPALETTE**(scherm**,** index**,** rgb)

Retourneert:

De rode, groene of blauwe waarde van een kleurenpaletindex.

scherm schermnummer (0 - 3), of een negatieve waarde om een van de vier systeempaletten te selecteren: -1 voor systeempalet 0 (het standaard systeempalet), -2 voor systeempalet 1, -3 voor palet 2, of -4 voor palet 3.

index kleurenpaletindex.

rgb (0: rood, 1: groen, 2: blauw).

Voorbeeld: Gebruik van **RPALETTE**

```
10 SCREEN 320, 200, 4                : REM DEFINIEER EN OPEN SCHERM
20 R = RPALETTE(0, 3, 0)             : REM LEES ROOD
30 G = RPALETTE(0, 3, 1)             : REM LEES GROEN
40 B = RPALETTE(0, 3, 2)             : REM LEES BLAUW
50 SCREEN CLOSE                      : REM SLUIT SCHERM
60 PRINT "PALETINDEX 3 RGB ="; R; G; B

RUN
PALETINDEX 3 RGB = 0  15  15
```

RPEN

Token: $D0

Formaat: **RPEN**(n)

Retourneert:

De kleurindex van pen **n**.

n pennummer (0 – 2), waarbij:

- 0: Tekenpen.
- 1: Wispen.
- 2: Omtrekpen.

Voorbeeld: Gebruik van **RPEN**

```
 10 GRAPHIC CLR          : REM INITIALISEER
 20 SCREEN DEF 0, 1, 0, 4 : REM SCHERM 0: 640 X 200 X 4
 30 SCREEN OPEN 0        : REM OPEN
 40 SCREEN SET 0, 0      : REM TEKEN OP 0 EN TOON 0
 50 SCNCLR 0             : REM LEEGMAKEN
 60 PEN 0, 1             : REM KIES KLEUR
 70 X = RPEN(0)
 80 Y = RPEN(1)
 90 C = RPEN(2)
100 SCREEN CLOSE 0
110 PRINT "KLEUR VAN TEKENPEN = "; X

RUN
KLEUR VAN TEKENPEN =  1
```

RPLAY

Token: $CE $0F

Formaat: **RPLAY**(stem)

Retourneert:

Test of er muziek wordt afgespeeld op het opgegeven stemkanaal.

stem het te beoordelen stemkanaal, van 1 tot en met 6.

Retourneert 1 indien er muziek op het kanaal wordt afgespeeld, anders 0.

Voorbeeld: Gebruik van **RPLAY**

```
10 PLAY "O4ICDEFGABO5CR", "O2QCGEGCO1GCR"
20 IF RPLAY(1) OR RPLAY(2) THEN GOTO 20   : REM WACHT OP EINDE LIEDJE
```

RPT$

Token: $CE $14

Formaat: **RPT$**(tekenreeks, aantal)

Retourneert:
Een nieuwe tekenreeks die bestaat uit **tekenreeks** herhaald **aantal** keer.

Opmerkingen:
Indien de nieuwe tekenreeks langer zou zijn dan 255 tekens, wordt een String Too Long-fout gerapporteerd.

Indien **tekenreeks** een lege tekenreeks is of **aantal** nul bedraagt, retourneert **RPT$()** de lege tekenreeks.

Voorbeeld: Gebruik van **RPT$**

```
PRINT RPT$("=", 30)
==============================

PRINT RPT$("HALLO ", 3),"DAAR"
HALLO HALLO HALLO DAAR
```

RREG

Token: $FE $09

Formaat: **RREG** [{areg, xreg, yreg, zreg, sreg}]

Gebruik: Leest de waarden die zich in de CPU-registers bevonden na een **SYS**-aanroep, in de opgegeven variabelen.

areg accumulatoreind waarde.

xreg X-registerwaarde.

yreg Y-registerwaarde.

zreg Z-registerwaarde.

sreg statusregisterwaarde.

Opmerkingen:
De registerwaarden na een **SYS**-aanroep worden in het systeemgeheugen opgeslagen. Hierdoor kan **RREG** ze ophalen.

Voorbeeld: Gebruik van **RREG**

```
10 POKE $1800, $18, $8A, $65, $06, $60
20 REM        CLC  TXA  ADC   06  RTS
30 SYS $1800, 77, 11                : REM A = 77 EN X = 11
40 RREG AC, X, Y, Z, S
50 PRINT "REGISTER:"; AC; X; Y; Z; S
```

RSPCOLOR

Token: $CE $07

Formaat: **RSPCOLOR**(n)

Retourneert:
De kleurinstelling van een multicolor-spritekleur.

n multicolor-spritenummer:

- 1: Haal multicolor #1 op.
- 2: Haal multicolor #2 op.

Opmerkingen:
Raadpleeg **SPRITE** en **SPRCOLOR** voor meer informatie.

Voorbeeld: Gebruik van **RSPCOLOR**

```
10 SPRITE 1, 1      : REM ZET SPRITE 1 AAN
20 C1% = RSPCOLOR(1) : REM LEES KLEUR #1
30 C2% = RSPCOLOR(2) : REM LEES KLEUR #2
```

RSPEED

Token: $CE $0E

Formaat: **RSPEED**(n)

Retourneert:

De huidige CPU-kloksnelheid in MHz.

n numeriek dummyargument dat wordt genegeerd.

De retourwaarde is een geheel getal, dat overeenkomt met de werkelijke CPU-snelheid als volgt:

- 1: 1 MHz
- 3: 3.5 MHz
- 40: 40.5 MHz

Opmerkingen:

RSPEED(n) retourneert niet de juiste waarde indien eerder `POKE 0, 65` is gebruikt om de hoogste snelheid (40 MHz) in te schakelen.

De MEGA65 ondersteunt ook het draaien van de CPU op 2 MHz, vergelijkbaar met de Commodore 128. Dit is niet beschikbaar als retourwaarde voor **RSPEED**. In dat geval retourneert **RSPEED** 1.

Raadpleeg het **SPEED**-commando voor meer informatie.

Voorbeeld: Gebruik van **RSPEED**

```
10 X = RSPEED(0)                          : REM VRAAG KLOKSNELHEID OP
20 IF X = 1  THEN PRINT "1 MHZ"   : GOTO 50
30 IF X = 3  THEN PRINT "3.5 MHZ" : GOTO 50
40 IF X = 40 THEN PRINT "40 MHZ"
50 END
```

RSPPOS

Token: $CE $05

Formaat: **RSPPOS**(sprite, n)

Retourneert:
De positie of snelheid van een sprite.

sprite het spritenummer.

n spriteparameter om op te halen:

- 0: X-positie.
- 1: Y-positie.
- 2: Snelheid.

Opmerkingen:
Raadpleeg de commando's **MOVSPR** en **SPRITE** voor meer informatie.

Voorbeeld: Gebruik van **RSPPOS**

```
10 SPRITE 1, 1      : REM ZET SPRITE 1 AAN
20 XP = RSPPOS(1, 0) : REM LEES X VAN SPRITE 1
30 YP = RSPPOS(1, 1) : REM LEES Y VAN SPRITE 1
30 SP = RSPPOS(1, 2) : REM LEES SNELHEID VAN SPRITE 1
```

RSPRITE

Token: $CE $06

Formaat: **RSPRITE**(sprite, n)

Retourneert:

Een spriteparameter.

sprite sprite-nummer (0 - 7)

n spriteparameter om terug te geven (0 - 5):

- 0: Ingeschakeld (0 of 1). 0 betekent dat de sprite uitgeschakeld is.
- 1: Voorgrondkleur (0 - 15).
- 2: Achtergrondprioriteit (0 of 1).
- 3: X-uitgebreid (0 of 1). 0 betekent dat de sprite niet X-uitgebreid is.
- 4: Y-uitgebreid (0 of 1). 0 betekent dat de sprite niet Y-uitgebreid is.
- 5: Multikleur (0 of 1). 0 betekent dat de sprite geen multikleur heeft.

Opmerkingen:

Raadpleeg de commando's **MOVSPR** en **SPRITE** voor meer informatie.

Voorbeeld: Gebruik van **RSPRITE**

```
10 SPRITE 1, 1       : ZET SPRITE 1 AAN
20 EN = RSPRITE(1, 0) : REM SPRITE 1 INGESCHAKELD ?
30 FG = RSPRITE(1, 1) : REM SPRITE 1 VOORGRONDKLEURINDEX
40 BP = RSPRITE(1, 2) : REM SPRITE 1 ACHTERGRONDPRIORITEIT
50 XE = RSPRITE(1, 3) : REM SPRITE 1 X VERDUBBELD ?
60 YE = RSPRITE(1, 4) : REM SPRITE 1 Y VERDUBBELD ?
70 MC = RSPRITE(1, 5) : REM SPRITE 1 MEERKLEUREN ?
```

RSPRSYS

Token: $CE $17

Formaat: **RSPRSYS**(n)

Retourneert:
Een spritesysteemparameter.

n parameter om terug te geven (0):

- 0: Hoge-resolutiemodus van sprite. 1=ingeschakeld, 0=uitgeschakeld.

Opmerkingen:
Raadpleeg het commando **SPRITE SYS** voor meer informatie.

Voorbeeld: Gebruik van **RSPRSYS**

```
SPRITE SYS R1

PRINT RSPRSYS(0)
1
```

RUN

Token: $8A

Formaat: **RUN** [regelnummer]
RUN bestandsnaam [**,D** station] [**,U** eenheid]
[↑] bestandsnaam

Gebruik: Voert het BASIC-programma in het geheugen uit of laadt en voert een programma vanaf schijf uit.

Indien een bestandsnaam is opgegeven, wordt het programmabestand in het geheugen geladen en uitgevoerd; anders wordt het programma dat zich op dat moment in het geheugen bevindt, gebruikt.

De ↑ kan als sneltoets worden gebruikt wanneer deze in de directe modus in de linkerste kolom staat. U kunt hiermee een programma vanuit een directorylijst laden en uitvoeren door de cursor naar de regel met de bestandsnaam te verplaatsen, de ↑ aan het begin van de regel te typen en op return te drukken. Tekens vóór en achter de aangehaalde bestandsnaam worden genegeerd (vergelijkbaar met PRG bijvoorbeeld).

regelnummer een bestaand regelnummer van het programma in het geheugen vanaf welke het programma moet worden uitgevoerd.

bestandsnaam is een aangehaalde tekenreeks, bijv. "PROG" of een tekenreeksexpressie in haakjes, bijv. (PR$). Het bestandstype moet **PRG** zijn.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

RUN herstelt eerst alle interne pointers naar hun standaardwaarden. Er zijn derhalve geen variabelen, arrays of strings gedefinieerd. De runtijdstack wordt eveneens hersteld en de tabel met geopende bestanden wordt gewist.

Om de programma-uitvoering te starten of voort te zetten zonder alles te herstel, gebruikt u **GOTO**.

Voorbeelden:

Gebruik van **RUN**

```
RUN "VLUCHTSIM" : REM LAAD EN PROGRAMMA "FLIGHTSIM" EN VOER HET UIT

RUN 1000        : REM VOER PROGRAMMA IN HET GEHEUGEN UIT, BEGIN OP REGEL 1000

RUN             : REM VOER PROGRAMMA IN HET GEHEUGEN UIT
```

RWINDOW

Token: $CE $09

Formaat: **RWINDOW**(n)

Retourneert:

Een parameter van het huidige tekstvenster.

n de schermparameter om op te halen:

- 0: Breedte van het huidige tekstvenster.
- 1: Hoogte van het huidige tekstvenster.
- 2: Aantal kolommen op het scherm (40 of 80).
- 3: Aantal regels op het scherm (25 of 50).

Opmerkingen:

Raadpleeg het commando **WINDOW** voor meer informatie.

Voorbeeld: Gebruik van **RWINDOW**

```
10 W = RWINDOW(2)          : REM LEES SCHERMBREEDTE
20 IF W = 80 THEN BEGIN    : REM IS 80 KOLOMSMODE ACTIEF?
30   PRINT CHR$(27) + "K"; : REM JA, SCHAKEL NAAR 40 KOLOMMEN
40 BEND
```

SAVE

Token: $94

Formaat: **SAVE** bestandsnaam [**,** eenheid]
[←] bestandsnaam [**,** eenheid]

Gebruik: Slaat een BASIC-programma op in een bestand van het type **PRG**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

SAVE gebruikt de verouderde syntaxis voor besturingstekens in de bestandsnaam. Zo vervangt "@0:BESTANDSNAAM" een bestaand bestand op station 0 van een multi-stationeenheid met de naam BESTANDSNAAM. Deze syntaxis verschilt van die van nieuwere commando's zoals **DSAVE**.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

SAVE is verouderd en alleen ter compatibiliteit met oudere programma's geïmplementeerd. Gebruik in plaats daarvan **DSAVE**.

De sneltoets [←] bevindt zich naast **1**. Kan alleen in de directe modus worden gebruikt.

OPMERKING: Opslaan met vervanging wordt niet aanbevolen op 1541- en 1571-stations, vanwege een bug in oudere versies van het besturingssysteem van het schijfstation welke gegevensverlies kan veroorzaken.

Voorbeelden:

Gebruik van **SAVE**

```
SAVE "AVONTUUR"

SAVE "ZORK-I", 8

SAVE "1:KERKERGEWELF", 9
```

SAVEIFF

Token: $FE $44

Formaat: **SAVEIFF** bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Bitmapgraphics: slaat het huidige grafische scherm op in een schijfbestand in **IFF**-formaat.

IFF (Interchange File Format) wordt ondersteund door veel verschillende toepassingen en besturingssystemen. **SAVEIFF** slaat de afbeelding, het kleurengeheugen en de resolutieparameters op.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$) De maximale lengte van de bestandsnaam is 16 tekens. Indien het eerste teken van de bestandsnaam een '@' (apestaartje) is, wordt dit geïnterpreteerd als een "opslaan en vervangen"-bewerking. Het wordt niet aanbevolen deze optie te gebruiken op 1541- en 1571-stations, omdat deze een "opslaan en vervangen-fout" in hun DOS bevatten.

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Bestanden die met **SAVEIFF** zijn opgeslagen, kunnen worden geladen met **LOADIFF**. Er zijn programma's beschikbaar om veelgebruikte afbeeldingsformaten naar IFF te converteren. Deze programma's zijn beschikbaar voor diverse besturingssystemen, zoals Amiga OS, macOS, Linux en Windows. **ImageMagick** is bijvoorbeeld een gratis grafisch hulpprogramma dat een tool genaamd **convert** bevat, waarmee IFF-bestanden kunnen worden gemaakt in combinatie met de tool **ppmtoilbm** uit het **Netbpm**-pakket.

Voorbeeld: Gebruik van **SAVEIFF**

```
10 SCREEN 320, 200, 2        : REM SCHER, #0 320 X 200 X 2
20 PEN 1                     : REM TEKENPENKLEUR 1 (WIT)
30 LINE 25, 25, 295, 175     : REM TEKEN LIJN
40 SAVEIFF "LINE-EXAMPLE", U8 : REM BEWAAR HUIDIG BEELD IN BESTAND
50 SCREEN CLOSE              : REM GRAFISCH SCHERM UIT EN HERSTEL PALET
```

SCNCLR

Token: $E8

Formaat: **SCNCLR** [kleur]

Gebruik: Wist een tekstvenster of een bitmapgrafisch scherm.

SCNCLR (zonder argumenten) wist het huidige tekstvenster. Het standaardvenster beslaat het volledige scherm.

SCNCLR kleur wist het grafische scherm door het te vullen met de opgegeven **kleur**.

Voorbeeld: Gebruik van **SCNCLR**

```
 10 GRAPHIC CLR             : REM INITIALISEER
 20 SCREEN DEF 1, 0, 0, 2   : REM SCHERM #1 320 X 200 X 2
 30 SCREEN OPEN 1           : REM OPEN SCHERM 1
 40 SCREEN SET 1, 1         : REM OP SCHERM 1 TEKENEN, SCHERM 1 TONEN
 50 SCNCLR 0                : REM SCHERM SCHOONMAKEN
 60 PALETTE 1, 1, 15, 15, 15 : REM DEFIEER KLEUR 1 ALS WIT
 70 PEN 0, 1                : REM TEKENPEN
 80 LINE 25, 25, 295, 175   : REM TEKEN LIJN
 90 SLEEP 10                : REM WACHT 10 SECONDEN
100 SCREEN CLOSE 1          : REM GRAFISCH SCHERM UIT EN HERSTEL PALET
```

SCRATCH

Token: $F2

Formaat: **SCRATCH** bestandsnaam [**,D** station] [**,U** eenheid] [**,R**]

Gebruik: Verwijdert ("krast") een schijfbestand.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

R herstelt een eerder verwijderd bestand. Dit werkt alleen indien er geen schrijfbewerkingen hebben plaatsgevonden tussen verwijdering en herstel, waardoor de inhoud van de schijf mogelijk is gewijzigd.

Opmerkingen:

SCRATCH is een synoniem van **ERASE** en **DELETE**.

In de directe modus wordt het succes en het aantal verwijderde bestanden afgedrukt. De voorlaatste waarde uit het bericht bevat het aantal succesvol verwijderde bestanden.

Voorbeelden:

Gebruik van **SCRATCH**

```
SCRATCH "DRM", U9          : REM VERWIJDER BESTAND "DRM" OP UNIT 9
01, FILES SCRATCHED,01,00

SCRATCH "OUD*"             : REM VERWIJDER ALLE BESTANDEN BEGINNEND OP "OUD"
01, FILES SCRATCHED,04,00

SCRATCH "R*=PRG"           : REM VRWIJDER ALLE PROGRAMMABESTANDEN BEGINNEND OP 'R'
01, FILES SCRATCHED,09,00
```

SCREEN

Token: $FE $2E

Formaat: **SCREEN** [scherm,] breedte**,** hoogte**,** diepte
SCREEN CLR kleur
SCREEN DEF scherm**,** breedte-vlag**,** hoogte-vlag**,** diepte
SCREEN SET teken-scherm**,** weergave-scherm
SCREEN OPEN [scherm]
SCREEN CLOSE [scherm]

Gebruik: Bitmapgraphics: beheert een grafisch scherm.

Bij het voorbereiden van het scherm voor het tekenen van graphics zijn er twee benaderingen beschikbaar: een vereenvoudigde benadering en een gedetailleerde benadering.

Vereenvoudigde benadering:

De eerste versie van **SCREEN** (met pixelunits voor breedte en hoogte) is de eenvoudigste manier om een grafisch scherm te starten en is de aanbevolen methode indien slechts één scherm nodig is (d.w.z. er is geen tweede scherm nodig voor double buffering). Deze versie voert alle voorbereidende werkzaamheden voor u uit en aanvaardt commando's zoals **GRAPHIC CLR**, **SCREEN CLR**, **SCREEN DEF**, **SCREEN OPEN** en **SCREEN SET** namens u uit. De volgende parameters worden ondersteund:

SCREEN [scherm,] breedte, hoogte, diepte

- screen: Het schermnummer (0 - 3) is optioneel. Indien geen schermnummer is opgegeven, wordt scherm 0 gebruikt. Om deze benadering zo eenvoudig mogelijk te houden, wordt aangeraden het standaardscherm 0 te gebruiken.
- width: 320 of 640 (standaard 320).
- height: 200 of 400 (standaard 200).
- depth: 1..8 (standaard = 8), kleuren = 2 ^depth.

De argumentenparsing is foutbestendig en gebruikt standaardwaarden voor breedte (320) en hoogte (200) indien het geparsede argument ongeldig is.

Deze versie van **SCREEN** start met een voorgedefinieerd kleurengeheugen, stelt de achtergrond in op zwart en de pen in op wit, zodat er direct met de standaardwaarden kan worden getekend.

De gedetailleerde benadering vereist daarentegen het instellen van het kleurengeheugen en de penkleur voordat er kan worden getekend.

De **kleur**-waarde moet in het bereik 0 - 15 liggen. Zie bijlage G op bladzijde G-3 voor de lijst met kleuren in het standaard kleurenpalet.

Wanneer u klaar bent met uw grafische scherm, roept u eenvoudig **SCREEN CLOSE** [scherm] aan om terug te keren naar het tekstscherm.

Gedetailleerde benadering:

De overige versies van **SCREEN** voeren specifieke acties uit en zijn bedoeld voor geavanceerde grafische programma's die meerdere schermen openen of "double buffering" vereisen. Indien u de vereenvoudigde benadering heeft gekozen, heeft u geen van deze versies nodig, met uitzondering van **SCREEN CLOSE**.

SCREEN CLR kleur (of **SCNCLR** kleur).
Wist het actieve grafische scherm door het te vullen met **kleur**.

SCREEN DEF scherm, breedtevlag, hoogtevlag, dieptevlag
Definieert de resolutieparameters voor het geselecteerde scherm. De breedte- en hoogtevlag geven aan of hoge resolutie (1) of lage resolutie (0) is geselecteerd.

- screen: Schermnummer 0 - 3.
- breedtevlag: 0 - 1 (0:320, 1:640 pixel).
- hoogtevlag: 0 - 1 (0:200, 1:400 pixel).
- depth: 1 - 8 (2 - 256 kleuren).

Merk op dat de waarden voor breedte en hoogte hier **vlaggen** zijn en **geen pixeleenheden**.

SCREEN SET tekenscherm, toonscherm
Stelt schermnummers (0 - 3) in voor het teken- en het weergavescherm; terwijl het ene scherm wordt weergegeven, kunt u op een apart scherm tekenen en later tussen de schermen wisselen. Dit staat bekend als dubbele buffering.

SCREEN OPEN scherm
Kent bronnen toe en initialiseert de grafische context voor het geselecteerde **screen** (0 - 3). Een optionele variabelenaam als extra argument ontvangt het resultaat van het commando, waarna het op succes kan worden getest.

SCREEN CLOSE [scherm]
Sluit **scherm** (0 - 3) en geeft bronnen vrij. Indien geen waarde is opgegeven, wordt standaard 0 gebruikt. Merk op dat het sluiten van een scherm automatisch **PALETTE RESTORE** uitvoert.

Opmerkingen:

Grafische schermen worden toegewezen in bank 4 en 5 van het geheugen. Indien een BASIC-programma ook geheugen in bank 4 of 5

gebruikt, kunnen geheugengebieden worden gereserveerd om te voorkomen dat **SCREEN** deze toewijst. Dit gebeurt met het commando **MEM**. Zie **MEM**.

Voorbeelden:

Gebruik van **SCREEN**

```
 5 REM *** VEREENVOUDIGDE AANPAK ***
10 SCREEN 320, 200, 2    : REM SCHERM #0: 320 X 200 X 2
20 PEN 1                 : REM TEKENPENKLEUR = 1 (WIT)
30 LINE 25, 25, 295, 175 : REM TEKEN LIJN
40 GETKEY A$             : REM WACHT OP TOETSDRUK
50 SCREEN CLOSE          : REM SLUIT SCHERM 0 (HERSTEL PALET)
```

```
 5 REM *** UITGEBREIDE AANPAK ***
10 GRAPHIC CLR              : REM INITIALISEER
20 SCREEN DEF 1, 0, 0, 2    : REM SCHERM #1: 320 X 200 X 2
30 SCREEN OPEN 1            : REM OPEN SCHERM 1
40 SCREEN SET 1, 1          : REM TEKEN OP SCHERM 1 EN TOON SCHERM 1
50 SCREEN CLR 0             : REM SCHERM SCHOONMAKEN
60 PALETTE 1, 1, 15, 15, 15 : REM DEFINIEER KLEUR 1 ALS WIT
70 PEN 0, 1                 : REM TEKENPEN
80 LINE 25, 25, 295, 175    : REM TEKEN EEN LIJN
90 SLEEP 10                 : REM WACHT 10 SECONDEN
100 SCREEN CLOSE 1          : REM SLUIT SCHERM 1 (HERSTEL PALET)
```

SET

Token: $FE $2D

Formaat: **SET DEF** eenheid
SET DISK oud **TO** nieuw
SET FAST MOUSE <**ON** | **OFF**>
SET VERIFY <**ON** | **OFF**>

Gebruik: **SET DEF** herbepaalt de standaardeenheid voor schijftoegang, welke door de DOS op 8 is geïnitialiseerd. Commando's die geen eenheid expliciet specificeren, gebruiken deze standaardeenheid.

SET DISK wordt gebruikt om het eenheidsnummer van een schijfstation tijdelijk te wijzigen.

SET FAST MOUSE schakelt het verdubbelen van de snelheid van de BASIC **MOUSE** in of uit.

SET VERIFY schakelt de DOS verify-after-write-modus voor 3.5"-stations in of uit.

Opmerkingen:
Deze instellingen zijn geldig totdat een reset of afsluiting plaatsvindt.

Voorbeelden:
Gebruik van **SET**

```
DIR             : REM TOON DIRECTORY VAN UNIT 8

SET DEF 11      : REM UNIT 11 WORDT STANDAARD

DIR             : REM TOON DIRECTORY OP UNIT 11

DLOAD "*"       : REM LAAD EERSTE BESTAND VAN UNIT 11

SET DISK 8 TO 9 : REM VERANDER UNIT# VAN STATION 8 NAAR 9

DIR U9          : REM TOON DIRECTORY VAN UNIT 9 (VOORHEEN 8)

SET FAST MOUSE ON  : REM MUIS DUBBEL ZO SNEL

SET VERIFY ON   : REM ACTIVEER VERIFICATIE NA SCHRIJVEN
```

SETBIT

Token: $FE $2D $FE $4E

Formaat: **SETBIT** adres**,** bitnummer

Gebruik: Zet een enkel bit op het opgegeven **adres**.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

bitnummer waarde in het bereik van 0 - 7.

Een bankwaarde > 127 wordt gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enz.

Voorbeeld: Gebruik van **SETBIT**

```
10 BANK 128       : REM KIES SYSTEEMBANK
20 SETBIT $D011, 6 : REM SCHAKEL UITGEBREIDE ACHTERGRONDKLEURMODUS IN
30 SETBIT $D01B, 0 : REM ZET ACHTERGRONDPRIORITEIT VOOR SPRITE 0
```

SGN

Token: $B4

Formaat: **SGN**(numerieke expressie)

Retourneert:
Het teken van een numerieke expressie, als getal.

- -1: Negatief argument.
- 0: Nul.
- 1: Positief, niet-nul argument.

Voorbeeld: Gebruik van **SGN**

```
10 ON SGN(X) + 2 GOTO 100, 200, 300 : REM DOELEN VOOR MINUS, NUL, PLUS
20 Z = SGN(X) * ABS(Y)              : REM COMBINEER TEKEN VAN X MET WAARDE VAN Y
```

SIN

Token: $BF

Formaat: **SIN**(expressie in radialen)
SIND(expressie in graden)

Retourneert:

De sinus van een hoek.

De **SIN**-functie verwacht een argument in radialen.

De **SIND**-variant verwacht een argument in graden.

Het resultaat ligt in het bereik (-1.0 tot +1.0).

Voorbeelden:

Gebruik van **SIN**

```
PRINT SIN(0.7)
 0.64421769

X = 30 : PRINT SIN(X * π / 180)
 0.5

PRINT SIND(30)
 0.5
```

SLEEP

Token: $FE $0B

Formaat: **SLEEP** seconden

Gebruik: Pauzeert de uitvoering voor de opgegeven duur.

Het argument is een positief zwevend-kommagetal in seconden. De precisie bedraagt 1 microseconde.

Opmerkingen:

Het indrukken van RUN STOP onderbreekt de slaap.

Voorbeelden:

Gebruik van **SLEEP**

```
SLEEP 10     : REM WACHT 10 SECONDEN

SLEEP 0.0005 : REM SLAAP 500 MICROSECONDEN

SLEEP 0.01   : REM SLAAP  10 MILLISECONDEN

SLEEP DD     : REM HAAL DE SLAAPTIJD UIT VARIABLE DD

SLEEP 600    : REM SLAAP 10 MINUTEN
```

SOUND

Token: $DA

Formaat: **SOUND** stem**,** frequentie**,** duur [{**,** dir**,** minimum**,** zwaai**,** golf**,** puls}]
SOUND CLR

Gebruik: **SOUND** speelt een geluidseffect af.

voice stemnummer (1 - 6).

frequentie frequentie (0 - 65535).

duur duur in jiffy's (0 - 32767). Een jiffy is afhankelijk van de beeldstandaard. Bij PAL zijn er 50 jiffy's per seconde en bij NTSC 60.

dir richting (0: omhoog, 1: omlaag, 2: oscillerend).

min minimumfrequentie (0 - 65535).

sweep zwaaistap (0 - 65535): de hoeveelheid die elke jiffy bij de frequentie wordt opgeteld (richting 0) of daarvan wordt afgetrokken (richting 1). De frequentie zwaait tussen **min** en **freq**, slaat om naar het andere uiteinde zodra hij er voorbij komt, of keert om wanneer **dir** gelijk is aan 2 (oscilleren).

wave golfvorm (0: driehoek, 1: zaagtand, 2: vierkant, 3: ruis).

pulse pulsbreedte (0 - 4095).

SOUND CLR dempt alle klank van **SOUND** en **PLAY** en herstelt het klanksysteem en alle parameters naar hun standaardwaarde.

Opmerkingen:

SOUND begint het geluidseffect af te spelen en gaat onmiddellijk verder met de uitvoering van het volgende BASIC-statement terwijl het geluidseffect wordt afgespeeld. Hierdoor kunnen afbeeldingen of tekst worden weergegeven en tegelijkertijd geluiden worden afgespeeld.

SOUND gebruikt SID2 (voor stemmen 1 tot 3) en SID4 (voor stemmen 4 tot 6) van de vier SID-chips van het systeem. Standaard zijn SID1 en SID2 enigszins naar rechts gedraaid en SID3 en SID4 enigszins naar links in de stereomix.

De zes stemkanalen die door het **SOUND**-commando worden gebruikt (op SID2 en SID4) zijn onderscheiden van de zes kanalen die door het **PLAY**-commando worden gebruikt (op SID1 en SID3). Geluidseffecten onderbreken geen muziek en omgekeerd.

Voorbeeld: Gebruik van **SOUND**

```
10 IF PEEK($D06F) AND $80 THEN J = 60 : ELSE J = 50 : REM J IS JIFFIES PER SECONDE
20 SOUND 1, 7382, J                          : REM SPEEL BLOKGOLF OP STEM 1, 1 SECONDE LANG
30 SOUND 2,  800, J*60                       : REM SPEEL BLOKGOLF OP STEM 2, 1 MINUUT LANG
40 SOUND 3, 4000, 120, 2, 2000, 400, 1          : REM SPEEL ZWAAIENDE ZAAGTAND OP STEM 3
50 SLEEP 3                                      : REM LAAT DE GELUIDEN 3 SECONDEN SPELEN
60 SOUND CLR                                      : REM GELUID STIL, HERSTEL PARAMETERS
```

SPC

Token: $A6

Formaat: **SPC**(kolommen)

Retourneert:

Als argument voor **PRINT** een tekenreeks van PETSCII-codes voor cursor naar rechts, geschikt om de cursor het opgegeven aantal kolommen verder te plaatsen.

Het afdrukken hiervan komt overeen met het indrukken van → **<kolom>** keer.

Dit is geen echte functie en genereert geen tekenreeks. Het kan alleen als argument voor **PRINT** worden gebruikt.

Opmerkingen:

De naam van deze functie is afgeleid van “spaces”, wat misleidend is. De functie drukt **cursor-rechts tekens** af, geen spaties. De inhoud van de tekenposities die worden overgeslagen, wordt niet gewijzigd.

Voorbeeld: Gebruik van **SPC**

```
10 FOR I = 8 TO 12
20 PRINT SPC(-(I < 10)); I : REM TRUE = -1, FALSE = 0
30 NEXT I
RUN
 8
 9
10
11
12
```

SPEED

Token: $FE $26

Formaat: **SPEED** [snelheid]

Gebruik: Stelt de processorkloksnelheid in op 1 MHz, 3.5 MHz of 40.5 MHz.

snelheid kloksnelheid van de CPU, waarbij:

- 1: Stelt de CPU in op 1 MHz.
- 3: Stelt de CPU in op 3.5 MHz.
- Elke andere waarde dan **1** of **3** stelt de CPU in op 40.5 MHz.

Opmerkingen:

Hoewel **SPEED** met elk zwevend-kommagetal kan worden aangeroepen, wordt het deel na de decimale punt (de punt en alle cijfers daarna) genegeerd.

SPEED is een synoniem van **FAST**.

SPEED heeft geen effect indien `POKE 0, 65` eerder is gebruikt om de CPU op 40 MHz in te stellen.

De MEGA65 ondersteunt ook het draaien van de CPU op 2 MHz, vergelijkbaar met de Commodore 128. Dit is niet beschikbaar als optie voor **SPEED**.

Voorbeelden:

Gebruik van **SPEED**

```
SPEED     : REM SET SNELHEID OP MAXIMUM (40 MHZ)

SPEED 1   : REM SET SNELHEID OP 1 MHZ

SPEED 3   : REM SET SNELHEID OP 3.5 MHZ

SPEED 3.5 : REM SET SNELHEID OP 3.5 MHZ
```

SPRCOLOR

Token: $FE $08

Formaat: **SPRCOLOR** [{mc1**,** mc2}]

Gebruik: Stelt de meerkleuren van sprites in.

SPRITE, dat de attributen van een sprite instelt, stelt alleen de voorgrondkleur in. Gebruik **SPRCOLOR** voor het instellen van de twee extra kleuren van multikleur-sprites.

Opmerkingen:

De kleuren die met **SPRCOLOR** worden ingesteld, hebben invloed op alle sprites. Raadpleeg het **SPRITE**-commando voor meer informatie.

Het laatste argument van **SPRITE** schakelt de multikleur-modus voor de sprite in.

Voorbeeld: Gebruik van **SPRCOLOR**

```
10 SPRITE 1, 1, 2,,,, 1 : REM ZET SPRITE 1 AAN (FG = 2)
20 SPRCOLOR 4, 5        : REM MC1 = 4, MC2 = 5
```

SPRITE

Token: $FE $07

Formaat: **SPRITE CLR**
SPRITE LOAD bestandsnaam [**,D** station] [**,U** eenheid]
SPRITE SAVE bestandsnaam [**,D** station] [**,U** eenheid]
SPRITE num [{**,** toon**,** kleur**,** prio**,** expx**,** expy**,** modus}]
SPRITE SYS R hires-mode

Gebruik: **SPRITE CLR** wist alle spritegegevens en herstelt alle pointers en attributen naar hun standaardwaarden.

SPRITE LOAD laadt spritegegevens vanuit **bestandsnaam** in het spritegeheugen.

SPRITE SAVE slaat spritegegevens uit het spritegeheugen op in **bestandsnaam**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

De **SPRITE**-vorm schakelt een sprite in of uit en stelt de attributen in:

num sprite-nummer.

toon sprite tonen of verbergen (1: toon, 0: verberg).

kleur sprite-voorgrondkleur.

prio de prioriteit (0: sprite voor tekst, 1: sprite achter tekst).

expx X-vergroting in- of uitschakelen (1: aan, 0: uit).

expy Y-vergroting in- of uitschakelen (1: aan, 0: uit).

modus multikleur-modus van sprite in- of uitschakelen (1: aan, 0: uit).

SPRITE SYS stelt eigenschappen van het spritesysteem als geheel in. Het aanvaardt één benoemd argument: **R hires-mode** schakelt de hoge-resolutiemodus van sprites in (1) of uit (0). Indien ingeschakeld, gebruiken alle sprites hoge-resolutiepixels in zowel verticale als horizontale dimensie, waardoor ze half zo groot lijken als in de lage-resolutiemodus. De hoge-resolutiemodus gebruikt ook een ander coördinatenvlak voor sprites.

Opmerkingen:
SPRCOLOR moet worden gebruikt om extra kleuren voor multikleur-sprites in te stellen (mode = 1).

RSPRITE() retourneert eigenschappen van individuele sprites die met het **SPRITE**-commando zijn ingesteld.

RSPRSYS() retourneert eigenschappen van het spritesysteem die met het **SPRITE SYS**-commando zijn ingesteld.

Voorbeeld: Gebruik van **SPRITE**

```
10 CLR : SCNCLR : SPRITE CLR
20 SPRITE LOAD "DEMOSPRITES1"
30 FOR I = 0 TO 7 : C = I : IF C = 6 THEN C = 8
40 MOVSPR I, 60 + 30 * I, 0 TO 60 + 30 * I, 65 + 20 * I, 3 : SPRITE I, 1, C,, 1, 1 : NEXT : SLEEP 3
50 FOR I = 0 TO 7 : SPRITE I,,,, 0, 0 : NEXT : SLEEP 3 : SPRITE CLR
60 FOR I = 0 TO 7 : MOVSPR I, 45 * I#5 : NEXT : FOR I = 0 TO 7 : SPRITE I, 1 : NEXT
70 FOR I = 0 TO 7 : X = 60 + 30 * I : Y = 65 + 20 * I : DO
80 LOOP UNTIL(X = RSPPOS(I, .)) AND (Y = RSPPOS(I, 1)) : MOVSPR I, .#. : NEXT
```

SPRSAV

Token: $FE $16

Formaat: **SPRSAV** bron**,** bestemming

Gebruik: Kopieert spritegegevens tussen twee sprites, of tussen een sprite en een stringvariabele.

bron sprite-nummer of stringvariabele.

bestemming sprite-nummer of stringvariabele.

Opmerkingen:

Bron en bestemming kunnen een sprite-nummer of een stringvariabele zijn,

SPRSAV kan alleen met de basisvorm van sprites (C64-compatibel) worden gebruikt. Deze sprites beslaan 64 bytes geheugen en produceren tekenreeksen van lengte 64 indien de bestemmingparameter een stringvariabele is.

Uitgebreide sprites en sprites met variabele hoogte kunnen niet met **SPRSAV** worden gebruikt.

Een tekenreeksarray van spritegegevens kan worden gebruikt om veel vormen op te slaan en snel naar het spritegeheugen te kopiëren met het commando **SPRSAV**.

Het is ook een handige methode om vormen van individuele sprites van of naar een schijfbestand te lezen of te schrijven.

Voorbeeld: Gebruik van **SPRSAV**

```
10 SPRITE LOAD "SPRITEDATA" : REM LAAD DATA VOOR 8 SPRITES
20 SPRITE 1, 1              : REM ZET SPRITE 1 AAN
30 SPRSAV 1, 2              : REM KOPIEER DATA VAN SPRITE 1 NAAR SPRITE 2
40 SPRITE 2, 1              : REM ZET SPRITE 2 AAN
50 SPRSAV 1, A$             : REM BEWAAR DATA VAN SPRITE 1 IN STRING A$
```

SQR

Token: $BA

Formaat: **SQR**(numerieke expressie)

Retourneert:
De vierkantswortel van een numerieke expressie.

Opmerkingen:
Het argument mag niet negatief zijn. Indien **SQR()** een negatieve waarde krijgt, wordt een Illegal Quantity-fout gegenereerd.

Voorbeeld: Gebruik van **SQR**

```
PRINT SQR(2)
 1.4142136
```

ST

Token: N.v.t.

Formaat: **ST**

Gebruik: De status van de laatste I/O-bewerking.

Indien **ST** nul is, trad er geen fout op; anders is deze ingesteld op een apparaat-afhankelijke foutcode.

Opmerkingen:

ST is een gereserveerde systeemvariabele.

Voorbeeld: Gebruik van **ST**

```
100 MX = 100 : DIM T$(MX)                          : REM DATA-ARRAY
110 DOPEN#1, "DATA"                                : REM OPEN BESTAND
120 IF DS THEN PRINT"KON NIET OPENEN" : STOP
130 LINE INPUT#1, T$(N) : N = N + 1                : REM LEES EEN RECORD
140 IF N > MX THEN PRINT "TE VEEL DATA" : GOTO 160
150 IF ST = 0 THEN 130                             : REM ST = 64 FOR END-OF-FILE
160 DCLOSE#1
170 PRINT N; " RECORDS GELEZEN"
```

STEP

Token: $A9

Formaat: **FOR** index = begin **TO** einde [**STEP** stap] ... **NEXT** [index]

Gebruik: **STEP** is een optioneel onderdeel van een **FOR**-lus.

De **index**-variabele kan na elke iteratie met een constante waarde worden verhoogd of verlaagd. Standaard wordt de variabele met 1 verhoogd. De indexvariabele moet een reële variabele zijn.

begin beginwaarde van de index.

einde wordt aan het einde van een iteratie gecontroleerd en bepaalt of er nog een iteratie wordt uitgevoerd of dat de lus wordt afgebroken.

stap bepaalt de wijziging die op het einde van een lusiteratie op de **index** wordt toegepast. Positieve stapwaarden verhogen de index, negatieve waarden verlagen deze. Standaard is 1.0 indien niet opgegeven.

Opmerkingen:

Bij positieve stappen moet **einde** groter zijn dan of gelijk aan **begin**. Bij negatieve stappen moet **einde** kleiner zijn dan of gelijk aan **begin**.

Het is een slechte programmeerpraktijk om de waarde van de **index**-variabele binnen de lus te wijzigen of met **GOTO** in of uit een lus te springen.

Voorbeeld: Gebruik van **STEP**

```
10 FOR D = 0 TO 360 STEP 30
20 R = D * π / 180
30 PRINT D; R; SIN(R); COS(R); TAN(R)
40 NEXT D
```

STOP

Token: $90

Formaat: **STOP**

Gebruik: Onderbreekt de uitvoering van het BASIC-programma.

Er wordt een bericht weergegeven met het regelnummer waarop het programma is gestopt. De READY-prompt verschijnt en de computer schakelt over naar de directe modus, in afwachting van toetsenbordinvoer.

Opmerkingen:

Alle variabeledefinities blijven na **STOP** geldig. Deze kunnen worden bekeken of gewijzigd, en het programma kan worden hervat met **CONT**. Elke bewerking van de programmaregels maakt verdere hervatting evenwel onmogelijk.

De programma-uitvoering kan worden hervat met **CONT**.

Voorbeeld: Gebruik van **STOP**

```
10 IF V < 0 THEN STOP : REM PROGRAMMA STOPT INDIEN NEGATIEF
20 PRINT SQR(V)       : REM PRINT VIERKANTSWORTEL
```

STR$

Token: $C4

Formaat: **STR$**(numerieke expressie)

Retourneert:
Een tekenreeks met de opgemaakte waarde van het argument, alsof deze met **PRINT** naar de tekenreeks is afgedrukt.

Opmerkingen:
Om een getal naar een decimale tekenreekswaarde om te zetten, zie **VAL**.

Voorbeeld: Gebruik van **STR$**

```
10 A$ = "DE WAARDE VAN PI IS " + STR$(π)
20 PRINT A$

RUN
DE WAARDE VAN PI IS 3.1415927
```

STRBIN$

Token: $CE $12

Formaat: **STRBIN$**(numerieke expressie)

Retourneert:
De getalwaarde als tekenreeks van de binaire representatie.

Opmerkingen:
Breukdelen worden genegeerd.

Om een binaire tekenreeks naar een getal om te zetten, zie **DECBIN**.

Voorbeeld: Gebruik van **STRBIN$**

```
PRINT STRBIN$(245)
11110101
```

SYS

Token: **SYS** : $9E
SYS TO : $9E $A4

Formaat: **SYS** adres [{**,** areg**,** xreg**,** yreg**,** zreg**,** sreg}]
SYS TO address [{**,A** areg**,X** xreg**,Y** yreg**,Z** zreg**,S** sreg **,I** BASIC-interruptmode **,O** MAPL-offset }]

Gebruik: Roept een machinetaal-subroutine aan.

adres startadres van de subroutine. Dit kan een in het ROM aanwezige KERNAL-routine zijn of elke andere routine die eerder is geladen of met **POKE** in het RAM is geplaatst.

areg waarde van de CPU-accumulator.

xreg waarde van het CPU X-register.

yreg waarde van het CPU Y-register.

zreg waarde van het CPU Z-register.

sreg waarde van het CPU-statusregister.

SYS laadt de argumenten (indien aanwezig) in registers, en aanroep daarna de subroutine. De aangeroepen routine moet met een **RTS**-instructie worden afgesloten. Nadat de subroutine is teruggekeerd, slaat hij de nieuwe registerinhoud op en geeft hij de besturing terug aan het BASIC-programma.

Indien de adreswaarde 16-bits is ($0000 - $FFFF), wordt de **BANK**-waarde gebruikt om het feitelijke adres te bepalen. Indien het adres hoger is dan $FFFF, wordt het geïnterpreteerd als een lineair 24-bits adres en wordt de waarde van **BANK** genegeerd.

SYS-vorm: Het **SYS**-commando aanvaardt het adres en optionele registerinhoud als positieargumenten, gescheiden door komma's. Als de waarde van een register niet is opgegeven, is die waarde ongedefinieerd. Zorg er dus voor dat u elke registerparameter instelt die de subroutine gebruikt.

In tegenstelling tot andere BASIC-commando's die geheugen benaderen, gelden er beperkingen op welke adressen **SYS** kan benaderen:

- **SYS** kan alleen banken 0 - 5 benaderen en heeft geen toegang tot zolder-RAM of het bovenste geheugen, zelfs niet met lange adressen.
- Alleen offsets $2000 - $7FFF binnen een gegeven bank verwijzen daadwerkelijk naar het geheugen van die bank.
- **SYS** kan alleen offsets $0000 - $1FFF in bank 0 benaderen.

- Het benaderen van offsets $8000 - $FFFF benadert altijd geheugen alsof **BANK** op 128 staat (inclusief ROM- en I/O-registermappingen), zelfs wanneer **BANK** op een andere bank staat of wanneer lange adressen worden gebruikt.

SYS TO-vorm:

Het **SYS TO**-commando aanvaardt het adres als eerste argument en de overige argumenten als benoemde argumenten. Deze vorm ondersteunt bovendien extra mogelijkheden ten opzichte van **SYS**, waaronder de mogelijkheid om code overal in de eerste 384KB van het geheugen uit te voeren.

Om een register in te stellen voor het aanroepen van de subroutine, geef het benoemde argument dat bij het register hoort op. Om bijvoorbeeld het X-register in te stellen en vervolgens een subroutine op adres $1600 aan te roepen: `SYS TO $1600,X($FF)`. De registerparameters heten A, X, Y, Z en S.

MAPL-offset Overschrijft het MAP-offsetregister voor 16-bit-adressen $2000 to $7FFF. De benoemde O-parameter accepteert een drie-nybble-waarde die met 256 wordt vermenigvuldigd om de offset te bepalen. Dit commando koppelt bijvoorbeeld $2000 - $7FFF TO $54000 - $59FFF, en aanroep vervolgens $3100 (+ $52000 = $55100): `SYS TO $3100,O($520)`

BASIC-interruptmode Een aanroepvorm waarbij BASIC tijdens de aanroep ingeschakeld blijft. Om deze modus in te schakelen, geef het argument I met waarde 1 op. In deze modus blijft BASIC ROM in het geheugen aanwezig en blijven interrupt-gestuurde functies ingeschakeld. Dit is nuttig voor het aanroepen van korte machinetaalroutines vanuit BASIC terwijl **PLAY** muziek afspeelt of **MOVSPR** sprites bewegen.

Een genoemd register kan worden gevolgd door een decimale waarde of door een numerieke expressie tussen haakjes. Om een hexadecimale waarde te gebruiken, plaats deze tussen haakjes.

Opmerkingen:

De registerwaarden na een **SYS**-aanroep worden in het systeemgeheugen opgeslagen. **RREG** kan worden gebruikt om deze waarden op te halen.

Ondanks de ongebruikelijke beperkingen op adressen is **SYS** een krachtige manier om BASIC- en machinetaal te combineren. Voor korte routines is geheugen in bank 0 op offsets $1800 - $1EFF beschikbaar voor programmeergebruik. Indien er zorgvuldig voor wordt gezorgd dat het einde van het BASIC-programma niet wordt overschreven, kunnen machinetaalroutines elders in bank 0 worden geladen, tot offset $BFFF.

Het correcte gebruik van **SYS** (d.w.z. zonder het systeem te storen) vereist enige technische vaardigheid. Zie voor meer informatie en voorbeelden Hoofdstuk/Bijlage 15 op bladzijde 15-3.

Voorbeeld: Gebruik van **SYS**

```
10 REM VERANDER DE RANDKLEUR MET SYS
20 BANK 0
30 POKE $4000, $EE, $20, $D0, $60   : REM MACHINETAAL VOOR INC $D020 : RTS
40 SYS $4000                        : REM ROEP SUBROUTINE AAN OP $4000 / BANK $00
50 GETKEY A$ : IF A$ <> "Q" THEN 40 : REM GA DOOR TOTDAT OP 'Q' IS GEDRUKT

REM : ROEP $1600 AAN MET HET Y-REGISTER GEZET OP $CC
SYS $1600,,,$CC

REM : ROEP $1600 AAN MET HET Y-REGISTER GEZET OP $CC
SYS TO $1600,Y($CC)

REM : ROEP $1600 AAN MET BASIC ZICHTBAAR
SYS TO $1600,I1

REM : $2000-$7FFF BEELDT HET GEHEUGEN AF VANAF $54000-$59FFF, DAN WORDT $3100 AANGEROEPEN
SYS TO $3100,0($520)
```

T@&

Token: N.v.t.

Formaat: **T@&**(kolom**,** rij**)**

Gebruik: Leest of stelt de schermcode van een teken op de gegeven schermcoördinaten in.

Dit gedraagt zich als een arrayvariabele. Het toewijzen van een waarde wijzigt de schermcode van het gegeven teken onmiddellijk.

Opmerkingen:

T@& is een gereserveerde systeemvariabele.

Schermcodes zijn niet hetzelfde als PETSCII-codes. Zie bijlage C op bladzijde C-3 voor een lijst met schermcodes.

De coördinaten beginnen bij (0, 0) voor de linkerbovenhoek. Indien de gegeven coördinaten buiten het huidige schermformaat vallen, geeft het benaderen van de array een Bad Subscript-fout.

Voorbeeld: Gebruik van **T@&** en **C@&**

```
10 PRINT CHR$(147)
20 FOR R = 0 TO 3
30 FOR C = 0 TO 63
40 T@&(C, R) = R * 64 + C
50 C@&(C, R)= MOD(C, 16)
60 NEXT C
70 NEXT R
80 CURSOR 0, 4
```

TAB

Token: $A3

Formaat: **TAB**(kolom)

Retourneert:

Plaatst de cursor op **kolom**.

Dit wordt alleen gedaan als de doelkolom zich *rechts* van de huidige cursorpositie bevindt, anders zal de cursor niet bewegen. De kolomtelling begint met 0 als de meest linkse kolom.

Opmerkingen:

Deze functie moet niet worden verward met TAB, die de cursor naar het volgende tabulatiepunt verplaatst.

Voorbeeld: Gebruik van **TAB**

```
10 FOR I = 1 TO 5
20 READ A$
30 PRINT "* " A$ TAB(10) " *"
40 NEXT I
50 END
60 DATA EEN, TWEE, DRIE, VIER, VIJF

RUN
* EEN      *
* TWEE     *
* DRIE     *
* VIER     *
* VIJF     *
```

TAN

Token: $C0

Formaat: **TAN**(expressie in radialen)
TAND(expressie in graden)

Retourneert:

De tangens van een hoek.

De **TAN**-functie verwacht een argument in radialen.

De **TAND**-variant verwacht een argument in graden.

Het resultaat ligt in het bereik (-1.0 tot +1.0).

Voorbeeld: Gebruik van **TAN**

```
PRINT TAN(0.7)
 0.84228838

X = 45 : PRINT TAN(X * π / 180)
 1

PRINT TAND(45)
 1
```

TEMPO

Token: $FE $05

Formaat: **TEMPO** snelheid

Gebruik: Stelt de weergavesnelheid in voor **PLAY**.

snelheid bereik van 1 - 255.

De duur (in seconden) van een hele noot wordt berekend met $duration = 24/speed$.

Voorbeeld: Gebruik van **TEMPO**

```
10 VOL 8, 8
20 FOR T = 24 TO 18 STEP -2
30 TEMPO T
40 PLAY "T0M3O4QGAGFED", "T2O4M5P0H.DP5GB", "T5O3IGAGAGAABABAB"
50 IF RPLAY(1) THEN GOTO 50
60 NEXT T
70 PLAY "T0O5QCO4GEH.C", "T2O5IEFEDEDCEGO6P8CP0R", "T5O3ICDCDEFEDCO4C"
```

THEN

Token: $A7

Formaat: **IF** expressie **THEN** waar-clausule [**ELSE** onwaar-clausule]

Gebruik: **THEN** is onderdeel van een **IF**-structuur.

expressie is een logische of numerieke expressie. Een numerieke expressie wordt als **FALSE** geëvalueerd indien de waarde nul is en als **TRUE** bij elke niet-nul waarde.

waar-clausule is één of meer instructies die direct na **THEN** op dezelfde regel beginnen. Een regelnummer na **THEN** voert in plaats daarvan een **GOTO** naar die regel uit.

onwaar-clausule is één of meer instructies die direct na **ELSE** op dezelfde regel beginnen. Een regelnummer na **ELSE** voert in plaats daarvan een **GOTO** naar die regel uit.

Opmerkingen:

De standaard **IF** ... **THEN** ... **ELSE**-structuur is beperkt tot één regel. De **waar-clausule** en **onwaar-clausule** kunnen evenwel worden uitgebreid naar meerdere regels met behulp van een samengestelde instructie tussen **BEGIN** ... **BEND**.

Voorbeeld: Gebruik van **THEN**

```
10 ROOD$ = CHR$(28) : ZWART$ = CHR$(144) : WIT$ = CHR$(5)
20 INPUT "GEEF EEN GETAL"; V
30 IF V < 0 THEN PRINT ROOD$; : ELSE PRINT ZWART$;
40 PRINT V                                  : REM PRINT NEGATIEVE GETALLEN IN HET ROOD
50 PRINT WIT$
60 INPUT "PROGRAMMA BEEINDIGEN (J/N)"; A$
70 IF A$ = "J" THEN END
80 IF A$ = "N" THEN 20 : ELSE 60
```

TI

Token: N.v.t.

Formaat: **TI**

Gebruik: Een hogesnelheidstimer-variabele met een resolutie van 1 microseconde.

Deze wordt gestart of gereset met **CLR TI** en kan op dezelfde manier als elke andere variabele in expressies worden benaderd.

Opmerkingen:

TI is een gereserveerde systeemvariabele.

De waarde in **TI** is het aantal seconden (met 7 decimalen) sinds de laatste reset of start.

Voorbeeld: Gebruik van **TI**

```
100 CLR TI                              : REM START TIMER
110 FOR I% = 1 TO 10000 : NEXT          : REM DOE IETS
120 ET = TI                             : REM SLA VERLOPEN TIJD OP IN ET
130 PRINT "LOOPTIJD:"; ET; " SECONDEN"
```

TI$

Token: N.v.t.

Formaat: **TI$**

Gebruik: De huidige tijd van de dag, als een tekenreeks.

De tijdwaarde wordt bijgewerkt door de RTC (Real-Time Clock). De tekenreeks **TI$** heeft het formaat HH:MM:SS.

TI$ is een read-only variabele die de registers van de RTC uitleest en de waarden formatteert tot een tekenreeks. Dit verschilt van andere Commodore-computers die geen RTC hebben.

Opmerkingen:

TI$ is een gereserveerde systeemvariabele.

Het is mogelijk om de registers van de RTC direct te benaderen via **PEEK**s. Het beginadres van de registers is $FFD7110 (FFD.7110).

Raadpleeg voor meer informatie over het instellen van de Real-Time Clock de sectie Configuratiehulpprogramma op bladzijde Hoofdstuk/-Bijlage 5 op bladzijde 5-3.

Voorbeelden:

Gebruik van **TI$**

```
100 REM ****** LEES RTC: ALLE WAARDEN ZIJN IN BCD ******
110 RT = $FFD7110                : REM ADRES VAN RTC
120 FOR I = 0 TO 5               : REM SS, MM, UU, DD, MA, JJ
130 T(I) = PEEK(RT + I)          : REM LEES REGISTERS
140 NEXT I                       : REM GEBRUIK ALLEEN LAATSTE TWEE CIJFERS
150 T(2) = T(2) AND 127          : REM VLAG VOOR 24-UURS MODE WEGHALEN
160 T(5) = T(5) + $2000          : REM 2000 BIJ JAAR OPTELLEN
170 FOR I = 2 TO 0 STEP -1       : REM TIJDINFORMATIE
180 PRINT USING ">## "; HEX$(T(I));
190 NEXT I
RUN
12 52 36
```

```
PRINT DT$;TI$
05-APR-2021 15:10:00
```

TO

Token: $A4

Formaat: trefwoord **TO**

Gebruik: **TO** is een secundair trefwoord dat in combinatie met primaire trefwoorden wordt gebruikt, zoals **BACKUP**, **BSAVE**, **CHANGE**, **CONCAT**, **COPY**, **FOR**, **GO**, **RENAME** en **SET DISK**.

Opmerkingen:
TO kan niet op zichzelf worden gebruikt.

Voorbeeld: Gebruik van **TO**

```
10 GO TO 1000           : REM ZOALS GOTO 1000

10 GOTO 1000            : REM KORTER EN SNELLER

10 FOR I = 1 TO 10      : REM TO IS DEEL VAN DE LUS
20 PRINT I : NEXT       : REM LUSEINDE

COPY "CODES" TO "BACKUP" : REM KOPIEER ENKEL BESTAND
```

TRAP

Token: $D7

Formaat: **TRAP** [regelnummer]

Gebruik: Registreert (of wist) een foutafhandelings-subroutine voor BASIC.

Wanneer een foutafhandelaar is geregistreerd, wordt er bij een fout in een BASIC-programma de subroutine aangeroepen in plaats van het programma te beëindigen. Tijdens de subroutine bevat de systeemvariabele **ER** het foutnummer. De **TRAP**-foutafhandelaar kan dan beslissen of de uitvoering moet worden gestopt (**STOP**) of hervat (**RESUME**).

TRAP zonder argument schakelt de foutafhandelaar uit; fouten worden dan afgehandeld door de normale systeemroutines.

Voorbeeld: Gebruik van **TRAP**

```
 10 TRAP 100
 20 FOR I = 1 TO 100
 30 PRINT EXP(I)
 40 NEXT
 50 PRINT "STOPPED FOR I ="; I
 60 END
100 PRINT ERR$(ER) : RESUME 50
```

TROFF

Token: $D9

Formaat: **TROFF**

Gebruik: Schakelt de traceermodus uit (ingeschakeld door **TRON**).

Wanneer de traceermodus actief is, wordt elk regelnummer afgedrukt voordat het wordt uitgevoerd. **TROFF** schakelt de traceermodus uit.

Voorbeeld: Gebruik van **TROFF**

```
10 TRON             : REM ACTIVEER TRACEMODUS
20 FOR I = 85 TO 100
30 PRINT I; EXP(I)
40 NEXT
50 TROFF            : REM DEACTIVEER TRACEMODUS

RUN
[10][20][30] 85  8.2230125E36
[40][30] 86  2.2352466E37
[40][30] 87  6.0760302E37
[40][30] 88  1.6516362E38
[40][30] 89
?OVERFLOW ERROR IN 30
READY.
```

TRON

Token: $D8

Formaat: **TRON**

Gebruik: Schakelt de traceermodus in.

Wanneer de traceermodus actief is, wordt elk regelnummer afgedrukt voordat het wordt uitgevoerd. **TRON** schakelt de traceermodus in.

Dit is nuttig voor het debuggen van de besturingsstroom van een BASIC-programma. Voeg **TRON**- en **TROFF**-opdrachten toe aan het programma rond de regels die gedebugd moeten worden.

Voorbeeld: Gebruik van **TRON**

```
10 TRON             : REM ACTIVEER TRACEMODUS
20 FOR I = 85 TO 100
30 PRINT I; EXP(I)
40 NEXT
50 TROFF            : REM DEACTIVEER TRACEMODUS

RUN
[10][20][30] 85  8.2230125E36
[40][30] 86  2.2352466E37
[40][30] 87  6.0760302E37
[40][30] 88  1.6516362E38
[40][30] 89
?OVERFLOW ERROR IN 30
READY.
```

TYPE

Token: $FE $27

Formaat: **TYPE** [**P**] bestandsnaam [**,D** station] [**,U** eenheid]

Gebruik: Drukt de inhoud van een bestand af dat tekst bevat gecodeerd als PETSCII.

Indien de **P**-vlag is opgegeven, pauzeert de weergave bij elke schermvol tekst. Druk op **Q** om de paginamodus te beëindigen; elke andere toets gaat verder naar de volgende pagina.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

TYPE kan niet worden gebruikt om BASIC-programma's af te drukken. Gebruik **LIST** voor programma's. **TYPE** kan alleen **SEQ**-bestanden verwerken die PETSCII-tekst bevatten, gescheiden door het CR-teken (carriage return). Het CR-teken kan naar een bestand worden geschreven met **CHR$(13)**. **TYPE** opent het bestand als een **SEQ**-bestand. Gebruik van **TYPE** met een bestand van een ander type geeft een schijffout 64,FILE TYPE MISMATCH.

Zie het **EDIT**-commando voor een manier om tekstbestanden interactief te maken en aan te passen.

Voorbeelden:

Gebruik van **TYPE**

```
TYPE "README"

TYPE "README.LST", U9

TYPE P "MOBYDICK"
```

UNLOCK

Token: $FE $4F

Formaat: **UNLOCK** bestandsnaam/patroon [**,D** station] [**,U** eenheid]

Gebruik: Ontgrendelt een bestand op de schijf dat voorheen is vergrendeld met **LOCK**.

Het opgegeven bestand of de set bestanden die aan het patroon voldoen, wordt ontgrendeld en is niet meer beschermd. Het kan daarna worden verwijderd met de commando's **DELETE**, **ERASE** of **SCRATCH**.

bestandsnaam de naam van een bestand. Ofwel een aangehaalde tekenreeks zoals "DATA", of een tekenreeksexpressie in haakjes zoals: (FI$)

station stationnummer in dubbelschijfstations. Het stationnummer is standaard **0** en kan worden weggelaten bij enkelschijfstations zoals de 1541, 1571 of 1581.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

Het ontgrendelen van een bestand dat al ontgrendeld is, heeft geen effect.

In de directe modus wordt het aantal ontgrendelde bestanden afgedrukt. De voorlaatste waarde uit het bericht bevat het aantal ontgrendelde bestanden.

Voorbeelden:

Gebruik van **UNLOCK**

```
UNLOCK "SNOOPY", U9     : REM ONTGRENDEL BESTAND SNOOPY OP UNIT 9
03,FILES UNLOCKED,01,00

UNLOCK "BS*"            : REM ONTGRENDEL ALLE BESTANDEN BEGINNEND OP "BS"
03,FILES UNLOCKED,04,00
```

UNTIL

Token: $FC

Formaat: **DO** ... **LOOP**
DO [<**UNTIL** | **WHILE**> logische expressie] ... opdrachten [**EXIT**]
LOOP [<**UNTIL** | **WHILE**> logische expressie]

Gebruik: **DO** ... **LOOP** definieert de reikwijdte van een BASIC-lus. Wanneer **DO** en **LOOP** zonder modifiers worden gebruikt, ontstaat er een oneindige lus die alleen kan worden afgebroken met **EXIT**. De lus kan worden beheerd door **UNTIL** of **WHILE** toe te voegen na **DO** of **LOOP**.

Opmerkingen:
DO-lussen kunnen genest worden. Een **EXIT**-opdracht beëindigt alleen de huidige lus.

Voorbeelden:
Gebruik van **UNTIL**

```
10 PW$ = "" : DO
20 GET A$ : PW$ = PW$ + A$
30 LOOP UNTIL LEN(PW$) > 7 OR A$ = CHR$(13)

10 DO
20 GET A$
30 LOOP UNTIL A$ = "Y" OR A$ = "N"
```

USING

Token: $FB

Formaat: **PRINT**[**#** kanaal**,**] **USING** indeling**;** argument

Gebruik: Ontleedt de **formaat**-tekenreeks en evalueert het argument. Het argument kan een tekenreeks of een numerieke waarde zijn. Het formaat van de resulterende uitvoer wordt bepaald door de **formaat**-tekenreeks.

Het **kanaal**nummer, dat eerder is opgegeven bij een aanroep van commando's zoals **APPEND**, **DOPEN** of **OPEN**. Indien geen kanaal wordt opgegeven, gaat de uitvoer naar het scherm.

Een **indeling**-tekenreeksvariabele of een tekenreeksconstante die de opmaakregels vastlegt. Bij het gebruik van een getal als **argument** kan het patroon worden opgegeven als een sjabloon zoals `###.##` of met een %[0]<width>.<precision>-specificatie, zoals `%3D %7.2F %4X`. Als een breedtespecificatie een voorlopende nul heeft, wordt een positief getal links met nullen aangevuld tot aan de decimale punt: `%07.2F`

Het **argument**-nummer dat moet worden opgemaakt. Als het argument niet in het formaat past, bijvoorbeeld bij het afdrukken van een variabele met vier cijfers in een reeks van drie haken (###), worden in plaats daarvan sterretjes gebruikt.

Opmerkingen:

De indelingstekenreeks wordt slechts op één argument toegepast, maar het is mogelijk om meer dan één **USING indeling;argument**-volgorde aan elkaar te koppelen.

Het **argument** kan afdrukbare tekens en controlcodes bevatten. Afdrukbare tekens worden op de cursorpositie weergegeven, terwijl controlcodes worden uitgevoerd.

Het aantal #-tekens bepaalt de breedte van de uitvoer. Indien het eerste teken van de indelingstekenreeks een '=' (gelijkteken) is, wordt het argument gecentreerd. Indien het eerste teken een '>' (groter-dan-teken) is, wordt het argument rechts uitgelijnd.

Hoewel het breedte-precisieformaatpatroon lijkt op de `printf()`-functie van de programmeertaal C, gedraagt het zich niet helemaal hetzelfde. Het breedteargument is het totale aantal cijfers in het patroon. De punt en het optionele precisieargument geven aan dat een decimale punt moet worden gebruikt, met het opgegeven aantal precisiecijfers rechts van de punt. `%7.2F` is equivalent aan: `#####.##`

Het opvullen van links met nullen is alleen beschikbaar met een breedte-precisiepatroon.

Voorbeelden:

USING met een bijbehorende **PRINT#**

```
PRINT USING "##.##"; 4, USING " [%7.4F] "; SQR(2)
 3.14 [  1.4142]

PRINT USING "##.##"; 4, USING " [%07.4F] "; SQR(2)
 3.14 [001.4142]

PRINT USING " ( # # # ) "; 12 * 31
 ( 3 7 2 )

PRINT USING "###"; "ABCDE"
ABC

PRINT USING ">###"; "ABCDE"
CDE

PRINT USING "ADDRESS: $%4X"; 65000
ADDRESS: $FDE8

A$ = "####,####,####.#":PRINT USING A$; 1E7 / 3
  3,333,333.3
```

USR

Token: $B7

Formaat: **USR**(numerieke expressie)

Gebruik: Roept een assembleertaal-routine aan waarvan het geheugenadres is opgeslagen op $02F8 - $02F9.

Het resultaat van de **numerieke expressie** wordt geschreven naar floating-point-accumulator 1.

Na het uitvoeren van de assembleroutine geeft BASIC de inhoud van floating-point-accumulator 1 terug.

Opmerkingen:

Banken 0 - 127 geven toegang tot RAM- of ROM-banken. Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enzovoort.

De floating-point-accumulator is een voorziening van de KERNAL.

Voorbeeld: Gebruik van **USR**

```
10 WPOKE $2F8, $7F33 : REM NEGATIEROUTINE
20 PRINT USR(π)
30 PRINT USR(-5)

RUN
-3.1415927
 5
```

VAL

Token: $C5

Formaat: **VAL**(tekenreeksexpressie)

Retourneert:
Het decimale floating point getal dat door een tekenreeks wordt weergegeven.

Opmerkingen:
VAL verwerkt tekens vanaf het begin van de tekenreeks die lijken op een BASIC-decimaal getal, inclusief een optioneel minteken, cijfers, een decimaal scheidingsteken en een exponent. Indien een ongeldig teken wordt aangetroffen, stopt de verwerking en wordt het resultaat tot aan dat punt in de tekenreeks geretourneerd. Indien het eerste teken ongeldig is, wordt 0 geretourneerd.

Om een getal naar een tekenreeks om te zetten, zie **STR$**. Indien u alleen een getal wilt afdrukken, onthoud dan dat **PRINT** numerieke expressies accepteert zonder dat een conversie naar een tekenreeks nodig is.

Voorbeeld: Gebruik van **VAL**

```
PRINT VAL("78E2")
 7800

PRINT VAL("7+5")
 7

PRINT VAL("1.256")
 1.256

PRINT VAL("$FFFF")
 0
```

VERIFY

Token: $95

Formaat: **VERIFY** bestandsnaam [**,** eenheid [**,** binvlag]]

Gebruik: **VERIFY** zonder **binvlag** vergelijkt een BASIC-programma in het geheugen met een schijfbestand van het type **PRG**. Het doet hetzelfde als **DVERIFY**, maar de syntaxis verschilt.

VERIFY met **binvlag** vergelijkt een binair bestand in het geheugen met een schijfbestand van het type **PRG**. Het doet hetzelfde als **BVERIFY**, maar de syntaxis verschilt.

bestandsnaam is een tekenreeks tussen aanhalingstekens, bijv. "PROG" of een tekenreeksexpressie.

eenheid apparaatnummer op de IEC-bus. Meestal van 8 t/m 11 voor schijfstations. Indien een variabele wordt gebruikt, moet deze tussen haakjes worden geplaatst. Het standaard eenheidsnummer is **8**.

Opmerkingen:

VERIFY kan alleen op gelijkheid testen. Het geeft geen informatie over het aantal of de positie van verschillende bytes. **VERIFY** eindigt met het bericht OK of met een Verify-fout.

VERIFY is verouderd in BASIC 65. Het is alleen geïmplementeerd voor achterwaartse compatibiliteit. Het wordt aanbevolen om **DVERIFY** en **BVERIFY** te gebruiken in plaats hiervan.

Voorbeelden:

Gebruik van **VERIFY**

```
VERIFY "ADVENTURE"

VERIFY "ZORK-I", 9

VERIFY "1:DUNGEON", 10
```

VIEWPORT

Token: $FE $31

Formaat: **VIEWPORT CLR**
VIEWPORT DEF x, y, breedte, hoogte

Gebruik: Bitmapgrafiek: beheert de viewport van een scherm.

VIEWPORT DEF definieert een knipgebied met de oorsprong (bovenste linker positie) ingesteld op **x, y** en de **breedte** en **hoogte**. Alle volgende grafiekcommando's zijn beperkt tot het **VIEWPORT**-gebied.

VIEWPORT CLR vult het knipgebied met de kleur van de tekenpen.

Opmerkingen:
Het knipgebied kan worden gereset naar het volledige scherm met het commando `VIEWPORT DEF 0, 0, WIDTH, HEIGHT` met dezelfde waarden voor breedte en hoogte als in het **SCREEN**-commando.

Voorbeeld: Gebruik van **VIEWPORT**

```
10 SCREEN 320, 200, 2
20 VIEWPORT DEF 20, 30, 100, 120 : REM GEBIED 20->119, 30->149
30 PEN 1                         : REM KIES KLEUR 1
40 VIEWPORT CLR                  : REM VUL GEBIED MET PENKLEUR
50 GETKEY A$                     : REM WACHT OP TOETSDRUK
60 SCREEN CLOSE
```

VOL

Token: $DB

Formaat: **VOL** right, left

Gebruik: Stelt het volume in voor de geluidsuitvoer met **SOUND** of **PLAY**. De waarde varieert van 0 (uit) tot 15 (hardst).

rechts volume voor SID's 1 en 2.

links volume voor SID's 3 en 4.

Opmerkingen:

De termen "rechts" en "links" verwijzen naar de standaard pan-instellingen voor de SID-chips in de audiomixer. Het werkelijke volume en de panpositie voor elk paar SID's hangt af van de huidige instellingen van de audiomixer. U kunt de audio-instellingen aanpassen in het Freezer-menu.

Voorbeeld: Gebruik van **VOL**

```
10 TEMPO 22
20 FOR V = 2 TO 12 STEP 2
30 VOL V, 16-V
40 PLAY "T0M3O4QGAGFED", "T2O4M5P0H.DP5GB", "T5O3IGAGAGAABABAB", "G"
50 IF RPLAY(1) THEN GOTO 50
60 NEXT V
70 PLAY "T0O5QCO4GEH.C", "T2O5IEFEDEDCEGO6P9CP0R", "T5O3ICDCDEFEDCO4C", "C"
```

VSYNC

Token: $FE $54

Formaat: **VSYNC** rasterlijn

Gebruik: Wacht tot de geselecteerde rasterlijn actief is.

rasterlijn PAL (0 - 311), NTSC (0 - 261).

Dit onderbreekt de uitvoering van het BASIC-programma totdat de schermupdate de opgegeven verticale pixelcoördinaat bereikt. Het betreft een zeer korte onderbreking: het scherm wordt 50 keer per seconde bijgewerkt bij PAL en 60 keer per seconde bij NTSC. Dit is nuttig om grafische parameters op specifieke momenten in de schermupdate aan te passen en de programmalogica van BASIC te synchroniseren met de verversingssnelheid van het scherm.

Voorbeeld: Gebruik van **VSYNC**

```
10 BORDER 3  : REM VERANDER RANDKLEUR IN CYAAN
20 VSYNC 100 : REM WACHT OP RASTERLIJN 100
30 BORDER 7  : REM VERANDER RANDKLEUR IN GEEL
40 VSYNC 260 : REM WACHT OP RASTERLIJN 260
50 GOTO 10   : REM LOOP
```

WAIT

Token: $92

Formaat: **WAIT** adres**,** enmasker [**,** xormasker]

Gebruik: Onderbreekt het BASIC-programma totdat het gevraagde bitpatroon van het opgegeven adres is gelezen.

adres adres in de huidige geheugenbank dat wordt gelezen.

enmasker masker dat via **AND** wordt toegepast.

xormasker masker dat via **XOR** wordt toegepast.

WAIT leest de byte-waarde van **adres** en past de maskers toe:
result = PEEK(adres) AND enmasker XOR xormasker.

De onderbreking eindigt als het resultaat ongelijk is aan nul; anders wordt de leesactie herhaald. Indien de voorwaarde niet wordt bereikt, kan de computer voor onbepaalde tijd vastlopen.

Opmerkingen:

WAIT wordt doorgaans gebruikt om hardware-registers of systeemvariabelen te onderzoeken en te wachten op een gebeurtenis, bijv. een joystick-gebeurtenis, muisevenement, toetsaanslag of wanneer een bepaalde rasterlijn op het scherm zal worden getekend.

Voorbeeld: Gebruik van **WAIT**

```
10 BANK 128
20 WAIT 54801, 1 : REM WACHT TOTDAT OP DE LINKER-SHIFT-TOETS WORDT GEDRUKT
```

WHILE

Token: $FD

Formaat: **DO** ... **LOOP**
DO [<**UNTIL** | **WHILE**> logische expressie] ... opdrachten [**EXIT**]
LOOP [<**UNTIL** | **WHILE**> logische expressie]

Gebruik: **DO** ... **LOOP** definieert de reikwijdte van een BASIC-lus. Wanneer **DO** en **LOOP** zonder modifiers worden gebruikt, ontstaat er een oneindige lus die alleen kan worden afgebroken met **EXIT**. De lus kan worden beheerd door **UNTIL** of **WHILE** toe te voegen na **DO** of **LOOP**.

Opmerkingen:
DO-lussen kunnen genest worden. Een **EXIT**-opdracht beëindigt alleen de huidige lus.

Voorbeelden:
Gebruik van **WHILE**

```
10 DO WHILE ABS(EPS) > 0.001
20 GOSUB 2000              : REM ITERATIESUBROUTINE
30 LOOP

10 I% = 0                  : REM INTEGERLUS 1-100
20 DO I% = I% + 1
30 LOOP WHILE I% < 101
```

WINDOW

Token: $FE $1A

Formaat: **WINDOW** links**,** boven**,** rechts**,** onder[**,** wissen]

Gebruik: Bepaalt de positie en grootte van het venster op het tekstscherm.

links linkerkolom.

boven bovenste rij.

rechts rechterkolom.

onder onderste rij.

wissen vlag voor het wissen van het tekstvenster.

Standaard vinden tekstupdates plaats op het volledige beschikbare tekstscherm. **WINDOW** beperkt het updategebied tot een rechthoek van de beschikbare schermruimte.

Opmerkingen:

De rijwaarden lopen van 0 tot 24. De kolomwaarden lopen van 0 tot respectievelijk 39 of 79, afhankelijk van de schermmodus.

Er kan slechts één venster op het scherm aanwezig zijn. Tweemaal op CLR HOME drukken of **PRINT**en van **CHR$(19)CHR$(19)** herstelt het venster naar de standaardinstelling (volledig scherm).

Voorbeelden:

Gebruik van **WINDOW**

```
WINDOW  0,  1, 79, 24   : REM SCHERM ZONDER TOPREGEL

WINDOW  0,  0, 79, 24, 1 : REM VENSTER OP VOLLEDIG SCHERM LEEGGEMAAKT

WINDOW  0, 12, 79, 24   : REM ONDERSTE HELFT VAN SCHERM

WINDOW 20,  5, 59, 15   : REM KLEIN GECENTREERD VENSTER
```

WPEEK

Token: $CE $10

Formaat: **WPEEK**(adres)

Retourneert:

De 16-bits woordwaarde die in het geheugen is opgeslagen op **adres** (lage byte) en **adres**+1 (hoge byte), als een natuurlijk getal van 16 bit.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

Opmerkingen:

Banken 0 - 127 geven toegang tot RAM- of ROM-banken. Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enzovoort.

Voorbeeld: Gebruik van **WPEEK**

```
10 UA = WPEEK($02F8)                    : REM DOELADRES VAN USR-JUMP
20 PRINT "DOELADRES USR-FUNCTIE"; UA
```

WPOKE

Token: $FE $1D

Formaat: **WPOKE** adres**,** woord [**,** woord ...]

Retourneert:

Schrijft een of meer 16-bits woorden in het geheugen of geheugengekoppelde I/O, beginnend bij **adres**.

Indien het adres in het bereik van $0000 tot $FFFF (0 - 65535) valt, wordt de geheugenbank gebruikt die door **BANK** is ingesteld.

Adressen groter dan of gelijk aan $10000 (decimaal 65536) worden beschouwd als vlakke geheugenadressen en als zodanig gebruikt, waarbij de instelling van **BANK** wordt genegeerd.

woord waarde van 0 - 65535. Het eerste woord wordt opgeslagen op adres (lage byte) en adres+1 (hoge byte). Het tweede woord wordt opgeslagen op adres+2 (lage byte) en adres+3 (hoge byte), enzovoort. Indien een waarde groter is dan 65535, worden alleen de twee laagste bytes gebruikt.

Opmerkingen:

Het adres wordt met twee verhoogd voor elk datawoord, zodat een geheugenbereik met één enkele **WPOKE**-opdracht kan worden geschreven.

Banken groter dan 127 worden gebruikt om toegang te krijgen tot I/O en de onderliggende systeemhardware zoals de VIC, SID, FDC enz.

Voorbeeld: Gebruik van **WPOKE**

```
10 BANK 128          : REM KIES SYSTEEMBANK
20 WPOKE $02F8, $1800 : REM ZET USR-VECTOR OP $1800
```

XOR

Token: $E9

Formaat: operand **XOR** operand

Gebruik: Voert een bitsgewijze logische Exclusive OR-bewerking uit op twee 16-bits waarden.

Operanden van het type geheel getal worden ongewijzigd gebruikt. Reële operanden worden omgezet naar een geheel getal van 16 bit (waarbij precisie verloren gaat). Logische operanden worden omgezet naar een geheel getal van 16 bit, waarbij $FFFF (decimaal -1) voor TRUE en $0000 (decimaal 0) voor FALSE wordt gebruikt.

Expressie	Resultaat
0 XOR 0	0
0 XOR 1	1
1 XOR 0	1
1 XOR 1	0

Opmerkingen:

Het resultaat is van het type geheel getal. Indien het resultaat in een logische context wordt gebruikt, wordt de waarde 0 als ONWAAR beschouwd en alle andere niet-nul-waarden als WAAR.

Voorbeeld: Gebruik van **XOR**

```
FOR I = 0 TO 8 : PRINT I XOR 5; : NEXT I
 5  4  7  6  1  0  3  2  13
```

BIJLAGE C

Schermcodes

- Schermcodes

SCHERMCODES

Een tekstteken wordt in het schermgeheugen weergegeven door een schermcode. Er bestaan 256 mogelijke schermcodes, die elk verwijzen naar een afbeelding in de huidige tekenset.

Een volledige tekenset bevat twee groepen van 256 afbeeldingen: één voor de hoofdlettermodus en één voor de kleinelettermodus, derhalve 512 afbeeldingen in totaal. Slechts één modus kan tegelijk worden weergegeven. De ingebouwde tekensets gebruiken de eerste 128 tekens van elke groep voor normale tekens en de volgende 128 voor de geïnverteerde versies van dezelfde tekens.

In BASIC biedt de speciale array **T@&()** toegang tot de tekens op het scherm via kolom- en rij-indexen. De waarden in deze speciale array zijn schermcodes. Het **FONT**-commando schakelt tussen de ingebouwde tekensets. Het **CHARDEF**-commando wijzigt de afbeelding die bij een schermcode hoort.

OPMERKING: Schermcodes verschillen van PETSCII-codes. PETSCII-codes worden gebruikt om tekstgegevens op te slaan, te verzenden en te ontvangen en om te bepalen hoe strings op het scherm worden weergegeven. Wanneer een PETSCII-teken op het scherm wordt weergegeven, wordt de bijbehorende schermcode in het schermgeheugen geschreven. Voor een overzicht van PETSCII-codes, zie bijlage D op bladzijde D-3.

De volgende tabel geeft een overzicht van de schermcodes. Indien een code afhankelijk van de modus een ander teken oplevert, wordt het teken vermeld als "hoofdletters / kleine letters".

0 @	**14** N / n	**28** £	**42** *
1 A / a	**15** O / o	**29**]	**43** +
2 B / b	**16** P / p	**30** ↑	**44** ,
3 C / c	**17** Q / q	**31** ←	**45** -
4 D / d	**18** R / r	**32** spatie	**46** .
5 E / e	**19** S / s	**33** !	**47** /
6 F / f	**20** T / t	**34** "	**48** 0
7 G / g	**21** U / u	**35** #	**49** 1
8 H / h	**22** V / v	**36** $	**50** 2
9 I / i	**23** W / w	**37** %	**51** 3
10 J / j	**24** X / x	**38** &	**52** 4
11 K / k	**25** Y / y	**39** '	**53** 5
12 L / l	**26** Z / z	**40** (	**54** 6
13 M / m	**27** [	**41**)	**55** 7

56 8
57 9
58 :
59 ;
60 <
61 =
62 >
63 ?
64 ⊟
65 ♠ / A

66 ◫ / B
67 ⊟ / C
68 ⊟ / D
69 ◫ / E
70 ⊟ / F
71 ◫ / G
72 ◫ / H
73 ◱ / I
74 ◹ / J
75 ◰ / K

76 ◻ / L
77 ◹ / M
78 ◰ / N
79 ◻ / O
80 ❒ / P
81 ● / Q
82 ▭ / R
83 ♥ / S
84 ◧ / T
85 ◪ / U

86 ☒ / V
87 ◙ / W
88 ♣ / X
89 ◨ / Y
90 ◆ / Z
91 ⊞
92 ◩
93 ◫
94 π / ▩

OPMERKING: In de ingebouwde tekensets zijn codes 128–255 geïnverteerde versies van codes 0–127.

BIJLAGE D

PETSCII-codes

- PETSCII-codes

PETSCII-CODES

PETSCII is de tekencodering die Commodore-8-bitcomputers gebruiken voor gegevens. De codering omvat afdrukbare tekens zoals letters, cijfers, leestekens en grafische symbolen; Controlcodes om de weergaveparameters te wijzigen en de cursor van de schermeditor te bedienen; en voorstellingen van toetsaanslagen die via het toetsenbord kunnen worden ingevoerd. Een programma kan PETSCII-codes gebruiken voor uitvoer naar het scherm, voor invoer via het toetsenbord of voor het lezen en schrijven van tekengegevens in bestanden of via seriële communicatie.

In BASIC aanvaardt de functie **CHR$()** een PETSCII-code en geeft die code als string terug. Zo drukt `PRINT CHR$(65)` de letter A af. De functie **ASC()** aanvaardt een string van één teken en geeft het getal van de PETSCII-code terug. De commando's **GET** en **GETKEY** lezen een toetsaanslag van het toetsenbord en geven de bijbehorende PETSCII-code terug.

Machinetaalprogramma's kunnen de GETIN-routine van de KERNAL of het PETSCIIKEY-hardwareregister gebruiken om toetsenbordinvoer als PETSCII-codes te lezen.

0

1 Alternatief palet

2 Onderstrepen aan

3

4 Standaardpalet

5 Wit

6

7 Bel

8

9 TAB

10 Nieuwe regel

11 Schakelt SHIFT + ▚ uit

12 Schakelt SHIFT + ▚ in

13 RETURN

14 Kleine letters

15 Knipperen aan

16 F9

17 ↓

18 RVS ON

19 CLR HOME

20 INST DEL

21 F10 / Vorig woord

22 F11

23 F12 / Volgend woord

24 Tab instellen/wissen

25 F13

26 F14 / Vorig tabpunt

27 ESCAPE

28 Rood

29 →

30 Groen

31 Blauw

32 SPACE

33 !

34 "

35 #

36 $

37 %

38 &

39 '

40 (

41)

42 *

43 +

44 ,

45 -

46 .

47 /

48 0

49 1

50 2

51 3

52 4

53 5

54 6

55 7

56 8

57 9

58 :

59 ;

60 <

61 =

62 >

63 ?

64 @

65 A

66 B

67 C

68 D

69 E

70 F

71 G

72 H
73 I
74 J
75 K
76 L
77 M
78 N
79 O
80 P
81 Q
82 R
83 S
84 T
85 U
86 V
87 W
88 X
89 Y
90 Z
91 [
92 £
93]
94 ↑
95 ←

De codes 96 tot en met 127 drukken tekens af die vergelijkbaar zijn met die van de codes 192 tot en met 223. Zij worden niet als toetsenbordinvoer teruggegeven.

128
129 Oranje
130 Onderstrepen uit
131 SHIFT RUN STOP
132 HELP
133 F1
134 F3
135 F5
136 F7
137 F2
138 F4
139 F6
140 F8
141 SHIFT RETURN
142 Hoofdletters
143 Knipperen uit
144 Zwart
145 ↑
146 RVS OFF
147 SHIFT CLR HOME
148 SHIFT INST DEL
149 Bruin
150 Lichtrood (roze)
151 Donkergrijs
152 Middengrijs
153 Lichtgroen
154 Lichtblauw
155 Lichtgrijs
156 Paars
157 ←
158 Geel
159 Cyaan
160 SPACE
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

216 ♣	**218** ♦	**220** ▌	**222** π
217 ♥	**219** ♠	**221** ▐	**223** ▄

De codes 224 tot en met 254 drukken tekens af die vergelijkbaar zijn met die van de codes 160 tot en met 190. Zij worden niet als toetsenbordinvoer teruggegeven.

De codes 126, 222 en 255 drukken alle drie het pi-symbool af (π). Wanneer u dit teken typt, is de toetsenbordinvoercode 222.

Wanneer u de kleineletters-/hoofdlettermodus gebruikt (door + SHIFT in te drukken), let dan op het volgende:

- De hoofdletters in het gebied 65-90 van bovenstaande tabel worden vervangen door kleine letters.
- De grafische tekens in het gebied 97-122 van bovenstaande tabel worden vervangen door hoofdletters.
- De kleine letters van PETSCII (65-90) en de hoofdletters (97-122) vallen in het ASCII-gebied voor respectievelijk hoofdletters (65-90) en kleine letters (97-122).
- Verscheidene symbolen hebben alternatieve symbolen in het lettertype voor kleine en hoofdletters, met name de PETSCII-codes 126, 127, 168 en 183.

BIJLAGE E

Toetsen van de schermeditor

- Toetsen van de schermeditor
- Control-codes
- Shiftcodes
- Escape-sequenties

TOETSEN VAN DE SCHERMEDITOR

De volgende toetscombinaties voeren handelingen uit in de schermeditor van de MEGA65.

In sommige gevallen kan een programma de equivalente PETSCII-codes weergeven om dezelfde handelingen uit te voeren. Zo kan een programma CTRL + G, welke een beltoon voortbrengt, als CHR$(7) weergeven. Gebruik voor een ESC-sequentie CHR$(27) om de ESC-toets weer te geven, gevolgd door de volgende toets van de reeks.

CONTROL-CODES

Toetsencombinatie	Functie
Kleuren	
CTRL + 1 tot 8	Kiezen uit het eerste kleurbereik. Zie bijlage G op bladzijde G-3 voor de lijst van kleuren in het systeempalet.
⌘ + 1 tot 8	Kiezen uit het tweede kleurbereik.
CTRL + E	Herstelt de kleur van de cursor naar de standaardkleur (wit).
CTRL + D	Schakelt de VIC-IV over naar kleurbereik 0-15 (standaardkleuren). Deze kleuren zijn bereikbaar met CTRL en de toetsen 1 tot 8 (voor de eerste 8 kleuren) of met ⌘ en de toetsen 1 tot 8 (voor de overige 8 kleuren).
CTRL + A	Schakelt de VIC-IV over naar kleurbereik 16-31 (alternatieve/regenboog-kleuren). Deze kleuren zijn bereikbaar met CTRL en de toetsen 1 tot 8 (voor de eerste 8 kleuren) of met ⌘ en de toetsen 1 tot 8 (voor de overige 8 kleuren).

Toetsencombinatie	Functie
Tabposities	
CTRL + Z	Verplaatst de cursor via een tab naar links. Indien er geen tabposities meer resteren, blijft de cursor aan het begin van de regel staan.
CTRL + I	Verplaatst de cursor via een tab naar rechts. Indien er geen tabposities meer resteren, blijft de cursor aan het einde van de regel staan.
CTRL + X	Stelt de huidige schermkolom als tabpositie in of wist deze. Gebruik CTRL + Z en I om heen en weer te springen naar alle posities die met X zijn ingesteld.
Verplaatsen	
CTRL + Q	Verplaatst de cursor telkens één regel omlaag. Gelijk aan ↓.
CTRL + J	Verplaatst de cursor één positie omlaag. Indien u zich op een lange regel BASIC-code bevindt die over twee regels doorloopt, verplaatst de cursor zich twee rijen omlaag om op de volgende regel te komen.
CTRL +]	Gelijk aan →.
CTRL + T	Wist het teken direct links van de cursor en schuift alle tekens die rechts daarvan staan één positie naar links. Dit is gelijk aan INST DEL.
CTRL + M	Voert een regelterugloop uit, gelijk aan RETURN.
Verplaatsen per woord	

Toetsencombinatie	Functie
CTRL + U	Verplaatst de cursor terug naar het begin van het vorige woord. Indien er op de huidige regel geen vorig woord staat, verplaatst de cursor zich naar de eerste kolom van de huidige regel en vervolgens naar de vorige regel, totdat een regel met een woord wordt aangetroffen.
CTRL + W	Verplaatst de cursor vooruit naar het begin van het volgende woord. Indien er op de huidige regel geen volgend woord staat, verplaatst de cursor zich naar de eerste kolom van de volgende regel, totdat een regel met een woord wordt aangetroffen.
Scrollen	
CTRL + P	Schuift de BASIC-listing één regel omlaag. Gelijk aan F9.
CTRL + V	Schuift de BASIC-listing één regel omhoog. Gelijk aan F11.
CTRL + S	Gelijk aan NO SCROLL.
Opmaak	
CTRL + B	Schakelt de onderstrepingsmodus voor tekst in. U kunt de onderstrepingsmodus uitschakelen door ESC en vervolgens O in te drukken.
CTRL + O	Schakelt de flitsmodus voor tekst in. U kunt de flitsmodus uitschakelen door ESC en vervolgens O in te drukken.
Hoofdlettergebruik	
CTRL + N	Wijzigt het hoofdlettergebruik van de tekst van uitsluitend hoofdletters naar kleine letters.

Toetsencombinatie	Functie
CTRL + K	Vergrendelt de omschakeling tussen hoofdletters en kleine letters, welke gewoonlijk met ▙ + SHIFT wordt uitgevoerd.
CTRL + L	Schakelt de omschakeling tussen hoofdletters en kleine letters in, welke met ▙ + SHIFT wordt uitgevoerd.
Diversen	
CTRL + G	Brengt een beltoon voort.
CTRL + [	Gelijk aan het indrukken van ESC.
CTRL + *	Opent de Matrix Mode Debugger.

SHIFTCODES

Toetsencombinatie	Functie
SHIFT + INST DEL	Voegt op de huidige cursorpositie een teken in en verplaatst alle tekens één positie naar rechts.
SHIFT + HOME	Wist het volledige scherm en plaatst de cursor op de startpositie.

ESCAPE-SEQUENTIES

Om een escape-sequentie uit te voeren, drukt u ESC kort in en laat u deze weer los. Druk vervolgens één van de volgende toetsen in om de sequentie uit te voeren.

Toets	Sequentie
Gedrag van de editor	
ESC X	Wist het scherm en schakelt tussen de tekstmodi 40 × 25 en 80 × 25.
ESC 4	Wist het scherm en schakelt over naar de tekstmodus 40 × 25.
ESC 8	Wist het scherm en schakelt over naar de tekstmodus 80 × 25.
ESC 5	Schakelt over naar de tekstmodus 80 × 50.
Let op dat sommige programma's verwachten dat zij in de tekstmodus 80 × 25 worden gestart en zich mogelijk niet correct gedragen wanneer zij in de tekstmodus 80 × 50 worden gestart.	
ESC @	Wist een gedeelte van het scherm, vanaf de huidige cursorpositie tot het einde van het scherm.
ESC O	Schakelt de zinnebeeldmodus, de modus voor omgekeerde tekst, de onderstrepingsmodus en de flitsmodus uit.
Scrollen	
ESC V	Schuift het hele scherm één regel omhoog.
ESC W	Schuift het hele scherm één regel omlaag.
ESC L	Schakelt het scrollen in wanneer ↓ onderaan het scherm wordt ingedrukt.
ESC M	Schakelt het scrollen uit. Wanneer ↓ onderaan het scherm wordt ingedrukt, verplaatst de cursor zich naar de bovenkant van het scherm. Wanneer ↑ echter bovenaan het scherm wordt ingedrukt, blijft de cursor op de eerste regel staan.

Toets	Sequentie
ESC N	Schakelt het “regelduwen” in: typen of afdrukken in de meest rechtse kolom schuift de daaropvolgende regels één positie omlaag.
ESC R	Schakelt het “regelduwen” uit: typen of afdrukken in de meest rechtse kolom verplaatst de cursor naar het begin van de volgende regel, maar schuift geen regels omlaag. Schakel zowel het “regelduwen” (ESC R) als het scrollen (ESC M) uit om met **PRINT** in de meest rechtse kolom af te drukken zonder de rest van de weergave te verstoren.
Invoegen en verwijderen	
ESC I	Voegt een lege regel in op de huidige cursorpositie en schuift alle daaropvolgende regels één positie omlaag.
ESC D	Verwijdert de huidige regel en schuift de regels onder de cursor één positie omhoog.
ESC P	Wist alle tekens van de cursor tot aan het begin van de huidige regel.
ESC Q	Wist alle tekens van de cursor tot aan het einde van de huidige regel.
Verplaatsen	
ESC J	Verplaatst de cursor naar het begin van de huidige regel.
ESC K	Verplaatst de cursor naar het laatste niet-witruimteteken op de huidige regel.

Toets	Sequentie
ESC ↑	Slaat de huidige cursorpositie op. Gebruik ESC ← (naast 1) om de cursor terug te verplaatsen naar de opgeslagen positie. Let op: de ↑ die hier gebruikt wordt, staat naast RESTORE.
ESC ←	Herstelt de cursorpositie naar de positie die eerder is opgeslagen door het indrukken van de toets ESC ↑ (naast RESTORE). Let op: de ← die hier gebruikt wordt, staat naast 1.
ESC HOME	Herstelt de cursorpositie naar de positie die eerder is opgeslagen door het indrukken van HOME.
Vensters	
ESC T	Stelt de linkerbovenhoek van het venstergebied in. Alle ingevoerde tekens en schermactiviteit blijven tot dit gebied beperkt. Zie ook ESC B. U kunt de venstermodus uitschakelen door CLR HOME tweemaal in te drukken.
ESC B	Stelt de rechterbenedenhoek van het venstergebied in. Alle ingevoerde tekens en schermactiviteit blijven tot dit gebied beperkt. Zie ook ESC T. U kunt de venstermodus uitschakelen door CLR HOME tweemaal in te drukken.
Gedrag van de cursor	

Toets	Sequentie
ESC A	Schakelt de automatische invoegmodus in. Elke ingedrukte toets wordt op de huidige cursorpositie ingevoegd, waarna alle tekens op de huidige regel rechts van de cursor één positie naar rechts opschuiven.
ESC C	Schakelt de automatische invoegmodus uit en keert terug naar de overschrijfmodus.
ESC E	Schakelt de cursor over naar de niet-flitsende modus.
ESC F	Schakelt de cursor over naar de gewone flitsmodus.
Gedrag van de bel	
ESC G	Schakelt de bel in. U kunt de bel laten klinken met CTRL en G.
ESC H	Schakelt de bel uit, zodat het indrukken van CTRL en G geen effect heeft.
Kleuren	
ESC U	Schakelt de VIC-IV over naar kleurbereik 0-15 (standaardkleuren). Deze kleuren zijn bereikbaar met CTRL en de toetsen 1 tot 8 (voor de eerste 8 kleuren) of met ALT en de toetsen 1 tot 8 (voor de overige kleuren).
ESC S	Schakelt de VIC-IV over naar kleurbereik 16-31 (alternatieve/rainbow-kleuren). Deze kleuren zijn bereikbaar met CTRL en de toetsen 1 tot 8 (voor de eerste 8 kleuren) of met ALT en de toetsen 1 tot 8 (voor de overige kleuren).
Tabposities	

Toets	Sequentie
ESC Y	Stelt de standaardtabposities in (om de 8 spaties) voor het volledige scherm.
ESC Z	Wist alle tabposities. Met CTRL en I verplaatst een tab de cursor naar het einde van de regel.

BIJLAGE F

Het toetsenbord van de MEGA65

- De typgebeurteniswachtrij gebruiken
- Werking van het toetsenbord
- C65-toetsenbordmatrix
- Synthetische toetsgebeurtenissen
- Besturing van de toetsenbord-LED's
- Native toetsenbordmatrix

De MEGA65 heeft een volledig mechanisch toetsenbord, dat compatibel is met de toetsenborden van de C65 en de C64 en dat vier afzonderlijke cursortoetsen heeft, welke zowel in de C64- als in de C65-modus werken, evenals elf nieuwe C65-toetsen die normaal gesproken uitsluitend in de C65-modus werken.

Er zijn verschillende manieren om het toetsenbord vanuit programma's te lezen. BASIC-programma's kunnen de commando's **GET** of **GETKEY** gebruiken, die een PETSCII-toetscode in een variabele lezen, al dan niet wachtend op een typgebeurtenis. Programma's in machinetaal die de KERNAL actief houden, kunnen de `getin`-routine van de KERNAL op $FFE4 gebruiken, welke een PETSCII-code in de accumulator laadt. Deze routines hebben als voordeel dat zij ook andere voorzieningen van de invoerstroom van de KERNAL ondersteunen, zoals functietoetsmacro's en invoeromleiding.

Voor een machinetaalprogramma dat de KERNAL niet gebruikt en het toetsenbord wil inzetten voor standaardtoepassingen voor tekstinvoer (zoals een tekstverwerker), biedt de MEGA65 een *typgebeurteniswachtrij* die rechtstreeks in de hardware is geïmplementeerd. Een *typgebeurtenis* ontstaat wanneer de gebruiker een toets indrukt die zelf geen modificatietoets is, terwijl daarbij nul of meer modificatietoetsen ingedrukt gehouden worden. De typgebeurteniswachtrij onthoudt de ingedrukte toets en de vastgehouden modificatietoetsen, tot een bepaald maximum aantal gebeurtenissen. Van het programma wordt verwacht dat het elke gebeurtenis uit de wachtrij haalt zodra deze is verwerkt, zodat er in het hardwaregeheugen weer ruimte voor nieuwe gebeurtenissen vrijkomt. Een programma kan typgebeurtenissen rechtstreeks uit hardwareregisters lezen, als PETSCII- of als ASCII-waarde. De modificatietoetsen die bij de gebeurtenis werden vastgehouden, zijn eveneens uit een register te lezen.

De KERNAL van de MEGA65 gebruikt de typgebeurteniswachtrij als onderliggend mechanisme voor `getin` en de BASIC-commando's **GET** en **GETKEY**. Het lezen van een toets met een van deze methoden haalt een typgebeurtenis uit de wachtrij.

De MEGA65 biedt daarnaast rechtstreekse hardwaretoegang tot de toetsenbordmatrix via speciale registers, zodat MEGA65-programma's elke combinatie van gelijktijdig ingedrukte toetsen kunnen vaststellen. Dit is nuttig voor spellen die het toetsenbord als spelbesturing gebruiken.

Oudere C64- en C65-programma's, waaronder de oorspronkelijke C65-ROM, gebruiken de CIA-chips als hardware-interface voor het toetsenbord. Het toetsenbord van de MEGA65 implementeert de C65-toetsenbordmatrix voor de CIA-chip. Deze interface kent de inherente beperkingen die ook bij Commodore-computers voorkomen, waaronder een tragere toetsrespons en "spooktoetsen" bij bepaalde toetscombinaties. MEGA65-programma's behalen betere resultaten met de methoden via de KERNAL, de typgebeurteniswachtrij of de toetsenbordmatrixregisters.

DE TYPGEBEURTENISWACHTRIJ GEBRUIKEN

De status van de typgebeurteniswachtrij is af te lezen uit de volgende registers:

Naam	I/O-adres	Omschrijving
ASCIIKEY	$D610	De bovenste gebeurtenis als ASCII, $FF indien er geen interpretatie is, of $00 indien de wachtrij leeg is
PETSCIIKEY	$D619	De bovenste gebeurtenis als PETSCII, of $FF indien er geen interpretatie is
MODKEY	$D60A[0:6]	Modificatietoetsen die tijdens de gebeurtenis werden vastgehouden
KEYQUEUE	$D60A[7]	1 indien de wachtrij niet leeg is

De bits van MODKEY $D60A komen overeen met de volgende modificatietoetsen. Is een bit gezet, dan werd de bijbehorende toets tijdens de gebeurtenis ingedrukt gehouden:

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CAPS LOCK	NO SCROLL	ALT		CTRL	SHIFT right	SHIFT left

Leest ASCIIKEY ($D610) de waarde $00, dan is de typgebeurteniswachtrij leeg. Een programma moet ASCIIKEY op $00 testen, of KEYQUEUE (bit 7 van $D60A) raadplegen, om te bepalen of de wachtrij leeg is. Het PETSCIIKEY-register heeft geen waarde die voor de lege wachtrij staat.

Een gegeven typgebeurtenis heeft al dan niet een ASCII- of PETSCII-interpretatie. Een typgebeurtenis zonder beide wordt niet aan de wachtrij toegevoegd. Heeft een gebeurtenis wel een ASCII-interpretatie maar geen PETSCII-interpretatie, dan bevat ASCIIKEY de ASCII-waarde en bevat PETSCIIKEY $FF. Evenzo bevat ASCIIKEY $FF voor typgebeurtenissen zonder ASCII-interpretatie. Van het programma wordt verwacht dat het met behulp van de overige registers bepaalt of de wachtrij leeg is en dat het typgebeurtenissen overslaat waarvan de interpretatie voor het programma niet nuttig is.

OPMERKING: ASCII gebruikt $FF eveneens voor het "umlaut-y"-teken (Alt+Y). In dit geval (en uitsluitend in dit geval) is PETSCIIKEY $59 (de gewone Y). Indien uw programma onderscheid moet kunnen maken tussen een typgebeurtenis zonder ASCII-interpretatie en Alt+Y, test dan of PETSCIIKEY $59 is wanneer ASCIIKEY $FF is.

Om de bovenste typgebeurtenis uit de wachtrij te halen, schrijft u een willekeurige waarde naar ASCIIKEY ($D610) of naar PETSCIIKEY ($D619). Bevat de wachtrij nog een gebeurtenis, dan wordt deze zichtbaar in de registers. Anders gaan de registers over naar de toestand van een lege wachtrij.

ASCII-typgebeurtenissen interpreteren

Met het ASCIIKEY-register kunt u de ASCII-symbolen invoeren die in PETSCII niet beschikbaar zijn, waaronder {, _ en |, welke op een gewoon C64- of C128-toetsenbord niet te typen zijn. Het toetsenbord en de ASCII-interpretatie ondersteunen de meeste tekens van de Latin-1-codepagina, waardoor veel geaccentueerde tekens kunnen worden ingevoerd. Deze toetsen voert u in door  ingedrukt te houden en daarbij andere toetsen of toetscombinaties in te drukken. Voor het gebruik van ASCII- of Unicode-basic-Latin-symbolen die niet in de PETSCII-tekenset voorkomen, is een lettertype nodig dat deze symbolen bevat, evenals software die ze ondersteunt.

De volgende tabellen tonen de hexadecimale waarde en het Unicode-teken die elke toetscombinatie in het ASCIIKEY-register oplevert. Er is één tabel voor het indrukken van een toets op zichzelf en één tabel voor elk van de toetsen , ,  en .

Toetsen alleen ingedrukt:

Toets	Code	Unicode	Toets	Code	Unicode	Toets	Code	Unicode
INST DEL	$14		RETURN	$0D		→	$1D	
F7	$F7	÷	F1	$F1	ñ	F3	$F3	ó
F5	$F5	õ	↓	$11		3	$33	3
W	$77	w	A	$61	a	4	$34	4
Z	$7A	z	S	$73	s	E	$65	e
SHIFT left	$00		5	$35	5	R	$72	r
D	$64	d	6	$36	6	C	$63	c
F	$66	f	T	$74	t	X	$78	x
7	$37	7	Y	$79	y	G	$67	g
8	$38	8	B	$62	b	H	$68	h
U	$75	u	V	$76	v	9	$39	9
I	$69	i	J	$6A	j	0	$30	0
M	$6D	m	K	$6B	k	O	$6F	o
N	$6E	n	+	$2B	+	P	$70	p
L	$6C	l	-	$2D	-	.	$2E	.
:	$3A	:	@	$40	@	,	$2C	,
£	$A3	£	*	$2A	*	;	$3B	;
CLR HOME	$13		SHIFT right	$00		=	$3D	=
↑	$AF	¯	/	$2F	/	1	$31	1
←	$5F	_	CTRL	$00		2	$32	2
SPC	$20		C=	$00		Q	$71	q
RUN STOP	$03		NO SCROLL	$00		TAB	$09	
ALT	$00		HELP	$1F		F9	$F9	ù
F11	$FB	û	F13	$FD	ý	ESC	$1B	

Toetsen met SHIFT ingedrukt

Toets	Code	Unicode	Toets	Code	Unicode	Toets	Code	Unicode
INST DEL	$94		RETURN	$0D		→	$9D	
F7	$F8	ø	F1	$F2	ò	F3	$F4	ô
F5	$F6	ö	↓	$91		3	$23	#
W	$57	W	A	$41	A	4	$24	$
Z	$5A	Z	S	$53	S	E	$45	E
SHIFT left	$00		5	$25	%	R	$52	R
D	$44	D	6	$26	&	C	$43	C
F	$46	F	T	$54	T	X	$58	X
7	$27	'	Y	$59	Y	G	$47	G
8	$28	(	B	$42	B	H	$48	H
U	$55	U	V	$56	V	9	$29	)
I	$49	I	J	$4A	J	0	$7B	{
M	$4D	M	K	$4B	K	O	$4F	O
N	$4E	N	+	$00		P	$50	P
L	$4C	L	-	$00		.	$3E	>
:	$5B	[	@	$00		,	$3C	<
£	$E0	à	*	$00		;	$5D	]
CLR HOME	$93		SHIFT right	$00		=	$5F	_
↑	$00		/	$3F	?	1	$21	!
←	$60	`	CTRL	$00		2	$22	"
SPC	$20		C=	$00		Q	$51	Q
RUN STOP	$A3	£	NO SCROLL	$00		TAB	$0F	
ALT	$00		HELP	$1F		F9	$FA	ú
F11	$FC	ü	F13	$FE	þ	ESC	$1B	

Toetsen met CTRL ingedrukt

Toets	Code	Unicode	Toets	Code	Unicode	Toets	Code	Unicode
INST DEL	$94		RETURN	$0D		→	$9D	
F7	$F8	ø	F1	$F2	ò	F3	$F4	ô
F5	$F6	ö	↓	$91		3	$1C	
W	$17		A	$01		4	$9F	
Z	$1A		S	$13		E	$05	
SHIFT left	$00		5	$9C		R	$12	
D	$04		6	$1E		C	$03	
F	$06		T	$14		X	$18	
7	$1F		Y	$19		G	$07	
8	$9E		B	$02		H	$08	
U	$15		V	$16		9	$12	
I	$09		J	$0A		0	$00	
M	$0D		K	$0B		O	$0F	
N	$0E		+	$2B	+	P	$10	
L	$0C		-	$2D	-	.	$2E	.
:	$3A	:	@	$40	@	,	$2C	,
£	$00		*	$EF	ï	;	$3B	;
CLR HOME	$93		SHIFT right	$00		=	$3D	=
↑	$00		/	$2F	/	1	$90	
←	$60	`	CTRL	$00		2	$05	
SPC	$20		C=	$00		Q	$11	
RUN STOP	$A3	£	NO SCROLL	$00		TAB	$0F	
ALT	$00		HELP	$1F		F9	$FA	ú
F11	$FC	ü	F13	$FE	þ	ESC	$1B	

Toetsen met [C=] ingedrukt

Toets	Code	Unicode	Toets	Code	Unicode	Toets	Code	Unicode
INST DEL	$94		RETURN	$0D		→	$ED	í
F7	$F8	ø	F1	$F2	ò	F3	$F4	ô
F5	$F6	ö	↓	$EE	î	3	$96	
W	$D7	×	A	$C1	Á	4	$97	
Z	$DA	Ú	S	$D3	Ó	E	$C5	Å
SHIFT left	$00		5	$98		R	$D2	Ò
D	$C4	Ä	6	$99		C	$C3	Ã
F	$C6	Æ	T	$D4	Ô	X	$D8	Ø
7	$9A		Y	$D9	Ù	G	$C7	Ç
8	$9B		B	$C2	Â	H	$C8	È
U	$D5	Õ	V	$D6	Ö	9	$92	
I	$C9	É	J	$CA	Ê	0	$81	
M	$CD	Í	K	$CB	Ë	O	$CF	Ï
N	$CE	Î	+	$2B	+	P	$D0	Ð
L	$CC	Ì	-	$2D	-	.	$7C	|
:	$7B	{	@	$40	@	,	$7E	~
£	$00		*	$2A	*	;	$7D	}
CLR HOME	$93		SHIFT right	$00		=	$5F	_
↑	$00		/	$5C	\	1	$81	
←	$60	`	CTRL	$00		2	$95	
SPC	$20		[C=]	$00		Q	$D1	Ñ
RUN STOP	$A3	£	NO SCROLL	$00		TAB	$EF	ï
ALT	$00		HELP	$1F		F9	$FA	ú
F11	$FC	ü	F13	$FE	þ	ESC	$1B	

Toetsen met ALT ingedrukt

Toets	Code	Unicode	Toets	Code	Unicode	Toets	Code	Unicode
INST DEL	$7F		RETURN	$00		→	$DF	ß
F7	$DE	Þ	F1	$B9	¹	F3	$B2	²
F5	$B3	³	↓	$00		3	$A4	¤
W	$AE	®	A	$E5	å	4	$A2	¢
Z	$F7	÷	S	$A7	§	E	$E6	æ
SHIFT left	$00		5	$B0	°	R	$AE	®
D	$F0	ð	6	$A5	¥	C	$E7	ç
F	$00		T	$FE	þ	X	$D7	×
7	$B4	´	Y	$FF	ÿ	G	$E8	è
8	$E2	â	B	$FA	ú	H	$FD	ý
U	$FC	ü	V	$D3	Ó	9	$DA	Ú
I	$ED	í	J	$E9	é	0	$DB	Û
M	$B5	µ	K	$E1	á	O	$F8	ø
N	$F1	ñ	+	$B1	±	P	$B6	¶
L	$F3	ó	-	$AC	¬	.	$BB	»
:	$E4	ä	@	$A8	¨	,	$AB	«
£	$A3	£	*	$B7	·	;	$E4	ä
CLR HOME	$DC	Ü	SHIFT right	$DD	Ý	=	$A6	¦
↑	$AF	¯	/	$BF	¿	1	$A1	¡
←	$B8	¸	CTRL	$00		2	$AA	ª
SPC	$A0			$00		Q	$A9	©
RUN STOP	$BA	º	NO SCROLL	$00		TAB	$C0	À
ALT	$00		HELP	$1F		F9	$BC	¼
F11	$BD	½	F13	$BE	¾	ESC	$DB	Û

De keymap voor Unicode Basic-Latin

De volgende tabellen zijn een handig naslagwerk om een toetscombinatie te vinden die de gewenste ASCII/Unicode-basic-Latin-tekencode oplevert. Let op: enkele codes zijn in de praktijk moeilijk in te typen, omdat zij aan toetscombinaties zijn toegewe-

zen die andere functies uitvoeren. Met name de combinatie + TAB wordt normaal gesproken buiten werking gesteld door de hardware-debugfunctie Matrix Mode.

		+$00		+$10		+$20		+$30
$00		ALT + RETURN		F9		SPC	0	0
$01		A		↓	!	1	1	1
$02		B		9	"	2	2	2
$03		RUN STOP		CLR HOME	#	3	3	3
$04		D		INST DEL	$	4	4	4
$05		2		F9	%	5	5	5
$06		F		F11	&	6	6	6
$07		G		F11	'	7	7	7
$08		H		TAB	(	8	8	8
$09		TAB		F13	)	9	9	9
$0a		J		F13	*	*	:	:
$0b		K		ESC	+	+	;	;
$0c		L		£	,	,	<	,
$0d		RETURN		→	-	-	=	=
$0e		N		↑	.	.	>	.
$0f		O		ALT + HELP	/	/	?	/

		+$40		+$50		+$60		+$70
$00	@	@	P	P	`	←	p	P
$01	A	A	Q	Q	a	A	q	Q
$02	B	B	R	R	b	B	r	R
$03	C	C	S	S	c	C	s	S
$04	D	D	T	T	d	D	t	T
$05	E	E	U	U	e	E	u	U
$06	F	F	V	V	f	F	v	V
$07	G	G	W	W	g	G	w	W
$08	H	H	X	X	h	H	x	X
$09	I	I	Y	Y	i	I	y	Y
$0a	J	J	Z	Z	j	J	z	Z
$0b	K	K	[	:	k	K	{	:
$0c	L	L	\	£	l	L	|	.
$0d	M	M	]	;	m	M	}	;
$0e	N	N	^	↑	n	N	~	,
$0f	O	O	_	←	o	O		ALT + INST DEL

		+$80		+$90		+$A0		+$B0
$00				1		ALT + SPC	°	ALT + 5
$01		1		↓	¡	ALT + 1	±	ALT + +
$02				0	¢	ALT + 4	2	ALT + F3
$03		RUN STOP		CLR HOME	£	ALT + £	3	ALT + F5
$04		HELP		INST DEL	¤	ALT + 3	´	ALT + 7
$05		F1		2	¥	ALT + 6	µ	ALT + M
$06		F3		3	¦	ALT + =	¶	ALT + P
$07		F5		4	§	ALT + S	·	ALT + *
$08		F7		5	¨	ALT + @	¸	ALT + ←
$09		F1		6	©	ALT + Q	1	ALT + F1
$0a		F3		7	ª	ALT + 2	º	ALT + RUN STOP
$0b		F5		8	«	ALT + ,	»	ALT + .
$0c		F7		5	¬	ALT + -	¼	ALT + F9
$0d		RETURN		→		Z	½	ALT + F11
$0e				8	®	ALT + W	¾	ALT + F13
$0f				4	¯	ALT + ↑	¿	ALT + /

		+$C0		+$D0		+$E0		+$F0
$00	À	ALT + TAB	Đ	P	à	SHIFT + £	ð	ALT + D
$01	Á	A	Ñ	Q	á	ALT + K	ñ	ALT + N
$02	Â	B	Ò	R	â	ALT + 8	ò	◤ + F1
$03	Ã	C	Ó	ALT + V	ã		ó	ALT + L
$04	Ä	D	Ô	T	ä	ALT + :	ô	◤ + F3
$05	Å	E	Õ	U	å	ALT + A	õ	F5
$06	Æ	F	Ö	V	æ	ALT + E	ö	◤ + F5
$07	Ç	G	×	ALT + X	ç	ALT + C	÷	ALT + Z
$08	È	H	Ø	X	è	ALT + G	ø	ALT + O
$09	É	I	Ù	Y	é	ALT + J	ù	F9
$0a	Ê	J	Ú	ALT + 9	ê		ú	ALT + B
$0b	Ë	K	Û	ALT + 0	ë		û	F11
$0c	Ì	L	Ü	ALT + CLR HOME	ì		ü	ALT + U
$0d	Í	M	Ý	ALT + SHIFT right	í	ALT + I	ý	ALT + H
$0e	Î	N	Þ	ALT + F7	î	◤ + ↓	þ	ALT + T
$0f	Ï	O	ß	ALT + →	ï	◤ + TAB	ÿ	ALT + Y

WERKING VAN HET TOETSENBORD

Het toetsenbord van de MEGA65 is een volledig mechanisch toetsenbord, opgebouwd als een matrix. Elke toetsschakelaar is voorzien van een diode, waardoor de toetsenbordhardware kan detecteren wanneer er tegelijk een willekeurige combinatie van toetsen wordt ingedrukt. Deze matrix wordt door de firmware in de CPLD-chip op de printplaat van het toetsenbord vele duizenden keren per seconde gescand.

De matrixindeling van het MEGA65-toetsenbord maakt geen gebruik van de matrixindeling van de C65. In plaats daarvan zet de CPLD de natuurlijke matrix van het toetsenbord om naar de volgorde van de C65-toetsenbordmatrix en verzendt de C65-matrixcoördinaten vervolgens serieel via de toetsenbordkabel naar het moederbord van de MEGA65. De MEGA65-core leest deze seriële gegevens en gebruikt die om in de FPGA een virtueel toetsenbord te reconstrueren dat compatibel is met de C65. Afhankelijk van het MEGA65-model kan dit virtueel toetsenbord ook invoer aanvaarden van het schermtoetsenbord, van het mechanisme voor het injecteren van synthetische toetsaanslagen en van andere bronnen van toetsenbordinvoer.

Van begin tot eind bedraagt de latentie van het toetsenbord minder dan één milliseconde.

C65-TOETSENBORDMATRIX

Het MEGA65-toetsenbord is ontworpen om de toetsenbordmatrix van de Commodore 65 na te bootsen, welke uit negen kolommen en acht rijen bestaat. Oudere C65-software (zoals de C65-ROM) gebruikt de CIA-chip om de eerste acht kolommen van de matrix te scannen, in dezelfde indeling als waarin de toetsen op een Commodore 64 voorkomen. De toetsen die uitsluitend op de C65 voorkomen (No Scroll, Tab, Alt, enz.) maken gebruik van een extra kolom die niet via de CIA toegankelijk is. Een dergelijk ouder programma kan toegang krijgen tot de negende kolom met behulp van bit 1 van I/O-poort E, waarbij het gegevensregister op $D607 en het gegevensrichtingsregister op $D608 liggen.

Programma's die voor de MEGA65 zijn geschreven, kunnen een speciaal daarvoor bedoelde set registers gebruiken, waarmee elke combinatie van gelijktijdig ingedrukte toetsen te detecteren is:

- $D614: kolomkeuze voor de C65-toetsenbordmatrix; schrijf een geheel getal in het bereik 0 – 8
- $D613: kolomgegevens van de C65-toetsenbordmatrix; de bit voor de rij is gewist indien de toets ingedrukt is
- $D611[0:6]: gegevens van de modificatietoetsen; de bit is gezet indien de modificatietoets ingedrukt is
- $D60F[0:1]: bits voor het onderscheiden van cursor links en cursor omhoog; de bit is gezet indien de toets ingedrukt is

In de onderstaande C65-toetsenbordmatrix komt elke fysieke toets die geen modificatietoets is precies één keer voor: in één van de negen kolommen, genummerd van 0 tot 8, en in één van de acht rijen, genummerd van 0 tot 7.

	0	1	2	3	4	5	6	7	8
0	INST DEL	3	5	7	9	+	£	1	NO SCROLL
1	RETURN	W	R	Y	I	P	*	←	TAB
2	→	A	D	G	J	L	;	CTRL	ALT
3	F7	4	6	8	0	-	CLR HOME	2	HELP
4	F1	Z	C	B	M	.	SHIFT right	SPC	F9
5	F3	S	F	H	K	:	=		F11
6	F5	E	T	U	O	@	↑	Q	F13
7	↓	SHIFT left	X	V	N	,	/	RUN STOP	ESC

Om te testen of een toets wordt ingedrukt, schrijft u het kolomnummer als binair gecodeerd geheel getal naar register $D614 (54804 decimaal) en leest u daarna uit register $D613 (54803 decimaal) de bit die bij de rij hoort. Als de bit nul is, is de toets ingedrukt. (Een bitwaarde van één betekent dat de toets *niet* wordt ingedrukt.)

Om bijvoorbeeld te testen of de Help-toets wordt ingedrukt:

```
LDA #$08              ; column 8
STA $D614
LDA $D613
AND #%00001000        ; row 3
BEQ help_key_is_pressed  ; unset = pressed
```

Register $D611 beschrijft de huidige toestand van de modificatietoetsen. Als een bit gezet is, is de bijbehorende toets ingedrukt.

Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CAPS LOCK	NO SCROLL	ALT		CTRL	SHIFT right	SHIFT left

Het oorspronkelijke toetsenbord van de Commodore 64 had slechts twee cursortoetsen: één voor omlaag/omhoog en één voor rechts/links, waarbij de gebruiker SHIFT ingedrukt hield om de tegengestelde richting op de toets te kiezen. Het toetsenbord van de Commodore 65 heeft afzonderlijke cursortoetsen voor alle vier de richtingen. Voor de compatibiliteit handhaaft de C65 in de toetsenbordmatrix de weergave met ingedrukte SHIFT voor cursor omhoog en cursor links: cursor omhoog komt overeen met de rechter SHIFT-toets samen met cursor omlaag en cursor links met de rechter SHIFT-toets samen met cursor rechts.

Om te kunnen onderscheiden wanneer twee cursortoetsen tegelijk worden ingedrukt om diagonalen te vormen (zoals omhoog en links tegenover omhoog en rechts), beschikt de MEGA65 over nog een register: $D60F, bits 0 en 1. Bit 0 is gezet indien cursor links wordt ingedrukt en bit 1 indien cursor omhoog wordt ingedrukt.

Al met al maken deze registers het voor een programma mogelijk om vrijwel elke combinatie van gelijktijdig ingedrukte toetsen vast te stellen.

SYNTHETISCHE TOETSGEBEURTENISSEN

De logica van de toetsenbordinterface van de MEGA65 staat het gebruik toe van uiteenlopende toetsenbordtypen en alternatieven. Dit dient deels om de vroege ontwikkeling op FPGA-borden voor algemeen gebruik te ondersteunen, aan de MEGAphone met zijn aanraakinterface en aan de desktopversies van de MEGA65-architectuur.

Eén mogelijke vorm van toetsenbordinvoer, beschikbaar op alle modellen, is de wachtrij voor virtuele toetsaanslagen. Een programma of de seriële debugger kan maximaal drie fysieke toetsen als ingedrukt laten lijken door de scancode van elke toets te schrijven naar de registers $D615 - $D617.

De scancode van een fysieke toets kan vanuit de C65-toetsenbordmatrix worden berekend. Voor de eerste acht kolommen is de scancode het kolomnummer vermenigvuldigd met acht, plus het rijnummer. De Home-toets bevindt zich bijvoorbeeld in kolom 6 en rij 3, zodat de scancode $6 \times 8 + 3 = 51$ is. De scancodes voor de negende kolom zijn 64 - 71, toegekend van boven naar beneden. De Tab-toets heeft bijvoorbeeld scancode 65.

Een virtuele toets geldt als ingedrukt totdat het register opnieuw wordt bijgewerkt. Om een toets weer los te laten, schrijft u $7F naar het register.

RESTORE staat, evenals op de C64, buiten de toetsenbordmatrix. Om het ingedrukt houden van RESTORE te simuleren, schrijft u $52 (de ASCII-code voor een hoofdletter R). Om een kort tikje op RESTORE te simuleren, schrijft u $72 (de ASCII-code voor een kleine letter r). Nadat u de waarde $72 heeft weggeschreven, moet u nog een andere waarde wegschrijven indien u RESTORE meerdere keren wilt indrukken.

Het volgende voorbeeld laat zien hoe u het register voor virtuele toetsen vanuit BASIC kunt gebruiken om te simuleren dat de *-toets kort wordt ingedrukt:

```
POKE $D615, 6*8+1:FOR I = 1 TO 100:NEXT:POKE $D615, 127
```

De FOR-lus zorgt voor een geschikte vertraging om te simuleren dat de toets korte tijd ingedrukt gehouden wordt. Alle statements moeten, indien u ze rechtstreeks in de BASIC-interpreter intikt, op één regel staan zoals hier, omdat de MEGA65 anders blijft

doen alsof de *-toets nog steeds ingedrukt gehouden wordt, waardoor het behoorlijk lastig wordt om de overige commando's in te voeren!

BESTURING VAN DE TOETSENBORD-LED'S

Het systeem bestuurt de lampjes op het toetsenbord van de MEGA65 normaal gesproken automatisch, bijvoorbeeld om de stroomstatus (het linker lampje) of de schijfactiviteit (het rechter lampje) weer te geven. U kunt ze echter ook vanuit een programma besturen. Elk van de twee lampjes bestaat uit twee RGB-LED's. De intensiteit van de rode, groene en blauwe componenten van elk lampje kunt u afzonderlijk instellen.

U kunt de LED-kleurwaarden instellen met twee registers: het stuurregister $D61D (decimaal 54813) en het register voor het bijwerken van waarden $D61E.

Bit 7 van het stuurregister $D61D bepaalt of de lampjes door het systeem worden bestuurd (gewist) of door een programma (gezet). De lagere bits selecteren één van de twaalf RGB-kanalen (vier LED's, elk met een rode, groene en blauwe) die moet worden bijgewerkt.

0 linkerhelft van de DRIVE-LED, ROOD

1 linkerhelft van de DRIVE-LED, GROEN

2 linkerhelft van de DRIVE-LED, BLAUW

3 rechterhelft van de DRIVE-LED, ROOD

4 rechterhelft van de DRIVE-LED, GROEN

5 rechterhelft van de DRIVE-LED, BLAUW

6 linkerhelft van de POWER-LED, ROOD

7 linkerhelft van de POWER-LED, GROEN

8 linkerhelft van de POWER-LED, BLAUW

9 rechterhelft van de POWER-LED, ROOD

10 rechterhelft van de POWER-LED, GROEN

11 rechterhelft van de POWER-LED, BLAUW

Om de intensiteit van een LED-kleurcomponent in te stellen, schrijft u het componentnummer en bit 7 hoog (+ 128) naar $D61D en vervolgens de intensiteitswaarde naar $D61E. De intensiteit is een waarde tussen 0 en 255.

De toetsenbordlogica heeft na het schrijven ongeveer één milliseconde nodig om bij te werken. Een goede gewoonte is om $FF (decimaal 255) naar $D61D te schrijven voordat u het componentnummer instelt.

Om de toetsenbord-LED's weer onder systeembesturing te brengen, wist u bit 7 van $D61D.

Het volgende voorbeeld laat de toetsenbord-LED's afwisselend rood en blauw oplichten:

```
10 REM SCHAKEL SOFTWAREBESTURING VAN LEDS IN
20 POKEDEC("D61D"),128
30 REM ZET ALLES LEDS UIT
40 POKEDEC("D61E"),0
50 FORI=0TO11:POKEDEC("D61D"),128+I:NEXT
60 REM KIES ROOD-KANAAL VAN RECHTERLED
70 POKEDEC("D61D"),128
80 REM VAN ZWART NAAR ROOD EN WEER TERUG
90 FORI=0TO255:POKEDEC("D61E"),I:NEXT
100 FORI=255TO0STEP-1:POKEDEC("D61E"),I:NEXT
110 REM KIES BLAUW-KANAAL VAN LINKERLED
120 POKEDEC("D61D"),128+8
130 REM VAN ZWART NAAR BLAUW EN WEER TERUG
140 FORI=0TO255:POKEDEC("D61E"),I:NEXT
150 FORI=255TO0STEP-1:POKEDEC("D61E"),I:NEXT
160 GOTO70
```

NATIVE TOETSENBORDMATRIX

De native toetsenbordmatrix is uitsluitend vanuit de CPLD op het toetsenbord van de MEGA65 toegankelijk. Indien u de MEGA65-computer programmeert, hoeft u deze normaliter niet te gebruiken. Dit staat hier vermeld voor het gemak van de platform-ontwikkelaars.

0 F5
1 9
2 I
3 K
4 <
5 INST DEL
6 CLR HOME
7 O
8 F3
9 8
10 U
11 J
12 M
13 →
14 £
15 =
16 F1
17 7
18 Y
19 H
20 N
21 ↓
22 -
23 ;
24 Gereserveerd
25 6
26 T
27 G
28 B
29 ← (cursor links)
30 +
31 :
32 NO SCROLL
33 5
34 R
35 F
36 V
37 SPACE
38 0
39 L
40 CAPS LOCK
41 4
42 E
43 D
44 C
45 Gereserveerd
46 HELP
47 RETURN

48 ALT

49 3

50 W

51 S

52 X

53 ↑ (cursor omhoog)

54 F13

55 ↑ (naast RESTORE)

56 ESC

57 2

58 Q

59 A

60 Z

61 rechter SHIFT

62 F11

63 *

64 Gereserveerd

65 1

66 Gereserveerd

67 Gereserveerd

68 linker SHIFT en SHIFT LOCK

69 /

70 F9

71 @

72 RUN STOP

73 ← (naast 1)

74 TAB

75 CTRL

76 ■

77 >

78 F7

79 P

BIJLAGE G

Systeempalet

- Systeempalet

SYSTEEMPALET

De volgende tabel beschrijft het systeempalet zoals het standaard is vastgelegd. U kunt deze paletkleuren wijzigen met het commando **PALETTE COLOR** en terugzetten op hun standaardwaarden met het commando **PALETTE RESTORE**.

Paletindexen worden gebruikt als waarden in de speciale array **C@&()** en als argumenten voor BASIC-commando's zoals **BACKGROUND**, **BORDER**, **COLOR**, **FOREGROUND**, **HIGHLIGHT**, **PEN** en **SCREEN CLR**.

Index	Rood	Groen	Blauw	Kleur
0	0	0	0	Zwart
1	15	15	15	Wit
2	15	0	0	Rood
3	0	15	15	Cyaan
4	15	0	15	Paars
5	0	15	0	Groen
6	0	0	15	Blauw
7	15	15	0	Geel
8	15	6	0	Oranje
9	10	4	0	Bruin
10	15	7	7	Lichtrood (roze)
11	5	5	5	Donkergrijs
12	8	8	8	Middengrijs
13	9	15	9	Lichtgroen
14	9	9	15	Lichtblauw
15	11	11	11	Lichtgrijs
16	14	0	0	Guru-meditatie
17	15	5	0	Rambtoetan
18	15	11	0	Wortel
19	14	14	0	Citroentaart
20	7	15	0	Pandangroen
21	6	14	6	Zeeziek groen
22	0	14	3	Soylent Groen
23	0	15	9	Slijmgroen
24	0	13	13	Het andere cyaan
25	0	9	15	Zeehemel
26	0	3	15	Smurfenblauw
27	0	0	14	Scherm des Doods
28	7	0	15	Pruimensaus
29	12	0	15	Zure druif
30	15	0	11	Kauwgom
31	15	3	6	Hete tamales

U kunt deze 32 kleuren in de schermterminal gebruiken door PETSCII-Controlcodes te typen of weer te geven. De tekenstatus van de cursor is ofwel één van de eerste 16 kleuren ofwel één van de tweede 16 kleuren, en deze status kan met Controlcodes worden gewijzigd. Elk van de 16 Controlcodes kiest de kleur van de cursor binnen die kleurenset.

Om kleurenset A te kiezen, drukt u op CTRL + D of geeft u PETSCII-code 4 weer.

Om kleurenset B te kiezen, drukt u op CTRL + A of geeft u PETSCII-code 1 weer.

Toets	PETSCII	Kleurenset A	Kleurenset B
CTRL + 1	144	Zwart	Guru-meditatie
CTRL + 2	5	Wit	Rambtoetan
CTRL + 3	28	Rood	Wortel
CTRL + 4	159	Cyaan	Citroentaart
CTRL + 5	156	Paars	Pandangroen
CTRL + 6	30	Groen	Zeeziek groen
CTRL + 7	31	Blauw	Soylent Groen
CTRL + 8	158	Geel	Slijmgroen
C= + 1	129	Oranje	Het andere cyaan
C= + 2	149	Bruin	Zeehemel
C= + 3	150	Lichtrood (roze)	Smurfenblauw
C= + 4	151	Donkergrijs	Scherm des Doods
C= + 5	152	Middengrijs	Pruimensaus
C= + 6	153	Lichtgroen	Zure druif
C= + 7	154	Lichtblauw	Kauwgom
C= + 8	155	Lichtgrijs	Hete tamales

BIJLAGE H

Decimaal, binair en hexadecimaal

- Getallen
- Notaties en grondtallen
- Bewerkingen
- Gehele en natuurlijke getallen
- Bitgewijze logische operatoren
- Getallen omrekenen

GETALLEN

Eenvoudige computerprogramma's, zoals de meeste inleidende BASIC-programma's in dit boek, vragen geen kennis van wiskunde en evenmin veel kennis van de inwendige werking van de computer. Dat komt doordat BASIC als hogere programmeertaal geldt. Deze taal laat ons de computer enigszins indirect programmeren, maar geeft ons nog steeds controle over de mogelijkheden van de computer. Meestal hoeven we ons niet bezig te houden met de interne architectuur van de computer, waardoor BASIC gebruiksvriendelijk en toegankelijk is.

Naarmate u meer kennis opdoet en meer ervaring krijgt, wilt u de computer vaak taken laten uitvoeren die complexer of gespecialiseerder zijn dan de voorbeelden in dit boek. Wellicht wilt u, bijvoorbeeld om redenen van efficiëntie, ook directe en precieze controle uitoefenen op de inhoud van het geheugen van de computer. Dat geldt vooral voor toepassingen die met geavanceerde grafiek en geluid werken. Zulke bewerkingen zitten dichter bij de hardware en worden daarom als laagniveau beschouwd. Om deze mogelijkheden op laag niveau effectief te kunnen benutten, is enige eenvoudige wiskundige kennis nodig.

De gezamenlijke stand van de piepkleine schakelaars in de computer—of elke schakelaar nu aan of uit staat—vormt de toestand van de computer. Het ligt voor de hand om aan deze toestand numerieke begrippen te koppelen. Getallen laten ons de binnenkant van de machine begrijpen en met logische en rekenkundige bewerkingen manipuleren. Getallen laten ons ook de twee wezenlijke en belangrijke soorten informatie coderen die in elk computerprogramma schuilgaan: *instructies* en *gegevens*.

De instructies van een programma vertellen een computer wat hij moet doen en hoe hij dat moet doen. Het weergeven van een tekenreeks op het scherm via het statement **PRINT** is bijvoorbeeld een instructie. Ook het tonen van een sprite en het wijzigen van de randkleur van het scherm zijn instructies. Achter de schermen gaat elke instructie die u aan de computer geeft gepaard met één of meer getallen (welke op hun beurt corresponderen met de piepkleine schakelaars in de computer, die aan of uit staan). Meestal zullen die instructies er voor u niet uitzien als getallen; in plaats daarvan nemen zij bijvoorbeeld de vorm aan van statements in BASIC.

De gegevens van een programma bestaan uit informatie. De begroeting "HELLO MEGA65!" is bijvoorbeeld PETSCII-tekengegevens in de vorm van een tekenreeks. Het grafische ontwerp van een sprite kan pixelgegevens zijn in de vorm van een held voor een spel. En de kleurgegevens van de rand van het scherm kunnen oranje voorstellen. Ook hier gaat achter de schermen elk gegeven dat u aan de computer geeft gepaard met één of meer getallen. Gegevens staan soms direct naast het statement waarop zij betrekking hebben; zulke gegevens worden een parameter of argument genoemd (zoals bij het wijzigen van de schermkleur met een **BACKGROUND 1**-statement). Gegevens kunnen ook binnen het programma worden opgegeven, via het BASIC-statement **DATA**, dat een lijst van door komma's gescheiden waarden aanvaardt.

Al die getallen—ongeacht of zij nu instructies of gegevens voorstellen—bevinden zich in het geheugen van de computer. Hoewel het geheugen van de computer sterk ge-

structureerd is, maakt de computer geen onderscheid tussen instructies en gegevens, noch heeft hij aparte geheugengebieden voor elke soort informatie. In plaats daarvan gebruikt de computer gewoon de geheugenlocatie die het handigst is om beide op te slaan. Of de inhoud van een bepaalde geheugenlocatie nu deel uitmaakt van de instructies van het programma of van de gegevens ervan, hangt grotendeels af van uw standpunt, van het programma dat geschreven wordt en van de behoeften van de programmeur.

Hoewel BASIC een hogere programmeertaal is, biedt het toch statements waarmee programmeurs het geheugen van de computer efficiënt kunnen manipuleren. Het statement **PEEK** laat ons de informatie op een opgegeven geheugenlocatie lezen: we kunnen de inhoud van een geheugenadres bekijken. Het statement **POKE** laat ons informatie op een opgegeven geheugenlocatie opslaan: we kunnen de inhoud van een geheugenadres zodanig wijzigen dat deze op een gegeven waarde komt te staan.

NOTATIES EN GRONDTALLEN

Laten we nu eens naar getallen kijken.

Getallen zijn ideeën over hoeveelheid en grootte. Om getallen te kunnen bewerken en onderlinge verbanden te bepalen, is het belangrijk dat elk getal een unieke vorm heeft. Dit brengt ons bij de symbolische voorstelling van getallen met behulp van een positionele notatie. In deze bijlage beperken wij onze bespreking tot gehele getallen, welke ook wel *integers* worden genoemd.

De *decimale* voorstelling van getallen is degene waarmee u het beste vertrouwd bent, want dat is de voorstelling die u op school heeft geleerd. Decimale notatie gebruikt de tien Arabisch-Indische cijfers 0, 1, 2, 3, 4, 5, 6, 7, 8 en 9 en wordt daarom aangeduid als een talstelsel met grondtal 10. Zoals we later zullen zien, gebruiken we voor het schrijven van grote getallen in decimale vorm een positioneel stelsel, waarin we cijfers naast elkaar in kolommen zetten om een groter getal te vormen.

53280 is bijvoorbeeld een decimaal getal. Elk van die cijfers (0 tot en met 9) in een decimaal getal stelt een veelvoud voor van een bepaalde macht van 10. Wanneer een BASIC-statement (zoals **PEEK** of **POKE**) een geheel getal als parameter nodig heeft, geeft u die parameter op in decimale vorm.

Hoewel decimale notatie voor mensen natuurlijk en gemakkelijk aanvoelt, gebruiken moderne computers op hun meest fundamentele niveau een andere notatie. Deze notatie heet *binair*. Men spreekt ook van een talstelsel met grondtal 2, omdat het slechts twee Arabisch-Indische cijfers gebruikt: 0 en 1. Binair weerspiegelt het feit dat elke piepkleine schakelaar in de computer zich in precies één van twee elkaar uitsluitende toestanden moet bevinden: aan of uit. Het getal 0 hoort bij uit en het getal 1 hoort bij aan. Binair is de eenvoudigste notatie die dit idee weergeeft. Om grote getallen binair te schrijven, gebruiken we een positioneel stelsel, waarin we cijfers naast elkaar in kolommen zetten om een groter getal te vormen, met een %-teken als voorvoegsel.

%1001 0110 is bijvoorbeeld een binair getal. Elk van die cijfers (0 of 1) in een binair getal stelt een veelvoud voor van een bepaalde macht van 2.

Later zien we hoe we met speciale BASIC-statements de patronen van enen en nullen in een binair getal kunnen bewerken om de toestand van de bijbehorende schakelaars te wijzigen. In feite kunnen we zo afzonderlijke schakelaars naar behoefte aan- of uitzetten.

Vaak wordt ook een derde notatie gebruikt, *hexadecimaal* genaamd. Dit is een talstelsel met grondtal 16. Omdat het meer dan tien cijfers gebruikt, hebben we letters nodig om de extra cijfers weer te geven. Hexadecimaal gebruikt de tien Arabisch-Indische cijfers 0 tot en met 9, aangevuld met de zes Latijnse letters als "cijfers" (A, B, C, D, E en F) om de getallen 10 tot en met 15 voor te stellen. Er zijn zo in totaal zestien symbolen voor de getallen 0 tot en met 15. Om een groot getal hexadecimaal te schrijven, gebruiken we een positioneel stelsel, waarin we cijfers naast elkaar in kolommen zetten om een groter getal te vormen, met een $-teken als voorvoegsel.

$E7 is bijvoorbeeld een hexadecimaal getal. Elk van die cijfers (0 tot en met 9 en A tot en met F) in een hexadecimaal getal stelt een veelvoud voor van een bepaalde macht van 16.

Bij het programmeren in BASIC wordt hexadecimaal niet vaak gebruikt. Het komt vaker voor bij het programmeren in lagere talen, zoals machinetaal of assembleertaal. Ook verschijnt het in geheugenindelingen van computers. Omdat het zo bondig is, is het een nuttige notatie; wij beschrijven het hier dan ook.

Denk er altijd aan dat decimaal, binair en hexadecimaal slechts verschillende notaties voor getallen zijn. Een notatie verandert alleen de manier waarop het getal geschreven wordt (d.w.z. hoe het er op papier of op het scherm uitziet); de eigenlijke waarde blijft onveranderd. Een notatie is in wezen niets anders dan een andere manier om hetzelfde weer te geven. Dat we verschillende notaties gebruiken, komt doordat elke notatie zich van nature leent voor een andere taak.

Wanneer u langere tijd met decimale, binaire en hexadecimale getallen werkt, zult u het handig vinden om een wetenschappelijke zakrekenmachine met een programmeermodus bij de hand te hebben. Zulke rekenmachines kunnen met één druk op de knop tussen grondtallen omrekenen. Zij kunnen tevens optellen, aftrekken, vermenigvuldigen, delen en diverse bitgewijze logische bewerkingen uitvoeren. Zie Hoofdstuk/-Bijlage AA op bladzijde AA-3: daar staat een tabel voor de Getalstelselomrekening, met decimale, binaire en hexadecimale weergaven van gehele getallen van 0 tot en met 255.

De BASIC-listing bij deze bijlage is een hulpprogramma dat afzonderlijke getallen naar verschillende grondtallen omrekent. Het kan ook meerdere getallen binnen een opgegeven bereik omrekenen.

Hoewel deze begrippen nu misschien nog nieuw zijn, worden zij met enige oefening al snel een tweede natuur. We gaan nader in op de manieren om getallen uit te drukken. Later onderzoeken we ook de verschillende bewerkingen die we op zulke getallen kunnen uitvoeren.

Decimaal

Wanneer we gehele getallen in decimale notatie weergeven, staat elke kolom in het getal voor een andere macht van 10. De rechterpositie stelt het aantal eenheden voor (want $10^0 = 1$) en elke kolom links daarvan is 10 keer zo groot als de kolom ervoor. De meest rechtse kolom heet de eenhedenkolom. De kolommen links daarvan heten tientallen (want $10^1 = 10$), honderdtallen (want $10^2 = 100$), duizendtallen (want $10^3 = 1000$), enzovoort.

Om een voorbeeld te geven: het gehele getal 53280 staat voor de som van 5 maal 10000, 3 maal 1000, 2 maal 100, 8 maal 10 en 0 eenheden. Dit wordt duidelijker wanneer we het getal per kolom in afzonderlijke delen opsplitsen.

Aangezien

$$53280 = 50000 + 3000 + 200 + 80 + 0$$

kunnen we dit in een tabel weergeven, met onderaan de som van elke kolom.

TIENDUIZENDEN	**DUIZENDEN**	**HONDERDEN**	**TIENTALLEN**	**EENHEDEN**
$10^4 = 10000$	$10^3 = 1000$	$10^2 = 100$	$10^1 = 10$	$10^0 = 1$
5	0	0	0	0
	3	0	0	0
		2	0	0
			8	0
				0
5	3	2	8	0

Een andere manier om dit te formuleren is de uitdrukking decimaal te schrijven, met veelvouden van machten van 10.

$$53280 = (5 \times 10^4) + (3 \times 10^3) + (2 \times 10^2) + (8 \times 10^1) + (0 \times 10^0)$$

Als alternatief

$$53280 = (5 \times 10000) + (3 \times 1000) + (2 \times 100) + (8 \times 10) + (0 \times 1)$$

We introduceren nu enige nuttige terminologie die bij decimale getallen hoort.

Het meest rechtse cijfer van een decimaal getal heet het minst significante cijfer, omdat dit cijfer, als kleinste vermenigvuldiger van een macht van 10, het minst bijdraagt aan de grootte van het getal. Elk cijfer links van dit cijfer heeft telkens een grotere betekenis. Het meest linkse (niet-nul) cijfer van het decimale getal heet het meest significante cijfer, omdat dit cijfer, als grootste vermenigvuldiger van een macht van 10, het meest bijdraagt aan de grootte van het getal.

In het decimale getal 53280 is het cijfer 0 bijvoorbeeld het minst significante cijfer en is het cijfer 5 het meest significante cijfer.

Een decimaal getal a is m *ordes van grootte* groter dan het decimale getal b indien $a = b \times (10^m)$. Het getal 50000 is bijvoorbeeld drie ordes van grootte groter dan

50, omdat het drie nullen meer heeft. Deze terminologie kan nuttig zijn bij het vergelijken van getallen, of bij het vergelijken van de tijds- of ruimte-efficiëntie van twee programma's in verhouding tot de grootte van de gegeven invoer.

Merk op dat decimale getallen, anders dan binaire getallen (met het gebruikelijke % voorvoegsel) en hexadecimale getallen (met het gebruikelijke $ voorvoegsel), geen speciaal voorvoegsel krijgen. In sommige leerboeken komen dergelijke getallen in plaats daarvan met een subscript voor. Decimale getallen krijgen dus een 10 als subscript, binaire getallen een 2 en hexadecimale getallen een 16.

Een ander nuttig begrip is het onderscheid tussen gehele en natuurlijke getallen.

Een decimaal geheel getal kan positief, negatief of nul zijn. Om een decimaal geheel getal weer te geven, zetten we er een $+$-teken of een $-$-teken voor. (Volgens conventie krijgt nul, dat noch positief noch negatief is, het $+$-teken.)

Is een decimaal geheel getal daarentegen een natuurlijk getal, dan kan het alleen nul of positief zijn en heeft het geen negatieve voorstelling. Dit valt te illustreren met de BASIC-statements **PEEK** en **POKE**. Gebruiken we **PEEK** om de waarde op te vragen die in een geheugenlocatie staat, dan krijgen we een natuurlijk getal terug. Het statement **PRINT (PEEK (49152))** geeft de inhoud van geheugenlocatie 49152 bijvoorbeeld als natuurlijk getal op het scherm weer. Merk op dat het geheugenadres dat we aan **PEEK** gaven, zelf een natuurlijk getal is. Gebruiken we **POKE** om een waarde in een geheugenlocatie op te slaan, dan worden zowel het geheugenadres als de op te slaan waarde opgegeven als natuurlijk getal. Het statement **POKE 49152, 128** slaat bijvoorbeeld het natuurlijke getal 128 weg in het geheugenadres dat door het natuurlijke getal 49152 wordt aangegeven.

Elke geheugenlocatie in de MEGA65 kan een geheel getal tussen 0 en 255 bevatten. Dit komt overeen met de kleinste en de grootste gehele getallen die we met acht binaire cijfers (acht bits) kunnen weergeven. Ook zijn de geheugenadressen gehele getallen tussen 0 en 65535. Dit komt overeen met de kleinste en de grootste gehele getallen die we met zestien binaire cijfers (zestien bits) kunnen weergeven.

Merk op dat het grootste getal dat u met d decimale cijfers kunt uitdrukken, $10^d - 1$ is. (In zijn voorstelling bestaat dit getal uit d negens.)

Binair

De binaire notatie gebruikt machten van 2 (en niet van 10, zoals bij het decimale stelsel). De meest rechtse positie geeft het aantal eenheden weer (omdat $2^0 = 1$) en elke kolom links daarvan is twee keer zo groot als de kolom ervoor. De kolommen links van de meest rechtse kolom zijn achtereenvolgens de kolom van de tweeën (omdat $2^1 = 2$), de kolom van de vieren (omdat $2^2 = 4$), de kolom van de achten (omdat $2^3 = 8$) enzovoort.

Zo bestaat het getal %1101 0011 bijvoorbeeld uit precies acht binaire cijfers en stelt het de som voor van 1 keer 128, 1 keer 64, 0 keer 32, 1 keer 16, 0 keer 8, 0 keer 4, 1 keer 2 en 1 eenheid.

We kunnen dit getal per kolom opsplitsen in afzonderlijke delen.

Aangezien

%1101 0011 = %1000 0000 + %100 0000 + %00 0000 + %1 0000 + %0000 + %000 + %10 + %1

kunnen we dit in een tabel weergeven, met onderaan de som van elke kolom.

HONDERD-ACHTENTWINTIGEN	**VIEREN-ZESTIGEN**	**TWEEËN-DERTIGEN**	**ZESTIENEN**	**ACHTEN**	**VIEREN**	**TWEEËN**	**EENHEDEN**
$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
1	0	0	0	0	0	0	0
	1	0	0	0	0	0	0
		0	0	0	0	0	0
			1	0	0	0	0
				0	0	0	0
					0	0	0
						1	0
							1
1	1	0	1	0	0	1	1

Een andere manier om dit te formuleren is de uitdrukking decimaal te schrijven, met veelvouden van machten van 2.

$$\%11010011 = (1\times2^7)+(1\times2^6)+(0\times2^5)+(1\times2^4)+(0\times2^3)+(0\times2^2)+(1\times2^1)+(1\times2^0)$$

Als alternatief

$$\%11010011 = (1\times128)+(1\times64)+(0\times32)+(1\times16)+(0\times8)+(0\times4)+(1\times2)+(1\times1)$$

hetgeen gelijkstaat aan het schrijven van

$$\%11010011 = 128+64+16+2+1$$

De binaire notatie kent ook een eigen terminologie. Elk binair cijfer in een binair getal heet een *bit*. In een 8-bits getal worden de bits opeenvolgend genummerd, met het minst significante (d.w.z. meest rechtse) bit als bit 0 en het meest significante (d.w.z. meest linkse) bit als bit 7. In een 16-bits getal is het meest significante bit bit 15. Men zegt dat een bit *gezet* is indien het gelijk is aan 1. Men zegt dat een bit *gewist* is indien het gelijk is aan 0. Heeft een bepaald bit een bijzondere betekenis, dan noemen we het soms een *vlag*.

1	1	0	1	0	0	1	1
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Zoals eerder vermeld, kan elke geheugenlocatie een geheel getal tussen 0 en 255 bevatten. Het minimum komt overeen met %0000 0000 en het maximum met %1111 1111; dit zijn de kleinste en de grootste getallen die we met precies acht bits kunnen

weergeven. De geheugenadressen bestaan uit 16 bits. Het kleinste geheugenadres, genoteerd in precies zestien bits, is %0000 0000 0000 0000 en komt overeen met het kleinste 16-bits getal. Evenzo is het grootste geheugenadres, genoteerd in precies zestien bits, %1111 1111 1111 1111 en komt dit overeen met het grootste 16-bits getal.

Het is vaak handig om met verschillende namen naar groepen bits te verwijzen. Zo vormen acht bits samen een *byte* en 1024 bytes samen een *kilobyte*. De helft van een byte heet een nibble.Zie Hoofdstuk/Bijlage AA op bladzijde AA-3 voor de Opslageenheden-tabel met nadere informatie.

Merk op dat het grootste getal dat u met d binaire cijfers kunt uitdrukken, (decimaal) $2^d - 1$ bedraagt. (Dit getal heeft in zijn voorstelling d enen.)

Hexadecimaal

Hexadecimale notatie gebruikt machten van 16. Elk van de zestien hexadecimale cijfers heeft een bijbehorende decimale waarde.

Hexadecimaal Cijfer	Decimaal Equivalent
$0	0
$1	1
$2	2
$3	3
$4	4
$5	5
$6	6
$7	7
$8	8
$9	9
$A	10
$B	11
$C	12
$D	13
$E	14
$F	15

De meest rechtse positie in een hexadecimaal getal geeft het aantal enen weer (aangezien $16^0 = 1$). Elke kolom links van dit cijfer is 16 keer zo groot als de kolom ervoor. De kolommen links van de meest rechtse kolom zijn de 16-kolom (aangezien $16^1 = 16$), de 256-kolom (aangezien $16^2 = 256$), de 4096-kolom (aangezien $16^3 = 4096$) enzovoort.

Het gehele getal $A3F2 bestaat bijvoorbeeld uit precies vier hexadecimale cijfers en stelt de som voor van 10 maal 4096 (omdat $A = 10), 3 maal 256 (omdat $3 = 3), 15 maal 16 (omdat $F = 15) en 2 eenheden (omdat $2 = 2). We kunnen dit gehele getal per kolom opsplitsen in afzonderlijke delen.

Aangezien

$$\$A3F2 = \$A000 + \$300 + \$F0 + \$2$$

kunnen we dit weergeven in een tabel met de som van elke kolom onderaan.

VIERDUIZEND ZESENNEGEN-TIGEN	TWEEHONDERD ZESENVIJF-TIGEN	ZESTIENEN	EENHEDEN
$16^3 = 4096$	$16^2 = 256$	$16^1 = 16$	$16^0 = 1$
A	0	0	0
	3	0	0
		F	0
			2
A	3	F	2

Een andere manier om dit te formuleren is de uitdrukking decimaal te schrijven, met veelvouden van machten van 16.

$$\$A3F2 = (10 \times 16^3) + (3 \times 16^2) + (15 \times 16^1) + (2 \times 16^0)$$

Als alternatief

$$\$A3F2 = (10 \times 4096) + (3 \times 256) + (15 \times 16) + (2 \times 1)$$

hetgeen gelijkstaat aan

$$\$A3F2 = 40960 + 768 + 240 + 2$$

Ook hier geldt, net als bij binair en decimaal, dat het meest rechtse cijfer het minst significante is en het meest linkse cijfer het meest significante.

Elke geheugenlocatie kan een geheel getal tussen 0 en 255 bevatten, hetgeen overeenkomt met de hexadecimale getallen $00 en $FF. Het hexadecimale getal $FFFF komt overeen met 65535—het grootste 16-bits getal.

Hexadecimale notatie is vaak handiger in gebruik en bewerking dan binaire notatie. Binaire getallen bestaan uit een langere reeks enen en nullen, terwijl hexadecimaal veel korter en compacter is. Dat komt doordat één hexadecimaal cijfer exact gelijk is aan vier bits. Een hexadecimaal getal van twee cijfers omvat dus acht bits, waarbij de lage nibble overeenkomt met het rechtercijfer en de hoge nibble met het linkercijfer.

Merk op dat het grootste getal dat u met d hexadecimale cijfers kunt uitdrukken, (decimaal) $16^d - 1$ bedraagt. (Dit getal heeft in zijn voorstelling d $F-symbolen.)

BEWERKINGEN

In deze sectie nemen we enkele bekende bewerkingen door, zoals tellen en rekenen. We zien tevens hoe deze bewerkingen van toepassing zijn op getallen die binair en hexadecimaal genoteerd zijn.

Daarna bekijken we allerlei logische bewerkingen die met logische poorten worden uitgevoerd. Deze bewerkingen zijn eenvoudig te begrijpen. Ze zijn tevens zeer belangrijk bij het schrijven van programma's met uitgebreide numerieke, grafische of geluidsmogelijkheden.

Tellen

Indien we het proces van *tellen* in het decimale stelsel zorgvuldig bekijken, begrijpen we beter hoe tellen werkt bij het gebruik van binair en hexadecimaal.

Laten we aannemen dat we in het decimale stelsel tellen en dat we daarbij bij 0 beginnen. Denk eraan dat de lijst met cijfers voor het decimale stelsel (in volgorde) 0, 1, 2, 3, 4, 5, 6, 7, 8 en 9 is. Merk op dat we 1 krijgen wanneer we 1 bij 0 optellen en dat we 2 krijgen wanneer we 1 bij 1 optellen. We kunnen op deze manier doorgaan, telkens 1 erbij optellend:

0 + 1 = 1
1 + 1 = 2
2 + 1 = 3
3 + 1 = 4
4 + 1 = 5
5 + 1 = 6
6 + 1 = 7
7 + 1 = 8
8 + 1 = 9

Aangezien 9 het hoogste cijfer in onze lijst met cijfers voor het decimale stelsel is, hebben we een manier nodig om de volgende speciale optelling te verwerken: $9 + 1$. Het antwoord is dat we onze oude cijfers helemaal opnieuw kunnen gebruiken. In deze belangrijke stap zetten we de eenhedenkolom terug op 0 en tellen we (tegelijkertijd) 1 bij de tientallenkolom op. Aangezien de tientallenkolom een 0 bevatte, geeft dit $9 + 1 = 10$. We zeggen dat we de 1 naar de tientallenkolom hebben "overgedragen", terwijl de eenhedenkolom terugliep naar 0.

Met deze techniek kunnen we zo hoog tellen als we willen. Het telprincipe voor binair en hexadecimaal verloopt vrijwel op dezelfde manier; het enige verschil is dat we in plaats van tien symbolen respectievelijk twee en zestien symbolen tot onze beschikking hebben.

Laten we eens kijken naar tellen in het binaire stelsel. Zoals u zich herinnert, bestaat de lijst met cijfers voor binair (in deze volgorde) slechts uit 0 en 1. Als we dus beginnen te tellen bij %0 en vervolgens %1 optellen, krijgen we %1 als resultaat:

%0 + %1 = %1

Nu dwingt de som $\%1 + \%1$ ons tot de vergelijkbare stap: we zetten de eenhedenkolom weer op nul en tellen tegelijk %1 op bij de kolom van de tweeën. Omdat de kolom van de tweeën een %0 bevatte, levert dit $\%1 + \%1 = \%10$ op. We zeggen dat we de %1 naar de kolom van de tweeën hebben "overgedragen", terwijl de eenhedenkolom terugliep naar %0. Als we op deze manier doorgaan, kunnen we nog hoger tellen.

%1 + %1 = %10
%10 + %1 = %11
%11 + %1 = %100
%100 + %1 = %101
%101 + %1 = %110
%110 + %1 = %111
%111 + %1 = %1000

Nu bekijken we tellen in het hexadecimale stelsel. De lijst met cijfers voor hexadecimaal is (in deze volgorde) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E en F. Als we beginnen te tellen bij $0 en steeds $1 optellen, krijgen we:

$0 + $1 = $1
$1 + $1 = $2
$2 + $1 = $3
$3 + $1 = $4
$4 + $1 = $5
$5 + $1 = $6
$6 + $1 = $7
$7 + $1 = $8
$8 + $1 = $9
$9 + $1 = $A
$A + $1 = $B
$B + $1 = $C
$C + $1 = $D
$D + $1 = $E
$E + $1 = $F

Wanneer we $F + $1 berekenen, moeten we de eenhedenkolom weer op $0 zetten en $1 optellen bij de 16-kolom, omdat dat getal wordt "overgedragen".

$F + $1 = $10

Ook dit proces stelt ons in staat om zo hoog te tellen als we willen.

Rekenkunde

De standaardrekenkundige bewerkingen (optellen, aftrekken, vermenigvuldigen en delen) zijn allemaal mogelijk met binaire en hexadecimale getallen.

Optellen gebeurt op dezelfde manier als in het decimale stelsel; het enige verschil is dat we al naargelang het geval grondtal 2 of grondtal 16 gebruiken. Beschouw het volgende voorbeeld van de optelling van twee binaire getallen.

```
  % 1 1 0
+ % 1 1 1
---------
% 1 1 0 1
=========
```

Het resultaat verkrijgen we door eerst de eenhedenkolommen van beide getallen op te tellen. Dat geeft %0 + %1 = %1, zonder overdracht naar de volgende kolom. Daarna

tellen we de kolommen van de tweeën van beide getallen op: %1 + %1 = %0, met een %1 die naar de volgende kolom wordt overgedragen. Vervolgens tellen we de kolommen van de vieren op (inclusief de overdracht), wat (%1 + %1) + %1 = %1 oplevert, met een %1 die naar de volgende kolom wordt overgedragen. Als laatste komen de kolommen van de achten. Omdat deze in feite beide nul zijn, houden we ons alleen met de overdracht bezig en die is %1. Zo is (%0 + %0) + %1 = %1. De som is dus %1101.

Hierna volgt een voorbeeld van de optelling van twee hexadecimale getallen.

```
  $ 7 D
+ $ 6 9
-------
  $ E 6
=======
```

We beginnen met het optellen van de eenhedenkolommen van beide getallen. Dat geeft $D + $9 = $6, met een $1 die naar de volgende kolom wordt overgedragen. Vervolgens tellen we de 16-kolommen op (inclusief de overdracht), wat ($7 + $6) + $1 = $E oplevert zonder overdracht, zodat $E6 de som is.

Nu bekijken we het aftrekken. Zoals u wellicht vermoedt, verloopt aftrekken met binaire en hexadecimale getallen volgens een vergelijkbaar proces als aftrekken met decimale gehele getallen.

Beschouw de volgende aftrekking van twee binaire getallen.

```
  % 1 0 1 1
- %   1 1 0
-----------
  %   1 0 1
===========
```

In de eenhedenkolommen voeren we de aftrekking %1 - %0 = %1 uit. Daarna voeren we in de kolommen van de tweeën nog een aftrekking uit: %1 - %1 = %0. Als laatste trekken we de kolommen van de vieren van elkaar af. Deze keer is %0 kleiner dan %1, waardoor we een %1 moeten lenen uit de kolom van de achten van het bovenste getal om de aftrekking uit te voeren. Daartoe berekenen we %10 - %1 = %1 en halen we %1 af van de kolom van de achten. De kolommen van de achten zijn nu beide nul. Omdat %0 - %0 = %0 en dit cijfer het voorloopcijfer van het resultaat is, mogen we het weglaten in het eindantwoord. Dit levert %101 als resultaat op.

Laten we nu de aftrekking van twee hexadecimale getallen bekijken.

```
  $ 3 D
- $ 1 F
-------
  $ 1 E
=======
```

Om deze aftrekking uit te voeren, berekenen we het verschil van de eenhedenkolommen. Daarbij merken we op dat we, omdat $D kleiner is dan $F, $1 moeten lenen uit de 16-kolom van het bovenste getal om de aftrekking te kunnen maken. We berekenen daartoe $1D - $F = $E en tevens $3 - $1 = $2 in de 16-kolom, ter verrekening van de $1 die we zojuist hebben geleend. Vervolgens berekenen we het verschil van de 16-kolom als $2 - $1 = $1. Dit levert als eindantwoord $1E op.

We geven geen uitvoerige voorbeelden van vermenigvuldiging en deling in binaire en hexadecimale notatie. Het volstaat te zeggen dat de beginselen overeenkomen met die van het decimale stelsel: vermenigvuldigen is herhaald optellen en delen is herhaald aftrekken.

We wijzen u echter wel op een bijzondere vorm van vermenigvuldiging en deling, zowel voor binair als voor hexadecimaal. Deze is met name nuttig voor het bewerken van binaire en hexadecimale getallen.

Voor binair is vermenigvuldigen met twee eenvoudig—schuif alle bits één positie naar links en vul het minst significante bit in met een %0. Delen door twee is eveneens eenvoudig—schuif alle bits één positie naar rechts en vul het meest significante bit in met een %0. Door deze bewerkingen herhaaldelijk uit te voeren, kunnen we moeiteloos vermenigvuldigen met en delen door machten van twee.

Zo krijgt het binaire getal %111, met acht vermenigvuldigd, drie extra nullen aan het eind en is het gelijk aan %111000. (Denk eraan dat $2^3 = 8$.) En het binaire getal %10100 krijgt, door vier gedeeld, twee cijfers minder en is gelijk aan %101. (Denk eraan dat $2^2 = 4$.)

Deze bewerkingen heten *bitschuivingen* naar links en naar rechts. Als we dus zeggen dat we een getal vier bitposities naar links schuiven, dan bedoelen we daarmee in werkelijkheid dat we het met $2^4 = 16$ hebben vermenigvuldigd.

Voor hexadecimaal is de situatie vergelijkbaar. Vermenigvuldigen met zestien is eenvoudig—schuif alle cijfers één positie naar links en vul het meest rechtse cijfer in met een $0. Delen door zestien is eveneens eenvoudig—schuif alle cijfers één positie naar rechts. Door dit herhaaldelijk uit te voeren, kunnen we moeiteloos vermenigvuldigen met en delen door machten van zestien.

Zo krijgt het hexadecimale getal $F, met 256 vermenigvuldigd, twee extra nullen aan het eind en is het gelijk aan $F00. (Denk eraan dat $16^2 = 256$.) En het hexadecimale getal $EA0 krijgt, door zestien gedeeld, één cijfer minder en is gelijk aan $EA. (Denk eraan dat $16^1 = 16$.)

Logische poorten

Er bestaan verschillende zogenaamde *logische poorten*. De fundamentele poorten zijn NOT, AND, OR en XOR.

Met behulp van deze poorten kunnen we specifieke binaire cijfers zetten, wissen en inverteren. Werken we bijvoorbeeld met sprites, dan willen we wellicht voor een bepaald register van een grafische chip tegelijkertijd bit 6 wissen (d.w.z. het gelijk maken aan 0) en bit 1 zetten (d.w.z. het gelijk maken aan 1). Bepaalde logische poorten maken dit, in combinatie gebruikt, mogelijk.

Het is van groot belang om te leren hoe deze logische poorten werken, want zij vormen de sleutel tot het begrijpen van hoe en waarom de computer programma's uitvoert zoals hij dat doet.

Alle logische poorten aanvaarden één of meer invoeren en leveren precies één uitvoer op. Deze invoeren en uitvoeren zijn altijd losse binaire cijfers (d.w.z. het zijn 1-bits getallen).

De NOT-poort is de enige poort die precies één bit als invoer aanvaardt. Alle andere poorten—AND, OR en XOR—aanvaarden precies twee bits als invoer. Alle poorten leveren precies één uitvoer op en die uitvoer is één enkele bit.

Laten we eerst kijken naar de eenvoudigste poort, de NOT-poort.

De NOT-poort werkt door het invoerbit te inverteren en dit resulterende bit als uitvoer terug te geven. Dit is samengevat in de volgende tabel.

INVOER X	UITVOER
0	1
1	0

We schrijven NOT x, waarbij x het invoerbit is.

Vervolgens bekijken we de AND-poort.

Zoals eerder vermeld, aanvaardt de AND-poort twee bits als invoer en levert één bit als uitvoer op. De AND-poort gedraagt zich als volgt. Zijn beide invoerbits gelijk aan 1, dan is het resultaat van het uitvoerbit 1. In alle andere gevallen is het resultaat van het uitvoerbit 0. Dit is samengevat in de volgende tabel.

INVOER X	INVOER Y	UITVOER
0	0	0
0	1	0
1	0	0
1	1	1

We schrijven x AND y, waarbij x en y de invoerbits zijn.

Vervolgens bekijken we de OR-poort.

De OR-poort aanvaardt twee bits als invoer en levert één bit als uitvoer op. De OR-poort gedraagt zich als volgt. Zijn beide invoerbits gelijk aan 0, dan is het resultaat 0. In alle andere gevallen is het resultaat van het uitvoerbit 1. Dit is samengevat in de volgende tabel.

INVOER X	INVOER Y	UITVOER
0	0	0
0	1	1
1	0	1
1	1	1

We schrijven x OR y, waarbij x en y de invoerbits zijn.

Als laatste bekijken we de XOR-poort.

De XOR-poort aanvaardt twee bits als invoer en levert één bit als uitvoer op. De XOR-poort gedraagt zich als volgt. Hebben beide invoerbits dezelfde waarde, dan is het

uitvoerbit 0. Is dat niet het geval, dan zijn beide invoerbits ongelijk van waarde en is het uitvoerbit 1. Dit is samengevat in de volgende tabel.

INVOER X	INVOER Y	UITVOER
0	0	0
0	1	1
1	0	1
1	1	0

We schrijven x XOR y, waarbij x en y de invoerbits zijn.

Merk op dat er nog enkele andere poorten bestaan. Die zijn eenvoudig op te bouwen.

- NAND-poort: dit is een AND-poort gevolgd door een NOT-poort
- NOR-poort: dit is een OR-poort gevolgd door een NOT-poort
- XNOR-poort: dit is een XOR-poort gevolgd door een NOT-poort

GEHELE EN NATUURLIJKE GETALLEN

Tot nu toe hebben we ons grotendeels op natuurlijke getallen gericht. Natuurlijke getallen hebben geen positief of negatief teken. We gaan er altijd van uit dat zij positief zijn. (Nul geldt hierbij als positief.)

Gehele getallen kunnen, zoals eerder vermeld, zowel een positief als een negatief teken hebben.

Gehele getallen worden weergegeven door het meest significante bit als tekenbit te behandelen. Dit bit kan voor niets anders worden gebruikt. Is het meest significante bit 0, dan interpreteren we het resultaat als een getal met een positief teken. Anders is het meest significante bit 1 en interpreteren we het resultaat als een getal met een negatief teken.

Een 8-bits geheel getal kan getallen met een positief teken tussen 0 en 127 weergeven en getallen met een negatief teken tussen -1 en -128.

Een 16-bits geheel getal kan getallen met een positief teken tussen 0 en 32767 weergeven en getallen met een negatief teken tussen -1 en -32768.

Het reserveren van het meest significante bit als teken van het gehele getal halveert in feite het bereik van de beschikbare positieve getallen (d.w.z. in vergelijking met natuurlijke getallen); daarvoor krijgen we echter een even grote hoeveelheid negatieve getallen terug.

Om de negatie van een geheel getal te bepalen, moeten we elk bit in dat getal inverteren en daarna %1 bij het resultaat optellen. Zo geeft de negatie van %0000 0101 (het gehele getal +5) %1111 1011 als resultaat (het gehele getal -5). Zoals verwacht verkrijgen we door de negatie van dit negatieve getal weer +5.

BITGEWIJZE LOGISCHE OPERATOREN

De BASIC-statements **NOT**, **AND**, **OR** en **XOR** hebben een werking die lijkt op die van de logische poorten waarnaar zij zijn genoemd.

Het **NOT**-statement moet een 16-bits decimaal geheel getal als parameter krijgen. Het geeft als resultaat een 16-bits decimaal geheel getal terug.

In het volgende voorbeeld zijn alle zestien bits van het decimale gehele getal +0 gelijk aan 0. Het **NOT**-statement inverteert alle zestien bits overeenkomstig de NOT-poort. Daardoor worden alle zestien bits gezet. Interpreteren we het resultaat als decimaal geheel getal, dan verkrijgen we als antwoord -1.

```
PRINT (NOT 0)
-1
```

Zoals verwacht zijn we, wanneer we het **NOT**-statement opnieuw op de parameter -1 toepassen, terug bij het beginpunt, omdat alle zestien gezette bits worden gewist.

```
PRINT (NOT -1)
 0
```

Het **AND**-statement moet twee 16-bits decimale gehele getallen als parameters krijgen. De tweede parameter heet het *bitmasker*. Het statement geeft als resultaat een 16-bits decimaal geheel getal terug, nadat het elk bit overeenkomstig de AND-poort heeft gewijzigd.

In het volgende voorbeeld dient het getal +253 als eerste parameter. Als 16-bits decimaal geheel getal is dit gelijkwaardig aan het volgende binaire getal: %0000 0000 1111 1101. Het **AND**-statement gebruikt als tweede parameter een bitmasker met de 16-bits decimale gehele waarde +239. In binaire notatie is dit het getal %0000 0000 1110 1110. Passen we de tabel van de AND-poort toe op de overeenkomstige paren van bits, dan verkrijgen we het 16-bits decimale gehele getal +237 (wat in binaire notatie %0000 0000 1110 1100 is).

```
PRINT (253 AND 239)
 237
```

In de volgende tabel is dit proces duidelijker te zien.

	%	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1
AND	%	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0
	%	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0

Merk op dat elk bit in de bovenste rij ongewijzigd blijft op iedere positie waar het maskerbit eronder een 1 bevat. Zo niet, dan wordt het bit op die positie gewist.

Het **OR**-statement moet twee 16-bits decimale gehele getallen als parameters krijgen. De tweede parameter heet het *bitmasker*. Het statement geeft als resultaat een 16-bits decimaal geheel getal terug, nadat het elk bit overeenkomstig de OR-poort heeft gewijzigd.

In het volgende voorbeeld dient het getal +240 als eerste parameter. Als 16-bits decimaal geheel getal is dit gelijkwaardig aan het volgende binaire getal: %0000 0000 1111 0000. Het **OR**-statement gebruikt als tweede parameter een bitmasker met de 16-bits decimale gehele waarde +19. In binaire notatie is dit het getal %0000 0000 0001 0011. Passen we de tabel van de OR-poort toe op de overeenkomstige paren van bits, dan verkrijgen we het 16-bits decimale gehele getal +243 (wat in binaire notatie %0000 0000 1111 0011 is).

```
PRINT (240 OR 19)
 243
```

In de volgende tabel is dit proces duidelijker te zien.

	%	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
OR	%	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
	%	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1

Merk op dat elk bit in de bovenste rij ongewijzigd doorgaat, overal waar in het maskerbit eronder een 0 staat. Anders wordt het bit op die positie gezet.

Nu bekijken we het **XOR**-statement. Dit statement vereist twee 16-bits decimale natuurlijke getallen als parameters. De tweede parameter heet het *bitmasker*. Het statement geeft een 16-bits decimaal natuurlijk getal als resultaat terug, waarbij elk bit volgens de XOR-poort is gewijzigd.

In het volgende voorbeeld gebruiken we het getal 14091 als eerste parameter. Als 16-bits decimaal natuurlijk getal komt dit overeen met het volgende binaire getal: %0011 0111 0000 1011. Het **XOR**-statement gebruikt als tweede parameter een bitmasker met een 16-bits decimale natuurlijke waarde van 8653. In binaire notatie is dit het getal %0010 0001 1100 1101. Passen we de tabel van de XOR-poort toe op de overeenkomstige paren van bits, dan verkrijgen we het 16-bits decimale natuurlijke getal 5830 (in binaire notatie: %0001 0110 1100 0110).

```
PRINT (XOR(14091,8653))
5830
```

In de volgende tabel is dit proces duidelijker te zien.

	%	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1	1
XOR	%	0	0	1	0	0	0	0	1	1	1	0	0	1	1	0	1
	%	0	0	0	1	0	1	1	0	1	1	0	0	0	1	1	0

Merk op dat het resulterende bit 0 is wanneer de bits gelijk zijn. Anders is het resulterende bit 1.

Veel van het nut van deze bitgewijze logische operatoren komt voort uit het combineren ervan tot één samengesteld statement. Het VIC-II-register om sprites in te schakelen bevindt zich bijvoorbeeld op geheugenadres 53269. Er zijn acht sprites (genummerd van 0 tot 7), waarbij elk bit overeenkomt met de status van een sprite. Stel nu dat we sprite 5 willen uitschakelen en sprite 1 willen inschakelen, waarbij de statussen van de andere sprites ongewijzigd blijven. Dat kunnen we doen met het volgende BASIC-statement, dat een **AND**-statement combineert met een **OR**-statement.

```
POKE 53269, (((PEEK(53269)) AND 223) OR 2)
```

De techniek om **PEEK** op een geheugenadres toe te passen, het resultaat met bitgewijze logische operatoren te combineren en daarna een **POKE** naar datzelfde geheugenadres uit te voeren, is zeer gebruikelijk.

GETALLEN OMREKENEN

Het onderstaande programma is geschreven in BASIC. Het zet getallen voor u om. Typ het in en sla het op onder de naam "CONVERT.BAS".

Om het programma uit te voeren, typt u **RUN** in en drukt u op RETURN.

Het programma biedt u een reeks tekstmenu's aan. U kunt ervoor kiezen om één decimaal, binair of hexadecimaal getal om te rekenen. U kunt er ook voor kiezen om een bereik van zulke getallen om te rekenen.

Het programma kan getallen in het bereik van 0 tot en met 65535 omrekenen.

```
10 REM ***************************
20 REM *                         *
30 REM *  GETALOMREKENPROGRAMMA  *
40 REM *                         *
50 REM ***************************
60 POKE 0,65: BORDER 6: BACKGROUND 6: FOREGROUND 1
70 DIM P(15)
80 E$ = "START-INTEGER MOET KLEINER ZIJN DAN EIND-INTEGER"
90 FOR N = 0 TO 15
100 :  P(N) = 2 ↑ N
110 NEXT N
120 REM *** TOON HET HOOFDMENU ***
130 PRINT CHR$(147)
140 PRINT: PRINT "GETALOMREKENPROGRAMMA"
150 L = 22: GOSUB 1930: PRINT L$
160 PRINT: PRINT "MAAK EEN KEUS (E, M OF B):": PRINT
170 PRINT "[S](SPACE*2)CONVERSIE VAN EEN ENKELE INTEGER"
180 PRINT "[M](SPACE*2)CONVERSIE VAN MEERDERE INTEGERS"
190 PRINT "[Q](SPACE*2)BEEINDIG PROGRAMMA"
200 GET M$
210 IF (M$="E") THEN GOSUB 260: GOTO 140
220 IF (M$="M") THEN GOSUB 380: GOTO 140
230 IF (M$="M") THEN END
240 GOTO 200
250 REM *** TOON MENU VOOR ENKELE CONVERSIE ***
260 PRINT: PRINT "(SPACE*2)MAAK EEN KEUS (D, B, H OF T):": PRINT
270 PRINT "(SPACE*2)[D](SPACE*2)CONVERTEER EEN DECIMALE INTEGER"
280 PRINT "(SPACE*2)[B](SPACE*2)CONVERTEER EEN BINAIRE INTEGER"
290 PRINT "(SPACE*2)[H](SPACE*2)CONVERTEER EEN HEXADECIMALE INTEGER"
300 PRINT "(SPACE*2)[T](SPACE*2)TERUG NAAR HOOFDMENU"
310 GET M1$
320 IF (M1$="D") THEN GOSUB 500: GOTO 260
330 IF (M1$="B") THEN GOSUB 760: GOTO 260
340 IF (M1$="H") THEN GOSUB 810: GOTO 260
350 IF (M1$="T") THEN RETURN
360 GOTO 310
370 REM *** TOON MENU VOOR MEERVOUDIGE CONVERSIE ***
380 PRINT: PRINT "(SPACE*2)SELECT AN OPTION (D, B, H OF T):": PRINT
390 PRINT "(SPACE*2)[D](SPACE*2)CONVERTEER EEN REEKS DECIMALE INTEGERS"
400 PRINT "(SPACE*2)[B](SPACE*2)CONVERTEER EEN REEKS BINAIRE INTEGERS"
410 PRINT "(SPACE*2)[H](SPACE*2)CONVERTEER EEN REEKS HEXADECIMALE INTEGERS"
```

```
420 PRINT "{SPACE*2}[R]{SPACE*2}TERUG NAAR HOOFDMENU"
430 GET M2$
440 IF (M2$="D") THEN GOSUB 1280: GOTO 380
450 IF (M2$="B") THEN GOSUB 1670: GOTO 380
460 IF (M2$="H") THEN GOSUB 1800: GOTO 380
470 IF (M2$="R") THEN RETURN
480 GOTO 430
490 REM *** CONVERTEER ENKELE DECIMALE INTEGER ***
500 D$ = ""
510 PRINT: INPUT "VOER DECIMALE INTEGER IN (T/M 65535): ",D$
520 GOSUB 1030: REM VALIDEER DECIMALE INVOER
530 IF (V = 0) THEN GOTO 510
540 PRINT: PRINT " DEC";SPC(4);"BIN";SPC(19);"HEX"
550 L = 5: GOSUB 1930: L1$ = L$
560 L = 20: GOSUB 1930: L2$ = L$
570 PRINT SPC(1);L1$;SPC(2);L2$;SPC(2);L1$
580 FOREGROUND 7
590 B$ = ""
600 D1 = 0
610 IF (D < 256) THEN GOTO 660
620 D1 = INT(D / 256)
630 FOR N = 1 TO 8
640 :  IF ((D1 AND P(8 - N)) > 0) THEN B$ = B$ + "1": ELSE B$ = B$ + "0"
650 NEXT N
660 IF (D < 256) THEN B$ = "%" + B$: ELSE B$ = "%" + B$ + " "
670 D2 = D - 256*D1
680 FOR N = 1 TO 8
690 :  IF ((D2 AND P(8 - N)) > 0) THEN B$ = B$ + "1": ELSE B$ = B$ + "0"
700 NEXT N
710 H$ = HEX$(D)
720 IF (D < 256) THEN H$ = "{SPACE*2}$" + RIGHT$(H$,2): ELSE H$ = "$" + H$
730 IF (D < 256) THEN PRINT SPC(6 - LEN(D$)); D$;SPC(12) + MID$(B$,1,5) +
" " + MID$(B$,6,10); "{SPACE*2}" + H$: FOREGROUND 1: RETURN
740 PRINT SPC(6 - LEN(D$));D$;"{SPACE*2}" + MID$(B$,1,5) + " " + MID$(B$,6,4) +
MID$(B$,10,5) + " " + MID$(B$,15,4); "{SPACE*2}" + H$: FOREGROUND 1: RETURN
750 REM *** CONVERTEER ENKELE BINAIRE INTEGER ***
760 I$=""
770 PRINT: INPUT "VOER BINAIRE INTEGER IN (T/M 16 BITS): ",I$
780 GOSUB 1110: REM VALIDEER BINAIRE INVOER
790 IF (V = 0) THEN GOTO 760: ELSE GOTO 540
800 REM *** CONVERTER ENKELE HEXADECIMALE INTEGER ***
```

```
810 H$=""
820 PRINT: INPUT "ENTER HEXADECIMAL INTEGER (UP TO 4 DIGITS): ",H$
830 GOSUB 1220: REM VALIDATE HEXADECIMAL INPUT
840 IF (V = 0) THEN GOTO 810: ELSE GOTO 540
850 REM *** VALIDATE DECIMAL INPUT STRING ***
860 FOR N = 1 TO LEN(D$)
870 :  M = ASC(MID$(D$,N,1)) - ASC("0")
880 :  IF ((M < 0) OR (M > 9)) THEN V = 0
890 NEXT N: RETURN
900 REM *** VALIDATE BINARY INPUT STRING ***
910 FOR N = 1 TO LEN(I$)
920 :  M = ASC(MID$(I$,N,1)) - ASC("0")
930 :  IF ((M < 0) OR (M > 1)) THEN V = 0
940 NEXT N: RETURN
950 REM *** VALIDEER HEXADECIMALE INVOERSTRING ***
960 FOR N = 1 TO LEN(H$)
970 :  M = ASC(MID$(H$,N,1)) - ASC("0")
980 :  IF (NOT (((M >= 0) AND (M <= 9)) OR
((M >= 17) AND (M <= 22)))) THEN V = 0
990 NEXT N: RETURN
1000 REM *** TOON FOUTMELDING ***
1010 FOREGROUND 2: PRINT: PRINT A$: FOREGROUND 1: RETURN
1020 REM *** VALIDEER DECIMALE INVOER ***
1030 V = 1: GOSUB 860: REM VALIDEER DECIMALE INVOERSTRING
1040 IF (V = 0) THEN A$ = "ONGELDIG DECIMAAL GETAL": GOSUB 1010
1050 IF (V = 1) THEN BEGIN
1060 :  D = VAL(D$)
1070 :  IF ((D < 0) OR (D > 65535)) THEN A$ = "DECIMAAL GETAL BUITEN BEREIK":
GOSUB 1010: V = 0
1080 BEND
1090 RETURN
1100 REM *** VALIDEER BINAIRE INVOER ***
1110 V = 1: GOSUB 910: REM VALIDEER BINAIRE INVOERSTRING
1120 IF (V = 0) THEN A$ = "ONGELDIG BINAIR GETAL": GOSUB 1010: RETURN
1130 IF (LEN(I$) > 16) THEN A$ = "BINAIRE GETAL BUITEN BEREIK":
GOSUB 1010: V = 0 : RETURN
1140 IF (V = 1) THEN BEGIN
1150 :  I = 0
1160 :  FOR N = 1 TO LEN(I$)
1170 :    I = I + VAL(MID$(I$,N,1)) * P(LEN(I$) - N)
1180 :  NEXT N
```

```
1190 BEND
1200 D$ = STR$(I): D = I: RETURN
1210 REM *** VALIDATE HEXADECIMALE INVOER ***
1220 V = 1: GOSUB 960: REM VALIDEER HEXADECIMALE INVOERSTRING
1230 IF (V = 0) THEN A$ = "ONGELDIG HEXADECIMAAL GETAL": GOSUB 1010: RETURN
1240 IF (LEN(H$) > 4) THEN A$ = "HEXADECIMAAL GETAL BUITEN BEREIK":
GOSUB 1010: V = 0: RETURN
1250 D = DEC(H$): D$ = STR$(D): H = D: RETURN
1260 RETURN
1270 REM *** CONVERTEER MEERDERE DECIMALE INTEGERS ***
1280 DB$=""
1290 PRINT: INPUT "VOER BEGIN IN ALS DECIMALE INTEGER (T/M 65535): ", DB$
1300 D$=DB$: GOSUB 1030: D$="": REM VALIDATE DECIMAL INPUT
1310 IF (V = 0) THEN GOTO 1290
1320 DE$=""
1330 PRINT: INPUT "VOER EIND IN ALS DECIMALE INTEGER (T/M 65535): ", DE$
1340 D$=DE$: GOSUB 1030: D$="": REM VALIDATE DECIMAL INPUT
1350 IF (V = 0) THEN GOTO 1330
1360 DB=VAL(DB$): DE=VAL(DE$)
1370 IF (DE < DB) THEN A$ = E$: GOSUB 1010: GOTO 1280
1380 SC = 1: SM = INT(((DE - DB) / 36) + 1)
1390 D = DB
1400 FOR J = SC TO SM
1410 :  PRINT CHR$(147) + "RANGE: " + DB$ + " TO " + DE$ + "(SPACE*10)SCHERM: "
+ STR$(J) + " OF " + STR$(SM)
1420 :  PRINT: PRINT "DEC";SPC(4);"BIN";SPC(19);"HEX";SPC(8);"DEC";SPC(4);
"BIN";SPC(19);"HEX"
1430 L = 5: GOSUB 1930: L1$ = L$
1440 L = 20: GOSUB 1930: L2$ = L$
1450 :    PRINT SPC(1);L1$;SPC(2);L2$;SPC(2);L1$;SPC(6);L1$;SPC(2);
L2$;SPC(2);L1$
1460 :  FOR K = 0 TO 17
1470 :      FOREGROUND (7 + MOD(K,2))
1480 :      D$ = STR$(D): GOSUB 590: D = D + 1
1490 :      IF (D > DE) THEN GOTO 1630
1500 :  NEXT K
1510 :  PRINT CHR$(19): PRINT: PRINT: PRINT
1520 :  FOR K = 0 TO 17
1530 :      FOREGROUND (7 + MOD(K,2))
1540 :      D$ = STR$(D): PRINT TAB(40): GOSUB 590: D = D + 1
```

```
1550 :      IF (D ) DE) THEN GOTO 1630
1560 :  NEXT K
1570 :  FOREGROUND 1: PRINT: PRINT SPC(19);"DRUK OP E VOOR EINDE OF SPATIE OM DOOR TE GAAN..."
1580 :  GET B$
1590 :  IF B$="E" THEN RETURN
1600 :  IF B$=" " THEN GOTO 1620
1610 :  GOTO 1580
1620 NEXT J
1630 PRINT CHR$(19): FOR I = 1 TO 22: PRINT: NEXT I
1640 PRINT SPC(20); "KLAAR. DRUK OP SPATIE OM DOOR TE GAAN..."
1650 GET B$: IF B$()" " THEN GOTO 1650: ELSE RETURN
1660 REM *** CONVERTEER MEERDERE BINAIRE INTEGERS ***
1670 IB$=""
1680 PRINT: INPUT "VOER START IN ALS BINAIRE INTEGER (T/M 16 BITS): ", IB$
1690 I$=IB$: GOSUB 1110: I$="": REM VALIDEEER BINARIRE INVOER
1700 IF (V = 0) THEN GOTO 1680
1710 IB = I
1720 IE$=""
1730 PRINT: INPUT "VOER EIND IN ALS BINAIRE INTEGER (UP TO 16 BITS): ", IE$
1740 I$=IE$: GOSUB 1110: I$="": REM VALIDEER BINAIRE INVOER
1750 IF (V = 0) THEN GOTO 1730
1760 IE = I
1770 IF (IE < IB) THEN A$ = E$: GOSUB 1010: GOTO 1670
1780 DB = IB: DE = IE: DB$ = STR$(IB): DE$ = STR$(IE): GOTO 1380
1790 REM *** CONVERTEER MEERDERE HEXADECIMALE INTEGERS ***
1800 HB$=""
1810 PRINT: INPUT "ENTER STARTING HEXADECIMAL INTEGER (UP TO 4 DIGITS): ", HB$
1820 H$=HB$: GOSUB 1220: H$="": REM VALIDEER HEXADECIMALE INVOER
1830 IF (V = 0) THEN GOTO 1810
1840 HB = H
1850 HE$=""
1860 PRINT: INPUT "ENTER ENDING HEXADECIMAL INTEGER (UP TO 4 DIGITS): ", HE$
1870 H$=HE$: GOSUB 1220: H$="": REM VALIDEER HEXADECIMALE INVOER
1880 IF (V = 0) THEN GOTO 1860
1890 HE = H
1900 IF (HE < HB) THEN A$ = E$: GOSUB 1010: GOTO 1800
1910 DB = HB: DE = HE: DB$ = STR$(HB): DE$ = STR$(HE): GOTO 1380
1920 REM *** MAKE LINES ***
1930 L$=""
1940 FOR K = 1 TO L: L$ = L$ + "-": NEXT K
1950 RETURN
```

BIJLAGE I

KERNAL-sprongtabel

- Gebruik van de KERNAL-sprongtabel
- Snelnaslag van de MEGA65-KERNAL-sprongtabel
- Naslagwerk van de MEGA65-KERNAL-sprongtabel

GEBRUIK VAN DE KERNAL-SPRONGTABEL

De *KERNAL* vormt het hart van het besturingssysteem van de MEGA65. De KERNAL stelt het besturingssysteem in wanneer de computer wordt ingeschakeld of gereset, onderhoudt de communicatie met randapparaten, voert schijfbewerkingen uit, beheert de in- en uitvoerstromen en stuurt de kernfuncties van de schermeditor en BASIC aan. Zolang de KERNAL actief is, beheert hij de geheugenindeling van de CPU en de interrupthandlers en reserveert hij geheugengebieden om zijn functies te ondersteunen.

De KERNAL biedt een *Application Programming Interface* (API), zodat machinetaalprogramma's door subroutines aan te roepen van zijn functies gebruik kunnen maken, bijvoorbeeld om een bestand van schijf te lezen. Een programma roept deze subroutines aan via adressen in een *sprongtabel*: een gebied van ROM-code waarin de API garandeert dat de adressen altijd op dezelfde wijze naar dezelfde subroutines verwijzen, zelfs bij verschillende versies van de ROM-code. Het programma gebruikt `jsr` of `jmp` (afhankelijk van wat de routine vereist), met het sprongtabeladres als absoluut doel. Het geheugen op het sprongtabeladres bevat een `jmp`-instructie naar het juiste interne adres.

Om bijvoorbeeld een teken naar de uitvoerstroom te schrijven (standaard de schermterminal), stelt een programma het accumulatorregister (A) in op de PETSCII-code van het te schrijven teken en roept vervolgens de subroutine CHROUT aan met het sprongtabeladres $FFD2:

```
lda #$41
jsr $ffd2
```

Randvoorwaarden voor sprongtabelroutines

Alle KERNAL-sprongtabelroutines verwachten dat bij de aanroep aan de volgende voorwaarden is voldaan:

1. Het 16-bits adresbereik $E000 – $FFFF moet gemapt zijn op de KERNAL-ROM op 3.E000 – 3.FFFF.
2. Het 16-bits adresbereik $0000 – $1FFF moet gemapt zijn op het interne variabelengeheugen van de KERNAL op 0.0000 – 0.1FFF.
3. Het basispaginaregister (B) van de CPU moet $00 zijn.

Sommige sprongtabelroutines hebben aanvullende randvoorwaarden. Zie de naslag hieronder.

De geheugenindelingen KERNAL BASIC en SYS handhaven deze voorwaarden standaard. Indien een machinetaalprogramma de registers MAP of B niet wijzigt, kan het zonder aanvullende voorbereiding KERNAL-sprongtabelroutines aanroepen.

De KERNAL maakt gebruik van andere geheugengebieden. Zie hoofdstuk 15.

Tenzij anders vermeld, moet een programma ervan uitgaan dat een aanroep van een KERNAL-routine de gegevens in alle CPU-registers en in statusvlaggen die met waarden te maken hebben overschrijft. Gevolgen voor de stack en voor modusgerelateerde statusvlaggen (interrupts) worden expliciet vermeld.

Geschiedenis en compatibiliteit

De KERNAL-sprongtabel van de MEGA65 is gebaseerd op de sprongtabel van de Commodore 65 en daarmee compatibel. Deze API is afgeleid van de KERNAL-API's van de Commodore 64 en de Commodore 128, maar is *niet* volledig compatibel.

Deze documentatie neemt, voor zover mogelijk, de routinenamen uit de documentatie van de Commodore 65 over en verzint unieke namen voor routines die nieuw zijn op de MEGA65. Een routine met een naam die lijkt op een naam uit de C64- of C128-documentatie is *niet* noodzakelijk API-compatibel. Voor zover bekend, worden verschillen in deze naslag vermeld.

De "GO64"-modus van de MEGA65 gebruikt de C64-KERNAL. Raadpleeg een naslagwerk over C64-programmering voor informatie over het gebruik van de C64-KERNAL.

Voor een volledige geschiedenis van de KERNAL-sprongtabel op alle Commodore-computers, zie: https://www.pagetable.com/?p=926

SNELNASLAG VAN DE MEGA65-KERNAL-SPRONGTABEL

Hieronder volgt een snelnaslag van de MEGA65-KERNAL-sprongtabel, in adresvolgorde. Raadpleeg de volledige naslag voor details en voorbeelden.

Adres	Categorie	Naam	Beschrijving
$FF2F	Besturingssysteem	VERSIONQ	Laadt het ROM-versienummer in Q
$FF32	Besturingssysteem	RESET_RUN	Herstart de computer op verschillende manieren
$FF35	Editor	CURSOR	Schakelt de cursor in of uit
$FF38	Gereserveerd	n/a	GERESERVEERD, NIET GEBRUIKEN
$FF3B	Bestanden	SAVEFL	Slaat op in een bestand, met vlaggen
$FF3E	Gereserveerd	n/a	GERESERVEERD, NIET GEBRUIKEN
$FF41	I/O	GETIO	Leest de huidige invoer- en uitvoerapparaten
$FF44	Bestanden	GETLFS	Leest bestand, apparaat en secundair adres
$FF47	Editor	SYSFLAGS	Leest/zet toetsenbordvergrendelingen
$FF4A	Editor	ADDKEY	Voegt een teken toe aan de invoerbuffer van het softwaretoetsenbord
$FF4D	Gereserveerd	SPIN_SPOUT	Stelt de snelle seriële poorten in
$FF50	Bestanden	CLOSE_ALL	Sluit alle bestanden op een apparaat
$FF53	Besturingssysteem	C64MODE	Zet de computer terug naar de GO64-modus
$FF56	Besturingssysteem	MonitorCall	Roept de monitor aan
$FF59	Besturingssysteem	BOOT_SYS	Start een alternatief systeem vanaf schijf op
$FF5C	Besturingssysteem	PHOENIX	Roept de koude start van een cartridge of de schijf-bootloader aan
$FF5F	Bestanden	LKUPLA	Zoekt een logisch bestandsnummer dat in gebruik is
$FF62	Bestanden	LKUPSA	Zoekt een secundair adres dat in gebruik is
$FF65	Editor	SWAPPER	Wisselt tussen de tekstmodi 40x25 en 80x25
$FF68	Editor	PFKEY	Programmeert een functietoets van de editor
$FF6B	Bestanden	SETBNK	Stelt de bank in voor I/O en bestandsnaamgeheugen
$FF6E	Geheugen	JSRFAR	Roept een subroutine in een willekeurige bank aan
$FF71	Geheugen	JMPFAR	Springt naar een adres in een willekeurige bank
$FF74	Geheugen	LDA_FAR	Leest een byte van een adres in een willekeurige bank
$FF77	Geheugen	STA_FAR	Schrijft een byte naar een adres in een willekeurige bank
$FF7A	Geheugen	CMP_FAR	Vergelijkt een byte met een adres in een willekeurige bank
$FF7D	Hulpmiddelen	PRIMM	Drukt een inline, met nul afgesloten korte string af
$FF81	Editor	CINT	Initialiseert de schermeditor
$FF84	Besturingssysteem	IOINIT	Initialiseert I/O-apparaten
$FF87	Besturingssysteem	RAMTAS	Initialiseert het RAM-geheugen en de buffers
$FF8A	Besturingssysteem	RESTOR	Initialiseert de KERNAL-vectortabel
$FF8D	Besturingssysteem	VECTOR	Leest/stelt de KERNAL-vectortabel in
$FF90	Besturingssysteem	SETMSG	Schakelt KERNAL-meldingen in of uit
$FF93	Serieel	SECND	Stuurt een secundair adres naar de listener
$FF96	Serieel	TKSA	Stuurt een secundair adres naar de talker
$FF99	Gereserveerd	MEMTOP	GERESERVEERD, NIET GEBRUIKEN
$FF9C	Gereserveerd	MEMBOT	GERESERVEERD, NIET GEBRUIKEN
$FF9F	Hulpmiddelen	KEY	Scant het toetsenbord

vervolg ...

...vervolg

Adres	Categorie	Naam	Beschrijving
$FFA2	Besturingssysteem	MONEXIT	Verlaat de monitor en keert terug naar BASIC
$FFA5	Serieel	ACPTR	Aanvaardt een byte van de talker
$FFA8	Serieel	CIOUT	Stuurt een byte naar de listener
$FFAB	Serieel	UNTLK	Stuurt het "untalk"-commando
$FFAE	Serieel	UNLSN	Stuurt het "unlisten"-commando
$FFB1	Serieel	LISTN	Stuurt het "listen"-commando
$FFB4	Serieel	TALK	Stuurt het "talk"-commando
$FFB7	I/O	READSS	Geeft de status van de laatste I/O-bewerking terug
$FFBA	Bestanden	SETLFS	Stelt bestand, apparaat en secundair adres in
$FFBD	Bestanden	SETNAM	Stelt de verwijzingen naar de bestandsnaam in
$FFC0	Bestanden	OPEN	Opent een logisch bestand
$FFC3	Bestanden	CLOSE	Sluit een logisch bestand
$FFC6	I/O	CHKIN	Stelt het invoerkanaal in
$FFC9	I/O	CKOUT	Stelt het uitvoerkanaal in
$FFCC	I/O	CLRCH	Herstelt de standaardkanalen
$FFCF	I/O	BASIN	Leest een teken van het invoerapparaat
$FFD2	I/O	BSOUT	Schrijft een teken naar het uitvoerapparaat
$FFD5	Bestanden	LOAD	Laadt/verifieert uit bestand
$FFD8	Bestanden	SAVE	Slaat op in een bestand
$FFDB	Hulpmiddelen	SETTIM	Stelt de 24-uursklok van CIA1 in
$FFDE	Hulpmiddelen	RDTIM	Leest de 24-uursklok van CIA1
$FFE1	Hulpmiddelen	STOP	Rapporteert de status van de Stop-toets (zie ScanStopKey)
$FFE4	I/O	GETIN	Leest een teken van het invoerapparaat, zonder te wachten
$FFE7	Bestanden	CLALL	Sluit alle bestanden en kanalen
$FFEA	Hulpmiddelen	ScanStopKey	Scant de Stop-toets
$FFED	Editor	SCRORG	Vraagt de huidige afmetingen van het schermvenster op
$FFF0	Editor	PLOT	Leest/stelt de cursorpositie in
$FFF3	Gereserveerd	IOBASE	GERESERVEERD, NIET GEBRUIKEN

NASLAGWERK VAN DE MEGA65-KERNAL-SPRONGTABEL

ACPTR

Adres: JSR $FFA5

Omschrijving:
Aanvaardt een byte van de talker

Uitvoer: **A** De in ontvangst genomen byte

Opmerkingen:
Dit is een routine op laag niveau voor de seriële bus. De meeste programma's geven de voorkeur aan de I/O-routines op hoger niveau (OPEN en dergelijke).

Nadat een programma het TALK-commando naar een randapparaat op de seriële bus heeft gezonden, om het apparaat uitvoer te laten leveren, en desgewenst een secundair adres met TKSA heeft meegezonden, roept het programma ACPTR aan om de gegevens één byte tegelijk uit het apparaat te lezen. Zie TALK.

Een programma kan dit gedrag met de IACPTR-vector overschrijven of uitbreiden.

Voorbeeld:
```
acptr = $ffa5

        jsr $ffa5
        sta data
```

ADDKEY

Adres: JSR $FF4A

Omschrijving:

Voegt een teken aan de toetsenbordinvoerbuffer toe

Invoer: **A** De PETSCII-code

Uitvoer: **C-vlag** 0 bij succes, 1 bij mislukking

Opmerkingen:

De MEGA65-KERNAL leest toetsenbordgebeurtenissen uit drie bronnen: de actieve functietoetsmacro, een toetsenbordbuffer die de KERNAL beheert en de wachtrij met typegebeurtenissen van de hardware. Wanneer GETIN van het toetsenbord leest, gaat het deze bronnen achtereenvolgens na.

ADDKEY voegt een PETSCII-code aan de toetsenbordbuffer toe. GETIN verwerkt deze buffer wanneer er geen functietoetsmacro actief is. Deze buffer heeft voorrang op de wachtrij met typegebeurtenissen van de hardware.

De toetsenbordbuffer heeft een vaste grootte. Indien de buffer vol is, wordt de toets niet toegevoegd en wordt de C-vlag gezet.

OPMERKING: Bij de MEGA65 wordt de buffer uitgelezen door aanroepen van GETIN, en niet door de KERNAL-IRQ zoals bij een C64. Een programma moet naar de schermeditor terugkeren opdat GETIN wordt aangeroepen. Indien een programma meer dan enkele tekens in de wachtrij moet plaatsen, is één optie om een IIRQ-vector te gebruiken en de tekens geleidelijk aan de buffer toe te voegen.

Voorbeeld: `addkey = $ff4a`

```
; Laat RUN en een regelterugloop intypen
; zodra de schermeditor actief is.
lda #'R'
jsr addkey
bcs err
lda #'U'
jsr addkey
bcs err
lda #'N'
jsr addkey
bcs err
lda #13
jsr addkey
bcs err
```

BASIN

Adres: JSR $FFCF

Omschrijving:
Leest een teken van het invoerapparaat

Uitvoer: Bij succes:
C-vlag = 0
A Het gelezen teken

Bij een fout:
C-vlag = 1
A Foutcode, indien aanwezig

Opmerkingen:
Het invoerapparaat is standaard het toetsenbord. Een programma kan het invoerapparaat wijzigen met OPEN en CHKIN.

BASIN behandelt het invoerapparaat als gebufferd en regelgeoriënteerd, waarbij elke regel eindigt met een regelterugloop (PETSCII 13).

Indien het apparaat RS-232 of serieel is en de buffer leeg is, wacht BASIN op het volgende teken. Verkeert het apparaat in een fouttoestand, dan slaat BASIN de invoer over en retourneert het een regelterugloop.

Indien het apparaat het toetsenbord is, activeert BASIN de cursor en wacht de routine tot de gebruiker een logische regel heeft getypt en op Return heeft gedrukt. Elke aanroep van BASIN levert een teken uit de regel op, tot en met de regelterugloop. Dit plaatst de cursorpositie tevens op het laatste teken van de logische regel—maar niet erachter. (Schrijft u op dit punt een teken weg, bijvoorbeeld met BSOUT, dan overschrijft dat het laatste teken op de regel. Het is doorgaans gepast om een enkele regelterugloop naar de schermterminal te schrijven.)

Indien het apparaat het scherm is, gedraagt BASIN zich alsof de gebruiker bij toetsenbordinvoer op Return heeft gedrukt en leest het vanaf de huidige cursorpositie tot het einde van de logische regel. Het laatste teken van een logische regel is een regelterugloop.

Om te proberen een teken te lezen zonder op een lege buffer te wachten, gebruikt u GETIN.

Via de IBASIN-vector kunt u deze routine overschrijven of uitbreiden.

BASIN is vergelijkbaar met "CHRIN" op de Commodore 64.

Voorbeeld: `basin = $ffcf`

```
; Lees een regel in het geheugen, tot aan
; 255 tekens
ldy #0
```

```
read_more:
    jsr basin
    sta data,y
    iny
    beq out_of_memory
    cmp #$0d   ; regelterugloop?
    bne read_more
```

BOOT_SYS

Adres: JMP $FF59

Omschrijving:

Start een alternatief systeem vanaf een schijf op

Opmerkingen:

De Commodore 65 voorzag een functie waarbij een diskette in het interne 3.5 inch-diskettestation software kon opstarten met behulp van maximaal 512 bytes aan machinetaal in een "home"-sector, op spoor 0, sector 1.

De KERNAL leest de home-sector van de schijf wanneer de KERNAL-routine BOOT_SYS wordt aangeroepen, de gebruiker het **BOOT SYS**-commando uitvoert of de gebruiker de Alt-toets ingedrukt houdt tijdens het opstarten. Is de eerste byte $4C, dan laadt de KERNAL de sector in het geheugen op 0.0400 en roept hij deze aan met JMP. Is er geen schijf aanwezig of is de eerste byte niet $4C, dan onderneemt de KERNAL geen actie.

De bootsector-functie is iets anders dan het **BOOT**-commando van BASIC 10, dat een bestand met de naam `AUTOBOOT.C65` laadt en uitvoert. Deze functie is niet compatibel met bootsectoren in C128-stijl.

Deze routine keert niet terug. Gebruik bij deze routine uitsluitend JMP.

OPMERKING: Voor de MEGA65 is deze functie niet getest.

Voorbeeld:

```
boot_sys = $ff59

        jmp boot_sys
```

BSOUT

Adres: JSR $FFD2

Omschrijving:
Schrijft een teken naar het uitvoerapparaat

Invoer: **A** Het uit te voeren teken

Uitvoer: **C-vlag** Gezet bij een fout
A Foutcode, indien aanwezig

Opmerkingen:
Het uitvoerapparaat is standaard het scherm. Een programma kan het uitvoerapparaat wijzigen met OPEN en CKOUT.

BSOUT behandelt het uitvoerapparaat als gebufferd.

Indien het apparaat RS-232 is en de uitvoerbuffer vol is, wacht BSOUT tot er weer ruimte in de buffer is.

Indien het apparaat serieel is, gebruikt CIOUT een buffer van één teken en wordt het eerder gebufferde teken verzonden.

Indien het apparaat het scherm is, wordt het teken op de huidige cursorpositie naar het schermgeheugen geschreven.

Zie OPEN voor de lijst met mogelijke foutcodes.

Via de IBSOUT-vector kunt u deze routine overschrijven of uitbreiden.

BSOUT is vergelijkbaar met "CHROUT" op de Commodore 64.

Voorbeeld:
```
bsout = $ffd2

        lda #$0d
        jsr bsout
```

C64MODE

Adres: JMP $FF53

Omschrijving:
Zet de computer terug naar de GO64-modus

Opmerkingen:
Deze routine schakelt over naar de geheugenindeling van de GO64-modus, reset de VIC-modi en springt naar de GO64-startroutine.

Deze routine keert niet terug. Gebruik uitsluitend JMP om deze routine aan te roepen.

Voorbeeld:
```
c64mode = $ff53

        jmp c64mode
```

CHKIN

Adres: JSR $FFC6

Omschrijving:
Stelt het invoerkanaal in

Invoer: **X** Het logische adres van het invoerapparaat

Uitvoer: **C-vlag** Gezet bij een fout
A Foutcode, indien aanwezig

Opmerkingen:
Deze routine stelt het invoerkanaal in op een opgegeven apparaat, dat via zijn logische adres wordt aangeduid. Roep CHKIN aan na een aanroep van OPEN. BASIN en GETIN lezen van het huidige invoerkanaal.

Het invoerapparaat is standaard het toetsenbord. Een programma kan CLRCH aanroepen om de standaard kanaalinstellingen te herstellen.

CHKIN voert apparaatspecifieke taken uit, zoals het verzenden van een TALK-commando naar een serieel kanaal. Het instellen van een apparaat dat geen invoerapparaat is, resulteert in een fout.

Zie OPEN voor de lijst met mogelijke foutcodes.

Een programma kan deze routine met de ICHKIN-vector overschrijven of uitbreiden.

Voorbeeld:
```
chkin = $ffc6

      ldx #8
      jsr chkin
```

CINT

Adres: JSR $FF81

Omschrijving:

Initialiseert de schermeditor

Opmerkingen:

CINT reset alle eigenschappen van de schermeditor, waaronder de indirecte vectoren, de functietoetsmacro's, de VIC-registers en de SID-registers. De routine wist tevens het scherm.

CINT wordt doorgaans samen met IOINIT aangeroepen, welke aanvullende eigenschappen van de schermeditor reset.

Omdat de routine de indirecte vectoren bijwerkt, moet deze worden aangeroepen terwijl interrupts zijn uitgeschakeld.

Voorbeeld: `cint = $ff81`

```
sei
jsr cint
cli
```

CIOUT

Adres: JSR $FFA8

Omschrijving:
Stuurt een byte naar de listener

Invoer: **A** De te verzenden byte

Opmerkingen:
Dit is een routine op laag niveau voor de seriële bus. De meeste programma's geven de voorkeur aan de I/O-routines op hoger niveau (OPEN en dergelijke).

Nadat een programma het LISTN-commando naar een randapparaat op de seriële bus heeft gezonden, om het apparaat te laten luisteren naar invoer, en desgewenst een secundair adres met SECND heeft meegezonden, gebruikt het programma CIOUT om de gegevens één byte tegelijk naar het apparaat te verzenden.

Een programma kan dit gedrag met de ICIOUT-vector overschrijven of uitbreiden.

Voorbeeld:
```
ciout = $ffa8

        lda data
        jsr ciout
```

CKOUT

Adres: JSR $FFC9

Omschrijving:
Stelt het uitvoerkanaal in

Invoer: **X** Het logische adres van het uitvoerapparaat

Uitvoer: **C-vlag** Gezet bij een fout
A Foutcode, indien aanwezig

Opmerkingen:
Deze routine stelt het uitvoerkanaal in op een opgegeven apparaat, dat via zijn logische adres wordt aangeduid. Roep CKOUT aan na een aanroep van OPEN. BSOUT schrijft naar het huidige uitvoerkanaal.

Het uitvoerapparaat is standaard de schermterminal. Een programma kan CLRCH aanroepen om de standaard kanaalinstellingen te herstellen.

CKOUT voert apparaatspecifieke taken uit, zoals het verzenden van een LISTN-commando naar een serieel kanaal. Het instellen van een apparaat dat geen uitvoerapparaat is, resulteert in een fout.

Zie OPEN voor de lijst met mogelijke foutcodes.

Een programma kan deze routine met de ICKOUT-vector overschrijven of uitbreiden.

CKOUT is vergelijkbaar met "CHKOUT" op de Commodore 64.

Voorbeeld:
```
ckout = $ffc9

      ldx #8
      jsr chkout
```

CLALL

Adres: JSR $FFE7

Omschrijving:

Sluit alle bestanden en kanalen

Opmerkingen:

Deze routine sluit alle openstaande logische bestanden, alsof ze met CLOSE werden gesloten. Dit omvat het uitvoeren van apparaatspecifieke afsluitacties, evenals het herstellen van de standaard invoer- en uitvoerkanalen (alsof dit via CLRCH gebeurt).

Een programma kan dit gedrag met de ICLALL-vector overschrijven of uitbreiden.

Voorbeeld:

```
clall = $ffe7

      jsr clall
```

CLOSE

Adres: JSR $FFC3

Omschrijving:
Sluit een logisch bestand

Invoer: **A** Het te sluiten logische bestand
C-vlag Gezet om het logische bestand te verwijderen zonder het afsluit-commando te zenden

Uitvoer: **C-vlag** Gezet bij een fout
A Foutcode, indien aanwezig

Opmerkingen:
Deze routine voert apparaatspecifieke afsluitacties uit voor het opgegeven openstaande logische bestandsnummer.

Indien het bestand al gesloten is, of het logische bestandsnummer naar het toetsenbord of het scherm verwijst, keert CLOSE zonder fout terug.

Indien het apparaat RS-232 is, wacht deze routine tot de schrijfbuffer leeg is voordat wordt afgesloten.

Indien het apparaat serieel is, zendt de routine een eventueel gebufferd uitvoerteken, zendt indien van toepassing een "close"-commando en geeft het apparaat opdracht UNLSTN uit te voeren.

In een bijzonder geval, indien de C-vlag is gezet, het apparaat een schijf is (FA is 8 of hoger) en het secundaire adres het commandokanaal (15) is, wordt geen "close"-commando gezonden en wordt het logische bestand uit de bestandenlijst verwijderd. Dit is noodzakelijk om te voorkomen dat de schijf andere geopende bestanden sluit wanneer het commandokanaal wordt gesloten. Zie OPEN voor een volledig voorbeeld.

OPMERKING: Wanneer CLOSE wordt gebruikt op een bestand dat door CHKIN of CKOUT als invoer- of uitvoerkanaal was ingesteld, zet deze routine het invoer-/uitvoerkanaal *niet* terug, en een daaropvolgende lees- of schrijfactie kan zich onjuist gedragen. Gebruik CLRCH na CLOSE om de invoer- en uitvoerkanalen terug te zetten.

Zie OPEN voor de lijst met mogelijke foutcodes.

Een programma kan dit gedrag met de ICLOSE-vector overschrijven of uitbreiden.

Voorbeeld:
```
close = $ffc3
clrch = $ffcc

      clc           ; normale sluiting
      lda #$01
      jsr close
```

```
        jsr clrch     ; I/O-kanalen wissen na
                      ; het sluiten van het bestand
```

CLOSE_ALL

Adres: JSR $FF50

Omschrijving:
Sluit alle bestanden op een apparaat

Invoer: **A** Het apparaatnummer (0-31)

Opmerkingen:
Deze routine roept CLOSE aan voor elk openstaand bestand van het opgegeven apparaat (FA).

Indien één van de gesloten bestanden het huidige invoer- of uitvoerkanaal is, worden de standaardkanalen hersteld.

Voorbeeld:
```
close_all = $ff50

        lda #$08
        jsr close_all
```

CLRCH

Adres: JSR $FFCC

Omschrijving:
Herstelt de standaardkanalen

Opmerkingen:
Deze routine verbreekt alle openstaande kanalen en herstelt de standaard-I/O-kanalen van het systeem, waaronder het toetsenbord als invoerkanaal en de schermterminal als uitvoerkanaal.

Indien de huidige invoer- of uitvoerkanalen naar seriële apparaten wijzen, zendt CLRCH, naargelang het geval, UNTLK of UNLSN.

Via de ICLRCH-vector kunt u deze routine overschrijven of uitbreiden.

CLRCH is vergelijkbaar met "CLRCHN" op de Commodore 64.

Voorbeeld:
```
clrch = $ffcc

      jsr clrch
```

CMP_FAR

Adres: JSR $FF7A

Omschrijving:
Vergelijkt een byte met een adres in een willekeurige bank

Invoer: **A** De te vergelijken gegevens
X Basispagina-aanwijzer
Y Index vanaf het adres waarnaar verwezen wordt
Z Bank

Uitvoer: CPU-statusvlaggen op basis van de vergelijking

Opmerkingen:
In wezen is dit het equivalent van `cmp (X),y` in bank Z.

Deze routine heeft toegang tot elk adres in de eerste 1 MB van het geheugen. Anders dan JMPFAR wijzigt zij de geheugenindeling niet. Om het lange adres te benaderen, gebruikt zij een DMA-taak.

Op de MEGA65 ondersteunt de 45GS02 een instructie die dit kan met een 32-bits adres op de basispagina, zonder dat een KERNAL-routine nodig is: `cmp [zp4],z`

Voorbeeld: `cmp_far = $ff7a`

```
; Vergelijk het geheugen op adres 4.4502
; naar de accumulator
lda #$00
sta $fe
lda #$45
sta $ff
ldx #$fe  ; basispagina-aanwijzer -> $4500
ldy #$02  ; Y-index
ldz #$04  ; bank
lda #$99  ; de te vergelijken gegevens
jsr cmp_far
beq they_are_equal  ; ...
```

CURSOR

Adres: JSR $FF35

Omschrijving:
Schakelt de knipperende cursor in of uit

Invoer: **C-vlag** 0 om in te schakelen, 1 om uit te schakelen

Opmerkingen:
In het normale gebruik schakelt de editor de knipperende cursor in bij de READY.-prompt of bij de uitvoering van het commando **INPUT** op de terminal en houdt hij de cursor anders uitgeschakeld bij de uitvoering van andere commando's, inclusief machinetaal die met **SYS** wordt aangeroepen. Een programma kan het knipperen van de cursor weer inschakelen wanneer het vergelijkbare gebruikersinteracties biedt, zoals het afwachten van toetsenbordinvoer van de gebruiker met **GETIN**.

Het inschakelen van de cursor heeft slechts tot gevolg dat de schermeditor de cursorpositie aangeeft, door het teken op die positie te animeren. Het is de verantwoordelijkheid van het programma om naar toetsenbordinvoer te luisteren en tekens naar het scherm te sturen.

In BASIC hebben de commando's **CURSOR OFF** en **CURSOR ON** hetzelfde effect als deze routine.

Voorbeeld:

```
bsout = $ffd2
cursor = $ff35
getin = $ffe4

    ; Schakel de cursor in.
    clc
    jsr cursor

    ; Wacht op een toetsaanslag.
-   jsr getin
    beq -
    pha  ; Bewaar het getypte teken.

    ; Schakel de cursor uit.
    sec
    jsr cursor

    ; Geef het getypte teken door.
    pla
    jsr bsout
```

GETIN

Adres: JSR $FFE4

Omschrijving:
Leest een teken van het invoerapparaat, zonder te wachten

Uitvoer: **A** Het gelezen teken
C-vlag Gezet bij een fout

Opmerkingen:
Het invoerapparaat is standaard het toetsenbord. Een programma kan het invoerapparaat wijzigen met OPEN en CHKIN.

GETIN behandelt het invoerapparaat als gebufferd, maar wacht niet op invoer. Indien de invoerbuffer van het apparaat leeg is, retourneert GETIN $00.

Is het invoerapparaat bij de MEGA65 het toetsenbord, dan gebruikt GETIN meerdere toetsenbordinvoerbuffers: de KERNAL-toetsenbordbuffer, de functietoetsmacrobuffer en de hardwaretoetsenbordbuffer van de MEGA65. In tegenstelling tot de C65 vult de MEGA65-KERNAL de KERNAL-toetsenbordbuffer niet in de IRQ-handler. In plaats daarvan haalt GETIN rechtstreeks één typegebeurtenis uit de hardwaretoetsenbordbuffer van de MEGA65. De KERNAL-toetsenbordbuffer blijft evenwel ondersteund, zodat softwarefuncties die toetsaanslagen injecteren blijven werken.

Is het invoerapparaat serieel of het scherm, dan gedraagt GETIN zich zoals BASIN.

Een programma kan deze routine met de IGETIN-vector overschrijven of uitbreiden.

Voorbeeld:

```
getin = $ffe4

get_a_key:
    jsr getin
    beq get_a_key
    sta data
```

GETIO

Adres: JSR $FF41

Omschrijving:
Leest de huidige invoer- en uitvoerapparaten

Uitvoer: **X** huidig invoerapparaat (FA)
Y huidig uitvoerapparaat (FA)

Opmerkingen:
Een invoerapparaat van 0 vertegenwoordigt het toetsenbord.

Een uitvoerapparaat van 3 vertegenwoordigt het scherm.

CHKIN en CKOUT wijzigen deze instellingen.

Voorbeeld:
```
getio = $ff41

      jsr getio
      cpy #3
      beq is_screen_output
```

GETLFS

Adres: JSR $FF44

Omschrijving:
Leest bestand, apparaat en secundair adres

Uitvoer: **A** Het logische bestandsnummer (LA)
X Het apparaatnummer (FA)
Y Het secundaire adres, of $FF (SA)

Opmerkingen:
Deze instellingen worden zowel door SETLFS gewijzigd als door andere KERNAL-aanroepen en BASIC-commando's die via het logische bestandsnummer informatie over geopende bestanden opzoeken.

Een programma kan dit gebruiken om te bepalen van welk schijfstation het programma is geladen, door deze routine direct na het starten aan te roepen, voordat het andere schijf-I/O uitvoert.

Voorbeeld:
```
getlfs = $ff44

       jsr getlfs
```

IOBASE

Adres: JSR $FFF3

Omschrijving:
GERESERVEERD, NIET GEBRUIKEN

Opmerkingen:
Vroeger gaf deze routine het startadres van de I/O-registers terug. Bij de C65 is dit altijd $D000.

IOINIT

Adres: JSR $FF84

Omschrijving:
Initialiseert I/O-apparaten

Opmerkingen:
IOINIT reset de CIA-chips, de 45GS02-poort, de VIC-chip, de UART en het DOS.

Deze routine moet worden aangeroepen met uitgeschakelde interrupts.

Voorbeeld:
```
ioinit = $ff84

        sei
        jsr ioinit
        cli
```

JMPFAR

Adres: JMP $FF71

Omschrijving:
Springt naar een adres in een willekeurige bank

Invoer:
$02 Bank van het adres (0-5)
$03 Hoge byte van het adres
$04 Lage byte van het adres
$05 Statusregister van de CPU
$06 A-register van de CPU
$07 X-register van de CPU
$08 Y-register van de CPU
$09 Z-register van de CPU

Opmerkingen:
Deze routine stelt de geheugenindeling van de CPU zodanig bij dat het gevraagde adres als een 16-bits adres zichtbaar wordt, en springt er vervolgens naartoe. Hij keert niet terug.

Deze routine kent vergelijkbare beperkingen als het **SYS**-commando van BASIC en kan niet elk adres bereiken. Hij kan uitsluitend springen naar adressen in banken ongelijk aan nul, van X.2000 - X.7FFF.

De gewijzigde geheugenindeling laat $0000-$1FFF en $8000-$DFFF ongemapt (zodat zij naar bank 0 respectievelijk de I/O-registers verwijzen) en mapt $E000-$FFFF op KERNAL-ROM.

Een programma stelt de parameters voor JMPFAR in door naar de basispagina van de KERNAL te schrijven. Tot de parameters behoren de drie adresbytes en de toestand van de CPU-registers zoals die bij binnenkomst op de nieuwe locatie verwacht wordt.

Deze routine keert niet terug. Roep deze routine uitsluitend aan met JMP.

Voorbeeld: `jmpfar = $ff71`

```
; MAP $2000-$7FFF naar bank 4, dan JMP
; naar 4.4500
lda #$04
sta $02
lda #$45
sta $03
lda #$00
sta $04
lda #$04  ; Start de routine met
          ; interrupts uitgeschakeld
sta $05
jmp jmpfar
```

JSRFAR

Adres: JSR $FF6E

Omschrijving:
Roept een subroutine in een willekeurige bank aan

Invoer: **$02** Bank van het adres (0-5)
$03 Hoge byte van het adres
$04 Lage byte van het adres
$05 Statusregister van de CPU
$06 A-register van de CPU
$07 X-register van de CPU
$08 Y-register van de CPU
$09 Z-register van de CPU

Uitvoer: De status- en de A-, X-, Y- en Z-registers van de CPU bevatten bij terugkeer dezelfde waarden als aan het einde van de subroutine.

Opmerkingen:
Deze routine stelt de geheugenindeling van de CPU zodanig bij dat het gevraagde adres als een 16-bits adres zichtbaar wordt, en voert er vervolgens een JSR naartoe uit. Bij terugkeer herstelt hij de vorige, door de KERNAL beheerde geheugenindeling (bijv. de SYS-indeling).

Deze routine kent vergelijkbare beperkingen als het **SYS**-commando van BASIC en kan niet elk adres bereiken. Hij kan uitsluitend springen naar adressen in banken ongelijk aan nul, binnen X.2000 - X.7FFF.

De gewijzigde geheugenindeling laat $0000-$1FFF en $8000-$DFFF ongemapt (waardoor zij naar bank 0 en de I/O-registers verwijzen), terwijl $E000-$FFFF op de KERNAL-ROM is gemapt.

Een programma stelt de parameters voor JSRFAR in door naar de basispagina van de KERNAL te schrijven. Tot de parameters behoren de drie adresbytes en de toestand van de CPU-registers zoals die bij binnenkomst op de nieuwe locatie verwacht wordt.

Roep deze routine uitsluitend aan met JSR.

Voorbeeld:
```
jsrfar = $ff6e

        ; MAP $2000-$7FFF naar bank 4, dan
        ; JSR naar 4.4500
        lda #$04
        sta $02
        lda #$45
        sta $03
        lda #$00
        sta $04
        lda #$04  ; Start de routine met
```

```
          ; interrupts uitgeschakeld
sta $05
jsr jsrfar

; Het programma gaat verder...
```

KEY

Adres: JSR $FF9F

Omschrijving:

Voert een toetsenbordscan uit.

Opmerkingen:

Bij de MEGA65 worden typegebeurtenissen beheerd door een buffermechanisme in de hardware en niet door de KERNAL. Een programma dat een typegebeurtenis wil verwerken, kan GETIN aanroepen of rechtstreeks de registers van de typegebeurtenisbuffer raadplegen, zonder KEY aan te roepen.

KEY beheert speciale toetsenbordgedragingen, waaronder Mega + Shift, No Scroll en Ctrl-S. De KERNAL van de MEGA65 roept KEY binnen de IRQ-handler meerdere keren per seconde aan, vergelijkbaar met andere Commodore-computers. Een programma dat de IRQ-handler van de KERNAL uitschakelt maar deze gedragingen wel in stand wil houden, kan KEY vanuit een eigen handler aanroepen.

OPMERKING: Bij de MEGA65 wijkt KEY af van de C65 en biedt de routine geen aansluiting via vectoren. Vectoren om toetsaanslagen te onderscheppen zijn nog in ontwikkeling.

KEY heeft hetzelfde adres als de routine "SCNKEY" op de Commodore 64, maar gedraagt zich anders.

Voorbeeld:

```
key = $ff9f

      jsr key
```

LDA_FAR

Adres: JSR $FF74

Omschrijving:
Leest een byte van een adres in een willekeurige bank

Invoer: **X** Basispagina-aanwijzer
Y Index vanaf het aangewezen adres
Z Bank

Uitvoer: **A** De waarde op het adres

Opmerkingen:
In wezen is dit het equivalent van `lda (X),y` in bank Z.

Deze routine kan elk adres in de eerste 1 MB van het geheugen bereiken. In tegenstelling tot JMPFAR wijzigt hij de geheugenindeling niet. Hij gebruikt een DMA-taak om het lange adres te benaderen.

Bij de MEGA65 ondersteunt de 45GS02 een instructie die dit kan met een 32-bits adres dat op de basispagina staat, zonder dat daarvoor een KERNAL-routine nodig is: `lda [zp4],z`

Voorbeeld: `lda_far = $ff74`

```
; Laad het geheugen op adres 4.4502
; in de accumulator
lda #$00
sta $fe
lda #$45
sta $ff
ldx #$fe  ; basispagina-aanwijzer -> $4500
ldy #$02  ; Y-index
ldz #$04  ; bank
jsr lda_far
```

LISTN

Adres: JSR $FFB1

Omschrijving:
Stuurt het "listen"-commando

Invoer: **A** Het apparaatnummer (FA, 0-31)

Opmerkingen:
Dit is een seriële routine op laag niveau. De meeste programma's gebruiken liever de I/O-routines op hoger niveau (OPEN en dergelijke).

Volgens het seriële protocol op laag niveau wordt van de computer verwacht dat hij alle apparaatcommunicatie op de seriële bus coördineert. Elk apparaat heeft een vastgeprogrammeerd apparaatnummer; sommige apparaten, zoals schijfstations, hebben schakelaars waarmee het herkende apparaatnummer te wijzigen is. Alle apparaten zien al het verkeer op de bus en van elk apparaat wordt verwacht dat het van modus wisselt wanneer het een commando met zijn apparaatnummer ziet.

Wanneer de computer gegevens naar een apparaat wil sturen, zendt hij het "listen"-commando naar dat apparaat. Doorgaans verzendt de computer gegevens naar slechts één apparaat tegelijk. Om een andere listener aan te wijzen, zendt de computer een "untalk"-commando uit naar alle apparaten en stuurt hij vervolgens een nieuw "listen"-commando met een ander apparaatnummer.

De computer geeft telkens aan één apparaat toestemming om als "talker" op de seriële bus op te treden. Om een andere talker aan te wijzen, zendt de computer een "untalk"-commando uit naar alle apparaten en stuurt hij vervolgens een nieuw "talk"-commando.

Gebruik UNLSN om alle apparaten te laten weten dat zij de talk-modus moeten beëindigen.

Een programma kan dit gedrag met de ILISTEN-vector overschrijven of uitbreiden.

LISTN is vergelijkbaar met "LISTEN" op de Commodore 64.

Voorbeeld:
```
listn = $ffb1

      lda #8
      jsr listn
```

LKUPLA

Adres: JSR $FF5F

Omschrijving:
Zoekt een logisch bestandsnummer dat in gebruik is

Invoer: **A** Het logische bestandsnummer (LA)

Uitvoer: Indien gevonden:
C-vlag = 0
A Het logische bestandsnummer (LA)
X Het apparaatnummer (FA)
Y Het secundaire adres (SA), of $FF indien niet van toepassing

Indien niet gevonden:
C-vlag = 1

Opmerkingen:
Deze routine wordt vooral door BASIC gebruikt om een ongebruikt logisch bestandsnummer te vinden. In de praktijk bepalen de meeste programma's volledig zelf welke logische bestandsnummers ze gebruiken, zodat opzoeken niet nodig is.

Deze routine is tevens bruikbaar om aan de hand van het logische bestandsnummer te achterhalen bij welk apparaat een bestand geopend is.

Voorbeeld: `lkupla = $ff5f`

```
        lda #$01
        jsr lkupla
        bcs not_in_use
```

LKUPSA

Adres: JSR $FF62

Omschrijving:
Zoekt een secundair adres dat in gebruik is

Invoer: **Y** Het secundaire adres (SA)

Uitvoer: Indien gevonden:
C-vlag = 0
A Het logische bestandsnummer (LA)
X Het apparaatnummer (FA)
Y Het secundaire adres (SA), of $FF indien niet van toepassing

Indien niet gevonden:
C-vlag = 1

Opmerkingen:
Deze routine wordt vooral door BASIC gebruikt om een ongebruikt secundair adres te vinden. In de praktijk bepalen de meeste programma's volledig zelf welke secundaire adressen ze gebruiken, zodat opzoeken niet nodig is.

Voorbeeld: `lkupsa = $ff62`

```
ldy #$60
jsr lkupsa
bcs not_in_use
```

LOAD

Adres: JSR $FFD5

Omschrijving:
Laadt/verifieert uit bestand

Invoer: **A** = 0 om te laden, 1 om te verifiëren; zet bit 6 voor de "raw"-modus
X Laadadres, lage byte
Y Laadadres, hoge byte

Uitvoer: **C-vlag** Gezet bij een fout
A Foutcode, indien aanwezig
X Eindadres, lage byte
Y Eindadres, hoge byte

Opmerkingen:

Voordat het programma LOAD aanroept, moet het SETBNK, SETLFS en SETNAM aanroepen om de parameters voor het te laden bestand in te stellen. LOAD kan niet worden gebruikt met RS-232-apparaten, het scherm of het toetsenbord. Het logische bestandsnummer wordt niet gebruikt.

Bij gebruik met een schijfstation laadt LOAD uitsluitend een bestand van het type PRG. Om een bestand van een ander type te laden, gebruikt u SETNAM, waarin u het type in de naamstring opneemt (bijvoorbeeld `"MIJNBESTAND,S"` voor een SEQ-bestand), samen met OPEN, CHKIN, BASIN en CLOSE. Zie OPEN voor een voorbeeld.

Indien bit 0 van A gewist is, laadt LOAD het bestand van het invoerapparaat en schrijft het naar het geheugen.

Indien bit 0 van A ingesteld is, verifieert LOAD het bestand: het laadt het bestand en vergelijkt het met de inhoud van het geheugen. Komt het bestand niet overeen met het geheugen, dan retourneert LOAD een foutstatus. Het geheugen wordt niet gewijzigd.

Indien bit 6 van A ingesteld is (de "raw"-modus), behandelt LOAD de eerste twee bytes van het bestand als gegevens. Is deze bit gewist, dan wordt verwacht dat de eerste twee bytes van het bestand het opgeslagen laadadres bevatten, zoals gebruikelijk is bij een PRG-bestand. Een PRG-bestand dat in de raw-modus wordt geladen, moet desondanks minstens twee bytes bevatten, anders wordt een "File Not Found"-fout gemeld.

Is het secundaire adres (SA) $00, dan wordt de opgegeven waarde van X/Y als laadadres gebruikt en worden de eerste twee bytes van het bestand overgeslagen indien de "raw"-modus niet actief is. Is SA niet $00, dan worden X/Y genegeerd en wordt het feitelijke laadadres ontleend aan het opgeslagen laadadres in de eerste twee bytes van het bestand.

Dit komt overeen met het gedrag van het volgende BASIC-commando: `LOAD "FILE",8,1`

De volledige LOAD-bewerking moet binnen één bank passen. Deze mag geen bankgrenzen overschrijden.

LOAD verricht alle taken die nodig zijn om het bestand te openen, het te lezen en het vervolgens te sluiten. Anders dan OPEN laat het geen logische bestanden open staan.

Bij succes wordt het adres van de laatst geladen (of geverifieerde) byte *plus één* teruggegeven in de registers X en Y.

LOAD stelt de I/O-status op de juiste wijze in. Deze is met READSS uit te lezen.

Zie OPEN voor de lijst met mogelijke foutcodes.

Een programma kan dit gedrag met de ILOAD-vector overschrijven of uitbreiden.

Voorbeeld:

```
load = $ffd5
setbnk = $ff6b
setlfs = $ffba
setnam = $ffbd

    lda #$00    ; geheugen in bank 0
    ldx #$00    ; bestandsnaam in bank 0
    jsr setbnk

    lda #$00    ; (niet gebruikt bij laden)
    ldx #$08    ; apparaatnummer
    ldy #$00    ; laden
    jsr setlfs

    lda #bestandsnam_eind-bestandsnaam
    ldx #<bestandsnaam
    ldy #>bestandsnaam
    jsr setnam

    lda #0      ; laden
    ldx #$00    ; naar 0.4500
    ldy #$45
    jsr load

    rts

bestandsnaam:
    !pet "mijnbestand"
bestandsnaam_eind:
```

MEMBOT

Adres: JSR $FF9C

Omschrijving:
GERESERVEERD, NIET GEBRUIKEN

Opmerkingen:
Historisch gezien verwijzen MEMBOT en MEMTOP naar parameters voor de geheugentoewijzing die de KERNAL gebruikt. De C65-KERNAL gebruikt deze voorziening niet. Deze routines worden wel genoemd in de C65-documentatie, maar doen niets.

MEMTOP

Adres: JSR $FF99

Omschrijving:
GERESERVEERD, NIET GEBRUIKEN

Opmerkingen:
Historisch gezien verwijzen MEMBOT en MEMTOP naar parameters voor de geheugentoewijzing die de KERNAL gebruikt. De C65-KERNAL gebruikt deze voorziening niet. Deze routines worden wel genoemd in de C65-documentatie, maar doen niets.

MONEXIT

Adres: JMP $FFA2

Omschrijving:
Verlaat de monitor en keert terug naar BASIC

Opmerkingen:
Deze routine wordt uitsluitend door de monitor gebruikt en is voor programma's niet nuttig.

MONEXIT heeft hetzelfde adres als de KERNAL-routine "SETTMO" op de Commodore 64. SETTMO is niet beschikbaar op de MEGA65.

Voorbeeld:
```
monexit = $ffa2

        jmp monexit
```

MonitorCall

Adres: JMP $FF56

Omschrijving:
Roept de monitor aan

Opmerkingen:
Deze routine stopt het programma en roept vervolgens de monitor aan.

Deze routine keert niet terug. Roep deze routine uitsluitend aan met JMP.

Wanneer de monitor wordt afgesloten, keert deze terug naar de READY.-prompt.

Voorbeeld:
```
monitor_call = $ff56

        jmp monitor_call
```

OPEN

Adres: JSR $FFC0

Omschrijving:
Opent een logisch bestand

Uitvoer: **C-vlag** Gezet bij een fout
A Foutcode, indien aanwezig

Opmerkingen:

Voordat het programma OPEN aanroept, moet het SETBNK, SETLFS en SETNAM aanroepen om de parameters voor het te openen bestand in te stellen.

Het gevraagde logische bestandsnummer (LA) mag momenteel nog niet openstaan. Gebruik, indien nodig, LKUPLA om na te gaan welke logische bestandsnummers ongebruikt zijn. Er kunnen maximaal tien logische bestanden tegelijk openstaan.

OPEN voert apparaatspecifieke taken uit voor het openen van seriële, RS-232-, toetsenbord- en schermapparaten, waaronder het wissen van de vorige status en het verzenden van een eventuele bestandsnaam of commandostring die aan SETNAM en SETBNK is meegegeven.

OPEN stelt de I/O-status op de juiste wijze in. Deze is met READSS uit te lezen.

Met CHKIN kan een programma een open bestand tot het huidige invoerapparaat maken en met CKOUT tot het huidige uitvoerapparaat. BASIN en GETIN lezen van het invoerapparaat en BSOUT schrijft naar het uitvoerapparaat.

Een programma moet een open bestand sluiten zodra het er klaar mee is. Gebruik CLOSE om een specifiek bestand te sluiten. Gebruik CLOSE_ALL om alle bestanden van een apparaat te sluiten. Gebruik CLALL om alle openstaande bestanden te sluiten. Zet bij het sluiten van het commandokanaal van een schijf de C-vlag voordat het programma CLOSE aanroept; zie CLOSE voor meer informatie.

De mogelijke foutcodes die BASIN, BSOUT, CHKIN, CKOUT, CLOSE, LOAD, OPEN, SAVE en SAVEFL kunnen teruggeven, zijn als volgt:

Foutcode	Betekenis
1	Te veel bestanden
2	Bestand open
3	Bestand niet open
4	Bestand niet gevonden
5	Apparaat niet aanwezig
6	Geen invoerbestand
7	Geen uitvoerbestand
8	Bestandsnaam ontbreekt
9	Ongeldig apparaatnummer

OPMERKING: Niet alle routines kunnen alle fouten detecteren. In het bijzonder detecteert OPEN geen "File not found"-fout. Deze fouttoestand blijft onopgemerkt totdat het programma probeert het bestand te lezen. READSS meldt sommige fouttoestanden pas na een bewerking. Schijfstations kunnen via het commandokanaal andere fouttoestanden melden.

Een programma kan het gedrag van OPEN met de IOPEN-vector overschrijven of uitbreiden.

Voorbeelden:

Schijfstations accepteren argumenten in de bestandsnaamstring die met SETNAM is ingesteld; de argumenten volgen op de bestandsnaam en worden door komma's gescheiden. Het volgende voorbeeld opent een bestand van het type SEQ om te schrijven ("...,s,w") en schrijft vervolgens de bytes 0 tot en met 255. (Ter wille van de beknoptheid is foutcontrole weggelaten.)

```
open = $ffc0
close = $ffc3
ckout = $ffc9
bsout = $ffd2
setbnk = $ff6b
setlfs = $ffba
setnam = $ffbd

    lda #$00     ; geheugen in bank 0
    ldx #$00     ; bestandsnaam in bank 0
    jsr setbnk

    lda #$01     ; logisch bestandsnummer
    ldx #$08     ; apparaatnummer
    ldy #$02     ; secundair adres
                 ; (niet 0, 1 of 15)
    jsr setlfs

    lda #bestandsnaam_eind-bestandsnaam
    ldx #<bestandsnaam
```

```
        ldy #>bestandsnaam
        jsr setnam

        jsr open
        ldx #$01
        jsr ckout

        lda #0
-       jsr bsout
        inc
        bne -

        clc
        lda #$01
        jsr close

        rts

bestandsnaam:
        !pet "mijnseqbestand,s,w"
bestandsnaam_eind:
```

Schijfstations hebben een *commandokanaal* dat toegankelijk is via het secundaire adres 15. Wanneer een logisch bestand naar SA 15 van het station wordt geopend, is de bestandsnaam de tekst van de opdracht die moet worden uitgevoerd. De opdracht wordt onmiddellijk uitgevoerd. Het volgende voorbeeld is vergelijkbaar met het BASIC-commando `OPEN 1,8,15,"I0"`, dat de initialisatieopdracht naar het schijfstation met apparaatnummer 8 stuurt.

```
open = $ffc0
close = $ffc3
setbnk = $ff6b
setlfs = $ffba
setnam = $ffbd

        lda #cmd_initialiseer_eind-cmd_initialiseer
        ldx #<cmd_initialiseer
        ldy #>cmd_initialiseer
        jsr setnam

        ldx #0
        jsr setbnk

        lda #1
        ldx #8
        ldy #15
        jsr setlfs
```

```
    jsr open    ; voer het I0-commando uit
    bcs error

    lda #1
    sec         ; "speciale sluiting" (zie CLOSE)
    jsr close

    ; ...

cmd_initialiseer:
    !pet "i0"
cmd_initialiseer_eind:
```

PFKEY

Adres: JSR $FF68

Omschrijving:
Programmeert een functietoets van de editor

Invoer: **A** Basispagina-aanwijzer naar het stringadres (lo/hi/bank)
X Nummer van de functietoets (1-16)
Y Lengte van de string

Uitvoer: **C-vlag** Gezet als het bijwerken is mislukt

Opmerkingen:
De schermeditor ondersteunt door de gebruiker gedefinieerde macro's die aan functietoetsen zijn gekoppeld. De PFKEY-aanroep werkt de definitie van een macro bij, vergelijkbaar met het **KEY**-commando van BASIC.

Als invoer stelt het programma de accumulator in op een basispagina-adres, waarop het 24-bits adres van de macrostring in little-endian-volgorde staat. Het Y-register bevat de lengte van de string, tot ten hoogste 240 tekens.

Het X-register geeft de te definiëren toets aan: 1 - 14 voor de bijbehorende genummerde functietoets, 15 voor de Help-toets en 16 voor de Stop-toets (Shift + Run/Stop).

De string bevat de PETSCII-codes die de macro bij uitvoering intypt. Dit is anders dan bij de BASIC-stringexpressie die u bij het **KEY**-commando zou gebruiken: speciale tekens, zoals dubbele aanhalingstekens, moeten in de string die bij PFKEY wordt gebruikt als PETSCII-codes verschijnen. (Zie het voorbeeld hieronder.)

Bij succes kopieert PFKEY de string naar het macrogeheugen; de string op het opgegeven adres hoeft daarna niet te blijven bestaan.

Voorbeeld: `pfkey = $ff68`

```
; stringadres op basispagina $fd-$ff
lda #<macro
sta $fd
lda #>macro
sta $fe
lda #0
sta $ff

; Installeer de macrostring voor de F6-toets
lda #$fd
ldy #macro_eind-macro
ldx #6
```

```
    jsr pfkey

    rts

macro:
    !pet "print ",34,"het werkt!",34,13
macro_eind:
```

PHOENIX

Adres: JMP $FF5C

Omschrijving:
Roept de schijf-bootloader aan

Opmerkingen:
Zie BOOT_SYS voor een beschrijving van bootdisks.

Enkele vroege prototype-ROM's van de Commodore 65 vervingen deze routine door een routine die diagnostiek uitvoerde en de resultaten afdrukte. Latere versies vervingen dit door een aanroep van de bootreeks.

Deze routine keert niet terug. Gebruik uitsluitend JMP om deze routine aan te roepen.

OPMERKING: Voor de MEGA65 is deze functie niet getest.

Voorbeeld:
```
phoenix = $ff5c

        jmp phoenix
```

PLOT

Adres: JSR $FFF0

Omschrijving:
Leest/stelt de cursorpositie in

Invoer: Om in te stellen:
C-vlag = 0
X Cursorregel
Y Cursorkolom

Om te lezen:
C-vlag = 1

Uitvoer: Bij het instellen:
C-vlag Gezet bij een fout

Bij het lezen:
X Cursorregel
Y Cursorkolom

Opmerkingen:
De CPU-registers **X** en **Y** komen *niet* overeen met de coördinaatassen die daar traditioneel naar vernoemd worden. Het register **X** staat voor het regelnummer (op en neer langs de verticale as) en het register **Y** voor het kolomnummer (links en rechts langs de horizontale as). Dit komt overeen met het oorspronkelijke ontwerp van Commodore.

De cursorpositie wordt uitgedrukt ten opzichte van het actieve venster. Gebruik SCRORG om de huidige venstergrootte op te vragen.

Indien bij het instellen de gevraagde positie buiten het actieve venster valt, keert de routine terug met de C-vlag gezet.

Over de toestand van X en Y na het instellen van de cursorpositie worden geen garanties gegeven.

Voorbeeld:

```
plot = $fff0
primm = $ff7d

    ; Zet de cursor in kolom 78
    ; van de derde regel
    clc
    ldx #2
    ldy #77
    jsr plot

    jsr primm
    !pet "bericht",0
```

```
; Lees de nieuwe cursorpositie
sec
jsr plot
stx $1600
sty $1601
```

PRIMM

Adres: JSR $FF7D

Omschrijving:

Drukt een inline, met nul afgesloten korte string af.

Opmerkingen:

De af te drukken string volgt in het geheugen onmiddellijk op de JSR-instructie.

De string moet eindigen op een null-byte ($00) en mag inclusief de null-byte ten hoogste 255 bytes lang zijn.

Tekens worden naar de uitvoerstroom geschreven, zoals bij BSOUT.

Gebruik uitsluitend JSR om PRIMM aan te roepen.

PRIMM behoudt de registers. Dit maakt deze routine geschikt om tijdelijk midden in een programma in te voegen voor foutopsporing.

Voorbeeld:

```
primm = $ff7d

      jsr primm
      !pet "een petscii-string om af te drukken.",0

      ; Het programma gaat verder...
```

RAMTAS

Adres: JSR $FF87

Omschrijving:

Initialiseert het RAM-geheugen en de buffers

Opmerkingen:

RAMTAS initialiseert de variabelen van de KERNAL op de basispagina opnieuw en stelt de aanwijzers naar de boven- en onderkant van het systeem-RAM opnieuw in.

Deze routine is in de eerste plaats bedoeld voor gebruik door de opstart- en resetsequenties van het systeem. Een machinetaalprogramma dat vanuit een opstartprogramma in BASIC wordt uitgevoerd, kan na een aanroep van RAMTAS niet netjes naar BASIC terugkeren en RAMTAS kan evenmin vanuit de monitor worden aangeroepen.

Voorbeeld:

```
ramtas = $ff87

       jsr ramtas
```

RDTIM

Adres: JSR $FFDE

Omschrijving:
Leest de 24-uursklok van CIA 1

Uitvoer: **Y** Uren, 0-23, in BCD
X Minuten, 0-59, in BCD
A Seconden, 0-59, in BCD
Z Tienden van een seconde, 0-9

Opmerkingen:
Deze routine leest de time-of-day-klok (TOD) van het CIA 1-apparaat uit. Deze klok telt tienden van seconden en houdt uren, minuten, seconden en tienden van seconden bij.

Gebruik SETTIM om de waarde van de TOD-klok in te stellen.

Bij de MEGA65 heeft de TOD-klok van CIA 1 geen enkele relatie met de batterijgevoede realtimeklok (RTC). De waarde die RDTIM teruggeeft, is de TOD-waarde van CIA 1 en niet de RTC-waarde. De speciale variabelen TI$ en DT$ van BASIC 65 gebruiken de RTC en niet de TOD.

De waarden hebben het Binary Coded Decimal-formaat, waarbij elke nibble één decimaal cijfer vertegenwoordigt. Als RDTIM bijvoorbeeld de hexadecimale waarde $59 in het accumulatorregister teruggeeft, vertegenwoordigt dit een decimale waarde van 59 seconden.

Voorbeeld:

```
rdtim = $ffde
bsout = $ffd2

    ; Druk uren en minuten af, als HH:MM
    jsr rdtim
    tya
    jsr print_bcd
    lda #':'
    jsr bsout
    txa
    jsr print_bcd
    rts

print_bcd:
    pha
    lsr
    lsr
    lsr
    lsr
    adc #'0'
    jsr bsout
```

```
pla
and #$0f
adc #'0'
jsr bsout
rts
```

READSS

Adres: JSR $FFB7

Omschrijving:
Geeft de status van de laatste I/O-bewerking terug

Uitvoer: **A** De statusbyte

Opmerkingen:
Deze routine geeft de status terug van de laatste I/O-bewerking die voor een serieel of RS-232-apparaat is uitgevoerd. Dit werkt alleen bij I/O-bewerkingen op hoog niveau, zoals OPEN, LOAD en SAVE, evenals bij het lezen van en schrijven naar kanalen die met OPEN zijn geopend.

Een gebruikelijke toepassing van READSS is het bepalen van het einde van een bestand of stroom.

Bij een RS-232-apparaat wist READSS de statusbyte en geeft de vorige waarde terug.

Bij seriële apparaten worden de statusbits als volgt gezet:

Bit	Betekenis
0	Time-out bij schrijven
1	Time-out bij lezen
6	Einde van de invoer
7	Apparaat niet aanwezig

READSS is vergelijkbaar met “READST” op de Commodore 64.

Voorbeeld:
```
readss = $ffb7

    ldx #0
loop:
    jsr basin
    sta buffer,x
    inx

    jsr readss
    beq loop    ; lezen geslaagd, ga door
    and #$40
    bne eof     ; status is einde van bestand

    ; verwerk andere fout...

eof:
    ; ...
```

RESET_RUN

Adres: JSR $FF32

Omschrijving:
Herstart de computer, met de optie om een programma uit te voeren

Invoer: **A** De resetmodus

Opmerkingen:
Kies een resetmodus door het accumulatorregister **A** in te stellen op de juiste waarde:

- **A=0** KERNAL-warmstart.
- **A=1** Hypervisor-reset.
- **A=2** KERNAL-warmstart, waarna **RUN** het BASIC-programma in het geheugen uitvoert.

Bij een KERNAL-warmstart start de computer opnieuw bij het opstartproces van de MEGA65-KERNAL, inclusief het resetten van de KERNAL-status en het weergeven van de BASIC-banner. Resetmodus 0 probeert bovendien de gekoppelde schijf automatisch op te starten.

Een Hypervisor-reset zet het opstartproces van de Hypervisor in gang, vergelijkbaar met het indrukken van de resetknop aan de zijkant van de computer. Hierbij voert de computer enkele extra opstartstappen uit, zoals het weergeven van het MEGA65-opstartscherm en het opnieuw laden van de ROM vanaf de SD-kaart. Niet al het geheugen wordt gereset, maar wel mogelijk meer registers dan bij een KERNAL-warmstart.

Resetmodus 2 wijzigt het opstartproces van de KERNAL zodat **RUN** het BASIC-programma uitvoert dat in het geheugen staat (te beginnen op adres $2001). Dit is nuttig voor programma's die op hun beurt andere, willekeurige programma's starten, zoals kioskmenu's of bestandskiezers. In deze modi voert de KERNAL het commando **NEW** niet uit en start hij de gekoppelde schijf ook niet automatisch op.

Resetmodi 0 tot en met 2 negeren de waarden in de CPU-registers **X**, **Y** en **Z**. Aanvullende modi die parameters in deze registers aanvaarden, kunnen in de toekomst worden toegevoegd. Alle waarden van **A** groter dan 2 zijn gereserveerd voor toekomstige uitbreiding.

Deze routine keert niet terug.

Voorbeeld: `reset_run = $ff32`

```
lda #1          ; Hypervisor-reset
jmp reset_run
```

RESTOR

Adres: JSR $FF8A

Omschrijving:
Initialiseert de KERNAL-vectortabel

Opmerkingen:
Deze routine reset alle KERNAL-gerelateerde vectoren naar hun standaardwaarden, welke naar routines in de ROM wijzen.

Hij reset de vectoren die de schermeditor en BASIC gebruiken niet.

Deze routine moet worden aangeroepen met uitgeschakelde interrupts.

Zie VECTOR.

Voorbeeld:
```
restor = $ff8a

        sei
        jsr restor
        cli
```

SAVE

Adres: JSR $FFD8

Omschrijving:
Slaat gegevens op in een bestand

Invoer: **A** Basispagina-aanwijzer naar het startadres
X Eindadres + 1, lage byte
Y Eindadres + 1, hoge byte

Uitvoer: **C-vlag** Gezet bij een fout
A Foutcode, indien aanwezig

Opmerkingen:
Voordat het programma SAVE aanroept, moet het SETBNK, SETLFS en SETNAM aanroepen om de parameters voor het op te slaan bestand in te stellen. SAVE kan niet worden gebruikt met RS-232-apparaten, het scherm of het toetsenbord. Het logische bestandsnummer wordt niet gebruikt.

Bij gebruik met een schijfstation slaat SAVE standaard een bestand van het type PRG op. Om een bestand van een ander type op te slaan, gebruikt u SETNAM met het in de naamstring vermelde type (bijv. `"MIJNBESTAND,S"` voor een SEQ-bestand), samen met OPEN, CKOUT, BSOUT en CLOSE. Zie OPEN voor een voorbeeld.

SAVE schrijft altijd het startadres als eerste twee bytes weg, ongeacht het bestandstype. Indien de gegevens in een andere bank dan bank 0 staan, wordt die informatie niet vastgelegd. Om de adresbytes in het bestand weg te laten, gebruikt u SAVEFL met de juiste vlag.

Standaard is het opslaan van een bestand met de naam van een bestaand bestand een fout en het bestaande bestand wordt niet gewijzigd. Om het bestand te overschrijven indien dit bestaat, zet u het volgende vóór de bestandsnaamstring: `@:` Bijvoorbeeld: `"@:MIJNBESTAND"`

SAVE schrijft een geheugengebied weg naar een bestand op een serieel apparaat. Het startadres van het geheugengebied moet op de basispagina worden vastgelegd als een 16-bits adres in little-endian-volgorde. Het programma geeft het basispaginaadres van deze aanwijzer door in het A-register.

Het programma geeft het eindadres van het geheugengebied als een 16-bits waarde *plus één* door in de X- en Y-registers.

Het volledige geheugengebied moet binnen één bank passen. Het mag geen bankgrenzen overschrijden.

SAVE voert alle taken uit die nodig zijn om het bestand te openen, weg te schrijven en vervolgens te sluiten. Hij laat geen logische bestanden openstaan (zoals OPEN dat doet).

SAVE stelt de I/O-status op de juiste wijze in. Deze is met READSS uit te lezen.

Zie OPEN voor de lijst met mogelijke foutcodes.

Een programma kan dit gedrag met de ISAVE-vector overschrijven of uitbreiden.

Voorbeeld:

```
save = $ffd8
setbnk = $ff6b
setlfs = $ffba
setnam = $ffbd

    ; vul het geheugen met testgegevens
    ldx #0
-   txa
    sta $4500,x
    inx
    bne -

    lda #$00    ; geheugen in bank 0
    ldx #$00    ; bestandsnaam in bank 0
    jsr setbnk

    lda #$00    ; (niet gebruikt bij opslaan)
    ldx #$08    ; apparaatnummer
    ldy #$01    ; opslaan
    jsr setlfs

    lda #bestandsnaam_eind-bestandsnaam
    ldx #<bestandsnaam
    ldy #>bestandsnaam
    jsr setnam

    ; Sla 0.4500 tot 0.45FF op
    lda #$00
    sta $fe
    lda #$45
    sta $ff
    lda #$fe
    ldx #$00
    ldy #$46

    jsr save
    bcs fout
```

```
    ; ...

bestandsnaam:
    !pet "mijnbestand"
bestandsnaam_eind:
```

SAVEFL

Adres: JSR $FF3B

Omschrijving:
Slaat gegevens op in een bestand, met vlaggen

Invoer: **A** Basispagina-aanwijzer naar het startadres
X Eindadres + 1, lage byte
Y Eindadres + 1, hoge byte
Z Vlaggen: bit 6 = "raw"-modus

Uitvoer: **C-vlag** Gezet bij een fout
A Foutcode, indien aanwezig

Opmerkingen:
SAVEFL is identiek aan SAVE, met het extra argument voor de vlaggen in het Z-register. Zie SAVE voor meer informatie.

Momenteel ondersteunt SAVEFL één vlag: indien bit 6 is gezet, slaat SAVEFL het bestand op zonder de adreskop van twee bytes. Dit is symmetrisch met de "raw"-modus van de routine LOAD.

ScanStopKey

Adres: JSR $FFEA

Omschrijving:
Scant de Stop-toets

Uitvoer: **A** $7F indien de Stop-toets is ingedrukt, $FF anders
N-vlag Gewist indien de Stop-toets is ingedrukt, anders gezet

Opmerkingen:
In tegenstelling tot STOP roept ScanStopKey de ISTOP-vector niet aan, sluit het geen kanalen en legt de toetsenbordbuffer niet leeg.

ScanStopKey heeft hetzelfde adres als de KERNAL-routine "UDTIM" op de Commodore 64. UDTIM is niet beschikbaar op de MEGA65.

Voorbeeld:
```
scan_stop_key = $ffea

    ; Wacht op de Stop-toets
-   jsr scan_stop_key
    bmi -
```

SCRORG

Adres: JSR $FFED

Omschrijving:
Geeft de huidige afmetingen van het schermvenster terug

Uitvoer: **C-vlag** Tekstschermbreedte: 0=80, 1=40
X Vensterbreedte
Y Vensterhoogte
A Schermgeheugenadres van de linkerbovenhoek van het venster, lage byte
Z Schermgeheugenadres van de linkerbovenhoek van het venster, hoge byte

Opmerkingen:
De schermeditor beheert een "venster", een rechthoek op het tekstscherm, waarin terminaluitvoer wordt geschreven en invoer wordt geaccepteerd. De grootte en de positie van het venster zijn standaard die van het volledige scherm en kunnen worden aangepast met het BASIC-commando **WINDOW**.

SCRORG geeft de eigenschappen van dit venster terug, welke kunnen worden gebruikt om tekens in het schermgeheugen te plaatsen.

SCRORG is vergelijkbaar met "SCREEN" op de Commodore 64.

Voorbeeld: `scrorg = $ffed`

```
; Zet een schoppen-teken in de linkerboven-
; hoek van het actieve venster.
jsr scrorg
sta $fe
stz $ff
lda #65
ldy #0
sta ($fe),y
```

SECND

Adres: JSR $FF93

Omschrijving:
Stuurt een secundair adres naar de listener

Invoer: **A** Secundair adres (SA)

Opmerkingen:
Dit is een routine op laag niveau voor de seriële bus. De meeste programma's geven de voorkeur aan de I/O-routines op hoger niveau (OPEN en dergelijke).

Nadat een programma het LISTN-commando naar een randapparaat op de seriële bus heeft gezonden, om het apparaat invoer te laten accepteren, stuurt SECND een secundair adres naar het apparaat. Dit wordt doorgaans gebruikt om het apparaat te configureren voordat het programma er gegevens naartoe verzendt. Het gedrag hangt af van het apparaat.

Om een secundair adres te sturen naar een apparaat waaraan TALK is gezonden, gebruikt u TKSA.

Een programma kan dit gedrag met de ISECOND-vector overschrijven of uitbreiden.

SECND is vergelijkbaar met "SECOND" op de Commodore 64.

Voorbeeld:
```
listn = $ffb1
secnd = $ff93

    lda #8
    jsr listn
    lda #15
    jsr secnd
```

SETBNK

Adres: JSR $FF6B

Omschrijving:

Stelt megabyte en bank in voor het I/O-geheugenadres en het bestandsnaamadres

Invoer: Voor banken 0 - 5:
A De geheugenbank
X De bestandsnaambank

Voor 28-bits adressen:
A Bit 7 gezet, bits 0 - 3 = bits 24 - 27 van het geheugenadres
Y Bits 16 - 23 van het geheugenadres
X Bit 7 gezet, bits 0 - 3 = bits 24 - 27 van het bestandsnaamadres
Z Bits 16 - 23 van het bestandsnaamadres

Opmerkingen:

SETBNK moet voor elke geheugen-I/O-bewerking worden gebruikt, samen met SETLFS en SETNAM voor het openen van bestanden. De lage twee bytes van elk adres dienen als argument voor andere aanroepen.

Voor achterwaartse compatibiliteit met de C65-KERNAL kent SETBNK twee aparte modi: één voor banken 0 - 5 in de eerste megabyte van het geheugen en één voor 28-bits adressen.

Is bit 7 van het eerste register gewist, dan stelt het een adres in de eerste megabyte voor en is de waarde het banknummer. Indien bijvoorbeeld A = $04, dan bevindt het geheugenadres zich in bank 4 van de eerste megabyte ($004.xxxx).

Is bit 7 van het eerste register gezet, dan stelt het een 28-bits adres voor en is de lage nibble van het eerste register de hoogste nibble van het adres. Het tweede register stelt de daaropvolgende twee nibbles van het adres voor. Indien bijvoorbeeld A = $88 en Y = $03, dan is het geheugenadres een 28-bits adres van de vorm $803.xxxx.

Het geheugenadres wordt met de invoerregisters A en Y ingesteld en het bestandsnaamadres met de invoerregisters X en Z. De geheugenbank en de bestandsnaambank kunnen verschillen en onafhankelijk van elkaar de bankmodus of de 28-bitsmodus gebruiken.

Zie OPEN voor een volledig voorbeeld.

Voorbeeld: `setbnk = $ff6b`

```
; Geheugenadres:      $4.xxxx
; Bestandsnaamadres:   $0.xxxx
lda #$04
ldx #$00
```

```
jsr setbnk

; Geheugenadres:   $803.xxxx
; Bestandsnaamadres:   $0.xxxx
lda #$88
ldy #$03
ldx #$00
jsr setbnk

; Geheugenadres:   $803.xxxx
; Bestandsnaamadres: $805.xxxx
lda #$88
ldy #$03
ldx #$88
ldz #$05
jsr setbnk
```

SETLFS

Adres: JSR $FFBA

Omschrijving:
Stelt bestand, apparaat en secundair adres in

Invoer: **A** Het logische bestandsnummer (LA)
X Het apparaatnummer (FA)
Y Het secundaire adres, of $FF (SA)

Opmerkingen:
SETLFS moet worden gebruikt voor bestandsbewerkingen op hoog niveau (OPEN, LOAD, SAVE), samen met SETBNK en SETNAM.

Het logische bestandsnummer moet uniek zijn onder de geopende bestanden. Indien het met OPEN wordt gebruikt, mag het nog niet geopend zijn.

Indien het secundaire adres niet van toepassing is, stelt u Y in op $FF. Voor schijfstations zijn de secundaire adressen 0 en 1 gereserveerd voor LOAD en SAVE en 15 is het commandokanaal.

Zie OPEN voor een volledig voorbeeld.

Voorbeeld: `setlfs = $ffba`

```
lda #$01
ldx #$08
ldy #$02
jsr setlfs
```

SETMSG

Adres: JSR $FF90

Omschrijving:
Schakelt KERNAL-meldingen in of uit

Invoer: **A** bit 7 = controlemeldingen, bit 6 = foutmeldingen

Opmerkingen:
De KERNAL kan meldingen op het scherm weergeven wanneer het schijf-bewerkingen uitvoert, zoals “LOADING”, “SAVING”, “VERIFYING” en “I/O ERROR”. Deze meldingen staan standaard uit.

Stel bit 7 in om de controlemeldingen van de KERNAL in te schakelen. Stel bit 6 in om de I/O-foutmeldingen van de KERNAL in te schakelen.

De meldingen van de KERNAL zijn niet hetzelfde als de meldingen die BASIC weergeeft.

Voorbeeld: `setmsg = $ff90`

```
; Schakel alle KERNAL-meldingen in
lda #$c0
jsr setmsg
```

SETNAM

Adres: JSR $FFBD

Omschrijving:
Stelt de verwijzingen naar de bestandsnaam in

Invoer: **A** De lengte van de string, of $00
X Het lage byte van het stringadres
Y Het hoge byte van het stringadres

Opmerkingen:
SETNAM moet worden gebruikt voor bestandsbewerkingen op hoog niveau (OPEN, LOAD, SAVE), samen met SETBNK en SETLFS.

Indien een bestandsnaam niet geschikt is voor het I/O-apparaat, moet SETNAM worden aangeroepen met A = $00 (en willekeurige waarden voor X en Y).

SETBNK stelt de bank voor het adres in.

Zie OPEN voor een volledig voorbeeld.

Voorbeeld:

```
setnam = $ffbd

        lda #bestandsnaam_eind-bestandsnaam
        ldx #<bestandsnaam
        ldy #>bestandsnaam
        jsr setnam

        ; ...

bestandsnaam:
        !pet "mijnbestand"
bestandsnaam_eind:
```

SETTIM

Adres: JSR $FFDB

Omschrijving:
Stelt de 24-uursklok van CIA1 in

Invoer: **Y** Uren, 0-23, in BCD
X Minuten, 0-59, in BCD
A Seconden, 0-59, in BCD
Z Tienden van een seconde, 0-9

Opmerkingen:
Deze routine stelt de time-of-day-klok (TOD) van het CIA1-apparaat in. Deze klok telt tienden van seconden en houdt uren, minuten, seconden en tienden van seconden bij.

Gebruik RDTIM om de huidige waarde van de TOD-klok te lezen.

Bij de MEGA65 heeft de TOD-klok van CIA1 geen betrekking op de batterijgevoede realtimeklok (RTC). Het instellen van de TOD-klok heeft geen invloed op de RTC. De speciale variabelen TI$ en DT$ van BASIC 65 gebruiken de RTC en niet de TOD.

De waarden hebben het Binary Coded Decimal-formaat, waarbij elke nibble één decimaal cijfer vertegenwoordigt. Om bijvoorbeeld het aantal seconden op 59 in te stellen, gebruikt u de hexadecimale waarde $59.

Voorbeeld: `settim = $ffdb`

```
; Stel de TOD-klok in op 14:30:57.5
; (alias 14:30:57.5)
ldy #$14
ldx #$30
lda #$57
ldz #$05
jsr settim
```

SPIN_SPOUT

Adres: JSR $FF4D

Omschrijving:
Stelt de snelle seriële poorten in

Invoer: **C-vlag** Gewist voor SPINP, gezet voor SPOUT

Opmerkingen:
Deze routine heeft betrekking op de aansturing van het "snelle seriële"-protocol van de C1571/1581 op het laagste niveau.

Uit de C65-specificatie:

Het snelle seriële protocol gebruikt CIA_1 (6526 op $DC00) en een speciaal drivercircuit dat gedeeltelijk door het FSDIR-register wordt aangestuurd. SPINP en SPOUT zijn routines die het systeem gebruikt om het CIA- en het drivercircuit voor de snelle seriële verbinding in te stellen voor invoer of uitvoer. SPINP stelt CRA in (CIA_1-register 14) en wist de FSDIR-bit voor invoer. SPOUT stelt CRA, ICR (CIA_1-register 13) en timer A (CIA_l-registers 4 & 5) in en zet de FSDIR-bit voor uitvoer. Merk op dat de toestand van de TOD_IN-bit van CRA altijd behouden blijft. Deze routines zijn alleen nodig voor toepassingen die de snelle seriële bus zelf vanaf het laagste niveau aansturen.

Voorbeeld:
```
spin_spout = $ff4d

        clc
        jsr $ff4d
```

STA_FAR

Adres: JSR $FF77

Omschrijving:
Schrijft een byte naar een adres in een willekeurige bank

Invoer: **A** De op te slaan gegevens
X Basispagina-aanwijzer
Y Index vanaf het aangewezen adres
Z Bank

Opmerkingen:
In wezen is dit het equivalent van `sta (X),y` in bank Z.

Deze routine heeft toegang tot elk adres in de eerste 1 MB van het geheugen. In tegenstelling tot JMPFAR wijzigt zij de geheugenindeling niet. Zij gebruikt een DMA-taak om het lange adres te benaderen.

Op de MEGA65 ondersteunt de 45GS02 een instructie die dit kan doen met een 32-bits adres dat op de basispagina is opgeslagen, zonder dat een KERNAL-routine nodig is: `sta [zp4],z`

Voorbeeld: `sta_far = $ff77`

```
; Bewaar de waarde op adres 4.4502
lda #$00
sta $fe
lda #$45
sta $ff
ldx #$fe  ; basispagina-aanwijzer -> $4500
ldy #$02  ; Y-index
ldz #$04  ; bank
lda #$99  ; de op te slaan waarde
jsr sta_far
```

STOP

Adres: JSR $FFE1

Omschrijving:
Rapporteert de Stop-toets en zet I/O terug indien de toets is ingedrukt

Uitvoer: **Z-vlag** Gezet indien de Stop-toets is ingedrukt

Opmerkingen:
Indien de Stop-toets (Shift + Run/Stop) bij de meest recente aanroep van ScanStopKey is ingedrukt, zet STOP de Zero-vlag (Z), sluit alle actieve kanalen en legt de toetsenbordwachtrij leeg. De KERNAL roept ScanStopKey routinematig aan, of het programma kan de routine expliciet aanroepen indien de KERNAL is uitgeschakeld.

Een programma kan dit gedrag met de ISTOP-vector overschrijven of uitbreiden.

Om de Stop-toets zonder neveneffecten te testen, raadpleegt u ScanStopKey.

Voorbeeld:
```
stop = $ffe1

     ; Wacht op de Stop-toets
-    jsr stop
     bne -
```

SWAPPER

Adres: JSR $FF65

Omschrijving:
Wisselt tussen de tekstmodi 40x25 en 80x25

Opmerkingen:
Deze routine ondersteunt de tekstmodus 80x50 nog niet.

Voorbeeld:
```
swapper = \$ff65

        jsr swapper
```

SYSFLAGS

Adres: JSR $FF47

Omschrijving:
Leest/zet systeemvlaggen

Invoer: Om te zetten:
C-vlag = 0
A De nieuwe vlagwaarde

Om te lezen:
C-vlag = 1

Uitvoer: Bij het lezen: **A** De vlagwaarde

Opmerkingen:
Vergrendelt bits:
Bit 4: GO64-modus (0=ontgrendeld, 1=vergrendeld)
Bit 5: Functietoetsen (0=macro's, 1=ruwe toetsen)
Bit 6: No Scroll vergrendeld (0=ontgrendeld, 1=vergrendeld)
Bit 7: Mega+Shift vergrendeld (0=ontgrendeld, 1=vergrendeld)

De bits 0-3 zijn gereserveerd voor toekomstig gebruik. Om compatibiliteit met de toekomst te waarborgen, leest u altijd de vlagwaarde en behoudt u de waarden van deze bits.

Voorheen bekend als "KEYLOCKS".

Voorbeeld: `sysflags = $ff47`

```
sec
jsr sysflags
ora #%00100000  ; functietoetsmacro's uit
clc
jsr sysflags
```

TALK

Adres: JSR $FFB4

Omschrijving:

Stuurt een "talk"-commando naar een apparaat

Invoer: **A** Het apparaatnummer (FA, 0-31)

Opmerkingen:

Dit is een routine op laag niveau voor de seriële bus. De meeste programma's geven de voorkeur aan de I/O-routines op hoger niveau (OPEN en dergelijke).

Volgens het seriële protocol op laag niveau wordt van de computer verwacht dat hij alle apparaatcommunicatie op de seriële bus coördineert. Elk apparaat heeft een vastgeprogrammeerd apparaatnummer; sommige apparaten, zoals schijfstations, hebben schakelaars waarmee het herkende apparaatnummer te wijzigen is. Alle apparaten zien al het verkeer op de bus en van elk apparaat wordt verwacht dat het van modus wisselt wanneer het een commando met zijn apparaatnummer ziet.

Wanneer de computer gegevens van een apparaat wil lezen, zendt hij het "talk"-commando naar dat apparaat. Telkens heeft slechts één apparaat toestemming om als "talker" op de seriële bus op te treden. Om een andere talker aan te wijzen, zendt de computer een "untalk"-commando uit naar alle apparaten en stuurt hij vervolgens een nieuw "talk"-commando met een ander apparaatnummer.

Gebruik UNTLK om alle apparaten te laten weten dat zij de talk-modus moeten beëindigen.

Een programma kan dit gedrag met de ITALK-vector overschrijven of uitbreiden.

Voorbeeld:

```
talk = $ffb4

       lda #8
       jsr talk
```

TKSA

Adres: JSR $FF96

Omschrijving:

Stuurt een secundair adres naar de talker

Invoer: **A** Secundair adres (SA)

Opmerkingen:

Dit is een routine op laag niveau voor de seriële bus. De meeste programma's geven de voorkeur aan de I/O-routines op hoger niveau (OPEN en dergelijke).

Nadat een programma het TALK-commando naar een randapparaat op de seriële bus heeft gezonden, om het apparaat uitvoer te laten leveren, stuurt TKSA een secundair adres naar het apparaat. Dit wordt doorgaans gebruikt om het apparaat te configureren voordat het programma er gegevens uit leest. Het gedrag hangt af van het apparaat.

Om een secundair adres te sturen naar een apparaat waaraan LISTN is gezonden, gebruikt u SECND.

Een programma kan dit gedrag met de ITALKSA-vector overschrijven of uitbreiden.

Voorbeeld:

```
talk = $ffb4
tksa = $ff96

    lda #8
    jsr talk
    lda #15
    jsr tksa
```

UNLSN

Adres: JSR $FFAE

Omschrijving:
Stuurt het "unlisten"-commando

Opmerkingen:
Dit is een routine op laag niveau voor de seriële bus. De meeste programma's geven de voorkeur aan de I/O-routines op hoger niveau (OPEN en dergelijke).

UNLSN stuurt het "unlisten"-commando naar alle randapparaten op de seriële bus. Zie LISTN.

Een programma kan dit gedrag met de IUNLISTEN-vector overschrijven of uitbreiden.

Voorbeeld:
```
unlsn = $ffae

      jsr unlsn
```

UNTLK

Adres: JSR $FFAB

Omschrijving:
Stuurt het "untalk"-commando

Opmerkingen:
Dit is een routine op laag niveau voor de seriële bus. De meeste programma's geven de voorkeur aan de I/O-routines op hoger niveau (OPEN en dergelijke).

UNTLK stuurt het "untalk"-commando naar alle randapparaten op de seriële bus. Zie TALK.

Een programma kan dit gedrag met de IUNTALK-vector overschrijven of uitbreiden.

Voorbeeld:
```
untlk = $ffab

      jsr untlk
```

VECTOR

Adres: JSR $FF8D

Omschrijving:
Leest/stelt de KERNAL-vectortabel in

Invoer: Om te lezen:
C-vlag = 1
X Doeladres, lage byte
Y Doeladres, hoge byte

Om in te stellen:
C-vlag = 0
X Bronadres, lage byte
Y Bronadres, hoge byte

Opmerkingen:
Een programma kan een aantal gedragingen van de KERNAL, de schermeditor en BASIC overschrijven of uitbreiden door eigen routines te installeren. De KERNAL heeft een vaste reeks uitbreidingspunten, bekend als *vectoren*, waar de KERNAL onder bepaalde omstandigheden naar het bijbehorende vectoradres springt. De standaardwaarden van deze vectoren verwijzen naar routines in de KERNAL-ROM. Een programma installeert een routine door het oorspronkelijke vectoradres te vervangen door het adres van een eigen routine en de eigen routine doorgaans (maar optioneel) zodanig te herschrijven dat deze het oorspronkelijke KERNAL-vectoradres aanroept.

Om nieuwe vectoren te installeren:

1. Gebruik VECTOR om de vectortabel naar een locatie in het geheugen van het programma te kopiëren.
2. Kopieer indien nodig het oorspronkelijke vectoradres in de code van de eigen routine, bijvoorbeeld om na het uitvoeren van een eigen actie de besturing terug te geven aan de oorspronkelijke KERNAL-routine.
3. Schrijf het adres van de eigen routine in de vectortabel in het geheugen.
4. Gebruik VECTOR om de bijgewerkte vectoradressen te installeren.

Sommige Commodore-programmeurs zijn gewend om eigen vectoradressen rechtstreeks uit het interne KERNAL-geheugen te lezen en erheen te schrijven, zonder de VECTOR-routine te gebruiken. Voor de MEGA65 is het belangrijk dat programma's de VECTOR-toegangsroutine gebruiken en niet van de interne KERNAL-geheugenlocatie afhankelijk zijn, om de compatibiliteit met toekomstige versies van de ROM te waarborgen.

OPMERKING: De KERNAL gebruikt een geheugenindeling voor BASIC, welke het grootste deel van het programmageheugen voor de CPU verbergt. Eigen routines moeten in het bereik $1600 - $1EFF staan om zichtbaar te zijn terwijl BASIC en de schermeditor actief zijn. Dit geldt alleen in gevallen waarin het programma de besturing teruggeeft aan BASIC of de schermeditor, een veelvoorkomend gebruik van vectoren. Voor langere eigen routines kunt u een kleine tussenroutine gebruiken die interrupts uitschakelt, de geheugenindeling van de CPU wijzigt en vervolgens een routine ergens anders in het geheugen aanroept. Wanneer die routine terugkeert, herstelt de tussenroutine de KERNAL-geheugenindeling, schakelt de interrupts weer in en springt vervolgens naar de oorspronkelijke vector van de KERNAL.

Wanneer u vectoren bijwerkt die betrekking hebben op interruptafhandeling, schakelt u de interrupts uit voordat u VECTOR aanroept.

Wanneer u vectoren installeert die de besturing teruggeven aan de oorspronkelijke KERNAL-routines, is het verstandig om RESTOR aan te roepen voordat u de tabel met VECTOR leest. Zo ziet de installatieroutine de oorspronkelijke KERNAL-adressen en niet de adressen van een eerdere installatie.

Gebruik RESTOR om de vectortabel terug te zetten op de KERNAL-standaardwaarden.

Vectortabel:

De vectortabel is 56 bytes groot, twee bytes voor elk 16-bits vectoradres.

Vrijwel alle vectoren horen bij KERNAL-sprongtabelroutines. Wanneer een programma één van deze vectoren vervangt, gaan alle aanroepen van het bijbehorende KERNAL-sprongtabeladres voortaan naar de eigen routine. Van de eigen routine wordt verwacht dat hij voldoet aan de gedocumenteerde randvoorwaarden van de KERNAL-routine (zoals CPU-registerargumenten). Indien hij vervolgens de KERNAL-routine aanroept, moet hij bij terugkeer ook aan die randvoorwaarden voldoen. Indien hij de KERNAL-routine niet aanroept, moet hij bij terugkeer voldoen aan de eindvoorwaarden van de routine (zoals de CPU-registerwaarden die als retourwaarden worden teruggegeven).

De vectoren IIRQ, IBRK en INMI verwijzen naar de interrupt-handlers van de KERNAL. De KERNAL gebruikt éénmaal per frame een raster-IRQ. De interrupt-handlers van de KERNAL bewaren alle CPU-registers op de stack, in deze volgorde gepusht: A, X, Y, Z, B. Bij een IRQ toetst de handler vervolgens of de IRQ door het rasterframe of door een BRK-instructie wordt veroorzaakt en springt hij dienovereenkomstig naar de IIRQ- of de IBRK-handler. Bij een NMI schakelt de handler de IRQ's uit en roept hij vervolgens de INMI-handler aan. Indien een eigen routine aan het einde de oorspronkelijke KERNAL-vector aanroept, herstelt de KERNAL de CPU-registers en verlaat hij de interrupt-handler op nette wijze. Indien de

eigen routine ervoor kiest om de oorspronkelijke KERNAL-vector niet aan te roepen, moet hij de vijf bytes op de stack zelf afhandelen.

CTLVEC, SHFVEC en ESCVEC geven een eigen routine de mogelijkheid om afgedrukte en getypte tekens, welke met de Control-, Shift- of Escape-toets te maken hebben, af te vangen. Het accumulatorregister bevat de ongewijzigde PETSCII-code. Een eigen routine kan verdere verwerking annuleren door met RTS terug te keren, of kan de besturing teruggeven aan de KERNAL door de oorspronkelijke KERNAL-vector aan te roepen. Hoewel dit in de eerste plaats bedoeld is om interactief gedrag aan de schermeditor toe te voegen wanneer deze tekens worden getypt, kunnen programma's al deze codes ook afdrukken om dezelfde effecten te bereiken.

KEYSCAN geeft een eigen routine de mogelijkheid om typegebeurtenissen af te vangen terwijl ze van het toetsenbord worden gelezen. Deze vector wordt aangeroepen tijdens de uitvoering van GETIN door de KERNAL, wanneer het invoerapparaat het toetsenbord is. Het accumulatorregister bevat de ongewijzigde PETSCII-code, of $FF om aan te geven dat er geen toets is ingedrukt. Y bevat een bitmasker van de modificatietoetsen; zie de beschrijving van MODKEY in Bijlage F. Indien de C-vlag gewist is, komt de typegebeurtenis in aanmerking om een functietoetsmacro te activeren; indien de vlag gezet is, activeert de gebeurtenis geen macro. De vectorroutine kan elk van deze attributen lezen of wijzigen. Hij moet de besturing teruggeven aan het oorspronkelijke KERNAL-adres in de vector.

Offset	**Naam**	**Beschrijving**
0	IIRQ	Raster-IRQ-handler
2	IBRK	BRK-handler
4	INMI	NMI-handler
6	IOPEN	OPEN
8	ICLOSE	CLOSE
10	ICHKIN	CHKIN
12	ICKOUT	CKOUT
14	ICLRCH	CLRCH
16	IBASIN	BASIN
18	IBSOUT	BSOUT
20	ISTOP	STOP
22	IGETIN	GETIN
24	ICLALL	CLALL
26	EXMON	Gereserveerd, niet wijzigen
28	ILOAD	LOAD
30	ISAVE	SAVE
32	ITALK	TALK

vervolg ...

...vervolg

Offset	Naam	Beschrijving
34	ILISTEN	LISTN
36	ITALKSA	TKSA
38	ISECOND	SECND
40	IACPTR	ACPTR
42	ICIOUT	CIOUT
44	IUNTALK	UNTLK
46	IUNLISTEN	UNLSN
48	CTLVEC	Controlcode
50	SHFVEC	Shiftcode
52	ESCVEC	Escapecode
54	KEYSCAN	Afhandeling van toetsenbordinvoer

Voorbeelden:

Het volgende voorbeeld installeert een korte routine op adres $1600, die de schermcode in de linkerbovenhoek van het scherm verhoogt en stelt vervolgens de IIRQ-vector zodanig in dat de KERNAL deze routine één keer per rasterframe aanroept. Nadat dit programma naar BASIC is teruggekeerd, wordt de schermcode continu bijgewerkt, terwijl BASIC en de schermeditor operationeel blijven.

```
vector = $ff8d

    ; Installeer de custom_irq-routine op $1600,
    ; zodat het zichtbaar is wanneer BASIC
    ; actief is.
    ldx #verlegde_irq_eind-verlegde_irq+1
-   lda custom_irq-1,x
    sta $1600-1,x
    dex
    bne -

    ; Lees de vectortabel in het geheugen in
    sei
    jsr restor
    cli
    sec
    ldx #<vectabel
    ldy #>vectabel
    jsr vector

    ; kopieer de iirq-vector naar custom_irq's
    ; jmp-instructie
    lda vectabel
```

```
    sta $1600+verlegde_irq_return-verlegde_irq+1
    lda vectable+1
    sta $1600+verlegde_irq_return-verlegde_irq+2

    ; verlegde_irq-adres naar iirq (tabelregel 1)
    lda #$00
    sta vectabel
    lda #$16
    sta vectabel+1

    ; Installeer de bijgewerkte vectortabel
    clc
    ldx #<vectabel
    ldy #>vectabel
    sei
    jsr vector
    cli

    ; Terug naar BASIC
    rts

verlegde_irq:
    ; Roteer het teken linksboven
    inc $0800
verlegde_irq_return:
    jmp $0000
verlegde_irq_eind:

vectabel:
    !fill $38
```

Het volgende voorbeeld gebruikt de KEYSCAN-vector om de Help-toets te onderscheppen. In plaats van de macro van de functietoets F15 aan te roepen, wijzigt het indrukken van de Help-toets de randkleur. De PHP- en PLP-instructies zorgen ervoor dat de C-vlag bewaard blijft. Alleen de vectorroutine wordt getoond; de installatiecode lijkt op die van het vorige voorbeeld, waarbij vectabel+54 als adres van de KEYSCAN-vermelding in de vectortabel dient.

```
verlegde_keyscan:
    php
    cmp #132  ; Help-toets
    bne +
    lda #$ff  ; "Help" annuleren
    inc $d020
+   plp
verlegde_keyscan_return:
    jmp $0000
```

```
verlegde_keyscan_eind:
```

VERSIONQ

Adres: JSR $FF2F

Omschrijving:
Laadt het ROM-versienummer in Q.

Opmerkingen:
Deze routine laadt het ROM-versienummer in het Q-register (A, X, Y, Z), als 32-bits geheel getal in little-endian-volgorde.

Voorbeeld:
```
versionq = $ff2f

         jsr versionq
         stq $1700
```

BIJLAGE J

Systeemgeheugenindeling

- Inleiding
- Eigen geheugenindeling van de MEGA65
- $D000 – $DFFF I/O-persoonlijkheden
- Banking van het CPU-geheugen
- C64/C65-ROM-emulatie

INLEIDING

De MEGA65-computer heeft een grote 28-bits adresruimte, waarmee tot 256 MB aan geheugen en in het geheugen gemapte apparaten kunnen worden aangesproken. Deze geheugenindeling kent verschillende verschijningsvormen, afhankelijk van de modus waarin de computer draait. In grove lijnen zijn er vijf hoofdmodi: (1) hypervisormodus; (2) C64-compatibiliteitsmodus; (3) C65-compatibiliteitsmodus; (4) UltiMAX-compatibiliteitsmodus; en (5) MEGA65-modus, of een van de andere modi waarbij de programmeur gebruikmaakt van de uitgebreide functies van de MEGA65.

Het is belangrijk te begrijpen dat de C65 en MEGA65, in tegenstelling tot de C128, toegang bieden tot alle uitgebreide functies vanuit C64-modus, indien de programmeur dat wenst. Dit betekent dat wij, hoewel wij vaak over "C64-modus," "C65-modus" en "MEGA65-modus" spreken, eenvoudigweg gemaksnamen gebruiken voor de MEGA65 waarvan de geheugenindeling (en soms andere functies) zodanig is ingesteld dat zij een omgeving biedt die overeenkomt met de desbetreffende modus. De kern hiervan is de flexibele geheugenindeling van de MEGA65.

In deze bijlage beschrijven wij eerst de eigen geheugenindeling van de MEGA65, dat wil zeggen waar al het geheugen, de I/O-apparaten en andere functies hun plaats hebben in de 28-bits adresruimte. Vervolgens leggen wij uit hoe C64- en C65-compatibele geheugenindelingen vanuit deze 28-bits adresruimte worden benaderd.

EIGEN GEHEUGENINDELING VAN DE MEGA65

De eerste zestien 64KB-banken

De MEGA65 gebruikt voor het eerste MB geheugen een soortgelijke geheugenindeling als de C65, te weten 16 geheugenbanken van 64 KB elk. Dit komt doordat de 4510-CPU van de C65 slechts 1 MB adresruimte kan benaderen. Deze banken kunnen vanuit BASIC 65 worden benaderd met de commando's **BANK**, **DMA**, **PEEK** en **POKE**. De volgende tabel vat de inhoud van de eerste 16 banken samen:

HEX	DEC	Adres	Inhoud
0	0	$0xxxx	Eerste 64KB RAM. Dit is het RAM-geheugen dat in de C64-modus zichtbaar is.
1	1	$1xxxx	Tweede 64KB RAM. Dit is het tweede 64KB RAM-geheugen dat op een C65 aanwezig is.
2	2	$2xxxx	Eerste helft van de C65-ROM (C64-modus en gedeelde onderdelen) *of* RAM
3	3	$3xxxx	Tweede helft van de C65-ROM (C65-modus-onderdelen) *of* RAM
4	4	$4xxxx	Extra RAM (384KB of grotere chip-RAM-modellen)
5	5	$5xxxx	Extra RAM (384KB of grotere chip-RAM-modellen)
6	6	$6xxxx	Extra RAM (*512KB of grotere chip-RAM-modellen)
7	7	$7xxxx	Extra RAM (*512KB of grotere chip-RAM-modellen)
8	8	$8xxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
9	9	$9xxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
A	10	$Axxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
B	11	$Bxxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
C	12	$Cxxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
D	13	$Dxxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
E	14	$Exxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
F	15	$Fxxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)

* Let op: de MEGA65 biedt momenteel alleen een model met 384KB chip-RAM. Toekomstige modellen kunnen grotere hoeveelheden chip-RAM bevatten (zoals 512KB en 1MB).

De belangrijkste kenmerken van deze adresruimte zijn de 128KB RAM in de eerste twee banken, die ook op de C65 aanwezig is. Indien u programma's wilt schrijven die ook op een C65 kunnen draaien, dient u alleen deze twee banken RAM te gebruiken.

Op alle modellen is het mogelijk om alle of een deel van de 128KB "ROM"-ruimte als RAM te gebruiken. Daartoe moet u eerst de Hypervisor verzoeken de schrijfbeveiliging van dit gebied op te heffen, voordat u de inhoud kunt wijzigen. Indien u een

programma schrijft dat vanuit C64-modus start, of anderszins overschakelt naar het C64-deel van de ROM in plaats van het C65-deel, dan kan de tweede helft van die ruimte, te weten BANK 3, veilig voor uw programma's worden gebruikt. Dit geeft een totaal van 192KB RAM, dat op alle modellen van de MEGA65 beschikbaar is.

Op modellen met 384KB of meer chip-RAM zijn BANK 4 en 5 eveneens beschikbaar. Modellen die 1MB of meer chip-RAM bieden, hebben daarnaast BANK 6 tot en met 15 beschikbaar, wat een totaal van 896KB (of 960KB, indien alleen het C64-deel van de ROM nodig is) RAM oplevert voor uw programma's. Let op: de ingebouwde freeze-cartridge van de MEGA65 bevriest momenteel alleen de eerste 384KB RAM.

Kleurengeheugen

De VIC-IV-videokaart van de MEGA65 ondersteunt veel grotere schermen dan de VIC-II of VIC-III. Zij heeft daarom toegang tot een apart kleurengeheugen (colour-RAM), vergelijkbaar met dat van de C64. Voor compatibiliteit met de C65 zijn de eerste twee kilobytes hiervan toegankelijk op $1F800 - $1FFFF. Het volledige 32KB kleurengeheugen bevindt zich op $FF80000 - $FF87FFF. Dit is het eenvoudigst toegankelijk via geavanceerde DMA-bewerkingen of de 32-bits indirecte adresseermethode op de basispagina (zero page) van de processor.

Op het moment van schrijven kunnen de **BANK**- en **DMA**-commando's niet worden gebruikt om het resterende kleurengeheugen te bereiken, omdat het kleurengeheugen zich niet in het eerste megabyte van de adresruimte bevindt. Dit kan in een toekomstige revisie van de MEGA65 worden gecorrigeerd, waardoor toegang tot het volledige kleurengeheugen via BANK 15 of een equivalente DMA-taak mogelijk wordt.

Extra RAM

Naast de 384KB chip-RAM die standaard op alle MEGA65-modellen aanwezig is, hebben de meeste modellen (devkit, release-borden en xemu, maar momenteel NIET op Nexys-borden) ook 8MB extra RAM vanaf $8000000, aangeduid als 'ATTIC RAM'. Dit is niet zichtbaar voor de andere chips (vic/sid/enz.) en kan niet worden gebruikt voor audio-DMA, maar er kan code op worden uitgevoerd (weliswaar langzamer) of het kan worden gebruikt om inhoud op te slaan en via DMA in en uit het chip-RAM te verplaatsen.

Er zijn ook plannen om een PMOD hyperRAM-module te ondersteunen (geïnstalleerd via het klepje onder de MEGA65) om nog eens 8MB RAM vanaf $8800000 te bieden, aangeduid als 'CELLAR RAM'.

28-bits adresruimte

Naast de C65-stijl 1MB adresruimte breidt de MEGA65 deze uit tot 256MB met 28-bits adressen. De volgende tabel toont de indeling op hoofdlijnen van deze adresruimte.

HEX	DEC	Grootte	Inhoud
0000000	0	1	Richtingsregister van de CPU-I/O-poortgegevens
0000001	1	1	CPU-I/O-poortgegevens
0000002 - 005FFFF	2 - 384KB	384KB	Snel chip-RAM (40MHz)
0060000 - 0FFFFFF	384KB - 16MB	15.6MB	Gereserveerd voor toekomstige chip-RAM-uitbreiding
1000000 - 3FFFFFF	16MB - 64MB	48MB	Gereserveerd
4000000 - 7FFFFFF	64MB - 128MB	64MB	Cartridgepoort en andere apparaten op de langzame bus (1 - 10 MHz)
8000000 - 87FFFFF	128MB - 135MB	8MB	8MB ATTIC-RAM (momenteel op alle modellen behalve Nexys)
8800000 - 8FFFFFF	135MB - 144MB	8MB	8MB CELLAR-RAM (geplande PMOD-module, geïnstalleerd via het klepje)
9000000 - EFFFFFF	144MB - 240MB	96MB	Gereserveerd voor toekomstig uitbreidings-RAM
F000000 - FF7DFFF	240MB - 255.49MB	15.49MB	Gereserveerd voor toekomstige I/O-uitbreiding
FF7E000 - FF7EFFF	255.49MB - 255.49MB	4KB	VIC-IV-teken-ROM (alleen schrijven)
FF80000 - FF87FFF	255.5MB - 255.53MB	32KB	VIC-IV-kleurengeheugen (32KB kleurengeheugen - beschikbaar op alle modellen)
FF88000 - FF8FFFF	255.53MB - 255.57MB	32KB	Gereserveerd
FF90000 - FFCAFFF	255.53MB - 255.80MB	216KB	Gereserveerd
FFCB000 - FFCBFFF	255.80MB - 255.80MB	4KB	Geëmuleerd C1541-RAM
FFCC000 - FFCFFFF	255.80MB - 255.81MB	16KB	Geëmuleerde C1541-ROM
FFD0000 - FFD0FFF	255.81MB - 255.81MB	4KB	C64-$Dxxx-I/O-persoonlijkheid
FFD1000 - FFD1FFF	255.81MB - 255.82MB	4KB	C65-$Dxxx-I/O-persoonlijkheid
FFD2000 - FFD2FFF	255.82MB - 255.82MB	4KB	MEGA65-$Dxxx-Ethernet-I/O-persoonlijkheid
FFD3000 - FFD3FFF	255.82MB - 255.82MB	4KB	MEGA65-$Dxxx-normale-I/O-persoonlijkheid

Vervolgd ...

...Vervolgd

HEX	DEC	Grootte	Inhoud
FFD4000 - FFD5FFF	255.82MB - 255.83MB	8KB	Gereserveerd
FFD6000 - FFD67FF	255.83MB - 255.83MB	2KB	Tijdelijk geheugen voor de hypervisor
FFD6000 - FFD6BFF	255.83MB - 255.83MB	3KB	Tijdelijk geheugen voor de hypervisor
FFD6C00 - FFD6DFF	255.83MB - 255.83MB	512	Sectorbuffer van de F011-diskettecontroller
FFD6E00 - FFD6FFF	255.83MB - 255.83MB	512	Sectorbuffer van de SD-kaartcontroller
FFD7000 - FFD70FF	255.83MB - 255.83MB	256	I2C-apparaten van de MEGAphone r1
FFD7100 - FFD71FF	255.83MB - 255.83MB	256	I2C-apparaten van de MEGA65 r2
FFD7200 - FFD72FF	255.83MB - 255.83MB	256	MEGA65-HDMI-I2C-registers (alleen voor R2 en oudere modellen met de ADV7511-HDMI-driverchip)
FFD7300 - FFD7FFF	255.83MB - 255.84MB	3.25KB	Gereserveerd voor toekomstige I2C-apparaten
FFD8000 - FFDBFFF	255.83MB - 255.86MB	16KB	Hypervisor-ROM (alleen zichtbaar in de hypervisormodus)
FFDC000 - FFDDFFF	255.86MB - 255.87MB	8KB	Gereserveerd voor ROM-uitbreiding van de hypervisormodus
FFDE000 - FFDE7FF	255.87MB - 255.87MB	2KB	Gereserveerd voor uitbreiding van de Ethernet-buffer
FFDE800 - FFDEFFF	255.87MB - 255.87MB	2KB	Ethernet-frameleesbuffer (alleen lezen) en Ethernet-frameschrijfbuffer (alleen schrijven)
FFDF000 - FFDFFFF	255.87MB - 255.87MB	4KB	Virtuele FPGA-registers (alleen op geselecteerde modellen)
FFE0000 - FFFFFFF	255.87MB - 256MB	128KB	Gereserveerd

$D000 – $DFFF I/O-PERSOONLIJKHEDEN

De MEGA65 ondersteunt vier verschillende I/O-persoonlijkheden. Deze worden geselecteerd door de juiste waarden naar het $D02F KEY-register te schrijven, dat in alle

vier de I/O-persoonlijkheden zichtbaar is. In Hoofdstuk/Bijlage 12 op bladzijde 12-3 vindt u meer informatie over het gebruik van het KEY-register.

De volgende tabel toont welke I/O-apparaten in elk van deze I/O-modi zichtbaar zijn, evenals de KEY-registerwaarden die worden gebruikt om de I/O-persoonlijkheid te selecteren.

HEX	C64	C65	MEGA65 ETHERNET	MEGA65
KEY	$00	$A5, $96	$45, $54	$47, $53
$D000 - $D02F	VIC-II	VIC-II	VIC-II	VIC-II
$D030 - $D07F	VIC-II[1]	VIC-III	VIC-III	VIC-III
$D080 - $D08F	VIC-II	F011	F011	F011
$D090 - $D09F	VIC-II	-	SD-kaart	SD-kaart
$D0A0 - $D0FF	VIC-II	RAM EXPAND CONTROL	-	-
$D100 - $D1FF	VIC-II	ROOD PALET	ROOD PALET	ROOD PALET
$D200 - $D2FF	VIC-II	GROEN PALET	GROEN PALET	GROEN PALET
$D300 - $D3FF	VIC-II	BLAUW PALET	BLAUW PALET	BLAUW PALET
$D400 - $D41F	SID rechts #1	SID rechts #1	SID rechts #1	SID rechts #1
$D420 - $D43F	SID rechts #2	SID rechts #2	SID rechts #2	SID rechts #2
$D440 - $D45F	SID links #1	SID links #1	SID links #1	SID links #1
$D460 - $D47F	SID links #2	SID links #2	SID links #2	SID links #2
$D480 - $D49F	SID rechts #1	SID rechts #1	SID rechts #1	SID rechts #1
$D4A0 - $D4BF	SID rechts #2	SID rechts #2	SID rechts #2	SID rechts #2
$D4C0 - $D4DF	SID links #1	SID links #1	SID links #1	SID links #1
$D4E0 - $D4FF	SID links #2	SID links #2	SID links #2	SID links #2
$D500 - $D5FF	SID-kopieën	-	Gereser-veerd	Gereser-veerd
$D600 - $D63F	-	UART	UART	UART
$D640 - $D67F	-	UART-kopieën	HyperTrap-registers	HyperTrap-registers
$D680 - $D6FF	-	-	MEGA65-apparaten	MEGA65-apparaten
$D700 - $D7FF	-	-	MEGA65-apparaten	MEGA65-apparaten
$D800 - $DBFF	KLEURENGE-HEUGEN	KLEURENGE-HEUGEN	ETHERNET-buffer	KLEURENGE-HEUGEN
$DC00 - $DDFF	CIAs	CIAs / KLEURENGE-HEUGEN	ETHERNET-buffer	CIAs / KLEURENGE-HEUGEN
$DE00 - $DFFF	CART I/O	CART I/O	ETHERNET-buffer	CART I/O / SD-sectorbuffer

Vervolgd ...

...Vervolgd

HEX	C64	C65	MEGA65 ETHERNET	MEGA65

[1] In de C64-I/O-persoonlijkheid gedraagt $D030 zich zoals op de C128, waarmee kan worden geschakeld tussen 1MHz en 2MHz CPU-snelheid.
[2] De extra MEGA65-SID's zijn zichtbaar in alle I/O-persoonlijkheden.
[3] Sommige modellen kunnen de herhaalde kopieën van de eerste vier SID's vervangen door vier extra SID's, voor een totaal van 8 SID's.

BANKING VAN HET CPU-GEHEUGEN

De 45GS02-processor kan, net als de 6502, slechts 64 KB geheugen tegelijk "zien". Toegang tot aanvullend geheugen verloopt via een reeks bankswitchingmechanismen. Voor backward-compatibiliteit met de C64 en C65 worden de geheugenbankmechanismen van beide computers ondersteund op de MEGA65:

1. Banking via de MAP-instructie in C65-stijl
2. $D030-banking in C65-stijl
3. Cartridge-banking in C64-stijl
4. $00 / $01-banking in C64-stijl

Het MAP-register overschrijdt alle andere bankmechanismen. Dit mechanisme selecteert welke van de acht 8 KB-gebieden van de 16-bits adresruimte $0000 - $FFFF via een offset op andere adressen worden gemapt. Wanneer een gebied gemapt is, zijn de overige bankmechanismen niet van toepassing. Dit geldt zelfs wanneer de offset 0 is, waardoor de 16-bits adressen toegang krijgen tot het RAM in bank 0 (zoals adres 0.D000).

C65-stijl $D030-banking en C64-stijl $00 / $01-banking selecteren beide gebieden die naar bank 2 worden gemapt, die standaard de C64-ROM-code bevat. Deze twee mechanismen overlappen in de gebieden die zij naar ROM kunnen mappen. Wanneer een van de twee mechanismen een gebied naar ROM mappt (en het niet elders door het MAP-register is gemapt), wordt het naar ROM gemapt.

De volgende figuur toont de verschillende vormen van banking die van toepassing kunnen zijn op de verschillende gebieden van de 64 KB die de CPU kan zien.

MAP	MAP LO (4 x 8KB-gebieden)	MAP HI (4 x 8KB-gebieden)				
I/O/CART		CART ROMLO	CART ROMHI		I/O	CART ROMHI
D030		CHARROM	BASIC	INTER-FACE		KERNAL
C64			BASIC		CHAR ROM	KERNAL
RAM	RAM	RAM	RAM	RAM	RAM	RAM
	$0000 - $7FFF	$8000 - $9FFF	$A000 - $BFFF	$C000 - $CFFF	$D000 - $DFFF	$E000 - $FFFF

Er zijn enkele bijkomende complicaties. Wanneer de cartridge bijvoorbeeld de UltiMAX™-gamemodus selecteert, is slechts de eerste 4 KB RAM zichtbaar en wordt de rest van de adresruimte niet gemapt, zodat deze door de cartridge kan worden aangeboden.

C64/C65-ROM-EMULATIE

De C64 en C65 gebruiken ROM-geheugen om het KERNAL- en BASIC-systeem op te slaan. De MEGA65 werkt anders: zij gebruikt 128 KB van de 384 KB snelle chip-RAM op $20000 - $3FFFF (banken 2 en 3) om deze systeemprogramma's op te slaan. Hierdoor is het mogelijk de "ROM" die de MEGA65 gebruikt te wijzigen of bij te werken zonder de computer te openen. Het is zelfs mogelijk het Freeze-menu van de MEGA65 te gebruiken om de gebruikte "ROM" te wijzigen terwijl een programma draait.

De geheugenbankmethoden van de C64 en C65 gebruiken dit 128 KB-gebied wanneer zij ROM-banken zichtbaar maken. Wanneer de RAM-banken gemapt zijn, zijn zij altijd alleen-lezen. Wanneer echter de MAP-instructie of DMA wordt gebruikt om dat adresgebied te bereiken, is het mogelijk ernaar te schrijven. Voor verbeterde backward-compatibiliteit is het volledige 128 KB-geheugengebied standaard op alleen-lezen ingesteld.

Een programma kan evenwel lees-schrijftoegang tot dit 128 KB-geheugengebied aanvragen, zodat het programma het volledige 384 KB chip-RAM van de MEGA65 kan benutten. Dit wordt gerealiseerd door de *ROM Write-enable* systeemtrap van de hypervisor te activeren. Het volgende codefragment demonstreert hoe dit werkt.

```
LDA #$02
STA $D641
CLV
```

Dit fragment werkt door subfunctie $02 (ROM write-enable) van hypervisortrap $01 aan te roepen. Subfunctie $00 van dezelfde trap schakelt de schrijfbeveiliging weer in. De instructie na de STA moet een instructie met één cyclus zijn, doorgaans CLV: de CPU voert die bij het terugkeren uit de Hypervisor al dan niet uit. NOP wordt niet

aanbevolen, omdat deze op de 45GS02 als prefix voor de volgende instructie kan werken. De MEGA65 I/O-persoonlijkheid moet eerst worden geselecteerd, zodat de hypervisor-trap-registers ($D640 - $D67F) zichtbaar worden.

De huidige schrijfbeveiligingstoestand kan worden getest door te proberen naar dit geheugengebied te schrijven. Daarnaast kunt u de huidige toestand inspecteren en wijzigen in het Freeze-menu van de MEGA65.

Opmerking: wanneer u het programma vanuit C65-modus start, moet u er eerst voor zorgen dat het I/O-gebied zichtbaar is op $D000-$DFFF. De eenvoudigste manier om dit te doen is de MAP-instructie te gebruiken met alle waarden nul in de registers. Het volgende fragment demonstreert dit en zorgt er bovendien voor dat de MEGA65 I/O-context actief is, zodat de hypervisortrap kan worden geactiveerd:

```
; Wis de C65-geheugenindeling
LDA #$00
TAX
TAY
TAZ
MAP
; Bank I/O in via het C64-mechanisme
LDA #$35
STA $01
; Activeer de MEGA65 / VIC-IV-I/O-persoonlijkheid
LDA #$47
STA $D02F
LDA #$53
STA $D02F
; Einde van de MAP-reeks, zodat interrupts weer kunnen optreden
EOM
; Hypervisor-aanroep om de schrijfbeveiliging van het ROM-gebied op te heffen
LDA #$02
STA $D641
CLV
```

Indeling van de C65-compatibiliteits-ROM

De indeling van het 128KB C65-compatibiliteits-ROM-gebied komt overeen met die van de C65:

HEX	Inhoud
$3E000 -- $3FFFF	C65 KERNAL
$3D000 -- $3DFFF	CHARSET B
$3C000 -- $3CFFF	GERESERVEERD
$38000 -- $3BFFF	C65-BASIC-grafische routines
$32000 -- $37FFF	C65-BASIC
$30000 -- $31FFF	MONITOR (wordt op $6000 -- $7FFF gemapt)
$2E000 -- $2FFFF	C64 KERNAL
$2D000 -- $2DFFF	CHARSET C
$2C800 -- $2CFFF	INTERFACE
$2C000 -- $2C7FF	GERESERVEERD
$2A000 -- $2BFFF	C64-BASIC
$29000 -- $29FFF	CHARSET A
$24000 -- $28FFF	GERESERVEERD
$20000 -- $23FFF	DOS (wordt op $8000 -- $BFFF gemapt)

Het INTERFACE-programma is een reeks routines die door de C65 worden gebruikt om te wisselen tussen C64-modus, C65-modus en het ingebouwde DOS van de C65. Het DOS bevindt zich in het onderste achtste van het ROM.

BIJLAGE K

45GS02-microprocessor

- Inleiding
- Verschillen ten opzichte van de 6502
- In het geheugen gemapte registers van de C64-CPU
- Nieuwe in het geheugen gemapte CPU-registers
- MEGA65 CPU-versnellingsregisters voor wiskunde
- MEGA65-hypervisormodus

INLEIDING

De 45GS02 is een verbeterde versie van het processorgedeelte van de CSG4510 en van de F018 "DMAgic" DMA-controller die werd gebruikt in de prototypes van de Commodore 65-computer. De 4510 is op zijn beurt een verbeterde versie van de 65CE02. Voor de achterwaarts compatibele werking van de 45GS02 verwijzen wij naar de uitgebreide documentatie die beschikbaar is voor de 6502- en 65CE02-processors.

Dit hoofdstuk zal zich richten op de verschillen tussen de 45GS02 en de eerdere 6502-klasse processors, en op de beschrijving van de vele ingebouwde in het geheugen gemapte I/O-registers van de 45GS02.

VERSCHILLEN TEN OPZICHTE VAN DE 6502

De 45GS02 heeft een aantal belangrijke verschillen met eerdere 6502-klasse processors:

Supervisor-/hypervisormodus

In tegenstelling tot de eerdere 6502-varianten beschikt de 45GS02 over een geprivilegieerde modus. Deze modus is bedoeld voor gebruik door een besturingssysteem of een type-1-hypervisor. De dubbelzinnigheid tussen besturingssysteem en Hypervisor op de MEGA65 vloeit voort uit het feit dat het besturingssysteem van de MEGA65 in feite niet meer is dan een loader en taakwisselaar voor C64- en C65-omgevingen, oftewel: het functioneert in feite als een hypervisor, maar biedt slechts beperkte virtualisatie van de hardware.

De belangrijkste verschillen tussen de normale modus en de supervisor-modus op de MEGA65 zijn dat in supervisor-modus:

- Een speciaal 16KB-geheugengebied wordt toegewezen aan $8000 - $BFFF, dat zowel het programma als de gegevens van het hypervisor-/supervisorprogramma bevat. Dit is doorgaans het Hyppo-programma. Dit geheugen kan niet worden toegewezen wanneer de processor zich in de normale modus bevindt (de chipselectlijn ernaartoe is geblokkeerd), waardoor het wordt beschermd tegen onbedoelde of kwaadwillende toegang.
- De 64 SYSCALL-trapregisters in de MEGA65 I/O-modus op $D640 - $D67F worden vervangen door de virtualisatieregisters. Deze registers bieden volledige controle over het systeem, en het is hun toegang die het privilege van de supervisor-modus werkelijk definieert.

- De processor werkt altijd op volle snelheid (40 MHz) en in de 4510-processorpersoonlijkheid.

De hypervisormodus wordt later in deze bijlage uitgebreider beschreven.

Niet-bedoelde 6502-instructies

De 65C02-, 65CE02- en CSG4510-processors hebben de originele 6502-processor uitgebreid door eerder niet-toegewezen opcodes van de 6502 te gebruiken om extra instructies te bieden. Alle software die de officiële documentatie van de 6502-processor volgde, zal daarom op deze nieuwere processors werken, mogelijk met een andere instructietiming. De veelgebruikte praktijk op de C64 en andere homecomputers van het gebruik van ongedefinieerde of niet-bedoelde opcodes (vaak aangeduid als "illegale opcodes", hoewel er geen wet is die het gebruik ervan verbiedt), betekent echter dat veel bestaande programma's niet op deze nieuwere processors zullen werken.

Om dit probleem te verhelpen, kan de 45GS02 wisselen tussen processorpersoonlijkheden van de 4510 en 6502. Het effect is dat in 6502-modus geen van de nieuwe opcodes van de 65C02, 65CE02, 4510 of 45GS02 beschikbaar zijn, en worden vervangen door het originele, vaak vreemde gedrag van de ongedefinieerde opcodes van de 6502.

> **WAARSCHUWING:** Deze functie is onvolledig en ongetest. De meeste niet-bedoelde 6502-instructies werken niet correct wanneer de 6502-persoonlijkheid is ingeschakeld.

Bugcompatibiliteit van Read-Modify-Write-instructies

De 65CE02-processor heeft een groep instructies geoptimaliseerd die Read-Modify-Write (RMW)-instructies worden genoemd. Bij dergelijke instructies, zoals INC, die de inhoud van een geheugenlocatie verhogen, zou de 6502 de oorspronkelijke waarde lezen en deze ongewijzigd terugschrijven, voordat deze met de nieuwe verhoogde waarde werd teruggeschreven. Voor de meeste doelen veroorzaakte dit geen problemen. Het bleek echter een snelle manier te zijn om VIC-II-onderbrekingen te bevestigen, omdat het terugschrijven van de oorspronkelijke waarde (wat de instructie niet hoeft te doen) de onderbreking bevestigt. Deze methode is sneller en gebruikt minder bytes dan elk alternatief, en werd daarom veel gebruikt in C64-software.

Het probleem ontstond bij de C65 met de van de 65CE02 afgeleide CSG4510, die deze extra schrijfbeweging tijdens de RMW-instructies niet uitvoerde. Hierdoor waren de RMW-instructies één cyclus sneller, waardoor software iets sneller draaide. Dit betekende helaas ook dat veel bestaande C64-software eenvoudigweg niet op een C65 zal werken, tenzij de onderbrekingsbevestigingscode in elk programma wordt aangepast om dit probleem te omzeilen. Dit is de belangrijkste reden waarom veel C64-spellen en andere softwaretitels niet op een C65 werken.

Omdat dit probleem zo vaak voorkomt, bevat de 45GS02 van de MEGA65 bugcompatibiliteit met deze veelgebruikte functie van de originele 6502. Dit doet de pro-

cessor door te controleren of het doel van een RMW-instructie $D019 is, oftewel het onderbrekingsstatusregister van de VIC-II. Indien dit het geval is, voert de 45GS02 de dummy-schrijfbeweging uit, waardoor veel C64-softwaretitels die niet op een C65-prototype werken, ongewijzigd op de MEGA65 kunnen worden uitgevoerd. Door de dummy-schrijfbeweging alleen uit te voeren wanneer het adres $D019 is, behoudt de MEGA65 C64-compatibiliteit, zonder de snelheidsverbetering op te offeren voor alle andere toepassingen van deze instructies.

Variabele CPU-snelheid

De 45GS02 kan draaien op 1MHz, 2MHz, 3.5MHz en 40MHz, om software te ondersteunen die is ontworpen voor de C64, C128 in C64-modus, C65 en MEGA65.

Langzame werking (1 MHz - 3.5 MHz)

In deze modi vertraagt de 45GS02-processor, zodat hetzelfde aantal instructies per videoframe wordt uitgevoerd als op een PAL- of NTSC-C64, C128 in C64-modus of een C65-prototype. Dit stelt bestaande software in staat om op de MEGA65 op de juiste snelheid en met minimale weergaveproblemen te draaien. De VIC-IV-videoregelaar levert cyclusindicatiepulsen aan de 45GS02 die worden gebruikt om de tijd bij te houden.

In deze modi hebben opcodes hetzelfde aantal cycli als een 6502. Geheugentoegangen binnen een instructie zijn echter niet gegarandeerd in dezelfde cyclus te vinden als op een 1MHz-6502. Normaal gesproken is het effect dat instructies sneller voltooid worden en de processor wacht totdat het juiste aantal cycli is verstreken. Dit betekent dat de timing tot 7 microseconden onjuist kan zijn. Dit is doorgaans geen probleem, en zelfs veel C64-snelladers werken correct. Bijvoorbeeld kan het GEOS™ Graphical Operating System voor de C64 worden opgestart en gebruikt vanaf een 1541 die op de seriële poort van de MEGA65 is aangesloten.

Enkele geavanceerde VIC-II-grafiektrucs, zoals Variable Screen Position (VSP), zullen hoogstwaarschijnlijk niet correct werken vanwege de onzekerheid in de timing van de geheugenschrijfcycli van instructies. In de meeste gevallen kunnen dergelijke problemen echter eenvoudig worden opgelost door de geavanceerde mogelijkheden van de VIC-IV-videoregelaar van de MEGA65 te gebruiken. VSP is bijvoorbeeld overbodig op de MEGA65, omdat u het schermRAM-adres op elke locatie in het geheugen kunt instellen.

Instructietiming op volle snelheid (40 MHz)

Wanneer de processor van de MEGA65 op volle snelheid werkt (momenteel 40 MHz), komt de instructietiming niet langer exact overeen met die van de 6502: instructies die in minder cycli kunnen worden uitgevoerd, doen dat ook. Zo vereisen sprongen op de 45GS02 doorgaans minder cycli. Er zijn ook instructies die op de 45GS02 meer cycli vereisen, met name de instructies LDA, LDX, LDY en LDZ. Deze instructies vereisen doorgaans één extra cyclus. Omdat de processor echter op 40 MHz werkt, worden

deze instructies nog steeds veel sneller uitgevoerd dan op een C65 of C64 met een accelerator.

Direct Memory Access (DMA)

Direct Memory Access (DMA) is een methode om geheugenregio's snel te vullen, te kopiëren of te verwisselen. De MEGA65 implementeert een verbeterde versie van de F018 "DMAgic" DMA-controller van de C65-prototypen. Anders dan bij de C65-prototypen maakt de DMA-controller deel uit van de CPU op de MEGA65.

Gedetailleerde informatie over het gebruik van de DMA-controller en deze geavanceerde mogelijkheden is te vinden in Hoofdstuk/Bijlage P op bladzijde P-3

Geheugen benaderen tussen 64 KB en 1 MB

De C65 bood vier manieren om geheugen voorbij het 64 KB-punt te benaderen: drie methoden die beperkt, gespecialiseerd of beide zijn, en twee algemeen toepasbare methoden. Eerst bespreken we de beperkte methoden, waarna we de algemeen toepasbare methoden documenteren.

Geheugenbanking in C64-stijl

De eerste methode is het gebruik van de C64-stijl $00/$01 ROM/RAM-banking. Deze methode is echter zeer beperkt, omdat zij alleen het in- en uitbanken van de twee 8 KB-regio's toestaat die overeenkomen met de C64 BASIC- en KERNAL-ROM's. Deze bevinden zich op $2A000 en $2E000 in de 20-bits C65-adresruimte, d.w.z. $002A000 en $002E000 in de 28-bits adresruimte van de MEGA65. Zij kan ook toegang geven tot de C64 character-ROM-gegevens op $D000, die zich op $2D000 in de C65-geheugenindeling bevinden, en dus op $002D000 in de adresruimte van de MEGA65. Behalve dat deze methode beperkt is in welke regio's zij kan benaderen, biedt zij ook alleen leestoegang tot deze geheugenregio's, d.w.z. dat zij niet kan worden gebruikt om deze geheugenregio's te wijzigen.

VIC-III "ROM"-banking

De C65 bevat nog een voorziening om C64-ROM-regio's zichtbaar te maken in de C65-modus, naast het regelen van de zichtbaarheid van het kleurengeheugen in de I/O-registerregio. Deze worden in- en uitgebankt met behulp van diverse bits van het $D030-register van de VIC-III:

$D030 Bit	Signaal-naam	20-bitsadres	16-bitsadres	Lees-/Schrijf-toegang
0	CRAM2K	$1F800 - $1FFFF, $FF80000 - $FF807FF	$D800 - $DFFF	J

Vervolgd ...

...Vervolgd

$D030 Bit	Signaal-naam	20-bitsadres	16-bitsadres	Lees-/Schrijf-toegang
3	ROM8	$28000 – $29FFF	$8000 – $9FFF	N
4	ROMA	$2A000 – $2BFFF	$A000 – $BFFF	N
5	ROMC	$2C000 – $2CFFF	$C000 – $CFFF	N
7	ROME	$2E000 – $2FFFF	$E000 – $FFFF	N

De eerste 2 KB van het kleurengeheugen zijn altijd zichtbaar op $1F800 – $1FFFF. De eerste 1 KB hiervan is ook zichtbaar op $D800 – $DBFF. Wanneer het CRAM2K-signaal 1 is, wordt de tweede kilobyte beschikbaar gesteld op $DC00 – $DFFF. Anders dan bij de C64 is het kleurengeheugen op de MEGA65 altijd zichtbaar als 8-bits bytes. Ook is het kleurengeheugen op de MEGA65 32 KB groot en bevindt het zich op $FF80000 – $FF87FFF. De zichtbaarheid van het kleurengeheugen op $1F800 – $1FFFF wordt bereikt door schrijfbewerkingen naar beide regio's te spiegelen wanneer het kleurengeheugen via dit mechanisme wordt benaderd.

Zowel de VIC-III-banking als de C64-stijl-banking kunnen van toepassing zijn op $A000 – $BFFF en $E000 – $FFFF. Zij hebben hetzelfde effect van het mappen van die regio's naar ROM-adressen in bank 2. Indien een van beide mechanismen actief is voor een regio, dan wordt de regio naar ROM gemapt.

VIC-ROM-banking heeft alleen toegang tot de C64-ROM. De C65/MEGA65-ROM wordt benaderd met behulp van het MAP-register.

Display Address Translator van de VIC-III

De derde gespecialiseerde manier om toegang te krijgen tot geheugen boven het 64 KB-punt is het gebruik van de Display Address Translator van de VIC-III. Het gebruik van dit mechanisme is gedocumenteerd in Hoofdstuk/Bijlage Q op bladzijde Q-5.

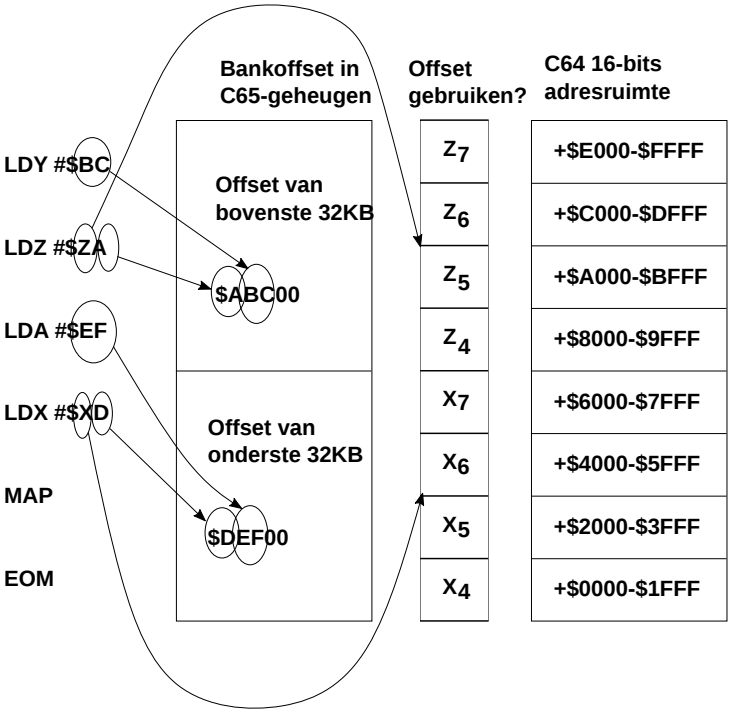
De MAP-instructie

De eerste algemeen toepasbare manier van toegang tot het geheugen is de MAP-instructie van de 4510-processor. De 45GS02-processor van de MEGA65 ondersteunt dit mechanisme eveneens. Deze instructie verdeelt het 64 KB-adres van de 6502 in acht blokken van 8 KB elk. Voor elk van deze blokken kan het blok normaal worden benaderd, d.w.z. dat een 8 KB-regio van de eerste 64 KB RAM van het systeem wordt benaderd. Alternatief kan elk blok in plaats daarvan worden hermapt (vandaar de naam van de MAP-instructie) naar een andere plaats in de adresruimte, door een offset aan het adres toe te voegen. Gemapte adressen in de eerste 32 KB gebruiken een offset, de lagere offset, en de tweede 32 KB gebruikt een andere, de hogere offset. Het hermappen van geheugen met behulp van de MAP-instructie gaat vóór alle andere bankingmechanismen.

De offsets moeten een veelvoud van 256 bytes zijn en dus uit 12 bits bestaan, om een willekeurige offset in de 1 MB-adresruimte van de C65 mogelijk te maken. Aangezien elk 8 KB-blok in een 32 KB-helft van het geheugen al dan niet kan worden gemapt, is hiervoor één bit per 8 KB-blok vereist. De processor vereist dus 16 bits aan informatie voor elke geheugenhelft, voor in totaal 32 bits aan informatie. Dit wordt bereikt door de A- en X-registers in te stellen voor de onderste geheugenhelft en de Y- en Z-registers voor de bovenste geheugenhelft, alvorens de MAP-instructie uit te voeren.

De MAP-instructie kopieert de inhoud van deze registers naar de interne registers van de processor die de mappinginformatie bevatten.

Het volgende diagram illustreert hoe de MAP-instructie de waarden van de vier A-, X-, Y- en Z-registers gebruikt om de bovenste en onderste adresoffsets te berekenen en de bank-enablebits in te stellen voor elk van de acht 8KB-geheugenregio's van de 6502-adresruimte:

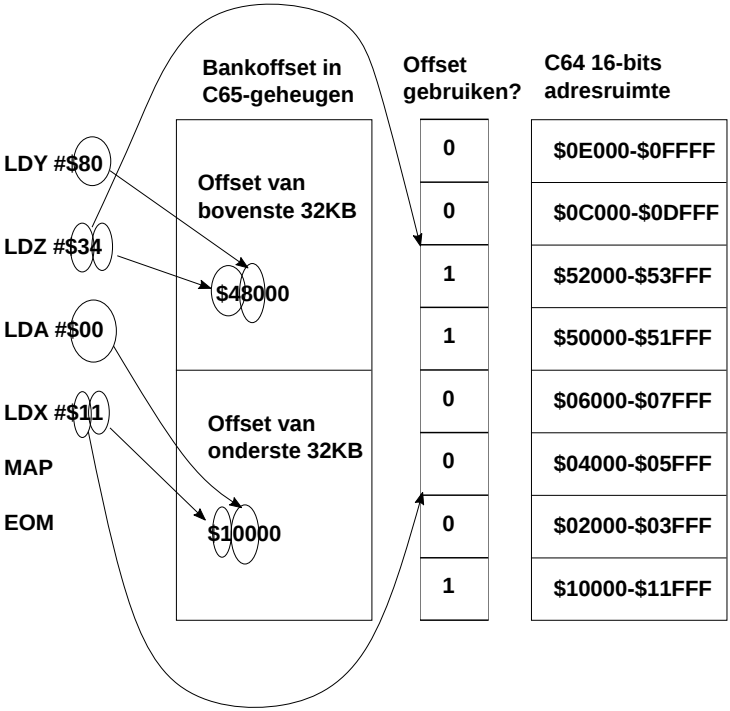


De inhoud van het A-register en de onderste nibble van het X-register vormen samen een 12-bit waarde die met 256 wordt vermenigvuldigd om de offset te produceren die wordt gebruikt voor elk van de 8KB-banken in het onderste 32KB-gedeelte van de 16-bit adresruimte van de 6502. De bovenste nibble van het X-register wordt als vlaggen gebruikt om aan te geven welke van de vier 8KB-blokken in dat 32KB-gedeelte van de 6502-adresruimte de offset bij hun adres moeten optellen om het werkelijke adres te berekenen.

De Y- en Z-registers worden op vergelijkbare manier gebruikt om de offset te produceren voor het bovenste 32KB-gedeelte van de 6502-adresruimte, en de vlaggen om aan te geven of de offset wordt gebruikt voor elk van de vier 8KB-blokken in dat gedeelte van de adresruimte.

De onderste 8 bits van de offset kunnen niet worden ingesteld. De offset is dus een veelvoud van 256 bytes, in tegenstelling tot sommige uitgebreide 6502-processoren. In de praktijk is deze beperking echter zelden van invloed.

Om te begrijpen hoe dit in de praktijk werkt, laat het volgende voorbeeld zien hoe dit werkt met een concreet voorbeeld, met de adresbereiken die zichtbaar zijn in elk van de 8KB-segmenten van de 64KB-adresruimte van de 6502:



Merk op dat de offsets voor elk van de twee 32KB-adresbereiken worden opgeteld bij het 6502-adres. Vandaar dat de offset van $48000 voor het bovenste 32KB-gedeelte een adres van $50000 genereert bij het 6502-adres $8000.

Zie ook "De MAP-instructie gebruiken om >1MB te benaderen" voor verdere uitleg.

DMA-controller

De F018/F018A DMA-controller van de C65 maakt het mogelijk om gegevens in het geheugen op willekeurige locaties in de 1MB-adresruimte snel te vullen, te kopiëren

en te wisselen. Gedetailleerde informatie over de F018 DMA-controller en de verbeteringen die de MEGA65 daarop heeft aangebracht, vindt u in Hoofdstuk/Bijlage P op bladzijde P-3

Platte geheugentoegang

Toegang tot geheugen buiten het 1MB-punt

De MEGA65 kan tot 256MB geheugen ondersteunen. Dit is meer dan de 1MB-adresruimte van de CSG4510 waarop deze is gebaseerd. Er zijn verschillende manieren om dit te bereiken.

De MAP-instructie gebruiken om >1MB te benaderen

De volledige adresruimte is beschikbaar voor de MAP-instructie voor traditionele C65-stijl geheugenmapping, hoewel enige voorzichtigheid geboden is, omdat de MAP-instructie tot drie keer moet worden aangeroepen. De reden hiervoor is dat de MAP-instructie eerst moet worden aangeroepen om te selecteren welk megabyte geheugen zal worden gebruikt voor de onderste en bovenste mapregio's, voordat deze opnieuw op de normale manier wordt aangeroepen om de geheugenmapping in te stellen. Omdat tussen deze twee aanroepen de geheugenmapping-offset een mix van oude en nieuwe adressen zal zijn, moet alle mapping eerst worden uitgeschakeld via de MAP-instructie. Dit betekent dat de code voor het hermappen van geheugen zich in het onderste 64KB RAM of in een van de ROM-bankbare regio's moet bevinden, zodat deze tijdens het mappingproces zichtbaar blijft.

Als deze situatie niet goed wordt afgehandeld, zal de processor instructies uitvoeren vanuit een onverwachte geheugenlocatie halverwege het proces, doordat de routine die hij uitvoert om de mapping uit te voeren plotseling niet meer in het geheugen gemapt is.

Vanwege de relatieve complexiteit van dit proces en de andere beperkingen van de MAP-instructie als middel om geheugen te benaderen, raden we u aan om voor gegevens buiten de huidige geheugenindeling ofwel DMA of de flat-memory adresvoorzieningen van de 45GS02 te gebruiken die hieronder worden beschreven. Toegang tot de volledige adresruimte via de MAP-instructie is alleen beschikbaar voor de volledigheid.

Als een ander voorbeeld van hoe de MAP-instructie kan worden gebruikt om een geheugengebied uit de uitgebreide adresruimte te mappen, laat het volgende programma de Ethernet-framebuffer vanaf de natuurlijke locatie op $FFDE8000 op $6800 verschijnen. Om het voorbeeld zo eenvoudig mogelijk te houden, gaan we ervan uit dat de code in het onderste 64KB RAM draait en niet in het gebied tussen $6000 - $8000.

Omdat de MAP-instructie normaal gesproken alleen de C65-stijl 20-bit adressen kent, moet de MEGA65-uitbreiding op de instructie worden gebruikt om de bovenste 8 bits van de 28-bit MEGA65-adressen in te stellen, dat wil zeggen welk megabyte adresruimte moet worden gebruikt voor de adresvertaling. Dit wordt gedaan door het X-

register in te stellen op $0F wanneer het megabyte-nummer voor de onderste 32KB van de C64-stijl 64KB-adresruimte wordt gedefinieerd. Dit veroorzaakt geen incompatibiliteit met enig redelijk gebruik van de MAP-instructie op een C65, omdat deze waarde aangeeft dat geen van de vier 8KB-geheugenblokken opnieuw zal worden gemapt, maar tegelijkertijd specificeert dat de bovenste 4 bits van de adresoffset voor het hergemapte blok de niet-nul-waarde $F vormt. Het megabyte-nummer wordt vervolgens gespecificeerd door het A-register in te stellen.

Dezelfde aanpak geldt voor de bovenste 32KB, maar met de Z- en Y-registers in plaats van de X- en A-registers. In dit geval hoeven we de bovenste 32KB geheugen niet te hermappen; we laten de Z- en Y-registers op nul staan. We moeten echter X en A instellen om het megabyte-nummer voor de onderste 32KB op $FF te zetten. A moet dus de waarde $FF hebben. Om de onderste 20 bits van de adresoffset in te stellen, gebruiken we de MAP-instructie een tweede keer, dit keer op de normale C65-manier. Omdat we $6800 naar $FFDE800 willen hermappen en al met de $FFxxxxx-offset via het megabyte-nummer hebben afgerekend, hoeven we alleen de offset toe te passen zodat $6800 naar $DE800 wijst. $DE800 minus $6800 = $D8000. Omdat de MAP-instructie werkt met een hermappinggranulariteit van 256 bytes = $100, kunnen we de laatste twee cijfers van $D8000 laten vallen om de MAP-offset van $D80 te verkrijgen. De onderste 8 bits, $80, moeten in het A-register worden geladen. De bovenste 4 bits, $D, moeten in de onderste nibble van het X-register worden geladen. Omdat we de mapping alleen op het vierde van de vier 8KB-blokken willen toepassen die de onderste 32KB-helft van de C64-geheugenindeling vormen, moeten we het 4e bit van de bovenste nibble instellen. Dat wil zeggen, de bovenste nibble moet worden ingesteld op %1000, oftewel $8. Het X-register moet dus worden geladen met $8D. Hiermee verkrijgen we het volledige voorbeeldprogramma:

```
; Map de Ethernet-registers op $6000 - $7FFF

; De Ethernet-controller bevindt zich feitelijk op $FFDE000 - $FFDEFFF, dus selectee
LDA #$ff
LDX #$0f
LDY #$00
LDZ #$00
map

; Activeer nu het mappen van $DE000-$DFFFF op $6000
; De MAP's zijn op offset gebaseerd, dus moet $6000 van het doeladres worden afgetro
; $DE000 - $6000 = $D8000
LDA #$80
LDX #$8d
LDY #$00
LDZ #$00
map
EOM

; De Ethernet-buffer is nu zichtbaar op $6800 - $6FFF
```

De EOM (End Of Mapping)-instructie (hetzelfde als NOP op een 6502, oftewel opcode $EA) werd alleen na de laatste MAP-instructie opgenomen om er zeker van te zijn dat er geen interrupts konden optreden terwijl de geheugenindeling gemengde waarden bevatte met het megabyte-nummer ingesteld, maar de onderste bits van het mappingadres nog niet waren bijgewerkt.

Er is geen voorbeeld in BASIC voor de MAP-instructie mogelijk, omdat de MAP een machinetaalinstructie is van de 4510/45GS02-processoren.

Platte geheugentoegang

De 45GS02 maakt het eenvoudig om een byte vanuit elke geheugenlocatie te lezen of te schrijven door de Zero-Page Indirect-adresseermethode een 32-bit pointer te laten gebruiken in plaats van de gebruikelijke 16-bit pointer. Dit wordt bereikt door de Z-geïndexeerde Zero-Page Indirect-adresseermethode voor de toegang te gebruiken, en de instructie direct te laten voorafgaan door een NOP-instructie (opcode $EA). Bijvoorbeeld:

```
NOP
LDA ($45),Z
```

Als u de ACME-assembler of een andere assembler gebruikt die de 45GS02-uitbreidingen ondersteunt, kunt u in plaats daarvan vierkante haakjes gebruiken om aan te geven dat u een flat-memory-bewerking uitvoert. Dergelijke assemblers voegen de $EA-prefix automatisch voor u in. Bijvoorbeeld:

```
LDA [$45],Z
```

Ongeacht welk hulpmiddel u gebruikt, in dit voorbeeld zouden de vier bytes Zero-Page-geheugen op $45 - $48 lezen om een 32-bit geheugenadres te vormen, en de waarde van het Z-register hieraan optellen om het werkelijke adres te vormen waaruit wordt gelezen. De bytevolgorde in het adres is hetzelfde als bij de 6502, oftewel het rechtse (minst significante) byte van het adres wordt als eerste gelezen vanaf het eerste adres ($45 in dit geval), enzovoort, totdat het meest linkse (meest significante) byte wordt gelezen vanaf $48. Om bijvoorbeeld vanaf geheugenlocatie $12345678 te lezen, moet de inhoud van het geheugen beginnend op $45 bestaan uit 78 56 34 12.

Deze methode is veel efficiënter en ook eenvoudiger dan het gebruik van de MAP-instructie of de DMA-controller voor enkelvoudige geheugentoegangen, en is wat wij doorgaans aanbevelen. De DMA-controller kan worden gebruikt voor het verplaatsen/vullen van grotere geheugenregio's. Wij bevelen de MAP-instructie alleen aan voor het banken van code, of in zeldzame situaties waarbij uitgebreide toegang tot een klein geheugengebied vereist is en de extra cycli voor het lezen van de 32-bit-adressen problematisch zijn.

Virtueel 32-bit-register

De 45GS02 maakt het mogelijk om zijn vier algemene registers, A, X, Y en Z (A is LSB, Z is MSB), als één enkel virtueel 32-bit-register te gebruiken, ook wel het *Q-pseudoregister* genoemd. Dit kan veelvoorkomende bewerkingen aanzienlijk vereenvoudigen en versnellen, en helpt veelvoorkomende programmeerfouten te voorkomen. Het optellen van twee 16-bit- of 32-bit-waarden kan bijvoorbeeld eenvoudig worden uitgevoerd met iets als:

```
; Wis de carry v  r het optellen, zoals gebruikelijk
CLC
; Laat twee NEG-instructies voorafgaan om de virtuele 32-bit-registermodus te sele
NEG
NEG
LDA $1234  ; Laadt de inhoud van $1234-$1237 in respectievelijk A, X, Y en Z
; En nogmaals, voor de optelling
NEG
NEG
ADC $1238  ; Telt de inhoud van $1238-$123B op
; Het resultaat van de optelling staat nu in A, X, Y en Z.
; En kan geheel of gedeeltelijk worden weggeschreven

; Om het geheel weg te schrijven is wederom de NEG + NEG-prefix nodig
NEG
NEG
STA $123C ; Schrijft het geheel weg naar $123C-$123F

; Of om de laagste bytes weg te schrijven, kan eenvoudigweg de inhoud van A en X n
STA $1240
STX $1241
```

Deze aanpak werkt met de LDA-, STA-, ADC-, SBC-, CMP-, EOR-, AND-, BIT-, ORA-, ASL-, ASR-, LSR-, ROL-, ROR-, INC- en DEC-instructies. Indien u ACME of een andere 45GS02-ondersteunende assembler gebruikt, kunt u in plaats daarvan de nieuwe **LDQ**, **STQ**, **ADCQ**, **SBCQ**, **CPQ**, **EORQ**, **ANDQ**, **BITQ**, **ORQ**, **ASLQ**, **ASRQ**, **LSRQ**, **ROLQ**, **RORQ**, **INQ** en **DEQ** mnemonics gebruiken. Het vorige voorbeeld wordt dan:

```
; Wis de carry v  r het optellen, zoals gebruikelijk
CLC
LDQ  $1234  ; Laadt de inhoud van $1234-$1237 in respectievelijk A, X, Y en Z
ADCQ $1238  ; Telt de inhoud van $1238-$123B op
; Het resultaat van de optelling staat nu in A, X, Y en Z.
; En kan geheel of gedeeltelijk worden weggeschreven

STQ $123C ; Schrijft het geheel weg naar $123C-$123F

; Of om de laagste bytes weg te schrijven, kan eenvoudigweg de inhoud van A en X r
STA $1240
STX $1241
```

De virtuele 32-bit-adresseermethode werkt met elke adresseermethode. Geïndexeerde adresseermethoden, waarbij X, Y of Z bij het adres wordt opgeteld, moeten echter met voorzichtigheid worden gebruikt, omdat deze registers in feite een deel van een 32-bit-waarde kunnen bevatten.

De uitzondering is de Zero-Page Indirect Z-geïndexeerde adresseermethode: in dit geval wordt het Z-register NIET bij het doeladres opgeteld (met uitzondering van de LDQ-opcode), in tegenstelling tot wat normaal het geval is. Hierdoor kan het virtuele 32-bit-register worden gebruikt met platte-geheugentoegang met de gecombineerde prefix **NEG NEG NOP** vóór de instructie, zodat een 32-bit-waarde ergens in het geheugen met één instructie kan worden benaderd.

Merk op dat het virtuele 32-bit-register niet kan worden gebruikt in de immediate-modus, bijvoorbeeld om een constante in de vier algemene registers te laden of om een constante waarde op te tellen of af te trekken. Dit dient om problemen met instructies van variabele lengte te voorkomen.

Voor LDQ en STQ bespaart dit hoogstens één byte ten opzichte van LDA #$nn ... LDZ #$nn, en zou het niet sneller zijn. Voor veel veelvoorkomende waarden, zoals #$00000000, bestaan er zelfs verkortingen, zoals:

```
LDA #$00
TAX
TAY
TAZ
```

Indien u een 32-bit-immediate-waarde moet optellen of aftrekken, kan dit vereisen dat u de argumenten herschikt of andere kleine aanpassingen uitvoert. Om bijvoorbeeld de som van de inhoud van het geheugen en een immediate-waarde te berekenen, kunt u de A-, X-, Y- en Z-registers laden met de immediate-waarde en vervolgens **ADCQ** gebruiken met het geheugenadres:

```
; Haal de immediate-waarde #$12345678 in Q
LDA #$78
LDX #$56
LDY #$34
LDZ #$12
; Telt de inhoud van de geheugenlocaties $1234-$1237 op
NEG
NEG
ADC $1234
; Slaat het resultaat op in $1234-$1237
NEG
NEG
STA $1234
```

Wanneer u opnieuw de ACME- of een andere 45GS02-ondersteunende assembler gebruikt, kan dit compacter en duidelijker worden geschreven als volgt. Merk op dat in beide gevallen dezelfde bytevolgorde van machinetaal wordt geproduceerd en het programma hetzelfde aantal cycli zal kosten om uit te voeren.

```
; Haal de immediate-waarde #$12345678 in Q
LDA #$78
LDX #$56
LDY #$34
LDZ #$12
; Telt de inhoud van de geheugenlocaties $1234-$1237 op
ADCQ $1234
; Slaat het resultaat op in $1234-$1237
STQ $1234
```

IN HET GEHEUGEN GEMAPTE REGISTERS VAN DE C64-CPU

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
00	0	PORTDDR							
01	1	PORT					C64MAP		

- **C64MAP** Geheugenmapper van de processorport
- **PORT** 6510/45GS10 CPU-portgegevens
- **PORTDDR** 6510/45GS10 CPU-port-DDR. Schrijf 65 om de 40MHz-modus af te dwingen, of 64 om de normale snelheidscontroles weer in te schakelen.

NIEUWE IN HET GEHEUGEN GEMAPTE CPU-REGISTERS

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D640	54848	HTRAP00							
D641	54849	HTRAP01							
D642	54850	HTRAP02							
D643	54851	HTRAP03							
D644	54852	HTRAP04							
D645	54853	HTRAP05							
D646	54854	HTRAP06							
D647	54855	HTRAP07							
D648	54856	HTRAP08							
D649	54857	HTRAP09							
D64A	54858	HTRAP0A							
D64B	54859	HTRAP0B							
D64C	54860	HTRAP0C							
D64D	54861	HTRAP0D							
D64E	54862	HTRAP0E							
D64F	54863	HTRAP0F							
D650	54864	HTRAP10							
D651	54865	HTRAP11							
D652	54866	HTRAP12							
D653	54867	HTRAP13							
D654	54868	HTRAP14							
D655	54869	HTRAP15							
D656	54870	HTRAP16							
D657	54871	HTRAP17							
D658	54872	HTRAP18							
D659	54873	HTRAP19							
D65A	54874	HTRAP1A							
D65B	54875	HTRAP1B							
D65C	54876	HTRAP1C							
D65D	54877	HTRAP1D							
D65E	54878	HTRAP1E							
D65F	54879	HTRAP1F							
D660	54880	HTRAP20							
D661	54881	HTRAP21							
D662	54882	HTRAP22							
D663	54883	HTRAP23							
D664	54884	HTRAP24							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D665	54885	HTRAP25							
D666	54886	HTRAP26							
D667	54887	HTRAP27							
D668	54888	HTRAP28							
D669	54889	HTRAP29							
D66A	54890	HTRAP2A							
D66B	54891	HTRAP2B							
D66C	54892	HTRAP2C							
D66D	54893	HTRAP2D							
D66E	54894	HTRAP2E							
D66F	54895	HTRAP2F							
D670	54896	HTRAP30							
D671	54897	HTRAP31							
D672	54898	HTRAP32							
D673	54899	HTRAP33							
D674	54900	HTRAP34							
D675	54901	HTRAP35							
D676	54902	HTRAP36							
D677	54903	HTRAP37							
D678	54904	HTRAP38							
D679	54905	HTRAP39							
D67A	54906	HTRAP3A							
D67B	54907	HTRAP3B							
D67C	54908	HTRAP3C							
D67D	54909	HTRAP3D							
D67E	54910	HTRAP3E							
D67F	54911	HTRAP3F							
D710	55056	-		BADEXTRA		BRCOST	-	SLIEN	BADLEN
D7EF	55279	HWRNGRAND							
D7FA	55290	FRAMECOUNT							
D7FB	55291	-						CARTEN	-
D7FD	55293	NOEX-ROM	NOGAME	-					POWEREN
D7FE	55294	HWRNG-NOTRDY	-	SLOWS-DRAM	SELS-DRAM	SLOWCACHE ADV	SLOWCA-CHE	OCEANA	PRE-FETCH

- **BADEXTRA** Kosten van badlines minus 40. D.w.z. 00=40 cycli, 11 = 43 cycli.
- **BADLEN** Schakelt badline-emulatie in
- **BRCOST** 1=extra cyclus(sen) rekenen voor genomen vertakkingen
- **CARTEN** 1=cartridges inschakelen

- **FRAMECOUNT** Telt het aantal verstreken videoframes
- **HTRAPXX** Schrijven activeert een hypervisor-trap $XX
- **HWRNGNOTRDY** Echte RNG-willekeurig getal van de hardware nog niet gereed (alleen-lezen)
- **HWRNGRAND** Echt willekeurig getal van de hardware-RNG (lezen activeert de generatie)
- **NOEXROM** Overschrijving voor /EXROM: moet 0 zijn om het /EXROM-signaal in te schakelen
- **NOGAME** Overschrijving voor /GAME: moet 0 zijn om het /GAME-signaal in te schakelen
- **OCEANA** Schakelt emulatie van cartridge van het type Ocean A in
- **POWEREN** Zet op nul om de computer uit te schakelen op ondersteunde systemen. ALLEEN-SCHRIJVEN.
- **PREFETCH** Schakelt de prefetchlogica voor uitgebreid RAM in
- **SELSDRAM** Selecteert SDRAM in plaats van HyperRAM voor Attic RAM waar beschikbaar
- **SLIEN** Schakelt interrupts in stijl 6502 (langzaam, 7 cycli) in
- **SLOWCACHE** Schakelt langzamere regelcaching voor Attic RAM in
- **SLOWCACHEADV** Schakelt de geavanceerde langzamere regelcaching voor Attic RAM in
- **SLOWSDRAM** Selecteert de langzame (81MHz) SDRAM-klok

MEGA65 CPU-VERSNELLINGSREGISTERS VOOR WISKUNDE

Elke MEGA65 bevat een gecombineerde 32-bit-hardwarevermenigvuldiger en -deler. Dit apparaat neemt twee 32-bit-invoeren, **MULTINA** en **MULTINB**, en berekent tegelijkertijd:

- het 64-bit-product **MULTOUT** van **MULTINA** en **MULTINB**
- het 32-bit-gehele deel **DIVOUT**(4-7) van **MULTINA** gedeeld door **MULTINB**
- het 32-bit-breukdeel **DIVOUT**(0-3) van **MULTINA** gedeeld door **MULTINB**

De uitvoer wordt altijd op basis van de invoer bijgewerkt, zodat er geen speciale actie nodig is bij het wijzigen van de invoer. De vermenigvuldiger heeft 1 cyclus nodig om te berekenen, zodat het bijgewerkte resultaat onmiddellijk beschikbaar is (een **MULBUSY**-bit is gedefinieerd, maar deze zal momenteel niet worden gezet). De hardwaredeler kan echter tot 20 cycli kosten, afhankelijk van de specifieke invoer. De programmeur moet de **DIVBUSY**-bit controleren indien de deler nog bezig is met berekenen:

```
loop:   BIT $D70F   ; brengt de DIVBUSY-bit over naar de N-vlag
        BMI loop    ; zolang deze is gezet, moet worden gewacht
```

De MEGA65 zal ook een programmeerbare rekenunit bevatten, die helpt bij het versnellen van de berekening van vastekommatformules. Deze is momenteel uitgeschakeld en zal verder worden gedocumenteerd zodra deze beschikbaar komt (adressen $D780 - $D7E3).

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D70F	55055	DIVBUSY	MULBUSY	-					
D768	55144	DIVOUT							
D769	55145	DIVOUT							
D76A	55146	DIVOUT							
D76B	55147	DIVOUT							
D76C	55148	DIVOUT							
D76D	55149	DIVOUT							
D76E	55150	DIVOUT							
D76F	55151	DIVOUT							
D770	55152	MULTINA							
D771	55153	MULTINA							
D772	55154	MULTINA							
D773	55155	MULTINA							
D774	55156	MULTINB							
D775	55157	MULTINB							
D776	55158	MULTINB							
D777	55159	MULTINB							
D778	55160	MULTOUT							
D779	55161	MULTOUT							
D77A	55162	MULTOUT							
D77B	55163	MULTOUT							
D77C	55164	MULTOUT							
D77D	55165	MULTOUT							
D77E	55166	MULTOUT							
D77F	55167	MULTOUT							
D780	55168	MATHIN0							
D781	55169	MATHIN0							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D782	55170	MATHIN0							
D783	55171	MATHIN0							
D784	55172	MATHIN1							
D785	55173	MATHIN1							
D786	55174	MATHIN1							
D787	55175	MATHIN1							
D788	55176	MATHIN2							
D789	55177	MATHIN2							
D78A	55178	MATHIN2							
D78B	55179	MATHIN2							
D78C	55180	MATHIN3							
D78D	55181	MATHIN3							
D78E	55182	MATHIN3							
D78F	55183	MATHIN3							
D790	55184	MATHIN4							
D791	55185	MATHIN4							
D792	55186	MATHIN4							
D793	55187	MATHIN4							
D794	55188	MATHIN5							
D795	55189	MATHIN5							
D796	55190	MATHIN5							
D797	55191	MATHIN5							
D798	55192	MATHIN6							
D799	55193	MATHIN6							
D79A	55194	MATHIN6							
D79B	55195	MATHIN6							
D79C	55196	MATHIN7							
D79D	55197	MATHIN7							
D79E	55198	MATHIN7							
D79F	55199	MATHIN7							
D7A0	55200	MATHIN8							
D7A1	55201	MATHIN8							
D7A2	55202	MATHIN8							
D7A3	55203	MATHIN8							
D7A4	55204	MATHIN9							
D7A5	55205	MATHIN9							
D7A6	55206	MATHIN9							
D7A7	55207	MATHIN9							
D7A8	55208	MATHINA							
D7A9	55209	MATHINA							
D7AA	55210	MATHINA							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D7AB	55211	MATHINA							
D7AC	55212	MATHINB							
D7AD	55213	MATHINB							
D7AE	55214	MATHINB							
D7AF	55215	MATHINB							
D7B0	55216	MATHINC							
D7B1	55217	MATHINC							
D7B2	55218	MATHINC							
D7B3	55219	MATHINC							
D7B4	55220	MATHIND							
D7B5	55221	MATHIND							
D7B6	55222	MATHIND							
D7B7	55223	MATHIND							
D7B8	55224	MATHINE							
D7B9	55225	MATHINE							
D7BA	55226	MATHINE							
D7BB	55227	MATHINE							
D7BC	55228	MATHINF							
D7BD	55229	MATHINF							
D7BE	55230	MATHINF							
D7BF	55231	MATHINF							
D7C0	55232	UNIT0INB					UNIT0INA		
D7C1	55233	UNIT1INB					UNIT1INA		
D7C2	55234	UNIT2INB					UNIT2INA		
D7C3	55235	UNIT3INB					UNIT3INA		
D7C4	55236	UNIT4INB					UNIT4INA		
D7C5	55237	UNIT5INB					UNIT5INA		
D7C6	55238	UNIT6INB					UNIT6INA		
D7C7	55239	UNIT7INB					UNIT7INA		
D7C8	55240	UNIT8INB					UNIT8INA		
D7C9	55241	UNIT9INB					UNIT9INA		
D7CA	55242	UNITAINB					UNITAINA		
D7CB	55243	UNITBINB					UNITBINA		
D7CC	55244	UNITCINB					UNITCINA		
D7CD	55245	UNITDINB					UNITDINA		
D7CE	55246	UNITEINB					UNITEINA		
D7CF	55247	UNITFINB					UNITFINA		
D7D0	55248	U0LATCH	U0MLADD	U0HIOUT	U0–LOWOUT		UNIT0OUT		
D7D1	55249	U1LATCH	U1MLADD	U1HIOUT	U1–LOWOUT		UNIT1OUT		

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D7D2	55250	U2LATCH	U2MLADD	U2HIOUT	U2-LOWOUT	UNIT2OUT			
D7D3	55251	U3LATCH	U3MLADD	U3HIOUT	U3-LOWOUT	UNIT3OUT			
D7D4	55252	U4LATCH	U4MLADD	U4HIOUT	U4-LOWOUT	UNIT4OUT			
D7D5	55253	U5LATCH	U5MLADD	U5HIOUT	U5-LOWOUT	UNIT5OUT			
D7D6	55254	U6LATCH	U6MLADD	U6HIOUT	U6-LOWOUT	UNIT6OUT			
D7D7	55255	U7LATCH	U7MLADD	U7HIOUT	U7-LOWOUT	UNIT7OUT			
D7D8	55256	U8LATCH	U8BSADD	U8HIOUT	U8-LOWOUT	UNIT8OUT			
D7D9	55257	U9LATCH	U9BSADD	U9HIOUT	U9-LOWOUT	UNIT9OUT			
D7DA	55258	UALATCH	UABSADD	UAHIOUT	UA-LOWOUT	UNITAOUT			
D7DB	55259	UBLATCH	UBBSADD	UBHIOUT	UB-LOWOUT	UNITBOUT			
D7DC	55260	UCLATCH	UCDVADD	UCHIOUT	UC-LOWOUT	UNITCOUT			
D7DD	55261	UDLATCH	UDDVADD	UDHIOUT	UD-LOWOUT	UNITDOUT			
D7DE	55262	UELATCH	UEDVADD	UEHIOUT	UE-LOWOUT	UNITEOUT			
D7DF	55263	UFLATCH	UFDVADD	UFHIOUT	UF-LOWOUT	UNITFOUT			
D7E0	55264	LATCHINT							
D7E1	55265	-						CALCEN	WREN
D7E2	55266	RESERVED							
D7E3	55267	RESERVED							

- **CALCEN** Maakt het vastleggen van uitvoerwaarden van rekenunits in de wiskunderegisters mogelijk (uitwissen pauzeert effectief iteratieve formules)
- **DIVBUSY** Gezet indien de hardwaredeler bezig is
- **DIVOUT** 64-bit-uitvoer van MULTINA ÷ MULTINB
- **LATCHINT** Vastleginterval voor vastgelegde uitvoer (in CPU-cycli)
- **MATHINX** 32-bit-invoer X van de rekenunit
- **MULBUSY** Gezet indien de hardwarevermenigvuldiger bezig is
- **MULTINA** Vermenigvuldiger-invoer A / teller van de deler (32 bit)
- **MULTINB** Vermenigvuldiger-invoer B / noemer van de deler (32 bit)
- **MULTOUT** 64-bit-uitvoer van MULTINA × MULTINB

- **RESERVED** Gereserveerd
- **UNITXINA** Selecteert welke van de 16 32-bit-wiskunderegisters invoer A is voor rekenfunctieunit X.
- **UNITXINB** Selecteert welke van de 16 32-bit-wiskunderegisters invoer B is voor rekenfunctieunit X.
- **UNITXOUT** Selecteert welke van de 16 32-bit-wiskunderegisters de uitvoer van rekenfunctieunit X ontvangt
- **UXBSADD** Indien gezet, treedt rekenfunctieunit Y op als een 32-bit-opteller in plaats van een 32-bit-barrelshifter.
- **UXDVADD** Indien gezet, treedt rekenfunctieunit X op als een 32-bit-opteller in plaats van een 32-bit-deler.
- **UXHIOUT** Indien gezet, wordt de hoge helft van de uitvoer van rekenfunctieunit X naar het wiskunderegister UNITXOUT geschreven.
- **UXLATCH** Indien gezet, wordt de uitvoer van rekenfunctieunit X vastgelegd.
- **UXLOWOUT** Indien gezet, wordt de lage helft van de uitvoer van rekenfunctieunit X naar het wiskunderegister UNITXOUT geschreven.
- **UXMLADD** Indien gezet, treedt rekenfunctieunit X op als een 32-bit-opteller in plaats van een 32-bit-vermenigvuldiger.
- **WREN** Maakt het instellen van de wiskunderegisters mogelijk (moet normaal gezet zijn)

MEGA65-HYPERVISORMODUS

Reset

Bij inschakeling of reset start de MEGA65 in de hypervisormodus op en verwacht een programma in het 16KB-hypervisorgeheugen, en begint het uitvoeren van instructies op adres $8100. Normaal bevindt zich op dit adres een JMP-instructie die naar een routine voor reset springt. Dat wil zeggen, de 45GS02 gebruikt niet de normale resetvector van de 6502. De functie daarvan wordt geëmuleerd door het Hyppo-hypervisorprogramma, dat het adres ophaalt uit de resetvector van de 6502 in het geladen clientbesturingssysteem wanneer het de hypervisormodus verlaat.

Het hypervisorgeheugen wordt bij reset automatisch op $8000 - $BFFF gemapt. Dit speciale geheugen kan niet worden gemapt of op enige andere manier worden benaderd, behalve wanneer de processor zich in de hypervisormodus bevindt. Het kan echter altijd worden benaderd vanuit de seriële monitor-/debuggerinterface via zijn 28-bit-adres, $FFF8000 - $FFFBFFF. Dit beschermt het tegen onbedoelde of kwaadwillende toegang vanuit een gastbesturingssysteem.

De hypervisormodus in- en uitstappen

Het betreden van de hypervisor vindt plaats wanneer een van de volgende gebeurtenissen optreedt:

- **Inschakelen** Wanneer de MEGA65 voor het eerst wordt ingeschakeld.
- **Reset** Wanneer de resetlijn laag wordt gemaakt of een reset door een watchdog wordt geactiveerd.
- **SYSCALL-register benaderd** De registers $D640 - $D67F in de MEGA65 I/O-modus activeren SYSCALLs wanneer ze worden benaderd. Dit is bedoeld als het mechanisme waarmee een clientbesturingssysteem of -proces de aandacht van de hypervisor of het besturingssysteem verzoekt.
- **Paginafout** Op MEGA65's met virtueel geheugen veroorzaakt een paginafout een trap naar de hypervisormodus.
- **Bepaalde toetsenbordgebeurtenissen** Het langer dan 0.5 seconden ingedrukt houden van RESTORE, of de combinatie van de  - en TAB-toets, activeert een trap naar de hypervisor. Doorgaans wordt het eerste gebruikt om het Freeze-menu te starten en het tweede om de weergave van de debuginterface te wisselen.
- **Benaderen van virtueel gemaakte I/O-apparaten** Indien bijvoorbeeld de F011 (interne 3.5"-schijfstationcontroller) is gevirtualiseerd, zal het pogingen om sectoren met dit apparaat te lezen of schrijven activeren tot trappen naar de hypervisor.
- **Uitvoeren van een instructie die de CPU zou blokkeren** Een aantal niet-gedocumenteerde opcodes op de 6502 zal de CPU blokkeren. Op de MEGA65 zal de computer in plaats daarvan een trap naar de hypervisor activeren. Dit kan worden gebruikt om alternatieve instructiegedragingen te implementeren, of eenvoudigweg om de gebruiker te melden dat er iets fout is gegaan.
- **Bepaalde speciale gebeurtenissen** Sommige apparaten kunnen hypervisor-niveau-onderbrekingen genereren. Deze worden geïmplementeerd als trappen naar de hypervisor.

De 45GS02 verwerkt al deze gebeurtenissen op een vergelijkbare manier intern:

1. Het SYSCALL- of trap-adres wordt berekend, op basis van de gebeurtenis.
2. De inhoud van alle CPU-registers wordt opgeslagen in de virtualisatiebesturingsregisters.
3. De geheugenindeling van de hypervisormodus wordt geactiveerd, de decimale vlag van de CPU en de special purpose-registers worden allemaal op geschikte waarden gezet. De inhoud van de A-, X-, Y- en Z-registers en de meeste andere CPU-vlaggen blijven behouden, zodat die benaderd kunnen worden vanuit de SYSCALL/trap-handlerroutine van de hypervisor, zonder ze te moeten laden, wat een paar cycli per aanroep bespaart.

4. De hypervisormodus-vlag wordt gezet en de programmateller (PC) wordt op het berekende adres gezet.

Al het bovenstaande gebeurt in één CPU-cyclus, d.w.z. in 25 nanoseconden. Terugkeren van een SYSCALL of trap bestaat eenvoudigweg uit het schrijven naar $D67F, wat 125 nanoseconden vereist, voor een totale overhead van 150 nanoseconden. Dit geeft de SYSCALL-prestaties van de MEGA65 die wedijveren met – zelfs sneller zijn dan – zelfs de snelste moderne computers, waar de systeemaanroeplatentie doorgaans honderden tot tienduizenden cycli bedraagt [2].

Hypervisor-geheugenindeling

Het hypervisorgeheugen heeft een grootte van 16KB. De eerste 512 bytes zijn gereserveerd voor SYSCALL- en systeem-trap-ingangspunten, met vier bytes voor elk. Het reset-ingangspunt is bijvoorbeeld $8100 - $8100 + 3 = $8100 - $8103. Dit laat 4 bytes toe voor een instructie, doorgaans een JMP-instructie, gevolgd door een NOP om deze tot 4 bytes op te vullen.

De volledige lijst van SYSCALLs en traps is:

HEX	DEC	Naam	Omschrijving
8000	32768	SYSCALL00	SYSCALL 0 ingangspunt
8004	32772	SYSCALL01	SYSCALL 1 ingangspunt
8008	32776	SYSCALL02	SYSCALL 2 ingangspunt
800C	32780	SYSCALL03	SYSCALL 3 ingangspunt
8010	32784	SYSCALL04	SYSCALL 4 ingangspunt
8014	32788	SYSCALL05	SYSCALL 5 ingangspunt
8018	32792	SYSCALL06	SYSCALL 6 ingangspunt
801C	32796	SYSCALL07	SYSCALL 7 ingangspunt
8020	32800	SYSCALL08	SYSCALL 8 ingangspunt
8024	32804	SYSCALL09	SYSCALL 9 ingangspunt
8028	32808	SYSCALL0A	SYSCALL 10 ingangspunt
802C	32812	SYSCALL0B	SYSCALL 11 ingangspunt
8030	32816	SYSCALL0C	SYSCALL 12 ingangspunt
8034	32820	SYSCALL0D	SYSCALL 13 ingangspunt
8038	32824	SYSCALL0E	SYSCALL 14 ingangspunt
803C	32828	SYSCALL0F	SYSCALL 15 ingangspunt
8040	32832	SYSCALL10	SYSCALL 16 ingangspunt
8044	32836	SECURENTR	Ingangspunt van de trap naar de beveiligde container
8048	32840	SECUREXIT	Verlaten van de beveiligde container trap-ingangspunt.
804C	32844	SYSCALL13	SYSCALL 19 ingangspunt
8050	32848	SYSCALL14	SYSCALL 20 ingangspunt
8054	32852	SYSCALL15	SYSCALL 21 ingangspunt

Vervolgd ...

...Vervolgd

HEX	DEC	Naam	Omschrijving
8058	32856	SYSCALL16	SYSCALL 22 ingangspunt
805C	32860	SYSCALL17	SYSCALL 23 ingangspunt
8060	32864	SYSCALL18	SYSCALL 24 ingangspunt
8064	32868	SYSCALL19	SYSCALL 25 ingangspunt
8068	32872	SYSCALL1A	SYSCALL 26 ingangspunt
806C	32876	SYSCALL1B	SYSCALL 27 ingangspunt
8070	32880	SYSCALL1C	SYSCALL 28 ingangspunt
8074	32884	SYSCALL1D	SYSCALL 29 ingangspunt
8078	32888	SYSCALL1E	SYSCALL 30 ingangspunt
807C	32892	SYSCALL1F	SYSCALL 31 ingangspunt
8080	32896	SYSCALL20	SYSCALL 32 ingangspunt
8084	32900	SYSCALL21	SYSCALL 33 ingangspunt
8088	32904	SYSCALL22	SYSCALL 34 ingangspunt
808C	32908	SYSCALL23	SYSCALL 35 ingangspunt
8090	32912	SYSCALL24	SYSCALL 36 ingangspunt
8094	32916	SYSCALL25	SYSCALL 37 ingangspunt
8098	32920	SYSCALL26	SYSCALL 38 ingangspunt
809C	32924	SYSCALL27	SYSCALL 39 ingangspunt
80A0	32928	SYSCALL28	SYSCALL 40 ingangspunt
80A4	32932	SYSCALL29	SYSCALL 41 ingangspunt
80A8	32936	SYSCALL2A	SYSCALL 42 ingangspunt
80AC	32940	SYSCALL2B	SYSCALL 43 ingangspunt
80B0	32944	SYSCALL2C	SYSCALL 44 ingangspunt
80B4	32948	SYSCALL2D	SYSCALL 45 ingangspunt
80B8	32952	SYSCALL2E	SYSCALL 46 ingangspunt
80BC	32956	SYSCALL2F	SYSCALL 47 ingangspunt
80C0	32960	SYSCALL30	SYSCALL 48 ingangspunt
80C4	32964	SYSCALL31	SYSCALL 49 ingangspunt
80C8	32968	SYSCALL32	SYSCALL 50 ingangspunt
80CC	32972	SYSCALL33	SYSCALL 51 ingangspunt
80D0	32976	SYSCALL34	SYSCALL 52 ingangspunt
80D4	32980	SYSCALL35	SYSCALL 53 ingangspunt
80D8	32984	SYSCALL36	SYSCALL 54 ingangspunt
80DC	32988	SYSCALL37	SYSCALL 55 ingangspunt
80E0	32992	SYSCALL38	SYSCALL 56 ingangspunt
80E4	32996	SYSCALL39	SYSCALL 57 ingangspunt
80E8	33000	SYSCALL3A	SYSCALL 58 ingangspunt
80EC	33004	SYSCALL3B	SYSCALL 59 ingangspunt
80F0	33008	SYSCALL3C	SYSCALL 60 ingangspunt
80F4	33012	SYSCALL3D	SYSCALL 61 ingangspunt
80F8	33016	SYSCALL3E	SYSCALL 62 ingangspunt

Vervolgd ...

...Vervolgd

HEX	DEC	Naam	Omschrijving
80FC	33020	SYSCALL3F	SYSCALL 63 ingangspunt
8100	33024	RESET	Inschakel-/reset-ingangspunt
8104	33028	PAGFAULT	Paginafout-ingangspunt (momenteel niet gebruikt)
8108	33032	RESTORKEY	Trap-ingangspunt voor lang indrukken van de RESTORE-toets
810C	33036	ALTTABKEY	ALT+TAB-trap-ingangspunt
8110	33040	VF011RD	Trap-ingangspunt voor virtueel gelezen F011-schijf
8114	33044	VF011WR	Trap-ingangspunt voor virtueel geschreven F011-schijf
8118	33048	BREAKPT	CPU-breukpunt aangetroffen
811C - 81FB	33048 - 33275	RESERVED	Gereserveerd trap-ingangspunt
81FC	33276	CPUKIL	KIL-instructie in 6502-modus trap-ingangspunt

Het resterende deel van het 16KB-hypervisorgeheugen staat ter beschikking van de programmeur, maar doorgaans worden de laatste 512 bytes gebruikt voor de stack en de zero page, wat leidt tot de volgende algemene geheugenindeling:

HEX	DEC	Omschrijving
8000 - 81FF	32768 - 33279	SYSCALL- en trap-ingangspunten
8200 - BDFF	33280 - 48639	Beschikbaar voor hypervisor- of besturingssysteem-programma
8E00 - BEFF	48640 - 48895	Processorstack voor hypervisor of besturingssysteem
8F00 - BFFF	48896 - 49151	Zero page-opslag van de processor voor hypervisor of besturingssysteem

De stack wordt gebruikt om het retouradres van functieaanroepen op te slaan. De zero page-opslag wordt doorgaans gebruikt om variabelen en andere gegevens voor korte termijn op te slaan, zoals gebruikelijk op de 6502.

Hypervisor-virtualisatieregisters

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D640	54848	REGA							
D641	54849	REGX							
D643	54851	REGZ							
D644	54852	REGB							
D645	54853	SPL							
D646	54854	SPH							
D647	54855	PFLAGS							
D648	54856	PCL							
D649	54857	PCH							
D64A	54858	MAPLO0							
D64B	54859	MAPLO1							
D64C	54860	MAPHI0							
D64D	54861	MAPHI1							
D64E	54862	MAPLOMB							
D64F	54863	MAPHIMB							
D650	54864	PORT00							
D651	54865	PORT01							
D652	54866	–					EXSID	VICMODE	
D653	54867	DMASRCMB							
D654	54868	DMADSTMB							
D655	54869	DMALADDR0							
D656	54870	DMALADDR1							
D657	54871	DMALADDR2							
D658	54872	DMALADDR3							
D659	54873	–						VFLOP1	VFLOP0
D65D	54877	VIRTPAGE0		–	VPG-ACTIVE	VPGDIRTY			
D65E	54878	VIRTPAGE1							
D65F	54879	VIRTPAGE2							
D660	54880	VPG0LOG0							
D661	54881	VPG0LOG1							
D662	54882	VPG0PHY0							
D663	54883	VPG0PHY1							
D664	54884	VPG1LOG0							
D665	54885	VPG1LOG1							
D666	54886	VPG1PHY0							
D667	54887	VPG1PHY0							
D668	54888	VPG2LOG0							
D669	54889	VPG2LOG1							
D66A	54890	VPG2PHY0							
D66B	54891	VPG2PHY0							
D66C	54892	VPG3LOG0							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D66D	54893	VPG3LOG1							
D66E	54894	VPG3PHY0							
D66F	54895	VPG3PHY0							
D670	54896	GEORAMBASE							
D671	54897	GEORAMMASK							
D672	54898	-	MATRIXEN	-					
D67C	54908	UARTDATA							
D67D	54909	WATCHDOG							
D67E	54910	HICKED							
D67F	54911	ENTEREXIT							

- **ASCFAST** Hypervisor: schakelt de ASC/DIN-CAPS-LOCK-toets in om de CPU-vertraging in de C64/C128/C65-modus in of uit te schakelen
- **CPUFAST** Hypervisor: dwingt de CPU naar 40.5MHz af voor de userland (de userland kan dit overschrijven via POKE0)
- **DMADSTMB** Hypervisor: DMAgic-bestemmings-MB
- **DMALADDR0** Hypervisor: bits 0-7 van het DMAGic-lijstadres
- **DMALADDR1** Hypervisor: bits 15-8 van het DMAGic-lijstadres
- **DMALADDR2** Hypervisor: bits 23-16 van het DMAGic-lijstadres
- **DMALADDR3** Hypervisor: bits 27-24 van het DMAGic-lijstadres
- **DMASRCMB** Hypervisor: DMAgic-bron-MB
- **ENTEREXIT** Schrijven activeert het terugkeren vanuit de hypervisor
- **EXSID** 0=interne SID's gebruiken, 1=externe(1) SID's gebruiken
- **F4502** Hypervisor: dwingt de CPU naar de 4502-persoonlijkheid af, zelfs in de C64-I/O-modus.
- **GEORAMBASE** Hypervisor: basisadres van GeoRAM (x MB)
- **GEORAMMASK** Hypervisor: adresmasker van GeoRAM (toegepast op het blok-register van GeoRAM)
- **HICKED** Hypervisor: al-geüpgraded-bit (schrijven zet dit permanent)
- **JMP32EN** Hypervisor: schakelt 32-bit JMP/JSR en dergelijke in
- **MAPHI0** Hypervisor: opslag van het MAPHI-register (hoge bits)
- **MAPHI1** Hypervisor: opslag van het MAPHI-register (lage bits)
- **MAPHIMB** Hypervisor: opslag van het megabyte-aantal van het MAPHI-register
- **MAPLO0** Hypervisor: opslag van het MAPLO-register (hoge bits)

- **MAPLO1** Hypervisor: opslag van het MAPLO-register (lage bits)
- **MAPLOMB** Hypervisor: opslag van het megabyte-aantal van het MAPLO-register
- **MATRIXEN** Schakelt de samengestelde Matrix-modus in en schakelt de UART-toegang tot de seriële monitor uit.
- **PCH** Hypervisor: opslag van het hoge PC-register
- **PCL** Hypervisor: opslag van het lage PC-register
- **PFLAGS** Hypervisor: opslag van het P-register
- **PIRQ** Hypervisor: vlag die aangeeft of er een IRQ in behandeling is bij het verlaten van de hypervisor / zet 1 om IRQ/NMI-uitstel voor 1.024 cycli af te dwingen bij het verlaten van de hypervisor.
- **PNMI** Hypervisor: vlag die aangeeft of er een NMI in behandeling is bij het verlaten van de hypervisor.
- **PORT00** Hypervisor: waarde van CPU-port $00
- **PORT01** Hypervisor: waarde van CPU-port $01
- **REGA** Hypervisor: opslag van het A-register
- **REGB** Hypervisor: opslag van het B-register
- **REGX** Hypervisor: opslag van het X-register
- **REGZ** Hypervisor: opslag van het Z-register
- **ROMPROT** Hypervisor: schrijfbeveiliging van de C65-ROM $20000-$3FFFF
- **RSVD** GERESERVEERD
- **SPH** Hypervisor: opslag van het SPH-register
- **SPL** Hypervisor: opslag van het SPL-register
- **UARTDATA** (schrijven) Hypervisor: schrijft seriële uitvoer naar de UART-monitor
- **VFLOP0** 1=SD-/Floppy0-toegang virtualiseren (doorgaans voor toegang via de seriële debuggerinterface)
- **VFLOP1** 1=SD-/Floppy1-toegang virtualiseren (doorgaans voor toegang via de seriële debuggerinterface)
- **VICMODE** VIC-II/VIC-III/VIC-IV-modusselectie
- **VIRTPAGE0** Hypervisor: huidig virtueel paginanummer (bits 0 - 1)
- **VIRTPAGE1** Hypervisor: huidig virtueel paginanummer (bits 2 - 9)
- **VIRTPAGE2** Hypervisor: huidig virtueel paginanummer (bits 10 - 17)
- **VPGACTIVE** Hypervisor: pagina actief

- **VPGDIRTY** Hypervisor: pagina gewijzigd
- **VPGXLOG0** Hypervisor: logische pagina van virtueel geheugenpagina X (bits 0 - 7)
- **VPGXLOG1** Hypervisor: logische pagina van virtueel geheugenpagina X (bits 8 - 15)
- **VPGXPHY0** Hypervisor: fysieke pagina van virtueel geheugenpagina X (bits 0 - 7)
- **VPGXPHY1** Hypervisor: fysieke pagina van virtueel geheugenpagina X (bits 8 - 15)
- **WATCHDOG** Hypervisor: watchdogregister: het schrijven van een willekeurige waarde wist de watchdog

Programmeren in de hypervisormodus

De eenvoudigste manier om een programma voor de hypervisormodus op de MEGA65 te schrijven, is het gebruik van KickC, een speciale versie van C die is gemaakt voor het schrijven van programma's voor 6502-klasse processors. De volgende voorbeeldprogramma's zijn afkomstig uit de meegeleverde voorbeelden van KickC. KickC produceert zeer efficiënte code en ondersteunt de hypervisormodus van de MEGA65 op eenvoudige wijze via een linkerdefinitiebestand met de volgende inhoud:

```
.file [name="%O.bin", type="bin", segments="XMega65Bin"]
.segmentdef XMega65Bin [segments="Syscall, Code, Data, Stack, Zeropage"]
.segmentdef Syscall [start=$8000, max=$81ff]
.segmentdef Code [start=$8200, min=$8200, max=$bdff]
.segmentdef Data [startAfter="Code", min=$8200, max=$bdff]
.segmentdef Stack [min=$be00, max=$beff, fill]
.segmentdef Zeropage [min=$bf00, max=$bfff, fill]
```

Dit bestand draagt de assembler van KickC op om een bestand van 16 KB te maken met het ingangspuntgebied van 512 bytes voor SYSCALL- en trap-verwijzingen aan het begin, gevolgd door code- en data-gebieden, en vervolgens de stack- en zeropage-gebieden. Het dwingt de grootte en locatie van deze velden af en zal tijdens de compilatie een foutmelding geven wanneer iets te groot is om te passen.

Wanneer dit bestand aanwezig is, kunt u een KickC-bronbestand aanmaken dat gegevensstructuren voor de SYSCALL/trap-tabel bevat:

```
// XMega65 KERNAL-ontwikkelingssjabloon
// Elke functie van het KERNAL is een functie zonder argumenten
// De functies worden in de SYSCALLS-tabel geplaatst, omringd door JMP en NOP

import "string"

// Gebruik een linkerdefinitiebestand (plaats de vorige listing in dat bestand)
#pragma link("mega65hyper.ld")

// Enkele definities van adressen en speciale waarden die dit programma gebruikt
const char* RASTER = 0xd012;
const char* VIC+MEMORY = 0xd018;
const char* SCREEN = 0x0400;
const char* BGCOL = 0xd021;
const char* COLS = 0xd800;
const char BLACK = 0;
const char WHITE = 1;

// Enige tekst om weer te geven
char[] MESSAGE = "hello world!";
```

```
void main() {
    // Initialiseer het schermgeheugen en selecteer het juiste lettertype
    *VIC+MEMORY = 0x14;
    // Vul het scherm met spaties
    memset(SCREEN, ' ', 40*25);
    // Stel de kleur van elk teken op het scherm in op wit
    memset(COLS, WHITE, 40*25);
    // Toon de "hello world!"-melding
    char* sc = SCREEN+40;  // Toon het   n regel lager op het scherm
    char* msg = MESSAGE; // De boodschap om weer te geven
    // Een eenvoudige kopieerroutine om de reeks te kopi ren
    while(*msg) {
        *sc++ = *msg++;
    }
    // Laat voor altijd twee witte lijnen als rasterbalken tonen
    while(true) {
        if(*RASTER==54 || *RASTER==66) {
            *BGCOL = WHITE;
        } else {
            *BGCOL = BLACK;
        }
    }
}

// Hier zijn een paar voorbeeld-SYSCALL-handlers die slechts een teken op het scherm
void syscall1() {
    *(SCREEN+79) = ')';
}

void syscall2() {
    *(SCREEN+78) = '(';
}

// Nu selecteren wij het SYSCALL-segment om de SYSCALL/trap-ingangspunttabel te beva
#pragma data+seg(Syscall)

// De structuur van elk ingangspunt is JMP (handleradres) + NOP.
// Wij hebben een char (xjmp) om de opcode voor de JMP-instructie te bevatten,
// en plaatsen daarna het adres van de SYSCALL/trap-handler in de volgende
// twee posities als een aanwijzer, en eindigen met de opcode van de NOP-instructie.
```

```
struct SysCall {
    char xjmp;          // Bevat $4C, de opcode van JMP $nnnn
    void()* syscall;    // Bevat het handleradres, zal het doel van de JMP zijn
    char xnop;          // Bevat $EA, de opcode van NOP
};

// Om niet steeds 0x4C en 0xEA te hoeven schrijven, defini ren wij ze als constanten
const char JMP = 0x4c;
const char NOP = 0xea;

// Nu kunnen wij een mooie tabel van maximaal 64 SYSCALL-handlers uitdrukken
// in een redelijk leesbaar en eenvoudig formaat.
// Elke regel is een instantie van de bovenstaande struct SysCall, met de JMP-
// opcodewaarde, het adres van de handler-routine en de NOP-opcodewaarde.
export struct SysCall[] SYSCALLS = {
    { JMP, &syscall1, NOP },
    { JMP, &syscall2, NOP }
    };

// In dit voorbeeld hadden wij slechts twee SYSCALLs gedefinieerd, dus in plaats van
// 62 regels te hebben, kunnen wij KickC vragen de TRAP-tabel te laten beginnen
// bij het eerstvolgende veelvoud van $100, dat wil zeggen op $8100.
export align(0x100) struct SysCall[] SYSCALL£+RESET = {
    { JMP, &main, NOP }
};
```

Wanneer u de eerste listing opslaat in een bestand met de naam mega65hyper.ld en de tweede in een bestand met de naam mega65hyper.kc, kunt u deze compileren met KickC met een commando als:

```
kickc -a mega65hyper
```

Vervolgens wordt een bestand met de naam mega65hyper.bin geproduceerd, dat u kunt uitproberen op uw MEGA65 of kunt uitvoeren in de XMega65-emulator met een commando als:

```
xmega65 -kickup mega65hyper.bin
```

BIJLAGE L

45GS02 & 6502-instructiesets

- Inleiding
- Stapelbewerkingen
- Adresseermethoden
- 6502-instructieset
- De instructieset van de 4510
- Samengestelde instructies van de 45GS02

INLEIDING

De 45GS02-CPU kan in de native-modus werken, waarbij hij compatibel is met de CSG 4510, en in de 6502-compatibiliteitsmodus, waarbij niet-gedocumenteerde 6502-instructies, ook wel illegale instructies genoemd, worden ondersteund voor compatibiliteit.

> **WAARSCHUWING:** Deze functie is onvolledig en niet getest. De meeste niet-gedocumenteerde 6502-opcodes werken niet correct wanneer de 6502-persoonlijkheid is ingeschakeld.

Bij 4510-compatibiliteit ondersteunt de 45GS02 ook een aantal uitbreidingen via *samengestelde instructies*. Dit werkt door de opcode van de gewenste instructie te laten voorafgaan door een of meer *prefix-bytes*, welke instructievolgordes representeren die normaal gesproken niet voorkomen. Twee **NEG**-instructies achter elkaar functioneren bijvoorbeeld als een prefix om de 45GS02 te vertellen dat de volgende instructie op 32 bits aan gegevens werkt, in plaats van de gebruikelijke 8 bits. Dit betekent dat een 45GS02-instructiestroom eenvoudig gedecodeerd of gedassembleerd kan worden, zonder dat speciale instructielengtevlaggen ingesteld hoeven te worden, zoals het geval is bij de 65816-familie van microprocessors. Het nadeel is een langere uitvoeringstijd, omdat de 45GS02 de prefix-bytes moet overslaan.

De rest van dit hoofdstuk introduceert de adresseermethoden, instructies, opcodes en instructietijding van de 45GS02, te beginnen met de 6502-compatibiliteitsmodus, waarna de 4510-compatibiliteitsmodus en de 45GS02-uitbreidingen worden behandeld.

STAPELBEWERKINGEN

De stapel (Engels: *stack*) is een gebied in het geheugen waar u gegevens op kunt plaatsen of de laatste gegevens van kunt ophalen (pop). Elke stapelbewerking die gegevens op de stapel plaatst (push) verlaagt de stapelaanwijzer (SP, *stack pointer*) en elke stapelbewerking die gegevens van de stapel afneemt (pop) verhoogt de SP. De stapel begint namelijk aan de bovenkant van het gebied en groeit naar onderen.

Wanneer u iets tegenkomt als

STAPEL ← WAARDE

betekent dat WAARDE op de STAPEL wordt geplaatst en de SP wordt verlaagd met het aantal bytes dat WAARDE groot is. Bij het plaatsen van meer dan één byte wordt eerst de meest significante byte (MSB) geplaatst, gevolgd door de minst significante byte (LSB). (**PHW** is een uitzondering: die plaatst de LSB eerst.)

In de tegenovergestelde richting geldt voor een bewerking als

MEMORY of REGISTER ← STAPEL

dat gegevens van de STAPEL worden opgehaald en de SP met elke verwijderde byte wordt verhoogd.

ADRESSEERMETHODEN

De 45GS02 ondersteunt 34 verschillende adresseermethoden, welke hieronder worden uitgelegd. Velen hiervan lijken sterk op elkaar en zijn variaties op de standaard 6502- of 65CE02-adresseermethoden, met als uitzondering dat ze ofwel 32-bits pointers accepteren, op 32 bits aan gegevens werken, of beide.

Impliciet

Bij deze methode zijn er geen operanden, omdat de exacte functie van de instructie impliciet door de instructie zelf wordt bepaald. De `INX`-instructie verhoogt bijvoorbeeld het X-register.

Accumulator

Bij deze methode is de Accumulator de operand. Dit wordt doorgaans gebruikt om de waarde van het accumulatorregister te verschuiven, te roteren of op een andere manier aan te passen. `INC A` verhoogt bijvoorbeeld de waarde in het accumulatorregister.

Het pseudoregister Q

Bij deze methode is het pseudoregister Q de operand. Dit wordt doorgaans gebruikt om de waarde van het pseudoregister Q te verschuiven, te roteren of op een andere manier aan te passen. `ASLQ` verschuift bijvoorbeeld de waarde in het pseudoregister Q één bit naar links.

Onthoud dat het pseudoregister Q eenvoudigweg de A-, X-, Y- en Z-registers zijn die samenwerken als een virtueel 32-bits register, waarbij A de minst significante bits bevat en Z de meest significante bits. Wanneer u Q aanpast, past u de daadwerkelijke registers aan, en op dezelfde manier verandert het aanpassen van een daadwerkelijk register het desbetreffende deel van het Q-register.

Er zijn enkele gevallen waarin het gebruik van een Q-modusinstructie nuttig kan zijn voor het bewerken van de vier daadwerkelijke registers, bijvoorbeeld om alle vier de registers snel te laden of op te slaan.

Immediate-modus

In deze modus is het argument van de instructie een waarde die rechtstreeks wordt gebruikt. Dit wordt aangegeven door de waarde te laten voorafgaan door een #-teken. De meeste assemblers staan toe dat waarden worden ingevoerd in decimaal,

of in hexadecimaal door de waarde te laten voorafgaan door een $-teken, of in binair door de waarde te laten voorafgaan door een %-teken. Om bijvoorbeeld het accumulatorregister op de waarde 5 te zetten, kunt u het volgende gebruiken:

```
LDA #5
```

Het immediate-argument wordt gecodeerd als een enkel byte dat op de instructie volgt. In het bovenstaande geval zou de instructiestroom $A9 bevatten, de opcode voor LDA in immediate-modus, gevolgd door $05, de immediate-operand.

Immediate Word-modus

In deze modus is het argument een 16-bits waarde die direct wordt gebruikt. Er is slechts één instructie die deze adresseermethode gebruikt: PHW. Om bijvoorbeeld het woord $1234 op de stapel te plaatsen, kunt u het volgende gebruiken:

```
PHW #$1234
```

De lage byte van de immediate-waarde volgt op de opcode van de instructie. De hoge byte van de immediate-waarde volgt daarop. In het bovenstaande voorbeeld zou de instructiestroom dus $F4 $34 $12 zijn.

Base-Page-modus

In deze modus is het argument een 8-bits adres. De bovenste 8 bits van het adres worden overgenomen uit het Base-Page-register. Op 6502-processors bestaat er geen Base-Page-register; in plaats daarvan worden de bovenste 8 bits altijd op nul gezet - vandaar de naam van deze modus op de 6502: Zero-Page. Op de 45GS02 is het mogelijk dit "Zero-Page" naar elke pagina in het 64KB-geheugenbeeld van de processor te verplaatsen door het Base-Page-register met behulp van de TAB-instructie in te stellen. De Base-Page-modus biedt snellere toegang tot een geheugengebied van 256 bytes en gebruikt daarvoor minder instructiebytes.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
LDA $12
```

zou de waarde lezen die is opgeslagen in locatie $12 in de Base-Page en deze in het accumulatorregister plaatsen. De instructiestroom hiervoor zou $85 $12 zijn.

Base-Page Quad-modus

Deze modus is identiek aan de Base-Page-modus, behalve dat deze een 32-bits woord leest dat begint op het opgegeven adres.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
LDQ $12
```

zou de waarden lezen die zijn opgeslagen in de locaties $12 - $15 in de Base-Page en deze in het pseudoregister Q plaatsen. De instructiestroom hiervoor zou $42 $42 $85 $12 zijn. Merk op dat dit hetzelfde is als bij de Base-Page-modus (Zero-Page), met als toevoeging de twee $42-prefixbytes. Opcode $42 is normaal gesproken NEG (het negatief maken van de waarde in het A-register). Indien deze twee keer achter elkaar wordt uitgevoerd, keert de A-waarde terug naar zijn oorspronkelijke waarde. De 45GS02-processor beschikt over speciale logica om deze reeks te herkennen, zodat hij weet dat de volgende instructie met behulp van het Q Pseudo Register moet worden uitgevoerd.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page.

Base-Page X-Indexed-modus

Deze modus is identiek aan de Base-Page-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het X-register bij op te tellen. In 6502-modus ligt het resultaat altijd in de Base-Page; dat wil zeggen dat eventuele carry als gevolg van de optelling van de lage byte naar de hoge byte van het adres wordt genegeerd. De codering van deze adresseermethode is identiek aan die van de Base-Page-modus.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
LDA $12,X
```

zou de waarde lezen die is opgeslagen in locatie ($12 + X) in de Base-Page en deze in het A-register plaatsen. De instructiestroom hiervoor zou $B5 $12 zijn.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page.

Base-Page Quad X-Indexed-modus

Deze modus is identiek aan de Base-Page Quad-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het X-register bij op te tellen. In 6502-modus ligt het resultaat altijd in de Base-Page; dat wil zeggen dat eventuele carry als gevolg van de optelling van de lage byte naar de hoge byte van het adres wordt genegeerd. De codering van deze adresseermethode is identiek aan die van de Base-Page Quad-modus.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
DEQ $12,X
```

zou het 32-bits woord dat is opgeslagen in ($12 + X) tot en met ($15 + X) in de Base-Page met één verhogen en het in het X-register plaatsen. De instructiestroom hiervoor zou $42 $42 $D6 $12 zijn.

Merk op dat LDQ niet beschikbaar is in deze adresseermethode.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page.
Zie de opmerking op pagina L-5 voor meer informatie over Quad-modusinstructies.

Base-Page Y-Indexed-modus

Deze modus is identiek aan de Base-Page-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het Y-register bij op te tellen. In 6502-modus ligt het resultaat altijd in de Base-Page; dat wil zeggen dat eventuele carry als gevolg van de optelling van de lage byte naar de hoge byte van het adres wordt genegeerd. De codering van deze adresseermethode is identiek aan die van de Base-Page-modus.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
LDX $12,Y
```

zou de waarde lezen die is opgeslagen in locatie ($12 + Y) in de Base-Page en deze in het X-register plaatsen. De instructiestroom hiervoor zou $B6 $12 zijn.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page.

Absolute modus

In deze modus is het argument een 16-bits adres. De lage 8 bits van het adres worden genomen uit het byte dat direct na de opcode van de instructie komt. De hoge 8 bits worden genomen uit het byte dat daarop volgt. Bijvoorbeeld de instructie

```
LDA $1234
```

zou de geheugenlocatie $1234 lezen en de gelezen waarde in het accumulatorregister plaatsen. Dit zou worden gecodeerd als $AD $34 $12.

Absolute Quad-modus

In deze modus is het argument een 16-bits adres. De lage 8 bits van het adres worden genomen uit het byte dat direct na de opcode van de instructie komt. De hoge 8 bits worden genomen uit het byte dat daarop volgt. Bijvoorbeeld de instructie

```
LDQ $1234
```

zou de geheugenlocaties $1234 - $1237 lezen en de gelezen waarden in het Q Pseudo Register plaatsen. Dit zou worden gecodeerd als $42 $42 $AD $34 $12.

Zie de opmerking op pagina L-5 voor meer informatie over Quad-modusinstructies.

Absolute X-Indexed-modus

Deze modus is identiek aan de Absolute-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het X-register bij op te tellen. Indien de indexering het adres een paginagrens doet overschrijden, dat wil zeggen indien de hoge byte van het adres verandert, kan dit een extra cyclus met zich meebrengen, afhankelijk van de processormodus en de snelheidsinstelling. De codering van deze adresseermethode is identiek aan die van de Absolute-modus. Bijvoorbeeld de instructie

```
LDA $1234,X
```

zou de geheugenlocatie ($1234 + X) lezen en de daar gelezen waarde in het A-register plaatsen. Dit zou worden gecodeerd als $BD $34 $12.

Absolute Quad X-Indexed-modus

Deze modus is identiek aan de Absolute Quad-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het X-register bij op te tellen. Indien de indexering het adres een paginagrens doet overschrijden, dat wil zeggen indien de hoge byte van het adres verandert, kan dit een extra cyclus met zich meebrengen, afhankelijk van de processormodus en de snelheidsinstelling. De codering van deze adresseermethode is identiek aan die van de Absolute Quad-modus.

Bijvoorbeeld de instructie

```
ROLQ $1234,X
```

zou de 32-bits waarde op de geheugenlocaties ($1234+X) - ($1237+X) naar links roteren en het resultaat terugschrijven naar dezezelfde geheugenlocaties. Dit zou worden gecodeerd als $42 $42 $3E $34 $12.

Zie de opmerking op pagina L-5 voor meer informatie over Quad-modusinstructies.

Absolute Y-Indexed-modus

Deze modus is identiek aan de Absolute-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het Y-register bij op te tellen. Indien de indexering het adres een paginagrens doet overschrijden, dat wil zeggen indien de hoge byte van het adres verandert, kan dit een extra cyclus met zich meebrengen, afhankelijk van de processormodus en de snelheidsinstelling. De codering van deze adresseermethode is identiek aan die van de Absolute-modus. Bijvoorbeeld de instructie

```
LDA $1234,Y
```

zou de geheugenlocatie ($1234 + Y) lezen en de daar gelezen waarde in het A-register plaatsen. Dit zou worden gecodeerd als $B9 $34 $12.

Absolute Indirect-modus

In deze modus is het 16-bits argument het adres dat naar het daadwerkelijk te lezen byte verwijst, dat wil zeggen de waarde van dat byte bevat. Indien geheugenlocatie $1234 bijvoorbeeld de waarde $78 bevat en geheugenlocatie $1235 de waarde $56, dan

```
JMP ($1234)
```

zou naar adres $5678 springen. De codering van deze adresseermethode is identiek aan die van de Absolute-modus en deze instructie zou dus worden gecodeerd als $6C $34 $12.

Absolute Indirect X-Indexed-modus

In deze modus is het 16-bits argument het adres dat naar het daadwerkelijk te lezen byte verwijst, dat wil zeggen de waarde van dat byte bevat. Deze modus is identiek aan de Absolute Indirect-modus, behalve dat de waarde van het X-register aan het pointeradres wordt toegevoegd. Indien het X-register bijvoorbeeld de waarde $04 bevat, geheugenlocatie $1238 de waarde $78 en geheugenlocatie $1239 de waarde $56, dan

```
JMP ($1234,X)
```

zou naar adres $5678 springen. De codering van deze adresseermethode is identiek aan die van de Absolute-modus en deze instructie zou dus worden gecodeerd als $7C $34 $12.

Base-Page Indirect X-Indexed-modus

Deze adresseermethode is identiek aan de Absolute Indirect X-Indexed-modus, behalve dat het adres van de pointer wordt gevormd uit het Base-Page-register (hoge byte) en de 8-bits operand (lage byte). De codering van deze adresseermethode is identiek aan die van de Base-Page-modus.

Indien het X-register bijvoorbeeld de waarde $04 bevat en de geheugenlocaties $16 en $17 in de huidige Base-Page respectievelijk $34 en $12 bevatten, dan

```
LDA ($12,X)
```

zou de inhoud van geheugenlocatie $1234 lezen en het resultaat opslaan in het A-register. Deze instructie wordt gecodeerd als $A1 $12.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page.

Base-Page indirect Y-geïndexeerde modus

Deze adresseermethode verschilt van de X-geïndexeerde indirecte modi doordat de waarde van het Y-register wordt opgeteld bij het adres dat van de pointer wordt gelezen, in plaats van bij de pointer zelf. Dit is een zeer nuttige modus die vaak wordt gebruikt omdat deze feitelijk toegang biedt tot "de byte met index Y in het geheugen op het adres waarnaar de operand verwijst." Met andere woorden, het derefereert een pointer. De codering voor deze adresseermethode is identiek aan de Base-Page-modus.

Bijvoorbeeld, als de Y-register de waarde $04 bevat en de geheugenlocaties $12 en $13 in de huidige Base-Page respectievelijk $78 en $56 bevatten, dan

```
LDA ($12),Y
```

zou de inhoud van geheugenlocatie $567C (d.w.z. $5678 + Y) lezen en het resultaat opslaan in het A-register. Deze instructie wordt gecodeerd als $B1 $12.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page.

Base-Page indirect Z-geïndexeerde modus

Deze adresseermethode verschilt van de X-geïndexeerde indirecte modi doordat de waarde van het Z-register wordt opgeteld bij het adres dat van de pointer wordt gelezen, in plaats van bij de pointer zelf. Dit is een zeer nuttige modus die vaak wordt gebruikt omdat deze feitelijk toegang biedt tot "de byte met index Z in het geheugen op het adres waarnaar de operand verwijst." Met andere woorden, het derefereert een pointer. De codering voor deze adresseermethode is identiek aan de Base-Page-modus.

Bijvoorbeeld, als de Z-register de waarde $04 bevat en de geheugenlocaties $12 en $13 in de huidige Base-Page respectievelijk $78 en $56 bevatten, dan

```
LDA ($12),Z
```

zou de inhoud van geheugenlocatie $567C (d.w.z. $5678 + Z) lezen en het resultaat opslaan in het A-register. Deze instructie wordt gecodeerd als $B2 $12.

Met andere woorden, het is equivalent aan de Base-Page indirect Y-geïndexeerde modus, maar met de Z-register in plaats van de Y-register om de offset te berekenen.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page.

Base-Page Quad indirect Z-geïndexeerde modus

Deze adresseermethode is identiek aan de Base-Page indirect Z-geïndexeerde modus, met als uitzondering dat er met 32 bits aan data wordt gewerkt. De codering

voor deze adresseermethode is identiek aan de Base-Page-modus, met als uitzondering dat deze wordt voorafgegaan door $42, $42. Bijvoorbeeld, als de Z-register de waarde $04 bevat en de geheugenlocaties $12 en $13 in de huidige Base-Page respectievelijk $CD en $AB bevatten, dan

```
LDQ ($12),Z
```

zou de inhoud van geheugenlocatie $ABD1 (d.w.z. $ABCD + Z) tot en met $ABD4 lezen en het resultaat opslaan in het Q Pseudo Register. Deze instructie wordt gecodeerd als $42 $42 $B2 $12.

Momenteel is LDQ de enige instructie die deze modus ondersteunt.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page. Zie de opmerking op pagina L-5 voor meer informatie over instructies in de Quad-modus.

32-bits Base-Page-indirecte Z-geïndexeerde modus

Deze modus ontstaat door een Base-Page-indirecte Z-geïndexeerde instructie te laten voorafgaan door de NOP-instructie (opcode $EA). Hierdoor leest de 45GS02 in plaats van een 16-bits adres een 32-bits adres uit het Base-Page-adres dat door de operand wordt aangegeven. De Z-index wordt bij die pointer opgeteld. Belangrijk is dat het 32-bits adres niet verwijst naar het huidige 64KB-geheugenbeeld van de processor, maar naar de werkelijke 28-bits adresruimte van de 45GS02. Hierdoor is elk geheugengebied eenvoudig toegankelijk, zonder dat complexe bank-switching- of DMA-operaties nodig zijn.

Indien adressen $12 tot $15 de bytes $20, $30, $FD, $0F bevatten, wat het 32-bits adres $FFD3020 weergeeft, d.w.z. het natuurlijke adres van het randkleurregister van de VIC-IV, en de Z-index de waarde $01 bevat, zou de volgende instructiesequentie de schermkleur op blauw zetten, omdat zich het schermkleurregister op $FFD3021 bevindt, dus $FFD3020 + Z:

```
LDA #$06
LDZ #$01
STA [$12],Z
```

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page.

32-bits Base-Page-(Zero-Page)-indirecte quad Z-geïndexeerde modus

Deze adresseermethode is identiek aan de 32-bits Base-Page-indirecte Z-geïndexeerde modus, met als uitzondering dat deze op 32 bits gegevens werkt op het 32-bits adres dat door het argument wordt gevormd, in tegenstelling tot de 32-bits Base-Page-indirecte Z-geïndexeerde modus die slechts op 8 bits gegevens werkt. De

codering van deze adresseermethode is $42, $42, $EA, gevolgd door de natuurlijke 6502-opcode van de instructie die wordt uitgevoerd.

Het is ook belangrijk op te merken dat Z in werkelijkheid meestal *niet* bij de pointer wordt opgeteld, omdat het deel uitmaakt van het Q Pseudo Register. Alleen `LDQ ($nn),Z` telt Z op. Bijvoorbeeld,

```
LDQ [$12],Z
```

zou de geheugenlocaties $12 tot $15 uit de Base-Page lezen en die waarden gebruiken om het 32-bits adres te vormen waaruit het Q Pseudo Register wordt geladen. De Z-index wordt bij het 32-bits adres opgeteld. Deze instructie wordt gecodeerd als $42 $42 $EA $B2 $12.

Zie de opmerking op pagina L-5 voor meer informatie over Base-Page en Zero-Page. Zie de opmerking op pagina L-5 voor meer informatie over instructies in de Quad-modus. Zie de opmerking op pagina L-11 voor meer informatie over 32-bits Base-Page-indirecte adressering.

Stapelrelatieve indirecte Y-geïndexeerde modus

Deze adresseermethode lijkt op de Base-Page-indirecte Y-geïndexeerde modus, met als uitzondering dat de operand in plaats van het adres van de pointer in de Base-Page de offset in de stapel aangeeft om de pointer te vinden. Deze adresseermethode derefereert effectief een pointer die op de stapel is geplaatst, bijv. als onderdeel van een functieaanroep vanuit een hogere programmeertaal. De codering is identiek aan die van de Base-Page-modus.

Bijvoorbeeld,

```
LDA ($12,SP),Y
```

Dit zou de inhoud van het geheugen op de huidige stapelaanwijzer plus $12 gebruiken om het adres van de pointer te berekenen. Aan deze pointer wordt vervolgens de waarde van het Y-register opgeteld om het uiteindelijke adres te verkrijgen waaruit wordt gelezen en om de waarde in het accumulatorregister op te slaan. De bytestroom van deze instructie is $E2 $12.

Indien de stapelaanwijzer momenteel naar $01E0 wijst, wordt het pointeradres gelezen van adressen $1F2 en $1F3, dus de twee bytes op $1E0 + $12 en $1E0 + $13. Indien de locaties $1F2 en $1F3 respectievelijk $78 en $56 bevatten en het Y-register de waarde $34 bevat, zou de uiteindelijke geheugenlocatie die wordt gelezen $5678 + $34 = $56AC zijn en wordt de inhoud van die geheugenlocatie in het accumulatorregister geladen.

Relatieve adresseermethode

Bij deze adresseermethode is de operand een 8-bits teken-offset ten opzichte van de huidige waarde van de programmateller (PC). Hiermee kunnen vertakkingen het

nabije adres coderen waar de uitvoering wordt voortgezet indien de vertakking wordt gevolgd.

Bijvoorbeeld,

```
BNE $2003
```

zou naar $2003 springen indien de Z-vlag van de processor niet gezet is. Indien deze instructie zich op adres $2000 bevindt, wordt deze gecodeerd als $D0 $01, dus een vertakking naar +1 bytes na de PC. Vertakkingsoffsets groter dan $7F vertakken achterwaarts, waarbij $FD naar de byte direct voor de vertakkinstructie vertakt en lagere waarden geleidelijk verder achterwaarts vertakken. Op deze manier kan effectief een vertaking worden gemaakt tussen -125 en +127 bytes vanaf de opcode-byte van de vertakinstructie. Voor langere vertaktingen ondersteunt de 45GS02 de relatieve word-adresseermethode, waarbij de offset met 2 bytes in plaats van 1 wordt gecodeerd.

Relatieve word-adresseermethode

Deze adresseermethode is identiek aan de relatieve adresseermethode, met als uitzondering dat de adresoffset een 16-bits waarde is. Hiermee kan een relatieve vertakking of sprong naar elke locatie in het huidige 64KB-geheugenbeeld worden gemaakt. Dit maakt het mogelijk volledig verplaatsbare software te schrijven, omdat absolute adressen bij het aanroepen van routines niet nodig zijn.

Bijvoorbeeld,

```
BNE $3000
```

zou naar $3000 springen indien de Z-vlag van de processor niet gezet is. Indien deze instructie zich op adres $2002 bevindt, wordt deze gecodeerd als $D3 $FC $0F, dus een vertakking naar +$FFC = 4092 bytes na de tweede byte van de instructie. Het feit dat de instructie 3 bytes lang is, wordt bij deze berekening buiten beschouwing gelaten.

6502-INSTRUCTIESET

Opmerking: de mechanismen voor het overschakelen van de 4510-CPU naar de 6502-CPU-persoonlijkheid zijn nog niet vastgelegd.

Officiële en onbedoelde instructies

De opcode-matrix van de 6502 heeft een grootte van 16 x 16 = 256 mogelijke opcodes. De opcodes die officieel gedocumenteerd zijn, vormen de verzameling van de legal instructies. Alle instructies van deze geldige verzameling worden voorafgegaan door een blauw gekleurde mnemoniek.

De overige opcodes vormen de verzameling van de onbedoelde instructies (soms worden deze "illegal"-instructies genoemd). Volledigheidshalve worden ook deze gedocumenteerd. Alle instructies van de onbedoelde verzameling worden voorafgegaan door een rood gekleurde mnemoniek.

OPMERKING: De onbedoelde instructies zijn momenteel niet geïmplementeerd en er wordt niet gegarandeerd dat zij exact dezelfde resultaten produceren als op andere CPU's van de 65xx-familie. Het is bekend dat veel van deze instructies instabiel zijn, zelfs wanneer zij op oude hardware draaien.

Opcodetabel

De opcodetabel vermeldt alle mogelijke opcodes, hun grootte, cycli en adresseermethode in een beknopt formaat.

Een cel met een lichtrode achtergrond duidt een onbedoelde instructie aan.

	$x0	$x1
$0x	size *cyc* OPC mode	size *cyc* OPC mode

De letters die aan het aantal cycli zijn toegevoegd, hebben de volgende betekenis:

	Betekenis
b	Voeg één cyclus toe als een branch wordt genomen. Voeg één cyclus extra toe als een genomen branch een paginagrens overschrijdt.
p	Voeg één cyclus toe als indexering een paginagrens overschrijdt.

Opcode-tabel 6502

	$x0	$x1	$x2	$x3	$x4	$x5	$x6	$x7
$0x	1 BRK *7* imp	2 ORA *6* bpiX	1 KIL *9* imp	2 SLO *8* bpiX	2 NOP *3* bp	2 ORA *3* bp	2 ASL *5* bp	2 SLO *5* bp
$1x	2 BPL *2b* rel	2 ORA *5p* ibpY	1 KIL *9* imp	2 SLO *8* ibpY	2 NOP *4* bpX	2 ORA *4p* bpX	2 ASL *6* bpX	2 SLO *6* bpX
$2x	3 JSR *6* abs	2 AND *6* bpiX	1 KIL *9* imp	2 RLA *8* bpiX	2 BIT *3* bp	2 AND *3* bp	2 ROL *5* bp	2 RLA *5* bp
$3x	2 BMI *2b* rel	2 AND *5p* ibpY	1 KIL *9* imp	2 RLA *8* ibpY	2 NOP *4* bpX	2 AND *4* bpX	2 ROL *6* bpX	2 RLA *6* bpX
$4x	1 RTI *6* imp	2 EOR *6* bpiX	1 KIL *9* imp	2 SRE *8* bpiX	2 NOP *3* bp	2 EOR *3* bp	2 LSR *5* bp	2 SRE *5* bp
$5x	2 BVC *2b* rel	2 EOR *5p* ibpY	1 KIL *9* imp	2 SRE *8* ibpY	2 NOP *4* bpX	2 EOR *4* bpX	2 LSR *6* bpX	2 SRE *6* bpX
$6x	1 RTS *6* imp	2 ADC *6* bpiX	1 KIL *9* imp	2 RRA *8* bpiX	2 NOP *3* bp	2 ADC *3* bp	2 ROR *5* bp	2 RRA *5* bp
$7x	2 BVS *2b* rel	2 ADC *5p* ibpY	1 KIL *9* imp	2 RRA *8* ibpY	2 NOP *4* bpX	2 ADC *4* bpX	2 ROR *6* bpX	2 RRA *6* bpX
$8x	2 NOP *2* imm	2 STA *6* bpiX	2 NOP *2* imm	2 SAX *6* bpiX	2 STY *3* bp	2 STA *3* bp	2 STX *3* bp	2 SAX *3* bp
$9x	2 BCC *2b* rel	2 STA *6* ibpY	1 KIL *9* imp	2 SHA *6* ibpY	2 STY *4* bpX	2 STA *4* bpX	2 STX *4* bpY	2 SAX *4* bpY
$Ax	2 LDY *2* imm	2 LDA *6* bpiX	2 LDX *2* imm	2 LAX *6* bpiX	2 LDY *3* bp	2 LDA *3* bp	2 LDX *3* bp	2 LAX *3* bp
$Bx	2 BCS *2b* rel	2 LDA *5p* ibpY	1 KIL *9* imp	2 LAX *5p* ibpY	2 LDY *4* bpX	2 LDA *4* bpX	2 LDX *4* bpY	2 LAX *4* bpY
$Cx	2 CPY *2* imm	2 CMP *6* bpiX	2 NOP *2* imm	2 DCP *8* bpiX	2 CPY *3* bp	2 CMP *3* bp	2 DEC *5* bp	2 DCP *5* bp
$Dx	2 BNE *2b* rel	2 CMP *5p* ibpY	1 KIL *9* imp	2 DCP *8* ibpY	2 NOP *4* bpX	2 CMP *4* bpX	2 DEC *6* bpX	2 DCP *6* bpX
$Ex	2 CPX *2* imm	2 SBC *6* bpiX	2 NOP *2* imm	2 ISC *8* bpiX	2 CPX *3* bp	2 SBC *3* bp	2 INC *5* bp	2 ISC *5* bp
$Fx	2 BEQ *2b* rel	2 SBC *5p* ibpY	1 KIL *9* imp	2 ISC *8* ibpY	2 NOP *4* bpX	2 SBC *4* bpX	2 INC *6* bpX	2 ISC *6* bpX

Opcode-tabel 6502

$x8	$x9	$xA	$xB	$xC	$xD	$xE	$xF	
1 PHP 3 imp	2 ORA 2 imm	1 ASL 2 acc	2 ANC 2 imm	3 NOP 4 abs	3 ORA 4 abs	3 ASL 6 abs	3 SLO 6 abs	$0x
1 CLC 2 imp	3 ORA 4p absY	1 NOP 2 imp	3 SLO 7 absY	3 NOP 4p absX	3 ORA 4p absX	3 ASL 7 absX	3 SLO 7 absX	$1x
1 PLP 4 imp	2 AND 2 imm	1 ROL 2 acc	2 ANC 2 imm	3 BIT 4 abs	3 AND 4 abs	3 ROL 6 abs	3 RLA 6 abs	$2x
1 SEC 2 imp	3 AND 4p absY	1 NOP 2 imp	3 RLA 7 absY	3 NOP 4p absX	3 AND 4p absX	3 ROL 7 absX	3 RLA 7 absX	$3x
1 PHA 3 imp	2 EOR 2 imm	1 LSR 2 acc	2 ALR 2 imm	3 JMP 3 abs	3 EOR 4 abs	3 LSR 6 abs	3 SRE 6 abs	$4x
1 CLI 2 imp	3 EOR 4p absY	1 NOP 2 imp	3 SRE 7 absY	3 NOP 4p absX	3 EOR 4p absX	3 LSR 7 absX	3 SRE 7 absX	$5x
1 PLA 4 imp	2 ADC 2 imm	1 ROR 2 acc	2 ARR 2 imm	3 JMP 5 ind	3 ADC 4 abs	3 ROR 6 abs	3 RRA 6 abs	$6x
1 SEI 2 imp	3 ADC 4p absY	1 NOP 2 imp	3 RRA 7 absY	3 NOP 4p absX	3 ADC 4p absX	3 ROR 7 absX	3 RRA 7 absX	$7x
1 DEY 2 imp	2 NOP 2 imm	1 TXA 2 imp	2 XAA 2 imm	3 STY 4 abs	3 STA 4 abs	3 STX 4 abs	3 SAX 4 abs	$8x
1 TYA 2 imp	3 STA 5 absY	1 TXS 2 imp	3 TAS 5 absY	3 SHY 5 absX	3 STA 5 absX	3 SHX 5 absY	3 SHA 5 absY	$9x
1 TAY 2 imp	2 LDA 2 imm	1 TAX 2 imp	2 LAX 2 imm	3 LDY 4 abs	3 LDA 4 abs	3 LDX 4 abs	3 LAX 4 abs	$Ax
1 CLV 2 imp	3 LDA 4p absY	1 TSX 2 imp	3 LAS 4p absY	3 LDY 4p absX	3 LDA 4p absX	3 LDX 4p absY	3 LAX 4p absY	$Bx
1 INY 2 imp	2 CMP 2 imm	1 DEX 2 imp	2 SBX 2 imm	3 CPY 4 abs	3 CMP 4 abs	3 DEC 6 abs	3 DCP 6 abs	$Cx
1 CLD 2 imp	3 CMP 4p absY	1 NOP 2 imp	3 DCP 7 absY	3 NOP 4p absX	3 CMP 4p absX	3 DEC 7 absX	3 DCP 7 absX	$Dx
1 INX 2 imp	2 SBC 2 imm	1 NOP 2 imp	2 SBC 2 imm	3 CPX 4 abs	3 SBC 4 abs	3 INC 6 abs	3 ISC 6 abs	$Ex
1 SED 2 imp	3 SBC 4p absY	1 NOP 2 imp	3 ISC 7 absY	3 NOP 4p absX	3 SBC 4p absX	3 INC 7 absX	3 ISC 7 absX	$Fx

ADC

Deze instructie telt het argument en de carry-vlag op bij de inhoud van het accumulatorregister. Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten, anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De C-vlag wordt gezet als het ongetekende resultaat >255 is, of >99 indien de D-vlag is gezet.

ADC : Add with carry **6502**

$A \leftarrow A + M + C$

			N	Z	I	C	D	V	E	
			+	+	·	+	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect,X)	ADC ($nn,X)	61	2	6						
zero-page	ADC $nn	65	2	3						
immediate 8bit	ADC #$nn	69	2	2						
absolute	ADC $nnnn	6D	3	4						
(indirect),Y	ADC ($nn),Y	71	2	5						p
zero-page,X	ADC $nn,X	75	2	4						
absolute,Y	ADC $nnnn,Y	79	3	4						p
absolute,X	ADC $nnnn,X	7D	3	4						p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

ALR [onbedoeld]

Deze instructie schuift het accumulatorregister één bit naar rechts na het uitvoeren van een bitgewijze EN van het accumulatorregister en het immediate-mode argument. Bit 7 wordt op nul gezet en bit 0 wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

- De C-vlag wordt gezet op bit 0 van het AND-resultaat, voor het verschuiven.

ALR : Binary AND and Logical Shift Right **6502**

A ← A *AND* M, C ← A(0), A ← A≫1, A(7) ← 0

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	ALR #$nn	4B	2	2

ANC [onbedoeld]

Deze instructie voert een bitgewijze EN-operatie uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister en die ook in het argument staan, worden na voltooiing ingesteld in het accumulatorregister. In tegenstelling tot de AND-instructie wordt de carry-vlag gezet alsof het resultaat één bit naar links is verschoven. Dat wil zeggen, de carry-vlag wordt op dezelfde manier gezet als de N-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

ANC : Binary AND, and Set Carry **6502**

A ← A *AND* M, C ← A(7)

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	ANC #$nn	0B	2	2
immediate 8bit	ANC #$nn	2B	2	2

AND

Deze instructie voert een bitgewijze EN-operatie uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister en die ook in het argument staan, worden na voltooiing ingesteld in het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

AND : Binary AND **6502**

A ← A *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	AND ($nn,X)	21	2	6	
zero-page	AND $nn	25	2	3	
immediate 8bit	AND #$nn	29	2	2	
absolute	AND $nnnn	2D	3	4	
(indirect),Y	AND ($nn),Y	31	2	5	p
zero-page,X	AND $nn,X	35	2	4	
absolute,Y	AND $nnnn,Y	39	3	4	p
absolute,X	AND $nnnn,X	3D	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

ARR [onbedoeld]

Deze instructie verschuift het accumulatorregister één bit naar rechts na het uitvoeren van een bitgewijze EN-operatie van het accumulatorregister en het argument M in de immediate-addreringswijze. Bit 7 wordt uitgewisseld met de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De V-vlag wordt op een of andere manier beïnvloed.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ARR : Binary AND and Rotate Right **6502**

A ← A *AND* M, A ← A$\gg$1, C ↔ A(7)

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	ARR #$nn	6B	2	2

ASL

Deze instructie verschuift het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt op nul gezet en bit 7 wordt verschoven naar de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ASL : Arithmetic Shift Left Memory or Accumulator **6502**

$A \leftarrow A \ll 1$ of $M \leftarrow M \ll 1$

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	ASL $nn	06	2	5
accumulator	ASL A	0A	1	2
absolute	ASL $nnnn	0E	3	6
zero-page,X	ASL $nn,X	16	2	6
absolute,X	ASL $nnnn,X	1E	3	7

BCC

Deze instructie vertakt naar het aangegeven adres als de carry-vlag gewist is.

BCC : Branch on Carry Flag Clear **6502**

$C{=}0 \Longrightarrow PC \leftarrow PC + R8$

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BCC $rr	90	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BCS

Deze instructie vertakt naar het aangegeven adres als de carry-vlag gezet is.

BCS : Branch on Carry Flag Set **6502**

C=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

			N	Z	I	C	D	V	E	
			.	.	.	.	.	.	.	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
relative	BCS $rr	B0	2	2	b					

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BEQ

Deze instructie vertakt naar het aangegeven adres als de zero-vlag gezet is. BEQ staat voor branch if equal, omdat een CMP ertoe leidt dat de zero-vlag gezet wordt als de operanden gelijk zijn.

BEQ : Branch on Zero Flag Set **6502**

Z=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

			N	Z	I	C	D	V	E	
			.	.	.	.	.	.	.	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
relative	BEQ $rr	F0	2	2	b					

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BIT

Deze instructie wordt gebruikt om de bits te testen die zijn opgeslagen op een geheugenlocatie of in het immediate-argument van de opcode.

Bits 6 en 7 van de inhoud van de geheugenlocatie worden rechtstreeks gekopieerd naar de overflow-vlag en de negatieve-vlag. De zero-vlag wordt gezet of gewist op basis van het resultaat van de binaire AND van het accumulatorregister en de inhoud van de aangegeven geheugenlocatie.

De immediate-test zet de N- en V-vlaggen met geldige toestanden, waarbij het argument wordt behandeld als de geheugenwaarde, wat bij de eerdere 65C02-implementatie niet het geval was.

Neveneffecten

- De N-vlag wordt gezet als bit 7 van de geheugenlocatie gezet is, anders wordt hij gewist.
- De V-vlag wordt gezet als bit 6 van de geheugenlocatie gezet is, anders wordt hij gewist.

- De Z-vlag wordt gezet als het resultaat van A *AND* M nul is, anders wordt hij gewist.

BIT : Perform Bit Test **6502**

N ← M(7), V ← M(6), Z ← A *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	BIT $nn	24	2	3
absolute	BIT $nnnn	2C	3	4

BMI

Deze instructie vertakt naar het aangegeven adres als de negatieve-vlag gezet is. BMI staat voor branch on minus.

BMI : Branch on Negative Flag Set **6502**

N=1 $\Longrightarrow$ PC ← PC + R8

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BMI $rr	30	2	2 ^{b}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BNE

Deze instructie vertakt naar het aangegeven adres als de zero-vlag gewist is. BNE staat voor branch if not equal, omdat een CMP ertoe leidt dat de zero-vlag gewist wordt als de operanden niet gelijk zijn.

BNE : Branch on Zero Flag Clear **6502**

Z=0 $\Longrightarrow$ PC ← PC + R8

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BNE $rr	D0	2	2 ^{b}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BPL

Deze instructie vertakt naar het aangegeven adres als de negatieve-vlag gewist is. BPL staat voor branch on plus.

BPL : Branch on Negative Flag Clear **6502**

N=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BPL $rr	10	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BRK

terruptserviceroutine onderscheid maken tussen IRQ-gebeurtenissen en BRK-gebeurtenissen. Bijvoorbeeld:

```
PLA           ; laad het statusregister
PHA           ; herstel de stapel
AND #$10      ; maskeer de break-vlag
BNE DO_BREAK  ; -> het was een BRK
...           ; anders door met de IRQ-server
```

Geciteerd uit: MCS6500 Microcomputer Family Programming Manual, januari 1976, tweede editie, MOS Technology Inc., blz. 144:

"De BRK is een instructie van één byte en de adresseermethode ervan is Implied."

Er wordt gediscussieerd over de vraag of BRK kan worden beschouwd als een instructie van twee bytes met de immediate-adresseermethode, waarbij het operand-byte wordt weggegooid. Het byte na de BRK zou dan als oproepargument voor de break-handler kunnen dienen. Commodore gebruikte echter de BRK, zoals in de handleiding vermeld, als een instructie van één byte, die in de ML-monitor breekt als deze aanwezig is. Deze ingebouwde monitors verlaagden de op de stapel geplaatste PC, zodat deze kon worden gebruikt om direct terug te keren naar of te springen naar het code-byte na de BRK.

BRK : SP $\leftarrow$ SP $-$ 2, STAPEL $\leftarrow$ PC + 2, PC $\leftarrow$ ($FFFE) **6502**

M+

N	Z	I	C	D	V	E
.	.	.	.	**e**	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	BRK	00	1	7

BVC

Deze instructie vertakt naar het aangegeven adres als de overflow-vlag (V) gewist is.

BVC : Branch on Overflow Flag Clear **6502**

V=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BVC $rr	50	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BVS

Deze instructie vertakt naar het aangegeven adres als de overflow-vlag (V) gezet is.

BVS : Branch on Overflow Flag Set **6502**

V=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BVS $rr	70	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

CLC

Deze instructie wist de carry-vlag.

Neveneffecten

- De C-vlag wordt gewist.

CLC : Clear Carry Flag **6502**

C $\leftarrow$ 0

N	Z	I	C	D	V	E
.	.	.	+	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	CLC	18	1	2

CLD

Deze instructie wist de decimalenvlag. Rekenkundige bewerkingen maken dan gebruik van gewoon binair rekenen, in plaats van met BCD-gewogen decimale getallen (BCD).

Neveneffecten

- De D-vlag wordt gewist.

CLD : Clear Decimal Flag				**6502**
D ← 0				
			N Z I C D V E	
			· · · · + · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	CLD	D8	1	2

CLI

Deze instructie wist de interruptblokkeervlag. Interrupts kunnen nu plaatsvinden.

Neveneffecten

- De I-vlag wordt gewist.

CLI : Clear Interrupt Disable Flag				**6502**
I ← 0				
			N Z I C D V E	
			· · + · · · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	CLI	58	1	2

CLV

Deze instructie wist de overflow-vlag.

Neveneffecten

- De V-vlag wordt gewist.

CLV : Clear Overflow Flag				**6502**
V ← 0				
			N Z I C D V E	
			· · · · · + ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	CLV	B8	1	2

CMP

Deze instructie voert A − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt deze gewist.

CMP : Compare Accumulator **6502**

N,C,Z ⇐ [A − M]

			N	Z	I	C	D	V	E
			+	+	·	+	·	·	·
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**					
(indirect,X)	CMP ($nn,X)	C1	2	6					
zero-page	CMP $nn	C5	2	3					
immediate 8bit	CMP #$nn	C9	2	2					
absolute	CMP $nnnn	CD	3	4					
(indirect),Y	CMP ($nn),Y	D1	2	5	p				
zero-page,X	CMP $nn,X	D5	2	4					
absolute,Y	CMP $nnnn,Y	D9	3	4	p				
absolute,X	CMP $nnnn,X	DD	3	4	p				

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

CPX

Deze instructie voert X − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van X − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van X − M nul of positief is, d.w.z. als X niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van X − M nul is; anders wordt deze gewist.

CPX : Compare X Register **6502**

N,C,Z ⇐ [X − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	CPX #$nn	E0	2	2
zero-page	CPX $nn	E4	2	3
absolute	CPX $nnnn	EC	3	4

CPY

Deze instructie voert Y − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Y − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van Y − M nul of positief is, d.w.z. als Y niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van Y − M nul is; anders wordt deze gewist.

CPY : Compare Y Register **6502**

N,C,Z ⇐ [Y − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	CPY #$nn	C0	2	2
zero-page	CPY $nn	C4	2	3
absolute	CPY $nnnn	CC	3	4

DCP [onbedoeld]

Deze instructie verlaagt de inhoud van de aangegeven geheugenlocatie met één en voert daarna A − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.

- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt deze gewist.

DCP : Decrement and Compare Accumulator **6502**

M ← M − 1, N,C,Z ⇐ [A − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	DCP ($nn,X)	C3	2	8
zero-page	DCP $nn	C7	2	5
absolute	DCP $nnnn	CF	3	6
(indirect),Y	DCP ($nn),Y	D3	2	8
zero-page,X	DCP $nn,X	D7	2	6
absolute,Y	DCP $nnnn,Y	DB	3	7
absolute,X	DCP $nnnn,X	DF	3	7

DEC

Deze instructie verlaagt het accumulatorregister of de aangegeven geheugenlocatie met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEC : Decrement **6502**

A ← A − 1 *of* M ← M − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	DEC $nn	C6	2	5
absolute	DEC $nnnn	CE	3	6
zero-page,X	DEC $nn,X	D6	2	6
absolute,X	DEC $nnnn,X	DE	3	7

DEX

Deze instructie verlaagt het X-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEX : Decrement X Register **6502**

X ← X − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	DEX	CA	1	2

DEY

Deze instructie verlaagt het Y-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEY : Decrement Y Register **6502**

Y ← Y − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	DEY	88	1	2

EOR

Deze instructie voert een bitgewijze uitsluitende-OF-operatie (XOR) uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister of die ingesteld staan in het argument, worden na voltooiing ingesteld in het accumulatorregister, maar niet allebei.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

EOR : Binary Exclusive OR					**6502**
A ← A *XOR* M					
			N Z I C D V E		
			+ + · · · · ·		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**	
(indirect,X)	EOR ($nn,X)	41	2	6	
zero-page	EOR $nn	45	2	3	
immediate 8bit	EOR #$nn	49	2	2	
absolute	EOR $nnnn	4D	3	4	
(indirect),Y	EOR ($nn),Y	51	2	5	p
zero-page,X	EOR $nn,X	55	2	4	
absolute,Y	EOR $nnnn,Y	59	3	4	p
absolute,X	EOR $nnnn,X	5D	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

INC

Deze instructie verhoogt het accumulatorregister of de aangegeven geheugenlocatie met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INC : Increment Memory or Accumulator					**6502**
A ← A + 1 of M ← M + 1					
			N Z I C D V E		
			+ + · · · · ·		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**	
zero-page	INC $nn	E6	2	5	
absolute	INC $nnnn	EE	3	6	
zero-page,X	INC $nn,X	F6	2	6	
absolute,X	INC $nnnn,X	FE	3	7	

INX

Deze instructie verhoogt het X-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.

- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INX : Increment X Register				**6502**
X ← X + 1				
			N Z I C D V E	
			+ + · · · · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	INX	E8	1	2

INY

Deze instructie verhoogt het Y-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INY : Increment Y Register				**6502**
Y ← Y + 1				
			N Z I C D V E	
			+ + · · · · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	INY	C8	1	2

ISC [onbedoeld]

Deze instructie verhoogt de aangegeven geheugenlocatie en voert vervolgens A — M - 1 + C uit en zet de processorvlaggen dienovereenkomstig. Het resultaat wordt opgeslagen in het accumulatorregister.

OPMERKING: Deze instructie wordt beïnvloed door de status van de decimalenvlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A — M negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A — M nul of positief is, d.w.z. als A niet kleiner is dan M, anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten, anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.

- De Z-vlag wordt gezet als het resultaat van A − M nul is, anders wordt de vlag gewist.

ISC : Increment Memory, Subtract With Carry **6502**

M ← M + 1, A ← − M − 1 + C

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	ISC ($nn,X)	E3	2	8
zero-page	ISC $nn	E7	2	5
absolute	ISC $nnnn	EF	3	6
(indirect),Y	ISC ($nn),Y	F3	2	8
zero-page,X	ISC $nn,X	F7	2	6
absolute,Y	ISC $nnnn,Y	FB	3	7
absolute,X	ISC $nnnn,X	FF	3	7

JMP

Deze instructie stelt het Program Counter (PC)-register in op het door de instructie aangegeven adres, waardoor de uitvoering vanaf dat adres wordt voortgezet.

JMP : Jump to Address **6502**

PC ← M2:M1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute	JMP $nnnn	4C	3	3
(indirect)	JMP ($nnnn)	6C	3	5

JSR

Deze instructie slaat het adres van de instructie na de JSR-instructie op in de stapel en stelt vervolgens het Program Counter (PC)-register in op het door de instructie aangegeven adres, waardoor de uitvoering vanaf dat adres wordt voortgezet. Omdat het terugkeeradres in de stapel is opgeslagen, kan de RTS-instructie worden gebruikt om vanuit de aangeroepen subroutine terug te keren en de uitvoering na de JSR-instructie voort te zetten.

OPMERKING: deze instructie plaatst in feite het adres van de laatste byte van de JSR-instructie op de stapel. De RTS-instructie weet dit uiteraard en verhoogt het adres wanneer het van de stapel wordt gehaald, voordat het Program Counter-register (PC) wordt ingesteld.

JSR : Jump to Sub-Routine **6502**

SP ← SP − 2, STAPEL ← PCH:PCL, PC ← M2:M1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute	JSR $nnnn	20	3	6

KIL [onbedoeld]

Op een 6502 zorgen deze instructies ervoor dat de processor in een oneindige lus in zijn interne logica terechtkomt, welke alleen onderbroken kan worden door de computer te resetten. Op de 45GS02 veroorzaken deze instructies Hypervisor Traps, zodra deze functionaliteit is geïmplementeerd. Zo kunnen zij worden gebruikt om vast te stellen of er een 6502 of een 45GS02 draait: op een 6502-processor keert de instructie nooit terug, terwijl zij op een 45GS02 een uitzondering veroorzaakt, waardoor het aanroepende programma waarschijnlijk wordt afgebroken of crasht.

KIL : Lock-up 6502 Processor **6502**

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	KIL	02	1	9
implied	KIL	12	1	9
implied	KIL	22	1	9
implied	KIL	32	1	9
implied	KIL	42	1	9
implied	KIL	52	1	9
implied	KIL	62	1	9
implied	KIL	72	1	9
implied	KIL	92	1	9
implied	KIL	B2	1	9
implied	KIL	D2	1	9
implied	KIL	F2	1	9

LAS [onbedoeld]

OPMERKING: dit monster van een instructie is, afgezien van het feit dat het elk denkbaar nuttig doel ontbeert, onstabiel op veel 6502-processoren en dient daarom ook om die reden te worden vermeden, indien u niet al was afgeschrikt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.
- Een gevoel van holle voldoening, wanneer u daadwerkelijk een nuttig doel voor deze instructie ontdekt.

LAS : Set A, X and SPL Register With Useless Value **6502**

SP, A, X ← SP *AND* M

			N	Z	I	C	D	V	E	
			+	·	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
absolute,Y	LAS $nnnn,Y	BB	3	4	p					

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LAX [onbedoeld]

Deze instructie laadt zowel het accumulatorregister als het X-register met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

OPMERKING: de LAX-instructie staat erom bekend onstabiel te zijn op veel 6502-processoren en dient niet te worden gebruikt. Niet-immediate adresseermethoden zijn mogelijk stabiel genoeg om te gebruiken, maar dienen in het algemeen te worden vermeden.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.

LAX : Load Accumulator and X Registers **6502**

A, X ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	LAX ($nn,X)	A3	2	6	
zero-page	LAX $nn	A7	2	3	
immediate 8bit	LAX #$nn	AB	2	2	
absolute	LAX $nnnn	AF	3	4	
(indirect),Y	LAX ($nn),Y	B3	2	5	p
zero-page,Y	LAX $nn,Y	B7	2	4	
absolute,Y	LAX $nnnn,Y	BF	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LDA

Deze instructie laadt het accumulatorregister met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.

LDA : Load Accumulator **6502**

A ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	LDA ($nn,X)	A1	2	6	
zero-page	LDA $nn	A5	2	3	
immediate 8bit	LDA #$nn	A9	2	2	
absolute	LDA $nnnn	AD	3	4	
(indirect),Y	LDA ($nn),Y	B1	2	5	p
zero-page,X	LDA $nn,X	B5	2	4	
absolute,Y	LDA $nnnn,Y	B9	3	4	p
absolute,X	LDA $nnnn,X	BD	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LDX

Deze instructie laadt het X-register met de opgegeven waarde, of met de inhoud van de opgegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

LDX : Load X Register **6502**

X ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDX #$nn	A2	2	2	
zero-page	LDX $nn	A6	2	3	
absolute	LDX $nnnn	AE	3	4	
zero-page,Y	LDX $nn,Y	B6	2	4	
absolute,Y	LDX $nnnn,Y	BE	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LDY

Deze instructie laadt het Y-register met de opgegeven waarde, of met de inhoud van de opgegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

LDY : Load Y Register **6502**

Y ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDY #$nn	A0	2	2	
zero-page	LDY $nn	A4	2	3	
absolute	LDY $nnnn	AC	3	4	
zero-page,X	LDY $nn,X	B4	2	4	
absolute,X	LDY $nnnn,X	BC	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LSR

Deze instructie verschuift ofwel het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt op nul gezet en bit 0 wordt naar de carry-vlag uitgeschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.
- De C-vlag wordt gezet als bit 0 van de waarde gezet was voordat de waarde is verschoven.

LSR : Logical Shift Right **6502**

A ← A≫1, C ← A(0) of M ← M≫1, C ← M(0)

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	LSR $nn	46	2	5
accumulator	LSR A	4A	1	2
absolute	LSR $nnnn	4E	3	6
zero-page,X	LSR $nn,X	56	2	6
absolute,X	LSR $nnnn,X	5E	3	7

NOP

Deze instructies fungeren als noopinstructies: zij voeren de busoperaties uit alsof het echte instructies zijn, maar doen vervolgens niets met de opgehaalde waarde. Zij kunnen derhalve worden gebruikt als vertragingsinstructies, of om registers te lezen die bij het lezen neveneffecten hebben, zonder een register te beschadigen.

Op de 6502 is alleen $EA een opcode die voor NOP bedoeld is. Alle andere zijn uitsluitend beschikbaar op NMOS-versies van de processor, of op de 45GS02 in de 6502-modus.

NOP : No-Operation (some are unintended opcodes) **6502**

			N Z I C D V E		
			. . . .		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**	
zero-page	NOP $nn	04	2	3	
absolute	NOP $nnnn	0C	3	4	
zero-page,X	NOP $nn,X	14	2	4	
implied	NOP	1A	1	2	
absolute,X	NOP $nnnn,X	1C	3	4	p
zero-page,X	NOP $nn,X	34	2	4	
implied	NOP	3A	1	2	
absolute,X	NOP $nnnn,X	3C	3	4	p
zero-page	NOP $nn	44	2	3	
zero-page,X	NOP $nn,X	54	2	4	
implied	NOP	5A	1	2	
absolute,X	NOP $nnnn,X	5C	3	4	p
zero-page	NOP $nn	64	2	3	
zero-page,X	NOP $nn,X	74	2	4	
implied	NOP	7A	1	2	
absolute,X	NOP $nnnn,X	7C	3	4	p
immediate 8bit	NOP #$nn	80	2	2	
immediate 8bit	NOP #$nn	82	2	2	
immediate 8bit	NOP #$nn	89	2	2	
immediate 8bit	NOP #$nn	C2	2	2	
zero-page,X	NOP $nn,X	D4	2	4	
implied	NOP	DA	1	2	
absolute,X	NOP $nnnn,X	DC	3	4	p
immediate 8bit	NOP #$nn	E2	2	2	
implied	NOP	EA	1	2	
zero-page,X	NOP $nn,X	F4	2	4	
implied	NOP	FA	1	2	
absolute,X	NOP $nnnn,X	FC	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

ORA

Deze instructie voert een binaire OR-bewerking uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al

in het accumulatorregister gezet waren, of die in het argument gezet waren, worden bij voltooiing in het accumulatorregister gezet, of beide.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

ORA : Binary OR **6502**

A ← A *OR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	ORA ($nn,X)	01	2	6	
zero-page	ORA $nn	05	2	3	
immediate 8bit	ORA #$nn	09	2	2	
absolute	ORA $nnnn	0D	3	4	
(indirect),Y	ORA ($nn),Y	11	2	5	p
zero-page,X	ORA $nn,X	15	2	4	p
absolute,Y	ORA $nnnn,Y	19	3	4	p
absolute,X	ORA $nnnn,X	1D	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

PHA

Deze instructie duwt de inhoud van het accumulatorregister op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHA : Push Accumulator Register onto the Stack **6502**

SP ← SP − 1, STAPEL ← A

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHA	48	1	3

PHP

Deze instructie duwt de inhoud van de processorvlaggen op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHP : Push Processor Flags onto the Stack **6502**

SP ← SP − 1, STAPEL ← P

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHP	08	1	3

PLA

Deze instructie vervangt de inhoud van het accumulatorregister door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLA : Pull Accumulator Register from the Stack **6502**

A ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PLA	68	1	4

PLP

Deze instructie vervangt de inhoud van de processorvlaggen door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

OPMERKING: deze instructie vervangt NIET de 'Extended Stack Disable'-vlag (de E-vlag) of de software-interruptvlag (de B-vlag)

PLP : Pull Processor Flags from the Stack **6502**

P ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	+	+	+	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PLP	28	1	4

RLA [onbedoeld]

Deze instructie verschuift de inhoud van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt ingesteld op de huidige waarde van de carry-vlag en bit 7 wordt uitgeschoven in de carry-vlag. Het resultaat wordt vervolgens via een logische AND-bewerking met het accumulatorregister gecombineerd.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

RLA : Rotate Left Memory, and AND with Accumulator **6502**

$M \leftarrow M \ll 1$, $M(0) \leftarrow C$, $C \leftarrow M(7)$, $A \leftarrow A$ *AND* M

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	RLA ($nn,X)	23	2	8
zero-page	RLA $nn	27	2	5
absolute	RLA $nnnn	2F	3	6
(indirect),Y	RLA ($nn),Y	33	2	8
zero-page,X	RLA $nn,X	37	2	6
absolute,Y	RLA $nnnn,Y	3B	3	7
absolute,X	RLA $nnnn,X	3F	3	7

ROL

Deze instructie verschuift de inhoud van het accumulatorregister of van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt ingesteld op de huidige waarde van de carry-vlag en bit 7 wordt uitgeschoven in de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ROL : Rotate Left Memory or Accumulator **6502**

M ← M≪1, C ← M(7), M(0) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	ROL $nn	26	2	5
accumulator	ROL A	2A	1	2
absolute	ROL $nnnn	2E	3	6
zero-page,X	ROL $nn,X	36	2	6
absolute,X	ROL $nnnn,X	3E	3	7

ROR

Deze instructie verschuift ofwel het accumulatorregister ofwel de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt ingesteld op de huidige waarde van de carry-vlag, en bit 0 wordt in de carry-vlag verschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was gezet, voordat er werd verschoven.

ROR : Rotate Right Memory or Accumulator **6502**

M ← M≫1, C ← M(0), M(7) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	ROR $nn	66	2	5
accumulator	ROR A	6A	1	2
absolute	ROR $nnnn	6E	3	6
zero-page,X	ROR $nn,X	76	2	6
absolute,X	ROR $nnnn,X	7E	3	7

RRA [onbedoeld]

Deze instructie verschuift de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt ingesteld op de huidige waarde van de carry-vlag, en bit 0 wordt in de carry-vlag verschoven. Het resultaat wordt aan het accumulatorregister toegevoegd.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als de optelling in het accumulatorregister een overflow oplevert.

RRA : Rotate Right Memory, and Add to Accumulator **6502**

$M \leftarrow M \gg 1$, $C \leftarrow M(0)$, $M(7) \leftarrow C$, $A \leftarrow A + M$

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	RRA ($nn,X)	63	2	8
zero-page	RRA $nn	67	2	5
absolute	RRA $nnnn	6F	3	6
(indirect),Y	RRA ($nn),Y	73	2	8
zero-page,X	RRA $nn,X	77	2	6
absolute,Y	RRA $nnnn,Y	7B	3	7
absolute,X	RRA $nnnn,X	7F	3	7

RTI

Deze instructie haalt de processorvlaggen van de stapel en vervolgens het Program Counter (PC)-register van de stapel, waardoor een onderbroken programma kan worden hervat.

- De 6502-processorvlaggen worden hersteld van de stapel.
- Door deze instructie wordt noch de B-vlag (software-interrupt) noch de E-vlag (uitgebreide stapel) gezet.

RTI : Return From Interrupt **6502**

$P \leftarrow$ STAPEL, PCH:PCL $\leftarrow$ STAPEL, $SP \leftarrow SP - 3$

N	Z	I	C	D	V	E
+	+	+	+	+	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	RTI	40	1	6

RTS

Deze instructie haalt het Program Counter (PC)-register van de stapel en telt vervolgens een optioneel byte-argument op bij de stapelaanwijzer, waardoor een subroutine naar haar aanroeper kan terugkeren.

Het optionele argument is bedoeld om een aanroeper in staat te stellen argumenten voor de subroutine op de stapel te plaatsen. De subroutine leest de argumenten uit het stapelgeheugen en keert vervolgens terug met het aantal argumentbytes. De CPU haalt naast het retouradres ook de argumenten van de stapel, zodat de aanroeper de argumenten niet expliciet hoeft te verwijderen.

RTS : Return From Subroutine **6502**

PCH:PCL ← STAPEL, SP ← SP + 2 (+ M)

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	RTS	60	1	6

SAX [onbedoeld]

Deze instructie werkt als een combinatie van AND en CMP. Het resultaat wordt in het X-register opgeslagen. Omdat de instructie functionaliteit van CMP bevat in plaats van SBC, wordt de carry-vlag niet gebruikt bij de aftrekking, hoewel zij door de instructie wordt gewijzigd.

OPMERKING: deze instructie wordt beïnvloed door de status van de decimalenvlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt de vlag gewist. Als de vlag is gezet, geeft dit aan dat een getekende overflow is opgetreden.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt de vlag gewist.

SAX : AND Accumulator and X, and Subtract Without Carry **6502**

X ← (A *AND* X) − M

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	SAX ($nn,X)	83	2	6
zero-page	SAX $nn	87	2	3
absolute	SAX $nnnn	8F	3	4
zero-page,Y	SAX $nn,Y	97	2	4

SBC

Deze instructie voert A − M − 1 + C uit en zet de processorvlaggen dienovereenkomstig. Het resultaat wordt in het accumulatorregister opgeslagen.

OPMERKING: Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt de vlag gewist.

SBC : Subtract With Carry **6502**

A ← − M − 1 + C

			N	Z	I	C	D	V	E	
			+	+	·	+	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect,X)	SBC ($nn,X)	E1	2	6						
zero-page	SBC $nn	E5	2	3						
immediate 8bit	SBC #$nn	E9	2	2						
immediate 8bit	SBC #$nn	EB	2	2						
absolute	SBC $nnnn	ED	3	4						
(indirect),Y	SBC ($nn),Y	F1	2	5						p
zero-page,X	SBC $nn,X	F5	2	4						
absolute,Y	SBC $nnnn,Y	F9	3	4						p
absolute,X	SBC $nnnn,X	FD	3	4						p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

SBX [onbedoeld]

Deze instructie laadt het X-register met de binaire EN van het accumulatorregister en het X-register, verminderd met het immediate-argument.

OPMERKING: het aftrekkingseffect in deze instructie is te danken aan CMP , niet aan . De negatieve-vlag wordt derhalve gezet volgens de werking van CMP, niet van SBC. Dat wil zeggen dat de carry-vlag niet in de berekening wordt gebruikt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat nul of positief is; anders wordt de vlag gewist.

SBX : AND and Subtract **6502**

X ← (A *AND* X) − M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	SBX #$nn	CB	2	2

SEC

Deze instructie zet de carry-vlag.

Neveneffecten

- De C-vlag wordt gezet.

SEC : Set Carry Flag **6502**

C ← 1

N	Z	I	C	D	V	E
·	·	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	SEC	38	1	2

SED

Deze instructie zet de decimalenvlag. Rekenkundige bewerkingen maken vanaf nu gebruik van de BCD-modus (Binary Coded Decimal).

OPMERKING: de interrupt-handler van de C64 wist de decimalenvlag niet, waardoor het gevaarlijk is de decimalenvlag te zetten zonder eerst de interruptblokkeervlag te zetten.

Neveneffecten

- De D-vlag wordt gezet.

SED : Set Decimal Flag **6502**

D ← 1

N	Z	I	C	D	V	E
·	·	·	·	+	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	SED	F8	1	2

SEI

Deze instructie zet de interruptblokkeervlag. Gewone (IRQ-)interrupts kunnen niet langer optreden. Niet-maskeerbare interrupts (NMI) blijven optreden, zoals hun naam al aangeeft.

Neveneffecten

- De I-vlag wordt gezet.

SEI : Set Interrupt Disable Flag **6502**

I ← 1

N	Z	I	C	D	V	E
·	·	+	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	SEI	78	1	2

SHA [onbedoeld]

OPMERKING: deze instructie is onstabiel op veel 6502-processoren en dient te worden vermeden.

Deze instructie slaat de bitgewijze EN van de inhoud van het accumulatorregister, het X-register en de derde byte van de instructie op in de aangegeven geheugenlocatie.

SHA : Store binary AND of A, X and 3rd Instruction Byte **6502**

M ← A *AND* X AND B3

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect),Y	SHA ($nn),Y	93	2	6
absolute,Y	SHA $nnnn,Y	9F	3	5

SHX [onbedoeld]

OPMERKING: deze instructie is onstabiel op veel 6502-processoren en dient te worden vermeden.

Deze instructie slaat de bitgewijze EN van de inhoud van het X-register en de derde byte van de instructie op in de aangegeven geheugenlocatie.

SHX : Store Binary AND of X Register and 3rd Instruction Byte **6502**

M ← X *AND* B3

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute,Y	SHX $nnnn,Y	9E	3	5

SHY [onbedoeld]

OPMERKING: deze instructie is onstabiel op veel 6502-processoren en dient te worden vermeden.

Deze instructie slaat de bitgewijze EN van de inhoud van het Y-register en de derde byte van de instructie op in de aangegeven geheugenlocatie.

SHY : Store Binary AND of Y Register and 3rd Instruction Byte **6502**

M ← Y *AND* B3

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute,X	SHY $nnnn,X	9C	3	5

SLO [onbedoeld]

Deze instructie schuift de inhoud van de opgegeven geheugenlocatie één bit naar links en voert daarna een OR-bewerking uit van het resultaat met het accumulatorregister en slaat het resultaat op in het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het accumulatorregister na voltooiing van de instructie $00 bevat, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de geheugeninhoud was ingesteld, voor het verschuiven.

SLO : Shift Left and OR **6502**

C ← M(7), M ← M≪1, A ← A *OR* M

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	SLO ($nn,X)	03	2	8
zero-page	SLO $nn	07	2	5
absolute	SLO $nnnn	0F	3	6
(indirect),Y	SLO ($nn),Y	13	2	8
zero-page,X	SLO $nn,X	17	2	6
absolute,Y	SLO $nnnn,Y	1B	3	7
absolute,X	SLO $nnnn,X	1F	3	7

SRE [onbedoeld]

Deze instructie schuift de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt op nul gezet en bit 0 wordt naar de carry-vlag uitgeschoven. Het resultaat wordt vervolgens met behulp van een uitsluitende-OF-bewerking (XOR) met het accumulatorregister gecombineerd en het eindresultaat wordt in het accumulatorregister opgeslagen.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 0 van de waarde was ingesteld, voor het verschuiven.

SRE : Logical Shift Right and Exclusive OR with Accumulator **6502**

C ← M(0), M ← M≫1, A ← A *XOR* M

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	SRE ($nn,X)	43	2	8
zero-page	SRE $nn	47	2	5
absolute	SRE $nnnn	4F	3	6
(indirect),Y	SRE ($nn),Y	53	2	8
zero-page,X	SRE $nn,X	57	2	6
absolute,Y	SRE $nnnn,Y	5B	3	7
absolute,X	SRE $nnnn,X	5F	3	7

STA

Deze instructie slaat de inhoud van het accumulatorregister op in de aangeduide locatie.

STA : Store Accumulator **6502**

M ← A

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	STA ($nn,X)	81	2	6
zero-page	STA $nn	85	2	3
absolute	STA $nnnn	8D	3	4
(indirect),Y	STA ($nn),Y	91	2	6
zero-page,X	STA $nn,X	95	2	4
absolute,Y	STA $nnnn,Y	99	3	5
absolute,X	STA $nnnn,X	9D	3	5

STX

Deze instructie slaat de inhoud van het X-register op in de aangeduide locatie.

STX : Store X Register **6502**

M ← X

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	STX $nn	86	2	3
absolute	STX $nnnn	8E	3	4
zero-page,Y	STX $nn,Y	96	2	4

STY

Deze instructie slaat de inhoud van het Y-register op in de aangeduide locatie.

STY : Store Y Register **6502**

M ← Y

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	STY $nn	84	2	3
absolute	STY $nnnn	8C	3	4
zero-page,X	STY $nn,X	94	2	4

TAS [onbedoeld]

OPMERKING: Dit monster van een instructie, behalve dat deze van elk denkbaar nut is ontdaan, is onstabiel op veel 6502-processoren en zou om die reden eveneens vermeden moeten worden, indien u er nog niet van afgeschrikt was.

Neveneffecten

- Opmerkelijk genoeg wijzigt het, ondanks de overgecompliceerde bewerking die het uitvoert, geen van de processorvlaggen.
- Verlies van verstand als u het probeert te gebruiken, of zelfs maar precies probeert uit te vogelen hoe het werkt.

TAS : Munge X Register and Stack Pointer **6502**

SP ← A *AND* X, M ← SP AND B3

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute,Y	TAS $nnnn,Y	9B	3	5

TAX

Deze instructie laadt het X-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet is, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

TAX : Transfer Accumulator Register into the X Register **6502**

X ← A

N	Z	I	C	D	V	E
+	+	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TAX	AA	1	2

TAY

Deze instructie laadt het Y-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet is, anders wordt de vlag gewist.

- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

TAY : Transfer Accumulator Register into the Y Register **6502**

Y ← A

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TAY	A8	1	2

TSX

Deze instructie laadt het X-register met de inhoud van het Stack Pointer Low (SPL)-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TSX : Transfer Stack Pointer Low Register into the X Register **6502**

X ← SPL

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TSX	BA	1	2

TXA

Deze instructie laadt het accumulatorregister met de inhoud van het X-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TXA : Transfer X Register into the Accumulator Register **6502**

A ← X

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TXA	8A	1	2

TXS

Deze instructie zet de lage byte van het stapelaanwijzerregister (SPL) op de inhoud van het X-register.

TXS : Transfer X Register into Stack Pointer Low Register **6502**

SPL ← X

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TXS	9A	1	2

TYA

Deze instructie laadt het accumulatorregister met de inhoud van het Y-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TYA : Transfer Y Register into the Accumulator Register **6502**

A ← Y

N	Z	I	C	D	V	E
+	+	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TYA	98	1	2

XAA [onbedoeld]

Deze instructie laadt het accumulatorregister met de binaire AND van het X-register en het immediate-modusargument.

OPMERKING: Deze instructie is op veel 6502-processors instabiel en mag niet gebruikt worden.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

XAA : Transfer X into A and AND with operand **6502**

A ← X *AND* VALUE

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	XAA #$nn	8B	2	2

DE INSTRUCTIESET VAN DE 4510

Instructietiming

Merk op dat het aantal cycli afhangt van de snelheidsinstelling van de processor: sommige instructies nemen meer of minder cycli wanneer de processor op volle snelheid draait, op een C65-compatibele 3.5 MHz-snelheid of op een C64-compatibele 1 MHz/2 MHz-snelheid. Meer gedetailleerde informatie hierover staat vermeld bij de beschrijving van elke instructie, maar het overzicht op hoofdlijnen is:

- Wanneer de processor op 1 MHz draait, duren alle instructies ten minste twee cycli en worden er dummy-cycli opnieuw ingevoegd in de Read-Modify-Write-instructies, zodat alle instructies exact hetzelfde aantal cycli duren als op een 6502.
- De Read-Modify-Write-instructies en alle instructies die een waarde uit het geheugen lezen, vereisen allemaal een extra cyclus wanneer zij op volle snelheid draaien, om de signalen de gelegenheid te geven zich binnen de processor te verplaatsen.
- De Read-Modify-Write-instructies vereisen een extra cyclus wanneer de operand $D019 is, omdat de dummy-schrijfoperatie in dat geval wordt uitgevoerd. Dit dient om de compatibiliteit te verbeteren met C64-software die deze "bug" van de 6502 vaak gebruikt om VIC-II-interrupts sneller te bevestigen.
- De strafcycli voor paginagrensoverschrijding en het nemen van branches zijn niet van toepassing wanneer de processor op volle snelheid draait.
- Veel instructies vereisen minder cycli wanneer de processor op volle snelheid draait, omdat de niet-bus-cycli over het algemeen worden verwijderd. Het plaatsen op en halen van waarden van en naar de stapel vereist slechts 2 cycli in plaats van de 4 die de 6502 voor dergelijke instructies nodig heeft.

Opcodetabel

De gekleurde cellen duiden een uitgebreide 45GS02-opcode aan. Een opcode met het pseudoregister Q is blauw gemarkeerd en een Base-Page-indirecte, Z-geïndexeerde opcode die gebruik kan maken van 32-bits pointers is cyaan.

	$x0	$x1	$x2
$0x	bytes *cyc* OPC modus	bytes *cyc* QOP Qmodus	bytes *cyc* FARQ Q lbpZ

De letters die aan het aantal cycli zijn toegevoegd, hebben de volgende betekenis:

	Betekenis
b	Voeg één cyclus toe als een branch wordt genomen. Voeg één cyclus extra toe als een genomen branch een paginagrens overschrijdt.
d	Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
i	Voeg één cyclus toe wanneer de kloksnelheid 40 MHz bedraagt.
m	Trek de niet-bus-cycli af wanneer de CPU op 40 MHz draait.
p	Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r	Voeg één cyclus toe wanneer de kloksnelheid 40 MHz bedraagt.
s	De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

Opcode-tabel 4510/45GS02

	$x0	$x1	$x2	$x3	$x4	$x5	$x6	$x7
$0x	1 7 BRK imp	2 6rp ORA bpiX	1 1 CLE imp	1 1 SEE imp	2 3r TSB bp	2 3r Q ORA bp	2 4r Q ASL bp	2 4rb RMB0 bp
$1x	2 2b BPL rel	2 5rp ORA ibpY	2 5rp Q ORA ibpZ	3 3b BPL relfar	2 5r TRB bp	2 3r ORA bpX	2 4r Q ASL bpX	2 4rb RMB1 bp
$2x	3 5s JSR abs	2 5rp AND bpiX	3 5r JSR ind	3 5rp JSR indX	2 3r Q BIT bp	2 3r AND bp	2 4r Q ROL bp	2 4r RMB2 bp
$3x	2 2r BMI rel	2 5rp AND ibpY	2 5rp Q AND ibpZ	3 3b BMI relfar	2 3rp BIT bpX	2 4rp AND bpX	2 5rp Q ROL bpX	2 4r RMB3 bp
$4x	1 6m RTI imp	2 5r EOR bpiX	1 1 NEG acc	1 1 Q ASR acc	2 4r ASR bp	2 3r EOR bp	2 4r Q LSR bp	2 4r RMB4 bp
$5x	2 2b BVC rel	2 5rp EOR ibpY	2 5rp Q EOR ibpZ	3 3b BVC relfar	2 5rp Q ASR bpX	2 3p EOR bpX	2 3rp Q LSR bpX	2 4r RMB5 bp
$6x	1 6m RTS imp	2 5r ADC bpiX	2 4 RTS imm	3 3b BSR relfar	2 3 STZ bp	2 3r Q ADC bp	2 5r Q ROR bp	2 5r RMB6 bp
$7x	2 2b BVS rel	2 5rp ADC ibpY	2 5rp Q ADC ibpZ	3 3b BVS relfar	2 3 STZ bpX	2 3r ADC bpX	2 5rmdp Q ROR bpX	2 4r RMB7 bp
$8x	2 2b BRA rel	2 5p STA bpiX	2 6p STA ispY	3 3b BRA relfar	2 3 STY bp	2 3 Q STA bp	2 3 STX bp	2 4r SMB0 bp
$9x	2 2b BCC rel	2 5p STA ibpY	2 5p Q STA ibpZ	3 3b BCC relfar	2 3p STY bpX	2 3p STA bpX	2 3p STX bpY	2 4r SMB1 bp
$Ax	2 2 LDY imm	2 5rp LDA bpiX	2 2 LDX imm	2 2 LDZ imm	2 3r LDY bp	2 3r Q LDA bp	2 3r LDX bp	2 4r SMB2 bp
$Bx	2 2b BCS rel	2 5rp LDA ibpY	2 5rp Q LDA ibpZ	3 3b BCS relfar	2 3rp LDY bpX	2 3rp LDA bpX	2 5rp LDX bpY	2 4r SMB3 bp
$Cx	2 2 CPY imm	2 5rp CMP bpiX	2 2 CPZ imm	2 7mdr DEW bp	2 3r CPY bp	2 3r Q CMP bp	2 5mdr Q DEC bp	2 4r SMB4 bp
$Dx	2 2b BNE rel	2 5rp CMP ibpY	2 5rp Q CMP ibpZ	3 3b BNE relfar	2 3r CPZ bp	2 3rp CMP bpX	2 5mdrp Q DEC bpX	2 4r SMB5 bp
$Ex	2 2 CPX imm	2 3pm SBC bpiX	2 6rmp LDA ispY	2 7mdr INW bp	2 3r CPX bp	2 3r Q SBC bp	2 5mdr Q INC bp	2 4r SMB6 bp
$Fx	2 2b BEQ rel	2 3rp SBC ibpY	2 5rp Q SBC ibpZ	3 3b BEQ relfar	3 5m PHW im16	2 3rp SBC bpX	2 5dmrp Q INC bpX	2 4r SMB7 bp

Opcode-tabel 4510/45GS02

$x8	$x9	$xA	$xB	$xC	$xD	$xE	$xF	
1 PHP 2 imp	2 ORA 2 imm	1 ASL 1 Q acc	1 TSY 1 imp	3 TSB 5r abs	3 ORA 4r Q abs	3 ASL 5r Q abs	3 BBR0 0rb bpr8	$0x
1 CLC 1 imp	3 ORA 4r absY	1 INC 1 Q acc	1 INZ 1 imp	3 TRB 4r abs	3 ORA 4rp absX	3 ASL 5rp Q absX	3 BBR1 5b bpr8	$1x
1 PLP 4m imp	2 AND 2 imm	1 ROL 1 Q acc	1 TYS 1 imp	3 BIT 4r Q abs	3 AND 4r Q abs	3 ROL 5r Q abs	3 BBR2 5b bpr8	$2x
1 SEC 1 imp	3 AND 4r absY	1 DEC 1 Q acc	1 DEZ 1 imp	3 BIT 4rp absX	3 AND 4rp absX	3 ROL 5rp Q absX	3 BBR3 4b bpr8	$3x
1 PHA 2 imp	2 EOR 2 imm	1 LSR 1 Q acc	1 TAZ 1 imp	3 JMP 3 abs	3 EOR 4r Q abs	3 LSR 5r Q abs	3 BBR4 4rb bpr8	$4x
1 CLI 1 imp	3 EOR 4rp absY	1 PHY 2 imp	1 TAB 1 imp	1 MAP 1 imp	3 EOR 4rp absX	3 LSR 5rp Q absX	3 BBR5 4rb bpr8	$5x
1 PLA 4m imp	2 ADC 2 imm	1 ROR 1 Q acc	1 TZA 1 imp	3 JMP 5r ind	3 ADC 4r Q abs	3 ROR 6r Q abs	3 BBR6 4rb bpr8	$6x
1 SEI 1s imp	3 ADC 4r absY	1 PLY 4m imp	1 TBA 1 imp	3 JMP 6mp indX	3 ADC 4r absX	3 ROR 5rmdp Q absX	3 BBR7 4br bpr8	$7x
1 DEY 1s imp	2 BIT 2 imm	1 TXA 1 imp	3 STY 4p absX	3 STY 4 abs	3 STA 4 Q abs	3 STX 4 abs	3 BBS0 4br bpr8	$8x
1 TYA 1 imp	3 STA 4p absY	1 TXS 1 imp	3 STX 4p absY	3 STZ 4 abs	3 STA 4p absX	3 STZ 4p absX	3 BBS1 4br bpr8	$9x
1 TAY 1 imp	2 LDA 2 imm	1 TAX 1 imp	3 LDZ 4r abs	3 LDY 4r abs	3 LDA 4r Q abs	3 LDX 4r abs	3 BBS2 4br bpr8	$Ax
1 CLV 1 imp	3 LDA 4rp absY	1 TSX 1 imp	3 LDZ 4rp absX	3 LDY 4rp absX	3 LDA 4rp absX	3 LDX 4rp absY	3 BBS3 4br bpr8	$Bx
1 INY 1s imp	2 CMP 2 imm	1 DEX 1s imp	3 ASW 7rmd abs	3 CPY 4r abs	3 CMP 4r Q abs	3 DEC 6mdr Q abs	3 BBS4 4br bpr8	$Cx
1 CLD 1 imp	3 CMP 4rp absY	1 PHX 3m imp	1 PHZ 3m imp	3 CPZ 4r abs	3 CMP 4rp absX	3 DEC 6mdrp Q absX	3 BBS5 4br bpr8	$Dx
1 INX 1s imp	2 SBC 2 imm	1 EOM 1 imp	3 ROW 5rmd abs	3 CPX 4r abs	3 SBC 4r Q abs	3 INC 6dmr Q abs	3 BBS6 4br bpr8	$Ex
1 SED 1s imp	3 SBC 4rp absY	1 PLX 4m imp	1 PLZ 4m imp	3 PHW 7m abs	3 SBC 4rp absX	3 INC 6drp Q absX	3 BBS7 4br bpr8	$Fx

ADC

Deze instructie telt het argument en de carry-vlag op bij de inhoud van het accumulatorregister. Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten, anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De C-vlag wordt gezet als het ongetekende resultaat >255 is, of >99 indien de D-vlag is gezet.

ADC : Add with carry **4510**

$A \leftarrow A + M + C$

			N	Z	I	C	D	V	E	
			+	+	·	+	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect,X)	ADC ($nn,X)	61	2	5	r
base-page	ADC $nn	65	2	3	r
immediate 8bit	ADC #$nn	69	2	2	
absolute	ADC $nnnn	6D	3	4	r
(indirect),Y	ADC ($nn),Y	71	2	5	pr
(indirect),Z	ADC ($nn),Z	72	2	5	pr
base-page,X	ADC $nn,X	75	2	3	r
absolute,Y	ADC $nnnn,Y	79	3	4	r
absolute,X	ADC $nnnn,X	7D	3	4	r

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

AND

Deze instructie voert een bitgewijze EN-operatie uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister en die ook in het argument staan, worden na voltooiing ingesteld in het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

AND : Binary AND **4510**

A ← A *AND* M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect,X)	AND ($nn,X)	21	2	5	pr
base-page	AND $nn	25	2	3	r
immediate 8bit	AND #$nn	29	2	2	
absolute	AND $nnnn	2D	3	4	r
(indirect),Y	AND ($nn),Y	31	2	5	pr
(indirect),Z	AND ($nn),Z	32	2	5	pr
base-page,X	AND $nn,X	35	2	4	pr
absolute,Y	AND $nnnn,Y	39	3	4	r
absolute,X	AND $nnnn,X	3D	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ASL

Deze instructie verschuift het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt op nul gezet en bit 7 wordt verschoven naar de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ASL : Arithmetic Shift Left Memory or Accumulator **4510**

A ← A≪1 of M ← M≪1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	ASL $nn	06	2	4	r
accumulator	ASL A	0A	1	1	s
absolute	ASL $nnnn	0E	3	5	r
base-page,X	ASL $nn,X	16	2	4	r
absolute,X	ASL $nnnn,X	1E	3	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ASR

Deze instructie schuift de inhoud van het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt als tekenbit beschouwd en behouden. De inhoud van bit 0 wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 0 van de waarde voor het schuiven was gezet.

ASR : Arithmetic Shift Right **4510**

A ← A>>1 of M ← M>>1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
accumulator	ASR A	43	1	1	s
base-page	ASR $nn	44	2	4	r
base-page,X	ASR $nn,X	54	2	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ASW

Deze instructie schuift een 16-bits waarde in het geheugen één bit naar links.

Indien adres $1234 de waarde $87 bevat en adres $1235 de waarde $A9 bevat, geeft ASW $1234 als resultaat dat adres $1234 de waarde $0E bevat en adres $1235 de waarde $53, en wordt de carry-vlag gezet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de hoge byte voor het verschuiven gezet was; anders wordt de vlag gewist.

ASW : Arithmetic Shift Word Left **4510**

$M \leftarrow M \ll 1$

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute	ASW $nnnn	CB	3	7	dmr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR0

Deze instructie vertakt naar het opgegeven adres indien bit 0 in de opgegeven basispagina-locatie nul is.

BBR0 : Branch on Bit 0 Reset **4510**

$M(0)=0 \Longrightarrow PC \leftarrow PC + R8$

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR0 $nn,$rr	0F	3	0	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR1

Deze instructie vertakt naar het opgegeven adres indien bit 1 in de opgegeven basispagina-locatie nul is.

BBR1 : Branch on Bit 1 Reset **4510**

M(1)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
base-page+rel	BBR1 $nn,$rr	1F	3	5 ^{b}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BBR2

Deze instructie vertakt naar het opgegeven adres indien bit 2 in de opgegeven basispagina-locatie nul is.

BBR2 : Branch on Bit 2 Reset **4510**

M(2)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
base-page+rel	BBR2 $nn,$rr	2F	3	5 ^{b}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BBR3

Deze instructie vertakt naar het opgegeven adres indien bit 3 in de opgegeven basispagina-locatie nul is.

BBR3 : Branch on Bit 3 Reset **4510**

M(3)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
base-page+rel	BBR3 $nn,$rr	3F	3	4 ^{b}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BBR4

Deze instructie vertakt naar het opgegeven adres indien bit 4 in de opgegeven basispagina-locatie nul is.

BBR4 : Branch on Bit 4 Reset **4510**

M(4)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR4 $nn,$rr	4F	3	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR5

Deze instructie vertakt naar het opgegeven adres indien bit 5 in de opgegeven basispagina-locatie nul is.

BBR5 : Branch on Bit 5 Reset **4510**

M(5)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR5 $nn,$rr	5F	3	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR6

Deze instructie vertakt naar het opgegeven adres indien bit 6 in de opgegeven basispagina-locatie nul is.

BBR6 : Branch on Bit 6 Reset **4510**

M(6)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR6 $nn,$rr	6F	3	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR7

Deze instructie vertakt naar het opgegeven adres indien bit 7 in de opgegeven basispagina-locatie nul is.

BBR7 : Branch on Bit 7 Reset **4510**

M(7)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR7 $nn,$rr	7F	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS0

Deze instructie vertakt naar het opgegeven adres indien bit 0 in de opgegeven basispagina-locatie één is.

BBS0 : Branch on Bit 0 Set **4510**

M(0)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS0 $nn,$rr	8F	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS1

Deze instructie vertakt naar het opgegeven adres indien bit 1 in de opgegeven basispagina-locatie één is.

BBS1 : Branch on Bit 1 Set **4510**

M(1)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS1 $nn,$rr	9F	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS2

Deze instructie vertakt naar het opgegeven adres indien bit 2 in de opgegeven basispagina-locatie één is.

BBS2 : Branch on Bit 2 Set **4510**

M(2)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS2 $nn,$rr	AF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS3

Deze instructie vertakt naar het opgegeven adres indien bit 3 in de opgegeven basispagina-locatie één is.

BBS3 : Branch on Bit 3 Set **4510**

M(3)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS3 $nn,$rr	BF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS4

Deze instructie vertakt naar het opgegeven adres indien bit 4 in de opgegeven basispagina-locatie één is.

BBS4 : Branch on Bit 4 Set **4510**

M(4)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS4 $nn,$rr	CF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS5

Deze instructie vertakt naar het opgegeven adres indien bit 5 in de opgegeven basispagina-locatie één is.

BBS5 : Branch on Bit 5 Set **4510**

M(5)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS5 $nn,$rr	DF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS6

Deze instructie vertakt naar het opgegeven adres indien bit 6 in de opgegeven basispagina-locatie één is.

BBS6 : Branch on Bit 6 Set **4510**

M(6)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS6 $nn,$rr	EF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS7

Deze instructie vertakt naar het opgegeven adres indien bit 7 in de opgegeven basispagina-locatie één is.

BBS7 : Branch on Bit 7 Set **4510**

M(7)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS7 $nn,$rr	FF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BCC

Deze instructie vertakt naar het aangegeven adres als de carry-vlag gewist is.

BCC : Branch on Carry Flag Clear **4510**

C=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BCC $rr	90	2	2	b
16-bit relative	BCC $rrrr	93	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BCS

Deze instructie vertakt naar het aangegeven adres als de carry-vlag gezet is.

BCS : Branch on Carry Flag Set **4510**

C=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BCS $rr	B0	2	2	b
16-bit relative	BCS $rrrr	B3	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BEQ

Deze instructie vertakt naar het aangegeven adres als de zero-vlag gezet is. BEQ staat voor branch if equal, omdat een CMP ertoe leidt dat de zero-vlag gezet wordt als de operanden gelijk zijn.

BEQ : Branch on Zero Flag Set **4510**

Z=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BEQ $rr	F0	2	2	b
16-bit relative	BEQ $rrrr	F3	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BIT

Deze instructie wordt gebruikt om de bits te testen die zijn opgeslagen op een geheugenlocatie of in het immediate-argument van de opcode.

Bits 6 en 7 van de inhoud van de geheugenlocatie worden rechtstreeks gekopieerd naar de overflow-vlag en de negatieve-vlag. De zero-vlag wordt gezet of gewist op basis van het resultaat van de binaire AND van het accumulatorregister en de inhoud van de aangegeven geheugenlocatie.

De immediate-test zet de N- en V-vlaggen met geldige toestanden, waarbij het argument wordt behandeld als de geheugenwaarde, wat bij de eerdere 65C02-implementatie niet het geval was.

Neveneffecten

- De N-vlag wordt gezet als bit 7 van de geheugenlocatie gezet is, anders wordt hij gewist.
- De V-vlag wordt gezet als bit 6 van de geheugenlocatie gezet is, anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat van A *AND* M nul is, anders wordt hij gewist.

BIT : Perform Bit Test **4510**

N ← M(7), V ← M(6), Z ← A *AND* M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page	BIT $nn	24	2	3	r
absolute	BIT $nnnn	2C	3	4	r
base-page,X	BIT $nn,X	34	2	3	pr
absolute,X	BIT $nnnn,X	3C	3	4	pr
immediate 8bit	BIT #$nn	89	2	2	

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BMI

Deze instructie vertakt naar het aangegeven adres als de negatieve-vlag gezet is. BMI staat voor branch on minus.

BMI : Branch on Negative Flag Set **4510**

N=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BMI $rr	30	2	2	r
16-bit relative	BMI $rrrr	33	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BNE

Deze instructie vertakt naar het aangegeven adres als de zero-vlag gewist is. BNE staat voor branch if not equal, omdat een CMP ertoe leidt dat de zero-vlag gewist wordt als de operanden niet gelijk zijn.

BNE : Branch on Zero Flag Clear **4510**

Z=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BNE $rr	D0	2	2	b
16-bit relative	BNE $rrrr	D3	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BPL

Deze instructie vertakt naar het aangegeven adres als de negatieve-vlag gewist is. BPL staat voor branch on plus.

BPL : Branch on Negative Flag Clear **4510**

N=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BPL $rr	10	2	2	b
16-bit relative	BPL $rrrr	13	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BRA

Deze instructie vertakt naar het opgegeven adres.

BRA : Branch Unconditionally **4510**

PC ← PC + R8 *of* PC ← PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BRA $rr	80	2	2	*b*
16-bit relative	BRA $rrrr	83	3	3	*b*

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BRK

terruptserviceroutine onderscheid maken tussen IRQ-gebeurtenissen en BRK-gebeurtenissen. Bijvoorbeeld:

```
PLA           ; laad het statusregister
PHA           ; herstel de stapel
AND #$10      ; maskeer de break-vlag
BNE DO_BREAK  ; -> het was een BRK
...           ; anders door met de IRQ-server
```

Geciteerd uit: MCS6500 Microcomputer Family Programming Manual, januari 1976, tweede editie, MOS Technology Inc., blz. 144:

"De BRK is een instructie van één byte en de adresseermethode ervan is Implied."

Er wordt gediscussieerd over de vraag of BRK kan worden beschouwd als een instructie van twee bytes met de immediate-adresseermethode, waarbij het operand-byte wordt weggegooid. Het byte na de BRK zou dan als oproepargument voor de break-handler kunnen dienen. Commodore gebruikte echter de BRK, zoals in de handleiding vermeld, als een instructie van één byte, die in de ML-monitor breekt als deze aanwezig is. Deze ingebouwde monitors verlaagden de op de stapel geplaatste PC, zodat deze kon worden gebruikt om direct terug te keren naar of te springen naar het code-byte na de BRK.

BRK : SP ← SP − 2, STAPEL ← PC + 2, PC ← ($FFFE) **4510**

M+

N	Z	I	C	D	V	E
.	.	.	.	**e**	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	BRK	00	1	7

BSR

Deze instructie vertakt naar het opgegeven adres en slaat het adres van de volgende instructie op de stapel op, zodat met een RTS-instructie uit de routine kan worden teruggekeerd.

Deze instructie is nuttig bij het gebruik van verplaatsbare code, omdat het een relatief geadresseerd alternatief voor JSR biedt.

BSR : Branch Sub-Routine **4510**

SP ← SP − 2, STAPEL ← PC + len(V), PC ← PC + V

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
16-bit relative	BSR $rrrr	63	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BVC

Deze instructie vertakt naar het aangegeven adres als de overflow-vlag (V) gewist is.

BVC : Branch on Overflow Flag Clear **4510**

V=0 $\Longrightarrow$ PC ← PC + R8 of PC ← PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BVC $rr	50	2	2	b
16-bit relative	BVC $rrrr	53	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BVS

Deze instructie vertakt naar het aangegeven adres als de overflow-vlag (V) gezet is.

BVS : Branch on Overflow Flag Set **4510**

V=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BVS $rr	70	2	2	b
16-bit relative	BVS $rrrr	73	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

CLC

Deze instructie wist de carry-vlag.

Neveneffecten

- De C-vlag wordt gewist.

CLC : Clear Carry Flag **4510**

C $\leftarrow$ 0

N	Z	I	C	D	V	E
·	·	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLC	18	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CLD

Deze instructie wist de decimalenvlag. Rekenkundige bewerkingen maken dan gebruik van gewoon binair rekenen, in plaats van met BCD-gewogen decimale getallen (BCD).

Neveneffecten

- De D-vlag wordt gewist.

CLD : Clear Decimal Flag **4510**

D $\leftarrow$ 0

N	Z	I	C	D	V	E
·	·	·	·	+	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLD	D8	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CLE

Deze instructie wist de Extended Stack Disable-vlag. Hierdoor kan de stapel in lengte meer dan 256 bytes omvatten, doordat de processor de waarde van de hoge byte van het stapeladres (SPH) kan wijzigen.

Neveneffecten

- De E-vlag wordt gewist.

CLE : Clear Extended Stack Disable Flag **4510**

E ← 0

N	Z	I	C	D	V	E
·	·	·	·	·	·	+

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLE	02	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CLI

Deze instructie wist de interruptblokkeervlag. Interrupts kunnen nu plaatsvinden.

Neveneffecten

- De I-vlag wordt gewist.

CLI : Clear Interrupt Disable Flag **4510**

I ← 0

N	Z	I	C	D	V	E
·	·	+	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLI	58	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CLV

Deze instructie wist de overflow-vlag.

Neveneffecten

- De V-vlag wordt gewist.

CLV : Clear Overflow Flag **4510**

V ← 0

N	Z	I	C	D	V	E
·	·	·	·	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLV	B8	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CMP

Deze instructie voert A − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt deze gewist.

CMP : Compare Accumulator **4510**

N,C,Z ⇐ [A − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	CMP ($nn,X)	C1	2	5	pr
base-page	CMP $nn	C5	2	3	r
immediate 8bit	CMP #$nn	C9	2	2	
absolute	CMP $nnnn	CD	3	4	r
(indirect),Y	CMP ($nn),Y	D1	2	5	pr
(indirect),Z	CMP ($nn),Z	D2	2	5	pr
base-page,X	CMP $nn,X	D5	2	3	pr
absolute,Y	CMP $nnnn,Y	D9	3	4	pr
absolute,X	CMP $nnnn,X	DD	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CPX

Deze instructie voert X − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van X − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van X − M nul of positief is, d.w.z. als X niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van X − M nul is; anders wordt deze gewist.

CPX : Compare X Register **4510**

N,C,Z ⇐ [X − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	CPX #$nn	E0	2	2	
base-page	CPX $nn	E4	2	3	r
absolute	CPX $nnnn	EC	3	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CPY

Deze instructie voert Y − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Y − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van Y − M nul of positief is, d.w.z. als Y niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van Y − M nul is; anders wordt deze gewist.

CPY : Compare Y Register **4510**

N,C,Z ⇐ [Y − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	CPY #$nn	C0	2	2	
base-page	CPY $nn	C4	2	3	r
absolute	CPY $nnnn	CC	3	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CPZ

Deze instructie voert Z − M uit en zet de processorvlaggen overeenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Z − M negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van Z − M nul of positief is, d.w.z. als Z niet kleiner is dan M; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat van Z − M nul is; anders wordt de vlag gewist.

CPZ : Compare Z Register **4510**

N,C,Z ⇐ [Z − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	CPZ #$nn	C2	2	2	
base-page	CPZ $nn	D4	2	3	*r*
absolute	CPZ $nnnn	DC	3	4	*r*

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

DEC

Deze instructie verlaagt het accumulatorregister of de aangegeven geheugenlocatie met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEC : Decrement **4510**

A ← A − 1 *of* M ← M − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
accumulator	DEC A	3A	1	1	*s*
base-page	DEC $nn	C6	2	5	*dmr*
absolute	DEC $nnnn	CE	3	6	*dmr*
base-page,X	DEC $nn,X	D6	2	5	*dmpr*
absolute,X	DEC $nnnn,X	DE	3	6	*dmpr*

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

DEW

Deze instructie verlaagt het aangegeven geheugenwoord in de basispagina met één. Het laagst genummerde adres bevat de minst significante bits. Wanneer geheugenlocatie $12 bijvoorbeeld $78 bevat en geheugenlocatie $13 $56, zorgt de instructie DEW $12 ervoor dat geheugenlocatie $12 de waarde $77 krijgt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

DEW : Decrement Memory Word **4510**

M16 ← M16 − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	DEW $nn	C3	2	7	*dmr*

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

DEX

Deze instructie verlaagt het X-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEX : Decrement X Register **4510**

$X \leftarrow X - 1$

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	DEX	CA	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

DEY

Deze instructie verlaagt het Y-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEY : Decrement Y Register **4510**

$Y \leftarrow Y - 1$

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	DEY	88	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

DEZ

Deze instructie verlaagt het Z-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

DEZ : Decrement Z Register **4510**

$Z \leftarrow Z - 1$

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	DEZ	3B	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

EOM

Anders dan op de 6502 vervult de NOP-instructie op de 45GS02 twee extra rollen wanneer de processor in de 4502-modus draait.

Ten eerste geeft de instructie het einde aan van een geheugenmappingreeks die door een MAP-instructie is veroorzaakt, waardoor interrupts opnieuw kunnen optreden.

Ten tweede instrueert hij de processor om een 32-bits pointer te gebruiken in plaats van een 16-bits pointer in 6502-stijl, indien de volgende instructie de adresseermethode Base-Page Indirect Z Indexed gebruikt. Dergelijke 32-bits adressen worden niet beïnvloed door de geheugenbanken van de C64, C65 of MEGA65. Dit maakt snelle en eenvoudige toegang tot de volledige adresruimte van de MEGA65 mogelijk, zonder dat u enige vorm van banking hoeft uit te voeren, daarvan bewust hoeft te zijn of de DMA-controller hoeft te gebruiken. Deze adresseermethode brengt twee extra cycli met zich mee, vanwege de tijd die nodig is om de twee extra bytes van de pointer te lezen.

OPMERKING: wees voorzichtig wanneer u EOM/NOP na een Hypervisor Call gebruikt voor een vertraging, omdat dit uw volgende instructie zou kunnen wijzigen. CLV kan als alternatief worden gebruikt.

Neveneffecten

- Heft het door de MAP-instructie opgelegde verbod op alle interrupts op, waardoor niet-maskeerbare interrupts weer kunnen optreden en tevens IRQ-interrupts, indien de interruptblokkeervlag niet gezet is.

EOM : End of Mapping Sequence / No-Operation **4510**

Speciaal

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	EOM	EA	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

EOR

Deze instructie voert een bitgewijze uitsluitende-OF-operatie (XOR) uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister of die ingesteld staan in het argument, worden na voltooiing ingesteld in het accumulatorregister, maar niet allebei.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

EOR : Binary Exclusive OR **4510**

A ← A *XOR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	EOR ($nn,X)	41	2	5	r
base-page	EOR $nn	45	2	3	r
immediate 8bit	EOR #$nn	49	2	2	
absolute	EOR $nnnn	4D	3	4	r
(indirect),Y	EOR ($nn),Y	51	2	5	pr
(indirect),Z	EOR ($nn),Z	52	2	5	pr
base-page,X	EOR $nn,X	55	2	3	p
absolute,Y	EOR $nnnn,Y	59	3	4	pr
absolute,X	EOR $nnnn,X	5D	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

INC

Deze instructie verhoogt het accumulatorregister of de aangegeven geheugenlocatie met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INC : Increment Memory or Accumulator **4510**

A ← A + 1 *of* M ← M + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
accumulator	INC A	1A	1	1	s
base-page	INC $nn	E6	2	5	dmr
absolute	INC $nnnn	EE	3	6	dmr
base-page,X	INC $nn,X	F6	2	5	$dmpr$
absolute,X	INC $nnnn,X	FE	3	6	dpr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

INW

Deze instructie verhoogt het aangegeven geheugenwoord in de basispagina met één. Het laagst genummerde adres bevat de minst significante bits. Wanneer geheugenlocatie $12 bijvoorbeeld $78 bevat en geheugenlocatie $13 $56, zorgt de instructie INW $12 ervoor dat geheugenlocatie $12 de waarde $79 krijgt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

INW : Increment Memory Word **4510**

M16 ← M16 + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	INW $nn	E3	2	7	dmr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

INX

Deze instructie verhoogt het X-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INX : Increment X Register **4510**

X ← X + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	INX	E8	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

INY

Deze instructie verhoogt het Y-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INY : Increment Y Register **4510**

Y ← Y + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	INY	C8	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

INZ

Deze instructie verhoogt het Z-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

INZ : Increment Z Register **4510**

Z ← Y + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	INZ	1B	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

JMP

Deze instructie stelt het Program Counter (PC)-register in op het door de instructie aangegeven adres, waardoor de uitvoering vanaf dat adres wordt voortgezet.

JMP : Jump to Address **4510**

PC ← M2:M1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute	JMP $nnnn	4C	3	3	
(indirect)	JMP ($nnnn)	6C	3	5	r
(indirect,X)	JMP ($nnnn,X)	7C	3	6	mp

m Trek de niet-buscycli af bij 40 MHz.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

JSR

Deze instructie slaat het adres van de instructie na de JSR-instructie op in de stapel en stelt vervolgens het Program Counter (PC)-register in op het door de instructie aangegeven adres, waardoor de uitvoering vanaf dat adres wordt voortgezet. Omdat het terugkeeradres in de stapel is opgeslagen, kan de RTS-instructie worden gebruikt om vanuit de aangeroepen subroutine terug te keren en de uitvoering na de JSR-instructie voort te zetten.

OPMERKING: deze instructie plaatst in feite het adres van de laatste byte van de JSR-instructie op de stapel. De RTS-instructie weet dit uiteraard en verhoogt het adres wanneer het van de stapel wordt gehaald, voordat het Program Counter-register (PC) wordt ingesteld.

JSR : Jump to Sub-Routine **4510**

SP ← SP − 2, STAPEL ← PCH:PCL, PC ← M2:M1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute	JSR $nnnn	20	3	5	
(indirect)	JSR ($nnnn)	22	3	5	r
(indirect,X)	JSR ($nnnn,X)	23	3	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDA

Deze instructie laadt het accumulatorregister met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.

LDA : Load Accumulator **4510**

A ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	LDA ($nn,X)	A1	2	5	pr
base-page	LDA $nn	A5	2	3	r
immediate 8bit	LDA #$nn	A9	2	2	
absolute	LDA $nnnn	AD	3	4	r
(indirect),Y	LDA ($nn),Y	B1	2	5	pr
(indirect),Z	LDA ($nn),Z	B2	2	5	pr
base-page,X	LDA $nn,X	B5	2	3	pr
absolute,Y	LDA $nnnn,Y	B9	3	4	pr
absolute,X	LDA $nnnn,X	BD	3	4	pr
(immediate,SP),Y	LDA ($nn,SP),Y	E2	2	6	mpr

m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDX

Deze instructie laadt het X-register met de opgegeven waarde, of met de inhoud van de opgegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

LDX : Load X Register **4510**

X ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDX #$nn	A2	2	2	
base-page	LDX $nn	A6	2	3	r
absolute	LDX $nnnn	AE	3	4	r
base-page,Y	LDX $nn,Y	B6	2	5	pr
absolute,Y	LDX $nnnn,Y	BE	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDY

Deze instructie laadt het Y-register met de opgegeven waarde, of met de inhoud van de opgegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

LDY : Load Y Register **4510**

Y ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDY #$nn	A0	2	2	
base-page	LDY $nn	A4	2	3	r
absolute	LDY $nnnn	AC	3	4	r
base-page,X	LDY $nn,X	B4	2	3	pr
absolute,X	LDY $nnnn,X	BC	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDZ

Deze instructie laadt het Z-register met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

LDZ : Load Z Register **4510**

Z ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDZ #$nn	A3	2	2	
absolute	LDZ $nnnn	AB	3	4	r
absolute,X	LDZ $nnnn,X	BB	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LSR

Deze instructie verschuift ofwel het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt op nul gezet en bit 0 wordt naar de carry-vlag uitgeschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.
- De C-vlag wordt gezet als bit 0 van de waarde gezet was voordat de waarde is verschoven.

LSR : Logical Shift Right **4510**

A ← A≫1, C ← A(0) of M ← M≫1, C ← M(0)

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	LSR $nn	46	2	4	r
accumulator	LSR A	4A	1	1	s
absolute	LSR $nnnn	4E	3	5	r
base-page,X	LSR $nn,X	56	2	3	pr
absolute,X	LSR $nnnn,X	5E	3	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

MAP

Deze instructie stelt de geheugenindeling in de C65- of MEGA65-stijl in, afhankelijk van de waarden in het accumulatorregister en de X-, Y- en Z-registers.

Het is een goede gewoonte om geen MAP-instructie uit te voeren in een geheugenregio waarvan de geheugenindeling verandert. De CPU wijzigt de geheugeninstellingen onmiddellijk wanneer de MAP-instructie wordt uitgevoerd. De programmateller gebruikt de nieuwe geheugenindeling om de volgende instructie te vinden.

OPMERKING: De MAP-instructie schakelt interrupts uit, ongeacht de I-vlag, tot aan de volgende EOM-instructie. EOM (byte EA) wordt daarnaast gebruikt als samengestelde instructie-prefix voor 32-bit indirect adresseren. Een dergelijke instructie gedraagt zich na een MAP-instructie als een EOM-instructie.

Meer informatie over deze instructie vindt u in bijlage K.

Neveneffecten

- De geheugenindeling wordt onmiddellijk gewijzigd in de aangevraagde.
- Alle interrupts, waaronder niet-maskeerbare interrupts (NMI's), worden geblokkeerd totdat een EOM-instructie (NOP) wordt aangetroffen.

MAP : Set Memory Map				**4510**
Speciaal				
		N Z I C D V E		
		. . . .		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	MAP	5C	1	1 ^{s}

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

NEG

Deze instructie vervangt de inhoud van het accumulatorregister door het tweecomplement van de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

NEG : Negate Accumulator				**4510**
A $\leftarrow$ (A *XOR* $FF) + 1				
		N Z I C D V E		
		+ + ...		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
accumulator	NEG A	42	1	1 ^{s}

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ORA

Deze instructie voert een binaire OR-bewerking uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al in het accumulatorregister gezet waren, of die in het argument gezet waren, worden bij voltooiing in het accumulatorregister gezet, of beide.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

ORA : Binary OR **4510**

A ← A *OR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	ORA ($nn,X)	01	2	6	pr
base-page	ORA $nn	05	2	3	r
immediate 8bit	ORA #$nn	09	2	2	
absolute	ORA $nnnn	0D	3	4	r
(indirect),Y	ORA ($nn),Y	11	2	5	pr
(indirect),Z	ORA ($nn),Z	12	2	5	pr
base-page,X	ORA $nn,X	15	2	3	r
absolute,Y	ORA $nnnn,Y	19	3	4	r
absolute,X	ORA $nnnn,X	1D	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

PHA

Deze instructie duwt de inhoud van het accumulatorregister op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHA : Push Accumulator Register onto the Stack **4510**

SP ← SP − 1, STAPEL ← A

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHA	48	1	2

PHP

Deze instructie duwt de inhoud van de processorvlaggen op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHP : Push Processor Flags onto the Stack **4510**

SP ← SP − 1, STAPEL ← P

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHP	08	1	2

PHW

Deze instructie schuift een 16-bits letterlijke waarde of het aangegeven geheugenwoord op de stapel en verlaagt de waarde van de stapelaanwijzer met 2.

OPMERKING: anders dan JSR, waarbij de meest significante byte op het hogere stapeladres achterblijft, plaatst PHW eerst de minst significante byte en daarna de meest significante byte. Het woord ligt dus op de stack met de meest significante byte op het lagere adres (big-endian) en kan daarom niet rechtstreeks worden gebruikt als little-endian-pointer met stapelrelatieve adressering. Zo laat PHW #$1234 $12 op SP+1 en $34 op SP+2 achter.

PHW : Push Word onto the Stack **4510**

SP ← SP − 2, STAPEL ← M1:M2

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 16bit	PHW #$nnnn	F4	3	5	m
absolute	PHW $nnnn	FC	3	7	m

m Trek de niet-buscycli af bij 40 MHz.

PHX

Deze instructie schuift de inhoud van het X-register op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHX : Push X Register onto the Stack **4510**

SP ← SP − 1, STAPEL ← X

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PHX	DA	1	3	m

m Trek de niet-buscycli af bij 40 MHz.

PHY

Deze instructie schuift de inhoud van het Y-register op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHY : Push Y Register onto the Stack **4510**

SP ← SP − 1, STAPEL ← Y

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHY	5A	1	2

PHZ

Deze instructie schuift de inhoud van het Z-register op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHZ : Push Z Register onto the Stack **4510**

SP $\leftarrow$ SP $-$ 1, STAPEL $\leftarrow$ Z

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PHZ	DB	1	3	m

m Trek de niet-buscycli af bij 40 MHz.

PLA

Deze instructie vervangt de inhoud van het accumulatorregister door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLA : Pull Accumulator Register from the Stack **4510**

A $\leftarrow$ STAPEL, SP $\leftarrow$ SP $+$ 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLA	68	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

PLP

Deze instructie vervangt de inhoud van de processorvlaggen door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

OPMERKING: deze instructie vervangt NIET de 'Extended Stack Disable'-vlag (de E-vlag) of de software-interruptvlag (de B-vlag)

PLP : Pull Processor Flags from the Stack **4510**

P $\leftarrow$ STAPEL, SP $\leftarrow$ SP $+$ 1

N	Z	I	C	D	V	E
+	+	+	+	+	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLP	28	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

PLX

Deze instructie vervangt de inhoud van het X-register door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLX : Pull X Register from the Stack **4510**

X ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLX	FA	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

PLY

Deze instructie vervangt de inhoud van het Y-register door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLY : Pull Y Register from the Stack **4510**

Y ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLY	7A	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

PLZ

Deze instructie vervangt de inhoud van het Z-register door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLZ : Pull Z Register from the Stack **4510**

Z ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLZ	FB	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

RMB0

Deze instructie wist bit nul van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB0 : Reset Bit 0 in Base Page **4510**

M(0) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB0 $nn	07	2	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB1

Deze instructie wist bit 1 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB1 : Reset Bit 1 in Base Page **4510**

M(1) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB1 $nn	17	2	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB2

Deze instructie wist bit 2 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB2 : Reset Bit 2 in Base Page **4510**

M(2) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB2 $nn	27	2	4	^{r}

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB3

Deze instructie wist bit 3 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB3 : Reset Bit 3 in Base Page **4510**

M(3) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB3 $nn	37	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB4

Deze instructie wist bit 4 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB4 : Reset Bit 4 in Base Page **4510**

M(4) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB4 $nn	47	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB5

Deze instructie wist bit 5 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB5 : Reset Bit 5 in Base Page **4510**

M(5) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB5 $nn	57	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB6

Deze instructie wist bit 6 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB6 : Reset Bit 6 in Base Page				**4510**
M(6) ← 0				
			N Z I C D V E	
			. . . .	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
base-page	RMB6 $nn	67	2	5 ^{r}

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB7

Deze instructie wist bit 7 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB7 : Reset Bit 7 in Base Page				**4510**
M(7) ← 0				
			N Z I C D V E	
			. . . .	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
base-page	RMB7 $nn	77	2	4 ^{r}

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ROL

Deze instructie verschuift de inhoud van het accumulatorregister of van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt ingesteld op de huidige waarde van de carry-vlag en bit 7 wordt uitgeschoven in de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ROL : Rotate Left Memory or Accumulator **4510**

M ← M≪1, C ← M(7), M(0) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	ROL $nn	26	2	4	r
accumulator	ROL A	2A	1	1	s
absolute	ROL $nnnn	2E	3	5	r
base-page,X	ROL $nn,X	36	2	5	pr
absolute,X	ROL $nnnn,X	3E	3	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ROR

Deze instructie verschuift ofwel het accumulatorregister ofwel de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt ingesteld op de huidige waarde van de carry-vlag, en bit 0 wordt in de carry-vlag verschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was gezet, voordat er werd verschoven.

ROR : Rotate Right Memory or Accumulator **4510**

M ← M≫1, C ← M(0), M(7) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	ROR $nn	66	2	5	r
accumulator	ROR A	6A	1	1	s
absolute	ROR $nnnn	6E	3	6	r
base-page,X	ROR $nn,X	76	2	5	$dmpr$
absolute,X	ROR $nnnn,X	7E	3	5	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ROW

Deze instructie roteert de inhoud van het opgegeven geheugenwoord één bit naar links. Bit 0 van de lage byte wordt ingesteld op de huidige waarde van de carry-vlag en bit 7 van de hoge byte wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de hoge byte voor het verschuiven gezet was.

ROW : Rotate Word Left **4510**

M2:M1 ← M2:M1≪1, C ← M2(7), M1(0) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute	ROW $nnnn	EB	3	5	dmr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RTI

Deze instructie haalt de processorvlaggen van de stapel en vervolgens het Program Counter (PC)-register van de stapel, waardoor een onderbroken programma kan worden hervat.

- De 6502-processorvlaggen worden hersteld van de stapel.
- Door deze instructie wordt noch de B-vlag (software-interrupt) noch de E-vlag (uitgebreide stapel) gezet.

RTI : Return From Interrupt **4510**

P ← STAPEL, PCH:PCL ← STAPEL, SP ← SP − 3

N	Z	I	C	D	V	E
+	+	+	+	+	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	RTI	40	1	6	m

m Trek de niet-buscycli af bij 40 MHz.

RTS

Deze instructie haalt het Program Counter (PC)-register van de stapel en telt vervolgens een optioneel byte-argument op bij de stapelaanwijzer, waardoor een subroutine naar haar aanroeper kan terugkeren.

Het optionele argument is bedoeld om een aanroeper in staat te stellen argumenten voor de subroutine op de stapel te plaatsen. De subroutine leest de argumenten uit het stapelgeheugen en keert vervolgens terug met het aantal argumentbytes. De CPU haalt naast het retouradres ook de argumenten van de stapel, zodat de aanroeper de argumenten niet expliciet hoeft te verwijderen.

RTS : Return From Subroutine **4510**

PCH:PCL ← STAPEL, SP ← SP + 2 (+ M)

			N	**Z**	**I**	**C**	**D**	**V**	**E**
			.	.	.	.	.	.	.

Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**	
implied	RTS	60	1	6	m
immediate 8bit	RTS #$nn	62	2	4	

m Trek de niet-buscycli af bij 40 MHz.

SBC

Deze instructie voert A − M − 1 + C uit en zet de processorvlaggen dienovereenkomstig. Het resultaat wordt in het accumulatorregister opgeslagen.

OPMERKING: Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt de vlag gewist.

SBC : Subtract With Carry **4510**

A ← − M − 1 + C

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	SBC ($nn,X)	E1	2	3	mp
base-page	SBC $nn	E5	2	3	r
immediate 8bit	SBC #$nn	E9	2	2	
absolute	SBC $nnnn	ED	3	4	r
(indirect),Y	SBC ($nn),Y	F1	2	3	pr
(indirect),Z	SBC ($nn),Z	F2	2	5	pr
base-page,X	SBC $nn,X	F5	2	3	pr
absolute,Y	SBC $nnnn,Y	F9	3	4	pr
absolute,X	SBC $nnnn,X	FD	3	4	pr

m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SEC

Deze instructie zet de carry-vlag.

Neveneffecten

- De C-vlag wordt gezet.

SEC : Set Carry Flag **4510**

C ← 1

N	Z	I	C	D	V	E
·	·	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	SEC	38	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SED

Deze instructie zet de decimalenvlag. Rekenkundige bewerkingen maken vanaf nu gebruik van de BCD-modus (Binary Coded Decimal).

OPMERKING: de interrupt-handler van de C64 wist de decimalenvlag niet, waardoor het gevaarlijk is de decimalenvlag te zetten zonder eerst de interruptblokkeervlag te zetten.

Neveneffecten

- De D-vlag wordt gezet.

SED : Set Decimal Flag				**4510**
D ← 1				
			N Z I C D V E	
			· · · · + · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	SED	F8	1	1 *s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SEE

Deze instructie zet de Extended Stack Disable Flag. Hierdoor werkt de stapel zoals op de 6502, d.w.z. beperkt tot één geheugenpagina. De geheugenpagina waarin de stapel zich bevindt, kan nog steeds worden gewijzigd door het Stack Pointer High (SPH)-register in te stellen.

Neveneffecten

- De E-vlag wordt gezet.

SEE : Set Extended Stack Disable Flag				**4510**
E ← 1				
			N Z I C D V E	
			· · · · · · +	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	SEE	03	1	1 *s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SEI

Deze instructie zet de interruptblokkeervlag. Gewone (IRQ-)interrupts kunnen niet langer optreden. Niet-maskeerbare interrupts (NMI) blijven optreden, zoals hun naam al aangeeft.

Neveneffecten

- De I-vlag wordt gezet.

SEI : Set Interrupt Disable Flag **4510**

I ← 1

N	Z	I	C	D	V	E
·	·	+	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	SEI	78	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SMB0

Deze instructie zet bit nul van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB0 : Set Bit 0 in Base Page **4510**

M(0) ← 1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB0 $nn	87	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB1

Deze instructie zet bit 1 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB1 : Set Bit 1 in Base Page **4510**

M(1) ← 1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB1 $nn	97	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB2

Deze instructie zet bit 2 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB2 : Set Bit 2 in Base Page **4510**

M(2) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB2 $nn	A7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB3

Deze instructie zet bit 3 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB3 : Set Bit 3 in Base Page **4510**

M(3) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB3 $nn	B7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB4

Deze instructie zet bit 4 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB4 : Set Bit 4 in Base Page **4510**

M(4) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB4 $nn	C7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB5

Deze instructie zet bit 5 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB5 : Set Bit 5 in Base Page				**4510**
M(5) ← 1				
			N Z I C D V E	
			. . . .	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
base-page	SMB5 $nn	D7	2	4 r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB6

Deze instructie zet bit 6 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB6 : Set Bit 6 in Base Page				**4510**
M(6) ← 1				
			N Z I C D V E	
			. . . .	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
base-page	SMB6 $nn	E7	2	4 r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB7

Deze instructie zet bit 7 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB7 : Set Bit 7 in Base Page				**4510**
M(7) ← 1				
			N Z I C D V E	
			. . . .	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
base-page	SMB7 $nn	F7	2	4 r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

STA

Deze instructie slaat de inhoud van het accumulatorregister op in de aangeduide locatie.

STA : Store Accumulator **4510**

M ← A

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	STA ($nn,X)	81	2	5	p
(immediate,SP),Y	STA ($nn,SP),Y	82	2	6	p
base-page	STA $nn	85	2	3	
absolute	STA $nnnn	8D	3	4	
(indirect),Y	STA ($nn),Y	91	2	5	p
(indirect),Z	STA ($nn),Z	92	2	5	p
base-page,X	STA $nn,X	95	2	3	p
absolute,Y	STA $nnnn,Y	99	3	4	p
absolute,X	STA $nnnn,X	9D	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

STX

Deze instructie slaat de inhoud van het X-register op in de aangeduide locatie.

STX : Store X Register **4510**

M ← X

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	STX $nn	86	2	3	
absolute	STX $nnnn	8E	3	4	
base-page,Y	STX $nn,Y	96	2	3	p
absolute,Y	STX $nnnn,Y	9B	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

STY

Deze instructie slaat de inhoud van het Y-register op in de aangeduide locatie.

STY : Store Y Register **4510**

M ← Y

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	STY $nn	84	2	3	
absolute,X	STY $nnnn,X	8B	3	4	p
absolute	STY $nnnn	8C	3	4	
base-page,X	STY $nn,X	94	2	3	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

STZ

Deze instructie slaat de inhoud van het Z-register op in het opgegeven geheugenadres.

STZ : Store Z Register **4510**

M ← Z

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	STZ $nn	64	2	3	
base-page,X	STZ $nn,X	74	2	3	
absolute	STZ $nnnn	9C	3	4	
absolute,X	STZ $nnnn,X	9E	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

TAB

Deze instructie stelt het Base Page-register in op de inhoud van het accumulatorregister. Hierdoor kan de zero page van de 6502 naar elke pagina in het geheugen worden verplaatst.

TAB : Transfer Accumulator into Base Page Register **4510**

B ← A

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TAB	5B	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TAX

Deze instructie laadt het X-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet is, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

TAX : Transfer Accumulator Register into the X Register **4510**

X ← A

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TAX	AA	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TAY

Deze instructie laadt het Y-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet is, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

TAY : Transfer Accumulator Register into the Y Register **4510**

Y ← A

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TAY	A8	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TAZ

Deze instructie laadt het Z-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit gezet is; anders wordt de vlag gewist.

- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

TAZ : Transfer Accumulator Register into the Z Register **4510**

Z ← A

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TAZ	4B	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TBA

Deze instructie laadt het accumulatorregister met de inhoud van het Base Page-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit gezet is; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

TBA : Transfer Base Page Register into the Accumulator **4510**

A ← B

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TBA	7B	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TRB

Deze instructie voert een binair EN op de ontkrachte versie van het accumulatorregister en de aangegeven geheugenlocatie uit en slaat het resultaat daar op. Dat wil zeggen dat elk bit dat in het accumulatorregister gezet is, wordt gereset in de aangegeven geheugenlocatie.

Deze instructie voert ook een test uit op bits die zowel in het accumulatorregister als in de aangegeven geheugenlocatie voorkomen. Hiermee kunnen eenvoudige gedeeld-geheugen multiprocessorsystemen worden gebouwd, doordat een atomair mechanisme wordt geboden om een semafoor te zetten of een vergrendeling te verkrijgen.

Neveneffecten

- De Z-vlag wordt gezet als het binaire EN van het accumulatorregister en de inhoud van de aangegeven geheugenlocatie voor de uitvoering van de instructie nul is.

TRB : Test and Reset Bit **4510**

M ← M *AND* (*NOT* A)

N	Z	I	C	D	V	E
·	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	TRB $nn	14	2	5	r
absolute	TRB $nnnn	1C	3	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

TSB

Deze instructie voert een binair OF op het accumulatorregister en de aangegeven geheugenlocatie uit en slaat het resultaat daar op. Dat wil zeggen dat elk bit dat in het accumulatorregister gezet is, wordt gezet in de aangegeven geheugenlocatie.

Deze instructie voert ook een test uit op bits die zowel in het accumulatorregister als in de aangegeven geheugenlocatie voorkomen. Hiermee kunnen eenvoudige gedeeld-geheugen multiprocessorsystemen worden gebouwd, doordat een atomair mechanisme wordt geboden om een semafoor te zetten of een vergrendeling te verkrijgen.

Neveneffecten

- De Z-vlag wordt gezet als het binaire EN van het accumulatorregister en de inhoud van de aangegeven geheugenlocatie voor de uitvoering van de instructie nul is.

TSB : Test and Set Bit **4510**

M ← M *OR* A

N	Z	I	C	D	V	E
·	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	TSB $nn	04	2	3	r
absolute	TSB $nnnn	0C	3	5	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

TSX

Deze instructie laadt het X-register met de inhoud van het Stack Pointer Low (SPL)-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TSX : Transfer Stack Pointer Low Register into the X Register **4510**

X ← SPL

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TSX	BA	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TSY

Deze instructie laadt het Y-register met de inhoud van het Stapelpointer High (SPH)-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit ingesteld is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

TSY : Transfer Stack Pointer High Register into the Y Register **4510**

Y ← SPH

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TSY	0B	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TXA

Deze instructie laadt het accumulatorregister met de inhoud van het X-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TXA : Transfer X Register into the Accumulator Register **4510**

A ← X

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TXA	8A	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TXS

Deze instructie zet de lage byte van het stapelaanwijzerregister (SPL) op de inhoud van het X-register.

TXS : Transfer X Register into Stack Pointer Low Register **4510**

SPL ← X

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TXS	9A	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TYA

Deze instructie laadt het accumulatorregister met de inhoud van het Y-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TYA : Transfer Y Register into the Accumulator Register **4510**

A ← Y

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TYA	98	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TYS

Deze instructie zet het hoge byte van de stapelaanwijzerregister (SPH) op de inhoud van het Y-register. Hierdoor kan de geheugenpagina waar de stapel zich bevindt,

worden gewijzigd (indien de “Extended Stack Disable”-vlag (E) gezet is), of kan de huidige stapelpointer op elke pagina in het geheugen worden ingesteld, indien de “Extended Stack Disable”-vlag (E) gereset is.

TYS : Transfer Y Register into Stack Pointer High Register **4510**

SPH ← Y

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TYS	2B	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TZA

Deze instructie laadt het accumulatorregister met de inhoud van het Z-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel als het meest significante bit gezet is; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

TZA : Transfer Z Register into the Accumulator Register **4510**

A ← Z

N	Z	I	C	D	V	E
+	+	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TZA	6B	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SAMENGESTELDE INSTRUCTIES VAN DE 45GS02

Omdat de 4510 geen ongebruikte opcodes heeft, gebruikt de 45GS02 samengestelde instructies om de uitbreiding te implementeren. Deze samengestelde instructies bestaan uit één of meer instructies van één byte, welke onmiddellijk vóór een conventionele instructie worden geplaatst. Deze prefixes geven de 45GS02 de opdracht de volgende instructie anders te behandelen, zoals beschreven in Hoofdstuk/Bijlage K op bladzijde K-3.

OPMERKING: Prefix-instructies worden door de CPU uitgevoerd alsof het zelfstandige instructies zijn, nog voordat de samengestelde instructie wordt herkend. Dit kan

van invloed zijn op de vlagstatus. De instructiebytes 42 42 8D voeren bijvoorbeeld twee NEG-instructies uit, gevolgd door de samengestelde STQ-instructie. De NEG-instructies kunnen van invloed zijn op de N- en Z-vlaggen, afhankelijk van de inhoud van het accumulatorregister.

U vindt de samengestelde instructies gemarkeerd in de 4510-opcodetabel. Zie Hoofdstuk/Bijlage L op bladzijde L-58.

ADC

Deze instructie telt het argument en de carry-vlag op bij de inhoud van het accumulatorregister. Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten, anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De C-vlag wordt gezet als het ongetekende resultaat >255 is, of >99 indien de D-vlag is gezet.

ADC : Add with carry **45GS02**

$A \leftarrow A + M + C$

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	ADC [$nn],Z	EA 72	3	7	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ADCQ

Deze instructie telt het argument en de carry-vlag op bij de inhoud van het 32-bits Q-pseudoregister.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Opmerking: indien de D-vlag gezet is, is de bewerking ongedefinieerd en kan deze nog wijzigen.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt deze gewist. Indien de vlag gezet is, geeft dit aan dat een tekenoverflow heeft plaatsgevonden.
- De C-vlag wordt gezet als het ongetekende resultaat $\geq 2^{32}$ is.

ADCQ : Add with carry Quad **45GS02**

Q ← Q + M + C

			N	Z	I	C	D	V	E	
			+	+	·	+	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page quad	ADCQ $nn	42 42 65	4	8	r					
absolute quad	ADCQ $nnnn	42 42 6D	5	9	r					
(indirect quad)	ADCQ ($nn)	42 42 72	4	10	ipr					
[indirect quad]	ADCQ [$nn]	42 42 EA 72	5	13	ipr					

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

AND

Deze instructie voert een bitgewijze EN-operatie uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister en die ook in het argument staan, worden na voltooiing ingesteld in het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

AND : Binary AND **45GS02**

A ← A *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	AND [$nn],Z	EA 32	3	7	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ANDQ

Deze instructie voert een binair EN-bewerking uit van het argument met het Q-pseudoregister en slaat het resultaat op in het accumulatorregister. Alleen bits die reeds in het Q-pseudoregister stonden en die ook in het argument zijn ingesteld, blijven in het Q-pseudoregister ingesteld na afloop.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

ANDQ : Binary AND Quad **45GS02**

Q ← Q *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	ANDQ $nn	42 42 25	4	8	r
absolute quad	ANDQ $nnnn	42 42 2D	5	9	r
(indirect quad)	ANDQ ($nn)	42 42 32	4	10	ipr
[indirect quad]	ANDQ [$nn]	42 42 EA 32	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ASLQ

Deze instructie verschuift het Q-pseudoregister, of de inhoud van de opgegeven geheugenlocatie en de drie daaropvolgende, één bit naar links, waarbij deze als een

little-endian 32-bits waarde worden behandeld. Bit 0 wordt op nul gezet en bit 31 wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De C-vlag wordt gezet als bit 31 van de waarde is ingesteld vóórdat deze werd verschoven; anders wordt deze gewist.

ASLQ : Arithmetic Shift Left Quad **45GS02**

Q ← Q<<1 *of* M ← M<<1

			N	Z	I	C	D	V	E	
			+	+	·	+	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page quad	ASLQ $nn	42 42 06	4	12	dmr					
pseudoregister Q	ASLQ Q	42 42 0A	3	3						
absolute quad	ASLQ $nnnn	42 42 0E	5	13	dmr					
base-page quad,X	ASLQ $nn,X	42 42 16	4	12	$dmpr$					
absolute quad,X	ASLQ $nnnn,X	42 42 1E	5	13	$dmpr$					

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ASRQ

Deze instructie verschuift het Q-pseudoregister, of de inhoud van de opgegeven geheugenlocatie en de drie daaropvolgende, één bit naar rechts, waarbij deze als een little-endian 32-bits waarde worden behandeld. Bit 31 wordt als tekenbit beschouwd en blijft behouden. De inhoud van bit 0 wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De C-vlag wordt gezet als bit 0 van de waarde is ingesteld vóórdat deze werd verschoven; anders wordt deze gewist.

ASRQ : Arithmetic Shift Right Quad **45GS02**

Q ← Q>>1 *of* M ← M>>1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
pseudoregister Q	ASRQ Q	42 42 43	3	3	
base-page quad	ASRQ $nn	42 42 44	4	12	dmr
base-page quad,X	ASRQ $nn,X	42 42 54	4	12	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.

m Trek de niet-buscycli af bij 40 MHz.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BITQ

Deze instructie wordt gebruikt om de bits te testen die zijn opgeslagen in een geheugenlocatie en de drie daaropvolgende, waarbij deze als een little-endian 32-bits waarde worden behandeld. Bits 30 en 31 van de inhoud van de geheugenlocatie worden rechtstreeks gekopieerd naar de Overflow-vlag en de Negativiteit-vlag. De Zero-vlag wordt gezet of gereset op basis van het resultaat van het uitvoeren van de binaire EN-bewerking van het Q-register en de inhoud van de aangegeven geheugenlocatie.

Neveneffecten

- De N-vlag wordt gezet als bit 31 van de geheugenlocatie is ingesteld; anders wordt deze gewist.
- De V-vlag wordt gezet als bit 30 van de geheugenlocatie is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van Q *AND* M nul is; anders wordt deze gewist.

BITQ : Perform Bit Test Quad **45GS02**

N ← M(31), V ← M(30), Z ← Q *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	BITQ $nn	42 42 24	4	8	r
absolute quad	BITQ $nnnn	42 42 2C	5	9	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CMP

Deze instructie voert A − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt deze gewist.

CMP : Compare Accumulator **45GS02**

N,C,Z ⇐ [A − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	CMP [$nn],Z	EA D2	3	7	^{ipr}

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CMPQ

Deze instructie voert Q − M uit en stelt de processorvlaggen dienovereenkomstig in, maar wijzigt de inhoud van het Q-register niet.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Q − M negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van Q − M nul of positief is, d.w.z. als Q niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van Q − M nul is; anders wordt deze gewist.

CMPQ : Compare Q Pseudo Register **45GS02**

N,C,Z $\Leftarrow$ [Q − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	CMPQ $nn	42 42 C5	4	8	r
absolute quad	CMPQ $nnnn	42 42 CD	5	9	r
(indirect quad)	CMPQ ($nn)	42 42 D2	4	10	ipr
[indirect quad]	CMPQ [$nn]	42 42 EA D2	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

DEQ

Deze instructie verlaagt het Q-pseudoregister of de opgegeven geheugenlocatie met één.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEQ : Decrement Quad **45GS02**

Q $\leftarrow$ Q − 1 of M $\leftarrow$ M − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
pseudoregister Q	DEQ Q	42 42 3A	3	3	
base-page quad	DEQ $nn	42 42 C6	4	12	dmr
absolute quad	DEQ $nnnn	42 42 CE	5	13	dmr
base-page quad,X	DEQ $nn,X	42 42 D6	4	12	$dmpr$
absolute quad,X	DEQ $nnnn,X	42 42 DE	5	13	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

EOR

Deze instructie voert een bitgewijze uitsluitende-OF-operatie (XOR) uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister of die ingesteld staan in het argument, worden na voltooiing ingesteld in het accumulatorregister, maar niet allebei.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

EOR : Binary Exclusive OR **45GS02**

A ← A *XOR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	EOR [$nn],Z	EA 52	3	7	^{ipr}

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

EORQ

Deze instructie voert een binaire EXCLUSIEF-OF-bewerking uit van het argument met het Q-pseudoregister en slaat het resultaat op in het Q-pseudoregister. Alleen bits die reeds in het Q-pseudoregister stonden of die in het argument zijn ingesteld, worden bij afronding in het Q-pseudoregister ingesteld, maar niet bits die in beide stonden.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

EORQ : Binary Exclusive OR Quad **45GS02**

Q ← Q *XOR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	EORQ $nn	42 42 45	4	8	r
absolute quad	EORQ $nnnn	42 42 4D	5	9	r
(indirect quad)	EORQ ($nn)	42 42 52	4	10	ipr
[indirect quad]	EORQ [$nn]	42 42 EA 52	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

INQ

Deze instructie verhoogt het Q-pseudoregister of de opgegeven geheugenlocatie.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

INQ : Increment Memory or Accumulator **45GS02**

Q ← Q + 1 of M ← M + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
pseudoregister Q	INQ Q	42 42 1A	3	3	
base-page quad	INQ $nn	42 42 E6	4	13	dmr
absolute quad	INQ $nnnn	42 42 EE	5	14	dmr
base-page quad,X	INQ $nn,X	42 42 F6	4	13	$dmpr$
absolute quad,X	INQ $nnnn,X	42 42 FE	5	14	dpr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDA

Deze instructie laadt het accumulatorregister met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.

LDA : Load Accumulator **45GS02**

A ← M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**		**Cycli**					
[indirect],Z	LDA [$nn],Z	EA B2	3		7	^{ipr}				

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDQ

Deze instructie laadt het Q-pseudoregister met de opgegeven waarde of met de inhoud van de opgegeven locatie. Omdat het Q-register een alias is voor A, X, Y en Z tezamen, zet deze bewerking die vier registers. A bevat de minst significante bits, X de volgende minst significante bits, vervolgens Y, en Z bevat de meest significante bits.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

LDQ : Load Q Pseudo Register **45GS02**

Q ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	LDQ $nn	42 42 A5	4	8	r
absolute quad	LDQ $nnnn	42 42 AD	5	9	r
(indirect quad),Z	LDQ ($nn),Z	42 42 B2	4	10	ipr
[indirect quad],Z	LDQ [$nn],Z	42 42 EA B2	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LSRQ

Deze instructie verschuift het Q-pseudoregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 31 wordt op nul gezet en bit 0 wordt in de carry-vlag verschoven.

Opmerking: het geheugenadres wordt beschouwd als het eerste adres van een little-endian 32-bits waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De C-vlag wordt gezet als bit 0 van de waarde voor het verschuiven gezet was; anders wordt deze gewist.

LSRQ : Logical Shift Right Quad **45GS02**

Q ← Q≫1, C ← A(0) of M ← M≫1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	LSRQ $nn	42 42 46	4	12	dmr
pseudoregister Q	LSRQ Q	42 42 4A	3	3	
absolute quad	LSRQ $nnnn	42 42 4E	5	13	dmr
base-page quad,X	LSRQ $nn,X	42 42 56	4	12	$dmpr$
absolute quad,X	LSRQ $nnnn,X	42 42 5E	5	13	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ORA

Deze instructie voert een binaire OR-bewerking uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al in het accumulatorregister gezet waren, of die in het argument gezet waren, worden bij voltooiing in het accumulatorregister gezet, of beide.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

ORA : Binary OR **45GS02**

A ← A *OR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	ORA [$nn],Z	EA 12	3	7	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ORQ

Deze instructie voert een binaire OR-bewerking uit van het argument met het Q-pseudoregister en slaat het resultaat op in het Q-pseudoregister. Alleen bits die reeds

in het Q-pseudoregister stonden of die in het argument zijn ingesteld, of beide, worden bij afronding in het Q-pseudoregister ingesteld.

Opmerking: deze bewerking behandelt het geheugenadres als het eerste adres van een 32-bits little-endian-waarde. Dat wil zeggen dat het geheugenadres en de drie daaropvolgende worden gebruikt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

ORQ : Binary OR Quad **45GS02**

Q ← Q *OR* M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page quad	ORQ $nn	42 42 05	4	8	r					
absolute quad	ORQ $nnnn	42 42 0D	5	9	r					
(indirect quad)	ORQ ($nn)	42 42 12	4	10	pr					
[indirect quad]	ORQ [$nn]	42 42 EA 12	5	13	pr					

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RESQ

Deze uitgebreide opcodes zijn gereserveerd en hun functie is niet gedefinieerd en kan in toekomstige revisies van de 45GS02 nog wijzigen. Ze mogen derhalve in geen enkel programma worden gebruikt.

RESQ : Reserved extended opcode **45GS02**

UNDEFINED

			N	Z	I	C	D	V	E	
			.	.	.	.	.	.	.	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect quad,X)	RESQ ($nn,X)	42 42 01	4	10	ipr
(indirect quad),Y	RESQ ($nn),Y	42 42 11	4	10	ipr
base-page quad,X	RESQ $nn,X	42 42 15	4	8	pr
absolute quad,Y	RESQ $nnnn,Y	42 42 19	5	9	pr
absolute quad,X	RESQ $nnnn,X	42 42 1D	5	9	pr
(indirect quad,X)	RESQ ($nn,X)	42 42 21	4	10	ir
(indirect quad),Y	RESQ ($nn),Y	42 42 31	4	10	ipr
base-page quad,X	RESQ $nn,X	42 42 34	4	8	pr
base-page quad,X	RESQ $nn,X	42 42 35	4	8	pr
absolute quad,Y	RESQ $nnnn,Y	42 42 39	5	10	pr
absolute quad,X	RESQ $nnnn,X	42 42 3C	5	9	pr
absolute quad,X	RESQ $nnnn,X	42 42 3D	5	10	pr
(indirect quad,X)	RESQ ($nn,X)	42 42 41	4	10	ipr
(indirect quad),Y	RESQ ($nn),Y	42 42 51	4	10	ipr
base-page quad,X	RESQ $nn,X	42 42 55	4	8	pr
absolute quad,Y	RESQ $nnnn,Y	42 42 59	5	9	pr
absolute quad,X	RESQ $nnnn,X	42 42 5D	5	9	pr
(indirect quad,X)	RESQ ($nn,X)	42 42 61	4	10	ir
(indirect quad),Y	RESQ ($nn),Y	42 42 71	4	10	ipr
base-page quad,X	RESQ $nn,X	42 42 75	4	8	pr
absolute quad,Y	RESQ $nnnn,Y	42 42 79	5	10	pr
absolute quad,X	RESQ $nnnn,X	42 42 7D	5	10	pr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ROLQ

Deze instructie verschuift het Q-pseudoregister of de inhoud van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt ingesteld op de huidige waarde van de carry-vlag en bit 31 wordt in de carry-vlag geschoven.

OPMERKING: het geheugenadres wordt beschouwd als het eerste adres van een little-endian gecodeerde 32-bits waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.

- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De C-vlag wordt gezet als bit 31 van de waarde is ingesteld vóórdat deze werd verschoven; anders wordt deze gewist.

ROLQ : Rotate Left Quad **45GS02**

M ← M≪1, C ← M(31), M(0) ← C or Q ← Q≪1, C ← Q(31), Q(0) ← C

			N	Z	I	C	D	V	E	
			+	+	·	+	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page quad	ROLQ $nn	42 42 26	4	12	dmr					
pseudoregister Q	ROLQ Q	42 42 2A	3	3						
absolute quad	ROLQ $nnnn	42 42 2E	5	13	dmr					
base-page quad,X	ROLQ $nn,X	42 42 36	4	12	$dmpr$					
absolute quad,X	ROLQ $nnnn,X	42 42 3E	5	13	$dmpr$					

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RORQ

Deze instructie verschuift het Q-pseudoregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 31 wordt ingesteld op de huidige waarde van de carry-vlag en bit 0 wordt in de carry-vlag geschoven

Opmerking: het adres wordt beschouwd als het eerste adres van een little-endian 32-bits waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit ervan gezet is en anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is en anders wordt hij gewist.
- De C-vlag wordt gezet als bit 31 van de waarde gezet was, vóór deze verschoven werd en anders wordt hij gewist.

RORQ : Rotate Right Quad **45GS02**

M ← M≫1, C ← M(0), M(31) ← C or Q ← Q≫1, C ← Q(0), Q(31) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	RORQ $nn	42 42 66	4	12	dmr
pseudoregister Q	RORQ Q	42 42 6A	3	3	
absolute quad	RORQ $nnnn	42 42 6E	5	13	dmr
base-page quad,X	RORQ $nn,X	42 42 76	4	12	$dmpr$
absolute quad,X	RORQ $nnnn,X	42 42 7E	5	13	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RSVQ

Deze uitgebreide opcodes zijn gereserveerd en hun functie is niet gedefinieerd en kan in toekomstige revisies van de 45GS02 nog wijzigen. Ze mogen derhalve in geen enkel programma worden gebruikt.

RSVQ : Reserved extended opcode **45GS02**

UNDEFINED

			N	Z	I	C	D	V	E	
			.	.	.	.	.	.	.	
Adresseermethode	**Instructie**	**Opcode**		**Bytes**		**Cycli**				
(indirect quad,X)	RSVQ ($nn,X)	42 42 81		4		10	ip			
(indirect quad,SP),Y	RSVQ ($nn,SP),Y	42 42 82		4		10	ip			
(indirect quad),Y	RSVQ ($nn),Y	42 42 91		4		10	ip			
base-page quad,X	RSVQ $nn,X	42 42 95		4		8	p			
absolute quad,Y	RSVQ $nnnn,Y	42 42 99		5		9	p			
absolute quad,X	RSVQ $nnnn,X	42 42 9D		5		9	p			
(indirect quad,X)	RSVQ ($nn,X)	42 42 A1		4		10	ipr			
(indirect quad,X)	RSVQ ($nn,X)	42 42 C1		4		10	ipr			
(indirect quad),Y	RSVQ ($nn),Y	42 42 D1		4		10	ipr			
base-page quad,X	RSVQ $nn,X	42 42 D5		4		8	pr			
absolute quad,Y	RSVQ $nnnn,Y	42 42 D9		5		9	pr			
absolute quad,X	RSVQ $nnnn,X	42 42 DD		5		9	pr			
(indirect quad,X)	RSVQ ($nn,X)	42 42 E1		4		10	ipr			
(indirect quad),Y	RSVQ ($nn),Y	42 42 F1		4		10	ipr			
base-page quad,X	RSVQ $nn,X	42 42 F5		4		8	pr			
absolute quad,Y	RSVQ $nnnn,Y	42 42 F9		5		8	pr			
absolute quad,X	RSVQ $nnnn,X	42 42 FD		5		9	pr			

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SBC

Deze instructie voert A $-$ M $-$ 1 $+$ C uit en zet de processorvlaggen dienovereenkomstig. Het resultaat wordt in het accumulatorregister opgeslagen.

OPMERKING: Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A $-$ M negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A $-$ M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.

- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt de vlag gewist.

SBC : Subtract With Carry				**45GS02**
A ← − M − 1 + C				
			N Z I C D V E	
			+ + · + · + ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
[indirect],Z	SBC [$nn],Z	EA F2	3	0

SBCQ

Deze instructie voert Q − M − 1 + C uit en stelt de processorvlaggen overeenkomstig in. Het resultaat wordt opgeslagen in het Q-pseudoregister.

Let op: de aangegeven geheugenplaats wordt behandeld als het eerste byte van een 32-bits little-endian waarde.

Let op: als de D-vlag gezet is, is de bewerking ongedefinieerd en onderhevig aan verandering.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Q − M negatief is, d.w.z. als het meest significante bit ervan gezet is en anders wordt hij gewist.
- De C-vlag wordt gezet als het resultaat van Q − M nul of positief is, d.w.z. als A niet kleiner is dan M en anders wordt hij gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten en anders wordt hij gewist. Als de vlag gezet is, geeft dit aan dat er een getekende overflow is opgetreden.
- De Z-vlag wordt gezet als het resultaat van Q − M nul is en anders wordt hij gewist.

SBCQ : Subtract With Carry Quad **45GS02**

Q ← Q − M − 1 + C

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	SBCQ $nn	42 42 E5	4	8	r
absolute quad	SBCQ $nnnn	42 42 ED	5	9	r
(indirect quad)	SBCQ ($nn)	42 42 F2	4	10	ipr
[indirect quad]	SBCQ [$nn]	42 42 EA F2	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

STA

Deze instructie slaat de inhoud van het accumulatorregister op in de aangeduide locatie.

STA : Store Accumulator **45GS02**

M ← A

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	STA [$nn],Z	EA 92	3	8	ip

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

STQ

Deze instructie slaat de inhoud van het Q-pseudoregister op in de opgegeven geheugenlocatie.

Aangezien Q is opgebouwd uit A, X, Y en Z, betekent dit dat deze vier registers respectievelijk worden geschreven naar de opgegeven geheugenlocatie en de drie daaropvolgende geheugenlocaties.

Opmerking: De CPU voert het 42 42-voorvoegsel van de STQ-instructie uit als twee NEG-instructies, voordat hij de STQ-instructie herkent en uitvoert. Dit beïnvloedt de N- en Z-vlag, afhankelijk van de inhoud van het accumulatorregister.

STQ : Store Q **45GS02**

M ← A, M+1 ← X, M+2 ← Y, M+3 ← Z

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	STQ $nn	42 42 85	4	8	
absolute quad	STQ $nnnn	42 42 8D	5	9	
(indirect quad)	STQ ($nn)	42 42 92	4	10	ip
[indirect quad]	STQ [$nn]	42 42 EA 92	5	13	ip

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

BIJLAGE M

Ontwikkeling van systeemprogramma's

- Inleiding
- Flashmenu
- Format/FDISK-hulpprogramma
- Toetsenbordtest-hulpprogramma
- MEGA65-configuratiehulpprogramma
- Freezer-menu
- Hulpprogramma's van het Freezer-menu
- Hypervisor

- OpenROM

INLEIDING

De MEGA65 heeft een aantal systeemprogramma's en hulpprogramma's die bij diverse gelegenheden verschillende functies vervullen. Daartoe behoren de hulpprogramma's via het hulpprogrammamenu, het Freezer-menu en de eigen hulpprogramma's, alsmede het flashmenu.

Een aantal van deze systeemprogramma's is vooraf in de bitstream van de MEGA65 geladen, terwijl andere op de SD-kaart staan. Bij de versies die vooraf in de bitstream zijn geladen, gebeurt dit door middel van voorgeïnitialiseerde geheugengebieden die het juiste programma bevatten. De hulpprogramma's die via het hulpprogrammamenu toegankelijk zijn, bevinden zich bijvoorbeeld allemaal in het colour RAM, terwijl het flashmenu op het adres $50000 - $57FFF staat.

De eenvoudigste manier om nieuwe versies van deze hulpprogramma's te testen, is het genereren van een nieuwe bitstream met de bijgewerkte versies. Het synthesiseren van een nieuwe bitstream is echter tijdrovend: het duurt doorgaans een uur op een redelijk snelle computer. Dit hoofdstuk legt daarom de procedure uit voor het laden van een alternatieve versie van elk van deze systeemprogramma's, en verstrekt enkele nuttige gegevens over deze programma's, hun werking en de omgeving waarin zij werken in vergelijking met gewone programma's in C64- of C65-modus.

FLASHMENU

Het flashmenu bevindt zich in voorgeïnitialiseerd RAM op $50000 - $57FFF. Het wordt iedere keer uitgevoerd tijdens de eerste opstart na inschakelen van de MEGA65. Het programma is buitenissig in de zin dat het in de Hypervisor-context draait. Dit is nodig om toegang te krijgen tot het QSPI-flashgeheugen, dat buiten de Hypervisor-modus niet beschikbaar is, zodat gebruikersprogramma's de cores die in de flash zijn opgeslagen niet kunnen beschadigen.

Het is ook belangrijk op te merken dat het flashmenu-programma *volledig* onder $8000 moet passen wanneer het wordt geladen *en* wordt uitgevoerd, omdat de Hypervisor op $8000 - $BFFF in het geheugen geladen is en gemakkelijk kan worden beschadigd door een flashmenu-programma dat zich misdraagt. In dit opzicht kan het flashmenu worden beschouwd als een uitbreiding van de Hypervisor die na de eerste opstart wordt verwijderd. Dit verschilt van alle andere systeemprogramma's, die in een afzonderlijk geheugencontext werken waarin de Hypervisor beschermd is tegen beschadiging. Het betekent ook dat u het flashmenu niet kunt comprimeren om het te laten passen, omdat het tijdens het uitpakken de Hypervisor zou overschrijven.

Ook wordt het flashmenu zeer vroeg in het opstartproces uitgevoerd, zodat alleen het vooraf opgenomen OpenROM ROM-image beschikbaar is. U moet er daarom voor zorgen dat uw flashmenu-programma compatibel is met die ROM.

De Hypervisor houdt een vlag bij die aangeeft of het flashmenu al dan niet is uitgevoerd. Deze vlag wordt bijgewerkt op het moment dat de Hypervisor voor de eerste

keer naar de gebruikersmodus overschakelt, omdat de inhoud van $50000 - $57FFF vanaf dat moment niet meer betrouwbaar is als opslagplaats van het flashmenu. Dit betekent dat u, indien u de Hypervisor een nieuwe versie van het flashmenu wilt laten uitvoeren die u hebt geladen, eerst moet voorkomen dat de Hypervisor naar de gebruikersmodus overschakelt.

De eenvoudigste manier om dit te bereiken, is de ALT-toets ingedrukt te houden terwijl u de MEGA65 inschakelt. Hierdoor toont de Hypervisor het hulpprogrammamenu in plaats van naar de gebruikersmodus over te schakelen. Op dit moment kunt u het `m65`-hulpprogramma gebruiken om het vervangende flashmenu-programma te laden met een commando als het volgende:

```
m65 -@ newflashmenu.prg@50000
```

Dat commando laadt het bestand `newflashmenu.prg` op geheugenadres $50000 geladen.

Daarna kunt u eenvoudig de resetknop aan de zijkant van de MEGA65 indrukken terwijl u NO SCROLL ingedrukt houdt. De computer start opnieuw op en omdat tijdens de vorige opstartcyclus niet uit de Hypervisor-modus is overschakeld, wordt uw bijgewerkte flashmenu-programma uitgevoerd.

Het zou ook mogelijk moeten zijn om dit proces volledig te automatiseren, door eerst `m65 -b` te gebruiken om een nieuwe bitstream te laden (een koude start nabootsen), vervolgens snel `m65` opnieuw aan te roepen om het indrukken van de ALT-toets na te bootsen (of eenvoudig de processor te stoppen), daarna `m65 -@ ...` en ten slotte `m65 -F` om de computer te resetten. Het schrijven van een script of hulpprogramma dat deze automatisering correct implementeert, laten wij over als een oefening voor de lezer.

FORMAT/FDISK-HULPPROGRAMMA

Het Format/FDISK-hulpprogramma is toegankelijk via het hulpprogrammamenu. Deze hulpprogramma's worden gecompileerd, gecomprimeerd en gelinkt met het `utilpacker`-programma. Indien u de broncode van de mega65-core hebt uitgecheckt, kunt u het colour RAM-image opnieuw bouwen met:

```
make bin/COLOURRAM.BIN
```

U moet uiteraard eerst het Format/FDISK-hulpprogramma hebben aangepast, dat zich normaal gesproken in de subdirectory `src/mega65-freeze-menu` bevindt.

Vervolgens moet u dit aangepaste colour RAM-image in de draaiende computer laden. Net als bij het bijwerken van het flashmenu, toont de Hypervisor het hulpprogrammamenu alleen bij de eerste opstart, voordat er voor de eerste keer naar de

gebruikersmodus wordt overgeschakeld, omdat de Hypervisor anders niet zeker kan zijn dat het colour RAM de geldige hulpprogramma's bevat.

Net als bij het flashmenu schakelt u de MEGA65 uit en schakelt u deze vervolgens weer in terwijl u de ALT-toets ingedrukt houdt, zodat het hulpprogrammamenu wordt weergegeven. Op dat moment kunt u het volgende commando gebruiken om uw aangepaste bestand `COLOURRAM.BIN` te laden:

```
m65 -c COLOURRAM.BIN
```

U kunt nu ALT ingedrukt houden en de resetknop aan de linkerkant van de MEGA65 indrukken. Het hulpprogrammamenu verschijnt dan opnieuw, ditmaal met uw aangepaste Format/FDISK-hulpprogramma.

TOETSENBORDTEST-HULPPROGRAMMA

De procedure voor het bijwerken van het toetsenbordtest-hulpprogramma is in beginsel dezelfde als voor het Format/FDISK-hulpprogramma, omdat het zich in het colour RAM bevindt.

MEGA65-CONFIGURATIEHULPPROGRAMMA

De procedure voor het bijwerken van het MEGA65-configuratiehulpprogramma is in beginsel dezelfde als voor het Format/FDISK-hulpprogramma, omdat het zich in het colour RAM bevindt.

FREEZER-MENU

Het Freezer-menu is een gewoon programma dat als `FREEZER.M65` op het FAT32-bestandssysteem van de SD-kaart staat.

Om het Freezer-menu bij te werken, gebruikt u het `m65ftp`-hulpprogramma of een andere methode om uw bijgewerkte bestand FREEZER.M65 naar het FAT32-bestandssysteem van de SD-kaart te uploaden. Het formaat van het programma is eenvoudigweg een PRG-bestand in C64-modus, hernoemd naar FREEZER.M65.

HULPPROGRAMMA'S VAN HET FREEZER-MENU

De hulpprogramma's van het Freezer-menu worden op dezelfde manier bijgewerkt als het Freezer-menu zelf.

HYPERVISOR

De Hypervisor wordt doorgaans gebouwd als HICKUP.M65, een bestand van 16 KB dat het volledige Hypervisor-programma bevat. MEGA65-bitstreams bevatten een vooraf gebouwde versie op het adres $FFF8000 – $FFFBFFF. Bijgewerkte versies van de Hypervisor kunnen op twee manieren worden getest:

- 1. Plaats het bijgewerkte bestand HICKUP.M65 op het FAT32-bestandssysteem van de SD-kaart en schakel de MEGA65 vervolgens uit en weer in. Dit werkt omdat de Hypervisor code bevat die een bijgewerkte versie van zichzelf zoekt en deze, indien gevonden, laadt. Deze aanpak is echter problematisch doordat bij het installeren van een nieuwere bitstream de Hypervisor alsnog wordt teruggedraaid naar de versie die in het bestand HICKUP.M65 op de SD-kaart staat. Deze methode wordt alleen aanbevolen voor ontwikkelaars die hun aangepaste Hypervisor-code vanuit een koude start moeten testen. Ook dan wordt aanbevolen het bestand HICKUP.M65 onmiddellijk na het testen te verwijderen om onverwachte terugval in de toekomst te voorkomen.
- 2. Gebruik de `-k`-optie van het m65-commando om de Hypervisor ter plekke te vervangen en reset de MEGA65 vervolgens met de resetknop aan de linkerkant van de behuizing. Dit dient te gebeuren wanneer de Hypervisor *niet* actief is, zodat beschadiging van de huidige uitvoering niet kan optreden. Het moet echter ook gebeuren voordat er een ROM is geladen die het standaard OpenROM-image vervangt. De Hypervisor zal bij de eerste opstart namelijk een aanroep doen naar de ROM ter voorbereiding op het flashmenu, en ervan uitgaan dat de OpenROM aanwezig is, omdat hij een speciale, voor OpenROM bestemde aanroep gebruikt om delen van de systeemtoestand voor het flashmenu te initialiseren. Het beste doet u dit met een commando als `m65 -k bin/HICKUP.M65 -R bin/MEGA65.ROM` om zowel een nieuw Hypervisor-programma te laden als het OpenROM-image opnieuw te laden.

OPENROM

Om een nieuwe versie van een ROM te laden, bestaan er verschillende mogelijkheden, waaronder het tegelijkertijd vervangen van de Hypervisor en de ROM zoals hierboven beschreven. Doorgaans is de eenvoudigste methode het kopiëren van de

nieuwe ROM naar het FAT32-bestandssysteem van de SD-kaart als MEGA65.ROM of MEGA65x.ROM, waarbij x wordt vervangen door een cijfer tussen 0 en 9. Bij het resetten van de MEGA65 wordt dan normaal gesproken MEGA65.ROM geladen, of indien een cijfer tussen 0 en 9 wordt ingedrukt op het toetsenbord tijdens het resetten, laadt de Hypervisor in plaats daarvan MEGA65x.ROM, waarbij x het cijfer is dat wordt ingedrukt.

BIJLAGE N

MEGA65 Hyppo-diensten

- Inleiding
- Hyppo-diensten: Beknopt overzicht
- Algemene diensten
- Station-/opslagservices
- Schijfimage-services
- Taak- en procesdiensten
- Systempartitieservices
- Freezerservices

INLEIDING

Een onderdeel van de MEGA65 is het systeemprogramma Hyppo dat:

- De MEGA65 opstart.
- De ROM's en andere bestanden van de SD-kaart laadt.
- Geheugenbanken 2 en 3 ROM-achtig maakt door ze tegen schrijven te beschermen.
- De diskettestationcontroller virtualiseert zodat u schijfimages kunt gebruiken.
- Diverse hulpprogramma's start zoals de Freezer en de Matrix Mode Debugger.
- Diensten biedt die specifiek zijn voor de MEGA65 en die u in uw programma's kunt gebruiken.

Wanneer u bekend bent met hypervisors en virtuele machines, is Hyppo een zeer beperkte hypervisor. Verwacht niet dat u meerdere virtuele machines tegelijkertijd met volledige isolatie kunt uitvoeren. Hyppo voert taken uit die meer doen denken aan de taken en processen van een modern besturingssysteem dan aan de virtuele machines van een hypervisor zoals u die wellicht kent.

Hyppo biedt drie bedrijfsmodi.

- **De C64-achtige bedrijfsmodus** voert C64-programma's en MEGA65-programma's uit die draaien in de C64-modus van de MEGA65. Wanneer u opstart met de  ingedrukt of het **GO64**-commando gebruikt, start Hyppo een proces in de C64-achtige bedrijfsmodus om BASIC 2 of het MEGA65-programma uit te voeren.
- **De C65-achtige bedrijfsmodus** is de normale bedrijfsmodus van de MEGA65. Hier worden de reguliere MEGA65-programma's uitgevoerd, waaronder BASIC 65-programma's.
- **De MEGA65-bedrijfsmodus** voert de systeemprogramma's van de MEGA65 uit, zoals de Freezer, het configuratiehulpprogramma en de Matrix Mode Debugger. Wellicht verrassend draaien reguliere MEGA65-programma's niet in de MEGA65-bedrijfsmodus. Zij draaien in de C65-achtige bedrijfsmodus. De MEGA65-bedrijfsmodus is uitsluitend ontworpen voor de MEGA65 en probeert niet compatibel te zijn met of zelfs maar te lijken op eerdere systemen.

In tegenstelling tot de C128 kan een programma de bedrijfsmodus effectief wijzigen terwijl het draait, door eenvoudigweg de verschillende hardwarefuncties in of uit te schakelen.

Hyppo biedt zeer beperkte virtualisatie van de hardware van de MEGA65. Het kan de diskettestationcontroller virtualiseren. Er bestaan plannen om de seriële bus te virtualiseren zodat de MEGA65 schijfimages kan gebruiken voor apparaten zoals de 1541.

Er zijn delen van de hardware die alleen Hyppo kan benaderen. Het is het enige onderdeel dat directe toegang heeft tot de interne en externe SD-kaarten. U dient de diensten van Hyppo te gebruiken als u vanuit uw programma's de bestanden en directories op de SD-kaarten wilt benaderen.

Terminologie

Wanneer u kennismaakt met Hyppo, kunt u in aanraking komen met termen die verwarrend kunnen zijn als u al bekend bent met andere onderdelen van de MEGA65.

Op de SD-kaart bevindt zich waarschijnlijk een bestand met de naam HICKUP.M65. Dit bestand werkt Hyppo bij naar nieuwe versies zonder dat een bijgewerkte core geïnstalleerd hoeft te worden. Het zou kunnen voorkomen dat u Hyppo soms Hickup hoort noemen vanwege deze sterke associatie.

Er zijn drie verschillende disk-besturingssystemen in de MEGA65.

- Binnen Hyppo bevindt zich Hyppo DOS, of kortweg HDOS. HDOS is bedoeld voor toegang tot het FAT32-bestandssysteem op de SD-kaarten. HDOS weet niets van Commodore-bestandssystemen. Het kan een image van een Commodore-bestandssysteem koppelen, maar het begrijpt niet wat er in de image staat.
- Binnen de KERNAL bevindt zich CBDOS. CBDOS is bedoeld voor toegang tot 1581-achtige bestandssystemen. CBDOS gebruikt de 45IO27 multifunctionele I/O-controller om de sectoren van een fysieke schijf te benaderen. CBDOS weet niets van SD-kaarten en het FAT32-bestandssysteem daarop. Hyppo virtualiseert een deel van de 45IO27 zodat CBDOS schijfimages kan benaderen alsof het fysieke schijven zijn.
- De externe schijfstations die op de seriële bus zijn aangesloten, hebben elk hun eigen DOS. Deze worden gebruikt voor toegang tot de bestandssystemen op hun respectievelijke fysieke schijven.

Het woord drive heeft voor elk van deze DOS's een andere betekenis.

- De drives in Hyppo zijn de partities van de interne en externe SD-kaarten. Wanneer de MEGA65 opstart, wijst Hyppo nummers toe aan de partities die het kan lezen.
- De drives in CBDOS zijn de fysieke schijfstations die op de 45IO27 multifunctionele I/O-controller zijn aangesloten — zoals het interne schijfstation — of de schijfimages die op de gevirtualiseerde 45IO27 zijn gekoppeld. De CBDOS-drives zijn doorgaans zichtbaar als stations 8 en 9.
- De drives in een extern station dat op de seriële bus is aangesloten, zijn de schijfstations binnen dat station.

Versies

Dit hoofdstuk beschrijft de diensten die beschikbaar zijn in Hyppo 1.2.

Er kunnen nieuwe Hyppo-diensten beschikbaar komen en bestaande Hyppo-diensten kunnen wijzigen of komen te vervallen. Een robuust programma gebruikt de hyppo_getversion-dienst om te controleren of het compatibel is met de Hyppo in de MEGA65 waarop het wordt uitgevoerd.

Gebruik

Wanneer u een Hyppo-dienst wilt gebruiken, gebruikt u geen JSR. Hyppo bevindt zich namelijk in een afzonderlijke ruimte ten opzichte van gewone code. Om er toegang toe te krijgen, moet de CPU overschakelen naar zijn hypervisor-modus.

Op de adressen $D640 - $D67F staat een reeks hypervisor-traps. Het schrijven naar deze adressen werkt niet zoals het schrijven naar andere adressen. In plaats van naar geheugen of I/O te schrijven, schakelt de CPU over naar de hypervisor-modus en start een Hyppo-dienst. Hoe de CPU dit doet, wordt beschreven in Hoofdstuk/Bijlage K op bladzijde K-3.

Welke Hyppo-dienst wordt gestart, hangt af van welke waarde u uit het A-register schrijft en welke trap u aanspreekt. Elke dienst die in dit hoofdstuk wordt beschreven, vermeldt welke waarde u moet schrijven en welke trap u moet gebruiken. U moet het A-register gebruiken wanneer u een trap activeert. Het schrijven van dezelfde waarde vanuit een ander register werkt niet.

Wanneer de Hyppo-dienst is afgerond, schakelt de CPU terug naar uw programma. Met uitzondering van de registers die een dienst gebruikt om waarden terug te geven, blijven de registers ongewijzigd.

Belangrijk De CPU kan het volgende byte in uw programma uitvoeren nadat de Hyppo-dienst is afgerond, maar dit is niet gegarandeerd. Plaats een CLV-instructie na de STA. Of de CPU de CLV nu wel of niet uitvoert, mag geen invloed hebben op uw programma. Indien uw programma afhankelijk is van de V-vlag, kunt u in plaats daarvan de NOP-instructie gebruiken. Wanneer u NOP gebruikt, moet u erop letten wanneer de CPU de NOP interpreteert als een prefix voor de volgende instructie. Om die reden verdient het de voorkeur om CLV boven NOP te gebruiken.

Foutcodes:

Indien de dienst succesvol was, wordt de C-vlag gezet.

Indien de dienst niet succesvol was, wordt de C-vlag gewist en komt een foutcode in het A-register. Er staat een tabel met foutcodes in de beschrijving van hyppo_geterrorcode.

Voorbeelden

De voorbeelden gebruiken de ACME-assembler. De ACME-assembler is niet verplicht. De Hyppo-diensten kunnen met elke assembler worden gebruikt.

De voorbeelden zijn doorgaans niet volledig. Ze gaan ervan uit dat ergens een foutverwerker met de naam error is gedefinieerd. Ze gaan er eveneens van uit dat ergens een overdrachtsgebied is gedefinieerd. hyppo_setup_transfer_area en hyppo_setname laten zien hoe u een overdrachtsgebied definieert.

HYPPO-DIENSTEN: BEKNOPT OVERZICHT

Hieronder volgt een beknopt overzicht van de Hyppo-diensten. Raadpleeg de volledige naslag voor details en voorbeelden.

Hyppo-naam	Categorie	Invoer	STA	Beschrijving
geterrorcode	Algemeen	A=$38	$D640	Huidige foutcode van Hyppo
getversion	Algemeen	A=$00	$D640	Versie van Hyppo, HDOS
setup_transfer_area	Algemeen	A=$3A, Y	$D640	Stelt het gebied in dat Hyppo gebruikt om gegevens over te dragen
chdir	SD-kaart	A=$0C	$D640	Wijzigt de huidige werkmap
cdrootdir	SD-kaart	A=$3C, X	$D640	Voert selectdrive uit en stelt de huidige werkmap in op de root
closeall	SD-kaart	A=$22	$D640	Sluit alle bestandsdescriptoren
closedir	SD-kaart	A=$16, X	$D640	Sluit een bestandsdescriptor voor een directory
closefile	SD-kaart	A=$20, X	$D640	Sluit een bestandsdescriptor voor een bestand
filedate	SD-kaart	A=$2C	$D640	Niet geïmplementeerd
findfile	SD-kaart	A=$34	$D640	Zoekt het eerste bestand waarvan de bestandsnaam overeenkomt met de huidige Hyppo-bestandsnaam
findfirst	SD-kaart	A=$30	$D640	Zoekt het eerste bestand waarvan de bestandsnaam overeenkomt met de huidige Hyppo-bestandsnaam
findnext	SD-kaart	A=$32	$D640	Zoekt een volgend bestand waarvan de bestandsnaam overeenkomt met de huidige Hyppo-bestandsnaam
fstat	SD-kaart	A=$28	$D640	Niet geïmplementeerd
getcurrentdrive	SD-kaart	A=$04	$D640	Het nummer van de momenteel geselecteerde SD-kaartpartitie
getcwd	SD-kaart	A=$0A	$D640	Niet geïmplementeerd
getdefaultdrive	SD-kaart	A=$02	$D640	De SD-kaartpartitie die Hyppo heeft geselecteerd tijdens het opstarten

vervolg ...

...vervolg

Hyppo-naam	Categorie	Invoer	STA	Beschrijving
getdrivesize	SD-kaart	A=$08	$D640	Niet geïmplementeerd
loadfile	SD-kaart	A=$36, XYZ	$D640	Laadt een bestand in chip-geheugen
loadfile_attic	SD-kaart	A=$3E, XYZ	$D640	Laadt een bestand in hyper-geheugen
mkdir	SD-kaart	A=$0E	$D640	Niet geïmplementeerd
mkfile	SD-kaart	A=$1E	$D640	Niet geïmplementeerd
opendir	SD-kaart	A=$12	$D640	Opent de huidige werkmap om de bestandsvermeldingen daarin te lezen
openfile	SD-kaart	A=$18	$D640	Opent een bestand op een station
readdir	SD-kaart	A=$14, XY	$D640	Leest de volgende FAT-maprecord in een doelgebied
readfile	SD-kaart	A=$1A	$D640	Leest de volgende sector van het huidige bestand in de sectorbuffer
rename	SD-kaart	A=$2A	$D640	Niet geïmplementeerd
rmdir	SD-kaart	A=$10	$D640	Niet geïmplementeerd
rmfile	SD-kaart	A=$26	$D640	Verwijdert een bestand
seekfile	SD-kaart	A=$24	$D640	Niet geïmplementeerd
selectdrive	SD-kaart	A=$06, X	$D640	Stelt de huidig geselecteerde SD-kaartpartitie in
setname	SD-kaart	A=$2E, Y	$D640	Stelt de huidige Hyppo-bestandsnaam in
writefile	SD-kaart	A=$1C	$D640	Schrijft de sectorbuffer naar het huidige bestand
attach	Schijfimage	A=$4A, X	$D640	Koppelt een schijfimage aan of los van het gevirtualiseerde F011-station
d81attach0	Schijfimage	A=$40	$D640	Verouderd
d81attach1	Schijfimage	A=$46	$D640	Verouderd
d81detach	Schijfimage	A=$42	$D640	Verouderd
d81write_en	Schijfimage	A=$44	$D640	Maakt schrijven naar alle schijfimages die aan de gevirtualiseerde F011-stations 0 en 1 zijn gekoppeld, mogelijk
create_task_c64	Taak	A=$66	$D640	Niet geïmplementeerd
create_task_c65	Taak	A=$68	$D640	Niet geïmplementeerd
create_task_native	Taak	A=$62	$D640	Niet geïmplementeerd
exit_and_switch_to_task	Taak	A=$6A	$D640	Niet geïmplementeerd

vervolg ...

...vervolg

Hyppo-naam	**Categorie**	**Invoer**	**STA**	**Beschrijving**
exit_task	Taak	A=$6E	$D640	Niet geïmplementeerd
get_mapping	Taak	A=$74	$D640	Kopieert de huidige 45GS02-geheugenindeling naar een doelgebied
get_proc_desc	Taak	A=$58	$D640	Kopieert het huidige taak-blok naar een doelgebied
gettasklist	Taak	A=$50	$D640	Niet geïmplementeerd
load_into_task	Taak	A=$64	$D640	Niet geïmplementeerd
readoutoftask	Taak	A=$58	$D640	Niet geïmplementeerd
receivemessage	Taak	A=$54	$D640	Niet geïmplementeerd
reset	Taak	A=$7E	$D640	Voert een warme herstart van de MEGA65 uit
rom_writeenable	Taak	A=$02	$D641	Maakt schrijven naar bank 2 en 3 mogelijk
rom_writeprotect	Taak	A=$00	$D641	Schakelt schrijven naar bank 2 en 3 uit
sendmessage	Taak	A=$52	$D640	Niet geïmplementeerd
serial_monitor_wait _and_write	Taak	A=byte	$D643	Wacht tot de seriële debug-ger gereed is en schrijft een teken naar deze debugger
serial_monitor_write	Taak	A=$7C, Y=byte	$D640	Schrijft een teken naar de seriële debugger, zonder te wachten
set_mapping	Taak	A=$76, Y	$D640	Kopieert het brongebied naar de huidige 45GS02-geheugenindeling
switch_to_task	Taak	A=$6C	$D640	Niet geïmplementeerd
terminateothertask	Taak	A=$60	$D640	Niet geïmplementeerd
toggle_force_4502	Taak	A=$72	$D640	Wisselt de CPU tussen de 45GS02- en 6502-modus
toggle_rom_writeprotect	Taak	A=$70	$D640	Schakelt de schrijfbeveili-ging van bank 2 en 3 om
writeintotask	Taak	A=$56	$D640	Niet geïmplementeerd
configsector_apply	Configuratie	A=$04	$D642	Past het systeemconfigura-tiesector toe dat momenteel in het geheugen is geladen
configsector_read	Configuratie	A=$00	$D642	Leest het systeemconfigura-tiesector in het geheugen
configsector_write	Configuratie	A=$02	$D642	Schrijft het systeemcon-figuratiesector vanuit het geheugen

vervolg ...

...vervolg

Hyppo-naam	Categorie	Invoer	STA	Beschrijving
dmagic_autoset	Configuratie	A=$06	$D642	Stelt de DMAgic-revisie in op basis van de geladen ROM
freeze_self	Freezer	A=lukraak	$D67F	Start de Freezer
get_slot_count	Freezer	A=$16	$D642	Vraagt het aantal freeze-slots op
locate_freeze_slot	Freezer	A=$10	$D642	Zoekt de eerste sector van een freeze-slot
read_freeze_region_list	Freezer	A=$14	$D642	Leest de lijst met freezeregio's
unfreeze_from_slot	Freezer	A=$12	$D642	Herstelt vanuit een freeze-slot

ALGEMENE DIENSTEN

hyppo_geterrorcode

Trap: LDA #$38 : STA $D640 : CLV

Dienst: Geeft de huidige foutcode van Hyppo terug.

Voorbeding: De C-vlag was gewist door de vorige dienst.

Resultaat: **A** De foutcode van de voorgaande mislukte dienst.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: De foutcode is alleen geldig als de vorige Hyppo-dienst de C-vlag heeft gewist. Indien de C-vlag was gezet, is er geen fout opgetreden en is de Hyppo-foutcode ongedefinieerd.

De betekenissen hier zijn algemeen. Zie de secties van de diensten voor meer specifieke betekenissen.

Error codes: Dit is mogelijk geen volledige lijst.

Hex	Dec	Naam	Algemene betekenis
$01	1	partition not interesting	De partitie heeft geen ondersteund type.
$02	2	bad signature	De handtekeningsbytes aan het einde van een partitietabel of van de eerste sector van een partitie ontbraken of waren onjuist.
$03	3	is small FAT	Dit is een FAT12- of FAT16-partitie. Er wordt alleen FAT32 ondersteund.
$04	4	too many reserved clusters	De partitie heeft meer dan 65.535 gereserveerde sectoren.
$05	5	not two FATs	De partitie heeft niet precies twee kopieën van de FAT-structuur.
$06	6	too few clusters	De partitie bevat te weinig clusters.
$07	7	read timeout	Het duurde te lang om van de SD-kaart te lezen.
$08	8	partition error	Er is een ongespecificeerde fout opgetreden bij het verwerken van een partitie.
$10	16	invalid address	Er is een ongeldig adres opgegeven in een parameter.
$11	17	illegal value	Er is een ongeldige waarde opgegeven in een parameter.

Hex	Dec	Naam	Algemene betekenis
$20	32	read error	Er is een ongespecificeerde fout opgetreden bij het lezen.
$21	33	write error	Er is een ongespecificeerde fout opgetreden bij het schrijven.
$80	128	no such drive	Het opgegeven Hyppo-stationnummer bestaat niet.
$81	129	name too long	De opgegeven bestandsnaam was te lang.
$82	130	not implemented	De Hyppo-service is niet geïmplementeerd.
$83	131	file too long	Het bestand is groter dan 16 MB.
$84	132	too many open files	Alle bestandsdescriptoren zijn in gebruik.
$85	133	invalid cluster	Het opgegeven clusternummer is ongeldig.
$86	134	is a directory	Er is geprobeerd een handeling uit te voeren op een directory, terwijl een normaal bestand werd verwacht.
$87	135	not a directory	Er is geprobeerd een handeling uit te voeren op een normaal bestand, terwijl een directory werd verwacht.
$88	136	file not found	Het bestand kon niet worden gevonden in de huidige directory van het huidige station.
$89	137	invalid file descriptor	Er is een ongeldige of gesloten bestandsdescriptor opgegeven.
$8A	138	image wrong length	Het schijfimage-bestand heeft een onjuiste lengte.
$8B	139	image fragmented	Het schijfimage is niet aaneengesloten opgeslagen op de SD-kaart.
$8C	140	no space	De SD-kaart heeft geen vrije ruimte voor de gevraagde bewerking.
$8D	141	file exists	Er bestaat al een bestand met de opgegeven naam.
$8E	142	directory full	De directory kan geen extra vermeldingen bevatten.
$8F	143	double attach	Het image is al gekoppeld aan het andere station.
$FF	255	eof	Het einde van een bestand of directory werd bereikt.
$FF	255	no such trap	Er is geen Hyppo-service beschikbaar voor de trap. Het programma is mogelijk onverenigbaar met deze versie van Hyppo.

hyppo_getversion

Trap: LDA #$00 : STA $D640 : CLV

Dienst: Geeft de versie van Hyppo **A.X** en HDOS **Y.Z** weer.

Resultaat: **A** Het hoofdversienummer van Hyppo

X Het subversienummer van Hyppo

Y Het hoofdversienummer van HDOS

Z Het subversienummer van HDOS

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: De HDOS in Hyppo staat los van de CBDOS binnen de KERNAL of de DOS in de schijfstations die op de seriële poort zijn aangesloten.

Voorbeeld: Test of de versie van Hyppo $\geq$ 1.2 is en $<$ 2.0.

```
; Verkrijg de versienummers
LDA #$00 : STA $D640 : CLV
; Controleer of het hoofdversienummer van Hyppo in A gelijk is aan
; 1
CMP #1 : BNE incompatible
; Controleer of het subversienummer van Hyppo in X gelijk is aan 2
; of hoger
TXA : CMP #2 : BMI incompatible
; Deze Hyppo-versie is compatibel
```

hyppo_setup_transfer_area

Trap: LDA #$3A : STA $D640 : CLV

Dienst: Stelt het gebied in dat Hyppo gebruikt om gegevens naar uw programma over te dragen en ervan te ontvangen.

Invoer: **Y** Het hoogste byte van het adres van het overdrachtsgebied

Foutcodes: **$10 invalid address** Het adres van het overdrachtsgebied in Y > $7E

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Het overdrachtsgebied moet tussen $0000 en $7E00 liggen. Het moet ook op een paginagrens beginnen. Het laagste byte van het adres moet $00 zijn.

Het overdrachtsgebied is 256 bytes lang voor de meeste services.

Het overdrachtsgebied wordt aangeduid met het huidige geheugenindeling van de CPU op het moment dat een service wordt uitgevoerd. Het is echter goed gebruik om het altijd in de onderste 32 KB van bank 0 te plaatsen.

Voorbeeld: Reserveert 256 bytes op een paginagrens en stelt deze in als het overdrachtsgebied.

```
;-------------------------------------------------------------------------------
; Ergens in het gegevensgebied van het programma

    ; Uitlijnen op de nieuwe paginagrens
    !align 255, 0

transferarea:
    ; 256 bytes reserveren
    !skip 256

;-------------------------------------------------------------------------------
; Elders in de programmacode

setuptransferarea:
    ; Stel de transferarea in als het Hyppo-overdrachtsgebied
    LDY #>transferarea : LDA #$3A : STA $D640 : CLV : BCC error
```

STATION-/OPSLAGSERVICES

In Hyppo zijn stations de partities van de interne en externe SD-kaarten. Het zijn niet de station 0 en station 1 van de F011-floppycontroller. Het zijn ook niet de station 0 en station 1 van dual-drive-units die op de seriële bus zijn aangesloten.

hyppo_chdir

Trap: LDA #$0C : STA $D640 : CLV

Dienst: Wijzigt de huidige werkmap.

Voorbeding: De FAT-dir-vermelding van de directory waarnaar u wilt overschakelen, is gevonden. hyppo_findfile wordt doorgaans gebruikt om een FAT-dir-vermelding te vinden. hyppo_findfirst, hyppo_findnext en hyppo_readdir kunnen ook worden gebruikt.

Foutcodes: **$87 not a directory** De laatst gevonden FAT-dir-vermelding is geen directory. Bit 4 van het attribuutbyte van de FAT-dir-vermelding is gezet voor directories.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: U kunt naar de bovenliggende directory navigeren door de FAT-dir-vermelding .. te vinden.

U kunt niet meer dan één directory tegelijk omhoog of omlaag navigeren.

Gebruik hyppo_cdrootdir om direct terug te keren naar de hoofdmap.

Voorbeeld: Wijzigt het pad naar een willekeurig pad op het huidige station. Roep aan met Y:X als het adres van het pad. Het laatste teken van elk onderdeel in het pad moet bit 7 ingesteld hebben. Het gehele pad wordt afgesloten met een $00. Om bijvoorbeeld naar DIR1 en vervolgens DIR2 te wisselen, is het pad `!text "DIR", '1'+$80, "DIR" '2'+$80, 0`.

Bij succes wordt de C-vlag ingesteld. Als een deel van het pad niet bestaat, wordt de C-vlag gewist en is de huidige werkmap de map die het laatst succesvol is bereikt.

```
    !addr pathptr = $C4
chdirpath:
    STX pathptr : STY pathptr+1 ; Stel pathptr in op Y:Z
@chdirpath10:
    JSR @copycomponent ; Haal het volgende onderdeel van het pad op
    BEQ @chdirpath20 ; Stop bij een nulbyte in het pad
```

... gaat verder op de volgende bladzijde ...

```
    JSR @trychdir ; Probeer van map te veranderen
    BCC @chdirpath30 ; Stop als we niet van map kunnen veranderen
    BRA @chdirpath10 ; Herhaal met het volgende onderdeel
@chdirpath20: SEC
@chdirpath30: RTS
;-------------------------------------------------------------------------------
@trychdir:
    ; Stel de hyppo-bestandsnaam in vanuit transferbuffer
    LDY #>transferbuffer : LDA #$2E : STA $D640 : CLV : BCC @trychdir10
    ; Zoek de FAT-dir-entry
    LDA #$34 : STA $D640 : CLV : BCC @trychdir10
    ; Verander van map
    LDA #$0C : STA $D640 : CLV
@trychdir10:
    RTS
;-------------------------------------------------------------------------------
    ; Kopieer het volgende onderdeel van het pad naar transferbuffer
@copycomponent:
    LDY #0
@copycomponent10:
    LDA (pathptr),Y
    BEQ @copycomponent20
    TAX
    AND #$7F : STA transferbuffer,Y
    INY
    TXA : BPL @copycomponent10
    ; Voeg een nulbyte toe aan transferbuffer
    LDA #0 : STA transferbuffer,Y
    ; Verplaats pathptr
    TYA
    CLC : ADC pathptr : STA pathptr : LDA #0 : ADC pathptr+1 : STA pathptr+1
@copycomponent20:
    RTS
```

hyppo_cdrootdir

Trap: LDA #$3C : STA $D640 : CLV

Dienst: Wisselt het station door hyppo_selectdrive aan te roepen en stelt de huidige map in op de hoofdmap.

Invoer: **X** Het stationnummer dat het nieuwe huidige station wordt

Voorbeding: Geen.

Resultaat: **A** Een foutmelding die door hyppo_selectdrive kan worden geretourneerd

Geschiedenis: Beschikbaar vanaf Hyppo 1.16

hyppo_closeall

Trap: LDA #$22 : STA $D640 : CLV

Dienst: Sluit alle bestandsdescriptors.

Nawerking: Het gebruik van een willekeurige bestandsbestandsdescriptor met hyppo_closedir of hyppo_closefile slaagt.

Het gebruik van een willekeurige bestandsdescriptor met hyppo_readdir of hyppo_readfile mislukt.

hyppo_opendir en hyppo_openfile hergebruiken de bestandsdescriptor.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: U kunt ook afzonderlijke bestandsdescriptors sluiten met hyppo_closedir of hyppo_closefile.

hyppo_closedir

Trap: LDA #$16 : STA $D640 : CLV

Dienst: Sluit een bestandsdescriptor voor een map.

Voorbeding: De bestandsdescriptor die is opgegeven in het X-register is geopend met hyppo_opendir.

Invoer: **X** De bestandsdescriptor voor de map

Nawerking: Het opnieuw gebruiken van de bestandsdescriptor met hyppo_closedir slaagt.

Het opnieuw gebruiken van de bestandsdescriptor met hyppo_readdir mislukt.

hyppo_opendir en hyppo_openfile hergebruiken de bestandsdescriptor.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: U kunt ook alle geopende bestandsdescriptors sluiten met hyppo_closeall.

Voorbeeld: Zie het voorbeeld in hyppo_opendir.

hyppo_closefile

Trap: LDA #$20 : STA $D640 : CLV

Dienst: Sluit een bestandsdescriptor voor een bestand.

Voorbeding: De bestandsdescriptor die is opgegeven in het X-register is geopend met hyppo_openfile.

Invoer: **X** De bestandsdescriptor voor het bestand

Nawerking: Het opnieuw gebruiken van de bestandsdescriptor met hyppo_closefile slaagt.

Het opnieuw gebruiken van de bestandsdescriptor met hyppo_readfile mislukt.

hyppo_opendir en hyppo_openfile hergebruiken de bestandsdescriptor.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: U kunt ook alle geopende bestandsdescriptors sluiten met hyppo_closeall.

hyppo_filedate

Trap: LDA #$2C : STA $D640 : CLV

Dienst: Stelt de tijdstempel van een bestand in.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_findfile

Trap: LDA #$34 : STA $D640 : CLV

Dienst: Zoekt het eerste bestand waarvan de bestandsnaam overeenkomt met de huidige Hyppo-bestandsnaam.

Voorbeding: De huidige Hyppo-bestandsnaam is ingesteld met hyppo_setname.

Nawerking: Er zijn geen extra bestandsdescriptors geopend.

Foutcodes: **$88 file not found** Er is geen overeenkomstig bestand gevonden in de huidige map van het huidige station.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Hyppo vindt alleen bestanden waarvan de lange bestandsnaam geheel in hoofdletters staat. Hyppo converteert de huidige bestandsnaam naar ASCII-hoofdletters voordat het een overeenkomst probeert te vinden. Bytes $61 - $7B worden $41 - $5A.

Hyppo ondersteunt nog niet de jokertekens * en ?. Ondersteuning daarvoor is gepland in een toekomstige versie.

Dit vindt alleen het eerste overeenkomende bestand. U kunt meerdere overeenkomsten vinden met hyppo_findfirst en hyppo_findnext.

Voorbeeld: Zie het voorbeeld in hyppo_openfile.

hyppo_findfirst

Trap: LDA #$30 : STA $D640 : CLV

Dienst: Zoekt het eerste bestand waarvan de bestandsnaam overeenkomt met de huidige Hyppo-bestandsnaam.

Voorbeding: De huidige Hyppo-bestandsnaam is ingesteld met hyppo_setname.

Resultaat: **A** De bestandsdescriptor voor het lezen van de huidige werkmap. U bent mogelijk verantwoordelijk voor het sluiten van deze bestandsdescriptor met hyppo_closedir. Zie de opmerkingen.

Nawerking: hyppo_findnext vindt het volgende overeenkomende bestand of mislukt met een foutmelding dat het bestand niet is gevonden.

Neveneffect: Stelt de huidige bestandsdescriptor in.

Foutcodes: **$88 file not found** Er is geen overeenkomstig bestand gevonden in de huidige map van het huidige station.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Als Hyppo een eerste overeenkomend bestand vindt, stelt het de C-vlag in en retourneert het een bestandsdescriptor in het A-register. Dit is een bestandsdescriptor voor het lezen van de huidige werkmap. U bent verantwoordelijk voor het sluiten ervan met hyppo_closedir. Het is een standaard mapbestandsdescriptor. U kunt hyppo_readdir gebruiken om de FAT-mapitems na het gevonden bestand te lezen.

Als Hyppo geen overeenkomende bestanden vindt, mislukt het met een foutmelding dat het bestand niet is gevonden. In dat geval heeft Hyppo de bestandsdescriptor al gesloten en hoeft u die niet te sluiten.

Hyppo vindt alleen bestanden waarvan de lange bestandsnaam geheel in hoofdletters staat. Hyppo converteert de huidige bestandsnaam naar ASCII-hoofdletters voordat het een overeenkomst probeert te vinden. Bytes $61 - $7B worden $41 - $5A.

Hyppo ondersteunt nog niet de jokertekens * en ?. Ondersteuning daarvoor is gepland in een toekomstige versie.

Als u alleen de eerste overeenkomst zoekt, kunt u hyppo_findfile gebruiken. hyppo_findfile sluit de bestandsdescriptor altijd voor u. U kunt het echter niet gebruiken om meerdere overeenkomende bestanden te vinden.

Voorbeeld: Zie het voorbeeld in hyppo_findnext.

hyppo_findnext

Trap: LDA #$32 : STA $D640 : CLV

Dienst: Zoekt een volgend bestand waarvan de bestandsnaam overeenkomt met de huidige Hyppo-bestandsnaam.

Voorbeding: De huidige Hyppo-bestandsnaam is ingesteld met hyppo_setname.

Het eerste overeenkomende bestand is reeds succesvol gevonden met hyppo_findfirst.

Nawerking: Het opnieuw gebruiken van hyppo_findnext vindt het volgende overeenkomende bestand of mislukt met een foutmelding dat het bestand niet is gevonden.

Foutcodes: **$88 file not found** Er is geen volgend overeenkomend bestand gevonden in de huidige map van het huidige station.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Als Hyppo geen volgend overeenkomend bestand vindt, mislukt de aanroep met een file not found-fout. Hyppo sluit tevens de bestandsdescriptor die in hyppo_findfirst is uitgegeven.

Indien u de zoekopdracht niet uitput door hyppo_findnext te blijven aanroepen totdat deze met een file not found-fout mislukt, bent u verplicht de bestandsdescriptor zelf te sluiten via hyppo_closedir.

Voorbeeld: Geeft het X-register terug met het aantal bestanden dat overeenkomt met de huidige Hyppo-bestandsnaam in de huidige werkmap van het huidige station. Hoewel een directory in theorie meerdere bestanden met een identieke naam kan bevatten, wordt dit voorbeeld nuttiger zodra Hyppo jokertekens met * en ? ondersteunt.

```
    ; Neem aan dat de huidige hyppo-bestandsnaam al ingesteld is.
    ; Herstel de teller in X.
    LDX #0
    ; Probeer het eerste overeenkomende bestand te vinden.
    LDA #$30
@10:
    STA $D640 : CLV : BCC @20
    ; Verhoog de teller in X
    INX
    ; Probeer het volgende overeenkomende bestand te vinden.
    LDA #$32
    BRA @10
@20:
```

... gaat verder op de volgende bladzijde ...

```
; Als de foutcode in A $88 is, zijn er geen overeenkomende
; bestanden meer; anders is er een fout opgetreden.
CMP #$88 : BNE error
; Sluiten is niet nodig omdat de zoekactie voltooid is.
```

hyppo_fstat

Trap: LDA #$28 : STA $D640 : CLV

Dienst: Geeft informatie over een bestand.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_getcurrentdrive

Trap: LDA #$04 : STA $D640 : CLV

Dienst: Geeft het nummer van het huidige geselecteerde station (SD-kaartpartitie) terug.

Resultaten: **A** Het huidige stationsnummer

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: hyppo_selectdrive wijzigt het huidige stationsnummer. hyppo_cdrootdir kan het eveneens wijzigen.

Voorbeeld: Drukt het nummer van het huidige geselecteerde station in de linkerbovenhoek van het scherm. Dit voorbeeld gaat ervan uit dat er niet meer dan 10 stations (stations 0 tot en met 9) zijn. Het gaat er tevens vanuit dat het schermgeheugen niet is verplaatst vanaf $800.

```
; Verkrijg het huidige station
LDA #$04 : STA $D640 : CLV : BCC error

; Zet het stationsnummer in A om naar een schermcode
CLC : ADC #$30

; Zet de schermcode in de linkerbovenhoek van het schermgeheugen
STA $0800
```

hyppo_getcwd

Trap: LDA #$0A : STA $D640 : CLV

Dienst: Geeft informatie over de huidige geselecteerde directory of subdirectory.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_getdefaultdrive

Trap: LDA #$02 : STA $D640 : CLV

Dienst: Geeft het stationsnummer (SD-kaartpartitie) terug dat Hyppo tijdens het opstarten heeft geselecteerd.

Resultaten: **A** Het standaard stationsnummer

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Voorbeeld: Selecteert het standaardstation.

```
; Verkrijg het standaardstation
LDA #$02 : STA $D640 : CLV : BCC @error

; Kopieer het stationsnummer van A naar X
TAX

; Selecteer het standaardstation
LDA #$06 : STA $D640 : CLV : BCC @error
```

hyppo_getdrivesize

Trap: LDA #$08 : STA $D640 : CLV

Dienst: Geeft informatie over de grootte van het huidige geselecteerde station (SD-kaartpartitie).

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_loadfile

Trap: LDA #$36 : STA $D640 : CLV

Dienst: Laadt een bestand in chip memory.

Voorwaarden: De naam van het te laden bestand is ingesteld met hyppo_setname.

Invoer: **X** Het LSB van het adres waarmee het laden begint

Y Het middelste byte van het adres waarmee het laden begint

Z Het MSB van het adres waarmee het laden begint

Nawerkingen: Er zijn geen extra bestandsdescriptors geopend.

Foutcodes: **$84 too many open files** hyppo_loadfile maakt intern gebruik van één bestandsdescriptor, maar alle bestandsdescriptors zijn in gebruik. Sluit enkele of alle bestandsdescriptors met hyppo_closedir, hyppo_closefile of hyppo_closeall.

$88 file not found Het bestand is niet gevonden in de huidige directory van het huidige station.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Deze dienst kan bestanden tot 16 MB groot laden in de eerste 16 MB van chip memory. Chip memory is het geheugen van 384 KB of meer in de CPU-module.

Het laden begint op het 28-bit-adres $00ZZYYXX. Indien het laden voorbij $00FFFFFF probeert te gaan, slaat het om en gaat verder op $00000000.

U kunt hyppo_loadfile_attic gebruiken om een bestand in hyper memory te laden. Hyper memory is het geheugen van 8 MB of meer in de externe RAM-chips.

Voorbeeld: Laadt een bestand in het geheugen vanaf $48000.

```
; Neem aan dat de huidige hyppo-bestandsnaam al is ingesteld.
; Het is niet nodig het bestand eerst te zoeken.
LDZ #$04 ; Meest significante byte
LDY #$80 ; Middelste byte
LDX #$00 ; Minst significante byte
LDA #$36 : STA $D640 : CLV : BCC error
```

hyppo_loadfile_attic

Trap: LDA #$3E : STA $D640 : CLV

Dienst: Laadt een bestand in hyper memory.

Voorwaarden: De naam van het te laden bestand is ingesteld met hyppo_setname.

Invoer: **X** Het LSB van het adres waarmee het laden begint

Y Het middelste byte van het adres waarmee het laden begint

Z Het MSB van het adres waarmee het laden begint

Nawerkingen: Er zijn geen extra bestandsdescriptors geopend.

Foutcodes: **$84 too many open files** hos_loadfile_attic maakt intern gebruik van één bestandsdescriptor, maar alle bestandsdescriptors zijn in gebruik. Sluit enkele of alle bestandsdescriptors met hyppo_closedir, hyppo_closefile of hyppo_closeall.

$88 file not found Het bestand is niet gevonden in de huidige directory van het huidige station.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Deze dienst kan bestanden tot 16 MB groot laden in de eerste 16 MB van hyper memory. Hyper memory is het geheugen van 8 MB of meer in de externe RAM-chips.

Het laden begint op het 28-bit-adres $08ZZYYXX. Indien het laden voorbij $08FFFFFF probeert te gaan, slaat het om en gaat verder op $08000000.

U kunt hyppo_loadfile gebruiken om een bestand in chip memory te laden. Chip memory is het geheugen van 384 KB of meer in de CPU-module.

hyppo_mkdir

Trap: LDA #$0E : STA $D640 : CLV

Dienst: Maakt een subdirectory aan.

Foutcodes: **$8D file exists** Er bestaat al een subdirectory of bestand met de huidige Hyppo-bestandsnaam in de huidige werkmap van het huidige station.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_mkfile

Trap: LDA #$1E : STA $D640 : CLV

Dienst: Maakt een bestand aan.

Foutcodes: **$8D file exists** Er bestaat al een subdirectory of bestand met de huidige Hyppo-bestandsnaam in de huidige werkmap van het huidige station.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_opendir

Trap: LDA #$12 : STA $D640 : CLV

Dienst: Opent de huidige werkmap om de bestandsvermeldingen erin te lezen.

Voorwaarden: Het station en de directory die u wilt lezen, zijn indien nodig al ingesteld met hyppo_selectdrive en hyppo_chdir.

Resultaten: **A** De bestandsdescriptor voor het lezen van de directory. U bent zelf verantwoordelijk voor het sluiten van deze bestandsdescriptor met hyppo_closedir.

Nawerkingen: hyppo_readdir leest de eerste FAT-dir-entry in de directory.

Foutcodes: **$84 too many open files** Alle bestandsdescriptors zijn in gebruik. hyppo_opendir en hyppo_openfile maken gezamenlijk gebruik van dezelfde kleine voorraad bestandsdescriptors. Sluit enkele of alle bestandsdescriptors met hyppo_closedir, hyppo_closefile of hyppo_closeall.

$87 not a directory De laatst gevonden FAT-dir-entry is van een bestand. Gebruik hyppo_openfile voor bestanden.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Voorbeeld: Roept processdirentry aan voor elke FAT-dir-entry in de huidige werkmap. processdirentry wordt elders gedefinieerd verondersteld.

```
    ; Open de huidige werkmap
    LDA #$12 : STA $D640 : CLV : BCC error
    ; Verplaats de bestandsdescriptor naar X
    TAX
    ; Stel Y in op het MSB van het overdrachtsgebied
    LDY #)transferarea
@10:
    ; Lees het directory-item
    LDA #$14 : STA $D640 : CLV : BCC @20
    ; Roep processdirentry aan (wordt elders gedefinieerd)
    PHX : PHY : JSR processdirentry : PLY : PLX
    BRA @10
@20:
    ; Als de foutcode in A $85 is, hebben we het einde van de map
    ; bereikt; anders is er een fout opgetreden
    CMP #$85 : BNE error
    ; Sluit de bestandsindicator in X
    LDA #$16 : STA $D640 : CLV : BCC error
```

hyppo_openfile

Trap: LDA #$18 : STA $D640 : CLV

Dienst: Opent een bestand op een station.

Voorwaarden: Het bestand is al gevonden. Bestanden kunnen worden gevonden met hyppo_findfile, hyppo_findfirst en hyppo_findnext. hyppo_readdir kan ook worden gebruikt om een bestand te vinden.

Resultaten: **A** Het bestandsdescriptor voor toegang tot het bestand. U bent zelf verantwoordelijk voor het sluiten van deze bestandsdescriptor met hyppo_closefile.

Nawerkingen: Het gebruik van hyppo_readfile met deze bestandsdescriptor leest de eerste sector van het bestand.

Neveneffecten: Stelt het huidige bestand in op het nieuw geopende bestand. hyppo_readfile leest vanuit het huidige bestand.

Foutcodes: **$84 too many open files** Alle bestandsdescriptors zijn in gebruik. hyppo_opendir en hyppo_openfile maken gezamenlijk gebruik van dezelfde kleine voorraad bestandsdescriptors. Sluit enkele of alle bestandsdescriptors met hyppo_closedir, hyppo_closefile of hyppo_closeall.

$86 is a directory De laatst gevonden FAT-dir-entry is van een directory. Gebruik hyppo_opendir voor directories.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: U kunt dit niet gebruiken om een bestand binnen een schijfimage te openen. Gebruik in plaats daarvan hyppo_attach om de schijfimage te koppelen en gebruik vervolgens de KERNAL om het bestand te lezen of programmeer de virtuele F011-floppycontroller.

Voorbeeld: Zoekt en opent een bestand.

```
; Neem aan dat de huidige hyppo-bestandsnaam al is ingesteld.
; Zoek het bestand
LDA #$34 : STA $D640 : CLV : BCC error
; Open het bestand
LDA #$18 : STA $D640 : CLV : BCC error
```

hyppo_readdir

Trap: LDA #$14 : STA $D640 : CLV

Dienst: Leest de volgende FAT-dir-entry in een bestemmingsgebied.

Voorwaarden: De bestandsdescriptor die in het X-register is opgegeven, is geopend met hyppo_opendir en hyppo_closedir is sindsdien niet gebruikt om deze te sluiten.

Het bestemmingsgebied bevindt zich op een paginagrens tussen $0000 en $7E00 en is ten minste 87 bytes groot.

Invoer: **X** De bestandsdescriptor voor de directory.

Y Het MSB van het bestemmingsgebied.

Resultaten: Vanaf $YY00 heeft de FAT-dir-entry de volgende structuur.

Offset	Type	Omschrijving
$00	asciiz	De lange bestandsnaam
$40	byte	De lengte van de lange bestandsnaam
$41	ascii	De "8.3"-bestandsnaam. Het naamgedeelte wordt aangevuld met spaties zodat het precies 8 bytes is. De 3 bytes van de extensie volgen. Er staat geen punt tussen de naam en de extensie. Er is geen NULL-byte.
$4E	dword	Het clusternummer waar het bestand begint. Voor sub-directories is dit de plaats waar de FAT-dir-entries voor die subdirectory beginnen.
$52	dword	De lengte van het bestand in bytes.
$56	byte	Het type- en attribuutbyte.

Dit is de betekenis van de bits in het laatste byte. Bits 6 en 7 zijn ongedefinieerd.

Bit	Betekenis indien bit ingesteld
0	Alleen lezen
1	Verborgen
2	Systeem
3	Volumelabel
4	Sub-directory
5	Archief

Nawerkingen: Het opnieuw gebruiken van hyppo_readdir leest de volgende FAT-dir-entry in de directory.

Foutcodes: **$08 partition error** Er is een ongespecificeerde fout opgetreden bij het verwerken van de huidig geselecteerde partitie.

$10 invalid address Het Y-register is > $7E.

$85 invalid cluster Er is geprobeerd voorbij het einde van de directory te lezen.

Opmerkingen: Indien de lange bestandsnaam in de FAT-dir-entry te lang is om in het bestemmingsgebied te kopiëren, slaat Hyppo de entry geheel over.

De bestandsnamen in FAT zijn gecodeerd als UTF-16. Hyppo leest alleen het LSB van elk 16-bit teken. Hyppo converteert bestandsnamen niet naar PETSCII.

Zie hyppo_setup_transfer_area voor meer details over de waarde van het Y-register.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Voorbeeld: Zie het voorbeeld bij hyppo_opendir.

hyppo_readfile

Trap: LDA #$1A : STA $D640 : CLV

Dienst: Leest het volgende sector van het huidige bestand in de sectorbuffer.

Voorwaarden: Er is een huidig bestand geopend. Bestanden kunnen worden geopend met hyppo_openfile.

Resultaten: **X** Het LSB van het aantal gelezen bytes

Y Het MSB van het aantal gelezen bytes

Nawerkingen: De volgende aanroep van hyppo_readfile leest het volgende sector van het huidige bestand of geeft het einde van het bestand aan.

Foutcodes: **$89 invalid file descriptor** Er is geen huidig bestand.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Om de gegevens te benaderen, moet u het volgende doen:

- de sectorbuffer in de 16-bit adresruimte te plaatsen;
- een verbeterde DMA-overdracht te gebruiken om de sectorbuffer op $FFD6E00 - $FFD6FFF naar een buffer te kopiëren die al in de 16-bit adresruimte is geplaatst; of
- 32-bit laadinstructies te gebruiken om de sectorbuffer direct te benaderen.

Indien een volledig sector is gelezen, is Y:X gelijk aan $0200. Voor het laatste sector van het bestand kan Y:X kleiner zijn. Bytes in de sectorbuffer na Y:X zijn ongedefinieerd en niet noodzakelijk nul.

Indien u voorbij het einde van het laatste sector leest, is Y:X gelijk aan $0000, is het A-register gelijk aan $FF en is de C-vlag ingesteld.

Hoewel meerdere bestanden tegelijkertijd kunnen worden geopend, kan alleen het huidige bestand worden gelezen. Het huidige bestand is meestal het laatst geopende bestand, maar niet altijd.

Voorbeeld: Plaatst de sectorbuffer op $DE00 en leest vervolgens elk sector van het bestand door voor elk gelezen sector een aanroep te doen naar proccesssector. proccesssector wordt elders gedefinieerd verondersteld.

```
; Neem aan dat het bestand al geopend is.
; Koppel kleuren-RAM los van $DC00 om de sectorbuffer te kunnen
; inladen
```

... gaat verder op de volgende bladzijde ...

```
    LDA $D030 : PHA : AND #%11111110 : STA $D030
@10:
    ; Lees de volgende sector
    LDA #$1A : STA $D640 : CLV : BCC @20
    ; Breng de sectorbuffer naar $DE00
    LDA #$81 : STA $D680
    ; Roep processsector aan (wordt elders gedefinieerd)
    JSR processsector
    ; Koppel de sectorbuffer los van $DE00
    LDA #$82 : STA $D680
    BRA @10
@20:
    ; Als de foutcode in A $FF is, hebben we het einde van het bestand
    ; bereikt; anders is er een fout opgetreden
    CMP #$FF : BNE error
    ; Breng kleuren-RAM terug naar $DC00 als het daarvoor in het
    ; geheugen stond
    PLA : STA $D030
```

hyppo_rename

Trap: LDA #$2A : STA $D640 : CLV

Dienst: Hernoemt een bestand of subdirectory.

Foutcodes: **$8D file exists** Er bestaat al een subdirectory of bestand met de huidige Hyppo-bestandsnaam in de huidige werkmap van het huidige station.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_rmdir

Trap: LDA #$10 : STA $D640 : CLV

Dienst: Verwijdert een subdirectory.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_rmfile

Trap: LDA #$26 : STA $D640 : CLV

Dienst: Verwijdert een bestand.

Voorbeding: Het bestand is reeds gevonden. Bestanden kunnen worden gevonden met hyppo_findfile, hyppo_findfirst en hyppo_findnext. hyppo_readdir kan ook worden gebruikt om een bestand te vinden.

Nawerking: Het bestand is verwijderd en de sectoren zijn vrijgegeven.

Geschiedenis: Beschikbaar vanaf Hyppo 1.3

Opmerkingen: Deze documentatie is onvolledig en in opbouw!

hyppo_seekfile

Trap: LDA #$24 : STA $D640 : CLV

Dienst: Zoekt naar een bepaalde sector in een bestand.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_selectdrive

Trap: LDA #$06 : STA $D640 : CLV

Dienst: Stelt het momenteel geselecteerde station (SD-kaartpartitie) in.

Voorbeding: Hyppo heeft een stationsnummer toegewezen aan de SD-kaartpartitie.

Invoer: **X** Het stationsnummer dat het nieuwe huidige station wordt

Nawerking: hyppo_getcurrentdrive geeft de waarde terug die in het X-register stond.

Hyppo-services werken op het nieuw geselecteerde station.

Foutcodes: **$80 no such drive** Het station in het X-register bestaat niet. Hyppo wijst alleen stationsnummers toe aan SD-kaartpartities die het kan lezen.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Voorbeeld: Test of station 2 bestaat door het te selecteren. Keert terug met de C-vlag ingesteld als station 2 bestaat.

```
doesdrive2exist:
    ; Bewaar het huidige station zodat we het later kunnen herstellen
    LDA #$04 : STA $D640 : CLV : BCC error
    PHA
    ; Probeer station 2 te selecteren
    LDX #2 : LDA #$06 : STA $D640 : CLV : BCC @10
    ; Herstel het eerder geselecteerde station
    PLX
    LDA #$06 : STA $D640 : CLV : BCC error
    ; De C-vlag was reeds gezet door de Hyppo-service
    RTS

@10:
    ; Als de foutcode in A $80 is, bestaat het station niet; anders is
    ; een andere fout opgetreden
    CMP #$80
    BNE error
    ; Negeer het huidige station dat we bewaarden, want het is niet
    ; gewijzigd
    PLX
    ; Wis de C-vlag omdat het station niet bestaat
    CLC
    RTS
```

hyppo_setname

Trap: LDA #$2E : STA $D640 : CLV

Dienst: Stelt de huidige Hyppo-bestandsnaam in.

Voorbeding: De bestandsnaam is opgeslagen in ASCII en eindigt op een $00-byte.

De bestandsnaam begint op een paginagrens tussen $0000 en $7E00 en is korter dan 63 tekens, exclusief de $00-byte.

Invoer: **Y** Het MSB van het bestandsnaamadres.

Nawerking: Hyppo heeft de bestandsnaam gekopieerd naar het eigen gegevensgebied.

De hyppo_find*- en hyppo_load*-services gebruiken deze bestandsnaam.

Neveneffect: Stelt het overdrachtsgebied in op $YY00.

Foutcodes: **$10 invalid address** Het Y-register is > $7E.

$81 name too long De bestandsnaam is langer dan 63 tekens.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: De bestandsnaam moet tussen $0000 en $7E00 liggen. Deze moet ook op een paginagrens beginnen. Het adres moet dus eindigen op $00. De huidige geheugenindeling wordt gebruikt. Het is echter aan te raden om de bestandsnaam in de onderste 32KB van bank 0 te plaatsen.

De bestandsnamen in FAT worden gecodeerd in UTF-16. Hyppo leest alleen het LSB van elk 16-bit teken. Hyppo voert geen conversie uit tussen ASCII en PETSCII.

Hyppo geeft toegang tot de bestanden in het FAT-bestandssysteem op de interne en externe SD-kaarten. Het geeft geen toegang tot bestanden op schijven in floppy-stations of in schijfimages.

Voorbeeld: Stel de huidige Hyppo-bestandsnaam in op GAME.MAP.

```
; Ergens in het gegevensgebied van het programma
   !align 255, 0 ; Uitlijnen op de volgende paginagrens
filename:
   !text "GAME.MAP", 0 ; Moet eindigen met een nulbyte

; Elders in de programmacode
setfilename:
   LDY #>filename : LDA #$2E : STA $D640 : CLV : BCC error
```

hyppo_writefile

Trap: LDA #$1C : STA $D640 : CLV

Dienst: Schrijft de sectorbuffer naar het huidige bestand.

Voorbeding: Er is een huidig bestand geopend. Bestanden kunnen worden geopend met hyppo_openfile.

Nawerking: De volgende aanroep van hyppo_writefile schrijft de volgende sector van het huidige bestand.

Foutcodes: **$89 invalid file descriptor** Er is geen huidig bestand.

Geschiedenis: Beschikbaar vanaf Hyppo 1.3

Opmerkingen: Deze documentatie is onvolledig en in opbouw!

Om de gegevens aan te leveren, heeft u een van de volgende mogelijkheden:

- plaats de sectorbuffer in de 16-bit adresruimte;
- gebruik een verbeterde DMA-overdracht om de sectorbuffer van $FFD6E00 - $FFD6FFF te kopiëren naar een buffer die reeds in de 16-bit adresruimte is geladen; of
- gebruik 32-bit laadinstructies om direct toegang tot de sectorbuffer te krijgen.

SCHIJFIMAGE-SERVICES

De 45IO27 multifunctionele I/O-controller bevat een F011-compatibele floppycontroller. Het interne floppy-station is hierop aangesloten als station 0.

Hyppo kan de F011 floppy-controller virtualiseren zodat schijfimages kunnen worden gekoppeld in plaats van floppy-stations. Zodra een schijfimage is gekoppeld, vangt Hyppo de I/O-registers van de F011 af en emuleert de commando's op de schijfimage.

U kunt BASIC, de KERNAL en de F011 I/O-registers gebruiken om een schijfimage net als een fysieke schijf te bewerken. De virtualisatie gedraagt zich niet in alle gevallen hetzelfde als een floppy-station. Indien u wilt dat uw programma met zowel schijfimages als fysieke schijven werkt, test het dan met beide.

hyppo_attach

Trap: LDA #$4A : STA $D640 : CLV

Dienst: Koppelt of ontkoppelt een schijfimage aan het virtuele F011-station.

Voorbeding: (koppelen) De huidige Hyppo-bestandsnaam is ingesteld met hyppo_setname.

Invoer: **X.0** Selecteer station 0 of 1.

X.1 Bij X.0=1, selecteer beide stations (alleen ontkoppelen).

X.6 Herkoppel geen fysieke stations (alleen ontkoppelen).

X.7 Werkingsmodus: 0=koppelen, 1=ontkoppelen.

Foutcodes: **$81 name to long** De naam van de schijfimage is te lang voor de procesdescriptor (max. 31 tekens).

$88 file not found De schijfimage is niet gevonden in de huidige map van het huidige station.

$8A image wrong length De lengte van de image komt niet overeen met een van de ondersteunde imagetypen.

$8B image fragmented De gegevens van de image bevinden zich niet in één aaneengesloten reeks van sectoren; dit wordt niet ondersteund.

$8F double attach De gevraagde image is al gekoppeld aan het andere station.

Geschiedenis: Beschikbaar vanaf Hyppo 1.3

Opmerkingen: Tenzij dit is gewijzigd, is station 0 van de virtuele F011 floppy-controller apparaat 8 en station 1 apparaat 9.

Voorbeeld: Koppelt en ontkoppelt de schijfimage DISK2.D81 aan virtueel F011 station 0.

```
; Ergens in het gegevensgebied van het programma
    !align 255, 0 ; Lijn uit op de volgende paginagrens
disk2name:
    !text "DISK2.D81", 0 ; Moet eindigen met een nulbyte

; Elders in de code van het programma
attachdisk:
    ; Stel de hyppo-bestandsnaam in op DISK2.D81
    LDY #>disk2name : LDA #<2E : STA $D640 : CLV : BCC error
    ; Koppel de schijfimage aan
```

... gaat verder op de volgende bladzijde ...

```
    LDX #$00 : LDA #$4A : STA $D640 : CLV : BCC error
    RTS

detachdisk:
    ; Ontkoppel de schijfimage en herstel het fysieke station
    LDX #$80 : LDA #$4A : STA $D640 : CLV : BCC error
    RTS

good:
    LDX #$53 : STX $800 : RTS

error:
    LDX #$56 : STX $800 : STA $801 : RTS

main:
    JSR attachdisk
    JSR detachdisk
```

hyppo_d81attach0

Trap: LDA #$40 : STA $D640 : CLV

Dienst: Koppelt een D81-schijfimage aan virtueel F011 station 0.

Voorbeding: De huidige Hyppo-bestandsnaam is ingesteld met hyppo_setname.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2
VEROUDERD sinds Hyppo 1.3. Deze dienst kan in een toekomstige versie worden verwijderd. Nieuwe programma's moeten in plaats daarvan attach gebruiken.

Opmerkingen: Deze service aanroept eigenlijk hyppo_attach met **X** = $00

hyppo_d81attach1

Trap: LDA #$46 : STA $D640 : CLV

Dienst: Koppelt een D81-schijfimage aan virtueel F011 station 1.

Voorbeding: De huidige Hyppo-bestandsnaam is ingesteld met hyppo_setname.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

VEROUDERD sinds Hyppo 1.3. Deze dienst kan in een toekomstige versie worden verwijderd. Nieuwe programma's moeten in plaats daarvan attach gebruiken.

Opmerkingen: Deze dienst roept eigenlijk hyppo_attach aan met **X** = $01

hyppo_d81detach

Trap: LDA #$42 : STA $D640 : CLV

Dienst: Koppelt alle schijfimages los van de virtuele F011-stations 0 en 1.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

VEROUDERD sinds Hyppo 1.3. Deze dienst kan in een toekomstige versie worden verwijderd. Nieuwe programma's moeten in plaats daarvan attach gebruiken.

Opmerkingen: Deze dienst roept eigenlijk hyppo_attach aan met **X** = $C2

hyppo_d81write_en

Trap: LDA #$44 : STA $D640 : CLV

Dienst: Schakelt schrijven in voor alle schijfimages die gekoppeld zijn aan virtuele F011-stations 0 en 1.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Images worden standaard gekoppeld met schrijven ingeschakeld, zodat deze aanroep niet nodig is.

TAAK- EN PROCESDIENSTEN

hyppo_create_task_c64

Trap: LDA #$66 : STA $D640 : CLV

Dienst: Maakt een Hyppo-taak aan in de C64-achtige werking.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_create_task_c65

Trap: LDA #$68 : STA $D640 : CLV

Dienst: Maakt een Hyppo-taak aan in de C65-achtige werking.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_create_task_native

Trap: LDA #$62 : STA $D640 : CLV

Dienst: Maakt een Hyppo-taak aan in de MEGA65-werking.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_exit_and_switch_to_task

Trap: LDA #$6A : STA $D640 : CLV

Dienst: Sluit de huidige Hyppo-taak en schakelt de context over naar een andere Hyppo-taak.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_exit_task

Trap: LDA #$6E : STA $D640 : CLV

Dienst: Sluit de huidige Hyppo-taak.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_get_mapping

Trap: LDA #$74 : STA $D640 : CLV

Dienst: Kopieert de huidige 45GS02-geheugenindeling naar een bestemmingsgebied.

Voorwaarden: Het bestemmingsgebied begint op een paginagrens tussen $0000 en $7E00 en is minstens 6 bytes lang.

Invoer: **Y** Het hoogste byte van het bestemmingsgebied.

Uitvoer: Beginnend op $YY00 heeft de huidige indelingsinformatie de volgende structuur.

Offset	Type	Omschrijving
0	word	MAPLO (big-endian)
2	word	MAPHI (big-endian)
4	byte	Het megabyte-offset voor MAPLO
5	byte	Het megabyte-offset voor MAPHI

Foutcodes: **$10 invalid address** Het Y-register is > $7E.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: MAPLO is de indeling voor $0000 - $7FFF.

MAPHI is de indeling voor $8000 - $FFFF.

Zie Hoofdstuk/Bijlage K op bladzijde K-3 voor meer informatie over geheugenindeling en banken van de MEGA65.

hyppo_get_proc_desc

Trap: LDA #$48 : STA $D640 : CLV

Dienst: Kopieert het huidige taakblok naar een bestemmingsgebied.

Voorwaarden: Het bestemmingsgebied begint op een paginagrens tussen $0000 en $7E00 en is minstens 256 bytes lang.

Invoer: **Y** Het hoogste byte van het bestemmingsgebied.

Uitvoer: Beginnend op $YY00 heeft het huidige taakblok de volgende structuur.

Offset	Type	Omschrijving
$00	byte	Het ID van de huidige taak.
$01	text	De naam van de huidige taak. Maximaal 16 tekens. Opgevuld met $00-bytes. Bij 16 tekens staan er geen afsluitende $00-bytes.
$11	byte	Vlaggen voor de D81-schijfimage die gekoppeld is aan station 0 van de virtuele F011-floppycontroller.
$12	byte	Idem maar voor station 1.
$13	byte	De lengte van de bestandsnaam van de D81-schijfimage die gekoppeld is aan station 0.
$14	byte	Idem maar voor station 1.
$15	text	De bestandsnaam van de D81-schijfimage die gekoppeld is aan station 0. Maximaal 32 tekens. Opgevuld met $20-bytes. Er staan geen afsluitende $00-bytes.
$35	text	Idem maar voor station 1.
$55		De betekenis van deze bytes is ongedefinieerd en kan veranderen.
$80		Bestandsdescriptor 0.
$A0		Bestandsdescriptor 1.
$C0		Bestandsdescriptor 2.
$E0		Bestandsdescriptor 3.

Elke bestandsdescriptor heeft de volgende structuur.

Offset	Type	Omschrijving
$00	byte	Het nummer van de SD-kaartpartitie waarin het bestand staat. $FF betekent dat de bestandsdescriptor gesloten is.
$01	dword	Het cluster waar het bestand begint
$05	dword	Het huidige cluster
$09	byte	De huidige sector binnen het huidige cluster
$0A	dword	De lengte van het bestand
$0E	dword	De huidige positie binnen de buffer van het bestand
$12	dword	Het cluster van de directory waarin het bestand staat
$16	word	De index van het bestand binnen zijn directory
$18	dword	Het absolute 32-bits adres van de buffer van het bestand
$1C	word	Het aantal bytes dat in de buffer van het bestand wordt gebruikt
$1E	word	De huidige offset binnen de buffer van het bestand

Foutcodes: **$10 invalid address** Het Y-register is > $7E.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_gettasklist

Trap: LDA #$50 : STA $D640 : CLV

Dienst: Haalt de lijst met taken op uit Hyppo.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_load_into_task

Trap: LDA #$64 : STA $D640 : CLV

Dienst: Laadt een bestand vanaf een SD-kaartpartitie in het geheugen van een Hyppo-taak.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_readoutoftask

Trap: LDA #$58 : STA $D640 : CLV

Dienst: Leest uit het geheugen van een andere Hyppo-taak.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_receivemessage

Trap: LDA #$54 : STA $D640 : CLV

Dienst: Ontvangt berichten die door andere Hyppo-taken zijn verzonden.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_reset

Trap: LDA #$7E : STA $D640 : CLV

Dienst: Voert een warme start van de MEGA65 uit.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_rom_writeenable

Trap: LDA #$02 : STA $D641 : CLV

Dienst: Verandert $20000 - $3FFFF zodat ze zich als RAM gedragen door de schrijfbeveiliging uit te schakelen.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: $20000 - $3FFFF bevat normaal de KERNAL-, BASIC-, CBDOS- en lettertype-ROM's.

hyppo_rom_writeprotect schakelt de schrijfbeveiliging in en blokkeert schrijfbewerkingen. hyppo_toggle_rom_writeprotect wisselt de schrijfbeveiliging om.

hyppo_rom_writeprotect

Trap: LDA #$00 : STA $D641 : CLV

Dienst: Verandert $20000 - $3FFFF zodat ze zich als ROM gedragen door de schrijfbeveiliging in te schakelen.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: $20000 - $3FFFF bevat normaal de KERNAL-, BASIC-, CBDOS- en lettertype-ROM's.

hyppo_rom_writeenable schakelt de schrijfbeveiliging uit en staat schrijfbewerkingen toe. hyppo_toggle_rom_writeprotect wisselt de schrijfbeveiliging om.

hyppo_sendmessage

Trap: LDA #$52 : STA $D640 : CLV

Dienst: Verstuurt een bericht naar een andere Hyppo-taak.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_serial_monitor_wait_and_write

Trap: LDA #$xx : STA $D643 : CLV

Dienst: Wacht totdat de seriële monitor of de Matrix Mode Debugger gereed is om te ontvangen en schrijft vervolgens een teken naar het apparaat.

Invoer: **A** Het ASCII-teken dat geschreven moet worden.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: De service wacht tot de seriële monitor gereed is om te ontvangen. Dit kan uw programma vertragen of vast laten lopen. Indien u dat niet wilt en het geen bezwaar vindt dat het teken verloren gaat wanneer de seriële monitor niet gereed is, gebruik dan hyppo_serial_monitor_write.

hyppo_serial_monitor_write

Trap: LDA #$7C : STA $D640 : CLV

Dienst: Schrijft een teken naar de seriële monitor of de Matrix Mode Debugger.

Voorwaarden: De seriële monitor is gereed om te ontvangen.

Invoer: **Y** Het ASCII-teken dat geschreven moet worden.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Het teken gaat verloren indien de seriële monitor niet gereed is om te ontvangen. Indien u niet wilt dat het teken verloren gaat, gebruik dan hyppo_serial_monitor_wait_and_write.

hyppo_set_mapping

Trap: LDA #$76 : STA $D640 : CLV

Dienst: Kopieert het brongebied naar de huidige 45GS02-geheugenindeling.

Voorwaarden: Het brongebied begint op een paginagrens tussen $0000 en $7E00 en is minimaal 6 bytes.

Invoer: **Y** Het meest significante byte van het brongebied.

Vanaf $YY00 heeft de huidige mappingsinformatie de volgende structuur.

Offset	Type	Omschrijving
0	word	MAPLO (big-endian)
2	word	MAPHI (big-endian)
4	byte	Het megabyte-offset voor MAPLO
5	byte	Het megabyte-offset voor MAPHI

Nawerkingen: De CPU gaat verder met de nieuwe geheugenindeling.

Foutcodes: **$10 invalid address** Het Y-register is > $7E.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: Wees voorzichtig bij het wijzigen van de geheugenindeling. Hyppo zal geen maatregelen nemen om te garanderen dat de instructies na de STA worden uitgevoerd ongeacht de indeling. Indien u de indeling van het blok waar het programmateller naar wijzigt, zal de CPU hervatten met de instructies in het nieuw toegewezen blok.

MAPLO is de mapping voor $0000 - $7FFF.

MAPHI is de mapping voor $8000 - $FFFF.

Zie Hoofdstuk/Bijlage K op bladzijde K-3 voor meer informatie over de geheugenindeling en bankindeling van de MEGA65.

hyppo_switch_to_task

Trap: LDA #$6C : STA $D640 : CLV

Dienst: Schakelt de context over naar een andere Hyppo-taak.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_terminateothertask

Trap: LDA #$60 : STA $D640 : CLV

Dienst: Beëindigt een andere Hyppo-taak.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

hyppo_toggle_force_4502

Trap: LDA #$72 : STA $D640 : CLV

Dienst: Schakelt de CPU-persoonlijkheid om tussen 45GS02 en 6502.

Uitvoer: **A** Indien bit 5 is gezet, bevindt de CPU zich in de 45GS02-persoonlijkheid. Indien bit 5 is gewist, bevindt de CPU zich in de 6502-persoonlijkheid.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: De overige bits van het A-register zijn ongedefinieerd. Verwacht niet dat ze nul zijn.

In de 6502-persoonlijkheid zijn geen van de nieuwe opcodes van de 65C02, 65CE02, 4510 of 45GS02 beschikbaar. Deze worden vervangen door het oorspronkelijke — en vaak vreemde — gedrag van de ongedefinieerde opcodes van de 6502.

Waarschuwing Deze functie is onvolledig en niet getest. De meeste ongedocumenteerde 6502-opcodes werken niet correct wanneer de 6502-persoonlijkheid is ingeschakeld.

Voorbeeld: Schakelt de 45GS02-persoonlijkheid in ongeacht de huidige persoonlijkheid van de CPU.

```
@10
    ; Wissel de CPU-persoonlijkheid
    LDA #$72 : STA $D640 : CLV : BCC error
    ; Wissel opnieuw indien bit 5 van A niet ingesteld is
    AND #%00100000 : BEQ @10
```

hyppo_toggle_rom_writeprotect

Trap: LDA #$70 : STA $D640 : CLV

Dienst: Schakelt de schrijfbescherming voor $20000 - $3FFFF om.

Uitvoer: **A** Indien bit 2 is gezet, kan naar $20000 - $3FFFF niet worden geschreven.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

Opmerkingen: De overige bits van het A-register zijn ongedefinieerd. Verwacht niet dat ze nul zijn.

Indien u de bescherming eenvoudig wilt uitschakelen of inschakelen, kunt u hyppo_rom_writeenable en hyppo_rom_writeprotect gebruiken.

hyppo_writeintotask

Trap: LDA #$56 : STA $D640 : CLV

Dienst: Schrijft in het geheugen van een andere Hyppo-taak.

Opmerkingen: **NIET GEÏMPLEMENTEERD**

SYSTEMPARTITIESERVICES

hyppo_configsector_apply

Trap: LDA #$04 : STA $D642 : CLV

Dienst: Past de systeemconfiguratie-sector toe die momenteel in het geheugen is geladen.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_configsector_read

Trap: LDA #$00 : STA $D642 : CLV

Dienst: Leest de systeemconfiguratie-sector in het geheugen.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_configsector_write

Trap: LDA #$02 : STA $D642 : CLV

Dienst: Schrijft de systeemconfiguratie-sector vanuit het geheugen.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_dmagic_autoset

Trap: LDA #$06 : STA $D642 : CLV

Dienst: Stelt de DMAgic-revisie in op basis van de geladen ROM.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

FREEZERSERVICES

hyppo_freeze_self

Trap: LDA #$xx : STA $D67F : CLV

Dienst: Start de Freezer.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_get_slot_count

Trap: LDA #$16 : STA $D642 : CLV

Dienst: Geeft het aantal freeze-slots op.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_locate_freeze_slot

Trap: LDA #$10 : STA $D642 : CLV

Dienst: Zoekt de eerste sector van een freeze-slot.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_read_freeze_region_list

Trap: LDA #$14 : STA $D642 : CLV

Dienst: Leest de freeze-regiolijst.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2

hyppo_unfreeze_from_slot

Trap: LDA #$12 : STA $D642 : CLV

Dienst: Maakt een freeze-slot ongedaan.

Geschiedenis: Beschikbaar vanaf Hyppo 1.2



BIJLAGE

Machinetaalmonitor

- Inleiding
- De machinetaalmonitor van de MEGA65
- De Matrix Mode/seriële monitor

INLEIDING

Een machinetaalmonitor is een programma waarmee u de interne toestand van de processor en het geheugen van een computer kunt bekijken en ermee kunt experimenteren. Zo'n monitor is handig bij het opsporen van fouten in machinetaalprogramma's en bij het onderzoeken hoe de computer werkt. Een typische monitor kan onder meer de inhoud van de CPU-registers en van het geheugen weergeven en wijzigen, machinetaal disassembleren, instructies rechtstreeks in het geheugen assembleren en het geheugen op andere manieren bewerken.

De MEGA65 heeft twee afzonderlijke machinetaalmonitors ingebouwd: één in de ROM (de MEGA65-monitor) en één die deel uitmaakt van de Matrix Mode-debuginterface (de Matrix Mode/seriële monitor). Elke monitor heeft zijn eigen mogelijkheden en toepassingen.

De monitor in de ROM van de MEGA65 lijkt op de monitor die bij de oorspronkelijke C65-ROM hoorde, maar is volledieger en biedt meer mogelijkheden. Hij is opgenomen in zowel de gesloten MEGA65-ROM als in de uitgebreide MEGA65 OpenROM. Indien u een oorspronkelijke C65-ROM gebruikt, werkt de monitor daarin op vergelijkbare wijze.

DE MACHINETAALMONITOR VAN DE MEGA65

De machinetaalmonitor die bij de ROM van de MEGA65 hoort, kunt u op verschillende manieren openen. Vanuit BASIC roept u de monitor aan met het commando MONITOR. U kunt er ook voor zorgen dat de MEGA65 rechtstreeks in de monitor opstart door tijdens het opstarten de toets STOP ingedrukt te houden.

De MEGA65-monitor wordt automatisch geactiveerd wanneer het programma de instructie **BRK** tegenkomt (bytewaarde 00). U kunt uw programma op een specifiek punt laten pauzeren en de monitor openen, door op die plek in de programmacode een **BRK**-instructie op te nemen.

Wanneer de monitor start, toont hij eerst de inhoud van de registers en wacht daarna op een commando:

```
MONITOR
   PC   SR AC XR YR ZR BP SP
; 000000 00 00 00 00 00 00 F8
```

Getallen

De monitor gebruikt hexadecimale notatie (grondtal 16) om getallen weer te geven, zoals geheugenadressen en waarden. Hexadecimale waarden verschijnen zonder voorafgaand '$'-teken, zoals `5A` of `D020`.

U kunt een waarde in hexadecimale, decimale (grondtal 10), octale (grondtal 8) of binaire (grondtal 2) notatie invoeren. Elk getalformaat gebruikt een teken als prefix, zoals in de onderstaande tabel staat. Indien u de prefix weglaat, gaat de monitor ervan uit dat de waarde hexadecimaal is. (Dat is anders dan bij BASIC, dat zonder prefix uitgaat van decimale notatie.) U kunt een bytewaarde ook als PETSCII-teken invoeren, met de bijbehorende prefix.

Ook voor instructies in assembleertaal is hexadecimaal het standaardgetalformaat. Dat kan verwarrend zijn! Bedenk wat er gebeurt als u de monitor de volgende instructie laat assembleren:

```
LDA #10
```

Een gewone assembler ziet dit als het laden van de decimale waarde 10 (hex `0A`) in de accumulator. De monitor gaat er echter van uit dat dit de hexadecimale waarde `$10` is (decimaal 16). Om verwarring te voorkomen geeft u bij getallen in instructies in assembleertaal altijd expliciet een prefixteken op. De disassembler van de monitor toont alle waarden als hexadecimaal, beginnend met het prefix `$`.

Alle getallen moeten letterlijke waarden zijn. De monitor begrijpt geen rekenkundige uitdrukkingen.

U kunt de monitor ook gebruiken om een gegeven getal of PETSCII-teken in alle grondtallen te tonen. Typ daartoe het getal in alsof het een commando is, met de bijbehorende prefix. Voor hexadecimale getallen is de `$` in dit geval verplicht, om het onderscheid te maken met andere monitorcommando's.

In de volgende tabel staan de prefixes voor elke manier waarop u een waarde kunt invoeren:

grondtal	naam	prefix	toegelaten tekens	voorbeeld
16	hexadecimaal		0123456789ABCDEF	100
16	hexadecimaal	$	0123456789ABCDEF	$100
10	decimaal	+	0123456789	+256
8	octaal	&	01234567	&400
2	binair	%	01	%100000000
	teken	'	alle	'A

De assembler

De monitor heeft een ingebouwde mini-assembler waarmee u machinetaal kunt schrijven met de standaardmnemonieken, zoals **LDA** of **STA**.

Dit is geen volwaardige assembler zoals u die zou gebruiken om grote programma's te schrijven. Met name ondersteunt hij geen symbolen, zoals labels voor subroutines

of KERNAL-adressen. In plaats van **JSR CHROUT** moet u het adres van **CHROUT** opzoeken en de numerieke waarde ervan intypen: **JSR FFD2**

Voor het gemak kunt u bij een vertakkingsinstructie een volledig adres opgeven. De monitor berekent dan de relatieve adresoffset om de instructie te assembleren.

De assembler kent alle instructies en adresseermethoden van de MEGA65-CPU 45GS02. Een instructie zoals **LDA [TXTPTR],Z** wordt geassembleerd als het laden van de accumulator met behulp van een 32-bits pointer op de adressen **TXTPTR, TXTPTR+1, TXTPTR+2, TXTPTR+3**.

Verschillen met de C65-monitor

De MEGA65-monitor lijkt op de monitor van de C65. Hij kent echter tal van uitbreidingen ten opzichte van de C65-monitor, zowel om de monitor in het algemeen nuttiger te maken als om hem uit te breiden met ondersteuning voor mogelijkheden die specifiek zijn voor de architectuur van de MEGA65.

De MEGA65-monitor beschikt over de volgende unieke mogelijkheden:

Adressen:
Alle adressen worden gebruikt als 32-bits adressen (4 bytes). Zo krijgt u toegang tot het volledige adresbereik van de MEGA65, dat 28 bits beslaat. Dit is vooral nuttig voor de toegang tot de RAM-blokken van 8 MB: attic RAM op $8000000 (ingebouwd) en cellar RAM op $8800000 (optioneel). Stelt u bit 31 van een adres in op 1, dan krijgt u toegang tot een speciale (in banken ingedeelde) configuratie. In dat geval bedekken het I/O-gebied op $D000 en de ROM-gebieden $6000 - $7FFF (monitor-ROM) en $E000 - $FFFF (KERNAL-ROM) de huidige bank.

Commando's:
Het extra commando **B** geeft bitmaps van tekens weer.

Schijftoegang:
Het schijfcommandoteken kent twee extra functies: **U1** om een reeks schijfblokken naar het geheugen te lezen en **U2** om een geheugenbereik naar schijfblokken te schrijven. Dit maakt het bewerken van schijven mogelijk, bijvoorbeeld het wijzigen van directory-items, maar u kunt het zelfs gebruiken om een back-up te maken van de volledige inhoud van diskettes of schijfimages. De attic RAM is groot genoeg om de inhoud van 8 volledige 1581-diskettes te bevatten.

Disassembler:
De disassembler kan alle extra adresseermethoden decoderen, zoals de 32-bits indirecte modus $[\$nn],Z$ en de samengestelde instructies die het 32-bits Q-register gebruiken.

Register:
Het registeroverzicht toont de volledige 16-bits stackpointer en het Base-Page-register en aanvaardt nieuwe instellingen voor beide.

Tabel met MEGA65-monitorcommando's

C	mnemoniek	beschrijving
A	ASSEMBLE	Assembleert een regel 45GS02-code
B	BITMAPS	Geeft 8x8-bitmaps (tekens) weer
C	COMPARE	Vergelijkt twee geheugengebieden
D	DISASSEMBLE	Disassembleert een regel 45GS02-code
F	FILL	Vult een geheugengebied met een waarde
G	GO	Start de uitvoering op het opgegeven adres
H	HUNT	Zoekt opgegeven gegevens in een geheugengebied
J	JUMP	Springt naar een subroutine op het opgegeven adres
L	LOAD	Laadt een bestand van schijf
M	MEMORY	Toont een geheugengebied
R	REGISTERS	Toont de inhoud van de 45GS02-registers
S	SAVE	Slaat een geheugengebied op in een schijfbestand
T	TRANSFER	Kopieert geheugen naar een andere locatie
V	VERIFY	Vergelijkt een geheugengebied met een schijfbestand
X	EXIT	Verlaat de monitormodus
.	<punt>	Assembleert een regel 45GS02-code
>	<groter dan>	Wijzigt het geheugen
;	<puntkomma>	Wijzigt de registerinhoud
@	<apestaartje>	Schijfcommando, directory of status
$	<hex>	Geeft de waarde in hexadecimaal, decimaal, octaal en binair weer
+	<decimaal>	Geeft de waarde in hexadecimaal, decimaal, octaal en binair weer
&	<octaal>	Geeft de waarde in hexadecimaal, decimaal, octaal en binair weer
%	<binair>	Geeft de waarde in hexadecimaal, decimaal, octaal en binair weer

A : ASSEMBLE

Formaat: **A adres mnemoniek operand**

Gebruik: Met de mini-assembler voert u machinetaalinstructies in met behulp van gemakkelijk te onthouden mnemonieken in plaats van opcodes. De operand kunt u invoeren als hexadecimaal getal, decimaal getal, binair getal

of teken. Sprongdoelen worden automatisch omgerekend naar relatieve afstanden. Na elke ingevoerde instructie genereert de mini-assembler de 1 tot 5 bytes lange machinetaal, drukt deze code samen met de instructie af en verhoogt de programmateller. Er verschijnt een nieuwe regel met het commando **A**, waarin de nieuwe waarde van de programmateller wordt afgedrukt. Dit vergemakkelijkt het snel invoeren van instructies. De assembleerinvoermodus wordt uitsluitend beëindigd door op RETURN te drukken. U kunt elke regel van de ingevoerde code of een regel in disassembly-indeling wijzigen door de cursor naar die regel te verplaatsen en het gewenste element te veranderen, bijvoorbeeld de mnemoniek of de operand. Hexadecimale waarden die vóór de mnemoniek staan, worden genegeerd.

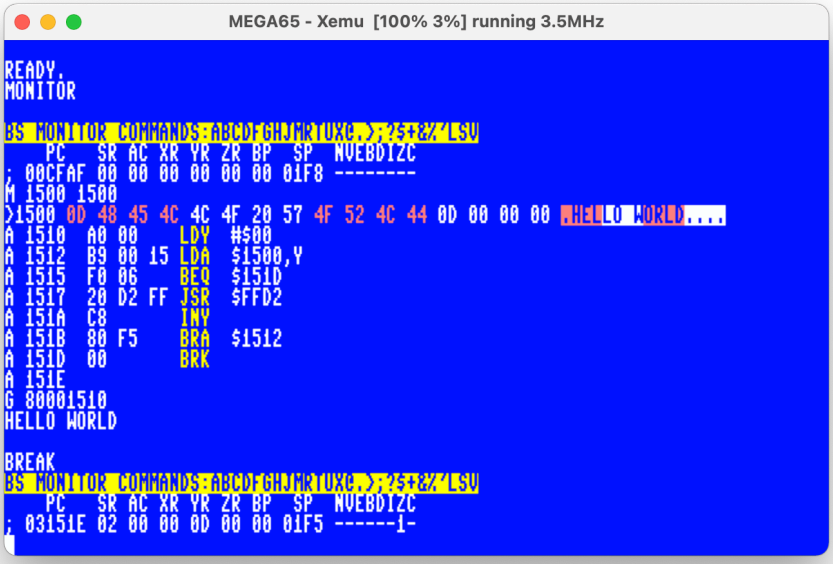
Indien u na het uitvoeren van de code wilt terugkeren naar de monitor, moet de laatste instructie een **BRK**-instructie zijn en moet het programma worden aangeroepen terwijl I/O en de monitor-ROM actief zijn. Dat doet u door bit 31 van het uitvoeradres in te stellen. Indien het programma in bank 0 op adres 1500 is ingevoerd, start u het met: `G 80001500`.

Indien de ingevoerde code een subroutine is, moet deze eindigen met een **RTS**-instructie.

Opmerkingen:

De assembler herkent alle 45GS02-instructies van de MEGA65, met inbegrip van de instructies die het Q-register gebruiken. **NOP** en **AUG** kunnen worden ingevoerd, maar worden geïnterpreteerd als **EOM** respectievelijk **MAP**.

32-bits Base-Page-Indirect-instructies gebruiken vierkante haken (**[** en **]**).



B : BITMAPS

Formaat: **B begin**

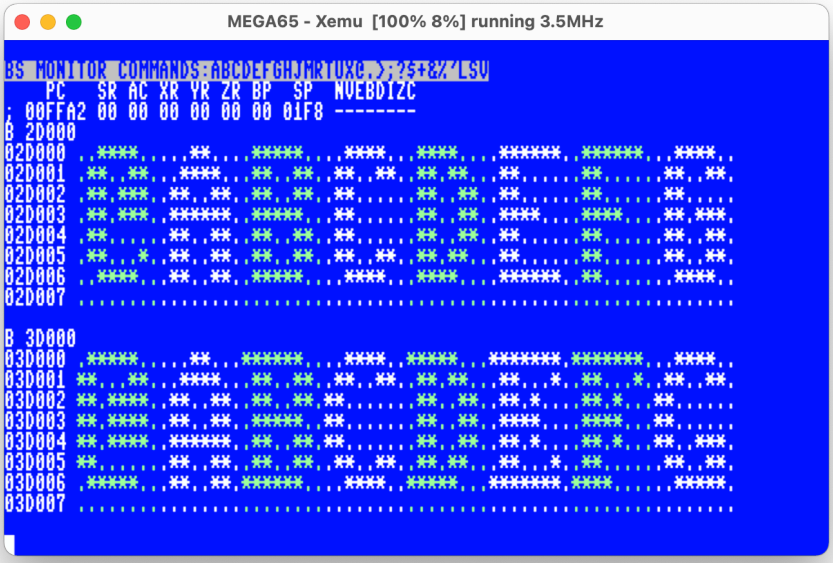
Gebruik: Geeft geheugenwaarden weer, geïnterpreteerd als bitmaps van tekens.

Opmerkingen:

Het commando B geeft de inhoud van geheugencellen bitsgewijs weer, waarbij voor een 1 een sterretje en voor een 0 een punt wordt afgedrukt. Daarbij wordt rekening gehouden met de bijzondere indeling van tekengegevens, waarbij 8 bytes samen één tekencel vormen. Bij elke aanroep worden 8 tekens weergegeven.

In de 92XXXX-ROM's zijn drie ROM-tekensets ingebouwd:

```
FONT A : REM $029000 : ASCII [£]↑← (!)π inbegrepen
FONT B : REM $03D000 : versie van A met schreef
FONT C : REM $02D000 : origineel C64-lettertype
```



C : COMPARE

Formaat: **C begin einde begin-b**

Gebruik: Vergelijkt twee geheugengebieden en geeft vervolgens de adressen weer waarvan de waarden verschillen.

Opmerkingen:

Het eindadres ligt één adres voorbij het laatste te vergelijken adres.

Het tweede geheugengebied dat vergeleken wordt, begint op **begin-b** en heeft dezelfde grootte als het gebied dat door **begin** en **einde** wordt omschreven.

Adressen worden als hexadecimale waarden weergegeven.

D : DISASSEMBLE

Formaat: **D [van [tot]]**

Gebruik: Toont een listing in machinetaal voor het opgegeven adresbereik, in de veronderstelling dat het code bevat. Wanneer slechts één argument is opgegeven, disassembleert de disassembler de volgende 21 bytes. Indien geen argument is opgegeven, gaat de disassembly verder op het laatst gebruikte adres voor disassemblage. De inhoud wordt als hexadecimale waarden weergegeven.

Opmerkingen:

De regels beginnen met het puntteken '.'. Hierdoor kunt u de disassembly direct op het volledige scherm bewerken. Drukt u in een willekeurige regel RETURN in, dan assembleert de monitor het gewijzigde commando van de cursorregel weer naar het geheugen, indien zich daar schrijfbare RAM bevindt. Zie het monitorcommando **.**.

De disassembler kent de instructieset van de MEGA65-CPU, de 45GS02. Heeft een instructie het voorvoegsel **NEG NEG**, dan worden die hexadecimale waarden weggelaten en verschijnt in plaats daarvan een sterretje (*) tussen het adres en de hexadecimale waarden.

Voorbeeld: Gebruik van **D**

```
DL"*

SEARCHING FOR 0:*
LOADING
READY.
MONITOR

MONITOR
    PC   SR AC XR YR ZR SP
; 000000 00 00 00 00 00 F8
D 31F7
. 0031F7  A4 29     LDY  $29
. 0031F9  AB 29 00  LDZ  $0029
. 0031FC  B1 AC     LDA  ($AC),Y
. 0031FE  91 57     STA  ($57),Y
. 003200  EA        NOP
. 003201  B2 AE     LDA  ($AE),Z
. 003203  EA        NOP
. 003204  92 59     STA  ($59),Z
. 003206  3B        DEZ
. 003207  88        DEY
. 003208  10 F2     BPL  $31FC
. 00320A  18        CLC
. 00320B  A5 AC     LDA  $AC
```

F : FILL

Formaat: **F begin einde waarde**

Gebruik: Vult een geheugengebied met een waarde.

Opmerkingen:
Het eindadres ligt één adres voorbij het laatste adres dat gevuld wordt.

G : GO

Formaat: **G [adres]**

Gebruik: Voert de code op een adres uit. Wordt **adres** weggelaten, dan gebruikt de monitor de huidige programmateller als adres.

Opmerkingen:
Dit is gelijkwaardig aan een **JMP**-instructie.

Het commando **G** zonder argument is een handige manier om de uitvoering van een programma te hervatten nadat de monitor met een **BRK**-instructie is gestart.

Omdat dit een **JMP** gebruikt, mag u dit commando niet gebruiken om subroutines aan te roepen. Zie **J**.

H : HUNT

Formaat: **H begin einde byte [byte...]**

Gebruik: Zoekt in een bereik van het geheugen naar elk voorkomen van een byte-reeks en geeft vervolgens het beginadres van elk voorkomen weer.

Opmerkingen:
Het eindadres ligt één adres voorbij het laatste adres dat doorzocht wordt.

Adressen worden als hexadecimale waarden weergegeven.

J : JUMP

Formaat: **J adres**

Gebruik: Springt naar een subroutine op een adres.

Opmerkingen:
Dit is gelijkwaardig aan een **JSR**-instructie. Keert de subroutine terug met RTS, dan neemt de monitor het weer over.

Wilt u de uitvoering gewoon op een adres voortzetten zonder terugkeer naar de monitor te verwachten, gebruik dan **G**.

L : LOAD

Formaat: **L "bestandsnaam",eenheid [begin]**

Gebruik: Laadt gegevens uit een bestand in het geheugen.

Opmerkingen:

De bestandsnaam moet tussen dubbele aanhalingstekens staan.

Het nummer van de **eenheid** (zoals 8 of 9) is verplicht. **Eenheid** wordt opgegeven in het normale talstelsel van de monitor (standaard hexadecimaal). Zie Getallen.

Ontbreekt het **begin**-adres, dan neemt de monitor het beginadres uit de eerste twee bytes van het bestand. Het bestand moet van het type PRG zijn. Indien u **begin** opgeeft, negeert de monitor de PRG-kop van twee bytes.

M : MEMORY

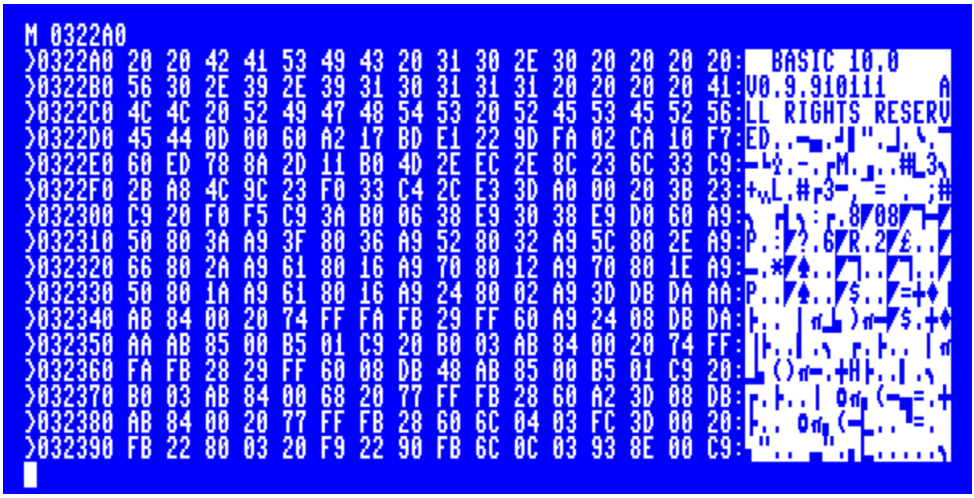
Formaat: **M [van [,tot]]**

Gebruik: Drukt een geheugendump af voor het opgegeven adresbereik. De dump toont de geheugeninhoud, ingedeeld in rijen van 16 opeenvolgende adressen, te beginnen met het adres dat als eerste argument is opgegeven. De dump loopt door totdat een rij is afgedrukt die de waarde bevat van het adres dat als tweede argument is opgegeven. Ontbreekt het tweede argument, dan toont de dump een volledige pagina van 256 bytes in 16 rijen. De inhoud wordt afgedrukt als 16 bytewaarden in hexadecimale notatie, gevolgd door de tekenweergave.

Opmerkingen:

De rijen beginnen met het teken '>'. Hierdoor kunt u de dump direct op het volledige scherm bewerken. Drukt u in een willekeurige rij RETURN in, dan schrijft de monitor de gewijzigde waarden van de cursorrij terug naar het geheugen, indien zich daar schrijfbare RAM bevindt. Zie het monitorcommando **>**.

Voorbeeld: Gebruik van **M**



R : REGISTERS

Formaat: **R**

Gebruik: Toont de inhoud van de registers zoals die waren vlak voordat de monitor werd gestart.

Opmerkingen:

De uitvoer begint met het teken ;, zodat u de registerwaarden kunt wijzigen door de cursor omhoog naar de regel te bewegen, de waarden aan te passen en vervolgens op RETURN te drukken.

S : SAVE

Formaat: **S "bestandsnaam",eenheid begin einde**

Gebruik: Slaat een geheugengebied op in een bestand op schijf, van het type PRG.

Opmerkingen:

De bestandsnaam moet tussen dubbele aanhalingstekens staan.

Het nummer van de **eenheid** (zoals 8 of 9) is verplicht. **Eenheid** wordt opgegeven in het normale talstelsel van de monitor (standaard hexadecimaal). Zie Getallen.

Het eindadres ligt één adres voorbij het laatste adres dat opgeslagen wordt.

Om een bestaand bestand te overschrijven, plaatst u `0:` vóór de bestandsnaam, bijvoorbeeld: `"0:MYFILE"`.

T : TRANSFER

Formaat: **T begin einde bestemmingsbegin**

Gebruik: Kopieert een geheugengebied naar een ander gebied.

Opmerkingen:

Het eindadres ligt één adres voorbij het laatste adres dat gekopieerd wordt.

De monitor veronderstelt dat het doelgebied even lang is als het brongebied. Alle bytes worden gekopieerd.

V : VERIFY

Formaat: **V "bestandsnaam",eenheid begin**

Gebruik: Controleert of een geheugengebied overeenkomt met een bestand op schijf.

Opmerkingen:

De bestandsnaam moet tussen dubbele aanhalingstekens staan.

Het nummer van de **eenheid** (zoals 8 of 9) is verplicht. **Eenheid** wordt opgegeven in het normale talstelsel van de monitor (standaard hexadecimaal). Zie Getallen.

De monitor veronderstelt dat de te controleren gegevens even lang zijn als het bestand.

Komen de gegevens in het geheugen niet overeen, dan toont de monitor een foutmelding. Anders verschijnt er niets meer en kunt u ervan uitgaan dat de controle is geslaagd.

X : EXIT

Formaat: **X**

Gebruik: Verlaat de monitor en keert terug naar BASIC.

Opmerkingen:

Indien de monitor werd gestart door een **BRK**-instructie in een programma, betekent het verlaten van de monitor in feite dat het programma wordt afgebroken. Om een gepauzeerd programma voort te zetten, gebruikt u in plaats daarvan **G**.

. : ASSEMBLE

Formaat: **. adres opcode operand**

Gebruik: Een alias voor **A**.

Opmerkingen:

De uitvoer van de disassembler begint elke regel met het teken .. U kunt de door de disassembler afgedrukte machinetaalinstructies bewerken door de cursor omhoog naar een regel te bewegen, de regel aan te passen en vervolgens op RETURN te drukken. Hierdoor verandert het . in een A, zodat u kunt zien welke regels opnieuw zijn geassembleerd.

> : MODIFY MEMORY

Formaat: **> adres byte [byte...]**

Gebruik: Wijzigt één of meer bytes vanaf een adres.

Opmerkingen:

De uitvoer van het commando **M** begint elke regel met het teken >. U kunt de door het commando **M** afgedrukte geheugenwaarden bewerken door de cursor omhoog naar een regel te bewegen, de regel aan te passen en vervolgens op RETURN te drukken.

; : MODIFY REGISTERS

Formaat: **; pc sr ac xr yr zr bp sp nvebdizc**

Gebruik: Wijzigt de CPU-registers. De argumenten hebben dezelfde indeling als de uitvoer van het commando **R**.

Opmerkingen:

De uitvoer van het commando **R** begint de regel met het teken ;. U kunt de door het commando **R** afgedrukte registerwaarden bewerken door de cursor omhoog naar de regel te bewegen, de regel aan te passen en vervolgens op RETURN te drukken.

@ : DISK COMMAND

Formaat: **@ [eenheid] [,commando]**

Gebruik: Voert een schijfcommando uit.

Opmerkingen:

Zonder **commando** toont dit de status van het schijfstation.

Standaard is de **eenheid** 8. De **eenheid** wordt opgegeven in het normale talstelsel van de monitor (standaard hexadecimaal). Zie Getallen.

Een **commando** van **$[pattern]** vraagt bij de opgegeven **eenheid** een directoryoverzicht op en toont dit. U kunt optioneel een **patroon** opgeven om de getoonde bestanden te beperken. De jokertekens * en **?** kunnen worden gebruikt

Het **commando** kan elk Commodore-DOS-commando zijn. Om bijvoorbeeld een bestand te hernoemen: `@R0:NEWNAME=OLDNAME`

Het **commando** kan U1 zijn om ruwe schijfsectoren naar het geheugen te lezen, of U2 om geheugen naar ruwe schijfsectoren weg te schrijven. Deze **commando's** hebben de volgende vorm:

@8,U1 geheugenbegin track sector [sectoraantal]

De tracks zijn genummerd van 1 tot 80 en de sectoren van 0 tot 39. Wordt **sectoraantal** weggelaten, dan wordt slechts één sector gelezen of geschreven.

DE MATRIX MODE/SERIËLE MONITOR

De Matrix Mode-monitor kan iets wat de MEGA65-monitor niet kan: in plaats van een programma te pauzeren om de monitor te draaien, draait de Matrix Mode-monitor gelijktijdig met het actieve programma mee. Dat houdt in dat u het geheugen van de MEGA65 kunt bekijken en wijzigen, *terwijl een programma wordt uitgevoerd.*

Dit werkt dankzij speciale hardware in het ontwerp van de MEGA65, waarin een kleine hulpprocessor is opgenomen die deze monitorinterface draait. Deze hardware heeft een speciaal toegangsmechanisme voor het geheugen en de CPU van de MEGA65.

In vergelijking met de monitors in de ROM, die op de hoofdprocessor van de MEGA65 draaien, heeft de Matrix Mode-monitor een aantal voor- en nadelen:

- Hij kan worden gebruikt terwijl een programma draait.
- Hij kan worden gebruikt, zelfs wanneer het ROM-gebied voor programmacode of gegevens wordt gebruikt in plaats van een standaard C65- of MEGA65-ROM te bevatten.
- Hij is bereikbaar via de seriële debug-interface op de JB1-connector.
- Hij kan worden opgedragen de processor te stoppen zodra het programmatellerregister (PC) van de hoofdprocessor een door de gebruiker opgegeven adres bereikt. Dat wil zeggen dat hij één hardware-breakpoint ondersteunt.

- Hij kan worden opgedragen de processor te stoppen telkens wanneer naar een opgegeven geheugenadres wordt geschreven. Dat wil zeggen dat hij een "write watch" op één geheugenadres ondersteunt. Het geheugenadres wordt opgegeven als een volledig 28-bits adres, zodat hij geheugenschrijfbewerkingen via elke denkbare weg kan detecteren. Let op: lopende DMA-bewerkingen worden nog voltooid voordat het watchpoint ingaat.
- Hij kan worden opgedragen de processor te stoppen telkens wanneer specifieke CPU-vlaggen worden gezet of gewist, wat ook kan worden gebruikt om het debuggen van programma's te ondersteunen.
- Op sommige modellen van de MEGA65 is de ingebouwde ROM van de monitorprocessor zeer klein, waardoor de functionaliteit beperkt kan zijn. Deze monitor kent bijvoorbeeld geen "assemble"-commando. Dat kan in toekomstige coreupdates worden verholpen voor MEGA65-modellen waarvan de monitorprocessor ruimte biedt voor een grotere ROM.

Tabel met Matrix Mode-monitorcommando's

C	mnemoniek	beschrijving
<Enter>	STEP	Voert de volgende instructie uit
#	HYPERTRAP	Schakelt CPU-hypervisor-traps in of uit
+	UARTDIVISOR	Stelt de UART-baudratedeler in
?	HELP	Hulp
@	CPUMEM	(toont geheugen vanuit de CPU-context)
B	BREAKPOINT	Stelt het breakpoint voor de CPU-uitvoering in of wist het
D	DISASSEMBLE	Disassembleert het geheugen
E	FLAGWATCH	Stelt het watchpoint voor de CPU-vlaggen in of wist het
F	FILL	Vult een geheugengebied met een waarde
G	SETPC	Stelt de programmateller van de CPU in
H	HELP	hulp
I	INTERRUPTS	Schakelt CPU-IRQ's in of uit (wist/zet de I-vlag van de CPU)
J	DEBUGMON	Diverse debugfuncties voor de monitor zelf
L	LOADMEMORY	Laadt gegevens in het geheugen
M	MEMORY	Toont de geheugeninhoud
R	REGISTERS	Toont de registers
S	SETMEMORY	Stelt de geheugeninhoud in
T	TRACE	stelt de traceer- of uitvoeringsmodus van de CPU in
W	MEMORYWATCH	Stelt het watchpoint voor geheugenschrijfacties in of wist het
Z	CPUHISTORY	CPU-geschiedenis

De monitor aanroepen

Om de monitor te openen of te verlaten houdt u  ingedrukt en drukt u op . U ziet een animatie van groene tekens die als een regen vanaf de bovenkant van het scherm naar beneden vallen, waarna u een eenvoudige teksterminal-interface aangeboden krijgt. Die is transparant, zodat u tegelijkertijd de schermuitvoer van uw draaiende programma kunt zien.

<Enter> : Single-step

Formaat: **<Enter-toets> Voert de volgende instructie uit**

Gebruik: <Enter-toets>

Opmerkingen:

Bevindt de CPU zich in de trace-modus **t1**, dan voert het indrukken van de Enter- (of Return-)toets bij de prompt zonder commando de volgende CPU-instructie uit en verhoogt de programmateller. Zie hieronder bij "Trace".

: Hypervisor-trap in-/uitschakelen

Formaat: **# Schakelt hypervisor-traps in of uit**

Gebruik: #[0|1]

Opmerkingen:

Is het argument 1, dan worden de hypervisor-traps ingeschakeld; in het andere geval worden ze uitgeschakeld.

Het is niet bevestigd of dit commando momenteel functioneert.

+ : Stel UART-baudratedeler seriële interface in

Formaat: **+ Stelt de UART-baudratedeler van de seriële interface in**

Gebruik: + deler

Opmerkingen:

Stelt de deler voor de seriële monitorinterface in. Met dit commando kunt u de baudrate van de seriële monitorinterface wijzigen ten opzichte van de standaardwaarde van 2000000 bits per seconde. De baudrate bedraagt $40,500,000 \div (divisor - 1)$. Dit heeft alleen betrekking op de seriële UART-interface en verandert niets aan de toegang tot deze monitor via de samengestelde weergave van Matrix Mode.

Wilt u de seriële monitorinterface bijvoorbeeld vertragen naar 19200 bits per seconde, dan moet de deler $40,500,000 \div (19,200 - 1) = 2108$ zijn. Het commando + vereist vervolgens dat u deze waarde naar hexadecimaal omrekent; het commando luidt dan +83c.

Let op: dit commando controleert de opgegeven waarde *niet* op geldigheid. Geeft u per ongeluk een voor uw doeleinden onjuiste waarde op, dan kunt u dit herstellen door de Matrix Mode-interface te activeren: houd  ingedrukt en tik op , en geef daarna het juiste commando om de deler te corrigeren, bijvoorbeeld +14 om terug te keren naar de standaardwaarde van 2000000 bits per seconde.

Vervolgens moet u Matrix Mode weer verlaten door de toetsencombinatie  +  te herhalen, voordat de seriële UART-interface weer actief wordt. Dat komt doordat Matrix Mode de seriële UART-interface uitschakelt zolang de modus actief is.

@ : CPUMEMORY

Formaat: **@ [adres]**

Gebruik: Drukt een geheugendump af voor het opgegeven 16-bits adres, *zoals geïnterpreteerd door de huidige geheugenmapping van de CPU*. Wilt u de inhoud van het geheugen ergens in de 28-bits adresruimte bekijken, gebruik dan in plaats daarvan het commando M.

De dump toont de geheugeninhoud, ingedeeld in rijen van 16 opeenvolgende adressen, te beginnen met het opgegeven adres. De dump toont een volledige pagina van 256 bytes in 16 rijen. De inhoud wordt afgedrukt als 16 bytewaarden in hexadecimale notatie, gevolgd door de tekenweergave.

Opmerkingen:

Ontbreekt een adres, dan toont de monitor de volgende 256 bytes.

? of H : HELP

Formaat: **?** *of* **h**

Gebruik: Toont een (zeer) korte melding die de monitor identificeert. Op sommige MEGA65-modellen waarvoor meer geheugen beschikbaar is voor de monitorprocessor kan dit commando informatie over elk van de beschikbare commando's weergeven.

B : BREAKPOINT

Formaat: **b [adres]**

Gebruik: Stelt het hardware-breakpoint in of wist het. Indien geen adres is opgegeven, wordt het breakpoint uitgeschakeld. Anders wordt het breakpoint ingesteld op het opgegeven 16-bits adres.

Zodra het programmatellerregister (PC) van de processor van de MEGA65 gelijk is aan de waarde die bij dit commando is opgegeven, stopt de processor en toont de Matrix Mode-monitorinterface de laatst

uitgevoerde instructie en de huidige registerwaarden, om u op deze gebeurtenis te wijzen. Hierbij wordt de Matrix Mode-weergave niet geactiveerd. Er wordt normaliter van uitgegaan dat de Matrix Mode al actief is of dat u via de seriële interface communiceert.

D : DISASSEMBLE

Formaat: **<d|D> [adres]**

Gebruik: Disassembleert de instructie op het opgegeven 28-bits adres en geeft deze weer.

Om instructies te disassembleren vanuit de huidige geheugencontext van de CPU, met inachtneming van de huidige geheugenbanking, plaatst u `777` vóór het adres. Zo disassembleert `d777080D` bijvoorbeeld de instructie op $080D, zoals die momenteel zichtbaar is voor de processor van de MEGA65.

Met `D` in plaats van `d` disassembleert u per keer 16 instructies in plaats van slechts één.

E : FLAGWATCH

Formaat: **e [waarde]**

Gebruik: Stelt het watchpoint voor CPU-vlaggen in of wist dit: Indien geen argument is opgegeven, wordt het watchpoint uitgeschakeld. Indien een waarde is opgegeven, wordt deze als 16-bits waarde opgevat, waarbij de eerste twee hexadecimale cijfers aangeven welke processorvlaggen het watchpoint activeren indien zij gezet zijn. De laatste twee hexadecimale cijfers geven aan welke processorvlaggen het watchpoint activeren indien zij gewist zijn. Op deze manier kunt u elke combinatie van processorvlagwaarden bewaken.

Dit commando functioneert ten tijde van het schrijven van deze handleiding niet correct.

Voorbeeld: Om het watchpoint te laten afgaan wanneer de N-vlag is gezet, gebruikt u het commando `e8000`.

F : FILL

Formaat: **f [begin] [einde+1] [waarde]**

Gebruik: Vult het opgegeven 28-bits adresbereik met de opgegeven waarde.

Opmerkingen:

Het eindadres moet één adres voorbij het laatste adres liggen dat u wilt vullen.

G : SETPC

Formaat: **g adres**

Gebruik: Stelt het programmatellerregister (PC) van de processor van de MEGA65 in op het opgegeven 16-bits adres. Draait de processor op dat moment, dan gaat de uitvoering onmiddellijk vanaf dat adres verder. Is de processor op dat moment gestopt, bijvoorbeeld doordat het commando t1 is gebruikt, dan blijft hij stopgezet, met de programmateller ingesteld op het opgegeven adres, gereed voor het moment waarop de processor weer mag draaien.

I : INTERRUPTS

Formaat: **i[0|1]**

Gebruik: Schakelt interruptverzoeken (IRQ's) van de processor van de MEGA65 in of uit. Het uitschakelen van IRQ's kan nuttig zijn wanneer u een programma instructie voor instructie doorloopt: anders loopt u doorgaans uitsluitend door de code van de interruptafhandelaar, omdat interrupts vaker optreden dan uw stappen door de code.

i1 zet de CPU-vlag "IRQs disabled" (I). De CPU reageert daardoor niet langer op IRQ-bronnen, zoals CIA-timers of rasterstraalbewakingen van de VIC. Dit schakelt non-maskable interrupts (NMI's) niet uit. Er bestaat geen manier om NMI's uit te schakelen vanuit de seriële debugger (of vanuit een programma).

i0 wist de I-vlag van de CPU. Dit schakelt de IRQ's weer in.

Opmerkingen:

Als de CPU door **t1** is stopgezet, toont het registeroverzicht (**r**) de gewijzigde I-vlag pas nadat de volgende instructie is uitgevoerd.

J : DEBUGMON

Formaat: **j [waarde]**

Gebruik: Toont interne signalen van de Matrix Mode-monitorinterface en stelt die desgewenst in.

L : LOADMEMORY

Formaat: **l <beginadres> <eindadres + 1>**

Gebruik: Laadt een geheugenblok snel in via de seriële monitorinterface. Onmiddellijk nadat u dit commando heeft verstuurd, moet u de te laden geheugenbytes naar de seriële monitorinterface sturen. De bytes worden ongewijzigd ingelezen en moeten derhalve als gewone bytes worden aangeboden, niet hexadecimaal gecodeerd. Hiermee laadt u gegevens met een snelheid van circa 200KB per seconde bij de standaard seriële baudrate van 2000000 bits per seconde.

M : MEMORY

Formaat: **<m|M> [adres]**

Gebruik: Drukt een geheugendump af voor het opgegeven 28-bits adres. Wilt u de inhoud van het geheugen bekijken zoals de processor die op dit moment ziet met de huidige instellingen voor geheugenbanking, gebruik dan in plaats daarvan het commando `@`.

De dump toont de geheugeninhoud, ingedeeld in rijen van 16 opeenvolgende adressen, te beginnen met het opgegeven adres. De dump toont een volledige pagina van 256 bytes in 16 rijen. De inhoud wordt afgedrukt als 16 bytewaarden in hexadecimale notatie, gevolgd door de tekenweergave.

Om het geheugen te bekijken vanuit de huidige geheugencontext van de CPU, waarbij rekening wordt gehouden met de huidige geheugenbanking, plaatst u 777 vóór het adres; met `m777080D` bekijkt u bijvoorbeeld het geheugen op $080D, zoals dat op dat moment zichtbaar is voor de processor van de MEGA65.

Opmerkingen:

Is er geen adres opgegeven, dan toont de monitor de volgende 256 bytes.

R : REGISTERS

Formaat: **r**

Gebruik: Toont de huidige waarde van verschillende processorregisters en -vlaggen, alsmede een disassembly van de meest recent uitgevoerde instructie.

S : SETMEMORY

Formaat: **s adres <waarde ...>**

Gebruik: Stelt de inhoud van de opgegeven geheugenlocatie in op de aangeleverde waarde. Geeft u meer dan één door spaties gescheiden waarde op, dan worden meerdere opeenvolgende geheugenlocaties ingesteld.

Dit commando gebruikt 28-bits adressen en negeert derhalve de momenteel geselecteerde instellingen voor geheugenbanking.

T : TRACE

Formaat: **t<0|1|c>**

Gebruik: Kiest de traceer- of uitvoeringsmodus van de processor: `t0` betekent dat de processor vrij draait, `t1` stopt de processor, en `tc` laat de processor in de doorlopende traceermodus draaien, waarbij elke instructie en de registerwaarden onmiddellijk na de uitvoering worden weergegeven, alsof `t1` was gekozen en de gebruiker daarna onmiddellijk op Return of Enter zou drukken om de uitvoering van de volgende instructie te vragen.

Is `t1` gekozen, dan wordt in de Matrix Mode-monitor de volgende instructie uitgevoerd zodra u op Enter of Return drukt.

Het commando `t0` wordt ook gebruikt nadat een breakpoint of watchpoint is afgegaan, zodat de processor de uitvoering kan hervatten.

W : WATCHPOINT

Formaat: **w [adres]**

Gebruik: Stelt het hardware-watchpoint in of wist het. Is er geen adres opgegeven, dan wordt het watchpoint uitgeschakeld. In het andere geval wordt het watchpoint ingesteld op het opgegeven 28-bits adres.

Zodra de processor van de MEGA65 naar het adres schrijft dat aan dit commando is meegegeven, stopt de processor en toont de Matrix Mode-monitorinterface de laatst uitgevoerde instructie en de huidige registerwaarden om u op deze gebeurtenis te wijzen. Hierbij wordt de Matrix Mode-weergave niet geactiveerd. Er wordt normaliter van uitgegaan dat de Matrix Mode al actief is, of dat u via de seriële interface communiceert.

Z : CPUHISTORY

Formaat: **z [adres]**

Gebruik: Toont informatie over de instructies die de processor van de MEGA65 onlangs heeft uitgevoerd.

Opmerkingen:

Er wordt vermoed dat dit commando op het moment van schrijven niet correct functioneert.

BIJLAGE P

F018-compatibele controller voor Direct Memory Access (DMA)

- F018A/B-DMA-taken
- MEGA65 Enhanced DMA Jobs
- Textuurschaling en lijntekening
- Inline-DMA-lijsten
- Audio-DMA
- F018 “DMAgic” DMA-controller
- Uitbreidingen van de MEGA65-DMA-controller
- Niet-geïmplementeerde functionaliteit

De MEGA65 bevat een F018/F018A-compatibele DMA-controller. Anders dan in de C65, waar de DMA-controller als een afzonderlijke chip bestaat, maakt deze in de MEGA65 deel uit van de 45GS02-processor. Omdat het gebruik van de DMA-controller echter een logisch afzonderlijk onderwerp vormt, wordt het apart in deze bijlage besproken.

De DMA-controller van de MEGA65 biedt diverse belangrijke verbeteringen ten opzichte van de F018/F018A-DMAgic-chips van de C65:

- **Snelheid** De MEGA65 voert DMA-bewerkingen uit met 40 MHz, waardoor per seconde 40 MB kan worden gevuld of 20 MB kan worden gekopieerd. Zo is het bijvoorbeeld mogelijk om een volledige bitmapafbeelding van 8 KB in C64-stijl in circa 200 microseconden te kopiëren, wat overeenkomt met minder dan vier rasterlijnen!
- **Toegang tot het volledige geheugen** De DMA-controller van de MEGA65 biedt toegang tot alle 256 MB van de adresruimte.
- **Ondersteuning voor het kopiëren van texturen** De DMA-controller van de MEGA65 kan fractionele adresberekeningen uitvoeren ter ondersteuning van hardwarematige textuurschaling, evenals adresstriding, waardoor het in principe mogelijk is om een textuur gelijktijdig te schalen en vanuit het geheugen naar het scherm te tekenen. Dat zou nuttig zijn als iemand gek genoeg zou zijn om een spel in de stijl van Wolfenstein of Doom op de MEGA65 te implementeren.
- **Ondersteuning voor transparantie- en maskerwaarden** De DMA-controller van de MEGA65 kan de opdracht krijgen om bij het kopiëren van geheugen een speciale waarde te negeren, waardoor de inhoud van het bestemmingsgeheugen ongewijzigd blijft. Dit maakt het maskeren van transparante gebieden bij een DMA-kopie mogelijk, wat het blitten van grafische vormen aanzienlijk vereenvoudigt.
- **Optielijst per taak** Voor elke taak in een geketende lijst van DMA-taken kunnen diverse opties worden geconfigureerd, zoals het selecteren van de F018- of F018B-modus, het wijzigen van de transparantiewaarde, fractionele adresstappen of het bron- of bestemmingsgeheugengebied.
- **Achtergrond-audio-DMA** De MEGA65 beschikt over mogelijkheden voor achtergrond-audio-DMA, vergelijkbaar met de Amiga™-serie computers. Belangrijke verschillen zijn dat de MEGA65 samples van 8 of 16 bits kan gebruiken, zeer hoge samplerates tot ongeveer 1 MHz ondersteunt, 256 volume-instellingen per kanaal heeft en geen modulatie tussen de kanalen kent.

F018A/B-DMA-TAKEN

Om een DMA-taak met de F018-serie DMA-controllers uit te voeren, moet u de lijst met DMA-taken in het geheugen opbouwen en vervolgens het adres van deze lijst

in de DMA-adresregisters schrijven. De DMA-taak wordt uitgevoerd wanneer u naar het register ADDRLSBTRIG ($D700) schrijft. Om die reden moet u eerst het MSB en het banknummer van de DMA-lijst in $D701 en $D702 schrijven en pas daarna het LSB, nadat u deze beide registers hebt ingesteld. Indien u met dezelfde lijststructuur in het geheugen meerdere DMA-taken wilt uitvoeren, kunt u gewoon opnieuw naar ADDRLSBTRIG schrijven nadat u de inhoud van de lijst hebt bijgewerkt, mits geen ander programma de inhoud van $D701 of $D702 heeft gewijzigd. Houd er rekening mee dat BASIC 65 de DMA-controller gebruikt om het scherm te laten scrollen. Het is dan ook doorgaans het veiligst om altijd naar alle drie de registers te schrijven.

Nadat er naar ADDRLSBTRIG is geschreven, wordt de DMA-taak onmiddellijk afgerond. Anders dan in de C65 maakt de DMA-controller in de MEGA65 deel uit van de processor. Dit betekent dat de processor stopt met het uitvoeren van instructies totdat de DMA-taak is voltooid. De enige uitzondering hierop is dat audio-DMA doorgaat en cycli onttrekt aan elke andere DMA-activiteit, zodat de geluidsweergave niet wordt beïnvloed.

Dit gedrag betekent dat DMA-taken, anders dan op de C65, niet kunnen worden onderbroken. Heeft uw programma gevoelige timingvereisten, dan moet u grotere DMA-taken wellicht opsplitsen in verscheidene kleinere taken. Deze zorg wordt enigszins gecompenseerd door de hoge snelheid van de DMA in de MEGA65, die per seconde 40.5 MB geheugen kan vullen en 20.25 MB kan kopiëren, vergeleken met circa 3.5 MB en 1.7 MB per seconde op een C65. Daardoor kunnen grotere DMA-taken worden uitgevoerd zonder dat u zich zorgen hoeft te maken over de gevolgen voor realtime-elementen van een programma. Zo is het bijvoorbeeld mogelijk om met de DMA-controller van de MEGA65 een tekstscherm van 80 kolommen bij 50 rijen te vullen in slechts 200 microseconden.

Indeling van de F018-DMA-taaklijst

De DMA-controller van de MEGA65 ondersteunt de twee formaten voor DMA-taaklijsten die de oorspronkelijke F018 gebruikte, welke in de vroege C65-prototypen werd toegepast (tot revisie 2B), en die de F018B en latere revisies gebruikten, welke in de C65-prototypen van revisie 3 - 5 werden toegepast. Het belangrijkste verschil is de toevoeging van een tweede commandobyte, zoals de volgende tabellen tonen:

Het is belangrijk om te weten welke stijl de DMA-controller verwacht. De Hypervisor van de MEGA65 stelt de modus in op basis van de gedetecteerde versie van de C65-ROM, indien er één wordt uitgevoerd. Indien het een oudere versie betreft, wordt de F018-stijl verwacht; anders wordt de nieuwere F018B-stijl verwacht. U kunt nagaan welke stijl is geselecteerd door bit 0 van $D703 op te vragen: staat deze op 1, dan wordt het nieuwere F018B-lijstformaat van 12 bytes verwacht; staat deze op 0, dan wordt het oudere F018-lijstformaat van 11 bytes verwacht. De verwachte stijl kan worden ingesteld door naar dit register te schrijven.

Tenzij u software schrijft die ook op een C65-prototype moet draaien, kunt u waarschijnlijk het beste gebruikmaken van de Enhanced DMA-taken van de MEGA65, waarvan het lijstformaat expliciet in de lijst zelf wordt opgegeven. Omdat de Enhanced

DMA-taken een uitbreiding zijn van de F018/F018B-DMA-taken, moet u het volgende toch lezen, tenzij u het gedrag van de F018-DMA-controller al kent.

Structuur van de 11-byte F018-DMA-lijst

Offset	Inhoud
$00	Commando LSB
$01	Aantal LSB
$02	Aantal MSB
$03	Bronadres LSB
$04	Bronadres MSB
$05	Bronadres BANK en FLAGS
$06	Bestemmingsadres LSB
$07	Bestemmingsadres MSB
$08	Bestemmingsadres BANK en FLAGS
$09	Modulo LSB
$0a	Modulo MSB

* Het commando-MSB is $00 bij gebruik van dit lijstformaat.

Structuur van de 12-byte F018B-DMA-lijst

Offset	Inhoud
$00	Commando LSB
$01	Aantal LSB
$02	Aantal MSB
$03	Bronadres LSB
$04	Bronadres MSB
$05	Bronadres BANK en FLAGS
$06	Bestemmingsadres LSB
$07	Bestemmingsadres MSB
$08	Bestemmingsadres BANK en FLAGS
$09	Commando MSB
$0a	Modulo LSB / Modus
$0b	Modulo MSB / Modus

De structuur van het commandowoord is als volgt:

Bit(s)	Inhoud
0 - 1	DMA-bewerkingstype
2	Ketting (d.w.z. er volgt nog een DMA-lijst)
3	Toegeven aan interrupts
4	MINTERM -SA,-DA bit
5	MINTERM -SA,DA bit
6	MINTERM SA,-DA bit
7	MINTERM SA,DA bit
8 - 9	Adresseermethode van de bron
10 - 11	Adresseermethode van de bestemming
12 - 15	GERESERVEERD. Altijd op 0 instellen

Het commando-veld kent de volgende vier waarden:

Waarde	Inhoud
%00 (0)	Kopiëren
%01 (1)	Mengen (via MINTERMs)
%10 (2)	Wisselen
%11 (3)	Vullen

* Alleen Kopiëren en Vullen zijn op het moment van schrijven geïmplementeerd.

De velden voor de adresseermethode kennen de volgende vier waarden:

Waarde	Inhoud
%00 (0)	Lineair (normaal) adresseren
%01 (1)	Modulo (rechthoekig) adresseren
%10 (2)	Vasthouden (constant adres)
%11 (3)	XY MOD (rechthoekige bitmap) adresseren

* Alleen Lineair, Modulo en Vasthouden zijn op het moment van schrijven geïmplementeerd.

Het BANK- en FLAGS-veld voor de bron- en bestemmingsadressen maakt het mogelijk adressen binnen een adresruimte van 1 MB te selecteren. Om geheugen buiten de eerste 1 MB te benaderen, is het noodzakelijk een Enhanced DMA-taak te gebruiken met de juiste optiebytes om de MB van het bron- en/of bestemmingsgeheugen te selecteren. Het BANK- en FLAGS-veld heeft de volgende structuur:

Bit(s)	Inhoud
0 - 3	Geheugenbank binnen de geselecteerde MB
4	HOLD, d.w.z. wijzig het adres niet
5	MODULO, d.w.z. pas het MODULO-veld toe voor de wrap-around binnen een beperkte geheugenruimte
6	DIRECTION. Indien gezet, wordt het adres verlaagd in plaats van verhoogd.
7	I/O. Indien gezet, zijn de I/O-registers tijdens de DMA zichtbaar op $D000 - $DFFF.

Eenvoudige DMA-bewerkingen uitvoeren

Voor informatie over het gebruik van de DMA-controller vanuit BASIC 65, zie de BASIC-opdracht **DMA** in Hoofdstuk/Bijlage B op bladzijde B-80.

Om de DMA-controller vanuit assembleertaal te gebruiken, bouwt u een gegevens-structuur met de DMA-lijst en stelt u daarna $D702 - $D700 in op het adres van de lijst. Om bijvoorbeeld het scherm in de C65-modus te wissen door het met spaties te vullen, zou de volgende routine kunnen worden gebruikt:

```
; De DMA-lijst bevindt zich in BANK 0
LDA #$00
STA $D702
; Stel het MSB van het DMA-lijstadres in
```

```
    LDA #>dmalist
    STA $D701
    ; Stel het LSB van het DMA-lijstadres in en voer de DMA uit
    LDA #<dmalist
    STA $D700
    RTS

  dmalist:
    .byte $03   ; Commando-laagbyte: FILL
    .word 2000  ; Aantal: 80x25 = 2000 bytes
    .word $0020 ; Vul met waarde $20
    .byte $00   ; Bronbank (genegeerd bij een FILL-bewerking)
    .word $0800 ; Bestemmingsadres van de schermgegevens
    .byte $00   ; Het scherm bevindt zich in bank 0
    .byte $00   ; Commando-hoogbyte
    .word $0000 ; Modulo (genegeerd vanwege het gekozen commando)
```

Het is ook mogelijk meer dan één DMA-taak tegelijk uit te voeren door de CHAIN-bit in het lage byte van het commando te zetten. Om bijvoorbeeld het scherm te wissen zoals hiervoor en daarnaast ook het kleurengeheugen (colour-RAM) van het scherm te wissen, zou u iets kunnen gebruiken als:

```
    ; De DMA-lijst bevindt zich in BANK 0
    LDA #$00
    STA $D702
    ; Stel het MSB van het DMA-lijstadres in
    LDA #>dmalist
    STA $D701
    ; Stel het LSB van het DMA-lijstadres in en voer de DMA uit
    LDA #<dmalist
    STA $D700
    RTS

  dmalist:
    .byte $07   ; Commando-laagbyte: FILL + CHAIN
    .word 2000  ; Aantal: 80x25 = 2000 bytes
    .word $0020 ; Vul met waarde $20
    .byte $00   ; Bronbank (genegeerd bij een FILL-bewerking)
    .word $0800 ; Bestemmingsadres van de schermgegevens
    .byte $00   ; Het scherm bevindt zich in bank 0
    .byte $00   ; Commando-hoogbyte
    .word $0000 ; Modulo (genegeerd vanwege het gekozen commando)

    ; De tweede DMA-taak volgt direct op de eerste
    .byte $03   ; Commando-laagbyte: FILL
    .word 2000  ; Aantal: 80x25 = 2000 bytes
    .word $0001 ; Vul met waarde $01 = wit
    .byte $00   ; Bronbank (genegeerd bij een FILL-bewerking)
    .word $F800 ; Bestemmingsadres van het kleurengeheugen
    .byte $01   ; Het kleurengeheugen bevindt zich in
                ; bank 1 ($1F800-$1FFFF)
```

```
    .byte $00   ; Commando-hoogbyte
    .word $0000 ; Modulo (genegeerd vanwege het gekozen commando)
```

Geheugen kopiëren lijkt sterk op het vullen van geheugen, met dit verschil dat het lage byte van het commando moet worden aangepast en dat het bronadresveld correct moet worden geïnitialiseerd. Om bijvoorbeeld de tekenset uit de ROM op $2D000 - $2DFFF te kopiëren naar $5000, zou u iets kunnen gebruiken als:

```
    ; De DMA-lijst bevindt zich in BANK 0
    LDA #$00
    STA $D702
    ; Stel het MSB van het DMA-lijstadres in
    LDA #>dmalist
    STA $D701
    ; Stel het LSB van het DMA-lijstadres in en voer de DMA uit
    LDA #<dmalist
    STA $D700
    RTS

dmalist:
    .byte $00   ; Commando-laagbyte: COPY
    .word $1000 ; Aantal: 4KB = 4096
    .word $D000 ; Kopieer vanaf $xD000
    .byte $02   ; Bronbank = $02 voor $2xxxx
    .word $5000 ; Bestemmingsadres van de schermgegevens
    .byte $00   ; Het scherm bevindt zich in bank 0
    .byte $00   ; Commando-hoogbyte
    .word $0000 ; Modulo (genegeerd vanwege het gekozen commando)
```

Het is ook mogelijk vanuit BASIC 2 in de C64-modus een DMA-bewerking uit te voeren door de benodigde waarden met POKE te schrijven, nadat u er eerst voor hebt gezorgd dat de MEGA65- of C65-I/O-modus is geselecteerd door de juiste waarden naar $D02F (53295) te schrijven. Om bijvoorbeeld het scherm in C64 BASIC 2 te wissen met behulp van de DMA-controller, zou u iets kunnen gebruiken als:

```
10 rem schakel de MEGA65-I/O in
20 poke53295,asc("g"):poke53295,asc("s")
30 rem DMA-lijst in data-regels
40 data 3: rem commando-lsb = vullen
50 data 232,3 : rem scherm is 1000 bytes = 3*256+232
60 data 32,0: rem vullen met spatie = 32
70 data 0: rem bronbank (niet gebruikt bij vullen)
80 data 0,4: rem schermadres = 1024 = 4*256
90 data 0: rem scherm bevindt zich in bank 0
100 data 0: rem hoge byte van commando
110 data 0,0: rem modulo (niet gebruikt in deze opgave)
120 rem zet de DMA-lijst op $c000 = 49152
130 fori=0to11:readа:poke49152+i,a:next
140 rem voer de taak uit
150 poke55042,0: rem DMA-lijst staat in bank 0
160 poke55041,192: rem DMA-lijst staat in $c0xx
170 poke55040,0: rem DMA-lijst staat in $xx00, en voer uit
```

Hoewel dit elke keer vrij omslachtig is, hoeft u, indien u het scherm opnieuw wilt wissen, niets anders te doen dan opnieuw **POKE 55040,0** uit te voeren, ervan uitgaande dat de DMA-lijst en de registers van de DMA-controller niet zijn gewijzigd sinds de vorige keer dat de DMA-taak is uitgevoerd.

De HOLD-, I/O- en andere opties kunnen ook worden gebruikt om interessante effecten te bereiken. Om bijvoorbeeld zeer snel een nieuwe waarde naar de achtergrondkleur van het scherm te schrijven, kunt u een geheugengebied kopiëren naar $D021, met de I/O-vlag gezet zodat het I/O-register in de DMA-taak zichtbaar wordt voor schrijven, en de HOLD-vlag gezet zodat herhaaldelijk naar hetzelfde adres wordt geschreven. Hiermee wordt met een snelheid van 20.5 MHz naar de achtergrondkleur geschreven, bijna zo snel als de videopixelklok (27 MHz). Zo kunnen wij de kleur vrijwel per pixel wijzigen.

Met een beetje zorg kunnen wij deze routine zo maken dat de uitvoering ervan precies één rasterlijn duurt en zo verticale rasterbalken trekken, of een soort frankenstein-videomodus creëren die een lineaire geheugenindeling gebruikt – ten koste van alle verwerkingstijd gedurende het actieve deel van het beeld.

Het volgende voorbeeld doet dit om verticale rasterbalken op het scherm te trekken. Het programma gaat ervan uit dat de MEGA65 op PAL is ingesteld. Voor NTSC moet de omvang van de DMA-overdracht enigszins worden verkleind. Verder valt bij dit programma op dat met MEGA65 Enhanced-DMA-optie $81 de bestemmingsmegabyte van het geheugen op $FFxxxxx wordt ingesteld, de bank op $D en het bestemmingsadres op $0021, om zo het volledige adres $FFD0021 te vormen. Dit is de werkelijke locatie van het randkleurregister van de VIC-IV. Het programma is geschreven met ACME-compatibele syntaxis.

```
basicheader:
  ;; 2020 SYS 2061
```

```
  !word $80A,2020
  !byte $9E,$32,$30,$36,$31,0,0,0

  ;; Werkelijke code begint op $080d = 2061
main:
  SEI
  LDA #$47      ; Schakel de MEGA65-I/O in
  STA $D02F
  LDA #$53
  STA $D02F
  LDA #65       ; Stel de CPU-snelheid in op snel
  STA 0
  LDA #0        ; Schakel het scherm uit, alles is schermrandkleur
  STA $D011

  LDA $D012     ; Wacht tot het begin van de volgende rasterlijn
raster_sync:    ; vóór het begin van de lus voor horizontale
  CMP $D012     ; uitlijning
  BEQ raster_sync

  ;; De volgende lus duurt precies één rasterlijn
  ;; bij 40,5 MHz in PAL
loop:
  JSR triggerdma
  JMP loop

triggerdma:
  LDA #0      ; Zorg dat het F018-lijstformaat actief is
  STA $D703

  LDA #0      ; DMA-lijstbank
  STA $D702
  LDA #>rasterdmalist
  STA $D701
  LDA #<rasterdmalist
  STA $D705
  RTS

rasterdmalist:
  !byte $81,$FF,$00
  !byte $00     ; COPY
  !word 619     ; De DMA-overdracht is 619 bytes lang
  !word rastercolours  ; bronadres
  !byte $00     ; bronbank
  !word $0020   ; bestemmingsadres
  !byte $1D     ; bestemmingsbank + HOLD
  ;; ongebruikt moduloveld
  !word $0000

rastercolours:
  !byte 0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
!byte 0,0,0,11,11,11,12,12,12,15,15,15
!byte 1,1,1,15,15,15,12,12,12,11,11,11,0,0,0
!byte 0,0,0,6,6,6,4,4,4,14,14,14,3,3,3
!byte 1,1,1,3,3,3,14,14,14,4,4,4,6,6,6,0,0,0
!byte 0,0,0,11,11,11,12,12,12,15,15,15
!byte 1,1,1,15,15,15,12,12,12,11,11,11,0,0,0
!byte 0,0,0,6,6,6,4,4,4,14,14,14,3,3,3
!byte 1,1,1,3,3,3,14,14,14,4,4,4,6,6,6,0,0,0
!byte 0,0,0,11,11,11,12,12,12,15,15,15
!byte 1,1,1,15,15,15,12,12,12,11,11,11,0,0,0
!byte 0,0,0,6,6,6,4,4,4,14,14,14,3,3,3
!byte 1,1,1,3,3,3,14,14,14,4,4,4,6,6,6,0,0,0
!byte 0,0,0,11,11,11,12,12,12,15,15,15
!byte 1,1,1,15,15,15,12,12,12,11,11,11,0,0,0
!byte 0,0,0,6,6,6,4,4,4,14,14,14,3,3,3
!byte 1,1,1,3,3,3,14,14,14,4,4,4,6,6,6,0,0,0
!byte 0,0,0,11,11,11,12,12,12,15,15,15
!byte 1,1,1,15,15,15,12,12,12,11,11,11,0,0,0
!byte 0,0,0,6,6,6,4,4,4,14,14,14,3,3,3
!byte 1,1,1,3,3,3,14,14,14,4,4,4,6,6,6,0,0,0
!byte 0,0,0,11,11,11,12,12,12,15,15,15
!byte 1,1,1,15,15,15,12,12,12,11,11,11,0,0,0
!byte 0,0,0,6,6,6,4,4,4,14,14,14,3,3,3
!byte 1,1,1,3,3,3,14,14,14,4,4,4,6,6,6,0,0,0
!byte 0,0,0,11,11,11,12,12,12,15,15,15
!byte 1,1,1,15,15,15,12,12,12,11,11,11,0,0,0
!byte 0,0,0,6,6,6,4,4,4,14,14,14,3,3,3
!byte 1,1,1,3,3,3,14,14,14,4,4,4,6,6,6,0,0,0
!byte 0,0,0,11,11,11,12,12,12,15,15,15
!byte 1,1,1,15,15,15,12,12,12,11,11,11,0,0,0
!byte 0,0,0,6,6,6,4,4,4,14,14,14,3,3,3
!byte 1,1,1,3,3,3,14,14,14,4,4,4,6,6,6,0,0,0
```

MEGA65 ENHANCED DMA JOBS

De implementatie van de DMAgic in de MEGA65 ondersteunt aanzienlijk uitgebreidere DMA-taken. Een uitgebreide DMA-taak wordt aangegeven door het lage byte van het adres van de DMA-lijst naar $D705 te schrijven in plaats van naar $D700. De MEGA65 zoekt vervolgens aan het begin van de DMA-lijst naar één of meer *taakoptietokens*. Die tokens worden geïnterpreteerd voordat de DMA-taak wordt uitgevoerd die onmiddellijk volgt op het *einde van de taakopties*-token ($00).

Taakoptietokens die een argument nodig hebben, hebben de meest significante bit gezet en worden altijd gevolgd door een optie van 1 byte. Taakoptietokens die

geen argument nodig hebben, hebben de meest significante bit niet gezet. Niet-ondersteunde taakoptietokens worden eenvoudigweg genegeerd. Zo kunnen toekomstige revisies van de DMAgic ondersteuning voor aanvullende opties toevoegen, zonder dat de achterwaartse compatibiliteit wordt verbroken.

Opties die door een taak zijn ingesteld blijven gelden voor de volgende taken in een geketende lijst. Ze worden alleen teruggezet naar hun standaardwaarden wanneer de laatste taak in de keten klaar is. Een taak in een keten die een optie weer op zijn standaardwaarde wil hebben, moet die optie dus opnieuw in zijn eigen optielijst instellen; het weglaten van de optie zet hem niet terug.

Deze opties worden ook gebruikt om geavanceerde functies te bereiken, zoals hardwarematige textuurschaling met snelheden tot 20 megapixels per seconde en hardwarematig lijntekenen met snelheden tot 40 megapixels per seconde. Deze geavanceerde functies zijn geïmplementeerd doordat tijdens de uitvoering complexe berekeningen kunnen worden toegepast op het bron- en/of bestemmingsadres van DMA-jobs.

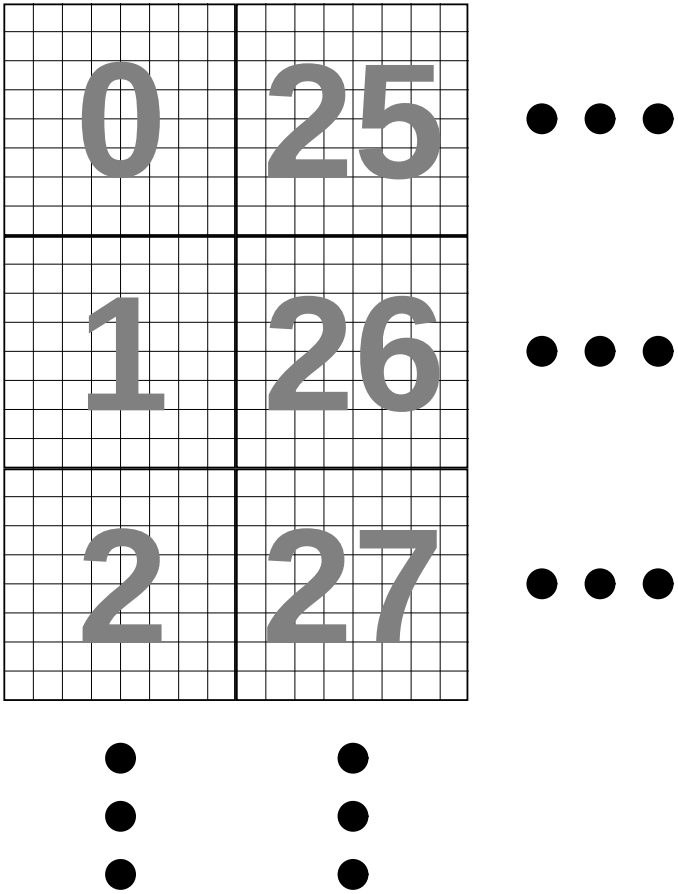
De lijst met geldige taakoptietokens is:

$00	Einde van de taakoptielijst
$06	Gebruik van de transparante waarde uitschakelen
$07	Gebruik van de transparante waarde inschakelen
$0A	Gebruik het 11-byte-F011A-DMA-lijstformaat
$0B	Gebruik het 12-byte-F011B-DMA-lijstformaat
$0D	Ruwe flux naar het diskettestation schrijven
$0E	Ruwe flux van het diskettestation lezen
$0F	Ruwe flux van het diskettestation lezen
$53	De modus 'Shallan Spiral' inschakelen
$80	Bronadres bits 20 - 27
$81	Bestemmingsadres bits 20 - 27
$82	Bron-skipwaarde (256ste van een byte)
$83	Bron-skipwaarde (gehele bytes)
$84	Bestemmings-skipwaarde (256ste van een byte)
$85	Bestemmings-skipwaarde (gehele bytes)
$86	Transparante waarde (bytes met een overeenkomende waarde worden niet geschreven)
$87	X-kolombytes instellen (LSB) voor het bestemmingsadres bij lijntekening
$88	X-kolombytes instellen (MSB) voor het bestemmingsadres bij lijntekening
$89	Y-rijbytes instellen (LSB) voor het bestemmingsadres bij lijntekening
$8A	Y-rijbytes instellen (MSB) voor het bestemmingsadres bij lijntekening
$8B	Helling (LSB) voor het bestemmingsadres bij lijntekening
$8C	Helling (MSB) voor het bestemmingsadres bij lijntekening
$8D	Beginfractie van de hellingaccumulator (LSB) voor het bestemmingsadres bij lijntekening
$8E	Beginfractie van de hellingaccumulator (MSB) voor het bestemmingsadres bij lijntekening
$8F	Lijntekenmodus inschakelen en opties voor het bestemmingsadres (in te stellen in het argumentbyte): bit 7 = lijnmodus inschakelen, bit 6 = richting X of Y kiezen, bit 5 = de helling is negatief.
$97	X-kolombytes instellen (LSB) voor het bronadres bij lijntekening
$98	X-kolombytes instellen (MSB) voor het bronadres bij lijntekening
$99	Y-rijbytes instellen (LSB) voor het bronadres bij lijntekening
$9A	Y-rijbytes instellen (MSB) voor het bronadres bij lijntekening
$9B	Helling (LSB) voor het bronadres bij lijntekening
$9C	Helling (MSB) voor het bronadres bij lijntekening
$9D	Beginfractie van de hellingaccumulator (LSB) voor het bronadres bij lijntekening
$9E	Beginfractie van de hellingaccumulator (MSB) voor het bronadres bij lijntekening
$9F	Lijntekenmodus inschakelen en opties voor het bronadres (in te stellen in het argumentbyte): bit 7 = lijnmodus inschakelen, bit 6 = richting X of Y kiezen, bit 5 = de helling is negatief.

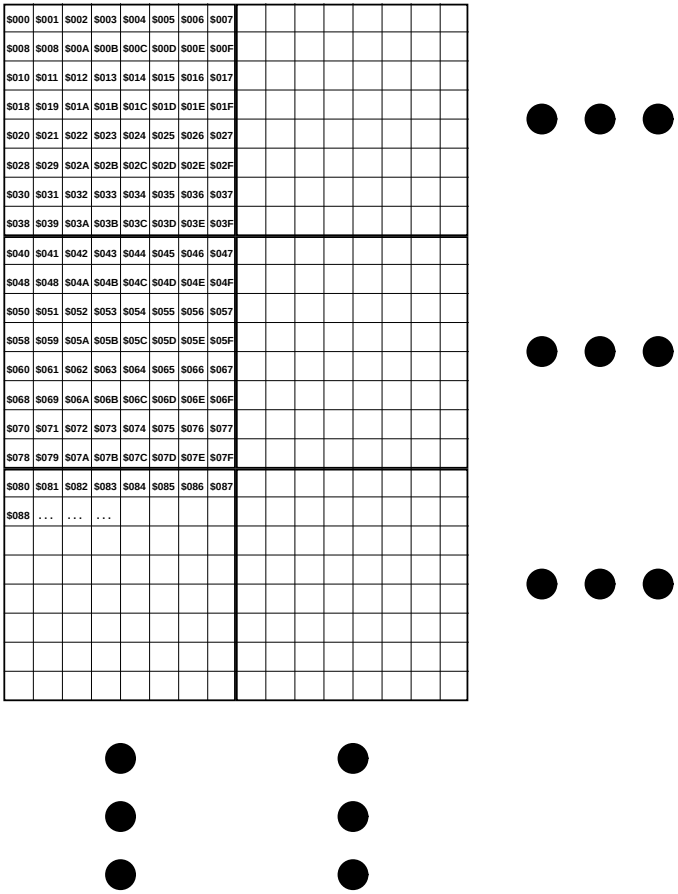
TEXTUURSCHALING EN LIJNTEKENING

De DMAgic beschikt over een geavanceerde interne adresrekenmachine waarmee hij geschaalde texturen en lijnen met willekeurige hellingen kan tekenen op VIC-IV-FCM-beeldschermen.

Voor het schalen van texturen moet het FCM-scherm verticaal worden gerangschikt, zoals hieronder is weergegeven:



Door de tekens zo in verticale kolommen te rangschikken, levert een verticale verplaatsing van één pixel telkens een constante toename van 8 bytes op, zoals hieronder wordt weergegeven:



De skipwaarden voor bron en bestemming maken het ook mogelijk de schaalfactoren in te stellen. Een skipwaarde van $0100 komt overeen met een verplaatsing van $01.00 pixel. Wilt u de verticaal gestapelde FCM-indeling gebruiken als doel voor het kopiëren van verticale textuurlijnen, dan moet de bestemmingsskipwaarde $0800 zijn, d.w.z. 8.0 bytes per pixel. Daarmee wordt een verticale lijn textuurgegevens zonder schaling gekopieerd. Door de bronverplaatsing lager dan $0100 in te stellen, worden sommige pixels herhaald, waardoor de textuur effectief wordt ingezoomd; stelt u de bronverplaatsing boven $0100 in, dan worden sommige pixels overgeslagen, waardoor de textuur effectief wordt uitgezoomd. De bestemmingsverplaatsing hoeft gewoonlijk niet te worden aangepast. Let op: de textuurgegevens moeten zo worden opgeslagen dat elke verticale strook aaneengesloten staat, opdat deze modus kan worden gebruikt.

Voor het tekenen van lijnen moet de DMA-controller de schermindeling kennen. Hij moet in het bijzonder weten welk getal moet worden opgeteld bij het adres van de meest rechtse pixel in een kolom FCM-tekens om het adres te berekenen van de pixel

die onmiddellijk rechts daarvan verschijnt. Evenzo moet hij weten hoeveel er moet worden opgeteld bij het adres van de onderste pixel in een rij FCM-tekens om het adres te berekenen van de pixel die onmiddellijk onder die pixel verschijnt. Dit biedt flexibele mogelijkheden voor de schermindeling en willekeurige schermafmetingen. U moet daarna ook de helling van de lijn opgeven, of de lijn de X-as of de Y-as als hoofdas heeft en of de helling positief of negatief is.

Het bestand `test_290.c` in de repository https://github.com/mega65/mega65-tools bevat een voorbeeld van het gebruik van deze voorzieningen om hardware-versnelde lijntekening te implementeren. Deze aanpak is *zeer* snel, want de lijnen worden getekend met de volledige DMA-vulsnelheid, d.w.z. ongeveer 40.5 miljoen pixels per seconde.

INLINE-DMA-LIJSTEN

Gewoonlijk moet u een apart geheugengebied inrichten dat de DMA-lijst bevat en vervolgens het adres van dat gebied in de DMA-adresregisters op $D70x laden. Omdat de DMA-controller van de MEGA65 deel uitmaakt van de CPU, ondersteunt deze bovendien een uiterst handige extra modus: de inline-DMA-lijstmodus.

Deze modus werkt zoals de Enhanced-DMA-modus, met dit verschil dat de DMA-lijst wordt gelezen vanaf de huidige waarde van het Program Counter (PC)-register. Om deze modus te gebruiken, schrijft u een willekeurige waarde naar $D707. Hiermee start u onmiddellijk een DMA-taak, waarvan de lijst staat in de bytes die direct volgen op de instructie die naar $D707 schrijft.

De DMA-lijst kan uit één enkele taak bestaan of, zoals bij elke andere DMA-taak, uit meerdere gekoppelde taken. Het echte kunststuk is dat de programmateller wordt gezet op het eerstvolgende adres na het einde van de DMA-lijst en dat de DMA-lijst wordt gelezen uit de huidige geheugenindeling van de CPU. Dit betekent dat u code met DMA-lijsten vanuit elke geheugenbank kunt uitvoeren, zonder dat u hoeft na te gaan in welke bank die code staat.

Het volgende codevoorbeeld wist bijvoorbeeld het scherm in de C65-modus en laat de rand daarna eindeloos flikkeren:

```
    STA $D707
    .byte $00   ; end of job options
    .byte $03   ; fill
    .word 2000  ; count
    .word $0020 ; value
    .byte $00   ; src bank
    .word $0800 ; dst
    .byte $00   ; dst bank
    .byte $00   ; cmd hi
    .word $0000 ; modulo / ignored
foo:
    INC $D020
    JMP foo
```

AUDIO-DMA

De MEGA65 beschikt over vier kanalen voor DMA-aangestuurde geluidsweergave, die in plaats van de directe digitale audioregisters op $D6F8-$D6FB gebruikt kunnen worden. U moet dus kiezen welke van deze twee bronnen u aan de audio-crossbar-mixer wilt toevoeren. Die keuze wordt gemaakt via het AUDEN-signaal (bit 7 van $D711), dat tegelijkertijd de audio-DMA-functie in de processor inschakelt en de audio-crossbar-mixer opdraagt de audio van deze bron te gebruiken in plaats van de digitale audioregisters op $D6F8-$D6FB. Indien u naast de audio-DMA-kanalen geen ander geluid wenst, kan de audio-crossbar-mixer worden omzeild en kan de DMA-audio op vol volume worden afgespeeld door het NOMIX-signaal te zetten (bit 4 van $D711). De omzeiling geldt alleen voor de PWM-audio-uitgangen (de 3,5mm-audioaansluiting op R3 en oudere borden): op die uitgangen is er geen geluid van de SID's, FM, microfoons of andere bronnen beschikbaar. HDMI-audio, en de audio-DAC op R4 en nieuwere borden, dragen altijd de uitvoer van de audio-crossbar-mixer. Alle overige bits van $D711 moeten gewoonlijk nul blijven; schrijf dus $80 naar $D711 om audio-DMA in te schakelen.

Twee kanalen vormen het linker digitale audiokanaal en de twee andere het rechter digitale audiokanaal. Deze linker- en rechterkanalen worden vervolgens aan de audio-crossbar-mixer van de MEGA65 toegevoerd.

Omdat de DMA-controller deel uitmaakt van de processor van de MEGA65 en de MEGA65 geen gereserveerde busesleuven heeft voor multimedia-bewerkingen, gebruikt de MEGA65 ongebruikte CPU-cycli om DMA op de achtergrond uit te voeren. Hiervoor moet de CPU van de MEGA65 in de modus "full speed" staan, d.w.z. op ongeveer 40 MHz. In deze modus treedt bij het lezen van een operand uit het geheugen telkens een wachtcyclus op. Elke instructie die een byte uit het geheugen laadt, creëert daardoor één impliciete audio-DMA-sleuf. In de praktijk is dit zelden een probleem, behalve wanneer de processor in een zeer strakke lus wacht. Om ervoor te zorgen dat de audio op de achtergrond blijft spelen, moeten dergelijke lussen een leesinstructie bevatten, zoals:

```
    ; Ensure loop has at least one idle cycle for
    ; audio DMA
loop:
    LDA $1234
    JMP loop
```

Elk van de vier DMA-kanalen wordt geconfigureerd met een blok van 16 registers op respectievelijk $D720, $D730, $D740 en $D750. Wij lichten de registers van het eerste kanaal, kanaal 0, toe op $D720 - $D72F.

Adresbeheer voor samples

Om een audiosample af te spelen, moet u eerst het startadres van het sample opgeven. Dit is een 24-bits adres dat in het hoofd-chipgeheugen van de MEGA65 moet liggen. U schrijft het adres daartoe in $D72A - $D72C. Dat is het adres van de eerste

samplewaarde die wordt afgespeeld. Vervolgens moet u het eindadres van het sample opgeven in $D727 - $D728. Let echter op: dit is slechts 16 bits. Dat komt doordat de MEGA65 bij de controle of het einde van een sample is bereikt, alleen de onderste 16 bits van het adres vergelijkt. In de praktijk betekent dit dat samples niet groter dan 64 KB kunnen zijn. Bevat het sample een gedeelte dat herhaald moet worden, dan moet het startadres van dat herhalende gedeelte in $D721 - $D723 worden geladen en moet de CH0LOOP-bit worden gezet (bit 6 van $D720).

U kunt op elk moment het huidige sampleadres bepalen door de registers op $D72A - $D72C te lezen. Let echter op: deze registers worden niet vergrendeld, zodat de waarden tijdens het lezen kunnen worden bijgewerkt, tenzij u het kanaal eerst stopt door het CH0EN-signaal te wissen.

Afspeelfrequentie en volume van samples

De MEGA65 regelt de afspeelsnelheid van audio-DMA-samples met een 24-bits teller. Telkens wanneer de 24-bits teller overloopt, wordt de volgende samplewaarde opgevraagd. De samplesnelheid stelt u in door de waarde te kiezen die elke CPU-cyclus aan deze teller wordt toegevoegd. Een waarde van $FFFFFF levert derhalve een samplerate van bijna 40.5 MHz op. In de praktijk zijn samplerates boven enkele megahertz echter niet mogelijk, omdat daarvoor onvoldoende ongebruikte CPU-cycli beschikbaar zijn en er anders vervormd geluid ontstaat. Ook onder die grens moet u ervoor zorgen dat de ongebruikte cycli voldoende vaak en voldoende verspreid door de instructiestroom van de processor voorkomen, om vervorming te vermijden. Bij typische samplerates onder 16 kHz en het gebruik van 8-bits samples zijn deze effecten voor normale instructiestromen doorgaans verwaarloosbaar, zodat voor normaal audio-afspelen gewoonlijk geen speciale maatregelen nodig zijn.

Aan de andere kant van de schaal zijn samplefrequenties van slechts 40.5 MHz/2^{24}= 2.4 samples per seconde mogelijk. Dat is laag genoeg voor zelfs de meest veeleisende infrageluidstoepassingen.

Het volume wordt geregeld door $D729 in te stellen. Het maximale volume wordt bereikt met de waarde $FF, terwijl een waarde van $00 het kanaal volledig dempt. De eerste twee audiokanalen worden normaal gesproken aan de linkerzijde toegewezen en de tweede twee aan de rechterzijde. De MEGA65 beschikt echter over afzonderlijke volumeregelaars voor de tegenoverliggende kanalen. Om audio-DMA-kanaal 0 bijvoorbeeld op vol volume aan zowel de linker- als de rechterkant van de audio-uitgang af te spelen, stelt u zowel $D729 als $D71C in op $FF. Dit maakt het pannen van de vier audio-DMA-kanalen mogelijk.

Zowel de frequentie als het volume kunnen tijdens het afspelen van een sample vrij worden aangepast om verschillende effecten te verkrijgen.

Zuivere sinusgolf

Wanneer het nodig is een stabiele sinusgolf te produceren, met name bij hogere frequenties, voorziet een speciale modus hierin. Door het CH0SINE-signaal te zetten,

speelt het audiokanaal een sinusgolfpatroon van 32 bytes met 16-bits samples af. De sample-adressen moeten nog steeds worden ingesteld, alsof de sinusgolftabel zich in de onderste 64 bytes van het geheugen bevindt; de normale adresgeneratielogica wordt namelijk ook in deze modus gebruikt. In deze modus worden geen audio-DMA-fetches uitgevoerd, zodat alle vervormingsbronnen worden vermeden die het gevolg zijn van de onregelmatige spreiding van inactieve cycli in de instructiestroom van de processor.

Dit kan worden gebruikt om sinusgolven te produceren in zowel het hoorbare bereik als ver daarboven in het ultrasone bereik, bij frequenties die 60000 Hz overschrijden, mits de MEGA65 is aangesloten op een geschikte luidsprekeropstelling.

Regeling van het afspelen van samples

Om een kanaal een sample te laten afspelen, zet u het CH0EN-signaal ($D720 bit 7). Het sample wordt afgespeeld tot het zijn einde bereikt, tenzij het CH0LOOP-signaal eveneens is gezet. Wanneer een sample klaar is met afspelen, wordt de CH0STP-vlag gezet. Het audio-DMA-subsysteem kan op dit moment geen interrupts genereren.

Anders dan de Amiga™ biedt het audio-DMA-systeem van de MEGA65 ondersteuning voor zowel 8- als 16-bits samples. Het ondersteunt ook samengepakte 4-bits samples, waarbij van elk sample-byte de lagere of de hogere nibble wordt afgespeeld. Hierdoor kunnen twee afzonderlijke samples hetzelfde byte bezetten, waarmee de benodigde opslagruimte voor twee samples van gelijke lengte effectief wordt gehalveerd.

F018 "DMAGIC" DMA-CONTROLLER

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D700	55040	ADDRLSBTRIG							
D701	55041	ADDRMSB							
D702	55042	WITHIO	ADDRBANK						

- **ADDRBANK** Adresbank van de DMA-lijst (bits 16 - 22). Schrijven wist $D704.
- **ADDRLSBTRIG** DMAgic-LSB van het adres van de DMA-lijst (bits 0 - 7); start DMA wanneer erin wordt geschreven
- **ADDRMSB** Hoge byte van het adres van de DMA-lijst (bits 8 - 15)
- **WITHIO** DMA-withio-vlag (onbenut)

UITBREIDINGEN VAN DE MEGA65-DMA-CONTROLLER

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D703	55043	-						NOMB-WRAP	EN018B
D704	55044	ADDRMB							
D705	55045	ETRIG							
D706	55046	ETRIGMAPD							
D707	55047	ETRIGINLINE							
D70E	55054	ADDRLSB							
D711	55057	AUDEN	BLKD	AUD-WRBLK	NOMIX	-	AUDBLKTO		
D712	55058	DBG-LFTCLP	DBG-RGTCLP	-				CHANS-WAP	-
D713	55059	IRQ-CH0FLG	IRQ-CH1FLG	IRQ-CH2FLG	IRQ-CH3FLG	IRQ-CH0ENA	IRQ-CH1ENA	IRQ-CH2ENA	IRQ-CH3ENA
D714	55060	IRQCH0OFFS							
D715	55061	IRQCH1OFFS							
D716	55062	IRQCH2OFFS							
D717	55063	IRQCH3OFFS							
D71C	55068	CH0RVOL							
D71D	55069	CH1RVOL							
D71E	55070	CH2LVOL							
D71F	55071	CH3LVOL							
D720	55072	CH0EN	CH0LOOP	CH0SGN	CH0SINE	CH0STP	-	CH0SBITS	
D721	55073	CH0BADDRLSB							
D722	55074	CH0BADDRMSB							
D723	55075	CH0BADDRBNK							
D724	55076	CH0FREQ0							
D725	55077	CH0FREQ1							
D726	55078	CH0FREQ2							
D727	55079	CH0TADDRLSB							
D728	55080	CH0TADDRMSB							
D729	55081	CH0VOLUME							
D72A	55082	CH0CURADDRLSB							
D72B	55083	CH0CURADDRMSB							
D72C	55084	CH0CURADDRBNK							
D72D	55085	CH0TMGCNT0							
D72E	55086	CH0TMGCNT1							
D72F	55087	CH0TMGCNT2							
D730	55088	CH1EN	CH1LOOP	CH1SGN	CH1SINE	CH1STP	-	CH1SBITS	

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D731	55089	CH1BADDRLSB							
D732	55090	CH1BADDRMSB							
D733	55091	CH1BADDRBNK							
D734	55092	CH1FREQ0							
D735	55093	CH1FREQ1							
D736	55094	CH1FREQ2							
D737	55095	CH1TADDRLSB							
D738	55096	CH1TADDRMSB							
D739	55097	CH1VOLUME							
D73A	55098	CH1CURADDRLSB							
D73B	55099	CH1CURADDRMSB							
D73C	55100	CH1CURADDRBNK							
D73D	55101	CH1TMGCNT0							
D73E	55102	CH1TMGCNT1							
D73F	55103	CH1TMGCNT2							
D740	55104	CH2EN	CH2LOOP	CH2SGN	CH2SINE	CH2STP	–	CH2SBITS	
D741	55105	CH2BADDRLSB							
D742	55106	CH2BADDRMSB							
D743	55107	CH2BADDRBNK							
D744	55108	CH2FREQ0							
D745	55109	CH2FREQ1							
D746	55110	CH2FREQ2							
D747	55111	CH2TADDRLSB							
D748	55112	CH2TADDRMSB							
D749	55113	CH2VOLUME							
D74A	55114	CH2CURADDRLSB							
D74B	55115	CH2CURADDRMSB							
D74C	55116	CH2CURADDRBNK							
D74D	55117	CH2TMGCNT0							
D74E	55118	CH2TMGCNT1							
D74F	55119	CH2TMGCNT2							
D750	55120	CH3EN	CH3LOOP	CH3SGN	CH3SINE	CH3STP	–	CH3SBITS	
D751	55121	CH3BADDRLSB							
D752	55122	CH3BADDRMSB							
D753	55123	CH3BADDRBNK							
D754	55124	CH3FREQ0							
D755	55125	CH3FREQ1							
D756	55126	CH3FREQ2							
D757	55127	CH3TADDRLSB							
D758	55128	CH3TADDRMSB							
D759	55129	CH3VOLUME							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D75A	55130	CH3CURADDRLSB							
D75B	55131	CH3CURADDRMSB							
D75C	55132	CH3CURADDRBNK							
D75D	55133	CH3TMGCNT0							
D75E	55134	CH3TMGCNT1							
D75F	55135	CH3TMGCNT2							

- **ADDRLSB** Lage byte van het DMA-lijstadres (bits 0 - 7) ZONDER EEN DMA-TAAK TE STARTEN (door de Hypervisor gebruikt om taken die DMA gebruiken te ontdooien)
- **ADDRMB** Mega-byte van het adres van de DMA-lijst (bits 20 - 27). Overlapt met ADDRBANK!
- **AUDBLKTO** Time-out van het audio-DMA-blok (alleen-lezen) DEBUG
- **AUDEN** Schakelt audio-DMA in
- **AUDWRBLK** Audio-DMA-blokschrijfbewerkingen (samples worden nog steeds gelezen)
- **BLKD** Audio-DMA geblokkeerd (alleen-lezen) DEBUG
- **CH0RVOL** Volume van het rechterkanaal van audio-DMA-kanaal 0
- **CH1RVOL** Volume van het rechterkanaal van audio-DMA-kanaal 1
- **CH2LVOL** Volume van het linkerkanaal van audio-DMA-kanaal 2
- **CH3LVOL** Volume van het linkerkanaal van audio-DMA-kanaal 3
- **CHANSWAP** Omwisseling van de audio-DMA-kanalen
- **CHXBADDRBNK** Basisadres van audio-DMA-kanaal X (bits 16 - 23)
- **CHXBADDRLSB** Basisadres van audio-DMA-kanaal X (bits 0 - 7)
- **CHXBADDRMSB** Basisadres van audio-DMA-kanaal X (bits 8 - 15)
- **CHXCURADDRBNK** Huidig adres van audio-DMA-kanaal X (bits 16 - 23)
- **CHXCURADDRLSB** Huidig adres van audio-DMA-kanaal X (bits 0 - 7)
- **CHXCURADDRMSB** Huidig adres van audio-DMA-kanaal X (bits 8 - 15)
- **CHXEN** Schakelt audio-DMA-kanaal X in
- **CHXENA** Audio-DMA-IRQ van kanaal X ingeschakeld
- **CHXFLG** Vlag die wordt gezet wanneer een gebeurtenis de audio-DMA-IRQ van kanaal X activeert
- **CHXFREQ0** Frequentie van audio-DMA-kanaal X (bits 0 - 7)

- **CHXFREQ1** Frequentie van audio-DMA-kanaal X (bits 8 - 15)
- **CHXFREQ2** Frequentie van audio-DMA-kanaal X (bits 16 - 23)
- **CHXLOOP** Schakelt looping van audio-DMA-kanaal X in
- **CHXOFFS** IRQ-adresoffset van audio-DMA-kanaal X (bits 15 - 8)
- **CHXSBITS** Sample-bits van audio-DMA-kanaal X (11=16, 10=8, 01=hogere nibble, 00=lagere nibble)
- **CHXSINE** Audio-DMA-kanaal X speelt een sinusgolf van 32 samples af in plaats van DMA-gegevens
- **CHXSTP** Stopvlag van audio-DMA-kanaal X
- **CHXTADDRL** Topadres van audio-DMA-kanaal X (bits 0 - 7)
- **CHXTADDRM** Topadres van audio-DMA-kanaal X (bits 8 - 15)
- **CHXTMGCNT0** Timingteller van audio-DMA-kanaal X (bits 0 - 7)
- **CHXTMGCNT1** Timingteller van audio-DMA-kanaal X (bits 8 - 15)
- **CHXTMGCNT2** Timingteller van audio-DMA-kanaal X (bits 16 - 23)
- **CHXUSGN** Sample-teken van audio-DMA-kanaal X: 0=met teken, 1=zonder teken
- **CHXVOLUME** Afspeelvolume van audio-DMA-kanaal X
- **DBGLFTCLP** Audio-DMA links geklipt (alleen-lezen)
- **DBGRGTCLP** Audio-DMA rechts geklipt (alleen-lezen)
- **EN018B** Schakelt de F018B-modus van de DMA in (voegt een subcommandobyte toe)
- **ETRIG** Stel de lage byte van het DMA-lijstadres in (bits 0 - 7) en start de Enhanced DMA-taak met het lijstadres gespecificeerd als 28-bits flat adres (gebruikt de DMA-optielijst)
- **ETRIGINLINE** Start de Enhanced DMAgic-taak inline (d.w.z. beginnend bij de PC-waarde, waarbij de PC na afloop aan het einde van de taak wordt gezet)
- **ETRIGMAPD** Stel de lage byte van het DMA-lijstadres in (bits 0 - 7) en start de Enhanced DMA-taak met de lijst in de huidige CPU-geheugenindeling (gebruikt de DMA-optielijst)
- **NOMBWRAP** Schakelt de wrap-around van het bron- en doeladres op de MB-grenzen uit (wordt genegeerd in de Hypervisor-modus)
- **NOMIX** Audio-DMA omzeilt de audiomixer

NIET-GEÏMPLEMENTEERDE FUNCTIONALITEIT

De DMAgic van de MEGA65 ondersteunt op dit moment noch memory-swap- noch mini-term-bewerkingen.

Mini-terms waren bedoeld voor het blitten van bitplanes, hetgeen voor de MEGA65 niet nodig is, aangezien deze veel geavanceerdere tekstmodi en trapsgewijze en fractionele DMA-adresverhoging biedt, waarmee teksturen efficiënt kunnen worden gekopieerd en geschaald. Er is bovendien geen bekende software die ooit van deze voorziening gebruik heeft gemaakt, en het blijft onzeker of deze ooit in enige revisie van de DMAgic-chip in de C65-prototypen is geïmplementeerd.

De memory-swap-bewerking zal naar verwachting worden geïmplementeerd, maar kan in de tussentijd worden omzeild door het eerste gebied naar een derde gebied te kopiëren dat als tijdelijk buffer fungeert, vervolgens het tweede gebied naar het eerste, en het derde naar het tweede.

BIJLAGE



VIC-IV Video Interface Controller

- Kenmerken
- Toegang tot VIC-III/IV-registers
- Video-uitvoerformaten, timing en compatibiliteit
- Geheugeninterface
- Hete registers
- Nieuwe modi
- Sprites
- VIC-III errata-niveau

- VIC-II / C64-registers
- VIC-III / C65-registers
- VIC-IV / MEGA65-specifieke registers

KENMERKEN

De VIC-IV is een vierde generatie Video Interface Controller die speciaal voor de MEGA65 is ontwikkeld en beschikt over een zeer goede backwards compatibility met de VIC-II die in de C64 werd gebruikt en de VIC-III die in de C65 werd gebruikt. De VIC-IV kan geprogrammeerd worden alsof het een van deze voorgangersystemen betreft. Daarnaast ondersteunt het een aantal nieuwe functies. Het is eenvoudig om oudere VIC-II/III-functies te combineren met de nieuwe VIC-IV-functies, waardoor de overstap van de VIC-II of VIC-III naar de VIC-IV net zo eenvoudig is als de overstap van de VIC-II naar de VIC-III destijds. Enkele van de nieuwe functies en verbeteringen van de VIC-IV zijn:

- **Directe toegang tot 384KB RAM** (ten opzichte van 16KB/64KB met de VIC-II en 128KB met de VIC-III).
- Ondersteuning voor **32KB 8-bits kleuren-/attribuutgeheugen** (ten opzichte van 2KB op de VIC-III), om zeer grote schermen te ondersteunen.
- **HDTV 720×576 / 800×600 native resolutie** bij zowel 50 Hz als 60 Hz voor **PAL en NTSC**, met **VGA en digitale video**uitvoer.
- **81 MHz pixelklok** (ten opzichte van ∼ 8 MHz met de VIC-II/III), waardoor een breed scala aan nieuwe functies mogelijk wordt.
- Nieuwe 16-kleuren (16×8 pixels per tekencel) en 256-kleuren (8×8 pixels per tekencel) **volkleur-tekstmodi**.
- Ondersteuning voor maximaal **8.192 unieke tekens in een tekenset**.
- **Vier 256-kleuren paletbanken** (ten opzichte van de enkele paletbank van de VIC-III), die elk een **23-bits kleurdiepte** ondersteunen (ten opzichte van de 12-bits kleurdiepte van de VIC-III) en die snel kunnen worden gewisseld om nog kleurrijkere afbeeldingen te produceren dan mogelijk is met de VIC-III.
- Scherm-, bitmap-, kleur- en tekengegevens kunnen op elk **adres met byte-nauwkeurigheid** worden gepositioneerd (in tegenstelling tot de vaste 1KB - 16KB-grenzen van de VIC-II/III).
- **Virtuele schermafmetingen**, die gecombineerd met byte-nauwkeurige gegevensposities **hardware-ondersteuning voor scrollen en pannen in zowel X- als Y-richting** bieden.
- **Nieuwe sprite-modi**: bitplanewijziging, **volkleuren** (15 voorgrondkleuren + transparantie) en getegelde modi, die een breed scala aan nieuwe en spannende sprite-gebaseerde effecten mogelijk maken.
- De mogelijkheid om sprites stapelvormig in bitplane-vorm te rangschikken om **sprites met maximaal 256 kleuren** te produceren.

- Sprites kunnen 64 bits gegevens per rasterlijn gebruiken, waardoor **sprites 64 pixels breed** kunnen zijn bij gebruik van de VIC-II/III mono-/multicolour-modus, of 16 pixels breed bij gebruik van de nieuwe VIC-IV volkleur-spritemodus.
- **Sprite-tegelmodus**, waarmee een sprite horizontaal kan worden herhaald over de volledige rasterlijn, waardoor sprites kunnen worden gebruikt om geanimeerde achtergronden te creëren op een geheugenefficiënte manier.
- Sprites kunnen worden geconfigureerd om een **apart 256-kleurenpalet** te gebruiken in plaats van het palet dat voor andere tekst en afbeeldingen wordt gebruikt, waardoor een kleurrijker beeld mogelijk is.
- **Super-uitgebreide attribuutmodus** die per tekenmodus twee schermgeheugenbytes en twee kleurengeheugenbytes gebruikt en een breed scala aan nieuwe functies ondersteunt, waaronder **alfa-menging/anti-aliasing**, **hardware-kerning/letters met variabele breedte**, hardwarematig horizontaal/verticaal spiegelen, selectie van een alternatief palet en andere krachtige functies die het gemakkelijk maken om zeer dynamische en kleurrijke weergaven te creëren.
- **Raster-Rewrite Buffer** waarmee **hardwarematig gegenereerde pseudo-sprites** mogelijk zijn, vergelijkbaar met "bobs" op Amiga™-computers, maar met het voordeel dat ze in de weergavepijplijn worden gerenderd en dus niet hoeven te worden gewist en opnieuw getekend om te animeren.
- Ook **verschillende 8-bits kleuren-playfields** zijn mogelijk via de Raster-Rewrite Buffer.

Kortom, de VIC-IV is een krachtige evolutie van de VIC-II/III, waarbij het karakter en de eigenaardigheden van de VIC-reeks videocontrollers behouden zijn gebleven.

Voor een volledige beschrijving van de extra registers die de VIC-IV biedt, alsmede documentatie van de legacy VIC-II- en VIC-III-registers, verwijst u naar de betreffende secties van deze bijlage. Het resterende deel van de bijlage richt zich op het beschrijven van de mogelijkheden en het gebruik van veel van de nieuwe functies van de VIC-IV.

TOEGANG TOT VIC-III/IV-REGISTERS

Omdat de nieuwe functies van de VIC-IV allemaal uitbreidingen zijn op de bestaande VIC-II/III-ontwerpen, bestaat er geen concept van het moeten selecteren van de modus waarin de VIC-IV zal werken: deze bevindt zich altijd in VIC-IV-modus. Voor backwards compatibility met software kunnen de vele extra registers van de VIC-IV echter verborgen worden, zodat de VIC-IV zich voordoet als een VIC-II of VIC-III. Dit gebeurt op dezelfde wijze als waarop de VIC-III zijn nieuwe functies verbergt voor legacy VIC-II-software.

Het mechanisme is het write-only KEY-register van de VIC-III ($D02F, 53295 decimaal). De VIC-III verbergt standaard zijn nieuwe functies totdat een "klop"-sequentie

wordt uitgevoerd. Deze bestaat uit het achtereenvolgens schrijven van twee speciale waarden naar $D02F. De volgende samenvattende tabel toont de ondersteunde klopsequenties van de VIC-IV en geeft aan welke VIC-IV-specifiek zijn en welke ook door de VIC-III worden ondersteund:

Eerste waarde Hex (Decimaal)	**Tweede waarde Hex (Decimaal)**	**Effect**	**VIC-IV-specifiek?**
$00 (0)	$00 (0)	Alleen VIC-II-registers zichtbaar (alle nieuwe VIC-III- en VIC-IV-registers zijn verborgen)	Nee
$A5 (165)	$96 (150)	Nieuwe VIC-III-registers zichtbaar	Nee
$47 (71)	$53 (83)	Zowel nieuwe VIC-III- als VIC-IV-registers zichtbaar	Ja
$45 (69)	$54 (84)	Geen VIC-II/III/IV-registers zichtbaar. In plaats daarvan zijn de buffers van de 45E100-Ethernetcontroller zichtbaar	Ja

VIC-II/III/IV detecteren

Het bepalen van welke generatie VIC-II/III/IV in een computer is geïnstalleerd, kan belangrijk zijn voor programma's die slechts bepaalde generaties ondersteunen of die hun grafische weergave willen laten variëren op basis van de mogelijkheden van de computer. Hoewel er hiervoor veel mogelijkheden bestaan, volgt hier een eenvoudige en effectieve methode. Deze berust op het feit dat de VIC-III en VIC-IV de VIC-II-registers niet door de hele I/O-adresruimte herhalen. Terwijl $D000 en $D100 synoniem zijn wanneer een VIC-II aanwezig is (of een VIC-III/IV zijn extra registers verbergt), is dit niet het geval wanneer een VIC-III of VIC-IV alle registers zichtbaar maakt. De aanwezigheid van een VIC-III/IV kan daarom worden vastgesteld door te testen of deze twee locaties aliasen zijn van hetzelfde register of afzonderlijke registers vertegenwoordigen. De detectiesequentie bestaat uit het gebruik van het KEY-register om extra VIC-IV- of VIC-III-registers zichtbaar te maken. Indien een van beide lukt, kan worden aangenomen dat de overeenkomstige VIC-generatie is geïnstalleerd. Omdat de VIC-IV de VIC-III-klopsequenties ondersteunt, moet eerst op de aanwezigheid van een VIC-IV worden getest. Tevens gaan we ervan uit dat de MEGA65 in VIC-IV-modus start, zelfs wanneer C65 BASIC wordt uitgevoerd. De test kan daarom vanuit BASIC worden uitgevoerd, zowel vanuit C64- als C65-modus als volgt:

```
0 REM IN C65-MODUS KUNNEN WE NIET VEILIG NAAR $D02F SCHRIJVEN,
1 REM DUS TESTEN WE HET OP EEN ANDERE MANIER
10 IF PEEK($D018) AND 32 THEN GOTO 65
20 POKE $D000,1:POKE $D02F,71:POKE $D02F,83
30 POKE $D000+256,0:IF PEEK($D000)=1 THEN PRINT"VIC-IV AANWEZIG":END
40 POKE $D000,1:POKE $D02F,165:POKE $D02F,150
50 POKE $D000+256,0:IF PEEK($D000)=1 THEN PRINT"VIC-III AANWEZIG":END
60 PRINT "VIC-II AANWEZIG":END
65 REM WE GAAN ERVAN UIT DAT WE HIER EEN C65 HEBBEN
70 V1=PEEK($D050):V2=PEEK($D050):V3=PEEK($D050)
80 IF V1<>V2 OR V1<>V3 OR V2<>V3 THEN PRINT "VIC-IV PRESENT":END
90 GOTO 40
```

Regel 10 van dit programma controleert of het scherm een veelvoud van 2KB is. Omdat het scherm op de C64 op 1KB staat, slaagt deze test niet en gaat de uitvoering verder naar regel 20. Regel 20 schrijft 1 naar een van de VIC-II-spritepositieregisters, 53248, en vervolgens de MEGA65-klop naar het KEY-register, 53295. Regel 30 schrijft naar 53248 + 256, wat op de C64 een spiegel is van 53248, maar op een MEGA65 met VIC-IV I/O ingeschakeld een van de rode paletregisters is. Na het schrijven naar 53248 + 256 controleert het programma of het register op 53248 is gewijzigd door de schrijfactie naar 53248 + 256. Indien dat het geval is, verwijzen de twee adressen naar hetzelfde register. Dit gebeurt op zowel een C64 als een C65, maar niet op een computer met een VIC-IV. Wanneer 53248 dus niet is gewijzigd, melden we dat we een VIC-IV hebben gedetecteerd. Indien het schrijven naar 53248 + 256 de waarde in register 53248 wel heeft gewijzigd, gaat het programma verder naar regel 40, die opnieuw naar 53248 schrijft en ditmaal de VIC-III-klop naar het KEY-register schrijft. Regel 50 is gelijk aan regel 30, maar omdat deze na een VIC-III-klop verschijnt, maakt ze de detectie van een VIC-III mogelijk. Indien ten slotte noch een VIC-IV noch een VIC-III wordt gedetecteerd, concluderen we dat er alleen een VIC-II aanwezig moet zijn.

Omdat de MEGA65 de enige C64-klassecomputer is die over een VIC-IV beschikt, kan dit als een *de facto*-test voor de aanwezigheid van een MEGA65-computer worden gebruikt. Detectie van een VIC-III kan op vergelijkbare wijze worden aangenomen dat deze de aanwezigheid van een C65 aanduidt.

VIDEO-UITVOERFORMATEN, TIMING EN COMPATIBILITEIT

Integrated Marvellous Digital Hookup™ (IMDH™) Digitale video-uitvoer

De MEGA65 beschikt over VGA-analoge video-uitvoer en Integrated Marvellous Digital Hookup™ (IMDH™). Dit wijkt in een aantal opzichten af van bestaande gangbare digitale videostandaarden:

1. We hebben er geen nieuwe connector voor bedacht: we hebben in plaats daarvan de meest gebruikte digitale videoconnector gebruikt die al in omloop is. Uw bestaande kabels zouden dus probleemloos moeten werken!
2. We hebben hem niet opzettelijk incompatibel gemaakt met een bestaande digitale videostandaard. Uw bestaande televisies en monitoren zouden dus probleemloos moeten werken!
3. We eisen geen woekerprijs van andere leveranciers voor het gebruik van de IMDH™-digitale videostandaard, door hen jaarlijks $10.000 in rekening te brengen, slechts voor de toestemming om één apparaat te mogen verkopen. Dit betekent dat de MEGA65 goedkoper is voor u!
4. De IMDH™-standaard staat geen inhoudsbescherming of andere soevereiniteitsaantastende ongein toe. Als u de video hebt geproduceerd, mag u ermee doen wat u wilt!

Aansluiten op eigenzinnige propriëtaire digitale videostandaarden

Er bestaan digitale videostandaarden die volkomen achterhaald zijn in vergelijking met IMDH™. Dankzij de open aanpak van IMDH™op het gebied van interoperabiliteit zouden deze in de meeste gevallen echter zonder problemen met de MEGA65 moeten werken. Zoek eenvoudigweg een videokabel die past op de IMDH™-connector op de achterkant van uw MEGA65 en sluit deze aan op uw MEGA65 en een televisie, monitor of projector die over dezelfde connector beschikt.

Helaas hebben niet alle fabrikanten hun apparaten ingediend voor de IMDH™-compliantietest bij het MEGA65-team. Dit betekent dat sommige televisies en monitoren helaas niet IMDH™-conform zijn. Hoewel de meeste televisies en monitoren dus met de MEGA65 zullen werken, zou het kunnen dat u er een paar moet uitproberen om een bevredigend resultaat te bereiken. Indien u een monitor vindt die niet met de MEGA65 werkt, horen we dat graag van u. Meld het probleem tevens bij de monitorfabrikant en beveel hen aan hun apparaten in te dienen voor de IMDH™-compliantietest.

De VIC-IV is ontworpen voor gebruik in de MEGA65 en verwante systemen, waaronder de MEGAphone-familie van draagbare apparaten. De VIC-IV ondersteunt zowel VGA- als digitale video-uitgang via de niet-propriëtaire IMDH™-interface. Daarnaast

ondersteunt de VIC-IV parallelle digitale video-uitgang voor het aansturen van LCD-beeldschermpanelen. Er is veel zorg besteed aan het ontwerpen van een gemeenschappelijke video-voorkant die deze drie uitgangsmodi ondersteunt.

Voor eenvoud en nauwkeurigheid van de frametiming voor bestaande software is het videoformaat doorgaans gebaseerd op de HDTV PAL- en NTSC-modi met 720×576/480 (576p en 480p) en een pixelklok van 27 MHz. Dit formaat is ideaal voor digitale video en LCD-beeldschermpanelen. Niet alle VGA-beeldschermen ondersteunen deze modi echter, met name 720×576 bij 50 Hz.

Met het oog op de achterwaartse compatibiliteit met de VIC-II en VIC-III heeft dit weergaveformaat verschillende gevolgen die voor de meeste programma's geen problemen veroorzaken, maar wel tot enkele gedragsverschillen kunnen leiden:

1. Omdat de VIC-IV-weergave progressief is in plaats van geïnterlinieerd, worden er twee fysieke rasterlijnen geproduceerd voor elke logische VIC-II- of VIC-III-rasterlijn. Dit betekent dat er 63 of 65 cycli per logische dubbele rasterlijn zijn, in plaats van per fysieke 576p/480p-rasterlijn. Dit kan enkele kleine visuele artefacten veroorzaken wanneer programma's aannemen op welk punt op een horizontale lijn de VIC-IV op dat moment tekent, bijvoorbeeld bij het wijzigen van de rand- of schermkleur.

2. De VIC-IV volgt niet het gedrag van de VIC-III, die wijzigingen in videomodi (bijv. tussen tekst- en bitmapmodus) op tekenen toestond, evenmin als de VIC-II-strategie waarbij dergelijke wijzigingen onmiddellijk van kracht werden. De VIC-IV past wijzigingen daarentegen toe aan het begin van elke rasterlijn. Dit kan enkele kleine artefacten veroorzaken.

3. De VIC-IV gebruikt een enkel raster-weergavebuffer die wordt gevuld met behulp van de interne 81 MHz-pixelklok van de VIC-IV, waarna deze wordt weergegeven met de 27 MHz-pixelklok voor de uitgang. Hierdoor wordt de weergave-inhoud van een rasterlijn doorgaans veel eerder in de rasterlijn gerenderd dan op de VIC-II of VIC-III. Dit kan artefacten veroorzaken op beeldschermen, met name in demo's die specifiek gedrag van de VIC-II op bepaalde cycli in een rasterlijn benutten, bijvoorbeeld voor effecten als VSP of FLI. Dergelijke effecten worden op de huidige revisie van de VIC-IV waarschijnlijk niet correct weergegeven. Er is verbeterde ondersteuning voor deze functies gepland voor een toekomstige revisie van de VIC-IV.

4. De 1280×200- en 1280×400-weergavemodi van de VIC-III worden momenteel niet ondersteund, omdat ze niet zinvol op een modern beeldscherm kunnen worden weergegeven en er geen software bekend is die deze functie ondersteunt of gebruikt.

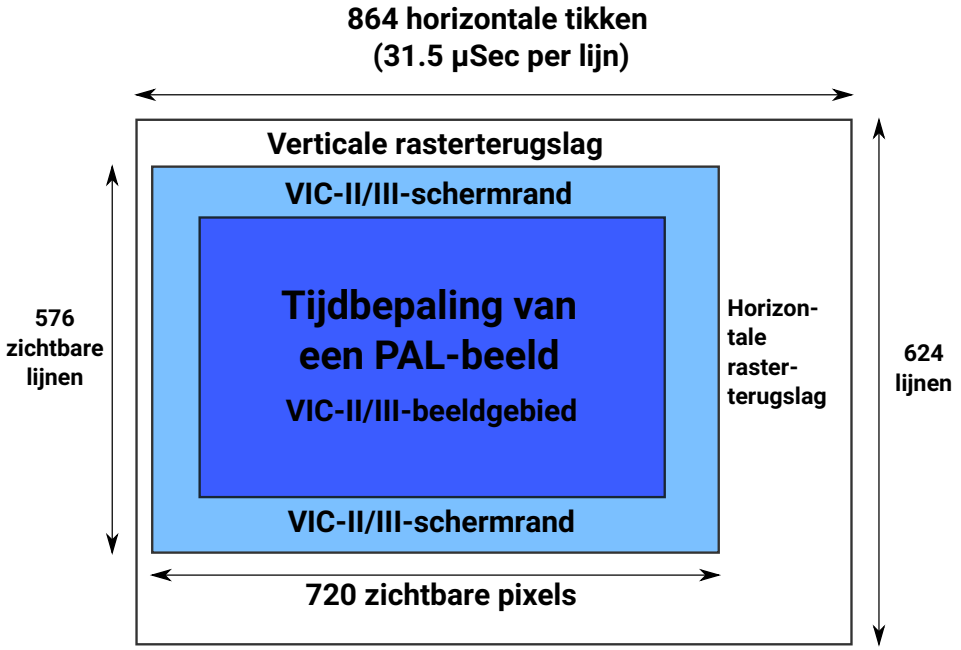
Frametiming

De frametiming is ontworpen om overeen te komen met die van de combinatie 6502 + VIC-II van de C64. Zowel PAL- als NTSC-timing wordt ondersteund, en het aantal cycli per logische rasterlijn, het aantal rasterlijnen per frame en het aantal cycli per frame worden dienovereenkomstig aangepast. Daartoe gebruikt de VIC-IV doorgaans

de HDTV-modi 576p 50 Hz (PAL) en 480p 60 Hz (NTSC), met timing die zo dicht mogelijk bij de dubbelgescande PAL- en NTSC-composiet-tv-modi van de VIC-II wordt afgestemd.

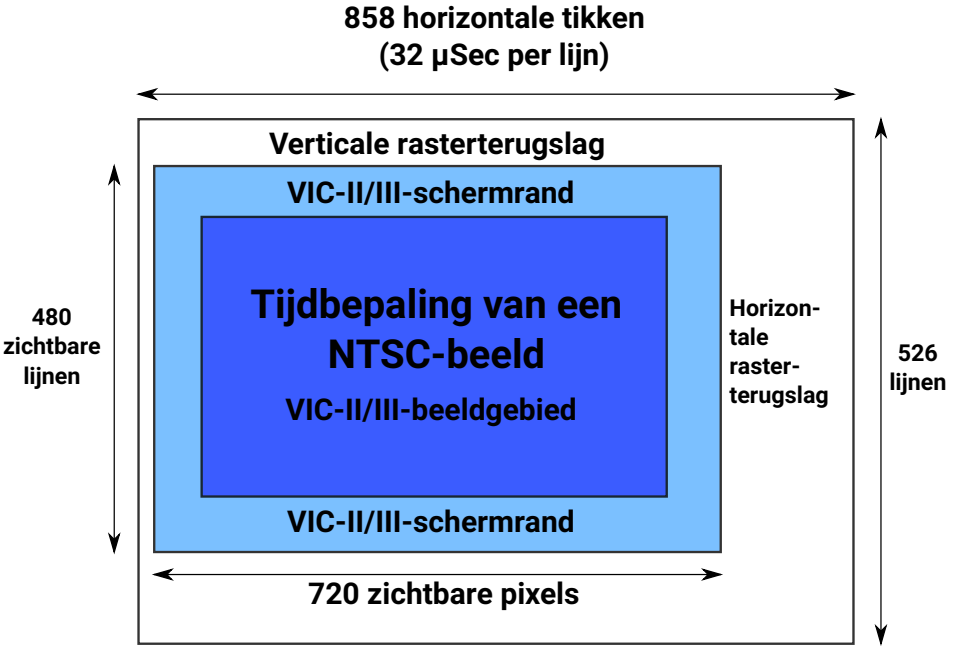
De VIC-IV genereert timingimpulsen van ongeveer 1 MHz die door de 45GS02-processor worden gebruikt, zodat de juiste effectieve frequentie wordt geleverd bij werking in de C64-, C128- en C65-compatibiliteitsmodi met respectievelijk 1 MHz, 2 MHz en 3.5 MHz. Hierdoor kan het enkele systeem overschakelen tussen nauwkeurige PAL- en NTSC-CPU-timing, evenals tussen videomodi. De exacte frequentie varieert tussen PAL- en NTSC-modi om het gedrag van PAL versus NTSC bij de C64-, C128- en C65-processor- en videotiming na te bootsen.

Het PAL-frame bestaat uit 624 fysieke rasterlijnen met elk 864 pixelklokken. De pixelklok is 27 MHz, oftewel 1/3 van de VIC-IV-pixelklok. Het zichtbare frame is 720×576 pixels, waarvan het geheel in de VIC-IV-modus kan worden gebruikt. In de VIC-II- en VIC-III-modi verkleint het randgebied de bruikbare grootte tot 640×400 pixels. In de VIC-II-modus en VIC-III 200H-modi wordt de weergave dubbelgescand, waarbij twee fysieke rasterlijnen van 31.5 microseconden overeenkomen met één VIC-II-stijl rasterlijn van 63 microseconden. Elk frame bestaat dus uit 312 VIC-II-rasterlijnen van elk 63 microseconden, exact gelijk aan die van een PAL-C64.



Het NTSC-frame bestaat uit 526 fysieke rasterlijnen met elk 858 pixelklokken. De pixelklok is 27 MHz, oftewel 1/3 van de VIC-IV-pixelklok. Het zichtbare frame is 720×480 pixels, waarvan het geheel in de VIC-IV-modus kan worden gebruikt. In de VIC-II- en VIC-III-modi verkleint het randgebied de bruikbare grootte tot 640×400

pixels. In de VIC-II-modus en VIC-III 200H-modi wordt de weergave dubbelgescand, waarbij twee fysieke rasterlijnen van 32 microseconden overeenkomen met één VIC-II-stijl rasterlijn van 64 microseconden. Elk frame bestaat dus uit 263 VIC-II-rasterlijnen van elk 64 microseconden, overeenkomstig de meest voorkomende NTSC-videotiming van de C64.



Omdat niet alle VGA-beeldschermen deze HDTV-videomodi ondersteunen, is er een compatibiliteitsmodus beschikbaar die een VGA-stijl modus van 640×480 biedt. Omdat de pixelklok van de MEGA65 vaststaat op 27 MHz, werkt deze modus op 63 Hz. Dit zou op de overgrote meerderheid van VGA-beeldschermen moeten werken. Er zullen geen problemen optreden met de PAL/NTSC-modi bij gebruik van de digitale video-uitgang van de MEGA65 op de overgrote meerderheid van IMDH™-compatibele beeldschermen en televisies.

Om te bepalen of de MEGA65 in PAL- of NTSC-modus werkt, kunt u het Freezer-menu openen, dat de huidige videomodus weergeeft, of vanuit een programma het PALNTSC-signaal controleren (bit 7 van $D06F, decimaal 53359). Als deze bit gezet is, werkt het systeem in NTSC-modus; als deze gewist is, in PAL-modus. Deze bit kan worden gewijzigd om tussen de modi te schakelen, bijv.:

```
10 REM C65+MEGA65-I/O INSCHAKELEN
20 IF PEEK($D018)<32 THEN POKE $D02F,ASC("G"):POKE $D02F,ASC("S")
30 REM NTSC-bit controleren
40 NTSC=PEEK($D06F) AND 128
50 REM STATUS WEERGEVEN EN OM OMSCHAKELING VRAGEN
60 PRINT"MEGA65 IS IN ";:IF NTSC THEN PRINT"NTSC MODE":ELSE PRINT"PAL MODE"
70 INPUT"SWITCH MODES (Y/N)? ",A$
80 REM NTSC-BIT OMSCHAKELEN
90 IF A$="Y" THEN POKE $D06F,PEEK($D06F) XOR 128:ELSE END
100 REM NIEUWE STATUS WEERGEVEN
110 NTSC=PEEK($D06F) AND 128
120 PRINT"MEGA65 IS IN ";:IF NTSC THEN PRINT"NTSC MODE":ELSE PRINT"PAL MODE"
```

Fysieke en logische rasters

Fysieke rasterlijnen per frame verwijzen naar het aantal daadwerkelijke rasterlijnen in de PAL- of NTSC-EDTV-videomodi die door de MEGA65 worden gebruikt. Logische rasterlijnen verwijzen naar het aantal VIC-II-stijl rasterlijnen per frame. Elke logische rasterlijn bestaat uit twee fysieke rasterlijnen per regel, aangezien EDTV-modi dubbelgescande modi zijn vergeleken met de oorspronkelijke PAL- en NTSC-SDTV-modi van de C64. De frame-parameters van de VIC-IV voor PAL en NTSC zijn als volgt:

Standaard	Cycli per rasterlijn	Fysieke rasterlijnen per frame	Logische rasterlijnen per frame
PAL	63	626	312
NTSC	65	526	263

Het resultaat is dat de frames op de VIC-IV exact hetzelfde aantal ~1 MHz-CPU-cycli bevatten als op de VIC-II.

Bad lines

De VIC-IV kent van nature geen "bad lines", omdat de VIC-IV eigen geheugenbussen heeft naar het hoofdgeheugen en het kleurengeheugen (colour RAM) van de MEGA65. Hierdoor kunnen zowel de processor als de VIC-IV tegelijkertijd toegang krijgen tot het geheugen, in tegenstelling tot de C64 of C65, waarbij de toegang wordt afgewisseld.

Om de compatibiliteit te verbeteren, geeft de VIC-IV echter aan wanneer een "bad line" zou hebben plaatsgevonden op de VIC-II. De 45GS02-processor van de MEGA65 aanvaardt deze bad-line-signalen en onderbreekt de CPU gedurende 40 klokcycli, tenzij de processor op volle snelheid draait, in welk geval deze worden genegeerd. Dit verbetert de timingcompatibiliteit met de VIC-II aanzienlijk. De timing is echter niet exact, omdat de huidige revisie van de 45GS02 precies 40 cycli onderbreekt in plaats van 40-43 cycli, afhankelijk van de instructie die op dat moment

wordt uitgevoerd. Daarnaast onderbreken de VIC-IV en 45GS02 momenteel niet voor sprite-ophaaloperaties.

De bad-line-emulatie wordt beheerd door bit 0 van $D710: het zetten van deze bit schakelt bad-line-emulatie in, en het wissen ervan voorkomt dat een bad line tijd van de processor steelt.

GEHEUGENINTERFACE

De VIC-IV ondersteunt tot 16 MB direct toegankelijk RAM voor videogegevens, maar momenteel bieden alle bestaande modellen slechts 384 KB adresseerbaar RAM. In MEGA65-systemen wordt het tweede blok van 128 KB RAM (in het geheugengebied van 128 KB tot 256 KB) doorgaans gebruikt om een C65-compatibele ROM op te slaan, waardoor 256 KB RAM voor de gebruiker beschikbaar blijft. Indien de software zo is geschreven dat het gebruik van C65-ROM-routines niet nodig is, kan de volledige 384 KB RAM door het programma worden gebruikt.

Alle MEGA65-modellen ondersteunen momenteel 32 KB kleurengeheugen (colour RAM), maar er zijn plannen om het nieuwste R3-bord 64 KB kleurengeheugen (of mogelijk zelfs 128 KB) te laten ondersteunen.

De VIC-IV ondersteunt alle methoden van de oudere VIC-II en VIC-III voor de toegang tot dit geheugen, waaronder het gebruik van 16KB-banken door de VIC-II en de Display Address Translator (DAT) van de VIC-III. Dit extra geheugen kan voor teken- en bitmapweergaven worden gebruikt, evenals voor sprites. De bitplanemodi van de VIC-III blijven echter beperkt tot het gebruik van slechts de eerste 128KB van het geheugen, omdat de VIC-IV de bitplanemodus niet uitbreidt.

Schermgeheugen lokaliseren en verplaatsen

De VIC kan worden geconfigureerd om het tekstschermgeheugen overal in de eerste 384KB te zoeken. Uw programma kan bepalen waar het schermgeheugen komt te staan en het startadres in de hardware-registers instellen. Dit adres wordt opgeslagen in de SCRNPTR-registers ($D060 - $D063, 53344 - 53347 decimaal). Om het schermgeheugen bijvoorbeeld op adres 12345 in te stellen:

```
REM C65+MEGA65-I/O INSCHAKELEN
IF PEEK($D018)<32 THEN POKE $D02F,ASC("G"):POKE $D02F,ASC("S")

WPOKE $D060,$2345:POKE $D062,$1
```

Om een bestaand scherm te gebruiken dat door de KERNAL is ingesteld (het scherm dat de gebruiker bij het opstarten ziet), kunt u het startadres uit deze registers lezen. Houd er rekening mee dat de KERNAL in sommige omstandigheden de locatie van het schermgeheugen wijzigt. De beste praktijk is daarom dit adres binnen het programma uit de registers te lezen en de locatie ervan niet te veronderstellen.

```
10 SC=WPEEK($D060)+(PEEK($D062)*65536)
20 POKE SC,1 : REM SCHRIJFT EEN "A" NAAR DE LINKERBOVENHOEK
```

Tekengeneratorgegevens verplaatsen

De locatie van de tekengeneratorgegevens kan eveneens met precisie op byteniveau worden ingesteld via de CHARPTR-registers op $D068 - $D06A (53352 - 53354 decimaal). Zoals gebruikelijk bevat het eerste van deze registers de minst significante byte en het laatste de meest significante byte. De drie bytes maken plaatsing van tekengeneratorgegevens overal in de eerste 16MB van het geheugen mogelijk. Voor systemen met minder dan 16MB geheugen waartoe de VIC-IV toegang heeft, dienen de bovenste adresbits op nul te staan.

Om bijvoorbeeld aan te geven dat de tekengeneratorgegevens vanaf $41200 (266752 decimaal) moeten worden gelezen, kan het volgende worden gebruikt. Houd er rekening mee dat het commando WPOKE kan worden gebruikt om twee bytes als een woord naar een geheugen- of I/O-locatie te schrijven. Wij gebruiken daarom WPOKE om $00 naar $D068 en $12 naar $D069 te schrijven, en een extra POKE om de hoge byte $A naar $D06A te schrijven door het adres door 65536 te delen:

```
REM C65+MEGA65-I/O INSCHAKELEN
IF PEEK($D018)<32 THEN POKE $D02F,ASC("G"):POKE $D02F,ASC("$")
REM HEX $41200 IS MAKKELIJK VERDEELD IN ZIJN 3 BYTES $00, $12, $4
REM WPOKE ZET DE ONDERSTE TWEE BYTES IN EEN COMMANDO EN
REM DE VOLGENDE POKE ZET DE BOVENSTE BYTE
A=$41200
WPOKE $D068,A
POKE $D06A,A/65536
```

Het kleuren-/attribuutgeheugen verplaatsen

Het gebruikte deel van het kleurengeheugen kan op vergelijkbare wijze worden ingesteld met de COLPTR-registers ($D064 - $D065, 53348 - 53349 decimaal). De waarde is namelijk een verschuiving vanaf het begin van het kleuren-/attribuutgeheugen. Dit komt omdat het kleuren-/attribuutgeheugen van de MEGA65, net als op de C64, een afzonderlijk geheugenonderdeel is met een eigen verbinding met de VIC-IV. Standaard zijn de COLPTR's op nul ingesteld, wat het gedrag van de VIC-II/III nabootst. Om de weergave het kleuren-/attribuutgeheugen vanaf verschuiving $4000 te laten gebruiken, kan men bijvoorbeeld iets als het volgende gebruiken:

```
REM C65+MEGA65-I/O INSCHAKELEN
IF PEEK($D018)<32 THEN POKE $D02F,ASC("G"):POKE $D02F,ASC("$")
REM STEL COLPTR IN OP $4000, VERDEELD IN $00-LSB en $40-MSB
POKE $D064,$00
POKE $D065,$40
```

Spritepointers en sprite-afbeeldingen verplaatsen

De locatie van de spritepointers kan eveneens worden verplaatst, en sprites kunnen hun gegevens overal in de eerste 4MB van het geheugen laten hebben. Dit wordt bereikt door eerst de locatie van de spritepointers in te stellen via de SPRPTRADR-registers ($D06C - $D06E, 53356 - 53358 decimaal; houd er echter rekening mee dat alleen de onderste 7 bits van $D06E worden gebruikt, omdat de hoogste bit voor het SPRPTR16-signaal wordt gebruikt). Hierdoor kan de lijst van acht spritepointers van het einde van het schermgeheugen naar een willekeurige locatie in de eerste 8MB van het geheugen worden verplaatst. SPRPTRADR moet op een 16-byte-grens (een veelvoud van 16) zijn uitgelijnd.

Om sprites zelf overal in de eerste 4MB van het geheugen te kunnen plaatsen, moet de SPRPTR16-bit in $D06E worden gezet. In deze modus worden twee bytes gebruikt om de locatie van elke sprite aan te geven, in plaats van één byte. De lijst van spritepointers is daardoor 16 bytes lang, in plaats van 8 bytes zoals op de VIC-II/III. Wanneer SPRPTR16 is ingeschakeld, moet de locatie van de spritepointers altijd expliciet via de SPRPTRADR-registers worden ingesteld.

Om de spritepointers bijvoorbeeld op locatie 800 - 815 te plaatsen, kunt u min of meer de volgende code gebruiken. Houd er rekening mee dat enige kunstgrepen nodig zijn om de SPRPTR16-bit ongewijzigd te laten, en ook om de AND-binaire operator, die niet met waarden groter dan 65535 werkt, te omzeilen:

```
REM C65+MEGA65-I/O INSCHAKELEN
IF PEEK($D018)<32 THEN POKE $D02F,ASC("G"):POKE $D02F,ASC("$")
POKE $D06C,(800-INT(800/65536)*65536) AND 255
POKE $D06D,INT(800/256) AND 255
POKE $D06E,(PEEK($D06E) AND 128)+INT(800/65536)
```

De locatie van elke sprite-afbeelding blijft een veelvoud van 64 bytes, waardoor er op elk moment tot 65,536 unieke sprite-afbeeldingen kunnen worden gebruikt, indien het systeem over voldoende geheugen beschikt (4MB of meer). In deze modus wordt de 16KB-banking van de VIC-II genegeerd, en is de locatie van de spritegegevens eenvoudigweg 64 × de pointerwaarde. Om de gegevens voor een sprite op $C000 (49152 decimaal) te hebben, zou dit bijvoorbeeld sprite-locatie 768 zijn, omdat 49152 gedeeld door 64 = 768. Vervolgens moeten wij 768 in hoge en lage bytes splitsen om de twee pointerbytes in te stellen: 768 = 256×3, met rest 0. Dit zou dus vereisen dat de twee spritepointerbytes 0 (lage byte, die als eerste komt) en 3 (hoge byte) zijn. Indien de spritepointers zich op $7F8 (2040 decimaal) bevonden, zou het instellen van de eerste sprite op sprite-afbeelding 768 kunnen worden gedaan met zoiets als:

```
POKE 2040,768-256*INT(768/256)
POKE 2041,INT(768/256)
```

HETE REGISTERS

Sommige VIC-IV-registers ondersteunen functies die vergelijkbaar zijn met die van de VIC-II en VIC-III, maar met uitgebreide mogelijkheden. Voor achterwaartse compatibiliteit zorgt het schrijven naar specifieke VIC-II- en VIC-III-registers er ook voor dat de bijbehorende VIC-IV-registers worden gereset met consistente waarden. Dit gedrag kan worden geconfigureerd met de HOTREG-vlag in bit 7 van $D05D.

De onderste vier bits van register $D018 (**CB**) stellen bijvoorbeeld het tekensetadres van de VIC-II in, als een veelvoud van 1 KiB. De VIC-IV kan de tekenset op elk 24-bits adres plaatsen met behulp van $D06A (**CHARPTRBNK**), $D068 (**CHARPTRLSB**) en $D069 (**CHARPTRMSB**). Indien u CB instelt, worden de VIC-IV-registers ook bijgewerkt om hiermee overeen te komen.

De volledige reeks "hete" VIC-II-registers die van invloed zijn op de VIC-IV-registers omvat:

- $D011 (53265): RB8, ECM, BMM, BLNK, RSEL, YSCL
- $D016 (53270): RST, MCM, CSEL, XSCL
- $D018 (53272): VS, CB
- $D031 (53297): VIC-III-modi: H640, FAST, ATTR, BPM, V400, H1280, MONO, INT
- De VIC-II-bankbits van $DD00 (56576) (CIA 2 PORTA)

Telkens wanneer een van die registers wordt gewijzigd, zelfs door de bestaande waarde er weer in terug te schrijven, worden alle bijbehorende VIC-IV-registers bijgewerkt op basis van de VIC-II-registerwaarden, zelfs die welke geen verband houden met het register waarnaar is geschreven. Dit omvat de FAST-vlag, welke zich op een byte-locatie van een heet register ($D031) bevindt, maar geen verband houdt met de videoparameters en geen bijbehorend VIC-IV-register heeft.

De registers die tijdens dit proces worden gewijzigd, staan hieronder vermeld. Houd er rekening mee dat sommige van deze registers intern zijn aan de VIC-IV en niet rechtstreeks door de gebruiker kunnen worden opgevraagd of gewijzigd. Waar dit het geval is, worden er geen adressen voor de registers vermeld.

- **X-positie van de linkerrandgrens.** Dit interne register wordt bijgewerkt om de linkerrand in te stellen op de breedte die in de "Single Side Border Width"-registers is aangegeven ($D05C bevat de LSB, en bits 0 – 5 van $D05D bevatten de MSB van de breedte van de zijrand. Houd er rekening mee dat de breedte van de zijrand is gebaseerd op de afmetingen van het videoframe op laag niveau, niet op de weergavegrootte van het scherm van de videomodus. Wanneer het 38/40-kolommenveld van $D016 op 38 kolommen is ingesteld, verschijnt de linkerrandgrens 14 pixels rechts van zijn normale positie.
- **X-positie van de rechterrandgrens.** Dit is hetzelfde als de linkerrandgrens, maar dan voor de rechterrand van het scherm. Houd er rekening mee dat wan-

neer de 38/40-kolommenvlag op 38 kolommen is ingesteld, de rechterrandgrens 17 pixels naar links van zijn normale positie wordt verplaatst.

- **Y-positie van de bovenste randgrens ($D048 LSB, bits 0 – 3 van $D049 MSB).** Dit interne register wordt ingesteld op de normale bovenste positie van het scherm, minus de waarde van het RASLINE0-veld in bits 0 – 5 van $D06F. Wanneer het 24/25-rijenveld van $D011 op 24 rijen is ingesteld, wordt de bovenste randgrens met 8 rasterlijnen verlaagd.
- **Y-positie van de onderste randgrens ($D04A LSB, bits 0 – 3 van $D04B MSB).** Dit interne register wordt ingesteld op de normale bovenste positie van het scherm, plus 400 rasterlijnen, om het normale 400px hoge primaire weergavegebied binnen de randen te creëren. Wanneer het 24/25-rijenveld van $D011 op 24 rijen is ingesteld, wordt de bovenste randgrens met 8 rasterlijnen verhoogd.
- **Verticale schaal van de tekengenerator ($D05B).** Dit register wordt ingesteld op 0 voor de V200- of 1 voor de V400-modus, zodat elke pixelrij in een teken respectievelijk 1 of 2 pixels hoog wordt, volgens de V400-vlag.
- **Aantal weer te geven tekenrijen ($D07B).** Dit register wordt ingesteld op 25-1 = 24 of 50-1 = 49 om respectievelijk 25 of 50 tekstrijen weer te geven. Houd er rekening mee dat wanneer $D011 wordt gebruikt om de verticale randen naar binnen te brengen om het aantal zichtbare tekenrijen te verminderen, de VIC-IV nog steeds alle 25 of 50 rijen tekent.
- **X-positie waar de tekenweergave begint ($D04C LSB, bits 0 – 3 van $D04D MSB).** Dit register wordt ingesteld op een positie ten opzichte van de rand van het 40-kolommen brede tekstscherm, plus 2× de positie van de vloeiende verschuiving die in $D016 is aangegeven.
- **Y-positie waar de tekenweergave begint ($D04E LSB, bits 0 – 3 van $D04F MSB).** Dit register wordt ingesteld op de bovenrand van de verticale rand, minus het VIC-II First Raster-aanpassingsregister (bits 0 – 5 van $D06F), plus eventuele verschuiving als gevolg van de verticale bits voor vloeiende verschuiving in $D011.
- **Virtuele rijbreedte ($D058 LSB, $D059 MSB), d.w.z. het aantal bytes scherm- en kleurengeheugen waarmee de VIC-IV vooruitgaat bij het weergeven van elke opeenvolgende tekenrij.** Dit register wordt ingesteld op 40 als de H640-vlag is gewist, of op 80 als de H640-vlag is gezet, waardoor de voortgang overeenkomt met het aantal weer te geven tekens.
- **Breedte van de weergaverij ($D05E LSB, bits 4 – 5 van $D063 MSB).** Dit register wordt ingesteld op 40 als de H640-vlag is gewist, of op 80 als de H640-vlag is gezet.
- **Basisadres van het schermgeheugen ($D060 – $D062, dat een 24-bits adres voorstelt).** Dit adres wordt gereset naar het adres zoals berekend aan de hand van $D018 en $DD00, zoals op de C64.

- **Spritepointer-adres van de VIC-II ($D06C - $D06E, dat een 24-bits adres voorstelt).** Dit register wordt gereset naar de normale locatie aan het einde van het schermgeheugen van de huidige modus. Indien de H640-vlag is gezet, gebeurt dit aan het einde van het 2KB-gebied van het schermgeheugen, of indien de H640-vlag niet is gezet, wijst het naar het einde van het 1KB-gebied van het schermgeheugen, zoals op een C64.
- **Basisadres van de tekenset ($D068 - $D06A, dat een 24-bits adres voorstelt).** Houd er rekening mee dat de hete registerfunctie alleen de onderste 16 bits van het tekensetadres instelt. Dat wil zeggen dat $D06A niet wordt gewist. Dit is bewust gedrag, waardoor het gemakkelijker wordt om in bestaande VIC-II-georiënteerde software de tekenset te vervangen door een andere tekenset in een andere geheugenbank.
- **Basisadres van het kleurengeheugen ($D064 LSB, $D065 MSB).** Deze registers worden naar nul gereset, waardoor de VIC-IV verwacht dat het kleurengeheugen voor het scherm zich in het eerste deel van het kleurengeheugen bevindt, om compatibel te zijn met de VIC-II en VIC-III.

Dit gedrag van de VIC-II-registers is vooral bedoeld voor software uit het verleden die geen weet heeft van (en daarom nooit naar) VIC-IV-registers schrijft. De heteregisterpropagatie kan worden uitgeschakeld door het HOTREG- ("hete register") signaal te wissen: bit 7 van $D05D (53341). Indien u de HOTREG-vlag wist, moet u de VIC-IV-registers rechtstreeks bijwerken wanneer u de parameters van de videomodus wilt wijzigen.

Indien u een wijziging in een heet register aanbrengt terwijl HOTREG is uitgeschakeld en HOTREG vervolgens weer inschakelt, worden alle hete registers onmiddellijk bijgewerkt. Er zijn zeldzame gevallen waarin u een wijziging in een heet register wilt aanbrengen zonder de VIC-IV-registers bij te werken, maar de hete registers na de wijziging toch ingeschakeld wilt hebben. Om dit te doen, kunt u de openstaande bijwerking annuleren door vlak voordat u de hete registers weer inschakelt opnieuw een 0 naar HOTREG te schrijven. De volledige procedure is:

1. Schrijf 0 naar HOTREG om hete registers uit te schakelen.
2. Werk het VIC-II-register bij dat u wilt wijzigen.
3. Schrijf 0 naar HOTREG om de openstaande bijwerking te annuleren.
4. Schrijf 1 naar HOTREG om hete registers weer in te schakelen.

In assembleertaal:

```
lda #%10000000
trb $d05d        ; hete registers uitschakelen

lda #%01000000
tsb $d031        ; een VIC-II/III-register bijwerken

lda #%10000000
trb $d05d        ; openstaande bijwerking wissen
tsb $d05d        ; hete registers weer inschakelen
```

NIEUWE MODI

Waarom de nieuwe VIC-IV-modi teken- en bitmapmodi zijn en geen bitplanemodi

De nieuwe videomodi van de VIC-IV zijn afgeleid van de teken- en bitmapmodi van de VIC-II, in plaats van de bitplanemodi van de VIC-III. Deze keuze was gebaseerd op diverse praktische aspecten van het programmeren van een geheugenbeperkte 8-bit thuiscomputer:

1. Bitplanes vereisen dat dezelfde hoeveelheid geheugen aan elk gebied op het scherm wordt toegewezen, ongeacht of het lege ruimte of complexe afbeeldingen weergeeft. Met bitplanes is het niet mogelijk om inhoud uit een afbeelding opnieuw te gebruiken in een ander deel van die afbeelding. De meeste C64-spellen maken echter gebruik van sterk repetitieve schermen, waarbij veelgebruikte elementen op diverse plaatsen op het scherm voorkomen, waarvan Boulder Dash en Super Giana Sisters goede voorbeelden zijn.
2. Bitplanes maken het ook moeilijk om een beeldscherm bij te werken, omdat elke pixel uniek is: er is geen mogelijkheid om een wijziging, bijvoorbeeld in de animatie van een schermelement, door te voeren op alle plaatsen tegelijk. De diamantanimaties in Boulder Dash zijn een goed voorbeeld van dit probleem. Het moeten wijzigen van meerdere afzonderlijke bytes in elke bitplane vormt een verhoogde rekenkundige belasting, hetgeen de reden was dat er voor de Amiga AAA-chipset werd gevraagd zogenaamde “chunky”-modi op te nemen, in plaats van uitsluitend op bitplanes gebaseerde modi. Hoewel de Display Address Translator (DAT) en DMAgic van de C65 enige verlichting bieden voor dit probleem, is die verlichting slechts gedeeltelijk.
3. Scrollen met behulp van de bitplanes van de C65 vereist het kopiëren van de gehele bitplane, omdat de hardwareondersteuning voor vloeiend scrollen zich niet uitstrekt tot het op fijne wijze wijzigen van het bronadres van de bitplane. Zelfs met gebruikmaking van de DMAgic vereist het scrollen van een 320×200 256-kleurenscherm in het gunstigste geval 128.000 klokcycli (het lezen en schrijven van 320×200 = 64.000 bytes). Op 3.5 MHz op de C65 zou dit ongeveer 36 milliseconden kosten, oftewel ongeveer 2 volledige videoframes. Voor vloeiend

scrollen van een dergelijk scherm is daarom een dubbelgebufferde regeling nodig, die 128.000 van de 131.072 bytes geheugen zou verbruiken.

De bekende tekenmodi van de VIC-II worden daarentegen veelvuldig in spellen gebruikt, omdat ze met een geringe hoeveelheid schermgeheugen kunnen bepalen welk blok van 8×8 pixels wordt weergegeven, wat zeer snel scrollen, minder geheugengebruik en effectieve hardwareversnelling van animatie van veelgebruikte elementen mogelijk maakt. De verbeteringen in de VIC-IV zijn dan ook vooral op de tekenmodus gericht. Omdat de bitmapmodus op de VIC-II feitelijk een speciaal geval van de tekenmodus is, met impliciete tekennummers, wordt deze op de VIC-IV als het ware gratis meegeleverd en zal alleen in het kader van een zeer klein aantal bitmapmodus-specifieke verbeteringen worden genoemd die triviaal te implementeren waren en waarvan het dus onverstandig leek ze niet te realiseren, voor het geval ze toch van pas komen.

Weergeven van meer dan 256 unieke tekens via "Super-Uitgebreide Attribuutmodus"

De belangrijkste innovatie is de toevoeging van de Super-Uitgebreide Attribuutmodus (SEAM). De VIC-II gebruikt al 12 bits per teken: elke cel van 8×8 pixels wordt gedefinieerd door 12 bits gegevens: 8 bits schermgeheugengegevens, standaard van $0400 - $07E7 (1024 - 2023 decimaal), die aangeven welke tekens moeten worden weergegeven, en 4 bits kleurgegevens van het 1 KB nibble kleurengeheugen op $D800 - $DBFF (55296 - 56319 decimaal). De VIC-III van de C65 gebruikt 16 bits, omdat het kleurengeheugen nu 8 bits telt in plaats van 4, waarbij de extra 4 bits kleurengeheugen worden gebruikt om attributen te ondersteunen (knipperen, vet, onderstrepen en inverse weergave). Het wordt aanbevolen om de werking hiervan door te nemen voordat u verder leest. Een goede inleiding op de tekstmodus van de VIC-II is op veel plaatsen te vinden. Super-Uitgebreide Attribuutmodus verdubbelt het aantal bits per teken van de 16 van de VIC-III tot 32: twee bytes schermgeheugen en twee bytes kleuren-/attribuutgeheugen.

Super-Uitgebreide Attribuutmodus wordt ingeschakeld door bit 0 in $D054 (53332 decimaal) te zetten. Vergeet niet eerst de VIC-IV-modus in te schakelen om dit register toegankelijk te maken. Wanneer deze bit is gezet, worden twee bytes gebruikt voor zowel het schermgeheugen als het kleurengeheugen voor elk teken dat op het scherm wordt weergegeven. In vergelijking met de 12 bits informatie die de C64 per teken gebruikt en de 16 bits van de VIC-III, heeft de VIC-IV 32 bits informatie. Hoe deze 32 bits worden gebruikt, varieert enigszins per modus, zoals beschreven in de volgende tabellen, inclusief of de GOTOX-bit is gezet.

Let er ook op dat het tegelijkertijd inschakelen van de BOLD- en REVERSE-attributen op de MEGA65 een alternatief palet selecteert, waardoor feitelijk 512 kleuren op het scherm mogelijk zijn, maar elk 8×8-teken slechts kleuren uit één palet van 256 kleuren kan gebruiken.

De twee tabellen die volgen beschrijven de bitvelden die voor dit teken beschikbaar zijn, die variëren afhankelijk van of de **GOTOX_flag** is gezet of gewist:

- **GOTOX_flag = 0** creëert een zichtbaar teken (blauwe bitvelden) met attributen die dit teken beïnvloeden.
- **GOTOX_flag = 1** creëert een onzichtbaar teken (rode bitvelden) met attributen die volgende tekens beïnvloeden.

Bitvelden indien GOTOX_flag is gewist:

Bit(s)	Functie wanneer GOTOX_flag is gewist
Schermgeheugen-byte 0	
Bits 7 - 0	**char_number (lo)**: (bits 7 - 0) Laagste byte van tekennummer (hetzelfde als de VIC-II en VIC-III)
Schermgeheugen-byte 1	
Bits 7 - 5	**rhs_trim (lo)**: (bits 2 - 0) Pixels aan de rechterkant van het teken wegsnijden)
Bits 4 - 0	**char_number (hi)**: (bits 12 - 8) Bovenste 5 bits van tekennummer, waardoor 8.192 unieke tekens kunnen worden aangesproken
Kleurengeheugen-byte 0	
Bit 7	**vert_flip**: het teken verticaal spiegelen
Bit 6	**horz_flip**: het teken horizontaal spiegelen
Bit 5	**alpha_mode**: de alpha-blendmodus inschakelen. De pixelwaarden worden dan als alpha-waarden behandeld die tussen de voorgrondkleur en de achtergrondkleur mengen (hiervoor moet $D054.7 - ALPHEN worden ingesteld)
Bit 4	**GOTOX_flag** is **gewist** **(op 0 gesteld)** Verschaft de blauwe bitvelden die in deze tabel worden getoond. Dit zijn attributen die invloed hebben op dit teken.
Bit 3	**ncm_flag**: • 1 = NCM-teken: gebruikt 4 bits per pixel en is 16 pixels breed (minus eventuele trim-bits aan de rechterkant) • 0 = FCM-teken: gebruikt 8 bits per pixel en is 8 pixels breed (minus eventuele trim-bits aan de rechterkant)
Bit 2	**rhs_trim (hi)**: (bit 3) pixels aan de rechterkant van het teken wegsnijden
Bits 1 - 0	GERESERVEERD, op 0 gesteld
Kleurengeheugen-byte 1	
Indien de VIC-II-multicolourmodus is ingeschakeld: ($D016.4 - MCM)	
Bits 7 - 4	**colour (hi)**: bovenste 4 bits van de voorgrondkleur van het teken
Indien de uitgebreide attributen van de VIC-III zijn ingeschakeld: ($D031.5 - ATTR)	
Bit 7	**underline**: Hardwarematige onderstreping van het teken
Bit 6	**bold**: Vetattribuut van het teken (hardware) *
Bit 5	**reverse**: Omgekeerde weergave van het teken (hardware) *
Bit 4	**blink**: Knipperen van het teken (hardware)
Het overgebleven bitveld is gemeenschappelijk:	
Bits 3 - 0	**colour (lo)**: onderste 4 bits van de voorgrondkleur van het teken

Bitvelden indien GOTOX_flag is ingesteld:

Bit(s)	Functie wanneer GOTOX_flag is ingesteld
Schermgeheugen-byte 0	
Bits 7 - 0	**gotox_pos (lo)**: (bits 7 - 0) onderste 8 bits van de nieuwe X-positie waar volgende tekens moeten beginnen met tekenen, ten opzichte van het begin van het tekenen van het teken. Door dit op 0 te stellen wordt het volgende teken over het meest linkse teken heen getekend.
Schermgeheugen-byte 1	
Bits 7 - 5	**fcm_yoffs**: Y-offset van FCM-tekengegevens: wanneer niet nul, worden FCM-tekengegevens opgehaald vanaf het tekenadres + 8 × de Y-waarde, wat equivalent is aan één pixellijn FCM-tekengegevens per Y. Door zorgvuldig plannen kan dit worden gebruikt om meerdere lagen RRB-inhoud soepel verticaal te scrollen.
Bit 4	**fcm_yoffs_dir**: wanneer ingesteld, wordt het **fcm_yoffs**-veld in plaats daarvan in mindering gebracht.
Bits 3 - 2	GERESERVEERD, op 0 ingesteld
Bits 1 - 0	**gotox_pos (hi)**: (bits 9 - 8) bovenste 2 bits van de nieuwe X-positie (het hoogste bit is het tekenbit van het tweecomplement)
Kleurengeheugen-byte 0	
Bit 7	**transparent_flag**: wanneer ingesteld, worden achtergrond-/transparante pixels niet getekend voor volgende tekens, waardoor lagen mogelijk zijn
Bit 6	**background_flag**: wanneer ingesteld, worden volgende tekens als achtergrond gerenderd, waardoor sprites ervoor kunnen verschijnen, zelfs wanneer sprites op achtergrond zijn ingesteld.
Bit 5	GERESERVEERD, op 0 ingesteld
Bit 4	**GOTOX_flag** is **op 1 ingesteld**. Biedt de in deze tabel getoonde rode bitvelden, dit zijn attributen die volgende tekens beïnvloeden. Eén opvallend veld is **gotox_pos**, dat later in de sectie Raster-Rewrite Buffer wordt besproken.)
Bit 3	**rowmask_flag**: wanneer ingesteld, beslist het **rowmask**-veld welke pixellijnen van volgende tekens moeten worden gerenderd. Dit kan worden gebruikt om tekens verticaal te scrollen met RRB, door elk teken op twee rijen te tekenen: verschoven naar beneden op de bovenste rij, en verschoven omhoog op de onderste rij, en gemaskeerd zodat alleen de pixellijnen van het gescrollde teken worden weergegeven, en niet de gegevens van vóór/na de gegevens van dat teken.
Bit 2	**foreground_flag**: wanneer ingesteld, worden volgende tekens als voorgrond getekend, ongeacht hun kleur, waardoor sprites erachter kunnen verschijnen.

vervolg ...

...vervolg

Bit(s)	Functie wanneer GOTOX_flag is ingesteld
Bits 1 - 0	GERESERVEERD, op 0 ingesteld
Kleurengeheugen-byte 1	
Bits 7 - 0	**rowmask**: maskerwaarden voor pixellijnen. Bits 7 - 0 = Rijen 7 - 0. 1 = rij weergeven, 0 = rij verbergen

We zien dat we nog steeds de onderste 8 bits van het tekennummer in de C64-stijl hebben in de eerste schermgeheugen-byte. De tweede byte van het schermgeheugen krijgt daarvoor vijf extra bits, waardoor er 2^{13} = 8.192 verschillende tekens op één scherm kunnen worden gebruikt. Dat is meer dan voldoende voor unieke tekens die een 80×50-scherm bestrijken (wat mogelijk is om te maken met de VIC-IV). De resterende bits maken het bijsnijden van het teken mogelijk. Hierdoor kunnen tekens met variabele breedte worden gebruikt, waarmee dingen gedaan kunnen worden die normaal niet mogelijk zouden zijn, zoals het gebruik van tekstmodus voor vrije horizontale plaatsing van tekens (of delen daarvan). Dit werd oorspronkelijk toegevoegd om hardwarematige ondersteuning voor proportionele breedte-lettertypen mogelijk te maken.

Voor het kleurengeheugen is de tweede byte (byte 1) gelijk aan die van de C65, oftewel de onderste helft biedt vier bits voor de voorgrondkleur, zoals op de C64, plus de optionele VIC-III uitgebreide attributen. Het C65-specificatiedocument beschrijft het gedrag wanneer meer dan één van deze tegelijk wordt gebruikt, waarvan de meeste logisch zijn, maar er zijn een paar combinaties die zich anders gedragen dan verwacht. Het combineren van vetgedrukt met knipperen zorgt er bijvoorbeeld voor dat het teken schakelt tussen de vetgedrukte en de normale modus. Vetgedrukte modus zelf wordt geïmplementeerd door effectief te fungeren als bit 4 van de voorgrondkleurwaarde, waardoor de kleur van andere paletposities wordt getekend dan normaal.

Indien u de VIC-III uitgebreide attributen niet nodig heeft, kunt u in plaats daarvan de bovenste vier bits van de tweede byte van het kleurengeheugen gebruiken om meer bits van de kleurindex te bevatten, waardoor een selectie uit het volledige bereik van 256 kleurinstellingen mogelijk is. Deze modus wordt geactiveerd door de multicolour-modus van de VIC-II in te schakelen terwijl de volledige kleurmodus actief is.

De C65 / VIC-III-attributen en het gebruik van 256-kleuren 8-bit-waarden voor verschillende VIC-II-kleurregisters wordt ingeschakeld door bit 5 van $D031 (53297 decimaal) in te stellen. Het wordt sterk aanbevolen dit te doen bij het gebruik van de VIC-IV-modus, omdat anders bepaalde functies niet naar behoren werken. Merk op dat BOLD+REVERSE samen de betekenis heeft van het selecteren van een alternatief palet op de MEGA65, wat verschilt van de C65.

Dankzij Super-Uitgebreide Attribuutmodus zijn veel effecten mogelijk. Een paar mogelijkheden worden in de volgende subsecties uitgelegd.

Super-Uitgebreide Attribuutmodus gebruiken

Super-Uitgebreide Attribuutmodus vereist twee keer zoveel schermgeheugen en kleurengeheugen als de VIC-II/III-tekstmodi. Dit komt doordat er twee bytes van elk nodig zijn om elk teken te definiëren, in plaats van één. Het schermgeheugen kan op elke willekeurige locatie in het 384KB hoofdgeheugen worden geplaatst met behulp van de registers $D060 - $D062 (53344 - 53346 decimaal). Het kleurengeheugen kan op elke willekeurige locatie in het 32KB kleurengeheugen worden geplaatst. Alleen de eerste 1 of 2KB van het kleurengeheugen is zichtbaar op $D800 - $DBFF of $D800 - $DFFF (indien het *CRAM2K*-signaal is ingesteld in bit 0 van $D030, 53296 decimaal). Indien een scherm groter dan 40×25 tekens wordt gebruikt, is het gebruik van de DMA-controller of een andere manier vereist om het volledige kleurengeheugen te benaderen. We bespreken daarom eerst het gebruik van Super-Uitgebreide Attribuutmodus met een 40x25-tekenweergave.

De eerste stap is het inschakelen van Super-Uitgebreide Attribuutmodus door de *FCLRHI*- en *CHR16*-signalen te activeren, door bits 2 en 0 van $D054 (53332 decimaal) in te stellen. Aangezien dit een VIC-IV-register is, moet eerst de VIC-IV I/O-modus worden ingeschakeld. De VIC-IV moet ook worden geconfigureerd naar 40-kolomsmodus, door het *H640*-signaal uit te schakelen door bit 7 van $D031 (53297 decimaal) te wissen. Dit komt doordat elk tekenpaar samen één teken op het scherm vormt, waarbij één teken twee schermgeheugenbytes vereist, waardoor er 80 schermgeheugenbytes nodig zijn om 40 tekens weer te geven. Eveneens zijn er 80 kleurengeheugenbytes nodig.

Om dit visueel te begrijpen, is het nuttig eerst de normale C64-schermgeheugenindeling te bekijken:

$400	$401	$402	$403	$404	$405	$406	$407	$408	$409	$40a	$40b	$40c	$40d	$40e	$40f	$410	$411	$412	$413	$414	$415	$416	$417	$418	$419	$41a	$41b	$41c	$41d	$41e	$41f	$420	$421	$422	$423	$424	$425	$426	$427
$428	$429	$42a	$42b	$42c	$42d	$42e	$42f	$430	$431	$432	$433	$434	$435	$436	$437	$438	$439	$43a	$43b	$43c	$43d	$43e	$43f	$440	$441	$442	$443	$444	$445	$446	$447	$448	$449	$44a	$44b	$44c	$44d	$44e	$44f
$450	$451	$452	$453	$454	$455	$456	$457	$458	$459	$45a	$45b	$45c	$45d	$45e	$45f	$460	$461	$462	$463	$464	$465	$466	$467	$468	$469	$46a	$46b	$46c	$46d	$46e	$46f	$470	$471	$472	$473	$474	$475	$476	$477
$478	$479	$47a	$47b	$47c	$47d	$47e	$47f	$480	$481	$482	$483	$484	$485	$486	$487	$488	$489	$48a	$48b	$48c	$48d	$48e	$48f	$490	$491	$492	$493	$494	$495	$496	$497	$498	$499	$49a	$49b	$49c	$49d	$49e	$49f
$4a0	$4a1	$4a2	$4a3	$4a4	$4a5	$4a6	$4a7	$4a8	$4a9	$4aa	$4ab	$4ac	$4ad	$4ae	$4af	$4b0	$4b1	$4b2	$4b3	$4b4	$4b5	$4b6	$4b7	$4b8	$4b9	$4ba	$4bb	$4bc	$4bd	$4be	$4bf	$4c0	$4c1	$4c2	$4c3	$4c4	$4c5	$4c6	$4c7
$4c8	$4c9	$4ca	$4cb	$4cc	$4cd	$4ce	$4cf	$4d0	$4d1	$4d2	$4d3	$4d4	$4d5	$4d6	$4d7	$4d8	$4d9	$4da	$4db	$4dc	$4dd	$4de	$4df	$4e0	$4e1	$4e2	$4e3	$4e4	$4e5	$4e6	$4e7	$4e8	$4e9	$4ea	$4eb	$4ec	$4ed	$4ee	$4ef
$4f0	$4f1	$4f2	$4f3	$4f4	$4f5	$4f6	$4f7	$4f8	$4f9	$4fa	$4fb	$4fc	$4fd	$4fe	$4ff	$500	$501	$502	$503	$504	$505	$506	$507	$508	$509	$50a	$50b	$50c	$50d	$50e	$50f	$510	$511	$512	$513	$514	$515	$516	$517
$518	$519	$51a	$51b	$51c	$51d	$51e	$51f	$520	$521	$522	$523	$524	$525	$526	$527	$528	$529	$52a	$52b	$52c	$52d	$52e	$52f	$530	$531	$532	$533	$534	$535	$536	$537	$538	$539	$53a	$53b	$53c	$53d	$53e	$53f
$540	$541	$542	$543	$544	$545	$546	$547	$548	$549	$54a	$54b	$54c	$54d	$54e	$54f	$550	$551	$552	$553	$554	$555	$556	$557	$558	$559	$55a	$55b	$55c	$55d	$55e	$55f	$560	$561	$562	$563	$564	$565	$566	$567
$568	$569	$56a	$56b	$56c	$56d	$56e	$56f	$570	$571	$572	$573	$574	$575	$576	$577	$578	$579	$57a	$57b	$57c	$57d	$57e	$57f	$580	$581	$582	$583	$584	$585	$586	$587	$588	$589	$58a	$58b	$58c	$58d	$58e	$58f
$590	$591	$592	$593	$594	$595	$596	$597	$598	$599	$59a	$59b	$59c	$59d	$59e	$59f	$5a0	$5a1	$5a2	$5a3	$5a4	$5a5	$5a6	$5a7	$5a8	$5a9	$5aa	$5ab	$5ac	$5ad	$5ae	$5af	$5b0	$5b1	$5b2	$5b3	$5b4	$5b5	$5b6	$5b7
$5b8	$5b9	$5ba	$5bb	$5bc	$5bd	$5be	$5bf	$5c0	$5c1	$5c2	$5c3	$5c4	$5c5	$5c6	$5c7	$5c8	$5c9	$5ca	$5cb	$5cc	$5cd	$5ce	$5cf	$5d0	$5d1	$5d2	$5d3	$5d4	$5d5	$5d6	$5d7	$5d8	$5d9	$5da	$5db	$5dc	$5dd	$5de	$5df
$5e0	$5e1	$5e2	$5e3	$5e4	$5e5	$5e6	$5e7	$5e8	$5e9	$5ea	$5eb	$5ec	$5ed	$5ee	$5ef	$5f0	$5f1	$5f2	$5f3	$5f4	$5f5	$5f6	$5f7	$5f8	$5f9	$5fa	$5fb	$5fc	$5fd	$5fe	$5ff	$600	$601	$602	$603	$604	$605	$606	$607
$608	$609	$60a	$60b	$60c	$60d	$60e	$60f	$610	$611	$612	$613	$614	$615	$616	$617	$618	$619	$61a	$61b	$61c	$61d	$61e	$61f	$620	$621	$622	$623	$624	$625	$626	$627	$628	$629	$62a	$62b	$62c	$62d	$62e	$62f
$630	$631	$632	$633	$634	$635	$636	$637	$638	$639	$63a	$63b	$63c	$63d	$63e	$63f	$640	$641	$642	$643	$644	$645	$646	$647	$648	$649	$64a	$64b	$64c	$64d	$64e	$64f	$650	$651	$652	$653	$654	$655	$656	$657
$658	$659	$65a	$65b	$65c	$65d	$65e	$65f	$660	$661	$662	$663	$664	$665	$666	$667	$668	$669	$66a	$66b	$66c	$66d	$66e	$66f	$670	$671	$672	$673	$674	$675	$676	$677	$678	$679	$67a	$67b	$67c	$67d	$67e	$67f
$680	$681	$682	$683	$684	$685	$686	$687	$688	$689	$68a	$68b	$68c	$68d	$68e	$68f	$690	$691	$692	$693	$694	$695	$696	$697	$698	$699	$69a	$69b	$69c	$69d	$69e	$69f	$6a0	$6a1	$6a2	$6a3	$6a4	$6a5	$6a6	$6a7
$6a8	$6a9	$6aa	$6ab	$6ac	$6ad	$6ae	$6af	$6b0	$6b1	$6b2	$6b3	$6b4	$6b5	$6b6	$6b7	$6b8	$6b9	$6ba	$6bb	$6bc	$6bd	$6be	$6bf	$6c0	$6c1	$6c2	$6c3	$6c4	$6c5	$6c6	$6c7	$6c8	$6c9	$6ca	$6cb	$6cc	$6cd	$6ce	$6cf
$6d0	$6d1	$6d2	$6d3	$6d4	$6d5	$6d6	$6d7	$6d8	$6d9	$6da	$6db	$6dc	$6dd	$6de	$6df	$6e0	$6e1	$6e2	$6e3	$6e4	$6e5	$6e6	$6e7	$6e8	$6e9	$6ea	$6eb	$6ec	$6ed	$6ee	$6ef	$6f0	$6f1	$6f2	$6f3	$6f4	$6f5	$6f6	$6f7
$6f8	$6f9	$6fa	$6fb	$6fc	$6fd	$6fe	$6ff	$700	$701	$702	$703	$704	$705	$706	$707	$708	$709	$70a	$70b	$70c	$70d	$70e	$70f	$710	$711	$712	$713	$714	$715	$716	$717	$718	$719	$71a	$71b	$71c	$71d	$71e	$71f
$720	$721	$722	$723	$724	$725	$726	$727	$728	$729	$72a	$72b	$72c	$72d	$72e	$72f	$730	$731	$732	$733	$734	$735	$736	$737	$738	$739	$73a	$73b	$73c	$73d	$73e	$73f	$740	$741	$742	$743	$744	$745	$746	$747
$748	$749	$74a	$74b	$74c	$74d	$74e	$74f	$750	$751	$752	$753	$754	$755	$756	$757	$758	$759	$75a	$75b	$75c	$75d	$75e	$75f	$760	$761	$762	$763	$764	$765	$766	$767	$768	$769	$76a	$76b	$76c	$76d	$76e	$76f
$770	$771	$772	$773	$774	$775	$776	$777	$778	$779	$77a	$77b	$77c	$77d	$77e	$77f	$780	$781	$782	$783	$784	$785	$786	$787	$788	$789	$78a	$78b	$78c	$78d	$78e	$78f	$790	$791	$792	$793	$794	$795	$796	$797
$798	$799	$79a	$79b	$79c	$79d	$79e	$79f	$7a0	$7a1	$7a2	$7a3	$7a4	$7a5	$7a6	$7a7	$7a8	$7a9	$7aa	$7ab	$7ac	$7ad	$7ae	$7af	$7b0	$7b1	$7b2	$7b3	$7b4	$7b5	$7b6	$7b7	$7b8	$7b9	$7ba	$7bb	$7bc	$7bd	$7be	$7bf
$7c0	$7c1	$7c2	$7c3	$7c4	$7c5	$7c6	$7c7	$7c8	$7c9	$7ca	$7cb	$7cc	$7cd	$7ce	$7cf	$7d0	$7d1	$7d2	$7d3	$7d4	$7d5	$7d6	$7d7	$7d8	$7d9	$7da	$7db	$7dc	$7dd	$7de	$7df	$7e0	$7e1	$7e2	$7e3	$7e4	$7e5	$7e6	$7e7

Dat wil zeggen: elk tekencel gebruikt één byte van het schermgeheugen en de adressen nemen geleidelijk toe, zowel binnen regels als tussen regels. Super-Uitgebreide Attribuutmodus vereist twee bytes per tekencel. Indien u $D054 bijvoorbeeld op $05 instelt, krijgt u schermadressen als volgt:

| | | |

$400	$402	$404	$406	$408	$40a	$40c	$40e	$410	$412	$414	$416	$418	$41a	$41c	$41e	$420	$422	$424	$426	$428	$42a	$42c	$42e	$430	$432	$434	$436	$438	$43a	$43c	$43e	$440	$442	$444	$446	$448	$44a	$44c	$44e
$450	$452	$454	$456	$458	$45a	$45c	$45e	$460	$462	$464	$466	$468	$46a	$46c	$46e	$470	$472	$474	$476	$478	$47a	$47c	$47e	$480	$482	$484	$486	$488	$48a	$48c	$48e	$490	$492	$494	$496	$498	$49a	$49c	$49e
$4a0	$4a2	$4a4	$4a6	$4a8	$4aa	$4ac	$4ae	$4b0	$4b2	$4b4	$4b6	$4b8	$4ba	$4bc	$4be	$4c0	$4c2	$4c4	$4c6	$4c8	$4ca	$4cc	$4ce	$4d0	$4d2	$4d4	$4d6	$4d8	$4da	$4dc	$4de	$4e0	$4e2	$4e4	$4e6	$4e8	$4ea	$4ec	$4ee
$4f0	$4f2	$4f4	$4f6	$4f8	$4fa	$4fc	$4fe	$500	$502	$504	$506	$508	$50a	$50c	$50e	$510	$512	$514	$516	$518	$51a	$51c	$51e	$520	$522	$524	$526	$528	$52a	$52c	$52e	$530	$532	$534	$536	$538	$53a	$53c	$53e
$540	$542	$544	$546	$548	$54a	$54c	$54e	$550	$552	$554	$556	$558	$55a	$55c	$55e	$560	$562	$564	$566	$568	$56a	$56c	$56e	$570	$572	$574	$576	$578	$57a	$57c	$57e	$580	$582	$584	$586	$588	$58a	$58c	$58e
$590	$592	$594	$596	$598	$59a	$59c	$59e	$5a0	$5a2	$5a4	$5a6	$5a8	$5aa	$5ac	$5ae	$5b0	$5b2	$5b4	$5b6	$5b8	$5ba	$5bc	$5be	$5c0	$5c2	$5c4	$5c6	$5c8	$5ca	$5cc	$5ce	$5d0	$5d2	$5d4	$5d6	$5d8	$5da	$5dc	$5de
$5e0	$5e2	$5e4	$5e6	$5e8	$5ea	$5ec	$5ee	$5f0	$5f2	$5f4	$5f6	$5f8	$5fa	$5fc	$5fe	$600	$602	$604	$606	$608	$60a	$60c	$60e	$610	$612	$614	$616	$618	$61a	$61c	$61e	$620	$622	$624	$626	$628	$62a	$62c	$62e
$630	$632	$634	$636	$638	$63a	$63c	$63e	$640	$642	$644	$646	$648	$64a	$64c	$64e	$650	$652	$654	$656	$658	$65a	$65c	$65e	$660	$662	$664	$666	$668	$66a	$66c	$66e	$670	$672	$674	$676	$678	$67a	$67c	$67e
$680	$682	$684	$686	$688	$68a	$68c	$68e	$690	$692	$694	$696	$698	$69a	$69c	$69e	$6a0	$6a2	$6a4	$6a6	$6a8	$6aa	$6ac	$6ae	$6b0	$6b2	$6b4	$6b6	$6b8	$6ba	$6bc	$6be	$6c0	$6c2	$6c4	$6c6	$6c8	$6ca	$6cc	$6ce
$6d0	$6d2	$6d4	$6d6	$6d8	$6da	$6dc	$6de	$6e0	$6e2	$6e4	$6e6	$6e8	$6ea	$6ec	$6ee	$6f0	$6f2	$6f4	$6f6	$6f8	$6fa	$6fc	$6fe	$700	$702	$704	$706	$708	$70a	$70c	$70e	$710	$712	$714	$716	$718	$71a	$71c	$71e
$720	$722	$724	$726	$728	$72a	$72c	$72e	$730	$732	$734	$736	$738	$73a	$73c	$73e	$740	$742	$744	$746	$748	$74a	$74c	$74e	$750	$752	$754	$756	$758	$75a	$75c	$75e	$760	$762	$764	$766	$768	$76a	$76c	$76e
$770	$772	$774	$776	$778	$77a	$77c	$77e	$780	$782	$784	$786	$788	$78a	$78c	$78e	$790	$792	$794	$796	$798	$79a	$79c	$79e	$7a0	$7a2	$7a4	$7a6	$7a8	$7aa	$7ac	$7ae	$7b0	$7b2	$7b4	$7b6	$7b8	$7ba	$7bc	$7be
$7c0	$7c2	$7c4	$7c6	$7c8	$7ca	$7cc	$7ce	$7d0	$7d2	$7d4	$7d6	$7d8	$7da	$7dc	$7de	$7e0	$7e2	$7e4	$7e6	$7e8	$7ea	$7ec	$7ee	$7f0	$7f2	$7f4	$7f6	$7f8	$7fa	$7fc	$7fe	$800	$802	$804	$806	$808	$80a	$80c	$80e
$810	$812	$814	$816	$818	$81a	$81c	$81e	$820	$822	$824	$826	$828	$82a	$82c	$82e	$830	$832	$834	$836	$838	$83a	$83c	$83e	$840	$842	$844	$846	$848	$84a	$84c	$84e	$850	$852	$854	$856	$858	$85a	$85c	$85e
$860	$862	$864	$866	$868	$86a	$86c	$86e	$870	$872	$874	$876	$878	$87a	$87c	$87e	$880	$882	$884	$886	$888	$88a	$88c	$88e	$890	$892	$894	$896	$898	$89a	$89c	$89e	$8a0	$8a2	$8a4	$8a6	$8a8	$8aa	$8ac	$8ae
$8b0	$8b2	$8b4	$8b6	$8b8	$8ba	$8bc	$8be	$8c0	$8c2	$8c4	$8c6	$8c8	$8ca	$8cc	$8ce	$8d0	$8d2	$8d4	$8d6	$8d8	$8da	$8dc	$8de	$8e0	$8e2	$8e4	$8e6	$8e8	$8ea	$8ec	$8ee	$8f0	$8f2	$8f4	$8f6	$8f8	$8fa	$8fc	$8fe
$900	$902	$904	$906	$908	$90a	$90c	$90e	$910	$912	$914	$916	$918	$91a	$91c	$91e	$920	$922	$924	$926	$928	$92a	$92c	$92e	$930	$932	$934	$936	$938	$93a	$93c	$93e	$940	$942	$944	$946	$948	$94a	$94c	$94e
$950	$952	$954	$956	$958	$95a	$95c	$95e	$960	$962	$964	$966	$968	$96a	$96c	$96e	$970	$972	$974	$976	$978	$97a	$97c	$97e	$980	$982	$984	$986	$988	$98a	$98c	$98e	$990	$992	$994	$996	$998	$99a	$99c	$99e
$9a0	$9a2	$9a4	$9a6	$9a8	$9aa	$9ac	$9ae	$9b0	$9b2	$9b4	$9b6	$9b8	$9ba	$9bc	$9be	$9c0	$9c2	$9c4	$9c6	$9c8	$9ca	$9cc	$9ce	$9d0	$9d2	$9d4	$9d6	$9d8	$9da	$9dc	$9de	$9e0	$9e2	$9e4	$9e6	$9e8	$9ea	$9ec	$9ee
$9f0	$9f2	$9f4	$9f6	$9f8	$9fa	$9fc	$9fe	$a00	$a02	$a04	$a06	$a08	$a0a	$a0c	$a0e	$a10	$a12	$a14	$a16	$a18	$a1a	$a1c	$a1e	$a20	$a22	$a24	$a26	$a28	$a2a	$a2c	$a2e	$a30	$a32	$a34	$a36	$a38	$a3a	$a3c	$a3e
$a40	$a42	$a44	$a46	$a48	$a4a	$a4c	$a4e	$a50	$a52	$a54	$a56	$a58	$a5a	$a5c	$a5e	$a60	$a62	$a64	$a66	$a68	$a6a	$a6c	$a6e	$a70	$a72	$a74	$a76	$a78	$a7a	$a7c	$a7e	$a80	$a82	$a84	$a86	$a88	$a8a	$a8c	$a8e
$a90	$a92	$a94	$a96	$a98	$a9a	$a9c	$a9e	$aa0	$aa2	$aa4	$aa6	$aa8	$aaa	$aac	$aae	$ab0	$ab2	$ab4	$ab6	$ab8	$aba	$abc	$abe	$ac0	$ac2	$ac4	$ac6	$ac8	$aca	$acc	$ace	$ad0	$ad2	$ad4	$ad6	$ad8	$ada	$adc	$ade
$ae0	$ae2	$ae4	$ae6	$ae8	$aea	$aec	$aee	$af0	$af2	$af4	$af6	$af8	$afa	$afc	$afe	$b00	$b02	$b04	$b06	$b08	$b0a	$b0c	$b0e	$b10	$b12	$b14	$b16	$b18	$b1a	$b1c	$b1e	$b20	$b22	$b24	$b26	$b28	$b2a	$b2c	$b2e
$b30	$b32	$b34	$b36	$b38	$b3a	$b3c	$b3e	$b40	$b42	$b44	$b46	$b48	$b4a	$b4c	$b4e	$b50	$b52	$b54	$b56	$b58	$b5a	$b5c	$b5e	$b60	$b62	$b64	$b66	$b68	$b6a	$b6c	$b6e	$b70	$b72	$b74	$b76	$b78	$b7a	$b7c	$b7e
$b80	$b82	$b84	$b86	$b88	$b8a	$b8c	$b8e	$b90	$b92	$b94	$b96	$b98	$b9a	$b9c	$b9e	$ba0	$ba2	$ba4	$ba6	$ba8	$baa	$bac	$bae	$bb0	$bb2	$bb4	$bb6	$bb8	$bba	$bbc	$bbe	$bc0	$bc2	$bc4	$bc6	$bc8	$bca	$bcc	$bce

Het is mogelijk om Super-Uitgebreide Attribuutmodus te gebruiken vanuit C65-modus, door het scherm op 80 kolommen in te stellen, omdat de C65-ROM 2 KB reserveert voor zowel het schermgeheugen als het kleurengeheugen, en automatisch $D058 en $D059 op de juiste waarde zet voor $40 \times 2 = 80$ bytes per schermregel. De gebruiker hoeft elk tekenpaar slechts als één Super-Uitgebreid Attribuutteken te behandelen en Super-Uitgebreide Attribuutmodus in te schakelen, zoals hierboven beschreven.

Omdat er paren van kleurengeheugen- en schermgeheugenbytes worden gebruikt om elk teken te definiëren, moet voorzichtig te werk worden gegaan bij het initialiseren en bewerken van het scherm. Een goede aanpak is om de tekstkleur op zwart in te stellen (kleurcode 0) en vervolgens het scherm te vullen met @-tekens, omdat dat tekencode 0 is. Vervolgens zijn er verschillende manieren om het scherm te bewerken. U kunt het normale PRINT-commando gebruiken en zorgvuldig tekenreeksen samenstellen die de juiste waarden in elk scherm- en kleurenbytepaar schrijven. Een andere aanpak is om de BANK- en POKE-commando's te gebruiken om de inhoud van het schermgeheugen en kleurengeheugen direct te instellen.

Het beheren van een scherm in Super-Uitgebreide Attribuutmodus op deze manier met BASIC 65 is natuurlijk eerder een noodoplossing en wordt alleen voorgesteld als een relatief eenvoudige manier om te beginnen met experimenteren. U wilt bijna zeker snel overschakelen op aangepaste schermverwerkingscode, het liefst in assembleertaal, om schermen in Super-Uitgebreide Attribuutmodus te bewerken. De aanpak met BASIC 65 kan echter behoorlijk krachtig zijn, omdat het bestaande scrollen van het scherm en andere bewerkingen beschikbaar blijven.

XXX Voorbeeldprogramma

De volgende beschrijvingen gaan ervan uit dat u een van de hierboven beschreven methoden heeft geïmplementeerd om het schermgeheugen en kleurengeheugen in te stellen.

Volkleur-tekstmodus (256 kleuren per teken) (FCM)

In de normale VIC-II/III-tekstmodus wordt één byte gebruikt voor elke pixelrij in een teken. Ter herinnering aan de werking van die modes: in hoge-resolutiemodus is elke pixel ofwel de achtergrond- ofwel de voorgrondkleur, gebaseerd op één bit in de byte. Kleurenmodus gebruikt twee bits om tussen vier mogelijke kleuren te kiezen, maar omdat er nog steeds slechts 8 bits beschikbaar zijn om elke rij van 8 pixels te beschrijven, heeft elk pixelpaar dezelfde kleur. De volkleur-tekstmodus van de VIC-IV schaft deze beperkingen op en maakt het mogelijk om elke pixel van een teken te kiezen uit 256 kleuren van de primaire of alternatieve paletbank, zonder horizontale resolutie op te offeren.

Hiertoe heeft elk teken nu 64 bytes aan gegevens nodig. Het adres van de gegevens is 64 $\times$ het tekennummer, ongeacht het adres van de tekenset. FCM hoort normaal gesproken te worden gebruikt met Super-Uitgebreide Attribuutmodus (SEAM), zodat meer dan 256 unieke tekens kunnen worden aangesproken. Omdat SEAM de keuze van 8.192 unieke tekens mogelijk maakt, kunnen FCM-tekengegevens overal in de eerste 512 KB chip-RAM worden geplaatst (let er echter op dat de meeste MEGA65-modellen slechts 384 KB chip-RAM hebben).

Let erop dat pixelwaarde $ff niet direct de bijbehorende kleurcode selecteert. In plaats daarvan selecteert deze de kleurcode die is gedefinieerd door het kleurengeheugen.

Nibble-kleuren-tekstmodus (16 kleuren per teken) (NCM)

De Nibble-kleuren-modus (NCM) voor tekst lijkt op Volkleur-tekstmodus, met als verschil dat elke byte gegevens twee pixels beschrijft met elk 4 bits. Dit maakt NCM uniek, omdat de tekens 16 pixels breed zullen zijn in plaats van de gebruikelijke 8 pixels. Dit kan worden gebruikt om kleurrijke weergaven te maken, zonder zoveel geheugen te gebruiken als FCM, omdat er minder tekens nodig zijn om het scherm te bedekken. In tegenstelling tot de multicolourmodus (MCM) van de VIC-II heeft deze modus geen verlies van horizontale resolutie tot gevolg.

Bij NCM komen de onderste vier bits van de pixelkleur van de bovenste of onderste vier bits van de pixelgegevens. De bovenste vier bits van de kleurcode komen van de kleurengeheugengegevens van het weergegeven teken. Hierdoor is het mogelijk om alle paletingangen in NCM te gebruiken, hoewel de beperking van 16 kleuren per teken blijft bestaan. Vergelijkbaar met het gedrag van FCM selecteert pixelgegevens-waarde $f de pixelkleur die is ingesteld in het kleurengeheugen.

Een verder voordeel van NCM is dat het minder buscycli per pixel gebruikt dan FCM, omdat er minder tekengegevens-ophaalbewerkingen per rasterlijn hoeven plaats te vinden. Samen met de verminderde geheugenbehoeften maakt dit NCM bijzonder

geschikt voor het maken van kleurrijke grafische lagen. Hiermee kan de VIC-IV arcade-achtige weergaven met meer kleuren tonen dan bij veel 16-bits computers.

XXX

Alphablending / Anti-aliasing

De VIC-IV ondersteunt het mengen van tekens met de achtergrondkleur, waardoor effecten zoals anti-aliased lettertype-weergave mogelijk zijn. Het mengen is mogelijk op per-teken-basis. Het wordt ingeschakeld voor een specifiek teken als aan de volgende voorwaarden wordt voldaan:

1. Het *ALPHAEN*-signaal is ingesteld in bit 7 van $D054.
2. Het *CHR16*-signaal is ingesteld in bit 0 van $D054 om Super-Uitgebreide Attributmodus (SEAM) in te schakelen.
3. Volkleur-tekstmodus (FCM) is ingeschakeld voor het teken.

Indien alphablending is ingeschakeld voor een teken, worden de 8-bit pixelwaarden behandeld als alphawaarden in plaats van paletindices. De werkelijke pixelkleur wordt bepaald door de achtergrondkleur te mengen met de voorgrondkleur van het teken zoals gedefinieerd in het kleurengeheugen. Een alphawaarde van 0 staat voor volledige transparantie, waarbij alleen de achtergrondkleur van die pixel wordt weergegeven.

Merk op dat de alphablending alleen wordt toegepast tussen de achtergrondkleur en de voorgrondkleur van het teken. Dit betekent dat tekens achter het huidige teken feitelijk niet zichtbaar zullen zijn (tekenlagen kunnen worden samengesteld via GOTOX-herpositionering). Programmeurs moeten er echter van uitgaan dat het mengen met eerdere lagen in een toekomstige implementatie zal worden ondersteund. Om problemen met een dergelijke wijziging te voorkomen, plaatst u geen tekens achter een teken waarvoor alphablending is ingeschakeld.

Tekens omdraaien

XXX

Variabele-breedte lettertypen

Er zijn 4 bits waarmee pixels van de rechterrand van tekens kunnen worden weggesneden wanneer deze worden weergegeven. Dit heeft het effect dat tekens smaller worden. Dit kan nuttig zijn voor aantrekkelijkere tekstweergave, waarbij smallere tekens zoals "i" minder ruimte innemen dan bredere tekens zoals "m", zonder dat een bitmap-display nodig is. Deze functie kan worden gebruikt om variabele-breedte tekstweergave zeer efficiënt te maken, zowel wat betreft geheugengebruik als verwerkingstijd.

Deze functie kan worden gecombineerd met de volkleur-tekstmodus, de alfablendmodus en de 4-bits-per-pixel-modus om tekens mogelijk te maken met 15 intensiteitsniveaus tussen de achtergrond- en voorgrondkleur, en tot 16 pixels breed. Bovendien kan het GOTO-bit worden gebruikt om negatieve kerning toe te passen,

zodat tekenparen zoals A en T geen overbodige witruimte tussen zich hebben wanneer ze naast elkaar worden afgedrukt. Het zorgvuldig gebruik van deze functies kan leiden tot indrukwekkende tekstweergave, vergelijkbaar met die op moderne 32-bits en 64-bits systemen, maar die nog steeds efficiënt genoeg is om te worden geïmplementeerd op een relatief beperkt systeem zoals de MEGA65. De presentatiesoftware "MegaWAT!?"" voor de MEGA65 maakt gebruik van verschillende van deze functies om zijn aantrekkelijke, antialiased proportionele tekstweergave op dia's te produceren.

XXX MEGAWat!?!-screenshot

XXX Voorbeeldprogramma

Raster Re-write Buffer

Als het GOTO-bit is gezet voor een teken in Super-Uitgebreide Attribuutmodus, wordt in plaats van het teken te tekenen de positie op de rasterlijn teruggedraaid (of vooruitgeschoven) naar de pixelpositie die is opgegeven in de onderste 10 bits van de schermgeheugenbytes. Als het vertical flip-bit is gezet, heeft dit de alternatieve betekenis dat de achtergrondkleur niet wordt getekend. Deze combinatie kan worden gebruikt om tekstmateriaal over ander tekstmateriaal te plaatsen, wat een grove aanvulling vormt op de 8 hardware-sprites. De hoeveelheid materiaal is alleen beperkt door de rastertijd van de VIC-IV. Enige experimentatie is vereist om te bepalen hoeveel er in PAL- en NTSC-modus kan worden bereikt.

Als het GOTO-bit is gezet voor een teken en de tekenbreedte-verminderingsbits ook zijn gezet, worden deze geïnterpreteerd als een Y-offset die wordt opgeteld bij het tekengeheugenadres, maar alleen in Volkleurmodus. Instellen van Y=1 zorgt ervoor dat de tekendata worden opgehaald van 8 bytes verder, d.w.z. de eerste rij tekendata komt van het adres waar normaal de tweede rij tekendata zou worden opgehaald. Voor hogere waarden worden de tekendata opgehaald van verderop in de tekens. Door tekens zorgvuldig in het geheugen te rangschikken, kan deze functie worden gebruikt om vrije verticale positionering van soft sprites mogelijk te maken, zonder dat de tekendata hoeven te worden gekopieerd.

Als het GOTO-bit is gezet voor een teken en het Nybl Colour Mode (NCM)-bit ook is gezet, wordt het tweede kleurengeheugenbyte voor dat teken gebruikt om de rijmaskerbits in te stellen. Voor elk bit dat in het rijmasker is gezet, wordt de overeenkomstige rij tekens in de regel niet weergegeven. Dit kan worden gecombineerd met de Y-offsetfunctie om effectief een soepele verticale schuifregeling per teken mogelijk te maken.

De mogelijkheid om meerdere lagen tekst en grafisch materiaal te tekenen is zeer krachtig. Dit kan bijvoorbeeld worden gebruikt om meerdere overlappende lagen van apart schuifbare grafische elementen te bieden. Dit geeft veel van de voordelen van bitplane-gebaseerde playfields op andere computers, zoals de Amiga, maar zonder de nadelen van bitplanes.

Een goede introductie van de Raster Re-write Buffer en het gebruik ervan is te vinden in deze video:

https://www.youtube.com/watch?v=00bm5uBeBos&feature=youtu.be

Een belangrijk aspect van de RRB is dat de VIC-IV alleen de tekendata weergeeft die zich links van, en inclusief, het laatst getekende teken bevindt. Dit betekent dat wanneer u de GOTO-token gebruikt om meerdere lagen grafisch materiaal te overschrijven, u ofwel moet zorgen dat de laatste laag tot aan de rechterrand van het scherm reikt, ofwel een GOTO-token moet toevoegen dat de renderpositie naar de rechterrand van het scherm verplaatst.

XXX Voorbeeldprogramma

SPRITES

VIC-II/III-spritebesturing

De besturing van sprites voor C64 / VIC-II/III-compatibiliteit is ongewijzigd ten opzichte van de C64. De enige praktische verschillen zijn zeer gering. In het bijzonder gebruikt de VIC-IV een ringbuffer voor de gegevens van elke sprite bij het renderen van een rasterlijn. Dit betekent dat een sprite meerdere keren per rasterlijn kan worden weergegeven, wat horizontale multiplexing mogelijk maakt.

Uitgebreide sprite-afbeeldingssets

Op de VIC-II en VIC-III moeten alle sprites hun afbeeldingsdata op een bepaald moment uit één enkel 16KB-geheugengebied halen. Dit beperkt het aantal verschillende sprite-afbeeldingen tot 256, omdat elke sprite-afbeelding 64 bytes in beslag neemt. In de praktijk moet hetzelfde 16KB-geheugengebied ook bitmap-, tekst- of bitplane-data bevatten, wat het aantal sprite-afbeeldingen dat tegelijk kan worden gebruikt aanzienlijk beperkt.

De VIC-IV heft deze beperking op doordat spritedata overal in het geheugen kunnen worden geplaatst, zij het nog steeds op 64-byte-grenzen. Dit wordt gedaan door het SPRPTR16-signaal in te stellen (bit 7, $D06E, decimaal 53358), wat de VIC-IV vertelt dat er twee bytes per spritepointer in plaats van één moeten worden verwacht. Deze adressen zijn dan absolute adressen en negeren de 16KB VIC-II-bankselectielogica. Er zijn dus 16 bytes nodig in plaats van 8 bytes. De lijst met pointers kan ook overal in het geheugen worden geplaatst door de SPRPTRADR ($D06C - $D06D, 53356 - 53357 decimaal) en SPRPTRBNK-signalen (bits 0 - 6, $D06E, 53358 decimaal) in te stellen. Hierdoor kunnen spritedata overal in de eerste 4MB RAM worden geplaatst en de spritepointerlijst overal in de eerste 8MB RAM. Let op dat typische installaties van de VIC-IV slechts 384KB aangesloten RAM hebben, zodat deze beperkingen geen praktisch effect hebben. De bovenste bits van het SPRPTRBNK-signaal moeten echter op nul worden gezet om problemen met voorwaartse compatibiliteit te voorkomen.

Een reden voor ondersteuning van meer sprite-afbeeldingen is dat sprites op de VIC-IV meer dan één 64-byte-afbeeldingsslot kunnen vereisen. Het inschakelen van de modus voor extra-brede sprites betekent bijvoorbeeld dat een sprite $8 \times 21 = 168$ bytes

nodig heeft en dus vier VIC-II-stijl 64-byte-spriteafbeeldingsslots inneemt. Wanneer sprites met variabele hoogte worden gebruikt, kan dit oplopen tot 8×255 = 2.040 bytes per sprite.

Variabele spritegrootte

Sprites kunnen met de VIC-IV een van drie breedtes hebben:

1. Normale VIC-II-breedte (24 pixels breed).
2. Extra Breed, waarbij 64 bits (8 bytes) data per rasterlijn worden gebruikt in plaats van de 24 van de VIC-II. Dit resulteert in sprites die 64 pixels breed zijn, tenzij Volkleur-spritemodus is geselecteerd voor een sprite, in welk geval de sprite 64 bits ÷ 4 bits per pixel = 16 pixels breed zal zijn.
3. Tegelmodus, waarbij de sprite wordt herhaald getekend tot het einde van de rasterlijn.
   Tegelmodus moet normaal gesproken alleen worden gebruikt met extra-brede sprites, omdat het tegelen altijd plaatsvindt met 64 bits spritedata per regel. Het inschakelen van tegelmodus met normale 24 bits brede mono- of multicolour-spritedata tekent 2 en 2/3 rijen spritedata als één regel, zelfs als de sprite niet in extra-brede modus staat, wat tot verwarrende weergave leidt.

Om een sprite 64 pixels breed te maken (of 16 pixels als Volkleur-spritemodus is ingeschakeld), zet u het overeenkomstige bit voor de sprite in het SPRX64EN-register op ($D057, 53335 decimaal). Het inschakelen van Volkleurmodus voor een sprite schakelt impliciet de uitgebreide breedtemodus in en zorgt ervoor dat deze sprites 16 pixels breed zijn.

Op vergelijkbare wijze kunnen sprites diverse hoogtes hebben: sprites zijn ofwel 21 pixels hoog zoals bij de VIC-II, ofwel, als het overeenkomstige bit voor de sprite is ingeschakeld in het SPRHGTEN-signaal ($D055, 53333 decimaal), heeft die sprite de hoogte in pixels die is ingesteld in het SPRHGT-register ($D056, 53334 decimaal), van 0 tot 255. Let op dat alle sprites met SPRHGTEN ingeschakeld dezelfde hoogte delen. Een sprite kan altijd de onderkant van zijn afbeeldingsdata transparant laten.

Om tegelmodus in te schakelen voor een sprite, zet u het bijbehorende bit van SPRTILEN. Voor sprites 0 tot en met 3 zet u bits 4 tot en met 7 van $D04D (53325 decimaal). Voor sprites 4 tot en met 7 zet u bits 4 tot en met 7 van $D04F (55327 decimaal).

Variabele sprite-resolutie

Standaard hebben sprites dezelfde resolutie als op de VIC-II, oftewel elke sprite-pixel is twee fysieke pixels breed en hoog. Het is echter mogelijk om sprites de native resolutie te laten gebruiken, waarbij sprite-pixels één fysiek pixel breed en/of hoog zijn. Dit wordt bereikt door het relevante bit voor de sprite in het SPRENV400-register ($D076, 53366 decimaal) te zetten, zodat de verticale resolutie per sprite wordt vergroot. De horizontale resolutie van alle sprites is ofwel de normale VIC-II-resolutie, of wanneer

het SPR640-signaal is gezet (bit 4 van $D054, 53332 decimaal), hebben sprites dezelfde horizontale resolutie als de fysieke pixels van het beeldscherm.

Sprite-paletbank

De VIC-IV heeft vier paletbanken, in vergelijking met de enkele paletbank van de VIC-III. De VIC-IV maakt het mogelijk om afzonderlijke paletbanken te selecteren voor bitmap-/tekstgraphics en voor sprites. Dit maakt het eenvoudig om zeer kleurrijke beeldschermen te creëren, waarbij sprites andere kleuren hebben dan de rest van het beeldscherm, of om paletanimatie te gebruiken voor interessante visuele effecten in sprites, zonder het palet dat door andere elementen van het beeldscherm wordt gebruikt te verstoren.

De sprite-paletbank wordt geselecteerd door het SPRPALSEL-signaal in bits 2 en 3 van het register $D070 (53360 decimaal) te zetten. Het is mogelijk om deze op dezelfde bank als de bitmap-/tekstweergave in te stellen, of om een andere paletbank te selecteren. De selectie van de paletbank wordt onmiddellijk van kracht. Vergeet niet dat u, om een palet te kunnen wijzigen, het palet ook in het bereik van de paletbankregisters op $D100 - $D3FF moet plaatsen door het MAPEDPAL-signaal in bits 6 en 7 van $D070 te zetten.

Vollekleuren-spritemodus

Naast de monochrome en multicolour-modi ondersteunt de VIC-IV een nieuwe vollekleuren-spritemodus. In deze modus worden vier bits gebruikt om elke sprite-pixel te coderen. In tegenstelling tot de multicolour-modus, waarin paren van bits paren van pixels coderen, behouden de pixels in de vollekleurenmodus hun normale horizontale resolutie. De kleur nul wordt als transparant beschouwd. Indien u zwart wilt gebruiken in een vollekleurensprite, moet u de paletbank die voor sprites is geselecteerd zodanig configureren dat een van de 15 kleuren voor de betreffende sprite de kleur zwart codeert.

De vollekleuren-spritemodus kan per sprite worden geselecteerd door het bijbehorende bit in het SPR16EN-register ($D06B, 53355 decimaal) te zetten.

Om de acht sprites elk 15 unieke kleuren te laten hebben, wordt de spritekleur weergegeven met behulp van de paletingang die overeenkomt met: $spritenummer \times 16 + nibblewaarde$, waarbij $spritenummer$ het nummer van de sprite is (van 0 tot en met 7) en $nibblewaarde$ de waarde is van het halfbyte dat de spritegegevens voor de pixel bevat. Indien bovendien de bitplane-modus voor deze sprite is ingeschakeld, wordt 128 opgeteld bij de kleurwaarde, waardoor het eenvoudig wordt om tussen twee kleurenschema's voor een gegeven sprite te schakelen door slechts één bit in het SPRBPMEN-register te wijzigen.

Omdat de vollekleuren-spritemodus vier bits per pixel vereist, zijn sprites slechts zes pixels breed, tenzij extra-breed voor een sprite is ingeschakeld, in welk geval de sprite 16 pixels breed is. Tegelmodus kan gecombineerd worden met de vollekleuren-spritemodus, en heeft dan tot gevolg dat de 16 vollekleuren-pixels van de sprite worden herhaald tot aan het einde van de rasterlijn.

Het volgende BASIC-programma tekent een vollekleurensprite in C64- of C65-modus:

```
10 PRINT CHR$(147)
20 REM C65-/C64-MODUS DETECTEREN
30 IF PEEK(53272) AND 32 THEN GOTO 100
40 POKE 53295,ASC("6"):POKE 53295,ASC("$")
100 REM Sprite instellen
110 AD=4096                          :REM $1000-spriteadres
120 TC=10                            :REM transparante kleur
130 SPR=PEEK(53356)+PEEK(53357)*256 :REM sprite-tabeladres ophalen
140 POKE SPR,AD/64                   :REM spriteadres instellen
150 FOR I=AD TO AD+168               :REM sprite met TC wissen
160 POKE I,TC+TC*16                  :REM een byte = 2 pixels
170 NEXT
180 POKE 53287,TC                    :REM transparante kleur instellen
190 POKE 53248,100                   :REM sprite op 100,100 zetten...
200 POKE 53249,100                   :REM op het scherm op 100,100
210 POKE 53355,1                     :REM sprite 0 16-kleurig maken
220 POKE 53335,1                     :REM sprite 0 16X4-bits laten gebruiken
230 POKE 53269,1                     :REM sprite 0 inschakelen
240 GOSUB 900                        :REM meerkleurensprite lezen
250 END

900 REM Sprite uit gegevens laden
910 READ N$:IF N$="END" THEN RETURN
920 GOSUB 1000                       :REM regel decoderen
930 GOTO 910
```

```
1000 REM Tekenreeks van nibbles in N$ op adres AD decoderen
1010 IF LEN(N$)<>16 THEN PRINT "ILLEGAL SPR DATA!":END
1020 FOR I=1 TO 16 STEP 2
1030 N=(ASC(MID$(N$,I,1))-ASC("@"))    :REM hoge nibble
1040 IF N<0 THEN N=TC                  :REM . is transparant
1050 M=(ASC(MID$(N$,I+1,1))-ASC("@"))  :REM lage nibble
1060 IF M<0 THEN M=TC                  :REM . is transparant
1070 POKE AD,(N AND 15)*16 + (M AND 15):REM 2 pixels instellen
1080 AD=AD+1                           :REM AD ophogen
1090 NEXT I
1100 RETURN

1998 REM Spritegegevens
1999 REM . = transparant, @-O = kleuren 0 tot 15
2000 DATA "..AAFF...HHCC..."
2010 DATA ".AAFF.....HHCC.."
2020 DATA "AAFF.......HHCC."
2030 DATA "AFF...@@@...HHC."
2040 DATA "FF..@@GGG@@..HH."
2050 DATA "..@@GGGGGGG@@..."
2060 DATA ".@GGGGGGGGGG@.."
2070 DATA ".@GGGGGGGGGG@.."
2080 DATA "@GGG@@GGG@@GGG@."
2090 DATA "@GG@GGGGGG@GG@."
2100 DATA "@GGGGGGGGGGGG@."
2110 DATA "@GGGGGBGBGGGGG@."
2120 DATA "@GGGBBBBBBBGGG@."
2130 DATA ".@GGGBBBGGGG@.."
2140 DATA ".@GGGGGBGGGGG@.."
2150 DATA "..@@GGGGGGG@@..."
2160 DATA "II..@@GGG@@..KK."
2170 DATA "DII...@@@...KKE."
2180 DATA "DDII.......KKEE."
2190 DATA ".DDII.....KKEE.."
2200 DATA "..DDII...KKEE..."
2210 DATA "END"
```

VIC-III ERRATA-NIVEAU

Er zijn een aantal gevallen waarin de VIC-III-chip in de bekende Commodore 65-prototypen niet overeenkomstig de specificatie functioneert, de specificatie onvoldoende detail bevat en de implementatie opmerkelijk inconsistent is, of het ontwerp zelf gebreken of inconsistenties vertoont. Het standaardgedrag van de VIC-IV is om de VIC-III in deze gevallen zo nauwkeurig mogelijk te emuleren. In sommige gevallen waarin het gedrag van de VIC-III tekortschiet, biedt de VIC-IV verbeterd gedrag, dat door software kan worden geselecteerd met behulp van het HWERRATA-register op $D08F.

Omdat deze correcties achterwaarts onverenigbaar zijn met de VIC-III en met eerdere versies van de VIC-IV en de MEGA65-core, moet software deze correcties expliciet activeren door het HWERRATA-register te zetten. Dit beschermt software tegen verdere wijzigingen die in toekomstige versies van de MEGA65-core kunnen worden geïntroduceerd. De beginstand van dit register is $00, wat volledige compatibiliteit met de VIC-III aanvraagt, inclusief het foutieve gedrag. De MEGA65-ROM zal dit register bij het starten van programma's altijd op $00 laten staan. Een programma kan HWERRATA instellen om een set correcties te activeren die de ontwikkelaar bij het schrijven van het programma kende, en achterwaarts onverenigbare correcties uit te sluiten die in latere versies kunnen worden geïntroduceerd.

Het aanvragen van een errata-niveau activeert alle correcties tot dat niveau. Er is bewust geen mogelijkheid om een niveau aan te vragen en specifieke correcties op lagere niveaus uit te sluiten. U moet uw programma zodanig schrijven dat het alle correcties tot het gevraagde niveau ondersteunt.

In gevallen waarin het activeren van een correctie het gedrag van een heet register wijzigt, zal het instellen van het errata-niveau hete-registerpropagatie *niet* activeren. Het programma moet de hete-registerpropagatie zelf activeren door een 1 te schrijven naar het HOTREG-register.

De tot nu toe geïmplementeerde errata-niveaus zijn als volgt:

VIC-III errata-niveaus

Niveau	Gecorrigeerd gedrag	Geïntroduceerd in
0	Geen correcties. Volledig VIC-III-compatibel.	N/A
1	**X-scrollpositie verschoven in de H640-modus.** De VIC-III geeft de X-scrollpositie voor het vloeiend scrollen van tekst ($D016 XSCL) onjuist weer in de H640-modus. De correctie verschuift de scrollpositie met 1 logische pixel (2 fysieke pixels) naar rechts. Dit wordt pas van kracht na de hot register-propagatie.	v0.96

vervolg ...

Niveau	Gecorrigeerd gedrag	Geïntroduceerd in
X	**Gedrag van de busy-vlag van de SD-kaart.** De busy-vlag van de SD-kaart (bit 1 van $D680) geeft aan of de SD-kaartcontroller op laag niveau bezig is. Door de toevoeging van de read-ahead-functionaliteit van de SD-kaartcontroller zal oudere software, die erop rekent dat dit bit vrijkomt na voltooiing van een leesbewerking, ten onrechte wachten totdat de volledige read-ahead-sequentie is afgerond. Dit kan er op zijn beurt toe leiden dat software ten onrechte meent dat het lezen van de sector is mislukt vanwege de langere benodigde tijd, of dat een ander leesverzoek niet onmiddellijk kan worden ingediend. Daarom geeft de busy-vlag van de SD-kaart onder dit errata-niveau niet aan of de SD-kaartcontroller een read-ahead-bewerking op de achtergrond voor een sector uitvoert. Op of boven dit errata-niveau wordt deze informatie niet verborgen: bit 0 komt vrij wanneer de gevraagde sector beschikbaar is, maar bit 1 blijft gezet zolang een read-ahead-bewerking doorgaat.	Nog niet geïmplementeerd

VIC-II / C64-REGISTERS

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D000	53248	S0X							
D001	53249	S0Y							
D002	53250	S1X							
D003	53251	S1Y							
D004	53252	S2X							
D005	53253	S2Y							
D006	53254	S3X							
D007	53255	S3Y							
D008	53256	S4X							
D009	53257	S4Y							
D00A	53258	S5X							
D00B	53259	S5Y							
D00C	53260	S6X							
D00D	53261	S6Y							
D00E	53262	S7X							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D00F	53263	S7Y							
D010	53264	SXMSB							
D011	53265	RC8	ECM	BMM	BLNK	RSEL	YSCL		
D012	53266	RC							
D013	53267	LPX							
D014	53268	LPY							
D015	53269	SE							
D016	53270	-		RST	MCM	CSEL	XSCL		
D017	53271	SEXY							
D018	53272	VS					CB		-
D019	53273	-				ILP	ISSC	ISBC	RIRQ
D01A	53274	-					MISSC	MISBC	MRIRQ
D01B	53275	BSP							
D01C	53276	SCM							
D01D	53277	SEXX							
D01E	53278	SSC							
D01F	53279	SBC							
D020	53280	-				BORDERCOL			
D021	53281	-				SCREENCOL			
D022	53282	-				MC1			
D023	53283	-				MC2			
D024	53284	-				MC3			
D025	53285	SPRMC0							
D026	53286	SPRMC1							
D027	53287	SPR0COL							
D028	53288	SPR1COL							
D029	53289	SPR2COL							
D02A	53290	SPR3COL							
D02B	53291	SPR4COL							
D02C	53292	SPR5COL							
D02D	53293	SPR6COL							
D02E	53294	SPR7COL							
D02F	53295	KEY							
D030	53296	-							C128-FAST

- **BLNK** Weergave inschakelen: 0 = weergave uitschakelen, 1 = weergave inschakelen
- **BMM** Bitmapmodus
- **BORDERCOL** Randkleur van het scherm (16 kleuren)
- **BSP** Bitstelling sprite-/achtergrondprioriteit

- **C128FAST** 2 MHz-selectie (voor C128 2 MHz-emulatie)
- **CB** Adreslocatie van de tekenset (× 1 KiB)
- **CSEL** 38/40-kolommenselectie
- **ECM** Uitgebreide achtergrondmodus
- **ILP** Lichtpen indicator of bevestiging
- **ISBC** Sprite-/bitmapbotsing indicator of bevestiging
- **ISSC** Sprite-/spritebotsing indicator of bevestiging
- **KEY** $00 en dan $00 schrijven om C64/VIC-II I/O-registers in te schakelen
- **LPX** Grove horizontale bundelpositie (was lichtpen X)
- **LPY** Grove verticale bundelpositie (was lichtpen Y)
- **MC1** Multikleur 1 (16 kleuren)
- **MC2** Multikleur 2 (16 kleuren)
- **MC3** Multikleur 3 (16 kleuren)
- **MCM** Multikleurmodus
- **MISBC** Masker sprite-/bitmapbotsing IRQ
- **MISSC** Masker sprite-/spritebotsing IRQ
- **MRIRQ** Masker rasterIRQ
- **RC** Rastervergelijking bits 0 tot 7
- **RC8** Rastervergelijking bit 8
- **RIRQ** Rastervergelijking indicator of bevestiging
- **RSEL** 24/25-rijselectie
- **RST** Schakelt de video-uitvoer op de MAX Machine(tm) VIC-II 6566 uit. Wordt genegeerd op normale C64-computers en de MEGA65
- **SBC** Indicatiebits voor sprite/voorgrondbotsing
- **SCM** Inschakelbits voor de sprite-multikleur
- **SCREENCOL** Schermkleur (16 kleuren)
- **SE** Inschakelbits voor sprites
- **SEXX** Inschakelbits voor horizontale sprite-vergroting
- **SEXY** Inschakelbits voor verticale sprite-vergroting
- **SNX** Horizontale positie van sprite N
- **SNY** Verticale positie van sprite N

- **SPRMC0** Sprite-multikleur 0
- **SPRMC1** Sprite-multikleur 1
- **SPRNCOL** Kleur van sprite N / transparantiekleur van de sprite in 16-kleurenmodus (lagere nibble)
- **SSC** Indicatiebits voor sprite/spritebotsing
- **SXMSB** MSB's van de horizontale sprite-positie
- **VS** Schermadres (× 1KiB)
- **XSCL** Horizontaal vloeiend scrollen
- **YSCL** 24/25 verticaal vloeiend scrollen

VIC-III / C65-REGISTERS

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D020	53280	BORDERCOL							
D021	53281	SCREENCOL							
D022	53282	MC1							
D023	53283	MC2							
D024	53284	MC3							
D025	53285	SPRMC0							
D026	53286	SPRMC1							
D02F	53295	KEY							
D030	53296	ROME	CROM9	ROMC	ROMA	ROM8	PAL	EXTSYNC	CRAM2K
D031	53297	H640	FAST	ATTR	BPM	V400	H1280	MONO	INT
D032	53298	BPENABLE							
D033	53299	B0ADODD			-	B0ADEVN			-
D034	53300	B1ADODD			-	B1ADEVN			-
D035	53301	B2ADODD			-	B2ADEVN			-
D036	53302	B3ADODD			-	B3ADEVN			-
D037	53303	B4ADODD			-	B4ADEVN			-
D038	53304	B5ADODD			-	B5ADEVN			-
D039	53305	B6ADODD			-	B6ADEVN			-
D03A	53306	B7ADODD			-	B7ADEVN			-
D03B	53307	BPCOMP							
D03C	53308	BPX							
D03D	53309	BPY							
D03E	53310	HPOS							
D03F	53311	VPOS							
D040	53312	B0PIX							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D041	53313	B1PIX							
D042	53314	B2PIX							
D043	53315	B3PIX							
D044	53316	B4PIX							
D045	53317	B5PIX							
D046	53318	B6PIX							
D047	53319	B7PIX							
D100 - D1FF	53504 - 53759	PALRED							
D200 - D2FF	53760 - 54015	PALGREEN							
D300 - D3FF	54016 - 54271	PALBLUE							

- **ATTR** Uitgebreide attributen en 8-bits kleuringangen inschakelen
- **BNPIX** Display Address Translater (DAT) Bitplane N-poort
- **BORDERCOL** Weergaverandkleur (256 kleuren)
- **BPCOMP** Complement-vlaggen voor bitplanes
- **BPENABLE** Inschakelbits voor bitplanes
- **BPM** Bitplane-modus
- **BPX** Bitplane X
- **BPY** Bitplane Y
- **BXADEVN** Even rijen van even Bitplane X-adres[15:13] (voor v200 wordt de even rij tweemaal getekend)
- **BXADODD** Oneven rijen van even Bitplane X-adres[15:13] (wordt alleen gebruikt in de v400-modus)
- **BYADEVN** Even rijen van oneven Bitplane Y-adres[15:13] (+64KB) (voor v200 wordt de even rij tweemaal getekend)
- **BYADODD** Oneven rijen van oneven Bitplane Y-adres[15:13] (+64KB) (wordt alleen gebruikt in de v400-modus)
- **CRAM2K** Mapt de 2e KB van het kleurengeheugen $DC00-$DFFF
- **CROM9** Selecteert tussen de C64- en C65-tekenset (niet geïmplementeerd)
- **EXTSYNC** Externe videosynchronisatie inschakelen (genlock-ingang)
- **FAST** C65 FAST-modus inschakelen (~3.5MHz)
- **H1280** 1280 horizontale pixels inschakelen (niet geïmplementeerd, schakelt PALEMU uit)

- **H640** C64 640 horizontale pixels / 80-kolommenmodus inschakelen
- **HPOS** Bitplane X-offset
- **INT** Interlaced modus van de VIC-III inschakelen
- **KEY** $A5 en dan $96 schrijven om de C65/VIC-III I/O-registers in te schakelen
- **MC1** Multikleur 1 (256 kleuren)
- **MC2** Multikleur 2 (256 kleuren)
- **MC3** Multikleur 3 (256 kleuren)
- **MONO** VIC-III MONO-composietvideo-uitvoer inschakelen (kleur indien uitgeschakeld)
- **PAL** Gebruikt ingangen van het PALETTE ROM (0) of het RAM (1) voor kleuren 0 - 15
- **PALBLUE** Blauwe paletwaarden (omgekeerde nibble-volgorde)
- **PALGREEN** Groene paletwaarden (omgekeerde nibble-volgorde)
- **PALRED** Rode paletwaarden (omgekeerde nibble-volgorde)
- **ROM8** Mapt de C65-ROM $8000
- **ROMA** Mapt de C65-ROM $A000
- **ROMC** Mapt de C65-ROM $C000
- **ROME** Mapt de C65-ROM $E000
- **SCREENCOL** Schermkleur (256 kleuren)
- **SPRMC0** Sprite-multikleur 0 (8-bits voor selectie van elke paletkleur)
- **SPRMC1** Sprite-multikleur 1 (8-bits voor selectie van elke paletkleur)
- **V400** 400 verticale pixels inschakelen (schakelt PALEMU uit)
- **VPOS** Bitplane Y-offset

VIC-IV / MEGA65-SPECIFIEKE REGISTERS

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D020	53280	BORDERCOL							
D021	53281	SCREENCOL							
D022	53282	MC1							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D023	53283	MC2							
D024	53284	MC3							
D025	53285	SPRMC0							
D026	53286	SPRMC1							
D02F	53295	KEY							
D048	53320	TBDRPOS							
D049	53321	SPRBPMEN				TBDRPOS			
D04A	53322	BBDRPOS							
D04B	53323	SPRBPMEN				BBDRPOS			
D04C	53324	TEXTXPOS							
D04D	53325	SPRTILEN				TEXTXPOS			
D04E	53326	TEXTYPOS							
D04F	53327	SPRTILEN				TEXTYPOS			
D050	53328	XPOSLSB							
D051	53329	NORRDEL	DBLRR	XPOSMSB					
D052	53330	FNRASTERLSB							
D053	53331	FNRST	SHDEMU	UPSCALE	BOL-DISALT	FNRASTERMSB			
D054	53332	ALPHEN	VFAST	PALEMU	SPRH640	SMTH	FCLRHI	FCLRLO	CHR16
D055	53333	SPRHGTEN							
D056	53334	SPRHGHT							
D057	53335	SPRX64EN							
D058	53336	LINESTEPLSB							
D059	53337	LINESTEPMSB							
D05A	53338	CHRXSCL							
D05B	53339	CHRYSCL							
D05C	53340	SDBDRWDLSB							
D05D	53341	HOTREG	RSTDELEN	SDBDRWDMSB					
D05E	53342	CHRCOUNT							
D05F	53343	SPRXSMSBS							
D060	53344	SCRNPTRLSB							
D061	53345	SCRNPTRMSB							
D062	53346	SCRNPTRBNK							
D063	53347	EXGLYPH	FCOLMCM	CHRCOUNT		SCRNPTRMB			
D064	53348	COLPTRLSB							
D065	53349	COLPTRMSB							
D068	53352	CHARPTRLSB							
D069	53353	CHARPTRMSB							
D06A	53354	CHARPTRBNK							
D06B	53355	SPR16EN							
D06C	53356	SPRPTRADRLSB							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D06D	53357	SPRPTRADRMSB							
D06E	53358	SPRPTR16	SPRPTRBNK						
D06F	53359	PALNTSC	VGAHDTV	RASLINE0					
D070	53360	MAPEDPAL		BTPALSEL		SPRPALSEL		ABTPALSEL	
D071	53361	BP16ENS							
D072	53362	SPRYADJ							
D073	53363	RASTERHEIGHT				ALPHADELAY			
D074	53364	SPRENALPHA							
D075	53365	SPRALPHAVAL							
D076	53366	SPRENV400							
D077	53367	SPRYMSBS							
D078	53368	SPRYSMSBS							
D079	53369	RASCMP							
D07A	53370	FNRST-CMP	EXTIRQS	NOBUG-COMPAT	CHARY16	SPTR-CONT	RASCMPMSB		
D07B	53371	DISPROWS							
D07C	53372	DEBUGC		VSYNCP	HSYNCP	NORRB-WRAP	BITPBANK		

- **ABTPALSEL** Bitmap-/tekstpaletbank van de VIC-IV (alternatief palet)
- **ALPHADELAY** Alfa-vertraging voor de compositor
- **ALPHEN** Alfa-compositor inschakelen
- **BBDRPOS** Positie van de onderrand
- **BITPBANK** Bepaalt welke 128KB-bank de bitplanes gebruiken
- **BOLDISALT** Indien gezet, selecteert de BOLD-attribuutbit het alternatieve palet, zonder dat REVERSE eveneens gezet hoeft te worden.
- **BORDERCOL** Weergaverandkleur (256 kleuren)
- **BP16ENS** Inschakelvlaggen voor de 16-kleurenbitplanes van de VIC-IV
- **BTPALSEL** Bitmap-/tekstpaletbank
- **CHARPTRBNK** Nauwkeurig basisadres van de tekenset (bits 23 - 16)
- **CHARPTRLSB** Nauwkeurig basisadres van de tekenset (bits 0 - 7)
- **CHARPTRMSB** Nauwkeurig basisadres van de tekenset (bits 15 - 8)
- **CHARY16** Alternatieve teken-ROM-bank op afwisselende rasterlijnen in V200
- **CHR16** 16-bits teekennummers inschakelen (twee schermbytes per teken)
- **CHRCOUNT** Aantal weer te geven tekens per rij (LSB)

- **CHRXSCL** Horizontale hardwarematige schaal van de tekstmodus (een honderdtwintigste pixel per pixel)
- **CHRYSCL** Verticale schaling van de tekstmodus (aantal fysieke rasterlijnen per tekstrij)
- **COLPTRLSB** Basisadres van het kleurengeheugen (bits 0 - 7)
- **COLPTRMSB** Basisadres van het kleurengeheugen (bits 15 - 8)
- **DBLRR** Wanneer deze is gezet, wordt de Raster Rewrite Buffer slechts om de tweede rasterlijn bijgewerkt, waardoor de resolutie wordt beperkt tot V200 maar er meer cycli voor Raster-Rewrite-acties overblijven.
- **DEBUGC** VIC-IV-debugpixelselectie: het rode (01), groene (10) of blauwe (11) kanaal is zichtbaar in $D07D
- **DISPROWS** Aantal tekstrijen dat moet worden weergegeven
- **EXGLYPH** Haalt full-colour-tekengegevens uit het uitbreidingsgeheugen
- **EXTIRQS** Schakelt extra IRQ-bronnen in, bijv. de raster-X-positie
- **FCLRHI** Schakelt de full-colour-modus in voor tekennummers >$FF
- **FCLRLO** Schakelt de full-colour-modus in voor tekennummers <=$FF
- **FCOLMCM** Schakelt 256 kleuren in de multikleur-tekstmodus in
- **FNRASTERLSB** Leest de fysieke rasterpositie
- **FNRASTERMSB** Leest de fysieke rasterpositie
- **FNRST** Leest de bron van de rastervergelijking (0=VIC-IV-fijne raster, 1=VIC-II-raster) en levert dezelfde waarde als in FNRSTCMP is ingesteld
- **FNRSTCMP** De rastervergelijking vindt plaats in fysieke rasterlijnen indien dit bit is gewist, of in VIC-II-lijnen indien het is gezet
- **HOTREG** Schakelt de VIC-II hot registers in. Wanneer deze zijn ingeschakeld, zorgt het aanraken van veel VIC-II-registers ervoor dat de VIC-IV de weergaveparameters herberekent, zoals randposities en -afmetingen. Het aanraken van registers terwijl deze functie is uitgeschakeld leidt tot een wijziging wanneer deze weer wordt ingeschakeld. Deze waarde op 0 instellen wist de herberekeningsvlag en annuleert de herberekening.
- **HSYNCP** hsync-polariteit
- **KEY** Schrijf $47 en dan $53 om de MEGA65/VIC-IV I/O-registers in te schakelen
- **LINESTEPLSB** Aantal bytes waarmee tussen elke tekstrij wordt opgeschoven (LSB)
- **LINESTEPMSB** Aantal bytes waarmee tussen elke tekstrij wordt opgeschoven (MSB)

- **MAPEDPAL** Ingemapte paletbank op $D100-$D3FF
- **MC1** Multikleur 1 (256 kleuren)
- **MC2** Multikleur 2 (256 kleuren)
- **MC3** Multikleur 3 (256 kleuren)
- **NOBUGCOMPAT** *VEROUDERD*, gebruik HWERRATA. Schakelt de VIC-III / C65-bugcompatibiliteitsmodus uit indien deze is gezet
- **NORRBWRAP** Schakelt de RRB-adreswrap-around uit wanneer deze is gezet
- **NORRDEL** Wanneer deze is gewist, wordt dubbelbuffering van de raster rewrite gebruikt
- **PALEMU** Schakelt PAL-CRT-achtige scanlijn-emulatie in (uitgeschakeld door V400 of H1280)
- **PALNTSC** NTSC-emulatiemodus (maximaal raster = 262)
- **RASCMP** Fysieke rastervergelijkingwaarde die moet worden gebruikt indien FNRSTCMP is gewist
- **RASCMPMSB** MSB van de rastervergelijkingwaarde
- **RASLINE0** Eerste VIC-II-rasterlijn
- **RASTERHEIGHT** Fysieke rasterlijnen per VIC-II-raster (1 tot 16)
- **RSTDELEN** Schakelt de rastervertraging in (vertraagt de rasterteller en interrupts met één regel om aan te sluiten bij de latentie van de uitvoerpipeline)
- **SCREENCOL** Schermkleur (256 kleuren)
- **SCRNPTRBNK** Nauwkeurig basisadres van het scherm-RAM (bits 23 - 16)
- **SCRNPTRLSB** Nauwkeurig basisadres van het scherm-RAM (bits 7 - 0)
- **SCRNPTRMB** Nauwkeurig basisadres van het scherm-RAM (bits 27 - 24)
- **SCRNPTRMSB** Nauwkeurig basisadres van het scherm-RAM (bits 15 - 8)
- **SDBDRWDLSB** Breedte van de zijrand (LSB)
- **SDBDRWDMSB** Breedte van de zijrand (MSB)
- **SHDEMU** Gesimuleerd schaduwmasker inschakelen (PALEMU moet ook zijn ingeschakeld)
- **SMTH** Horizontale vloeiendheid van de video-uitvoer inschakelen
- **SPR16EN** Inschakelbits voor de 16-kleurenmodus van sprites
- **SPRALPHAVAL** Alpha-vermengingswaarde van sprites
- **SPRBPMEN** Inschakelbits voor de sprite-bitplane-wijzigingsmodus
- **SPRENALPHA** Alpha-vermenging van sprites inschakelen

- **SPRENV400** Inschakelbits voor de V400-modus van sprites
- **SPRH640** H640-modus van sprites inschakelen
- **SPRHGHT** Grootte van de uitgebreide sprite-hoogte (in sprite-pixels)
- **SPRHGTEN** Uitgebreide sprite-hoogte inschakelen (één bit per sprite)
- **SPRMC0** Sprite-multikleur 0 (8-bits voor de keuze van elke paletkleur)
- **SPRMC1** Sprite-multikleur 1 (8-bits voor de keuze van elke paletkleur)
- **SPRPALSEL** Spritepaletbank
- **SPRPTR16** 16-bits sprite-pointermodus (ermee kunnen sprites op elke 64-bytegrens in chip-RAM worden geplaatst)
- **SPRPTRADRLSB** Sprite-pointeradres (bits 7 - 0)
- **SPRPTRADRMSB** Sprite-pointeradres (bits 15 - 8)
- **SPRPTRBNK** Sprite-pointeradres (bits 23 - 16)
- **SPRTILEN** Inschakelbits voor horizontale sprite-tegels
- **SPRX64EN** Inschakelbits voor de uitgebreide sprite-breedte (8 bytes per sprite-rij = 64 pixels breed voor normale sprites of 16 pixels breed voor de 16-kleuren-sprite-modus)
- **SPRXSMSBS** Super-MSB's van de X-positie van sprites in de H640-modus
- **SPRYADJ** Aanpassing van de Y-positie van sprites
- **SPRYMSBS** MSB's van de Y-positie van sprites in de V400-modus
- **SPRYSMSBS** Super-MSB's van de Y-positie van sprites in de V400-modus
- **SPTRCONT** Bewaakt voortdurend de sprite-pointer zodat de gegevensbron van de sprite kan worden gewijzigd terwijl een sprite wordt getekend
- **TBDRPOS** Positie van de bovenrand
- **TEXTXPOS** Horizontale positie van de tekengenerator
- **TEXTYPOS** Verticale positie van de tekengenerator
- **UPSCALE** Ingebouwde 720p-upscaler met lage latentie (130 usec) inschakelen
- **VFAST** MEGA65 FAST-modus (40.5MHz)
- **VGAHDTV** Selecteert indien gezet een meer VGA-compatibele modus in plaats van de cyclus-exacte frame-timing van de HDMI/HDTV-VIC-II. Kan helpen om een functioneel beeld op oudere VGA-monitoren te produceren.
- **VSYNCP** vsync-polariteit
- **XPOSLSB** Leest de LSB van de horizontale raster-scanpositie
- **XPOSMSB** Leest de MSB van de horizontale raster-scanpositie

BIJLAGE R

Sound Interface Device (SID)

- SID-registers

SID-REGISTERS

De MEGA65 heeft vier ingebouwde SID's, bereikbaar via de registerbereiken die beginnen bij $D400, $D420, $D440 en $D460. De registers in elk van deze bereiken zijn exact hetzelfde, dus de volgende tabel vermeldt alleen de eerste SID. Tel er 32 bij op om bij de volgende SID te komen.

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D400	54272	VOICE1FRQLO							
D401	54273	VOICE1FRQHI							
D402	54274	VOICE1PWLO							
D403	54275	VOICE1UNSD				VOICE1PWHI			
D404	54276	VOICE1-CTRLRNW	VOICE1-CTRLPUL	VOICE1-CTRLSAW	VOICE1-CTRLTRI	VOICE1-CTRLTST	VOICE1-CTRLRMO	VOICE1-CTRLRMF	VOICE1-CTRLGATE
D405	54277	ENV1ATTDUR				ENV1DECDUR			
D406	54278	ENV1SUSLEV				ENV1RELDUR			
D407	54279	VOICE2FRQLO							
D408	54280	VOICE2FRQHI							
D409	54281	VOICE2PWLO							
D40A	54282	VOICE2UNSD				VOICE2PWHI			
D40B	54283	VOICE2-CTRLRNW	VOICE2-CTRLPUL	VOICE2-CTRLSAW	VOICE2-CTRLTRI	VOICE2-CTRLTST	VOICE2-CTRLRMO	VOICE2-CTRLRMF	VOICE2-CTRLGATE
D40C	54284	ENV2ATTDUR				ENV2DECDUR			
D40D	54285	ENV2LEVDUR				ENV2RELDUR			
D40E	54286	VOICE3FRQLO							
D40F	54287	VOICE3FRQHI							
D410	54288	VOICE3PWLO							
D411	54289	VOICE3UNSD				VOICE3PWHI			
D412	54290	VOICE3-CTRLRNW	VOICE3-CTRLPUL	VOICE3-CTRLSAW	VOICE3-CTRLTRI	VOICE3-CTRLTST	VOICE3-CTRLRMO	VOICE3-CTRLRMF	VOICE3-CTRLGATE
D413	54291	ENV3ATTDUR				ENV3DECDUR			
D414	54292	ENV3LEVDUR				ENV3RELDUR			
D415	54293	FLTRCUTFRQLO							
D416	54294	FLTRCUTFRQHI							
D417	54295	FLTRRESON				FLTR-EXTINP	FLTR-V3OUT	FLTR-V2OUT	FLTR-V1OUT
D418	54296	FLTR-CUTV3	FLTR-HIPASS	FLTR-BDPASS	FLTR-LOPASS	FLTRVOL			
D419	54297	PADDLE1							
D41A	54298	PADDLE2							
D41B	54299	OSC3RNG							
D41C	54300	ENV3OUT							
D41D	54301	SIDPANLEFT							
D41E	54302	SIDPANRIGHT							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D63C	54844	-			SIDEXENA	SIDMODE			

- **ENV3OUT** Uitgang van envelopegenerator 3
- **ENVXATTDUR** Duur van de aanzetcyclus van envelopegenerator X
- **ENVXDECDUR** Duur van de terugvalcyclus van envelopegenerator X
- **ENVXRELDUR** Duur van de wegstervenscyclus van envelopegenerator X
- **ENVXSUSLEV** Aanhoudniveau van envelopegenerator X
- **FLTRBDPASS** Filter banddoorlaatmodus
- **FLTRCUTFRQHI** Hoge afsnijfrequentie van het filter
- **FLTRCUTFRQLO** Lage afsnijfrequentie van het filter
- **FLTRCUTV3** Filter uitschakelen op de uitgang van stem 3 (1 = uit)
- **FLTREXTINP** Externe ingang van het filter
- **FLTRHIPASS** Filter hoogdoorlaatmodus
- **FLTRLOPASS** Filter laagdoorlaatmodus
- **FLTRRESON** Filterresonantie
- **FLTRVOL** Uitgangsvolume van het filter
- **FLTRVXOUT** Filteruitgang van stem X
- **OSC3RNG** Generator voor willekeurige getallen van oscillator 3
- **PADDLE1** Analoog/digitaalomzetter: paddle 1 (0-255)
- **PADDLE2** Analoog/digitaalomzetter: paddle 2 (0-255)
- **SIDEXENA** Extra SID-registers inschakelen
- **SIDMODE** Selecteert de SID-modus: 0=6581, 1=8580
- **SIDPANLEFT** Volume van het linkerkanaal van de SID (vereist dat SIDEXENA is ingesteld)
- **SIDPANRIGHT** Volume van het rechterkanaal van de SID (vereist dat SIDEXENA is ingesteld)
- **VOICE1CTRLRMF** Stem 1 synchroniseert oscillator 1 met de frequentie van oscillator 3
- **VOICE1CTRLRMO** Stem 1 ringmoduleert oscillator 1 met de uitgang van oscillator 3

- **VOICE2CTRLRMF** Stem 2 synchroniseert oscillator 2 met de frequentie van oscillator 1
- **VOICE2CTRLRMO** Stem 2 ringmoduleert oscillator 2 met de uitgang van oscillator 1
- **VOICE3CTRLRMF** Stem 3 synchroniseert oscillator 3 met de frequentie van oscillator 2
- **VOICE3CTRLRMO** Stem 3 ringmoduleert oscillator 3 met de uitgang van oscillator 2
- **VOICEXCTRLGATE** Stem X gate-bit (1 = start, 0 = wegsterven)
- **VOICEXCTRLPUL** Pulsgolfvorm van stem X
- **VOICEXCTRLRNW** Willekeurige ruisgolfvorm van stem X
- **VOICEXCTRLSAW** Zaagtandgolfvorm van stem X
- **VOICEXCTRLTRI** Driehoeksgolf van stem X
- **VOICEXCTRLTST** Testbit van stem X - de oscillator uitschakelen
- **VOICEXFRQHI** Frequentie van stem X (hoge byte)
- **VOICEXFRQLO** Frequentie van stem X (lage byte)
- **VOICEXPWHI** Pulsgolfbreedte van stem X (hoge byte)
- **VOICEXPWLO** Pulsgolfbreedte van stem X (lage byte)
- **VOICEXUNSD** Ongebruikt

BIJLAGE S

6526 Complex Interface Adapter (CIA)-registers

- CIA 6526-registers
- CIA 6526-hypervisorregisters

CIA 6526-REGISTERS

CIA 1-registers

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DC00	56320	PORTA							
DC01	56321	PORTB							
DC02	56322	DDRA							
DC03	56323	DDRB							
DC04	56324	TIMERA							
DC05	56325	TIMERA							
DC06	56326	TIMERB							
DC07	56327	TIMERB							
DC08	56328	-				TODJIF			
DC09	56329	-		TODSEC					
DC0A	56330	-		TODMIN					
DC0B	56331	TOD-AMPM	-		TODHOUR				
DC0C	56332	SDR							
DC0D	56333	IR	ISRCLR		FLG	SP	ALRM	TB	TA
DC0E	56334	TOD50	SPMOD	IMODA	-	RMODA	OMODA	PBONA	STRTA
DC0F	56335	TODEDIT	IMODB		LOAD	RMODB	OMODB	PBONB	STRTB

- **ALRM** TOD-alarm
- **DDRA** DDR van poort A
- **DDRB** DDR van poort B
- **FLG** FLAG-flank gedetecteerd
- **IMODA** Tickbron van Timer A
- **IMODB** Tickbron van Timer B
- **IR** Interruptvlag
- **ISRCLR** Plaatshouder - lezen wist de gebeurtenissen
- **LOAD** Strobe-invoer om de timers geforceerd te laden
- **OMODA** Omschakelen of puls voor Timer A
- **OMODB** Omschakelen of puls voor Timer B
- **PBONA** PB6-uitgang van timer A
- **PBONB** PB7-uitgang van timer B
- **PORTA** Poort A

- **PORTB** Poort B
- **RMODA** Timer A one-shotmodus
- **RMODB** Timer B one-shotmodus
- **SDR** Schuifregistergegevensregister (schrijven start het verzenden)
- **SP** Schuifregister vol/leeg
- **SPMOD** Richting van de seriële poort
- **STRTA** Start van timer A
- **STRTB** Start van timer B
- **TA** Onderloop van timer A
- **TB** Onderloop van timer B
- **TIMERA** Teller van timer A (16-bits)
- **TIMERB** Teller van timer B (16-bits)
- **TOD50** 50/60Hz-selectie voor de TOD-klok
- **TODAMPM** PM-vlag van de TOD
- **TODEDIT** TOD-alarm bewerken
- **TODHOUR** Uren van de TOD
- **TODJIF** Tienden van seconden van de TOD
- **TODMIN** Minuten van de TOD
- **TODSEC** Seconden van de TOD

CIA2-registers

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DD00	56576	PORTA							
DD01	56577	PORTB							
DD02	56578	DDRA							
DD03	56579	DDRB							
DD04	56580	TIMERA							
DD05	56581	TIMERA							
DD06	56582	TIMERB							
DD07	56583	TIMERB							
DD08	56584	-				TODJIF			
DD09	56585	-		TODSEC					
DD0B	56587	TOD-AMPM	-		TODHOUR				

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DD0C	56588	SDR							
DD0D	56589	IR	ISRCLR		FLG	SP	ALRM	TB	TA
DD0E	56590	TOD50	SPMOD	IMODA	–	RMODA	OMODA	PBONA	STRTA
DD0F	56591	TODEDIT	IMODB		LOAD	RMODB	OMODB	PBONB	STRTB

- **ALRM** TOD-alarm
- **DDRA** DDR van poort A
- **DDRB** DDR van poort B
- **FLG** FLAG-flank gedetecteerd
- **IMODA** Tickbron van Timer A
- **IMODB** Tickbron van Timer B
- **IR** Interruptvlag
- **ISRCLR** Plaatshouder - Lezen wist de gebeurtenissen
- **LOAD** Strobe-invoer om de timers met dwang te laden
- **OMODA** Omschakelen of puls voor Timer A
- **OMODB** Omschakelen of puls voor Timer B
- **PBONA** Timer A PB6-uitvoer
- **PBONB** Timer B PB7-uitvoer
- **PORTA** Poort A
- **PORTB** Poort B
- **RMODA** One-shot-modus van Timer A
- **RMODB** One-shot-modus van Timer B
- **SDR** Schuifregistergegevensregister (schrijven start het verzenden)
- **SP** Schuifregister vol/leeg
- **SPMOD** Richting van de seriële poort
- **STRTA** Start van timer A
- **STRTB** Start van timer B
- **TA** Onderloop van timer A
- **TB** Onderloop van timer B
- **TIMERA** Teller van timer A (16-bits)

- **TIMERB** Teller van timer B (16-bits)
- **TOD50** 50/60Hz-selectie voor de TOD-klok
- **TODAMPM** PM-vlag van TOD
- **TODEDIT** Bewerken van het TOD-alarm
- **TODHOUR** Uren van TOD
- **TODJIF** Tienden van seconden van TOD
- **TODSEC** Seconden van TOD

CIA 6526-HYPERVISORREGISTERS

Naast de standaard CIA-registers die op de C64 en C65 beschikbaar zijn, biedt de MEGA65 een extra reeks registers die alleen zichtbaar zijn wanneer de computer zich in de hypervisormodus bevindt. Deze extra registers maken het mogelijk om de interne toestand van de CIA bij het bevriezen volledig uit te lezen, waardoor meer programma's na het bevriezen correct blijven functioneren. Zij zijn niet zichtbaar wanneer u de MEGA65 op normale wijze gebruikt en kunnen veilig worden genegeerd door programmeurs die de MEGA65 niet in de hypervisormodus programmeren.

CIA1-hypervisorregisters

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DC10	56336	TALATCH							
DC11	56337	TALATCH							
DC12	56338	TALATCH							
DC13	56339	TALATCH							
DC14	56340	TALATCH							
DC15	56341	TALATCH							
DC16	56342	TALATCH							
DC17	56343	TALATCH							
DC18	56344	IMFLG	IMSP	IMALRM	IMTB	TODJIF			
DC19	56345	TODSEC							
DC1A	56346	TODMIN							
DC1B	56347	TOD-AMPM	TODHOUR						
DC1C	56348	DD00-DELAY	ALRMJIF						
DC1D	56349	ALRMSEC							
DC1E	56350	ALRMMIN							
DC1F	56351	ALRM-AMPM	ALRMHOUR						

- **ALRMAMPM** AM/PM-vlag van het TOD-alarm
- **ALRMHOUR** TOD-alarmwaarde voor uren
- **ALRMJIF** TOD-alarmwaarde voor tienden van seconden (eigenlijk alle 8 bits)
- **ALRMMIN** TOD-alarmwaarde voor minuten
- **ALRMSEC** TOD-alarmwaarde voor seconden
- **DD00DELAY** Schakel het vertragen van schrijfbewerkingen naar $DD00 met 3 cycli in zodat dit overeenkomt met de timing van de echte 6502
- **IMALRM** Interruptmasker voor het TOD-alarm
- **IMFLG** Interruptmasker voor de FLAG-lijn
- **IMSP** Interruptmasker voor het schuifregister (seriële poort)
- **IMTB** Interruptmasker voor Timer B
- **TALATCH** Latchwaarde van Timer A (16 bit)
- **TODAMPM** TOD-vlag voor AM/PM
- **TODHOUR** TOD-waarde voor uren
- **TODJIF** TOD-waarde voor tienden van seconden
- **TODMIN** TOD-alarmwaarde voor minuten
- **TODSEC** TOD-alarmwaarde voor seconden

CIA2-hypervisorregisters

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DD10	56592	TALATCH							
DD11	56593	TALATCH							
DD12	56594	TALATCH							
DD13	56595	TALATCH							
DD14	56596	TALATCH							
DD15	56597	TALATCH							
DD16	56598	TALATCH							
DD17	56599	TALATCH							
DD18	56600	IMFLG	IMSP	IMALRM	IMTB	TODJIF			
DD19	56601	TODSEC							
DD1A	56602	TODMIN							
DD1B	56603	TOD-AMPM	TODHOUR						
DD1C	56604	DD00-DELAY	ALRMJIF						
DD1D	56605	ALRMSEC							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
DD1E	56606	ALRMMIN							
DD1F	56607	ALRM-AMPM	ALRMHOUR						

- **ALRMAMPM** AM/PM-vlag van het TOD-alarm
- **ALRMHOUR** Urenwaarde van het TOD-alarm
- **ALRMJIF** Tienden van seconden-waarde van het TOD-alarm (in feite alle 8 bits)
- **ALRMMIN** Minutenwaarde van het TOD-alarm
- **ALRMSEC** Secondenwaarde van het TOD-alarm
- **DD00DELAY** Uitstel van schrijfbewerkingen naar $DD00 met 3 cycli om de timing van een echte 6502 te benaderen
- **IMALRM** Interruptmasker voor het TOD-alarm
- **IMFLG** Interruptmasker voor de FLAG-lijn
- **IMSP** Interruptmasker voor het schuifregister (seriële poort)
- **IMTB** Interruptmasker voor timer B
- **TALATCH** Latchwaarde van timer A (16 bits)
- **TODAMPM** AM/PM-vlag van TOD
- **TODHOUR** Urenwaarde van TOD
- **TODJIF** Tienden van seconden-waarde van TOD
- **TODMIN** Minutenwaarde van het TOD-alarm
- **TODSEC** Secondenwaarde van het TOD-alarm

BIJLAGE T

4551 UART, GPIO en utiliteitscontroller

- C65 6551 UART-registers
- Algemene I/O & diverse interface-registers van de 4551

C65 6551 UART-REGISTERS

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D600	54784	DATA							
D601	54785	-				FRMERR	PTYERR	RXOVR-RUN	RXRDY
D602	54786	TXEN	RXEN	SYNCMOD		CHARSZ		PTYEN	PTYEVEN
D603	54787	DIVISOR							
D604	54788	DIVISOR							
D605	54789	IMTXIRQ	IMRXIRQ	IMTXNMI	IMRXNMI	-			
D606	54790	IFTXIRQ	IFRXIRQ	IFTXNMI	IFRXNMI	-			

- **CHARSZ** Tekengrootte van de UART: 00=8, 01=7, 10=6, 11=5 bits per byte
- **DATA** Gegevensregister van de UART (lezen of schrijven)
- **DIVISOR** UART-baudratedeler (16 bit). Baudrate = 7.09375MHz / DIVISOR, tenzij de snelle UART-modus van de MEGA65 is ingeschakeld, in welk geval baudrate = 80MHz / DIVISOR
- **FRMERR** UART RX-framingfoutvlag (wissen door $D600 te lezen)
- **IFRXIRQ** UART-interruptvlag: IRQ op RX (nog niet geïmplementeerd op de MEGA65)
- **IFRXNMI** UART-interruptvlag: NMI op RX (nog niet geïmplementeerd op de MEGA65)
- **IFTXIRQ** UART-interruptvlag: IRQ op TX (nog niet geïmplementeerd op de MEGA65)
- **IFTXNMI** UART-interruptvlag: NMI op TX (nog niet geïmplementeerd op de MEGA65)
- **IMRXIRQ** UART-interruptmasker: IRQ op RX (nog niet geïmplementeerd op de MEGA65)
- **IMRXNMI** UART-interruptmasker: NMI op RX (nog niet geïmplementeerd op de MEGA65)
- **IMTXIRQ** UART-interruptmasker: IRQ op TX (nog niet geïmplementeerd op de MEGA65)
- **IMTXNMI** UART-interruptmasker: NMI op TX (nog niet geïmplementeerd op de MEGA65)
- **PTYEN** UART-pariteit inschakelen: 1=ingeschakeld
- **PTYERR** UART-vlag voor RX-pariteitsfout (wissen door $D600 te lezen)

- **PTYEVEN** UART-pariteit: 1=even, 0=oneven
- **RXEN** UART ontvangen inschakelen
- **RXOVRRUN** UART-vlag voor RX-overrun (wissen door $D600 te lezen)
- **RXRDY** UART-vlag dat RX-byte gereed is (wissen door $D600 te lezen)
- **SYNCMOD** UART-vlaggen voor synchronisatiemodus (00=RX & TX beide asynchroon, 01=RX synchroon, TX asynchroon, 1x=TX synchroon, RX asynchroon (ongebruikt op de MEGA65)
- **TXEN** UART verzenden inschakelen

ALGEMENE I/O & DIVERSE INTERFACE-REGISTERS VAN DE 4551

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D609	54793	-							UFAST
D60A	54794	KEY-QUEUE	MODKEY-CAPS	MODKEY-SCRL	MODKEY-ALT	MODKEY-MEGA	MOD-KEYCTRL	MOD-KEYRSHFT	MOD-KEYLSHFT
D60B	54795	OSKZEN	OSKZON	PORTF					
D60C	54796	PORTFDDR		PORTFDDR					
D60D	54797	HDSCL	HDSDA	SDBSH	SDCS	SDCLK	SDDATA	RST41	CONN41
D60E	54798	BASHDDR							
D60F	54799	ACCES-SKEY	OSKDIM	REALHW	-			KEYUP	KEYLEFT
D610	54800	ASCIIKEY							
D611	54801	MDISABLE	MCAPS	MSCRL	MALT	MMEGA	MCTRL	MRSHFT	MLSHFT
D612	54802	LJOYB	LJOYA	JOYSWAP	OSKDE-BUG	-			
D613	54803	KEYMATRIXPEEK							
D614	54804	KEYMATRIXSEL							
D615	54805	OSKEN	VIRTKEY1						
D616	54806	OSKALT	VIRTKEY2						
D617	54807	OSKTOP	VIRTKEY3						
D618	54808	KSCNRATE							
D619	54809	PETSCIIKEY							
D61A	54810	SYSCTL							
D61D	54813	KEYLED-ENA	KEYLEDREG						
D61E	54814	KEYLEDVAL							
D620	54816	POTAX							
D621	54817	POTAY							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D622	54818	POTBX							
D623	54819	POTBY							
D625	54821	J21L							
D626	54822	J21H							
D627	54823	J21LDDR							
D628	54824	BOARDMINOR				J21HDDR			
D629	54825	M65MODEL							

- **ACCESSKEY** Toegankelijke toetsenbordinvoer inschakelen via de vuurknop van joystickpoort 2
- **ASCIIKEY** Bovenkant van de typgebeurteniswachtrij als ASCII. Schrijven om de gebeurtenis te wissen en de volgende gereed te maken.
- **BASHDDR** Data Direction Register (DDR) voor de $D60D-bitbashingpoort.
- **BOARDMINOR** PCB-minorrevisie lezen (enkel R5+, anders worden nullen gelezen)
- **CONN41** Interne 1541-stationskoppeling (1=het interne 1541-station op de IEC-bus aansluiten)
- **HDSCL** HDMI I2C-besturingsinterface SCL-klok
- **HDSDA** HDMI I2C-besturingsinterface SDA-datalijn
- **J21H** J21-pinnen 11 - 14 invoer/uitvoerwaarden
- **J21HDDR** J21-pinnen 11 - 14 data direction register
- **J21L** J21-pinnen 1 - 6, 9 - 10 invoer/uitvoerwaarden
- **J21LDDR** J21-pinnen 1 - 6, 9 - 10 data direction register
- **JOYSWAP** Joystickpoorten 1 & 2 verwisselen
- **KEYLEDENA** Besturing van de toetsenbord-LED's inschakelen
- **KEYLEDREG** Registerkeuze voor de toetsenbord-LED's (R,G,B-kanalen x 4 = 0 tot 11)
- **KEYLEDVAL** Waarde van het toetsenbord-LED-register (enkel schrijven)
- **KEYLEFT** De C65 Cursor links-toets rechtstreeks lezen
- **KEYMATRIXPEEK** 8-bits segment van de gecombineerde toetsenbordmatrix (LEZEN)
- **KEYMATRIXSEL** Bepaalt welk 8-bits segment van de gecombineerde toetsenbordmatrix gelezen wordt.

- **KEYQUEUE** 1 = De typgebeurteniswachtrij is niet leeg. Schrijf 0 naar deze bit om de wachtrij te spoelen.
- **KEYUP** De C65 Cursor omhoog-toets rechtstreeks lezen
- **KSCNRATE** Fysieke scanfrequentie van het toetsenbord ($00=50MHz, $FF=~200KHz)
- **LJOYA** Roteert de invoer van joystick A met 180 graden (voor linkshandig gebruik)
- **LJOYB** Roteert de invoer van joystick B met 180 graden (voor linkshandig gebruik)
- **M65MODEL** MEGA65-model-ID. Kan worden gebruikt om het model van de MEGA65 te bepalen waarop een programma wordt uitgevoerd, bijv. om aanraakbediening op de MEGAphone in te schakelen.
- **MALT** Toestand van de ALT-toets (direct; alleen lezen).
- **MCAPS** Toestand van de CAPS LOCK-toets (direct; alleen lezen).
- **MCTRL** Toestand van de CTRL-toets (direct; alleen lezen).
- **MDISABLE** Schakelt de modificatietoetsen uit.
- **MLSHFT** Toestand van de linker SHIFT-toets (direct; alleen lezen).
- **MMEGA** Toestand van de MEGA/C=-toets (direct; alleen lezen).
- **MODKEYALT** Toestand van de ALT-toets bovenaan de wachtrij van typegebeurtenissen. 1 = ingedrukt tijdens de gebeurtenis.
- **MODKEYCAPS** Toestand van de CAPS LOCK-toets bovenaan de wachtrij van typegebeurtenissen. 1 = ingedrukt tijdens de gebeurtenis.
- **MODKEYCTRL** Toestand van de CTRL-toets bovenaan de wachtrij van typegebeurtenissen. 1 = ingedrukt tijdens de gebeurtenis.
- **MODKEYLSHFT** Toestand van de linker SHIFT-toets bovenaan de wachtrij van typegebeurtenissen. 1 = ingedrukt tijdens de gebeurtenis.
- **MODKEYMEGA** Toestand van de MEGA/C=-toets bovenaan de wachtrij van typegebeurtenissen. 1 = ingedrukt tijdens de gebeurtenis.
- **MODKEYRSHFT** Toestand van de rechter SHIFT-toets bovenaan de wachtrij van typegebeurtenissen. 1 = ingedrukt tijdens de gebeurtenis.
- **MODKEYSCRL** Toestand van de NOSCRL-toets bovenaan de wachtrij van typegebeurtenissen. 1 = ingedrukt tijdens de gebeurtenis.
- **MRSHFT** Toestand van de rechter SHIFT-toets (direct; alleen lezen).
- **MSCRL** Toestand van de NOSCRL-toets (direct; alleen lezen).
- **OSKALT** Toont de alternatieve indeling van het schermtoetsenbord (gewoonlijk het nummerpad van de MEGA65-telefoon)

- **OSKDEBUG** Debugt de OSK-overlay (alleen schrijven)
- **OSKDIM** Dimt het achtergrondmateriaal achter het schermtoetsenbord licht of sterk
- **OSKEN** Schakelt de weergave van de samengestelde overlay van het schermtoetsenbord in
- **OSKTOP** 1=toont het schermtoetsenbord bovenaan, 0=toont het schermtoetsenbord onderaan het scherm.
- **OSKZEN** Toont de hardwarezoom van het gebied onder het eerste aanraakpunt voor het schermtoetsenbord
- **OSKZON** Toont altijd de hardwarezoom van het gebied onder het eerste aanraakpunt
- **PETSCIIKEY** Bovenaan de wachtrij van typegebeurtenissen als PETSCII. Schrijf erin om de gebeurtenis te wissen en gereed te maken voor de volgende.
- **PORTF** PMOD-poort A op het FPGA-bord (gegevens) (alleen Nexys4-borden)
- **PORTFDDR** PMOD-poort A op het FPGA-bord (DDR)
- **POTAX** Leest de paddle X van poort A, zonder dat u met de SID/CIA-instellingen hoeft te rommelen.
- **POTAY** Leest de paddle Y van poort A, zonder dat u met de SID/CIA-instellingen hoeft te rommelen.
- **POTBX** Leest de paddle X van poort B, zonder dat u met de SID/CIA-instellingen hoeft te rommelen.
- **POTBY** Leest de paddle Y van poort B, zonder dat u met de SID/CIA-instellingen hoeft te rommelen.
- **REALHW** Zet op 1 indien de MEGA65 op echte hardware wordt uitgevoerd, zet op 0 indien geëmuleerd (Xemu) of gesimuleerd (ghdl).
- **RST41** Interne reset van het 1541-schijfstation (1=reset, 0=operationeel)
- **SDBSH** Bitbash-modus van de SD-kaart inschakelen
- **SDCLK** SD-kaart SCLK
- **SDCS** SD-kaart CS_BO
- **SDDATA** SD-kaart MOSI/MISO
- **SYSCTL** Systeembeheervlagen (doelapparaatspecifiek)
- **UFAST** UART-baudklokbron van de C65: 1 = 7,09375 MHz, 0 = 80 MHz (VIC-IV-pixelpitch)
- **VIRTKEY1** Op $7F zetten voor geen toetsaanslag; geef anders het virtuele toetsaanslag aan

- **VIRTKEY2** Op $7F zetten voor geen toetsaanslag; geef anders de 2e virtuele toetsaanslag aan
- **VIRTKEY3** Op $7F zetten voor geen toetsaanslag; geef anders de 3e virtuele toetsaanslag aan

BIJLAGE U

45E100 Fast Ethernet-controller

- Overzicht
- In het geheugen gemapte registers
- Voorbeeldprogramma's

OVERZICHT

De 45E100 is een eenvoudige Fast Ethernet-controller die speciaal voor de MEGA65 en voor 8-bit computers in het algemeen is ontworpen. Naast het ondersteunen van 100 Mbit Fast Ethernet verschilt hij radicaal van andere Ethernet-controllers, zoals de RR-NET.

De 45E100 heeft 32 ontvangstbuffers op MEGA65 R3- en latere borden (andere borden kunnen er minder hebben, bijvoorbeeld vier op de Nexys4 DDR), waardoor tot 31 frames ontvangen kunnen worden terwijl een ander wordt verwerkt, of waardoor interrupts minder frequent verwerkt hoeven te worden. Deze ontvangstbuffers kunnen in het geheugen gemapt worden en ook direct benaderd worden via de DMA-controller van de MEGA65. Samen met automatische CRC32-controle bij ontvangst en automatische CRC32-generatie bij verzending, verlagen deze mogelijkheden de belasting van de processor aanzienlijk en maken ze het schrijven van programma's met Ethernet-ondersteuning veel eenvoudiger.

De 45E100 ondersteunt tevens echte full-duplex-werking met 100 Mbit per seconde, waardoor de totale bidirectionele doorvoer de 100 Mbit per seconde kan overschrijden. Het MAC-adres is softwarematig configureerbaar en promiscue modus wordt ondersteund, evenals het individueel aansturen van de ontvangst van broadcast- en multicast Ethernet-frames.

De 45E100 ondersteunt tevens zowel verzend- als ontvangstinterrupts, wat de prestaties in de praktijk aanzienlijk verbetert. Indien bijzonder lage latentie vereist is, kan de verzending van het huidige Ethernet-frame ook onmiddellijk afgebroken worden, zodat een frame met een hogere prioriteit onmiddellijk verstuurd kan worden. Deze mogelijkheden samen maken roundtriplatenties van minder dan een milliseconde mogelijk, wat vooral waardevol is voor interactieve toepassingen, zoals netwerkspellen voor meerdere spelers.

Verschillen met de RR-NET en vergelijkbare oplossingen

De RR-NET en andere Ethernet-controllers voor de serie 8-bit-homecomputers van de Commodore™ gebruiken doorgaans een Ethernet-controller die voor 16-bit pc's is ontworpen maar ook een zogenaamde "8-bit-modus" ondersteunt, die kampt met een aantal nadelen. Deze nadelen omvatten het ontbreken van werkende interrupts evenals processorintensieve toegang tot de Ethernet-framebuffers. Het gebrek aan interrupts dwingt programma's ertoe te pollen om de aankomst van nieuwe Ethernet-frames te controleren. Dit, gecombineerd met de complexiteit van het benaderen van de buffers, resulteert in een Ethernet-interface die zeer traag is en waarvan de praktijkdoorvoer aanzienlijk lager is dan de theoretische 10 Mbit per seconde. Zelfs een Commodore 64 met REU haalt geen snelheden boven enkele tientallen kilobytes per seconde.

Daarentegen ondersteunt de 45E100 zowel RX- (Ethernet-frame ontvangen) als TX- (klaar om te verzenden) interrupts, waarbij de processor het apparaat niet langer hoeft te pollen. Omdat de 45E100 RX-interrupts ondersteunt, is er geen behoefte aan grote

aantallen ontvangstbuffers, en daarom heeft de 45E100 slechts een klein aantal RX-buffers nodig om zeer hoge prestatieniveaus te bereiken.

Bovendien ondersteunt de 45E100 directe geheugentoegang tot de Ethernet-framebuffers, wat aanzienlijk efficiënter benaderen mogelijk maakt, ook via DMA. Met de geïntegreerde DMA-controller van de MEGA65 is het heel goed mogelijk transfersnelheden van enkele megabytes per seconde te bereiken – ongeveer 100 keer sneller dan de RR-NET.

Werking: het ontvangen van frames

De 45E100 is eenvoudig te bedienen: om het ontvangen van Ethernet-frames te beginnen, hoeft de programmeur alleen de RST- en TXRST-bits (bit 0 van register $D6E0) te wissen om de Ethernet-controller te resetten en vervolgens deze bits op 1 te zetten om de controller uit de resettoestand te bevrijden. De controller onderhandelt vervolgens automatisch een verbinding op de hoogst beschikbare snelheid, doorgaans 100 Mbit, full-duplex.

Indien u eenvoudig door pollen wilt controleren of er Ethernet-frames zijn aangekomen, controleer dan de RXQ-bit (bit 5 van $D6E1). Indien deze gezet is, is er ten minste één frame ontvangen. Om het volgende ontvangen frame te benaderen, schrijf $01 naar $D6E1 en vervolgens $03 naar $D6E1. Hiermee draait u de ring van ontvangstbuffers, zodat het volgende ontvangen frame door de processor benaderd kan worden. De ontvangstbuffer die voorheen door de processor benaderd kon worden, wordt als vrij gemarkeerd, en de 45E100 zal deze gebruiken om een volgend Ethernet-frame te ontvangen wanneer dat nodig is.

Omdat de 45E100 meerdere ontvangstbuffers heeft, is het mogelijk om meerdere frames achter elkaar te verwerken door deze procedure te volgen. Indien alle ontvangstbuffers ontvangen frames bevatten en de processor deze niet heeft aanvaard, dan wordt het RXBLKD-signaal geactiveerd, zodat de processor weet dat verdere ontvangen frames verloren zullen gaan. Programmeurs moeten voorzichtig zijn om deze situatie te vermijden. Omdat de 45E100 ontvangstinterrupts ondersteunt, is dit doorgaans eenvoudig te beheren – onderschat echter niet hoe vaak Ethernet-frames op een 100 Mbit Fast Ethernet-verbinding kunnen binnenkomen: wanneer een zender een continue stroom van frames van minimale lengte stuurt, kunnen deze elke 6 microseconden aankomen! Hoewel het onwaarschijnlijk is dat u met zo'n hoge ontvangstfrequentie te maken krijgt, moet u er rekening mee houden dat pakketten ten minste elke milliseconde verwerkt moeten worden. Met name een CIA- of raster-IRQ die eenmaal per frame optreedt, kan ertoe leiden dat enkele pakketten verloren gaan wanneer er meer dan drie in een videoframe van 16-20 ms aankomen. Het RXBLKD-signaal kan gebruikt worden om te bepalen of deze situatie zich waarschijnlijk heeft voorgedaan. Let er echter op dat het alleen aangeeft wanneer alle ontvangstbuffers bezet zijn, en niet of er nog extra frames arriveerden terwijl er geen vrije ontvangstbuffers beschikbaar waren.

De ontvangstbuffers zijn elk 2 KB groot en kunnen slechts één ontvangen Ethernet-frame tegelijk bevatten. Dit verschilt van enkele Ethernet-controllers die hun totale ontvangstbuffergeheugen als één simpele ringbuffer gebruiken. De reden hiervoor is

het mechanisme voor programmeurs zo eenvoudig mogelijk te houden. Door de vaste buffers kan de controller de ontvangen Ethernet-frames elke keer op exact dezelfde locatie in het geheugen mappen, waardoor het mogelijk is veel eenvoudigere ontvangstprogramma's te schrijven, omdat de locatie van de ontvangen Ethernet-frames constant verondersteld kan worden.

De structuur van een ontvangstbuffer met een Ethernet-frame is vrij eenvoudig: de eerste twee bytes geven de lengte van het ontvangen frame aan. Het frame volgt dan onmiddellijk. De effectieve Maximum Transport Unit (MTU)-lengte is 2042 bytes, aangezien de laatste vier bytes bezet zijn door de CRC32-controlesom van het ontvangen Ethernet-frame. De indeling van de ontvangstbuffers is als volgt:

HEX	DEC	Lengte	Omschrijving
0000	0	1	Het lage byte van de lengte van het ontvangen Ethernet-frame.
0001	1	1	Bits 0-2 bevatten de bovenste drie bits van de lengte van het ontvangen Ethernet-frame. Bit 3 is altijd 0. Bit 4 is gezet wanneer het ontvangen Ethernet-frame een multicast-frame is. Bit 5 is gezet wanneer het een broadcast-frame betreft. Bit 6 is gezet wanneer het bestemmingsadres van het frame overeenkomt met het geprogrammeerde MAC-adres van de 45E100. Bit 7 is gezet wanneer de CRC32-controle voor het ontvangen frame is mislukt, d.w.z. dat het frame of is afgekapt of tijdens de verzending beschadigd is geraakt.
0002 - 07FB	2 - 2,043	2,042	Het ontvangen frame. Frames korter dan 2042 bytes beginnen op offset 2.
07FC - 07FF	2,044 - 2,047	4	Gereserveerde ruimte voor het bewaren van de CRC32-code tijdens het ontvangen. De CRC32-code bevindt zich echter altijd direct na het ontvangen frame en neemt deze ruimte dus alleen in wanneer het ontvangen frame langer is dan 2038 bytes.

Vanwege het zeer hoge tempo waarin Fast Ethernet-frames kunnen worden ontvangen, dient een programmeur gebruik te maken van de ontvangstinterrupt-voorziening, die wordt ingeschakeld door RXQEN (bit 7 van $D6E1) te zetten. Polling is als alternatief mogelijk, maar wordt bij de 45E100 niet aangeraden, omdat bij de 100 Mbit-snelheid van Fast Ethernet pakketten om de 5 microseconden kunnen aankomen. Gelukkig is het mogelijk om dergelijke hoge gegevenssnelheden bij te houden in de volledige snelheidsmodus van 40 MHz van de MEGA65 en met behulp van de DMA-kopieerfunctie van 20 MB per seconde.

Toegang tot de Ethernet-framebuffers

Anders dan bij de RR-NET kunnen de Ethernet-framebuffers van de 45E100 in het geheugen worden gemapt, waardoor snelle toegang via DMA of via assembleertaalprogramma's mogelijk is. Het is eveneens mogelijk de buffers met enige voorzichtigheid vanuit BASIC te benaderen.

De frame-buffers kunnen ofwel worden benaderd vanaf hun natuurlijke locatie in de uitgebreide adresruimte van de MEGA65 op adres $FFDE800 - $FFDEFFF, ofwel kunnen ze worden gemapt in de normale C64/C65-I/O-adresruimte op $D800 - $DFFF. Hierbij is voorzichtigheid geboden, aangezien het mappen van de Ethernet-framebuffers in de I/O-adresruimte ertoe leidt dat alle andere I/O-apparaten gedurende die tijd niet beschikbaar zijn. Om die reden dienen CIA-gebaseerde interrupts te worden uitgeschakeld voordat u dit doet, of u nu BASIC of machinetaal gebruikt. Voor het programmeren in assembleertaal of machinetaal wordt daarentegen aangeraden de natuurlijke locatie te gebruiken en dit geheugengebied te benaderen via één van de drie mechanismen voor toegang tot de uitgebreide adresruimte, welke worden beschreven in Hoofdstuk/Bijlage K op bladzijde K-10.

De methode om interrupts uit te schakelen verschilt al naar gelang de context waarin een programma wordt geschreven. Voor programma's die worden geschreven met BASIC 2 van de C64-modus werkt het volgende:

```
POKE56333,127: REM SCHAKEL TIMER-IRQS VAN CIA UIT
```

Terwijl voor BASIC 65 van de MEGA65 in plaats daarvan het volgende moet worden gebruikt, omdat een VIC-III-rasterinterrupt wordt gebruikt in plaats van een op CIA gebaseerde timer-interrupt:

```
POKE53274,0: REM SCHAKEL RASTER-IRQS VAN VIC-II/III/IV UIT
```

Wanneer dit eenmaal is gebeurd, kan de I/O-context voor de Ethernet-controller worden geactiveerd door $45 (69 in decimaal, gelijk aan het teken 'E' in PETSCII) en $54 (84 in decimaal, gelijk aan het teken 'T' in PETSCII) in het KEY-register van de VIC-IV ($D02F, 53295 in decimaal) te schrijven, bijvoorbeeld:

```
POKE53295,ASC("E"):POKE53295,ASC("T")
```

Op dit punt kan de Ethernet-RX-buffer worden gelezen vanaf locatie $D800 (55296 in decimaal) en kan in de TX-buffer op hetzelfde adres worden geschreven. Raadpleeg "Werking: het ontvangen van frames" hierboven voor nadere uitleg hierover.

Wanneer u klaar bent met het benaderen van de Ethernet-framebuffer, kunt u de normale C64-, C65- of MEGA65-I/O-context herstellen door in het KEY-register van de VIC-III/IV te schrijven. In de meeste gevallen is het het meest zinvol om terug te keren naar de I/O-context van de MEGA65 door eerst $47 (71 in decimaal) en vervolgens $53 (83 in decimaal) in het KEY-register te schrijven, bijvoorbeeld:

```
POKE53295,ASC("G"):POKE53295,ASC("S")
```

Tot slot dient u vervolgens de interrupts weer in te schakelen, hetgeen opnieuw afhangt van de vraag of u programmeert vanuit de C64- of de C65-modus. Voor de C64-modus:

```
POKE56333,129
```

Voor de C65-modus zou dit zijn:

```
POKE53274,129
```

Werking: het verzenden van frames

Het verzenden van frames is eveneens eenvoudig: het programma moet het te verzenden frame eenvoudigweg in de zendbuffer laden, de lengte ervan in de TXSZLSB- en TXSZMSB-registers schrijven en vervolgens $01 in het COMMAND-register schrijven. Het frame begint daarna met verzenden, zodra de zender inactief is. Het is niet nodig een Ethernet-CRC32-veld te berekenen en toe te voegen, aangezien de 45E100 dit automatisch doet.

In tegenstelling tot bij de ontvanger is er voor de zender slechts één frame-buffer (dit kan in een toekomstige revisie worden gewijzigd). Dit betekent dat u het volgende frame niet kunt voorbereiden totdat het vorige frame reeds is verzonden. Dit vermindert de maximale gegevensdoorvoer enigszins, in ruil voor een zeer eenvoudige architectuur.

Houd er tevens rekening mee dat de zendbuffer vanuit de processorbusinterface uitsluitend schrijfbaar is. Dit betekent dat u de inhoud van de zendbuffer niet direct kunt lezen, maar dat u de waarden "blind" moet laden. Tot slot stelt de 45E100 u in staat om Ethernet-frames te verzenden.

Geavanceerde functies

Naast het functioneren als een eenvoudige en efficiënte Ethernet-framezender en -ontvanger, beschikt de 45E100 over een aantal geavanceerde functies die hier worden beschreven.

Broadcast- en multicast-verkeer en promiscue modus

De 45E100 ondersteunt filtering op basis van het Ethernet-bestemmingsadres, d.w.z. het MAC-adres. Standaard worden alleen frames ontvangen waarvan het Ethernet-bestemmingsadres overeenkomt met het Ethernet-adres dat in de registers MACADDR1 – MACADDR6 is geprogrammeerd. Indien de MCST-bit is gezet, worden echter ook multicast-Ethernet-frames ontvangen. Evenzo staat het instellen van de BCST-bit alle broadcast-frames toe, d.w.z. frames met het MAC-adres ff:ff:ff:ff:ff:ff. Tenslotte wordt het filter volledig uitgeschakeld indien de NOPROM-bit is gewist; de 45E100 ontvangt dan alle geldige Ethernet-frames.

Foutopsporings- en diagnosemogelijkheden

De 45E100 ondersteunt tevens een aantal functies om bij te dragen aan het opsporen van Ethernet-problemen. Indien de NOCRC-bit is gezet, worden ook Ethernetframes met ongeldige CRC32-waarden ontvangen. Dit kan helpen bij het opsporen van defecte Ethernet-apparaten in een netwerk.

Indien de STRM-bit is gezet, verzendt de Ethernet-zender een continue stroom van foutopsporingsframes via een speciale logging-interface met hoge bandbreedte. Standaard genereert de 45E100 een stroom van Ethernet-frames van ongeveer 2200 bytes, die gecomprimeerde video bevatten die wordt aangeboden door een VIC-IV of compatibele videocontroller die de video-over-Ethernet-interface van de MEGA65 ondersteunt. Door een aangepaste decoder voor deze stroom van Ethernet-frames te schrijven, is het mogelijk een extern beeldscherm van de MEGA65 via Ethernet te realiseren. Zo'n extern beeldscherm kan bijvoorbeeld worden gebruikt om de weergave van een MEGA65 digitaal vast te leggen.

De grootte en inhoud van de foutopsporingsframes kunnen worden beheerd door specifieke waarden in het COMMAND-register te schrijven. Het schrijven van $F1 staat de selectie toe van frames met een lengte van ongeveer 1200 bytes. Hoewel dit de prestaties van de foutopsporings- en streamingmogelijkheden vermindert, maakt het de ontvangst van deze frames mogelijk op systemen waarvan de Ethernet-controllers niet kunnen worden geconfigureerd om frames van ongeveer 2200 bytes te ontvangen.

Indien de STRM-bit is gezet en bit 2 van $D6E1 ook is gezet, wordt in plaats daarvan een gecomprimeerd logboek van de door de 45GS02-processor uitgevoerde instructies gestreamd, mits een compatibele processor op deze interface is aangesloten. Dit mechanisme oefent tegendruk uit en vertraagt de 45GS02-processor zodat de instructiegegevens kunnen worden uitgezonden. Doorgaans beperkt dit de snelheid van de aangesloten 45GS02-processor tot ongeveer 5 MHz, afhankelijk van de specifieke instructiemix.

Opmerking: de status van bit 2 van $D6E1 kan momenteel niet direct worden uitgelezen. Dit wordt mogelijk in een toekomstige revisie gecorrigeerd.

Tenslotte maakt het inschakelen van de videofunctionaliteit ook de ontvangst van synthetische toetsenbordgebeurtenissen via Ethernet mogelijk. Deze worden aangeboden aan de Keyboard Complex Interface Adapter (KCIA) van de MEGA65, waardoor volledige interactie op afstand met een MEGA65 via de Ethernet-interface mogelijk is. Deze functie is primair bedoeld voor ontwikkelingsdoeleinden.

IN HET GEHEUGEN GEMAPTE REGISTERS

De 45E100 Fast Ethernet-controller is een MEGA65-specifieke voorziening. Hij is daarom alleen beschikbaar in de MEGA65 I/O-context. Deze wordt ingeschakeld

door $47 en vervolgens $53 te schrijven naar het VIC-IV-register $D02F. Bij het programmeren in BASIC kan dit als volgt worden gedaan:

```
POKE53295,ASC("G"):POKE53295,ASC("S")
```

De 45E100 Fast Ethernet-controller heeft de volgende registers

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D02F	53295	KEY							
D6E0	55008	TXIDLE	RXBLKD	-	RCENAB-LED	DRXDV	DRXD	TXRST	RST
D6E1	55009	RXQEN	TXQEN	RXQ	TXQ	STRM	RXBF		-
D6E2	55010	TXSZLSB							
D6E3	55011	TXSZMSB							
D6E4	55012	COMMAND							
D6E5	55013	RXPH		MCST	BCST	TXPH		NOCRC	NOPROM
D6E6	55014	MIIMPHY			MIIMREG				
D6E7	55015	MIIMVLSB							
D6E8	55016	MIIMVMSB							
D6E9	55017	MACADDR1							
D6EA	55018	MACADDR2							
D6EB	55019	MACADDR3							
D6EC	55020	MACADDR4							
D6ED	55021	MACADDR5							
D6EE	55022	MACADDR6							

- **BCST** Aanvaard broadcast-frames
- **COMMAND** Ethernet-commandoregister (alleen schrijven)
- **DRXD** Lees de ethernet-RX-bits die op dit moment op de draad staan
- **DRXDV** Lees of de ethernet-RX-gegevens geldig zijn (debug)
- **KEY** Schrijf $45 en daarna $54 om de buffers van de 45E100-ethernetcontroller naar $D000-$DFFF te mappen
- **MACADDRX** Ethernet-MAC-adres
- **MCST** Aanvaard multicast-frames
- **MIIMPHY** Ethernet-MIIM-PHY-nummer (gebruik 0 voor Nexys4 en 1 voor MEGA65-r1-PCB's)
- **MIIMREG** Ethernet-MIIM-registernummer
- **MIIMVLSB** Ethernet-MIIM-registerwaarde (LSB)
- **MIIMVMSB** Ethernet-MIIM-registerwaarde (MSB)
- **NOCRC** Schakel de CRC-controle voor ontvangen pakketten uit

- **NOPROM** Schakel de promiscue modus van ethernet uit
- **RCENABLED** (alleen lezen) inschakelstatus van de ethernet-afstandsbediening
- **RST** Schrijf 0 om de ethernetcontroller in reset te houden
- **RXBF** Aantal vrije ontvangstbuffers
- **RXBLKD** Geeft aan of ethernet-RX is geblokkeerd totdat RX-buffers vrijkomen
- **RXPH** Ethernet-RX-klokfase-instelling
- **RXQ** Ethernet-RX-IRQ-status
- **RXQEN** Schakel de ethernet-RX-IRQ in
- **STRM** Schakel het streamen van de CPU-instructiestroom of de VIC-IV-weergave via ethernet in
- **TXIDLE** De ethernet-zendzijde is inactief, d.w.z. er kan een pakket worden verzonden.
- **TXPH** Ethernet-TX-klokfase-instelling
- **TXQ** Ethernet-TX-IRQ-status
- **TXQEN** Schakel de ethernet-TX-IRQ in
- **TXRST** Schrijf 0 om het zendsubsysteem van de ethernetcontroller in reset te houden
- **TXSZLSB** TX-pakketgrootte (lage byte)
- **TXSZMSB** TX-pakketgrootte (hoge byte)

Waarden van het COMMAND-register

De volgende waarden kunnen in het COMMAND-register worden geschreven om de beschreven functies uit te voeren. In normaal gebruik is alleen het STARTTX-commando nodig, bijvoorbeeld door de volgende **POKE** uit te voeren:

```
POKE55012,1
```

HEX	DEC	Signaal	Omschrijving
00	0	STOPTX	Stop onmiddellijk met het verzenden van het huidige Ethernet-frame. De zender stuurt een gedeeltelijk verzonden frame, waardoor dit frame zeer waarschijnlijk verloren gaat.
01	1	STARTTX	Zend pakket
D0	208	RXNORMAL	Schakel de effecten van RXONLYONE uit

vervolgd ...

...vervolgd

HEX	DEC	Signaal	Omschrijving
D4	212	DEBUGVIC	Selecteer de VIC-IV-foutopsporingsstroom via Ethernet wanneer $D6E1.3 is gezet
DC	220	DEBUGCPU	Selecteer de CPU-foutopsporingsstroom via Ethernet wanneer $D6E1.3 is gezet
DE	222	RXONLYONE	Ontvang precies één Ethernet-frame en behoud alle signaaltoestanden (voor het opsporen van fouten in het Ethernet-subsysteem)
F1	241	FRAME1K	Selecteer frames van 1 KiB voor de video-/CPU-foutopsporingsstroom (voor ontvangers die MTU's groter dan 2 KiB niet ondersteunen)
F2	242	FRAME2K	Selecteer frames van 2 KiB voor de video-/CPU-foutopsporingsstroom, voor optimale prestaties.

VOORBEELDPROGRAMMA'S

Voorbeeldprogramma's voor de Ethernet-controller bevinden zich in de MEGA65 Core-repository op GitHub in de mappen src/tests en src/examples, in wisselende staat van voltooiing.

Indien u de Ethernet-controller wilt gebruiken voor TCP/IP-verkeer, kunt u de port van WeeIP naar de MEGA65 bekijken op https://github.com/mega65/mega65-weeip. De code die de Ethernet-controller aanstuurt, bevindt zich in `eth.c`.

BIJLAGE V

45IO27 multifunctionele I/O-controller

- Overzicht
- F011-compatibele schijfstationcontroller
- SD-kaartcontroller en F011-virtualisatiefuncties
- Touchpadinterface
- Ondersteunende functies voor audio
- Diverse I/O-functies

OVERZICHT

De 45IO27 is een multifunctionele I/O-controller die de functies van de F011-schijfstationcontroller van de C65 verenigt met de interface van de SD-kaartcontroller van de MEGA65 en een aantal andere uiteenlopende I/O-functies.

Elk van deze hoofdfuncties wordt in een afzonderlijke sectie van dit hoofdstuk behandeld.

F011-COMPATIBELE SCHIJFSTATIONCONTROLLER

De MEGA65-computer is een van de zeer weinige moderne computers die nog steeds volwaardige ondersteuning bieden voor magnetische schijfstations. Hij bevat een schijfstationcontroller die achterwaarts compatibel is met de F011D-schijfstationcontroller van de C65.

Anders dan de F011D ondersteunt de schijfstationcontroller van de MEGA65 HD- en ED-media. Net als het 1541-schijfstation ondersteunt deze ook variabele datatempo's, zodat een vastberaden gebruiker schijfformaten kan ontwikkelen die meer gegevens opslaan, robuuste kopieerbeveiligingsschema's bevatten of beide.

Ook andere schijfformaten kunnen worden gebruikt, zij het met wisselende inspanning, zoals het GCR-schema van de 1541 en de 1571 of het track-at-once-MFM-schema dat de Amiga™-familie van computers gebruikt. Dit is mogelijk dankzij de track-DMA-functionaliteit die de 45IO27 en de 45GS02-processor gezamenlijk bieden. Deze twee apparaten werken samen om ruwe fluxsignalen van en naar diskettes te lezen of te schrijven, waardoor vrijwel elk schijfformaat kan worden gelezen of gereproduceerd.

Voor Amiga™-diskettes bevat de 45IO27 bovendien experimentele speciale schakelingen die het lezen van dergelijke diskettes vergemakkelijken. Zo kan de 45IO27 de unieke sectorsynchronisatiesignalen herkennen die op Amiga™-diskettes worden gebruikt en de gegevenssectoren lezen. Hij haalt echter niet de 16 bytes besturingssysteemspecifieke gegevens per sector op en voert ook geen de-interleaving uit van de gescheiden vectoren van oneven en even bits ten opzichte van de volgorde waarin de Amiga™-computer ze op de schijf schrijft. Het gebruik van deze speciale schakelingen is optioneel. De track-DMA-functionaliteit kan dus ook worden gebruikt om Amiga™-diskettes te lezen.

Ondersteuning voor meerdere schijfstations

Net als de F011-schijfstationcontroller van de C65 ondersteunt de 45IO27 maximaal acht stations. Van die stations wordt aangenomen dat de eerste twee, station 0 en station 1, zijn aangesloten op een standaard 34-pins diskettekabel, dezelfde als in

standaard-pc's, dus met een draai in de kabel zodat er twee stations zonder jumpers kunnen worden gebruikt.

Zoals in latere secties wordt beschreven, kunnen de posities van station 0 en station 1 worden omgewisseld zonder dat de bekabeling hoeft te worden gewijzigd. Evenzo kan elk van de eerste twee stations verwijzen naar een echt schijfstation of naar een op een aangesloten SD-kaart opgeslagen D81-schijfimage, en kan het ook worden omgeleid naar de virtualisatiedienst voor schijfstations, zodat de sectortoegangen door een aangesloten computer kunnen worden afgehandeld, bijvoorbeeld als onderdeel van een comfortabele en efficiënte cross-development-omgeving.

De overige zes stations worden alleen ondersteund in combinatie met een toekomstige C1565-compatibele externe stationpoort.

Bewerkingen met gebufferde sectoren

De 45IO27 ondersteunt twee manieren om sectoren van een schijf te lezen: byte voor byte en via een in het geheugen ingemapte sectorbuffer.

Het mechanisme byte voor byte bestaat uit een lus die wacht totdat het DRQ-signaal actief wordt en daarna het databyte uit het DATA-register ($D087) leest.

Bij de methode met de in het geheugen ingemapte sectorbuffer wacht u totdat de BUSY-vlag wordt gewist. Wanneer deze gewist is, is de volledige sector gelezen en kunt u direct de sectorbuffer op $FFD6C00 - $FFD6DFF benaderen. Let erop dat het BUFSEL-signaal (bit 7 van $D689) is gewist, zodat voor andere programma's dan de Hypervisor de diskettesectorbuffer zichtbaar is in plaats van de SD-kaart-sectorbuffer. Dit komt doordat alleen de Hypervisor toegang heeft tot de volledige 4KB-bufferruimte van de SD-controller: gewone programma's zien ofwel de diskettesectorbuffer ofwel de SD-kaart-sectorbuffer, acht keer herhaald tussen $FFD6000 en $FFD6FFF.

Alternatief kan de sectorbuffer worden ingemapt op $DE00 - $DFFF, d.w.z. in het 4KB-I/O-gebied, door $81 naar het SD-commandoregister op $D680 te schrijven. Eventuele I/O-perifierieën die dit gebied anders gebruiken, zoals cartridges of REU-emulatie, worden dan verborgen. Deze functie kan weer worden uitgeschakeld door $82 naar het SD-commandoregister te schrijven. Net als bij de normale geheugentoewijzing van de sectorbuffer op $FFD6xxx, bepaalt het BUFSEL-signaal (bit 7 van $D689) voor software die niet in hypervisormodus draait of de FDC-sectorbuffer of de SD-kaart-sectorbuffer zichtbaar is. Let op: wanneer u de Matrix Mode / serial-monitor-interface gebruikt om de inhoud van de sectorbuffer te bekijken, gebeurt dat in de hypervisorcontext. Het BUFSEL-signaal wordt dan genegeerd en de volledige 4KB-buffer is zichtbaar.

De in het geheugen ingemapte sectorbuffer heeft als voordeel dat hij via DMA kan worden benaderd, waardoor zeer efficiënte kopieën mogelijk zijn. Bovendien kan een sector op de achtergrond worden geladen terwijl uw programma zich in de tussentijd met interessantere zaken bezighoudt.

Sectoren van een schijf lezen

Om een sector van een schijf te kunnen lezen, moet u een aantal stappen doorlopen. Volgt u deze aanwijzingen op, dan werkt uw code zowel met fysieke schijven als met D81-schijfimages die zich op de SD-kaart bevinden:

- Schakel eerst de motor in en selecteer het juiste station. De F011 ondersteunt maximaal acht fysieke stations, hoewel het zelden voorkomt dat er meer dan twee fysiek zijn aangesloten. Om de motor in te schakelen schrijft u $60 naar $D080. Schrijf daarna een SPINUP-commando ($20) naar $D081 en wacht totdat de BUSY-vlag (bit 7 van $D082) wordt gewist. Het station draait dan op snelheid en is klaar om aanvragen af te handelen.
- Selecteer vervolgens de juiste zijde van de schijf door de SIDE1-vlag (bit 3 van $D080) te zetten of te wissen. Dit heeft onmiddellijk effect.
- Gebruik daarna, afhankelijk van de behoefte, de step-in- en step-out-commando's (door $10 en $18 naar $D081 te schrijven) om de leeskop naar het juiste spoor te verplaatsen. Ook na elk commando hoort u te wachten totdat de BUSY-vlag (bit 7 van $D082) wordt gewist voordat u het volgende commando geeft.

  U kunt controleren of de leeskop zich op spoor 0 bevindt door de TRACK0-vlag te controleren, maar er is geen onfeilbare manier om te weten of u zich op enig ander specifiek spoor bevindt. U kunt de registers op $D6A3 - $D6A5 gebruiken om de spoor-, sector- en zijdewaarde van de laatste sectorheader te zien die onder de leeskop is gepasseerd en zo een geïnformeerde schatting te maken van het momenteel geselecteerde spoor. Dit werkt echter alleen voor echte schijven, omdat schijfimages niet onder de leeskop ronddraaien. Houd er ook rekening mee dat sporen sectoren kunnen bevatten waarvan de spoorwaarden in de sectorheaders opzettelijk of per ongeluk onjuist zijn.
- Laad als vierde stap het gewenste spoor-, sector- en zijdenummer in de TRACK-, SECTOR- en SIDE-registers ($D084, $D085 en $D086). De FDC is nu gereed om een sector te lezen.
- Ten vijfde moet u een passende leesopdrachtwaarde in $D081 schrijven. Normaal is dit $40 (64). Daarna wacht u tot het RDREQ-signaal ($D083, bit 7) hoog wordt, wat aangeeft dat de sector gevonden is. Vervolgens wacht u ofwel op elke keer dat DRQ hoog wordt en leest u in zo'n lus byte voor byte, ofwel wacht u tot de BUSY-vlag gewist is en de DRQ- en EQ-vlaggen hoog worden, wat aangeeft dat de volledige sector in de buffer is gelezen.

Automatische spoorcorrectie

De 45IO27 bevat tevens een "auto-tune"-functie voor sporen, die wordt ingeschakeld door bit 7 van $D696 te wissen. Die functie leest de sectorheaders om te bepalen waar de kop zich momenteel bevindt, en stapt de kop in of uit om zo het juiste spoor te bereiken. Auto-tune is standaard ingeschakeld.

Sector-scheefstand en de Target Any-modus

Het is ook het vermelden waard dat het TARGANY-signaal geactiveerd kan worden om de schijfstationcontroller te laten weten dat hij eenvoudig de volgende sector moet lezen die onder de kop passeert. Dit geldt alleen bij het gebruik van echte diskettes, waar het aanzienlijke voordeel biedt dat u de sectoren kunt lezen in de volgorde waarin ze zich op de schijf bevinden. Zo kunt u in één keer een volledig spoor lezen, zonder te hoeven wachten tot het indexgat voorbij is, of te hoeven weten welke sector als volgende onder de kop zal passeren.

De C65-DOS formatteert bijvoorbeeld diskettes met een scheefstandsfactor van 7, terwijl PC's een andere scheefstandsfactor kunnen gebruiken. Als u de scheefstandsfactor van de diskette niet kent, kunt u het lezen van de sectoren op het spoor in een suboptimale volgorde plannen. Dit kan leiden tot overdrachtsnelheden van slechts 5 sectoren per seconde, vergeleken met het optimale geval van 50 sectoren per seconde. Met de juiste sectorvolgorde, of door de target any-modus te gebruiken, is het dus mogelijk om ongeveer twee volledige sporen per seconde te lezen, d.w.z. twee zijden $\times$ twee sporen, oftewel ongeveer 20 KB/seconde op DD-diskettes, of het dubbele daarvan op HD-diskettes, rond 40 KB/seconde. Dit steekt zeer gunstig af tegen de laadsnelheid van de C65-DOS, die in de C64-modus doorgaans dichter bij 1 KB/sec ligt.

Schijfindeling en logische 1581-sectoren

Het 1581-schijfformaat is ongebruikelijk in die zin dat de fysieke sectoren op de schijf een andere grootte hebben dan de datablokken die het aan de gebruiker presenteert. Concreet gebruiken de diskettes sectoren van 512 bytes, terwijl de 1581- (en C65-) DOS datablokken van 256 bytes presenteren. In elke fysieke sector worden twee blokken opgeslagen. De fysieke spoornummers lopen ook van 0 tot 79, terwijl de logische spoornummers van de DOS van 1 tot 80 lopen. Fysieke sectoren zijn ook genummerd van 1 tot 10, terwijl logische bloknummers lopen van 0 tot 39.

Dit betekent dat u, als u een logische 1581-sector wilt vinden, moet weten in welke fysieke sector deze te vinden is. Om de fysieke sector te bepalen die een blok bevat, trekt u eerst één af van het spoornummer en deelt u vervolgens het sectornummer door twee. Logische sectoren 0 tot 19 van elk spoor bevinden zich in fysieke sectoren 1 tot 10 op de eerste zijde van de schijf. Logische sectoren 20 tot 39 bevinden zich in fysieke sectoren 1 tot 10 op de achterzijde van de schijf.

We kunnen dus een logisch spoor en sector t,s als volgt afbeelden op het fysieke spoor, de zijde en de sector:

$track = t - 1$

$sector = (s/2) + 1, IFFs < 20, ELSE = ((s - 20)/2) + 1$

$side = 0IFFsector < 20$

Het is ook het vermelden waard dat de 45IO27 van sporen voorbij spoor 80 kan lezen, mits het schijfstation daartoe in staat is. Bijna alle 3.5-inchschijfstations kunnen

ten minste één extra spoor lezen, omdat fabrikanten van diskettes historisch gezien informatie over de schijf opsloegen op het 81ste spoor. Naar onze ervaring kunnen bijna alle stations ook een 82ste spoor benaderen.

FD2000-diskettes

De CMD™ FD2000™-schijfstations met hoge dichtheid van 3.5" voor Commodore™-computers gebruiken een ongebruikelijke schijfindeling die heel anders is dan die van PC's: Ze gebruiken 10 sectoren, net zoals op 720KB-diskettes met dubbele dichtheid, maar verdubbelen de *sectorgrootte* van 512 bytes naar 1024 bytes. De 45IO27 ondersteunt deze grotere sectoren momenteel niet voor gebufferde sectorbewerkingen. Ten minste ondersteuning voor alleen-lezen is gepland; deze zal in de toekomst via een core-update worden toegevoegd. FD2000™- en FD4000™-diskettes kunnen echter wel gelezen en geschreven worden met behulp van de track-DMA-functie.

Voor het aanmaken van nieuwe diskettes ondersteunt de 45IO27 *bovendien* al schijven en stations met hoge dichtheid, met veel hogere capaciteiten dan de FD2000 kon bieden.

HD-diskettes en diskettes met variabele dichtheid

De 45IO27 ondersteunt variabele datatempo's, waardoor het mogelijk is HD-stations en -media te gebruiken, met een flexibele benadering van schijfformaten voor gebruikersexperimenten en een gemakkelijke omgang met formaten met hoge capaciteit voor softwaredistributie.

U wordt eigenlijk alleen beperkt door uw verbeeldingskracht, de beschikbare tijd en het beperkte aantal mensen dat nog geïnteresseerd is in het in hun computer plaatsen van een diskette!

Het standaardformaat voor hoge dichtheid (HD) is "1.44MB", met 18 sectoren per spoor over 80 sporen. Dit levert 80 sporen × 18 sectoren × 2 zijden = 2880 sectoren op. Aangezien elke sector 512 bytes groot is, komt dit overeen met 1440KB. Dit voert ons naar het interessante wonderland van "floppy-disk marketing megabytes", een verschijnsel dat lang voorafging aan fabrikanten van SD-kaarten en harde schijven die megabytes van 1000000 bytes gebruiken.

Vreemd genoeg werd voor diskettes de "megabyte" van 1024000 bytes gebruikt, d.w.z. "1MB" = 1KB × 1KB, een vreemde hybride van binaire en decimale conventies. Misschien kwam het doordat de vorige standaard 720KB was en men dacht dat mensen het vreemd zouden vinden als het dubbele van 720KB 1.41MB was en zouden klagen over de ontbrekende kilo-bytes. We blijven voor de consistentie met deze historische ongerijmdheid de floppy-disk-marketing-mega-byte van 1024KB = 1000KB gebruiken.

HD-diskettes zijn echter fundamenteel in staat om veel meer dan 1.44MB te bevatten. De FD2000 sloeg bijvoorbeeld 1.6MB op door dubbelgrote sectoren te gebruiken om het equivalent van 20 sectoren per spoor eruit te halen, en de Amiga ging verder door track-at-once-schrijven te gebruiken om er 22 sectoren per spoor in te passen. Beide

formaten gebruikten over alle sporen een constant datatempo en dus een constant aantal sectoren per spoor.

De omtrek van de sporen op een 3.5"-diskette varieert echter behoorlijk: Het binnenste spoor op een straal van ongeveer 25 mm heeft een lengte van 157 mm, terwijl het buitenste spoor op een straal van 40 mm een lengte van 251 mm heeft, 1.6 maal zo lang. Het 1.44MB-formaat is zo ontworpen dat de gegevens betrouwbaar worden opgeslagen op die kortere binnenste sporen. Dat betekent dat we 60% meer gegevens op het buitenste spoor zouden moeten kunnen plaatsen dan op het binnenste spoor, binnen een aantal voorwaarden van de wetten van de fysica, het ontwerp van de schijfstation-elektronica, de kwaliteit van het gebruikte medium en diverse andere hinderlijke zaken. Vanwege deze variabiliteit en onzekerheid ondersteunt de schijfstationcontroller van de MEGA65 een volledig variabel datatempo per spoor afzonderlijk.

Track Information Blocks

Om variabele datatempo's te ondersteunen gebruikt de 45IO27 de zogeheten *Track Information Blocks* (TIB's) die informatie bevatten over het datatempo en de codering die op het spoor gebruikt worden. Zo kunnen gebruikers experimenteren met verschillende dichtheden op verschillende sporen en tóch functioneren de diskettes automatisch voor gebufferde sectorbewerkingen.

Het Track Information Block wordt automatisch aangemaakt bij gebruik van de functie voor het automatisch formatteren van sporen, maar moet handmatig worden aangemaakt bij gebruik van ongebufferde opmaak. De TIB zelf bestaat uit de volgende gegevens:

1. 3× $A1-syncbytes (geschreven met het klokbyte $FB)
2. het $65-marker van het MEGA65 Track Information Block (geschreven met het klokbyte $FF, evenals alle volgende bytes in het blok)
3. het spoornummer
4. de datatempo-deler, in hetzelfde formaat als $D6A2, d.w.z. datatempo = 40.5 MHz / waarde.
5. de track-coderingsinformatie: bit 7 = Track-at-once-vlag, 1 = geen gaps tussen sectoren (Amiga-stijl), 0 = met gaps tussen sectoren (normaal), bit 6 = de gegevenscodering, 0 = MFM, 1 = RLL2,7. De overige bits zijn gereserveerd en moeten bij het schrijven 0 zijn.
6. het aantal sectoren, d.w.z. het aantal sectoren op de track.
7. CRC-byte 1, volgens het normale CRC-algoritme voor diskettes.
8. CRC-byte 2, volgens het normale CRC-algoritme voor diskettes.

Het Track Information Block wordt altijd geschreven met het datatempo voor een 720KB-diskette met dubbele dichtheid, zodat het op elke diskette aanwezig kan zijn. Het schrijven van het Track Information Block en van de start-of-track-gaps met het DD-datatempo zorgt er ook voor dat de kop bij zeer hoge datatempo's nog voldoende

tijd heeft om naar de schrijfmodus over te schakelen, waardoor een van de vele problemen wordt vermeden die ontstaan bij het schrijven van gegevens met zeer hoge datatempo's.

Indien u diskettes ongebufferd formatteert, is het de verantwoordelijkheid van de programmeur om het datatempo om te schakelen nadat het Track Information Block is geschreven en enkele bytes meer zijn geschreven, zodat de floppy-coderingspijplijn de laatste byte van het Track Information Block kan doorspoelen. Dit wordt allemaal automatisch afgehandeld wanneer u de automatische trackformatteringsfunctie gebruikt.

De opname van het TIB stelt gebruikers in staat om op verschillende stations en media te spelen en de mogelijkheden van verschillende datatempo's te verkennen, terwijl alles in alle MEGA65's toch automatisch leesbaar blijft, omdat het TIB de controller in staat stelt om naar het juiste datatempo en de juiste codering over te schakelen. Het is waarschijnlijk dat na verloop van tijd enigszins gestandaardiseerde formaten zullen ontstaan, zeer waarschijnlijk in het bereik van 2 MB tot 3.5 MB - en daarmee de capaciteit van ED-media in ED-stations benaderen, zonder die stations of media nodig te hebben.

Schijven formatteren

Het formatteren van schijven is nu mogelijk met de 45IO27, ongebufferd of volledig automatisch. Om een track te formatteren geeft u een van de volgende commando's aan $D081:

- $A0 - automatisch formatteren, met gaps tussen sectoren en schrijfvoorcompensatie uitgeschakeld.
- $A1 - handmatig formatteren, schrijfvoorcompensatie uitgeschakeld.
- $A4 - automatisch formatteren, met gaps tussen sectoren en schrijfvoorcompensatie ingeschakeld.
- $A5 - handmatig formatteren, schrijfvoorcompensatie ingeschakeld.
- $A8 - automatisch formatteren, Amiga-stijl track-at-once en schrijfvoorcompensatie uitgeschakeld.
- $AC - automatisch formatteren, Amiga-stijl track-at-once en schrijfvoorcompensatie ingeschakeld.

Handmatig formatteren wordt niet aanbevolen, tenzij u track-at-once-geformatteerde diskettes voor softwaredistributie aanmaakt, vanwege de relatieve complexiteit daarvan. Ook moeten bij de hogere datatempo's bytes zo vaak als elke 20 cycli aan de floppy-controller worden geleverd, wat aanzienlijke zorg vereist bij het schrijven van de formatteringsroutine. Zie voor meer informatie over het handmatig formatteren van tracks, evenals voor voorbeelden van handmatig formatteren, de C64 Specifications Manual of de C65 ROM-DOS-broncode.

De automatische modi daarentegen formatteren een track met één enkel commando en zijn daardoor veel eenvoudiger te gebruiken, en worden voor algemeen gebruik aanbevolen. Schrijfvoorcompensatie moet normaliter worden ingeschakeld, omdat die bij hogere datatempo's vereist is en bij lagere datatempo's geen problemen veroorzaakt.

Schrijfvoorcompensatie

Schrijfvoorcompensatie is een familie van algoritmen die wordt gebruikt bij het schrijven van signalen met een hoog datatempo naar diskettes. Zij dient om de voorspelbare component van de tijdsvariatie van magnetische opnamen te anticiperen en teniet te doen. Er zijn diverse bronnen van deze tijdsvariatie, die het onderwerp zijn geweest van promotieonderzoeken en van veel bedrijfseigen onderzoek door fabrikanten van harde schijven. Wat voor ons belangrijk is om te begrijpen, is dat aangrenzende pulsen (in feite magnetische ompolingen) naar elkaar toe worden gedrukt wanneer ze door langere pulsen worden omgeven, en de neiging hebben uit elkaar te spreiden wanneer ze door kortere pulsen worden omgeven.

Er zijn ook andere fascinerend complexe en moeilijk te voorspellen factoren, die zaken veroorzaken zoals de "negatieve verschuiving van midlange pulsen", de "inverse F-verdeling van pulsaankomsttijden" en wie weet wat nog meer. Die laten we echter over aan de fabrikanten van harde schijven. Wij beperken ons tot het door het gegevenspatroon veroorzaakte effect dat in de vorige alinea is beschreven.

De 45IO27 ondersteunt twee instelbare coëfficiënten voor kleine en grote correcties hierop, die met behulp van een interne opzoek (look-up) tabel worden toegepast. Dit wordt echter allemaal automatisch afgehandeld wanneer u schrijfvoorcompensatie inschakelt. Dit maakt datatempo's mogelijk die veel dichter bij de verwachte limiet van HD-media liggen, hoewel de werkelijke limiet vanwege de andere verschrikkingen van magnetische media-opnamen die hierboven zijn aangestipt, niet wordt bereikt.

Gebufferd sectorschrijven

De 45IO27 kan naar schijfimages schrijven die zich op de SD-kaart bevinden, of wanneer u gevirtualiseerde schijftoegang gebruikt.

Om een sector te schrijven doorloopt u een soortgelijk proces als bij het lezen, behalve dat u $84 in het commandobyt schrijft in plaats van $40. De $80 geeft een schrijfbewerking aan en de $04 activeert de schrijfvoorcompensatie. Dit is belangrijk bij het schrijven naar echte diskettes, vooral HD- en ED-diskettes. Schrijfvoorcompensatie zorgt ervoor dat bits iets te vroeg of iets te laat worden geschreven, met behulp van een algoritme dat modelleert hoe de magnetische domeinen op een schijf de neiging hebben te bewegen nadat ze zijn geschreven.

Indien u de sectorbuffer niet wilt gebruiken, maar tijdens de schrijfbewerking elk byte afzonderlijk tegelijk wilt leveren, moet u $01 aan de commandocode toevoegen. Dit wordt op de MEGA65 echter niet aanbevolen, omdat bij het schrijven naar de SD-kaart of bij het gebruik van gevirtualiseerde schijfimages de volledige sectorbewerking vanuit het perspectief van uw programma ogenblikkelijk kan plaatsvinden. Dat betekent

dat het in deze modus niet mogelijk is om gegevens betrouwbaar te leveren. Behalve dat het minder handig is, is het dus ook minder betrouwbaar.

Zodra een schrijfbewerking is geactiveerd, geeft het DRQ-signaal aan wanneer u het volgende byte moet leveren indien u byte voor byte schrijft. In overige gevallen wordt aangenomen dat u de sectorbuffer vooraf hebt gevuld met de volledige benodigde 512 bytes aan gegevens.

Om naar schijven te schrijven die Track Information Blocks bevatten, moet u eerst wachten tot het TIB is gelezen bij het wisselen van tracks. Dit doet u door te wachten tot $D6A9 (aantal sectoren per track uit het TIB) een waarde ongelijk aan nul bevat.

Floppy-track-DMA

Zoals eerder beschreven, verbindt de 45IO27 zich met de 45GS02-CPU, en in het bijzonder met de interne DMA-controller daarvan, om een eenvoudig mechanisme te bieden voor het lezen en schrijven van de ruwe magnetische fluxovergangen van diskettes. Naast het mogelijk maken van software die elk mogelijk schijfformaat kan lezen, maakt het ook het schrijven en dupliceren van vrijwel elk schijfformaat mogelijk. Wij zeggen 'vrijwel', omdat het mogelijk is dat diskettes worden geschreven met speciale machines die mogelijkheden hebben die die van het floppy-stationmechanisme van de MEGA65 te boven gaan. Fat tracks is hiervan een voorbeeld. Een ander voorbeeld is dat de magnetische fluxovergangen zodanig worden geplaatst en/of gespreid dat een normaal floppy-station ze niet precies kan reproduceren.

Floppy-track-DMA gebruiken

De volgende DMA-opties vertellen de DMAgic-controller om in plaats van een geheugenlocatie ruwe fluxwaarden te lezen of schrijven. In deze modus wacht de DMAgic-controller op het ontvangen van elk opeenvolgend pulsinterval van de 45IO27. De gegevens die hij aldus leest (of schrijft) zijn de duur tussen elke opeenvolgende fluxompoling, in eenheden van 50ns.

- $0D - Schrijf ruwe fluxintervallen. Stel de DMA in op COPY-modus om dit te gebruiken.
- $0E - Lees ruwe fluxintervallen en negeer onwaarschijnlijk lange intervallen. Stel de DMA in op FILL-modus om dit te gebruiken.
- $0F - Lees ruwe fluxintervallen en retourneer onwaarschijnlijk lange intervallen als $FF. Stel de DMA in op FILL-modus om dit te gebruiken.

Zie floppytest.c in de mega65-tools-repository van de MEGA65-githubsite op https://github.com/mega65-toolsvoor voorbeeldcode die sporen leest met behulp van deze functionaliteit.

De beperkingen van schijfstations begrijpen

Bij het schrijven van ruwe fluxovergangen moet u rekening houden met de beperkingen van standaard schijfstations. Er zijn drie belangrijke factoren waarmee rekening moet worden gehouden:

- Twee overgangen kunnen niet te dicht bij elkaar worden geplaatst. Het station weigert ze namelijk eenvoudigweg te schrijven als ze minder dan een zekere afstand uit elkaar liggen. Zelfs wanneer u ze toch weet te schrijven, voegt het leesfiltercircuit van het schijfstation pulsen die te dicht bij elkaar staan samen.
- Twee overgangen kunnen niet te ver uit elkaar worden geplaatst. In dat geval schrijft het station ze zonder bezwaar, maar het leesfiltercircuit raakt ongerust als het een tijdje geen puls ziet, begint te denken dat de achtergrondruis echte signalen zijn en introduceert valse pulsen in de leesstroom die in werkelijkheid niet op de schijf staan.
- Magnetische pulsen verplaatsen zich op de schijf wanneer u ze schrijft! De natuurkunde hierachter is erg complex en hangt af van de reeks afstanden tussen opeenvolgende pulsen, van de gegevens die eerder op het spoor zijn geschreven en, zo voelt het, ook van de maanstand en de kleur van uw ondergoed. Er zijn enkele vuistregels die bij "typische"-pulsafstandsintervallen kunnen worden gebruikt om dit gedeeltelijk te beperken. Deze regels heten schrijfvoorcompensatie: de pulsen worden vóór het schrijven verplaatst, zodat ze na het schrijven op de juiste plaats terechtkomen. De 45IO27 past al schrijfvoorcompensatie toe voor HD-geformatteerde schijven. Het effect van het verplaatsen van pulsen is vooral uitgesproken bij pulsen die dichter bij elkaar liggen.
- Hoe hoger genummerd een spoor is, des te korter het is. Dat betekent dat de pulsen in de tijd verder uit elkaar moeten liggen om op een spoor op dezelfde afstand van elkaar te staan. Daarom past de 1541 minder gegevens op hoger genummerde sporen. Dit is vooral uitgesproken bij hogere datatempo's, waar de "magnetische resolutie" van de schijven een probleem wordt. Om het nog erger te maken is de sterkte van de magnetische signalen evenredig aan de snelheid waarmee het spoor passeert. Dat betekent dat bij hoger genummerde sporen, die dichter bij het midden van de schijf liggen, niet alleen de "magnetische resolutie" lager is door de kortere sporen, maar ook de lineaire snelheid van de schijf onder de kop lager is, wat resulteert in zwakkere signalen.

Het is de combinatie van deze factoren die doorgaans beperkt hoe dicht u gegevens op een diskette kunt pakken – samen uiteraard met de kwaliteit en toestand van de diskette en het schijfstation. De HD+-schijfformaten van de MEGA65, die rond de 3MiB op een standaard diskette van 1.44MiB kunnen passen, besteden bijzondere aandacht aan het beheersen van deze factoren. Daarom gebruiken die formaten veel verschillende dichtheidszones en gebruiken ze bovendien RLL27-codering in plaats van MFM-codering, omdat die de minimale afstand tussen pulsen vergroot.

F011-registers van de schijfstationcontroller

De volgende registers zijn de set F011-compatibiliteitsregisters van de 45IO27. Registers die verband houden met het gebruik van opslag op basis van SD-kaarten vindt u in de overeenkomstige sectie hieronder.

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D080	53376	IRQ	LED	MOTOR	SWAP	SIDE	DS		
D081	53377	WRCMD	RDCMD	FREE	STEP	DIR	ALGO	ALT	NOBUF
D082	53378	BUSY	DRQ	EQ	RNF	CRC	LOST	PROT	TK0
D083	53379	RDREQ	WTREQ	RUN	WGATE	DISKIN	INDEX	IRQ	DSKCHG
D084	53380	TRACK							
D085	53381	SECTOR							
D086	53382	SIDE							
D087	53383	DATA							
D088	53384	CLOCK							
D089	53385	STEP							
D08A	53386	PCODE							

- **ALGO** Selecteert het lees- en schrijfalgoritme (wordt momenteel genegeerd).
- **ALT** Selecteert de alternatieve DPLL-leesherstelmethode (niet geïmplementeerd)
- **BUSY** F011 FDC-busyvlag (commando wordt uitgevoerd) (alleen lezen)
- **CLOCK** Stelt het klokpatroon in dat wordt gebruikt bij het schrijven van adres- en datamerken. Normaal $FF laten staan
- **COMMAND** F011 FDC-commandoregister
- **CRC** F011 FDC CRC-controlefoutvlag (alleen lezen)
- **DATA** F011 FDC-gegevensregister (lezen/schrijven) voor toegang tot de 512 byte sectorbuffer van de schijfstationcontroller
- **DIR** Stelt de staprichting in (naar binnen ten opzichte van
- **DISKIN** F011 Schijfgevoeligheid (alleen lezen)
- **DRQ** F011 FDC DRQ-vlag (een of meer bytes gegevens zijn gereed) (alleen lezen)
- **DS** Stationselectie (0 tot 7). Intern station is 0. Tweede schijfstation op de interne kabel is 1. Andere waarden zijn gereserveerd voor de C1565 externe stationinterface.
- **DSKCHG** F011 Schijfwisselgevoeligheid (alleen lezen)
- **EQ** F011 FDC CPU- en diskschijfwaarden naar de sectorbuffer zijn gelijk, wat aangeeft dat de sectorbuffer vol of leeg is. (alleen lezen)

- **FREE** Commando is een vrij formaat (laag niveau) bewerking
- **INDEX** F011 Indexgatgevoeligheid (alleen lezen)
- **IRQ** De schijfstationcontroller heeft een interrupt gegenereerd (alleen lezen). Opmerking: interrupts worden momenteel niet geïmplementeerd op de 45IO27.
- **LED** Station-LED knippert wanneer ingesteld
- **LOST** F011 LOST-vlag (gegevens zijn verloren gegaan tijdens de overdracht, d.w.z. de CPU las de gegevens niet snel genoeg) (alleen lezen)
- **MOTOR** Schakelt de stationmotor en LED in (tenzij het LED-signaal ook is ingesteld, waardoor de station-LED knippert)
- **NOBUF** Herstel de lees-/schrijfpunten van de sectorbuffer
- **PCODE** (Alleen lezen) geeft de beschermingscode van het meest recent gelezen sector terug. Was bedoeld voor elementaire kopieerbescherming. Niet geïmplementeerd.
- **PROT** F011 Schrijfbeveiligingsvlag van de schijf (alleen lezen)
- **RDCMD** Commando is een leesbewerking wanneer ingesteld
- **RDREQ** F011 Leesverzoekvlag, d.w.z. het gevraagde sector werd gevonden tijdens een leesbewerking (alleen lezen)
- **RNF** F011 FDC-verzoek niet gevonden (RNF), d.w.z. een lees- of schrijfbewerking heeft het gevraagde sector niet gevonden (alleen lezen)
- **RUN** F011 Opeenvolgende overeenkomst. Een synoniem van RDREQ op de 45IO27 (alleen lezen)
- **SECTOR** F011 FDC-sectorselectieregister
- **SIDE** Stuurt het SIDE-signaal naar de schijfstationcontroller direct, d.w.z. selecteert welke zijde van het medium actief is.
- **STEP** Het schrijven van 1 laat de kop in de aangegeven richting stappen
- **SWAP** Wissel de boven- en onderste helft van de gegevensbuffer (d.w.z. inverteer bit 8 van de sectorbuffer)
- **TK0** F011 Kop bevindt zich boven spoor 0-vlag (alleen lezen)
- **TRACK** F011 FDC-spoorselectieregister
- **WGATE** F011 Schrijfpoortvlag. Geeft aan dat het station momenteel naar het medium schrijft. Er kunnen problemen optreden als een schrijfbewerking wordt afgebroken (alleen lezen)
- **WRCMD** Commando is een schrijfbewerking wanneer ingesteld
- **WTREQ** F011 Schrijfverzoekvlag, d.w.z. het gevraagde sector werd gevonden tijdens een schrijfbewerking (alleen lezen)

De volgende registers zijn uitsluitend van toepassing op de 45IO27, d.w.z. ze zijn specifiek voor de MEGA65:

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D6A0	54944	DENSITY	DBGMO-TORA	DBGMO-TORA	DBGDIR	DBGDIR	DBGW-DATA	DBGW-GATE	DBGW-GATE
D6A2	54946	DATARATE							

- **DATARATE** Stelt het aantal buscycli per diskettemagnetisch interval in (verlaag om het datatempo te verhogen)
- **DBGDIR** Stuurt de STEPDIR-lijn van de schijfstationcontroller
- **DBGMOTORA** Stuurt de MOTOR-lijn van het schijfstation aan
- **DBGWDATA** Stuurt de WDATA-lijn van het schijfstation aan
- **DBGWGATE** Stuurt de WGATE-lijn van het schijfstation aan
- **DENSITY** Stuurt de selectielijn van de dichtheid van het schijfstation aan

SD-KAARTCONTROLLER EN F011-VIRTUALISATIEFUNCTIES

Voor situaties waarin u geen echte diskettes wilt gebruiken, ondersteunt de 45IO27 twee complementaire alternatieve modi:

- Toegang tot schijfimages op basis van de SD-kaart.
- Gevirtualiseerde toegang tot schijfimages.

Dit komt bovenop de directe toegang tot een dual-bus-SD-kaartinterface.

Toegang tot schijfimages op basis van de SD-kaart

De 45IO27 is tegelijk een schijfstation en een SD-kaartcontroller. Hierdoor kan het apparaat transparant toegang bieden tot D81-schijfimages die op de SD-kaart zijn opgeslagen. Verder is het, omdat de controller is gecombineerd, mogelijk om het schijfstation nog steeds te laten stappen en draaien alsof het wordt gebruikt, wat voor aanzienlijke sfeer en een gevoel van realisme zorgt, zelfs wanneer u schijfimages gebruikt.

De 45IO27 ondersteunt zowel standaard D81-schijfimages van 800KB als "MEGA Images" van 64MB. Hoewel een besturingssysteem beperkingen kan opleggen op basis van de naam van een bestand, is de 45IO27 blind voor deze eisen. In plaats daarvan vereist het apparaat enkel dat een aaneengesloten blok van 800KB of 64MB van de SD-kaart wordt gebruikt om een schijfimage te bevatten.

Wanneer een schijfimage wordt ingeschakeld, worden de overeenkomstige sectoren op de SD-kaart feitelijk onder de controle van de gebruiker geplaatst en kan het besturingssysteem het lezen of schrijven van die sectoren niet langer verhinderen. U moet daarom nooit toegang inschakelen tot een image dat korter is dan de vereiste grootte, omdat het de gebruiker anders in staat stelt onbedoeld of kwaadwillig gegevens te openen en/of te wijzigen die geen deel uitmaken van het imagebestand.

Om dezelfde reden kan alleen de hypervisor het sectornummer wijzigen waar een schijfimage begint (de D?STARTSEC?-signalen), of het gebruik van schijfimages in plaats van het echte schijfstation toestaan (de USEREAL0- en USEREAL1-signalen). Zodra de hypervisor de beginsector van een schijfimage heeft ingesteld en het USEREAL0- of USEREAL1-signaal heeft gewist, kan de gebruiker nog steeds bepalen of een toegang naar het echte schijfstation of naar het schijfimage gaat, door het betreffende signaal respectievelijk te wissen of te zetten. Voor station 0 is dit D0IMG, en voor station 1 is het D1IMG.

Er zijn ook signalen die bepalen of een schijfimage een D81-image van 800KB of een MEGA Disk-image van 64MB is, of een schijfimage aanwezig is en of het schrijfbeveiligd is. Deze bevinden zich allemaal in het $D68B-register. Omdat manipulatie van deze registers gegevens kan beschadigen of er op ongeoorloofde wijze toegang toe kan geven, zijn deze signalen allemaal alleen-lezen, behalve vanuit de hypervisor.

De volgende tabel geeft de registers weer die worden gebruikt om de toegang tot de schijfimages op de SD-kaart te regelen:

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D68A	54922	D1D64	D0D64	-					
D68B	54923	D1MD	D0MD	D1WP	D1P	D1IMG	D0WP	D0P	D0IMG
D68C	54924	D0STARTSEC0							
D68D	54925	D0STARTSEC1							
D68E	54926	D0STARTSEC2							
D68F	54927	D0STARTSEC3							
D690	54928	D1STARTSEC0							
D691	54929	D1STARTSEC1							
D692	54930	D1STARTSEC2							
D693	54931	D1STARTSEC3							
D6A1	54945	-				SILENT	USE-REAL1	TARGANY	USE-REAL0

- **D0D64** Het schijfimage van F011-station 0 is D64 indien ingesteld, anders D81 (zie ook SDFDC:D0MD)
- **D0IMG** F011-station 0 gebruikt een schijfimage indien ingesteld, anders een echte diskette.
- **D0MD** Het schijfimage van F011-station 0 is D65 indien ingesteld, anders D81 (zie ook SDFDC:D0D64)
- **D0P** F011-station 0 medium aanwezig

- **D0STARTSEC0** Adres van het schijfimage van F011-station 0 op de SD-kaart (LSB)
- **D0STARTSEC1** F011-station 0 schijfimageadres op SD-kaart (2e byte)
- **D0STARTSEC2** F011-station 0 schijfimageadres op SD-kaart (3e byte)
- **D0STARTSEC3** F011-station 0 schijfimageadres op SD-kaart (MSB)
- **D0WP** Schrijven inschakelen voor F011-station 0
- **D1D64** F011-station 1 schijfimage is D64 indien gezet, anders D81 (zie ook SDFDC:D1MD)
- **D1IMG** F011-station 1 gebruikt schijfimage indien gezet, anders een echt schijfstation.
- **D1MD** F011-station 1 schijfimage is D65 indien gezet, anders D81 (zie ook SDFDC:D1D64)
- **D1P** F011-station 1 medium aanwezig
- **D1STARTSEC0** F011-station 1 schijfimageadres op SD-kaart (LSB)
- **D1STARTSEC1** F011-station 1 schijfimageadres op SD-kaart (2e byte)
- **D1STARTSEC2** F011-station 1 schijfimageadres op SD-kaart (3e byte)
- **D1STARTSEC3** F011-station 1 schijfimageadres op SD-kaart (MSB)
- **D1WP** Schrijven inschakelen voor F011-station 1
- **SILENT** Schakelt het draaien en tracken van de diskette uit voor SD-kaartbewerkingen.
- **TARGANY** Leest het volgende sector onder de kop indien gezet, waarbij de gevraagde zijde, track en sectornummer worden genegeerd.
- **USEREAL0** Gebruikt een echt schijfstation voor station 0 indien gezet (alleen-lezen, behalve vanuit de hypervisor)
- **USEREAL1** Gebruikt een echt schijfstation voor station 1 indien gezet (alleen-lezen, behalve vanuit de hypervisor)

F011-virtualisatie

Naast het toestaan van automatische lees- en schrijftoegang tot D81-images op basis van de SD-kaart, is het mogelijk een programma op de seriële monitorinterface aan te sluiten dat gegevens verschaft en aanvaardt alsof het de diskette was.

Dit wordt gewoonlijk gebruikt in een cross-ontwikkelomgeving, waarin u een schijfimage vaak wilt wijzigen dat door een programma wordt gebruikt dat u ontwikkelt - zonder dat u telkens nieuwe versies van het schijfimage eerst op de SD-kaart van de MEGA65 hoeft te zetten. Het heeft bovendien het bijkomende voordeel dat u

gemakkelijk kunt visualiseren welke sectoren worden gelezen en geschreven, wat de ontwikkeling en het opsporen van fouten in uw programma kan versnellen.

Deze functie werkt samen met de hypervisor van de MEGA65 door hyperrupts te activeren (dat zijn interrupts die de hypervisor activeren). In de hypervisor zit dan speciale code die via de seriële monitorinterface met het `m65`-programma communiceert.

Als dat allemaal tamelijk ingewikkeld klinkt: alles wat u hoeft te weten is dat u deze functie gebruikt door het `m65`-hulpprogramma uit te voeren met argumenten als `-d image.d81`. Hiermee zou de verbinding met de MEGA65 automatisch tot stand moeten komen. Als de BASIC-interpreter niet meer reageert, drukt u op de resetknop (niet op de netschakelaar) aan de linkerkant van uw MEGA65, en zou deze moeten terugkeren naar de `READY.`-prompt van BASIC - en als uw meegeleverde schijfimage een C65-auto-bootfunctie heeft, zou deze automatisch moeten beginnen met opstarten.

Deze functie werkt uitstekend indien de hostcomputer Linux draait en maakt laden mogelijk met een snelheid van ongeveer 60KB per seconde. Op Windows- of Apple OSX-systemen kan de snelheid echter aanzienlijk lager liggen.

Om dit te gebruiken heeft u bovendien een interface-module en/of kabel nodig om uw cross-ontwikkelingssysteem op de seriële monitorinterface van de MEGA65 aan te sluiten. Dit doet u het eenvoudigst met een Trenz TE0790-03 JTAG-adapter en een mini-USB-kabel.

Meer informatie over het gebruik van deze interface en het `m65`-hulpprogramma vindt u in Hoofdstuk/Bijlage 18 op bladzijde 18-3.

Dual-bus SD-kaartcontroller

De 45IO27 bevat een snelle dual-bus SD-kaartcontroller. Deze controller werkt in de SPI x1-modus met een kloksnelheid van 20MHz en biedt een maximale doorvoer van ongeveer 2MB per seconde. De kwaliteit van de SD-kaart is bepalend voor de prestaties: sommige kaarten leveren doorgaans 1.7MB per seconde, terwijl andere 1MB per seconde of minder halen. Over het algemeen presteren nieuwere kaarten die geschikt zijn voor video-opnamen beter. De controller ondersteunt SDHC-kaarten en heeft experimentele ondersteuning voor SDXC-kaarten. Oudere SD-kaarten met een capaciteit van 2GB of minder worden niet ondersteund, omdat deze een andere adresseringsmethode gebruiken.

De SD-controller zelf is eenvoudig aan te sturen: geef het sectornummer op in $D681-$D684 en geef vervolgens een lees- of schrijfcommando aan het commandoregister ($D680). De SD-controller ondersteunt alleen sectorgebaseerde gebufferde bewerkingen via de sectorbuffer. In hypervisormodus bevindt de sectorbuffer zich op $FFD6E00-$FFD6FFF, terwijl de SD-kaart en de schijfstationcontroller in de normale werkingsmodus van de computer één adres delen voor zowel de diskettesectorbuffer als de SD-kaartsectorbuffer. Welke buffer op dat adres zichtbaar is, wordt bepaald door het BUFSEL-signaal. Bij een waarde van 1 is de SD-kaartbuffer zichtbaar, bij 0 de diskettesectorbuffer. Zie ook subsectie V op bladzijde V-4 voor een nadere beschrijving van het precieze gedrag van deze buffer in de normale modus ten opzichte van de hypervisormodus, en hoe deze ook op $DE00 kan worden gemapt.

Write-gate

Bij het schrijven van een sector moet u eerst de write-gate openen. Dit is een mechanisme dat onbedoelde gegevenscorruptie op de SD-kaart voorkomt: u moet twee verschillende waarden achtereenvolgens naar het commandoregister ($D680) schrijven. Na het openen van de write-gate heeft u ongeveer één milliseconde om het schrijfcommando te geven, waarna de write-gate weer sluit en de SD-kaart wordt beschermd tegen schrijven totdat de write-gate opnieuw wordt geopend. Er zijn twee aparte write-gates: één voor het master boot record (sector 0) en één voor alle overige sectoren, beide vermeld in de commandotabel hieronder. Dit biedt extra bescherming van het zeer belangrijke master boot record tegen programma's die per ongeluk sector 0 als doel van een gewone schrijfoperatie berekenen.

Vulmodus

Wanneer u sectoren met een constante waarde wilt vullen, ondersteunt de 45IO27 hiervoor een speciale modus, zodat u de inhoud van de sectorbuffer niet hoeft te overschrijven. U activeert dit door de gewenste vulwaarde in het FILLVAL-register ($D686) te plaatsen, vervolgens het commando om de vulmodus in te schakelen ($83) te geven, de schrijfbewerkingen uit te voeren en tenslotte het commando om de vulmodus uit te schakelen ($84) te geven.

Selectie tussen meerdere SD-kaarten

De controller ondersteunt twee SD-kaartinterfaces, en het is mogelijk om in beide interfaces tegelijkertijd een kaart te hebben. Elke kaart moet echter apart worden gereset en aangestuurd. Er kan tegelijkertijd slechts één kaart worden aangestuurd. U kunt wel elke kaart eenmaal resetten en vervolgens tussen de kaarten wisselen om individuele bewerkingen uit te voeren.

Om de eerste SD-kaartslot te selecteren, schrijft u $C0 naar het SD-controllercommandoregister ($D680). Om de tweede SD-kaartslot te selecteren, schrijft u in plaats daarvan $C1.

Commandotabel van de SD-controller

De SD-controller ondersteunt de volgende commando's die naar het commandoregister op $D680 geschreven kunnen worden:

Commando	Functie
$00 (0)	Zet de SD-kaart in reset (verouderd. Gebruik in plaats daarvan commando $10)
$01 (1)	Haal de SD-kaart uit reset
$02 (2)	Lees een sector van de SD-kaart
$03 (3)	Schrijf één sector naar de SD-kaart
$04 (4)	Schrijf de eerste sector van een meervoudige schrijfbewerking naar de SD-kaart

vervolgd ...

...vervolgd

Commando	Functie
$05 (5)	Schrijf een volgende sector van een meervoudige schrijfbewerking naar de SD-kaart
$06 (6)	Schrijf de laatste sector van een meervoudige schrijfbewerking naar de SD-kaart
$0C (12)	Vraag om het doorspoelen van de schrijfbuffers van de SD-kaart (experimenteel)
$0E (14)	Trek de SD-handshakelijn laag (alleen voor foutopsporing)
$0F (15)	Trek de SD-handshakelijn hoog (alleen voor foutopsporing)
$10 (16)	Zet de SD-kaart in reset met ingestelde vlaggen (voorkeursmethode)
$11 (17)	Haal de SD-kaart uit reset (alternatieve methode)
$40 (64)	Wis de SDHC/SDXC-vlag en selecteer daarmee de verouderde SD-kaartmodus (verouderd)
$41 (65)	Zet de SDHC/SDXC-modusvlag
$44 (68)	Beëindig het geforceerd wissen van de foutvlag van de SD-kaart-toestandsmachine
$45 (69)	Begin het geforceerd wissen van de foutvlag van de SD-kaart-toestandsmachine
$4D (77)	Open de write-gate voor sector 0 (master boot record) gedurende ongeveer 1 milliseconde
$57 (87)	Open de write-gate voor alle sectoren > 0 gedurende ongeveer 1 milliseconde
$81 (129)	Schakel het mappen van de SD/FDC-sectorbuffer op $DE00 - $DFFF in
$82 (130)	Schakel het mappen van de SD/FDC-sectorbuffer op $DE00 - $DFFF uit
$83 (131)	Schakel de vulmodus van de SD-kaart in
$84 (132)	Schakel de vulmodus van de SD-kaart uit
$C0 (192)	Selecteer SD-kaartslot 0
$C1 (193)	Selecteer SD-kaartslot 1

Merk op dat de hypervisor de rechtstreekse toegang tot de SD-controller kan in- of uitschakelen. Het besturingssysteem van de hypervisor kan een mechanisme bieden om toestemming te vragen voor toegang tot de SD-kaartcontroller, bijvoorbeeld voor hulpprogramma's voor schijfbeheer.

De registers van de SD-kaartcontroller zijn als volgt:

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D680	54912	CMDANDSTAT							
D681	54913	SECTOR0							
D682	54914	SECTOR1							
D683	54915	SECTOR2							

vervolgd ...

...vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D684	54916	SECTOR3							
D686	54918	FILLVAL							
D68A	54922	-				VFDC1	VFDC0	VICIII	CDC00
D6AE	54958	FDCTIBEN	FDC-2XSEL	FDC-VARSPD	AUTO-2XSEL	FDCENC			
D6AF	54959	-		VLOST	VDRQ	VRNF	VEQINH	VW-FOUND	VRFOUND

- **AUTO2XSEL** Selecteert automatisch de DD- of HD-decoder voor de weergave van de laatste sector
- **CDC00** (alleen-lezen) Ingesteld indien kleurengeheugen op $DC00
- **CMDANDSTAT** SD-controllerstatus/commando
- **FDC2XSEL** Selecteert de HD-decoder voor de weergave van de laatste sector
- **FDCENC** Selecteert de diskettecodering (0=MFM, 1=RLL2,7, F=Raw-codering)
- **FDCTIBEN** Schakelt het gebruik van Track Info Block-instellingen in
- **FDCVARSPD** Schakelt automatische variabele snelheidsselectie in voor de diskettecontroller met behulp van Track Information Blocks op MEGA65 HD-diskettes
- **FILLVAL** ALLEENSCHRIJVEN: stelt het vulbyte in voor gebruik in de vulmodus, in plaats van SD-buffergegevens
- **SECTOR0** SD-controller SD-sectoradres (LSB)
- **SECTOR1** SD-controller SD-sectoradres (2e byte)
- **SECTOR2** SD-controller SD-sectoradres (3e byte)
- **SECTOR3** SD-controller SD-sectoradres (MSB)
- **VDRQ** Handmatig instellen van het f011_drq-signaal (uitsluitend bedoeld voor de virtuele F011-modus)
- **VEQINH** Handmatig instellen van het f011_eq_inhibit-signaal (uitsluitend bedoeld voor de virtuele F011-modus)
- **VFDC0** (alleen-lezen) Ingesteld indien station 0 is gevirtualiseerd (sectoren geleverd via de seriële monitorinterface)
- **VFDC1** (alleen-lezen) Ingesteld indien station 1 is gevirtualiseerd (sectoren geleverd via de seriële monitorinterface)
- **VICIII** (alleen-lezen) Ingesteld indien de VIC-IV- of Ethernet I/O-bank zichtbaar is
- **VLOST** Handmatig instellen van het f011_lost-signaal (uitsluitend bedoeld voor de virtuele F011-modus)

- **VRFOUND** Handmatig instellen van het f011_rsector_found-signaal (uitsluitend bedoeld voor de virtuele F011-modus)
- **VRNF** Handmatig instellen van het f011_rnf-signaal (uitsluitend bedoeld voor de virtuele F011-modus)
- **VWFOUND** Handmatig instellen van het f011_wsector_found-signaal (uitsluitend bedoeld voor de virtuele F011-modus)

TOUCHPADINTERFACE

Bepaalde MEGA65-varianten zijn uitgerust met een LCD-touchpad, met name de handheld MEGAphone-versie van de MEGA65. De aanraakinterface ondersteunt het detecteren van twee gelijktijdige aanraakgebeurtenissen. Sommige varianten ondersteunen mogelijk ook gebarenherkenning, hoewel deze nog zeer experimenteel is.

De aanraakdetectie-interface in de 45IO27 wordt aangevuld door de schermtoetsenbordinterface van de 4551-UART- en GPIO-controller. Raadpleeg paragraaf T voor verdere informatie. Bijzonder relevant is bit 7 van de registers $D615 -$D617, die het mogelijk maakt de schermtoetsenbordinterface te activeren, te kiezen of het schermtoetsenbord in het bovenste of onderste gedeelte van het scherm wordt geplaatst, en of het primaire of secundaire schermtoetsenbord wordt weergegeven.

De rechtstreekse verbindingen tussen de 4551 en de 45IO27 combineren informatie over het op dat moment weergegeven schermtoetsenbord en de aanraakinterfacecontroller, waardoor synthetische toetsenbordgebeurtenissen automatisch worden geactiveerd wanneer u op het schermtoetsenbordgedeelte van de aanraakinterface drukt. Hierdoor kan de aanraakinterface worden gebruikt om het schermtoetsenbord aan te sturen, zonder dat daarvoor ondersteuning van gebruikersprogramma's nodig is. Dit werkt zelfs wanneer het schermtoetsenbord beweegt tijdens de activering of wanneer het tussen de boven- en onderkant van het scherm wordt verplaatst.

Omdat aanraakinterfaces kalibratie kunnen vereisen, biedt de 45IO27 een lineaire transformatie van zowel de X- als de Y-coördinaten van een aanraakgebeurtenis. Er zijn specifiek schaalregisters (TCHXSCALE en TCHYSCALE) en offsetregisters (TCHXDELTA en TCHYDELTA) die deze transformatie verzorgen. Het is ook mogelijk de coördinaten van het touchpad te spiegelen op de X-as, de Y-as, of op beide assen. Deze kalibratieregisters beïnvloeden ook de werking van het schermtoetsenbord.

Er dient ook te worden opgemerkt dat sommige aanraakinterfaces geen constante horizontale of verticale resolutie hebben. Sommige panelen hebben bijvoorbeeld een gedeelte met een lage horizontale resolutie in het midden van het paneel, waarvoor enige zorg nodig kan zijn.

Om de primaire aanraakgebeurtenis te detecteren, kunnen de registers TOUCH1XLSB, TOUCH1XMSB, TOUCH1YLSB en TOUCH1YMSB worden gelezen. Voor de tweede aanraakgebeurtenis bestaan vergelijkbare registers: TOUCH2XLSB, TOUCH2XMSB, TOUCH2YLSB en TOUCH2YMSB. Elke aanraakgebeurtenis heeft een vlag van één bit

die aangeeft of de aanraakgebeurtenis op dat moment geldig is: de EV1- en EV2-bits van het register $D6B0. Er zijn ook overeenkomstige bitvelden die aangeven of een bepaalde aanraakgebeurtenis is ingedrukt of losgelaten, waardoor kan worden gedetecteerd wanneer een vinger zowel contact met het scherm maakt als het contact verbreekt. De UPDN1- en UPDN2-signalen verschaffen deze informatie. Binaire waarden van respectievelijk 01 en 10 geven aan of de vinger van het touchpad is verwijderd of ertegen is gedrukt. De waarden 00 en 11 betekenen dat een vinger al dan niet tegen het touchpad wordt gehouden.

De primaire aanraakgebeurtenis wordt tevens ingevoerd in de lichtpen-ingang van de VIC-IV en kan worden gedetecteerd met de gewone lichtpenregisters van de VIC-IV.

De registers voor de touchpadinterface zijn als volgt:

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D6B0	54960	YINV	XINV	UPDN2		UPDN1		EV2	EV1
D6B1	54961	CALXSCALELSB							
D6B2	54962	CALXSCALEMSB							
D6B3	54963	CALYSCALELSB							
D6B4	54964	CALYSCALEMSB							
D6B5	54965	CALXDELTALSB							
D6B7	54967	CALYDELTALSB							
D6B8	54968	CALYDELTAMSB							
D6B9	54969	TOUCH1XLSB							
D6BA	54970	TOUCH1YLSB							
D6BB	54971	-		TOUCH1YMSB		-		TOUCH1XMSB	
D6BC	54972	TOUCH2XLSB							
D6BD	54973	TOUCH2YLSB							
D6BE	54974	-		TOUCH2YMSB		-		TOUCH2XMSB	
D6C0	54976	GESTUREID				GESTUREDIR			

- **CALXDELTALSB** X-delta van de touchpad (LSB)
- **CALXSCALELSB** X-schaling van de touchpad (LSB)
- **CALXSCALEMSB** X-schaling van de touchpad (MSB)
- **CALYDELTALSB** Y-delta van de touchpad (LSB)
- **CALYDELTAMSB** Y-delta van de touchpad (MSB)
- **CALYSCALELSB** Y-schaling van de touchpad (LSB)
- **CALYSCALEMSB** Y-schaling van de touchpad (MSB)
- **EV1** Aanraakgebeurtenis 1 is geldig
- **EV2** Aanraakgebeurtenis 2 is geldig

- **GESTUREDIR** Gebaarrichtingen van de touchpad (links, rechts, omhoog, omlaag)
- **GESTUREID** Gebaren-ID van de touchpad
- **TOUCH1XLSB** X-coördinaat van aanraking #1 op de touchpad (LSB)
- **TOUCH1XMSB** X-coördinaat van aanraking #1 op de touchpad (MSBs)
- **TOUCH1YLSB** Y-coördinaat van aanraking #1 op de touchpad (LSB)
- **TOUCH1YMSB** Y-coördinaat van aanraking #1 op de touchpad (MSBs)
- **TOUCH2XLSB** X-coördinaat van aanraking #2 op de touchpad (LSB)
- **TOUCH2XMSB** X-coördinaat van aanraking #2 op de touchpad (MSBs)
- **TOUCH2YLSB** Y-coördinaat van aanraking #2 op de touchpad (LSB)
- **TOUCH2YMSB** Y-coördinaat van aanraking #2 op de touchpad (MSBs)
- **UPDN1** Status van aanraking 1 (losgelaten/ingedrukt)
- **UPDN2** Status van aanraking 2 (losgelaten/ingedrukt)
- **XINV** Inverteert de horizontale as
- **YINV** Inverteert de verticale as

ONDERSTEUNENDE FUNCTIES VOOR AUDIO

De 45IO27 verschaft de primaire interface naar de volledige cross-bar-audiomixer van de MEGA65. Dit omvat de interface voor het lezen of wijzigen van de mixercoëfficiënten, de toegang tot de feedbackregisters van de mixer en het instellen van de 16-bits digitale samplewaarden, welke tevens twee van de ingangskanalen van de audiomixer vormen.

De audiomixer bestaat uit 128 coëfficiënten, die elk 16 bits groot zijn. Elk audio-uitgangskanaal, zoals de luidspreker links, de luidspreker rechts, de hoofdtelefoon links, de hoofdtelefoon rechts, de cellulaire modem 1 (alleen op MEGAphone-modellen) enzovoort, wordt gegenereerd door elk audio-ingangskanaal te nemen, het met de passende coëfficiënt te vermenigvuldigen en het aan de totale uitvoer van het audio-uitgangskanaal toe te voegen.

Omdat elk audio-uitgangskanaal zijn eigen reeks coëfficiënten heeft die op alle audio-ingangskanalen worden toegepast, betekent dit dat het mogelijk is om volstrekt verschillende audio uit te voeren over elk audiokanaal: Zo is het bijvoorbeeld mogelijk om uw favoriete quadrafonische SID-muziek af te spelen via de hoofdtelefoon terwijl u voorbijgangers trakteert op Amiga-stijl MOD-audio. Dit is de reden dat de audiomixer een `full cross-bar-mixer` wordt genoemd, omdat er geen beperkingen zijn aan de

wijze waarop u elk audio-uitgangskanaal mixt. In dit opzicht lijkt hij sterk op een volwaardige audiomengtafel, en maakt hij verschillende mengniveaus voor verschillende luidsprekers mogelijk.

Omdat de audiocoëfficiënten elk 16 bits groot zijn, wordt elke coëfficiënt gevormd door twee opeenvolgende bytes van de audiocoëfficiëntenruimte. Wijzigingen aan de audiocoëfficiënten worden onmiddellijk van kracht, dus dient er voorzichtigheid te worden betracht bij het wijzigen van coëfficiënten om hoorbare klikken en plofjes te voorkomen. U moet bovendien 32 cycli laten verlopen voordat u de geselecteerde audiocoëfficiënt wijzigt; indien u dit niet doet, kan de wijziging worden verworpen indien de accumulator van de audiomixer die coëfficiënt nog niet opnieuw heeft verwerkt.

De audiobronnen op de MEGA65- en MEGAphone-apparaten zijn als volgt:

Kanaal-ID v/d ingang	**Aansluiting**
$0 (0)	Linker SID's
$1 (1)	Rechter SID's
$2 (2)	Modem-bay 1 (alleen MEGAphone)
$3 (3)	Modem-bay 2 (alleen MEGAphone)
$4 (4)	Bluetooth™ Links
$5 (5)	Bluetooth™ Rechts
$6 (6)	Hoofdtelefoon-interface 1
$7 (7)	Hoofdtelefoon-interface 2
$8 (8)	Digitale audio Links
$9 (9)	Digitale audio Rechts
$A (10)	MEMs-microfoon 0 (alleen Nexys4 en MEGAphone)
$B (11)	MEMs-microfoon 1 (alleen MEGAphone)
$C (12)	MEMs-microfoon 2 (alleen MEGAphone)
$D (13)	MEMs-microfoon 3 (alleen MEGAphone)
$E (14)	Hoofdtelefoon-audiojack-microfoon (alleen Nexys4 en MEGAphone)
$F (15)	OPL-compatibele FM-audio (*deelt de coëfficiënt met ingang 14*)

De OPL-compatibele FM-audio op bron 15 wordt aangestuurd door de coëfficiënt voor bron 14. De reden hiervoor is dat de coëfficiënt voor bron 15 het mastervolumeniveau voor elke uitvoer bepaalt. Het OPL-compatibele FM-audioapparaat is momenteel niet functioneel in de MEGA65-core.

De cross-bar-audiomixer van het audiosysteem ondersteunt de volgende acht uitgangskanalen:

Kanaal-ID v/d uitgang	Aansluiting
$0 (0)	Linker primaire luidspreker (digitale audio op de MEGA65 R2/R3, fysieke luidspreker op de MEGAphone, audiojack op de Nexys4)
$1 (1)	Rechter primaire luidspreker (digitale audio op de MEGA65 R2/R3, fysieke luidspreker op de MEGAphone, audiojack op de Nexys4)
$2 (2)	Modem-bay 1 audio-uitvoer (alleen MEGAphone)
$3 (3)	Modem-bay 2 audio-uitvoer (alleen MEGAphone)
$4 (4)	Bluetooth-audio links (alleen MEGAphone)
$5 (5)	Bluetooth-audio rechts (alleen MEGAphone)
$6 (6)	Hoofdtelefoon-uitvoer links (MEGA65 R2/R3 en MEGAphone. Op Nexys4-borden stuurt de primaire luidspreker de 3.5 mm-audiojack aan)
$7 (7)	Hoofdtelefoon-uitvoer rechts (MEGA65 R2/R3 en MEGAphone. Op Nexys4-borden stuurt de primaire luidspreker de 3.5 mm-audiojack aan)

Om het coëfficiëntregisternummer voor een bepaalde bron en uitvoer te bepalen, vermenigvuldigt u het uitvoernummer met 32 en het bronnummer met 2. Dit geeft het registernummer van de LSB van de 16-bits coëfficiënt. De MSB bevindt zich in het volgende register. Om bijvoorbeeld de coëfficiënt van de rechter SID's in te stellen op de tweede modem bay-audio-uitvoer, wordt de coëfficiënt $32\times3+1\times2 = 96+2 = 98$.

Overige audiofuncties

Het audiosysteem ondersteunt verscheidene andere functies, waarvan de documentatie nog gereed moet komen.

Mixer-feedbackregisters

Deze registers maken het mogelijk dat de processor de gemengde audio van een bepaald uitgangskanaal uitleest. Dit kan worden gebruikt om weergaven van audiogolfformen te maken of om bepaalde audio-effecten, zoals reverb, te realiseren.

8/16-bits stereo digitale audio

Er zijn registers beschikbaar om 8- of 16-bits samplewaarden rechtstreeks in specifieke invoerkanalen van de audiomixer in te voeren. Dit biedt een eenvoudige manier om digitaal audiogegevens af te spelen. Deze voorziening is met name nuttig voor procedureel gegenereerde audiogegevens. Voor opgenomen samples is het doorgaans eenvoudiger en beter om de Audio DMA-functionaliteit te gebruiken die het afspelen van digitaal audio volledig automatiseert. Audio DMA is vooral nuttig op hogere samplefrequenties, omdat het de belasting van de CPU vermindert en sample-jitter aanzienlijk verkleint.

Pulsbreedte versus pulsdichtheidsmodulatie

Voor MEGA65-modellen die een 1-bits audio-uitvoer met oversampling gebruiken (tot en met het R3/R3A-model) is het mogelijk te kiezen tussen deze twee verschillende oversamplingmethoden. Beide hebben vergelijkbare prestaties, maar gebruikers hebben wellicht een voorkeur voor de ene boven de andere. Deze keuze heeft geen effect op het R4 of nieuwere modellen die een dedicated audio-DAC gebruiken om audio naar de 3.5 mm-audiojack te voeren en die van nature een betere audiokwaliteit hebben. Voor eigenaren van oudere modellen kan de geplande interne uitbreidingskaart van de MEGA65 een verbeterd audio-uitvoercircuit bevatten, afhankelijk van de definitieve configuratie van die kaart. Er is momenteel nog geen termijn bekend waarop deze kaart beschikbaar komt. In de tussentijd is het gebruik van HDMI-audio-uitvoer de aanbevolen oplossing, omdat de audio digitaal over de HDMI-kabel wordt gecodeerd.

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D6F4	55028	MIXREGSEL							
D6F5	55029	MIXREGDATA							
D6F8	55032	DIGILLSB							
D6F9	55033	DIGILMSB							
D6FA	55034	DIGIRLSB							
D6FB	55035	DIGIRMSB							
D6FC	55036	READBACKLSB							
D6FD	55037	READBACKMSB							
D711	55057	-				PWMPDM	-		

- **DIGILEFTLSB** Digitale audio, linkerkanaal, LSB
- **DIGILEFTMSB** Digitale audio, linkerkanaal, MSB
- **DIGILLSB** 16-bits digitale audio-uitgang (linker LSB)
- **DIGILMSB** 16-bits digitale audio-uitgang (linker MSB)
- **DIGIRIGHTLSB** Digitale audio, linkerkanaal, LSB
- **DIGIRIGHTMSB** Digitale audio, linkerkanaal, MSB
- **DIGIRLSB** 16-bits digitale audio-uitgang (rechter LSB)
- **DIGIRMSB** 16-bits digitale audio-uitgang (rechter MSB)
- **MIXREGDATA** Leespoort van het audiomixerregister
- **MIXREGSEL** Selectie van het audiomixerregister
- **PWMPDM** Selectie van de PWM/PDM-audio-encodering
- **READBACKLSB** De LSB van de audio-read-back (de bron wordt geselecteerd door $D6F4)
- **READBACKMSB** De MSB van de audio-read-back (de bron wordt geselecteerd door $D6F4)

DIVERSE I/O-FUNCTIES

BIJLAGE W

4541 seriële buscontroller

- Overzicht
- Mogelijkheden van de 4541
- Werkingsprincipe
- Voorbeelden
- Commandoreferentie
- Registertabel
- Seriële bustiming
- Optionele geïntegreerde datalogger

OVERZICHT

De 4541 is een buscontroller die compatibel is met de seriële randapparatuurbus van Commodore™ en die de inspanning die nodig is om met apparaten op deze bus te communiceren sterk vermindert.

MOGELIJKHEDEN VAN DE 4541

Ondersteuning van versnelde seriële protocolvarianten

De 4541 ondersteunt de JiffyDOS™-uitbreidingen op dit protocol, waarmee gegevensoverdrachten ongeveer 10× sneller verlopen dan met het oorspronkelijke protocol. Van een toekomstige revisie van de 4541 wordt verwacht dat deze het snelle seriële protocol van Commodore ondersteunt, dat in de schijfstations 1571 en 1581 aanwezig is.

Processorontlasting via interruptondersteuning

De 4541 voert seriële communicatie uit onafhankelijk van de microprocessor. Samen met een IRQ-functionaliteit stelt dit software in staat om door te gaan met andere taken terwijl seriële communicatie met randapparaten plaatsvindt, waarbij slechts een korte onderbreking nodig is wanneer het volgende gebeurtenis op de seriële randapparatuurbus optreedt.

Onafhankelijkheid van processorsnelheid

Een groot voordeel van de 4541 is dat deze ook alle tijdsvereisten van communicatie op deze bus afhandelt, waardoor de bus kan worden aangestuurd door een processor die op verschillende snelheden kan werken, zonder de software van de buscontroller aan te hoeven passen.

Coëxistentie via open-collectorlogica

Omdat de 4541 open-collectorlogica gebruikt, kan deze parallel worden gebruikt met bestaande softwarematige implementaties van het busprotocol, waarmee compatibiliteit met bestaande software wordt gewaarborgd. In de MEGA65 is de 4541 in deze configuratie geïnstalleerd, waardoor de bestaande softwarematige communicatiesoftware voor de seriële randapparatuurbus ongewijzigd kan blijven werken.

WERKINGSPRINCIPE

De 4541 biedt een eenvoudige interface: u geeft opdrachten, wacht op een reactie en haalt vervolgens de gegevens op die de opdracht heeft verzameld. Sommige opdrachten vereisen ook een gegevensbyte, die via een apart register wordt aangeboden. Er is ook een register met apparaatinformatie, waarmee u kunt zien wat de 4541 van de huidige status van het meest recent gevraagde apparaat vindt, waaronder of het aanwezig is en of het een van beide of beide uitbreidingen van het JiffyDOS™- of Commodore™ 128-protocol ondersteunt.

Om bijvoorbeeld de Attention-lijn vrij te geven, schrijft u de juiste commandobytewaarde (65 = $41) naar het commandoregister op $D698 en controleert u vervolgens de voltooiingsstatus in het statusregister op $D697, zoals weergegeven in het volgende voorbeeld in BASIC65:

```
10 POKE $D698,$41
20 IF ( PEEK($D697) AND $20 ) = $00 GOTO 20
30 PRINT "GEREED"
```

De 4541 implementeert een reeks opdrachten die nauw aansluiten bij de KERNAL-aanroepen die worden gebruikt om de IEC-bus op de C64 en verwante computers te besturen. In de meeste gevallen is er één bijbehorende opdracht voor de 4541, hoewel u in een aantal gevallen twee opdrachten moet geven, zoals samengevat in de volgende tabel:

KERNAL-aanroep	**Betekenis en bijbehorende 4541-opdracht(en)**
$FF93 LSTNSA	Verzend het secundaire LISTEN-adres. 4541-equivalent: Gegevens = Binair OR van $60 en het gewenste secundaire adres. Opdracht $30. Vervolgens opdracht $41 om de Attention-lijn vrij te geven.
$FF96 TALKSA	Verzend het secundaire TALK-adres. 4541-equivalent: Gegevens = Binair OR van $60 en het gewenste secundaire adres. Opdracht $30. Vervolgens opdracht $41 om de Attention-lijn vrij te geven.
$FFA5 IECIN	Ontvang een byte van de seriële randapparatuurbus. 4541-equivalent : Opdracht $32. De ontvangen byte is beschikbaar in het gegevensregister na voltooiing.
$FFA8 IECOUT	Verzend een byte naar de seriële randapparatuurbus. 4541-equivalent : Gegevens = de te verzenden byte. Opdracht $31 (of $30 indien de byte onder Attention wordt verzonden).
$FFA8 UNTALK	Verzend UNTALK-opdracht naar de seriële randapparatuurbus. 4541-equivalent : Gegevens = $5F. Opdracht $30.

KERNAL-aanroep	Betekenis en bijbehorende 4541-opdracht(en)
$FFAB UNLISTN	Verzend UNLISTEN-opdracht naar de seriële randapparatuurbus. 4541-equivalent : Gegevens = $3F. Opdracht $30.
$FFB1 LISTEN	Verzend het LISTEN-commando naar de seriële randapparatuurbus. 4541-equivalent: Gegevens = $20 plus het apparaatnummer. Opdracht $30.
$FFB4 TALK	Verzend het TALK-commando naar de seriële randapparatuurbus. 4541-equivalent: Gegevens = $40 plus het apparaatnummer. Opdracht $30.
$FFB7 READSS	Lees de status van de seriële randapparatuurbus. 4541-equivalent: Lees de statusbits vanaf $D698. Voor het gemak hebben de bovenste bits van deze statusbyte dezelfde indeling als in de KERNAL.

De 4541 is zeer gehoorzaam: vraagt u hem iets nieuws, dan begint hij daar onmiddellijk mee, zelfs midden in een andere taak – zelfs al is hij halverwege het verzenden van een byte gegevens naar de seriële randapparatuurbus!

U dient dan ook altijd te wachten totdat de IRQREADY-bit in $D697 is gezet voordat u een commando geeft of de statusbits leest, zodat u er zeker van bent dat de controller de laatste gevraagde taak heeft afgerond.

VOORBEELDEN

De status van het DOS-kanaal lezen

Het volgende BASIC65-programma kan worden gebruikt om met een apparaat op de seriële randapparatuurbus te communiceren. Omdat het de 4541 gebruikt en niet de CBDOS van de C65/MEGA65, negeert dit programma het interne 3.5"-schijfstation, omdat dat station niet op de seriële randapparatuurbus is aangesloten.

```
10 REM BREEK EEN EVENTUEEL LOPEND 4541-COMMANDO af
20 POKE $D698,$00
30 REM RESET DE SERIELE RANDAPPARATUURBUS DOOR
40 REM DE RESETPIN 1 SECOND LANG LAAG TE HOUDEN
50 POKE $D698,$72 : SLEEP 1 : POKE $D698,$52
60 REM GUN AANGESLOTEN APPARATEN NA RESET TIJD OM GEREED TE KOMEN
70 SLEEP 2: REM HET DUURT EVEN VOOR DE 1541 GERESET IS!
80 REM SELECTEER HET APPARAATNUMMER WAARNAAR GELUISTERD WORDT
90 D = 8
100 PRINT"DEVICE"; D; "TALK"
110 POKE $D699,$40 + D : POKE $D698,$30 : GOSUB 1000
120 PRINT "SECUNDAIR ADDRES 15"
130 POKE $D699,$6F : POKE $D698,$30 : GOSUB 1000
140 PRINT"OMKEREREN NAAR LUISTEREN"
150 POKE $D698,$35 : GOSUB 1000
160 PRINT "LEES DOSSTATUS"
170 POKE $D698,$32 : GOSUB 1000
180 PRINT CHR$( PEEK($D699) );
190 CC = CC + 1 : IF CC < 200 GOTO 180
999 END
1000 REM WACHT TOT DE 4541 HET COMMANDO HEEFT AFGEROND
1010 S=PEEK($D697) : IF (S AND 32) = 32 GOTO 1020
1020 GOTO 1000
1030 IF S AND 128 THEN PRINT "DEVICE NIET AANWEZIG" : END
1040 RETURN
```

Als u een 1581 met een JiffyDOS™-ROM heeft aangesloten, toont dit programma bij uitvoering het volgende:

```
READY.
RUN
DEVICE 8 TALK
SECUNDAIR ADRES 15
OMKEREN NAAR LUISTEREN
LEES DOSSTATUS
73,(C) 1989 JIFFYDOS 6.0 1581,00,00
00, OK,00,00
00, OK,00,00
00, OK,00,00
...
00, OK,0
READY.
```

COMMANDOREFERENTIE

De volgende tabel geeft een overzicht van de door de 4541 ondersteunde commandocodes.

Commandobyte	Actie
Lopende commando's onderbreken	
$00	Onderbreek elk uitgevoerd commando.
$01	Herstel de toestand van de controller: ATN, CLK, DATA en SRQ loslaten en de standaard protocolselectie inschakelen.
Bit-bashing-commando's	
$41	Attention-lijn (ATN) loslaten naar 5V.
$61	Attention-lijn (ATN) naar 0V trekken.
$43	Kloklijn (CLK) loslaten naar 5V.
$63	Kloklijn (CLK) naar 0V trekken.
$44	Gegevenslijn (DATA) loslaten naar 5V.
$64	Gegevenslijn (DATA) naar 0V trekken.
$53	Snelle seriële kloklijn (SRQ) loslaten naar 5V.
$73	Snelle seriële kloklijn (SRQ) naar 0V trekken.
$52	Laat de resetlijn (RESET) los op 5V.
$72	Trek de resetlijn (RESET) naar 0V.
$4C	Laat alle lijnen (ATN, CLK, DATA, SRQ, RESET) los op 5V.
Protocolvariant-besturingscommando's	
$4A	Sta het aanvragen van de JiffyDOS™ uitbreiding voor het seriële protocol toe.
$6A	Sta het aanvragen van de JiffyDOS™ uitbreiding voor het seriële protocol niet toe.
$46	Sta het aanvragen van de Commodore™ 1571/1581 snelle seriële uitbreiding voor het seriële protocol toe.
$66	Sta het aanvragen van de Commodore™ 1571/1581 snelle seriële uitbreiding voor het seriële protocol niet toe.

Commandobyte	Actie
$50	Sta het aanvragen van alle seriële uitbreidingen toe.
$70	Sta het aanvragen van alle seriële uitbreidingen niet toe.

Seriële busprotocolcommando's

$30	Verzend een byte in $D699 onder Attention. De Attention-lijn wordt aan het einde van de transactie op 0V gehouden.
$31	Verzend een byte in $D699 zonder Attention. De Attention-lijn wordt losgelaten indien deze voorafgaand was geactiveerd.
$32	Ontvang een byte van de seriële randapparatuurbus. Er moet een vooraf geactiveerde spreker ("**talk**er") aanwezig zijn. De ontvangen byte wordt opgeslagen in $D699.
$33	GERESERVEERD
$34	Verzend een byte en geef einde-aanduiding (EOI) aan. De Attention-lijn wordt voorafgaand aan de verzending losgelaten indien deze was geactiveerd.
$35	Draai om van spreker ("**talk**er") naar luisteraar ("**listen**er").

Commando's voor de protocoltiming

$80	Reset de protocoltiming naar de standaardwaarden.
$81	Stel de T_R-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$82	Stel de T_{TK}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$83	Stel de T_{DC}-time-out in (1 - 255 milliseconden). De oude waarde kan worden gelezen uit $D699.
$84	Stel de T_{BB}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$85	Stel de T_{HA}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$86	Stel de T_{ST}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.

Commandobyte	Actie
$87	Stel de T_{VT}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$88	Stel de T_{AL}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$89	Stel de T_{AC}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$8A	Stel de T_{AT}-vertraging in (1 - 255 milliseconden). De oude waarde kan worden gelezen uit $D699.
$8B	Stel de T_H-vertraging in (1 - 255 milliseconden). De oude waarde kan worden gelezen uit $D699. *Deze waarde heeft momenteel geen effect, omdat de 4541 een werkelijk oneindige wachttijd door een luisteraar mogelijk maakt, met uitzondering van bytes die onder Attention worden verzonden, waarbij in plaats daarvan T_{HA} wordt gebruikt.*
$8C	Stel de T_{NE}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$8D	Stel de T_F-vertraging in (4 - 1020 microseconden). De oude waarde kan worden gelezen uit $D699, geschaald met een factor 4.
$8E	Stel de T_{YE}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$8F	Stel de T_{EI}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$90	Stel de T_{AR}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$91	Stel de T_{JT}-vertraging in (4 - 1020 microseconden). De oude waarde kan worden gelezen uit $D699, geschaald met een factor 4.
$92	Stel de T_{J0}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$93	Stel de T_{J1}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$94	Stel de T_{J2}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$95	Stel de T_{J3}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.

Commandobyte	Actie
$96	Stel de T_{J4}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$97	Stel de T_{J5}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$98	Stel de T_{J6}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$99	Stel de T_{J7}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$9A	Stel de T_{J8}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$9B	Stel de T_{J9}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$9C	Stel de T_{J10}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$9D	Stel de T_{J11}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$9E	Stel de T_{JR}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$9F	Stel de T_{FS}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$A0	Stel de T_{FF}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$A1	Stel de T_{PULLUP}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.
$A2	Stel de T_{JD}-vertraging in (4 - 1020 microseconden). De oude waarde kan worden gelezen uit $D699, geschaald met een factor 4.
$A3	Stel de T_{J12}-vertraging in (1 - 255 microseconden). De oude waarde kan worden gelezen uit $D699.

Diagnostische opdrachten

$D0	Activeer de optionele geïntegreerde datalogger indien ingeschakeld.
$D1	Begin met het uitzenden van een 1 kHz-pulstrein op de CLK- en DATA-lijnen. Dit duurt voort totdat een ander commando het onderbreekt.

REGISTERTABEL

De 4541 heeft de volgende registers:

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D694	54932	DATALOG0							
D695	54933	DATALOG1							
D697	54935	IRQFLAG	IRQRX	IRQRDY	IRQTO	IRQEN	IRQRXEN	IRQRDYEN	IRQTOEN
D698	54936	STNODEV	STNOEOI	STSRQ	STVERIFY	STC	STD	STTO	STDDIR
D699	54937	DATA							
D69A	54938	PRESENT	PROT		DIATN	DEVNUM			

- **DATA** De gegevensbyte die van de IEC-bus wordt gelezen
- **DATALOG0** Toegang tot de geïntegreerde datalogger in de IEC-controller
- **DATALOG1** Toegang tot de geïntegreerde datalogger in de IEC-controller
- **DEVNUM** De onderste 4 bits van het op dat moment geselecteerde apparaatnummer
- **DIATN** Het apparaat wordt momenteel in de attention-toestand gehouden
- **IRQEN** Schakelt de interrupts in indien deze is gezet
- **IRQFLAG** Interruptvlag. Deze wordt gezet indien een IRQ-gebeurtenis wordt geactiveerd.
- **IRQRDY** Gezet indien gereed om een opdracht te verwerken
- **IRQRDYEN** Schakelt de TX-interruptbron in indien deze is gezet
- **IRQRX** Gezet indien een byte van een luisteraar is ontvangen.
- **IRQRXEN** Schakelt de RX-interruptbron in indien deze is gezet
- **IRQTO** Gezet indien een protocol-timeout is opgetreden, bijv. wanneer het apparaat niet wordt gevonden.
- **IRQTOEN** Schakelt de timeout-interruptbron in indien deze is gezet
- **PRESENT** Het apparaat is aanwezig
- **PROT** Protocolondersteuning van het apparaat (5=C128/C65 FAST, bit 6 = JiffyDOS(tm))
- **STC** Toestand van de CLK-lijn
- **STD** Toestand van de DATA-lijn

- **STDDIR** De gegevensrichting op het moment dat een timeout optreedt.
- **STNODEV** Het apparaat is niet aanwezig
- **STNOEOI** End of Indicate (EOI/EOF)
- **STSRQ** Toestand van de SRQ-lijn
- **STTO** Er is een timeout opgetreden
- **STVERIFY** Er is een verificatiefout opgetreden

SERIËLE BUSTIMING

Dit hoofdstuk beschrijft de tijdsvereisten voornamelijk vanuit het perspectief van een buscontroller. Indien u seriële busapparaten ontwerpt, besteed dan zorgvuldig aandacht aan de manier waarop deze tijdsvereisten de apparaten beïnvloeden.

Een byte onder Attention verzenden

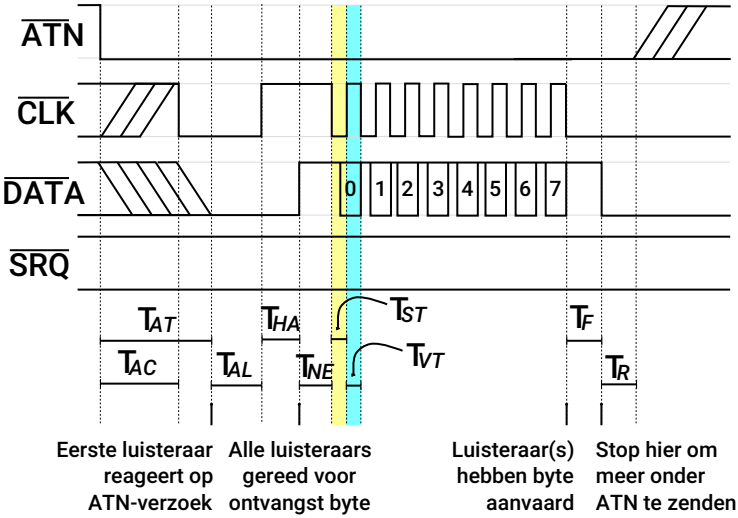
Wanneer de controller de aandacht van één of meer apparaten op de bus wil trekken, gebruikt hij de ATN-lijn om dit aan te geven: hij trekt deze laag naar 0V en stuurt vervolgens één of meer bytes die door alle apparaten op de bus worden ontvangen.

De SRQ-lijn is niet actief in deze transactie.

Samengevat:

1. De controller trekt de ATN naar 0V en laat de CLK-, DATA- en SRQ-lijnen los indien hij een van deze op 0V hield.
2. Alle apparaten op de bus merken op dat de ATN-lijn op 0V staat en geven dit aan door de DATA-lijn naar 0V te trekken. De controller kan niet zien noch hoeveel apparaten er hebben gereageerd, noch of alle apparaten op de bus hebben gereageerd; hij kan enkel vaststellen dat ten minste één apparaat heeft gereageerd.
3. Indien geen enkel apparaat reageert, blijft DATA op 5V, omdat niemand deze naar beneden trekt, en geeft de controller een DEVICE NOT PRESENT-fout aan.
4. Terwijl de controller wacht op reactie van de apparaten, trekt hij ook de CLK-lijn naar 0V, doorgaans kort nadat hij de ATN-lijn naar 0V heeft getrokken.
5. Nadat de controller enige tijd heeft gewacht, doorgaans 1 milliseconde, neemt hij aan dat alle apparaten die op het ATN-verzoek zullen reageren, dat inmiddels hebben gedaan.
6. Vervolgens laat de controller de CLK-lijn los naar 5V om aan te geven dat hij een byte gegevens op de bus wil verzenden. Dit vormt het begin van het verzenden van een byte onder Attention. Al deze voorgaande stappen hadden tot doel de ATN-communicatie tot stand te brengen.

7. De controller wacht nu tot alle apparaten hun bereidheid om een byte te ontvangen aangeven door de DATA-lijn terug naar 5V te laten gaan. Omdat de bus open collector is, kan zelfs één enkel apparaat dat nog niet gereed is, de DATA-lijn naar 0V blijven trekken, waardoor de controller moet wachten. Indien dit te lang duurt, kan de controller opgeven en een time-out aangeven.
8. Zodra het laatste apparaat heeft aangegeven gereed te zijn om gegevens te ontvangen, wacht de controller nog iets langer om er zeker van te zijn dat alle apparaten hun laatste voorbereidingen voor het ontvangen van een byte kunnen afronden. Dit duurt niet lang.
9. Na die wachttijd begint de controller met het verzenden van de 8 bits gegevens, te beginnen met het minst significante bit (LSB), oftewel bit 0. Voor elk bit trekt hij eerst de CLK-lijn laag om aan te geven dat er gegevens op de bus worden geladen, trekt de DATA-lijn naar 0V als hij een 0-bit wenst te verzenden, of laat deze naar 5V zweven als hij een 1-bit wenst te verzenden, en laat vervolgens de CLK-lijn weer los naar 5V en houdt deze daar enige tijd. De timing van dit proces is van kritiek belang: als de CLK te kort laag of hoog is, raken de apparaten in de war en kunnen zij een of meer bits missen, wat op de bus tot volledige chaos leidt.
10. Nadat de controller het laatste bit heeft verzonden, houdt hij de CLK-lijn laag en laat de DATA-lijn los. Hij verwacht dat een of meer apparaten binnen korte tijd de DATA-lijn naar 0V trekken om de ontvangst van de byte te bevestigen. Indien dit niet gebeurt, kan de controller een time-out of een DEVICE NOT PRESENT-fout melden.
11. Aan het einde hiervan houdt de controller de CLK-lijn op 0V en houden de apparaten de DATA-lijn op 0V. Deze combinatie dient om elkaar te vertellen dat de controller nog geen volgende byte wenst te verzenden en dat de apparaten nog niet gereed zijn om de volgende byte te ontvangen.
12. Indien de controller meer bytes onder Attention wenst te verzenden, herhaalt hij dit proces vanaf de stap waarin hij de CLK-lijn naar 5V liet los.
13. Indien het echter de laatste byte betrof die onder deze Attention-aanvraag werd verzonden, wacht de controller een korte tijd en laat vervolgens de ATN-lijn los.



Symbool	Min	4541	Max	Omschrijving
T_{AT}	-	1000	∞	Reactietijd van het apparaat op Attention. *Een apparaat moet binnen deze tijd op het ATN-verzoek reageren. Indien geen enkel apparaat binnen deze tijd reageert, volgt een DEVICE NOT PRESENT-fout. Apparaten dienen derhalve de typische waarde van 1000 microseconden aan te houden.*
T_{AC}	-	20	∞?	Tijd tussen het laag trekken van ATN en het laag trekken van CLK.
T_{AL}	-	1000	∞?	Tijd tussen de reactie van het eerste apparaat op ATN en het loslaten van CLK door de controller naar 5V.
T_{HA}	-	64 ms	∞	Luisteraar-wachttijd. *Het protocol staat een luisteraar toe om gedurende een willekeurige periode of helemaal niet te wachten. De 4541 staat dit niet toe bij ATN-verzoeken.*
T_{NE}	-	40	200	Niet-EOI-tijdskanaalreactie op "Gereed voor gegevens". *Indien dit interval te lang is, wordt de byte als een EOI-byte geïnterpreteerd en zullen de apparaten een EOI-reactie vereisen (zie T_{EI}).*
T_{ST}	20*	35	∞	Bitinsteltijd. *Wordt aangeduid als T_S in de C64 Programmer's Reference Guide.*

Symbool	Min	4541	Max	Omschrijving
T_{VT}	20*	35	∞	Bitgeldigheid vasthoudtijd. *Wordt aangeduid als T_V in de C64 Programmer's Reference Guide.*
T_F	-	1000	1000	Frame-handshake (bevestiging) time-out.
T_R	20	200	∞	Tijd tussen het ontvangen van de framebevestiging en het loslaten van de ATN-lijn naar 5V.

Alle tijdseenheden zijn uitgedrukt in microseconden, tenzij anders aangegeven.

* De Commodore™ 64 Programmer's Reference Guide stelt dat T_S en T_V elk een minimumduur van 20 microseconden dienen te hebben. Onze onderzoeken wijzen uit dat bij communicatie met een 1541 veilige waarden voor T_{ST} en T_{VT} dichter bij 70 microseconden liggen.

JiffyDOS™-protocolaanvraag

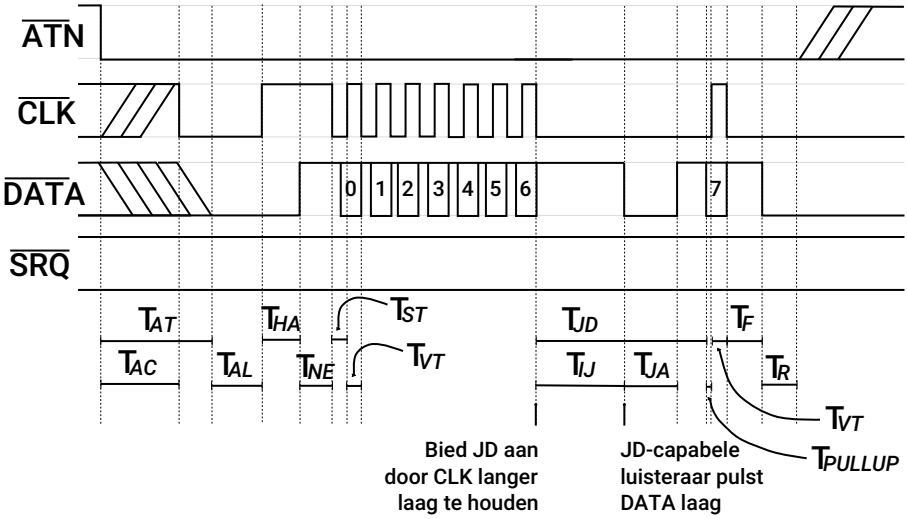
Indien ondersteuning voor de JiffyDOS™-protocoluitbreiding is ingeschakeld, vraagt de 4541 deze slechts aan tijdens de verzending van TALK- en LISTEN-commando's. Dat wil zeggen: wanneer een byte onder Attention wordt gestuurd die valt in het bereik $20 - $5F.

De SRQ-lijn is niet actief in deze transactie.

Samengevat:

1. De byte wordt normaal verzonden, tot het T_{ST}-interval vlak voor het laatste bit (bit 7) van de byte.
2. In plaats van T_{ST} af te wachten, moet de controller een veel langere periode afwachten, doorgaans 300 microseconden, zodat de luisteraar herkent dat hem gevraagd wordt of hij het JiffyDOS™ protocol ondersteunt. De controller laat ook de DATA-lijn naar 5V los.
3. Indien de luisteraar het JiffyDOS™ protocol ondersteunt, trekt hij DATA naar 0V wanneer hij opmerkt dat de CLK-lijn gedurende de langere periode laag is gehouden.
4. Nadat de luisteraar DATA lang genoeg op 0V heeft gehouden om er zeker van te zijn dat de controller dit heeft kunnen opmerken, laat hij de DATA-lijn los naar 5V. Het apparaat zal nu het JiffyDOS™ gebruiken voor de gevraagde TALK- of LISTEN-sessie.
5. De controller noteert of de DATA-lijn naar 0V is getrokken, en registreert in dat geval dat de luisteraar het JiffyDOS™-protocol heeft geselecteerd en dit protocol zal gebruiken voor de gevraagde TALK- of LISTEN-sessie.

6. De controller stuurt nu het laatste bit van de byte.
7. De controller zorgt ervoor dat de bitwaarde van het databit zichtbaar is op DATA voordat hij CLK activeert, doorgaans 1 - 4 microseconden.
8. De controller laat de CLK-lijn los naar 5V om aan de luisteraar aan te geven dat het laatste databit gereed is.
9. De rest van de byte wordt normaal overgedragen.



Symbool	Min	4541	Max	Omschrijving
T_{JD}	320	320	∞	Tijd dat CLK op 0V wordt gehouden tussen de twee laatste bits van de byte om het nevenkanaal te markeren; via het nevenkanaal geeft de controller aan dat hij de JiffyDOS™-protocoluitbeiding wenst te gebruiken.
T_{IJ}	100	-	295	Tijd voordat de luisteraar de selectie van de JiffyDOS™-protocoluitbreiding aanvaardt en de bevestiging ervan begint.
T_{IJ}	4/100*	-	200	Tijd dat de luisteraar de DATA-lijn laag houdt om de selectie van de JiffyDOS™-protocoluitbreiding te bevestigen.

Alle tijdseenheden zijn uitgedrukt in microseconden, tenzij anders aangegeven.

* De 4541 kan pulsbreedtes van slechts 4 microseconden detecteren. Software-implementaties van het protocol vereisen echter een grotere waarde. JiffyDOS™gebruikt 100 microseconden.

Verzenden van controller naar apparaat onder JiffyDOS™

In tegenstelling tot het standaard seriële protocol maakt het JiffyDOS™-protocol gebruik van verschillende protocollen voor verzenden en ontvangen. Dit is noodzakelijk bij software-implementaties om de maximale snelheid te bereiken, omdat de seriële apparaatlijnen op verschillende registerbits zijn gemapt, en een onderdeel van de kracht van het JiffyDOS™-protocol bestaat uit de slimme manipulatie van de ontvangen bits om het complete byte in de minst mogelijke tijd te reconstrueren, en eveneens bij het verzenden.

Het resultaat is opmerkelijk: JiffyDOS™ kan een byte verzenden in ongeveer dezelfde tijd die het originele protocol nodig heeft om één bit te verzenden!

Let op dat het JiffyDOS™-protocol databits met de tegenovergestelde polariteit verzendt vergeleken met het standaard protocol: een 1-bit wordt aangegeven met 0V, terwijl een 0-bit wordt aangegeven met 5V.

De SRQ-lijn is niet actief bij deze transactie.

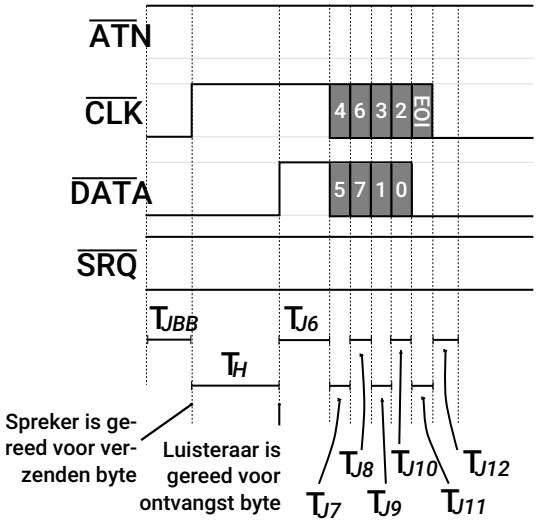
Samengevat:

1. Afhankelijk van de toestand van de controller op het moment dat deze met verzenden begint, houdt hij mogelijk de CLK-lijn op 0V. Zo ja, dan laat hij deze onmiddellijk los.
2. Vervolgens geeft het apparaat na een willekeurige tijd aan dat het gereed is om een byte van de controller te ontvangen, door de DATA-lijn naar 5V los te laten.
3. De timing voor het resterende deel van de byteoverdracht is nu cruciaal, omdat het apparaat direct na het loslaten van DATA de CLK- en DATA-signalen gaat uitlezen om de byte over te dragen.
4. De controller wacht enkele microseconden (T_{J6}).
5. De controller zet CLK op 0V als databit 4 gelijk is aan 1, en laat deze anders naar 5V los. De controller zet DATA op 0V als databit 5 gelijk is aan 1, en laat deze anders naar 5V los.
6. De controller wacht enkele microseconden (T_{J7}).
7. De controller zet CLK op 0V als databit 6 gelijk is aan 1, en laat deze anders naar 5V los. De controller zet DATA op 0V als databit 7 gelijk is aan 1, en laat deze anders naar 5V los.
8. De controller wacht enkele microseconden (T_{J8}).
9. De controller zet CLK op 0V als databit 3 gelijk is aan 1, en laat deze anders naar 5V los. De controller zet DATA op 0V als databit 1 gelijk is aan 1, en laat deze anders naar 5V los.
10. De controller wacht enkele microseconden (T_{J9}).

11. De controller zet CLK op 0V als databit 2 gelijk is aan 1, en laat deze anders naar 5V los. De controller zet DATA op 0V als databit 0 gelijk is aan 1, en laat deze anders naar 5V los.
12. De controller wacht enkele microseconden (T_{J10}).
13. De controller trekt de DATA-lijn naar 0V. Hij laat ook de CLK-lijn naar 5V los, tenzij hij EOI wil aangeven, in welk geval hij de CLK-lijn naar 0V trekt.
14. De controller wacht enkele microseconden, gedurende welke tijd het apparaat de EOI-waarde uitleest, en trekt vervolgens DATA naar 0V. (T_{J11}).
15. De byteoverdracht is voltooid.

Let er bij het verzenden van de eerste byte onder het JiffyDOS™protocol op dat het CLK-signaal op 0V kan staan. De tijd tussen bytes (T_{JBB}) is niet vereist (wordt op nul gezet), omdat de korte ontvanglus op het apparaat eenvoudigweg aangeeft wanneer het gereed is om de volgende byte te ontvangen, wat ook bijdraagt aan de hoge snelheid.

Let ook op dat de onderstaande timing gebaseerd is op het moment waarop de signalen ingesteld moeten worden. Bij software-implementaties moet de latentie van de benodigde processorinstructies in aanmerking worden genomen. Dit vormt geen probleem bij de 4541, omdat deze een latentie van minder dan 50 ns heeft.



Symbool	Min	4541	Max	Omschrijving
T_{JBB}	0	0	∞	Tijd tussen bytes. Er is geen minimale tijd tussen bytes bij het JiffyDOS™-protocol, mits T_{J12} voldoende groot is, omdat dat de latentie van de ontvangroutine effectief verbergt.
T_{J6}	10	10	10	JiffyDOS™ verzenden: insteltijd ontvanger.
T_{J7}	13	13	13	JiffyDOS™ verzenden: vasthoudtijd, eerste di-bit.
T_{J8}	11	11	11	JiffyDOS™ verzenden: vasthoudtijd, tweede di-bit.
T_{J9}	11	11	11	JiffyDOS™ verzenden: vasthoudtijd, derde di-bit.
T_{J10}	12	12	12	JiffyDOS™ verzenden: vasthoudtijd, vierde di-bit.
T_{J11}	15	15	15	JiffyDOS™ verzenden: vasthoudtijd, statusbits. CLK draagt de EOI-melding, en DATA wordt op 0V gezet om een succesvolle verzending aan te geven. De ontvanger kan een framefout aangeven indien DATA 5V wordt waargenomen.
T_{J12}	18	18	18	JiffyDOS™ hersteltijd na verzending. CLK wordt naar 0V getrokken, en DATA wordt op 0V gezet om een succesvolle verzending aan te geven. De ontvanger kan een framefout aangeven indien DATA 5V wordt waargenomen.
T_{JR}	15	15	15	

De overige tijden zijn gelijk aan die bij het verzenden van een byte onder Attention.

Alle tijdseenheden zijn uitgedrukt in microseconden, tenzij anders aangegeven.

Ontvangst van randapparaat naar controller onder JiffyDOS™

In tegenstelling tot het standaard seriële protocol gebruikt het JiffyDOS™-protocol verschillende protocollen voor verzenden en ontvangen. Dit is noodzakelijk bij software-implementaties om de maximale snelheid te bereiken, omdat de seriële randapparatuurbus-lijnen op verschillende registerbits worden gemapt. Een essentieel onderdeel van het JiffyDOS™-protocol is hoe het op intelligente wijze de ontvangen bits manipuleert om het volledige byte in de kortst mogelijke tijd te reconstrueren, en op vergelijkbare wijze bij verzending.

Het resultaat is opmerkelijk: JiffyDOS™ kan een byte verzenden in ongeveer dezelfde tijd die het originele protocol nodig heeft om één bit te verzenden.

Bij verzending via het JiffyDOS™-protocol worden de databits geïnverteerd; bij ontvangst van een randapparaat dat het JiffyDOS™-protocol gebruikt, worden de bits op natuurlijke wijze weergegeven, oftewel met 1 = 5V en 0 = 0V.

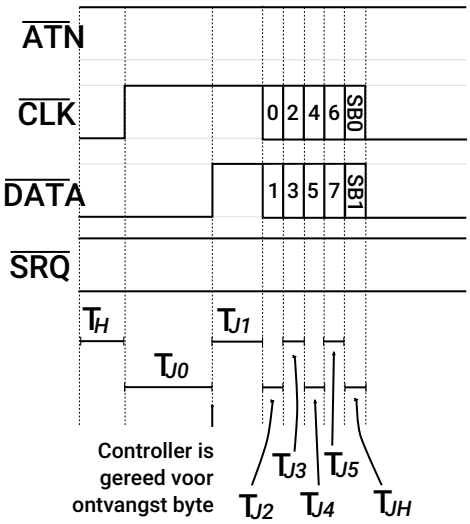
De SRQ-lijn is niet actief in deze transactie.

Ter samenvatting:

1. Het randapparaat kan een willekeurige periode wachten voordat het een byte naar de controller kan verzenden. Wanneer het zover is, laat het de CLK-lijn los naar 5V.
2. Wanneer de controller dit opmerkt, laat hij de DATA-lijn los naar 5V en begint het kritieke tijdsgebied van het protocol.
3. Het randapparaat wacht kort. Voor apparaten met een software-implementatie, oftewel alle Commodore™-stations, is de vertraging een natuurlijk gevolg van de processorinstructie die nodig is om het verzenden van de byte in te luiden; hetzelfde geldt voor de volgende stappen.
4. Het randapparaat plaatst het eerste di-bit van gegevens op de CLK- en DATA-lijnen en wacht kort.
5. Het randapparaat plaatst het tweede di-bit van gegevens op de CLK- en DATA-lijnen en wacht kort.
6. Het randapparaat plaatst het derde di-bit van gegevens op de CLK- en DATA-lijnen en wacht kort.
7. Het randapparaat plaatst het vierde di-bit van gegevens op de CLK- en DATA-lijnen en wacht kort.
8. Het randapparaat plaatst de statusbits van de byte op de CLK- en DATA-lijnen en wacht kort.
9. De controller trekt DATA naar 0V om de ontvangst van de byte te bevestigen en te voorkomen dat het randapparaat nog een byte verzendt voordat hij gereed is.
10. Het randapparaat trekt de CLK-lijn naar 0V indien het niet onmiddellijk nog een byte gereed heeft om te verzenden; in het andere geval blijft de lijn op 5V.

De statusbits hebben de volgende betekenis:

CLK	**DATA**	**Betekenis**
0V	0V	Er is een fout opgetreden.
0V	5V	Byte ontvangen met einde-aanduiding (EOI).
5V	0V	Normale byte ontvangen (geen EOI).
5V	5V	Er is een fout opgetreden.



Symbool	Min	4541	Max	Omschrijving
T_{JH}	10	–	∞	Wachttijd van het randapparaat totdat het gegevens heeft om te verzenden.
T_{J0}	0	–	∞	Wachttijd van de controller totdat deze gereed is om een byte te ontvangen.
T_{J1}	37	37	∞	Deze tijd wordt door software-implementaties gebruikt om de byte voor te bereiden voor onmiddellijke verzending.
T_{J2}	14	14	14	Eerste di-bit van gegevens verzenden
T_{J3}	10	10	10	Tweede di-bit van gegevens verzenden
T_{J4}	11	11	11	Derde di-bit van gegevens verzenden
T_{J5}	11	11	11	Vierde di-bit van gegevens verzenden
T_{JH}	13	13	13	Statusbits verzenden

De overige tijden zijn gelijk aan die bij het verzenden van een byte onder Attention.

Alle tijdseenheden zijn uitgedrukt in microseconden, tenzij anders aangegeven.

Omschakelen van spreker naar luisteraar

Wanneer de buscontroller een apparaat beveelt om te verzenden, wordt de transactie afgesloten met ATN op 0V. De controller houdt de CLK-lijn op 0V, en het apparaat dat

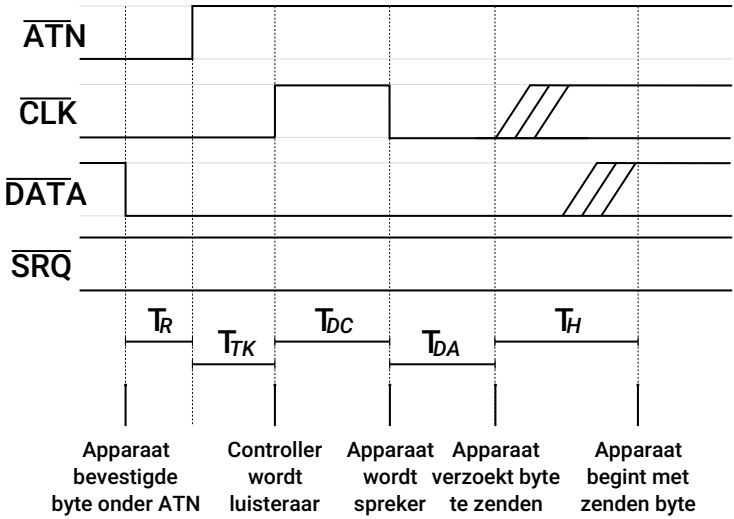
verzond hield de DATA-lijn vast. Deze situatie moet worden omgekeerd: de controller moet de besturing van de bus afstaan en deze overdragen aan het apparaat.

De SRQ-lijn is niet actief bij deze transactie.

Ter samenvatting:

1. De controller wacht lang genoeg voordat hij de ATN-lijn vrijgeeft, om er zeker van te zijn dat het randapparaat de bevestiging van de byte die onder Attention werd verzonden, heeft afgerond. Dit is de vertraging T_R.
2. De controller geeft de ATN-lijn vrij naar 5V.
3. De controller wacht enige tijd. Dit is de vertraging T_{TK}.
4. De controller geeft de CLK-lijn vrij naar 5V en trekt de DATA-lijn naar 0V. Vanaf dit moment heeft de controller de besturing van de bus afgestaan en wordt hij luisteraar.
5. Na een bepaalde tijdsduur trekt het randapparaat dat op het punt staat te spreken, de CLK-lijn naar 0V en geeft de DATA-lijn vrij, die op 0V blijft staan omdat de controller deze nog steeds naar 0V trekt ter voorbereiding op het ontvangen. Vanaf dit moment is het randapparaat de enige spreker op de bus. De vertraging voordat de CLK-lijn door het randapparaat naar 0V wordt getrokken, is de time-out T_{DC}.
6. Het randapparaat wacht enige tijd voordat het met spreken mag beginnen. Dit is de vertraging T_{DA}.

Na voltooiing van dit proces zijn de rollen van controller en randapparaat omgekeerd, waarbij het randapparaat nu de rol van spreker ("**talk**er") heeft en de controller (en mogelijk andere apparaten op de bus) die van luisteraar ("**listen**er").



Symbool	Min	4541	Max	Omschrijving
T_{DA}	4/80*	-	∞	Duur van de bevestiging dat apparaat nu spreker is. De controller houdt CLK.
T_{DC}	0	64 ms	∞	Duur van de zendbevestiging, d.w.z. de tijd die een randapparaat heeft voordat het spreker wordt. *Kan onmiddellijk beginnen zodra CLK op 5V wordt waargenomen.*
T_H	-	N/A	∞	Luisteraar-wachttijd. *De luisteraar kan de ontvangstbevestiging willekeurig lang uitstellen of ook helemaal niet.*
T_R	0/20^	200	-	Tijd van frame tot vrijgave van ATN.
T_{TK}	20	40	100	Vrijgave van de zendvoorwaarde.

Alle tijdseenheden zijn uitgedrukt in microseconden, tenzij anders aangegeven.

* T_{DA} wordt geleverd door het randapparaat, niet door de controller. De 4541 is in staat veel sneller op seriële bus-gebeurtenissen te reageren dan een C64 en heeft derhalve slechts 4 microseconden nodig om te garanderen dat de CLK-lijn de tijd heeft om naar 5V te stijgen. Voor controllers met een softwarematige implementatie van het protocol, zoals de C64, bedraagt het minimum 80 microseconden. Randapparaten dienen derhalve steeds een waarde van ten minste 80 microseconden voor T_{DA} te gebruiken.

^ Het Commodore™ 64 Programmer's Reference Guide vermeldt dat T_R een minimumduur van 20 microseconden heeft. Wij zien geen aanleiding om te veronderstellen dat enige implementatie een vertraging vereist voordat de ATN-lijn kan worden vrijgegeven na het overdragen van een byte.

Byte verzenden met einde-aanduiding (EOI)

Na het verzenden van een stroom gegevens wil een apparaat de luisteraar ("**listen**er") er vaak van op de hoogte brengen dat het einde van de gegevens is bereikt, bijvoorbeeld zodat deze met de verwerking kan beginnen. Dit gebeurt door de byte op een bijzondere manier te markeren met het einde-aanduiding (EOI)-attribuut.

Over de betekenis van de afkorting EOI bestaat spraakverwarring. Op de parallele IEE-488 bus heet één van de pinnen in de aansluiting *end or identify*. Deze pin heeft inderdaad een dubbele functie: Als ATN niet actief is, dan dient de pin als einde-aanduiding (*end*). Is ATN wel actief, dan dient de pin om parallel de status van apparaten op te vragen (*identify*). Op de seriële Commodore-bus is de *identify*-functie niet aanwezig en is EOI te allen tijde een einde-aanduiding. Wellicht hierom wordt EOI, zelfs in officiële Commodore-documenten, vaak beschreven als *end of indicator*. De meest triviale manier om dit te lezen, "einde van (een) aanduiding", is logisch onzinnig. De juiste manier om dit te lezen is "eind van"-aanduiding, de aanduiding van het

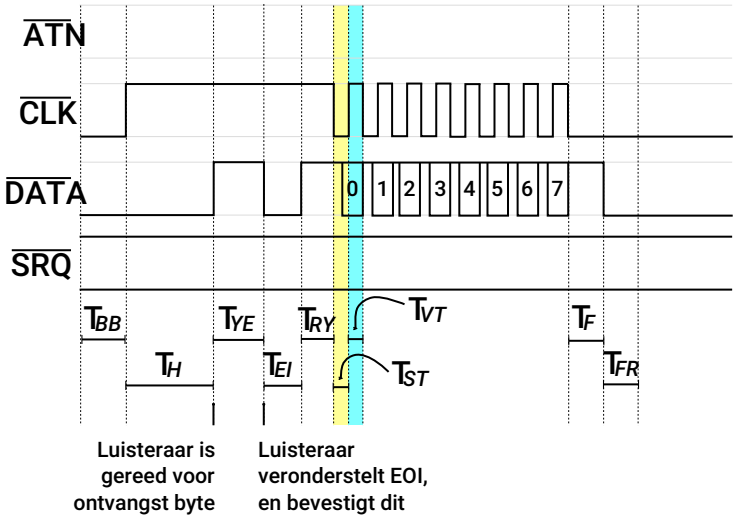
eind van gegevensoverdracht. Dit is weer een mooi voorbeeld waarin begrippen in de Engelse taal voor meerdere uitleg vatbaar zijn. We zullen deze spraakverwarring niet meenemen naar het Nederlands en in de handleiding stelselmatig "einde-aanduiding (EOI)" gebruiken.

De einde-aanduiding wordt gerealiseerd met behulp van een timing-nevenkanaal, zodat er geen afzonderlijke ader in de kabel voor nodig is. Wanneer de spreker de CLK-lijn naar 5V heeft gebracht om aan te geven dat hij een byte wil verzenden, en de luisteraar de DATA-lijn naar 5V heeft gebracht om zijn bereidheid om te ontvangen aan te geven, kan de spreker ten minste 200 microseconden wachten om de luisteraar aan te geven dat de byte met EOI-status moet worden ontvangen. De luisteraar bevestigt dit aan de spreker door de DATA-lijn enige tijd naar 0V te trekken, waarna de verzending vrijwel identiek aan een normale transactie verloopt.

De SRQ-lijn is niet actief bij deze transactie.

Ter samenvatting:

1. De spreker geeft de CLK-lijn vrij naar 5V om aan te geven dat hij een byte wil verzenden.
2. Na een willekeurige tijdsduur geeft de luisteraar de DATA-lijn vrij naar 5V om zijn bereidheid om een byte te ontvangen aan te geven.
3. Normaal zou de spreker op dit punt na een korte tijd beginnen met het verzenden van de bits. Om EOI aan te geven, wacht hij evenwel stilzwijgend totdat de luisteraar doorheeft dat deze byte afwijkend is, hetgeen ertoe leidt dat de luisteraar de DATA-lijn terug naar 0V trekt.
4. De luisteraar houdt de DATA-lijn lang genoeg op 0V om er zeker van te zijn dat de spreker dit heeft opgemerkt, en geeft deze vervolgens weer vrij.
5. Zodra de spreker heeft waargenomen dat de luisteraar de DATA-lijn naar 0V heeft getrokken en weer heeft vrijgegeven, begint hij de byte na een korte vertraging normaal te verzenden.
6. Indien de spreker te lang wacht, neemt de luisteraar aan dat de spreker EOI wilde aangeven zonder daadwerkelijk een byte te verzenden.
7. De verzending van de byte wordt op dezelfde manier afgerond als bij een niet-EOI-byte.



Symbool	Min	4541	Max	Omschrijving
T_{BB}	100	100	∞	Tijd tussen bytes. Sprekers moeten lang genoeg wachten om de luisteraar de tijd te geven het einde van de verzending van de vorige byte te registreren, voordat de volgende byte kan worden verzonden.
T_H	–	N/A	∞	Luisteraar-wachttijd. *De luisteraar kan de ontvangstbevestiging willekeurig lang uitstellen of ook helemaal niet.*
T_{YE}	200	250	∞	EOI-indicatievertraging. De zender moet ten minste 200 microseconden wachten zodat de ontvanger kan herkennen dat een byte met EOI wordt verzonden. Indien de vertraging korter is, herkent de ontvanger de EOI niet.
T_{EI}	60	80	∞	EOI-bevestiging vasthoudtijd. De ontvanger moet de DATA-lijn ten minste 60 microseconden weer op 0V trekken, zodat de zender deze EOI-bevestigingspuls kan detecteren.
T_{RY}	60	80	100	De spreker dient binnen een strak tijdvenster te reageren. Indien hij te vroeg is, is het mogelijk dat de luisteraar zich na bevestiging van de EOI, nog niet heeft voorbereid op ontvangst en kan de synchronisatie verloren gaan. Indien hij te laat is, dan kan de luisteraar concluderen dat de spreker gestopt is met spreken.

Symbool	Min	4541	Max	Omschrijving
T_{ST}	20*	35	∞	Databit-insteltijd. *In de C64 Programmer's Reference Guide aangeduid als T_S.*
T_{VT}	20*	35	∞	Databit-geldigheidsvasthoudtijd. *In de C64 Programmer's Reference Guide aangeduid als T_V.*
T_F	-	1000	1000	Frame-handshake (bevestiging) time-out.
T_{FR}	0/60	0	∞	EOI-frame-bevestiging. De luisteraar trekt DATA na een korte periode naar 0V om de met EOI verzonden byte te bevestigen.

Alle tijdseenheden zijn uitgedrukt in microseconden, tenzij anders aangegeven.

* T_{ST} wordt geleverd door het randapparaat, niet door de controller. De 4541 is in staat veel sneller te reageren op seriële bus-gebeurtenissen dan een C64, en heeft derhalve slechts 4 microseconden nodig om er zeker van te zijn dat de CLK-lijn de tijd heeft gehad naar 5V te stijgen. Voor controllers die een softwarematige implementatie van het protocol gebruiken, zoals de C64, bedraagt het minimum 80 microseconden. Randapparaten dienen derhalve altijd een waarde van ten minste 80 microseconden voor T_{DA} te gebruiken.

^ De Commodore™ 64 Programmer's Reference Guide vermeldt T_{FR} met een minimumduur van 60 microseconden. Wij zien geen enkel bewijs dat er een vertraging nodig zou zijn voordat de DATA-lijn door de luisteraar laag kan worden getrokken om ontvangst van een byte te bevestigen.

Byte ontvangen

Het ontvangen van bytes op de IEC-bus verloopt identiek aan het verzenden. Het belangrijkste verschil is dat de 4541 veel ruimere marginale waarden voor tijdsparameters toelaat dan softwarematige implementaties van het bus-protocol. De 4541 heeft niet meer dan 4 microseconden nodig voor een willekeurige bus-toestand; dit is de tijd die de pull-up-weerstanden op de bus nodig hebben om de lijnen te laten terugkeren naar 5V. Intern is de 4541 in staat bit-tijden korter dan 1 microseconde te detecteren.

OPTIONELE GEÏNTEGREERDE DATALOGGER

De 4541 is verkrijgbaar in een variant die een ingebouwde seriële randapparatuurbus-datalogger bevat. Deze is ontworpen om het oplossen van protocolfouten op deze

bus te vergemakkelijken. Het vastleggen van gegevens begint zodra opdracht $D0 aan de 4541 wordt gegeven, en duurt ongeveer 4 milliseconden.

- **DATALOG0** Het lage byte van de optionele geïntegreerde seriële randapparatuurbus-datalogger. Het schrijven van $00 naar dit register reset het leespunt van de datalogger naar het begin van de opname. Het schrijven van $01 naar dit register verplaatst het leespunt van de datalogger met één tijdstap. De tijdstappen bedragen ongeveer 1 microseconde, met extra tijdstappen wanneer een van de signalen op de seriële perifere bus van toestand verandert.
- **DATALOG1** Het hoge byte van de optionele geïntegreerde seriële randapparatuurbus-datalogger. Verschillende velden kunnen op dit adres worden gelezen door de volgende waarden naar het DATALOG0-register te schrijven:
  - $02 - Lees lage byte van het statemachine-toestandsnummer van de IEC-controller.
  - $03 - Lees hoge byte van het statemachine-toestandsnummer van de IEC-controller.
  - $04 - Lees lage byte van het aantal cycli dat de vorige bustoestand is vastgehouden.
  - $05 - Lees hoge byte van het aantal cycli dat de vorige bustoestand is vastgehouden.

Elke tijdstap legt 16 bits informatie vast over de seriële perifere bus:

- **DATALOG0 Bit 0** DATA-lijn invoerwaarde
- **DATALOG0 Bit 1** CLK-lijn invoerwaarde
- **DATALOG0 Bit 2** SRQ-lijn invoerwaarde
- **DATALOG0 Bit 3** DATA-lijn uitvoertoestand
- **DATALOG0 Bit 4** CLK-lijn uitvoertoestand
- **DATALOG0 Bit 5** SRQ-lijn uitvoertoestand
- **DATALOG0 Bit 6** ATN-lijn uitvoertoestand
- **DATALOG0 Bit 7** RESET-lijn uitvoertoestand
- **DATALOG1** Toegang tot de aanvullende datalogger-velden.

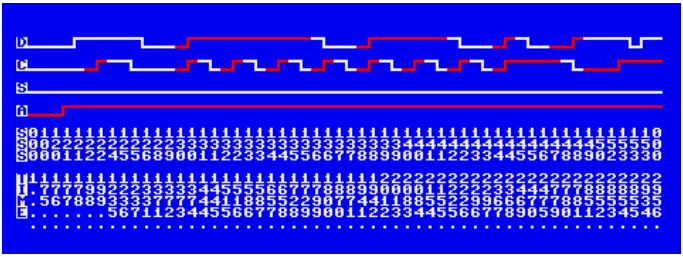
De invoerwaarden zijn de spanningen die de 4541 op de betreffende pinnen uitleest. Los daarvan legt de datalogger vast of de 4541 een van die lijnen laag trekt. Dit maakt het mogelijk te bepalen of een aangesloten randapparaat of de 4541 een bepaald signaal laag trekt. Dit biedt veel meer informatie dan het simpelweg signalen op de seriële randapparatuurbus meten, waarbij niet eenvoudig is vast te stellen wie een bepaalde lijn laag trekt.

De volgende waarheidstabel legt uit hoe deze signalen moeten worden geïnterpreteerd:

Invoerwaarde	Uitvoerwaarde	Betekenis
1	1	Het signaal zweeft op 5V. Geen enkel apparaat trekt het laag.
0	1	Het signaal staat op 0V. De 4541 trekt het niet laag, ofwel een aangesloten randapparaat, ofwel de CIA trekt het laag.
0	0	Het signaal staat op 0V. De 4541 trekt het laag. Andere apparaten kunnen het ook laag trekken, maar tussen deze twee situaties kan niet worden onderscheiden.
1	0	Het signaal zweeft op 5V, maar de 4541 trekt het naar 0V. Uw computer is waarschijnlijk defect, of iemand heeft 5V op het signaal aangesloten zonder stroombeperkende weerstand ertussen; in dat geval is uw computer op het punt defect te raken.

Gegevens uit de datalogger halen

Met het volgende programma wordt de inhoud van de datalogger opgehaald en als PETSCII-golfvorm weergegeven, zoals de hier getoonde:



```
10 WK=$50000
20 NL=PEEK($ED)
30 FETCH 3, 4096*2, 0, WK
40 PRINT"␣DUMPING..."
50 POKE$D694,0
60 FORI=0 TO 4095*2 STEP 2
70 POKEWK+I,PEEK($D694):POKE WK+1+I,PEEK($D695)
80 POKE$D694,1
90 NEXT I
100 PRINT"␣PROCESSING..."
110 PRINT"␣␣␣
```

```
400 PRINT A$;"■■";
410 GOSUB 550:GOSUB 490
420 PRINT"■■■■■■■■■■■■■■■■";
430 PS%=S%
440 PV%=V%
450 IF C%=77 THEN  0,18:C%=0
460 NEXT D
470 PRINT"■";:FORL=0TONL-3:PRINT"■";:NEXTL
480 END
490 N$=LEFT$(STR$(D/2)+".........",6)
500 PRINT"■";
510 FORI=1TO6
520 PRINTCHR$(27);"K";MID$(N$,I,1)
530 NEXT I
540 RETURN
550 S$=MID$(STR$(S%),2,LEN(STR$(S%))):N$=RIGHT$("000"+S$,3)
560 PRINT"■";
570 FORI=1TO3
580 PRINTCHR$(27);"K";MID$(N$,I,1)
590 NEXT I
600 RETURN
```

BIJLAGE X

Registers van de real-time klok (RTC)

- Algemene RTC-registers (alle revisies)
- Uitgebreide RTC-registers (moederbord R4 en nieuwer)
- Registers op de Grove-RTC-vervangingsunit

ALGEMENE RTC-REGISTERS (ALLE REVISIES)

HEX	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
FFD7110	RTCSEC							
FFD7111	RTCMIN							
FFD7112	Onge-bruikt	RTCHOUR						
FFD7113	RTCDAY							
FFD7114	RTCMONTH							
FFD7115	RTCYEAR							
FFD7116	RTCDOW							
FFD7140 - FFD717F	NVRAM							

- **NVRAM** 64 byte niet-vluchtig RAM; kan worden gebruikt om de machineconfiguratie op te slaan
- **RTCDAY** Waarde die de dag van de maand aangeeft in de realtimeklok (binair gecodeerd decimaal)
- **RTCDOW** Waarde die de dag van de week aangeeft in de realtimeklok (0-6; binair gecodeerd decimaal)
- **RTCHOUR** Waarde die het uur aangeeft in de realtimeklok (binair gecodeerd decimaal)
- **RTCHOUR24EN** Schakelt de 24-uursmodus van de realtimeklok in
- **RTCHOURPM** PM-indicator van de realtimeklok (alleen in de 12-uursmodus)
- **RTCMIN** Waarde die de minuten aangeeft in de realtimeklok (binair gecodeerd decimaal)
- **RTCMONTH** Waarde die de maand aangeeft in de realtimeklok (binair gecodeerd decimaal)
- **RTCSEC** Waarde die de seconden aangeeft in de realtimeklok (binair gecodeerd decimaal)
- **RTCYEAR** Waarde die het jaar aangeeft in de realtimeklok (binair gecodeerd decimaal)
- **Ongebruikt** Ongebruikt; altijd 1

UITGEBREIDE RTC-REGISTERS (MOEDERBORD R4 EN NIEUWER)

Deze moederborden bevatten de RV-3032-C7, welke enkele extra registers biedt. Lees de documentatie voor deze RTC-chip zorgvuldig door voordat u deze registers wijzigt!

HEX	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
FFD7116	RTCDOW							
FFD7117	100THSEC							
FFD7118	AE_M	ALMINUTES						
FFD7119	AE_H	-	ALHOURS					
FFD711A	AE_D	-	ALDAY					
FFD711B	PCTCR0							
FFD711C	-				PCTCR1			
FFD711D	THF	TLF	UF	TF	AF	EVF	PORF	VLF
FFD711E	TEMPL				EEF	EEbusy	CLKF	BSF
FFD711F	TEMPH							
FFD7120	Onge-bruikt	Onge-bruikt	GP0	USEL	TE	EERD	TD	
FFD7121	-	CLKIE	UIE	TIE	AIE	EIE	GP1	STOP
FFD7122	Ongebruikt			BSIE	THE	TLE	THIE	TLIE
FFD7123	Ongebruikt		EVR	THR	TLR	EVOW	THOW	TLOW
FFD7124	CLKD	INTDE	CEIE	CAIE	CTIE	CUIE	CTHIE	CTLIE
FFD7125	CLKDE	EHL	ET		Ongebruikt			ESYN
FFD7126	TLT							
FFD7127	THT							
FFD7128	TSTLowCount							
FFD7129	-	TSTLowSec						
FFD712A	-	TSTLowMin						
FFD712B	-	TSTLowHrs						
FFD712C	-	TSTLowDat						
FFD712D	-	TSTLowMonth						
FFD712E	TSTLowYear							
FFD712F	TSTHighCount							
FFD7130	-	TSTHighSec						
FFD7131	-	TSTHighMin						
FFD7132	-	TSTHighHrs						
FFD7133	-	TSTHighDat						
FFD7134	-	TSTHighMonth						
FFD7135	TSTHighYear							
FFD7136	TSEVICount							

vervolgd ...

...vervolgd

HEX	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
FFD7137	TSEVI100th							
FFD7138	–	TSEVISec						
FFD7139	–	TSEVIMin						
FFD713A	–		TSEVIHrs					
FFD713B	–		TSEVIDat					
FFD713C	–			TSEVIMonth				
FFD713D	TSEVIYear							

- **100THSEC** Honderdsten van seconden van de realtimeklok
- **AE_D** Inschakelbit van het alarm (samen met AE_M en AE_H)
- **AE_H** Inschakelbit van het alarm (samen met AE_M en AE_D)
- **AE_M** Inschakelbit van het minuten-alarm; schakelt het alarm in samen met AE_H en AE_D
- **AF** Statusregister: 1 geeft het optreden van een alarminterrupt aan
- **AIE** Inschakelbit van het alarminterrupt
- **ALDAY** Datumwaarde van het alarm, 01-31 (binair gecodeerd decimaal)
- **ALHOURS** Uurwaarde van het alarm, 00-23 (binair gecodeerd decimaal)
- **ALMINUTES** Minutenwaarde van het alarm, 00-59 (binair gecodeerd decimaal)
- **BSF** 1 geeft aan dat een omschakeling van de hoofdvoeding naar de back-up heeft plaatsgevonden
- **BSIE** Inschakelbit van het interrupt voor de back-upomschakeling
- **CAIE** Klokuitvoer wanneer de alarm-interrupt is ingeschakeld
- **CEIE** Klokuitvoer wanneer de EVI-interrupt is ingeschakeld
- **CLKD** Inschakelbit voor de uitschakelvertragingswaarde van CLKOUT na I2C STOP
- **CLKDE** Inschakelbit van de uitschakelvertraging van CLKOUT na I2C-STOP
- **CLKF** 1 geeft het optreden van een interruptgestuurde klokuitvoer op de CLKOUT-pin aan
- **CLKIE** Inschakelbit van de interruptgestuurde klokuitvoer
- **CTHIE** Klokuitvoer wanneer de THigh-interrupt is ingeschakeld
- **CTIE** Klokuitvoer wanneer de interrupt van de periodieke afteltimer is ingeschakeld
- **CTLIE** Klokuitvoer wanneer de TLow-interrupt is ingeschakeld

- **CUIE** Klokuitvoer wanneer de Periodic Time Update-interrupt is ingeschakeld
- **EEF** 1 geeft aan dat de EEPROM-schrijftoegang is mislukt
- **EERD** Bit voor het uitschakelen van de EEPROM-geheugenverversing
- **EEbusy** 1 geeft aan dat de EEPROM momenteel bezig is en verdere commando's negeert
- **EHL** Selectie van het hoge/lage niveau van de gebeurtenis (stijgende/dalende flank) voor detectie
- **EIE** Inschakelbit voor de interrupt van een externe gebeurtenis
- **ESYN** Synchronisatiebit voor een externe gebeurtenis (EVI)
- **ET** Instelling van de filtertijd voor gebeurtenissen
- **EVF** Statusregister: 1 geeft aan dat een externe gebeurtenis is opgetreden
- **EVOW** Overschrijfbit van de EVI-tijdstempel
- **EVR** Resetbit van de EVI-tijdstempel
- **GP0** Registerbit voor algemeen gebruik
- **GP1** Registerbit voor algemeen gebruik
- **INTDE** Inschakelbit voor interruptvertraging na het inschakelen van CLKOUT
- **PCTCR0** Besturingsregister van periodieke afteltimer 0 (lage byte)
- **PCTCR1** Besturingsregister van periodieke afteltimer 1 (hoge byte)
- **PORF** Statusregister: 1 geeft aan dat gegevens ongeldig zijn vanwege een stroomprobleem; registers moeten opnieuw worden geïnitialiseerd
- **RTCDOW** Waarde van de dag van de week, 0-6 (binair gecodeerd decimaal)
- **STOP** 1 stopt de timer en herstelt de klokvoorscalerfrequenties
- **TD** Selectie van de timerklokfrequentie
- **TE** Inschakelbit van de periodieke afteltimer
- **TEMPH** Geheel deel van de temperatuurwaarde [11.0] in tweecomplementformaat
- **TEMPL** Fractiedeel van de temperatuurwaarde [11:0] in tweecomplementformaat
- **TF** Statusregister: 1 geeft het optreden van een interrupt van de periodieke afteltimer aan
- **THE** Inschakelbit voor hoge temperatuur
- **THF** Statusregister: 1 geeft aan dat de temperatuur de hoge drempelwaarde THT heeft overschreden

- **THIE** Inschakelbit voor de hoge-temperatuurinterrupt
- **THOW** Overschrijfbit van de THigh-tijdstempel
- **THR** Resetbit van de THigh-tijdstempel
- **THT** Temperatuurdrempelregister: hoge drempelwaarde (THigh)
- **TIE** Inschakelbit voor de interrupt van de periodieke afteltimer
- **TLE** Inschakelbit voor lage temperatuur
- **TLF** Statusregister: 1 geeft aan dat de temperatuur onder de lage drempelwaarde TLT is gezakt
- **TLIE** Inschakelbit voor de lage-temperatuurinterrupt
- **TLOW** Bit voor het overschrijven van TLow in de tijdstempel.
- **TLR** Bit voor het resetten van TLow in de tijdstempel.
- **TLT** Temperatuurdrempelregister: lage drempelwaarde (TLow)
- **TSEVI100th** Tijdstempelevenementregister: TS EVI honderdsten van seconden
- **TSEVICount** Tijdstempelevenementregister: TS EVI-telling
- **TSEVIDat** Tijdstempel-THigh-register: TS TLow-datum
- **TSEVIHrs** Tijdstempel-THigh-register: TS TLow-uren
- **TSEVIMin** Tijdstempel-THigh-register: TS TLow-minuten
- **TSEVIMonth** Tijdstempel-THigh-register: TS TLow-maand
- **TSEVISec** Tijdstempel-THigh-register: TS TLow-seconden
- **TSEVIYear** Tijdstempel-THigh-register: TS TLow-jaar
- **TSTHighCount** Tijdstempel-THigh-register: TS TLow-telling
- **TSTHighDat** Tijdstempel-THigh-register: TS TLow-datum
- **TSTHighHrs** Tijdstempel-THigh-register: TS TLow-uren
- **TSTHighMin** Tijdstempel-THigh-register: TS TLow-minuten
- **TSTHighMonth** Tijdstempel-THigh-register: TS TLow-maand
- **TSTHighSec** Tijdstempel-THigh-register: TS TLow-seconden
- **TSTHighYear** Tijdstempel-THigh-register: TS TLow-jaar
- **TSTLowCount** Tijdstempel-TLow-register: TS TLow-telling
- **TSTLowDat** Tijdstempel-TLow-register: TS TLow-datum
- **TSTLowHrs** Tijdstempel-TLow-register: TS TLow-uren
- **TSTLowMin** Tijdstempel-TLow-register: TS TLow-minuten

- **TSTLowMonth** Tijdstempel-TLow-register: TS TLow-maand
- **TSTLowSec** Tijdstempel-TLow-register: TS TLow-seconden
- **TSTLowYear** Tijdstempel-TLow-register: TS TLow-jaar
- **UF** Statusregister: 1 geeft aan dat er een periodieke timer-update-interruptgebeurtenis heeft plaatsgevonden.
- **UIE** Bit voor het inschakelen van de periodieke tijdupdate-interrupt.
- **USEL** Bit voor het selecteren van de update-interrupt
- **Ongebruikt** Niet geïmplementeerd. Geeft altijd 0 terug
- **VLF** Statusregister: 1 geeft aan dat er ongeldige gegevens zijn door een voedingsprobleem. De registers moeten opnieuw worden geïnitialiseerd.

REGISTERS OP DE GROVE-RTC-VERVANGINGSUNIT

Deze registers zijn beschikbaar indien een Grove-vervangingsunit is geïnstalleerd. Dit onderdeel is alleen nodig indien uw RTC defect is. Dit heeft alleen betrekking op de eerste R3-moederborden die werden verzonden.

HEX	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
FFD7400	EXTRTCSEC							
FFD7401	EXTRTCMIN							
FFD7402	-	EXTRTC-HOUR12EN	EXTRTC-HOURPM	EXTRTCHOUR				
FFD7403	EXTRTCDOW							
FFD7404	EXTRTCDAY							
FFD7405	EXTRTC-CENTURY	-		EXTRTCMONTH				
FFD7406	EXTRTCYEAR							
FFD7407	EXTRTC-A1M1	EXTRTCA1SEC						
FFD7408	EXTRTC-A1M2	EXTRTCA1MIN						
FFD7409	EXTRTC-A1M3	EXTRTC-A1HOUR12EN	EXTRTC-A1HOURPM	EXTRTCA1HOUR				
FFD740A	EXTRTC-A1M4	EXTRTC-A1DOWSEL	EXTRTCA1DATE					
FFD740B	EXTRTC-A2M2	EXTRTCA2MIN						
FFD740C	EXTRTC-A2M3	EXTRTC-A2HOUR12EN	EXTRTC-A2HOURPM	EXTRTCA2HOUR				

vervolgd ...

...vervolgd

HEX	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
FFD740D	EXTRTC-A2M4	EXTRTC-A2DOWSEL	EXTRTCA2DATE					
FFD740E	EXTRTCCTRL							
FFD740F	EXTRTCST							
FFD7410	EXTRTCAGINGOFS							
FFD7411	EXTRTCTEMPMSB							
FFD7412	EXTRTCTEMPMSB							

- **EXTRTCA1DATE** Datumwaarde van alarm 1 van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCA1DOW** Waarde die de dag van de week aangeeft voor alarm 1 van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCA1DOWSEL** Selecteert de overeenkomst van de dag van de week voor alarm 1 van de externe realtimeklok
- **EXTRTCA1HOUR** Uurwaarde van alarm 1 van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCA1HOUR12EN** Schakelt de 12-uursmodus van alarm 1 van de externe realtimeklok in
- **EXTRTCA1HOURPM** PM-indicator van het uur van alarm 1 van de externe realtimeklok
- **EXTRTCA1M1** Maskerbit 1 van alarm 1 van de externe realtimeklok
- **EXTRTCA1M2** Maskerbit 2 van alarm 1 van de externe realtimeklok
- **EXTRTCA1M3** Maskerbit 3 van alarm 1 van de externe realtimeklok
- **EXTRTCA1M4** Maskerbit 4 van alarm 1 van de externe realtimeklok
- **EXTRTCA1MIN** Minuutwaarde van alarm 1 van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCA1SEC** Secondenwaarde van alarm 1 van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCA2DATE** Datumwaarde van alarm 2 van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCA2DOW** Waarde die de dag van de week aangeeft voor alarm 2 van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCA2DOWSEL** Selecteert de overeenkomst van de dag van de week voor alarm 2 van de externe realtimeklok
- **EXTRTCA2HOUR** Uurwaarde van alarm 2 van de externe realtimeklok (binair gecodeerd decimaal)

- **EXTRTCA2HOUR12EN** Schakelt de 12-uursmodus van alarm 2 van de externe realtimeklok in
- **EXTRTCA2HOURPM** PM-indicator van het uur van alarm 2 van de externe realtimeklok
- **EXTRTCA2M2** Maskerbit 2 van alarm 2 van de externe realtimeklok
- **EXTRTCA2M3** Maskerbit 3 van alarm 2 van de externe realtimeklok
- **EXTRTCA2M4** Maskerbit 4 van alarm 2 van de externe realtimeklok
- **EXTRTCA2MIN** Minuutwaarde van alarm 2 van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCAGINGOFS** Ouderdomscompensatie van de externe realtimeklok (niet wijzigen!)
- **EXTRTCCENTURY** Jaaroverdracht van de externe realtimeklok
- **EXTRTCCTRL** Besturing van de externe realtimeklok
- **EXTRTCDAY** Waarde die de dag van de maand aangeeft in de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCDOW** Waarde die de dag van de week aangeeft in de externe realtimeklok (1-7; binair gecodeerd decimaal)
- **EXTRTCHOUR** Uurwaarde van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCHOUR12EN** Schakelt de 12-uursmodus van de externe realtimeklok in
- **EXTRTCHOURPM** PM-indicator van de externe realtimeklok (alleen 12-uursmodus)
- **EXTRTCMIN** Minuutwaarde van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCMONTH** Maandwaarde van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCSEC** Secondenwaarde van de externe realtimeklok (binair gecodeerd decimaal)
- **EXTRTCST** Besturings- en statusregister van de externe realtimeklok
- **EXTRTCTEMPMSB** Temperatuur van de externe realtimeklok (MSB)
- **EXTRTCYEAR** Jaarwaarde van de externe realtimeklok (binair gecodeerd decimaal)

BIJLAGE Y

Speciale registers

- UUID64

UUID64

HEX	Signaal	Omschrijving
FFD7100 - FFD7107	UUID64	64-bits UUID. Kan worden gebruikt om het Ethernet-MAC-adres in te zaaien

BIJLAGE Z

5417E Spreadsheet Hardware Integrated Technology Engine

- Overzicht
- Architectuur
- Niet-geïmplementeerde functionaliteit

OVERZICHT

Hierbij presenteren wij met trots de MEGA SheetStorm 65 Spreadsheet Hardware Integrated Technology Engine (ook wel aangeduid als SHITE), de baanbrekende chip die voor het MEGA65-systeem is ontworpen, welke het verwerken van spreadsheets gaat revolutioneren op manieren die de bedrijfstak nog nooit heeft gezien. Met de innovatieve "Data Fan"-technologie is deze chip niet louter een verbetering ten opzichte van bestaande oplossingen; zij herdefinieert snelheid en efficiëntie in gegevensbeheer volledig.

Wanneer uw spreadsheetgegevens de Data Fan van de MEGA SheetStorm 65 bereiken, is dat een ervaring zonder weerga. De Data Fan, een unieke innovatie op het gebied van gegevensverdeling, zorgt ervoor dat uw cellen sneller en grondiger worden bedekt met bijgewerkte gegevens dan u ooit tevoren hebt aanschouwd. Het is als een frisse windvlaag die door uw spreadsheets trekt en elke cel verkwikt met nieuwverworven snelheid en nauwkeurigheid. In de wereld van de gegevensverwerking geldt: indien snelheid het spel is, dan is de MEGA SheetStorm 65 de onbetwiste kampioen.

De kracht van de MEGA SheetStorm 65 is zo formidabel dat deze chip de concurrentie doeltreffend wegspoelt. Waar anderen struikelen en verstopt raken onder de druk van reusachtige gegevensverzamelingen, handhaaft de SheetStorm 65 een informatiestroom die zo soepel verloopt dat het is alsof u toekijkt hoe een meesterlijk ontworpen systeem hanteert wat anderen als een stortvloed van gegevens zouden beschouwen. Dit is niet louter verwerking; dit is gegevensbeheer als kunstvorm.

In de zakenwereld van vandaag, waarin alles in hoog tempo verloopt, komt het vastzitten in trage gegevensverwerking neer op geld door het putje spoelen. De MEGA SheetStorm 65 pakt dit probleem frontaal aan en biedt een mate van efficiëntie die niet alleen de stroom van uw gegevens op gang houdt, maar ook de algehele gezondheid van uw informatiesystemen versterkt. Dit is een chip die niet alleen hard werkt; zij werkt bovenal slim en zorgt ervoor dat elk gegeven zich daar bevindt waar het behoort te zijn, precies op het moment dat het daar behoort te zijn.

Kortom: de MEGA SheetStorm 65 is meer dan slechts een technologische vooruitgang; zij vormt een revolutie in de spreadsheetverwerking. Het is een bevestiging dat in het rijk der gegevens het juiste gereedschap niet slechts een verschil maakt; het maakt hét verschil. Met de SheetStorm 65 bereidt u zich voor om uw gegevens en uw onderneming in een geheel nieuw licht te zien. Welkom in de toekomst van gegevensverwerking. Welkom bij de MEGA SheetStorm 65.

ARCHITECTUUR

De SHITE-processor bestaat uit een gedeeld geheugengebied dat in de adresruimte is ingemapt dat zowel de hoofdprocessor als de SHITE-processor gelijktijdige toegang tot de spreadsheetgegevens verleent. Correct gedefinieerde spreadsheetcellen worden door de SHITE continu in hardware met hoge snelheid bijgewerkt, hetgeen dra-

matisch snellere prestaties oplevert dan met een conventionele processor kan worden verkregen.

Voor snelle toegang door de processor kan bovendien een deelverzameling van deze cellen in de 16-bits adresruimte worden aangesloten met behulp van de Streamlined Electronic Workflow for Easy Retrieval (SEWER)-verbinding.

NIET-GEÏMPLEMENTEERDE FUNCTIONALITEIT

De SHITE-processor wordt op dit moment geïmplementeerd en is derhalve nog niet beschikbaar. Verwacht wordt dat deze beschikbaar zal komen in een toekomstige core-update van de MEGA65.

BIJLAGE AA

Naslagtabellen

- Opslageenheden
- Getalstelselomrekening

OPSLAGEENHEDEN

Eenheid	Is gelijk aan	Afkorting
1 Bit		
1 Nibble	4 Bits	
1 Byte	8 bits	B
1 Kilobyte	1024 B	KB
1 Megabyte	1024KB of 1048576 B	MB

GETALSTELSELOMREKENING

Decimaal	Binair	Hexadecimaal
0	%0	$0
1	%1	$1
2	%10	$2
3	%11	$3
4	%100	$4
5	%101	$5
6	%110	$6
7	%111	$7
8	%1000	$8
9	%1001	$9
10	%1010	$A
11	%1011	$B
12	%1100	$C
13	%1101	$D
14	%1110	$E
15	%1111	$F
16	%10000	$10
17	%10001	$11
18	%10010	$12
19	%10011	$13
20	%10100	$14
21	%10101	$15
22	%10110	$16
23	%10111	$17
24	%11000	$18
25	%11001	$19
26	%11010	$1A
27	%11011	$1B
28	%11100	$1C
29	%11101	$1D
30	%11110	$1E
31	%11111	$1F

Decimaal	Binair	Hexadecimaal
32	%100000	$20
33	%100001	$21
34	%100010	$22
35	%100011	$23
36	%100100	$24
37	%100101	$25
38	%100110	$26
39	%100111	$27
40	%101000	$28
41	%101001	$29
42	%101010	$2A
43	%101011	$2B
44	%101100	$2C
45	%101101	$2D
46	%101110	$2E
47	%101111	$2F
48	%110000	$30
49	%110001	$31
50	%110010	$32
51	%110011	$33
52	%110100	$34
53	%110101	$35
54	%110110	$36
55	%110111	$37
56	%111000	$38
57	%111001	$39
58	%111010	$3A
59	%111011	$3B
60	%111100	$3C
61	%111101	$3D
62	%111110	$3E
63	%111111	$3F

Decimaal	Binair	Hexadecimaal
64	%1000000	$40
65	%1000001	$41
66	%1000010	$42
67	%1000011	$43
68	%1000100	$44
69	%1000101	$45
70	%1000110	$46
71	%1000111	$47
72	%1001000	$48
73	%1001001	$49
74	%1001010	$4A
75	%1001011	$4B
76	%1001100	$4C
77	%1001101	$4D
78	%1001110	$4E
79	%1001111	$4F
80	%1010000	$50
81	%1010001	$51
82	%1010010	$52
83	%1010011	$53
84	%1010100	$54
85	%1010101	$55
86	%1010110	$56
87	%1010111	$57
88	%1011000	$58
89	%1011001	$59
90	%1011010	$5A
91	%1011011	$5B
92	%1011100	$5C
93	%1011101	$5D
94	%1011110	$5E
95	%1011111	$5F

Decimaal	Binair	Hexadecimaal
96	%1100000	$60
97	%1100001	$61
98	%1100010	$62
99	%1100011	$63
100	%1100100	$64
101	%1100101	$65
102	%1100110	$66
103	%1100111	$67
104	%1101000	$68
105	%1101001	$69
106	%1101010	$6A
107	%1101011	$6B
108	%1101100	$6C
109	%1101101	$6D
110	%1101110	$6E
111	%1101111	$6F
112	%1110000	$70
113	%1110001	$71
114	%1110010	$72
115	%1110011	$73
116	%1110100	$74
117	%1110101	$75
118	%1110110	$76
119	%1110111	$77
120	%1111000	$78
121	%1111001	$79
122	%1111010	$7A
123	%1111011	$7B
124	%1111100	$7C
125	%1111101	$7D
126	%1111110	$7E
127	%1111111	$7F

Decimaal	Binair	Hexadecimaal
128	%10000000	$80
129	%10000001	$81
130	%10000010	$82
131	%10000011	$83
132	%10000100	$84
133	%10000101	$85
134	%10000110	$86
135	%10000111	$87
136	%10001000	$88
137	%10001001	$89
138	%10001010	$8A
139	%10001011	$8B
140	%10001100	$8C
141	%10001101	$8D
142	%10001110	$8E
143	%10001111	$8F
144	%10010000	$90
145	%10010001	$91
146	%10010010	$92
147	%10010011	$93
148	%10010100	$94
149	%10010101	$95
150	%10010110	$96
151	%10010111	$97
152	%10011000	$98
153	%10011001	$99
154	%10011010	$9A
155	%10011011	$9B
156	%10011100	$9C
157	%10011101	$9D
158	%10011110	$9E
159	%10011111	$9F

Decimaal	Binair	Hexadecimaal
160	%10100000	$A0
161	%10100001	$A1
162	%10100010	$A2
163	%10100011	$A3
164	%10100100	$A4
165	%10100101	$A5
166	%10100110	$A6
167	%10100111	$A7
168	%10101000	$A8
169	%10101001	$A9
170	%10101010	$AA
171	%10101011	$AB
172	%10101100	$AC
173	%10101101	$AD
174	%10101110	$AE
175	%10101111	$AF
176	%10110000	$B0
177	%10110001	$B1
178	%10110010	$B2
179	%10110011	$B3
180	%10110100	$B4
181	%10110101	$B5
182	%10110110	$B6
183	%10110111	$B7
184	%10111000	$B8
185	%10111001	$B9
186	%10111010	$BA
187	%10111011	$BB
188	%10111100	$BC
189	%10111101	$BD
190	%10111110	$BE
191	%10111111	$BF

Decimaal	Binair	Hexadecimaal
192	%11000000	$C0
193	%11000001	$C1
194	%11000010	$C2
195	%11000011	$C3
196	%11000100	$C4
197	%11000101	$C5
198	%11000110	$C6
199	%11000111	$C7
200	%11001000	$C8
201	%11001001	$C9
202	%11001010	$CA
203	%11001011	$CB
204	%11001100	$CC
205	%11001101	$CD
206	%11001110	$CE
207	%11001111	$CF
208	%11010000	$D0
209	%11010001	$D1
210	%11010010	$D2
211	%11010011	$D3
212	%11010100	$D4
213	%11010101	$D5
214	%11010110	$D6
215	%11010111	$D7
216	%11011000	$D8
217	%11011001	$D9
218	%11011010	$DA
219	%11011011	$DB
220	%11011100	$DC
221	%11011101	$DD
222	%11011110	$DE
223	%11011111	$DF

Decimaal	Binair	Hexadecimaal
224	%11100000	$E0
225	%11100001	$E1
226	%11100010	$E2
227	%11100011	$E3
228	%11100100	$E4
229	%11100101	$E5
230	%11100110	$E6
231	%11100111	$E7
232	%11101000	$E8
233	%11101001	$E9
234	%11101010	$EA
235	%11101011	$EB
236	%11101100	$EC
237	%11101101	$ED
238	%11101110	$EE
239	%11101111	$EF
240	%11110000	$F0
241	%11110001	$F1
242	%11110010	$F2
243	%11110011	$F3
244	%11110100	$F4
245	%11110101	$F5
246	%11110110	$F6
247	%11110111	$F7
248	%11111000	$F8
249	%11111001	$F9
250	%11111010	$FA
251	%11111011	$FB
252	%11111100	$FC
253	%11111101	$FD
254	%11111110	$FE
255	%11111111	$FF

BIJLAGE **AB**

Het flashen van de FPGA's en CPLD's in de MEGA65

- Aanbevolen specificaties voor de pc
- Waarschuwing
- Vivado installeren
- De FTDI-stuurprogramma's installeren
- De hoofd-FPGA flashen met Vivado
- De CPLD in het toetsenbord van de MEGA65 flashen met Lattice Diamond
- Het flashen van de MAX10-FPGA op het MEGA65-moederbord met INTEL QUARTUS

De MEGA65 is een computer met open-source hardware en open-source ontwerp. Dit betekent dat u niet alleen programma's kunt schrijven die op de MEGA65 als een voltooide computer draaien, maar ook de programmeerbare chips in de MEGA65 kunt gebruiken om er allerlei andere dingen van te maken.

Als u alleen een upgradecore voor de MEGA65 wilt installeren, of een core waarmee u uw MEGA65 als een ander type computer kunt gebruiken, kijkt u beter in Hoofdstuk/Bijlage 6 op bladzijde 6-5.

Dit hoofdstuk is vooral bedoeld voor mensen die willen helpen bij het ontwikkelen van cores voor de MEGA65. Het kan ook interessant zijn voor eigenaren van Nexys4-borden die hun apparaten vanaf het QSPI-flashgeheugen op het bord willen opstarten (in plaats van een bitstreambestand vanaf de SD-kaart). Hiervoor moet een .mcs-bestand op de QSPI-chip van het bord geflashed worden om een initiële bitstream in de positie "Slot 0" te plaatsen.

Deze programmeerbare chips heten Field Programmable Gate Arrays (FPGA's) of Complex Programmable Logic Devices (CPLD's) en kunnen een breed scala aan circuits implementeren. Ze worden doorgaans geprogrammeerd met een taal zoals VHDL of Verilog. Dit zijn talen die de meeste mensen niet vaak tegenkomen. Ze zijn ook op een aantal punten heel anders dan "normale" programmeertalen en het kan even duren voordat u begrijpt hoe ze werken. Met enige inzet en volharding kunt u echter spannende dingen ermee creëren.

AANBEVOLEN SPECIFICATIES VOOR DE PC

Reken erop dat u vele gigabytes aan software moet installeren op een Linux- of Windows-pc voordat u programma's kunt schrijven voor de FPGA's en CPLD's in de MEGA65. Het "compileren" van complexe ontwerpen kan ook enkele uren duren, afhankelijk van de snelheid en het geheugen van uw computer. Wij raden een computer met ten minste 12 GB RAM (bij voorkeur 16 GB) aan als u programma's voor FPGA's en CPLD's wilt schrijven. Als u daarentegen alleen programma's op de FPGA's en CPLD's van uw MEGA65 wilt laden die door anderen zijn geschreven, volstaan de meeste computers met een recente versie van Windows of Linux.

- Besturingssysteem: Linux of Windows
- Snelheid van de processor: zo snel als u kunt vinden!
- Aantal cores: bij voorkeur 8 of meer, omdat de gratis licentie van Vivado maximaal 8 cores kan gebruiken.
- Harde schijf: ongeveer 70 GB of meer. De exacte hoeveelheid hangt af van hoeveel onderdelen van Vivado u installeert (hou er rekening mee dat het volledige installatiebestand zelf ongeveer 50 GB is)

- Geheugen: minimaal 12 GB (bij voorkeur meer, voor de zekerheid)

WAARSCHUWING

Voordat we verdergaan, moeten we u waarschuwen voor het opnieuw programmeren van de FPGA's en CPLD's in de MEGA65. Het opnieuw programmeren van de FPGA van de MEGA65 kan schade veroorzaken of uw MEGA65 in een onbereikbare staat achterlaten waarvan het moeilijk is te herstellen, oftewel "gebrickt". Als u ervoor kiest om uw MEGA65 te openen en een van de FPGA's erin opnieuw te programmeren, kunnen wij de correcte werking van het apparaat niet langer garanderen. Wij kunnen de fabrieksgarantie van het apparaat als computer derhalve niet meer redelijkerwijs honoreren. U bent gewaarschuwd!

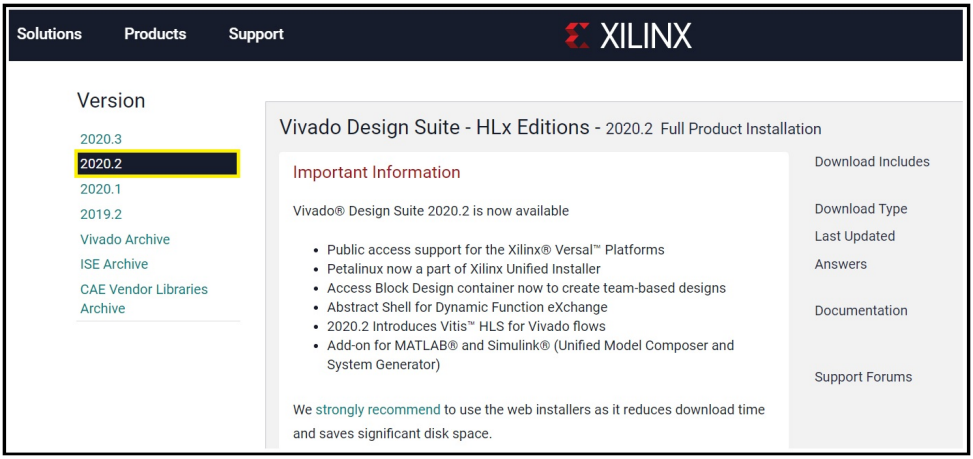
VIVADO INSTALLEREN

U moet Vivado installeren om het QSPI-flashgeheugen in uw doelapparaat te flashen, of het nu een MEGA65 R2/R3/R3A/R4, Nexys4/Nexys4DDR/NexysA7, MEGAphone of ander apparaat is.

Vivado is ook het hulpprogramma dat wordt gebruikt voor het compileren (beter bekend als synthese) van FPGA-bitstreams.

Ga naar https://www.xilinx.com/support/download.html

Selecteer versie 2020.2



Opmerking: sommige gebruikers hebben nog succes met oudere versies, omdat het hoofddoel hier is om een versie te installeren die de FPGA van uw doelhardware ondersteunt.

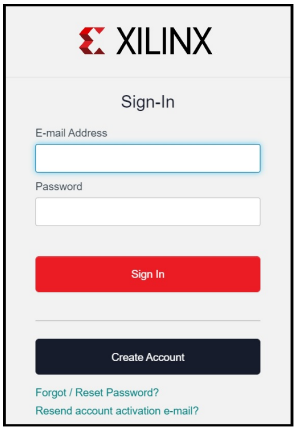
Dat wil zeggen de Artix7 100T (voor Nexys en R2) of 200T (R3/R3A/R4).

Klik op Xilinx Unified Installer 2020.2: Windows Self Extracting Web Installer EXE - 248.44MB

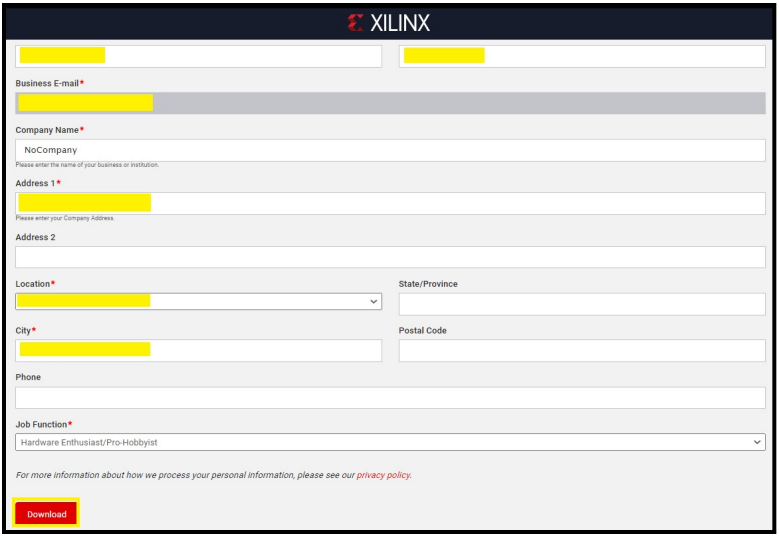


U wordt gevraagd een account aan te maken om in te kunnen loggen en het installatieprogramma te kunnen downloaden.

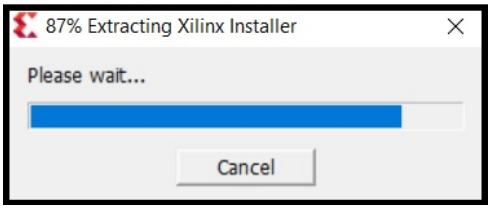
Uw inloggegevens worden ook gevraagd tijdens de installatie.



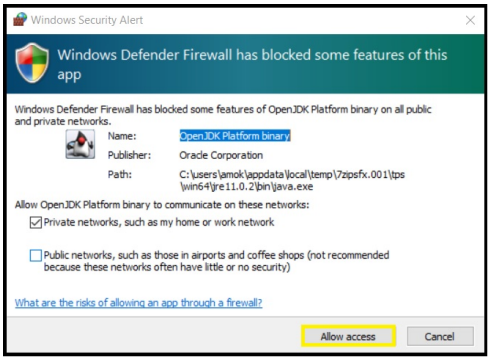
Nadat u ingelogd bent, moet u enkele persoonlijke gegevens invullen en dan op Download klikken.



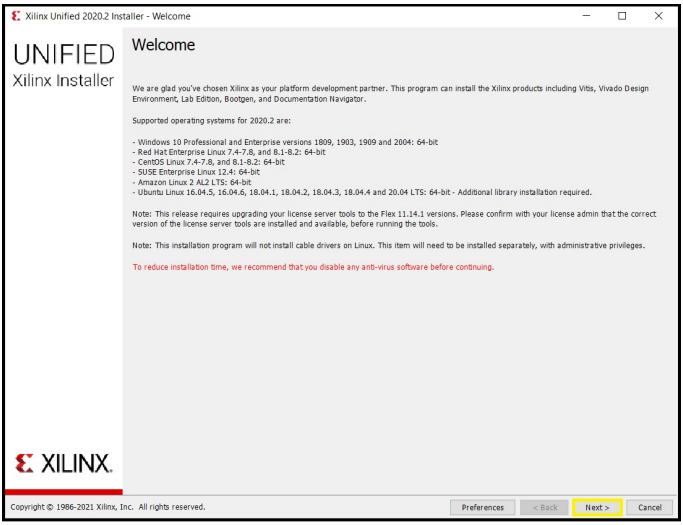
Voer het installatieprogramma uit als beheerder (Xilinx_Unified_2020.2_1118_1232_Win64.exe).



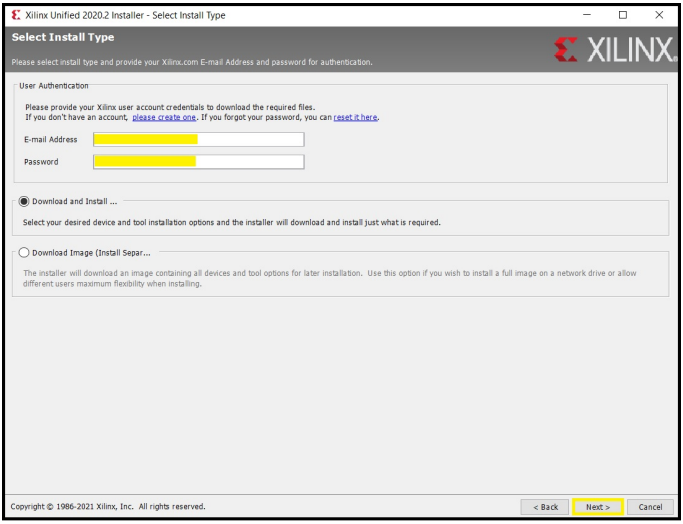
Klik op Allow Access.



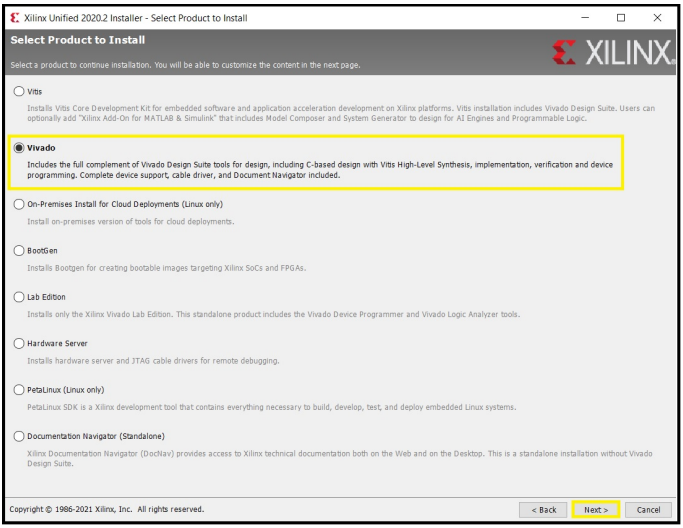
Klik op Next.



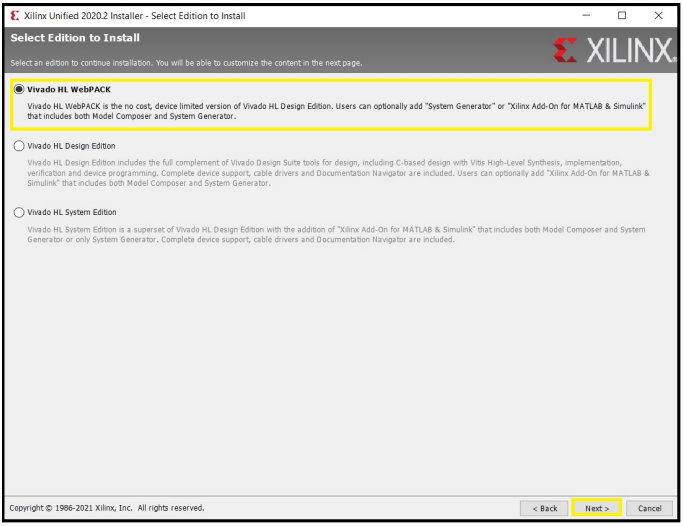
Voer uw inloggegevens in en klik op Next.



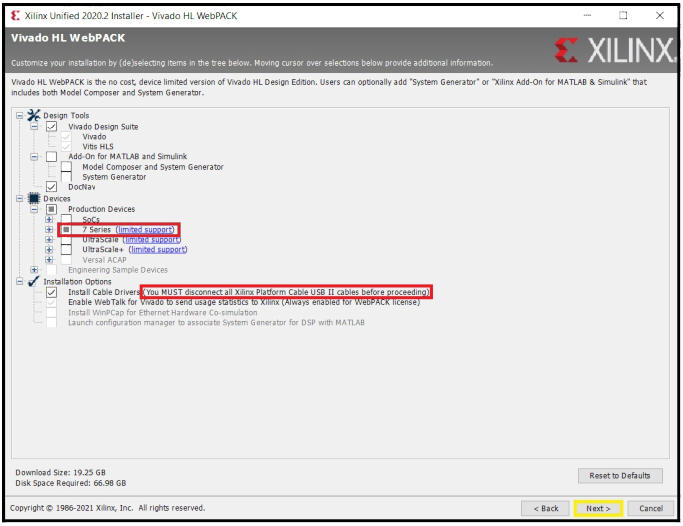
Selecteer Vivado en klik op Next.



Selecteer “Vivado HL WebPACK” en klik op “Next”.

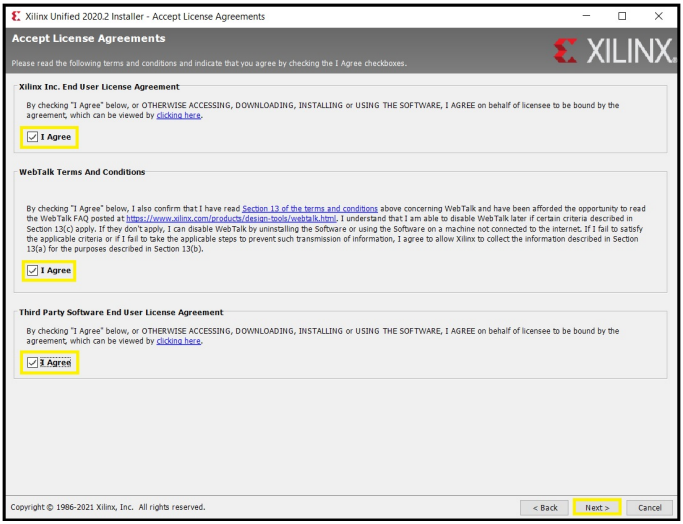


Wij raden aan om alleen de apparaten uit de “**7 Series**” te selecteren, omdat onze gekozen FPGA in deze serie valt, en het deselecteren van de andere series bespaart u ongeveer 6 GB aan downloadgrootte. Klik dan op “Next”.

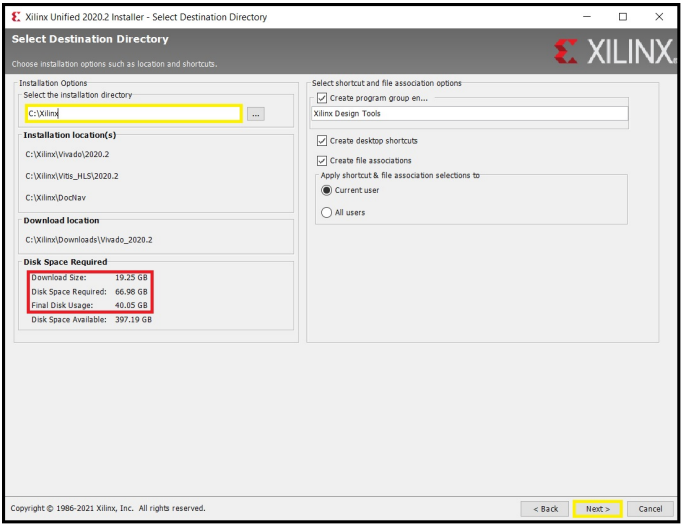


Waarschuwing: zoals gezegd, ontkoppel elke USB-kabel die op uw pc is aangesloten van het Nexys-bord.

Ga akkoord met alle bepalingen van de eindgebruikerslicentieovereenkomst en de voorwaarden en klik op “Next”.

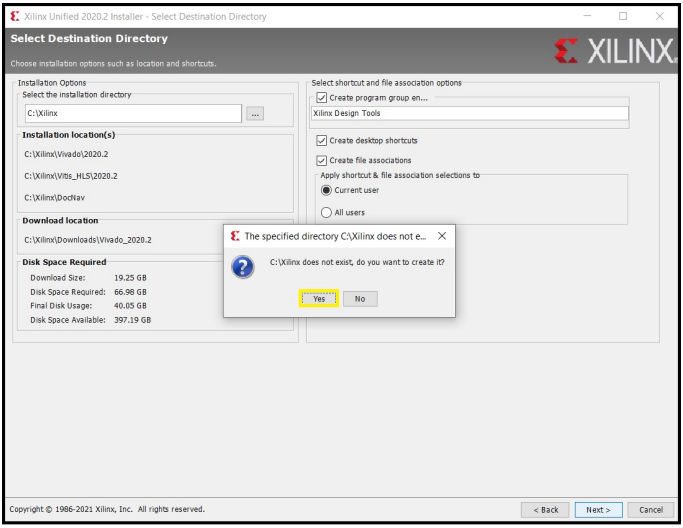


Kies de locatie waar u de software wilt installeren en klik op “Next”.

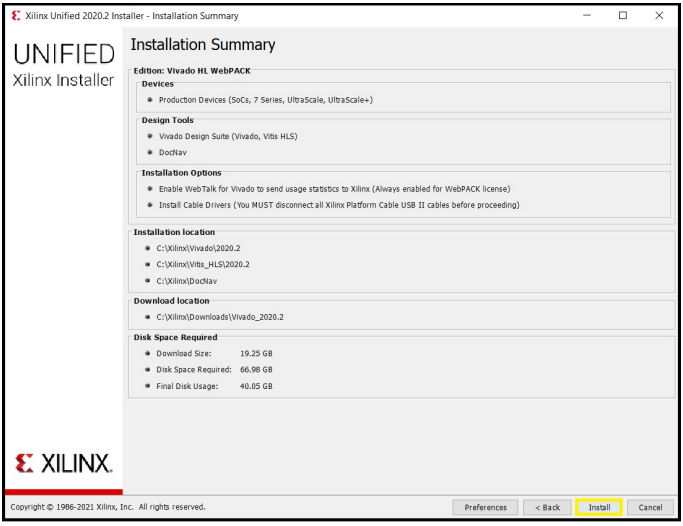


Waarschuwing: u staat op het punt om 20 GB aan software te downloaden en u hebt 70 GB nodig om de installatie uit te voeren.

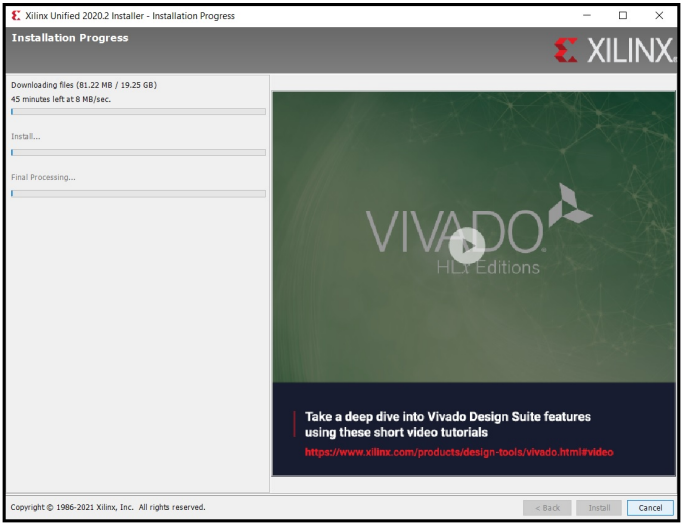
Klik op “Yes”.



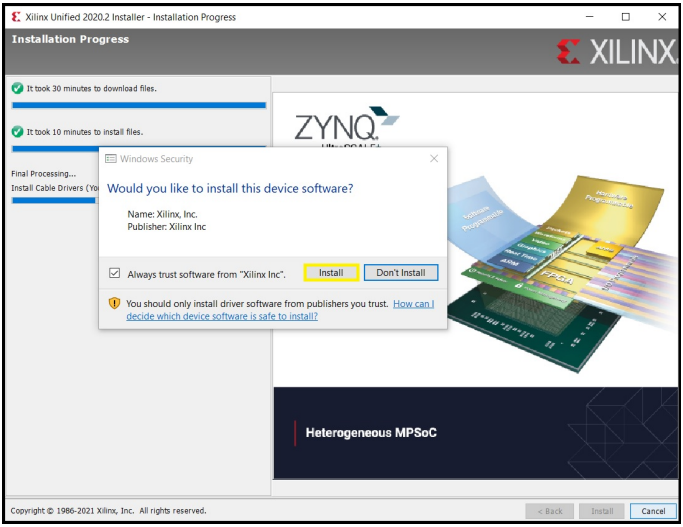
Klik op “Install”.



Wacht tot de installatie is voltooid. Aan het einde van de installatie wordt u gevraagd of u de Xilinx-apparatensoftware wilt installeren.

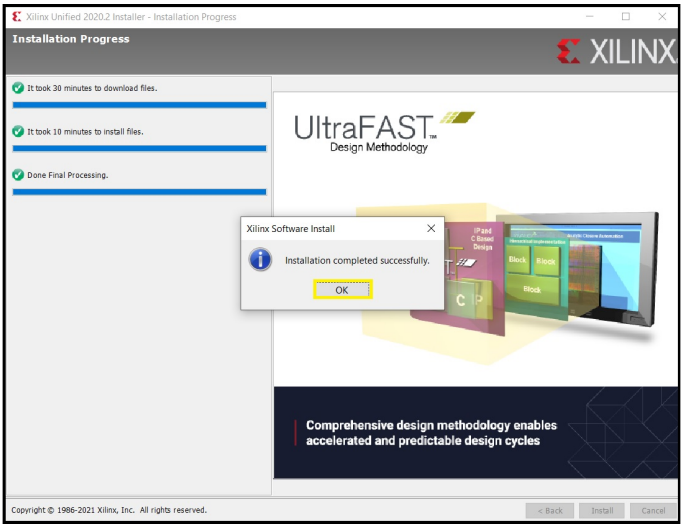


Klik op “Install”.



Laat de installatie voltooien.

De installatie is voltooid. Klik op “OK”.

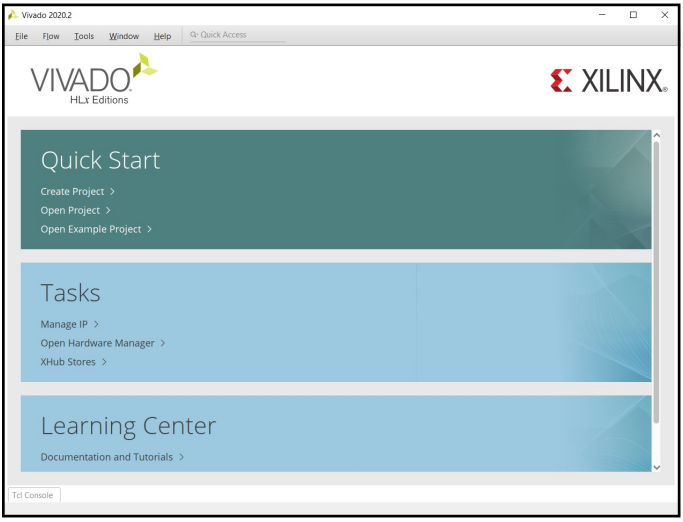


Op uw bureaublad verschijnen nu de volgende pictogrammen:

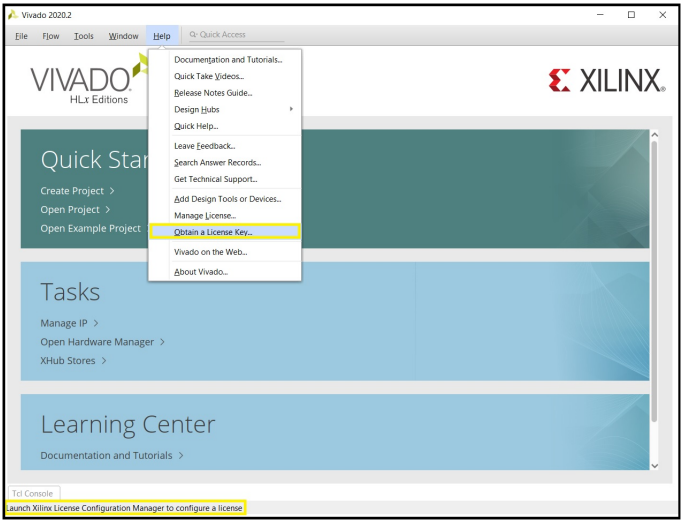


Let op: Ubuntu-gebruikers moeten mogelijk een ontbrekende afhankelijkheid installeren met: `sudo apt install libtinfo5`

Start Vivado 2020.2



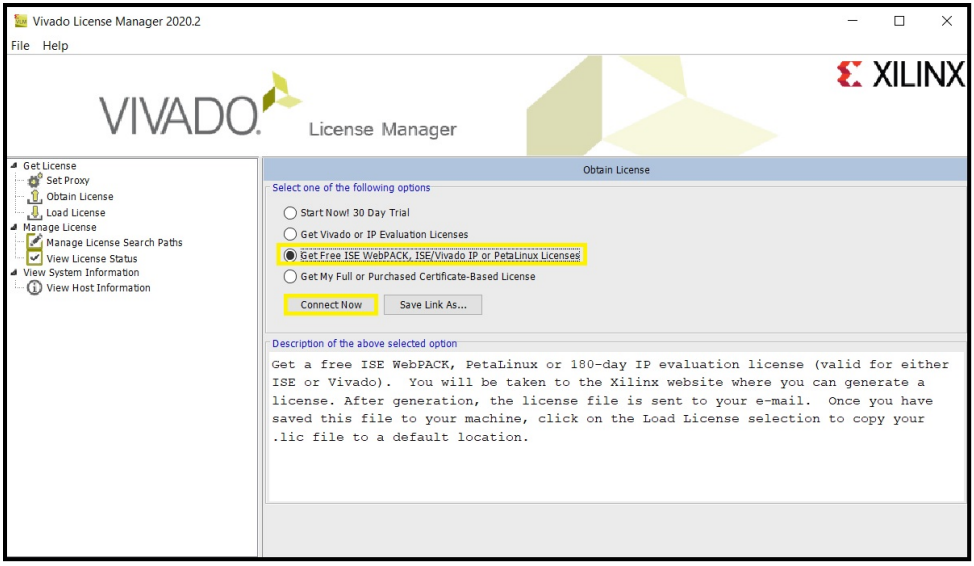
Klik op “Help”->“Obtain a licence Key”.



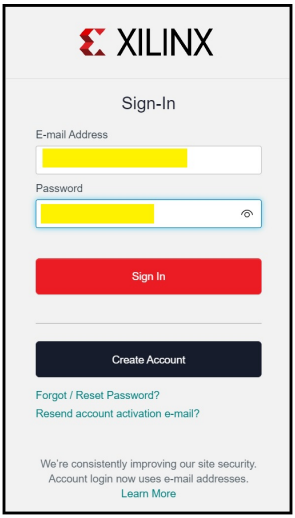
Hiermee opent u de Vivado-licentiemanager.

Selecteer “Get Free ISE WebPACK, ISE/Vivado IP or PetaLinux Licenses”

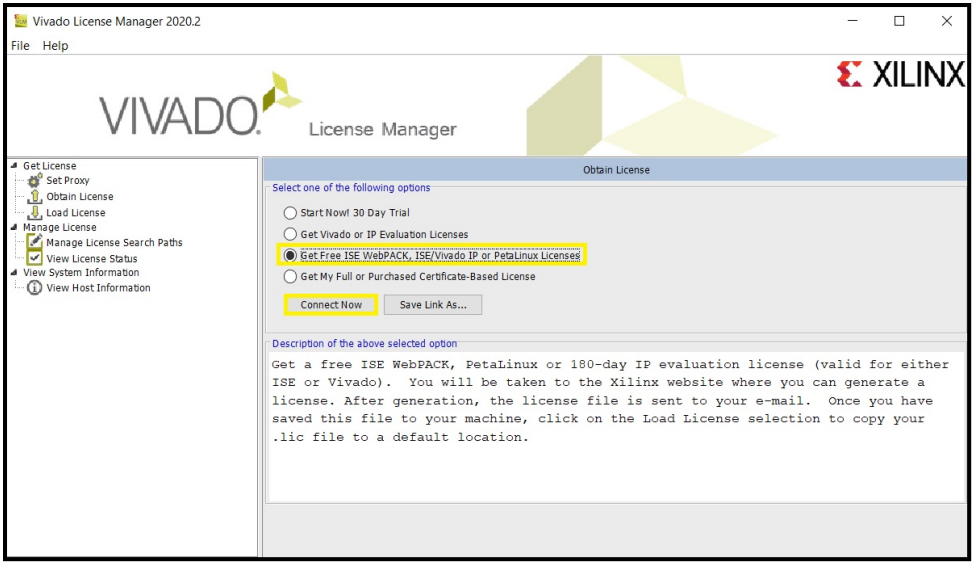
Klik op “Connect Now”.



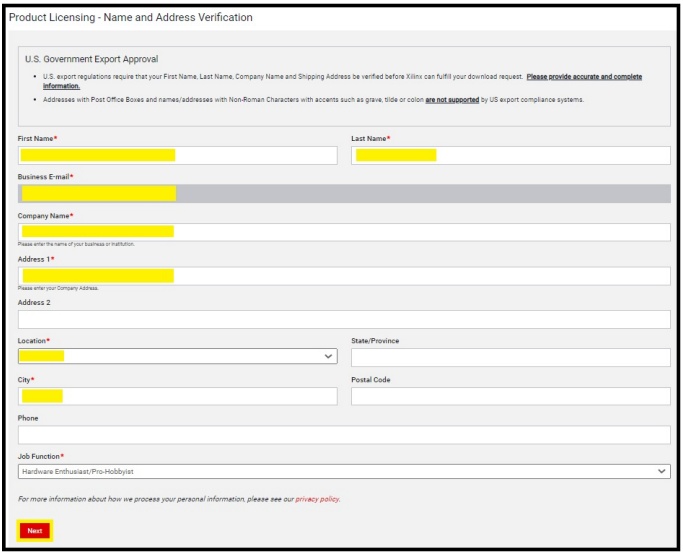
Verbind met het gebruikersaccount dat u hebt aangemaakt om de Vivado-software te kunnen downloaden. Als u nog niet op de website van Xilinx was ingelogd, wordt u naar de hoofdpagina geleid. Keer terug naar de licentiemanager (die niet is gesloten).



Klik opnieuw op “Connect Now” (controleer of “Get Free ISE WebPACK, ISE/Vivado IP or PetaLinux Licenses” nog steeds is geselecteerd).

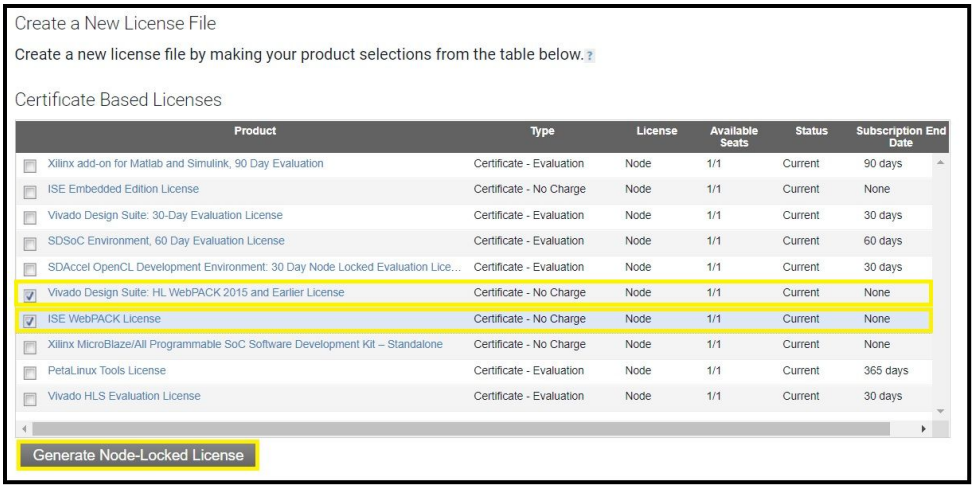


Vervolgens registreert u uw persoonlijke gegevens op de Vivado-website en klikt u op “Next”.

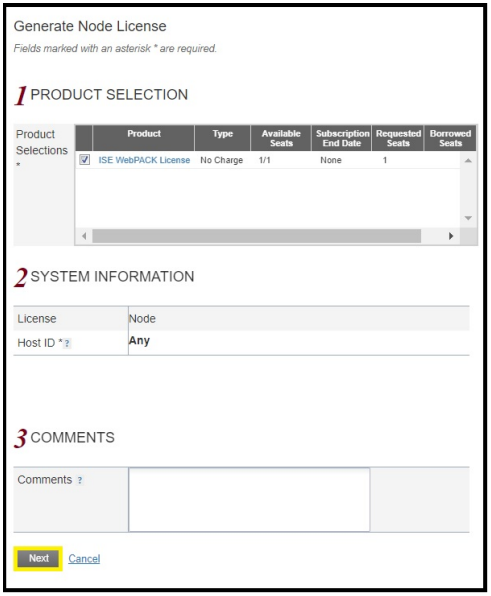


Selecteer “ISE WebPACK Licence” en “Vivado Design Suite: HL WebPACK 2015 and Earlier License”

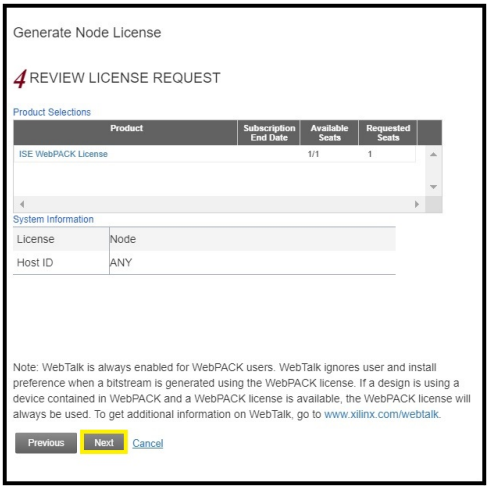
Klik vervolgens op “Generate Node-Locked Licence”.



Klik op “Next”.



Klik op “Next”.



Controleer uw e-mail: u hebt een e-mail ontvangen van Xilinx, Inc. met een licentie-bestand met de naam “Xilinc.lic” als bijlage.

Sla dit bestand op uw computer op en bewaar het op een veilige plaats.

Congratulations

Your new license file has been successfully generated and e-mailed to [redacted]. You can also view the license file under the Manage Licenses tab.

Please add this sender (xilinx.notification@entitlenow.com) to your address book.

License File Details
Node License
Host ID: ANY

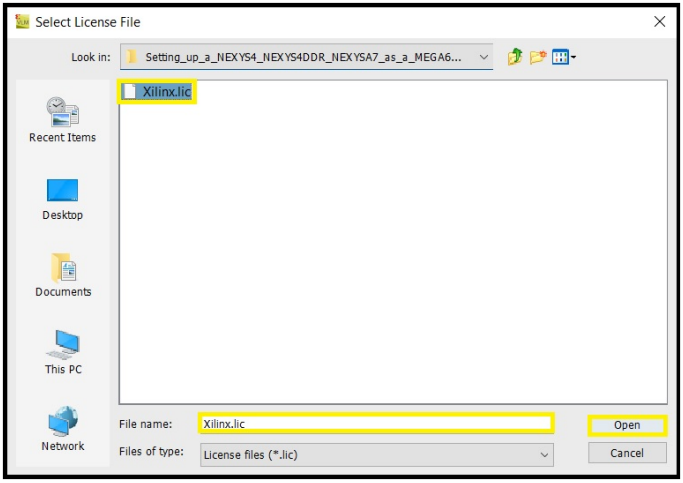
Products
ISE WebPACK License (No Charge): 1 seats

Ga terug naar de licentiemanager (die nog steeds actief is).

Selecteer “Load License” en klik op “Copy License”.



Blader naar de locatie waar u het bestand “Xilinc.lic” heeft opgeslagen, selecteer het en klik op “Open”.



Klik op “OK” en sluit de Vivado-licentiemanager.



Uw Vivado-software is geregistreerd en u kunt deze nu gebruiken.

DE FTDI-STUURPROGRAMMA’S INSTALLEREN

De FTDI-drivers zijn nodig om uw PC te laten communiceren met de JTAG-poort en de seriële communicatiepoort van de hardware (merk op dat de enkele fysieke USB-verbinding met uw PC feitelijk deze twee poorten aanbiedt).

Linux-drivers

Gebruikers van Linux melden dat zij de FTDI-drivers standaard in hun Linux-distributie aangetroffen hebben, terwijl anderen aangeven dat het volgende extra commando na de installatie van Vivado uitgevoerd moet worden:

```
cd /opt/Xilinx/Vivado/2018.3/data/xicom/cable_drivers/lin64/install_script/install_driv
sudo ./install_drivers
```

Windows-drivers

Download het volgende archief om de drivers te installeren:

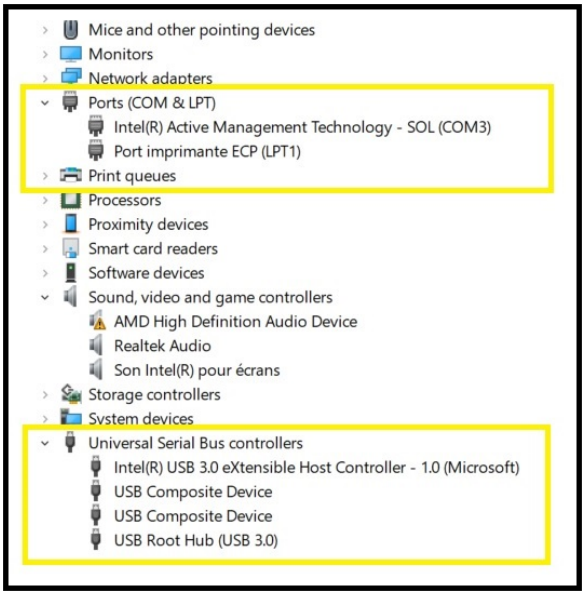
- https://ftdichip.com/wp-content/uploads/2021/08/CDM212364_Setup.zip

Pak het bestand **CDM212364_Setup.zip** uit; u verkrijgt het bestand **CDM212364_Setup.exe**.

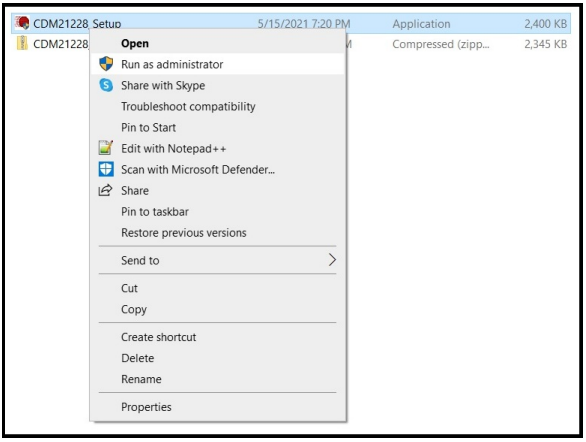
Waarschuwing:

Voordat u de drivers installeert, moet u de Nexys4-plaat uitschakelen en de USB-kabel loskoppelen van de PC.

Controleer welke apparaten er voor de installatie al aanwezig zijn:

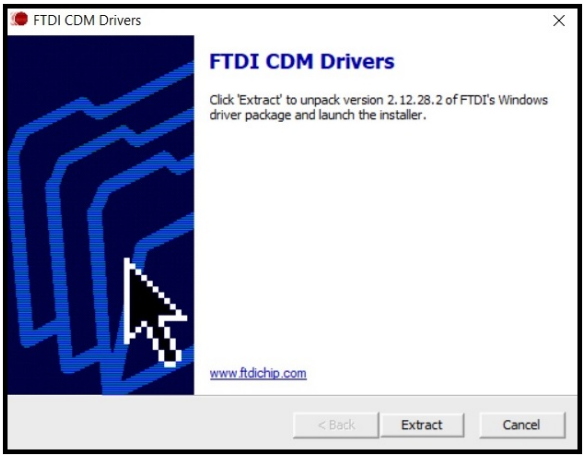


Voer het bestand CDM212364_Setup.exe uit als administrator:

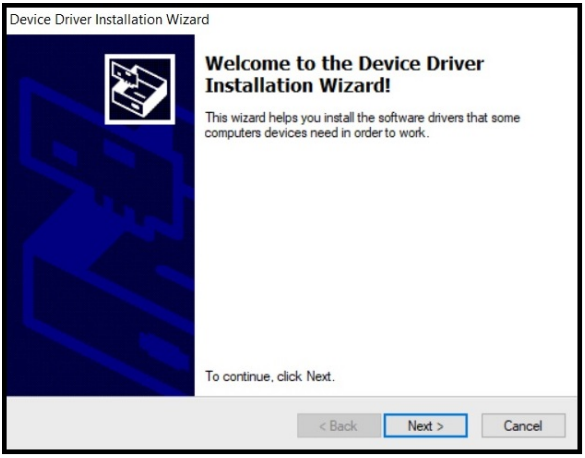


Bevestig dat u het programma wilt uitvoeren.

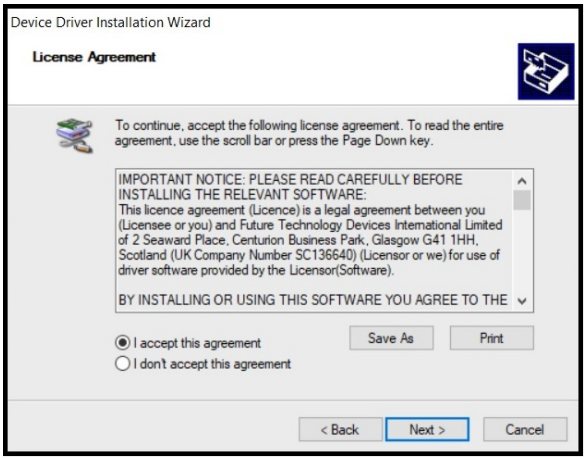
Klik op “Extract”.



Klik op “Next >”.



Aanvaard de overeenkomst en klik op “Next >”.



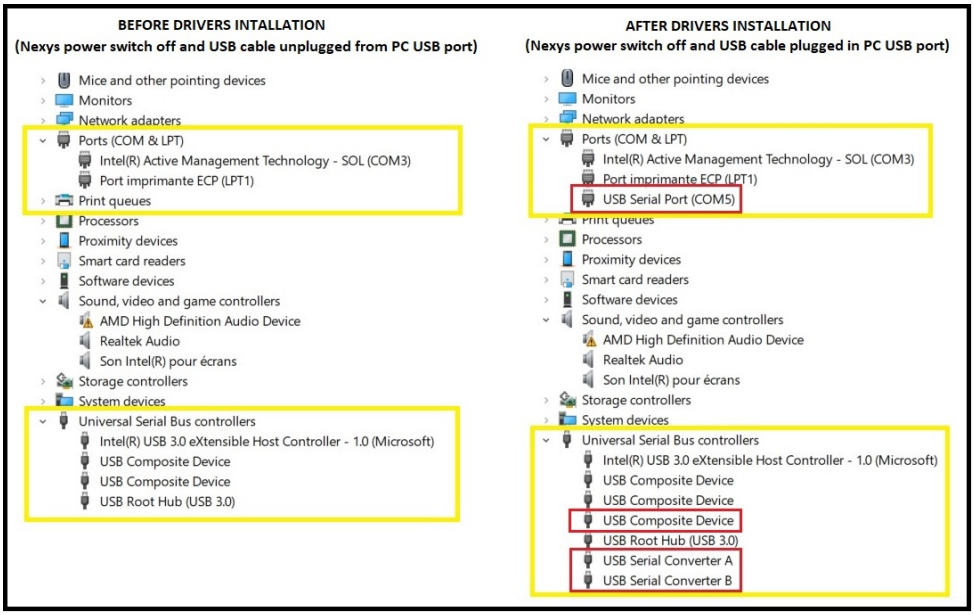
De installatie van de drivers wordt gestart.

Klik op “Finish”.



Sluit de USB-kabel aan op een USB-poort op de PC zonder de Nexys4-plaat in te schakelen.

Het aansluiten van de USB-kabel veroorzaakt het verschijnen van nieuwe apparaten.



- Er is een extra COM-poort geïnstalleerd: dit is de COM-poort die gebruikt wordt om met de Nexys4-plaat te communiceren.
- Er is een extra USB-samengesteld apparaat geïnstalleerd.
- Er zijn twee USB-serieel-omzetters geïnstalleerd.

Op dit moment is de Nexys4-plaat nog niet van stroom voorzien.

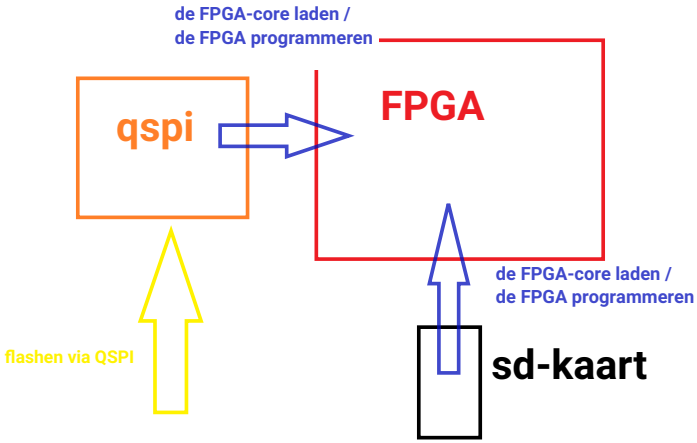
Voor meer informatie over de geïnstalleerde drivers kunt u de bijbehorende documentatie downloaden:

- https://ftdichip.com/wp-content/uploads/2020/08/AN_396-FTDI-Drivers-Installation-Guide-for-Windows-10.pdf
- https://ftdichip.com/wp-content/uploads/2021/01/AN_119_FTDI_Drivers_Installation_Guide_for_Windows7.pdf

DE HOOFD-FPGA FLASHEN MET VIVADO

Ter verduidelijking: wanneer wij spreken over het “flashen van de FPGA”, bedoelen wij in feite het flashen van het QSPI-flashgeheugen waarvan de FPGA bij het opstarten gebruikmaakt om de bitstream snel te laden.

De afbeelding hieronder toont twee veelgebruikte manieren waarop de FPGA bij het opstarten een bitstream kan laden:



- We kunnen eerst een bitstream-/core-bestand op het QSPI-flashgeheugen flashen, waarna de FPGA dit bij het opstarten snel kan laden. Het flashen van het QSPI is tamelijk traag, maar het voordeel van een korte opstarttijd is een duidelijke meerwaarde.
- Gebruikers van een Nexys-plaat kunnen een bitstream-bestand op hun SD-kaart plaatsen en de FPGA dit (enigszins langzamer) bij het opstarten vanaf daar laten laden. Hierdoor kunnen bitstreams snel gewisseld of bijgewerkt worden zonder een TE0790-03 JTAG-programmeermodule. Deze functionaliteit is echter voor later gepland voor de MEGA65.

In dit onderdeel beschrijven wij de werkwijze die gebruikmaakt van de QSPI.

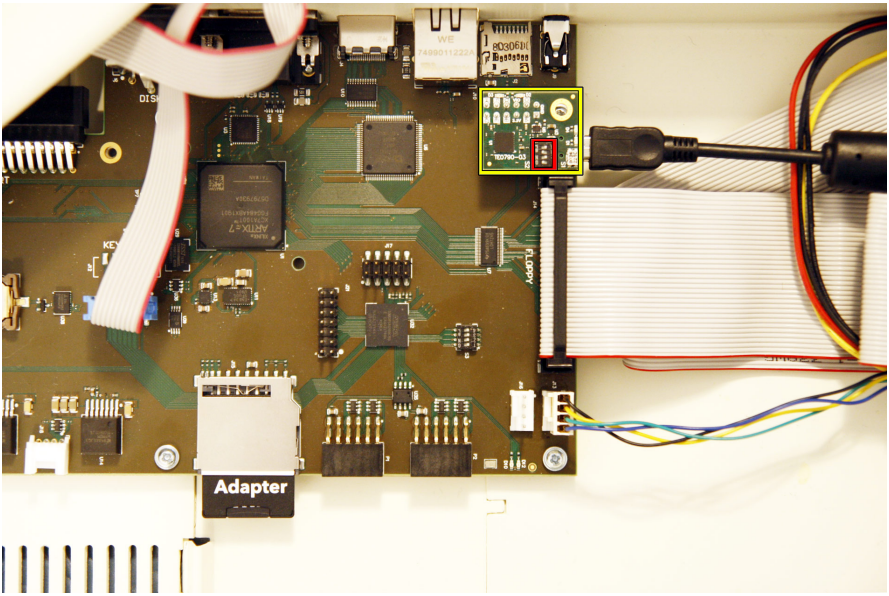
Veel van de volgende stappen in dit onderdeel zijn van toepassing op eigenaren van zowel de MEGA65 R2/R3/R3A/R4 als op eigenaren van een Nexys4-plaat. Er zijn onderweg enkele verschillen waarvan de lezer op de hoogte gesteld zal worden.

Indien u ervoor kiest om verder te gaan, heeft u een functionele installatie van het Vivado-programma van Xilinx nodig en de FTDI-drivers zoals beschreven in de voorgaande onderdelen.

U moet ook een .mcs-bitstreambestand downloaden of zelf bouwen (en een optioneel .prm-controlesomverificatiebestand) dat u via Vivado op de QSPI-chip wilt flashen. Zie het **grote Mega65-boek** voor meer details over waar dergelijke bestanden gedownload kunnen worden.

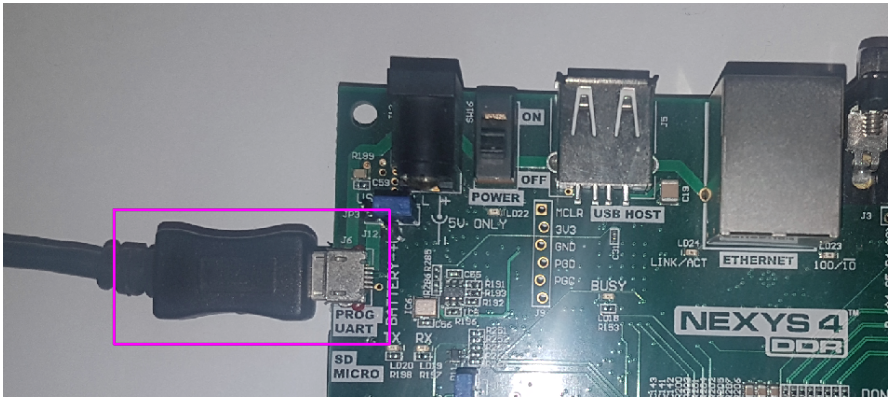
Voor eigenaren van de MEGA65 R2/R3/R3A/R4:

U heeft een TE0790-03 JTAG-programmeermodule nodig. Het is ook noodzakelijk dat dip-schakelaars 1 en 3 op de AAN-stand staan en dip-schakelaars 2 en 4 op de UIT-stand op de TE-0790. Met de MEGA65 losgekoppeld van de voeding moet de TE-0790 op de JB1-connector worden geïnstalleerd, die zich tussen de diskettedatakabel en de audiojack bevindt. De vergulde opening van de TE-0790 moet in lijn liggen met het schroefgat eronder. De mini-USB-kabel verbindt zich vervolgens aan de kant richting het 3.5" diskettestation. De volgende afbeelding toont de juiste positie: de TE0790 is omringd door de gele kast en de dip-schakelaars door de rode kast. Dip-schakelaar 1 is de dichtstbijzijnde bij de diskettedatakabel.

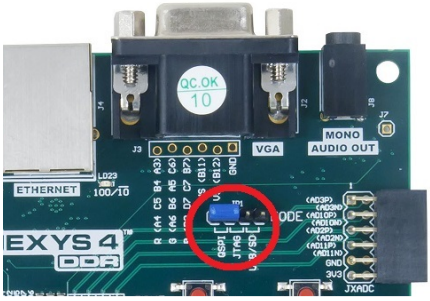


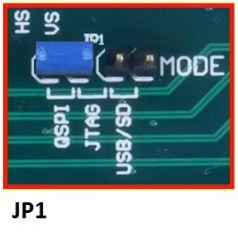
Voor eigenaren van een Nexys4-bord:

Sluit uw micro-USB-kabel aan tussen uw Nexys4-bord en uw pc via de poort met het label “PROG UART” (J6), zoals getoond:



Zet bovendien de J1-jumper op de QSPI-stand:

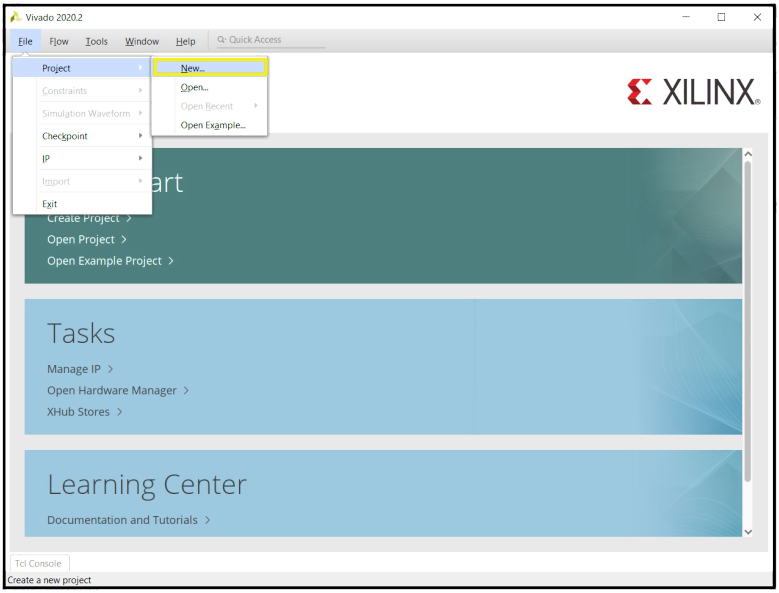




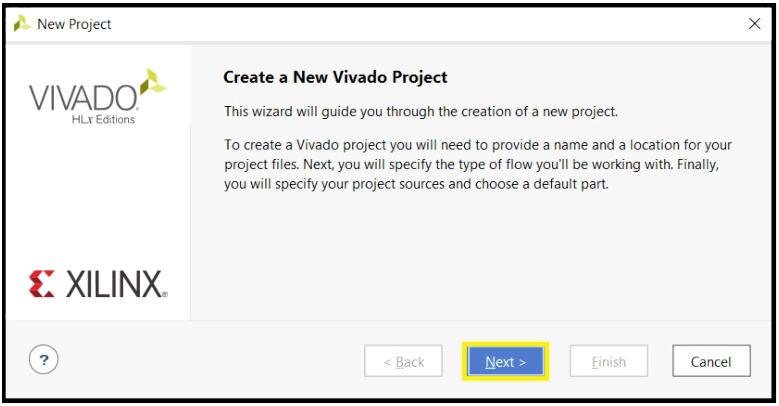
JP1

- Sluit uw niet-8-bit computer aan op het FPGA-programmeerapparaat met behulp van de juiste USB-kabel.
- Schakel de MEGA65-computer in.
- Open Vivado.

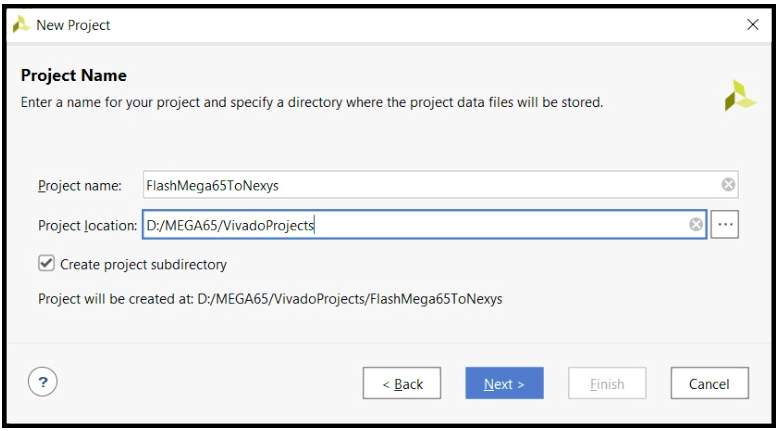
Stap 1a: Maak een nieuw Vivado-project aan via “File”, “Project”, “New...”.
LET OP: Bij toekomstige gelegenheden waarop u de QSPI moet flashen, heropent u dit project (het is niet nodig elke keer een nieuw project aan te maken).



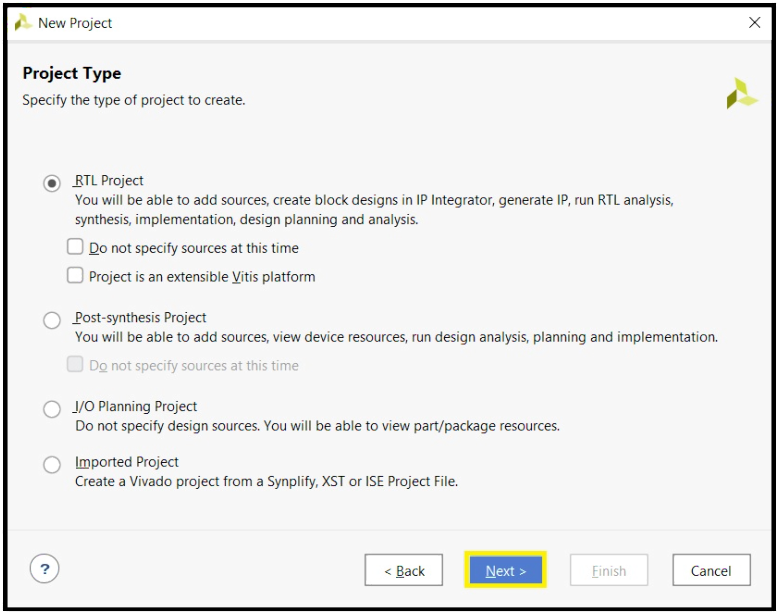
Stap 1b: De wizard “New Project” verschijnt. Klik op “Next”:



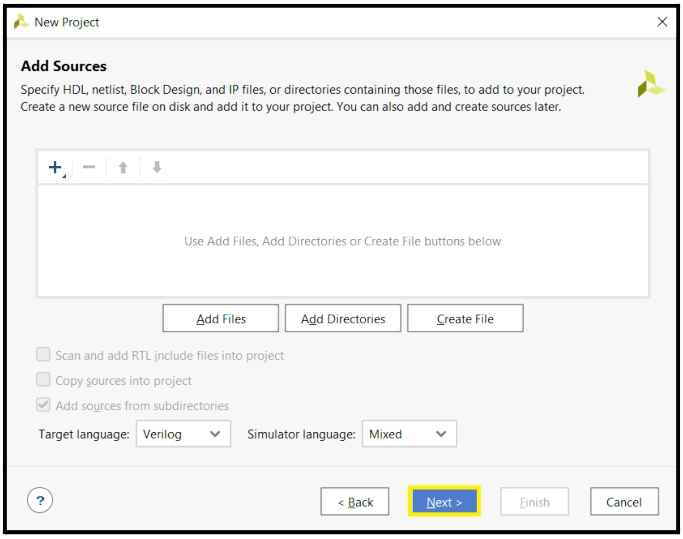
Stap 1c: Geef uw project een naam en kies de gewenste locatie en klik op “Next”:



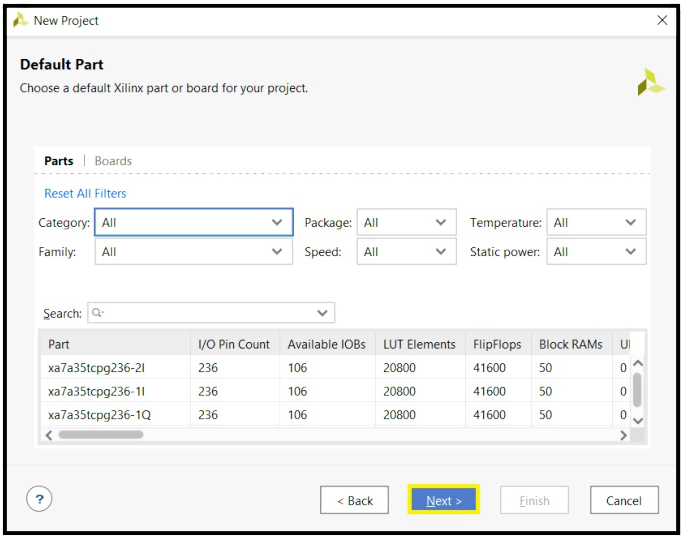
Stap 1d: Laat de standaard geselecteerde opties behouden en klik op “Next”:



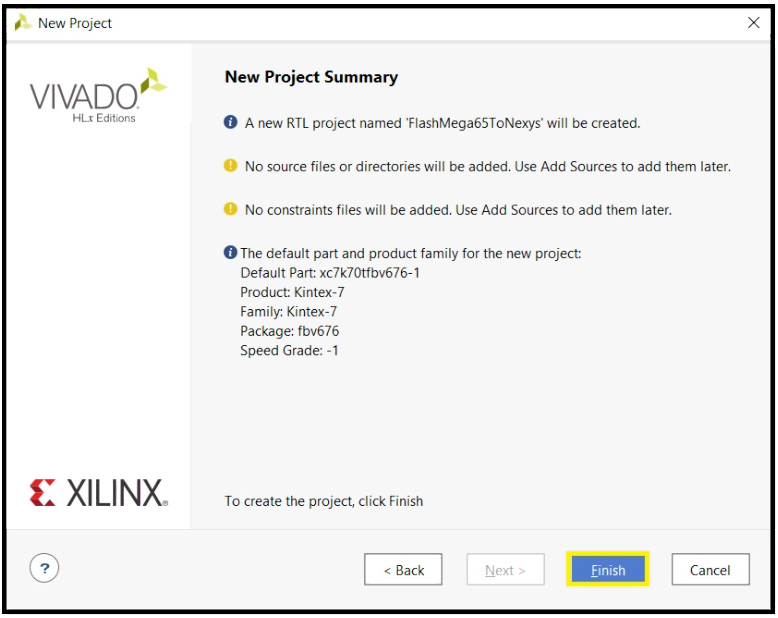
Stap 1e: Voeg geen bronnen toe, laat de standaard geselecteerde opties behouden en klik op “Next”:



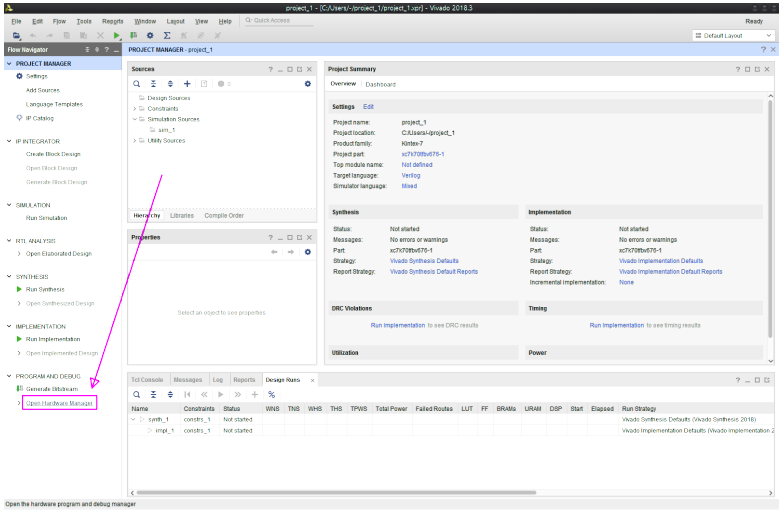
Stap 1f: Laat de standaard geselecteerde opties behouden en klik op “Next”:



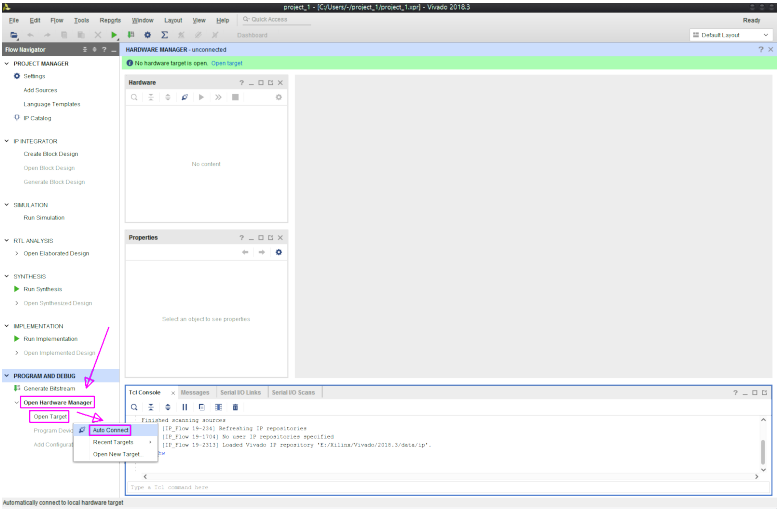
Stap 1g: Klik op “Finish”:



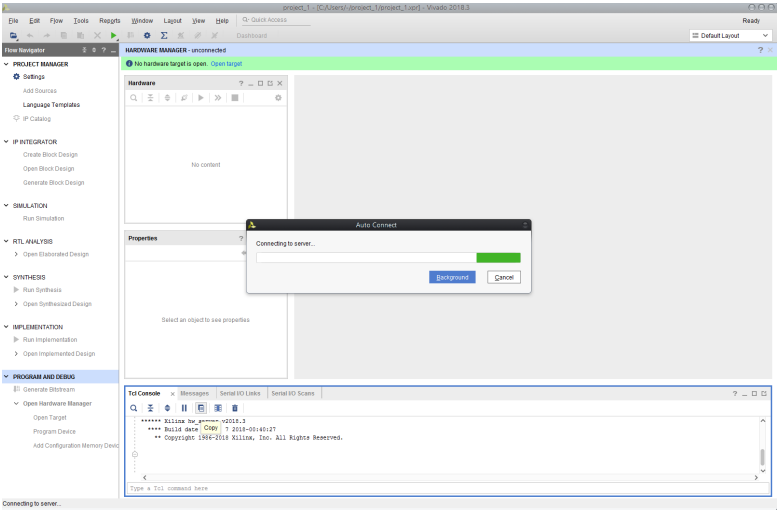
Stap 2: Selecteer in de linker kolom “Open Hardware Manager” helemaal onderaan.



Stap 3: Verbind met de FPGA:
Onder “Open Hardware Manager” kiest u “Open Target” en vervolgens “Auto Connect”.

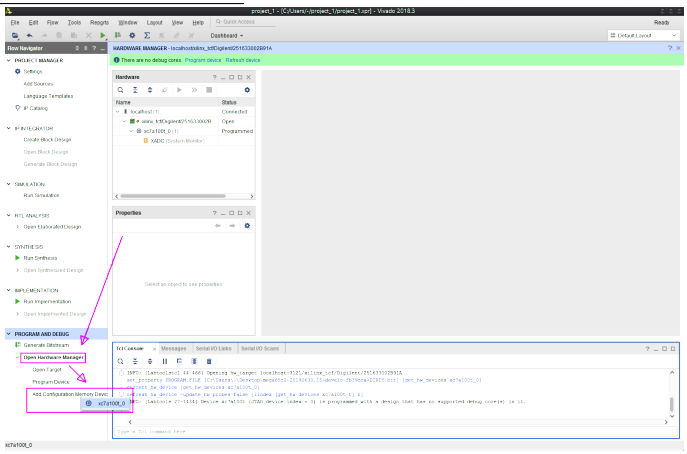


Stap 4: Wacht even; “Connecting to server...” zou automatisch moeten sluiten zonder een foutmelding aan de console te geven.



Stap 5: Onder “Open Hardware Manager” kiest u “Add Configuration Memory Device” en vervolgens:

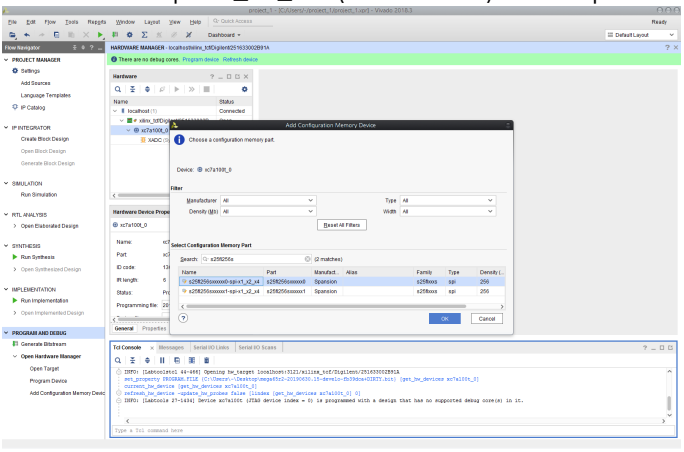
- Voor MEGA65R3/R3A/R4: “xc7a200t_0”
- Voor Nexys4 en MEGA65R2: “xc7a100t_0”.



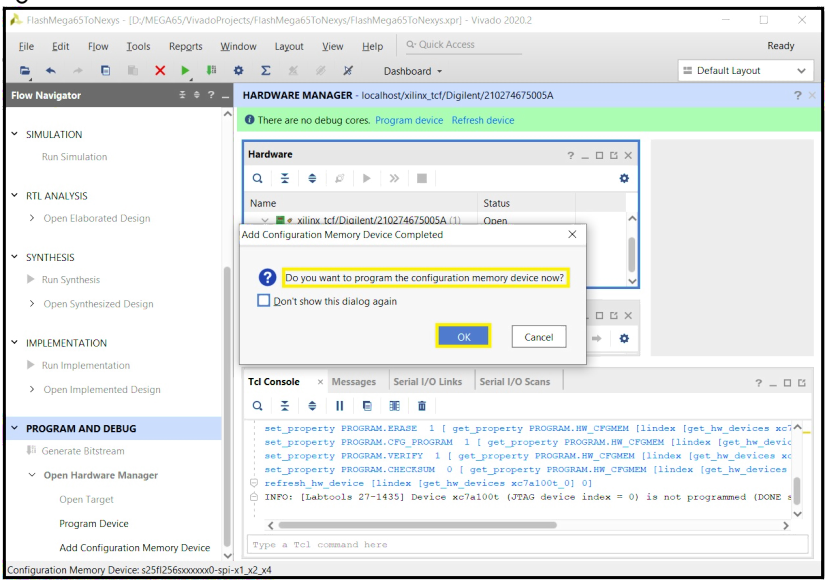
Stap 6a: Selecteer Memory Part:

In het nieuw geopende dialoogvenster:

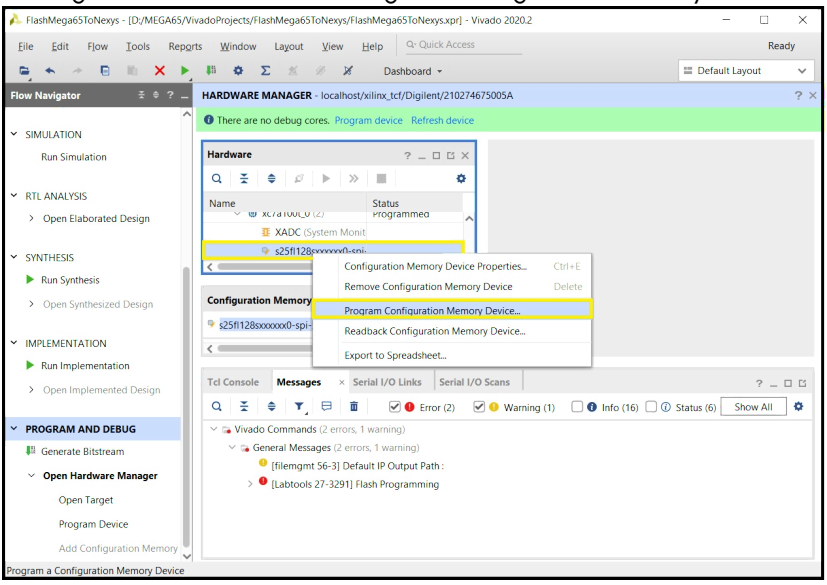
- Voor MEGA65R2/R3: typ “S25fl256s” (zonder aanhalingstekens), selecteer vervolgens “s25fl256sxxxxxxx0-spi-x1_x2_x4” (de bovenste) en klik op “OK”.
- Voor MEGA65R3A/R4: Vivado kan het grotere flashgeheugen op deze borden niet flashen. Gebruik het flashmenu op de MEGA65.
- Voor Nexys4: typ “S25fl128s” (zonder aanhalingstekens), selecteer vervolgens “s25fl128sxxxxxxx0-spi-x1_x2_x4” (de bovenste) en klik op “OK”.



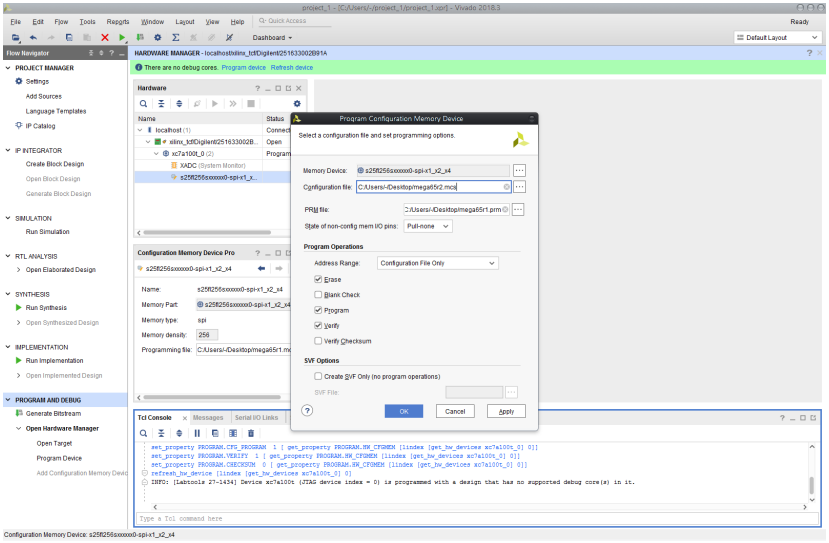
Stap 6b: Klik op “OK” om te bevestigen dat u het configuratiegeheugenapparaat nu wilt programmeren.



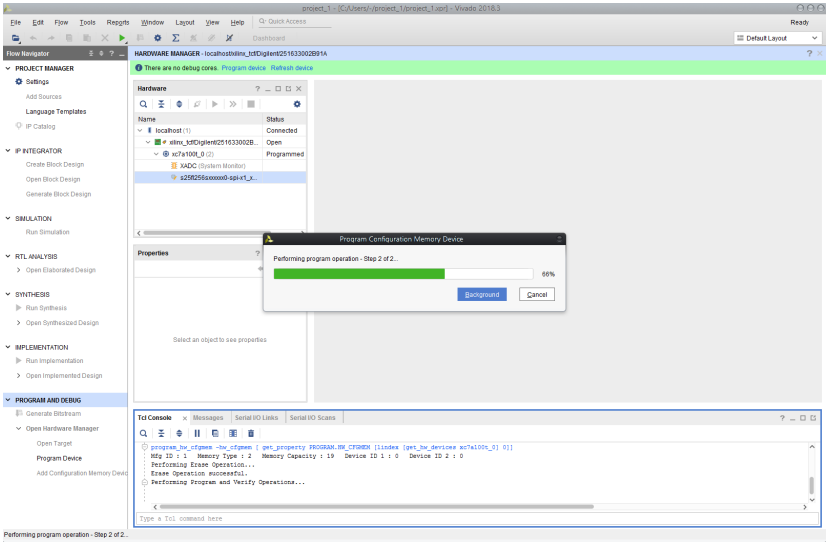
Stap 6c: Als u zo’n pop-up niet ziet of de QSPI op een later tijdstip opnieuw wilt programmeren, klikt u in het “Hardware”-venster met de rechtermuisknop op de geheugenconfiguratie en selecteert u “Program Configuration Memory Device”:



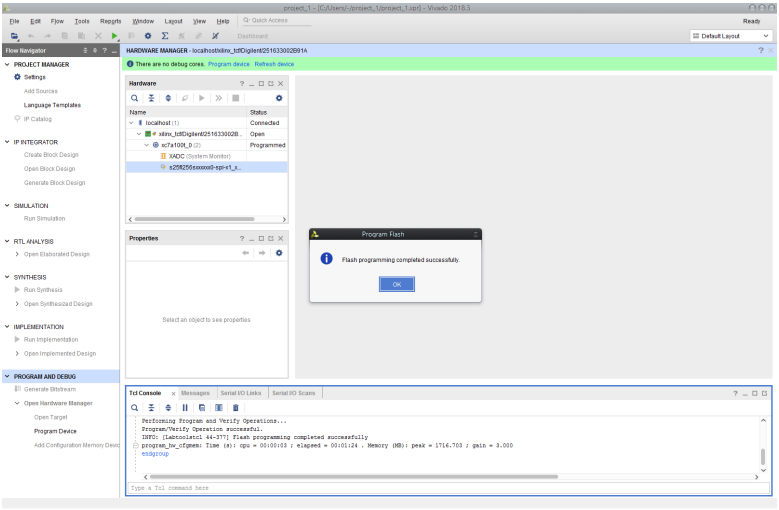
Stap 7: Stel de programmeeropties in:
Stel in het volgende dialoogvenster “Configuration file” in op het pad van uw “.mcs”-bitstream-bestand. U kunt optioneel ook het “PRM file”-veld instellen op het pad van uw “.prm”-bestand. Laat alle overige parameters zoals ze zijn (zie onderstaande screenshot).



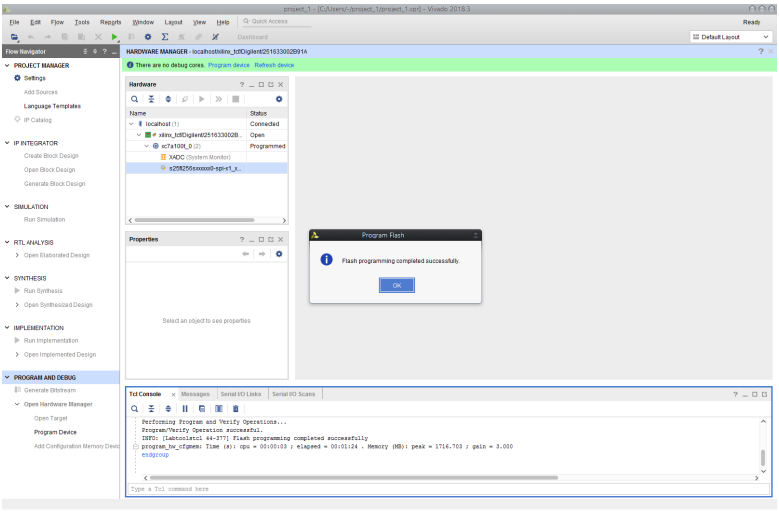
Stap 8: Wacht geduldig tot het programmeren is voltooid. Dit kan enkele minuten duren, omdat de Vivado-software het flashgeheugen dat wordt gebruikt om de FPGA bij het inschakelen te initialiseren, eerst wist en vervolgens opnieuw programmeert.



Stap 9: Als uw scherm eruitziet als de onderstaande screenshot, is uw nieuwe bitstream succesvol geflasht in Slot 0 van uw QSPI-flashgeheugen!

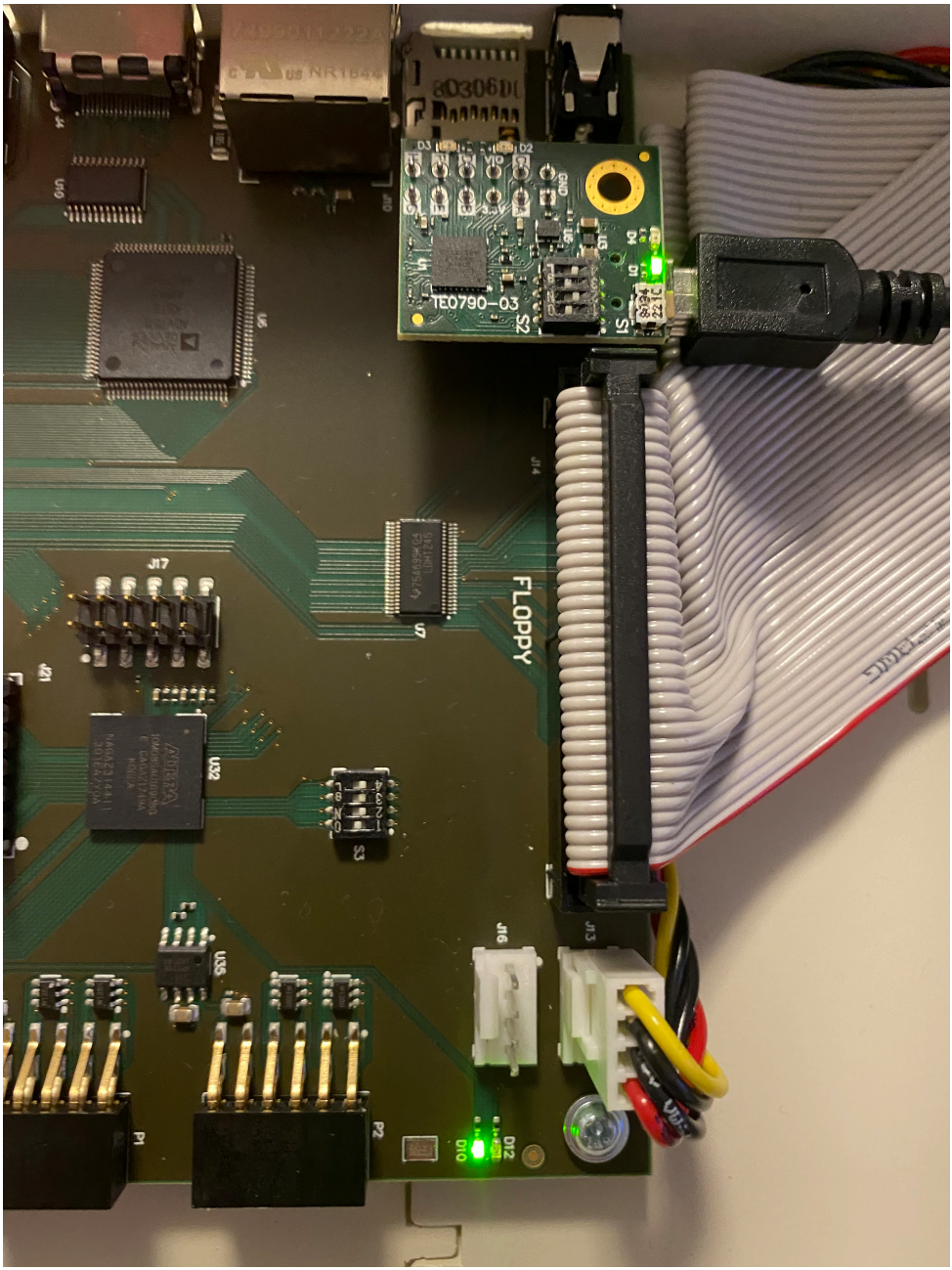


Stap 10: Als u de FPGA opnieuw wilt flashen, kan het zijn dat de optie “Add Configuration Memory Device” uit stap 5 grijs weergegeven wordt. Selecteer in dat geval “s25fl256sxxxxxxx0-spi-x1_x2_x4” in het “Hardware”-venster, druk op de rechtermuisknop en selecteer “Program Configuration Memory Device” om te flashen.



DE CPLD IN HET TOETSENBORD VAN DE MEGA65 FLASHEN MET LATTICE DIAMOND

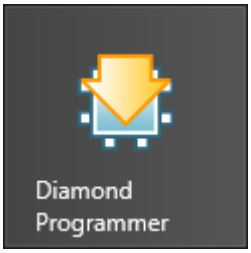
Indien u ervoor kiest om door te gaan, heeft u een TE0790-03 JTAG-programmeermodule en een werkende installatie van Lattice Diamond Programmer-software nodig. Dit kan op zowel Windows als Linux, maar in beide gevallen moet u de benodigde USB-stuurprogramma’s installeren. Het is ook noodzakelijk dat dip-schakelaars 1 en 3 op de AAN-stand staan en dip-schakelaars 2 en 4 op de UIT-stand op de TE-0790. Met de MEGA65 losgekoppeld van de voeding moet de TE-0790 op de JB1-connector worden geïnstalleerd, die zich tussen de diskettedatakabel en de audiojack bevindt. De vergulde opening van de TE-0790 moet in lijn liggen met het schroefgat eronder. De mini-USB-kabel verbindt zich vervolgens aan de kant richting het 3.5” diskettestation. De volgende afbeelding toont de juiste positie: de TE0790 is omringd door de gele kast en de dip-schakelaars door de rode kast. Dip-schakelaar 1 is de dichtstbijzijnde bij de diskettedatakabel.


TE0790-03
FLOPPY

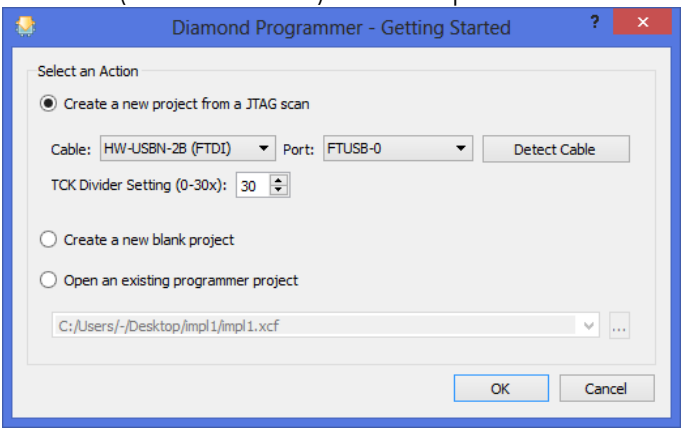
Op de PCB van een R2/R3/R3A/R4 MEGA65-moederbord moet dip-schakelaar 1 (de dichtstbijzijnde bij de gebruiker die voor de computer zit) op de AAN-stand staan. De overige schakelaars moeten UIT staan. Het toetsenbord gaat in de "ambulance-modus" (blauwe flitsende lichten) wanneer het correct is ingesteld.

Sluit uw niet-8-bit computer aan op het FPGA-programmeerapparaat met behulp van een mini-USB-kabel. Schakel de MEGA65-computer in. Open de Diamond Programmer, die van het internet kan worden gedownload.

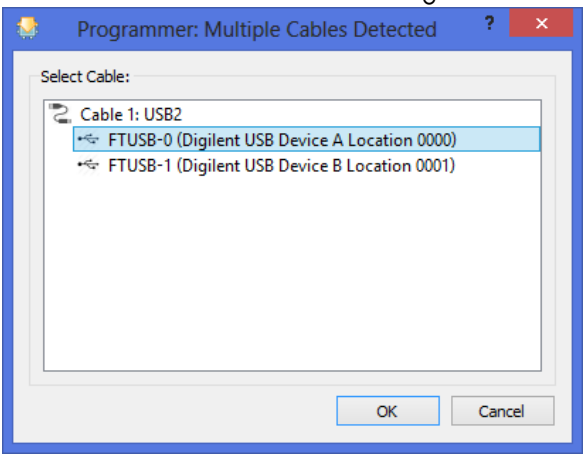
Stap 1: Open DIAMOND PROGRAMMER:
Selecteer "Create a new project from a JTAG scan". Indien het veld onder "Cable:" leeg is, klikt u op "Detect Cable".



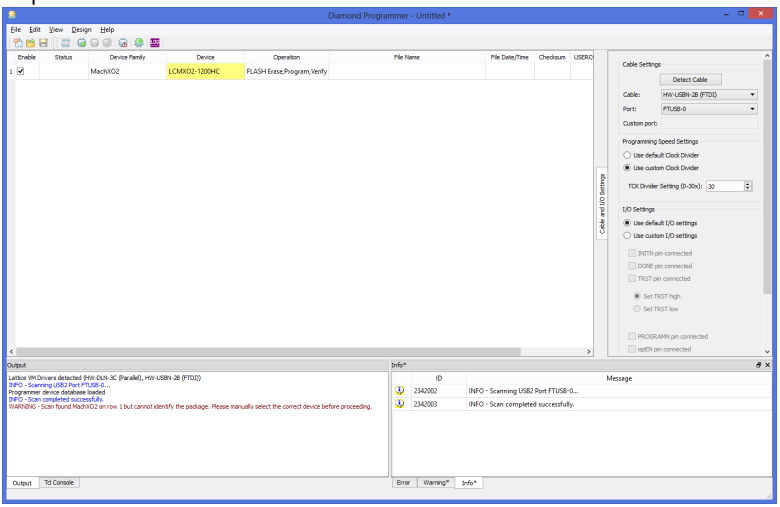
Stap 2: Maak een nieuw project aan:
Indien het dialoogvenster "Programmer: Multiple Cables Detected" verschijnt, selecteert u het eerste item ("Location 0000") en klikt u op "OK".



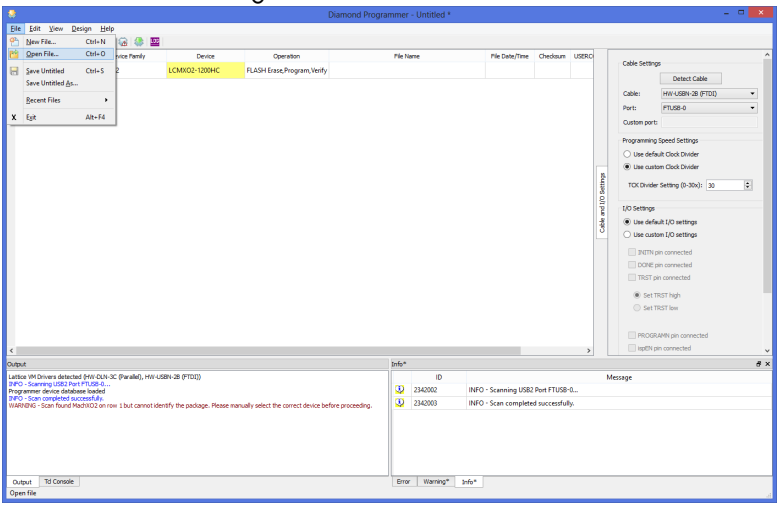
Stap 3: Selecteer kabel:
U heeft nu een nieuw project aangemaakt dat “MachXO2” onder “Device Family” en “LCMXO2-1200HC” onder “Device” zou moeten weergeven.



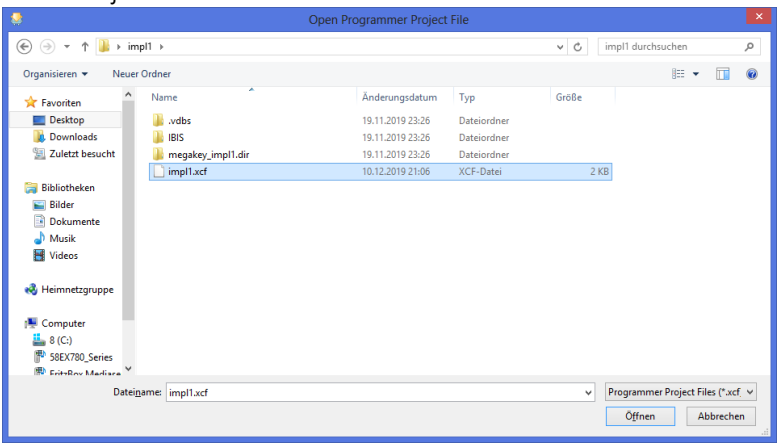
Stap 4: Nieuw Diamond Programmer-project:
Kies “File” en vervolgens “Open File” om het Diamond Programmer-project met de firmware-update voor het MEGA65-toetsenbord te laden.



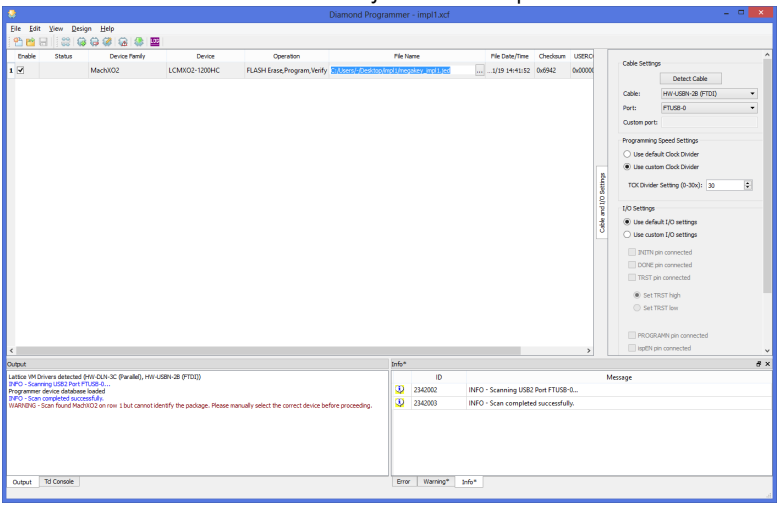
Stap 5: Open project:
Navigeer naar de map met de uitgepakte firmwarebestanden voor het MEGA65-toetsenbord die u hebt ontvangen en selecteer het bestand met de extensie ".xcf".



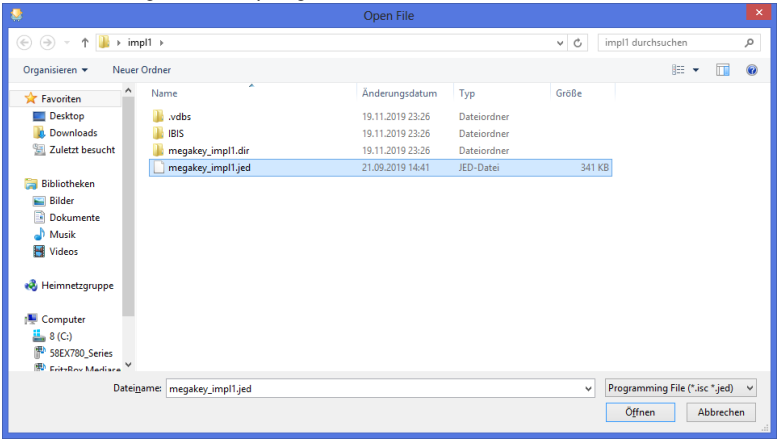
Stap 6: Selecteer projectbestand:
Klik op de drie puntjes onder "File Name" om het juiste pad in te stellen en het bestand met de extensie ".jed" te vinden.



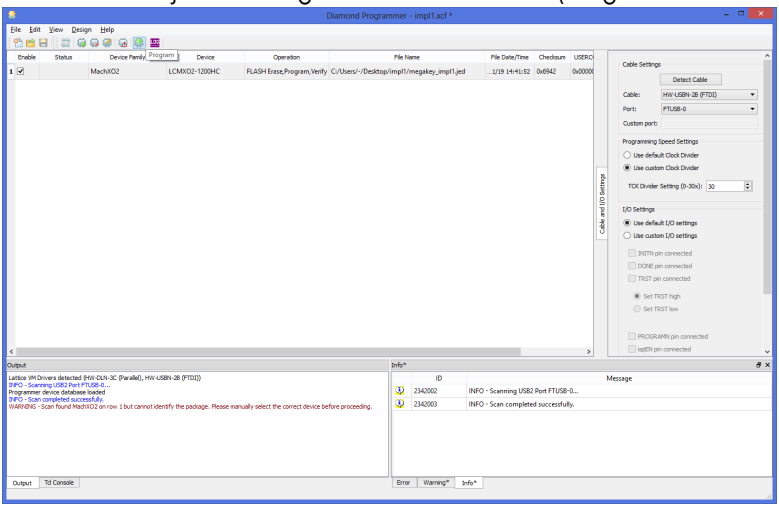
Stap 7: Kies het juiste pad van het .jed-bestand:
Selecteer het bestand met de extensie ".jed" en klik op "OK".



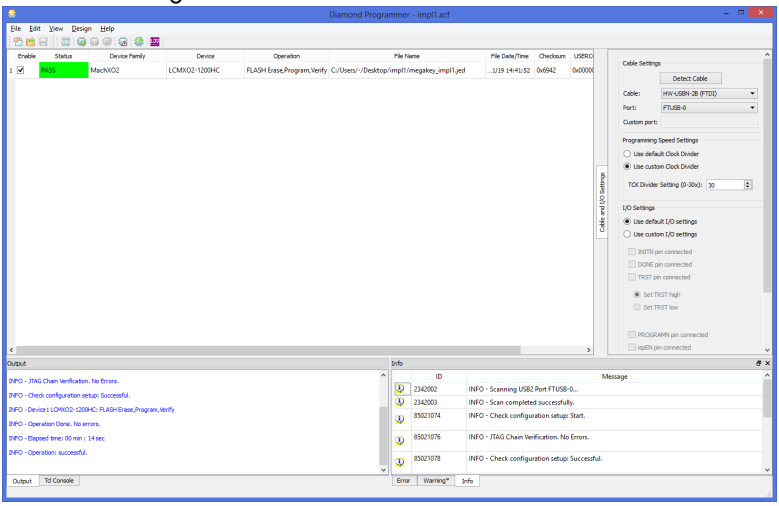
Stap 8: Selecteer .jed-bestand:
Klik op het icoon met de groene pijl naar beneden "PROGRAM", dat lijkt op het icoon van het Diamond Programmer-programma.



Stap 9: Selecteer kabel:
Na een moment zou het uitvoervenster “INFO - Operation: successful.” moeten weergeven en zou het celletje “Status” groen moeten worden (dit gebeurt niet altijd).



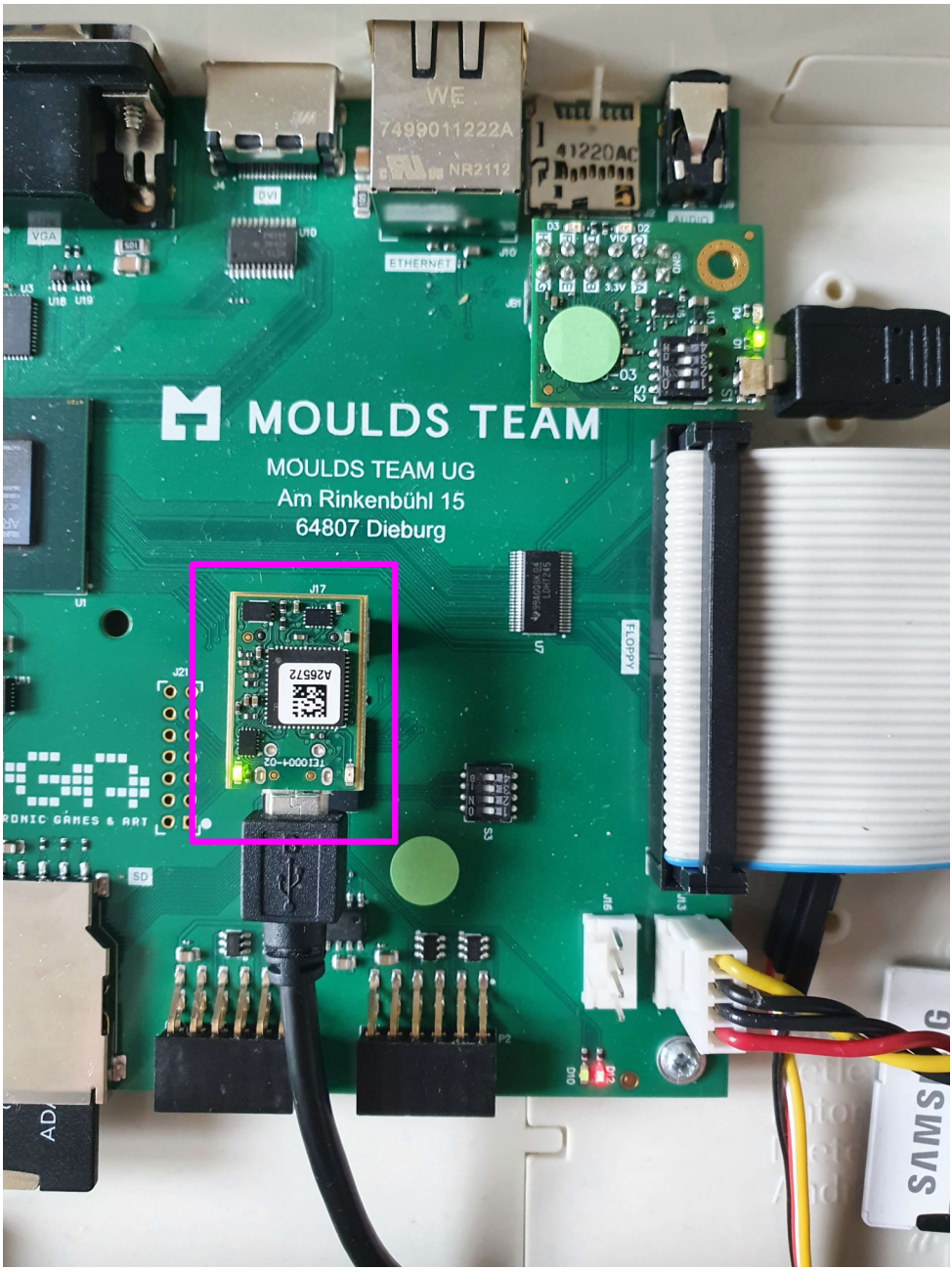
Stap 10: Operatie geslaagd:
U heeft het MEGA65-toetsenbord nu succesvol geflasht. Desgewenst kunt u het project opslaan voor later gebruik.



HET FLASHEN VAN DE MAX10-FPGA OP HET MEGA65-MOEDERBORD MET INTEL QUARTUS

Indien u verder wilt gaan, heeft u een TEI0004 - Arrow USB Programmer2-module nodig met het bijbehorende TEI0004-stuurprogramma geïnstalleerd en een goed functionerende installatie van Quartus Prime Programmer Lite Edition. Dit kan zowel op Windows als op Linux, maar in beide gevallen moet u de nodige USB-stuurprogramma's installeren. Sluit de MEGA65 niet aan op de netspanning en installeer de TEI0004 op de J17-connector, die zich tussen de datakabel van de diskette en de ARTIX 7-FPGA op het moederbord bevindt. De micro-USB-poort van de TEI0004 wijst in de tegenovergestelde richting van de HDMI- en LAN-aansluitingen, naar het klepje. De volgende afbeelding toont de juiste positie.

Op het PCB van een R2/R3/R3A/R4-MEGA65-moederbord moeten alle dipschakelaars in de positie uit staan. De hoofd-FPGA van de MEGA65 mag geen geldige bitstream bevatten. De eenvoudigste manier om dit te bereiken is met een TE0790-JTAG-adapter: u kunt dan het m65-hulpprogramma vanaf een aangesloten computer gebruiken om een bitstream via JTAG te verzenden en vervolgens Control-C gebruiken om het m65-hulpprogramma te onderbreken voordat de bitstream volledig is geladen.

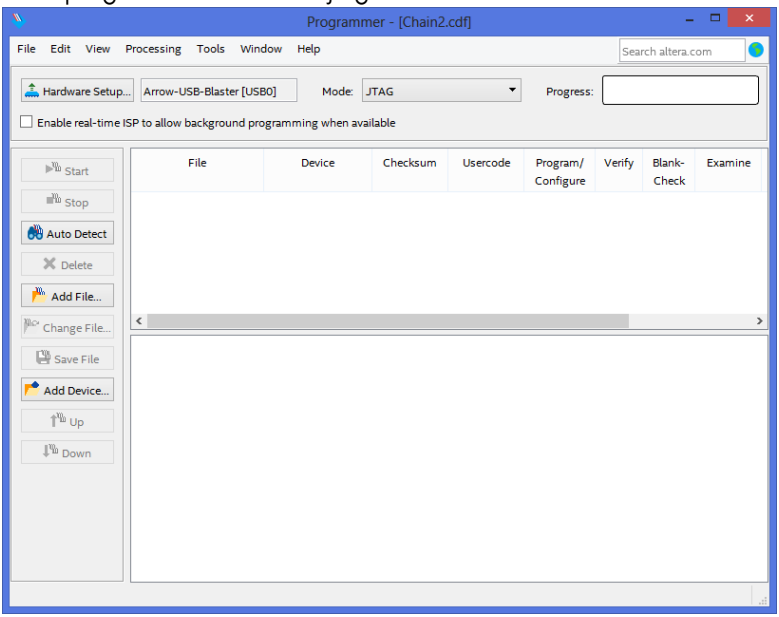

MOULDS TEAM
MOULDS TEAM UG
Am Rinkenbühl 15
64807 Dieburg
ETHERNET
DVI
VGA
FLOPPY
SD
J17
J21
A26572

Sluit uw niet-8-bit computer aan op het FPGA-programmeerapparaat met behulp van een micro-USB-kabel. Open Quartus Prime Programmer Lite Edition, dat van internet kan worden gedownload.

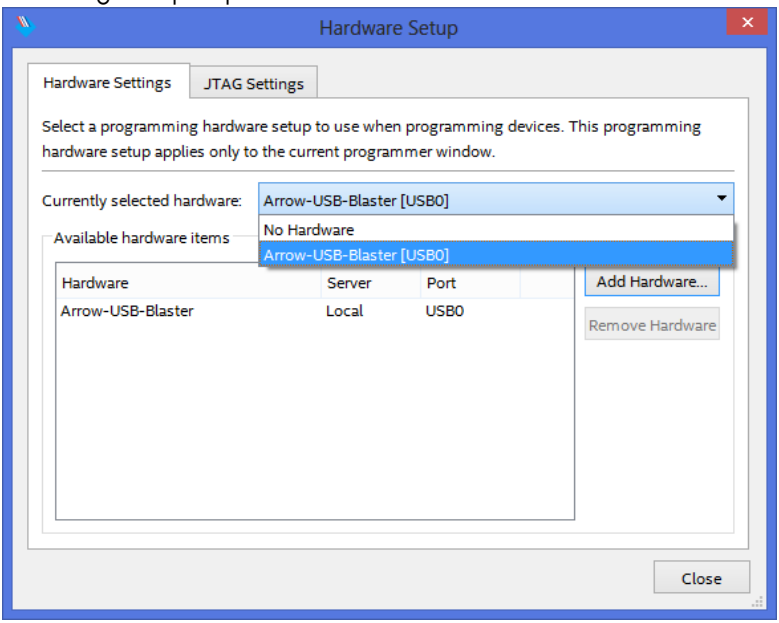
Stap 1: Open Quartus Prime Programmer Lite Edition:
Klik op de knop “Hardware Setup” in de linkerbovenhoek van het Quartus Prime Programmer-venster.



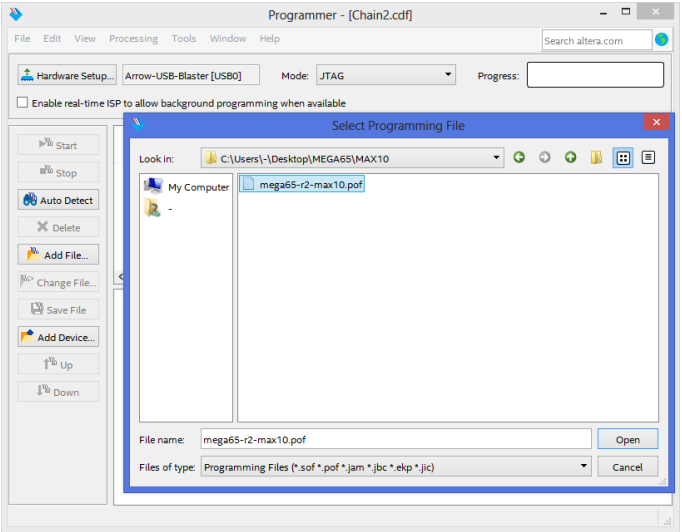
Stap 2: Hardware-instellingen invoeren:
In het nieuw verschenen venster kiest u onder “Currently selected hardware” voor “Arrow-USB-Blaster”. Indien “Arrow-USB-Blaster” niet verschijnt, controleert u of de kabel en stuurprogramma’s correct zijn geïnstalleerd.



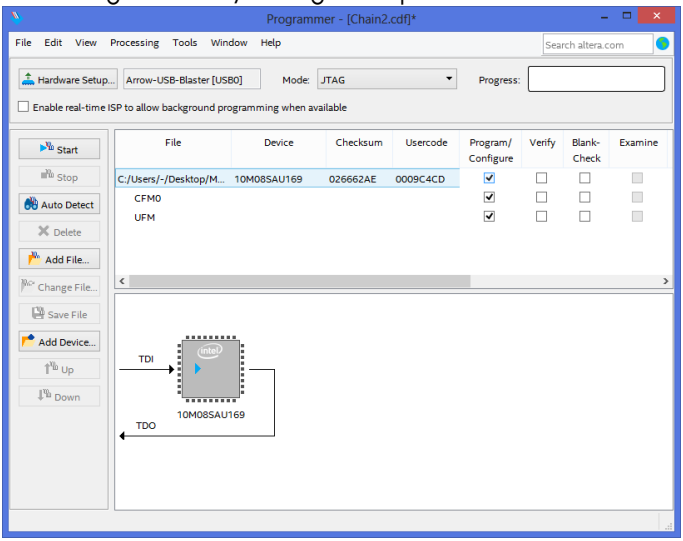
Stap 3: Selecteer Arrow USB-Blaster:
Klik op de knop “Add File” in de linkerrij en kies het nieuwste bestand met de extensie “.pof”. Klik vervolgens op “Open”.



Stap 4: Selecteer programmeerbestand:
Vink ten minste de drie vakjes onder “Program/Configure” aan. Het inschakelen van alle vakjes onder “Verify” en “Blank-Check” maakt het proces betrouwbaarder.



Stap 5: Selecteer Programmeer-/Configureeropties:

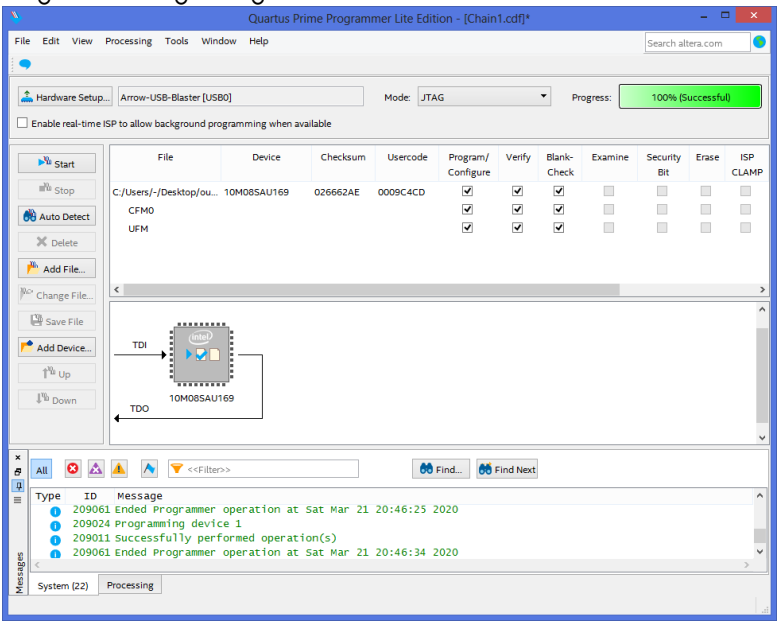


Houd de Reset-knop ingedrukt en schakel de MEGA65-computer in. Het toetsenbord gaat in de “ambulance mode” (blauwe flitsende lichten). Indien dit niet het geval is, is de hoofd-FPGA niet leeg: start het gehele proces opnieuw.

Klik nu op “Start” in de linkerrij van knoppen. De voortgangsbalk rechtsboven zou snel naar 100 procent moeten gaan en groen moeten worden. U heeft de MAX10-FPGA nu succesvol bijgewerkt.

Indien u in plaats daarvan een foutmelding ontvangt, controleer dan of de bitstream van de hoofd-FPGA is gewist en of u de reset-knop op de MEGA65 niet te vroeg heeft losgelaten. Schakel de MEGA65 uit en herhaal deze stap.

Stap 6: Programmeren geslaagd:



BIJLAGE AC

Problemen oplossen

- Hardware
- Vivado
- mega65_ftp

HARDWARE

Geen digitale video-uitvoer

MEGA65 R4, R5, R6

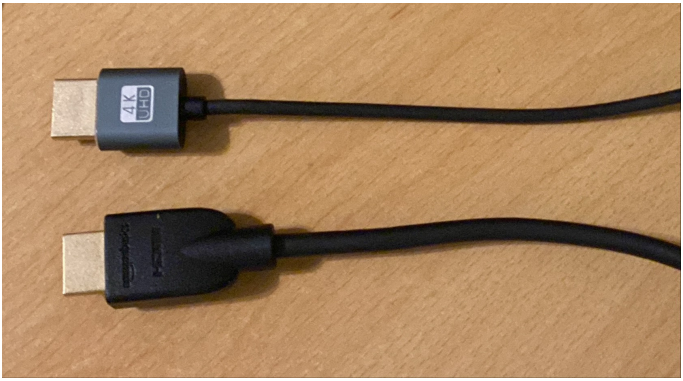
De digitale videopoort heeft een verbeterde isolatie tegen terugvoeding gekregen om het probleem "Geen lampjes bij het inschakelen" op te lossen. Een neveneffect hiervan is dat sommige digitale videokabels ten onrechte alleen de behuizing van de connector gebruiken om het grondsignaal door te geven, in plaats van de grondsignalen van de pinnen van de connector te gebruiken. Kort samengevat: als u geen beeld krijgt op de digitale videopoort, kunt u een van de volgende oplossingen proberen:

- Probeer verschillende digitale videokabels die u heeft. Dit probleem komt alleen voor bij bepaalde merken kabels, vooral dunnere exemplaren, waarvan wij sterk vermoeden dat de interne grondleidingen ontbreken om kosten te besparen. Dit is de beste oplossing, omdat de kabel daadwerkelijk de oorzaak van het probleem is doordat deze niet aan de specificaties voldoet. Zelfs als u een van de volgende oplossingen toepast, moet u toch op zoek gaan naar een betere HDMI-kabel die het probleem bij de bron oplost.
- Sluit een VGA-kabel aan op de MEGA65 en op hetzelfde beeldscherm. Hierdoor wordt een grondverbinding via de VGA-connector tot stand gebracht, waardoor de HDMI op mysterieuze wijze plotseling lijkt te werken.
- Sluit een TE0790 JTAG-adapter aan. Een gebruiker meldde dat dit het probleem eveneens op mysterieuze wijze omzeilde.
- Verbind de behuizing van de digitale video-connector met de behuizing van de interne SD-kaart op vol formaat.
- Sluit een splitter of repeater aan op de digitale videopoort.

Geen lampjes bij het inschakelen

Als de MEGA65 bij het inschakelen soms geen lampjes laat zien, heeft dit te maken met bepaalde digitale video-apparaten die op het moment dat de MEGA65 uitstaat aangesloten zijn en onvoldoende vermogen leveren om de CPLD van het toetsenbord correct van stroom te voorzien, maar wel genoeg om te voorkomen dat deze goed kan resetten wanneer de MEGA65 wordt ingeschakeld. Het verwijderen van de digitale videokabel en het opnieuw inschakelen van het apparaat lost het probleem op.

Figuur AC.1: Voorbeeld van een slechte (boven) en goede (onder) digitale videokabel. Let op hoe de slechte kabel verdacht dunner is dan de goede. Dit is een teken dat waarschijnlijk dedicated grondleidingen ontbreken, wat problemen veroorzaakt op de MEGA65 R4 en latere revisies. Elke kabel van "normale dikte" zou echter goed moeten werken. U hoeft geen van die peperdure "met zilverdraad gevlochten, bij maanlicht door elfen gesmeede" kabels te kopen. Die kabels bieden ongeveer evenveel praktisch nut als het strooien van glitter of het tekenen van een eenhoorn aan één uiteinde, maar dan met veel minder klasse.



VIVADO

RAM-vereisten

```
INFO: [Synth 8-256] done synthesizing module 'ram32x1024' [/home/....]
INFO: [Synth 8-256] synthesizing module 'charrom' [/home/....]
   /opt/Xilinx/Vivado/2019.2/bin/loader: line 280: 2317 killed
   WARNING: [Vivado 12-8222] Failed run(s) : 'synth\_1'
       ERROR: Application Exception: failed to launch run 'impl\_1' due
           to failures in the following run(s):
       synth\_1
       These failed run(s) need to be reset prior to launching 'impl\_1'
           again.
```

Deze fout wordt veroorzaakt doordat Vivado crasht omdat het apparaat onvoldoende RAM heeft om Vivado uit te voeren. Vivado heeft minimaal 4 GB RAM nodig om het MEGA65-doel te synthetiseren, maar 8 GB is beter.

MEGA65_FTP

Ontbrekende bibliotheek

```
/usr/bin/ld: cannot find -lncurses
collect2: error: ld returned 1 exit status
Makefile:474: recipe for target 'bin/mega65_ftp' failed
make: *** [bin/mega65_ftp] Error 1
```

Deze fout treedt op wanneer de ncurses-bibliotheek op de computer ontbreekt bij het bouwen van het mega65_ftp-programma. Om dit probleem op te lossen, moet u deze afhankelijkheid installeren.

```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

BIJLAGE AD

Modelspecifieke kenmerken

- MEGA65-modellen detecteren
- MEGA65-desktopcomputer, revisies 5 en 6
- MEGA65-desktopcomputer, revisie 4
- MEGA65-desktopcomputer, revisie 3 & revisie 3A
- MEGA65-desktopcomputer, revisie 2
- MEGAphone-handheld, revisies 1 en 2
- Nexys4 DDR FPGA-bord

MEGA65-MODELLEN DETECTEREN

Hoewel we verwachten dat de productieversie van de MEGA65 een stabiel platform is, kunnen er zich nog situaties voordoen waarin het nodig is vast te stellen op welke hardware uw programma draait. Dit is met name belangrijk voor de systeemsoftware van de MEGA65, welke mogelijk verschillende hardwareonderdelen op de verschillende modellen moet initialiseren. Omdat er al een handheld-versie van de MEGA65 in ontwikkeling is, welke een scherm met een iets andere resolutie gebruikt (800x480 in plaats van 720x576) en wel een touchscreen heeft, maar geen hardwaretoetsenbord. wilt u wellicht programma's maken die zich op een soepelere manier aan deze handheld-apparaten aanpassen. U kunt bijvoorbeeld touchscreeninvoer inschakelen en de keuzemogelijkheden op het scherm zo herindelen dat ze groot genoeg zijn om gemakkelijk met een vinger te activeren.

De eenvoudige manier om vast te stellen op welk model MEGA65 uw programma draait, is het controleren van het register $D629 (schakel echter eerst de MEGA65-I/O-persoonlijkheid in, via $D02F). Dit register bevat een 8-bits hardware-identificatie. De volgende waarden zijn momenteel gedefinieerd:

$01 (1) MEGA65 R1

$02 (2) MEGA65 R2

$03 (3) MEGA65 R3

$04 (4) MEGA65 R4

$05 – $0F (5 – 15) MEGA65 R5 – MEGA65 R15

$21 (33) MEGAphone (handheld) R1

$40 (64) Nexys4 PSRAM

$41 (65) Nexys4DDR

$42 (66) Nexys4DDR met widget-bord

$FD (253) QMTECH Wukong A100T-bord

$FE (254) VHDL-simulatierun

MEGA65-DESKTOPCOMPUTER, REVISIES 5 EN 6

De desktop-PCB's van R5 en R6 vormen betrekkelijk bescheiden herzieningen van de desktop-PCB van R4, die oorspronkelijk werd uitgevoerd om problemen met de toeleveringsketen voor enkele voedingsregelaarcomponenten op te lossen. Hierbij hebben de ontwerpers tevens de gelegenheid aangegrepen om enkele verbeteringen door te voeren:

- Enkele signalen op de uitbreidingspoort die voorheen alleen uitvoer of alleen invoer waren, zijn nu bidirectioneel, zodat de C64-core naar verwachting alle bekende cartridges kan ondersteunen en de cartridgecompatibiliteit van de MEGA65-core verder kan worden verbeterd.
- Op het moederbord zijn vier extra DIP-schakelaars toegevoegd. Alle acht DIP-schakelaars zijn nu aangesloten via een I2C-I/O-expander, in plaats van dat zij rechtstreeks FPGA-pinnen in beslag nemen.
- Deze nieuwe I/O-expander biedt tevens informatie over de major- en minorrevisie van het moederbord. De hoofdrevisie is ingesteld op 5 en de subrevisie aanvankelijk op 0. De minorrevisie kan veranderen wanneer er wijzigingen aan het bord worden doorgevoerd die de compatibiliteit niet aantasten, zoals het vervangen van de QSPI-flashchip, de HyperRAM- en/of SDRAM-chips. Zo kan software, met name de Hypervisor van de MEGA65, deze situatie vaststellen en daarnaar handelen.
- De microfooninterface op de 3.5 mm-audiojack kan nu voorspanning leveren, zodat bepaalde typen microfoons kunnen worden gebruikt die voorheen niet werkten. De MEGA65-core ondersteunt het gebruik van de microfoonpoort momenteel nog niet, al kan dat in de toekomst veranderen.

R6 is elektrisch identiek aan R5 en is het model dat is meegeleverd met de MEGA65-productie-exemplaren die in het jaar 2024 zijn verscheept.

MEGA65-DESKTOPCOMPUTER, REVISIE 4

De desktop-PCB van R4 is een aanzienlijke herziening van de desktop-PCB van R3, met enkele belangrijke wijzigingen:

- De ondersteunende Intel™ MAX10™-FPGA is volledig van het bord verwijderd, wat het ontwerp vereenvoudigt en helpt om een deel van de kostenstijgingen op te vangen die zich hebben voorgedaan sinds het R3-bord werd ontworpen. Dit elimineert tevens enkele lastige foutmodi die zouden kunnen optreden als de MAX10-FPGA per ongeluk zijn configuratie zou verliezen.
- De MEGA65 kan de joystickpoorten nu ook laag trekken. Dat zou het gebruik van apparaten zoals de protopad™ met de MEGA65 mogelijk moeten maken.
- De aansluitingen voor de interne luidspreker op het bord zijn verwijderd: zij werden weinig gebruikt, brachten kosten met zich mee en konden door het gebruikte type DAC's problemen met oververhittende luidsprekers geven.
- De real-time klok (RTC) is vervangen door een RV-3032-C7. Dit neemt de problemen weg die zich bij de R3-borden voordeden door RTC-chips van het oude type met wisselende kwaliteit, die in veel gevallen vervangen moesten worden.

- Op de digitale video-uitvoer is een driver-IC toegevoegd om problemen door terugvoeding te voorkomen.
- Er is een nieuwe 16-polige header, J21, toegevoegd, met directe toegang tot de uitvoerlijnen van de joystick, alsmede twee extra FPGA-pinnen. Deze is bedoeld om bepaalde ontwikkeltaken op laag niveau te vergemakkelijken en om eventueel verdere verbeteringen mogelijk te maken door de toevoeging van een gespecialiseerd dochterbord.
- Naast de 8MiB HyperRAM is een 64MiB SDRAM toegevoegd. Dat maakt het overzetten van meer MiSTer-cores mogelijk en biedt de MEGA65-core bovendien een grotere en snellere Attic RAM. Waarschijnlijk wordt de HyperRAM op productie-exemplaren van het R4-bord niet gemonteerd, zodra de MEGA65-core de SDRAM ondersteunt, want de SDRAM is op alle punten beter dan de HyperRAM: hij is sneller, heeft een lagere latency en heeft een 8× grotere capaciteit.
- Aanzienlijk verbeterde geluidskwaliteit op de externe 3.5 mm-audiojack door de toevoeging van een hoogwaardige audio-DAC-chip.

MEGA65-DESKTOPCOMPUTER, REVISIE 3 & REVISIE 3A

De desktop-PCB van R3 lijkt sterk op de desktop-PCB van R2, met twee belangrijke wijzigingen:

- Ten eerste heeft de R3-PCB geen ADV7511-driverchip voor digitale video, zodat het I2C-registerblok voor dat apparaat ontbreekt.
- Ten tweede gebruikt de R3-PCB een andere versterker op het bord voor de pc-luidsprekers, die nu in stereo aanwezig zijn in plaats van in mono zoals op de R2-PCB. De versterker op de R3-PCB is dezelfde als die op de PCB's van de MEGAphone R1 – R2. De I2C-registers bevinden zich echter op een ander adres. Op de R3-PCB van de MEGA65 liggen de registers op $FFD7 1DC – $FFD7 1EF.

Het enige verschil tussen de R3 en de R3A is de QSPI-flashchip die op het bord is gemonteerd. De PCB zelf is ongewijzigd. De QSPI-flashchip van de R3 biedt slechts ruimte voor vier coreslots (0 – 3), met inbegrip van de fabriekscore in slot 0. De R3A gebruikt een QSPI-flashchip met een grotere capaciteit, die ruimte biedt voor acht coreslots (0 – 7), eveneens met inbegrip van de fabriekscore in slot 0.

R3-borden werden gebruikt voor de MEGA65 DevKit-modellen die in het jaar 2020 werden geleverd. R3A-borden werden gebruikt voor de MEGA65-productie-exemplaren die in 2022 en 2023 werden geleverd.

MEGA65-DESKTOPCOMPUTER, REVISIE 2

De desktopversie van de MEGA65 bevat een real-time klok (RTC), waarin tevens een kleine hoeveelheid niet-vluchtig geheugen (NVRAM) is opgenomen dat zijn waarde behoudt, zelfs als de computer is uitgeschakeld en van de voeding is losgekoppeld. De NVRAM bewaart zijn waarden zolang de interne batterij over voldoende lading beschikt. Deze batterij voedt tevens de real-time klok zelf, welke een eeuwkalender voor de jaren 2000 - 2099 bevat.

De voornaamste kunstgreep om de RTC vanuit BASIC te benaderen is dat we een Enhanced-DMA-bewerking van de MEGA65 nodig hebben om de RTC-registers op te halen, omdat de RTC-registers boven de 1MB-grens liggen, de bovengrens van de normale DMA-bewerkingen van de C65. De eenvoudigste manier om dat te doen is om ergens in het geheugen een klein DMA-lijstje op te bouwen en een routine in assembleertaal te schrijven die deze lijst gebruikt. Iets als het onderstaande (met BASIC 65 in de C65-modus):

```
10 RESTORE 110:FORI=0TO43:READA$:POKE1024+I,DEC(A$):NEXT:BANK 128:SYS1042
20 S=PEEK(1056):M=PEEK(1057):H=PEEK(1058)
30 D=PEEK(1059):MM=PEEK(1060):Y=PEEK(1061)+DEC("2000")
40 IF H AND 128 GOTO 80
50 PRINT "THE TIME IS ";RIGHT$(HEX$(H AND 63),2);":";RIGHT$(HEX$(M),2);":";RIGHT$(HEX$(S),2)
60 IF H AND 32 THEN PRINT "PM": ELSE PRINT "AM"
70 GOTO 90
80 PRINT "THE TIME IS ";RIGHT$(HEX$(H AND 63),1);":";RIGHT$(HEX$(M),2);":";RIGHT$(HEX$(S),2)
90 PRINT "THE DATE IS ";RIGHT$(HEX$(D),2);"-";RIGHT$(HEX$(MM),2);"-";HEX$(Y)
100 END
110 DATA 0B,80,FF,81,00,00,00,08,00,10,71,0D,20,04,00,00,00,00
120 DATA A9,47,8D,2F,D0,A9,53,8D,2F,D0,A9,00,8D,02,D7,A9
130 DATA 04,8D,01,D7,A9,00,8D,05,D7,60
```

Dit programma werkt doordat het eerst een DMA-lijst opbouwt in het geheugen op 1024 ($0400), wat op de C65 normaal gesproken ongebruikt is, gevolgd door een routine op 1042 ($0412) die ervoor zorgt dat de MEGA65-registers niet langer verborgen zijn, daarna de registers van de DMA-controller passend instelt om de DMA-taak in gang te zetten en vervolgens terugkeert. De rest van de BASIC-code leest met PEEK de RTC-registers uit die de DMA-taak naar 1024 - 1032 ($0400 - $0407) heeft gekopieerd en interpreteert die op de juiste wijze om de tijd weer te geven.

Wie nieuwsgierig is, kan het commando MONITOR gebruiken en daarna D1012 intypen om de routine te bekijken.

Wilt u een klok die doorloopt, dan kunt u regel 100 vervangen door GOTO 10. U krijgt dan zoiets als het volgende:

```
THE TIME IS 10:05:36 PM
THE DATE IS 20.02.2020
THE TIME IS 10:05:36 PM
THE DATE IS 20.02.2020
THE TIME IS 10:05:36 PM
THE DATE IS 20.02.2020
THE TIME IS 10:05:36 PM
THE DATE IS 20.02.2020
...
```

Als u eerst POKE0,65 gebruikt om de CPU op volledige snelheid in te stellen, kan het gehele programma vele malen per seconde draaien. Er kan zich af en toe een hapering voordoen als de RTC-registers worden gelezen terwijl de computer ze bijwerkt. We zouden de waarden daarom eigenlijk moeten ontdenderen door de tijd een paar keer achter elkaar te lezen en, indien de waarden niet beide keren gelijk zijn, het proces te herhalen totdat dat wel het geval is. Dit laten wij over als een oefening voor de lezer.

OPMERKING: Deze registers zijn nog niet volledig gedocumenteerd.

MEGAPHONE-HANDHELD, REVISIES 1 EN 2

De MEGAphone-revisies 1 en 2 bevatten weliswaar een real-time klok (RTC), maar deze RTC heeft geen gebied met niet-vluchtig geheugen (NVRAM). Tot de overige specifieke kenmerken van de MEGAphone-revisies 1 en 2 behoren een drie-assige accelerometer met analoog-naar-digitaalomzetters (ADC's), een besturingseenheid voor de luidsprekerversterker en diverse I2C-I/O-expanders, die worden gebruikt om de joypad en andere randapparaten aan te sluiten. De I/O-expanders vormen een volledig geïntegreerd onderdeel van het MEGAphone-ontwerp, zodat u deze registers normaal gesproken niet rechtstreeks hoeft uit te lezen. De I/O-expanders verzorgen echter ook de voedingsregeling van de verschillende subsystemen van de MEGAphone.

OPMERKING: Deze registers zijn nog niet volledig gedocumenteerd.

NEXYS4 DDR FPGA-BORD

OPMERKING: Deze registers zijn nog niet volledig gedocumenteerd.

BIJLAGE AE

Elektronisch schema

- MEGA65 R6 elektronisch schema
- MEGA65 R3 elektronisch schema
- MEGA65 R2 elektronisch schema
- Nexys Widget Board elektronisch schema

MEGA65 R6 ELEKTRONISCH SCHEMA

U_B13
B13.SchDoc

U_B14
B14.SchDoc

U_B15
B15.SchDoc

U_B16
B16.SchDoc

U_B34
B34.SchDoc

U_FPGA-CFG
FPGA-CFG.SchDoc

U_FPGA-MGT
FPGA-MGT.SchDoc

U_FPGA-PWR
FPGA-PWR.SchDoc

U_PowerMain
PowerMain.SchDoc

U_POWER
POWER.SchDoc

U_HDMI
HDMI.SchDoc

U_VGA
VGA.SchDoc

U_Ethernet
Ethernet.SchDoc

U_SOUND
SOUND.SchDoc

U_KEYBOARD
KEYBOARD.SchDoc

U_Floppy
Floppy.SchDoc

U_JOY
JOY.SchDoc

U_EXT_HEADER
EXT_HEADER.SchDoc

U_EXP_Slot
EXP_Slot.SchDoc

U_LED_SW_BUT
LED_SW_BUT.SchDoc

PM1 FIDU-DOT - small
PM2 FIDU-DOT - small
PM3 FIDU-DOT - small
PM4 FIDU-DOT - small
PM5 FIDU-DOT - small
PM6 FIDU-DOT - small

Address 0x40 I/O Expander
Address 0x61 DC/DC Converter
Address 0x67 DC/DC Converter

KK1
ATS-CPX025025015-138-C2-R0
R234 0R
R235 0R
GND GND

Serial1
Serial
Serialnumber 6,3 x 6.3mm

LOGO1
TE Logo PRINT Layer
LOGO PRINT

CE
CE Logo on Top Overlay
CE-TOPOVERLAY

Shielding

DDR
SHIELD4
BMI-S-202-C

FPGA
SHIELD5
BMI-S-230-C

DCDC
SHIELD6
BMI-S-205-C

Shield1
GND
BMI-S-202-F

Shield2
GND
BMI-S-230-F-R

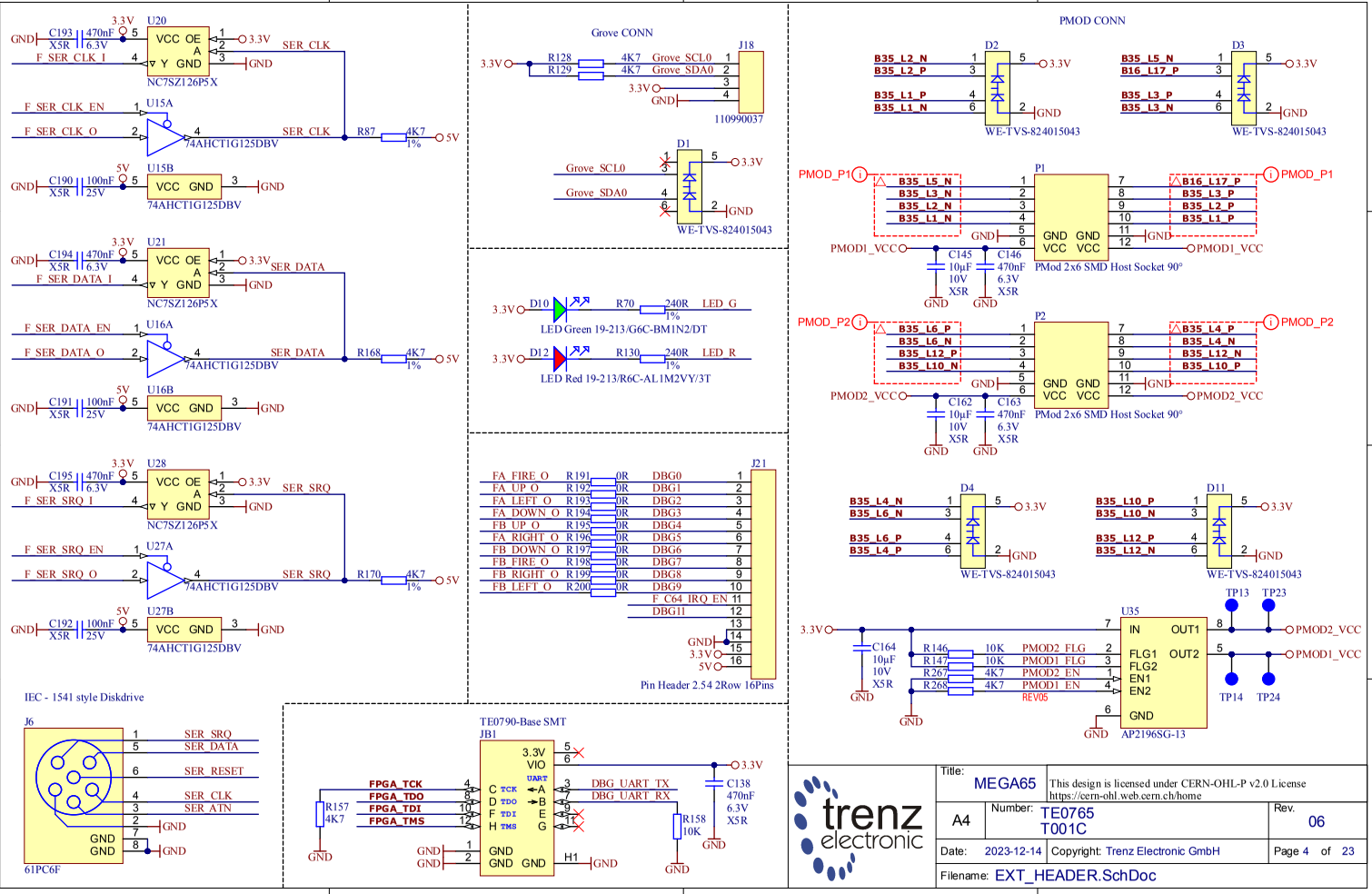
Shield3
GND
GND
BMI-S-205-F

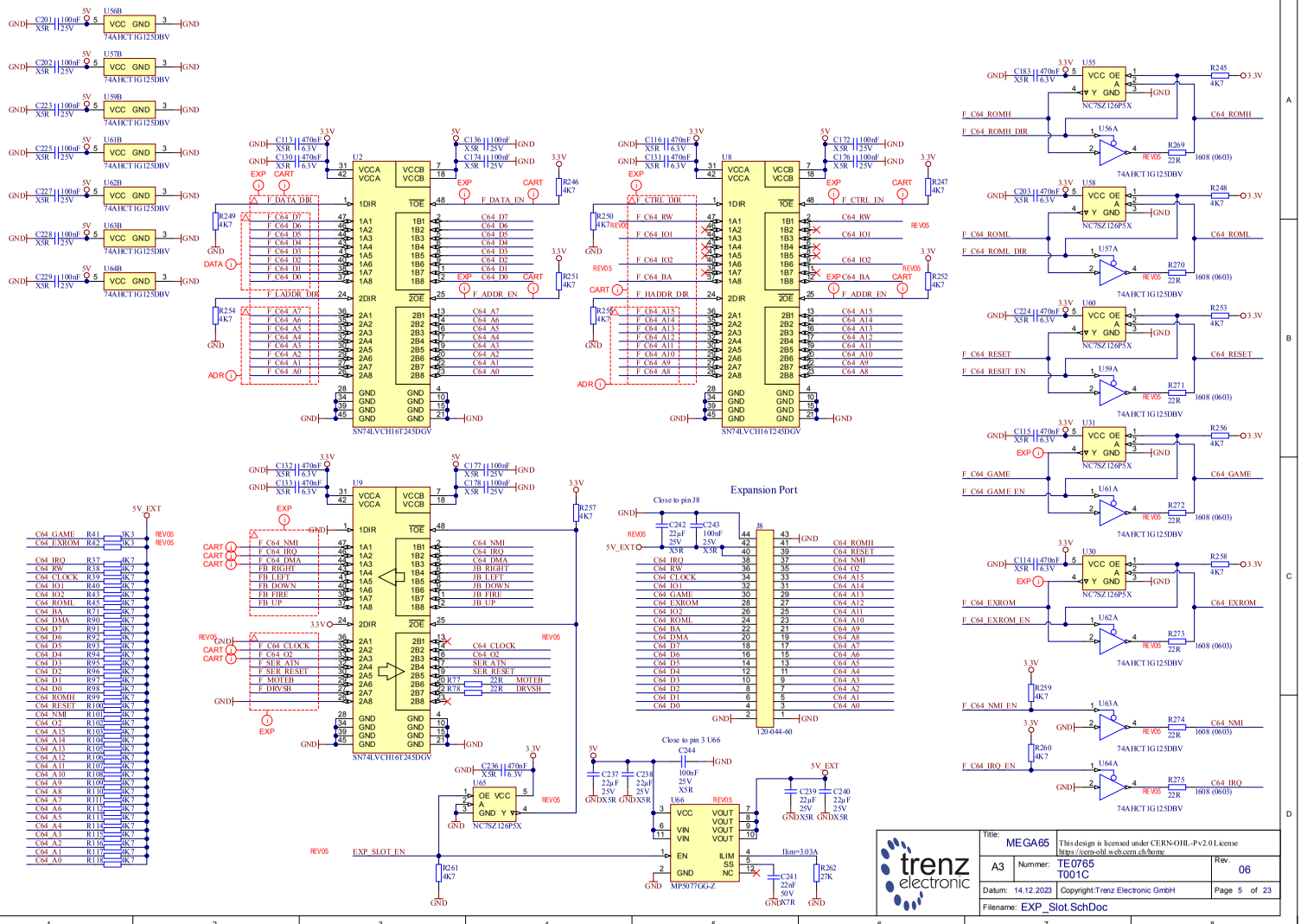
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm

Design drawn by:	VY
Checked by:	IG
Assembly variant:	T001C
Created by:	VY
Modified by:	VY
Modified at:	2023-07-14

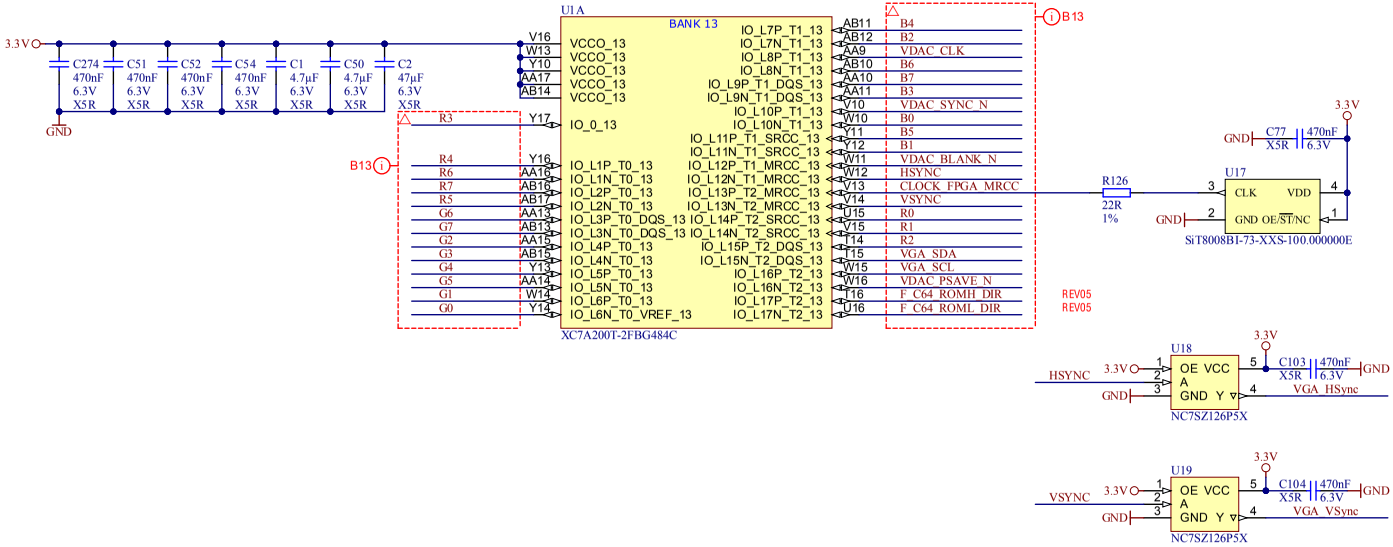
trenz electronic

Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
A4	Number: TE0765 T001C	Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 3 of 23
Filename: TE0765.SchDoc		


Grove CONN
PMOD CONN
IEC - 1541 style Diskdrive
Pin Header 2.54 2Row 16Pins
LED Green 19-213/G6C-BM1N2/DT
LED Red 19-213/R6C-AL1M2VY/3T
PMod 2x6 SMD Host Socket 90°
WE-TVS-824015043
NC7SZ126P5X
74AHCT1G125DBV
110990037
61PC6F
AP2196SG-13
TE0790-Base SMT
MEGA65
trenz electronic
This design is licensed under CERN-OHL-P v2.0 License
https://cern-ohl.web.cern.ch/home
TE0765
T001C
06
2023-12-14
Trenz Electronic GmbH
Page 4 of 23
EXT_HEADER.SchDoc

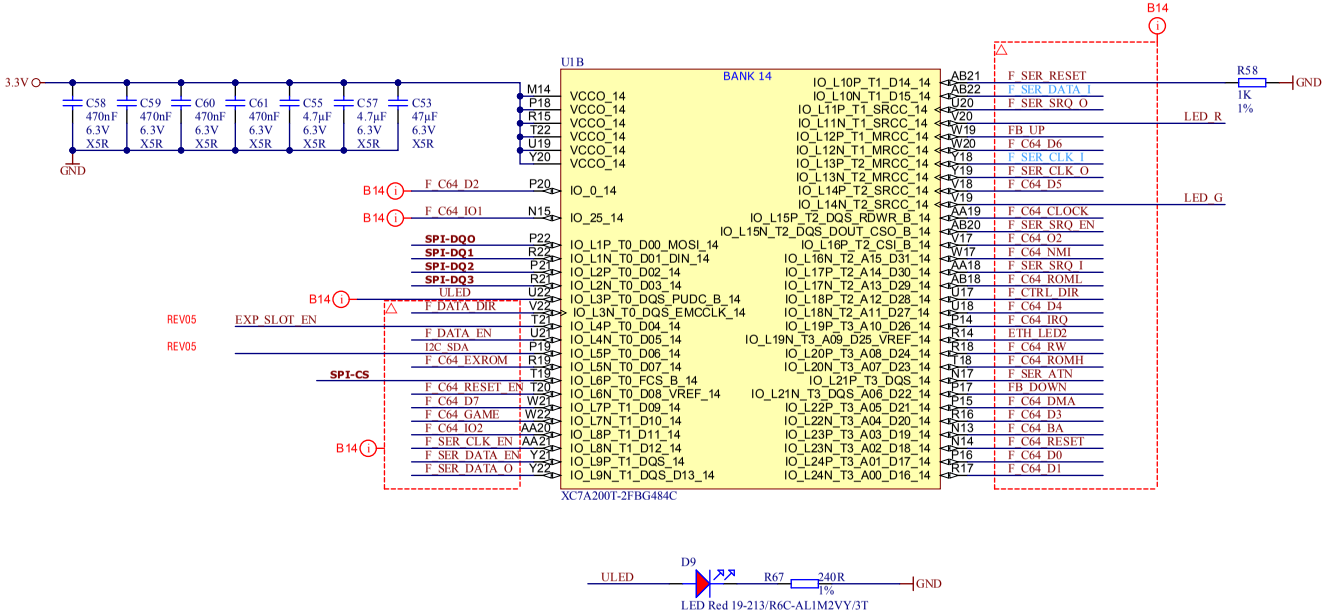


Title:	MEGA65	This design is licensed under CERN-OHL-Pv2.0 License https://cern-ohl.web.cern.ch/home	
A3	Nummer: TE0765 T001C		Rev: 06
Datum: 14.12.2023	Copyright:Trenz Electronic GmbH		Page 5 of 23
Filename: EXP_Slot.SchDoc			

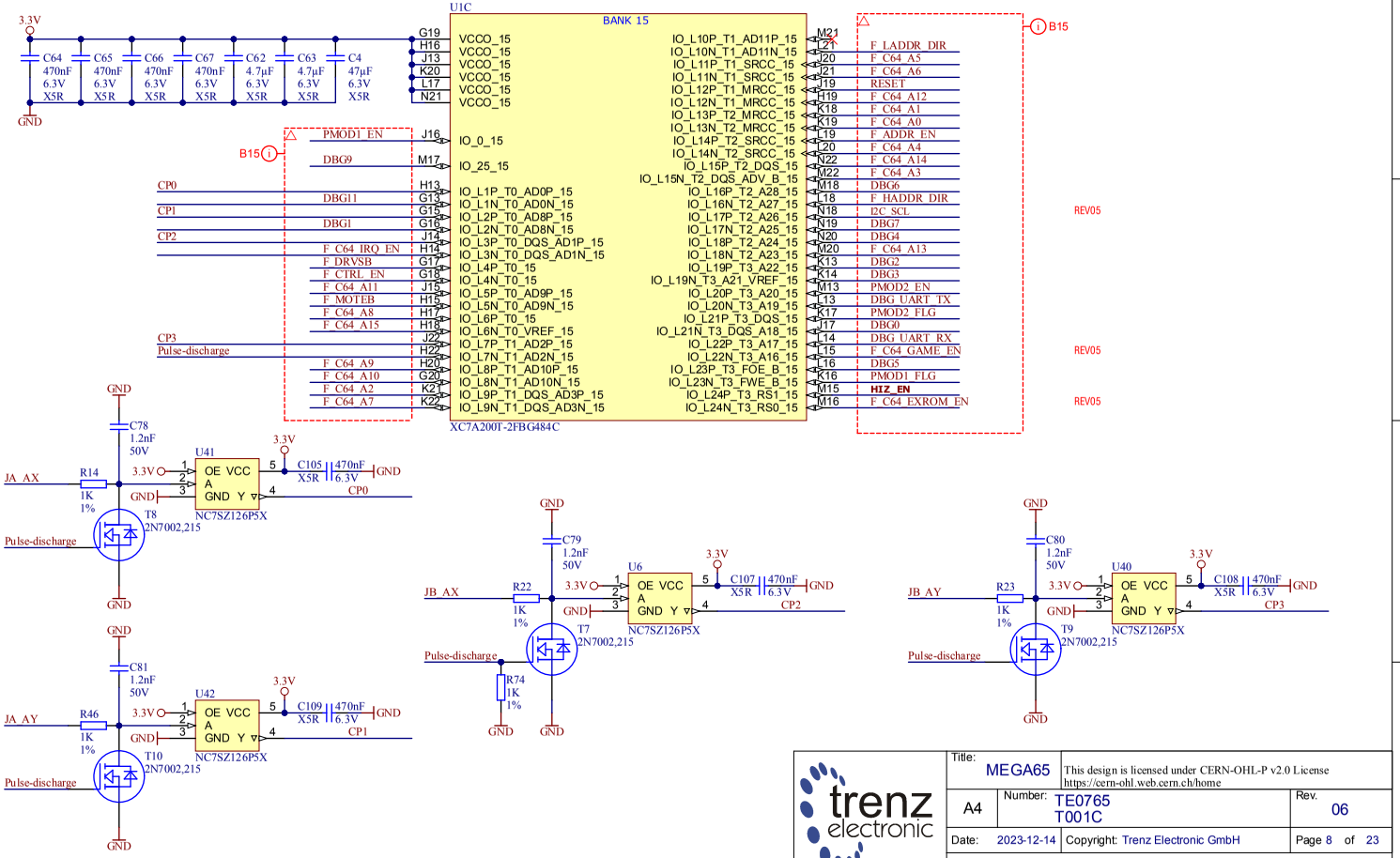


trenz electronic

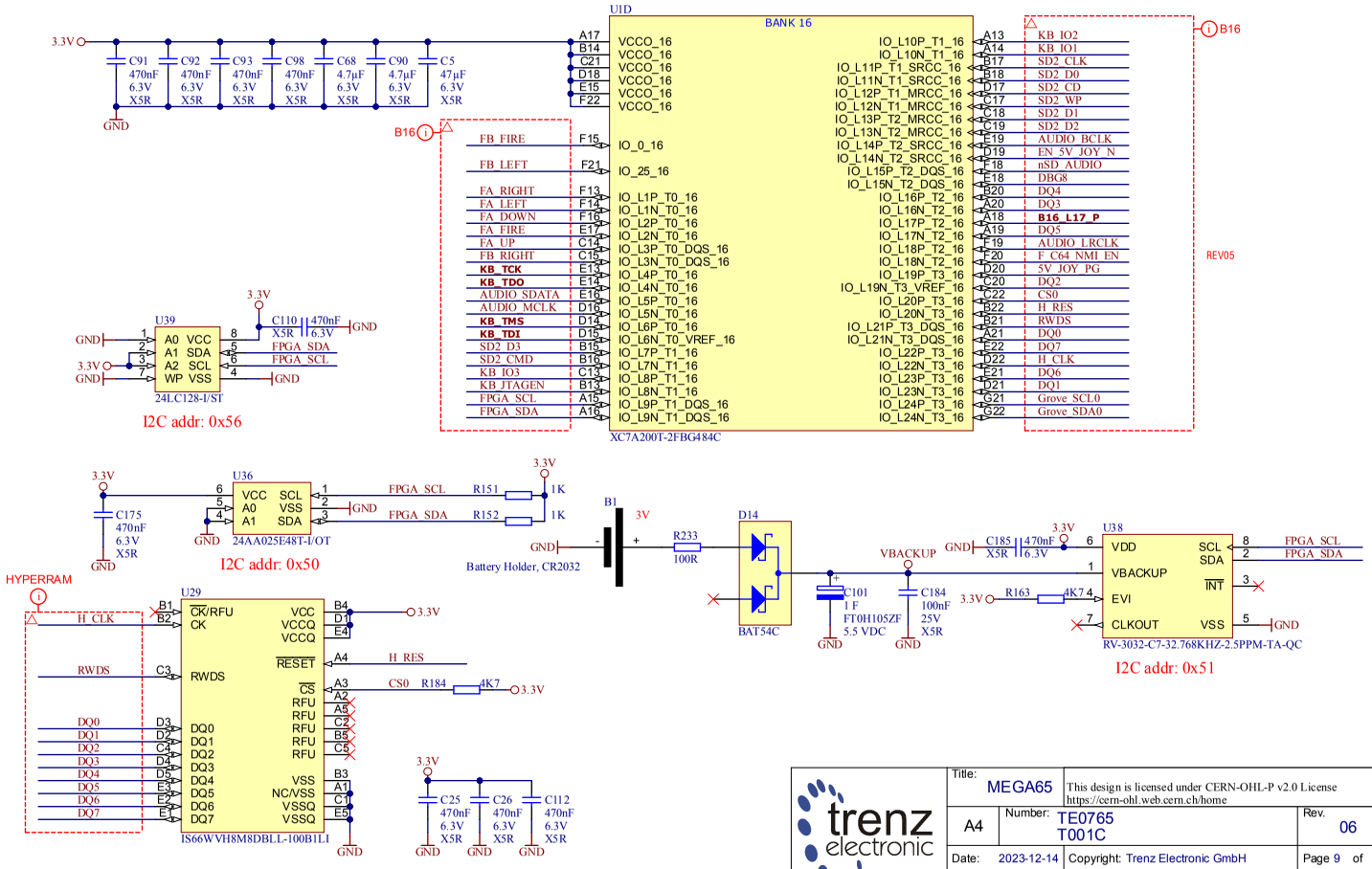
Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home		
A4	Number: TE0765 T001C		Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH		Page 6 of 23
Filename: B13.SchDoc			



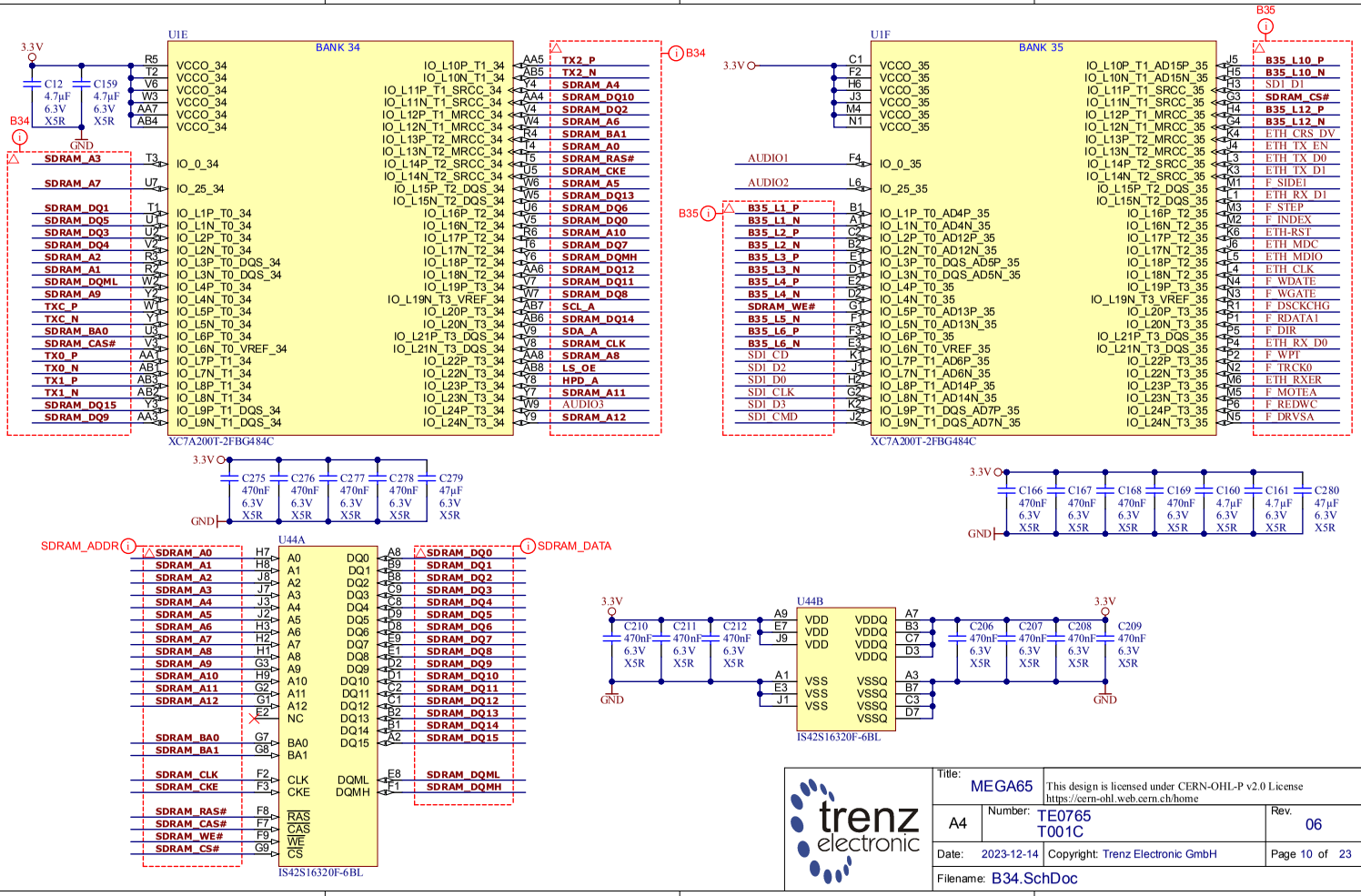
trenz electronic	Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
	A4	Number: TE0765 T001C	Rev. 06
	Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 7 of 23
	Filename: B14.SchDoc		

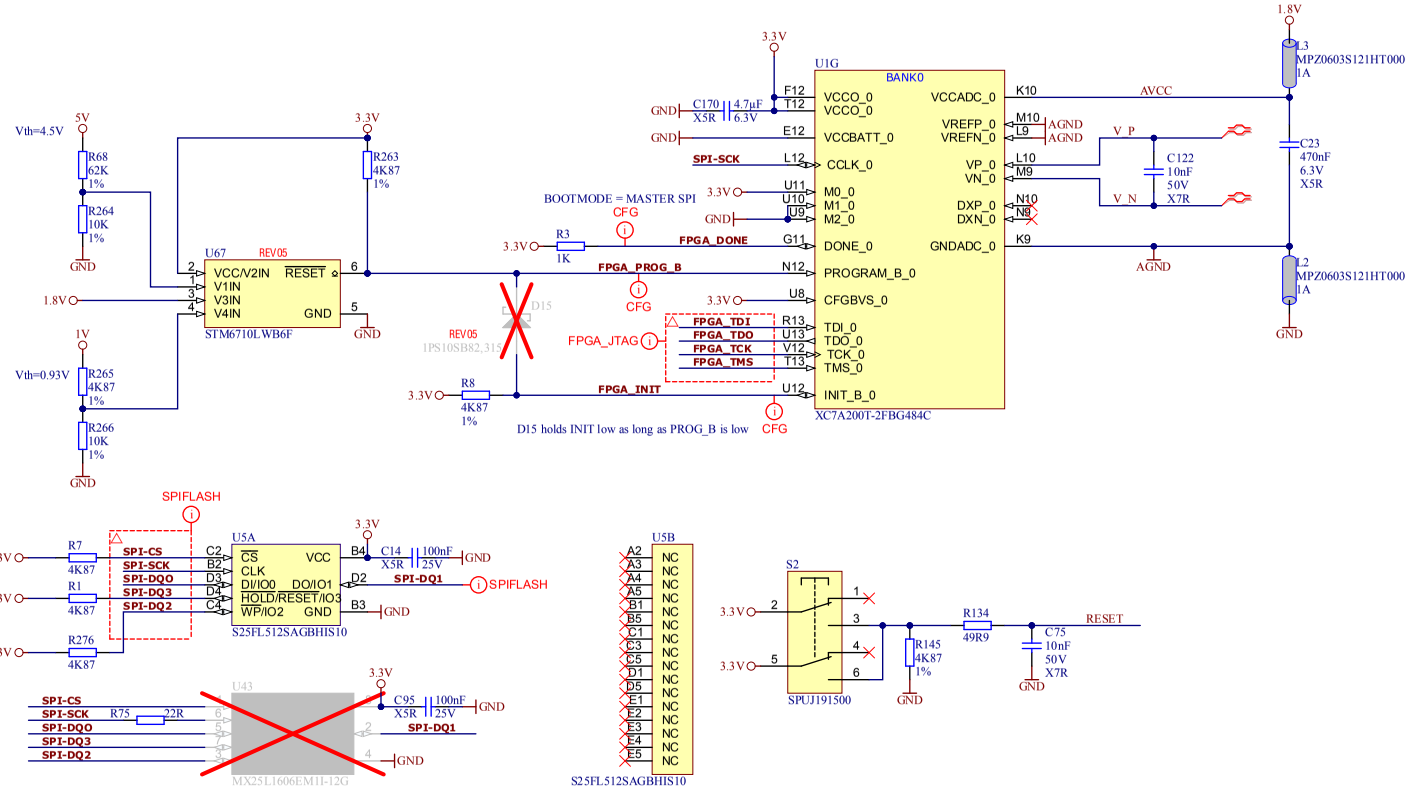


Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home		
A4	Number: TE0765 T001C		Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH		Page 8 of 23
Filename: B15.SchDoc			

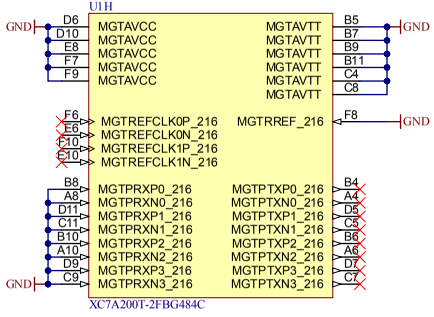


Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home		
A4	Number: TE0765 T001C		Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH		Page 9 of 23
Filename: B16.SchDoc			


U1E
BANK 34
XC7A200T-2FBG484C
U1F
BANK 35
XC7A200T-2FBG484C
B34
B35
3.3V
GND
U44A
IS42S16320F-6BL
U44B
IS42S16320F-6BL
SDRAM_ADDR
SDRAM_DATA
trenz electronic
MEGA65
This design is licensed under CERN-OHL-P v2.0 License
https://cern-ohl.web.cern.ch/home
A4
TE0765
T001C
06
2023-12-14
Copyright: Trenz Electronic GmbH
Page 10 of 23
B34.SchDoc

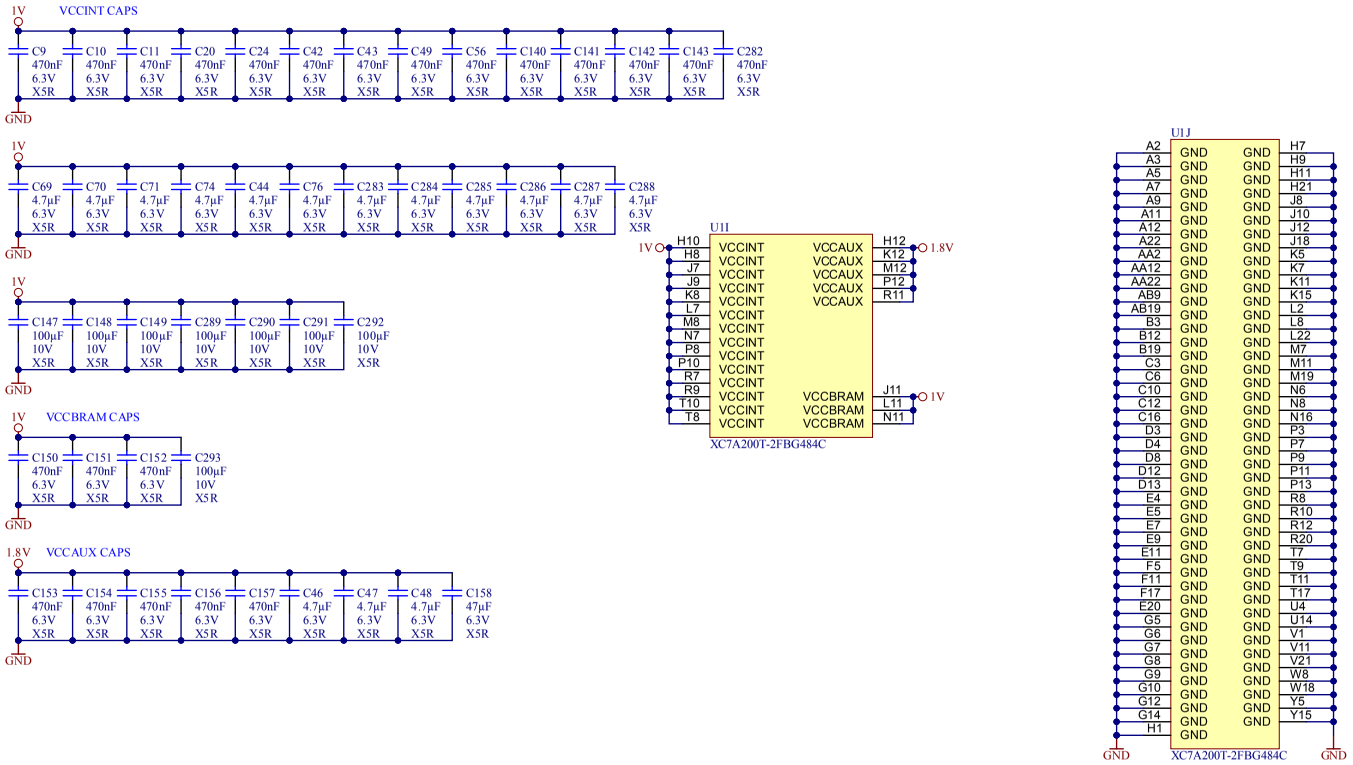


trenz electronic	Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
	A4	Number: TE0765 T001C	Rev. 06
	Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 11 of 23
	Filename: FPGA-CFG.SchDoc		



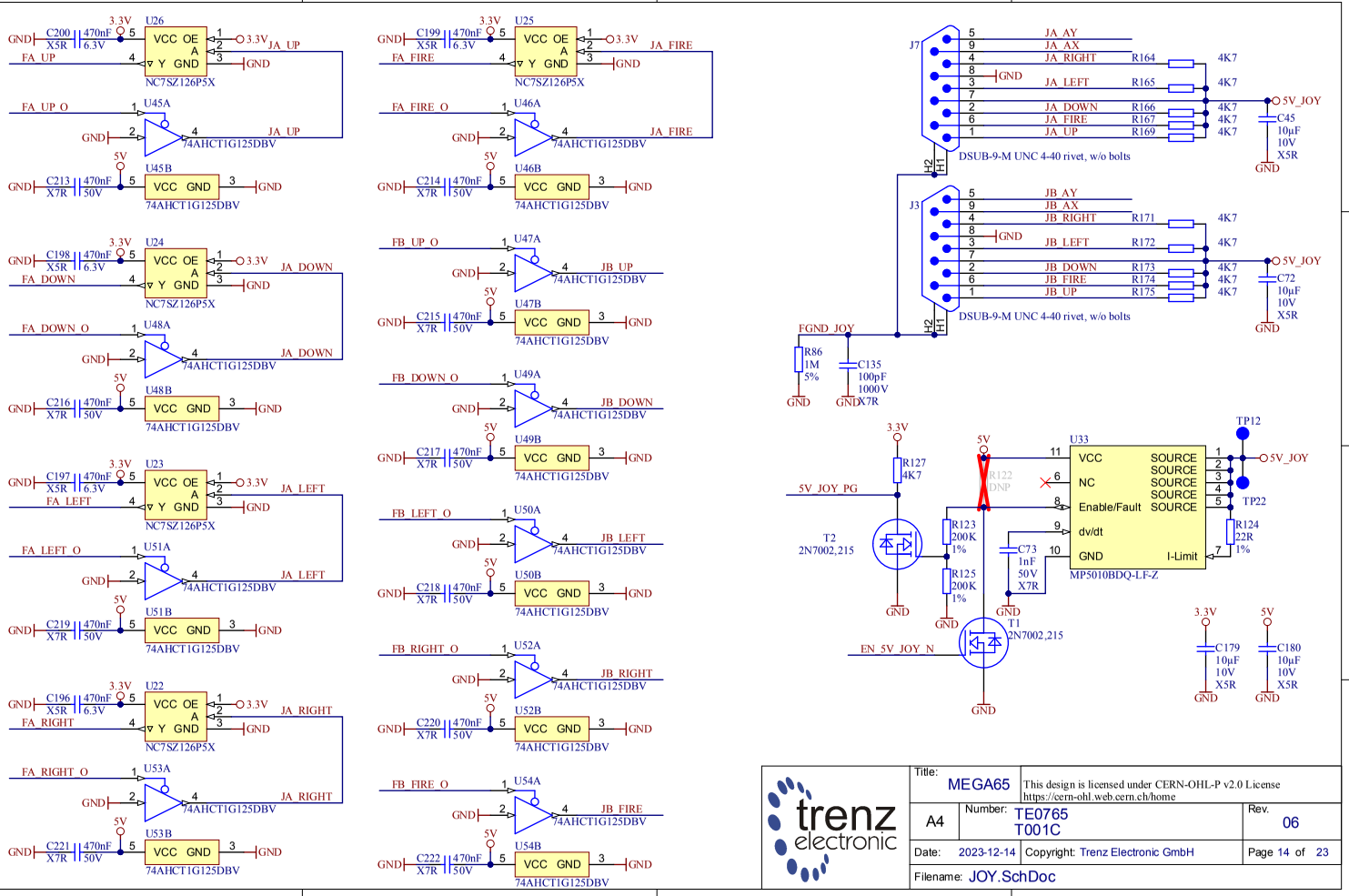
trenz electronic

Title: MEGA65		This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
A4	Number: TE0765 T001C		Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH		Page 12 of 23
Filename: FPGA-MGT.SchDoc			



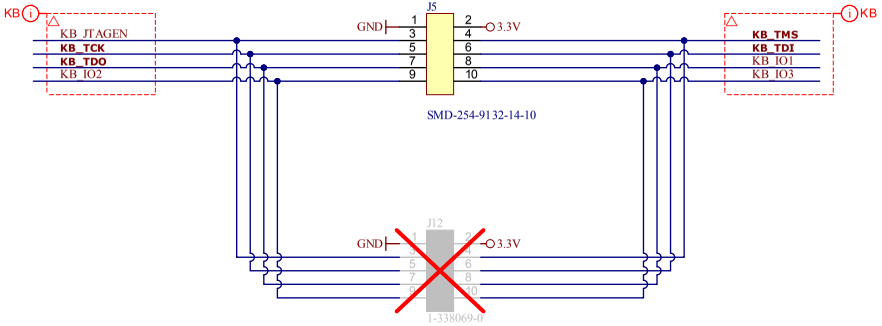
trenz electronic

Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
A4	Number: TE0765 T001C	Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 13 of 23
Filename: FPGA-PWR.SchDoc		

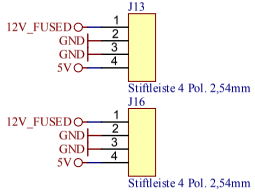


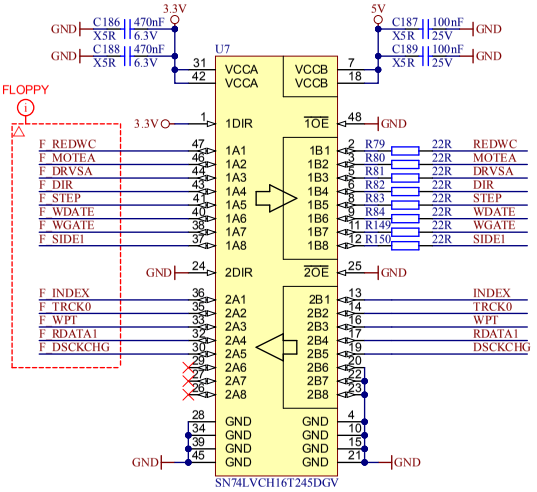
trenz electronic

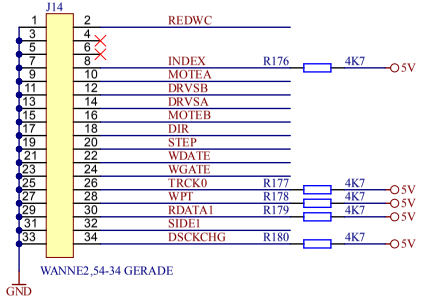
Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
A4	Number: TE0765 T001C	Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 14 of 23
Filename: JOY.SchDoc		



Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home		
A4	Number: TE0765 T001C		Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH		Page 15 of 23
Filename: KEYBOARD.SchDoc			

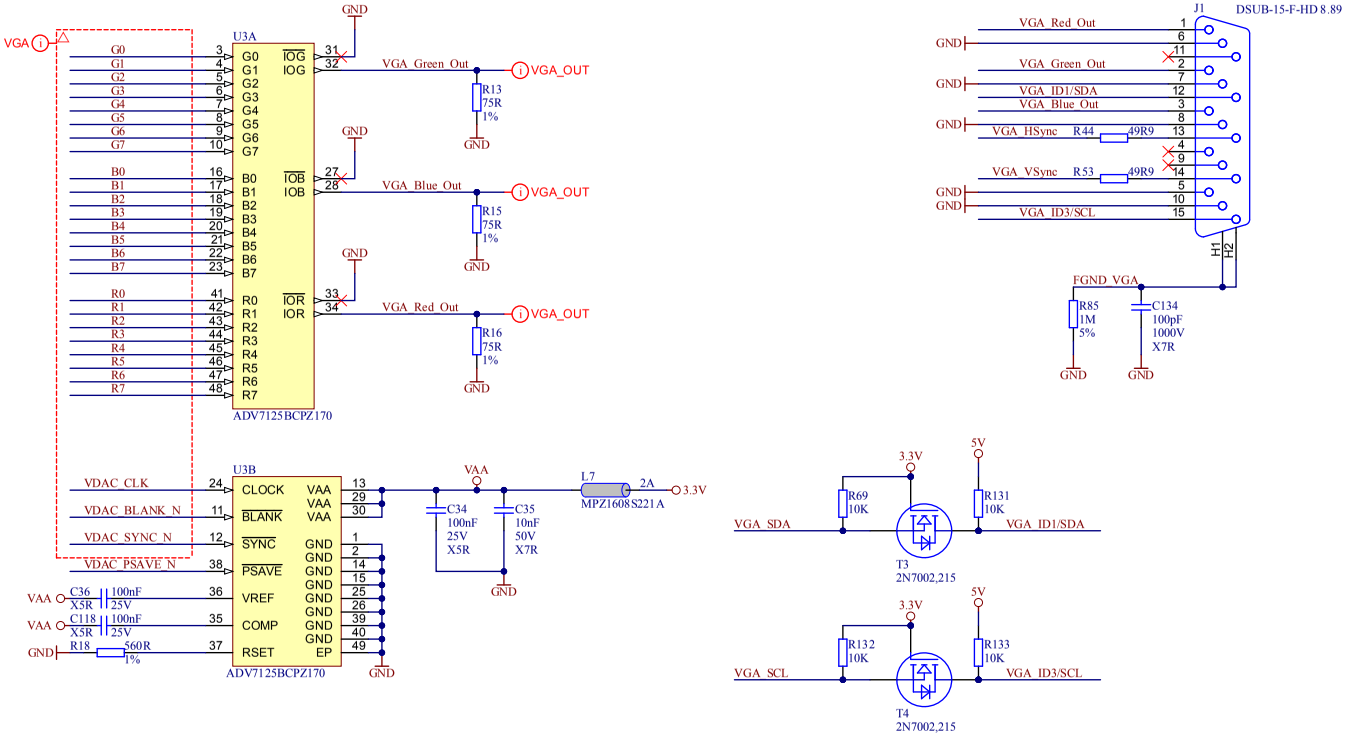




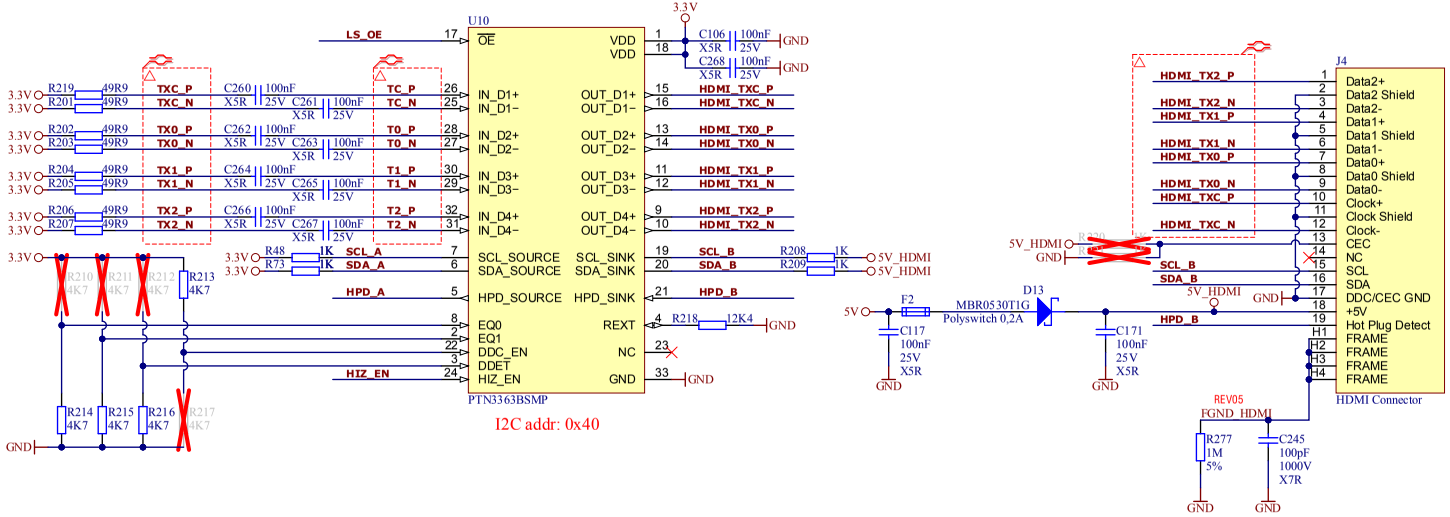


trenz electronic

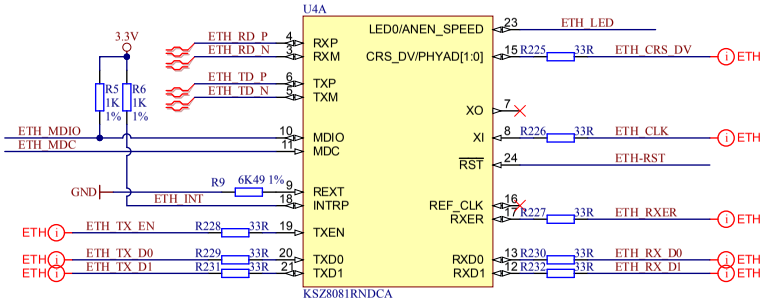
Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home		
A4	Number: TE0765 T001C		Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH		Page 16 of 23
Filename: Floppy.SchDoc			

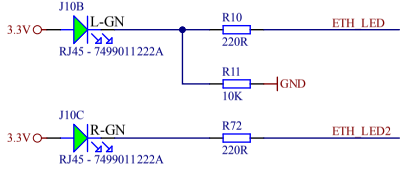


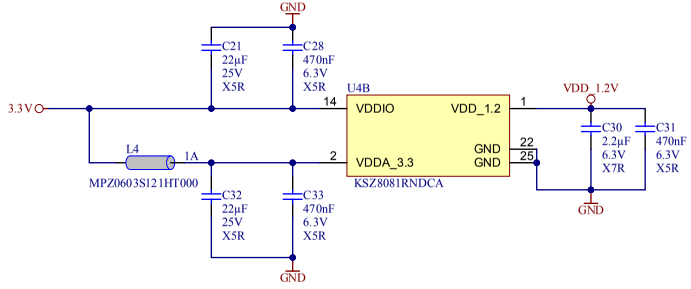
trenz electronic	Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
	A4	Number: TE0765 T001C	Rev. 06
	Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 17 of 23
	Filename: VGA.SchDoc		

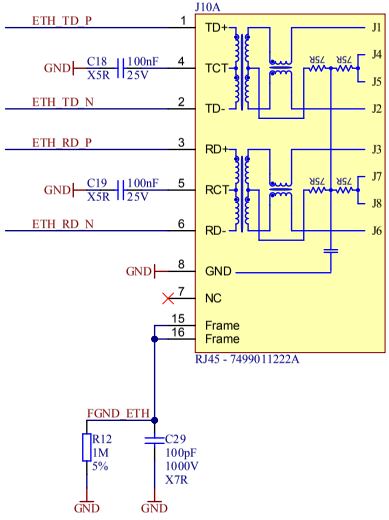


trenz electronic	Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
	A4	Number: TE0765 T001C	Rev. 06
	Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 18 of 23
	Filename: HDMI.SchDoc		



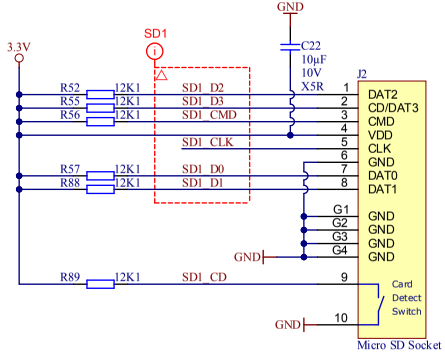


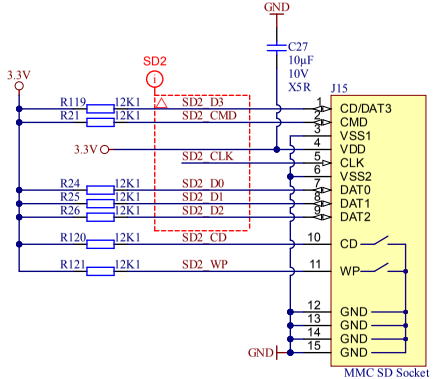


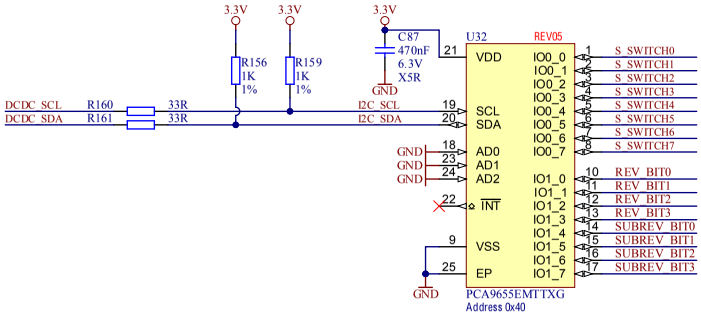


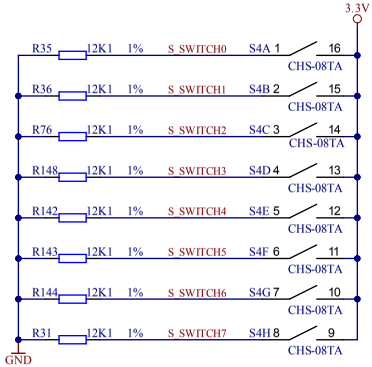
trenz electronic

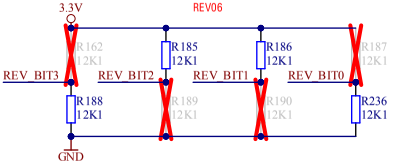
Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home		
A4	Number: TE0765 T001C		Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH		Page 19 of 23
Filename: Ethernet.SchDoc			



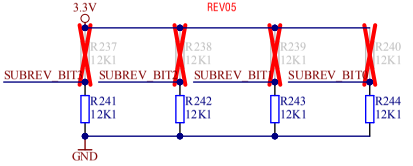








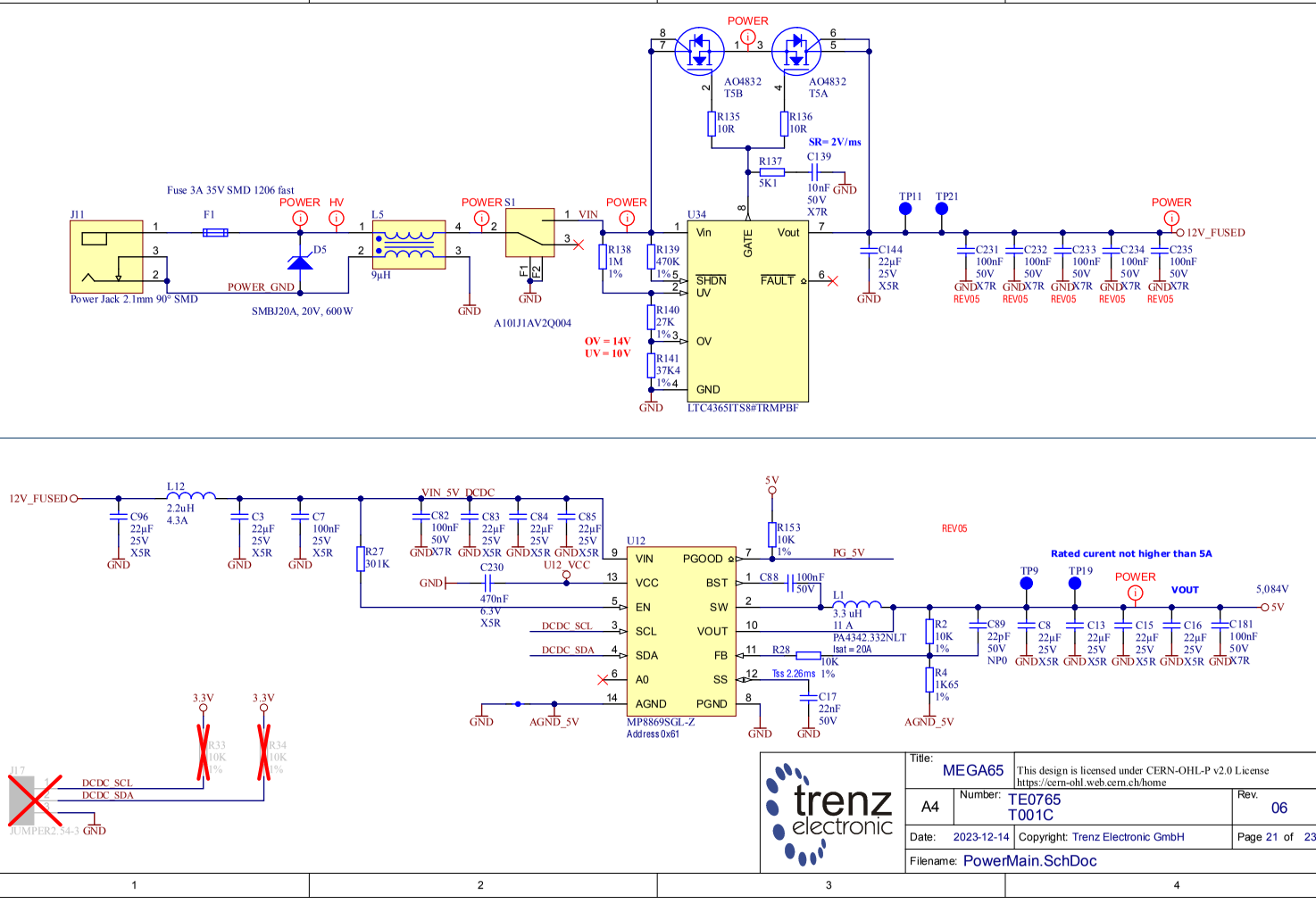
Revision	REV_BIT3	REV_BIT2	REV_BIT1	REV_BIT0
05	0	1	0	1
06	0	1	1	0



SUBREV	SUBREV_BIT3	SUBREV_BIT2	SUBREV_BIT1	SUBREV_BIT0
00	0	0	0	0

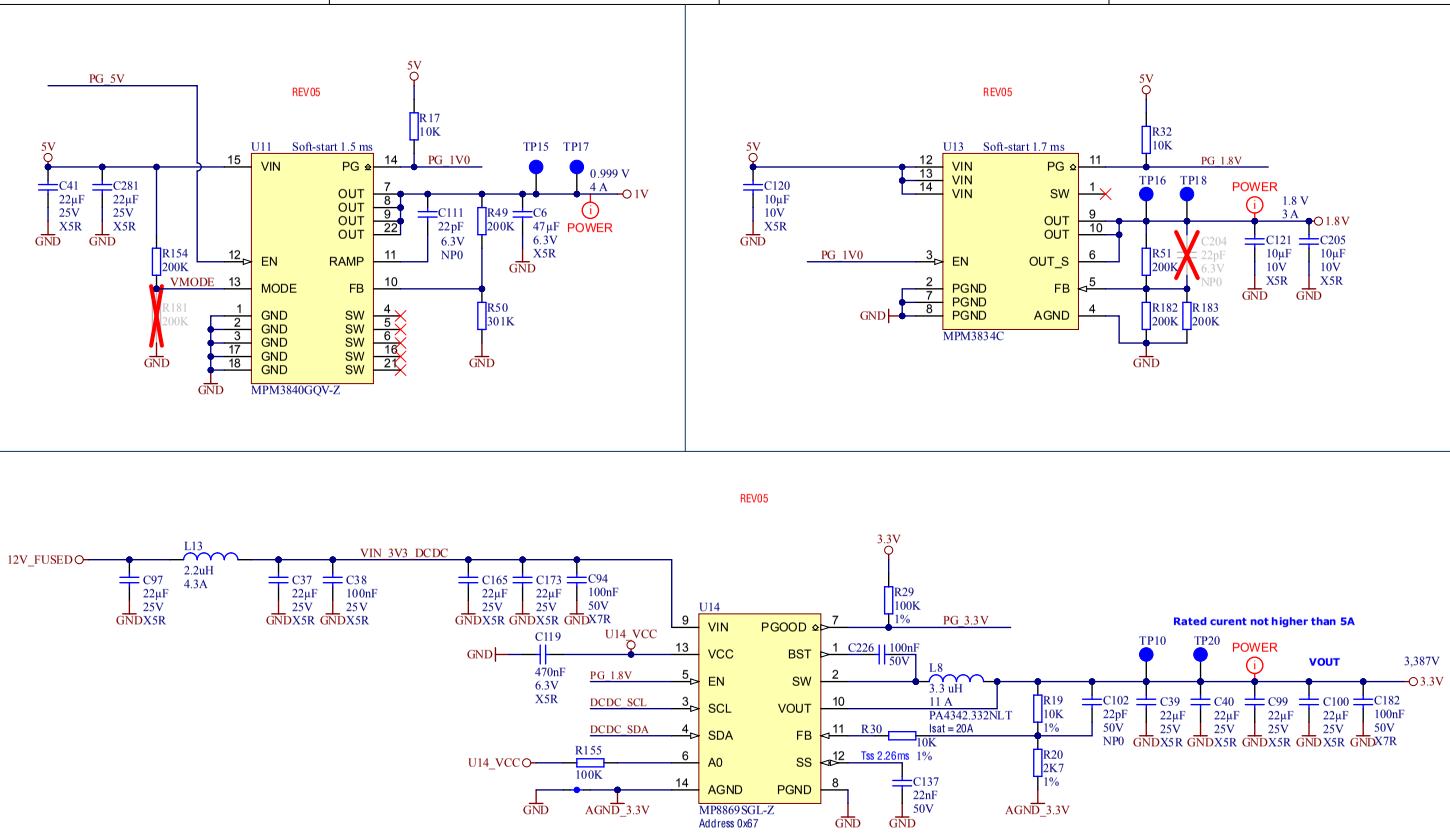
trenz electronic

Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
A4	Number: TE0765 T001C	Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 20 of 23
Filename: LED_SW_BUT.SchDoc		



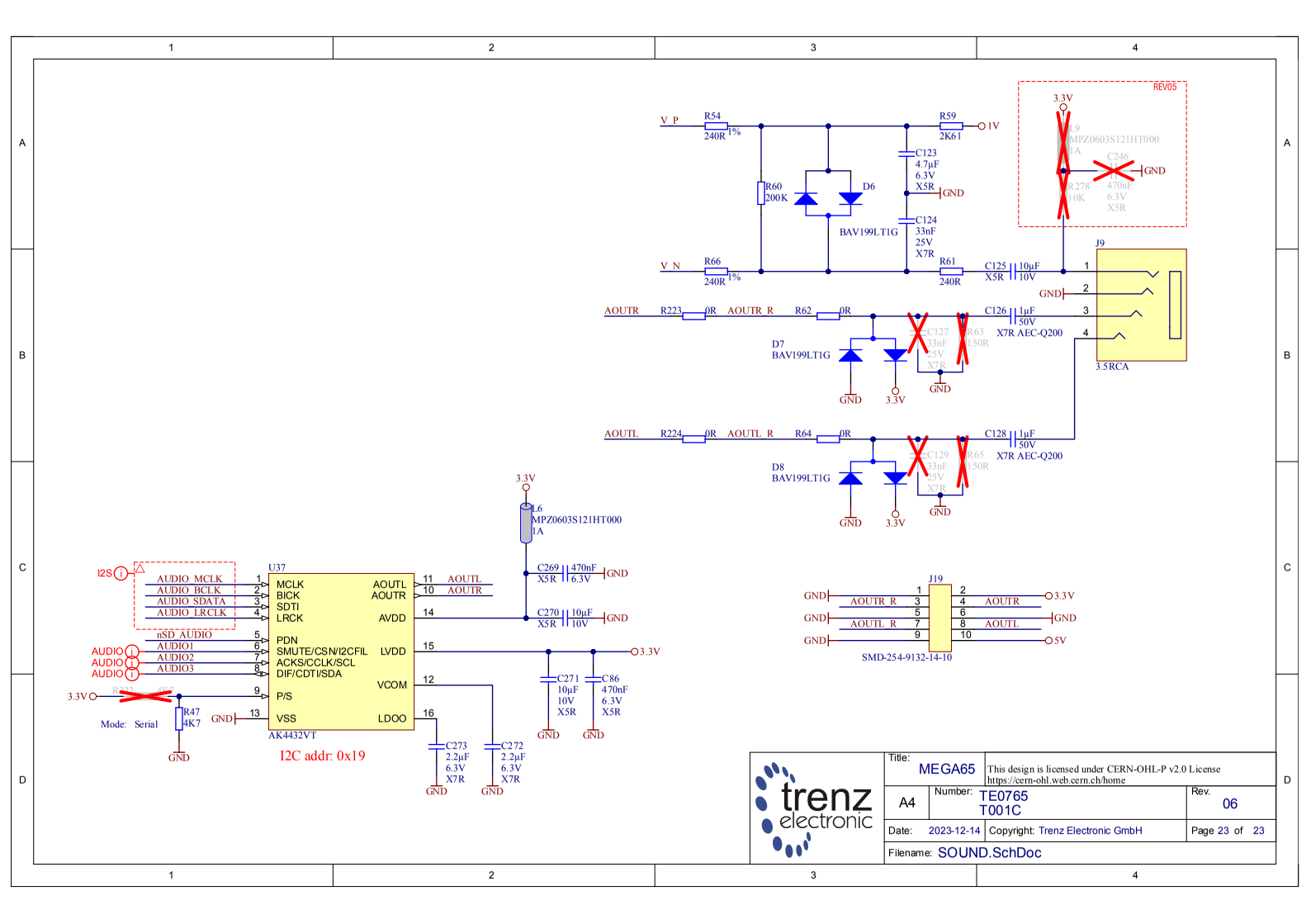
trenz electronic

Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
A4	Number: TE0765 T001C	Rev. 06
Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 21 of 23
Filename: PowerMain.SchDoc		

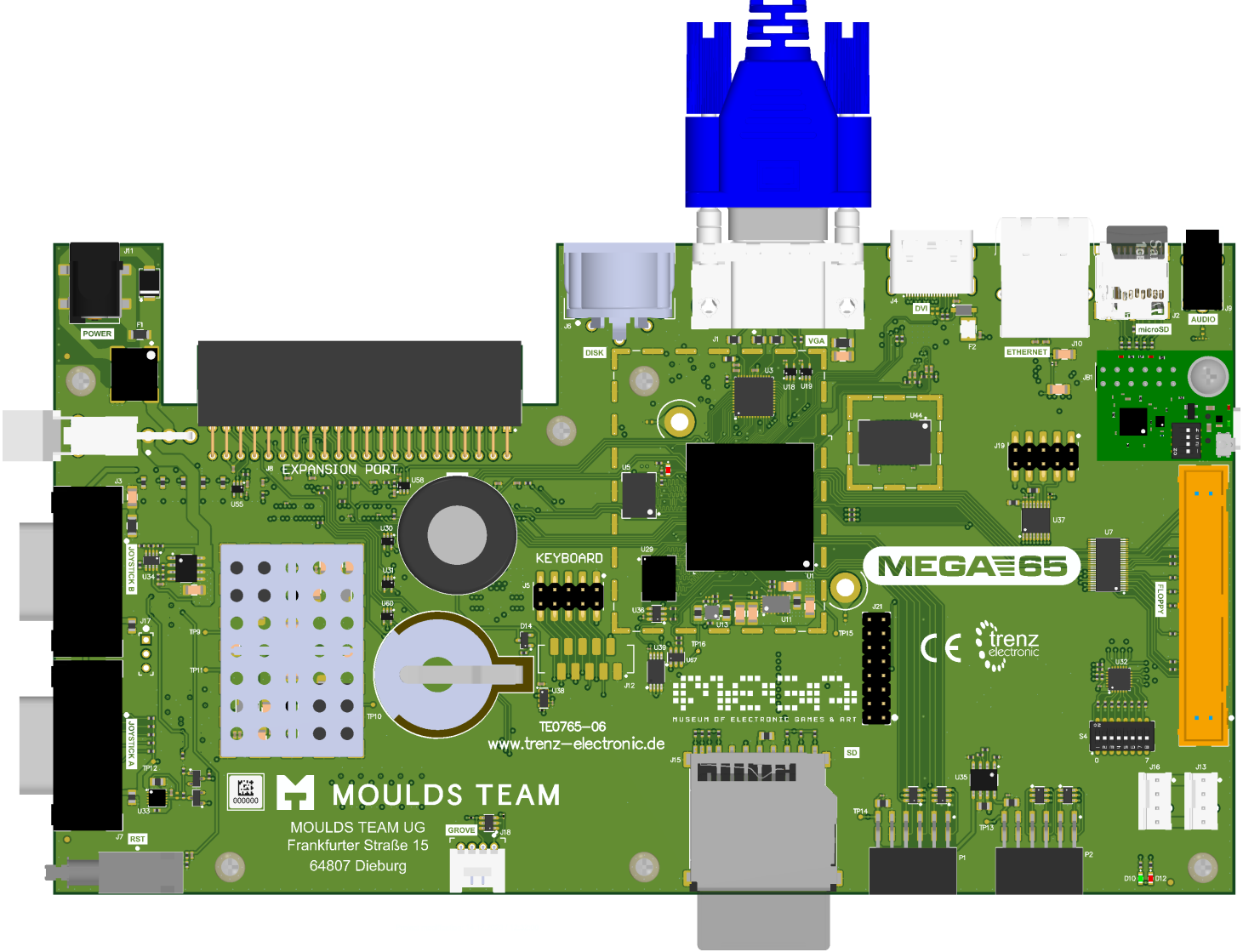


trenz electronic

Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home		
A4	Number: TE0765 T001C	Rev. 06	
Date: 2023-12-14	Copyright: Trenz Electronic GmbH / TT	Page 22 of 23	
Filename: POWER.SchDoc			



trenz electronic	Title: MEGA65	This design is licensed under CERN-OHL-P v2.0 License https://cern-ohl.web.cern.ch/home	
	A4	Number: TE0765 T001C	Rev. 06
	Date: 2023-12-14	Copyright: Trenz Electronic GmbH	Page 23 of 23
	Filename: SOUND.SchDoc		


POWER
DISK
VGA
DVI
ETHERNET
microSD
AUDIO
EXPANSION PORT
KEYBOARD
JOYSTICK B
JOYSTICK A
RST
GROVE
FLOPPY
SD
MEGA65
TE0765-06
www.trenz-electronic.de
MOULDS TEAM
MOULDS TEAM UG
Frankfurter Straße 15
64807 Dieburg
trenz electronic
MUSEUM OF ELECTRONIC GAMES & ART

MEGA65 R3 ELEKTRONISCH SCHEMA

U_HDMI
HDMI.SchDoc

U_Ethernet
Ethernet.SchDoc

U_LED_SW_BUT
LED_SW_BUT.SchDoc

U_SYS_MAX10_CTRL
SYS_MAX10_CTRL.SchDoc

U_SOUND
SOUND.SchDoc

U_B13
B13.SchDoc

U_B14
B14.SchDoc

U_B15
B15.SchDoc

U_B16
B16.SchDoc

U_B34
B34.SchDoc

U_FPGA-CFG
FPGA-CFG.SchDoc

U_FPGA-MGT
FPGA-MGT.SchDoc

U_FPGA-PWR
FPGA-PWR.SchDoc

U_JOY
JOY.SchDoc

U_KEYBOARD
KEYBOARD.SchDoc

U_POWER
POWER.SchDoc

U_PowerMain
PowerMain.SchDoc

U_VGA
VGA.SchDoc

U_EXT_HEADER
EXT_HEADER.SchDoc

U_EXP_Slot
EXP_Slot.SchDoc

U_Floppy
Floppy.SchDoc

PM1 FIDU-DOT - small
PM2 FIDU-DOT - small
PM3 FIDU-DOT - small
PM4 FIDU-DOT - small
PM5 FIDU-DOT - small
PM6 FIDU-DOT - small

Serial1
Serial
Serialnumber 6,3 x 6.3mm

LOGO1
TE Logo PRINT Layer
LOGO PRINT

GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm

Design drawn by:	IG
Checked by:	MR
Assembly variant:	[No Variations]
Created by:	VariantCreatedBy
Modified by:	VariantModifiedBy
Modified at:	VariantDateModification

trenz electronic

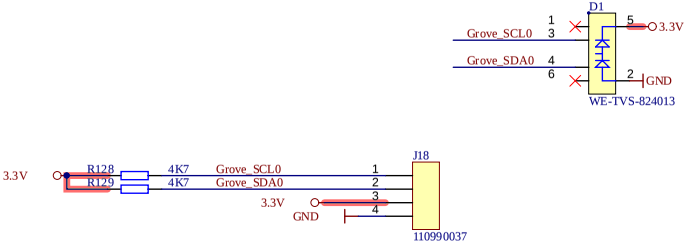
Title: MEGA65

A4 | Number: TE0765 [No Variations] | Rev. 03

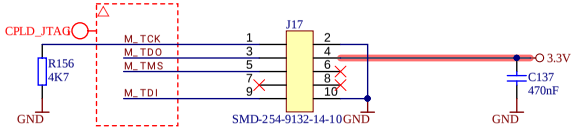
Date: 2020-06-29 | Copyright: Trenz Electronic GmbH | Page 1 of 23

Filename: TE0765.SchDoc

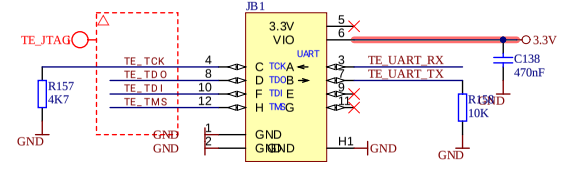
Grove CONN



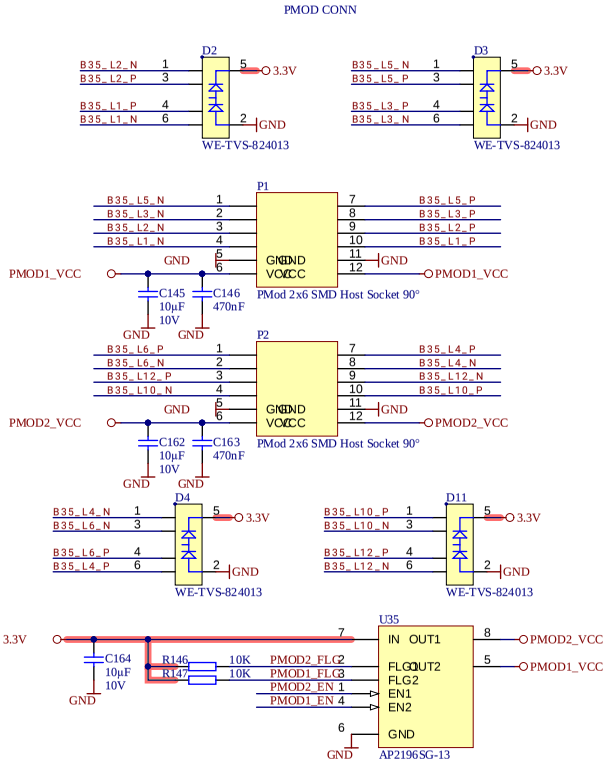
CPLD CONFIG



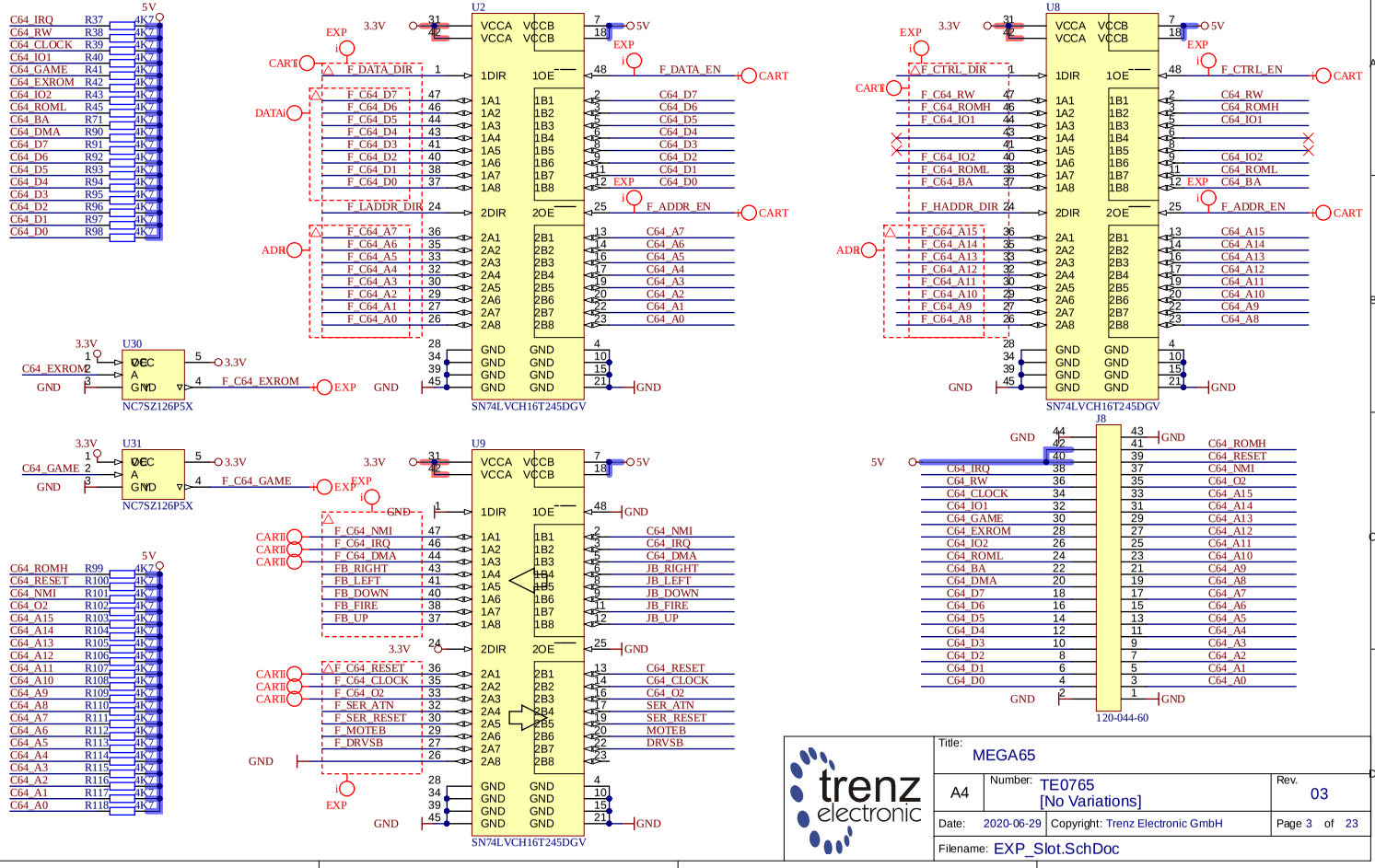
TE0790-Base SMT

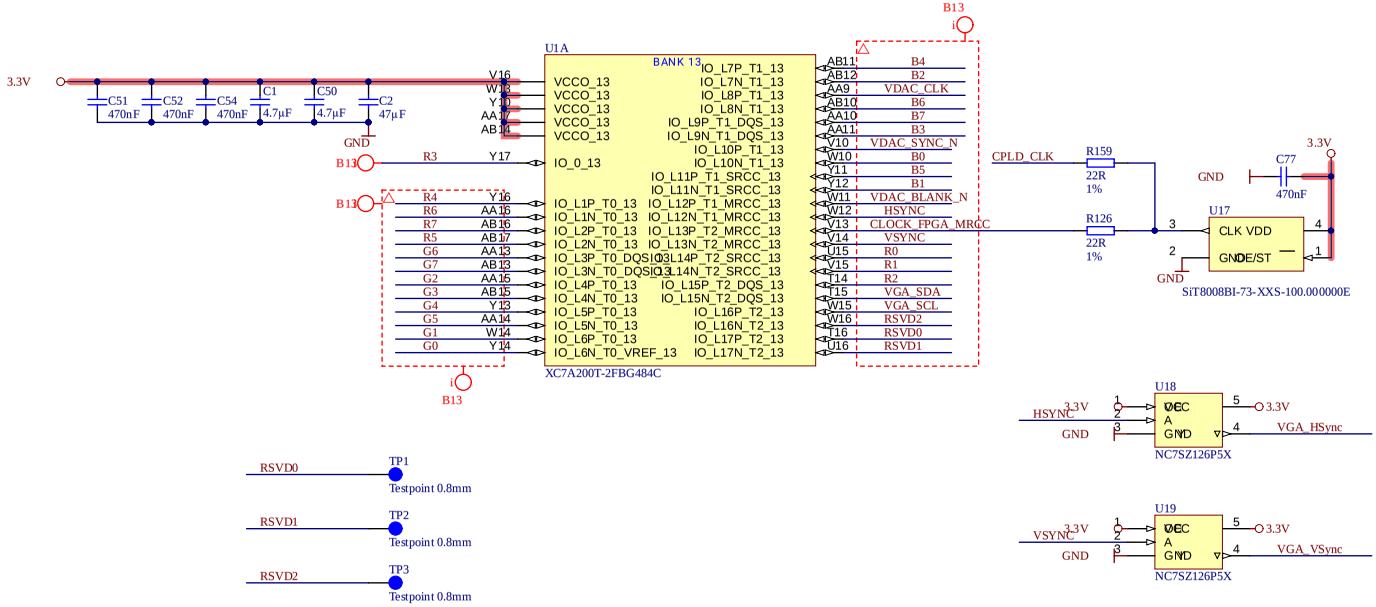


PMOD CONN

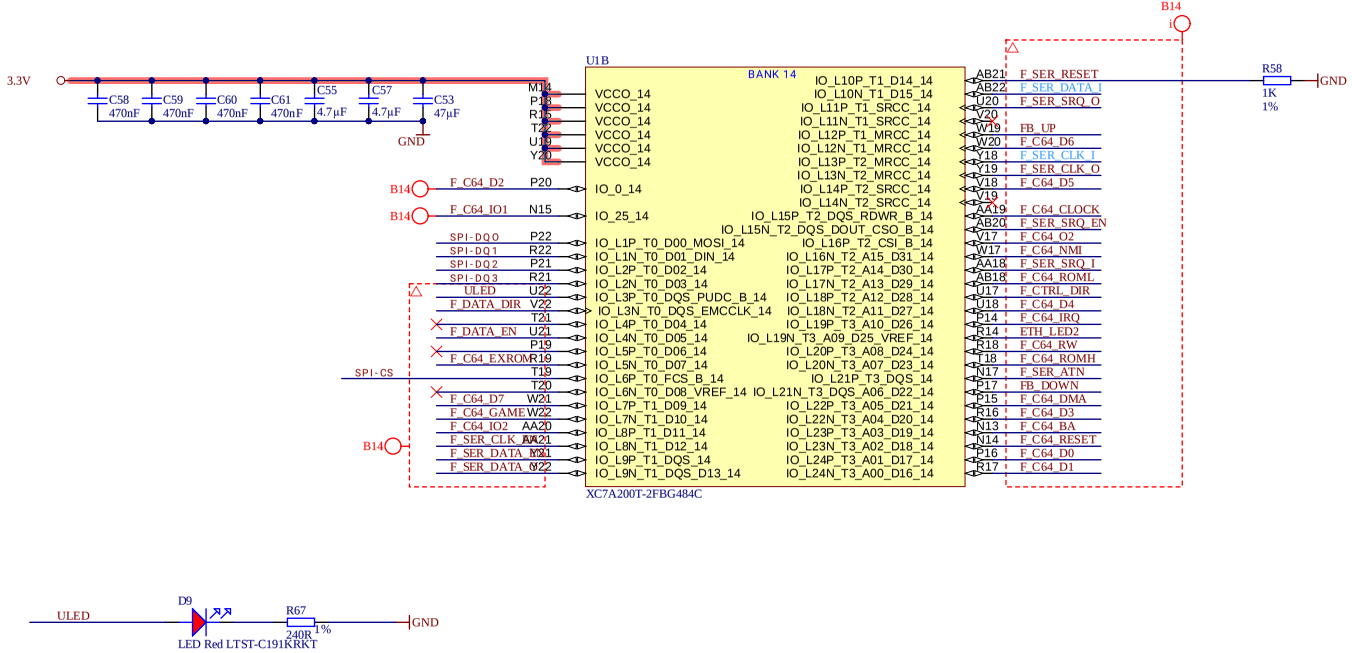


Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 2 of 23
Filename: EXT_HEADER.SchDoc			

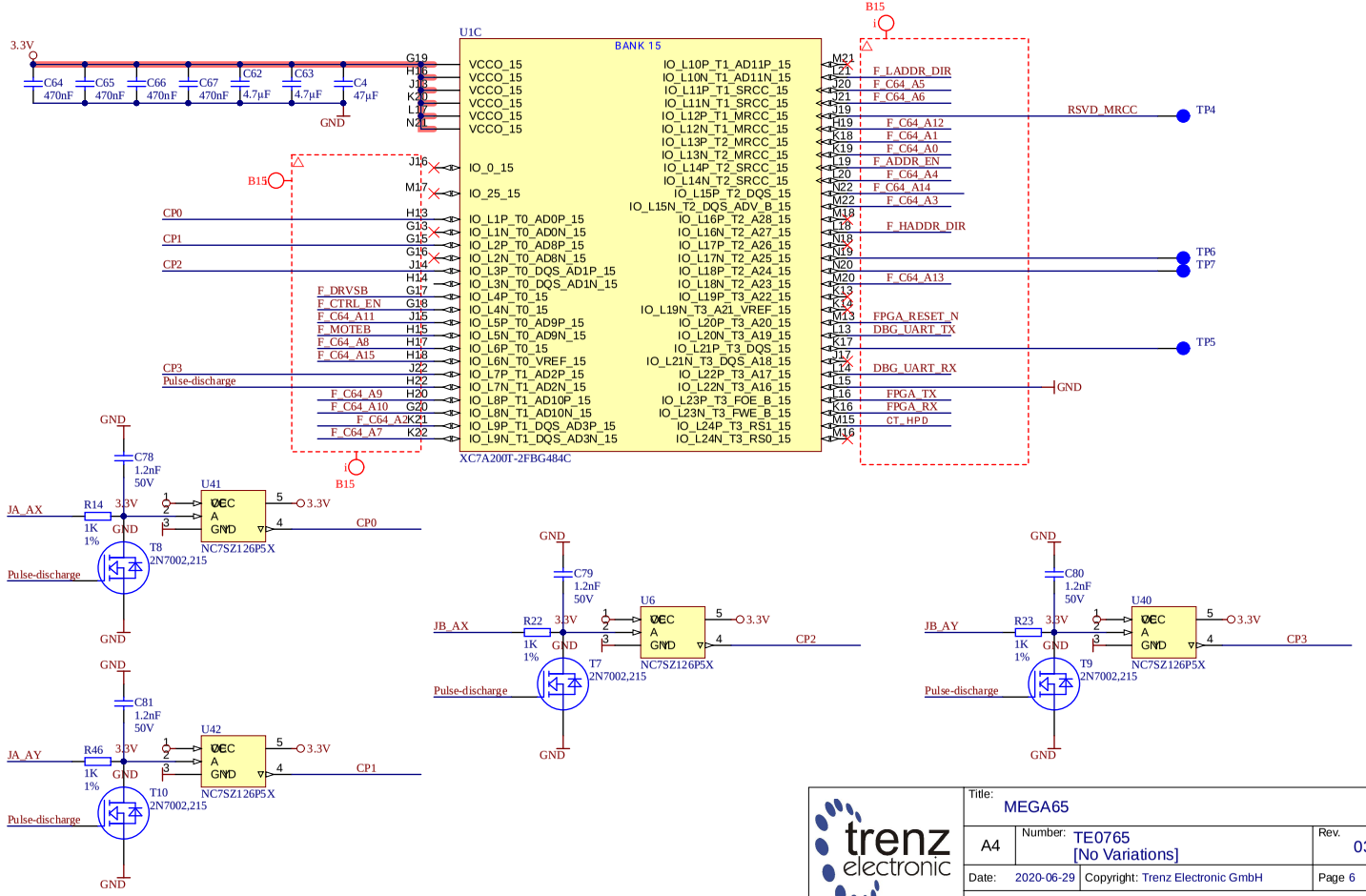

C64_IRQ R37 4K7
C64_RW R38 4K7
C64_CLOCK R39 4K7
C64_IO1 R40 4K7
C64_GAME R41 4K7
C64_EXROM R42 4K7
C64_IO2 R43 4K7
C64_ROML R45 4K7
C64_BA R71 4K7
C64_DMA R90 4K7
C64_D7 R91 4K7
C64_D6 R92 4K7
C64_D5 R93 4K7
C64_D4 R94 4K7
C64_D3 R95 4K7
C64_D2 R96 4K7
C64_D1 R97 4K7
C64_D0 R98 4K7
5V
U2
SN74LVCH16T245DGV
F_DATA_DIR
F_DATA_EN
F_LADDR_DIR
F_ADDR_EN
F_C64_D7
F_C64_D6
F_C64_D5
F_C64_D4
F_C64_D3
F_C64_D2
F_C64_D1
F_C64_D0
F_C64_A7
F_C64_A6
F_C64_A5
F_C64_A4
F_C64_A3
F_C64_A2
F_C64_A1
F_C64_A0
U30
NC7SZ126P5X
C64_EXROM
F_C64_EXROM
U31
C64_GAME
F_C64_GAME
C64_ROMH R99 4K7
C64_RESET R100 4K7
C64_NMI R101 4K7
C64_O2 R102 4K7
C64_A15 R103 4K7
C64_A14 R104 4K7
C64_A13 R105 4K7
C64_A12 R106 4K7
C64_A11 R107 4K7
C64_A10 R108 4K7
C64_A9 R109 4K7
C64_A8 R110 4K7
C64_A7 R111 4K7
C64_A6 R112 4K7
C64_A5 R113 4K7
C64_A4 R114 4K7
C64_A3 R115 4K7
C64_A2 R116 4K7
C64_A1 R117 4K7
C64_A0 R118 4K7
U9
F_C64_NMI
F_C64_IRQ
F_C64_DMA
FB_RIGHT
FB_LEFT
FB_DOWN
FB_FIRE
FB_UP
F_C64_RESET
F_C64_CLOCK
F_C64_O2
F_SER_ATN
F_SER_RESET
F_MOTEB
F_DRVSB
U8
F_CTRL_DIR
F_CTRL_EN
F_HADDR_DIR
F_C64_RW
F_C64_ROMH
F_C64_IO1
F_C64_IO2
F_C64_ROML
F_C64_BA
F_C64_A15
F_C64_A14
F_C64_A13
F_C64_A12
F_C64_A11
F_C64_A10
F_C64_A9
F_C64_A8
J8
120-044-60
Title: MEGA65
A4
Number: TE0765 [No Variations]
Rev. 03
Date: 2020-06-29
Copyright: Trenz Electronic GmbH
Page 3 of 23
Filename: EXP_Slot.SchDoc
trenz electronic



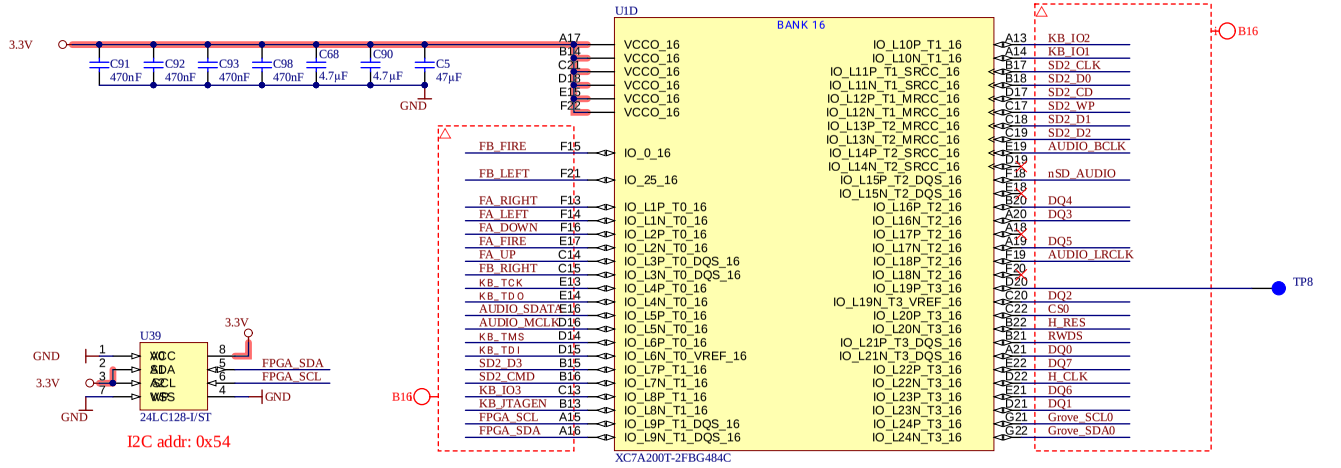
trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 4 of 23
	Filename: B13.SchDoc		



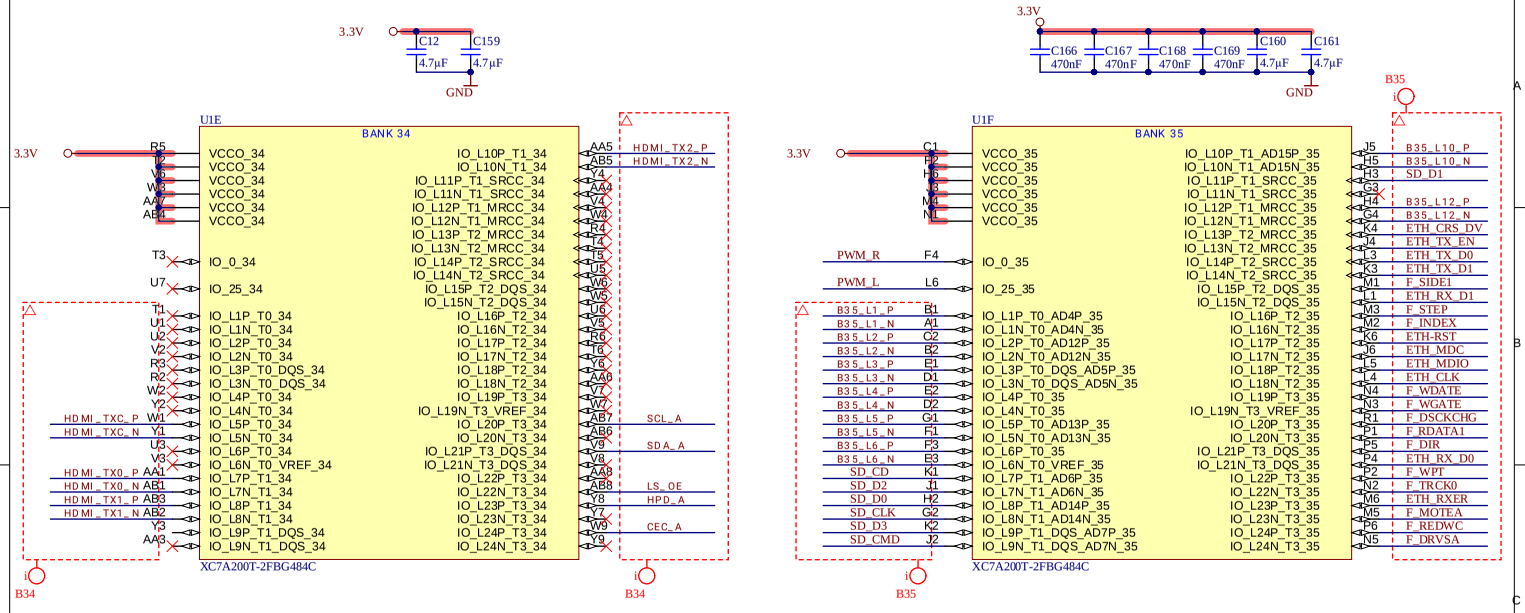
Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 5 of 23
Filename: B14.SchDoc			



Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 6 of 23
Filename: B15.SchDoc			

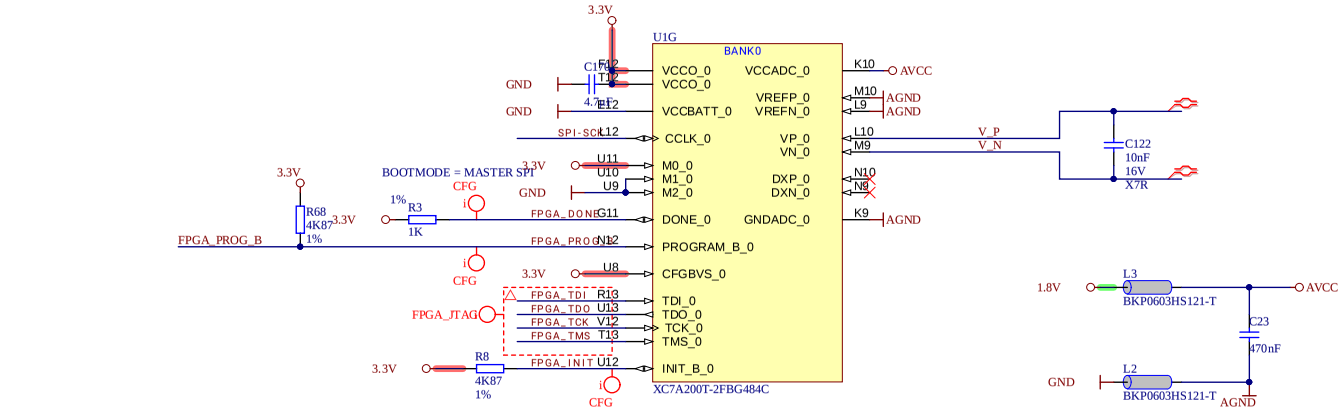


trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 7 of 23
	Filename: B16.SchDoc		

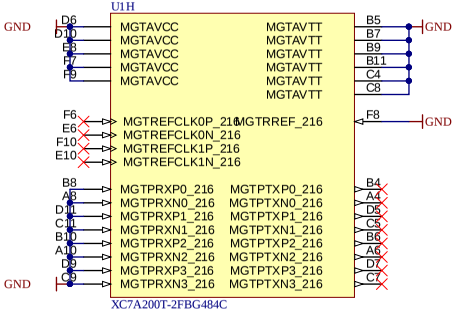


trenz electronic

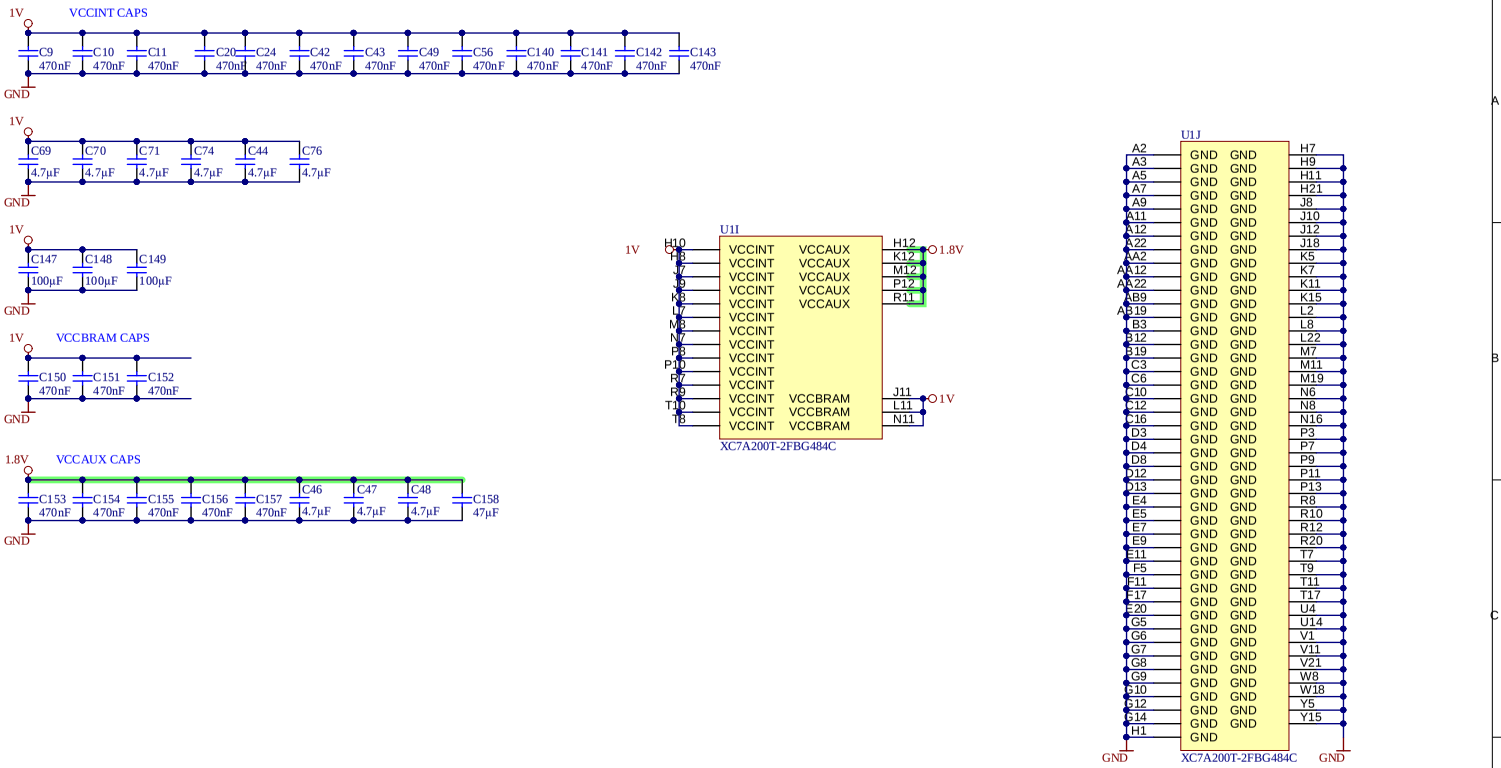
Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 8 of 23
Filename: B34.SchDoc			



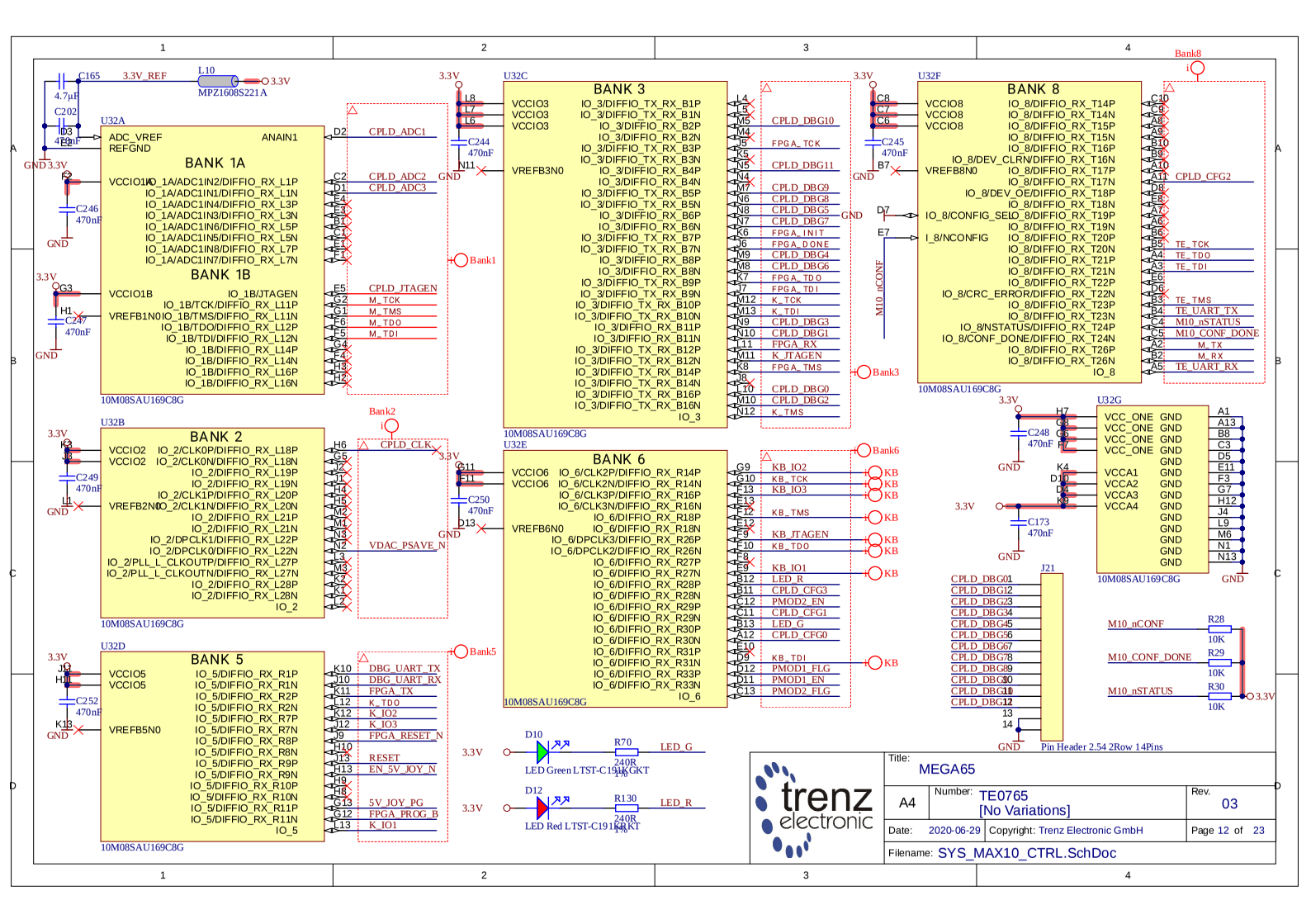
trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 9 of 23
	Filename: FPGA-CFG.SchDoc		

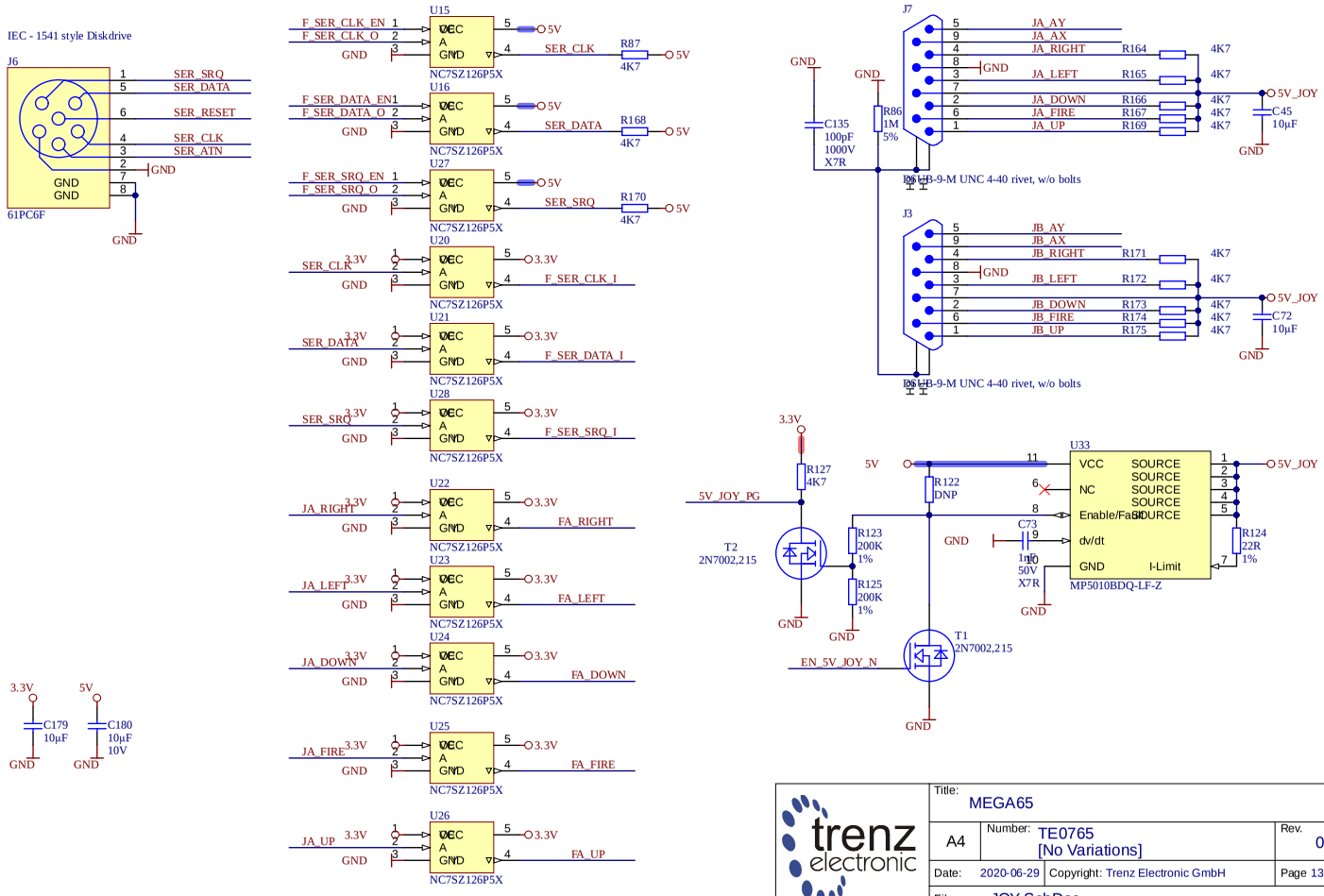


trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 10 of 23
	Filename: FPGA-MGT.SchDoc		

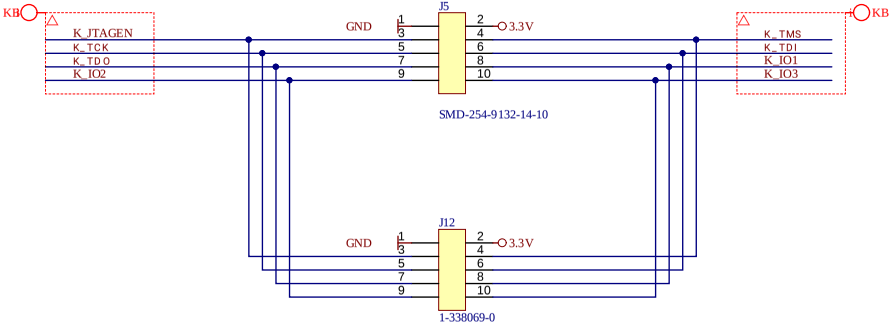


Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 11 of 23
Filename: FPGA-PWR.SchDoc			

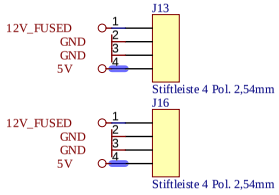

U32A
BANK 1A
BANK 1B
U32B
BANK 2
U32D
BANK 5
U32C
BANK 3
U32E
BANK 6
U32F
BANK 8
U32G
10M08SAU169C8G
3.3V
GND
Bank1
Bank2
Bank3
Bank5
Bank6
Bank8
CPLD_ADC1
CPLD_ADC2
CPLD_ADC3
CPLD_JTAGEN
M_TCK
M_TMS
M_TDO
M_TDI
CPLD_CLK
VDAC_PSAVE_N
DBG_UART_TX
DBG_UART_RX
FPGA_TX
K_TDO
K_IO2
K_IO3
FPGA_RESET_N
RESET
EN_5V_JOY_N
5V_JOY_PG
FPGA_PROG_B
K_IO1
CPLD_DBG10
FPGA_TCK
CPLD_DBG11
CPLD_DBG9
CPLD_DBG8
CPLD_DBG5
CPLD_DBG7
FPGA_INIT
FPGA_DONE
CPLD_DBG4
CPLD_DBG6
FPGA_TDO
FPGA_TDI
K_TCK
K_TDI
CPLD_DBG3
CPLD_DBG1
FPGA_RX
K_JTAGEN
FPGA_TMS
CPLD_DBG0
CPLD_DBG2
K_TMS
KB_IO2
KB_TCK
KB_IO3
KB_TMS
KB_JTAGEN
KB_TDO
KB_IO1
LED_R
CPLD_CFG3
PMOD2_EN
CPLD_CFG1
LED_G
CPLD_CFG0
KB_TDI
PMOD1_FLG
PMOD1_EN
PMOD2_FLG
M10_nCONF
CPLD_CFG2
TE_TCK
TE_TDO
TE_TDI
TE_TMS
TE_UART_TX
M10_nSTATUS
M10_CONF_DONE
M_TX
M_RX
TE_UART_RX
D10
LED Green LTST-C191KGKT
R70
240R
D12
LED Red LTST-C191KRKT
R130
240R
J21
Pin Header 2.54 2Row 14Pins
R28
R29
R30
10K
trenz electronic
Title: MEGA65
A4
Number: TE0765 [No Variations]
Rev. 03
Date: 2020-06-29
Copyright: Trenz Electronic GmbH
Page 12 of 23
Filename: SYS_MAX10_CTRL.SchDoc

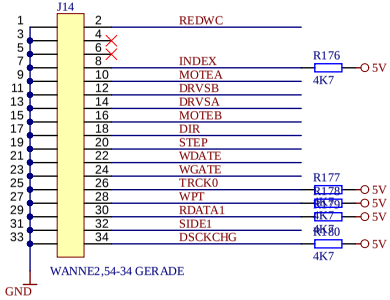


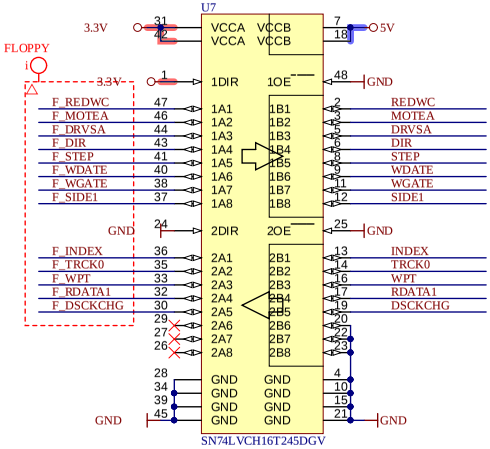
Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 13 of 23
Filename: JOY.SchDoc			



trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 14 of 23
	Filename: KEYBOARD.SchDoc		

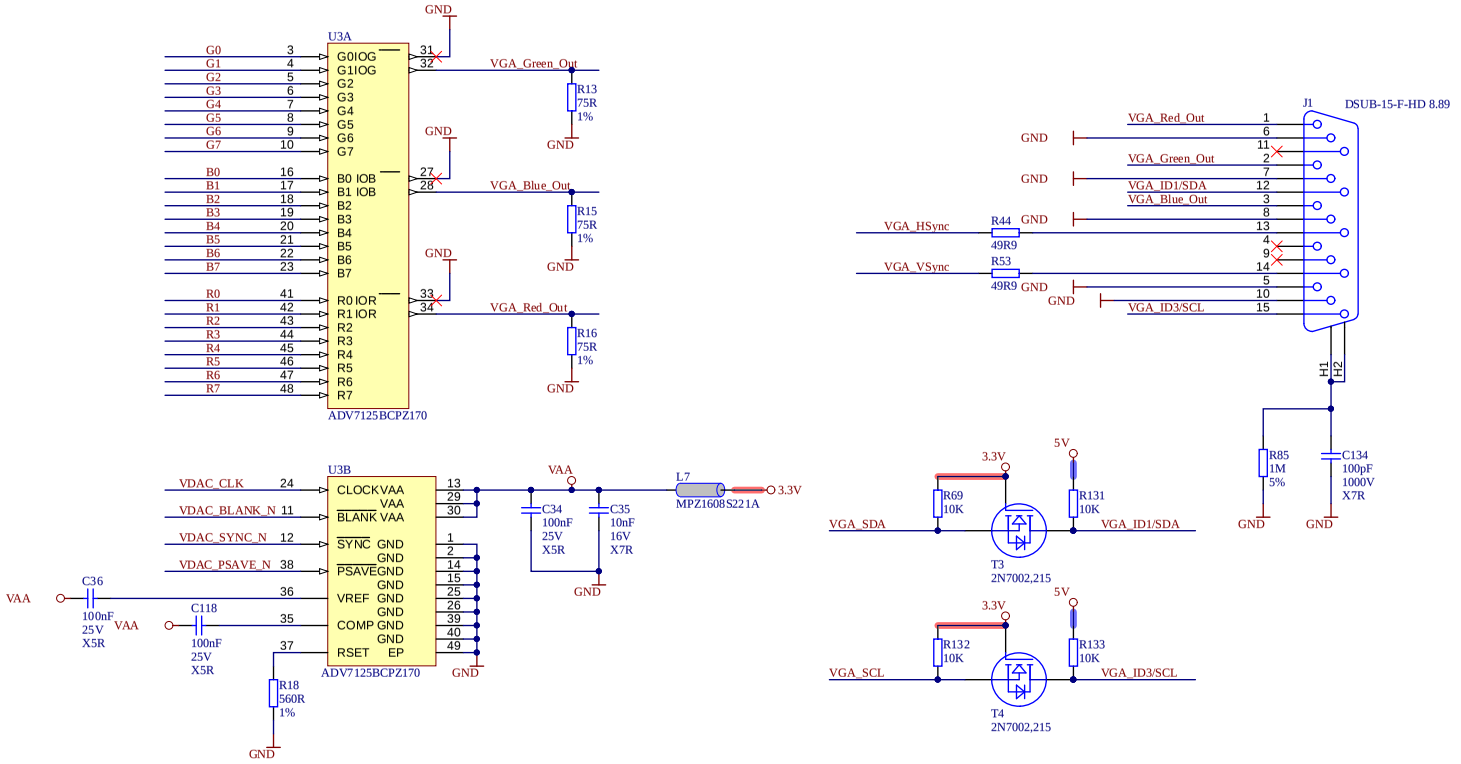




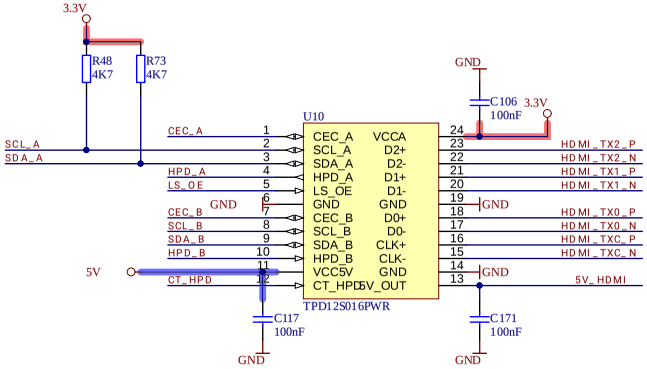


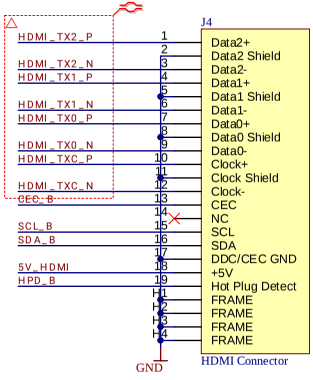


Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 15 of 23
Filename: Floppy.SchDoc			

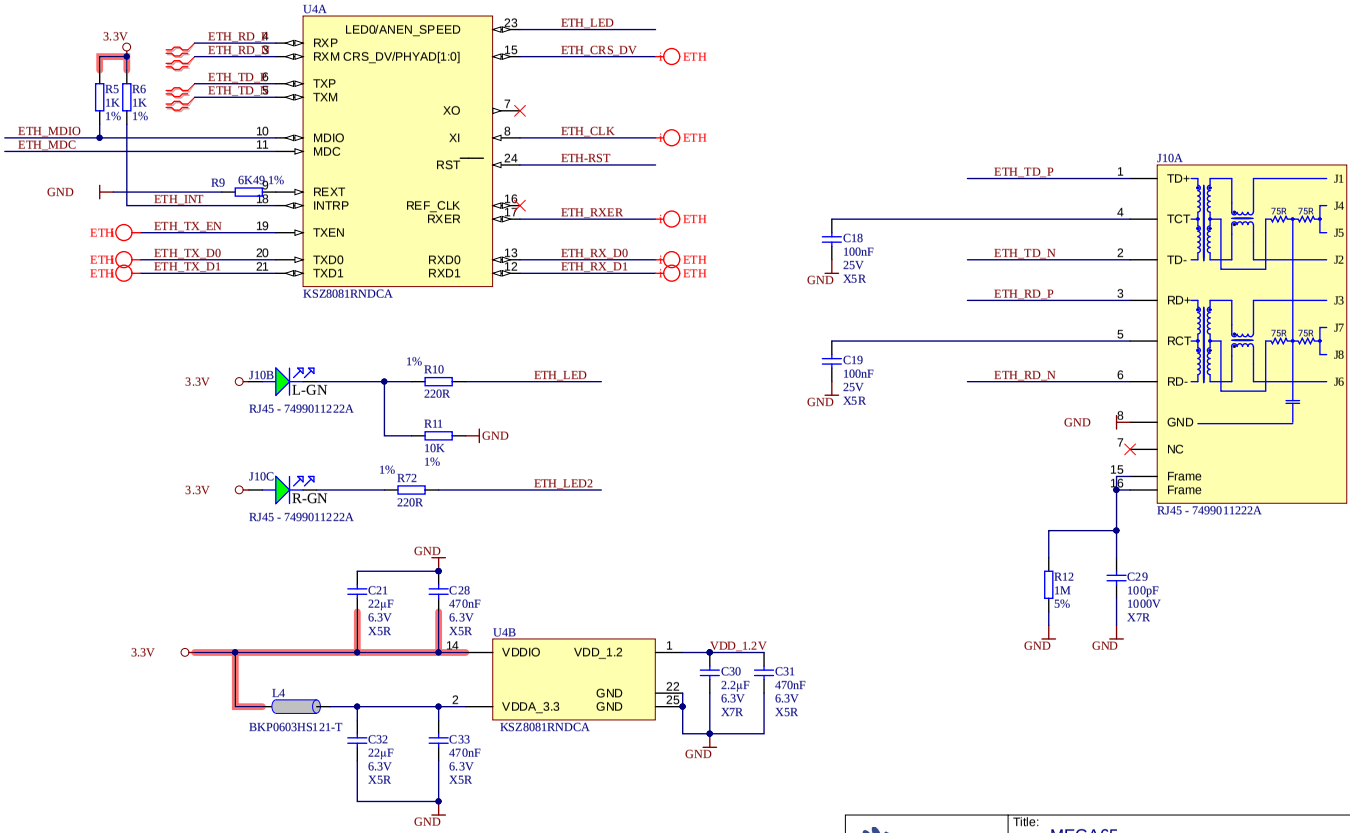


trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 16 of 23
	Filename: VGA.SchDoc		

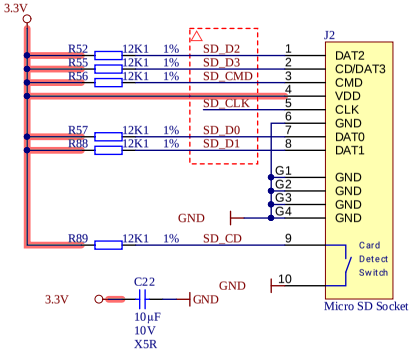


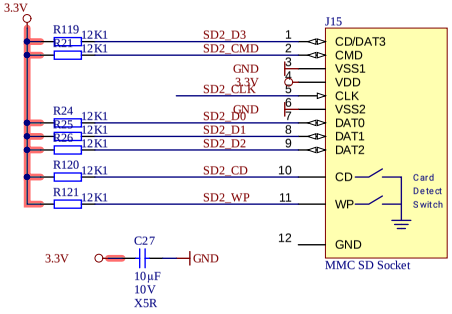


trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 17 of 23
	Filename: HDMI.SchDoc		

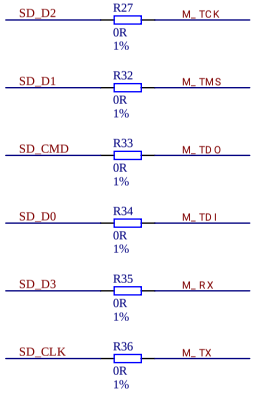


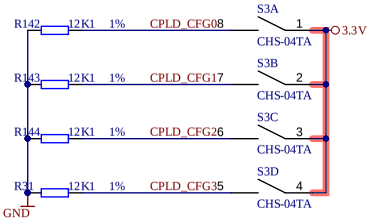
trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 18 of 23
	Filename: Ethernet.SchDoc		





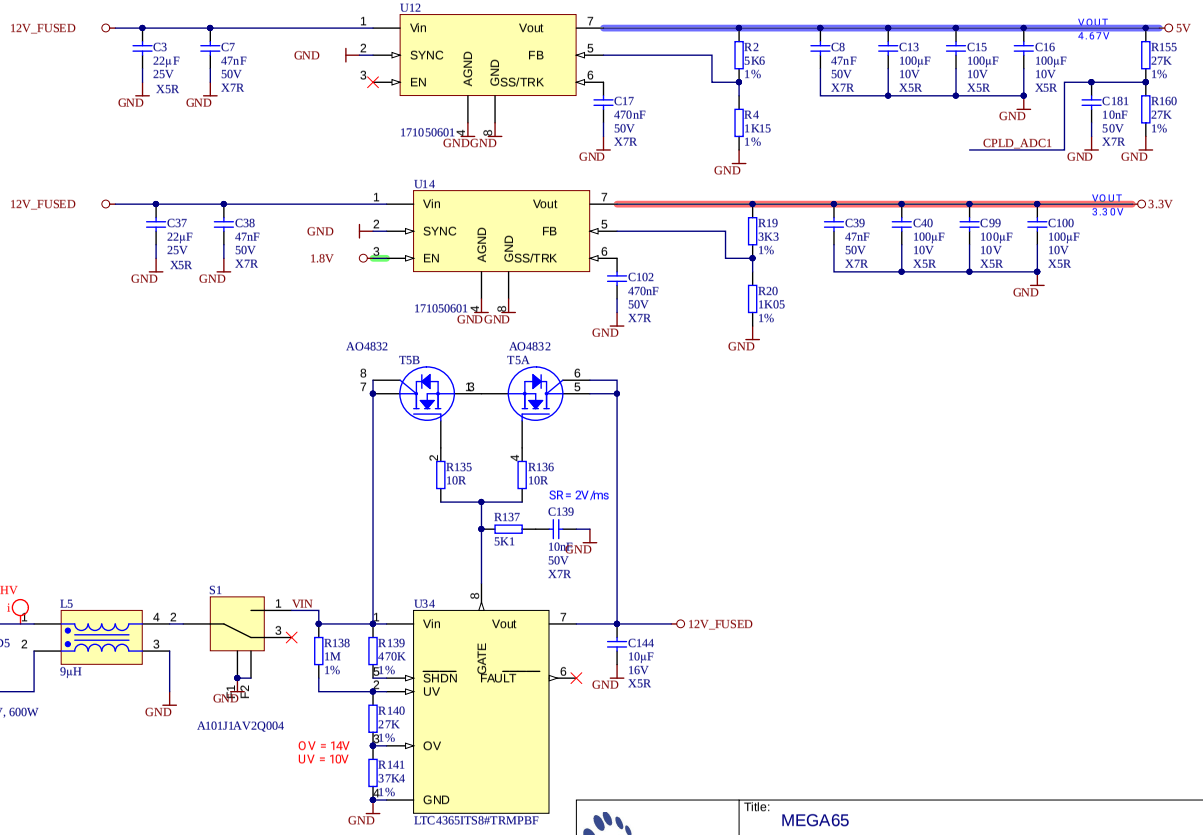
http://linux-sunxi.org/MicroSD_Breakout





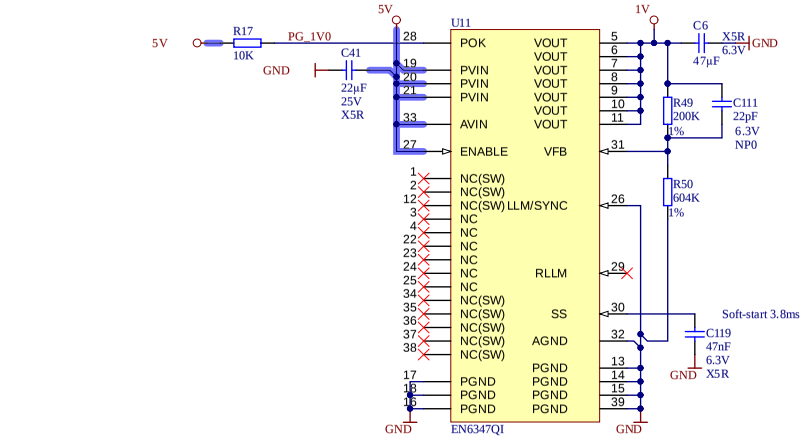
trenz electronic

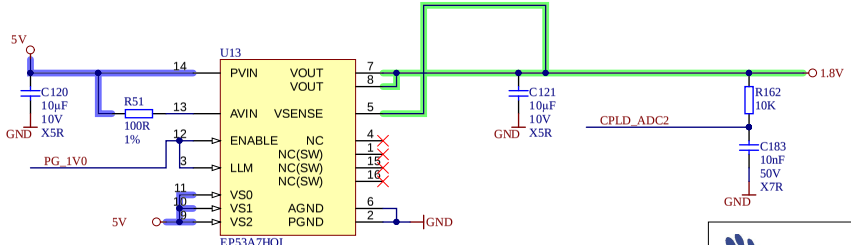
Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 19 of 23
Filename: LED_SW_BUT.SchDoc			



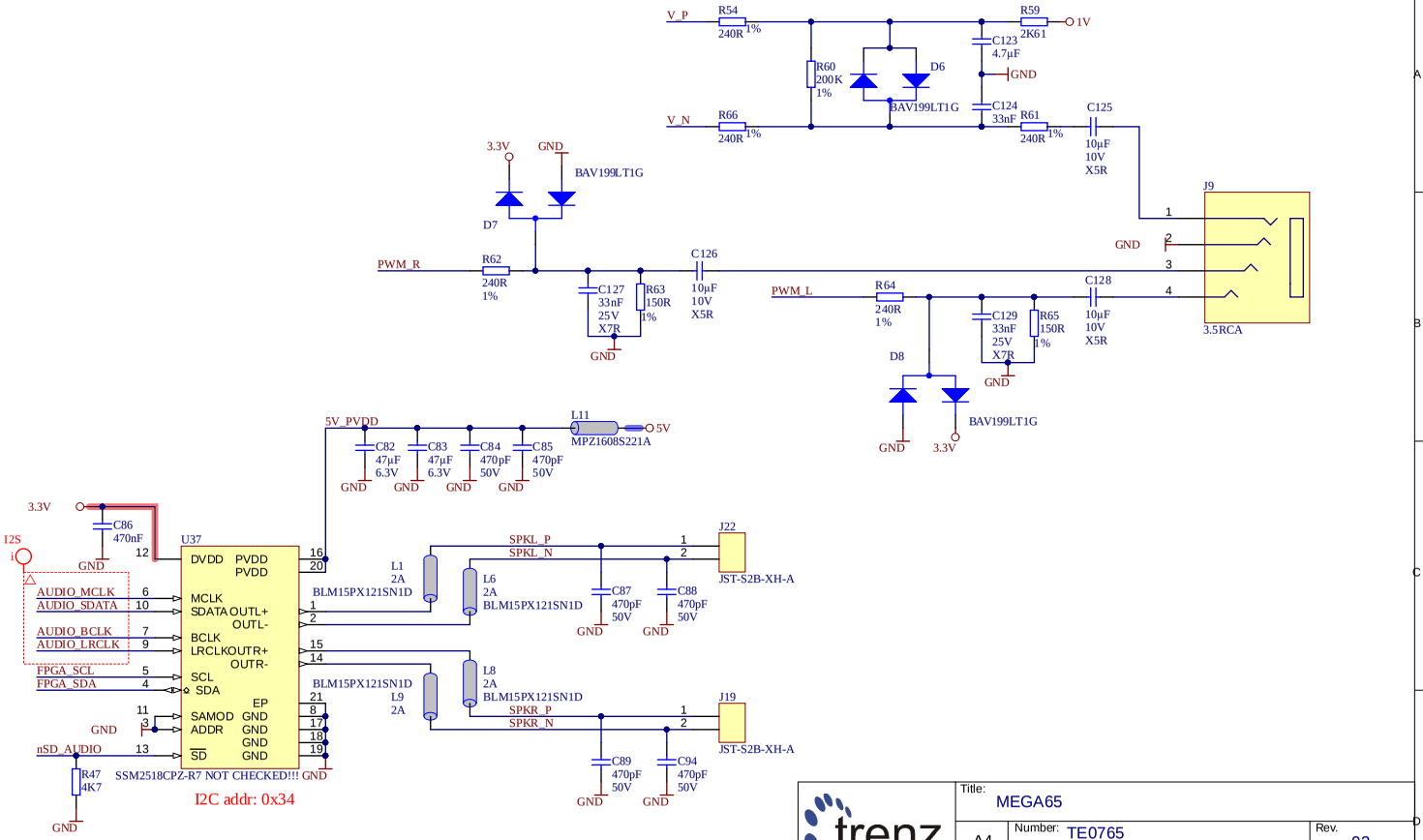
trenz electronic

Title: MEGA65			
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 20 of 23
Filename: PowerMain.SchDoc			





trenz electronic	Title: MEGA65		
	A4	Number: TE0765 [No Variations]	Rev. 03
	Date: 2020-06-29	Copyright: Trenz Electronic GmbH / TT	Page 21 of 23
	Filename: POWER.SchDoc		



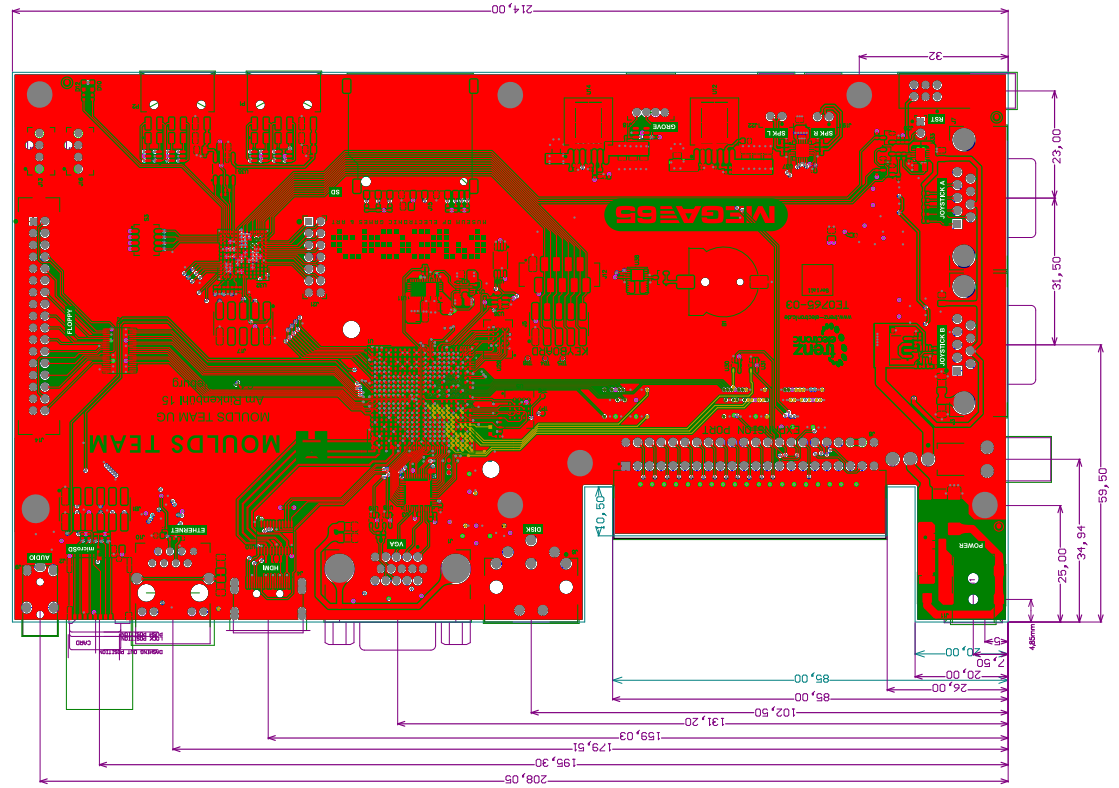
trenz electronic	Title: MEGA65		
A4	Number: TE0765 [No Variations]		Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH		Page 22 of 23
Filename: SOUND.SchDoc			

REVISION HISTORY

REV	Description	
-03	1. Added a VRP resistor on bank 65; 2. LDO U33 is changed on ADP7102ACPZ; 3. Signal FPGA IO0 is connected on AE18 pin of FPGA; 4. Signal DBG_LED3 is connected on AD18 pin of FPGA; 5. Signal MIO13_25 connected to J1 pin 33 instead MIO25. 6. Resistor R84 is removed; 7. LED D1 moved on edge of PCB; 8. Added THT testpoints J4 on CPLD_JTAGEN, R76 was removed; 9. Signals B49_XX_X are renamed in B88_XX_X; 10. C241 is changed on 1nF; 11. Length of CLK signals on RFADC and RFDAC are adjusted; 12. Wrong connection on U8 is fixed (PCB); 13. Wrong connection PGOOD1 pin of U7 is fixed;	IG

trenz electronic

Title: MEGA65		
A4	Number: TE0835 [No Variations]	Rev. 03
Date: 2020-06-29	Copyright: Trenz Electronic GmbH	Page 23 of 23
Filename: Revision_Changes.SchDoc		


214,00
32
23,00
31,50
59,50
34,94
25,00
10,50
7,50
20,00
26,00
85,00
85,00
102,50
131,20
159,03
179,51
195,30
208,05
MEGA65
MOULDS TEAM
MOULDS TEAM UG
An Beckedorf 15
POWER
ETHERNET
HDMI
VGA
DISK
AUDIO
JOYSTICK A
JOYSTICK B
SD
RST

MEGA65 R2 ELEKTRONISCH SCHEMA

U_HDMI
HDMI.SchDoc

U_Ethernet
Ethernet.SchDoc

U_LED_SW_BUT
LED_SW_BUT.SchDoc

U_SYS_MAX10_CTRL
SYS_MAX10_CTRL.SchDoc

U_SOUND
SOUND.SchDoc

U_B13
B13.SchDoc

U_B14
B14.SchDoc

U_B15
B15.SchDoc

U_B16
B16.SchDoc

U_B34
B34.SchDoc

U_FPGA-CFG
FPGA-CFG.SchDoc

U_FPGA-MGT
FPGA-MGT.SchDoc

U_FPGA-PWR
FPGA-PWR.SchDoc

U_JOY
JOY.SchDoc

U_KEYBOARD
KEYBOARD.SchDoc

U_POWER
POWER.SchDoc

U_PowerMain
PowerMain.SchDoc

U_VGA
VGA.SchDoc

U_EXT_HEADER
EXT_HEADER.SchDoc

U_EXP_Slot
EXP_Slot.SchDoc

U_Floppy
Floppy.SchDoc

PM1 FIDU-DOT - small
PM2 FIDU-DOT - small
PM3 FIDU-DOT - small
PM4 FIDU-DOT - small
PM5 FIDU-DOT - small
PM6 FIDU-DOT - small

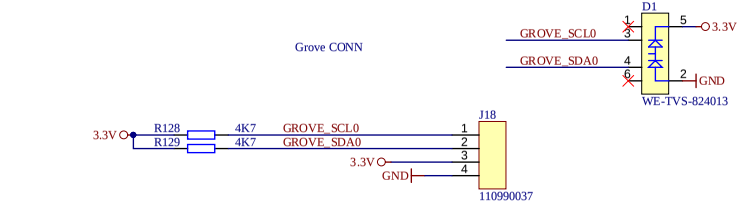
Serial1
Serial
Serialnumber 6,3 x 6.3mm

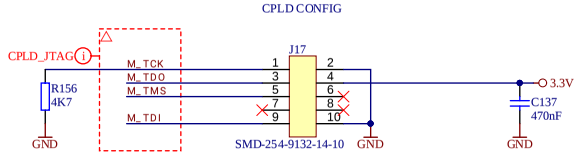
S/N1
1 0R 2
Serialnumber

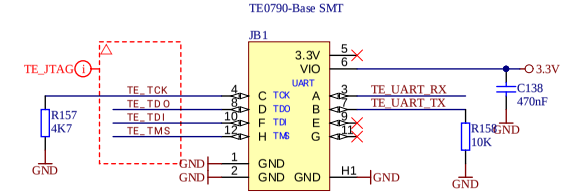
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm
GND Mount.Hole 3.2mm

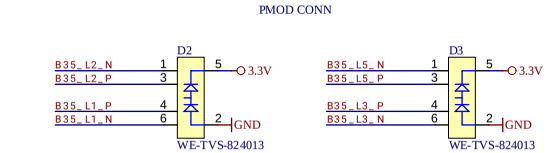
trenz electronic

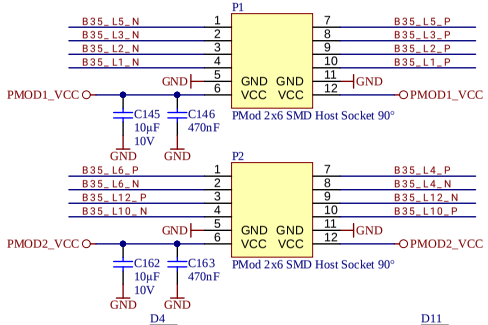
Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page1 of 23
Filename: TE0765.SchDoc			

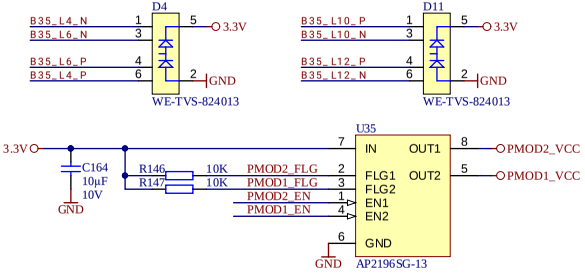






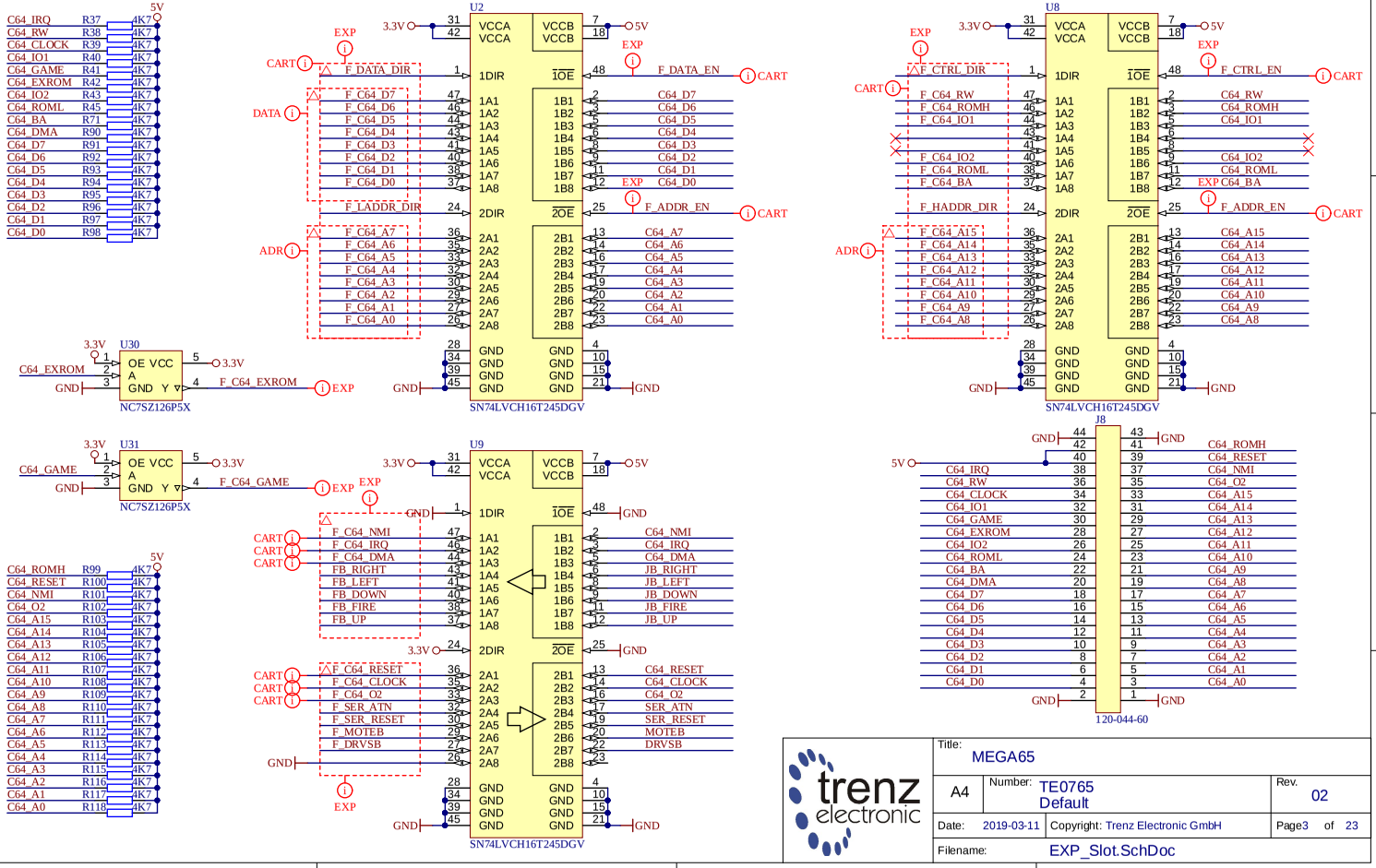




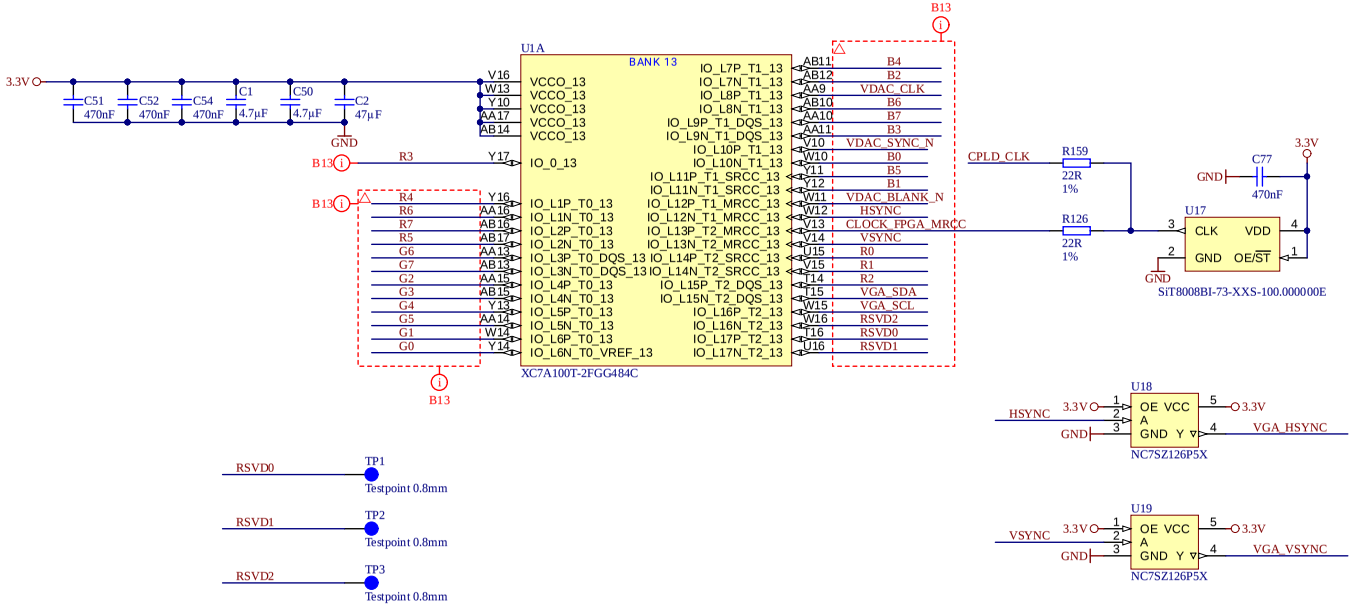


trenz electronic

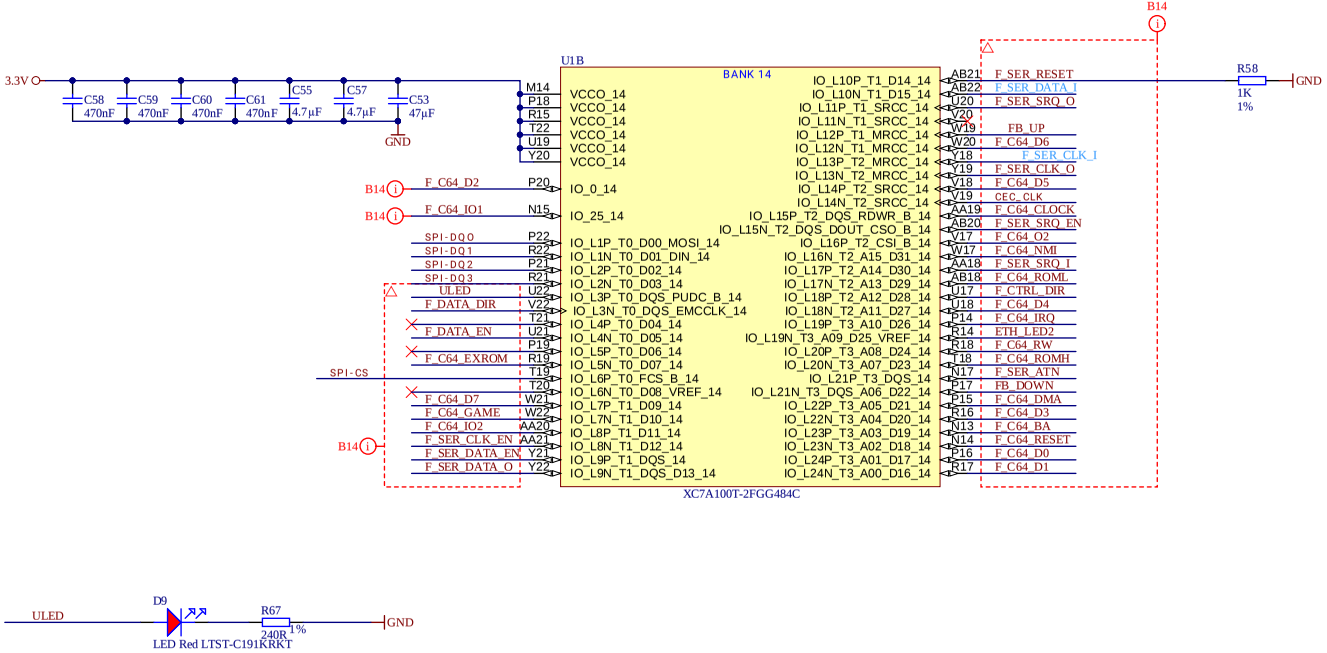
Title: MEGA65		
A4	Number: TE0765 Default	Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH	Page2 of 23
Filename: EXT_HEADER.SchDoc		



Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page3 of 23
Filename: EXP_Slot.SchDoc			

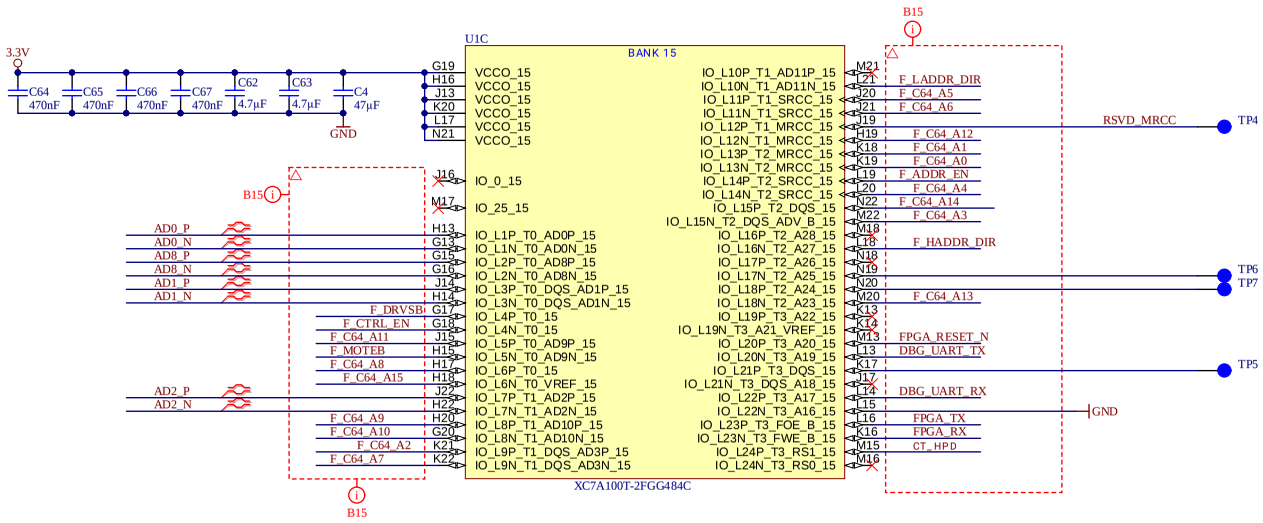


trenz electronic	Title: MEGA65		
	A4	Number: TE0765 Default	Rev. 02
	Date: 2019-03-11	Copyright: Trenz Electronic GmbH	Page4 of 23
	Filename: B13.SchDoc		



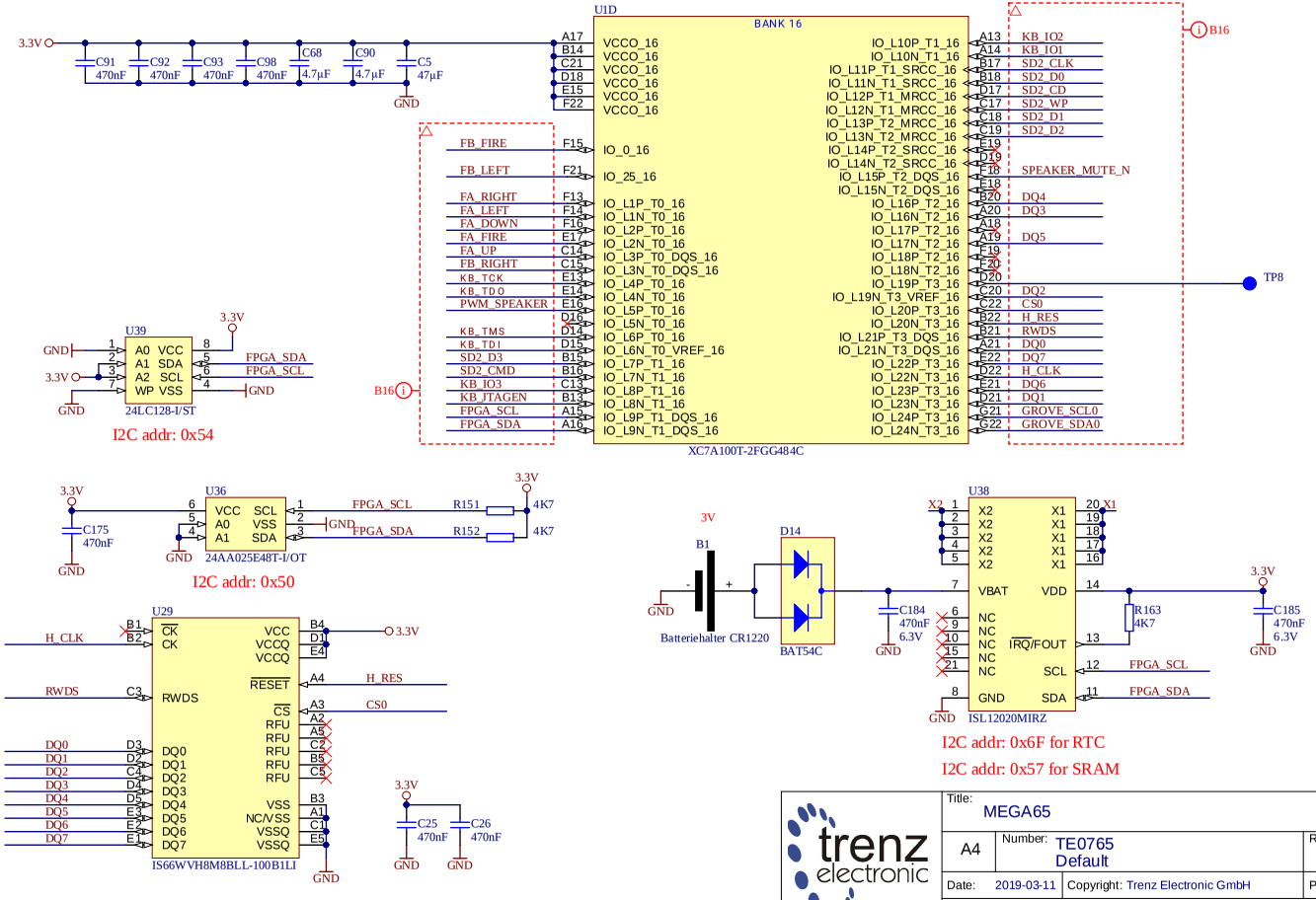
trenz electronic

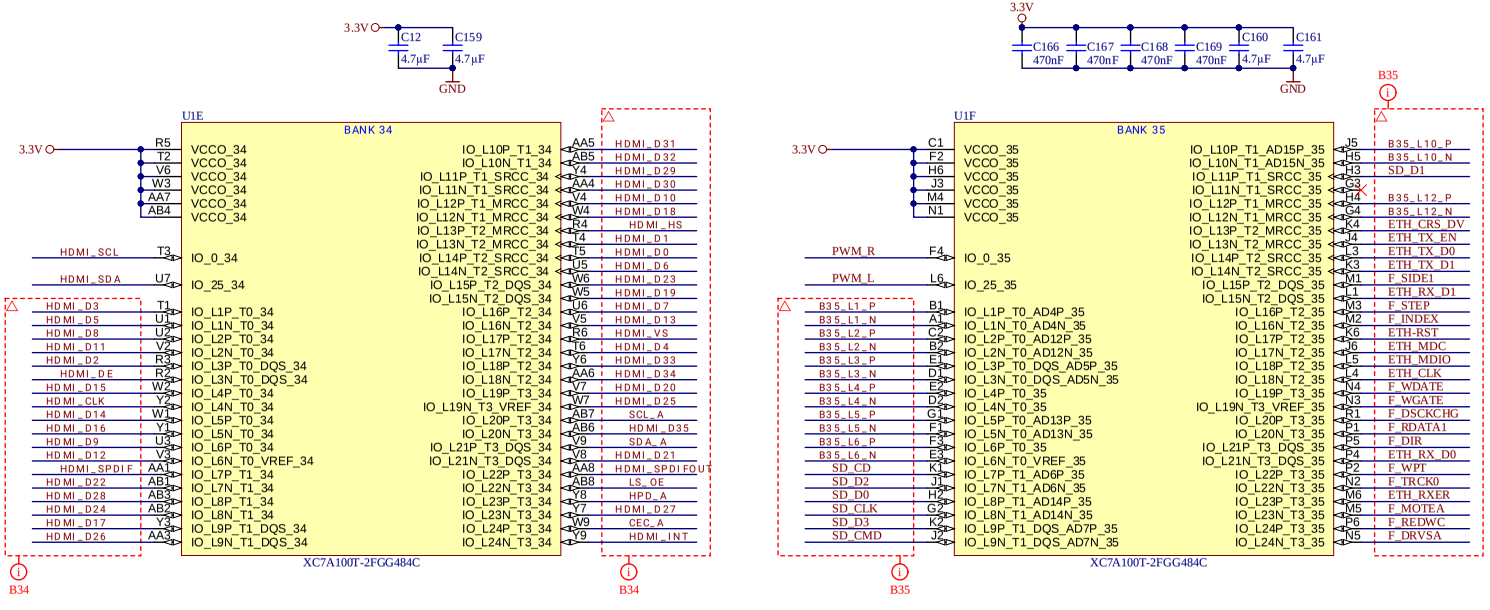
Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page5 of 23
Filename: B14.SchDoc			



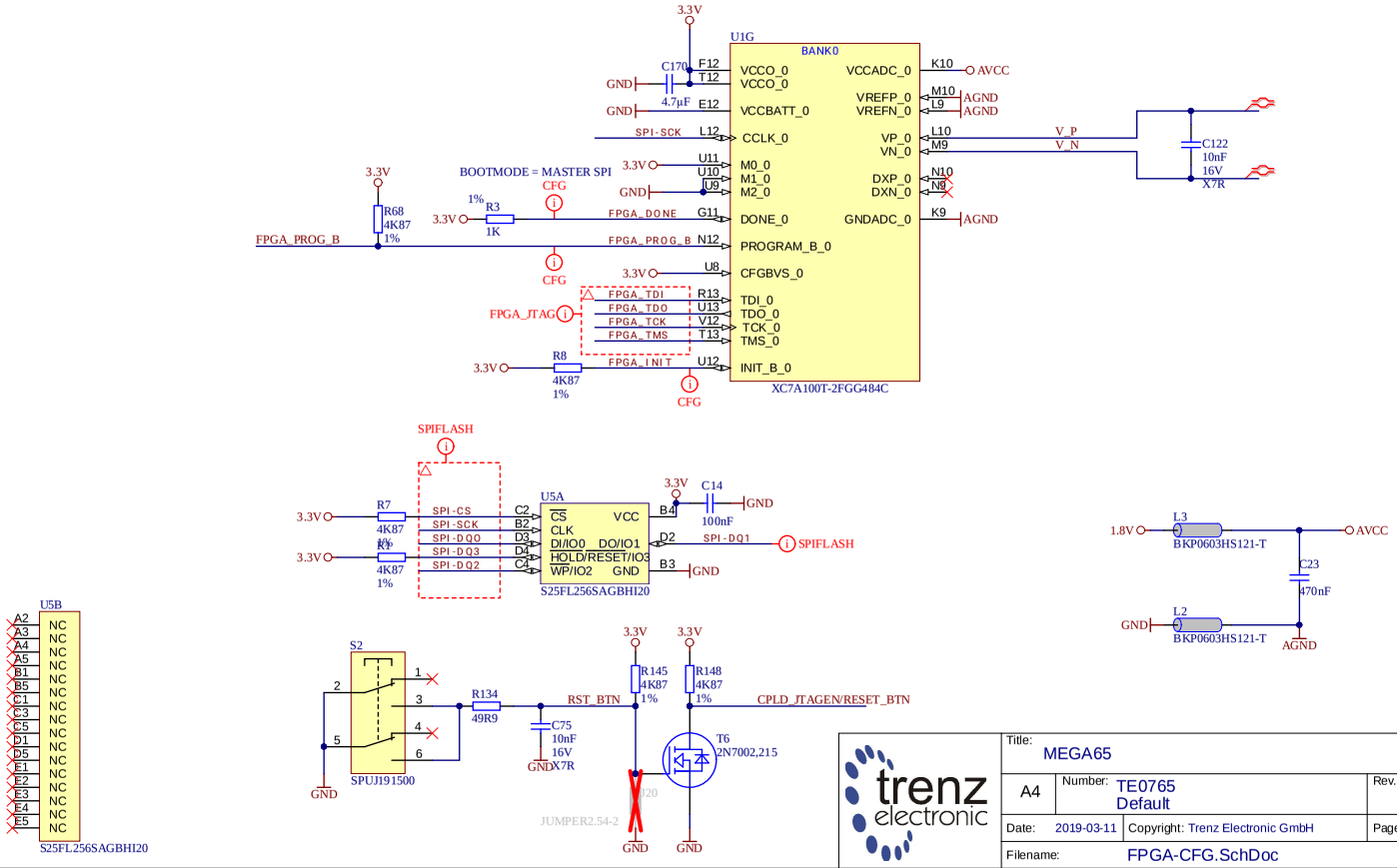
trenz electronic

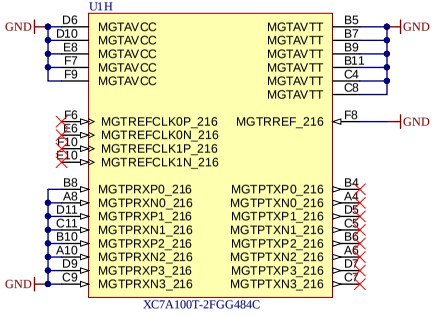
Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page6 of 23
Filename: B15.SchDoc			


U1D
BANK 16
XC7A100T-2FGG484C
3.3V
C91 470nF
C92 470nF
C93 470nF
C98 470nF
C68 4.7µF
C90 4.7µF
C5 47µF
GND
FB_FIRE
FB_LEFT
FA_RIGHT
FA_LEFT
FA_DOWN
FA_FIRE
FA_UP
FB_RIGHT
KB_TCK
KB_TDO
PWM_SPEAKER
KB_TMS
KB_TDI
SD2_D3
SD2_CMD
KB_IO3
KB_JTAGEN
FPGA_SCL
FPGA_SDA
B16
KB_IO2
KB_IO1
SD2_CLK
SD2_D0
SD2_CD
SD2_WP
SD2_D1
SD2_D2
SPEAKER_MUTE_N
DQ4
DQ3
DQ5
DQ2
CS0
H_RES
RWDS
DQ0
DQ7
DQ6
DQ1
TP8
U39
24LC128-I/ST
I2C addr: 0x54
U36
24AA025E48T-I/OT
I2C addr: 0x50
R151 4K7
R152 4K7
C175 470nF
U29
IS66WVH8M8BLL-100B1LI
C25 470nF
C26 470nF
3V
B1
Batteriehalter CR1220
D14
BAT54C
C184 470nF 6.3V
U38
ISL12020MIRZ
R163 4K7
C185 470nF 6.3V
I2C addr: 0x6F for RTC
I2C addr: 0x57 for SRAM
trenz electronic
Title: MEGA65
A4
Number: TE0765 Default
Rev. 02
Date: 2019-03-11
Copyright: Trenz Electronic GmbH
Page7 of 23
Filename: B16.SchDoc

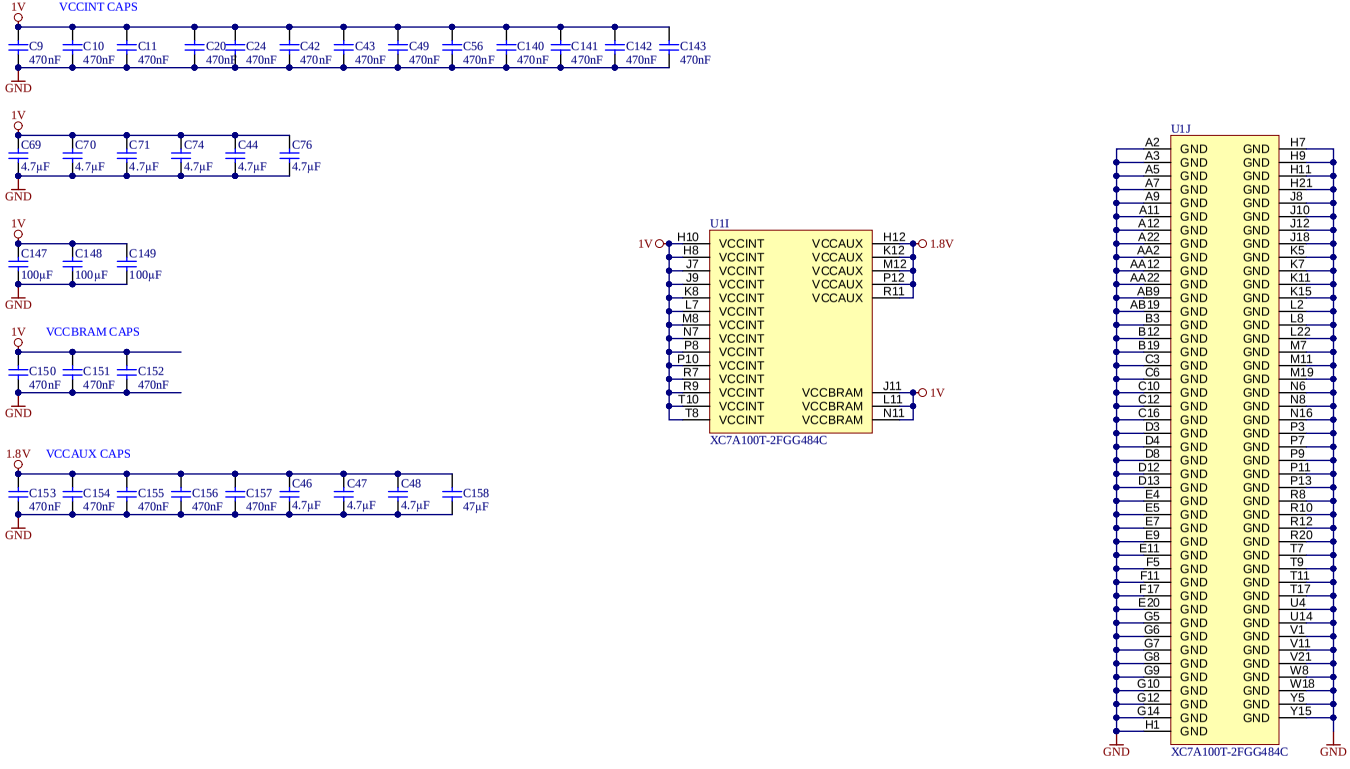


Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page8 of 23
Filename: B34.SchDoc			

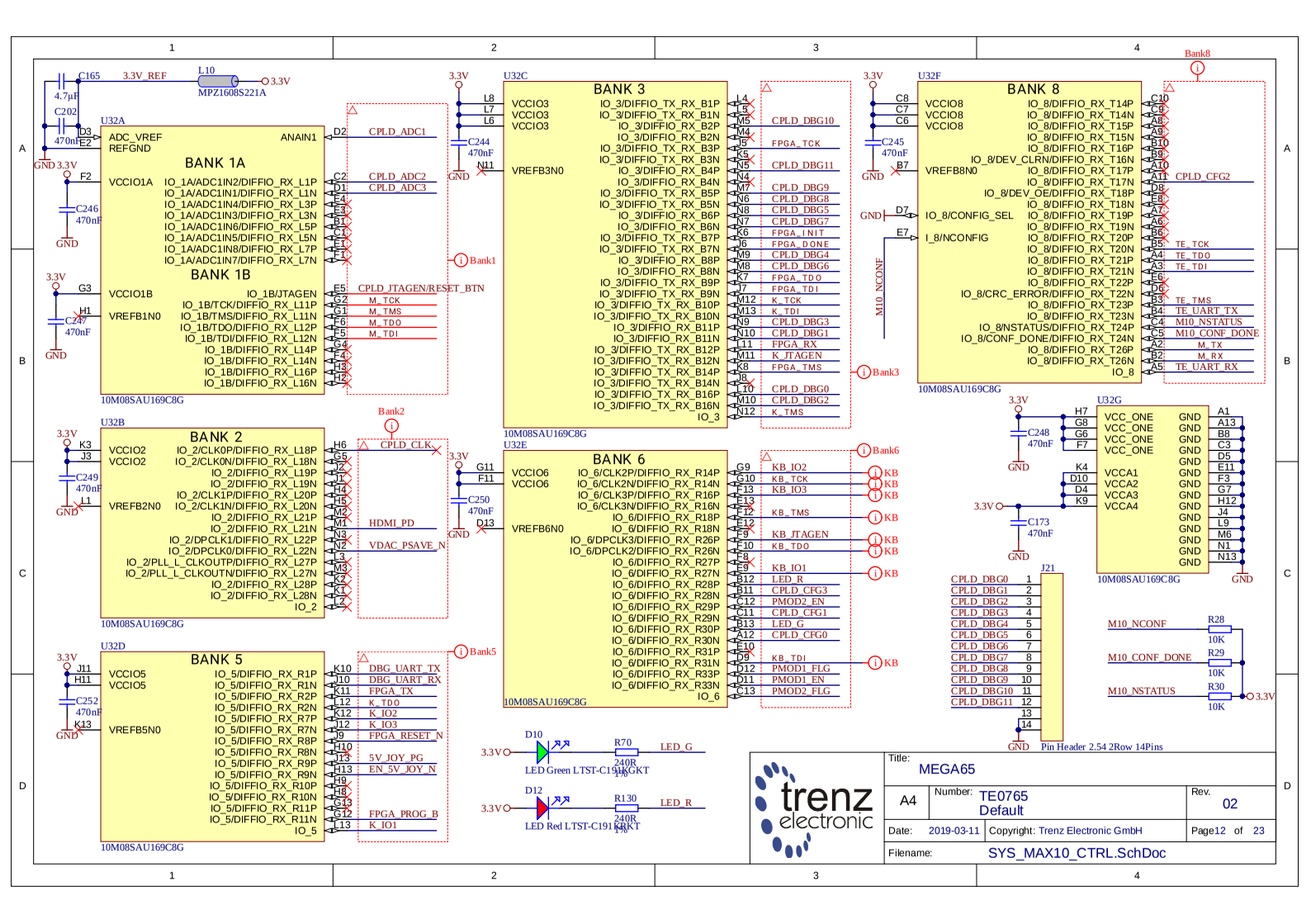

U1G
BANK0
XC7A100T-2FGG484C
VCCO_0
VCCADC_0
VCCBATT_0
VREFP_0
VREFN_0
CCLK_0
VP_0
VN_0
M0_0
M1_0
M2_0
DXP_0
DXN_0
DONE_0
GNDADC_0
PROGRAM_B_0
CFGBVS_0
TDI_0
TDO_0
TCK_0
TMS_0
INIT_B_0
3.3V
AVCC
AGND
GND
SPI-SCK
V_P
V_N
BOOTMODE = MASTER SPI
CFG
FPGA_DONE
FPGA_PROG_B
FPGA_JTAG
FPGA_TDI
FPGA_TDO
FPGA_TCK
FPGA_TMS
FPGA_INIT
C170
4.7µF
C122
10nF
16V
X7R
R68
4K87
1%
R3
1K
R8
SPIFLASH
U5A
S25FL256SAGBHI20
SPI-CS
SPI-DQ0
SPI-DQ1
SPI-DQ2
SPI-DQ3
C14
100nF
R7
L3
L2
BKP0603HS121-T
1.8V
C23
470nF
U5B
S2
SPUJ191500
R134
49R9
C75
10nF
16V
R145
R148
T6
2N7002,215
CPLD_JTAGEN/RESET_BTN
RST_BTN
JUMPER2.54-2
Title:
MEGA65
A4
Number:
TE0765
Default
Rev.
02
Date:
2019-03-11
Copyright: Trenz Electronic GmbH
Page9 of 23
Filename:
FPGA-CFG.SchDoc
trenz electronic

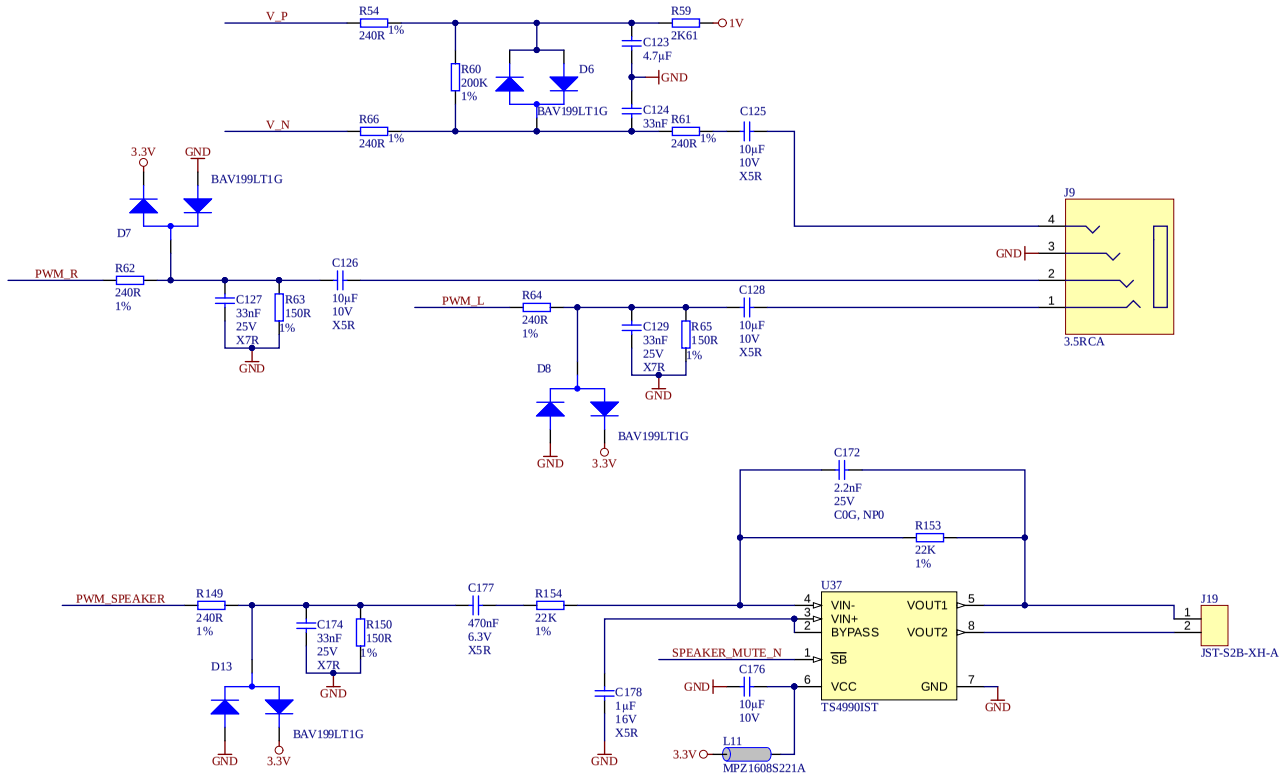


trenz electronic	Title: MEGA65		
	A4	Number: TE0765 Default	Rev. 02
	Date: 2019-03-11	Copyright: Trenz Electronic GmbH	Page10 of 23
	Filename: FPGA-MGT.SchDoc		

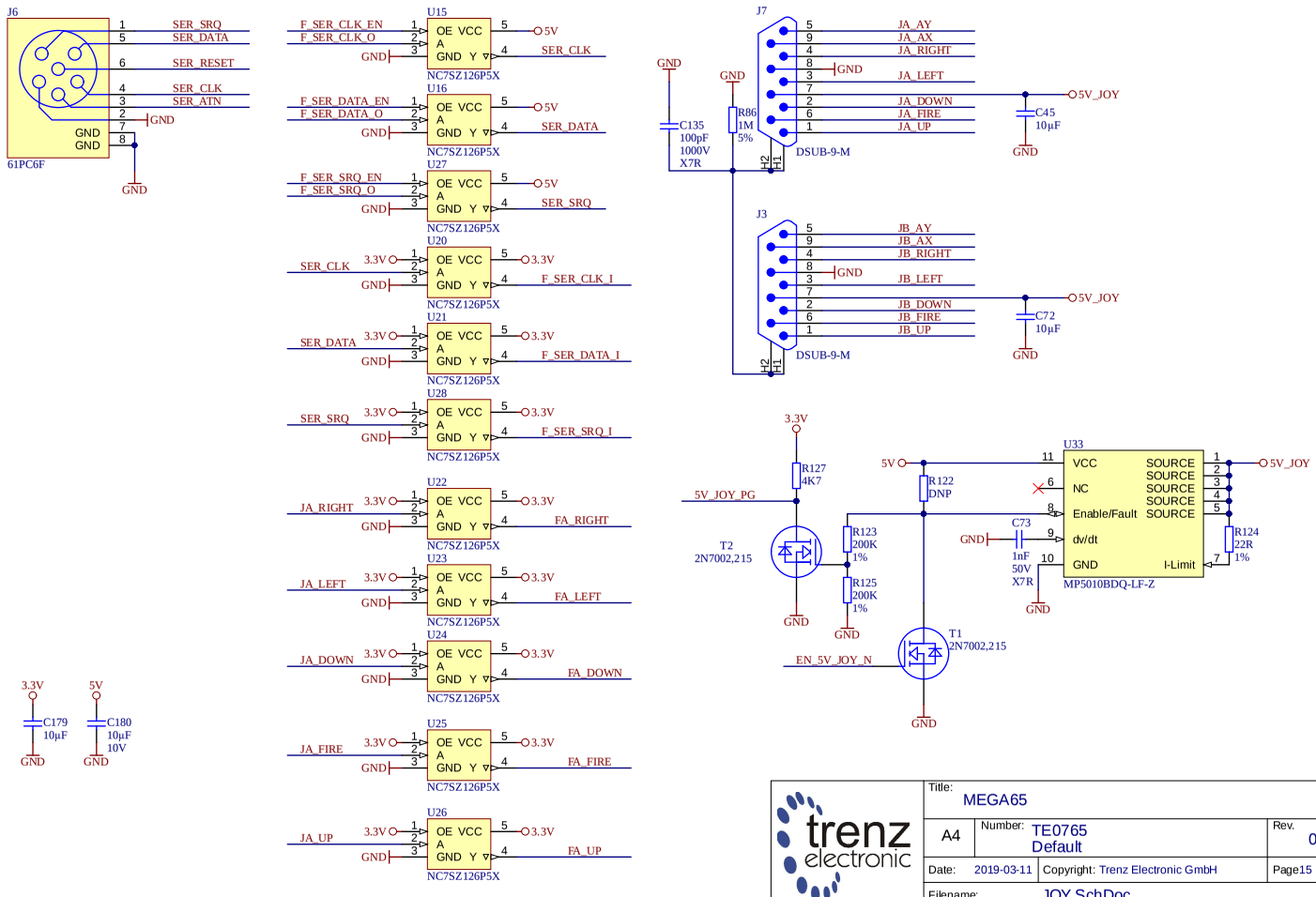


Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page 11 of 23
Filename: FPGA-PWR.SchDoc			

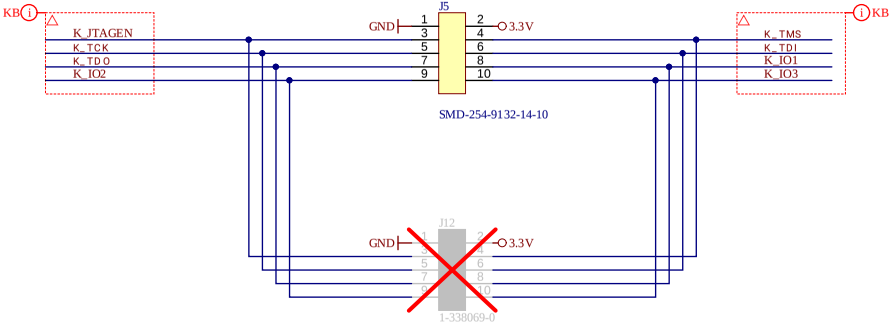

C165
3.3V_REF
L10
MPZ1608S221A
3.3V
4.7µF
C202
470nF
GND
U32A
ADC_VREF
REFGND
ANAIN1
BANK 1A
VCCIO1A
IO_1A/ADC1IN2/DIFFIO_RX_L1P
IO_1A/ADC1IN1/DIFFIO_RX_L1N
IO_1A/ADC1IN4/DIFFIO_RX_L3P
IO_1A/ADC1IN3/DIFFIO_RX_L3N
IO_1A/ADC1IN6/DIFFIO_RX_L5P
IO_1A/ADC1IN5/DIFFIO_RX_L5N
IO_1A/ADC1IN8/DIFFIO_RX_L7P
IO_1A/ADC1IN7/DIFFIO_RX_L7N
BANK 1B
VCCIO1B
IO_1B/JTAGEN
IO_1B/TCK/DIFFIO_RX_L11P
VREFB1N0
IO_1B/TMS/DIFFIO_RX_L11N
IO_1B/TDO/DIFFIO_RX_L12P
IO_1B/TDI/DIFFIO_RX_L12N
IO_1B/DIFFIO_RX_L14P
IO_1B/DIFFIO_RX_L14N
IO_1B/DIFFIO_RX_L16P
IO_1B/DIFFIO_RX_L16N
10M08SAU169C8G
CPLD_ADC1
CPLD_ADC2
CPLD_ADC3
CPLD_JTAGEN/RESET_BTN
M_TCK
M_TMS
M_TDO
M_TDI
Bank1
U32C
BANK 3
VCCIO3
VREFB3N0
IO_3/DIFFIO_TX_RX_B1P
IO_3/DIFFIO_TX_RX_B1N
IO_3/DIFFIO_RX_B2P
IO_3/DIFFIO_RX_B2N
IO_3/DIFFIO_TX_RX_B3P
IO_3/DIFFIO_TX_RX_B3N
IO_3/DIFFIO_RX_B4P
IO_3/DIFFIO_RX_B4N
IO_3/DIFFIO_TX_RX_B5P
IO_3/DIFFIO_TX_RX_B5N
IO_3/DIFFIO_RX_B6P
IO_3/DIFFIO_RX_B6N
IO_3/DIFFIO_TX_RX_B7P
IO_3/DIFFIO_TX_RX_B7N
IO_3/DIFFIO_RX_B8P
IO_3/DIFFIO_RX_B8N
IO_3/DIFFIO_TX_RX_B9P
IO_3/DIFFIO_TX_RX_B9N
IO_3/DIFFIO_TX_RX_B10P
IO_3/DIFFIO_TX_RX_B10N
IO_3/DIFFIO_RX_B11P
IO_3/DIFFIO_RX_B11N
IO_3/DIFFIO_TX_RX_B12P
IO_3/DIFFIO_TX_RX_B12N
IO_3/DIFFIO_TX_RX_B14P
IO_3/DIFFIO_TX_RX_B14N
IO_3/DIFFIO_TX_RX_B16P
IO_3/DIFFIO_TX_RX_B16N
IO_3
10M08SAU169C8G
CPLD_DBG10
FPGA_TCK
CPLD_DBG11
CPLD_DBG9
CPLD_DBG8
CPLD_DBG5
CPLD_DBG7
FPGA_INIT
FPGA_DONE
CPLD_DBG4
CPLD_DBG6
FPGA_TDO
FPGA_TDI
K_TCK
K_TDI
CPLD_DBG3
CPLD_DBG1
FPGA_RX
K_JTAGEN
FPGA_TMS
CPLD_DBG0
CPLD_DBG2
K_TMS
U32F
BANK 8
VCCIO8
VREFB8N0
IO_8/DEV_CLRN/DIFFIO_RX_T16N
IO_8/DEV_OE/DIFFIO_RT18P
IO_8/CONFIG_SEL
I_8/NCONFIG
IO_8/CRC_ERROR/DIFFIO_RX_T22N
IO_8/NSTATUS/DIFFIO_RX_T24P
IO_8/CONF_DONE/DIFFIO_RX_T24N
IO_8
10M08SAU169C8G
Bank8
CPLD_CFG2
TE_TCK
TE_TDO
TE_TDI
TE_TMS
TE_UART_TX
M10_NSTATUS
M10_CONF_DONE
M_TX
M_RX
TE_UART_RX
M10_NCONF
Bank3
U32B
BANK 2
VCCIO2
VREFB2N0
IO_2/CLK0P/DIFFIO_RX_L18P
IO_2/CLK0N/DIFFIO_RX_L18N
IO_2/DIFFIO_RX_L19P
IO_2/DIFFIO_RX_L19N
IO_2/CLK1P/DIFFIO_RX_L20P
IO_2/CLK1N/DIFFIO_RX_L20N
IO_2/DIFFIO_RX_L21P
IO_2/DIFFIO_RX_L21N
IO_2/DPCLK1/DIFFIO_RX_L22P
IO_2/DPCLK0/DIFFIO_RX_L22N
IO_2/PLL_L_CLKOUTP/DIFFIO_RX_L27P
IO_2/PLL_L_CLKOUTN/DIFFIO_RX_L27N
IO_2/DIFFIO_RX_L28P
IO_2/DIFFIO_RX_L28N
IO_2
10M08SAU169C8G
Bank2
CPLD_CLK
HDMI_PD
VDAC_PSAVE_N
U32E
BANK 6
VCCIO6
VREFB6N0
IO_6/CLK2P/DIFFIO_RX_R14P
IO_6/CLK2N/DIFFIO_RX_R14N
IO_6/CLK3P/DIFFIO_RX_R16P
IO_6/CLK3N/DIFFIO_RX_R16N
IO_6/DIFFIO_RX_R18P
IO_6/DIFFIO_RX_R18N
IO_6/DPCLK3/DIFFIO_RX_R26P
IO_6/DPCLK2/DIFFIO_RX_R26N
IO_6/DIFFIO_RX_R27P
IO_6/DIFFIO_RX_R27N
IO_6/DIFFIO_RX_R28P
IO_6/DIFFIO_RX_R28N
IO_6/DIFFIO_RX_R29P
IO_6/DIFFIO_RX_R29N
IO_6/DIFFIO_RX_R30P
IO_6/DIFFIO_RX_R30N
IO_6/DIFFIO_RX_R31P
IO_6/DIFFIO_RX_R31N
IO_6/DIFFIO_RX_R33P
IO_6/DIFFIO_RX_R33N
IO_6
10M08SAU169C8G
KB_IO2
KB_TCK
KB_IO3
KB_TMS
KB_JTAGEN
KB_TDO
KB_IO1
LED_R
CPLD_CFG3
PMOD2_EN
CPLD_CFG1
LED_G
CPLD_CFG0
KB_TDI
PMOD1_FLG
PMOD1_EN
PMOD2_FLG
Bank6
KB
U32D
BANK 5
VCCIO5
VREFB5N0
IO_5/DIFFIO_RX_R1P
IO_5/DIFFIO_RX_R1N
IO_5/DIFFIO_RX_R2P
IO_5/DIFFIO_RX_R2N
IO_5/DIFFIO_RX_R7P
IO_5/DIFFIO_RX_R7N
IO_5/DIFFIO_RX_R8P
IO_5/DIFFIO_RX_R8N
IO_5/DIFFIO_RX_R9P
IO_5/DIFFIO_RX_R9N
IO_5/DIFFIO_RX_R10P
IO_5/DIFFIO_RX_R10N
IO_5/DIFFIO_RX_R11P
IO_5/DIFFIO_RX_R11N
IO_5
10M08SAU169C8G
DBG_UART_TX
DBG_UART_RX
FPGA_TX
K_TDO
K_IO2
K_IO3
FPGA_RESET_N
5V_JOY_PG
EN_5V_JOY_N
FPGA_PROG_B
K_IO1
Bank5
U32G
VCC_ONE
VCCA1
VCCA2
VCCA3
VCCA4
GND
10M08SAU169C8G
J21
Pin Header 2.54 2Row 14Pins
M10_NCONF
M10_CONF_DONE
M10_NSTATUS
R28
R29
R30
10K
D10
LED Green LTST-C191KGKT
D12
LED Red LTST-C191KRKT
R70
R130
240R
LED_G
LED_R
trenz electronic
Title: MEGA65
A4
Number: TE0765 Default
Rev. 02
Date: 2019-03-11
Copyright: Trenz Electronic GmbH
Page12 of 23
Filename: SYS_MAX10_CTRL.SchDoc



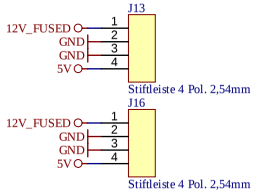
trenz electronic	Title: MEGA65		
	A4	Number: TE0765 Default	Rev. 02
	Date: 2019-03-11	Copyright: Trenz Electronic GmbH	Page14 of 23
	Filename: SOUND.SchDoc		

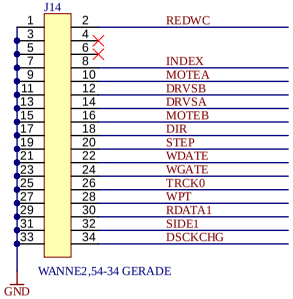


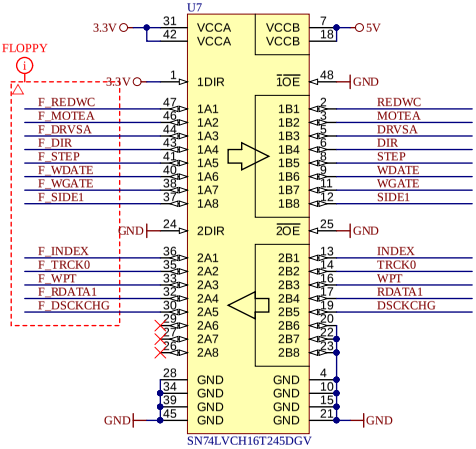
Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page15 of 23
Filename: JOY.SchDoc			



Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page16 of 23
Filename: KEYBOARD.SchDoc			

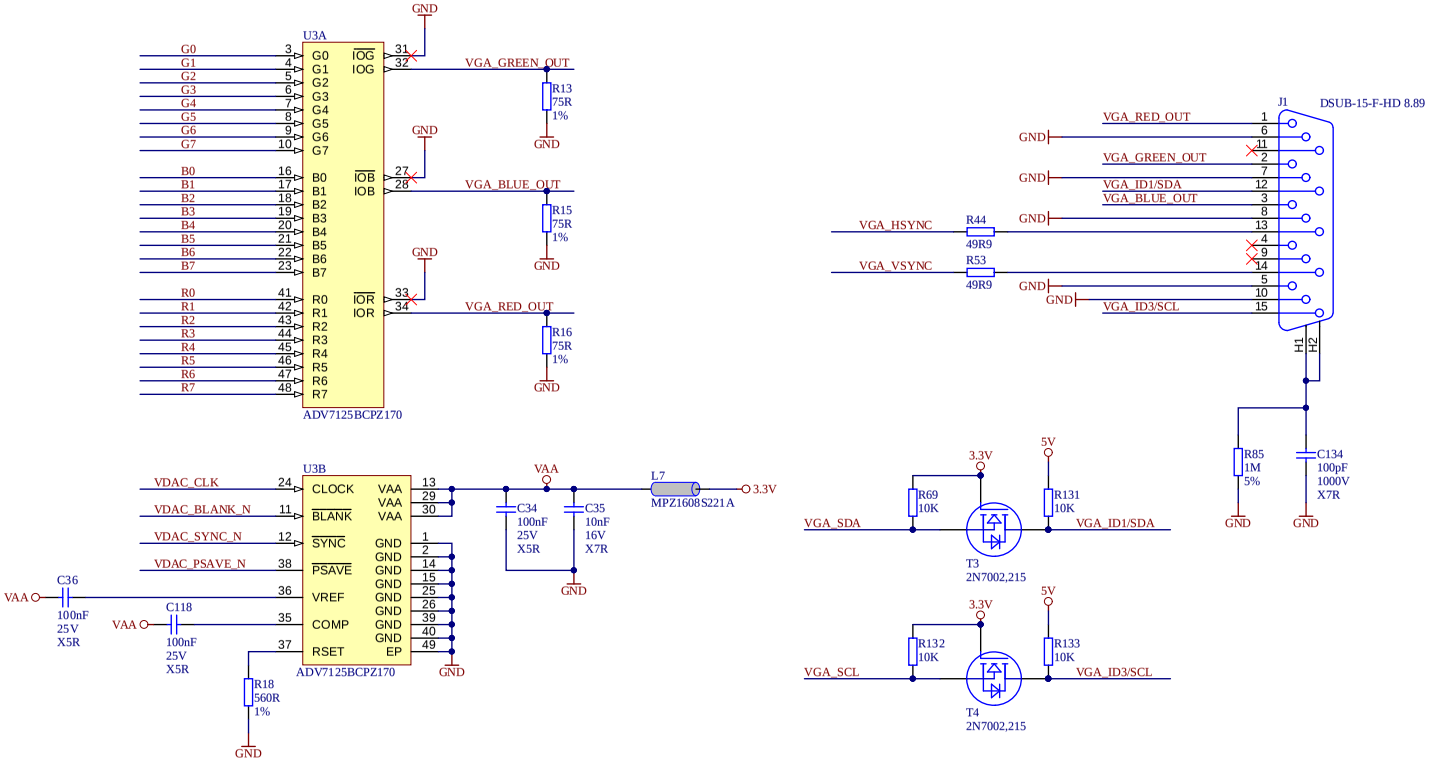




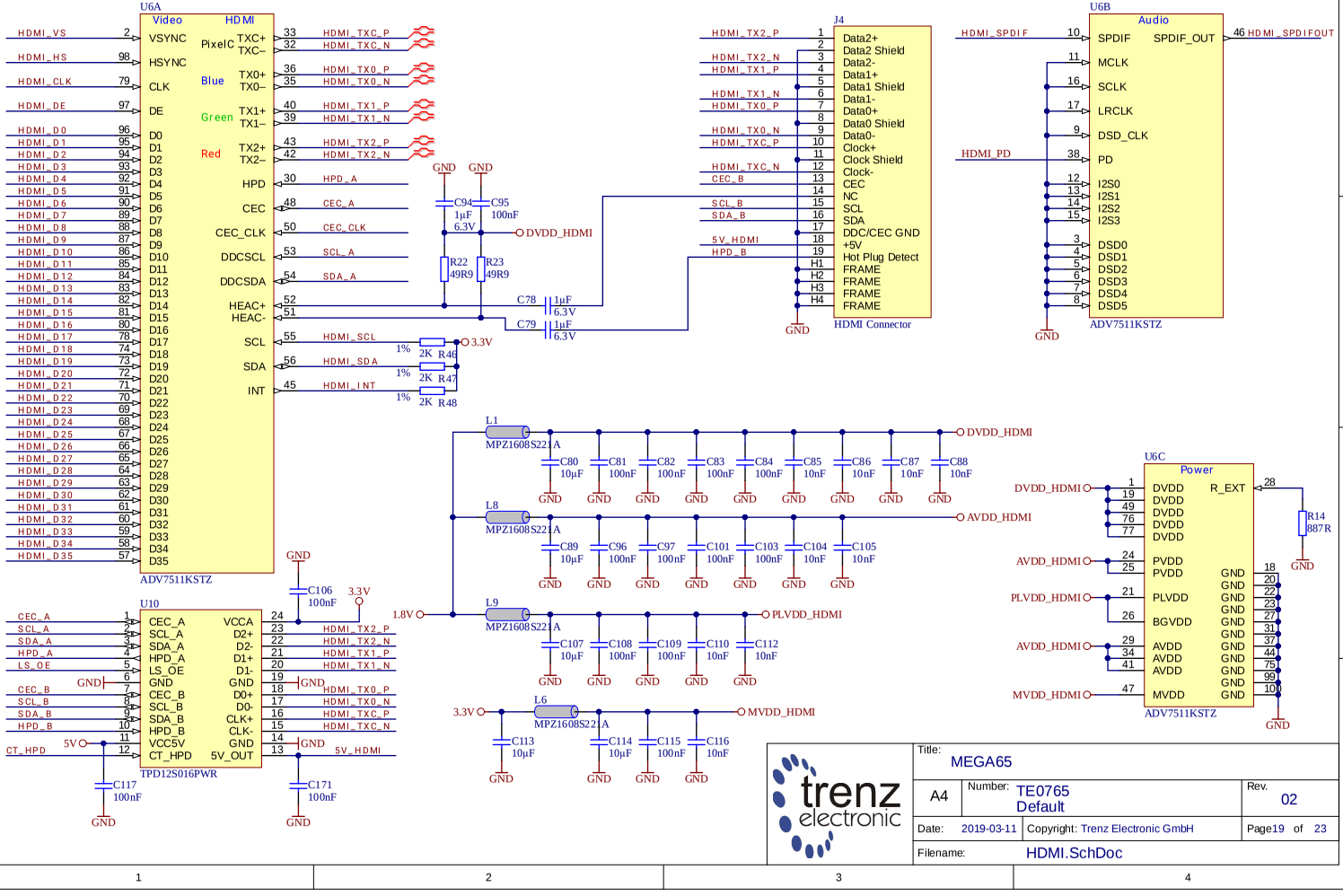


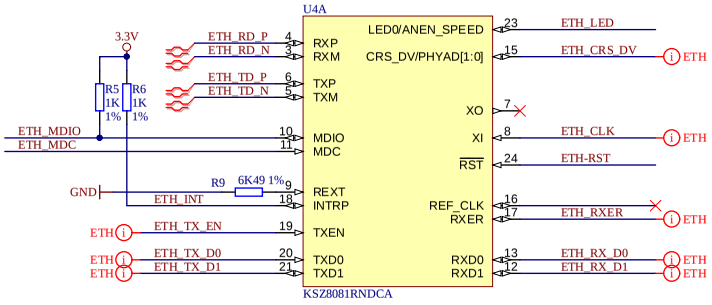
trenz electronic

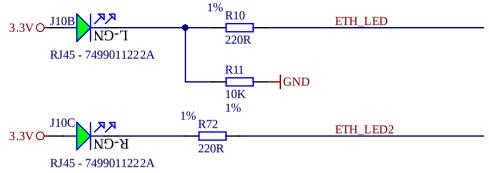
Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page17 of 23
Filename: Floppy.SchDoc			

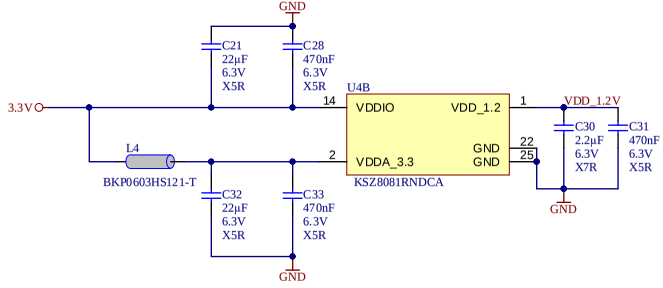


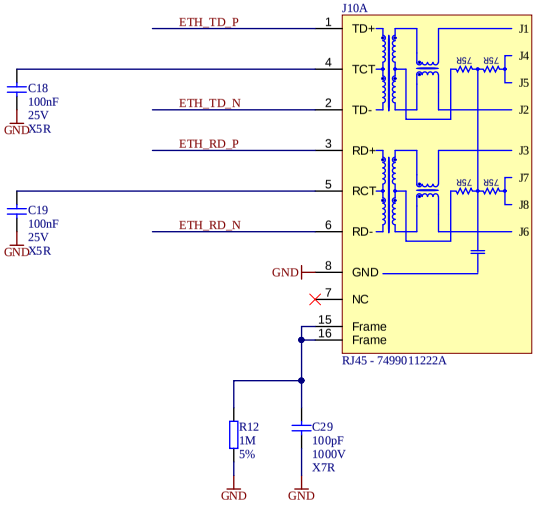
trenz electronic	Title: MEGA65		
	A4	Number: TE0765 Default	Rev. 02
	Date: 2019-03-11	Copyright: Trenz Electronic GmbH	Page18 of 23
	Filename: VGA.SchDoc		


U6A
Video
HDMI
VSYNC
HSYNC
CLK
DE
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
D16
D17
D18
D19
D20
D21
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
D32
D33
D34
D35
PixelC
Blue
Green
Red
TXC+
TXC–
TX0+
TX0–
TX1+
TX1–
TX2+
TX2–
HPD
CEC
CEC_CLK
DDCSCL
DDCSDA
HEAC+
HEAC-
SCL
SDA
INT
ADV7511KSTZ
HDMI_VS
HDMI_HS
HDMI_CLK
HDMI_DE
HDMI_D0
HDMI_D1
HDMI_D2
HDMI_D3
HDMI_D4
HDMI_D5
HDMI_D6
HDMI_D7
HDMI_D8
HDMI_D9
HDMI_D10
HDMI_D11
HDMI_D12
HDMI_D13
HDMI_D14
HDMI_D15
HDMI_D16
HDMI_D17
HDMI_D18
HDMI_D19
HDMI_D20
HDMI_D21
HDMI_D22
HDMI_D23
HDMI_D24
HDMI_D25
HDMI_D26
HDMI_D27
HDMI_D28
HDMI_D29
HDMI_D30
HDMI_D31
HDMI_D32
HDMI_D33
HDMI_D34
HDMI_D35
HDMI_TXC_P
HDMI_TXC_N
HDMI_TX0_P
HDMI_TX0_N
HDMI_TX1_P
HDMI_TX1_N
HDMI_TX2_P
HDMI_TX2_N
HPD_A
CEC_A
CEC_CLK
SCL_A
SDA_A
HDMI_SCL
HDMI_SDA
HDMI_INT
GND
C94 1µF 6.3V
C95 100nF
DVDD_HDMI
R22 49R9
R23 49R9
C78 1µF 6.3V
C79 1µF 6.3V
1% 2K R46
1% 2K R47
1% 2K R48
3.3V
J4
Data2+
Data2 Shield
Data2-
Data1+
Data1 Shield
Data1-
Data0+
Data0 Shield
Data0-
Clock+
Clock Shield
Clock-
CEC
NC
SCL
SDA
DDC/CEC GND
+5V
Hot Plug Detect
FRAME
FRAME
FRAME
FRAME
HDMI Connector
HDMI_TX2_P
HDMI_TX2_N
HDMI_TX1_P
HDMI_TX1_N
HDMI_TX0_P
HDMI_TX0_N
HDMI_TXC_P
HDMI_TXC_N
CEC_B
SCL_B
SDA_B
5V_HDMI
HPD_B
U6B
Audio
SPDIF
SPDIF_OUT
MCLK
SCLK
LRCLK
DSD_CLK
PD
I2S0
I2S1
I2S2
I2S3
DSD0
DSD1
DSD2
DSD3
DSD4
DSD5
ADV7511KSTZ
HDMI_SPDIF
HDMI_SPDIFOUT
HDMI_PD
U10
CEC_A
SCL_A
SDA_A
HPD_A
LS_OE
GND
CEC_B
SCL_B
SDA_B
HPD_B
VCC5V
CT_HPD
VCCA
D2+
D2-
D1+
D1-
GND
D0+
D0-
CLK+
CLK-
GND
5V_OUT
TPD12S016PWR
5V
5V_HDMI
C106 100nF
C117 100nF
C171 100nF
L1
MPZ1608S221A
L8
MPZ1608S221A
L9
MPZ1608S221A
L6
MPZ1608S221A
1.8V
C80 10µF
C81 100nF
C82 100nF
C83 100nF
C84 100nF
C85 10nF
C86 10nF
C87 10nF
C88 10nF
C89 10µF
C96 100nF
C97 100nF
C101 100nF
C103 100nF
C104 10nF
C105 10nF
C107 10µF
C108 100nF
C109 100nF
C110 10nF
C112 10nF
C113 10µF
C114 10µF
C115 100nF
C116 10nF
AVDD_HDMI
PLVDD_HDMI
MVDD_HDMI
U6C
Power
DVDD
PVDD
PLVDD
BGVDD
AVDD
MVDD
R_EXT
R14 887R
ADV7511KSTZ
trenz electronic
Title: MEGA65
A4
Number: TE0765 Default
Rev. 02
Date: 2019-03-11
Copyright: Trenz Electronic GmbH
Page19 of 23
Filename: HDMI.SchDoc



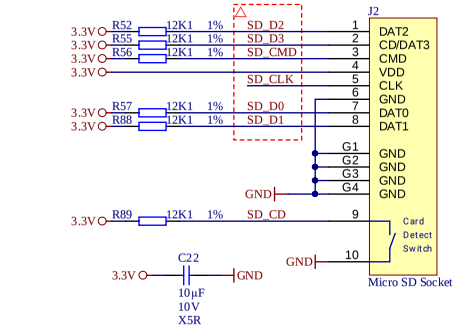


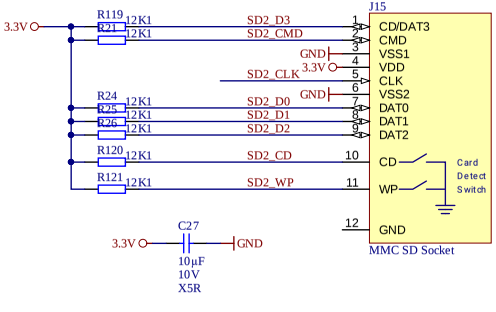




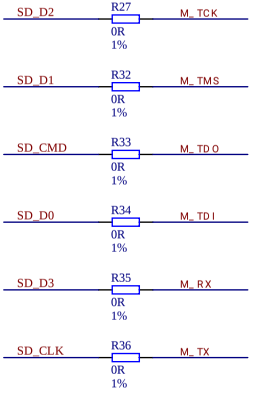
trenz electronic

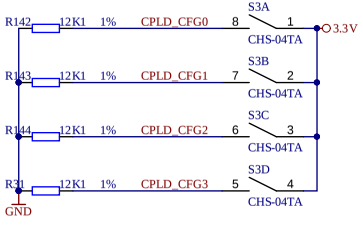
Title: MEGA65			
A4	Nummer: TE0765 Default		Rev. 02
Datum: 2019-03-11	Zeichner: Trenz Electronic GmbH		Blatt 20 von 23
Filename: Ethernet.SchDoc			



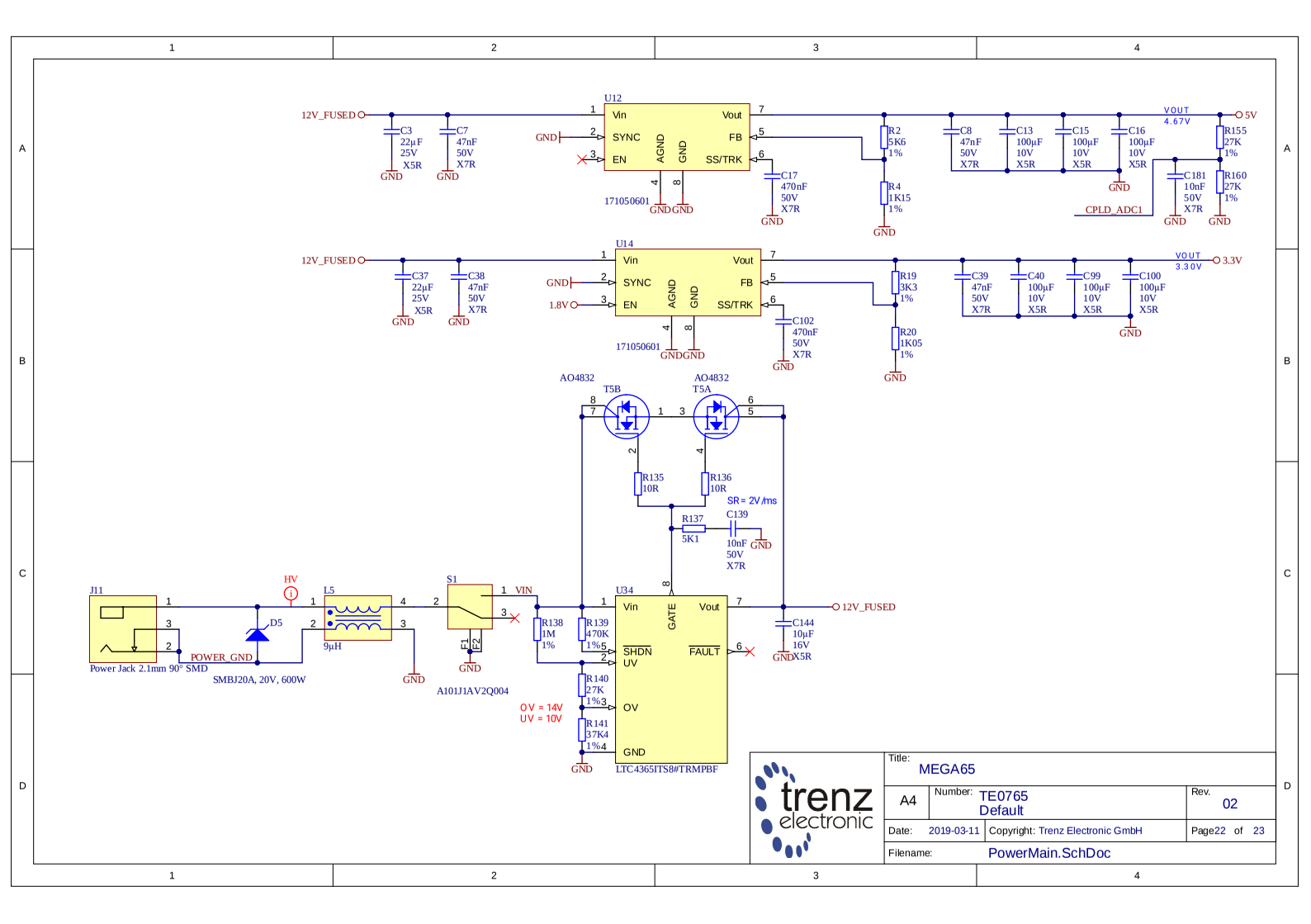


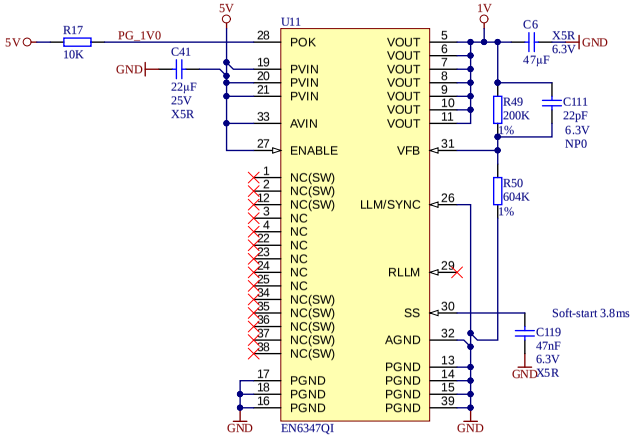
http://linux-sunxi.org/MicroSD_Breakout

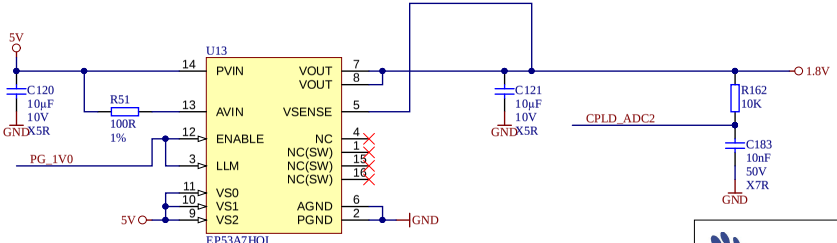


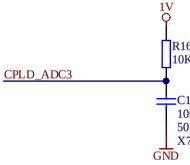


Title: MEGA65			
A4	Number: TE0765 Default		Rev. 02
Date: 2019-03-11	Copyright: Trenz Electronic GmbH		Page21 of 23
Filename: LED_SW_BUT.SchDoc			


12V_FUSED
U12
Vin
Vout
SYNC
FB
EN
AGND
GND
SS/TRK
171050601
C3 22µF 25V X5R
C7 47nF 50V X7R
C17 470nF 50V X7R
R2 5K6 1%
R4 1K15 1%
C8 47nF 50V X7R
C13 100µF 10V X5R
C15 100µF 10V X5R
C16 100µF 10V X5R
VOUT 4.67V
5V
R155 27K 1%
C181 10nF 50V X7R
R160 27K 1%
CPLD_ADC1
U14
1.8V
C37 22µF 25V X5R
C38 47nF 50V X7R
C102 470nF 50V X7R
R19 3K3 1%
R20 1K05 1%
C39 47nF 50V X7R
C40 100µF 10V X5R
C99 100µF 10V X5R
C100 100µF 10V X5R
VOUT 3.30V
3.3V
AO4832
T5B
T5A
R135 10R
R136 10R
SR = 2V/ms
R137 5K1
C139 10nF 50V X7R
J11
Power Jack 2.1mm 90° SMD
HV
D5
SMBJ20A, 20V, 600W
L5
9µH
S1
F1
F2
A101J1AV2Q004
VIN
R138 1M 1%
R139 470K 1%
U34
GATE
SHDN
UV
FAULT
OV
R140 27K 1%
R141 37K4 1%
0 V = 14V
UV = 10V
LTC4365ITS8#TRMPBF
C144 10µF 16V X5R
POWER_GND
Title: MEGA65
A4
Number: TE0765 Default
Rev. 02
Date: 2019-03-11
Copyright: Trenz Electronic GmbH
Page22 of 23
Filename: PowerMain.SchDoc
trenz electronic

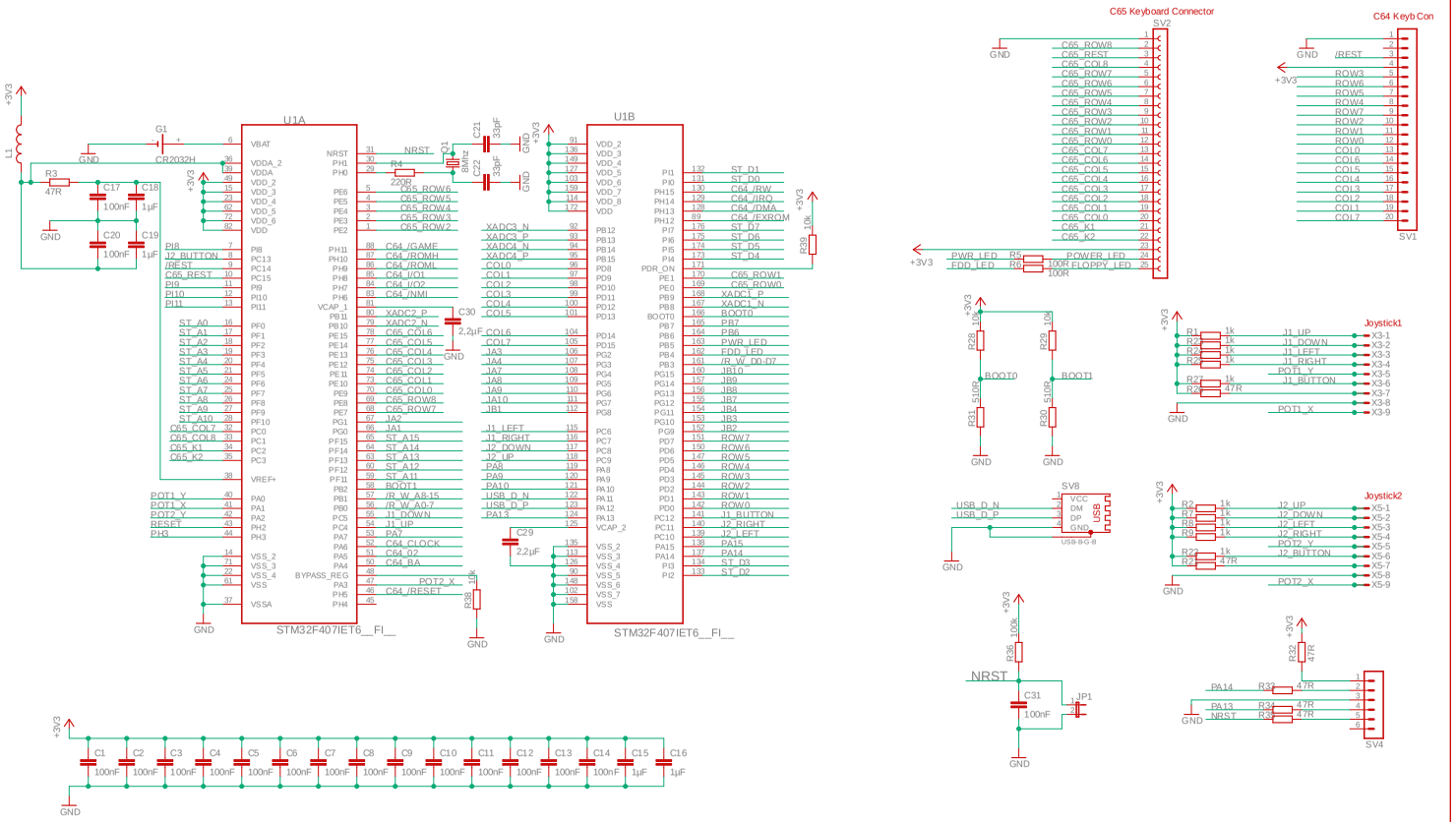




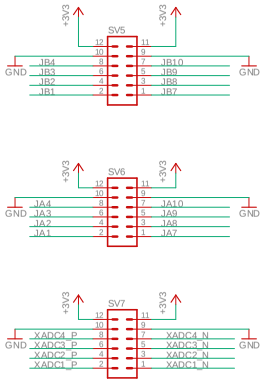


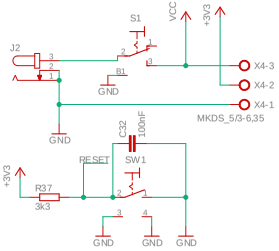
trenz electronic	Title: MEGA65		
	A4	Number: TE0765 Default	Rev. 02
	Date: 2019-03-11	Copyright: Trenz Electronic GmbH / TT	Page23 of 23
	Filename: POWER.SchDoc		

NEXYS WIDGET BOARD ELEKTRONISCH SCHEMA

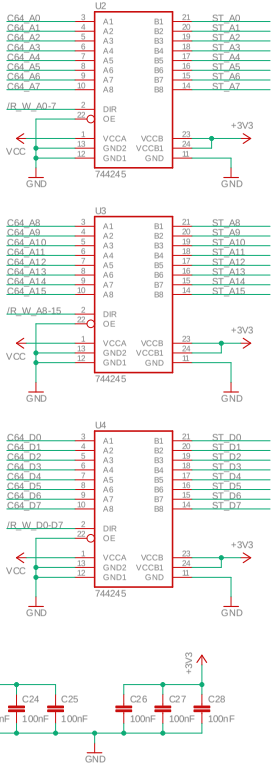

C65 Keyboard Connector
C64 Keyb Con
U1A
U1B
STM32F407IET6__FI__
STM32F407IET6__FI__
SV2
SV1
SV8
SV4
Joystick1
Joystick2
TITLE: c65Keyb
Document Number:
REV:
Date: 6/05/2022 10:40 AM
Sheet: 1/2

Nexys 4 Input

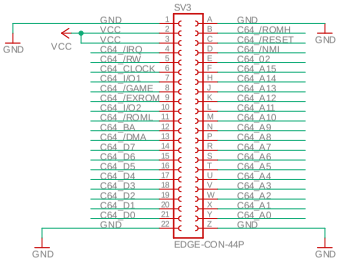


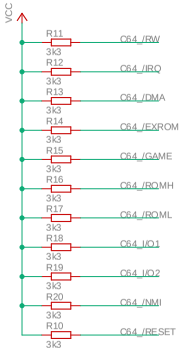


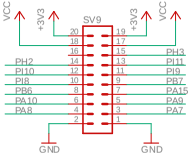
Output Driver



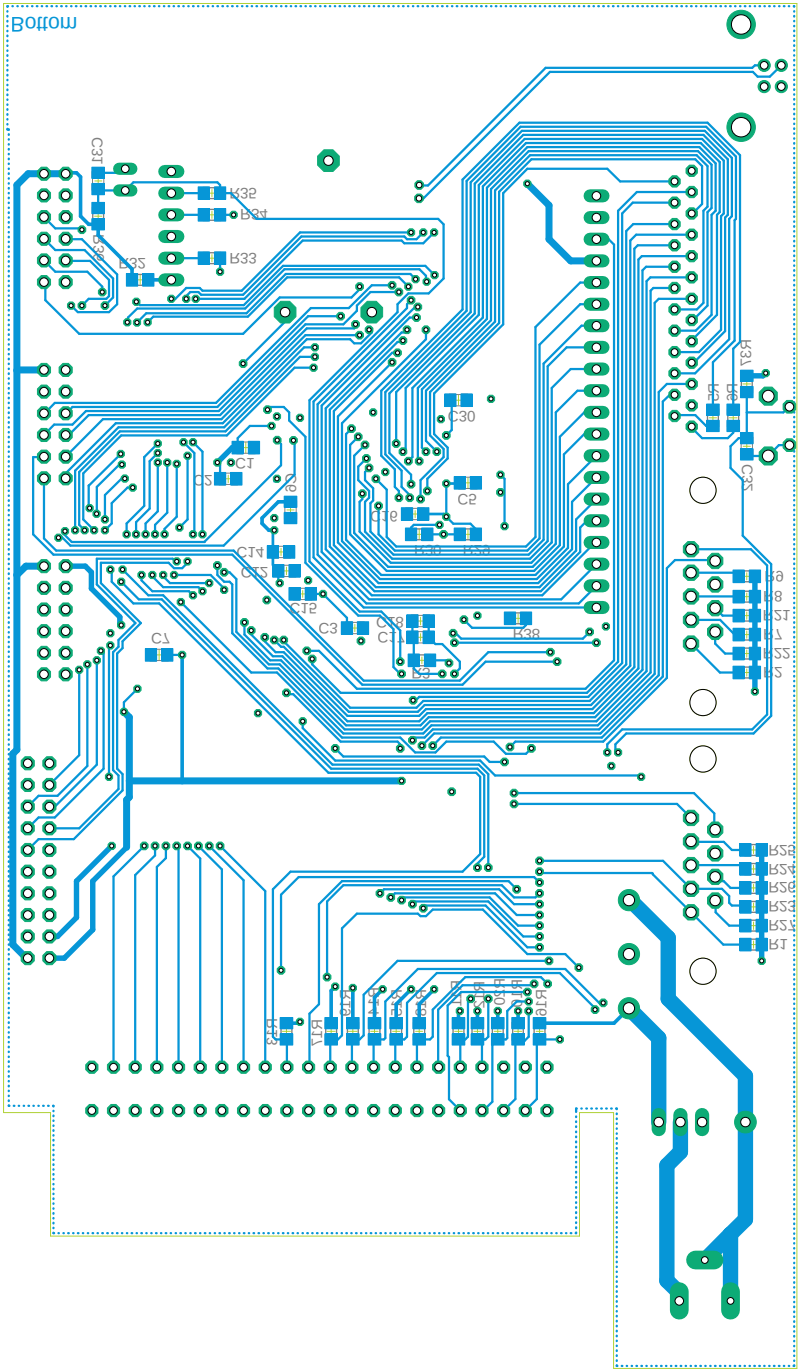
Expansion Slot







TITLE: c65Keyb		
Document Number:		REV:
Date: 6/05/2022 10:40 AM	Sheet: 2/2	


Bottom

BIJLAGE **AF**

Steuners & donateurs

- Organisaties
- Bijdragers
- Steuners

De MEGA65 zou niet tot stand zijn gekomen zonder de gulle steun van veel organisaties en individuen.

Wij stellen deze lijsten nog steeds samen; onze excuses indien u er nog niet in voorkomt. Indien u iemand kent die wij over het hoofd hebben gezien, laat het ons dan weten. Zo kunnen wij de bijdrage erkennen van iedereen die de MEGA65 mogelijk heeft gemaakt en die er het geweldige retrocomputingproject van heeft gemaakt dat de computer inmiddels is.

ORGANISATIES

Het MEGA Museum of Electronic Games & Art e.V. Duitsland
ALLES

Trenz Electronic, Duitsland
MOEDERBORD **FABRICAGE** **VERKOOP**

Hintsteiner, Oostenrijk
BEHUIZING

GMK, Duitsland
TOETSENBORD

KEVAG Telekom, Duitsland
WEBHOSTING

BIJDRAGERS

Andreas Liebeskind
(libi in paradize)
Financieel directeur MEGA eV

Thomas Hertzler
(grumpyninja)
Woordvoerder voor de VS

Russell Peake
(rdpeake)
Bugherder

Alexander Nik Petra
(n0d)
Vroeg ontwerp van de behuizing

Ralph Egas
(0-limits)
Zakelijk adviseur

Lucas Moss
PCB-ontwerp voor de MEGAphone

Daren Klamer
(Impakt)
Proeflezen van de handleiding

Daniël Mantione
(dmantione)
C64-hardwareguru

Dr. Canan Hastik
(indica)
Voorzitter MEGA eV

Simon Jameson
(Shallan)
Platformverbeteringen

Stephan Kleinert
(ubik)
Vernietiger van BASIC 10

Wayne Johnson
(sausage)
Aanvullingen op de handleiding

Raven L. A. Kleiss
(LAK132)
MegaWAT-presentatiesoftware

Maurice van Gils
(Maurice)
BASIC 65-voorbeeldprogramma's

Andrew Owen
(Cheveron)
Toetsenbord, Sinclair-ondersteuning

Adam Barnes
(amb5l)
HDMI-expert en printplaatherziening

Wayne Rittimann, Jr.
(johnwayner)
Bugverplettering op alle niveaus

STEUNERS

3c74ce64
8-Bit Classics
@11110110100
Aaron Smith
Achim Mrotzek
Adolf Nefischer
Adrian Esdaile
Adrien Guichard
Ahmed Kablaoui
Alan Bastian Witkowski
Alan Field
Alastair Paulin-Campbell
Alberto Mercuri
Alexander Haering
Alexander Kaufmann
Alexander Niedermeier
Alexander Soppart
Alfonso Ardire
Amiga On The Lake
André Kudra
André Simeit
André Wösten
Andrea Farolfi
Andrea Minutello
Andreas Behr
Andreas Freier
Andreas Grabski
Andreas Millinger
Andreas Nopper
Andreas Ochs
Andreas Wendel Manufaktur
Andreas Zschunke
Andrew Bingham
Andrew Dixon
Andrew Mondt
Andrzej Hłuchyj
Andrzej Sawiniec
Andrzej Śliwa
Anthony W. Leal
Arkadiusz Bronowicki
Arkadiusz Kwasny
Arnaud Léandre
Arne Drews
Arne Neumann
Arne Richard Tyarks
Axel Klahr
Balaz Ondrej
Barry Thompson
Bartol Filipovic
Benjamin Maas
Bernard Alaiz
Bernhard Zorn
Bieno Marti-Braitmaier
Bigby
Bill LaGrue
Bjoerg Stojalowski
Björn Johannesson
Bjørn Melbøe
Bo Goeran Kvamme
Boerge Noest
Bolko Beutner
Brett Hallen
Brian Gajewski
Brian Green
Brian Juul Nielsen
Brian Reiter
Bryan Pope
Burkhard Franke
Byron Goodman
Cameron Roberton (KONG)
Carl Angervall
Carl Danowski
Carl Stock
Carl Wall
Carlo Pastore
Carlos Silva
Carsten Sørensen
Cenk Miroglu Miroglu
Chang sik Park
Charles A. Hutchins Jr.
Chris Guthrey
Chris Hooper
Chris Stringer
Christian Boettcher
Christian Eick
Christian Gleinser
Christian Gräfe
Christian Heffner
Christian Kersting
Christian Schiller
Christian Streck
Christian Weyer
Christian Wyk
Christoph Haug
Christoph Huck
Christoph Pross
Christopher Christopher
Christopher Kalk
Christopher Kohlert
Christopher Nelson
Christopher Taylor
Christopher Whillock
Claudio Piccinini
Claus Skrepek
Collen Blijenberg
Constantine Lignos
Crnjaninja
Daniel Auger
Daniel Julien
Daniel Lobitz
Daniel O'Connor
Daniel Teicher
Daniel Tootill
Daniel Wedin
Daniele Benetti
Daniele Gaetano Capursi
Dariusz Szczesniak
Darrell Westbury
David Asenjo Raposo
David Dillard
David Gorgon
David Norwood
David Raulo
David Ross
de voughn accooe
Dean Scully
Dennis Jeschke
Dennis Schaffers
Dennis Schierholz

Dennis Schneck
denti
Dick van Ginkel
Diego Barzon
Dierk Schneider
Dietmar Krueger
Dietmar Schinnerl
Dirk Becker
Dirk Wouters
Domingo Fivoli
DonChaos
Donn Lasher
Douglas Johnson
Dr. Leopold Winter
Dusan Sobotka
Earl Woodman
Ed Reilly
Edoardo Auteri
Eduardo Gallardo
Eduardo Luis Arana
Eirik Juliussen Olsen
Emilio Monelli
EP Technical Services
Epic Sound
Erasmus Kuhlmann
ergoGnomik
Eric Hilaire
Eric Hildebrandt
Eric Hill
Eric Jutrzenka
Erwin Reichel
Espen Skog
Evangelos Mpouras
Ewan Curtis
Fabio Zanicotti
Fabrizio Di Dio
Fabrizio Lodi
FARA Gießen GmbH
FeralChild
First Choice Auto's
Florian Rienhardt
Forum64. de
Francesco Baldassarri
Frank Fechner
Frank Glaush
Frank Gulasch
Frank Haaland
Frank Hempel
Frank Koschel
Frank Linhares
Frank Sleeuwaert
Frank Wolf
FranticFreddie
Fredrik Ramsberg
Fridun Nazaradeh
Friedel Kropp
Garrick West
Gary Lake-Schaal
Gary Pearson
Gavin Jones
Geir Sigmund Straume
Gerd Mitlaender
Giampietro Albiero
Giancarlo Valente
Gianluca Girelli
Giovanni Medina
Glen Fraser
Glen R Perye III
Glenn Main
Gordon Rimac
GRANT BYERS
Grant Louth
Gregor Bubek
Gregor Gramlich
Guido Ling
Guido von Gösseln
Guillaume Serge
Gunnar Hemmerling
Günter Hummel
Guy Simmons
Guybrush Threepwood
Hakan Blomqvist
Hans Pronk
Hans-Jörg Nett
Hans-Martin Zedlitz
Harald Dosch
Harri Salokorpi
Harry Culpan
Harry Venema
Heath Gallimore
Heinz Roesner
Heinz Stampfli
Helge Förster
Hendrik Fensch
Henning Harperath
Henri Parfait
Henrik Kühn
Holger Burmester
Holger Sturk
Howard Knibbs
Hubert de Hollain
Huberto Kusters
Hugo Maria Gerardus v.d. Aa
Humberto Castaneda
Ian Cross
IDE64 Staff
Igor Ianov
Igor Kurtes
Immo Beutler
Ingo Katte
Ingo Keck
Insanely Interested Publishing
IT-Dienstleistungen Obsieger
Ivan Elwood
Jaap HUIJSMAN
Jace Courville
Jack Wattenhofer
Jakob Schönpflug
Jakub Tyszko
James Hart
James Marshburn
James McClanahan
James Sutcliffe
Jan Bitruff
Jan Hildebrandt
Jan Iemhoff
Jan Kösters
Jan Peter Borsje
Jan Schulze
Jan Stoltenberg-Lerche
Janne Tompuri
Jannis Schulte
Jari Loukasmäki
Jason Smith
Javier Gonzalez Gonzalez
Jean-Paul Lauque
Jeffrey van der Schilden
Jens Schneider

Jens-Uwe Wessling
Jesse DiSimone
Jett Adams
Johan Arneklev
Johan Berntsson
Johan Svensson
Johannes Fitz
John Cook
John Deane
John Dupuis
John Nagi
John Rorland
John Sargeant
John Traeholt
Jon Sandelin
Jonas Bernemann
Jonathan Prosise
Joost Honig
Jordi Pakey-Rodriguez
Jöre Weber
Jörg Jungermann
Jörg Schaeffer
Jörg Weese
Josef Hesse
Josef Soucek
Josef Stohwasser
Joseph Clifford
Joseph Gerth
Jovan Crnjanin
Juan Pablo Schisano
Juan S. Cardona Iguina
JudgeBeeb
Juliussen Olsen
Juna Luis Fernandez Garcia
Jürgen Endras
Jürgen Herm Stapelberg
Jyrki Laurila
Kai Pernau
Kalle Pöyhönen
Karl Lamford
Karl-Heinz Blum
Karsten Engstler
Karsten Westebbe
katarakt
Keith McComb
Kenneth Dyke
Kenneth Joensson
Kevin Edwards
Kevin Thomasson
Kim Jorgensen
Kim Rene Jensen
Kimmo Hamalainen
Konrad Buryło
Kosmas Einbrodt
Kurt Klemm
Lachlan Glaskin
Large bits collider
Lars Becker
Lars Edelmann
Lars Slivsgaard
Lasse Lambrecht
Lau Olivier
Lee Chatt
Loan Leray
Lorenzo Quadri
Lorenzo Travagli
Lorin Millsap
Lothar James Foss
Lothar Serra Mari
Luca Papinutti
Ludek Smetana
Lukas Burger
Lutz-Peter Buchholz
Luuk Spaetgens
Mad Web Skills
MaDCz
Magnus Wiklander
Maik Diekmann
Malte Mundt
Manfred Wittemann
Manuel Beckmann
Manzano Mérida
Marc "3D-vice" Schmitt
Marc Bartel
Marc Jensen
Marc Schmidt
Marc Theunissen
Marc Tutor
Marc Wink
Marcel Buchtmann
Marcel Kante
Marco Beckers
Marco Cappellari
Marco Rivela
Marco van de Water
Marcus Gerards
Marcus Herbert
Marcus Linkert
Marek Pernicky
Mario Esposito
Mario Fetka
Mario Teschke
Mariusz Tymków
Mark Adams
Mark Anderson
Mark Green
Mark Hucker
Mark Leitiger
Mark Spezzano
Mark Watkin
Marko Rizvic
Markus Bieler
Markus Bonet
Markus Dauberschmidt
Markus Fehr
Markus Fuchs
Markus Guenther-Hirn
Markus Liukka
Markus Merz
Markus Roesgen
Markus Uttenweiler
Martin Bauhuber
Martin Benke
Martin Gendera
Martin Groß
Martin Gutenbrunner
Martin Johansen
Martin Marbach
Martin Sonnleitner
Martin Steffen
Marvin Hardy
Massimo Villani
Mathias Dellacherie
Mathieu Chouinard
Matthew Adams
Matthew Browne
Matthew Carnevale
Matthew Palmer

Matthew Santos
Matthias Barthel
Matthias Dolenc
Matthias Fischer
Matthias Frey
Matthias Grandis
Matthias Guth
Matthias Lampe
Matthias Meier
Matthias Mueller
Matthias Nofer
Matthias Schonder
Maurice Al-Khaliedy
Max Ihlenfeldt
Meeso Kim
Michael Dailly
Michael Dötsch
Michael Dreßel
Michael Fichtner
Michael Fong
Michael Geoffrey Stone
Michael Gertner
Michael Grün
Michael Habel
Michael Härtig
Michael Haynes
Michael J Burkett
Michael Jensen
Michael Jurisch
Michael Kappelgaard
Michael Kleinschmidt
Michael Lorenz
Michael Mayerhofer
Michael Nurney
Michael Rasmussen
Michael Richmond
Michael Sachse
Michael Sarbak
Michael Schneider
Michael Scholz
Michael Timm
Michael Traynor
Michael Whipp
Michal Ursiny
Michele Chiti
Michele Perini
Michele Porcu
Miguel Angel Rodriguez Jodar
Mikael Lund
Mike Betz
Mike Kastrantas
Mike Pikowski
Mikko Hämäläinen
Mikko Suontausta
Mirko Roller
Miroslav Karkus
Morgan Antonsson
Moritz
Morten Nielsen
MUBIQUO APPS,SL
Myles Cameron-Smith
Neil Moore
Nelson
neoman
Nicholas Melnick
Nikolaj Brinch Jørgensen
Nils Andreas
Nils Eilers
Nils Hammerich
Nils77
Norah Smith
Norman King
Normen Zoch
Olaf Grunert
Ole Eitels
Oliver Boerner
Oliver Brüggmann
Oliver Graf
Oliver Smith
Olivier Bori
ONEPSI LLC
oRdYNe
Osaühing Trioflex
OSHA-PROS USA
Padawer
Patrick Becher
Patrick Bürckstümmer
Patrick de Zoete
Patrick Toal
Patrick Vogt
Paul Alexander Warren
Paul Gerhardt (KONG)
Paul Jackson
Paul Johnson
Paul Kuhnast (mindrail)
Paul Massay
Paul Westlake
Paul Woegerer
Pauline Brasch
Paulo Apolonia
Pete Collin
Pete of Retrohax.net
Peter Eliades
Peter Gries
Peter Habura
Peter Herklotz
Peter Huyoff
Peter Knörzer
Peter Leswell
Peter Weile
Petri Alvinen
Philip Marien
Philip Timmermann
Philipp Rudin
Pierre Kressmann
Pieter Labie
Piotr Kmiecik
Power-on.at
Przemysław Safonow
Que Labs
R Welbourn
R-Flux
Rafał Michno
Rainer Kappler
Rainer Kopp
Rainer Weninger
Ralf Griewel
Ralf Pöscha
Ralf Reinhardt
Ralf Schenden
Ralf Smolarek
Ralf Zenker
Ralph Bauer
Ralph Wernecke
Rédl Károly
Reiner Lanowski
Remi Veilleux
Riccardo Bianchi

Richard Englert
Richard Good
Richard Menedetter
Richard Sopuch
Rick Reynolds
Rico Gruninger
Rob Dean
Robert Bernardo
Robert Eaglestone
Robert Grasböck
Robert Miles
Robert Schwan
Robert Shively
Robert Tangmar
Robert Trangmar
Rodney Xerri
Roger Olsen
Roger Pugh
Roland Attila Kett
Roland Evers
Roland Schatz
Rolf Hass
Ronald Cooper
Ronald Hunn
Ronny Hamida
Ronny Preiß
Roy van Zundert
Rüdiger Wohlfromm
Ruediger Schlenter
Rutger WIllemsen
Sampo Peltonen
Sarmad Gilani
SAS74
Sascha Hesse
Scott Halman
Scott Hollier
Scott Robison
Sebastian Baranski
Sebastian Bölling
Sebastian Felzmann
Sebastian Lipp
Sebastian Rakel
Şemseddin Moldibi
Śeth Morabito
Shawn McKee
Siegfried Hartmann
Zytex Online Store
Sigurbjorn Larusson
Sigurdur Finnsson
Simon Lawrence
Simon Wolf
spreen.digital
Stefan Haberl
Stefan Kramperth
Stefan Richter
Stefan Schultze
Stefan Sonnek
Stefan Theil
Stefan Vrampe
Stefano Canali
Stefano Mozzi
Steffen Reiersen
Stephan Bielmann
Stephen Jones
Stephen Kew
Steve Gray
Steve Kurlin
Steve Lemieux
Steven Combs
Stewart Dunn
Stuart Marsh
Sven Neumann
Sven Stache
Sven Sternberger
Sven Wiegand
Szabolcs Bence
Tantrumedia Limited
Techvana Operations Ltd.
Teddy Turmeaux
Teemu Korvenpää
The Games Foundation
Thierry Supplisson
Thieu-Duy Thai
Thomas Bierschenk
Thomas Edmister
Thomas Frauenknecht
Thomas Gitzen
Thomas Gruber
Thomas Haidler
Thomas Jager
Thomas Karlsen
Thomas Laskowski
Thomas Marschall
Thomas Niemann
Thomas Scheelen
Thomas Schilling
Thomas Tahsin-Bey
Thomas Walter
Thomas Wirtzmann
Thorsten Knoll
Thorsten Nolte
Tim Krome
Tim Waite
Timo Weirich
Timothy Blanks
Timothy Henson
Timothy Prater
Tobias Butter
Tobias Heim
Tobias Köck
Tobias Lüthi
Tommi Vasarainen
Toni Ammer
Tore Olsen
Torleif Strand
Torsten Schröder
Tuan Nguyen
Uffe Jakobsen
Ulrich Hintermeier
Ulrich Nieland
Ulrik Kruse
Urban Lindeskog
Ursula Förstle
Uwe Anfang
Uwe Boschanski
Vedran Vrbanc
Verm Project
Wayne Rittimann
Wayne Sander
Wayne Steele
Who Knows
Winfried Falkenhahn
Wolfgang Becker
Wolfgang Stabla
Worblehat
www.patop69.net
Yan B
Zoltan Markus
Zsolt Zsila

Bibliografie

[1] N. Montfort, P. Baudoin, J. Bell, I. Bogost, J. Douglass, M. C. Marino, M. Mateas, C. Reas, M. Sample, en N. Vawter, *10 PRINT CHR $(205.5+ RND (1));: GOTO 10*. MIT Press, 2012.

[2] L. Soares en M. Stumm, "Flexsc: Flexible system call scheduling with exception-less system calls." in *Osdi*, dl. 10, 2010, blz. 1–8.

[3] Actraiser, "Vic-ii for beginners: Screen modes, cheaper by the dozen," 2013. [Online]. Beschikbaar: http://dustlayer.com/vic-ii/2013/4/26/vic-ii-for-beginners-screen-modes-cheaper-by-the-dozen

Index

, (komma), 10-22, 10-40
: (dubbele punt), 10-22, 10-40
<> (ongelijk aan), 10-31
$00 (PORTDDR), K-15
$00 (STOPTX), U-10
$01 (C64MAP), K-15
$01 (STARTTX), U-10
$D0 (RXNORMAL), U-10
$D4 (DEBUGVIC), U-11
$DC (DEBUGCPU), U-11
$DE (RXONLYONE), U-11
$F1 (FRAME1K), U-11
$F2 (FRAME2K), U-11

Aansluitingen
    IEC, 3-8, 7-3, 7-10
ADC, L-18, L-60, L-114
ADCQ, K-13, L-114
ALR, L-18
altpal, 21-8
Amiga™-stijl audio, P-17
ANC, L-19
AND, 10-44, L-19, L-60, L-115
ANDQ, K-13, L-116
Apple, 17-4
ARR, L-20
ASL, L-21, L-61
ASLQ, K-13, L-116
ASR, L-62
ASRQ, K-13, L-117
ASSEMBLE, O-6
ASW, L-62
audio-cross-bar-schakelaar, V-24
audiomixer, V-24
AUTOBOOT.C65-bestand, 7-11

BACKGROUND, 10-44
BASIC 65-arrays, B-7
BASIC 65-commando's
    APPEND, B-20
    AUTO, B-23
    BACKGROUND, 10-44, B-24
    BACKUP, 7-14, B-25
BANK, B-26
BEGIN, B-27
BEND, B-28
BLOAD, 15-19, B-29
BOOT, 7-11, 15-19, B-31
BORDER, 10-44, B-32
BOX, B-33
BSAVE, 15-19, B-35
BVERIFY, B-38
CATALOG, B-40
CHANGE, B-42
CHAR, B-44
CHARDEF, B-46
CHDIR, 7-7, 7-12, B-47
CIRCLE, B-49
CLOSE, B-51
CLR, B-52
CLRBIT, B-53
CMD, B-54
COLLECT, B-55
COLLISION, B-56
COLOR, B-57
CONCAT, B-58
CONT, 10-39, B-59
COPY, 7-14, B-60
CURSOR, B-63
CUT, B-64
DATA, B-65
DCLEAR, B-67
DCLOSE, B-68
DEF FN, B-71
DELETE, 7-14, B-72
DIM, B-74
DIR, 7-10, 7-12, 7-13, B-75
Directe modus, B-5
DISK, B-77
DLOAD, 7-12, 7-13, B-78
DMA, B-80
DMODE, B-81
DO, B-82
DOT, B-85
DPAT, B-86

DSAVE, 7-12, 7-13, B-90
DVERIFY, B-92
EDMA, B-95
ELLIPSE, B-98
ELSE, B-100
END, 10-39, B-102
ENVELOPE, B-103
ERASE, B-106
EXIT, B-108
FAST, B-110
FGOSUB, B-111
FGOTO, B-112
FILTER, B-113
FIND, B-114
FONT, B-117
FOR, 10-7, B-118
FOREGROUND, 10-44, B-119
FORMAT, 7-10, B-120
FREAD#, B-122
FREEZER, B-123
FWRITE#, B-124
GCOPY, B-125
GET, B-126
GET#, B-127
GETKEY, B-128
GO64, 4-15, B-129
GOSUB, B-130
GOTO, 10-46, B-131
GRAPHIC, B-132
HEADER, B-134
HELP, B-136
HIGHLIGHT, B-138
IF, 10-30, B-139
IMPORT, B-141
INFO, B-142
INPUT, 10-17, 10-24, B-143
INPUT#, B-145
INSTR, B-147
KEY, B-151
LET, 10-15, B-155
LINE, B-156
LINE INPUT, B-157
LINE INPUT#, B-158
LIST, 4-4, 10-11, 10-12, B-159
LOAD, B-161
LOADIFF, B-163
LOCK, B-165
LOOP, B-169
MEM, B-171
MERGE, B-173
MKDIR, B-175
MONITOR, B-177
MOUNT, 7-3, 7-6, B-178
MOUSE, B-180
MOVSPR, B-182
NEW, B-184
NEXT, B-185
OFF, B-187
ON, B-188
OPEN, B-190
PAINT, B-192
PALETTE, B-193
PASTE, B-195
PEEK, 15-6
PEN, B-197
PLAY, B-199
POKE, 15-6, 15-19
POLYGON, B-205
PRINT, 10-11, 15-6, B-208
PRINT USING, B-210
PRINT#, B-209
RCURSOR, B-213
READ, B-215
RECORD, B-216
REM, B-218
RENAME, 7-14, B-219
RENUMBER, 10-38, B-220
RESTORE, B-222
RESUME, B-223
RETURN, B-224
RMOUSE, B-227
RREG, B-233
RUN, 4-4, 7-11, 7-13, B-239
SAVE, 4-5, B-242

SAVEIFF, B-243
SCNCLR, B-244
SCRATCH, B-245
SCREEN, B-246
SET, 7-13, B-249
SETBIT, B-250
SLEEP, B-253
SOUND, B-254
SPEED, B-257
SPRCOLOR, B-258
SPRITE, B-259
SPRSAV, B-261
STEP, 10-13, B-264
STOP, 10-39, B-265
SYS, 15-19, B-268
TEMPO, B-274
THEN, 10-30, B-275
TO, B-278
TRAP, B-279
TROFF, B-280
TRON, B-281
TYPE, B-282
UNLOCK, B-283
UNTIL, B-284
USING, B-285
VERIFY, B-289
VIEWPORT, B-290
VOL, B-291
VSYNC, B-292
WAIT, 15-19, B-293
WHILE, B-294
WINDOW, B-295
BASIC 65-constanten, B-6
BASIC 65-functies
ABS, B-18
ASC, B-21
ATN, B-22
BUMP, B-37
CHR$, B-48
COS, B-62
DEC, B-69
DECBIN, B-70
ERR$, B-107
EXP, B-109
FN, B-71, B-116
FRE, B-121
HASBIT, B-133
HEX$, 15-5, B-137
INT, B-148
JOY, B-149
LEFT$, B-153
LEN, B-154
LOG, B-166
LOG10, B-167
LOG2, B-168
LPEN, B-170
MID$, B-174
MOD, B-176
PEEK, 15-19, B-196
PIXEL, B-198
POINTER, B-202
POKE, B-204
POS, B-206
POT, B-207
RCOLOR, B-212
RDISK, B-214
RGRAPHIC, B-225
RIGHT$, B-226
RND, 10-45, B-228
RPALETTE, B-229
RPEN, B-230
RPLAY, B-231
RPT$, B-232
RSPCOLOR, B-234
RSPEED, B-235
RSPPOS, B-236
RSPRITE, B-237
RSPRSYS, B-238
RWINDOW, B-241
SGN, B-251
SIN, B-252
SPC, B-256
SQR, B-262
STR$, B-266

STRBIN$, B-267
TAB, B-272
TAN, B-273
USR, B-287
VAL, B-288
WPEEK, B-296
WPOKE, B-297
BASIC 65-opdrachten
BANK, 15-18, J-4
DMA, J-4
BASIC 65-operatoren, B-9
AND, 10-44, B-19
NOT, B-186
OR, B-191
XOR, B-298
BASIC 65-systeemcommando's
EDIT, B-93
BASIC 65-systeemvariabelen
C, B-39
DS, B-88
DS$, B-89
DT$, B-91
EL, B-97
ER, B-104
ST, B-263
&, B-271
TI, B-276
TI$, B-277
BASIC 65-variabelen, B-6
DT, 3-8
TI$, 3-8
BASIC 65-voorbeelden
ABS, B-18, B-82, B-169, B-251, B-294
AND, B-19, B-27, B-37, B-144, B-146, B-150, B-186, B-189, B-207, B-217, B-255, B-260, B-277
APPEND, B-20
ASC, B-21
ATN, B-22
AUTO, B-23
BACKGROUND, B-24
BACKUP, 7-14, B-25
BANK, 12-8, B-26, B-53, B-133, B-164, B-196, B-203, B-204, B-250, B-270, B-293, B-297
BEGIN, B-27, B-28, B-101, B-140, B-241
BEND, B-27, B-28, B-101, B-140, B-241
BLOAD, B-30, B-183, B-214
BOOT, 7-11, B-31
BORDER, B-32, B-50, B-198, B-292
BOX, B-33, B-34, B-64, B-81, B-85, B-87, B-125, B-195
BSAVE, B-36
BUMP, B-37, B-56
BVERIFY, B-38
CATALOG, B-40, B-41
CHANGE, B-43
CHAR, B-45, B-99
CHARDEF, B-46
CHDIR, 7-7, 7-12, B-47
CHR, B-27, B-48, B-82, B-100, B-122, B-124, B-126, B-128, B-140, B-152, B-154, B-157, B-169, B-203, B-241, B-271, B-275, B-284
CIRCLE, B-50, B-99
CLOSE, B-45, B-50, B-51, B-54, B-64, B-81, B-85, B-87, B-99, B-108, B-125, B-127, B-132, B-156, B-164, B-192, B-194, B-195, B-197, B-198, B-205, B-225, B-229, B-230, B-243, B-244, B-248, B-290
CLR, B-52, B-132, B-164, B-183, B-194, B-197, B-225,

B-230, B-244, B-248, B-255, B-260, B-276, B-290
CLRBIT, B-53, B-133
CMD, B-54, B-190
COLLECT, B-55
COLLISION, B-56
COLOR, B-57, B-193
CONCAT, B-58
CONT, B-59
COPY, 7-14, B-61, B-278
COS, B-62, B-71, B-116, B-118, B-185, B-264
CURSOR, B-63, B-213, B-224, B-271
CUT, B-64, B-195
DATA, B-66, B-74, B-142, B-164, B-215, B-222, B-272
DCLEAR, B-67
DCLOSE, B-68, B-122, B-124, B-130, B-131, B-146, B-158, B-209, B-217, B-263
DEC, B-38, B-69
DECBIN, B-70
DEF, B-71, B-116, B-132, B-164, B-194, B-197, B-225, B-228, B-230, B-244, B-248, B-249, B-290
DELETE, 7-14, B-73
DIM, B-74, B-118, B-122, B-124, B-144, B-146, B-158, B-185, B-218, B-263
DIR, 7-12, 7-13, B-76, B-175, B-249
DISK, B-77, B-249
DLOAD, 7-3, 7-12, 7-13, B-79, B-173, B-214, B-249
DMA, 12-8, B-80
DMODE, B-81, B-87
DO, B-82, B-108, B-126, B-127, B-130, B-131, B-144, B-169, B-260, B-284, B-294
DOT, B-85, B-198
DPAT, B-87
DSAVE, 7-13, B-90, B-145
DVERIFY, B-92
EDMA, B-95, B-172
ELLIPSE, B-99
ELSE, B-100, B-101, B-113, B-140, B-255, B-275
END, B-56, B-66, B-74, B-97, B-100–B-102, B-105, B-107, B-112, B-126, B-128, B-130, B-131, B-140, B-223, B-224, B-235, B-272, B-275, B-279
ENVELOPE, B-103, B-201
ERASE, B-106
ERR, B-97, B-105, B-107, B-223, B-279
EXIT, B-108, B-127, B-144
EXP, B-109, B-136, B-223, B-279–B-281
FAST, B-110, B-249
FGOSUB, B-111
FGOTO, B-112
FILTER, B-113
FN, B-71, B-116, B-228
FONT, B-117
FOR, B-50, B-56, B-66, B-71, B-74, B-85, B-107, B-113, B-116, B-118, B-122, B-124, B-146, B-158, B-164, B-176, B-183, B-185, B-198, B-203, B-208, B-209, B-215, B-217, B-222–B-225, B-228, B-256, B-260, B-264, B-271, B-272, B-274, B-276–B-281, B-291, B-298

FORMAT, 7-10, B-120
FRE, B-121
FREAD, B-122, B-124
FREEZER, B-123
FWRITE, B-122, B-124
GCOPY, B-125
GET, B-27, B-82, B-108, B-126, B-127, B-169, B-198, B-284
GETKEY, B-45, B-50, B-64, B-81, B-85, B-87, B-99, B-107, B-125, B-128, B-132, B-156, B-192, B-194, B-195, B-197, B-205, B-248, B-270, B-290
GO64, 12-3, B-129
GOSUB, B-82, B-101, B-113, B-130, B-131, B-150, B-169, B-189, B-224, B-294
GOTO, B-27, B-59, B-111, B-113, B-126, B-128, B-130, B-131, B-144, B-146, B-150, B-158, B-189, B-221, B-231, B-235, B-251, B-263, B-274, B-278, B-291, B-292
GRAPHIC, B-132, B-164, B-194, B-197, B-225, B-230, B-244, B-248
HASBIT, B-133
HEADER, B-135
HELP, B-136
HEX, B-137, B-203, B-277
IF, B-19, B-27, B-28, B-37, B-50, B-88, B-89, B-97, B-100–B-102, B-105, B-107, B-108, B-113, B-126–B-128, B-130, B-131, B-140, B-144, B-146, B-150, B-158, B-186, B-189, B-191, B-198, B-206, B-217, B-227, B-231, B-235, B-241, B-255, B-260, B-263, B-265, B-270, B-274, B-275, B-291
IMPORT, B-141
INFO, B-142
INPUT, B-100, B-108, B-111, B-112, B-127, B-130, B-131, B-140, B-144, B-146, B-157, B-158, B-217, B-263, B-275
INSTR, B-147
INT, B-148, B-228
JOY, B-150, B-189
KEY, B-152, B-187, B-189
LEFT, B-153
LEN, B-27, B-82, B-154, B-169, B-284
LET, B-155
LINE, B-108, B-127, B-132, B-156–B-158, B-192, B-194, B-197, B-225, B-243, B-244, B-248, B-263
LIST, B-54, B-160, B-190
LOAD, 7-3, B-152, B-162, B-260, B-261
LOADIFF, B-164
LOCK, B-165
LOG, B-109, B-166
LOG10, B-167
LOG2, B-168
LOOP, B-82, B-108, B-126, B-127, B-130, B-131, B-144, B-169, B-260, B-284, B-294
LPEN, B-170
MAP, 12-8
MEM, B-172
MERGE, B-173

MID, B-174
MKDIR, B-175
MOD, B-107, B-176, B-271
MONITOR, B-177
MOUNT, 7-7, 7-8, B-179
MOUSE, B-181, B-187, B-189, B-198, B-227, B-249
MOVSPR, B-56, B-183, B-198, B-260
NEW, B-184
NEXT, B-50, B-56, B-66, B-71, B-74, B-85, B-97, B-105, B-107, B-113, B-116, B-118, B-122, B-124, B-146, B-158, B-164, B-176, B-183, B-185, B-198, B-203, B-208, B-209, B-215, B-217, B-222–B-225, B-228, B-256, B-260, B-264, B-271, B-272, B-274, B-276–B-281, B-291, B-298
NOT, B-186
OFF, B-145, B-152, B-179, B-181, B-187, B-198, B-224, B-227
ON, B-145, B-150, B-152, B-181, B-189, B-198, B-213, B-224, B-227, B-249, B-251
OPEN, B-51, B-54, B-108, B-127, B-132, B-164, B-190, B-194, B-197, B-225, B-230, B-244, B-248
OR, B-82, B-130, B-131, B-144, B-169, B-191, B-231, B-284
PAINT, B-192, B-194
PALETTE, B-50, B-132, B-164, B-192–B-194, B-197, B-244, B-248
PASTE, B-64, B-125, B-195
PEEK, 12-8, B-74, B-196, B-203, B-255, B-277
PEN, B-50, B-64, B-81, B-87, B-99, B-132, B-156, B-192, B-194, B-195, B-197, B-225, B-230, B-243, B-244, B-248, B-290
PIXEL, B-198
PLAY, B-103, B-113, B-200, B-201, B-231, B-274, B-291
POINTER, B-122, B-124, B-203
POKE, 12-5, 12-8, B-69, B-204, B-233, B-270
POLYGON, B-205
POS, B-206
POT, B-207
PRINT, B-18, B-19, B-21, B-22, B-27, B-28, B-37, B-48, B-51, B-52, B-54, B-56, B-57, B-59, B-62, B-63, B-66, B-69–B-71, B-74, B-88, B-89, B-91, B-95, B-97, B-100–B-102, B-105, B-107–B-109, B-111–B-113, B-116, B-118, B-121, B-122, B-124, B-126–B-128, B-130, B-131, B-133, B-136, B-137, B-140, B-144, B-146, B-150, B-153, B-154, B-157, B-158, B-166–B-168, B-170, B-174, B-176, B-185, B-186, B-189, B-191, B-196, B-203, B-206, B-208, B-209, B-211, B-213, B-215, B-217, B-222–B-230, B-232, B-233, B-235, B-238, B-241, B-252,

B-256, B-262–B-267, B-271–B-273, B-275–B-281, B-286–B-288, B-296, B-298
RCOLOR, B-212
RCURSOR, B-213
RDISK, B-214
READ, B-66, B-74, B-215, B-222, B-272
RECORD, B-217
REM, B-23–B-28, B-32, B-37, B-46–B-48, B-50, B-51, B-53, B-54, B-56, B-61, B-63, B-64, B-68, B-71, B-73, B-74, B-77, B-79, B-80, B-82, B-95, B-97, B-99, B-100, B-102, B-105–B-108, B-110–B-113, B-116, B-117, B-120, B-123, B-125–B-128, B-130–B-132, B-135, B-140, B-144–B-148, B-150, B-152, B-155, B-156, B-158, B-160, B-162, B-164, B-165, B-169, B-170, B-172, B-175, B-179, B-181, B-184, B-187, B-189, B-190, B-192–B-198, B-201, B-203–B-207, B-212, B-214, B-217–B-219, B-221, B-224, B-225, B-227–B-231, B-233–B-237, B-240, B-241, B-243–B-245, B-248–B-251, B-253, B-255–B-258, B-261, B-263, B-265, B-270, B-275–B-278, B-280, B-281, B-283, B-287, B-290, B-292–B-297
RENAME, 7-14, B-219
RENUMBER, B-221
RESTORE, B-152, B-164, B-184, B-222
RESUME, B-97, B-105, B-107, B-223, B-279
RETURN, B-56, B-101, B-111, B-113, B-130, B-131, B-150, B-189, B-224
RGRAPHIC, B-225
RIGHT, B-226
RMOUSE, B-181, B-198, B-227
RND, B-50, B-198, B-228
RPALETTE, B-229
RPEN, B-230
RPLAY, B-231, B-274, B-291
RPT, B-232
RREG, B-233
RSPCOLOR, B-234
RSPEED, B-235
RSPPOS, B-236, B-260
RSPRITE, B-237
RSPRSYS, B-238
RUN, 7-13, B-52, B-59, B-66, B-71, B-74, B-116, B-136, B-146, B-157, B-174, B-203, B-213, B-217, B-225, B-229, B-230, B-240, B-256, B-266, B-272, B-277, B-280, B-281, B-287
RWINDOW, B-241
SAVE, B-152, B-242
SAVEIFF, B-243
SCNCLR, B-50, B-63, B-107, B-132, B-164, B-183, B-224, B-225, B-230, B-244, B-260
SCRATCH, B-245

SCREEN, B-45, B-50, B-64, B-81, B-85, B-87, B-99, B-125, B-132, B-156, B-164, B-172, B-192, B-194, B-195, B-197, B-198, B-205, B-225, B-229, B-230, B-243, B-244, B-248, B-290
SET, B-132, B-164, B-194, B-197, B-225, B-230, B-244, B-248, B-249
SETBIT, B-133, B-181, B-250
SGN, B-251
SIN, B-71, B-116, B-118, B-185, B-252, B-264
SLEEP, B-63, B-113, B-164, B-183, B-224, B-244, B-248, B-253, B-255, B-260
SOUND, B-255
SPC, B-256
SPEED, B-142, B-257
SPRCOLOR, B-258
SPRITE, B-56, B-183, B-234, B-236–B-238, B-258, B-260, B-261
SPRSAV, B-261
SQR, B-97, B-105, B-107, B-208, B-209, B-211, B-262, B-265, B-286
STEP, B-71, B-113, B-116, B-118, B-185, B-217, B-264, B-274, B-277, B-291
STOP, B-88, B-89, B-108, B-127, B-146, B-215, B-227, B-263, B-265
STR, B-266
STRBIN, B-267
SYS, B-31, B-129, B-233, B-238, B-270
TAB, B-272
TAN, B-118, B-185, B-264, B-273
TEMPO, B-103, B-201, B-274, B-291
THEN, B-19, B-27, B-28, B-37, B-50, B-88, B-89, B-97, B-100–B-102, B-105, B-107, B-108, B-113, B-126–B-128, B-130, B-131, B-140, B-144, B-146, B-150, B-158, B-186, B-189, B-191, B-198, B-206, B-217, B-227, B-231, B-235, B-241, B-255, B-260, B-263, B-265, B-270, B-274, B-275, B-291
TO, B-25, B-36, B-43, B-50, B-56, B-58, B-61, B-66, B-71, B-74, B-85, B-107, B-113, B-116, B-118, B-122, B-124, B-146, B-158, B-164, B-176, B-183, B-185, B-198, B-203, B-208, B-209, B-215, B-217, B-219, B-222–B-225, B-228, B-249, B-256, B-260, B-264, B-270–B-272, B-274, B-276–B-281, B-291, B-298
TRAP, B-97, B-105, B-107, B-223, B-279
TROFF, B-280, B-281
TRON, B-280, B-281
TYPE, B-146, B-282
UNLOCK, B-283
UNTIL, B-82, B-126, B-130, B-131, B-144, B-169, B-260, B-284
USING, B-71, B-107, B-116, B-211, B-277, B-286

USR, B-287
VAL, B-288
VERIFY, B-249, B-289
VIEWPORT, B-85, B-290
VOL, B-103, B-201, B-274, B-291
VSYNC, B-292
WAIT, B-293
WHILE, B-82, B-169, B-294
WINDOW, B-295
WPEEK, B-203, B-296
WPOKE, B-287, B-297
XOR, B-298
BASIC Commands
MEM, 15-14
BBR0, L-63
BBR1, L-63
BBR2, L-64
BBR3, L-64
BBR4, L-64
BBR5, L-65
BBR6, L-65
BBR7, L-66
BBS0, L-66
BBS1, L-66
BBS2, L-67
BBS3, L-67
BBS4, L-67
BBS5, L-68
BBS6, L-68
BBS7, L-68
BCC, L-21, L-69
BCS, L-21, L-69
Behuizing
Aansluitingen, 3-4
Opening, 3-9
BEQ, L-22, L-69
bgcolor, 21-7
BIT, L-22, L-70
BITMAPS, O-8
BITQ, K-13, L-118
blink, 21-8
BMI, L-23, L-70
BNE, L-23, L-71
BORDER, 10-44
bordercolor, 21-7
box, 21-11
BPL, L-23, L-71
BRA, L-72
BREAKPOINT, O-20
BRK, L-24, L-72
BSR, L-73
BVC, L-24, L-73
BVS, L-25, L-73

C64MAP, K-15
cellcolor, 21-9
cgetc, 21-19
cinput, 21-20
CLC, L-25, L-74
CLD, L-25, L-74
CLE, L-75
clearattr, 21-9
CLI, L-26, L-75
clrscr, 21-5
CLV, L-26, L-75
CMP, L-26, L-46, L-76, L-119
CMPQ, L-119
Commodore 1351-muis, 3-3, 5-4
Commodore 64
Core, 4-16
GO64-modus, 4-6, 4-7, 4-15, B-129
Commodore Amiga-muis, 3-3, 5-4
COMPARE, O-9
Configuratie, 5-3
Audio, 5-6
Hulpprogramma, 4-7, 5-3, 7-3, 7-11
Introductie, 3-11, 5-6
MAC-adres, 5-7
Muis, 5-4
Netwerk, 5-7
Opstartschijf, 5-4, 5-5
Real-time klok, 5-5

Video, 5-6
conionit, 21-3
CONT, 10-39
contextafhankelijk, 10-19
Core, 6-5
C64-core, 4-16, 6-13
Corekeuzemenu, 6-9
Fabriekscore in slot 0 upgraden, 6-16
Upgraden, 6-10
Versies, 6-7
Wat is een core?, 6-5
CPQ, K-13
cprintf, 21-18
CPUHISTORY, O-25
CPUMEMORY, O-20
cputc, 21-16
cputcxy, 21-15, 21-18
cputdec, 21-17
cputhex, 21-16
cputnc, 21-16
cputncxy, 21-18
cputs, 21-17
cputsxy, 21-17
CPX, L-27, L-76
CPY, L-28, L-77
CPZ, L-78
cross-bar-schakelaar, audio, V-24

D64-schijfimage, 7-3
D81-schijfimage, 7-3
DCP, L-28
DEBUGCPU, U-11
DEBUGMON, O-22
DEBUGVIC, U-11
DEC, L-29, L-78
DEQ, K-13, L-120
DEW, L-79
DEX, L-29, L-79
DEY, L-30, L-80
DEZ, L-80
Digitale audio, P-17
Digitale video, 3-7
digitale video, Q-9
DIP-schakelaars, 8-6
Directe modus, 10-18
DISASSEMBLE, O-9, O-21
DMA
Inline-DMA-lijsten, P-16
DMA-audio, P-17
Drivenummer (IEC-apparaten), 7-3

Eigenaarscode, 6-8
emulator, 17-3
END, 10-39
EOM, L-81
EOR, L-30, L-82, L-121
EORQ, K-13, L-121
EXIT, O-14
Extra Ignored, 10-22

F011-schijfcontroller, 5-5
Field Programmable Gate Array (FPGA), 6-5
Filehost-website, 5-13, 6-7, 7-4, 8-4
FILL, O-10, O-21
fillrect, 21-11
FLAGWATCH, O-21
Flashmenu, M-5
Floppy-DMA, V-11
flushkeybuf, 21-20
FOR, 10-7
FOREGROUND, 10-44
Fouten
Extra Ignored, 10-22
Illegal Direct, 10-18
Syntax, 10-4
Type mismatch, 10-17
FRAME1K, U-11
FRAME2K, U-11
Freezer-menu, 4-5, 4-13, 6-5, 6-6, 7-3, 7-4, 7-6, 7-7, 7-9, 7-10, M-5

geblokkeerd, 10-21
Geheugenbanking, K-7-K-9

getcharsetaddr, 21-5
getcolramoffset, 21-4
getkeymodstate, 21-20
getmapedpal, 21-10
getpalbank, 21-10
getpalbanka, 21-10
getscreenaddr, 21-4
getscreensize, 21-5
GO, O-11
gohome, 21-13
GOTO, 10-46
gotox, 21-13
gotoxy, 21-13
gotoy, 21-13

Hardware-revisies, 6-7
HELP, O-20
Hete registers, Q-17
highlight, 21-8
hline, 21-12
Hoofdmap, N-17
Hulpprogrammamenu, 4-7, 5-3, 5-10, M-5
HUNT, O-11
HYPERTRAP, O-19
Hyppo-diensten, N-3
$D640 $00, N-13
$D640 $02, N-29
$D640 $04, N-27
$D640 $06, N-41
$D640 $08, N-30
$D640 $0A, N-28
$D640 $0C, N-15
$D640 $0E, N-33
$D640 $10, N-40
$D640 $12, N-34
$D640 $14, N-36
$D640 $16, N-19
$D640 $18, N-35
$D640 $1A, N-38
$D640 $1C, N-43
$D640 $1E, N-33
$D640 $20, N-20
$D640 $22, N-18
$D640 $24, N-40
$D640 $26, N-40
$D640 $28, N-26
$D640 $2A, N-40
$D640 $2C, N-21
$D640 $2E, N-42
$D640 $30, N-23
$D640 $32, N-24
$D640 $34, N-22
$D640 $36, N-31
$D640 $38, N-11
$D640 $3A, N-14
$D640 $3C, N-17
$D640 $3E, N-32
$D640 $40, N-46
$D640 $42, N-47
$D640 $44, N-47
$D640 $46, N-46
$D640 $48, N-50
$D640 $4A, N-45
$D640 $50, N-52
$D640 $52, N-56
$D640 $54, N-52
$D640 $56, N-63
$D640 $58, N-52
$D640 $60, N-60
$D640 $62, N-48
$D640 $64, N-52
$D640 $66, N-48
$D640 $68, N-48
$D640 $6A, N-48
$D640 $6C, N-60
$D640 $6E, N-48
$D640 $70, N-62
$D640 $72, N-61
$D640 $74, N-49
$D640 $76, N-59
$D640 $7C, N-58
$D640 $7E, N-53
$D641 $00, N-55
$D641 $02, N-54

$D642 $00, N-64
$D642 $02, N-64
$D642 $04, N-64
$D642 $06, N-64
$D642 $10, N-65
$D642 $12, N-65
$D642 $14, N-65
$D642 $16, N-65
$D643 $xx, N-57
$D67F $xx, N-65
Hyppo-foutcodes, N-5, N-11
$01, N-11
$02, N-11
$03, N-11
$04, N-11
$05, N-11
$06, N-11
$07, N-11
$08, N-11, N-36
$10, N-11, N-14, N-36, N-42, N-49, N-51, N-59
$11, N-11
$20, N-12
$21, N-12
$80, N-12, N-41
$81, N-12, N-42, N-45
$82, N-12
$83, N-12
$84, N-12, N-31, N-32, N-34, N-35
$85, N-12, N-36
$86, N-12, N-35
$87, N-12, N-15, N-34
$88, N-12, N-22-N-24, N-31, N-32, N-45
$89, N-12, N-38, N-43
$8A, N-12, N-45
$8B, N-12, N-45
$8C, N-12
$8D, N-12, N-33, N-40
$8E, N-12
$8F, N-12, N-45
$FF, N-12
Hyppo: verplaatsen naar hoofdmap, N-17

I/O
blokkeren, 10-21
IEC-controlleropdrachten, W-4
IF, 10-30
Illegal Direct Error, 10-18
IMDH™, Q-9
INC, L-31, L-82
Inline-DMA-lijsten, P-16
INPUT, 10-17, 10-24
INQ, K-13, L-122
Integrated Marvellous Digital Hookup™, Q-9
INTERRUPTS, O-22
Intro Disk, 3-13, 5-5, 7-3
Uitschakelen, 3-14
INW, L-83
INX, L-31, L-83
INY, L-32, L-84
INZ, L-84
ISC, L-32

JMP, L-33, L-85
Joystick, 3-3
JSR, L-33, L-85
JUMP, O-11

kbhit, 21-19
KERNAL-sprongtabel
ACPTR, I-8
ADDKEY, I-9
BASIN, I-10
BOOT_SYS, I-12
BSOUT, I-13
C64MODE, I-14
CHKIN, I-15
CINT, I-16
CIOUT, I-17
CKOUT, I-18
CLALL, I-19

CLOSE, I-20
CLOSE_ALL, I-22
CLRCH, I-23
CMP_FAR, I-24
CURSOR, I-25
GETIN, I-26
GETIO, I-27
GETLFS, I-28
IOBASE, I-29
IOINIT, I-30
JMPFAR, I-31
JSRFAR, I-32
KEY, I-34
LDA_FAR, I-35
LISTN, I-36
LKUPLA, I-37
LKUPSA, I-38
LOAD, I-39
MEMBOT, I-41
MEMTOP, I-42
MONEXIT, I-43
MonitorCall, I-44
OPEN, I-45
PFKEY, I-49
PHOENIX, I-51
PLOT, I-52
PRIMM, I-54
RAMTAS, I-55
RDTIM, I-56
READSS, I-58
RESET_RUN, I-59
RESTOR, I-60
SAVE, I-61
SAVEFL, I-64
ScanStopKey, I-65
SCRORG, I-66
SECND, I-67
SETBNK, I-68
SETLFS, I-70
SETMSG, I-71
SETNAM, I-72
SETTIM, I-73
SPIN_SPOUT, I-74
STA_FAR, I-75
STOP, I-76
SWAPPER, I-77
SYSFLAGS, I-78
TALK, I-79
TKSA, I-80
UNLSN, I-81
UNTLK, I-82
VECTOR, I-83
VERSIONQ, I-89
KIL, L-34

LAS, L-34
LAX, L-35
LDA, L-36, L-86, L-123
LDQ, K-13, L-123
LDX, L-37, L-87
LDY, L-37, L-87
LDZ, L-88
LET, 10-15
lichtpen, V-23
Lijntekening, P-12
DMA-optiebytes, P-12
LIST, 10-11, 10-12
LOAD, O-12
LOADMEMORY, O-23
LSR, L-38, L-88
LSRQ, K-13, L-124

M65Connect-toepassing, 8-4, 8-8
Machinetaalmonitor, 4-4
MONITOR-commando, B-177
macOS, 17-4
MAP, K-7-K-9, L-89
MEGA Flash, M-5
MEGA65-informatiehulpprogramma, 6-6
mega65_ftp-commando, 8-10
MEMORY, O-12, O-23
mixer, audio, V-24
MOD-stijl audio, P-17
MONITOR

Matrix Mode/seriële monitor, O-16
MEGA65-monitor, O-3
MONITOR-commando's, O-15
., O-15
>, O-15
ASSEMBLE, O-6
BITMAPS, O-8
BREAKPOINT, O-20
COMPARE, O-9
CPUHISTORY, O-25
CPUMEMORY, O-20
DEBUGMON, O-22
DISASSEMBLE, O-9, O-21
EXIT, O-14
FILL, O-10, O-21
FLAGWATCH, O-21
GO, O-11
HELP, O-20
HUNT, O-11
HYPERTRAP, O-19
INTERRUPTS, O-22
JUMP, O-11
LOAD, O-12
LOADMEMORY, O-23
MEMORY, O-12, O-23
REGISTERS, O-13, O-23
SAVE, O-13
SETMEMORY, O-24
SETPC, O-22
STEP, O-18
TRACE, O-24
TRANSFER, O-14
UARTDIVISOR, O-19
VERIFY, O-14
WATCHPOINT, O-24
mouSTer-adapter, 3-3, 5-4
movedown, 21-14
moveleft, 21-14
moveright, 21-14
moveup, 21-14

naamruimten, 10-17
NEG, L-90
Netwerk
Ethernet, 8-3
Luistermodus, 8-5
MAC-adres, 5-7, 8-4
niet gelijk aan, 10-31
NO SCROLL, 10-46
NOP, L-38
NTSC-schermmodus, 3-11

ongelijkheid, 10-31
OpenROMs, 17-6
ORA, L-39, L-90, L-125
ORQ, K-13, L-125

PAL-schermmodus, 3-11
pcprintf, 21-19
pcputc, 21-15
pcputs, 21-16
pcputsxy, 21-15
PETSCII, 4-3, 4-9, 4-12
PHA, L-40, L-91
PHP, L-40, L-91
PHW, L-92
PHX, L-92
PHY, L-92
PHZ, L-93
Pi1541-apparaat, 3-8
PLA, L-41, L-93
PLP, L-41, L-93
PLX, L-94
PLY, L-94
PLZ, L-94
PORTDDR, K-15
PRINT, 10-11
Programma's
bewerken, 10-25
regels vervangen, 10-25

quote mode, 10-35

Raad het getal, 10-33
READY-prompt, 3-15

Real-time klok
Batterij plaatsen, 3-8
Datum/tijd instellen, 5-5
Regels
bewerken, 10-25
hernummeren, 10-38
vervangen, 10-25
REGISTERS, O-13, O-23
Registers
$D000, Q-40
$D001, Q-40
$D002, Q-40
$D003, Q-40
$D004, Q-40
$D005, Q-40
$D006, Q-40
$D007, Q-40
$D008, Q-40
$D009, Q-40
$D00A, Q-40
$D00B, Q-40
$D00C, Q-40
$D00D, Q-40
$D00E, Q-40
$D00F, Q-41
$D010, Q-41
$D011, Q-41
$D012, Q-41
$D013, Q-41
$D014, Q-41
$D015, Q-41
$D016, Q-41
$D017, Q-41
$D018, Q-41
$D019, Q-41
$D01A, Q-41
$D01B, Q-41
$D01C, Q-41
$D01D, Q-41
$D01E, Q-41
$D01F, Q-41
$D020, Q-41, Q-43, Q-45
$D021, Q-41, Q-43, Q-45
$D022, Q-41, Q-43, Q-45
$D023, Q-41, Q-43, Q-46
$D024, Q-41, Q-43, Q-46
$D025, Q-41, Q-43, Q-46
$D026, Q-41, Q-43, Q-46
$D027, Q-41
$D028, Q-41
$D029, Q-41
$D02A, Q-41
$D02B, Q-41
$D02C, Q-41
$D02D, Q-41
$D02E, Q-41
$D02F, Q-41, Q-43, Q-46, U-9
$D030, Q-41, Q-43
$D031, Q-43
$D032, Q-43
$D033, Q-43
$D034, Q-43
$D035, Q-43
$D036, Q-43
$D037, Q-43
$D038, Q-43
$D039, Q-43
$D03A, Q-43
$D03B, Q-43
$D03C, Q-43
$D03D, Q-43
$D03E, Q-43
$D03F, Q-43
$D040, Q-43
$D041, Q-44
$D042, Q-44
$D043, Q-44
$D044, Q-44
$D045, Q-44
$D046, Q-44
$D047, Q-44
$D048, Q-46
$D049, Q-46
$D04A, Q-46

$D04B, Q-46
$D04C, Q-46
$D04D, Q-46
$D04E, Q-46
$D04F, Q-46
$D050, Q-46
$D051, Q-46
$D052, Q-46
$D053, Q-46
$D054, Q-46
$D055, Q-46
$D056, Q-46
$D057, Q-46
$D058, Q-46
$D059, Q-46
$D05A, Q-46
$D05B, Q-46
$D05C, Q-46
$D05D, Q-46
$D05E, Q-46
$D05F, Q-46
$D060, Q-46
$D061, Q-46
$D062, Q-46
$D063, Q-46
$D064, Q-46
$D065, Q-46
$D068, Q-46
$D069, Q-46
$D06A, Q-46
$D06B, Q-46
$D06C, Q-46
$D06D, Q-47
$D06E, Q-47
$D06F, Q-47
$D070, Q-47
$D071, Q-47
$D072, Q-47
$D073, Q-47
$D074, Q-47
$D075, Q-47
$D076, Q-47
$D077, Q-47
$D078, Q-47
$D079, Q-47
$D07A, Q-47
$D07B, Q-47
$D07C, Q-47
$D080, V-13
$D081, V-13
$D082, V-13
$D083, V-13
$D084, V-13
$D085, V-13
$D086, V-13
$D087, V-13
$D088, V-13
$D089, V-13
$D08A, V-13
$D100 - $D1FF, Q-44
$D200 - $D2FF, Q-44
$D300 - $D3FF, Q-44
$D400, R-3
$D401, R-3
$D402, R-3
$D403, R-3
$D404, R-3
$D405, R-3
$D406, R-3
$D407, R-3
$D408, R-3
$D409, R-3
$D40A, R-3
$D40B, R-3
$D40C, R-3
$D40D, R-3
$D40E, R-3
$D40F, R-3
$D410, R-3
$D411, R-3
$D412, R-3
$D413, R-3
$D414, R-3
$D415, R-3

$D416, R-3
$D417, R-3
$D418, R-3
$D419, R-3
$D41A, R-3
$D41B, R-3
$D41C, R-3
$D41D, R-3
$D41E, R-3
$D600, T-3
$D601, T-3
$D602, T-3
$D603, T-3
$D604, T-3
$D605, T-3
$D606, T-3
$D609, T-4
$D60A, T-4
$D60B, T-4
$D60C, T-4
$D60D, T-4
$D60E, T-4
$D60F, T-4
$D610, T-4
$D611, T-4
$D612, T-4
$D613, T-4
$D614, T-4
$D615, T-4
$D616, T-4
$D617, T-4
$D618, T-4
$D619, T-4
$D61A, T-4
$D61D, T-4
$D61E, T-4
$D620, T-4
$D621, T-4
$D622, T-5
$D623, T-5
$D625, T-5
$D626, T-5
$D627, T-5
$D628, T-5
$D629, T-5
$D63C, R-4
$D640, K-16, K-28, N-3
$D641, K-16, K-28, N-3
$D642, K-16, N-3
$D643, K-16, K-28, N-3
$D644, K-16, K-28, N-3
$D645, K-16, K-28, N-3
$D646, K-16, K-28, N-3
$D647, K-16, K-28, N-3
$D648, K-16, K-28, N-3
$D649, K-16, K-28, N-3
$D64A, K-16, K-28, N-3
$D64B, K-16, K-28, N-3
$D64C, K-16, K-28, N-3
$D64D, K-16, K-28, N-3
$D64E, K-16, K-28, N-3
$D64F, K-16, K-28, N-3
$D650, K-16, K-28, N-3
$D651, K-16, K-28, N-3
$D652, K-16, K-28, N-3
$D653, K-16, K-28, N-3
$D654, K-16, K-28, N-3
$D655, K-16, K-28, N-3
$D656, K-16, K-28, N-3
$D657, K-16, K-28, N-3
$D658, K-16, K-28, N-3
$D659, K-16, K-28, N-3
$D65A, K-16, N-3
$D65B, K-16, N-3
$D65C, K-16, N-3
$D65D, K-16, K-28, N-3
$D65E, K-16, K-28, N-3
$D65F, K-16, K-28, N-3
$D660, K-16, K-28, N-3
$D661, K-16, K-28, N-3
$D662, K-16, K-28, N-3
$D663, K-16, K-28, N-3
$D664, K-16, K-28, N-3
$D665, K-17, K-28, N-3

$D666, K-17, K-28, N-3
$D667, K-17, K-28, N-3
$D668, K-17, K-28, N-3
$D669, K-17, K-28, N-3
$D66A, K-17, K-28, N-3
$D66B, K-17, K-28, N-3
$D66C, K-17, K-28, N-3
$D66D, K-17, K-29, N-3
$D66E, K-17, K-29, N-3
$D66F, K-17, K-29, N-3
$D670, K-17, K-29, N-3
$D671, K-17, K-29, N-3
$D672, K-17, K-29, N-3
$D673, K-17, N-3
$D674, K-17, N-3
$D675, K-17, N-3
$D676, K-17, N-3
$D677, K-17, N-3
$D678, K-17, N-3
$D679, K-17, N-3
$D67A, K-17, N-3
$D67B, K-17, N-3
$D67C, K-17, K-29, N-3
$D67D, K-17, K-29, N-3
$D67E, K-17, K-29, N-3
$D67F, K-17, K-29, N-3
$D680, V-20
$D681, V-20
$D682, V-20
$D683, V-20
$D684, V-21
$D686, V-21
$D68A, V-16, V-21
$D68B, V-16
$D68C, V-16
$D68D, V-16
$D68E, V-16
$D68F, V-16
$D690, V-16
$D691, V-16
$D692, V-16
$D693, V-16
$D694, W-11
$D695, W-11
$D697, W-11
$D698, W-11
$D699, W-11
$D69A, W-11
$D6A0, V-15
$D6A1, V-16
$D6A2, V-15
$D6AE, V-21
$D6AF, V-21
$D6B0, V-23
$D6B1, V-23
$D6B2, V-23
$D6B3, V-23
$D6B4, V-23
$D6B5, V-23
$D6B7, V-23
$D6B8, V-23
$D6B9, V-23
$D6BA, V-23
$D6BB, V-23
$D6BC, V-23
$D6BD, V-23
$D6BE, V-23
$D6C0, V-23
$D6E0, U-9
$D6E1, U-9
$D6E2, U-9
$D6E3, U-9
$D6E4, U-9
$D6E5, U-9
$D6E6, U-9
$D6E7, U-9
$D6E8, U-9
$D6E9, U-9
$D6EA, U-9
$D6EB, U-9
$D6EC, U-9
$D6ED, U-9
$D6EE, U-9
$D6F4, V-27

$D6F5, V-27
$D6F8, V-27
$D6F9, V-27
$D6FA, V-27
$D6FB, V-27
$D6FC, V-27
$D6FD, V-27
$D700, P-19
$D701, P-19
$D702, P-19
$D703, P-20
$D704, P-20
$D705, P-20
$D706, P-20
$D707, P-20
$D70E, P-20
$D70F, K-19
$D710, K-17
$D711, P-20, V-27
$D712, P-20
$D713, P-20
$D714, P-20
$D715, P-20
$D716, P-20
$D717, P-20
$D71C, P-20
$D71D, P-20
$D71E, P-20
$D71F, P-20
$D720, P-20
$D721, P-20
$D722, P-20
$D723, P-20
$D724, P-20
$D725, P-20
$D726, P-20
$D727, P-20
$D728, P-20
$D729, P-20
$D72A, P-20
$D72B, P-20
$D72C, P-20
$D72D, P-20
$D72E, P-20
$D72F, P-20
$D730, P-20
$D731, P-21
$D732, P-21
$D733, P-21
$D734, P-21
$D735, P-21
$D736, P-21
$D737, P-21
$D738, P-21
$D739, P-21
$D73A, P-21
$D73B, P-21
$D73C, P-21
$D73D, P-21
$D73E, P-21
$D73F, P-21
$D740, P-21
$D741, P-21
$D742, P-21
$D743, P-21
$D744, P-21
$D745, P-21
$D746, P-21
$D747, P-21
$D748, P-21
$D749, P-21
$D74A, P-21
$D74B, P-21
$D74C, P-21
$D74D, P-21
$D74E, P-21
$D74F, P-21
$D750, P-21
$D751, P-21
$D752, P-21
$D753, P-21
$D754, P-21
$D755, P-21
$D756, P-21

$D757, P-21
$D758, P-21
$D759, P-21
$D75A, P-22
$D75B, P-22
$D75C, P-22
$D75D, P-22
$D75E, P-22
$D75F, P-22
$D768, K-19
$D769, K-19
$D76A, K-19
$D76B, K-19
$D76C, K-19
$D76D, K-19
$D76E, K-19
$D76F, K-19
$D770, K-19
$D771, K-19
$D772, K-19
$D773, K-19
$D774, K-19
$D775, K-19
$D776, K-19
$D777, K-19
$D778, K-19
$D779, K-19
$D77A, K-19
$D77B, K-19
$D77C, K-19
$D77D, K-19
$D77E, K-19
$D77F, K-19
$D780, K-19
$D781, K-19
$D782, K-20
$D783, K-20
$D784, K-20
$D785, K-20
$D786, K-20
$D787, K-20
$D788, K-20
$D789, K-20
$D78A, K-20
$D78B, K-20
$D78C, K-20
$D78D, K-20
$D78E, K-20
$D78F, K-20
$D790, K-20
$D791, K-20
$D792, K-20
$D793, K-20
$D794, K-20
$D795, K-20
$D796, K-20
$D797, K-20
$D798, K-20
$D799, K-20
$D79A, K-20
$D79B, K-20
$D79C, K-20
$D79D, K-20
$D79E, K-20
$D79F, K-20
$D7A0, K-20
$D7A1, K-20
$D7A2, K-20
$D7A3, K-20
$D7A4, K-20
$D7A5, K-20
$D7A6, K-20
$D7A7, K-20
$D7A8, K-20
$D7A9, K-20
$D7AA, K-20
$D7AB, K-21
$D7AC, K-21
$D7AD, K-21
$D7AE, K-21
$D7AF, K-21
$D7B0, K-21
$D7B1, K-21
$D7B2, K-21

$D7B3, K-21
$D7B4, K-21
$D7B5, K-21
$D7B6, K-21
$D7B7, K-21
$D7B8, K-21
$D7B9, K-21
$D7BA, K-21
$D7BB, K-21
$D7BC, K-21
$D7BD, K-21
$D7BE, K-21
$D7BF, K-21
$D7C0, K-21
$D7C1, K-21
$D7C2, K-21
$D7C3, K-21
$D7C4, K-21
$D7C5, K-21
$D7C6, K-21
$D7C7, K-21
$D7C8, K-21
$D7C9, K-21
$D7CA, K-21
$D7CB, K-21
$D7CC, K-21
$D7CD, K-21
$D7CE, K-21
$D7CF, K-21
$D7D0, K-21
$D7D1, K-21
$D7D2, K-22
$D7D3, K-22
$D7D4, K-22
$D7D5, K-22
$D7D6, K-22
$D7D7, K-22
$D7D8, K-22
$D7D9, K-22
$D7DA, K-22
$D7DB, K-22
$D7DC, K-22
$D7DD, K-22
$D7DE, K-22
$D7DF, K-22
$D7E0, K-22
$D7E1, K-22
$D7E2, K-22
$D7E3, K-22
$D7EF, K-17
$D7FA, K-17
$D7FB, K-17
$D7FD, K-17
$D7FE, K-17
$DC00, S-3
$DC01, S-3
$DC02, S-3
$DC03, S-3
$DC04, S-3
$DC05, S-3
$DC06, S-3
$DC07, S-3
$DC08, S-3
$DC09, S-3
$DC0A, S-3
$DC0B, S-3
$DC0C, S-3
$DC0D, S-3
$DC0E, S-3
$DC0F, S-3
$DC10, S-6
$DC11, S-6
$DC12, S-6
$DC13, S-6
$DC14, S-6
$DC15, S-6
$DC16, S-6
$DC17, S-6
$DC18, S-6
$DC19, S-6
$DC1A, S-6
$DC1B, S-6
$DC1C, S-6
$DC1D, S-6

$DC1E, S-6
$DC1F, S-6
$DD00, S-4
$DD01, S-4
$DD02, S-4
$DD03, S-4
$DD04, S-4
$DD05, S-4
$DD06, S-4
$DD07, S-4
$DD08, S-4
$DD09, S-4
$DD0B, S-4
$DD0C, S-5
$DD0D, S-5
$DD0E, S-5
$DD0F, S-5
$DD10, S-7
$DD11, S-7
$DD12, S-7
$DD13, S-7
$DD14, S-7
$DD15, S-7
$DD16, S-7
$DD17, S-7
$DD18, S-7
$DD19, S-7
$DD1A, S-7
$DD1B, S-7
$DD1C, S-7
$DD1D, S-7
$DD1E, S-8
$DD1F, S-8
$FFD7100 - $FFD7107, Y-3
$FFD7110, X-3
$FFD7111, X-3
$FFD7112, X-3
$FFD7113, X-3
$FFD7114, X-3
$FFD7115, X-3
$FFD7116, X-3, X-4
$FFD7117, X-4
$FFD7118, X-4
$FFD7119, X-4
$FFD711A, X-4
$FFD711B, X-4
$FFD711C, X-4
$FFD711D, X-4
$FFD711E, X-4
$FFD711F, X-4
$FFD7120, X-4
$FFD7121, X-4
$FFD7122, X-4
$FFD7123, X-4
$FFD7124, X-4
$FFD7125, X-4
$FFD7126, X-4
$FFD7127, X-4
$FFD7128, X-4
$FFD7129, X-4
$FFD712A, X-4
$FFD712B, X-4
$FFD712C, X-4
$FFD712D, X-4
$FFD712E, X-4
$FFD712F, X-4
$FFD7130, X-4
$FFD7131, X-4
$FFD7132, X-4
$FFD7133, X-4
$FFD7134, X-4
$FFD7135, X-4
$FFD7136, X-4
$FFD7137, X-5
$FFD7138, X-5
$FFD7139, X-5
$FFD713A, X-5
$FFD713B, X-5
$FFD713C, X-5
$FFD713D, X-5
$FFD7140 - $FFD717F, X-3
$FFD7400, X-8
$FFD7401, X-8
$FFD7402, X-8

$FFD7403, X-8
$FFD7404, X-8
$FFD7405, X-8
$FFD7406, X-8
$FFD7407, X-8
$FFD7408, X-8
$FFD7409, X-8
$FFD740A, X-8
$FFD740B, X-8
$FFD740C, X-8
$FFD740D, X-9
$FFD740E, X-9
$FFD740F, X-9
$FFD7410, X-9
$FFD7411, X-9
$FFD7412, X-9
100THSEC, X-4, X-5
53504 - 53759, Q-44
53760 - 54015, Q-44
54016 - 54271, Q-44
53248, Q-40
53249, Q-40
53250, Q-40
53251, Q-40
53252, Q-40
53253, Q-40
53254, Q-40
53255, Q-40
53256, Q-40
53257, Q-40
53258, Q-40
53259, Q-40
53260, Q-40
53261, Q-40
53262, Q-40
53263, Q-41
53264, Q-41
53265, Q-41
53266, Q-41
53267, Q-41
53268, Q-41
53269, Q-41
53270, Q-41
53271, Q-41
53272, Q-41
53273, Q-41
53274, Q-41
53275, Q-41
53276, Q-41
53277, Q-41
53278, Q-41
53279, Q-41
53280, Q-41, Q-43, Q-45
53281, Q-41, Q-43, Q-45
53282, Q-41, Q-43, Q-45
53283, Q-41, Q-43, Q-46
53284, Q-41, Q-43, Q-46
53285, Q-41, Q-43, Q-46
53286, Q-41, Q-43, Q-46
53287, Q-41
53288, Q-41
53289, Q-41
53290, Q-41
53291, Q-41
53292, Q-41
53293, Q-41
53294, Q-41
53295, Q-41, Q-43, Q-46, U-9
53296, Q-41, Q-43
53297, Q-43
53298, Q-43
53299, Q-43
53300, Q-43
53301, Q-43
53302, Q-43
53303, Q-43
53304, Q-43
53305, Q-43
53306, Q-43
53307, Q-43
53308, Q-43
53309, Q-43
53310, Q-43
53311, Q-43

53312, Q-43
53313, Q-44
53314, Q-44
53315, Q-44
53316, Q-44
53317, Q-44
53318, Q-44
53319, Q-44
53320, Q-46
53321, Q-46
53322, Q-46
53323, Q-46
53324, Q-46
53325, Q-46
53326, Q-46
53327, Q-46
53328, Q-46
53329, Q-46
53330, Q-46
53331, Q-46
53332, Q-46
53333, Q-46
53334, Q-46
53335, Q-46
53336, Q-46
53337, Q-46
53338, Q-46
53339, Q-46
53340, Q-46
53341, Q-46
53342, Q-46
53343, Q-46
53344, Q-46
53345, Q-46
53346, Q-46
53347, Q-46
53348, Q-46
53349, Q-46
53352, Q-46
53353, Q-46
53354, Q-46
53355, Q-46
53356, Q-46
53357, Q-47
53358, Q-47
53359, Q-47
53360, Q-47
53361, Q-47
53362, Q-47
53363, Q-47
53364, Q-47
53365, Q-47
53366, Q-47
53367, Q-47
53368, Q-47
53369, Q-47
53370, Q-47
53371, Q-47
53372, Q-47
53376, V-13
53377, V-13
53378, V-13
53379, V-13
53380, V-13
53381, V-13
53382, V-13
53383, V-13
53384, V-13
53385, V-13
53386, V-13
54272, R-3
54273, R-3
54274, R-3
54275, R-3
54276, R-3
54277, R-3
54278, R-3
54279, R-3
54280, R-3
54281, R-3
54282, R-3
54283, R-3
54284, R-3
54285, R-3

54286, R-3
54287, R-3
54288, R-3
54289, R-3
54290, R-3
54291, R-3
54292, R-3
54293, R-3
54294, R-3
54295, R-3
54296, R-3
54297, R-3
54298, R-3
54299, R-3
54300, R-3
54301, R-3
54302, R-3
54784, T-3
54785, T-3
54786, T-3
54787, T-3
54788, T-3
54789, T-3
54790, T-3
54793, T-4
54794, T-4
54795, T-4
54796, T-4
54797, T-4
54798, T-4
54799, T-4
54800, T-4
54801, T-4
54802, T-4
54803, T-4
54804, T-4
54805, T-4
54806, T-4
54807, T-4
54808, T-4
54809, T-4
54810, T-4
54813, T-4
54814, T-4
54816, T-4
54817, T-4
54818, T-5
54819, T-5
54821, T-5
54822, T-5
54823, T-5
54824, T-5
54825, T-5
54844, R-4
54848, K-16, K-28
54849, K-16, K-28
54850, K-16
54851, K-16, K-28
54852, K-16, K-28
54853, K-16, K-28
54854, K-16, K-28
54855, K-16, K-28
54856, K-16, K-28
54857, K-16, K-28
54858, K-16, K-28
54859, K-16, K-28
54860, K-16, K-28
54861, K-16, K-28
54862, K-16, K-28
54863, K-16, K-28
54864, K-16, K-28
54865, K-16, K-28
54866, K-16, K-28
54867, K-16, K-28
54868, K-16, K-28
54869, K-16, K-28
54870, K-16, K-28
54871, K-16, K-28
54872, K-16, K-28
54873, K-16, K-28
54874, K-16
54875, K-16
54876, K-16
54877, K-16, K-28

54878, K-16, K-28
54879, K-16, K-28
54880, K-16, K-28
54881, K-16, K-28
54882, K-16, K-28
54883, K-16, K-28
54884, K-16, K-28
54885, K-17, K-28
54886, K-17, K-28
54887, K-17, K-28
54888, K-17, K-28
54889, K-17, K-28
54890, K-17, K-28
54891, K-17, K-28
54892, K-17, K-28
54893, K-17, K-29
54894, K-17, K-29
54895, K-17, K-29
54896, K-17, K-29
54897, K-17, K-29
54898, K-17, K-29
54899, K-17
54900, K-17
54901, K-17
54902, K-17
54903, K-17
54904, K-17
54905, K-17
54906, K-17
54907, K-17
54908, K-17, K-29
54909, K-17, K-29
54910, K-17, K-29
54911, K-17, K-29
54912, V-20
54913, V-20
54914, V-20
54915, V-20
54916, V-21
54918, V-21
54922, V-16, V-21
54923, V-16
54924, V-16
54925, V-16
54926, V-16
54927, V-16
54928, V-16
54929, V-16
54930, V-16
54931, V-16
54932, W-11
54933, W-11
54935, W-11
54936, W-11
54937, W-11
54938, W-11
54944, V-15
54945, V-16
54946, V-15
54958, V-21
54959, V-21
54960, V-23
54961, V-23
54962, V-23
54963, V-23
54964, V-23
54965, V-23
54967, V-23
54968, V-23
54969, V-23
54970, V-23
54971, V-23
54972, V-23
54973, V-23
54974, V-23
54976, V-23
55008, U-9
55009, U-9
55010, U-9
55011, U-9
55012, U-9
55013, U-9
55014, U-9
55015, U-9

55016, U-9
55017, U-9
55018, U-9
55019, U-9
55020, U-9
55021, U-9
55022, U-9
55028, V-27
55029, V-27
55032, V-27
55033, V-27
55034, V-27
55035, V-27
55036, V-27
55037, V-27
55040, P-19
55041, P-19
55042, P-19
55043, P-20
55044, P-20
55045, P-20
55046, P-20
55047, P-20
55054, P-20
55055, K-19
55056, K-17
55057, P-20, V-27
55058, P-20
55059, P-20
55060, P-20
55061, P-20
55062, P-20
55063, P-20
55068, P-20
55069, P-20
55070, P-20
55071, P-20
55072, P-20
55073, P-20
55074, P-20
55075, P-20
55076, P-20
55077, P-20
55078, P-20
55079, P-20
55080, P-20
55081, P-20
55082, P-20
55083, P-20
55084, P-20
55085, P-20
55086, P-20
55087, P-20
55088, P-20
55089, P-21
55090, P-21
55091, P-21
55092, P-21
55093, P-21
55094, P-21
55095, P-21
55096, P-21
55097, P-21
55098, P-21
55099, P-21
55100, P-21
55101, P-21
55102, P-21
55103, P-21
55104, P-21
55105, P-21
55106, P-21
55107, P-21
55108, P-21
55109, P-21
55110, P-21
55111, P-21
55112, P-21
55113, P-21
55114, P-21
55115, P-21
55116, P-21
55117, P-21
55118, P-21

55119, P-21
55120, P-21
55121, P-21
55122, P-21
55123, P-21
55124, P-21
55125, P-21
55126, P-21
55127, P-21
55128, P-21
55129, P-21
55130, P-22
55131, P-22
55132, P-22
55133, P-22
55134, P-22
55135, P-22
55144, K-19
55145, K-19
55146, K-19
55147, K-19
55148, K-19
55149, K-19
55150, K-19
55151, K-19
55152, K-19
55153, K-19
55154, K-19
55155, K-19
55156, K-19
55157, K-19
55158, K-19
55159, K-19
55160, K-19
55161, K-19
55162, K-19
55163, K-19
55164, K-19
55165, K-19
55166, K-19
55167, K-19
55168, K-19
55169, K-19
55170, K-20
55171, K-20
55172, K-20
55173, K-20
55174, K-20
55175, K-20
55176, K-20
55177, K-20
55178, K-20
55179, K-20
55180, K-20
55181, K-20
55182, K-20
55183, K-20
55184, K-20
55185, K-20
55186, K-20
55187, K-20
55188, K-20
55189, K-20
55190, K-20
55191, K-20
55192, K-20
55193, K-20
55194, K-20
55195, K-20
55196, K-20
55197, K-20
55198, K-20
55199, K-20
55200, K-20
55201, K-20
55202, K-20
55203, K-20
55204, K-20
55205, K-20
55206, K-20
55207, K-20
55208, K-20
55209, K-20
55210, K-20

55211, K-21
55212, K-21
55213, K-21
55214, K-21
55215, K-21
55216, K-21
55217, K-21
55218, K-21
55219, K-21
55220, K-21
55221, K-21
55222, K-21
55223, K-21
55224, K-21
55225, K-21
55226, K-21
55227, K-21
55228, K-21
55229, K-21
55230, K-21
55231, K-21
55232, K-21
55233, K-21
55234, K-21
55235, K-21
55236, K-21
55237, K-21
55238, K-21
55239, K-21
55240, K-21
55241, K-21
55242, K-21
55243, K-21
55244, K-21
55245, K-21
55246, K-21
55247, K-21
55248, K-21
55249, K-21
55250, K-22
55251, K-22
55252, K-22
55253, K-22
55254, K-22
55255, K-22
55256, K-22
55257, K-22
55258, K-22
55259, K-22
55260, K-22
55261, K-22
55262, K-22
55263, K-22
55264, K-22
55265, K-22
55266, K-22
55267, K-22
55279, K-17
55290, K-17
55291, K-17
55293, K-17
55294, K-17
56320, S-3
56321, S-3
56322, S-3
56323, S-3
56324, S-3
56325, S-3
56326, S-3
56327, S-3
56328, S-3
56329, S-3
56330, S-3
56331, S-3
56332, S-3
56333, S-3
56334, S-3
56335, S-3
56336, S-6
56337, S-6
56338, S-6
56339, S-6
56340, S-6
56341, S-6

56342, S-6
56343, S-6
56344, S-6
56345, S-6
56346, S-6
56347, S-6
56348, S-6
56349, S-6
56350, S-6
56351, S-6
56576, S-4
56577, S-4
56578, S-4
56579, S-4
56580, S-4
56581, S-4
56582, S-4
56583, S-4
56584, S-4
56585, S-4
56587, S-4
56588, S-5
56589, S-5
56590, S-5
56591, S-5
56592, S-7
56593, S-7
56594, S-7
56595, S-7
56596, S-7
56597, S-7
56598, S-7
56599, S-7
56600, S-7
56601, S-7
56602, S-7
56603, S-7
56604, S-7
56605, S-7
56606, S-8
56607, S-8
ABTPALSEL, Q-47
ACCESSKEY, T-5
ADDRBANK, P-19
ADDRLSB, P-20, P-22
ADDRLSBTRIG, P-19
ADDRMB, P-20, P-22
ADDRMSB, P-19
AE_D, X-5
AE_H, X-5
AE_M, X-5
AF, X-5
AIE, X-5
ALDAY, X-4, X-5
ALGO, V-13
ALHOURS, X-4, X-5
ALMINUTES, X-4, X-5
ALPHADELAY, Q-47
ALPHEN, Q-47
ALRM, S-3, S-5
ALRMAMPM, S-7, S-8
ALRMHOUR, S-6-S-8
ALRMJIF, S-6-S-8
ALRMMIN, S-6-S-8
ALRMSEC, S-6-S-8
ALT, V-13
ASCFAST, K-29
ASCIIKEY, T-4, T-5
ATTR, Q-44
AUDBLKTO, P-20, P-22
AUDEN, P-22
AUDWRBLK, P-22
AUTO2XSEL, V-21
B1PIX, Q-44
B2ADEVN, Q-43
B2ADODD, Q-43
B2PIX, Q-44
B3ADEVN, Q-43
B3ADODD, Q-43
B3PIX, Q-44
B4ADEVN, Q-43
B4ADODD, Q-43
B4PIX, Q-44
B5ADEVN, Q-43

B5ADODD, Q-43
B5PIX, Q-44
B6ADEVN, Q-43
B6ADODD, Q-43
B6PIX, Q-44
B7ADEVN, Q-43
B7ADODD, Q-43
B7PIX, Q-44
BADEXTRA, K-17
BADLEN, K-17
BASHDDR, T-4, T-5
BBDRPOS, Q-46, Q-47
BCST, U-9
BITPBANK, Q-47
BLKD, P-22
BLNK, Q-41
BMM, Q-41
BNPIX, Q-43, Q-44
BOARDMINOR, T-5
BOLDISALT, Q-47
BORDERCOL, Q-41, Q-43-Q-45, Q-47
BP16ENS, Q-47
BPCOMP, Q-43, Q-44
BPENABLE, Q-43, Q-44
BPM, Q-44
BPX, Q-43, Q-44
BPY, Q-43, Q-44
BRCOST, K-17
BSF, X-5
BSIE, X-5
BSP, Q-41
BTPALSEL, Q-47
BUSY, V-13
BXADEVN, Q-43, Q-44
BXADODD, Q-43, Q-44
BYADEVN, Q-43, Q-44
BYADODD, Q-43, Q-44
C128FAST, Q-42
C64MAP, K-15
CAIE, X-5
CALCEN, K-22
CALXDELTALSB, V-23
CALXSCALELSB, V-23
CALXSCALEMSB, V-23
CALYDELTALSB, V-23
CALYDELTAMSB, V-23
CALYSCALELSB, V-23
CALYSCALEMSB, V-23
CARTEN, K-17
CB, Q-41, Q-42
CDC00, V-21
CEIE, X-5
CH0RVOL, P-20, P-22
CH1BADDRBNK, P-21
CH1BADDRLSB, P-21
CH1BADDRMSB, P-21
CH1CURADDRBNK, P-21
CH1CURADDRLSB, P-21
CH1CURADDRMSB, P-21
CH1FREQ0, P-21
CH1FREQ1, P-21
CH1FREQ2, P-21
CH1RVOL, P-20, P-22
CH1SBITS, P-20
CH1TADDRLSB, P-21
CH1TADDRMSB, P-21
CH1TMGCNT0, P-21
CH1TMGCNT1, P-21
CH1TMGCNT2, P-21
CH1VOLUME, P-21
CH2BADDRBNK, P-21
CH2BADDRLSB, P-21
CH2BADDRMSB, P-21
CH2CURADDRBNK, P-21
CH2CURADDRLSB, P-21
CH2CURADDRMSB, P-21
CH2FREQ0, P-21
CH2FREQ1, P-21
CH2FREQ2, P-21
CH2LVOL, P-20, P-22
CH2SBITS, P-21
CH2TADDRLSB, P-21
CH2TADDRMSB, P-21

CH2TMGCNT0, P-21
CH2TMGCNT1, P-21
CH2TMGCNT2, P-21
CH2VOLUME, P-21
CH3BADDRBNK, P-21
CH3BADDRLSB, P-21
CH3BADDRMSB, P-21
CH3CURADDRBNK, P-22
CH3CURADDRLSB, P-22
CH3CURADDRMSB, P-22
CH3FREQ0, P-21
CH3FREQ1, P-21
CH3FREQ2, P-21
CH3LVOL, P-20, P-22
CH3SBITS, P-21
CH3TADDRLSB, P-21
CH3TADDRMSB, P-21
CH3TMGCNT0, P-22
CH3TMGCNT1, P-22
CH3TMGCNT2, P-22
CH3VOLUME, P-21
CHANSWAP, P-22
CHARPTRBNK, Q-46, Q-47
CHARPTRLSB, Q-46, Q-47
CHARPTRMSB, Q-46, Q-47
CHARSZ, T-3
CHARY16, Q-47
CHR16, Q-47
CHRCOUNT, Q-46, Q-47
CHRXSCL, Q-46, Q-48
CHRYSCL, Q-46, Q-48
CHXBADDRBNK, P-20, P-22
CHXBADDRLSB, P-20, P-22
CHXBADDRMSB, P-20, P-22
CHXCURADDRBNK, P-20, P-22
CHXCURADDRLSB, P-20, P-22
CHXCURADDRMSB, P-20, P-22
CHXEN, P-22
CHXENA, P-22
CHXFLG, P-22
CHXFREQ0, P-20, P-22
CHXFREQ1, P-20, P-23
CHXFREQ2, P-20, P-23
CHXLOOP, P-23
CHXOFFS, P-20, P-23
CHXSBITS, P-20, P-23
CHXSINE, P-23
CHXSTP, P-23
CHXTADDRL, P-20, P-23
CHXTADDRM, P-20, P-23
CHXTMGCNT0, P-20, P-23
CHXTMGCNT1, P-20, P-23
CHXTMGCNT2, P-20, P-23
CHXUSGN, P-23
CHXVOLUME, P-20, P-23
CLKD, X-5
CLKDE, X-5
CLKF, X-5
CLKIE, X-5
CLOCK, V-13
CMDANDSTAT, V-20, V-21
COLPTRLSB, Q-46, Q-48
COLPTRMSB, Q-46, Q-48
COMMAND, U-9, V-13
CONN41, T-5
CPUFAST, K-29
CRAM2K, Q-44
CRC, V-13
CROM9, Q-44
CSEL, Q-42
CTHIE, X-5
CTIE, X-5
CTLIE, X-5
CUIE, X-6
D0D64, V-16
D0IMG, V-16
D0MD, V-16
D0P, V-16
D0STARTSEC0, V-16, V-17
D0STARTSEC1, V-16, V-17
D0STARTSEC2, V-16, V-17
D0STARTSEC3, V-16, V-17
D0WP, V-17
D1D64, V-17

D1IMG, V-17
D1MD, V-17
D1P, V-17
D1STARTSEC0, V-16, V-17
D1STARTSEC1, V-16, V-17
D1STARTSEC2, V-16, V-17
D1STARTSEC3, V-16, V-17
D1WP, V-17
DATA, T-3, V-13, W-11
DATALOG0, W-11
DATALOG1, W-11
DATARATE, V-15
DBGDIR, V-15
DBGLFTCLP, P-23
DBGMOTORA, V-15
DBGRGTCLP, P-23
DBGWDATA, V-15
DBGWGATE, V-15
DBLRR, Q-48
DD00DELAY, S-7, S-8
DDRA, S-3-S-5
DDRB, S-3-S-5
DEBUGC, Q-47, Q-48
DENSITY, V-15
DEVNUM, W-11
DIATN, W-11
DIGILEFTLSB, V-27
DIGILEFTMSB, V-27
DIGILLSB, V-27
DIGILMSB, V-27
DIGIRIGHTLSB, V-27
DIGIRIGHTMSB, V-27
DIGIRLSB, V-27
DIGIRMSB, V-27
DIR, V-13
DISKIN, V-13
DISPROWS, Q-47, Q-48
DIVBUSY, K-22
DIVISOR, T-3
DIVOUT, K-19, K-22
DMADSTMB, K-28, K-29
DMALADDR0, K-28, K-29
DMALADDR1, K-28, K-29
DMALADDR2, K-28, K-29
DMALADDR3, K-28, K-29
DMASRCMB, K-28, K-29
DRQ, V-13
DRXD, U-9
DRXDV, U-9
DS, V-13
DSKCHG, V-13
ECM, Q-42
EEbusy, X-6
EEF, X-6
EERD, X-6
EHL, X-6
EIE, X-6
EN018B, P-23
ENTEREXIT, K-29
ENV2ATTDUR, R-3
ENV2DECDUR, R-3
ENV2LEVDUR, R-3
ENV2RELDUR, R-3
ENV3ATTDUR, R-3
ENV3DECDUR, R-3
ENV3LEVDUR, R-3
ENV3OUT, R-3, R-4
ENV3RELDUR, R-3
ENVXATTDUR, R-3, R-4
ENVXDECDUR, R-3, R-4
ENVXRELDUR, R-3, R-4
ENVXSUSLEV, R-3, R-4
EQ, V-13
ESYN, X-6
ET, X-4, X-6
ETRIG, P-20, P-23
ETRIGINLINE, P-20, P-23
ETRIGMAPD, P-20, P-23
EV1, V-23
EV2, V-23
EVF, X-6
EVOW, X-6
EVR, X-6
EXGLYPH, Q-48

EXSID, K-29
EXTIRQS, Q-48
EXTRTCA1DATE, X-8, X-9
EXTRTCA1DOW, X-9
EXTRTCA1DOWSEL, X-9
EXTRTCA1HOUR, X-8, X-9
EXTRTCA1HOUR12EN, X-9
EXTRTCA1HOURPM, X-9
EXTRTCA1M1, X-9
EXTRTCA1M2, X-9
EXTRTCA1M3, X-9
EXTRTCA1M4, X-9
EXTRTCA1MIN, X-8, X-9
EXTRTCA1SEC, X-8, X-9
EXTRTCA2DATE, X-9
EXTRTCA2DOW, X-9
EXTRTCA2DOWSEL, X-9
EXTRTCA2HOUR, X-8, X-9
EXTRTCA2HOUR12EN, X-10
EXTRTCA2HOURPM, X-10
EXTRTCA2M2, X-10
EXTRTCA2M3, X-10
EXTRTCA2M4, X-10
EXTRTCA2MIN, X-8, X-10
EXTRTCAGINGOFS, X-9, X-10
EXTRTCCENTURY, X-10
EXTRTCCTRL, X-9, X-10
EXTRTCDAY, X-8, X-10
EXTRTCDOW, X-8, X-10
EXTRTCHOUR, X-8, X-10
EXTRTCHOUR12EN, X-10
EXTRTCHOURPM, X-10
EXTRTCMIN, X-8, X-10
EXTRTCMONTH, X-8, X-10
EXTRTCSEC, X-8, X-10
EXTRTCST, X-9, X-10
EXTRTCTEMPMSB, X-9, X-10
EXTRTCYEAR, X-8, X-10
EXTSYNC, Q-44
F4502, K-29
FAST, Q-44
FCLRHI, Q-48
FCLRLO, Q-48
FCOLMCM, Q-48
FDC2XSEL, V-21
FDCENC, V-21
FDCTIBEN, V-21
FDCVARSPD, V-21
FILLVAL, V-21
FLG, S-3, S-5
FLTRBDPASS, R-4
FLTRCUTFRQHI, R-3, R-4
FLTRCUTFRQLO, R-3, R-4
FLTRCUTV3, R-4
FLTREXTINP, R-4
FLTRHIPASS, R-4
FLTRLOPASS, R-4
FLTRRESON, R-3, R-4
FLTRVOL, R-3, R-4
FLTRVXOUT, R-4
FNRASTERLSB, Q-46, Q-48
FNRASTERMSB, Q-46, Q-48
FNRST, Q-48
FNRSTCMP, Q-48
FRAMECOUNT, K-17, K-18
FREE, V-14
FRMERR, T-3
GEORAMBASE, K-29
GEORAMMASK, K-29
GESTUREDIR, V-23, V-24
GESTUREID, V-23, V-24
GP0, X-6
GP1, X-6
H1280, Q-44
H640, Q-45
HDSCL, T-5
HDSDA, T-5
HICKED, K-29
HOTREG, Q-48
HPOS, Q-43, Q-45
HSYNCP, Q-48
HTRAP01, K-16
HTRAP02, K-16
HTRAP03, K-16

HTRAP04, K-16
HTRAP05, K-16
HTRAP06, K-16
HTRAP07, K-16
HTRAP08, K-16
HTRAP09, K-16
HTRAP0A, K-16
HTRAP0B, K-16
HTRAP0C, K-16
HTRAP0D, K-16
HTRAP0E, K-16
HTRAP0F, K-16
HTRAP10, K-16
HTRAP11, K-16
HTRAP12, K-16
HTRAP13, K-16
HTRAP14, K-16
HTRAP15, K-16
HTRAP16, K-16
HTRAP17, K-16
HTRAP18, K-16
HTRAP19, K-16
HTRAP1A, K-16
HTRAP1B, K-16
HTRAP1C, K-16
HTRAP1D, K-16
HTRAP1E, K-16
HTRAP1F, K-16
HTRAP20, K-16
HTRAP21, K-16
HTRAP22, K-16
HTRAP23, K-16
HTRAP24, K-16
HTRAP25, K-17
HTRAP26, K-17
HTRAP27, K-17
HTRAP28, K-17
HTRAP29, K-17
HTRAP2A, K-17
HTRAP2B, K-17
HTRAP2C, K-17
HTRAP2D, K-17
HTRAP2E, K-17
HTRAP2F, K-17
HTRAP30, K-17
HTRAP31, K-17
HTRAP32, K-17
HTRAP33, K-17
HTRAP34, K-17
HTRAP35, K-17
HTRAP36, K-17
HTRAP37, K-17
HTRAP38, K-17
HTRAP39, K-17
HTRAP3A, K-17
HTRAP3B, K-17
HTRAP3C, K-17
HTRAP3D, K-17
HTRAP3E, K-17
HTRAP3F, K-17
HTRAPXX, K-16, K-18
HWRNGNOTRDY, K-18
HWRNGRAND, K-17, K-18
IFRXIRQ, T-3
IFRXNMI, T-3
IFTXIRQ, T-3
IFTXNMI, T-3
ILP, Q-42
IMALRM, S-7, S-8
IMFLG, S-7, S-8
IMODA, S-3, S-5
IMODB, S-3, S-5
IMRXIRQ, T-3
IMRXNMI, T-3
IMSP, S-7, S-8
IMTB, S-7, S-8
IMTXIRQ, T-3
IMTXNMI, T-3
INDEX, V-14
INT, Q-45
INTDE, X-6
IR, S-3, S-5
IRQ, V-14
IRQCH1OFFS, P-20

IRQCH2OFFS, P-20
IRQCH3OFFS, P-20
IRQEN, W-11
IRQFLAG, W-11
IRQRDY, W-11
IRQRDYEN, W-11
IRQRX, W-11
IRQRXEN, W-11
IRQTO, W-11
IRQTOEN, W-11
ISBC, Q-42
ISRCLR, S-3, S-5
ISSC, Q-42
J21H, T-5
J21HDDR, T-5
J21L, T-5
J21LDDR, T-5
JMP32EN, K-29
JOYSWAP, T-5
KEY, Q-41-Q-43, Q-45, Q-46, Q-48, U-9
KEYLEDENA, T-5
KEYLEDREG, T-4, T-5
KEYLEDVAL, T-4, T-5
KEYLEFT, T-5
KEYMATRIXPEEK, T-4, T-5
KEYMATRIXSEL, T-4, T-5
KEYQUEUE, T-6
KEYUP, T-6
KSCNRATE, T-4, T-6
LATCHINT, K-22
LED, V-14
LINESTEPLSB, Q-46, Q-48
LINESTEPMSB, Q-46, Q-48
LJOYA, T-6
LJOYB, T-6
LOAD, S-3, S-5
LOST, V-14
LPX, Q-41, Q-42
LPY, Q-41, Q-42
M65MODEL, T-5, T-6
MACADDR2, U-9
MACADDR3, U-9
MACADDR4, U-9
MACADDR5, U-9
MACADDR6, U-9
MACADDRX, U-9
MALT, T-6
MAPEDPAL, Q-47, Q-49
MAPHI0, K-28, K-29
MAPHI1, K-28, K-29
MAPHIMB, K-28, K-29
MAPLO0, K-28, K-29
MAPLO1, K-28, K-30
MAPLOMB, K-28, K-30
MATHIN0, K-19, K-20
MATHIN1, K-20
MATHIN2, K-20
MATHIN3, K-20
MATHIN4, K-20
MATHIN5, K-20
MATHIN6, K-20
MATHIN7, K-20
MATHIN8, K-20
MATHIN9, K-20
MATHINA, K-20, K-21
MATHINB, K-21
MATHINC, K-21
MATHIND, K-21
MATHINE, K-21
MATHINF, K-21
MATHINX, K-19, K-22
MATRIXEN, K-30
MC1, Q-41-Q-43, Q-45, Q-49
MC2, Q-41-Q-43, Q-45, Q-46, Q-49
MC3, Q-41-Q-43, Q-45, Q-46, Q-49
MCAPS, T-6
MCM, Q-42
MCST, U-9
MCTRL, T-6
MDISABLE, T-6
MIIMPHY, U-9

MIIMREG, U-9
MIIMVLSB, U-9
MIIMVMSB, U-9
MISBC, Q-42
MISSC, Q-42
MIXREGDATA, V-27
MIXREGSEL, V-27
MLSHFT, T-6
MMEGA, T-6
MODKEYALT, T-6
MODKEYCAPS, T-6
MODKEYCTRL, T-6
MODKEYLSHFT, T-6
MODKEYMEGA, T-6
MODKEYRSHFT, T-6
MODKEYSCRL, T-6
MONO, Q-45
MOTOR, V-14
MRIRQ, Q-42
MRSHFT, T-6
MSCRL, T-6
MULBUSY, K-22
MULTINA, K-19, K-22
MULTINB, K-19, K-22
MULTOUT, K-19, K-22
NOBUF, V-14
NOBUGCOMPAT, Q-49
NOCRC, U-9
NOEXROM, K-18
NOGAME, K-18
NOMBWRAP, P-23
NOMIX, P-23
NOPROM, U-10
NORRBWRAP, Q-49
NORRDEL, Q-49
NVRAM, X-3
OCEANA, K-18
OMODA, S-3, S-5
OMODB, S-3, S-5
Ongebruikt, X-3, X-4, X-8
OSC3RNG, R-3, R-4
OSKALT, T-6
OSKDEBUG, T-7
OSKDIM, T-7
OSKEN, T-7
OSKTOP, T-7
OSKZEN, T-7
OSKZON, T-7
PADDLE1, R-3, R-4
PADDLE2, R-3, R-4
PAL, Q-45
PALBLUE, Q-44, Q-45
PALEMU, Q-49
PALGREEN, Q-44, Q-45
PALNTSC, Q-49
PALRED, Q-44, Q-45
PBONA, S-3, S-5
PBONB, S-3, S-5
PCH, K-28, K-30
PCL, K-28, K-30
PCODE, V-13, V-14
PCTCR0, X-4, X-6
PCTCR1, X-4, X-6
PETSCIIKEY, T-4, T-7
PFLAGS, K-28, K-30
PIRQ, K-30
PNMI, K-30
PORF, X-6
PORT, K-15
PORT00, K-28, K-30
PORT01, K-28, K-30
PORTA, S-3-S-5
PORTB, S-3-S-5
PORTDDR, K-15
PORTF, T-4, T-7
PORTFDDR, T-4, T-7
POTAX, T-4, T-7
POTAY, T-4, T-7
POTBX, T-5, T-7
POTBY, T-5, T-7
POWEREN, K-18
PREFETCH, K-18
PRESENT, W-11
PROT, V-14, W-11

PTYEN, T-3
PTYERR, T-3
PTYEVEN, T-4
PWMPDM, V-27
RASCMP, Q-47, Q-49
RASCMPMSB, Q-47, Q-49
RASLINE0, Q-47, Q-49
RASTERHEIGHT, Q-47, Q-49
RC, Q-41, Q-42
RC8, Q-42
RCENABLED, U-10
RDCMD, V-14
RDREQ, V-14
READBACKLSB, V-27
READBACKMSB, V-27
REALHW, T-7
REGA, K-28, K-30
REGB, K-28, K-30
REGX, K-28, K-30
REGZ, K-28, K-30
RESERVED, K-22, K-23
RIRQ, Q-42
RMODA, S-4, S-5
RMODB, S-4, S-5
RNF, V-14
ROM8, Q-45
ROMA, Q-45
ROMC, Q-45
ROME, Q-45
ROMPROT, K-30
RSEL, Q-42
RST, Q-42, U-10
RST41, T-7
RSTDELEN, Q-49
RSVD, K-30
RTCDAY, X-3
RTCDOW, X-3, X-4, X-6
RTCHOUR, X-3
RTCHOUR24EN, X-3
RTCHOURPM, X-3
RTCMIN, X-3
RTCMONTH, X-3
RTCSEC, X-3
RTCYEAR, X-3
RUN, V-14
RXBF, U-9, U-10
RXBLKD, U-10
RXEN, T-4
RXOVRRUN, T-4
RXPH, U-9, U-10
RXQ, U-10
RXQEN, U-10
RXRDY, T-4
S1X, Q-40
S1Y, Q-40
S2X, Q-40
S2Y, Q-40
S3X, Q-40
S3Y, Q-40
S4X, Q-40
S4Y, Q-40
S5X, Q-40
S5Y, Q-40
S6X, Q-40
S6Y, Q-40
S7X, Q-40
S7Y, Q-41
SBC, Q-41, Q-42
SCM, Q-41, Q-42
SCREENCOL, Q-41-Q-43, Q-45, Q-49
SCRNPTRBNK, Q-46, Q-49
SCRNPTRLSB, Q-46, Q-49
SCRNPTRMB, Q-46, Q-49
SCRNPTRMSB, Q-46, Q-49
SDBDRWDLSB, Q-46, Q-49
SDBDRWDMSB, Q-46, Q-49
SDBSH, T-7
SDCLK, T-7
SDCS, T-7
SDDATA, T-7
SDR, S-3-S-5
SE, Q-41, Q-42
SECTOR, V-13, V-14

SECTOR0, V-20, V-21
SECTOR1, V-20, V-21
SECTOR2, V-20, V-21
SECTOR3, V-21
SELSDRAM, K-18
SEXX, Q-41, Q-42
SEXY, Q-41, Q-42
SHDEMU, Q-49
SIDE, V-13, V-14
SIDEXENA, R-4
SIDMODE, R-4
SIDPANLEFT, R-3, R-4
SIDPANRIGHT, R-3, R-4
SILENT, V-17
SLIEN, K-18
SLOWCACHE, K-18
SLOWCACHEADV, K-18
SLOWSDRAM, K-18
SMTH, Q-49
SNX, Q-40, Q-42
SNY, Q-40, Q-42
SP, S-4, S-5
SPH, K-28, K-30
SPL, K-28, K-30
SPMOD, S-4, S-5
SPR16EN, Q-46, Q-49
SPR1COL, Q-41
SPR2COL, Q-41
SPR3COL, Q-41
SPR4COL, Q-41
SPR5COL, Q-41
SPR6COL, Q-41
SPR7COL, Q-41
SPRALPHAVAL, Q-47, Q-49
SPRBPMEN, Q-46, Q-49
SPRENALPHA, Q-47, Q-49
SPRENV400, Q-47, Q-50
SPRH640, Q-50
SPRHGHT, Q-46, Q-50
SPRHGTEN, Q-46, Q-50
SPRMC0, Q-41, Q-43, Q-45, Q-46, Q-50
SPRMC1, Q-41, Q-43, Q-45, Q-46, Q-50
SPRNCOL, Q-41, Q-43
SPRPALSEL, Q-47, Q-50
SPRPTR16, Q-50
SPRPTRADRLSB, Q-46, Q-50
SPRPTRADRMSB, Q-47, Q-50
SPRPTRBNK, Q-47, Q-50
SPRTILEN, Q-46, Q-50
SPRX64EN, Q-46, Q-50
SPRXSMSBS, Q-46, Q-50
SPRYADJ, Q-47, Q-50
SPRYMSBS, Q-47, Q-50
SPRYSMSBS, Q-47, Q-50
SPTRCONT, Q-50
SSC, Q-41, Q-43
STC, W-11
STD, W-11
STDDIR, W-12
STEP, V-13, V-14
STNODEV, W-12
STNOEOI, W-12
STOP, X-6
STRM, U-10
STRTA, S-4, S-5
STRTB, S-4, S-5
STSRQ, W-12
STTO, W-12
STVERIFY, W-12
SWAP, V-14
SXMSB, Q-41, Q-43
SYNCMOD, T-3, T-4
SYSCTL, T-4, T-7
TA, S-4, S-5
TALATCH, S-6-S-8
TARGANY, V-17
TB, S-4, S-5
TBDRPOS, Q-46, Q-50
TD, X-4, X-6
TE, X-6
TEMPH, X-4, X-6
TEMPL, X-4, X-6

TEXTXPOS, Q-46, Q-50
TEXTYPOS, Q-46, Q-50
TF, X-6
THE, X-6
THF, X-6
THIE, X-7
THOW, X-7
THR, X-7
THT, X-4, X-7
TIE, X-7
TIMERA, S-3-S-5
TIMERB, S-3, S-4, S-6
TK0, V-14
TLE, X-7
TLF, X-7
TLIE, X-7
TLOW, X-7
TLR, X-7
TLT, X-4, X-7
TOD50, S-4, S-6
TODAMPM, S-4, S-6-S-8
TODEDIT, S-4, S-6
TODHOUR, S-3, S-4, S-6-S-8
TODJIF, S-3, S-4, S-6-S-8
TODMIN, S-3, S-4, S-6-S-8
TODSEC, S-3, S-4, S-6-S-8
TOUCH1XLSB, V-23, V-24
TOUCH1XMSB, V-23, V-24
TOUCH1YLSB, V-23, V-24
TOUCH1YMSB, V-23, V-24
TOUCH2XLSB, V-23, V-24
TOUCH2XMSB, V-23, V-24
TOUCH2YLSB, V-23, V-24
TOUCH2YMSB, V-23, V-24
TRACK, V-13, V-14
TSEVI100th, X-5, X-7
TSEVICount, X-4, X-7
TSEVIDat, X-5, X-7
TSEVIHrs, X-5, X-7
TSEVIMin, X-5, X-7
TSEVIMonth, X-5, X-7
TSEVISec, X-5, X-7
TSEVIYear, X-5, X-7
TSTHighCount, X-4, X-7
TSTHighDat, X-4, X-7
TSTHighHrs, X-4, X-7
TSTHighMin, X-4, X-7
TSTHighMonth, X-4, X-7
TSTHighSec, X-4, X-7
TSTHighYear, X-4, X-7
TSTLowCount, X-4, X-7
TSTLowDat, X-4, X-7
TSTLowHrs, X-4, X-7
TSTLowMin, X-4, X-7
TSTLowMonth, X-4, X-8
TSTLowSec, X-4, X-8
TSTLowYear, X-4, X-8
TXEN, T-4
TXIDLE, U-10
TXPH, U-9, U-10
TXQ, U-10
TXQEN, U-10
TXRST, U-10
TXSZLSB, U-9, U-10
TXSZMSB, U-9, U-10
UARTDATA, K-29, K-30
UF, X-8
UFAST, T-7
UIE, X-8
UNIT1INA, K-21
UNIT1INB, K-21
UNIT1OUT, K-21
UNIT2INA, K-21
UNIT2INB, K-21
UNIT2OUT, K-22
UNIT3INA, K-21
UNIT3INB, K-21
UNIT3OUT, K-22
UNIT4INA, K-21
UNIT4INB, K-21
UNIT4OUT, K-22
UNIT5INA, K-21
UNIT5INB, K-21
UNIT5OUT, K-22

UNIT6INA, K-21
UNIT6INB, K-21
UNIT6OUT, K-22
UNIT7INA, K-21
UNIT7INB, K-21
UNIT7OUT, K-22
UNIT8INA, K-21
UNIT8INB, K-21
UNIT8OUT, K-22
UNIT9INA, K-21
UNIT9INB, K-21
UNIT9OUT, K-22
UNITAINA, K-21
UNITAINB, K-21
UNITAOUT, K-22
UNITBINA, K-21
UNITBINB, K-21
UNITBOUT, K-22
UNITCINA, K-21
UNITCINB, K-21
UNITCOUT, K-22
UNITDINA, K-21
UNITDINB, K-21
UNITDOUT, K-22
UNITEINA, K-21
UNITEINB, K-21
UNITEOUT, K-22
UNITFINA, K-21
UNITFINB, K-21
UNITFOUT, K-22
UNITXINA, K-21, K-23
UNITXINB, K-21, K-23
UNITXOUT, K-21, K-23
UPDN1, V-23, V-24
UPDN2, V-23, V-24
UPSCALE, Q-50
USEL, X-8
USEREAL0, V-17
USEREAL1, V-17
UUID64, Y-3
UXBSADD, K-23
UXDVADD, K-23
UXHIOUT, K-23
UXLATCH, K-23
UXLOWOUT, K-23
UXMLADD, K-23
V400, Q-45
VDRQ, V-21
VEQINH, V-21
VFAST, Q-50
VFDC0, V-21
VFDC1, V-21
VFLOP0, K-30
VFLOP1, K-30
VGAHDTV, Q-50
VICIII, V-21
VICMODE, K-28, K-30
VIRTKEY1, T-4, T-7
VIRTKEY2, T-4, T-8
VIRTKEY3, T-4, T-8
VIRTPAGE0, K-28, K-30
VIRTPAGE1, K-28, K-30
VIRTPAGE2, K-28, K-30
VLF, X-8
VLOST, V-21
VOICE1CTRLRMF, R-4
VOICE1CTRLRMO, R-4
VOICE2CTRLRMF, R-5
VOICE2CTRLRMO, R-5
VOICE2FRQHI, R-3
VOICE2FRQLO, R-3
VOICE2PWHI, R-3
VOICE2PWLO, R-3
VOICE2UNSD, R-3
VOICE3CTRLRMF, R-5
VOICE3CTRLRMO, R-5
VOICE3FRQHI, R-3
VOICE3FRQLO, R-3
VOICE3PWHI, R-3
VOICE3PWLO, R-3
VOICE3UNSD, R-3
VOICEXCTRLGATE, R-5
VOICEXCTRLPUL, R-5
VOICEXCTRLRNW, R-5

VOICEXCTRLSAW, R-5
VOICEXCTRLTRI, R-5
VOICEXCTRLTST, R-5
VOICEXFRQHI, R-3, R-5
VOICEXFRQLO, R-3, R-5
VOICEXPWHI, R-3, R-5
VOICEXPWLO, R-3, R-5
VOICEXUNSD, R-3, R-5
VPG1LOG0, K-28
VPG1LOG1, K-28
VPG1PHY0, K-28
VPG2LOG0, K-28
VPG2LOG1, K-28
VPG2PHY0, K-28
VPG3LOG0, K-28
VPG3LOG1, K-29
VPG3PHY0, K-29
VPGACTIVE, K-30
VPGDIRTY, K-28, K-31
VPGXLOG0, K-28, K-31
VPGXLOG1, K-28, K-31
VPGXPHY0, K-28, K-31
VPGXPHY1, K-28, K-31
VPOS, Q-43, Q-45
VRFOUND, V-22
VRNF, V-22
VS, Q-41, Q-43
VSYNCP, Q-50
VWFOUND, V-22
WATCHDOG, K-29, K-31
WGATE, V-14
WITHIO, P-19
WRCMD, V-14
WREN, K-23
WTREQ, V-14
XINV, V-24
XPOSLSB, Q-46, Q-50
XPOSMSB, Q-46, Q-50
XSCL, Q-41, Q-43
YINV, V-24
YSCL, Q-41, Q-43
RENUMBER, 10-38
RESQ, L-126
revers, 21-7
Rijmasker, Q-33
RLA, L-41
RMB0, L-94
RMB1, L-95
RMB2, L-95
RMB3, L-95
RMB4, L-96
RMB5, L-96
RMB6, L-96
RMB7, L-97
RND(), 10-45
ROL, L-42, L-97
ROLQ, K-13, L-127
ROM, 6-5
  Upgraden, 6-10
  Versie, 6-7
ROR, L-43, L-98
RORQ, K-13, L-128
ROW, L-99
RRA, L-43
RSVQ, L-129
RTI, L-44, L-99
RTS, L-44, L-100
RXNORMAL, U-10
RXONLYONE, U-11

SAVE, O-13
SAX, L-45
SBC, L-46, L-100, L-130
SBCQ, K-13, L-131
SBX, L-46
Scherm
  Aansluiten, 3-6
  PAL/NTSC instellen, 3-11, 5-6
Schermeditor, 4-11
Schermtekst- en kleurenarrays, B-8
Schijfstations, 7-3
  Aansluiten, 3-8
  Double-densityschijven (DD), 7-8
  Een schijf formatteren, 7-9
  High-densityschijven (HD), 7-8

Opstartbare schijven, 7-10
Unitnummers en drivenummers, 7-3
Virtuele schijfimages, 7-4, 7-6
SD-kaart, 17-5
SD-kaarten, 5-9
Bestanden overzetten, 8-3
Locaties, 3-8, 5-9
SD-kaarthulpprogramma, 5-10
SD2IEC device, 4-15
SD2IEC-apparaat, 3-8, 7-3
SEC, L-47, L-101
SED, L-47, L-101
SEE, L-102
SEI, L-48, L-102
set16bitcharmode, 21-6
setcharsetaddr, 21-4
setcolramoffset, 21-4
setextendedattrib, 21-6
sethotregs, 21-6
setmapedpal, 21-10
SETMEMORY, O-24
setpalbank, 21-9
setpalbanka, 21-9
setpalentry, 21-11
SETPC, O-22
setscreenaddr, 21-3
setscreensize, 21-5
SHA, L-48
SHX, L-48
SHY, L-49
SLO, L-49
SMB0, L-103
SMB1, L-103
SMB2, L-103
SMB3, L-104
SMB4, L-104
SMB5, L-104
SMB6, L-105
SMB7, L-105
Spellen
Raad het getal, 10-33
SRE, L-50
STA, L-51, L-105, L-132
STARTTX, U-10
STEP, 10-13, O-18
STOP, 10-39
STOPTX, U-10
STQ, K-13, L-132
string, 10-13
STX, L-51, L-106
STY, L-51, L-106
STZ, L-107
SYNTAX ERROR, 10-4

TAB, L-107
TAS, L-52
TAX, L-52, L-108
TAY, L-52, L-108
TAZ, L-108
TBA, L-109
teken, 10-13
tekenset, 10-13
textcolor, 21-7
Textuurschaling, P-14
THEN, 10-30
TIB, V-8, V-11
Toetsenbord, 4-3
ALT, 4-7
CAPS LOCK, 4-7
CLR HOME, 4-6
CTRL, 4-4, 4-8, E-3
Cursortoetsen, 4-5, E-9
Escape-reeksen, 4-11, E-6
Functietoetsen, 4-7
HELP, 4-7
INST DEL, 4-6
matrix, F-16
MEGA-toets, 4-6, 4-9, 12-4
NO SCROLL, 4-7
PETSCII-codes en CHR$, B-48, D-3
Pijltjestoetsen, 4-5
RESTORE, 4-4
RETURN, 4-3

RUN STOP, 4-4
Schermcodes, C-3
SHIFT LOCK, 4-4
Shifttoetsen, 4-3, 4-7, E-6
toetsenbord, 17-3
togglecase, 21-6, 21-7
TRACE, O-24
Track at once, V-11
Track Information Block, V-8, V-11
Track-DMA, V-11
TRANSFER, O-14
TRB, L-109
Trefwoorden, B-3
TSB, L-110
TSX, L-53, L-110
TSY, L-111
TXA, L-53, L-111
TXS, L-54, L-112
TYA, L-54, L-112
Type mismatch error, 10-17
TYS, L-112
TZA, L-113

UARTDIVISOR, O-19
underline, 21-8
Unitnummer (IEC-apparaten), 7-3, 7-10

variabele, 10-8
numeriek, 10-15
string, 10-15
Vensters
Escape-reeksen, 4-11
WINDOW-commando, B-295
vergelijkingsoperatoren, 10-30
VERIFY, O-14
vline, 21-12

Waarschuwingen
Extra Ignored, 10-22
WATCHPOINT, O-24
wetenschappelijke notatie, 10-45
wherex, 21-15
wherey, 21-15

XAA, L-54

zinnebeeld, 10-35
zinnebeeldmodus, 10-35

Over het grote Mega65-boek

C64- en C65-programma- en randapparatuurcompatibiliteit, prachtig geluid, echte arcadeklasse-grafiek, een prachtig afgewerkt stuk hardware met volledig mechanisch toetsenbord en uitgebreide netwerkmogelijkheden, en een van de snelste verkrijgbare 6502-klasseprocessors maken de MEGA65 werkelijk uniek voor thuis-, bedrijfs- of educatief gebruik.

Alles over de Mega65 verzamelt in één groot boekdeel alle informatie die u over uw MEGA65 moet weten. Door alle essentiële informatie uit de diverse MEGA65-gebruikershandleidingen en -naslagwerken op één plek te bundelen, biedt dit boek gedetailleerde informatie over elk onderwerp, waaronder BASIC-programmering, geluid, grafiek, netwerken, assembleertaalprogrammering, platformonafhankelijke ontwik- keling, waaronder het gebruik van hogere programmeertalen zoals C of KickC, de krachtige 8-bitschipset van de MEGA65 en hoe u de krachtige FPGA-core kunt gebruiken om andere computersystemen te implementeren.

Dit boek is het onmisbare naslagwerk voor de MEGA65, dat elke gebruiker, of hij nu beginner of gevorderd is, bij zich zou moeten hebben wanneer hij het volledige potentieel van de MEGA65 verkent.

Met auteurs waaronder professionele schrijvers, universiteitsdocenten en 8-bitexperts is de inhoud zowel toegankelijk als uitgebreid. Beginners vinden gedetailleerde uitleg over hoe ze kunnen beginnen, terwijl gevorderde gebruikers zeer gedetailleerde technische informatie vinden over de innerlijke werking van elk aspect van de MEGA65.

Kortom, dit boek is ontworpen om u het maximale uit de uitgebreide mogelijkheden van de MEGA65 te laten halen.

ISBN 123-45-67890-12-8

het MEGA Museum of Electronic Games & Art e.V. http://mega65.org

Redactie: Univ. hoofddoc. Dr. Paul Gardner-Stephen.

Vertaling: Daniël Mantione.