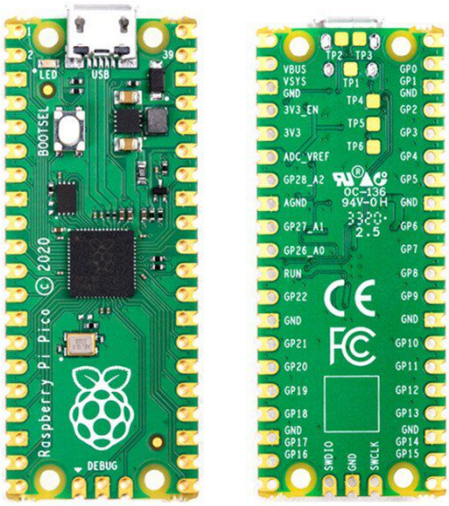
NEDERLANDS Ver. 12-5-2025

PicoMite

Handleiding
MMBasic Ver 6.00.01 (jan 2025)
voor de
Raspberry Pi Pico
Raspberry Pi Pico 2
Raspberry Pi Pico W
Raspberry Pi Pico 2 W
De Modules gebruiken de
RP2040 of de RP2350 microcontrollers



Voor updates van deze handleiding en meer details over MMBasic ga naar:

Opm: Als de link niet werkt dan eerst kopieeren en dan plakken.

Voor update firmware en handleidingen:	https://geoffg.net/picomite.html
Voor VGA:	https://geoffg.net/picomitevga.html
Voor WEB:	https://geoffg.net/webmite.html
MMBasic Help:	https://www.c-com.com.au/mmhelp/
Forum Picomite:	https://www.thebackshed.com/forum/ViewForum.php?FID=16
Forum Picomite: Site en dan Forum:	TheBackshed.com. Forum of "zoeken"
MM Edit 4 of 5:	https://www.c-com.com.au/MMedit.htm

Over

Peter Mather (matherp op het Back Shed Forum) porteerde MMBasic naar de Raspberry Pi Pico, schreef de stuurprogramma's vanwege de hardwarefuncties en heeft het project vooruit geholpen. De MMBasic-interpreter en deze handleiding zijn geschreven door Geoff Graham (http://geoffg.net). Mick Ames (Mixtel90 op het Back Shed Forum) schreef de PIO-compiler en de bijbehorende documentatie samen met het uitvoeren van een enorme hoeveelheid testen.

Steun

Ondersteuningsvragen moeten worden gesteld op het Back Shed-forum (https://www.thebackshed.com/forum/ViewForum.php?FID=16) waar veel enthousiaste MMBasic gebruikers zijn die maar al te graag willen helpen. De ontwikkelaars van de PicoMite-firmware is ook vaste klant op dit forum. Je aanmelden is gratis en duurt ongeveer een dag na registratie.

Copyright en dankbetuigingen

De PicoMite-firmware en MMBasic zijn copyright 2011-2024 door Geoff Graham en Peter Mather 2016-2024.

1-Wire Support is copyright 1999-2006 Dallas Semiconductor Corporation en 2012 Gerard Sexton.
FatFs (SD-kaart) driver is copyright 2014, ChaN.
WAV-bestandsondersteuning is copyright 2019 David Reid.
JPG-ondersteuning is te danken aan Rich Geldreich
De pico-sdk is copyright 2021 Raspberry Pi (Trading) Ltd.
TinyUSB is copyright tinyusb.org
De VGA-stuurprogrammacode is afgeleid van het werk van Miroslav Nemecek.

De gecompileerde objectcode (het **.uf2** bestand) voor de PicoMite is gratis software:
Alsjeblieft u kunt het gebruiken of herdistribueren zoals u wilt.
De broncode is beschikbaar op GitHub (https://github.com/UKTailwind/PicoMite) en kan worden onder voorwaarden vrij te gebruiken (zie de header op de bronbestanden).

Dit programma wordt verspreid in de hoop dat het nuttig zal zijn, maar ZONDER ENIGE GARANTIE, zonder zelfs de impliciete garantie van : VERKOOPBAARHEID of GESCHIKTHEID VOOR EEN BEPAALD DOEL.

Deze handleiding

De auteur van deze handleiding is Geoff Graham. Het is gebaseerd op voorlopig materiaal geschreven door Mick Ames en is gedistribueerd onder een Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Australia-licentie (CC BY NC-SA 3.0)

Dit is een vertaling vanuit de Engelse handleiding pdf en de afbeeldingen komen van het internet en zijn ook met ProfiCAD Home gemaakt. Er zijn veel voorbeeld programma's opgenomen om uit te proberen. Voor de orginele tekst: http://geoffg.net/picomite.html bij Downloads of Other.

Er zijn ook een paar onderwerpen van de volgende **Versie 6.00.02 (mei 2025)** in deze handleiding opgenomen waar men mogelijk de firmware eerst voor moet updaten. Maak een goede **backup** op een **externe SD drive B:** of kopieer het programma vanuit de Terminal en selecteer en kopieer het naar een andere drive, omdat bij een update het **flash** geheugen **drive A:** gewist kan worden.

Heeft men nog niet eerder met de programmeer taal Basic gewerkt dan kan men het beste eerst bestuderen: "**Bijlage I Programmeren in Basic. Een zelfstudie.**" op **blz. 343** t/m **blz. 387**.

Inhoud

Inleiding 11
  Aansluitingen modules 13
Firmwareversies 14
  Bestandsnamen 14
Beginnen 15
  De PicoMite firmware laden 15
  Virtuele seriële poort 16
  Terminal emulator 16
  De console 16
  Sommige testen 16
  Windows 7 en 8.1 18
  Apple Macintosh 18
  Linux 18
Snelstart handleiding 19
  Een eenvoudig programma 19
  Een knipper LED 19
  Uitleg over programmeren in de BASIC taal 20
PicoMite hardware 20
  Modules van derden 24
  Nieuwe CPU-variaties 24
  I/O pin limieten 26
  Voedingssysteem 26
  Energieverbruik en kloksnelheid 26
MMBasic gebruiken 27
  Opdrachten en programma invoer 27
  Programmastructuur 27
  De opdrachtregel bewerken 28
  Sneltoetsen 28
  Programma's opslaan met lokale FLASH en VAR SAVE commands 28
  DRIVE drive$ 30
  Een lopend programma onderbreken 32
  Opties instellen 32
  Opgeslagen variabelen 32
  Waakhond Timer 33
  PIN beveiliging 33
  De bibliotheek 34
  Programma initialisatie 35
  MM.STARTUP 35
  MM.PROMPT 35
  MM.END 36
Edit op volledig scherm 36
  Kleur gecodeerde editor weergave 38
Variabelen en uitdrukking 38
  Variabelen 38
  Constanten 39
  OPTIE STANDAARD (OPTION DEFAULT) 40
  OPTIE EXPLICIET 40

Inhoud

DIM en LOCAL 40
STATISCH 42
CONST 43
Uitdrukkingen en operatoren 43
Rekenkundige operatoren: 44
Shift operators: 44
Logische operatoren: 44
Tekenreeksoperators: 45
Drijvende komma en gehele getallen mengen 45
64-bits niet-ondertekende gehele getallen 45
Subroutines en functies 46
Subroutines 46
Functies 47
Argumenten doorgeven op basis van referentie 47
Arrays doorgeven 48
Vroege uitstap 49
Herhaling 49
Video uitgang 51
VGA video 51
HDMI video 52
VGA/PS2 Referentieontwerp (Raspberry Pi Pico) 54
De I/O-pinnen gebruiken 55
Digitale ingangen 55
Analoge ingangen 55
Ingangen tellen 56
Digitale uitgangen 56
Pulsbreedtemodulatie 57
Communicatie-interfaces (Serieel, SPI en I2C) 57
Interrupts 57
Geluidsuitgang 59
Laagdoorlaatfilter 59
WAV-bestanden afspelen 59
Sinusgolven genereren 60
Gespecialiseerde audio-uitgang 60
Met behulp van PLAY 60
PLAY STOP Hulpprogramma's 62
Speciale apparaatondersteuning 63
Infrarood Afstandsbediening Decoder NEC 38kHz 63
Infrarood Afstandsbediening Zender 12 bits Sony 64
Temperatuur meten DS18B20 65
Vochtigheid en temperatuur meten DHT22 66
Interface voor realtimeklok met batterij DS3231 67
Afstand meten HC-SR04 68
LCD scherm KS0066, HD44780 69
Toetsenbordinterface 4x3 en 4x4 keypad 70
WS2812 ondersteuning 71
PS2 toetsenbord 5V 72

Inhoud

Lcd scherm als console uitgang 73
PicoMiteVGA Console uitgang 73
Ondersteuning voor SD-kaarten met bijbehorende opdrachten 73
Een SD-kaart aansluiten 73
MMBasic-ondersteuning 74
Een bestand weergeven 77
XModem overdracht 77
Afbeelding laden en opslaan 77
Voorbeeld van sequentiële I/O 78
Willekeurige bestands-I/O 79
Displaypanelen 81
Op SPI gebaseerde weergavepanelen 81
I2C gebaseerde LCD-panelen 83
Op SSD1963 gebaseerde lcd-schermen 84
Een op SSD1963 gebaseerd LCD-paneel aansluiten 85
Een op SSD1963 gebaseerd LCD-paneel configureren 86
8 en 9 inch beeldschermen 87
Achtergrondverlichting 87
Voorbeeld LCD-paneelconfiguratie ILI9341 met aanraakscherm 88
Aanraakschermen 89
Aanraking configureren 90
Het touchscreen kalibreren 90
Aanraakfuncties 91
Het GUI BEEP commando 91
Touch Interrupts zonder geavanceerde GUI-bediening 91
Raak Onderbrekingen aan met geavanceerde GUI-bediening 92
Grafische opdrachten en functies 93
Kleuren 93
Lettertypen 95
Alleen lezen variabelen 96
Tekenenopdrachten 96
De tekenopdrachten zijn 97
Geroteerde tekst 98
Transparante tekst 98
Framebuffers en lagen 98
BLIT- en Sprite commando's..... 99
Afbeelding laden 100
Voorbeeld 101
PicoMite geavanceerde grafische afbeeldingen 102
Besturingselementen definiëren 102
Kader 103
LED 103
Selectievakje 103
Druk op de knop 104
Schakelaar 104
Radio knop 104
Displaydoos 104

Inhoud

Tekstvak 105
Nummervak 105
Opgemaakt nummervak 106
Spin Box 108
Ondertiteling 108
Ronde meter 108
Staafmeter 109
Gebied 110
Interactie met bedieningselementen 110
MsgBox() 111
WiFi- en internet functies 113
Verbinding maken met een WiFi-netwerk 113
Toegang tot console op afstand 114
Bestandsoverdracht 114
De tijd krijgen 115
Een webserver implementeren 116
Gegevens in de webpagina invoegen 117
Meerdere pagina's verzenden 117
Een afbeelding (image) verzenden 118
Pagina niet gevonden (404) Reactie 118
Live grafische gegevens op een webpagina 119
Een complete server voor algemeen gebruik 120
Een typische webpagina 120
Invoervelden en besturing 121
Een TCP-client implementeren 122
UDP gebruiken 123
E-mails verzenden 124
Basis 64 codering 126
MQTT client 126
Ping 127
Audio streamen 127
Long Strings 127
Variabelen met Long String 127
Samenvatting van de Commando's en Functies 128
Geavanceerde grafische programmeertechnieken 129
De gebruiker moet de controle hebben 129
Programmastructuur 129
Ongeldige bedieningselementen uitschakelen 130
Constanten gebruiken voor controlereferentienummers 131
Het hoofdprogramma loopt nog 131
Interrupts en SELECT CASE statements gebruiken 132
Touch-up onderbreken (Interrupt) 132
Houd onderbrekingen (Interrupt) erg kort 133
Meerdere schermen 134
Meerdere Interrupts 135
Basistekenopdrachten gebruiken 136
Overlappende bedieningselementen 136

Inhoud

MMBasiskenmerken 137
  Naamgevingsconventies 137
  Constanten 137
  Uitvoeringskenmerken 138
  Compatibiliteit 138
Vooraf gedefinieerde Alleen Lezen Variabelen 139
MM.xxxx 139
 A ... 139 F ... 139 O ... 143
 C ... 139 H ... 143 V ... 143
 D ... 139 I ... 139 W ... 143
 E ... 139 M ... 143
Opties 144
 A ... 144 E ... 147 K ... 149 R ... 153 V ... 156
 B ... 145 F ... 147 L ... 150 S ... 154 W ... 157
 C ... 145 G ... 148 M ... 151 T ... 156
 D ... 146 H ... 148 P ... 152 U ... 156
Operatoren 158
  Rekenkundige OPERATOR 158
  Shift OPERATOR 158
  Logische OPERATOR 158
  Tekenreeks operator 158
Commando's 159
 A ... 159 E ... 174 I ... 195 N ... 222 S ... 239 W ... 255
 B ... 161 F ... 177 K ... 198 O ... 222 T ... 251 X ... 261
 C ... 164 G ... 183 L ... 199 P ... 225 U ... 254
 D ... 170 H ... 195 M ... 207 R ... 236 V ... 254
Functies 263
 A ... 263 E ... 269 I ... 272 M ... 275 S ... 282
 B ... 263 F ... 269 J ... 273 O ... 279 T ... 286
 C ... 264 G ... 271 K ... 273 P ... 279 U ... 287
 D ... 265 H ... 272 L ... 273 R ... 282 V ... 287
Verouderde commando's en functies 288
Bijlage A – Seriële communicatie en voorbeeld 290
  I/O pinnen 290
  Option instellingen 290
  Commando's 291
  Het OPEN commando 291
  Voorbeelden 292
  Lezen en schrijven 292
  Onderbrekingen (Interrupts) 293
Bijlage B – I2C communicatie 294
  I/O pinnen 294
  Errors 294
  7-bits adressering 294
  I2C Master opdrachten 295
  I2C Slave opdrachten 296
  Voorbeelden 297

Inhoud

Bijlage C – OneWire communicatie 298
  Voorbeelden 299
Bijlage D – SPI communicatie en voorbeeld..... 300
  I/O-pinnen 300
  SPI openen 300
  Verzendformaat 301
  Standaard verzenden/ontvangen 301
  Bulk verzenden/ontvangen 302
  SPI Sluiten 302
  OPTION SYSTEM SPI 302
  Voorbeeld 1 303
Bijlage E - Regex syntaxis 304
Bijlage F - Het PIO programmeer pakket 306
  Inleiding tot de PIO 306
  Overzicht van PIO 306
  PIO programmeren 307
  PIO ASSEMBLE 308
  PIO PROGRAMMA REGELS 308
  PIO configureren 308
  FREQUENTIE 308
  PIN CONTROLE 308
  EXECUTE CONTROL (Voer controle uit) 309
  SHIFT CONTROLE (Schuiven met instructies IN en OUT Data FIFO).. 309
  DE CONFIGURATIE VAN DE STATE MACHINE SCHRIJVEN 310
  EEN STATE MACHINE STARTEN EN STOPPEN 310
  PIO ASSEMBLE V6.00.02b8 Commando's 310
  VOORBEELD PROGRAMMA 1 (Frequentie = 2029Hz uitgangs signaal) .. 311
  FIFO's 311
  VOORBEELD PROGRAMMA 2 (NES controller CD4021 met 8 drukknoppen) 312
  DMA NAAR EN VAN DE FIFOS 314
  VOORBEELD PROGRAMMA 3 (6 kanaals Logic Analyser) 315
  VOORBEELD PROGRAMMA 4 (Uitgangen met Frequentie = 200MHz) 316
  VOORBEELD PROGRAMMA 5 (Alleen voor RP2350) 318
PIO assembler met MMBasic Ver 6.00.02(beta) 320
  Dit zijn de 9 instructies voor de PIO 321
  Aan elk van de 4 state machines zijn 4 registers gekoppeld 321
PIO ASSEMBLE 321
  Wat voorbeeld code nieuwe PIO assembler 322

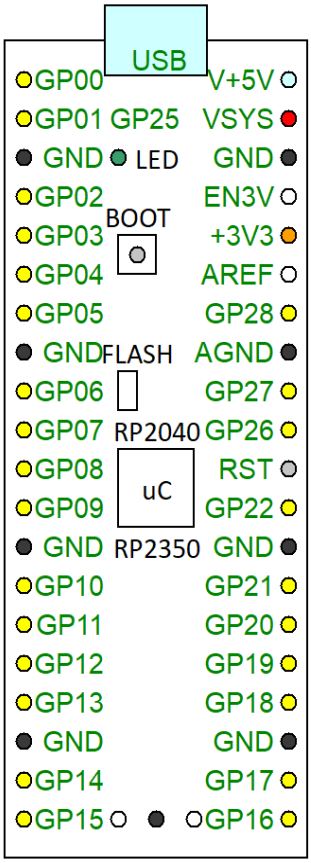
Inhoud

PIO Instructieset 324
  Samenvatting overzicht instructieset uit data blad 324
  JMP 325
  WAIT 326
  IN 328
  OUT 329
  PUSH 330
  PULL 331
  MOV (naar RX) 332
  MOV (van RX) 333
  MOV 334
  IRQ 336
  SET 338
  Functionele details 339
  Site Set 339
Bijlage G - Sprites 340
  ALLEEN VGA EN HDMI VERSIES 340
Bijlage H - Special Keyboard Keys 342
Bijlage I – Programmeren in BASIC - Een zelfstudie 343
  Commando prompt 343
  Structuur van een BASIC programma 344
  Opmerkingen 345
  Het PRINT commando 345
  Variabelen 347
  Uitdrukkingen 348
  De IF verklaring 349
  FOR lussen 351
  Tafel van vermenigvuldiging 352
  DO LOOP 353
  Console invoer 354
  GOTO en Labels 355
  Testen op priemgetallen 356
  Arrays 358
  Integers 360
  Strings 360
  Strings manipuleren 362
  Wetenschappelijke notatie 363
  DIM opdracht 364
  Waarom zijn deze twee commando's belangrijk? 365
  DIM LOCAL STATIC 366
  Constanten 369
  Subroutines 370
  Functies 373
  Lokale variabelen 374
  Statische variabelen 374
  Bereken dagen 376
Enkele Hints & Tips voor het schrijven van efficiënte code 379
  "Genees uzelf, dokter" 380

Inhoud

Enig inzicht in de namen van variabelen 381
Bereken dingen niet twee keer - gebruik een variabele 381
Over Absoluten en CONST's 382
Meerdere tests in IF-statements 383
Testen op toegestane tekens in een string 384
Een moment van hypocrisie 386
PWM regelingen 388
SERVO 388
VOORBEELD 1 Duty Cycle 389
VOORBEELD 2 PWM en SETTICK 390
PWM SYNC aan het werk zetten 393
Hier wat programma's voor een rotary encoder 394
Nieuwe versie van Sub rint: 395
Test programma voor een rotary encoder 395
Uitvoer op een terminal 396
Digitale ingangen 396
OneWire Voorbeeld 397
I2C commando's en functions met PicoMite Basic 400
I2C programma 401
I2C programma MCP23017 module 401
ADS1115 module met I2C 402
PicoMite Basic programma ADS1115 om PH en temperatuur te meten 404
I2C scanner 407
PCF8574-module 8 Digitale in of uitgangen 410
DHT20 of AHT20 I2C Temperatuur en Vochtigheid sensor 411
LCDI2CLib test voor bijna elk type PCF8574 I2C naar LCD-module. 415
Aangepaste grafische afbeeldingen maken op 1602/2004 LCD 418
COMMANDO's en BASIC TREFWOORDEN uitgeschreven 422
FUNCTIES, STRINGS en OPERATORS uitgeschreven 437
ASCII tabel 445
Een aantal sjablonen voor een Raspberry Pi Pico module 447
Alle hardware I/O functies RP2040 en Chip pin nummers 448
COMMANDO's en BASIC TREFWOORDEN V6.00.01 449
FUNCTIES, STRINGS en OPERATORS V6.00.01 450
COMMANDO's en BASIC TREFWOORDEN V6.00.02RC23 451
FUNCTIES, STRINGS en OPERATORS V6.00.02RC23 452
Option voorbeelden 453
Diverse geheugen overzichten 454
Vanaf versie V6.00.02(beta) enkele extra mogelijkheden 455
Olimex bord met HDMI en USB keyboard configureren methode 1 of 2 456
List Commands PicoMiteHDMI MMBasic USB RP2350A 457
List Functions PicoMiteHDMI MMBasic USB RP2350A 458
List Commands PicoMiteVGA MMBasic USB RP2040 Edition V6.00.02 .. 459
List Functions PicoMiteVGA MMBasic USB RP2040 Edition V6.00.02 .. 460
Overzichten en memory PicoMiteVGA MMBasic USB RP2040 V6.00.02 ... 461
Beschrijving van connector pinnen voor een Olimex print 462
Einde 462

Inleiding



RP2040 en RP2350 Pinnen

De **PicoMite** firmware voor de **Raspberry Pico Pico** en **Pico 2** is een compleet besturingssysteem met een compatibele Microsoft BASIC en uitgebreide hardware ondersteuning, waaronder touchscreen, SD-kaarten, WiFi/Internet en nog veel meer.

MMBasic is een Microsoft BASIC is een volledige implementatie van de BASIC taal met floating point, integer en string variabelen, arrays, long variabelenamen, een ingebouwde programma editor en vele andere functies.

Met MMBasic kunt u de I/O-pinnen bedienen en communicatie gebruiken protocollen zoals I2C of SPI om gegevens van verschillende sensoren te krijgen. Jij kan gegevens weergeven op goedkope kleuren-LCD-schermen, spanningen meten, detecteren digitale ingangen en stuuruitgangspinnen om lichten, relais, enz. Allemaal van binnenuit deze goedkope microcontroller. De PicoMite firmware is volledig gratis te downloaden en te gebruiken.

De nadruk bij MMBasic ligt op gebruiksgemak en ontwikkeling. De ontwikkelingscyclus is erg snel met de mogelijkheid om direct over te schakelen van bewerken naar uitvoeren. Fouten worden vermeld in gewoon Engels en wanneer er een fout optreedt een enkele toetsaanslag roept de ingebouwde editor op met de cursor op de regel die de fout veroorzaakte.

In samenvatting de kenmerken van de PicoMite zijn:

- **De BASIC-interpreter is volledig uitgerust** met drijvende komma, 64-bits gehele getallen en stringvariabelen, lang variabele namen, arrays van floats, integers of strings met meerdere dimensies, uitgebreide string handling en door de gebruiker gedefinieerde subroutines en functies.
  Meestal voert het een programma uit tot 100.000 regels per seconde. MMBasic maakt het inbedden van gecompileerde C-programma's mogelijk voor hoogwaardige functies en het lopende programma kan worden beveiligd tegen vermelding of wijziging door een pincode.

- **Ondersteuning voor alle Raspberry Pi Pico input/output pinnen.** Deze kunnen onafhankelijk worden geconfigureerd als, digitale ingang of uitgang, analoge ingang, frequentie- of periodemeting en -telling. Binnen MMBasic de I/O-pinnen kunnen dynamisch worden geconfigureerd als in- of uitgangen met of zonder pullups of pulldowns.
  MMBasic-commando's genereren pulsen en kunnen worden gebruikt om gegevens parallel over te dragen. Onderbrekingen kunnen zijn, gebruikt om te melden wanneer een ingangspin van status is veranderd. PWM-uitgangen kunnen worden gebruikt om verschillende geluiden te creëren, stuur servo's of genereer computergestuurde spanningen voor het aansturen van apparatuur die een analoge ingang gebruikt (bijv. motorcontrollers). Bovendien zijn pinnen toegankelijk die niet zichtbaar zijn op de Raspberry Pi Pico met MMBasic waardoor het kan worden gebruikt op andere modules die de **RP2040** of **RP2350** microcontrollers gebruiken.

- **Ondersteuning voor TFT LCD-schermpanelen** met parallelle, SPI- en I2C-interfaces waardoor de BASIC programma om tekst weer te geven en lijnen, cirkels, vakken, enz. in maximaal 16 miljoen kleuren te tekenen. Resistieve aanraking controllers op deze panelen worden ook ondersteund, waardoor ze kunnen worden gebruikt als geavanceerde invoerapparaten.
  LCD-panelen kosten slechts $7 en bieden een goedkope, hightech grafische gebruikersinterface. Voor hogere snelheid en grotere resolutie op SSD1963 gebaseerde TFT-schermen worden ook ondersteund.

LCD modules die de KS0066 gebruiken, HD44780 of HD1602 2x16, 4x16, 4x20
Deze werken met **LCD** parallel of met een extra **I2C module** met een **PCF8574** en de basic lib: **"LCDI2CLIB.bas".** Zie aan het einde van deze handleiding bij voorbeelden.

Grafische LCD schermen met de OPTION instelling in PicoMite Basic met ondersteuning.

LCDPANEEL met SPI databus:	OR:**L**=Landschap,**RL**=Omgedraait LS,**P**=Portret,**RP**=OmgedPor		
OPTION LCDPANEL **ILI9341,**	OR, DC, RESET, CS, Backlight (optie)	TFT	**320 x 240**
OPTION LCDPANEL **ILI9163,**	OR, DC, RESET, CS, Backlight (optie)		**128 x 128**
OPTION LCDPANEL **ILI9481,**	OR, DC, RESET, CS, Backlight (optie)		**480 x 320**
OPTION LCDPANEL **ILI9481IPS,**	OR, DC, RESET, CS, Backlight (optie)	3,5" IP	**480 x 320**
OPTION LCDPANEL **ILI9488,**	OR, DC, RESET, CS, Backlight (optie)		**480 x 320**
OPTION LCDPANEL **ILI9488W,**	OR, DC, RESET, CS, Backlight (optie)	3,5"	
OPTION LCDPANEL **N5110,**	OR, DC, RESET, CS [,contrast]		**84 x 48**
OPTION LCDPANEL **SSD1306SPI,**	OR, DC, RESET, CS [,offset]	0.96"-1.3"	**128 x 64**
OPTION LCDPANEL **SSD1331,**	OR, DC, RESET, CS		**96 x 64**
OPTION LCDPANEL **ST7735,**	OR, DC, RESET, CS, BackLight (optie)		**160 x 128**
OPTION LCDPANEL **ST7735S,**	OR, DC, RESET, CS, BackLight (optie)		**160 x 80**
OPTION LCDPANEL **ST7735S_W,**	OR, DC, RESET, CS ,BackLight (optie)		**128 x 128**
OPTION LCDPANEL **ST7789,**	OR, DC, RESET, CS, BackLight (optie)		**240 x 240**
OPTION LCDPANEL **ST7789_135,**	OR, DC, RESET, CS, BackLight (optie)	1.1" IP	**240 x 135**
OPTION LCDPANEL **ST7789_320,**	OR, DC, RESET, CS, BackLight (optie)		**320 x 240**
OPTION LCDPANEL **ST7796S,**	OR, DC, RESET, CS, BackLight (optie),INV.	**V6.00.02**	**480 x 320**
OPTION LCDPANEL **GC9A01,**	OR, DC, RESET, CS		**240 x 240**
OPTION LCDPANEL **ST7920,**	OR, DC, RESET		**128 x 64**
OLED LCDPANEEL met I2C databus:	OR:**L**=Landschap,**RL**=Omgedraait LS,**P**=Portret,**RP**=OmgedPor		
OPTION LCDPANEL **SSD1306I2C,**	OR [,offset]		**128 x 64**
OPTION LCDPANEL **SSD1306I2C32,**	OR		**128 x 32**
OPTION LCDPANEL **SSD1306I2C,L,2**	OR: **L,RL,P,RP** met **SSD1315 Oled**	**V6.00.02**	**128 x 64**

LCDPANEEL met parallelle databus:

SSD1963_4	Voor een 4,3-inch scherm.
SSD1963_5	Voor een 5 inch beeldscherm.
SSD1963_5A	Voor een alternatieve versie van het 5 inch.
SSD1963_7	Voor een 7 inch beeldscherm.
SSD1963_7A	Voor een alternatieve versie van het 7 inch.
SSD1963_8	Voor een 8 inch of 9 inch beeldscherm.

- **Volledige ondersteuning voor SD-kaarten.** Dit omvat het openen van bestanden voor lezen, schrijven of willekeurige toegang en programma's laden en opslaan. De firmware werkt met kaarten tot 32GB geformatteerd in FAT16 of FAT32 en de gemaakte bestanden kunnen ook worden gelezen en geschreven op pc's met Windows, Linux of het Mac-besturingssysteem.

- **Programmering en besturing gebeurt via de USB-interface**. Het enige dat nodig is, is een laptop desktop computer met een **VT100**-terminalemulator. Nadat het programma is geschreven en debuggen is de PicoMite kan worden geïnstrueerd om het programma automatisch uit tevoeren bij het opstarten zonder tussenkomst van de gebruiker. De PicoMiteVGA-versie ondersteunt een VGA-monitor en PS2-toetsenbord (in plaats van de USB-interface) waardoor een goedkope en op zichzelf staande "boot to BASIC computer".

- **Een schermvullende editor is ingebouwd in de PicoMite** en kan het hele programma in één sessie bewerken. Het bevat geavanceerde functies zoals kleurgecodeerde syntaxis, zoeken en kopiëren, knippen en plakken van en naar een klembord.

- **Programma's kunnen eenvoudig worden overgezet vanaf een desktop- of laptopcomputer** (Windows, Mac of Linux) met behulp van het XModem-protocol of door het programma via de seriële console-ingang te streamen.

- **Er is een uitgebreide reeks communicatieprotocollen geïmplementeerd**, waaronder **I2C**, asynchroon serieel, **RS232**, **SPI** en **1-Wire**. Deze kunnen worden gebruikt om te communiceren met veel sensoren (**temperatuur**, **vochtigheid**, **acceleratie**, enz.) en voor het verzenden van gegevens naar testapparatuur.

- **De PicoMite heeft ingebouwde commando's om direct te communiceren** met infrarood **afstandsbedieningen**, de **DS18B20** temperatuursensor, **LCD**-displaymodules, **klok** met batterijvoeding, numerieke **toetsenborden** en meer.

- **De stroomvereisten zijn VSYS** = **2,0** tot **5,5 V** bij **10** tot **42 mA** en elke **GP pin** = **3.3V** bij **4 mA** voor de **RP2040** of **GP pin** = **5V** **max** voor **RP2350** met voeding aan. **Pas op met externe power.**

Aansluitingen modules

Gaan we straks modules aansluiten op de PicoMite dan komen we verschillende benamingen tegen bij de voeding pinnen op de modules. Zoals **V+, Gnd, Vcc, Vss, Vdd, Vee** enz.
Wat betekenen deze afkortingen nu allemaal. Een geheugensteuntje is voor een npn transistor met CBE dat de C dicht bij de plus van de voeding zit en de E dicht bij de min van de voeding. Daar volgt dan uit dat **Vcc de plus** is en **Vee of Gnd de min**. Het zelfde voor een Fet met DGS dat de D dicht bij de plus en de S dicht bij de min dus de **Vdd is de plus** en **Vss of Gnd is de min**.

Voor de bus aansluiting I2C:	Klok signaal:	**SCL**	(Klok)
	Data signaal:	**SDA**	(Lezen/Schrijven/Data)
Voor de bus aansluiting SPI:	Klok signaal:	**CLK** of **SCK**	(Klok)
	Data ingang :	**RX**, **DI** of **MISO**	(MasterInputSlaveOutp)
	Data uitgang:	**TX**, **DO** of **MOSI**	(MasterOutputSlaveInp)
	Chip select:	**CS**	(Chip selcect op SPI-bus)
Voor de bus aansluiting Seriële:	Data ingang:	**RX**	(Ontvangen)
	Data uitgang:	**TX**	(Zenden)
Voor de 1-Wire modules:	Data signaal:	**D** of **DAT**	(Lezen/Schrijven/Data)

Verder zijn er nog speciale modules gerichte aansluitingen zoals: **D0-D8**, **Reset**, **RS**, **RW**, **IRQ** enz.

Firmwareversies

De PicoMite-firmware ondersteunt de originele **RP2040**-microcontroller die wordt gebruikt in de Raspberry Pi Pico en de nieuwere **RP2350** gebruikt in de Raspberry Pi Pico 2. De firmware is ook ontworpen om te werken met modules die zijn geproduceerd door andere leveranciers die dezelfde chips gebruiken.
De **RP2350** wordt geleverd in vier subversies: de **RP2350A, RP2350B, RP2354A** en de **RP2354B**.
De **RP2350B** is hetzelfde als de **RP2350A**, behalve dat deze **18 extra I/O-pinnen** heeft (**pinnen GP30 tot GP47**) die worden automatisch beschikbaar gemaakt in MMBasic. Beide chips worden ondersteund door dezelfde PicoMite firmware en werken hetzelfde. In onderstaande tekst zijn dus alle verwijzingen naar de **RP2350** in gelijke mate van toepassing op beide chips en dezelfde firmware kan worden gebruikt.
De **RP2354A** en de **RP2354B** worden niet ondersteund (hoewel dit in de toekomst mogelijk wel het geval zal zijn).

Bestandsnamen

Er zijn twaalf firmware-images in het zip-bestand voor de firmware-distributie. Deze zijn nodig ter dekking de variaties tussen de CPU (**RP2040** of **RP2350**), de toetsenbordondersteuning (**PS2** of **USB**), de video-uitgang (geen, **VGA** of **HDMI**) en de **WiFi/Internet**-mogelijkheid.
Deze functies kunnen worden gebruikt in drie brede toepassingsklassen:

- **Algemene Embedded Controller**. Dit is waar de Pico kan worden gebruikt als brein in een verwarming controller, inbraakalarm, etc. Voor deze toepassing kunt u bijvoorbeeld een firmware-image selecteren die ondersteunt aangesloten LCD-panelen.
- **Opstarten naar BASIC computer**. Dit is een op zichzelf staande computer met toetsenbord en video-uitgang doet denken aan de homecomputers uit de jaren zeventig en tachtig, zoals de **Apple II** en **Tandy TRS-80**. Hiervoor selecteert u een firmware-image die een **PS2**- of **USB**-toetsenbord en VGA- of HDMI-video ondersteunt uitgang.
- **Web/internet-**compatibele controller. Heeft een ingebouwde draadloze interface om verbinding te maken met een WiFi-netwerk en van daaruit toegang tot internet. Deze gebruikt de **Raspberry Pi Pico W** of de **Pico 2 W** en voert de **WebMite**-firmware.

Firmware Bestands Naam Bijvoorbeeld: 10-12-2024 PicoMiteRP2040 V6.00.01.uf2	CPU	Touch LCD Paneel	Toetsenbord		Video-uitgang		WiFi Internet
			PS2	USB	VGA	HDMI	
PicoMiteRP2040	RP2040	√	√				
PicoMiteRP2350	RP2350	√	√				
PicoMiteRP2040USB	RP2040	√		√			
PicoMiteRP2350USB	RP2350	√		√			
PicoMiteRP2040VGA	RP2040		√		√		
PicoMiteRP2350VGA	RP2350		√		√		
PicoMiteRP2040VGAUSB	RP2040			√	√		
PicoMiteRP2350VGAUSB	RP2350			√	√		
PicoMiteRP2350HDMI	RP2350		√			√	
PicoMiteRP2350HDMIUSB	RP2350			√		√	
WebMiteRP2040	RP2040	√	√				√
WebMiteRP2350	RP2350	√	√				√

Er zijn enkele beperkingen aan de VGA-versie en samenvattend zijn deze (RP2040) :

- Geen ondersteuning voor LCD of andere soorten displaypanelen.
  Aanraakinvoer en de geavanceerde GUI-functies die afhankelijk zijn van dit type display, worden ook niet ondersteund.
- Er zijn **kloksnelheden** van **126 MHz** en **252 MHz** beschikbaar.
- Een maximale programmagrootte van **100KB** (in plaats van **120KB** voor de PicoMite).
- Sommige I/O-pinnen zijn niet beschikbaar voor algemeen gebruik omdat ze worden gebruikt voor de VGA-uitgang en het toetsenbord invoer.

De PicoMiteVGA-versie van de firmware is een afzonderlijk laadbaar firmwarebestand en is opgenomen in de PicoMite ZIP-bestand te downloaden van: https://geoffg.net/picomite.html
De rest van deze handleiding concentreert zich op de standaard PicoMite-versie van de firmware met accentuering van opmerkingen opdrachten en functies die niet beschikbaar zijn in de PicoMiteVGA-versie. Voor een volledige beschrijving van de PicoMiteVGA-versie inclusief bedradingsschema zie Bijlage F achter in deze handleiding.

Beginnen

De PicoMite-firmware laden

De Raspberry Pi Pico wordt geleverd met een eigen ingebouwde firmware-lader die gemakkelijk te gebruiken is. Volg gewoon deze stappen:

- **Download de PicoMite-firmware van** http://geoffg.net/picomite.html en pak het bestand uit. Identificeer de firmware die zoiets als **"PicoMiteV6.00.02.uf2"** zou moeten heten.

- **Sluit de Raspberry Pi Pico** met een **USB-kabel** aan op uw **computer** (Windows, Linux of Mac) terwijl door de witte **BOOTSEL** knop op de **Raspberry Pi Pico** ingedrukt te houden.

- **De Raspberry Pi Pico** moet **verbinding maken** met uw computer en **een virtuele schijf maken** (hetzelfde als wanneer u had) aangesloten op een USB-geheugenstick) genaamd **"RPI-RP2"**. Deze schijf zal twee bestanden bevatten die u kunt negeren.

- **Kopieer het firmwarebestand** (met de extensie **.uf2**) naar deze **virtuele schijf.**

- **Wanneer het kopiëren is voltooid**, zal de Raspberry Pi Pico opnieuw opstarten en **een virtuele seriële poort** maken **op uw computer**. De **LED** op de Raspberry Pi Pico **knippert** langzaam om aan te geven dat de **PicoMite-firmware met MMBasic is nu actief**.

Hoewel de virtuele schijf gemaakt door de Raspberry Pi Pico eruitziet als een USB-geheugenstick, is het dat niet, de firmware bestand zal verdwijnen zodra het is gekopieerd en als u een ander type bestand probeert te kopiëren, wordt dit genegeerd.

Door de PicoMite-firmware te laden, wordt het **flash geheugen gewist,** inclusief het huidige programma, alle opgeslagen programma's in flash-geheugenslots en alle opgeslagen variabelen. Zorg er dus voor dat u een back-up maakt van deze gegevens op HD of SD voordat u de firmware upgrade uitvoert.

Virtuele seriële poort

De virtuele seriële poort gemaakt door de PicoMite-firmware werkt als een normale seriële poort, maar werkt via USB.
Windows 10 bevat een stuurprogramma voor deze virtuele seriële poort maar bij andere versies moet u mogelijk een stuurprogramma laden om laat het werken met het besturingssysteem (zie hieronder). Zodra dit is gebeurd, moet u het poortnummer noteren gemaakt door uw computer voor de virtuele serie verbinding. In Windows kan dit worden gedaan door te starten. Apparaatbeheer en controleer de "Poorten (COM & LPT)" voor een nieuwe COM-poort, zoals rechts weergegeven.

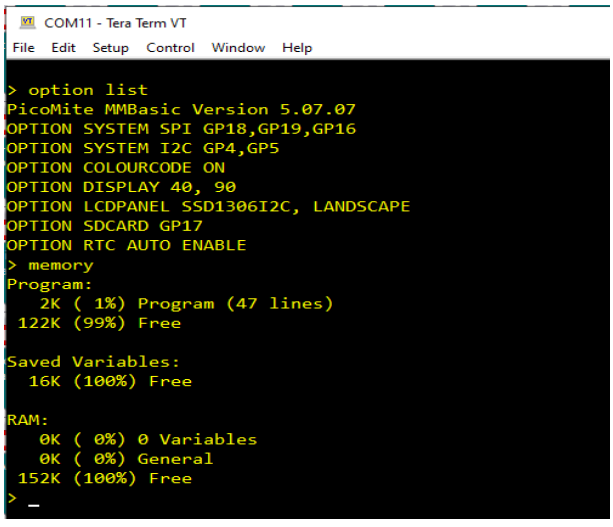
Terminal emulator

U hebt ook een **terminalemulator** programma op uw desktopcomputer nodig. Dit programma werkt als een ouderwetse computerterminal waar tekst wordt weergegeven die is ontvangen van een externe computer en eventuele toetsaanslagen worden verzonden naar de externe computer via de seriële link.
De terminalemulator die u gebruikt, moet **VT100** ondersteunen emulatie, want dat is wat de editor in de PicoMite heeft ingebouwd verwacht.

Voor Windows-gebruikers wordt aanbevolen om **Tera Term** te gebruiken omdat deze een goede **VT100 emulator** heeft en bekend staat om, werken met het XModem-protocol dat u kunt gebruiken om programma's overbrengen van en naar de PicoMite (Tera Term kan worden gedown-load van: http://tera-term.en.lo4d.com).
De schermafbeelding aan de rechterkant toont de instellingen voor Tera Term.
Houd er rekening mee dat de instelling **"Poort:"** varieert afhankelijk van welke **USB poort** waarop uw Raspberry Pi Pico was aangesloten. De PicoMite negeert de **baudrate-instelling**, zodat deze op elke snelheid kan worden ingesteld (andere dan 1200 baud, waardoor de Pico een firmwareup-grade krijgt modus).
Als u **Tera Term gebruikt**, stel dan **geen vertraging in tussen tekens** en als u **Putty gebruikt**, stelt u **de backspace-toets in** om het **backspace-teken te genereren**.

De console

Zodra u de virtuele seriële poort hebt geïdentificeerd en heb je terminal-emulator ermee ver-bonden, jij moet op return op je toetsenbord kunnen drukken en zie de MMBasic-prompt, die groter is dan symbool (bijv. **">"**).



Dit is de console en je gebruikt deze om uit te geven commando's om de PicoMite te configureren, laad de BASIC programma, bewerk en voer het uit.
MMBasic ook gebruikt de console om foutmelding-en weer te geven.
De console is de enige manier om te communiceren met de PicoMite en het programmeren ervan, zo is het belangrijk dat u er verbinding mee kunt maken.

Sommige testen

Hier zijn een paar dingen die u kunt uitproberen om te bewijzen dat u een werkende PicoMite heeft.
Al deze opdrachten moeten worden getypt bij de opdrachtprompt (**">"**). Wat u typt, wordt vet weer-gegeven en de uitvoer van PicoMite wordt weergegeven in normale tekst.

Probeer een eenvoudige berekening:

```
> PRINT 1/7
0.1428571429
```

Na reset of opstart PicoMite Basic:

```
PicoMite MMBasic RP2040 Edition V6.00.01
Copyright 2011-2024 Geoff Graham
Copyright 2016-2024 Peter Mather
```

Kijk hoeveel geheugen je hebt:

```
> memory
Program:
   0K ( 0%) Program (0 lines)
 128K (100%) Free

Saved Variables:
  16K (100%) Free

RAM:
   0K ( 0%) 0 Variables
   0K ( 0%) General
 156K (100%) Free
```

`> ? MM.INFO$(OPTION FLASH SIZE)` Zou de grootte van het flashgeheugen moeten geven.

`> ? MM.INFO$(DISK SIZE)` Zou de grootte van het flashgeheugen moeten geven.

`> ? MM.INFO$(FREE SPACE)` Zou de grootte van het flashgeheugen moeten geven.

Opm: Achter **>** een **?** geeft het zelfde resultaat als **Print**.

Wat is de huidige tijd? Merk op dat de klok van de PicoMite om middernacht begint bij het opstarten.

```
> PRINT TIME$
00:04:01
```

Zet de klok en datum op de huidige tijd en datum en is weer weg na zonder een RTC module.
Dit is voor opslag op de SD-kaart of een RTC module of het flash file systeem.

```
> TIME$ = "10:45:00"
> DATE$ = "29-12-2024"
```

Controleer de tijd nog eens. Zolang er spanning op de module blijft staan loopt de tijd altijd door.

```
> PRINT TIME$ of ? Time$
10:45:09
```

Hoeveel milliseconden zijn verstreken sinds het opstarten:

```
> PRINT TIMER
440782.748
```

Tel tot 20:

```
> FOR a = 1 to 20 : PRINT a; : NEXT a
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

Windows 7 en 8.1

De USB seriële poort maakt gebruik van het CDC-protocol en de stuurprogramma's hiervoor zijn standaard in Windows 10 en 11 en zullen automatisch laden.
De Raspberry Pi Foundation vermeldt Windows 7 of 8.1 als "niet-ondersteund", maar u kunt een tool zoals Zadig gebruiken.
(https://zadig.akeo.ie) om een generiek stuurprogramma voor een "usbser"-apparaat te installeren en waarmee deze computers aansluiten. Dit bericht beschrijft het proces:
https://github.com/raspberrypi/pico-feedback/issues/118

Apple Macintosh

De Apple Macintosh (OS X) is iets eenvoudiger omdat het apparaatstuurprogramma en de terminal emulator zijn ingebouwd.
Start eerst de applicatie 'Terminal' en vermeld bij de prompt de aangesloten seriële apparaten door in te typen:

ls /dev/tty.*.

De USB-naar-serieel-converter wordt weergegeven als /dev/tty.usbmodem12345. Terwijl nog steeds bij de Terminal-prompt u kunt de terminalemulator uitvoeren op 38400 baud met behulp van de opdracht:

screen /dev/tty.usbmodem12345 38400

Standaard zijn de functietoetsen niet correct gedefinieerd voor gebruik in de ingebouwde programma-editor van de PicoMite, dus u zult de bedieningssequenties moeten gebruiken zoals gedefinieerd in de sectie Full Screen Editor van deze handleiding. Vermijden hiermee kunt u de terminalemulator opnieuw configureren om deze codes te genereren wanneer de juiste functie-toetsen zijn: ingedrukt.

Documentatie voor de schermopdracht is hier:

Linux

Zie deze berichten voor Linux:
Voor Linux is er ook een versie in ontwikkeling **MMB4L** maar dat is nog in een begin fase.
De Terminal **Tera Term** is alleen voor Windows en voor Linux is er **Wine**.
Downloaden: https://github.com/thwill1000/mmb4l/releases/tag/0.6.0
README: https://github.com/thwill1000/mmb4l/blob/0.6.0/README.md

Snelstart handleiding

Een eenvoudig programma

Om een programma in te voeren, kunt u het **EDIT** commando gebruiken dat verderop in deze handleiding wordt beschreven. Echter, voor de moment, alles wat je moet weten is dat alles wat je typt zal worden ingevoegd bij de cursor, de pijltjestoetsen verplaatst de cursor en backspace verwijdert het teken voor de cursor. Probeer deze reeks om snel een idee te krijgen van hoe de PicoMite werkt (uw terminalemulator moet **VT100** compatibel zijn) :

- Typ bij de opdrachtprompt **EDIT** gevolgd door de **ENTER**-toets.
- De editor moet opstarten en u kunt deze regel invoeren: `PRINT "Hallo Wereld"`
  Met kleuren: `OPTION COLOURCODE ON` en Scherm afmeting: `OPTION DISPLAY 43,89` Reg.,Teken.
- Druk op de **F1**-toets in uw terminalemulator (of **CTRL-Q** die hetzelfde zal doen). Dit vertelt de editor om uw programma op te slaan en naar de opdrachtprompt te gaan.
- Typ bij de opdrachtprompt **RUN** gevolgd door de **ENTER**-toets.
- Je zou het bericht moeten zien: `Hallo wereld`

Gefeliciteerd. U hebt zojuist uw eerste programma op de PicoMite geschreven en uitgevoerd.
Als je **EDIT** opnieuw typt u bent weer terug in de editor waar u uw programma kunt wijzigen of toevoegen.

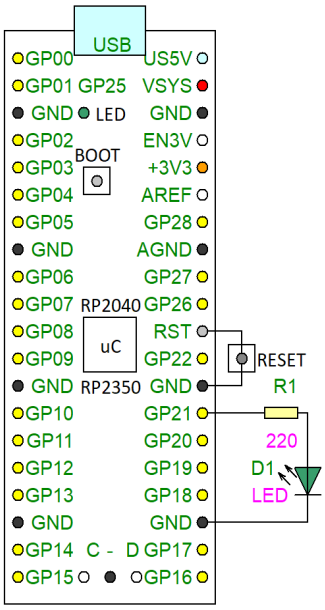
Een knipper LED

Sluit een **LED** aan op **pin GP21** en **GND** (nummers staan op de onderkant print of blz 20) zoals weergegeven in het schema aan de rechterkant.
GP21 is vrije pin. Heeft men geen **LED D1** en **R1** dan de led op de module gebruiken(**GP25**). Deze pin is nog ingebruik door de interne knipper led.

`> Option heartbeat off` Of herstellen: `> Option heartbeat on`

GP25 is nu beschikbaar. Of doe nu: `List pins` (v.a.**V.6.00.02**) lijst pinnen.
GP25 is nu de **LED** die men in het volgende voorbeeld kan gebruiken.
Gebruik vervolgens het **EDIT** commando om het volgende programma in te voeren: Daarna **F1** of **F2** vanuit **EDIT**. Stop het programma met **CTRL+C**

```
SETPIN GP21, DOUT ' Stel pin GP21(25) in als uitgang.
DO                ' Start loop.(Begin programma)
  PIN(GP21) = 1   ' Maak de uitgang GP21(25) wordt hoog.
  PAUSE 300       ' Een pauze van 300 mS = 0.3 sec.
  PIN(GP21) = 0   ' Maak de uitgang GP21(25) wordt laag.
  PAUSE 300       ' Een pauze van 300 mS = 0.3 sec.
LOOP              ' Einde programma en ga weer naar DO.
```



Knipper LED met GP21

Wanneer u dit programma hebt opgeslagen en uitgevoerd, zou u moeten worden begroet door de **LED** die aan en uit knippert. Het is niet een geweldig programma, maar het illustreert wel hoe de PicoMite via uw programmering kan communiceren met de fysieke wereld.
Het programma zelf is eenvoudig. De eerste regel stelt **pin GP21** in als uitgang. Dan komt het programma in een continu lus waarbij de output van die pin hoog is ingesteld om de **LED** in te schakelen, gevolgd door een korte **pauze** (**300** milliseconden). Met **F2** gelijk **RUN** via **Edit**.
De output wordt dan op laag gezet, gevolgd door nog een pauze. Het programma herhaalt vervolgens de lus.

De serie weerstand R1 is belangrijk omdat de stroom door een LED maximaal 20mA is en de uitgangstroom van de Raspberry Pi Pico GP21 is 4mA. Een LED werkt ook bij 4mA.
Als je het zo laat, zal de PicoMite daar voor altijd blijven zitten terwijl de LED knippert. Als je wilt veranderen iets (bijvoorbeeld de snelheid van knipperen) kunt u het programma onderbreken door **CTRL-C** op de console te typen en bewerk het vervolgens indien nodig. Dit is het grote voordeel van **MMBasic**, het is heel gemakkelijk om een programma te schrijven en te wijzigen.

Als u wilt dat dit programma automatisch start telkens wanneer de stroom wordt ingeschakeld, kunt u de opdracht gebruiken:

```
OPTION AUTORUN ON
```

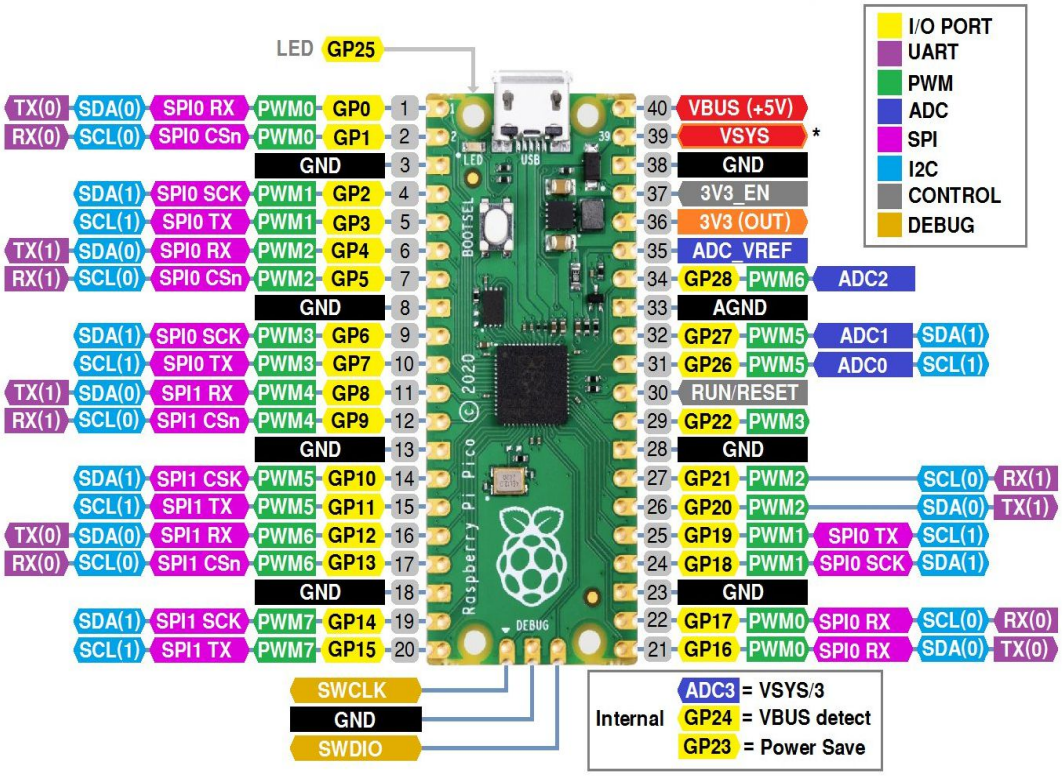
Om dit te testen, kunt u de voeding los koppelen en vervolgens opnieuw aansluiten. De PicoMite zou moeten opstarten door de LED te laten knipperen. Dan kan men ook een externe voeding VCC op VSYS gebruiken mits de PicoMite automatisch start. Deze losse voeding kan ook een batterij zijn maar moet lager zijn dan 5,5V anders gaat de Raspberry Pi Pico stuk.
Voorbeeld: 3 x 1,5V AA batterij = 4,5V. Met een of twee normale diodes in serie (0,6V of 1,2V) kan ook 4 x 1,5V = 6V - 0.6V of 1,2V = VSYS. Dit is niet van toepassing voor een RP2040-Zero print.
Heeft een lineaire voeding 5V → 3V3. PAS OP: Geen beveiligings diode D1 (pagina 21 diode D1).

Uitleg over programmeren in de BASIC taal

Als de programmeertaal **BASIC** nieuw voor u is, is het nu een goed moment om **Bijlage I** te raadplegen (**Programmeren in BASIC - A Tutorial**) achterin deze handleiding. Dit is een uitgebreide tutorial over de taal die u door de grondbeginselen leidt in een gemakkelijk te lezen formaat met veel voorbeelden. (pagina **338** - **382**)

PicoMite hardware

Dit diagram toont het mogelijke gebruik binnen MMBasic voor elke I/O pin op de Raspberry Pi Pico en de Raspberry Pi Pico 2:



Voor versies met **VGA-video**-uitgang zijn zes pinnen (**GP16** tot **GP21**) gereserveerd voor die functie. **HDMI-versies** hebben op vergelijkbare wijze acht pinnen (**GP12** tot **GP19**) die gereserveerd zijn voor die functie. Raadpleeg het hoofdstuk Video-uitgang voor meer informatie.

De versie van de firmware met **USB-toetsenbord/muisondersteuning** reserveert ook **pin 11** (**GP8**) voor de seriële console **Tx** en **pin 12** (**GP9**) voor **Rx**. Raadpleeg het hoofdstuk Toetsenbord/muis/ Gamepad voor meer informatie.

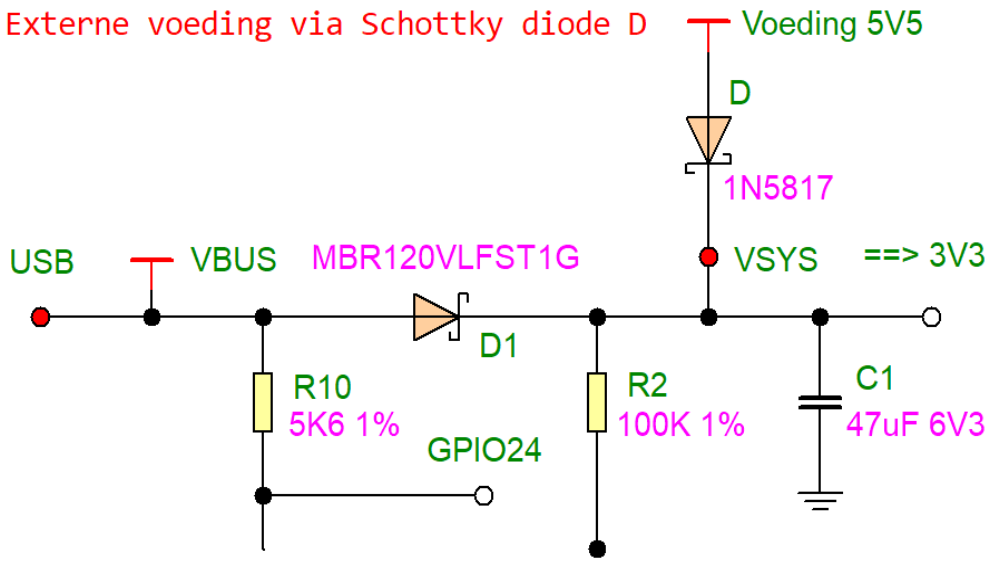
De notatie is als volgt:

GP0 tot **GP28**	Kan gebruikt worden voor digitale input of output.
COM1, **COM2**	Kan worden gebruikt voor asynchrone seriële I/O (UART0- en UART1-pinnen op de Pico data papier RX,TX).
I2C, **I2C2**	Kan worden gebruikt voor I2C-communicatie (I2C0- en I2C1-pinnen op het Pico-gegevensblad).
SPI, **SPI2**	Kan worden gebruikt voor SPI I/O (zie bijlage D). (SPI0- en SPI1-pinnen op het Pico-gegevensblad).
PWMnx	Kan worden gebruikt voor PWM-uitvoer (zie het PWM-commando).
GND	Gemeenschappelijke aarde.
VBUS	5V voeding rechtstreeks vanaf de USB-poort.
VSYS	5V-voeding gebruikt door de SMPS om 3,3V te leveren. Deze kan gebruikt worden als 5V uitgang of ingang.
3V3	EN Schakel 3.3V-regelaar in (laag = uit, hoog = ingeschakeld).
RUN	Reset pin, low houdt de PicoMite in reset.
ADCn	Deze pinnen kunnen worden gebruikt om spanning te meten analoge ingang.
ADC VREF	Referentiespanning voor spanningsmeting.
AGND	Analoge aarde.

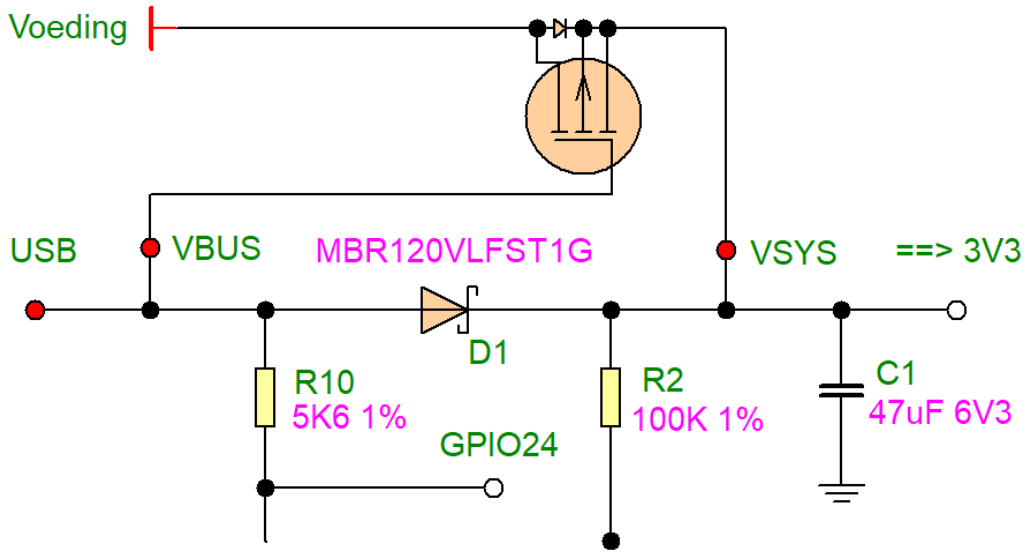
Onderstaande is alleen van toepassing voor een Raspberry Pi Pico of Raspberry Pi Pico 2.

1. USB-connector.
Geen tekening nodig. Sluit gewoon de micro-USB-kabel aan.

2. Batterij 1,8 V tot 5,5 V (Niet van toepassing bij een print met lineaire voeding en zonder **D1**)
Sluit de **+** van de batterij aan op de **Vsys-pin** (**pin 39**) en de **-** van de batterij op **GND** (**pin 38**).
Gebruik een diode om te voorkomen dat het verwisselen van pins schade veroorzaakt.

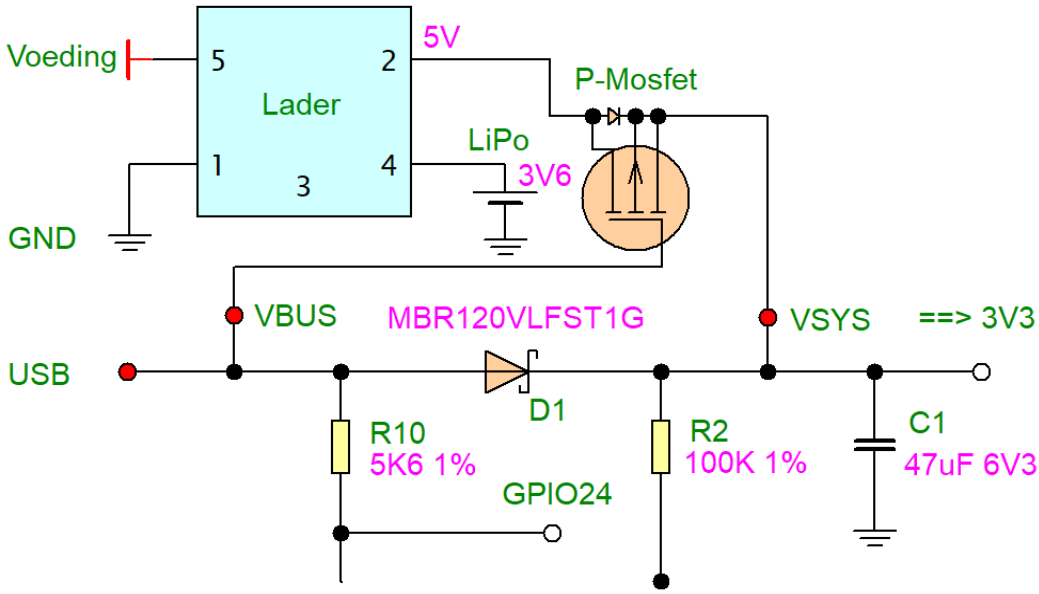


Voeding naar VSYS via externe Voeding met P-Channel Mosfet



of

Voeding met Lader en Batterij naar VSYS met P-Channel Mosfet



3. Accu 1,8V tot 5,5V en laadmodule
Er zijn veel verschillende laadmodules.

4. 5V voeding (lineaire regelaar)
Google gewoon **"LM7805 of LM117-5.0"** en sluit de **5V-uitgang** aan zoals in 2. boven.

5. Geschakelde 5V voeding
Daar kun je niets aan doen. De pico gebruikt **5V** niet rechtstreeks, maar maakt er alleen **3,3V** van

6. 3,3V voeding (lineaire regelaar)
Het beste is als u de ingebouwde **3,3V**-regelaar uitschakelt door **3,3V_EN** (**pin 37**) kort te sluiten met **GND** (**pin 38**)

7. Geschakelde 3V3 voeding
Gebruik een lineaire regelaar zoals in 6.

Normaal gesproken werken uw circuits betrouwbaarder als u een condensator (**10nF**) van de **RUN** (**pin 30**) aan aarde toevoegt. Dit voorkomt onbedoelde RESET's in omgevingen met elektrische ruis (industrieel/lasapparatuur/CF-lampen).

Alle pinnen kunnen worden gebruikt voor digitale invoer of uitvoer, maar ze zijn beperkt tot een maximale spanning van 3,6 V. Dit betekent dat een level shifter is vereist als ze worden gebruikt met apparaten die op 5V of hoger werken.

Binnen het MMBasic-programma kan naar I/O-pinnen verwezen worden met het fysieke pinnummer (d.w.z. 1 t/m 40) of de GP nummer (d.w.z. GP0 tot GP28).
Het volgende verwijst bijvoorbeeld naar dezelfde pin en werkt identiek:

```
  SETPIN 32, DOUT
```
en
```
  SETPIN GP27, DOUT
```

Op de PicoMite worden on-chip functies zoals de **SPI** en **I2C** interfaces niet toegewezen aan vaste pinnen, in tegenstelling tot (voor voorbeeld) de Micromite. De PicoMite maakt uitgebreid gebruik van het **SETPIN** commando, niet alleen om I/O pinnen, maar ook om de pinnen te configureren die worden gebruikt voor interfaces zoals serieel, **SPI**, **I2C**, enz.

De pinnen moeten volgens deze tekening worden toegewezen. De **SPI** **TX** kan bijvoorbeeld worden toegewezen aan pinnen **GP3**, **GP7** of **GP19**, maar het kan niet worden toegewezen aan pin **GP11** die alleen kan worden toegewezen aan het **SPI2** kanaal. Toewijzingen hoeven zich niet in hetzelfde "blok" te bevinden, dus u kunt bijvoorbeeld **SPI2** **TX** toewijzen aan pin **GP11** en **SPI2** **RX** aan pin **GP28**.

De PicoMiteVGA-versie van de firmware reserveert automatisch pinnen **21**, **22** en **26** (**GP16**, **GP17** en **GP20**) voor gebruik voor de VGA-uitgang. Bovendien worden bij het gebruik van de VGA-kleurmodus automatisch pinnen 24 gereserveerd, 25 en 27 (**GP18**, **GP19**, **GP21**). Bij het uitvoeren van de PicoMiteVGA-firmware zijn deze pinnen niet beschikbaar.

Pins die niet zichtbaar zijn op de Raspberry Pi Pico zijn nog steeds toegankelijk via MMBasic via een pseudo-pin nummer of hun GPn-nummer. Hierdoor kan MMBasic worden gebruikt op andere modules die de RP2040 microcontroller gebruiken. Deze verborgen pinnen zijn **Pin 41** of **GP23**, **Pin 42** of **GP24**, **Pin 43** of **GP25** en **Pin 44** of **GP29**.

Op de Raspberry Pi Pico worden deze pinnen als volgt gebruikt voor interne functies:

- **Pin 41** of **GP23** is een **digitale uitgang** die is ingesteld op de waarde **OPTION** **POWER**. (**ON=PWM**, **OFF=PFM**).
- **Pin 42** of **GP24** is een digitale ingang, die hoog is als **VBUS** aanwezig is.
- **Pin 43** of **GP25** is ook **PWM4B**. Het is een uitgang die is aangesloten op de on-board LED.
- **Pin 44** of **GP29** is ook **ADC3**, wat een analoge ingangswaarde ⅓ is van **VSYS**.

Modules van derden

Pins die niet op de Raspberry Pi Pico zichtbaar zijn, zijn nog steeds toegankelijk via MMBasic via een pseudo-pin nummer of hun GPn-nummer. Hierdoor kan MMBasic gebruikt worden op andere modules die gebruik maken van de **RP2040** of **RP2350** microcontrollers.

Op de Raspberry Pi Pico worden deze verborgen pinnen als volgt gebruikt voor interne functies:

- **Pin 41** of **GP23** is een digitale uitgang ingesteld op de waarde **OPTION POWER 'ON=PWM,OFF=PFM**
- **Pin 42** of **GP24** is een digitale ingang, die hoog is als **VBUS** aanwezig is.
- **Pin 43** of **GP25** is ook **PWM4B**. Het is een uitgang die is aangesloten op de ingebouwde **LED**.
- **Pin 44** of **GP29** is ook **ADC3**, wat een analoge ingang is die ⅓ van **VSYS** leest.

Op modules van derden die ze beschikbaar stellen, kunnen ze echter als volgt worden gebruikt:

- `Pin 41 of GP23: DIGITAL_IN:DIGITAL_OUT,SPI  TX,I2C2  SCL,         PWM3B`
- `Pin 42 of GP24: DIGITAL_IN:DIGITAL_OUT,SPI2 RX,COM2  TX, I2C SDA, PWM4A`
- `Pin 43 of GP25: DIGITAL_IN:DIGITAL_OUT,COM2 RX,I2C   SCL,         PWM4B`
- `Pin 44 of GP29: DIGITAL_IN:DIGITAL_OUT,COM1 RX,ANALOG_IN,I2C SCL, PWM6B`

WebMite-versie voor de Raspberry Pi Pico W of Pico 2 W

De WebMite-versie heeft ook enkele GPIO-pinnen die worden gebruikt voor interne bordfuncties:

- **GP29** (Input/Output) draadloze **SPI CLK/ADC** modus (**ADC3**) meet **VSYS/3**
- **GP25 SPI CS** (uitgang) zorgt er, wanneer hoog, ook voor dat de **GPIO29 ADC**-pin **VSYS** leest
- **GP24** (invoer/uitvoer) draadloze **SPI**-gegevens / **IRQ**
- **GP23** (uitgang) draadloos inschakelsignaal

De WebMite-firmware staat niet toe dat deze pinnen opnieuw worden toegewezen.
In tegenstelling tot de standaard Raspberry Pi Pico is de ingebouwde **LED** op de **Pico W** niet verbonden met een pin op de Pico **RP2040** of **RP2350**, maar in plaats daarvan naar een GPIO-pin op de draadloze chip en is niet toegankelijk vanuit een BASIC-programma.
De antenne bevindt zich op de printplaat, aan de andere kant van de USB-connector, en kan het beste in de vrije ruimte worden bewaard prestatie – plaats geen metaal onder of dichtbij de antenne.

Nieuwe CPU-variaties

De nieuwere chips uitgebracht door de Raspberry Pi Foundation en ondersteund door de PicoMite zijn:

RP2350A

Dit is verpakt in een **60 pins** pakket en wordt gebruikt in de **Raspberry Pi Pico 2** en vele andere derde partijen modules. De pinouts en functies van de I/O-pinnen zijn hetzelfde als die van de **RP2040** en zijn hierboven gedocumenteerd.

RP2350B

De **RP2350B** zit in een **80 pins** pakket met nog eens **18 GPIO pinnen**. De PicoMite firmware ondersteunt deze extra pinnen inclusief **PWM kanalen 8-11** (maakt maximaal 24 gelijktijdige PWM-uitgangen mogelijk).
De configuratie voor de **RP2350B** gebeurt automatisch door alle extra pinnen en drie PIO-kanalen te maken beschikbaar (de **VGA versie** heeft twee vrije PIO-kanalen).

Pindefinities voor de **RP2350B** moeten de GP gebruiken nomenclatuur (dwz **GP0** tot **GP47**).
De pinnen **GP40** t/m **GP47** kunnen worden gebruikt voor **analoge invoer**, de pinnen **GP26** tot **GP29** niet ondersteuning voor analoge ingang.
Het pingebruik voor de pinnen **GP0** tot **GP29** op de **RP2350B** is hetzelfde als dat voor de **RP2040** en **RP2350A**.

Pinnen GP30 t/m GP47 voor de RP2350B kunnen als volgt worden gebruikt:

```
GP30: DIGITAL_IN: DIGITAL_OUT, SPI2 SCK,   I2C2 SDA,           PWM7A
GP31: DIGITAL_IN: DIGITAL_OUT, SPI2 TX,    I2C2 SCL,           PWM7B
GP32: DIGITAL_IN: DIGITAL_OUT, COM1 TX,    SPI  RX,  I2C SDA,  EXT_PWM8A
GP33: DIGITAL_IN: DIGITAL_OUT, COM1 RX,    I2C  SCL,           PWM8B
GP34: DIGITAL_IN: DIGITAL_OUT, SPI  SCK,   I2C2 SDA,           PWM9A
GP35: DIGITAL_IN: DIGITAL_OUT, SPI  TX,    I2C2 SCL, PWM9B
GP36: DIGITAL_IN: DIGITAL_OUT, COM2 TX,    SPI  RX,  I2C SDA,  PWM10A
GP37: DIGITAL_IN: DIGITAL_OUT, COM2 RX,    I2C  SCL, PWM10B
GP38: DIGITAL_IN: DIGITAL_OUT, SPI  SCK,   I2C2 SDA, PWM11A
GP39: DIGITAL_IN: DIGITAL_OUT, SPI  TX,    I2C2 SCL, PWM11B
GP40: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN0, COM2 TX,  SPI2 RX,  I2C SDA, PWM8A
GP41: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN1, COM2 RX,  I2C  SCL, PWM8B
GP42: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN2, SPI2 SCK, I2C2 SDA, PWM9A
GP43: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN3, SPI2 TX,  I2C2 SCL, PWM9B
GP44: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN4, COM1 TX,  SPI2 RX,  I2C SDA, PWM10A
GP45: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN5, COM1 RX,  I2C  SCL, PWM10B
GP46: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN6, SPI2 SCK, I2C2 SDA, PWM11A
GP47: DIGITAL_IN: DIGITAL_OUT, ANALOG_IN7, SPI2 TX,  I2C2 SCL, PWM11B
```

RP2354A en RP2354B

Deze worden momenteel niet ondersteund door de PicoMite firmware.
(Deze krijgen **2Mb flash** geheugen in de microcontroller.)

PSRAM

De **RP2350B** ondersteunt **PSRAM** en sommige commerciële aanbiedingen hebben **8MB PSRAM** toegevoegd aan kaarten met de **80-pins RP2350B** (bijv. Pimoroni **PGA2350**).
Het **PSRAM** is toegankelijk via een Quad SPI-bus en is dus relatief traag, hoewel het wordt gebufferd via een cache wat dat probleem verzacht. Als **PSRAM** aanwezig en geconfigureerd is, zal MMBasic het toevoegen aan het algemene doel RAM-pool, zodat programma's een enorme hoeveelheid algemeen RAM-geheugen hebben om mee te werken, ook al kan dit een iets langzamer.
Om toegang te krijgen tot een **PSRAM** is een extra pin nodig voor de chipselectiefunctie en deze wordt geselecteerd met behulp van de **OPTION PSRAM PIN** opdracht. Geldige pinnen voor de **PSRAM** chip select zijn **GP0, GP8, GP19** en **GP47**.

I/O pin limieten

De maximale spanning die op een **I/O-pin** kan worden toegepast voor de **RP2040**, is **3,6V**.
De **Raspberry Pi Pico 2** die de **RP2350 microcontroller** gebruikt, kan **5V** accepteren terwijl de chip is ingeschakeld.
Als uitgangen kunnen alle **I/O**-pinnen individueel tot **8mA** aan- of afvoeren. Bij deze belasting zal de uitgangsspanning doorzakken ongeveer **2,3V**. Een meer praktische belasting is **5mA**, waarbij de uitgangsspanning doorgaans **3V** zou zijn.
Als uitgangen kunnen alle I/O-pinnen individueel maximaal **12mA** aan- of afvoeren. Bij deze belasting is de uitgangsspanning zal zakken tot ongeveer **2,3V**. Een meer praktische belasting is **5mA**, waar de uitgangsspanning normaal gesproken **3V** zou zijn. Naar stuur een rode **LED** aan op **5mA**, de aanbevolen weerstand is **220Ω**. Voor andere kleuren kan een andere waarde nodig zijn.
De maximale totale I/O-stroombelasting voor de gehele chip is **50mA**.

Voedingssysteem

De Raspberry Pi Pico heeft een flexibel voedingssysteem. De ingangsspanning van de USB- of VBUS-ingangen is via een Schottky-diode verbonden met de buck-boost SMPS (Switch Mode Power Supply) met een uitgang van **3.3V**. De SMPS zal input accommoderen spanningen van **1,8V** tot **5,5V** waardoor de PicoMite werken op een breed scala aan stroombronnen, waar-onder: batterijen.

Externe circuits kunnen worden gevoed door **VBUS** (normaal gesproken) **5V**) of door de **3V3** (**3,3V**) uitgang die tot **300mA**.

Voor embedded controller-toepassingen is over het algemeen een externe voedingsbron (**anders dan USB**) vereist en dit: kan via een **Schottky-diode op VSYS** worden aangesloten. Hierdoor kan de PicoMite worden aangedreven door wat dan ook voeding de hoogste spanning produceert (**USB** of **VSYS**). De diodes voorkomen terugkoppeling naar de lagere spanningsvoorziening.

Om de ruis van de voeding tot een minimum te beperken, is het mogelijk om de **3V3EN** te aarden om de SMPS uit te schakelen. Bij het afsluiten van de converter stopt met schakelen, interne regelcircuits worden uitgeschakeld en de belasting wordt losgekoppeld van de invoer. U kunt het bord dan van stroom voorzien via een lineaire regelaar van **3,3 V** die in de **3V3**-pin wordt gevoerd.

Energieverbruik en kloksnelheid

Standaard is de kloksnelheid voor de **PicoMite RP2040** is **133 MHz** en **420 MHz** maximaal.
126 MHz voor de **PicoMiteVGA** firmware, wat de aanbevolen maximum voor de **Raspberry Pi Pico**.
Standaard is de kloksnelheid voor de **PicoMite 2 RP2350A** is **150 MHz** en **378 MHz** maximaal.
Het stroomverbruik is afhankelijk van de kloksnelheid. Dit zijn typische metingen vanaf de **USB** en bevatten geen stroom afkomstig of aangesloten door de **I/O pinnen** of de **3V3 pin**:

```
RP2040: 420 MHz 82mA, 378 MHz 76mA, 252 MHz 50mA, 133 MHz 28mA, 48 MHz 13mA
RP2350: 378 MHz 84mA, 304 MHz 40mA, 252 MHz 37mA, 150 MHz 20mA, 48 MHz  9mA
```

Dit is voor een **Raspberry Pi Pico 2** en kan mogelijk afwijken bij andere borden met betrekking tot de frequentie kan vast lopen ivm een ander **FLASH.** Met **PSRAM** bij een **RP2350B** is het <= **372MHz**

`> OPTION CPUSPEED speed` Met dit commando kan men de **133 MHz** aanpassen.
`> OPTION CPUSPEED 133000 (KHz)`

`> ? MM.INFO$(CPUSPEED)` Met deze opdracht kan men de actuele snelheid opvragen.
`> ? MM.INFO$(CPUSPEED) 133000000 (133e6)`
Met `OPTION LIST` indien cpu snelheid geen **133 MHz** is.

MMBasic gebruiken

Opdrachten en programma invoer

Bij de opdrachtprompt kunt u een opdracht invoeren en deze wordt onmiddellijk uitgevoerd. Meestal zul je doe dit om de PicoMite te vertellen iets te doen zoals een programma uitvoeren of een optie instellen. Maar met deze functie kun je ook om opdrachten uit te testen bij de opdracht-prompt.
Om een programma in te voeren is de eenvoudigste methode het **EDIT** commando te gebruiken. Dit roept het programma op volledig scherm op editor die in de PicoMite is ingebouwd en verder-op in deze handleiding wordt beschreven. Het bevat geavanceerde functies zoals: zoals zoeken en kopiëren, knippen en plakken van en naar een klembord.
Je zou het programma ook op je desktopcomputer kunnen samenstellen met zoiets als **Kladblok** of **Notepad++** en dan breng het over naar de PicoMite via het **XModem**-protocol (zie het **XMODEM commando** en voorbeeld bij **commando's uitleg**) of door het naar de seriële verbinding van de console (zie de opdracht **AUTOSAVE**).
Een derde en handige methode voor het schrijven en debuggen van een programma is het gebruik van **MMEdit.** Dit is een programma draait op uw Windows-computer waarmee u uw programma op uw computer kunt bewerken en vervolgens kunt overbrengen naar de PicoMite met een enkele muisklik. **MMEdit** is geschreven door Jim Hiley en kan worden gedownload voor gratis van https://www.c-com.com.au/MMedit.htm.
Een ding dat je niet kunt doen, is de oude BASIC manier gebruiken om een programma in te voeren dat voor elke regel een prefix was met een regelnummer. Regelnummers zijn optioneel in MMBasic dus je kunt ze nog steeds gebruiken als je wilt, maar als je voer een regel in met een regelnummer bij de prompt MMBasic zal het gewoon onmiddellijk uitvoeren.

Programmastructuur

Een BASIC-programma begint bij de eerste regel en gaat door totdat het aan het einde van het programma afloopt of op **END command** komt, op dat moment zal MMBasic de opdrachtprompt **(>)** op de console weergeven en wachten op iets in te voeren.
Een programma bestaat uit een aantal instructies of opdrachten, die er elk voor zorgen dat de BASIC interpreter iets doen (de woorden statement en command betekenen over het algemeen hetzelfde en worden door elkaar gebruikt). Normaal gesproken staat elke instructie op zijn eigen regel, maar u kunt meerdere instructies op één regel hebben, gescheiden door de dubbele punt **(:)** Bijvoorbeeld.

```
A = 24.6 : PRINT A
```

Elke regel kan beginnen met een regelnummer. Regelnummers waren echter verplicht in de vroege BASIC-tolken moderne implementaties (zoals MMBasic) hebben ze niet nodig. Je kunt ze nog steeds gebruiken als je wilt, maar ze hebben geen voordeel en maken uw programma's over het algemeen rommelig.
Dit is een voorbeeld van een programma dat line nummers gebruikt.

```
50 A = 24.6
60 PRINT A
```

Een regel kan ook beginnen met een label dat kan worden gebruikt als doel voor een programma-sprong met behulp van de **GOTO** en **GOSUB** opdracht. Bijvoorbeeld (de labelnaam is **JmpBack**):

```
JmpBack:
  A = A + 1
  Print A
Goto JmpBack
```

Met de kleuren van de ingebouwde editor:

```
a = 20 : b = 30 : c = 40
Gosub LABEL1
End

LABEL1:
  Print "Label1"
Return
```

Een label heeft dezelfde specificaties (lengte, tekenset, enz.) als een variabelenaam, maar het kan niet de hetzelfde als een opdrachtnaam. Bij gebruik om een regel te labelen, moet het label aan het begin van een regel verschijnen maar na een regelnummer (indien gebruikt) en worden afgesloten met een dubbele punt **(:)**.

De opdrachtregel bewerken

Wanneer u een regel invoert bij de opdrachtprompt, kan de regel worden bewerkt met de linker- en rechterpijltoetsen om te verplaatsen langs de lijn, de Delete-toets om een teken te verwijderen en de Insert-toets om te wisselen tussen invoegen en overschrijven.
Op elk moment zal de Enter-toets de regel naar MMBasic sturen, die deze zal uitvoeren.
De pijltoetsen omhoog en omlaag bewegen door een geschiedenis van eerder ingevoerde opdracht-regels die kunnen worden bewerkt en hergebruikt.

Sneltoetsen

De functietoetsen op het toetsenbord of de seriële console kunnen worden gebruikt bij de opdrachtprompt om automatisch: voer algemene commando's in. Deze functietoetsen zullen de tekst invoegen gevolgd door de Enter-toets, zodat de commando wordt onmiddellijk uitgevoerd:

```
F2  RUN   ' Start het bestaande programma.    RUN  [bestandsnaam$][, cmdline$]
F3  LIST  ' List van het bestaande programma. LIST [bestandsnaam$][, .....
F4  EDIT  ' Edit het bestaande programma.
F10 AUTOSAVE
F11 XMODEM ONTVANGEN
F12 XMODEM VERZENDEN
```

Functietoetsen **F1** en **F5** tot **F9** kunnen worden geprogrammeerd met aangepaste tekst.
Zie de **OPTION FNKey** opdracht.

Programma's opslaan

Wanneer een programma wordt bewerkt via **EDIT** of geladen, wordt het daar opgeslagen.
Flash geheugen is niet vluchtig, dus het programma zal niet verloren gaan als de stroom uitvalt of de processor wordt gereset. `> OPTION LIST` geeft het nog beschiklbare geheugen.

Vrij Programma geheugen V6.00.02	RP2040	RP2350
PicoMite en PicoMite USB	128Kb	288Kb
PicoMite VGA en PicoMite VGA USB	100Kb	180Kb
WebMite	88Kb	208Kb

Programma's kunnen ook worden opgeslagen op een SD-kaart (indien geconfigureerd) of ze kunnen worden opgeslagen op een van de 3 genummerde geheugenlocaties (of 'slots') in het flash-geheugen. Deze kunnen worden gebruikt om eerdere versies van het programma op te slaan (voor het geval je terug moet naar een eerdere versie) of ze kunnen worden gebruikt om totaal verschillende programma's op te slaa die snel in het programmageheugen kan worden geladen en kan worden uitgevoerd.

Bovendien zal MMBasic een BASIC-programma toestaan om een ander programma te laden en uit te voeren dat is opgeslagen op een genummerde flash locatie met behoud van alle variabelen en instellingen van het originele programma – dit wordt ketenen genoemd en zorgt ervoor dat een veel groter programma kan worden uitgevoerd dan de hoeveelheid programmageheugen normaal zou toestaan.

Om deze genummerde locaties in Flash te beheren, kunt u de volgende opdrachten gebruiken (merk op dat in de volgende **n** is een getal van **1** tot **3** voor de **PicoMite** en **1** tot **3** voor de **PicoMiteVGA**):

FLASH SAVE n	Sla het programma op in het RAM op de flashlocatie n.
FLASH LOAD n	Laad een programma van flash-locatie n in het RAM.
FLASH RUN n	Voer een programma uit vanaf de flash-locatie n, wist alle variabelen maar wist niet of verander het programma in het hoofdprogrammageheugen.
FLASH LIST	Toon een lijst met de inhoud van alle flitslocaties.
FLASH LIST n [,all]	Geeft een overzicht van het programma op locatie n. Gebruik **FLASH LIST n, ALL** to list zonder pagina einden
FLASH ERASE n	Wis flitslocatie n.
FLASH ERASE ALL	Wis alle flitslocaties.
FLASH CHAIN n	Laad een programma en voer het uit vanaf flashlocatie n waarbij alle variabelen intact blijven. Net als bij **FLASH RUN**, wist of wijzigt deze opdracht niet de programma dat in het hoofdprogrammageheugen is opgeslagen.
FLASH OVERWRITE n	Wis de flashlocatie n en sla het programma vervolgens in het RAM op die locatie op.

Daarnaast kan het commando **OPTION AUTORUN** worden gebruikt om een in te stellen flash-programmalocatie te specificeren actief wanneer de stroom wordt ingeschakeld of de CPU opnieuw wordt opgestart. Deze optie kan ook worden gebruikt zonder een flitser op te geven locatie en in dat geval zal MMBasic automatisch het programma in het programmageheugen laden en uitvoeren.

Het wordt ten zeerste aanbevolen om een opmerking op te nemen die het programma beschrijft als de eerste regel van het programma.
Dit wordt dan weergegeven door de opdracht **FLASH LIST** en helpt bij het identificeren van het programma in de lijst.

Alle BASIC-programma's die in flash zijn opgeslagen, worden gewist als u de PicoMite-firmware upgradet (of downgradet). Dus zorg ervoor dat u er eerst een backup van maakt.

DRIVE drive$

drive$ kan "a:" of "b:" zijn, waarbij **a:** de **Flash drive** is en **b:** beschikbaar als er een **SD-kaart** is ingeschakeld. Dit stelt het standaardstation in waar commando's zoals **LOAD**, **SAVE**, **FILE**, **XMODEM** etc. worden uitgevoerd. NB vergelijkbaar met originele Maximite. Dus **> a:** of **> b:** indien aanwezig
Bij het opstarten is de standaardschijf de SD-kaart, indien ingeschakeld, anders de flashdrive.
Gebruik `> DRIVE "A:/FORMAT"` om de flashdrive opnieuw te initialiseren. (Niet voor de B: drive)

`> copy "a:/*" to "b:/test1"`	Kopieer alles van drive a: naar drive b:/test
`> copy "*" to "b:/test"`	Kopieer alles van bestaande drive naar drive b:/test
`> kill "b:/test/file?.txt"`	Wis in drive b: de map b:/test/file.
`> kill "b:test/*",all`	Wis de map in drive b:/test in zijn geheel.
`> copy "a:/*" to "a:/atest"`	Kopieer van drive a: hoofd map naar a:/atest map.
`> files "a:/atest"`	Vraag de map van drive a:/atest op, dus niet de inhoud
`> files "a:/atest/*"`	Vraag de inhoud van drive a:/atest/ op.
`> drive "A:/format"`	Wis de volledige inhoud A:/. Maak eerst een backup B:

`MM.INFO$(OPTION FLASH SIZE)` zou de grootte van het flash-geheugen moeten geven.
Controleer of dit werkt als u versies hebt met niet-standaard flash.

`MM.INFO$(DISK SIZE)` en `MM.INFO$(FREE SPACE)` werken op de standaard drive.

`LOAD`, `SAVE`, `XMODEM`, `LOAD IMAGE`, `LOAD JPG`, `OPEN`, `CLOSE` etc. zou allemaal moeten werken op de flashdrive, inclusief ondersteuning voor meerdere bestanden, open.

Nu geïmplementeerd `MKDIR`, `CHDIR` en `RMDIR` voor `FLASH`.
De sleutelcode is geïmporteerd uit de **CMM2** die de directorytracking in mijn firmware moet doen vanwege beperkingen in **FATFS** bij gebruik van **exFAT**. **LittleFS** heeft nog meer beperkingen dan **FatFS-exFAT** omdat het niet eens een **CHDIR** primitief heeft, dus alles is op het hoogste niveau en de firmware moet de **CHDIR's** van gebruikers volgen en dat pad toepassen op elke bestandsnaam die de gebruiker aanvraagt.

b.v.

```
> CHDIR "/peter/test"
> LOAD "../fred/myapp"
```

Wordt geconverteerd om `"/peter/fred/myapp.bas"` te laden.

Merk nogmaals op dat bestandsnamen in LittleFS hoofdlettergevoelig zijn. De firmware neemt kleine letters "myapp.bas" aan. Als u een bestand myapp.BAS maakt, zal LOAD "myapp" het niet vinden. Houd er ook rekening mee dat LittleFS geen tijd- en datumstempel van bestanden ondersteunt.

Op dit moment kunt u een bestandsnaam specificeren met een schijf, bijv. `LOAD "A:/mijnapp"`.
De schijfspecificatie wordt echter genegeerd en het bestand wordt geladen vanaf het apparaat dat is opgegeven door de laatste **DRIVE** opdracht.

U zou nu in bijna alle situaties volledig gekwalificeerde bestandsnamen met stationsletters moeten kunnen gebruiken, ongeacht de huidige werkende schijf.
Uitzonderingen zijn:
`MKDIR`, `CHDIR`, `RMDIR`, `NAME` die alleen werken op de huidige schijf,

b.v.

```
DRIVE "A:"
MKDIR "flashdir"
PLAY WAV "B:/wavfiles/wavsample"
COPY "B:/mydir/myprog.bas" To "a:/flashdir/myprog.bas"
DRIVE "B:"
FILES "A:/flashdir/"
OPEN "A:/testfile.dat" For Input As #1
OPEN "B:/anotherfile.dat" For Output As #2
a$ = Input$(100, 1)
Print #2, a$
Close #1
Close #2
OPEN "B:/myfile.dat" For Append As #1
' enzovoort.....
```

Hulp voor de nieuwe file en drive commando's als men het met files opvraagd.

Als men de onderstaande regels opslaat met **> SAVE** zorg er dan eerst voor dat het geheugen gewist is met **> NEW**. Dan is de geheugen omvang na save 0 byte.

```
0_SD-FLASH:_Load_Run_Save_Files_Kill_Mkdir.bas
1_SD-FLASH:_Chdir_Rmdir_Seek_Rename_Input$.bas
2_SD-FLASH:_Dir$_Cwd$_Eof_Loc_Lof_Copy_Name.bas
3__SERIEEL:_XMODEM_Send____'prbas'.bas
4__SERIEEL:_XMODEM_Receive_'prbas'.bas
5_FLASH1-4:_SAVE_LOAD_RUN_LIST_ERASE_CHAIN_OVERWRITE.bas
6_DRIVE_AB:_DRIVE_'A:'=FLASH-INTERN.bas
7_DRIVE_AB:_DRIVE_'B:'=SD-EXTERN.bas
8__SD_Copy:_Copy_'A:xxx'_to_'B:xxx'.bas
9__SD_Kill:_Kill_'A:*'_Kill_'A:xxx'.bas
```

U kunt nog steeds de andere vorm van kopiëren **A2B**, **B2A** gebruiken, in welk geval u relatieve padnamen kunt gebruiken voor de standaardmap die op elke schijf is ingesteld, b.v.

```
> A:
> files
> B:
> files
Drive "a:"
Chdir "peter"
Drive "b:"
Chdir "alles"
Copy a2b "file.dat2 To "new"
```

Dit kopieert :

```
a:/peter/file.dat2 To b:/anything/newfile.dat
```

En het zelfde resultaat:

```
Copy "A:/test.bas" To "B:/test/test.bas"
Copy "A:/*" To "B:/picomite"
Kill "A:/*" of Kill "A:/test1.bas"
```

Geïmplementeerde **DIR$** functie voor Flash drive en verbeterde functionaliteit voor SD
(Kan nu DIR$ een andere map dan de huidige).

Gebruik **DRIVE "A:/FORMAT"** om de **KILL** op enige versie van **MMBasic**, inclusief **CMM2**.
Kan geen nauwkeurig commentaar geven op de overhead van een groot aantal bestanden op gebruikte ruimte maar ik weet dat elk bestand **C: 1K** gebruikt als koptekst, dus elke 4 bestanden of zo zullen nog een blok van 4K opeten.

Kopieeren kan ook met **Tera Term** door na **> FILES** met de muis de file naam te selecteren en op de rechter muis knop te klikken, waardoor de file naam achter **>** komt te staan en dan met de pijltjes toetsen bewerken of na **> LIST** de text te selecteren en met **Edit** van **Tera Term Copy** te kiezen en in een ander editor programma bv. **Notepad++** of **MM Edit4** weer te plakken en daarna te bewerken of naar een andere com poort met **RP-Pico** contacten en plakken met rechter muis knop.

Een lopend programma onderbreken

Een programma wordt uitgevoerd door het **RUN** commando. U kunt MMBasic en het lopende programma op elk moment onderbreken tijd door "**CTRL**" + "**C**" in te drukken en MMBasic keert terug naar de opdrachtprompt.

Opties instellen

Veel opties kunnen worden ingesteld met behulp van opdrachten die beginnen met het trefwoord **OPTION.** Ze staan in hun eigen lijst gedeelte van deze handleiding. U kunt bijvoorbeeld de CPU-kloksnelheid wijzigen met het commando:

```
OPTION CPUSPEED snelheid  (Invoeren in KHz)
```

Opgeslagen variabelen

Omdat de PicoMite niet per se een normaal opslagsysteem heeft, moet hij gegevens opslaan die hersteld wanneer de stroom wordt hersteld. Dit kan worden gedaan met het **VAR SAVE** commando waarmee de variabelen vermeld op de opdrachtregel in niet-vluchtig flash-geheugen. De ruimte die is gereserveerd voor opgeslagen variabelen is **16 KB**. Bij andere systemen kan dit **EPROM** zijn

Deze variabelen kunnen worden hersteld met de opdracht **VAR RESTORE**, waarmee alle opgeslagen variabelen worden toegevoegd aan de variabelentabel van het lopende programma. Normaal gesproken wordt dit commando bij de start van een programma geplaatst, zodat: de variabelen zijn klaar voor gebruik door het programma.

Deze faciliteit is bedoeld voor het opslaan van kalibratiegegevens, of instellingen door de gebruiker geselecteerde opties en andere items die veranderen zelden. Het mag **niet** worden gebruikt voor snelle opslag, omdat u het **flashgeheugen kunt verslijten**. De gebruikte flits voor de Raspberry Pi Pico heeft een hoog uithoudingsvermogen, maar dit kan worden overschreden door een programma dat herhaaldelijk opslaat variabelen. Als u toch vaak gegevens wilt opslaan, moet u een real-time klokchip toevoegen. De **RTC** commando's kunnen:
vervolgens worden gebruikt om gegevens op te slaan en op te halen uit het door batterijen ondersteunde geheugen van de **RTC**. Zie het **RTC** commando voor: meer details.

Waakhond Timer

Het belangrijkste gebruik van de PicoMite is als een embedded controller. Het kan worden geprogrammeerd in MMBasic en wanneer de programma is debugged en klaar voor "prime time" de **OPTION AUTORUN** configuratie-instelling kan worden gedraaid aan. De module voert dan automatisch zijn programma uit wanneer de stroom wordt ingeschakeld en fungeert als een aan-gepast circuit het uitvoeren van een speciale taak. De gebruiker hoeft niets te weten over wat erin draait.
Het is echter mogelijk dat een fout in het programma ervoor kan zorgen dat MMBasic een fout genereert en terug naar de opdrachtprompt. Dit zou weinig zin hebben in een embedded situatie, aangezien de PicoMite dat niet zou doen iets op de console hebben aangesloten. Een andere mogelijkheid is dat het BASIC-programma vast kan komen te zitten om de een of andere reden een eindeloze lus. In beide gevallen zou het zichtbare effect hetzelfde zijn... het programma zou stoppen
draaien totdat de stroom werd gecycleerd.

Om dit te voorkomen kan de watchdog timer worden gebruikt. Dit is een timer die aftelt tot nul en wanneer het nul bereikt, wordt de processor automatisch opnieuw gestart (hetzelfde als toen de stroom voor het eerst werd ingeschakeld), dit zal treden zelfs op als MMBasic achter de opdrachtprompt zat. Na de herstart de automatische variabele **MM.WATCHDOG** wordt ingesteld op true om aan te geven dat de herstart werd veroorzaakt door een time-out van de watchdog.

Het **WATCHDOG** commando moet op strategische locaties in het programma worden geplaatst om de timer te blijven resetten en daardoor voorkomen dat het aftelt tot nul. Als er dan een fout optreedt, wordt de timer niet gereset, maar zal aftellen tot nul en het programma zal opnieuw worden gestart (ervan uitgaande dat de **AUTORUN** optie is ingesteld).

PIN beveiliging

Soms is het belangrijk om de data en het programma in een embedded controller vertrouwelijk te houden. In de PicoMite kan dit doen door het **OPTION PIN** commando te ge-bruiken. Deze opdracht stelt een pincode in (die wordt opgeslagen in flash) en telkens wanneer de PicoMite terugkeert naar de opdrachtprompt (om welke reden dan ook) de gebruiker op de con-sole wordt gevraagd om de pincode in te voeren. Zonder de juiste pincode kan de gebruiker niet bij de opdrachtprompt en hun enige optie is om de juiste pincode in te voeren of de PicoMite opnieuw op te starten. Wanneer het opnieuw wordt opgestart de gebruiker heeft nog steeds de juiste pincode nodig om toegang te krijgen tot de opdrachtprompt.

Omdat een indringer de opdrachtprompt niet kan bereiken, kunnen ze geen programma weergeven of kopiëren, ze kunnen niet wijzigen het programma of enig aspect van MMBasic of de PicoMite wijzigen. Eenmaal ingesteld kan de pincode alleen worden verwijderd door het verstrekken van de juiste pincode zoals ingesteld in de eerste plaats. Als het nummer verloren is gegaan, is de enige manier om het te herstellen, opnieuw te laden de PicoMite-firmware (die het programma en alle opties zal wissen).

Er zijn andere tijdrovende manieren om toegang te krijgen tot de gegevens (zoals het gebruik van een programmeur om de flitser te onderzoeken) geheugen), dus dit moet niet worden beschouwd als de ultieme beveiliging, maar het werkt wel als een belangrijk afschrikmiddel.

De bibliotheek

Met behulp van de **LIBRARY** functie is het mogelijk om **BASIC** functies, subroutines eninge-sloten lettertypen te maken en toe te voegen naar MMBasic om ze permanent en onderdeel van de taal te maken. U hebt bijvoorbeeld een geschreven reeks subroutines en functies die geavanceerde bitmanipulatie uitvoeren, deze kunnen worden opgeslagen als een bibliotheek en word onderdeel van MMBasic en voer hetzelfde uit als andere ingebouwde functies die al deel uitmaken de taal.
Een ingesloten lettertype kan ook op dezelfde manier worden toegevoegd en net als een normaal lettertype worden gebruikt. Om componenten in de bibliotheek te installeren, moet u de routines schrijven en testen zoals u met elke normale zou doen **BASIC** routines. Als ze correct werken, kunt u de opdracht **LIBRARY SAVE** gebruiken.
Dit zal draag de routines (zoveel als u wilt) over naar een niet-zichtbaar deel van het flashgeheugen waar ze beschikbaar zullen zijn elk **BASIC** programma, maar wordt niet weergegeven wanneer het **LIST** commando wordt gebruikt en wordt niet verwijderd wanneer een nieuw programma wordt geladen of **NEW** wordt gebruikt.
De opgeslagen subroutines en functies kunnen echter van binnenuit worden aangeroepen het hoofd-programma en kan zelfs worden uitgevoerd vanaf de opdrachtprompt (net als een inge-bouwde opdracht of functie).
Enkele aandachtspunten:

- **Bibliotheekroutines werken precies zoals normale BASIC code** en kunnen uit een wille keurig aantal **subroutines** bestaan, **functies**, ingebedde C-routines en **lettertypen**.
  Het enige verschil is dat ze niet laten zien wanneer een programma wordt weergegeven en wordt niet verwijderd wanneer een nieuw programma wordt geladen.
- **Bibliotheekroutines kunnen globale variabelen creëren** en benaderen en zijn onder-worpen aan dezelfde regels als de main programma - bijvoorbeeld **OPTION EXPLICIT** respecteren als het is ingesteld.
- **Wanneer de routines naar de bibliotheek worden overgebracht, zal MMBasic ze compri-meren door opmerkingen te verwijderen, extra spaties, lege regels en de hexadecimale codes in ingesloten C-routines en lettertypen**. Dit maakt de bibliotheek ruimtebesparend vooral bij het laden van grote lettertypen. Na het opslaan wordt het programmagebied gewist.
- **U kunt de opdracht LIBRARY SAVE meerdere keren gebruiken**. Sla bij elk de nieuwe inhoud van de programmaruimte kan worden toegevoegd aan de reeds bestaande code in de bibliotheek.
- **U kunt regelnummers gebruiken in de bibliotheek**, maar u kunt geen regelnummer ge-bruiken op een verder lege regel als het doel voor een **GOTO**, enz. Dit komt omdat het **LIBRARY SAVE** commando alle lege regels zal verwijderen.
- **U kunt READ opdrachten in de bibliotheek gebruiken**, maar deze lezen standaard **DATA** instructies in de geheugen van het hoofdprogramma. Als u wilt lezen uit **DATA** statements in de bibliotheek, moet u de **RESTORE** opdracht vóór de eerste **READ** opdracht. Hierdoor wordt de aanwijzer teruggezet naar de bibliotheekruimte.
- **De bibliotheek wordt opgeslagen in programmaflashgeheugen Slot 3** en dit zal niet be-schikbaar zijn voor het opslaan van een programma als **LIBRARY SAVE** wordt gebruikt.
- **U kunt zien wat er in de bibliotheek staat door de opdracht LIBRARY LIST te gebruiken die de inhoud van de bibliotheekruimte**.

Om de routines in de bibliotheekruimte te verwijderen, gebruikt u het **LIBRARY DELETE** commando

Dit maakt de ruimte vrij en breng de **Flash** **Slot 3** die door de bibliotheek wordt gebruikt weer beschikbaar voor opslag voor normale programma's. De enige andere manier om een biblio-theek te verwijderen is door **OPTION RESET** te gebruiken.

Programma initialisatie

De bibliotheek kan ook code bevatten die niet in een subroutine of functie zit. Deze code (indien aanwezig) wordt automatisch uitgevoerd voordat een programma begint te lopen (dwz via het RUN-commando). Deze functie kan worden gebruikt om constanten te initialiseren of MMBasic op de een of andere manier in te stellen. Als u bijvoorbeeld een aantal constanten wilt instellen kan de volgende regels in de bibliotheekcode opnemen:

```
CONST TRUE  = 1
CONST FALSE = 0
```

In alle opzichten zijn de identifiers **TRUE** en **FALSE** aan de taal toegevoegd en zullen dat ook zijn beschikbaar voor elk programma dat wordt uitgevoerd op de Micromite.

MM.STARTUP

Het kan nodig zijn om wat code uit te voeren bij de eerste keer opstarten, misschien om wat hardware te initialiseren, wat in te stellen opties of druk een aangepaste opstartbanner af. Dit kan worden bereikt door een subroutine te maken met de naam **MM.STARTUP**. Wanneer de PicoMite voor het eerst wordt opgestart of gereset, zoekt hij naar deze subroutine en, indien gevonden, het wordt een keer uitgevoerd.

Als de PicoMite bijvoorbeeld een realtime klok heeft, kan het programma de volgende code bevatten:

```
SUB MM.STARTUP
  RTC GETTIME
END SUB
```

Dit zou ertoe leiden dat de interne klok binnen **MMBasic** bij elke opstart of reset op de huidige tijd wordt ingesteld.
Nadat de code in **MM.STARTUP** is uitgevoerd, zal MMBasic doorgaan met het uitvoeren van de rest van het programma in programma geheugen. Als er geen andere code is, keert MMBasic terug naar de opdrachtprompt.

Merk op dat u **MM.STARTUP** niet moet gebruiken voor de algemene instelling van MMBasic (zoals het dimensioneren van arrays, openen communicatiekanalen, enz.) voordat u een programma uitvoert. De reden is dat wanneer u het **RUN** **commando** gebruikt MMBasic zal de status van de tolk wissen, klaar voor een nieuwe start.

MM.PROMPT

Als er een subroutine met deze naam bestaat, wordt deze automatisch uitgevoerd door MMBasic in plaats van de opdrachtprompt. Dit kan worden gebruikt om een aangepaste prompt weer te geven, kleuren in te stellen, variabelen te definiëren, enz zal actief zijn bij de opdrachtprompt. Merk op dat MMBasic alle variabelen en I/O-pininstellingen zal wissen wanneer een programma wordt uitgevoerd, dus alles wat hierin is ingesteld subroutine is alleen geldig voor opdrachten die bij de opdrachtprompt worden getypt (d.w.z. in de directe modus).

Als voorbeeld wordt in het volgende een aangepaste prompt weergegeven:

```
SUB MM.PROMPT
  PRINT TIME$ "> ";
END SUB
```

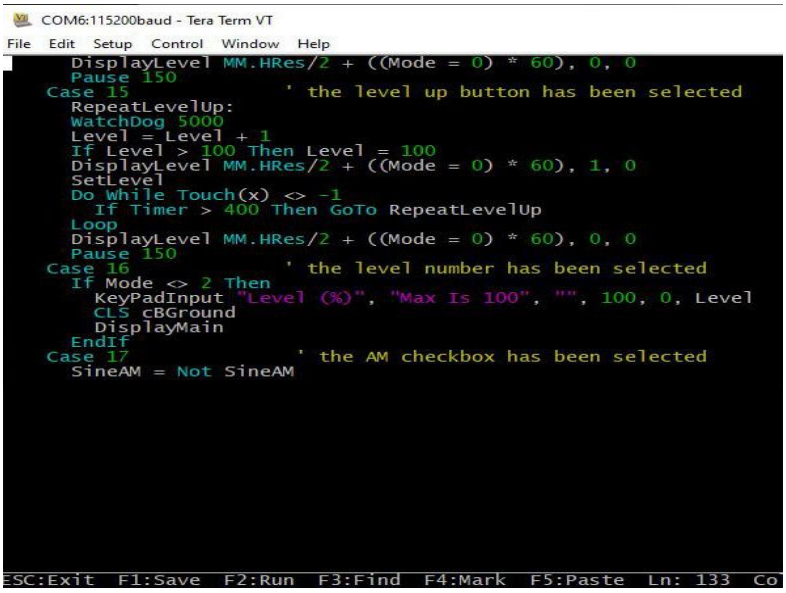
Merk op dat hoewel constanten kunnen worden gedefinieerd, ze niet zichtbaar zullen zijn omdat een constante gedefinieerd is binnen een subroutine is lokaal voor een subroutine. **DIM** zal echter globale variabelen maken die in plaats daarvan moeten worden gebruikt.

MM.END

Als er een subroutine met de naam **MM.END** in het programma bestaat, wordt deze uitgevoerd wanneer het programma eindigt met een werkelijke of impliciete **END** opdracht. Deze wordt niet uitgevoerd als het programma eindigt met een **Ctrl+C**. De optionele parameter 'noend' voor de **END** opdracht kan worden gebruikt om de uitvoering van de **MM.END** subroutine te blokkeren indien nodig (zie de **END** opdracht voor meer informatie)

Edit op volledig scherm

Een belangrijke productiviteitsfunctie is de ingebouwde editor voor volledig scherm. Tijdens het draaien ziet het er als volgt uit:



Wanneer de editor opstart, wordt de cursor automatisch gepositioneerd op de laatste plaats waar u aan het bewerken was of, als uw programma net is gestopt door een fout, wordt de cursor op de regel geplaatst die de fout heeft veroorzaakt.
Aan de onderkant van het scherm geeft de statusregel details weer zoals de huidige cursorpositie en de algemene functies ondersteund door de editor.

Als u eerder een editor zoals Windows Kladblok hebt gebruikt, zult u merken dat de werking van deze editor is: bekend. De pijltjestoetsen verplaatsen de cursor in de tekst, home en end brengen je naar het begin of het einde van de lijn.
Pagina omhoog en pagina omlaag zullen doen wat hun titels suggereren. De delete-toets verwijdert de teken bij de cursor en backspace verwijdert het teken voor de cursor. De invoegtoets zal wisselen, tussen invoeg- en overschrijf modus.

Ongeveer de enige ongebruikelijke toetscombinatie is dat twee keer indrukken van de hometoets zal, brengt u naar het begin van het programma en twee keer indrukken van de eindtoets brengt u naar het einde.
De volledige selectie van commando's is:

ESC	Dit zorgt ervoor dat de editor alle wijzigingen verlaat en terugkeert naar de opdrachtprompt met het bestand ongewijzigd. Als u de tekst heeft gewijzigd, wordt u gevraagd om op **ESC** . te drukken nog twee keer om deze actie te bevestigen.
F1: SAVE	Hiermee wordt het bestand opgeslagen en keert u terug naar de opdrachtprompt.
F2: RUN	Hiermee wordt het bestand opgeslagen en onmiddellijk uitgevoerd.
F3: FIND of **CTRL-R**	Hiermee gaat u naar de zoekmodus. U wordt gevraagd om de zoektekst en als u type dit in de editor zal de cursor automatisch op de eerst gevonden tekst plaatsen.
SHIFT-F3	Nadat u de zoekfunctie heeft gebruikt, kunt u de zoekopdracht herhalen door op **SHIFT-F3** te drukken
F4: MARK	Hiermee komt u in de selectiemodus. In deze modus kunt u de pijltjestoetsen, pijl-omlaag gebruiken om naar het volgende exemplaar te zoeken of de omhoog pijltjestoets voor het vorige voorval. De **Enter-toets** laat de cursor hier staan positie en keer terug naar de normale bewerkingsmodus. **F5** of **CTRL-V** vervangt de gezochte tekst met wat er op het klembord staat (zie hieronder). **Escape** zal het zoeken afbreken.
CTRL-T	**HOME** of **END** om tekst te selecteren en naar het klembord te kopiëren. Het wordt op het scherm gemarkeerd als jij het selecteert. Dan kopieert **F5** of **CTRL-C** de selectie naar het klembord, **F4** of **CTRL-X** kopieert en verwijdert de selectie. **DELETE** verwijdert de selectie gewoon en **ESCAPE** keert terug naar de normale bewerkingsmodus zonder iets te veranderen.
F5 : PASTE of **CTRL-Y**	Hiermee wordt (op de huidige cursorpositie) de tekst ingevoegd die eerder was geknipt of gekopieerd in de selectiemodus (zie hierboven).
END of **CTRL-K**	Verplaats de cursor naar het einde van de regel.
TAB	Verplaatst de cursor naar de volgende tabpositie zoals gedefinieerd door **OPTION TAB**.

In plaats van de hierboven genoemde functietoetsen kunt u ook bedieningstoetsen gebruiken. Deze bedieningstoetsen zijn:

LINKS	Ctrl-S	**RECHTS**	Ctrl-D	**OMHOOG**	Ctrl-E	**OMLAAG**	Ctrl-X
HOME	Ctrl-U	**END**	Ctrl-K	**PageUp**	Ctrl-P	**PageDn**	Ctrl-L
DEL	Ctrl-]	**INSERT**	Ctrl-N	**F1**	Ctrl-Q	**F2**	Ctrl-W
F3	Ctrl-R	**Shift F3**	Ctrl-G	**F4**	Ctrl-T	**F5**	Ctrl-Y

Als u **Tera Term**, **Putty**, **MMEdit** of **GFXterm** als terminalemulator gebruikt, is het ook mogelijk om de cursor door met de linkermuisknop op de muis van de pc te klikken in het venster van de terminal emulator. De beste manier om te leren hoe je de editor moet gebruiken, is door hem gewoon op te starten en te experimenteren.
De editor is een zeer productieve methode om een programma te schrijven. Met het commando **EDIT** kunt u uw programma kunt u het programma opslaan en uitvoeren door op de **F2**-toets te drukken. Als uw programma stopt met een fout door op de functietoets **F4** te drukken bij de opdrachtprompt wordt de opdracht **EDIT** uitgevoerd en gaat u terug naar de editor met de cursor op de regel die de fout veroorzaakte. Deze cyclus voor bewerken/uitvoeren/bewerken is erg snel.

Kleur gecodeerde editor weergave

De editor kan het bewerkte programma een kleurcode geven met trefwoorden, cijfers en opmerkingen die in verschillende kleuren. Deze functie kan worden in- of uitgeschakeld met het commando:

```
OPTION COLOURCODE ON of OPTION COLOURCODE OFF
```

Deze instelling wordt opgeslagen in een niet-vluchtig geheugen en automatisch toegepast bij het opstarten.

Normaal start **EDIT** scherm op in **23 regels** en **80 karakters** en werkt lastig met meer regels en meer karakters op een regel omdat de horizontale scrol balk ontbreekt. Dit is op te lossen door er een scherm vullend geheel van te maken met **OPTION DISPLAY 43, 197** en in **Tera Term** bij **Setup: - Font: - Lettertype: - Punten:** in te stellen op **11 punten.**

Voorbeeld van Option List met Maker Pi Pico bord:

```
PicoMite MMBasic Version 5.07.05RC1      ' Nieuw in V5.07.05RC1
OPTION SYSTEM SPI GP10,GP11,GP12         ' SPI SCK, MOSI=DOUT, MISO=DIN
OPTION LCDPANEL ILI9341, L,GP8,GP7,GP6   ' ILI9341, OR, DC, RESET,CS,BL
OPTION TOUCH GP9,GP16                    ' TOUCH CS, IRQ, BEEP
OPTION SYSTEM I2C GP4,GP5                ' I2C SDA, SCL
OPTION COLOURCODE ON                     ' COLOURCODE FOR EDIT ON/OFF
OPTION DISPLAY 40, 80                    ' DISPLAY aantal LINE, COL
OPTION GUI CONTROLS 59                   ' GUI RAM = 52 Bytes * 59 = 3KB
OPTION SDCARD GP15                       ' SDCARD CS
OPTION RTC AUTO ENABLE                   ' RTC AUTO = Tijd uit RTC
```

Voorbeeld van Option List met een BreadBoard en ILI9341 LCD met LED driver:

```
OPTION SYSTEM SPI GP18,GP19,GP16         ' SPI SCK, MOSI=DOUT, MISO=DIN
OPTION SYSTEM I2C GP4,GP5                ' I2C SDA, SCL
OPTION COLOURCODE ON                     ' COLOURCODE FOR EDIT ON/OFF
OPTION DISPLAY 40, 80                    ' DISPLAY aantal LINE, COL
OPTION LCDPANEL ILI9341,RL,GP15,GP14,GP13,GP12'ILI9341,OR,DC,RESET,CS,BL
OPTION GUI CONTROLS 70                   ' GUI RAM = 52 Bytes * 70 = 3K6
OPTION SDCARD GP22                       ' SDCARD CS
OPTION RTC AUTO ENABLE                   ' RTC AUTO = Tijd uit RTC
```

Variabelen en uitdrukkingen

In MMBasic zijn opdrachtnamen, functienamen, labels, variabelenamen, bestandsnamen, enz. niet hoofdlettergevoelig, dus dat **"Run"** en **"RUN"** equivalent zijn en **"dOO"** en **"Doo"** verwijzen naar dezelfde variabele.

Variabelen

Variabelen kunnen beginnen met een alfabetisch teken of onderstrepingsteken en kunnen elk alfabetisch of numeriek bevatten teken, de punt **(.)** en het onderstrepingsteken **(_)**. Ze mogen **maximaal 32 tekens** lang zijn.

Een variabelenaam of een label mag niet hetzelfde zijn als een functie of een van de volgende trefwoorden: `THEN, ELSE, GOTO, GOSUB, TO, STEP, FOR, WHILE, UNTIL, LOAD, MOD, NOT, AND, OR, XOR, AS.`
bijv. **step = 5** is illegaal omdat **STEP** een sleutelwoord is.

MMBasic ondersteunt drie soorten variabelen:

1. **Drijvend punt met dubbele precisie Float = !.**
   Deze kunnen een getal opslaan met een decimaalteken en een breuk (bijv. **45.386**), maar ze zullen nauwkeurigheid verliezen wanneer meer dan 14 cijfers nauwkeurig worden gebruikt. Variabelen met drijvende komma worden gespecificeerd door toe te voegen achtervoegsel '**!**' naar de naam van een variabele (bijv. **i!, nbr!,** etc). Ze zijn ook de standaard wanneer een variabele wordt gemaakt zonder achtervoegsel (bijv. **i, nbr,** etc).

2. **64-bits Getekend Integer = %.**
   Deze kunnen positieve of negatieve getallen opslaan met maximaal 19 decimale cijfers zonder de nauwkeurigheid te verliezen, maar ze kunnen geen breuken opslaan (d.w.z. het deel na de komma). Deze worden gespecificeerd door het toevoegen van de achtervoegsel '**%**' voor de naam van een variabele. Bijvoorbeeld **i%, nbr%,** enz.

3. **Een tekenreeks String = $.**
   Een string zal een reeks karakters opslaan (bijvoorbeeld **"Tom"**). Elk teken in de tekenreeks wordt opgeslagen als een acht bits getal en kan daarom een decimale waarde hebben van **0** tot **255.** Namen van stringvariabelen zijn afgesloten met een '**$**'-symbool (bijvoorbeeld **naam$, s$,** enz.). Strings kunnen maximaal **255 tekens** lang zijn.

Merk op dat het illegaal is om dezelfde variabelenaam met verschillende typen te gebruiken. bijv. met behulp van **nbr!** en **nbr%** in de hetzelfde programma zou een fout veroorzaken.

De meeste programma's gebruiken variabelen met drijvende komma voor rekenkunde, omdat deze kunnen omgaan met de getallen die worden gebruikt in typische situaties en zijn intuïtiever dan gehele getallen bij het omgaan met delen en breuken. Dus, als je dat niet bent stoort u zich aan de details, gebruik dan altijd een drijvende komma.

Constanten

Numerieke constanten kunnen beginnen met een **numeriek cijfer** (**0**-**9**) voor een **decimale** constante, **&H** voor een **hexadecimaal** constante, **&O** voor een **octale** constante of **&B** voor een **binaire** constante. Bijvoorbeeld **&B1000** is hetzelfde als de decimale constante **8**. Constanten die beginnen met **&H**, **&O** of **&B** worden altijd behandeld als een 64-bits geheel getal zonder teken constanten.

```
A = 5 : Z = 1.5
B = 1.23456E5  ->  hetzelfde als 123456
C = 1.23456E+5 ->  hetzelfde als 123456
D = 1.23456E-3 ->  hetzelfde als 0.00123456
E = &HABCD
E = &B00000101
E = &O377
```

Decimale constanten kunnen worden voorafgegaan door een min (-) of plus (+) en kunnen worden afgesloten met '**E**' gevolgd door een exponentnummer om exponentiële notatie aan te duiden. Bijvoorbeeld **1.6E+4** is hetzelfde als **16000**.

Wanneer een constant getal wordt gebruikt, wordt aangenomen dat het een geheel getal is als een decimaalteken of exponent dat niet is gebruikt. **1234** wordt bijvoorbeeld geïnterpreteerd als een geheel getal, terwijl **1234.0** wordt geïnterpreteerd als een zwevend punt nummer.

Stringconstanten moeten worden omgeven door dubbele aanhalingstekens **(")**.
Bijvoorbeeld `"Hallo wereld"`.

```
A$ = "Dit is mijn string" : B$ = "nog een string"
```

OPTIE STANDAARD (OPTION DEFAULT)

Een variabele kan worden gebruikt zonder achtervoegsel (d.w.z. **!, % of $**) en in dat geval gebruikt MMBasic het standaardtype drijvende punt. Het volgende creëert bijvoorbeeld een variabele met drijvende komma:

```
Nbr = 1234
```

Echter. de standaard kan worden gewijzigd met de opdracht **OPTION DEFAULT**. Bijvoorbeeld **OPTION DEFAULT INTEGER** specificeert dat alle variabelen zonder een specifiek type integer zijn. Dus het volgende zal een integer-variabele maken:

```
OPTION DEFAULT INTEGER
Nbr = 1234
```

De standaard kan worden ingesteld op **FLOAT** (wat de standaard is wanneer een programma wordt uitgevoerd), **INTEGER**, **STRING** of **NONE**.
In de laatste moeten alle variabelen specifiek worden getypt, anders treedt er een fout op.
Het **OPTION DEFAULT** commando kan overal in het programma worden geplaatst en op elk moment worden gewijzigd, maar goed de praktijk dicteert dat als het wordt gebruikt, het aan het begin van het programma moet worden geplaatst en ongewijzigd moet blijven.

OPTIE EXPLICIET

Standaard maakt MMBasic automatisch een variabele wanneer er voor het eerst naar wordt verwezen. Dus, **Nbr = 1234** zal maak de variabele aan en stel deze tegelijkertijd in op het getal **1234**. Dit is handig voor kort en snel programma's, maar het kan leiden tot subtiele en moeilijk te vinden bugs in grote programma's. Bijvoorbeeld, in de derde regel van dit fragment is de variabele Nbr verkeerd gespeld als Nbrs. Als gevolg hiervan zou de variabele Nbrs zijn gemaakt met een waarde van nul en de waarde van Totaal zou verkeerd zijn.

```
Nbr   = 1234
Incr  = 2
Total = Nbr + Incr
```

De opdracht **OPTION EXPLICIT** vertelt MMBasic om niet automatisch variabelen te maken. In plaats daarvan moeten ze zijn expliciet gedefinieerd met de commando's **DIM**, **LOCAL** of **STATIC** (zie hieronder) voordat ze worden gebruikt. Het gebruik van deze opdracht wordt aanbevolen ter ondersteuning van goede programmeerpraktijken. Als het wordt gebruikt, moet het worden geplaatst bij de start van het programma voordat er variabelen worden gebruikt.

DIM en LOCAL

De opdrachten **DIM** en **LOCAL** kunnen worden gebruikt om een variabele te definiëren en het type ervan in te stellen, en zijn verplicht wanneer de **OPTION EXPLICIT** commando wordt gebruikt.
Het **DIM** commando maakt een globale variabele die door het hele programma kan worden gezien en gebruikt, inclusief: binnen subroutines en functies. Als u echter wilt dat de definitie alleen zichtbaar is binnen een subroutine of functie, moet u het **LOCAL**-commando gebruiken aan het begin van de subroutine of functie. **LOCAL** heeft precies dezelfde syntaxis als **DIM**.

Voorbeeld1:

```
DIM A(100)    ' definieert een drijvende-komma-array met 101 elementen
                (index van 0 tot 100)
DIM ABC$(50) ' definieer een string-array met 51 elementen
                (index van 0 tot 50)
gebruikers = 4
DIM naam$(gebruikers)
DIM adres$(gebruikers)
DIM Tel$(gebruikers)
```

Voobeeld2:

```
DIM A(5)      ' alle elementen worden geinitialiseerd op 0
a(0) = 123
Print A(0)    ' print 123
Dim A(10)
Print A(0)    ' print dezelfde waarde 123
Print A(10)   ' print 0
```

Als **LOCAL** wordt gebruikt om een variabele op te geven met dezelfde naam als een globale variabele, dan is de globale variabele: verborgen voor de subroutine of functie en alle verwijzingen naar de variabele verwijzen alleen naar de variabele gedefinieerd door het **LOCAL**-commando. Elke variabele die door **LOCAL** is gemaakt, zal verdwijnen wanneer het programma de subroutine verlaat.

Op het eenvoudigste niveau kunnen **DIM** en **LOCAL** worden gebruikt om een of meer variabelen te definiëren op basis van hun typeachtervoegsel of de **OPTIE DEFAULT** die op dat moment van kracht was. Bijvoorbeeld:

```
DIM nbr%, s$
```

Maar het kan ook worden gebruikt om een of meer variabelen met een specifiek type te definiëren wanneer het achtervoegsel van het type niet wordt gebruikt:

```
DIM INTEGER nbr, nbr2, nbr3, enz
```

In dit geval worden nbr, nbr2, nbr3, enz. allemaal gemaakt als gehele getallen. Wanneer u de variabele binnen een programma gebruikt, doet u niet nodig om het achtervoegsel van het type op te geven.

MyStr in het volgende werkt bijvoorbeeld perfect als een stringvariabele:

```
DIM STRING MyStr
MyStr = "Hallo"
```

De **DIM** en **LOCAL** opdrachten accepteren ook de Microsoft-praktijk van het specificeren van het type van de variabele na de variabele met het trefwoord "**AS**". **Bijvoorbeeld:**

```
DIM NR AS INTEGER, s AS STRING
```

In dit geval wordt het type van elke variabele afzonderlijk ingesteld (niet als een groep zoals wanneer het type voor de lijst wordt geplaatst) van variabelen).

De variabelen kunnen ook worden geïnitialiseerd terwijl ze worden gedefinieerd.

Bijvoorbeeld:

```
DIM INTEGER a = 5, b = 4, c = 3
DIM s$ = "Wereld", i% = &H8FF8F
DIM msg AS STRING = "Hallo" + " " + s$
```

De waarde die wordt gebruikt om de variabele te initialiseren, kan een uitdrukking zijn die door de gebruiker gedefinieerde functies bevat.
De opdrachten **DIM** of **LOCAL** worden ook gebruikt om een array te definiëren en alle bovenstaande regels zijn van toepassing wanneer: het definiëren van een array. U kunt bijvoorbeeld gebruiken:

```
DIM INTEGER nbr(10), nbr2, nbr3(5, 8)
```

Bij het initialiseren van een array worden de waarden weergegeven als door komma's gescheiden waarden, waarbij de hele lijst wordt omringd door: beugels.

Bijvoorbeeld:

```
DIM INTEGER nbr(5) = (11, 12, 13, 14, 15, 16)
```

of

```
DIM days(7) AS STRING = ("","Zo","Ma","Di","Wo","Do","Vr","Za")
```

STATISCH

Binnen een subroutine of functie is het soms handig om een variabele aan te maken die alleen zichtbaar is binnen de subroutine of functie (zoals een **LOCAL** variabele) maar behoudt zijn waarde tussen oproepen naar de subroutine of functie.

U kunt dit doen door het **STATIC** commando te gebruiken. **STATIC** kan alleen worden gebruikt binnen een subroutine of functie en gebruikt dezelfde syntaxis als **LOCAL** en **DIM**. Het verschil is dat de waarde ervan behouden blijft tussen oproepen naar de subroutine of functie (d.w.z. deze wordt niet geïnitialiseerd bij de tweede en volgende aanroepen).

Als u bijvoorbeeld de volgende subroutine had en deze herhaaldelijk riep, zou de eerste oproep 5 afdrukken, de tweede 6, de derde 7 enzovoort.

```
SUB Foo
  STATIC var = 5
  PRINT var
  var = var + 1
END SUB
```

Merk op dat de initialisatie van de statische variabele **naar 5** (zoals in het bovenstaande voorbeeld) alleen van kracht zal zijn op de eerste oproep naar de subroutine. Bij volgende aanroepen wordt de initialisatie genegeerd omdat de variabele al was aangemaakt bij de eerste oproep.
Net als bij **DIM** en **LOCAL** kunnen de variabelen die met **STATIC** zijn gemaakt **float**, **integers** of **strings** en **arrays** van deze met of zonder initialisatie. De lengte van de variabelenaam gecreëerd door **STATIC** en de lengte van de **subroutine** of **functienaam** bij elkaar opgeteld mag niet langer zijn dan **32 tekens**.

Voorbeeld:

```
A = 10
B = 20
C = 30
mysub "Hallo"
PRINT A, B, C
End

SUB mysub(a$)
  LOCAL A, B
  A = 123
  B = 456
  C = 789
  D = 8888
  PRINT A$, D
END SUB
```

CONST

Vaak is het handig om een identifier te definiëren die een waarde vertegenwoordigt zonder het risico dat de waarde per ongeluk wordt weergegeven gewijzigd - wat kan gebeuren als variabelen voor dit doel werden gebruikt (deze praktijk moedigt een andere klasse van moeilijk om fouten te vinden).
Met behulp van het **CONST** commando kunt u een identifier maken die zich als een variabele gedraagt, maar is ingesteld op een waarde die niet kan veranderd worden. Bijvoorbeeld:

```
CONST InputVoltagePin = 31
CONST MaxValue        = 2.4
```

De identifiers kunnen vervolgens worden gebruikt in een programma waar ze logischer zijn voor de gewone lezer dan eenvoudig nummers. Bijvoorbeeld:

```
IF PIN(InputVoltagePin) > MaxValue THEN SoundAlarm
```

Op één regel kunnen een aantal constanten worden aangemaakt:

```
CONST InputVoltagePin = 31, MaxValue = 2.4, MinValue = 1.5
```

De waarde die wordt gebruikt om de constante te initialiseren, wordt geëvalueerd wanneer de constante wordt gemaakt en kan een uitdrukking zijn inclusief door de gebruiker gedefinieerde functies.
Het type constante wordt afgeleid van de waarde die eraan is toegekend; dus bijvoorbeeld, **MaxValue** hierboven is a constante met drijvende komma omdat **2.4** een getal met drijvende komma is. Het type van een constante kan ook expliciet worden ingesteld door een typeachter-voegsel te gebruiken (d.w.z. **!, %** of **$**), maar het moet overeenkomen met de toegewezen waarde.

Uitdrukkingen en operatoren

MMBasic zal een wiskundige uitdrukking evalueren met behulp van de standaard wiskundige regels. Bijvoorbeeld, vermenigvuldigen en delen worden eerst uitgevoerd, gevolgd door optellen en aftrekken. Dit worden de regels genoemd hebben voorrang en worden hieronder beschreven.

Dit betekent dat **2 + 3 * 6** zal oplossen tot **20**, dus **5 * 4** en ook **10 + 4 * 3 – 2**.

Als u de interpreter wilt dwingen om eerst delen van de uitdrukking te evalueren, kunt u dat deel van de . omringen uitdrukking tussen haakjes. Bijvoorbeeld, **(10 + 4) * (3 - 2)** zal worden opgelost tot **14** en niet tot **20** zoals het geval zou zijn geweest als de haakjes niet zijn gebruikt.

Het gebruik van haakjes vertraagt het programma niet noemenswaardig, dus gebruik ze royaal als er een kans is dat MMBasic uw bedoeling verkeerd zal interpreteren.

De volgende operatoren, in volgorde van prioriteit, zijn geïmplementeerd in MMBasic. Operators die zich op hetzelfde bevinden niveau (bijvoorbeeld + en -) worden verwerkt met een voorrang van links naar rechts zoals ze op de programmaregel voorkomen.

Rekenkundige operatoren:

^	**Machtsverheffen (bijv. b^n betekent bn).**
*** / \ MOD**	**Vermenigvuldiging, deling, gehele deling en modulus (rest).**
+ -	**Optellen en aftrekken.**

Shift-operators:

x << y x >> y	**Deze werken op een speciale manier**. **<<** betekent dat de geretourneerde waarde is de waarde van **x** verschoven met **y** bits naar links terwijl **>>** betekent de hetzelfde alleen rechts verschoven. Het zijn integerfuncties en alle bits verschoven worden weggegooid. Voor een verschuiving naar rechts worden alle ingevoerde bits ingesteld naar de waarde van het bovenste bit (**bit 63**). Voor een **linker shift** zijn alle **bits geïntroduceerd zijn op nul gezet**.

Logische operatoren:

NOT INV	**Keer de logische waarde aan de rechterkant om** (bijv. **NOT a = b is a <> b**) of **bitsgewijze inversie** van de **waarde** aan de **rechterkan**t (bijv. **a = INV b**)
<> < > <= =<	**Ongelijkheid**, **kleiner dan**, **groter dan**, **kleiner dan of gelijk aan**, **kleiner dan of kleiner dan of gelijk aan** (alternatieve versie),
>= =>	**Groter dan of gelijk aan.**
=	**Gelijkheid**
AND OR XOR	**Conjunctie, Disjunctie, eXclusief OFF.**

Voor Microsoft-compatibiliteit zijn de operatoren **AND**, **OR** en **XOR** integer bitsgewijze operatoren. Bijvoorbeeld, **PRINT (3 AND 6)** geeft het getal **2** weer. Omdat deze operators als beide logische operators kunnen fungeren (voor bijvoorbeeld **IF a=5 AND b=8 THEN …**) en als bitsgewijze operatoren (bijv. **y% = x% AND &B1010**) de interpreter zullen in de war raken als ze in dezelfde uitdrukking worden gemengd. Evalueer dus altijd logische en bitsgewijze uitdrukkingen in aparte uitdrukkingen.

Voorbeeld:

```
IF (a = 1) AND (b = 2) THEN PRINT "ok"
```

of

```
IF ((a = 2) AND (b = 3) AND (c = 3)) OR (d = 4) THEN PRINT "ok"
```

De andere logische bewerkingen resulteren in het **gehele getal 0** (**nul**) voor **onwaar** en **1** voor **waar** Bijvoorbeeld de verklaring **PRINT 4 >= 5** drukt het getal nul af op de uitvoer en de uitdrukking **A = 3 > 2** slaat **+1** op in **A.**

De **NOT**-operator zal de logische waarde aan de rechterkant omkeren (het is geen bitsgewijze inversie), terwijl de **INV**-operator zal een bitsgewijze inversie uitvoeren. Beide hebben de hoogste prioriteit, dus ze zullen stevig aan de volgende binden waarde. Voor normaal gebruik van **NOT** of **INV** moet de te opereren uitdrukking tussen haakjes worden geplaatst.

Bijv.:

```
IF NOT  (A = 3 OR  A = 8) THEN …
```

Tekenreeksoperators:

+	**Voeg twee strings samen**
<> < > <= =>	**Ongelijkheid, kleiner dan, groter dan, kleiner dan of gelijk aan, kleiner dan of gelijk aan (alternatieve versie), groter dan of gelijk aan, groter dan of gelijk aan (alternatieve versie)**
=	**Gelijkheid**

Stringvergelijkingen respecteren case. Bijvoorbeeld **"A"** is groter dan **"a"**.

Drijvende komma en gehele getallen mengen

MMBasic verwerkt automatisch de conversie van getallen tussen drijvende komma en gehele getallen. Als een operatie mengt zowel drijvende komma als gehele getallen (bijv. **PRINT A% + B!**) Het gehele getal wordt geconverteerd naar een drijvende komma nummer eerst, dan wordt de bewerking uitgevoerd en wordt een getal met drijvende komma geretourneerd. Als beide zijden van de operator zijn gehele getallen, dan wordt een gehele bewerking uitgevoerd en wordt een geheel getal geretourneerd.
De enige uitzondering is de **normale deling** (**"/"**) die beide zijden van de uitdrukking altijd zal converteren naar a getal met drijvende komma en retourneert vervolgens een getal met drijvende komma. Voor het delen van **gehele getallen** moet u het gehele getal **delingsoperator** gebruiken**"\"**.

MMBasic-functies retourneren een **float** of **integer**, afhankelijk van hun kenmerken. Bijvoorbeeld, **PIN()** zal retourneer een geheel getal wanneer de pin is geconfigureerd als een **digitale ingang**, maar een **float** wanneer deze is geconfigureerd als een **analoge ingang**.

Indien nodig kunt u een **float** naar een geheel getal converteren met de functie **INT()**. Het is niet nodig om specifiek converteer een geheel getal naar een **float**, maar als het nodig was, zou de **integer**-waarde kunnen worden toegewezen aan **een variabele** met **drijvende komma** en het wordt automatisch omgezet in de opdracht.

64-bits niet-ondertekende gehele getallen

MMBasic op de PicoMite **ondersteunt 64-bits integers met teken**. Dit betekent dat er **63 bits** zijn voor het vasthouden van de getal en één bit (de meest significante bit) die wordt gebruikt om het teken aan te geven (positief of negatief). Echter het is mogelijk om **volledige 64-bits** nietonder-tekende getallen te gebruiken, zolang je geen rekenkundige bewerkingen uitvoert op de getallen.

64-bits niet-ondertekende nummers kunnen worden gemaakt met de voorvoegsels **&H**, **&O** of **&B** voor een nummer en deze nummers kunnen worden opgeslagen in een **integer**-variabele. Je hebt dan een beperkt aantal bewerkingen die je hierop kunt uitvoeren.
Dit zijn **<<** (**naar links schuiven**), **>>** (**naar rechts schuiven**), **AND** (**bitsgewijs en**), **OR** (**bitsgewijs of**), **XOR** (**exclusief bits of**), **INV** (**bitsgewijze inversie**), **=** (**gelijk aan**) en **<>** (**niet gelijk aan**).

Rekenkundige operatoren zoals delen of optellen mogen verward worden door een 64-bits niet-ondertekend nummer en kan onzinnige resultaten opleveren.
Merk op dat verschuiving naar rechts een ondertekende bewerking is. Dit betekent dat als het bovenste bit een één is (een getal met een negatief teken) en je verschuift naar rechts, dan zal het in één verschuiven om het teken te behouden.

Om **64-bits** niet-ondertekende getallen weer te geven, moet u de functies **HEX$()**, **OCT$()** of **BIN$()** gebruiken.

De volgende 64-bits niet-ondertekende bewerking retourneert bijvoorbeeld de verwachte resultaten:

```
X% = &HFFFF0000FFFF0044
Y% = &H800FFFFFFFFFFF
X% = X% AND Y%
PRINT HEX$(X%, 16)
```

Geeft weer **"0800F0000FFFF0044"**

Subroutines en functies

Een door een programma gedefinieerde subroutine of **functie** is gewoon een **blok programmeer-code** dat is opgenomen in een **module** en kan overal in uw programma worden opgeroepen. Het is hetzelfde alsof je je eigen hebt toegevoegd commando of functie aan de taal.

Subroutines

Een subroutine werkt als een commando en kan argumenten hebben (soms een parameterlijst genoemd). In de definitie van de subroutine zien ze er als volgt uit:

```
SUB MYSUB arg1, arg2$, arg3
  <statements>
  <statements>
END SUB
```

En wanneer u de subroutine aanroept, kunt u waarden aan de argumenten toewijzen.

Bijvoorbeeld:

```
MYSUB 23, "Kat", 55
```

Binnen de subroutine heeft **arg1** de waarde **23**, **arg2$** de waarde van **"Kat"**, enzovoort. De argumenten werken zoals gewone variabelen, maar ze bestaan alleen binnen de subroutine en zullen verdwijnen wanneer de subroutine eindigt. U kunnen variabelen met dezelfde naam in het hoofdprogramma hebben en ze worden verborgen door de gedefinieerde argumenten voor de subroutine.

Bij het aanroepen van een subroutine kunt u minder dan het vereiste aantal waarden opgeven en in dat geval de ontbrekende waarden worden verondersteld nul of een lege tekenreeks te zijn. U kunt ook een waarde weglaten in het midden van de lijst en hetzelfde zal gebeuren. Bijvoorbeeld:

```
MYSUB 23, , 55
```

Zal ertoe leiden dat **arg2$** wordt ingesteld op de lege tekenreeks **""**.

In plaats van het type-achtervoegsel (bijv. de **$** in **arg2$**) kunt u het achtervoegsel **AS <type>** gebruiken in de definitie van de subroutine argument en dan zal het argument bekend staan als het gespecificeerde type, zelfs als het achtervoegsel dat niet is gebruikt. Bijvoorbeeld:

```
SUB MYSUB arg1, arg2 AS STRING, arg3
  IF arg2 = "Kat" THEN ...
END SUB
```

Binnen een subroutine kunt u een variabele definiëren met **LOCAL** (die dezelfde syntaxis heeft als **DIM**). deze variabele zal alleen bestaan binnen de subroutine en zal verdwijnen wanneer de subroutine wordt afgesloten.

Functies

Functies zijn vergelijkbaar met subroutines, met als belangrijkste verschil dat de functie wordt gebruikt om een waarde in een uitdrukking te retourneren. De regels voor de lijst met argumenten in een functie zijn vergelijkbaar met subroutines. Het enige verschil is dat haakjes nodig zijn rond de lijst met argumenten wanneer u een functie aanroept, zelfs als er geen argumenten zijn (ze zijn optioneel bij het aanroepen van een subroutine).

Om een waarde uit de functie terug te geven, wijst u een waarde toe aan de naam van de functie binnen de functie. Als de de naam van de functie wordt afgesloten met een **$**, een **%** of een **!** de functie zal dat type teruggeven, anders zal het terugkeren ongeacht de **OPTION DEFAULT** is ingesteld. U kunt ook het type functie specificeren door **AS <type>** toe te voegen aan het einde van de functiedefinitie.

Bijvoorbeeld:

```
FUNCTION Fahrenheit(C) AS FLOAT
  Fahrenheit = C * 1.8 + 32
END FUNCTION
```

Argumenten doorgeven op basis van referentie

Als u een gewone variabele (d.w.z. geen uitdrukking) als waarde gebruikt bij het aanroepen van een subroutine of een functie, argument binnen de subroutine/functie zal terugverwijzen naar de variabele die in de aanroep wordt gebruikt en eventuele wijzigingen in de argument zal ook worden gemaakt voor de opgegeven variabele. Dit wordt het doorgeven van argumenten door middel van verwijzing genoemd.
U kunt bijvoorbeeld als volgt een subroutine definiëren om twee waarden om te wisselen:

```
SUB Swap a, b
  LOCAL t
  t = a
  a = b
  b = t
END SUB
```

In je aanroepprogramma zou je variabelen gebruiken voor beide argumenten:

```
Swap nbr1, nbr2
```

En het resultaat zal zijn dat de waarden van **nbr1** en **nbr2** worden verwisseld.
Om dit te laten werken, moeten het type van de doorgegeven variabele (bijv. nbr1) en het gedefinieerde argument (bijv. a) hetzelfde zijn (in het bovenstaande voorbeeld zijn beide standaard zwevend).

Tenzij u een waarde via het argument moet retourneren, moet u een argument niet als algemeen doel gebruiken variabele binnen een subroutine of functie. Dit komt omdat een andere gebruiker van uw routine onbewust een variabele in hun oproep en die variabele kan **"magisch"** worden gewijzigd door uw routine. Het is veel veiliger om toe te wijzen het argument naar een lokale variabele en manipuleer dat in plaats daarvan.

Arrays doorgeven

Enkele elementen van een array kunnen worden doorgegeven aan een subroutine of functie en ze zullen hetzelfde worden behandeld als een normale variabele. Dit is bijvoorbeeld een geldige manier om de subroutine Swap aan te roepen (hierboven besproken):

```
Swap dat(i), dat(i + 1)
```

Dit type constructie wordt vaak gebruikt bij het sorteren van arrays.

U kunt ook een of meer complete arrays doorgeven aan een subroutine of functie door de array op te geven met leeg beugels in plaats van de normale afmetingen. Bijvoorbeeld **a()**. In de subroutine of functiedefinitie de bijbehorende parameter moet ook worden opgegeven met lege haakjes. Het type (d.w.z. float, integer of string) van de opgegeven argument en de parameter in de definitie moeten hetzelfde zijn.

In de subroutine of functie zal de array de afmetingen van de doorgegeven array erven en deze moeten: gerespecteerd bij het indexeren in de array. Indien nodig kunnen de afmetingen van de array als extra worden doorgegeven argumenten toe aan de subroutine of functie, zodat deze de array correct kan manipuleren. De array wordt doorgegeven referentie, wat betekent dat alle wijzigingen aan de array binnen de subroutine of functie ook van toepassing zijn op: de meegeleverde reeks.

Als bijvoorbeeld het volgende wordt uitgevoerd, worden de woorden **"Hello World"** afgedrukt:

```
DIM MyStr$(5, 5)
MyStr$(4, 4) = "Hallo" : MyStr$(4, 5) = "Wereld"
Concat MyStr$()
PRINT MyStr$(0, 0)

SUB Concat arg$()
  arg$(0, 0) = arg$(4, 4) + " " + arg$(4, 5)
END SUB
```

Vroege uitstap

Er kan slechts één **END SUB** of **END FUNCTION** zijn voor elke definitie van een subroutine of functie. Verlaten vroeg vanuit een subroutine (d.w.z. voordat het **END SUB** commando is bereikt) kunt u de **EXIT SUB** gebruiken opdracht. Dit heeft hetzelfde effect alsof het programma het **END SUB** statement heeft bereikt. Op dezelfde manier kunt u gebruik maken van **EXIT FUNCTION** om een functie voortijdig te verlaten.

Herhaling

Recursie is waar een subroutine of functie zichzelf aanroept. U kunt recursie doen in MMBasic, maar er zijn een aantal problemen (deze zijn een direct gevolg van de beperkingen van microcontrollers en de BASIC taal):

- Er is een vaste limiet aan de diepte van recursie. In de PicoMite is dit 50 niveaus.
- Als je veel argumenten hebt voor de subroutine of functie en veel **LOCAL** variabelen (vooral strings) je zou gemakkelijk geen geheugen meer kunnen hebben voordat je de limiet van 50 niveaus bereikt.
- Alle **FOR...NEXT** lussen en **DO...LOOP**s worden beschadigd als de subroutine of functie recursief is aangeroepen vanuit deze lussen.

Voorbeelden

Er is vaak behoefte aan een speciale opdracht of functie die moet worden geïmplementeerd in MMBasic, maar in veel gevallen deze kunnen worden geconstrueerd met behulp van een gewone subroutine of functie die dan precies hetzelfde zal werken als een ingebouwde in opdracht of functie.
Soms is er bijvoorbeeld een vereiste voor een **TRIM** functie die gespecificeerde tekens zal trimmen van het begin en einde van een string. Het volgende geeft een voorbeeld van hoe een dergelijke eenvoudige functie te construeren in: MMBasis.

Het eerste argument voor de functie is de tekenreeks die moet worden bijgesneden en het tweede is een tekenreeks die de bevat tekens om vanaf de eerste tekenreeks te trimmen. **RTrim$()** knipt de opgegeven tekens vanaf het einde van de tekenreeks, **LTrim$()** vanaf het begin en **Trim$()** vanaf beide uiteinden.

```
' Trim alle tekens in c$ vanaf het begin en einde van s$
Function Trim$(s$, c$)
  Trim$ = RTrim$(LTrim$(s$, c$), c$)
End Function

' Trim alle tekens in c$ vanaf het einde van s$
Function RTrim$(s$, c$)
  RTrim$ = s$
  Do While Instr(c$, Right$(RTrim$, 1))
    RTrim$ = Mid$(RTrim$, 1, Len(RTrim$) - 1)
  Loop
End Function
```

```
' Trim alle tekens in c$ vanaf het begin van s$
Function LTrim$(s$, c$)
  LTrim$ = s$
  Do While Instr(c$, Left$(LTrim$, 1))
    LTrim$ = Mid$(LTrim$, 2)
  Loop
End Function
```

Als voorbeeld van het gebruik van deze functies:

```
S$ = " ****23.56700 "
Print Trim$(s$, " ")
```

Geeft als resultaat: `****23.56700`

```
Print Trim$(s$, " *0")
```

Geeft als resultaat: `23.567`

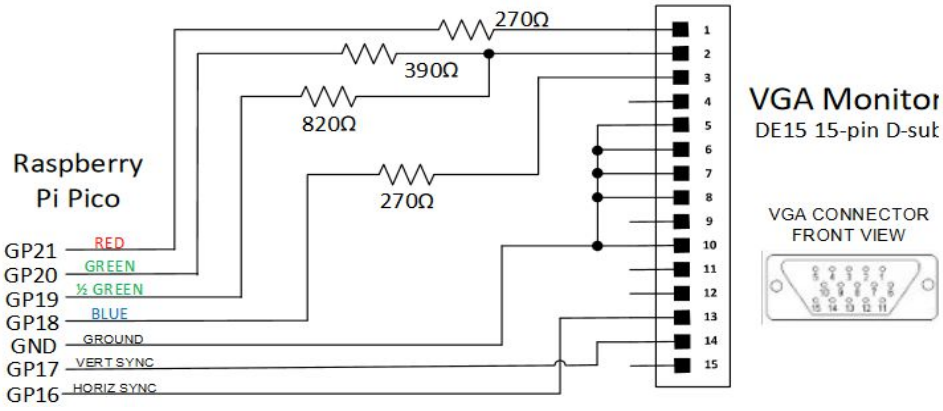
```
Print LTrim$(s$, " *0")
```

Geeft als resultaat: `23.56700`

Video uitgang

VGA video

Voor versies van de firmware die een **VGA** video uitgang ondersteunen, illustreert het volgende diagram hoe u een dergelijke monitor kunt aansluiten. De **VGA** uitgang wordt automatisch ingeschakeld bij het opstarten. Er zijn geen opties die moeten worden ingesteld.



De uitvoer is in het standaard **VGA** formaat met een pixelsnelheid van **25,175 MHz** en een framesnelheid van **60 Hz**.
Er zijn twee of drie modi die kunnen worden geselecteerd met de opdracht **MODE**:

MODE 1	**Monochroom**	met een resolutie van **640 x 480** (standaard bij opstarten)
MODE 2	**16 kleuren**	met een resolutie van **320 x 240**
MODE 3	**16 kleuren**	met een resolutie van **640 x 480** (**alleen RP2350**)

In **MODE 2** en **3** is de uitvoer **16 kleuren** in het 4-bits RGB121 formaat (d.w.z. 1 bit voor rood, 2 bits voor groen en 1 bit voor blauw).
In **MODE 2** worden de pixels gedupliceerd langs zowel de x als de y as, wat een resolutie van **320 x 240** oplevert, terwijl de monitor nog steeds een signaal van **640 x 480** ziet.
De uitvoer van MMBasic wordt als bitmap naar een framebuffer geschreven. De firmware gebruikt vervolgens de tweede CPU in de processor om deze framebuffergegevens, een pixellijn per keer, via DMA naar een van de programmeerbare I/O-controllers (PIO0) van de **RP2040** te sturen om het display te genereren. Omdat dit onafhankelijk van de hoofdprocessors van de **RP2040** draait, is er weinig of geen impact op MMBasic door het genereren van de VGA uitvoer.

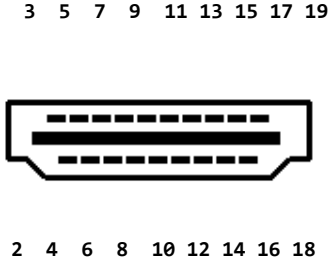
HDMI video

Voor versies van de firmware die HDMI video ondersteunen, geeft de volgende tabel de verbindingen met de standaard **HDMI Type A** aansluiting. De HDMI uitgang wordt automatisch ingeschakeld bij het opstarten. Er zijn geen opties die moeten worden ingesteld.
Bij standaard pin aansluitingen v.b.: `OPTION HDMI PINS 2, 0, 6, 4 '2=D1+,0=D2,6=CK,4=D0`
CK+ =GP14, CK- =GP15, D0+ =GP12, D0- =GP13, D1+ =GP18, D1- =GP19, D2+ =GP16, D2- =GP17

HDMI-pin 1 D2+ 0 : Pin 21 (**GP16**) via een **220Ω** weerstand
HDMI-pin 2 GND : GND
HDMI-pin 3 D2- 1 : Pin 22 (**GP17**) via een **220Ω** weerstand
HDMI-pin 4 D1+ 2 : Pin 24 (**GP18**) via een **220Ω** weerstand
HDMI-pin 5 GND : GND
HDMI-pin 6 D1- 3 : Pin 25 (**GP19**) via een **220Ω** weerstand
HDMI-pin 7 D0+ 4 : Pin 16 (**GP12**) via een **220Ω** weerstand
HDMI-pin 8 GND : GND
HDMI-pin 9 D0- 5 : Pin 17 (**GP13**) via een **220Ω** weerstand
HDMI-pin 10 CK+ 6 : Pin 19 (**GP14**) via een **220Ω** weerstand
HDMI-pin 11 GND : GND
HDMI-pin 12 CK- 7 : Pin 20 (**GP15**) via een **220Ω** weerstand

```
OPTION HDMI PINS 1, 3, 7, 5 'Pinnen voor Olimex.
```

HDMI-pin 13 : Geen verbinding
HDMI-pin 14 : Geen verbinding
HDMI-pin 15 : Geen verbinding
HDMI-pin 16 : Geen verbinding
HDMI-pin 17 : GND
HDMI-pin 18 : +5V via Schottky-barrièrediode
HDMI-pin 19 : Geen verbinding



De HDMI-signaalpennen worden aangestuurd op een hoge frequentie en daarom moet u het volgende doen:

- Houd de signaalsporen zo kort mogelijk.
- Zorg ervoor dat alle signaalsporen dezelfde lengte hebben.
- De **220Ω** weerstand moet bij voorkeur een surface mount type zijn (smd).

Om het DVI/HDMI signaal te genereren, moet de firmware de **RP2350** overklokken tot wel **372**MHz en de meeste Raspberry Pi Pico 2 modules zullen geen problemen hebben met deze snelheden. Dit kan echter niet worden gegarandeerd, vooral niet met modules van derden. Een voorbeeld is de Pimoroni Pico Plus 2 die niet kan worden overgeklokt naar de vereiste snelheden en daarom niet kan worden gebruikt met de HDMI versies van de PicoMite firmware.
Vergelijkbaar met hoe VGA wordt gegenereerd, wordt de uitvoer van MMBasic geschreven naar een framebuffer die, met behulp van de tweede CPU en DMA, wordt gevoerd naar het HSTX randapparaat dat op zijn beurt de parallelle videogegevens genereert. Het geproduceerde videosignaal is in feite DVI (HDMI ondersteunt DVI), wat betekent dat audio niet wordt ondersteund op de HDMI uitgang en geavanceerde HDMI functies zoals High Definition Content Protection (HDCP) en Ethernet worden ook niet ondersteund. HDMI video ondersteunt een aantal resoluties. Om deze in te stellen, gebruikt u de volgende opdracht:

```
OPTION RESOLUTION nn
```

Waarbij **'nn' 640x480** of **640** of **1280x720** of **1280** of **1024x768** of **1024** is.
Elke HDMI-resolutie kan in een aantal modi werken die worden ingesteld met de **MODE** opdracht. Let op dat veel modi de weergegeven resolutie verlagen om geheugen te besparen voor andere functies. Deze verlaging wordt gedaan door elke pixel te verdubbelen of te vervierdubbelen.
De monitor ziet echter altijd de resolutie (d.w.z. pixeldichtheid) die is ingesteld door de opdracht

```
OPTION RESOLUTION 640x480
```

MODE 1 Weergegeven afbeelding **640x480** in **monochroom** RGB555 tegels gebruik de opdracht **TILE**

MODE 2 Weergegeven afbeelding **320x240** in **16 kleuren** met kleurtoewijzing aan een RGB555-palet.

Twee optionele lagen

MODE 3 Weergegeven afbeelding **640x480** in **16 kleuren** met kleurtoewijzing aan een RGB555-palet.

Eén optionele laag

MODE 4 Weergegeven afbeelding **320x240** in **32.768 kleuren**. Eén optionele laag.

MODE 5 Weergegeven afbeelding **320x240** in **256 kleuren** met kleurtoewijzing aan een RGB555-palet.

Twee optionele lagen

OPTION RESOLUTION 1280x720

MODE 1 Weergegeven afbeelding **1280x720** in **monochroom** RGB332 tegels gebruik de opdracht **TILE**

MODE 2 Weergegeven afbeelding **320x180** in **16 kleuren** met kleurtoewijzing aan een RGB332-palet.

Twee optionele lagen

MODE 3 Weergegeven afbeelding **640x360** in **16 kleuren** met kleurtoewijzing aan een RGB332-palet.

Eén optionele laag

MODE 5 Weergegeven afbeelding **320x180** in **256 kleuren** met kleurtoewijzing aan een RGB332-palet.

Twee optionele lagen

OPTION RESOLUTION 1024x768

MODE 1 Weergegeven afbeelding **1024x768** in **monochroom** RGB332-tegels gebruik de opdracht **TILE**

MODE 2 Weergegeven afbeelding **256x192** in **16 kleuren** met kleurtoewijzing aan een RGB332-palet.

Twee optionele lagen

MODE 3 Weergegeven afbeelding **512x384** in **16 kleuren** met kleurtoewijzing aan een RGB332-palet.

Eén optionele laag

MODE 5 Weergegeven afbeelding **256x192** in **256 kleuren** met kleurtoewijzing aan een RGB332-palet.

Twee optionele lagen

De standaard is **RESOLUTION 640x480** en **MODE 1**

VGA/PS2 Referentieontwerp (Raspberry Pi Pico)

Dit is een eenvoudig te monteren ontwerp dat de VGA-uitgang, PS2-toetsenbordinterface en de SD-kaartsocket implementeert (dit ontwerp werd beschreven in Silicon Chip magazine).
Het gebruikt gangbare doorlopende componenten en kan in minder dan een uur worden gemonteerd.
Alle 40 pinnen op de Raspberry Pi Pico worden naar een 40-pins connector aan de achterkant van de PCB geleid in dezelfde configuratie als die wordt gebruikt door de Pico. Dit maakt het eenvoudig om externe apparaten aan te sluiten, omdat u het pin-outdiagram in deze handleiding kunt raadplegen en vervolgens de bijbehorende pinnen op de 40-pins connector kunt selecteren.
Rekening houdend met de I/O-pinnen die zijn gereserveerd voor de VGA-uitgang, het toetsenbord en de SD-kaart, zijn er 14 I/O-pinnen beschikbaar voor externe schakelingen.
Het bord is zo groot dat het in een Altronics snap-to-samen-behuizing past: 130 x 75 x 28 mm (onderdeelnummer H0376).
Het bouwpakket voor dit ontwerp kan worden gedownload van:
https://geoffg.net/picomitevga.html (onderaan de pagina).
HDMI/USB-referentieontwerp (Raspberry Pi Pico 2)
Dit is een volledig uitgerust ontwerp gebaseerd op de Raspberry Pi Pico 2
(met behulp van de RP2350-processors), inclusief:

- HDMI-video-uitgang
- Vier USB-interfaces voor toetsenbord, muis, gamepad, enz.
- Hoogwaardige audio-uitgang voor versterkte luidsprekers.
- USB-interface naar de seriële console.
- Realtimeklok met batterijondersteuning.
- Micro SD-kaartsocket.
- 14 I/O-pinnen beschikbaar op het achterpaneel.
- Op maat gemaakt voor een Multicomp MCRM2015S of Hammond RM2015S behuizing.
  Het bouwpakket voor dit ontwerp kan worden gedownload van:
  https://geoffg.net/picomitevga.html (onderaan de pagina).

Mogelijk indeling bij gebruik voor PicoGAME Mini:

- **VGA DB15:** GP16=Hsync, GP17=Vsync, GP18=Bl, GP19=GrL, GP20=GrH, GP21=Rood
- **Game DB9:** GP00 - GP03 voor drukknoppen en GP26 en GP27 voor joystick en GP14.
- **I2C2 RTC:** GP15=SCL en GP22=SDA.
- **I2C:** GP04=SDA of I/O, GP05=SCL of I/O.
- **Audio:** GP06=L en GP07=R.
- **PS2 keyboard:** GP08=DAT en GP09=CLK.
- **SD kaart:** GP10=MISO, GP11=SCLK, GP12=MOSI, GP13=CS.

De I/O-pinnen gebruiken

De Raspberry Pi Pico heeft 26 input/output-pinnen die vanuit het BASIC-programma kunnen worden aangestuurd met 3 hiervan ondersteunen een snelle ADC (analoog naar digitaal converter).
Een I/O-pin wordt aangeduid met zijn pincode en dit kan het nummer zijn (bijv. 2) of het GP-nummer (bijv. GP1).

Digitale ingangen

Een digitale ingang is het eenvoudigste type ingangsconfiguratie. Als de ingangsspanning hoger is dan **2,5 V**, is de logica niveau zal waar zijn (numerieke waarde van **1**) en alles onder **0,65 V** zal onwaar zijn (numerieke waarde van **0**). De ingangen gebruik een Schmitt-triggeringang zodat alles tussen deze niveaus het vorige logische niveau behoudt. Alle pinnen zijn beperkt tot een maximale spanning van **3,6 V**. Dit betekent dat een weerstandsdeler nodig is als ze worden gebruikt met ingangsspanningen groter dan dat.
In je **BASIC**-programma zou je de ingang instellen als een digitale ingang en de functie **PIN()** gebruiken om het niveau te krijgen. Bij een **RP2350** kan de ingang maximaal **5V5** verdragen.

Bijvoorbeeld:

```
SETPIN GP4, DIN
IF PIN(GP4) = 1 THEN PRINT "Hoog"
```

De opdracht **SETPIN** configureert pin **GP4** als een digitale ingang en de functie **PIN()** retourneert de waarde daarvan pin (het cijfer **1** als de pin hoog is). Het **IF** commando voert het commando dan uit na de **THEN** verklaring als de input hoog was. Als de invoer pin laag was, zou het programma gewoon doorgaan met de volgende regel in het programma.
Het **SETPIN** commando herkent ook een aantal opties die een interne weerstand vanaf de ingang verbinden naar de voeding of naar de aarde. Dit wordt een **"pullup"** of **"pulldown"** weerstand genoemd en is handig bij het aansluiten op een schakelaar omdat het niet nodig is om een externe weerstand te installeren om een spanning over de contacten te plaatsen.

Analoge ingangen

Pinnen gemarkeerd als **ADC** kunnen worden geconfigureerd om de spanning op de pin te meten. Het ingangsbereik is van nul tot **3,3 V** en de functie **PIN()** retourneert de spanning.

Bijvoorbeeld:

```
> SETPIN 31, AIN
> PRINT PIN(31)
2.345
>
```

Je hebt een **spanningsdeler** nodig als je spanningen hoger dan **3,3 V** wilt meten. Voor kleine spanningen kunt u een versterker nodig om de ingangsspanning binnen een redelijk meetbereik te brengen.

De meting gebruikt een voeding van **3,3 V** naar de **CPU** als referentie en er wordt aangenomen dat dit precies is **3.3 V**. Deze waarde kan worden gewijzigd met het **OPTION** commando.

De **ADC** opdrachten bieden een alternatieve methode voor het opnemen van analoge ingangen en zijn bedoeld voor hoge snelheid opname van vele metingen in een **array**.

Ingangen tellen

Elke vier pinnen kunnen worden gebruikt als telingangen om frequentie, periode of gewoon **pulsen** op de ingang te tellen.
De pinnen die voor deze functie worden gebruikt, kunnen worden geconfigureerd met de opdracht **OPTION COUNT**, maar als ze niet worden gewijzigd, standaard ingesteld op **GP6, GP7, GP8** en **GP9**.

Als voorbeeld zal het volgende de frequentie van het signaal op **pin GP7** afdrukken:

```
> SETPIN GP7, FIN
> PRINT PIN(GP7)
110374
>
```

In dit geval is de frequentie **110.374 kHz**.

Standaard is de gate-tijd één seconde, wat de tijdsduur is die MMBasic zal gebruiken om het nummer te tellen aantal cycli op de ingang en dit betekent dat de uitlezing één keer per seconde wordt bijgewerkt met een resolutie van **1 Hz**. Door het specificeren van een derde argument voor het **SETPIN** commando is het mogelijk om een alternatieve gate-tijd op te geven tussen: **10ms** en **100000ms**. Kortere tijden zullen ertoe leiden dat de metingen vaker worden bijgewerkt, maar de waarde teruggestuurd, zal een lagere resolutie hebben. De functie **PIN()** zal het geretourneerde getal altijd schalen als de frequentie in **Hz**, ongeacht de gebruikte gate tijd.

Het volgende stelt bijvoorbeeld de gate tijd in op **10 ms** met een bijbehorend verlies van resolutie:

```
> SETPIN GP7, FIN, 10
> PRINT PIN(GP7)
110300
>
```

Voor nauwkeurige meting van signalen kleiner dan **10 Hz** is het over het algemeen beter om de periode van het signaal te meten.
In deze modus meet de **PicoMite** het aantal **milliseconden** tussen opeenvolgende stijgende flanken van het ingangssignaal. De waarde wordt bijgewerkt bij de overgang van laag naar hoog, dus als uw signaal een periode heeft van (zeg) **100** seconden moet je bereid zijn om die hoeveelheid tijd te wachten voordat de **PIN()** functie een bijgewerkte teruggeeft waarde.

De telpennen kunnen ook het aantal pulsen op hun ingang tellen. Wanneer een pin is geconfigureerd als een teller (voor bijvoorbeeld **SETPIN 7, CIN**) wordt de teller op nul gezet en **PicoMite** telt dan elke overgang van een lage tot hoge spanning. De teller kan weer op nul worden gezet door: `PIN(7) = 0` uit te voeren.

Digitale uitgangen

Alle I/O-pinnen kunnen als digitale uitgang worden geconfigureerd. Dit betekent dat wanneer een uitgangspin is ingesteld op logisch laag, deze zal trek de output naar nul en wanneer deze hoog is ingesteld, trekt hij de output naar **3,3 V**. In MMBasic gebeurt dit met de **PIN** opdracht.
Bijvoorbeeld `PIN(GP15) = 0` zal pin **GP15** op laag zetten, terwijl `PIN(GP15) = 1` het hoog zal zetten.
Merk op dat de maximale spanning op een pin **3,6 V** is voor een **RP2040**, dus open collector uitgangen kunnen niet worden gebruikt om logica met een hogere spanning aan te sturen (d.w.z. **5V**)
OPM: **OC** werkt niet bij **RP2040** en **RP2350**.
De **RP2350** pinnen voor uitgang/ingang kunnen wel max. **5V** verdragen, uitgangs spanning is **3V3**.

Pulsbreedtemodulatie

Met het **PWM** commando (Pulse Width Modulation) kan de PicoMite blokgolven genereren met een programma gecontroleerde werkcyclus. Door de duty-cycle te variëren, kunt u een programma gestuurde spannings uitgang genereren voor gebruik bij het aansturen van externe apparaten die een analoge ingang nodig hebben (voedingen, motorcontrollers, enz.). de **PWM** uitgangen zijn ook handig voor het aansturen van servo's en voor het genereren van een geluidsuitgang via een kleine transducer.

De **PWM**-uitgangen bestaan uit maximaal **8 kanalen** (genummerd van **0** tot **7**) waarbij elk kanaal twee uitgangen heeft (A en B). Voor elk kanaal kan de frequentie worden geselecteerd en voor elke uitgang kan een andere duty cycle worden ingesteld.
Tot **16** pinnen kunnen worden geconfigureerd als **PWM** uitgangen met behulp van het **SETPIN** commando.

Communicatie interfaces (serieel, SPI en I2C)

Deze staan beschreven in de bijlagen achter in deze handleiding. Voordat deze interfaces kunnen worden gebruikt, moeten de pinnen die voor de relevante signalen moeten worden gebruikt, moeten worden geconfigureerd met het **SETPIN** commando.
Sommige apparaten zoals een **SD-kaart**, **LCD-panelen**, **touch**, enz. Gebruiken ook **SPI-** of **I2C** interfaces en de pinnen die worden gebruikt voor deze moeten op dezelfde manier worden geconfigureerd met de opdracht **OPTION SYSTEM** voordat ze kunnen worden gebruikt.

Interrupts

Interrupts zijn een handige manier om met een gebeurtenis om te gaan die zich op een onvoorspelbaar moment kan voordoen. Een voorbeeld is wanneer de gebruiker op een knop drukt. In uw programma zou u na elke instructie code kunnen invoegen om te controleren of de knop is ingedrukt, maar een onderbreking zorgt voor een schoner en beter leesbaar programma.

Wanneer er een interrupt optreedt, zal MMBasic een speciale subroutine uitvoeren en wanneer deze is voltooid terugkeren naar de main programma. Het hoofdprogramma is zich helemaal niet bewust van de onderbreking en gaat gewoon door.

Elke I/O-pin die als digitale ingang kan worden gebruikt, kan worden geconfigureerd om een interrupt te genereren met behulp van de **SETPIN** commando met maximaal tien actieve interrupts tegelijk. Onderbrekingen kunnen worden ingesteld om op te treden bij een stijgende of dalende digitaal ingangssignaal (of beide) en zal een onmiddellijke vertakking naar de gespecificeerde, door de gebruiker gedefinieerde subroutine veroorzaken. Het doel kan voor elke interrupt hetzelfde of verschillend zijn. Terugkeren van een interrupt is via de **END SUB** of **EXIT SUB** opdrachten.

Merk op dat er echter geen parameters kunnen worden doorgegeven aan de subroutine binnen de interrupt aanroepen naar andere subroutines en functies zijn toegestaan.
Als er twee of meer onderbrekingen tegelijkertijd optreden, worden ze verwerkt in de volgorde van de onderbrekingen zoals gedefinieerd onderstaand. Tijdens de verwerking van een interrupt worden alle andere interrupts uitgeschakeld totdat de interrupt-subroutine geeft terug. Tijdens een interrupt (en te allen tijde) is de waarde van de interrupt-pin toegankelijk met behulp van de **PIN()** functie.
Onderbrekingen kunnen op elk moment optreden, maar ze worden uitgeschakeld tijdens **INPUT** instructies.

Interrupts zijn ook niet herkend tijdens een aantal lange hardware-gerelateerde bewerkingen (bijv. de **TEMPR()** functie, **LCD**-tekening commando's en **SD**-toegangscommando's), hoewel ze worden herkend als ze nog steeds aanwezig zijn wanneer de operatie is afgelopen. Bij het gebruik van interrupts wordt het hoofdprogramma volledig onaangetast door de interrupt activiteit tenzij een variabele die door het hoofdprogramma wordt gebruikt tijdens de onderbreking wordt gewijzigd.

Omdat interrupts op de achtergrond worden uitgevoerd, kunnen ze moeilijk te diagnosticeren bugs veroorzaken. Houd rekening met de volgende factoren bij het gebruik van interrupts:

- Voor de meeste programma's reageert **MMBasic** op een interrupt in minder dan **15 µs,** maar sommige opdrachten (zoals zoals de functie **TEMPR()**) kan interrupt tot **200 ms** blokkeren, dus het is mogelijk voor een interrupt (bijv. druk op de knop) binnen dit venster verschijnen en verdwijnen en in dat geval zal het nooit worden herkend.
- Binnen een interrupt worden alle andere interrupts geblokkeerd, dus uw interrupts moeten kort zijn en eindigen als spoedig mogelijk. Gebruik bijvoorbeeld **nooit** **PAUSE** binnen een **interrupt** Als je wat lang hebt verwerking om te doen, moet u gewoon een vlag instellen en de onderbreking onmiddellijk afsluiten, en vervolgens uw hoofdprogramma loop kan de vlag detecteren en doen wat nodig is.
- De subroutine die de interrupt oproept (en alle andere subroutines of functies die daardoor worden aangeroepen) moet altijd exclusief zijn voor de interrupt. Als u een subroutine moet aanroepen die ook door een interrupt wordt gebruikt, moet u schakel eerst de interrupt uit (u kunt deze herstellen nadat u klaar bent met de subroutine).
- Vergeet niet om een onderbreking uit te schakelen wanneer u deze niet meer nodig hebt onderbrekingen op de achtergrond kunnen veroorzaken vreemde en niet-intuïtieve bugs.

Naast interrupts die worden gegenereerd door de statusverandering van een I/O-pin, kan een interrupt ook worden gegenereerd door andere secties van MMBasic inclusief timers en communicatiepoorten en de bovenstaande opmerkingen zijn ook op hen van toepassing.

De lijst van al deze interrupts (in rangorde van hoge tot lage prioriteit) is:

1. ON KEY individu
2. ON KEY algemeen
3. GUI Int Down (niet PicoMiteVGA)
4. GUI Int Up (niet PicoMiteVGA)
5. ADC-completion
6. I2C-Slave Rx
7. I2C Slave Tx
8. I2C2 Slave Rx
9. I2C2 Slave Tx
10. WAV Finished
11. COM1: Seriële poort
12. COM2: Seriële poort
13. IR-ontvangst
14. Toetsenbord
15. Interrupt commando/CSub Interrupt
16. I/O Pin Interrupts in volgorde van definitie
17. Vink Interrupts aan (1 t/m 4 in die volgorde)

Als voorbeeld: Als een **ON KEY** interrupt op hetzelfde moment als **COM1:** plaatsvond onderbreek de **ON KEY** interrupt subroutine zou eerst worden uitgevoerd en dan, wanneer de interrupt subroutine klaar is, de **COM1:** interrupt subroutine zou dan worden uitgevoerd.

Geluidsuitgang

De PicoMite kan stereo **WAV** bestanden op de **SD** kaart afspelen of nauwkeurige sinusgolven genereren met behulp van de **PLAY** commando.

Voordat deze commando's kunnen worden gebruikt, moeten de te gebruiken uitgangspinnen worden toegewezen als audio-uitgangen. Dit is gedaan met de opdracht **OPTION AUDIO**.

Bijvoorbeeld:

```
OPTION AUDIO PWM-A-PIN, PWM-B-PIN
```

Deze opdracht moet worden ingevoerd bij de opdrachtprompt en wordt opgeslagen, dus deze hoeft maar één keer te worden uitgevoerd.

Beide pinnen moeten op hetzelfde **PWM-kanaal** zitten met **PWM-A** het **linker audiokanaal** en **PWM-B** het **rechter**.

Bijvoorbeeld:

```
OPTION AUDIO GP0, GP1
```

Zal pin **GP0** configureren als het **linkerkanaal** en **GP1** als het **rechterkanaal.**

Laagdoorlaatfilter

Het audiosignaal wordt gesuperponeerd op een blokgolf als een pulsbreedtegemoduleerd (**PWM**) signaal. Dit betekent dat een laagdoorlaatfilter is vereist om het audiosignaal te herstellen, zoals hieronder wordt weergegeven. Dit is een eenvoudig voorbeeld dat vertrouwt op condensatorkoppeling in de volgende versterker (de meeste hebben dit) en heeft een uitgangsniveau van ongeveer **1V piek** tot **piek** (**650mV RMS**).

Voor schema zie orginele document pdf: http://geoffg.net/picomite.html

Dit is geschikt voor algemeen gebruik, maar indien nodig kunnen meer geavanceerde ontwerpen worden geïmplementeerd om te verbeteren de frequentierespons en verwerpen meer van de draaggolffrequentie. Deze schakeling is ook geschikt voor het genereren van een DC-uitgangssignaal met behulp van de **PWM opdrachten,** hoewel in dat geval beide **47 nF** condensatoren moeten worden **verhoogd** tot **4,7 µF**.

WAV bestanden afspelen

Het **PLAY** commando speelt een audiobestand op een **SD-kaart** af naar de geluidsuitvoer. Het kan worden gebruikt om zorg voor achtergrondmuziek, voeg geluidseffecten toe aan programma's en geef informatieve aankondigingen.

De opdracht is:

```
PLAY WAV file$, interrupt
```

file$ is de naam van het af te spelen audiobestand. Het moet op de **SD-kaart** staan en de juiste extensie (bijv. **.WAV**) wordt toegevoegd als deze ontbreekt. De audio wordt op de achtergrond afgespeeld (dwz het programma gaat verder zonder pauze). interrupt is optioneel en is de naam van een subroutine die wordt aangeroepen als het bestand klaar is spelen.

Sinusgolven genereren

Het **PLAY TONE** commando gebruikt de **audio-uitgang** om **sinusgolven** te genereren met selecteerbare frequenties voor de linker- en rechterkanalen. Deze functie is bedoeld voor het genereren van opvallende geluiden, maar omdat de frequentie is zeer nauwkeurig, het kan voor veel andere toepassingen worden gebruikt. Bijvoorbeeld het signaleren van DTMF-tonen een telefoonlijn af of het testen van de frequentierespons van luidsprekers.
De syntaxis van het commando is:

```
PLAY TONE left, right, duration, interrupt
```

left en **right** zijn de **frequenties** in **Hz** die moeten worden gebruikt voor de linker- en rechterkanalen. De toon speelt op de achtergrond (het programma blijft draaien na dit commando) en **'dur'** specificeert het aantal **milliseconden** dat de toon zal klinken voor.
duration is optioneel en indien niet gespecificeerd, zal de toon doorgaan tot expliciet gestopt of het programma eindigt.
interrupt (indien gespecificeerd) wordt geactiveerd wanneer de duur is verstreken.

De frequentie kan van **1 Hz** tot **20 KHz** zijn en is zeer nauwkeurig (het is gebaseerd op een kristaloscillator). De frequentie kan op elk moment worden gewijzigd door een nieuw **PLAY TONE** commando te geven. Merk op dat de sinusgolf is gegenereerd door door een opzoektabel te stappen, zodat de audio-uitvoer moet worden doorgegeven om de vervorming te verminderen door een laagdoorlaatfilter.

Gespecialiseerde audio-uitgang

Het **PLAY SOUND** commando genereert een uitvoer op basis van een combinatie van **sinus-, vierkant-,** enz. **golfvormen.** Zie de details in de opdrachtlijst.

Met behulp van PLAY

Het is belangrijk om te beseffen dat het **PLAY** commando de **audio** op **de achtergrond genereert**. Hierdoor kan een programma (bijvoorbeeld) om het geluid van een bel af te spelen terwijl u doorgaat met de bedieningsfunctie. Zonder de achtergrond faciliteit zou het hele BASIC programma bevriezen terwijl het geluid te horen was.
Het genereren van de audio op de achtergrond heeft echter enkele subtiele gevolgtrekkingen die nieuwkomers kunnen doen struikelen.

Neem bijvoorbeeld het volgende programma:

```
PLAY TONE 500, 500, 2000
END
```

Je mag verwachten dat de **500Hz** toon **2** seconden zal **klinken,** maar in de praktijk zal deze helemaal geen geluid maken. Dit is omdat MMBasic het **PLAY TONE** commando zal uitvoeren (dat het geluid begint te genereren in de achtergrond) en dan zal het onmiddellijk het **END** commando uitvoeren dat het programma en het achtergrondgeluid beëindigt. Dit gaat zo snel dat er niets meer te horen is.

Op dezelfde manier zal het volgende programma ook niet werken:

```
PLAY TONE 500, 500, 2000
PLAY TONE 300, 300, 5000
```

Dit komt omdat het eerste commando een **500 Hz** zal instellen voor het afspelen van de toon, maar dan zal het tweede **PLAY** commando vervang dat onmiddellijk door een toon van **300 Hz** en daarna zal het programma aan het einde lopen en de programma's beëindigen (en de achtergrondaudio), waardoor er **niets te horen is**.
Als u wilt dat MMBasic wacht terwijl het **PLAY** commando zijn werk doet, moet u de geschikte **PAUSE** commando gebruiken.

Bijvoorbeeld:

Dit geldt voor **alle versies** van het **PLAY** commando, inclusief **PLAY WAV**.

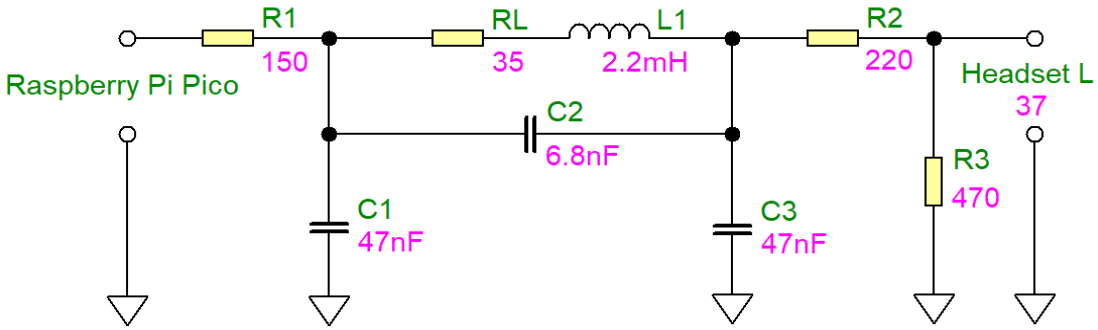
```
PLAY TONE 500, 500
PAUSE 2000
PLAY TONE 300, 300
PAUSE 5000
```

PLAY STOP Hulpprogramma's

Er zijn een aantal commando's die kunnen worden gebruikt om de geluidsuitvoer te beheren:

PLAY PAUSE	**Tijdelijk onderbreken (pauzeren) van het momenteel afgespeelde bestand of de toon.**
PLAY FLAC	**PLAY FLAC** hetzelfde als **PLAY WAV** maar voor **FLAC** bestanden
PLAY RESUME	**Hervat het afspelen van een bestand of toon die eerder was gepauzeerd**
PLAY STOP	**Beëindig het afspelen van het bestand of de toon.** De geluidsuitvoer zal ook automatisch stoppen wanneer het programma eindigt.
PLAY VOLUME L, R	**Stel het volume in tussen 0 en 100, waarbij 100 het maximale volume is.** Het volume wordt teruggezet naar het maximale niveau wanneer een programma wordt uitgevoerd. **Een logaritmische toonladder** wordt gebruikt zodat het **PLAY VOLUME 50, 50** half zo hard moet klinken als **100, 100**.

Het filter dat is herbewerkt op versie van het **PicomiteVGA** bord is dit: Frequentie tot **19kHz**



L1	= 2.2 mH(35)	spoel	Farnell 608622
R1	= 150 ohm	weerstand	Farnell 1700227
C2	= 6.8 nF 5%	condensator	Farnell
R2	= 220 ohm	weerstand	Farnell 1700228
C1,C3	= 47 nF	condensator	Farnell 1100392
R3	= 470 ohm	weerstand	Farnell 9343245

In de volgende versie **Ver 6.00.02** is er ondersteuning voor **I2S** waardoor met een complete module een **HiFi Stereo** audio signaal is te verkrijgen en **PLAY VOLUME** en **PLAY TONE, SOUND** of **PLAY WAV, MP3, FLAC** of **MOD files$** kunnen worden afgespeeld.

```
OPTION AUDIO I2S BCLKpin, DATApin
' OPTION AUDIO I2S BCLKpin=GP1, DATApin=GP5
```

Voorbeeld:

BCK=GP1, LCK=BCK+1=GP2, DIN=GP5, SCK=GND, VIN=5V, GND= -

De **PCM5102A I2S DAC** is een **HiFi Stereo** geluids module met een **DAC,** waardoor bovenstaand filter niet meer nodig is.
Deze modules zijn erg goedkoop (**€3,- PCM5102A** in China) te koop of in Nederland.
Alleen nog een **Versterker** of **Koptelefoon** of een **Multi-point Bluetooth Transmitter** nodig (**€12,-** in China) te koop of in Nederland of een vergelijkbaar type.
Voor de **RP2350A** en **RP2040** vanaf **Versie 6.00.02b10.**

Speciale apparaatondersteuning

Om het voor een programma gemakkelijker te maken om met de buitenwereld te communiceren, bevat de PicoMite stuurprogramma's voor een nummer van gewone randapparatuur.
Dit zijn:

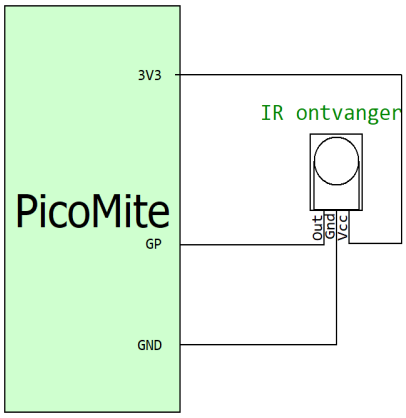
- **Infrarood afstandsbediening ontvanger en zender**
- **De DS18B20 temperatuursensor en DHT22 temperatuur-/vochtigheidssensor**
- **LCD-schermmodules**
- **Numerieke toetsenborden**
- **Klok met batterijvoeding**
- **Ultrasone afstandssensor**
- **WS2812 RGB-LED's**
- **PS2-toetsenbord**



Infrarood Afstandsbediening Decoder

U kunt eenvoudig een afstandsbediening aan uw project toevoegen met behulp van het IR-commando. Indien ingeschakeld, zal deze functie op de achtergrond draaien en het actieve programma onderbreken telkens wanneer een toets op de IR afstandsbediening wordt ingedrukt.

Het werkt met elke **NEC** of **Sony** compatibele afstandsbediening controles, inclusief degenen met uitgebreide berichten Meest goedkope programmeerbare afstandsbedieningen zal een van beide protocollen genereren en een van deze gebruiken u kan een verfijnde flair toevoegen aan uw op PicoMite gebaseerde projecteren. Het **NEC protocol** wordt ook door veel andere fabrikanten, waaronder **Apple, Pioneer, Sanyo, Akai** en **Toshiba** zodat hun afstandsbedieningen van hun merk kunnen worden gebruikt.



Aansluiting IR ontvanger

Om het IR signaal te detecteren heb je een IR **ontvanger** nodig. **NEC** afstandsbedieningen gebruiken een **38 kHz** modulatie van het **IR-signaal** en geschikte ontvangers die op deze frequentie zijn afgestemd, zijn de **Vishay TSOP4838**, **Jaycar ZD1952** en **Altronics Z1611A**. Merk op dat de I/O-pinnen op de **PicoMite** alleen zijn **3,3V** tolerant en dus mag de ontvanger maximaal **3,3V** van stroom voorzien. **Sony** afstands-bedieningen een modulatie van **40 kHz**, maar ontvangers voor deze frequentie kunnen moeilijk te vinden zijn. Over het alge-meen 38 kHz ontvangers zullen werken, maar maximale gevoeligheid wordt bereikt met een **40 kHz**-ontvanger.
De IR ontvanger kan worden aangesloten op elke pin op de **PicoMite**.

Deze pin moet worden geconfigureerd door het programma met behulp van de opdracht:

```
SETPIN n, IR
```

waarbij **n** de I/O-pin is die voor deze functie moet worden gebruikt.

Om de decoder in te stellen gebruik je het commando:

```
IR dev, key, interrupt
```

Waarbij **dev** een variabele is die wordt bijgewerkt met de apparaatcode en **key** is de variabele die moet worden bijgewerkt met de sleutelcode. **Interrupt** is de interrupt-subroutine die moet worden opgeroepen wanneer een nieuwe toetsaanslag is gedetecteerd. Hun het decoderen gebeurt op de achtergrond en het programma zal na deze opdracht zonder onderbreking doorgaan.
Dit is een voorbeeld van het gebruik van de IR-decoder die is aangesloten op de GP6-pin:

```
SETPIN GP6, IR                ' Definieert de te gebruiken pin
DIM INTEGER DevCode, KeyCode ' Variabelen gebruikt door de decoder
IR DevCode, KeyCode, IRInt    ' Start de IR-decoder
DO
   ' < Hoofdtekst van het programma >
LOOP

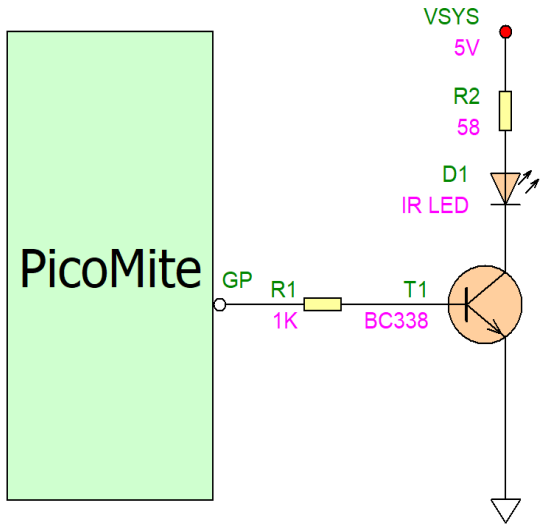
SUB IRInt                     ' Er is een toetsaanslag gedetecteerd
  PRINT "Ontvangen apparaat = " DevCode " key = " KeyCode
END SUB
```

IR-afstandsbedieningen kunnen veel verschillende apparaten aanspreken (videorecorder, tv, enz.), dus het programma zou normaal gesproken onderzoeken eerst de apparaatcode om te bepalen of het signaal bedoeld was voor het programma en, zo ja, dan actie ondernemen op basis van de ingedrukte toets. Er zijn veel verschillende apparaten en sleutelcodes, dus de beste methode om te bepalen welke codes uw afstandsbediening genereert, is om het bovenstaande programma te gebruiken om de codes te ontdekken.

Infrarood afstandsbediening zender

Met het **IRSEND** commando kunt u een **12-bits Sony** infrarood afstandsbedieningssignaal. Dit is bedoeld voor **PicoMite** om PicoMite- of Micromitecommunicatie, maar het zal ook werken met **Sony** apparatuur die **12 bits** codes gebruikt. Merk op dat alle **Sony** producten vereisen dat het bericht drie keer wordt verzonden met een **26ms** vertraging tussen elk bericht

Het circuit aan de rechterkant illustreert wat nodig is de transistor wordt gebruikt om de **infrarood-LED** aan te sturen omdat het uitgangsvermogen van de **PicoMite** is beperkt. Dit circuit levert ongeveer **50 mA** aan de **LED**.



Schema IR Zender

Om een signaal te sturen gebruik je het commando:

```
IRSEND pin, dev, key
```

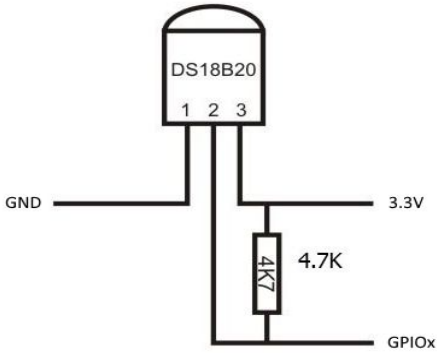
Waar pin de gebruikte I/O-pin is, is dev de te verzenden apparaatcode en is sleutel de sleutelcode. Elke I/O-pin op de **PicoMite** kan worden gebruikt en je hoeft het niet van tevoren in te stellen (**IRSEND** doet dat automatisch).

De gebruikte **modulatiefrequentie** is **38 kHz** en dit komt overeen met de gangbare **IR-ontvangers** (beschreven in de vorige pagina) voor maximale gevoeligheid bij communicatie tussen twee **PicoMites** of met een **Micromite**.

Temperatuur meten

De functie TEMPR() haalt de temperatuur van een DS18B20 temperatuursensor.
Dit apparaat kan op eBay worden gekocht voor ongeveer $ 5 in verschillende pakketten, waaronder een waterdichte sondeversie.

De **DS18B20** kan afzonderlijk worden gevoed door een voeding van **3,3 V** of het kan werken op parasitaire kracht van de **PicoMite** zoals rechts getoond.
Er kunnen meerdere sensoren worden gebruikt, maar een aparte I/O-pin en een **4.7K pullup** voor elk is een weerstand nodig. Om de huidige temperatuur te krijgen, gebruikt u gewoon de functie **TEMPR()** in een uitdrukking.



Bijvoorbeeld:

```
PRINT "Temperatuur: " TEMPR(pin)
```

Waarbij **'pin'** de I/O-pin is waarop de sensor is aangesloten. Je hoeft niet de I/O-pin te configureren, die wordt afgehandeld door MMBasic.

De geretourneerde waarde is in **graden C** met een resolutie van **0,25 C (2)** en is nauwkeurig tot **±0,5 C**. Als er tijdens de meting een fout is opgetreden, geretourneerde waarde is **1000.**
Voor afbeelding zie pdf: http://geoffg.net/picomite.html

De tijd die nodig is voor de algehele meting is **200 ms** en het lopende programma stopt voor deze periode terwijl de meting wordt uitgevoerd.
Dit betekent ook dat interrupts gedurende deze periode worden uitgeschakeld.
Als je dit niet wilt, kunt u de conversie afzonderlijk activeren met commando :
TEMPR START pin, pricisie opdracht en gebruik later de functie **TEMPR()** om de temperatuurmeting ophalen. De functie **TEMPR()** wacht altijd als de sensor nog steeds bezig is met meting.

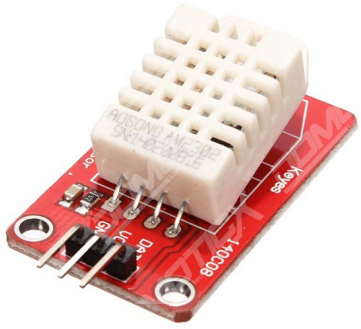
Voorbeeld:
Zet eerst goed: `OPTION SYSTEM I2C GP4, GP5`
`OPTION RTC AUTO ENABLE`

```
' Tijddatetemp
' *** TEMPERATUUR EN AFRONDING, TIJD, DATUM met DS18B20 en DS3231 ***
' RTC Settime 2022,2,13,14,57,10    ' JAAR,MAAND,DAG,UUR,MINUUT,SECONDE
Tijd:
a = TEMPR(GP2)                      ' TEMPERATUUR MET DS18B20 OP GP2
Print                               ' en TEMPR START pin,precisie=1(0-3)
Print "** Raspberry Pi Pico bord **"
Print "Tijd ......... :" Time$      ' TIJD  UITLEZEN MET I2C CHIP DS3231
Print "Datum ........ :" Date$      ' DATUM UITLEZEN MET I2C CHIP DS3231
Print "Temperatuur .. :" Str$(a, 4, 2); " C" ' TEMPERATUUR 0000,00 C
Pause 800                           ' Wacht 800mS ' STR$(a,m,n) m,n=afronden
GoTo Tijd
```

Vochtigheid en temperatuur meten

De opdracht **HUMID** commando leest de vochtigheid en temperatuur van een **DHT22 vochtigheids-/temperatuursensor**. Deze sensor wordt ook verkocht als de **RHT03** of **AM2302**, maar ze zijn allemaal compatibel en kan op eBay worden gekocht voor minder dan $5.
De **DHT11** sensor wordt ook ondersteund.
De **DHT22** moet worden gevoed door **3,3 V** (de maximale spanning) voor de I/O-pinnen van de **PicoMite**) en moet een **pullup-weerstand** hebben op de datalijn zoals weergegeven. Dit is geschikt voor lange kabels (tot **20 meter**) maar voor korte runs kan de weerstand worden weggelaten omdat de PicoMite ook een interne zwakke pullup biedt.
Om de temperatuur of vochtigheid te krijgen, gebruik je het **HUMID** commando met drie argumenten als volgt:

```
HUMID pin, tVar, hVar [,DHT11]
DHT22 pin, tvar, hvar [,DHT11]
' Werk ook zonder BITBANG, BITBANG wordt HUMIT.
```



Module AM2302 met DHT22

Waarbij **'pin'** de **I/O-pin** is waarop de sensor is aangesloten.
De I/O-pin wordt automatisch geconfigureerd door MMBasic.
'tVar' is een variabele met drijvende komma waarin de temperatuur wordt geretourneerd en **'hVar'** is een tweede variabele voor de vochtigheid. De temperatuur wordt teruggegeven als graden C met een resolutie van één decimaal en de vochtigheid wordt geretourneerd als een percentage relatieve vochtigheid (bijv. **54,3**).

Als de optionele **DHT11**-parameter is ingesteld op **1**, dan gebruikt de opdracht apparaattimings die geschikt zijn voor dat apparaat.
In dit geval worden de resultaten geretourneerd met een resolutie van **1 graad** en **1%** vochtigheid.

Voorbeeld1:

```
DIM FLOAT temp, humidity
HUMID GP3, temp, humidity
PRINT "De temperatuur is" temp " en de vochtigheid is" humidity
```

Voorbeeld2:

```
' Tera Term Font: Terminal   Chr$(248) voor graden teken.
' Tera Term Font: Lettertype Chr$(176) voor graden teken.
DIM FLOAT temp, humidity
HUMID GP3, temp, humidity          ' HUMIT pin, temp, humidity, type.
Print"De Temperatuur is"str$(temp,2,1);Chr$(176);"C"    ' Temperatuur in °C
Print "De Vochtigheid is" str$(humidity,2,1); " %"      ' Vochtigheid xx.x %
```

Op de terminal na edit en F1:

```
> run
De Temperatuur is 19.8°C
De Vochtigheid is 46.1 %
```

Werkt zowel voor binnen als buiten maar kan bij een hoge **%RV** verzadigt raken. (RV = > 100 %)
Voor buiten zou een **DS18B20** mogelijk beter werken en dan geen **%RV** of een andere type sensor zie verder in de handleiding maar deze is voor korte afstanden of **Droge/Natte** bol met 2x **DS18B20** voor binnen bij hoge %RV. (Buiten kan de het water bevriezen)

Interface met realtimeklok





Module met DS3231 en EEPROM DS3231 met C= 0.2F

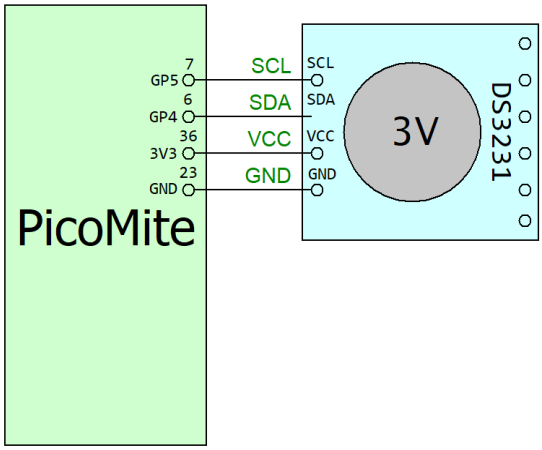
Met behulp van het RTC **GETTIME** commando is het gemakkelijk om de huidige tijd te krijgen van een :

PCF8563	**(Vcc=2V2-5V5)**
DS1307	**(Vcc=4V5-5V5)**
DS3231	**(Vcc=2V3-5V5)**
DS3232	**(Vcc=2V3-5V5)**

Real Time Clock en compatibele apparaten zoals de **M41T11**. Deze geïntegreerde schakelingen zijn populair en goedkoop, houdt de tijd nauwkeurig bij, zelfs als de stroom is verwijderd.
De **PCF8563** en **DS1307** houden de tijd binnen **5min/mnd** bij, terwijl de **DS3231** en **DS3232** zijn bijzonder nauwkeurig en blijven gedurende **een jaar** tot op **een minuut** nauwkeurig. Pas wel op dat de UBat tussen de **2V2** en **3V5** blijft en oppassen met een **GoldCap** en **5V** i.p.v. oplaadbare batterij dan een zener toevoegen of indien met een losse batterij (**CR2032**) de oranje diode of weerstand naast **DS3231** onderbreken anders wordt de batterij een van spanning voorzien.
Met **batterij 8 jaar** en met **Coldcap 1 week** en mogelijk nog langer met in serie een diode.
De Coldcap wordt alleen opgeladen via de zelfde **4K7** van de **SCL**, een lekstroom via **VCC** of SCL?

Deze chips zijn I2C apparaten en moeten worden aangesloten op de I2C I/O-pinnen van de PicoMite. Interne pullup-weerstanden (100KΩ) worden toegepast op de **I2C** I/O-pinnen, dus in veel gevallen externe weerstanden (zoals weergegeven in het diagram) zijn niet nodig.
Om de RTC in te schakelen, moet u eerst de te gebruiken **I2C-pinnen** toewijzen met behulp van de opdracht:

```
OPTION SYSTEM I2C SDApin, SCLpin
OPTION RTC AUTO ENABLE
```



RTC klok op een PicoMite

De door de RTC gebruikte tijd moet ook worden ingesteld. Dat gebeurt met het RTC **SETTIME** commando dat gebruikmaakt van de formaat:

```
RTC SETTIME  Jaar, Maand, Dag, Uur, Min, Sec.
```

Houd er rekening mee dat het uur in 24-uursnotatie moet zijn.
Het volgende zet bijvoorbeeld de realtimeklok op 10 november 2021 om 16:00 uur

```
RTC SETTIME 2021, 11, 10, 16, 0, 0  ' Jaar, Maand, Dag, Uur, Minuut, Seconde
```

Om de tijd te krijgen, gebruikt u het RTC **GETTIME** commando dat de tijd van de real-time klokchip leest en stel de klok in de PicoMite in. Normaal gesproken wordt dit commando aan het begin van het programma geplaatst of in de subroutine MM.STARTUP zodat de tijd bij het opstarten wordt ingesteld. Het commando OPTION RTC **AUTO ENABLE** kan ook worden gebruikt om een automatische update in te stellen.

Afstand meten

Met behulp van een **HC-SR04 ultrasone sensor** en de functie **DISTANCE()** kunt u de afstand tot een doel meten. Dit apparaat meet de afstand tot een doel van **3 cm** tot **3 meter**. Het werkt door een ultrasone geluidspuls te verzenden en de tijd te meten die nodig is voordat de echo wordt geretourneerd. Compatibele sensoren zijn de **SRF05**, **SRF06**, Parallax PING en de **DYP-ME007** (die waterdicht en daarom goed voor het bewaken van het niveau van een watertank).



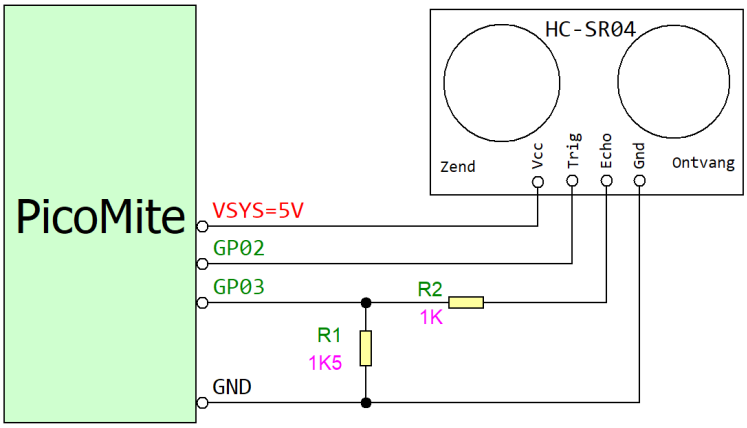
HC-SR04 afstand module

Op de PicoMite gebruik je de **AFSTAND** functie als volgt:

```
d = DISTANCE(trig, echo)
```

De geretourneerde waarde is de afstand in **centimeters** tot het doel.

Waar **trig** de **I/O-pin** is die is aangesloten op de **"trig"** **ingang** van de sensor en **echo** de pin is die is aangesloten op de **"echo"** **uitgang** van de sensor. U kunt ook 3-pins apparaten gebruiken en dan wordt er maar één pincode opge-geven. De maximale spanning op de I/O-pinnen van de **PicoMite** is **3V3**, (**RP2040**) dus er is een weerstandsdeler nodig om de PicoMite naar de echo pin van de **HC-SR04** sensor die werkt op **5V!**.



HC-SR04 aanluiten op een PicoMite

Bij de **RP2350** zijn de I/O-pinnen voor **3V3** - **5V5** geschikt dus is er geen spanningsdeler nodig.

Voorbeeld:

```
' Meet de afstand tot het doel 2 keer/seconde
Print "AFSTAND METING"
For i = 0 To 1000                  ' Voer de meting 1000 keer uit.
  Print DISTANCE(GP2, GP3); " cm" ' Afstand(trig, echo).
  Pause 500                        ' Wacht tussen elke meting 500 mS.
Next i
End
```

LCD scherm

Het LCD-commando zal tekst weergeven op een standaard LCD-module met de minimale programmeerinspanning.

Deze opdracht werkt met **LCD-modules** die de **KS0066** gebruiken, **HD44780** of **SPLC780** controller chip en hebben **1, 2 of 4 lijnen**. Typisch displays zijn onder meer de **LCD16X2.**

Om het display in te stellen, gebruikt u het **LCD INIT** commando:

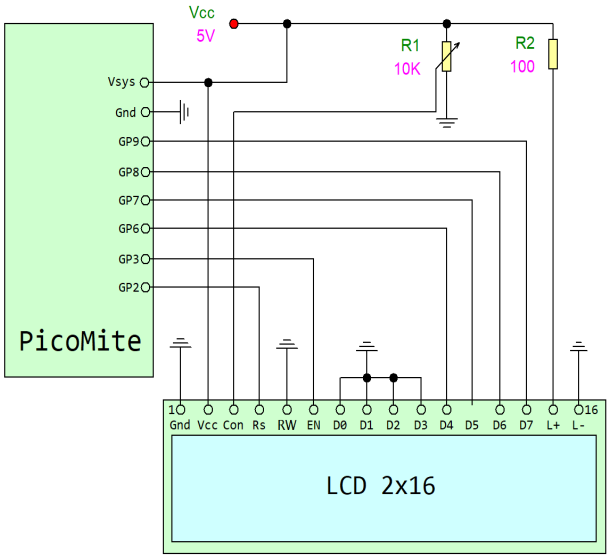
```
LCD INIT d4, d5, d6, d7, rs, en
```



LCD1602 of HD44780

d4, d5, d6 en **d7** zijn de nummers van de **I/O-pinnen** die zijn aan-gesloten op de ingangen **D4, D5, D6** en **D7** op de LCD-module (ingangen **D0** tot **D3** en **R/W** op de module moeten worden aangesloten op **GND**). **'rs'** is de pin die is aangesloten op de registreer select input op de module. (soms **CMD** of **DAT** genoemd) **'en'** is de pin die is aangesloten op de **enable** of chip select ingang op de module. Module werkt op **5V !**
Kan ook soms op **3V3** werken met contrast op max. en serie weerstand voor **LED** verkleinen. Alle I/O-pinnen op de PicoMite kunnen worden gebruikt en u hoeft ze niet van tevoren in te stellen (het LCD commando doet dat automatisch voor u).

Het volgende toont een typische opstelling.
Wees wel voorzichtig met het aansluiten van deze display's omdat de **GP pinnen** op **PicoMite RP2040** kunnen max. **3V6** hebben. Een controle welke LCD pin de **Gnd** is door ohms te meten t.o.v. metalen frame. En een controle met aansluiten door de LCD aan te sluiten alleen **Vcc** aan **Vsys** en de pot-meter en alle **Gnd** pinnen in het schema en dan te meten op **RS, EN, D4-D7**. Blijkt **5V** te zijn.
De **RP2350** pinnen zijn wel voor **3V3** - **5V** geschikt.



Om tekens op de module weer te geven, gebruikt u het LCD commando:

```
LCD line, pos, data$
```

Waar **line** is de lijn op het display (**1** tot **4**) en **pos** is de positie op de lijn waar de gegevens moeten Aansluitschema voor **LCD + level shifter** worden geschreven (de eerste positie op de lijn is 1). **Data$** is een tekenreeks die de gegevens bevat die naar het LCD scherm moeten worden geschreven. De karakters in **data$** overschrijven alles wat op dat deel van het LCD scherm stond.

Het volgende toont een typisch gebruik waarbij **D4** tot **D7** zijn aangesloten op pinnen **GP6** tot **GP9** op de **PicoMite**, **RS** is aangesloten op pin **GP2** en **EN** op pin **GP3**.

```
LCD INIT GP6, GP7, GP8, GP9, GP2, GP3
LCD 1, 2, "Temp. "           ' LCD 1, 2 = regel, positie, data$
LCD 2, 8, STR$(TEMPR(GP15)) ' DS18B20 aangesloten op pin GP15
                             ' LCD 2, 8 = regel, positie
```

Merk op dat dit voorbeeld ook de functie **TEMPR()** gebruikt om de temperatuur te krijgen .

Opm.: Bij een 1x16 display is

```
LCD 1, 1, data$   ' Teken 1- 8
LCD 2, 1, data$   ' Teken 9-16
```

Gebruik men een parallele naar **I2C** omzetter dan heb je ook te maken met **5V** en **3V3** spannings niveau's. De module is aangesloten op **5V** en dan zullen de **I2C** pinnen **SDA** en **SCL** ook **5V** hebben door de pullup weerstanden aangeloten op de ingangen. Door de SMD weerstanden te verwijderen op de module en extern te plaatsen kan gebruik van een levelshifter worden voorkomen. De **RP2350** pinnen zijn wel voor **5V** geschikt, dus geen level shifter nodig.

Toetsenbordinterface

Een toetsenbord is een low-tech methode om gegevens in een op PicoMite gebaseerd systeem in te voeren. De **PicoMite** ondersteunt ofwel een **4x3 keypad** of een **4x4 keypad** en het monitoren en decoderen van toetsaanslagen gebeurt op de achtergrond. Wanneer een toetsaanslag wordt ge-detecteerd, er wordt een interrupt gegeven waar het programma mee om kan gaan.

Voorbeelden van een **4x3 keypad** en een **4x4 keypad** zijn de **Altronics S5381** en **S5383**
Om de toetsenbordfunctie in te schakelen, gebruikt u de opdracht:

```
KEYPAD var,int,r1,r2,r3,r4,c1,c2,c3,c4
```

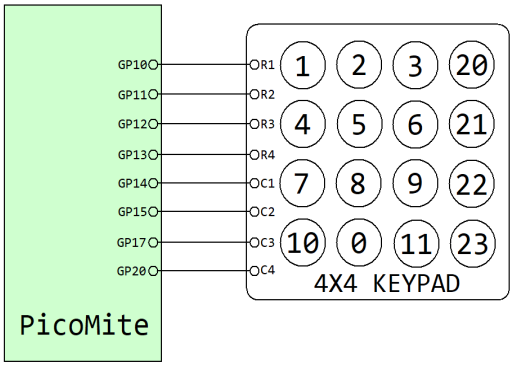
Waarbij var een variabele is die zal worden bijgewerkt met de sleutelcode en int de naam is van de interrupt-subroutine naar bellen wanneer een nieuwe toetsaanslag is gedetecteerd. **r1, r2, r3** en **r4** zijn de **pinnummers** die worden gebruikt voor de **vier rijen** aansluitingen op het bediendeel en **c1, c2, c3** en **c4** zijn de **kolom** aansluitingen. **c4** is alleen gebruikt met **4x4-toetsenborden** en moet worden weggelaten als u een **4x3-toetsenbord** gebruikt.



Altronics S5381 en S5383

Alle I/O-pinnen op de PicoMite kunnen worden gebruikt en u hoeft ze niet van tevoren in te stellen, het **KEYPAD** commando zal dat automatisch voor u doen.

De detectie en decodering van toetsaanslagen gebeurt op de achtergrond en het programma gaat hierna verder commando zonder onderbreking. Wanneer een toetsaanslag wordt gedetecteerd, wordt de waarde van de variabele var ingesteld op de getal dat de sleutel vertegenwoordigt (dit is het getal binnen de cirkels in het schema) dan de interruptie zal gebeld worden.



Schema van een 4x4 keypad op de PicoMite

```
Keypad KeyCode, KP_Int,GP2,GP3,GP4,GP5,GP21,GP22,GP23 ' 4x3 toetsenbord
DO
  < hoofdtekst van het programma >
LOOP
SUB KP_Int                          ' Een toetsaanslag is gedetecteerd.
  PRINT "Toets indrukken = " KeyCode
END SUB
```

WS2812 ondersteuning



Module met 9 x WS2812B

De PicoMite heeft ingebouwde ondersteuning voor de **WS2812** meerkleurige **LED-chip**. Deze chip heeft een zeer specifieke timing om correct te werken en met het **WS2812** commando is het eenvoudig om deze apparaten te bedienen met minimale inspanning.
Deze opdracht geeft de vereiste signalen af die nodig zijn om een keten van **WS2812 LED** chips aan te sturen die zijn aangesloten op de pin gespecificeerd en stel de kleuren van elke **LED** in de keten in
De syntaxis van het commando is:

```
WS2812 type, pin, nbr%, colours%[()]
```

Merk op dat de pin moet worden ingesteld op een digitale uitgang voordat deze opdracht wordt gebruikt. De **colour%()array** moet zijn bemeten om minstens hetzelfde aantal elementen te hebben als het aantal aan te sturen leds (**nbr%**). Elk element in de array moet de kleur in het normale **RGB888**-formaat (**0 - &hFFFFFF**) bevatten. **Colours%(&hFFFFFF)** **R=FF, G=FF, B=FF**
'nbr' is het aantal leds in de keten (**1** tot **256**).
Als er slechts één LED is aangesloten, moet een enkel geheel getal zijn gebruikt voor **nbr%** (dwz niet een array).
Waar een enkele LED naar toe moet worden aangestuurd, dan zou **colour%** een eenvoudige variabele moeten zijn.
Er worden maximaal **256 WS2812** chips in een string ondersteund.
'type' is een enkel teken dat het type chip aangeeft dat als volgt wordt aangestuurd:

```
O = WS2812 origineel (5V)
B = WS2812 B         (3V3)
S = SK6812
```

Voorbeeld op een **Maker Pi Pico bord**:

```
  Dim b%(7) = (RGB(0,0,10),RGB(0,10,0),RGB(0,10,10),RGB(10,0,0),RGB(10,0,10),
RGB(10,10,0),RGB(10,5,0),RGB(0,2,0))
  SetPin GP21, Dout
  WS2812 B, GP21, 8, b%()          ' Een WS2812 met 8 kleuren. I=3mA.
  WS2812 B, GP21, 1, 20            ' WS2812B op pin GP21  20= 13% blue
  WS2812 B, GP21, 1, &h000014      ' WS2812B op pin GP21  20=&h14 blue
  WS2812 B, GP21, 1, RGB(blue)     ' WS2812B op pin GP21 256=100% blue
  WS2812 B, GP21, 1, RGB(0,0,5)    ' WS2812B op pin GP21 5/256=2% blue
```

Het kleur nummer bepaald de **kleur(RGB)** en **helderheid**.
Let op: Stroom verbruik wat voor elke kleur **0-10mA** , dus voor alle kleuren max. **3x10=30mA**

Het is mogelijk dat een **WS2812** niet betrouwbaar werkt met de **3,3V** uitgang van de **PicoMite**.

In dit geval zijn een aantal oplossingen:

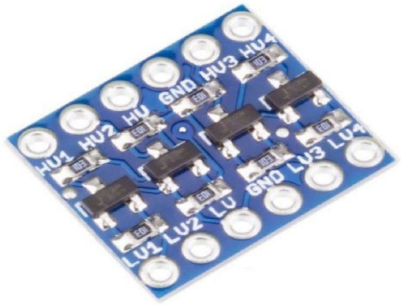
- Gebruik de **WS2812B** die werkt met een voeding en ingangen van **3,3 V**.
- Gebruik een **level shifter** om de **WS2812** te besturen.
- Gebruik een enkele **WS2812** gevoed door **3,3V** als eerste trap om de invoer van de eerst **"echte" LED** in de bedrading. De minimale voeding voor de **WS2812** is **4V**, maar in veel gevallen zal het werken op **3,3V**.
- De **RP2350** pinnen zijn wel voor **5V** geschikt, dus geen level shifter nodig.

PS2 toetsenbord

Op de PicoMite kunt u een **PS2-toetsenbord** aansluiten en de uitvoer van de tolk op het LCD-scherm weergeven (zie de LCD-scherm als het gedeelte Console-uitvoer hieronder).
Dit verandert de PicoMite in een volledig op zichzelf staande computer met eigen toetsenbord en beeldscherm. Met behulp van de ingebouwde **kleurgecodeerde editor** kunnen programma's worden ingevoerd, bewerkt en uitgevoerd zonder dat er een andere computer nodig is.
Omdat de **PicoMite I/O** is gespecificeerd voor maximaal **3,6 V** en **PS2-toetsenborden** run-off **5V**-niveauconversie moet worden gebruikt op de **CLOCK**- en **DATA**-pinnen.
Een geschikte in de handel verkrijgbare Adafruit 4-kanaals bidirectionele niveauconverter is rechts afgebeeld.
De **PS2 CLOCK pin** moet worden aangesloten via het niveau converter naar **PicoMite pin 11** (**GP8**) en de **PS2 DATA pin** naar **PicoMite pin 12** (**GP9**).



PS2-toetsenbord 5V uitvoering



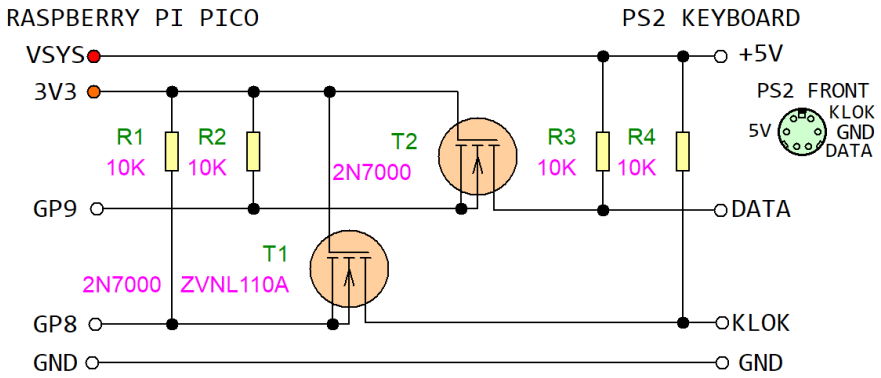
Level shifter module

Voordat het toetsenbord kan worden gebruikt, moet het eerst worden ingeschakeld door de taal op te geven van het toetsenbord:

```
OPTION KEYBOARD language [,capslock] [,numlock]
                         [,repeatstart][,repeatrate]
```

Waar **'taal'** een code van twee tekens is, zoals **VS** voor het standaardtoetsenbord dat wordt gebruikt in de **VS**, **Australië** en **Nieuw-Zeeland**. Andere toetsenbordindelingen die kunnen worden opgegeven, zijn **Verenigd Koninkrijk** (**VK**), **Frans** (**FR**), **Duits** (**GR**), **België** (**BE**), **Italiaans** (**IT**) of **Spaans** (**ES**).

Merk op dat de niet-Amerikaanse lay-outs enkele van de speciale toetsen toewijzen aanwezig op deze toetsenborden, maar het bijbehorende speciale teken wordt niet weergegeven omdat ze geen deel uitmaken van de standaard **PicoMite** lettertypen (in plaats daarvan wordt een ander teken gebruikt). Deze opdracht configureert de I/O-pinnen die aan het toetsenbord zijn toegewezen en initialiseert het voor gebruik, zoals bij soortgelijke commando's wordt deze optie opgeslagen in het flash geheugen en automatisch toegepast bij het opstarten. Als je moet verwijder het toetsenbord, dit doe je met het : **OPTION KEYBOARD DISABLE** commando.
De **RP2350 I/O-pinnen** zijn wel voor **3V3** - **5V** geschikt.



Level shifter discreet. MOSFET T1 en T2 kunnen TN070702N3-G, ZVNL110A of 2N7000 zijn.

Lcd scherm als console uitgang

Het toetsenbord kan op zichzelf worden gebruikt als een alternatieve invoermethode, maar het werkt vooral goed wanneer de LCD-scherm wordt gebruikt als console-uitgang. Het LCD-scherm moet een van de **SSD1963** versies zijn in de landschapsoriëntatie en moet eerst worden geconfigureerd met **OPTION LCDPANEL**.
Om de uitvoer naar het LCD-paneel in te schakelen, moet u de volgende opdracht gebruiken:

```
OPTION LCDPANEL CONSOLE [font [, fc [, bc [, blight]]]
```

'font' is het standaard lettertype, **'fc'** is de standaard voorgrondkleur, **'bc'** is de standaard achtergrondkleur en **'blight'** is de standaard helderheid van de achtergrondverlichting (**2** tot **100**). Deze instellingen worden in flash opgeslagen en worden gebruikt om MMBasic te configureren bij het opstarten. Ze zijn allemaal optioneel en standaard ingesteld op **lettertype 2**, **wit, zwart en 100%**.
Kleurcodering in de editor (zie hieronder) wordt ook ingeschakeld door dit commando (**OPTION COLOURCODE OFF**) zal het weer uitschakelen). Om het gebruik van het LCD-paneel als console uit te schakelen, is het commando **OPTION LCDPANEL NOCONSOLE**.
Option LCDPANEL CONSOLE kan worden gebruikt met **SPI** displays die **BLIT** ondersteunen (ILI9341, ST7789_320, ILI9841 indien **MISO** bedraad) maar het scrollen van tekst op het scherm zal traag zijn.

Bij gebruik met een **PS2-toetsenbord** verandert deze optie de **PicoMite** in een op zichzelf staande computer met een eigen toetsenbord en weergeven. Eerder een moderne versie van de Maximite (zie http://geoffg.net/maximite.html).

PicoMiteVGA Console-uitgang

Standaard gebruikt de **PicoMiteVGA** het VGA-scherm als console-apparaat. Dit kan worden uitgeschakeld met **OPTION LCDPANEL NOCONSOLE** en opnieuw ingeschakeld met:

OPTION LCDPANEL CONSOLE

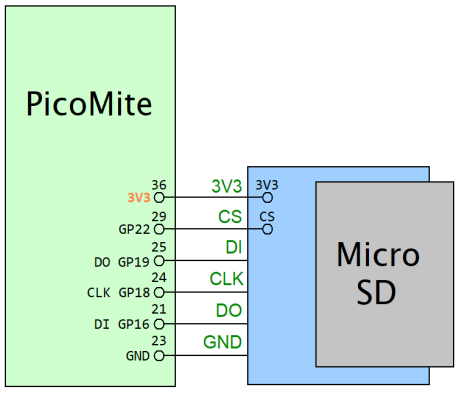
Ondersteuning voor SD-kaarten

De PicoMite heeft volledige ondersteuning voor SD-kaarten. Dit omvat het openen van bestanden voor lezen, schrijven of willekeurige toegang en het laden en opslaan van programma's.

De **firmware** werkt met kaarten tot **32 GB** die zijn geformatteerd in **FAT16** of **FAT32** en de gemaakte bestanden kunnen ook worden lezen/schrijven op pc's met Windows, Linux of het Mac besturingssysteem. De **PicoMite** gebruikt het **SPI-protocol** om met de kaart te praten en dit wordt niet beïnvloed door het kaarttype. Dus alle typen (**klasse 4, 10, UHS-1** enz.) worden ondersteund

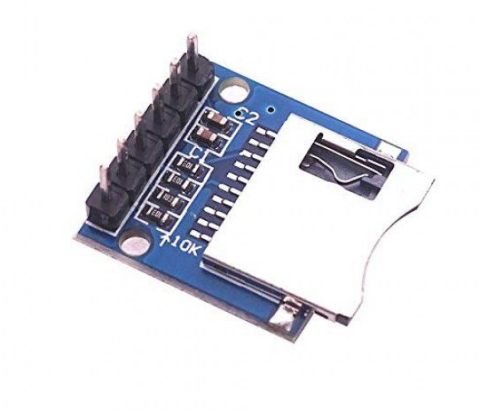
Een SD kaart aansluiten

SD-kaarten gebruiken het **SPI-protocol** voor communicatie en dit moet specifiek worden geconfigureerd voordat ze kunnen worden gebruikt. Eerst moet de **"systeem" SPI-poort** worden geconfigureerd. Dit is een poort die zal worden gebruikt voor systeemgebruik (**SD kaart**, **LCD scherm** en de **touch controller** op een LCD paneel).



Aansluiten micro SD kaart.

Er zijn een aantal poorten en pinnen die kunnen worden gebruikt (zie de sectie PicoMite-hardware), maar dit voorbeeld gebruikt SPI op pinnen **GP18, GP19** en **GP16** voor **CLK, MOSI** en **MISO**.



MicroSD adapter. Let op: Vcc = **3V3**

```
                    CLK    MOSI   MISO
OPTION SYSTEM SPI GP18, GP19, GP16
MOSI = MasterOutSlaveIn
MISO = MasterInSlaveOut
```

Dan moet MMBasic worden verteld dat er een SD-kaart is aangesloten en welke pin wordt gebruikt voor het Chip Select signaal:

```
OPTION SDCARD GP22   ' GP22 = CS pin
```

Deze opdrachten moeten worden ingevoerd bij de opdrachtprompt (niet in een programma) en zullen ervoor zorgen dat de **PicoMite** herstarten. Dit heeft als neveneffect dat de USB-console-interface wordt losgekoppeld, die opnieuw moet worden aangesloten.
Wanneer de PicoMite opnieuw wordt opgestart, zal MMBasic automatisch de SD-kaartinterface initialiseren. Deze **SPI-poort** zal dan niet beschikbaar zijn voor BASIC-programma's (d.w.z. het is gereserveerd). Om de configuratie te verifiëren, kunt u de commando **OPTION LIST** om alle opties weer te geven die zijn ingesteld, inclusief de configuratie van de SD-kaart.
Het basisschema voor het aansluiten van de SD-kaartconnector met behulp van deze pintoewijzingen wordt hieronder weergegeven. Let op: In dit schema komen de pinnummers niet overeen

Merk op dat u veel verschillende configuraties kunt gebruiken met verschillende pintoewijzingen - dit is slechts een voorbeeld op basis van de hierboven vermelde configuratieopdrachten.
Wees voorzichtig wanneer de SPI-poort wordt gedeeld tussen een aantal apparaten (SD-kaart, touch, enz.). In dit geval alle de Chip Select-signalen moeten worden geconfigureerd in MMBasic of anders worden uitgeschakeld door een permanente verbinding met **3,3 V.**

Als dit niet wordt gedaan, zullen zwevende Chip Select-signaallijnen ervoor zorgen dat de verkeerde controller reageert op opdrachten aan de SPI-bus.
Waar geen andere apparaten de **SPI-bus** delen, kan de SD-kaart worden ingesteld met:

```
OPTION SDCARD CSpin, CLKpin, MOSIpin, MISOpin  MOSI=MasterOutSlaveIn.
OPTION SDCARD  GP22,   GP18,  GP19,    GP16    MISO=MasterInSlaveOut.

Master Picomite SPI0 pin GP16 = Rx(MISO) naar Slave SDCARD SPI pin Tx(MISO)
Master PicoMite SPI0 pin GP19 = Tx(MOSI) naar Slave SDCARD SPI pin Rx(MOSI)
Master PicoMite SPI0 pin GP18 = CLK      naar Slave SDCARD SPI pin CLK
```

In dit geval kunnen de pinnen volledig flexibel worden toegewezen en hoeven ze niet in staat te zijn tot SPI-bediening.

MMBasic ondersteuning

MMBasic op de PicoMite ondersteunt de standaard BASIC commando's voor het werken met opslagsystemen.
Let daar op:

- Lange bestands-/mapnamen worden ondersteund naast het oude 8.3-formaat.
- De maximale bestands-/padlengte is **63 tekens**.

- **Hoofdletters/kleine letters** en spaties zijn toegestaan, hoewel het bestandssysteem niet hoofdlettergevoelig is.
- Directorypaden zijn toegestaan in bestands-/directorystrings. (d.w.z. `OPEN "\dir1\dir2\file.txt" FOR …`).
- Voorwaartse of achterwaartse slashes kunnen in paden worden gebruikt. bijv. **\dir\file.txt** is hetzelfde als `/dir/file.txt`.
- De huidige PicoMite-tijd wordt gebruikt voor het maken van bestanden en de laatste toegangs-tijden.
- Er kunnen maximaal **tien bestanden** tegelijk geopend zijn.
- Behalve voor `INPUT`, `LINE INPUT` en `PRINT` is de `#` in **#fnbr** optioneel en kan deze worden weggelaten.

Dit zijn de basiscommando's voor het lezen en schrijven van gegevens op de SD-kaart.

♦ `OPEN fname$ FOR mode AS #fnbr`
Opent een bestand om te lezen of te schrijven. **'fname$'** is de bestandsnaam in 8.3 formaat. **'modus'** kan `INPUT` zijn, `OUTPUT`, `APPEND` of `RANDOM`. **'#fnbr'** is het filenummer (**1** t/m **10**). Let op: **fname$** zo gebruiken `"fname$"`

♦ `PRINT #fnbr, expression [[,; ]expression] … enz`
OUTPUT tekst uit naar het bestand dat is geopend als **#fnbr**. Opmerking: `PRINT` en `?` zelfde.

♦ `INPUT #fnbr, lijst met variabelen`
Lees een lijst met door komma's gescheiden gegevens in de variabelen gespecificeerd uit het bestand dat eerder is geopend als **#fnbr**.

♦ `LINE INPUT #fnbr, variabele$`
Lees een volledige regel in de stringvariabele die is opgegeven uit het bestand dat eerder is geopend als **#fnbr.**

♦ `CLOSE #fnbr [,#fnbr] ...`
Sluit de eerder geopende bestanden af met het bestandsnummer '`#fnbr`'.

Programma's kunnen met twee commando's worden geladen van of opgeslagen op de SD-kaart.
Let op: Alle file opdrachten in fname$ moet tussen aanhalingstekens (`" "`).
Voorbeeld: `LOAD "I2C.bas", R  ' Laad programma I2C.bas van SD en start het`

♦ `LOAD fname$ [, R]`
Laad een BASIC-programma van de SD-kaart. Het optionele achtervoegsel **",R"** zorgt ervoor dat het programma wordt uitgevoerd nadat het is geladen (in dit geval moet **fname$** een string constante zijn). Voorbeeld: `LOAD "fname$`

♦ `RUN fname$`
Laad een BASIC-programma van de SD-kaart en voer het uit. **fname$** moet een tekenreeks-constante zijn. Voorbeeld: `RUN "fname$`

♦ `SAVE fname$`
Sla het huidige programma op de SD-kaart op. Voorbeeld: `SAVE "fname$"`

Afbeeldingen kunnen worden geladen van of opgeslagen op de SD-kaart met behulp van drie commando's.

- **LOAD IMAGE fname$ [,xleft%] [,ytop%]**
  Laad een **BMP**-bestand en geef het weer op het LCD-scherm. Let op: **fname$ = "fname$"**

- **LOAD JPG fname$ [,xleft%] [,ytop%]**
  Laad een **JPG**-bestand en geef het weer op het LCD-scherm.

- **SAVE IMAGE fname$**
  Sla het huidige LCD-schermbeeld op als een **BMP**-bestand (het LCD-paneel moet transparante tekst kunnen bevatten).

Basismanipulatie van bestanden en mappen kan worden gedaan vanuit een BASIC-programma.

- **FILES [jokerteken]**
  Zoek in de huidige map en maak een lijst van de gevonden bestanden/mappen.
  Ook deze tussen aanhalingstekens. **("").**
  Voorbeeld: `FILES "T*.*" ' Laat alle namen beginnen met t zien`

- **KILL fname$**
  Verwijder een bestand in de huidige map. Voorbeeld: **KILL "fname$"**

- **MKDIR dname$**
  Maak een subdirectory in de huidige directory.
  Ook **dname$** tussen aanhalingstekens. **"dname$".**
  Voobeeld: `MKDIR "Picomite" ' Maak subdir Picomite`

- **CHDIR dname$**
  Ga naar de **directory $dname**. **$dname** kan ook **".."** (punt punt) zijn voor één directory of **"\"** voor de root-map. Voorbeeld1: **CHDIR "Hobby"** geeft **A:/HOBBY** met de inhoud retour.
  Voorbeeld2: **CHDIR ".."** geeft **A:/** met de inhoud een map terug retour.

- **RMDIR dir$**
  Verwijder of verwijder de **map "dir$"** op de SD-kaart. Voorbeeld: **RMDIR "Hobby"**

- **SEEK #fnbr, pos**
  Plaatst de **lees-/schrijf aanwijzer** in een bestand dat is geopend voor **RANDOM** toegang tot de **'pos'**-byte.

- **RENAME fromname$ AS toname$**
  Zal het bestand hernoemen **vanname$** om de naam **toname$** te hebben. Ook zijn er een aantal functies die ondersteuning van de bovenstaande commando's.
  Voorbeeld: `> RENAME "ledje.bas" AS "led.bas"`

- **INPUT$(nbr, #fnbr)**
  Geeft een string terug die is samengesteld uit **'nbr'**-tekens die zijn gelezen uit een bestand dat eerder is geopend voor **INPUT** met de dossiernummer **'#fnbr'**. Als er minder dan **'nbr'**-tekens beschikbaar zijn, keert de functie terug met wat hij heeft (inclusief een lege tekenreeks als er geen tekens beschikbaar zijn). Voorbeeld: **> INPUT("1234567,#2")**

- **CWD$**
  Zal de huidige werkdirectory retourneren. Voorbeeld: **? CWD$** geeft **A:/** retour.

- **DIR$(fspec, type)**
  Zal op een SD-kaart naar bestanden zoeken en de namen van gevonden items retourneren. Voorbeeld: **? DIR$("led.bas")** geeft **led.bas** retour.

- **EOF(#fnbr)**
  Zal true retourneren als het bestand dat eerder is geopend voor **INPUT** met het bestands-nummer **'#fnbr'** is gepositioneerd op het einde van het bestand. Let op alleen als het bestand open is.

- **LOC(#fnbr)**
  Voor een bestand dat als **RANDOM** is geopend, wordt de huidige positie van de **lees-/schrijf**-aanwijzer in het bestand geretourneerd. Let op alleen als het bestand open is.

- **LOF(#fnbr)**
  Retourneert de huidige lengte van het bestand in bytes. Let op alleen als het bestand open is.

Een bestand weergeven

De inhoud van een bestand op de SD-kaart kan op de console worden weergegeven met het commando **LIST file$**

XModem overdracht

Naast de standaardmethode van XModem-overdracht die naar of van het programmageheugen kopieert, PicoMite kan ook kopiëren van en naar een bestand op de SD-kaart. De syntaxis is:

```
XMODEM SEND filename$
```
of
```
XMODEM RECEIVE filename$
```

Waarbij **'bestandsnaam$'** het bestand is dat moet worden opgeslagen of verzonden. Zoals gebruikelijk in MMBasic kan **'filename$'** een string zijn uitdrukking, variabel of constant.
Als het een constante is, moet de tekenreeks tussen aanhalingstekens staan.
(bijv. **XMODEM SEND "PRBAS"**) Als een bestand wordt ontvangen, wordt elk bestand op de SD-kaart met dezelfde naam overschreven.

Deze **Option disk** kan de option lijst opslaan of terug lezen van **A:** of **B:**

```
OPTION DISK SAVE/LOAD "filenaam$.opt"
```

Voorbeeld: `OPTION DISK SAVE "PicoMite_RP2350_V60001.opt"  ' Let op .opt aan einde.`

Afbeelding laden en opslaan

De opdracht **LOAD IMAGE** kan worden gebruikt om een bitmapafbeelding van de SD-kaart te laden en weer te geven op een aangesloten LCD-scherm. Dit kan worden gebruikt om een logo te tekenen of een achtergrond op het scherm toe te voegen. de syntaxis van de opdracht is:

```
LOAD IMAGE filename$ [, StartX, StartY]
```

Waar **'filename$'** de afbeelding is die moet worden geladen en **'StartX'/'StartY'** de coördinaten zijn van de linkerbovenhoek van de afbeelding (deze zijn optioneel en worden standaard in de linker-bovenhoek van het scherm weergegeven als ze niet zijn opgegeven).
De afbeelding moet in BMP-indeling zijn en MMBasic zal **".BMP"** aan de bestandsnaam toevoegen als er geen extensie is gespecificeerd. Alle typen van het BMP-formaat worden ondersteund, inclusief zwart-wit en 24-bits afbeeldingen in ware kleuren.

Het huidige beeld op een op **ILI9341** gebaseerd LCD-scherm kan met de volgende opdracht in een bestand worden opgeslagen:

```
SAVE IMAGE filename$ [,StartX, StartY, breedte, hoogte]
```

Hiermee wordt de afbeelding, of een deel van de afbeelding, opgeslagen als een 24-bits ware-kleuren BMP-bestand (de extensie .BMP) toegevoegd als er geen extensie is meegeleverd.

Voorbeeld van sequentiële I/O

In het onderstaande voorbeeld wordt een bestand gemaakt en worden er twee regels naar het bestand geschreven (met het PRINT-commando). Het bestand wordt dan gesloten.

```
OPEN "fox.txt" FOR OUTPUT AS #1
PRINT #1, "De snelle bruine vos"
PRINT #1, "springt over de luie hond"
CLOSE #1
```

U kunt de inhoud van het bestand lezen met de opdracht **LINE INPUT**. Bijvoorbeeld:

```
OPEN "fox.txt" FOR INPUT AS #1
LINE INPUT #1,a$
LINE INPUT #1,b$
CLOSE #1
```

LINE INPUT leest regel voor regel, dus de variabele a$ zal de tekst "The quick brown fox" en b$ bevatten bevat "springt over de luie hond".

Een andere manier om uit een bestand te lezen, is door de functie **INPUT$()** te gebruiken. Dit zal een bepaald aantal . lezen karakters. Bijvoorbeeld:

```
OPEN "fox.txt" FOR INPUT AS #1
ta$ = INPUT$(12, #1)
tb$ = INPUT$(3, #1)
CLOSE #1
```

De eerste **INPUT$()** zal **12** karakters bevatten en de tweede drie karakters. Dus de variabele **ta$** bevat **"De snelle br"** en de variabele **tb$** zullen **"own"** bevatten.

Bestanden bevatten normaal gesproken alleen tekst en de printopdracht converteert getallen naar tekst. Dus in het volgende de eerste regel zal bijvoorbeeld de regel **"123"** bevatten en de tweede **"56789"**.

```
nbr1 = 123 : nbr2 = 56789
OPEN "numbers.txt" FOR OUTPUT AS #1
PRINT #1, nr.1
PRINT #1, nr2
CLOSE #1
```

Nogmaals, u kunt de inhoud van het bestand lezen met het **LINE INPUT** commando, maar dan moet u converteer de tekst naar een getal met **VAL()**.

Bijvoorbeeld:

```
OPEN "numbers.txt" FOR INPUT AS #1
LINE INPUT #1, a$
LINE INPUT #1, b$
CLOSE #1
x = VAL(a$) : y = VAL(b$)
```

Hierna zou de variabele **x** de waarde **123** hebben en **y** de waarde **56789**.

Willekeurige bestands I/O

Voor willekeurige toegang moet het bestand worden geopend met het trefwoord **RANDOM**.

Bijvoorbeeld:

```
OPEN "bestandsnaam" FOR RANDOM AS #1
```

Om naar een record in het bestand te zoeken, gebruikt u het **SEEK** commando dat de lees-/schrijffunctie plaatst verwijzing naar een specifieke byte. De eerste byte in een bestand is genummerd met één, dus bijvoorbeeld de vijfde record in een bestand dat gebruikt **64 byte** records zou beginnen bij **byte 257**. In dat geval zou je het volgende gebruiken om ernaar te verwijzen:

```
SEEK #1, 257
```

Bij het lezen van een willekeurig toegankelijk bestand moet de functie **INPUT$()** worden gebruikt, omdat dit een vast getal zal lezen aantal bytes (d.w.z. een compleet record) uit het bestand. Als u bijvoorbeeld een record van **64 bytes** wilt lezen, gebruikt u:

```
dat$ = INPUT$(64, #1)
```

Bij het schrijven naar het bestand moet een vaste recordgrootte worden gebruikt en dit kan eenvoudig worden bereikt door toe te voegen voldoende opvultekens (normaal gesproken spaties) voor de te schrijven gegevens.

Bijvoorbeeld:

```
PRINT #1, dat$ + SPACE$(64 - LEN(dat$);
```

De functie **SPACE$()** wordt gebruikt om voldoende spaties toe te voegen om ervoor te zorgen dat de geschreven gegevens een exacte lengte hebben (**64 bytes** in dit voorbeeld). De puntkomma aan het einde van de printopdracht onderdrukt de toevoeging van de regelterugloop en linefeedtekens die de plaat langer zouden maken dan bedoeld.
Twee andere functies kunnen helpen bij het gebruik van willekeurige bestandstoegang. De functie **LOC()** retourneert de huidige byte positie van de lees-/schrijfaanwijzer en de **LOF()** functie retourneert de totale lengte van het bestand in bytes.

Het volgende programma demonstreert willekeurige bestandstoegang. Hiermee kun je aan het bestand toevoegen (om wat gegevens toe te voegen) in de eerste plaats) lees/schrijf vervolgens records met behulp van willekeurige recordnummers. De eerste record in het bestand is record nummer 1, de tweede is 2, enz.

```
RecLen = 64
OPEN "test.dat" FOR RND AS #1
DO
  abort: PRINT
  PRINT "Aantal records in het bestand =" LOF(#1) / RecLen
  INPUT "Opdracht (r=lezen,w=schrijven,a=toevoegen,q=afsluiten):",cmd$
  IF cmd$ = "q" THEN CLOSE #1: END
  IF cmd$ = "a" THEN
    SEEK #1, LOF(#1) + 1
  ELSE
    INPUT "Record Nummer: ", nbr
    IF nbr < 1 or nbr > LOF(#1) / RecLen THEN PRINT "Ongeldige record"
    GOTO abort
    SEEK #1, RecLen * (nbr - 1) + 1
  ENDIF
  IF cmd$ = "r" THEN
    PRINT "The record = " INPUT$(RecLen, #1)
  ELSE
    LINE INPUT "Voer de te schrijven gegevens in: ", dat$
    PRINT #1,dat$ + SPACE$(RecLen - LEN(dat$));
  ENDIF
LOOP
```

Willekeurige toegang kan ook worden gebruikt op een normaal tekstbestand. Dit zal bijvoorbeeld een bestand achterstevoren afdrukken:

```
OPEN "file.txt" FOR RND AS #1
  FOR i = LOF(#1) TO 1 STEP -1
    SEEK #1, i
    PRINT INPUT$(1, #1);
  NEXT i
CLOSE #1
```

Displaypanelen

De PicoMite biedt ondersteuning voor veel LCD-schermpanelen met een **SPI, I2C** of parallelle interface.
Deze opdrachten moeten worden ingevoerd bij de opdrachtprompt (niet in een programma) en zullen ervoor zorgen dat de PicoMite herstarten. Dit heeft als neveneffect dat de USB-consoleinterface wordt losgekoppeld, die opnieuw moet worden aangesloten.
Let daar op:

- De maximale spanning op alle **PicoMite** I/O-pinnen is **3,3 V.** Een levelshifter is vereist als een display gebruikt **5V** niveaus voor signalering.
- Ondersteuning voor lcd-scherm en aanraakinvoer is niet beschikbaar op de PicoMiteVGA-versie.

Op SPI gebaseerde weergavepanelen

De op SPI gebaseerde displaycontrollers delen de **SYSTEM SPI**-kanaalinterface op de PicoMite met aanraking regelaar (indien aanwezig). Een **SD-kaart** kan ook worden geconfigureerd om dezelfde pinnen te gebruiken. Als dit klaar is, zijn de pinnen toegewezen aan de **SYSTEM SPI** zijn niet beschikbaar voor andere MMBasic-opdrachten. De snelheid van tekenen naar SPI gebaseerde beeldschermen worden grotendeels niet beïnvloed door de CPU-snelheid.

Deze panelen worden geconfigureerd met behulp van de volgende opdrachten.
In alle commando's zijn de parameters:

- **OR** = Dit is de oriëntatie van het scherm en kan **LANDSCHAP, PORTRET, LANDSCHAP** of **RPORTRET.** Deze kunnen worden afgekort tot **L, P, RL** of **RP**. Het voorvoegsel **R** geeft het omgekeerde of **"upside"** aan naar beneden oriëntatie.
- **DC** = **Display Data/Command** control pin.
- **RESET** = Display **Reset**-pin (indien laag getrokken).
- **CS** = Display **Chip Select**-pin.

Alle vrije pinnen kunnen worden gebruikt.
Wil men OR aanpassen dan eerst OPTION LCDPANEL "....", ,DISABLE maken en opnieuw.

```
OPTION LCDPANEL ILI9341, OR, DC, RESET, CS
```

Initialiseert een **TFT-scherm** met behulp van de **ILI9341-controller**. Dit ondersteunt een resolutie van **320 * 240**. Displays die dit gebruiken controller zijn in staat tot transparante tekst en werken met de **BLIT**- en **BLIT READ** commando's.

```
OPTION LCDPANEL ILI9163, OR, DC, RESET, CS
```

Initialiseert een **TFT-scherm** met behulp van de **ILI9163-controller**. Dit ondersteunt een resolutie van **128 * 128**.

```
OPTION LCDPANEL ILI9481, OR, DC, RESET, CS
```

Initialiseert een **TFT-scherm** met behulp van de **ILI9481-controller**. Dit ondersteunt een resolutie van **480 * 320**.

```
OPTION LCDPANEL ILI9488, OR, DC, RESET, CS
```

Initialiseert een **TFT-display** met behulp van de **ILI9488-controller**. Dit ondersteunt een resolutie van **480 * 320**. Merk op dat dit controller heeft een probleem met de **MISO-pin** die de aanraak-controller verstoort. Om dit scherm te laten werken, **MISO-pin** mag niet worden aangesloten.

```
OPTION LCDPANEL ILI9488W, OR, DC, RESET, CS
```

Initialiseert een **TFT-display** met behulp van de **ILI9488-controller**. Dit ondersteunt het **Wave-share 3,5-inch** scherm zoals gebruikt op hun **Pico Eval bord** en de **normale 3,5"** beeldscherm-adapter.

```
OPTION LCDPANEL N5110, OR, DC, RESET, CS [,contrast]
```

Initialiseert een **LCD-scherm** met behulp van de **Nokia 5110-controller**. Dit ondersteunt een resolutie van **84 * 48** een extra parameter **LCDVOP** kan worden gespecificeerd om het contrast van het display te regelen. Probeer contrastwaarden tussen **&HA8** en **&HD0** voor uw scherm, standaard indien weggelaten is **&HB1**

```
OPTION LCDPANEL SSD1306SPI, OR, DC, RESET, CS [,offset]
```

Initialiseert een **OLED-display** met behulp van de **SSD1306-controller** met een **SPI-interface**. Dit ondersteunt **128 * 64** resolutie.
Er kan een extra parameteroffset worden gespecificeerd om de positie van het display te regelen. **0,96" displays** typisch hebben een **waarde van 0** nodig. Displays van **1,3 inch** hebben meestal een waarde van **2** nodig. De standaardwaarde indien weggelaten is **0**.

```
OPTION LCDPANEL SSD1331, OR, DC, RESET, CS
```

Initialiseert een **OLED-kleurenscherm** met behulp van de **SSD1331-controller**. Dit ondersteunt een resolutie van **96 * 64**.

```
OPTION LCDPANEL ST7735, OR, DC, RESET, CS
```

Initialiseert een **TFT-scherm** met behulp van de **ST7735-controller**. Dit ondersteunt een resolutie van **160 * 128**.

```
OPTION LCDPANEL ST7735S, OR, DC, RESET, CS
```
Initialiseert een **IPS-scherm** met behulp van de **ST7735S-controller**. Dit ondersteunt een resolutie van **160 * 80**.

```
OPTION LCDPANEL ST7789, OR, DC, RESET, CS
```

Initialiseert een **IPS-scherm** met behulp van de **7789-controller**. Dit ondersteunt **240 * 240** resolutie.
OPMERKING: weergaveborden zonder **CS-pin** worden momenteel niet ondersteund op de PicoMite, tenzij gewijzigd.

```
OPTION LCDPANEL ST7789_135, OR, DC, RESET, CS
```

Initialiseert een **IPS-scherm** met behulp van de **7789-controller**. Dit ondersteunt een resolutie van **240 * 135.**
OPMERKING: weergaveborden zonder **CS-pin** worden momenteel niet ondersteund op de PicoMite, tenzij gewijzigd.

```
OPTION LCDPANEL ST7789_320, OR, DC, RESET, CS
```

Initialiseert een **IPS-scherm** met behulp van de **7789-controller**. Dit type ondersteunt de **320 * 240** resolutie weergave van: Waveshare (https://www.waveshare.com/wiki/Pico-ResTouch-LCD-2.8). Deze zijn in staat tot transparante tekst en werken met de **BLIT**- en **BLIT READ** commando's. **OPMERKING:** weergaveborden zonder **CS-pin** worden momenteel niet ondersteund op de PicoMite, tenzij gewijzigd.

```
OPTION LCDPANEL GC9A01, OR, DC, RESET, CS
```

Initialiseert een **IPS-scherm** met behulp van de **GC9A01-controller**. Dit ondersteunt **240 * 240** resolutie.

```
OPTION LCDPANEL ST7920, OR, DC, RESET
```

Initialiseert een **LCD-scherm** met behulp van de **ST7920-controller**. Dit ondersteunt **128 * 64** resolutie. **Let op:** dit scherm doet ondersteunt geen chipselectie, dus de SPI-bus kan niet worden gedeeld als dit display wordt gebruikt.

```
OPTION LCDPANEL GDEH029A1, OR, DC, RESET, CS, BUSYpin [,refreshcount]
```

Initialiseert **een e-Ink-display** met behulp van de **GDEH029A1-controller**. Dit ondersteunt een resolutie van **128 * 296**. . **'BUSYpin'** is een invoer die door de bestuurder wordt gebruikt om de voltooiing van de opdracht op het langzame **e-Ink-display** vast te stellen. Een extra parameter **'refreshcount'** kan worden gespecificeerd om te bepalen wanneer het stuurprogramma een volledige update van het scherm uitvoert (**zwart/wit knipperen**).
De **standaardwaarde is 1**. d.w.z. het scherm vernieuwt elke schrijfbewerking volledig. Een grotere waarde instellen kan updates sneller en minder maken voor de hand liggend, maar met het risico dat er spookbeelden op het scherm ontstaan.

I2C gebaseerde LCD-panelen

Op I2C gebaseerde displaycontrollers gebruiken de **SYSTEM I2C** pinnen volgens de pinout voor het specifieke apparaat. Andere I2C apparaten kunnen de bus delen op voorwaarde dat hun adressen uniek zijn.

Als een **I2C-display** is geconfigureerd, is het niet nodig om de **I2C** te **"openen"** poort voor een extra apparaat (**I2C OPEN**), **I2C CLOSE** is geblokkeerd en alle **I2C**-apparaten moeten **400 kHz** kunnen verwerken. De I2C bussnelheid wordt niet beïnvloed door wijzigingen in de CPU-kloksnelheid.

Deze panelen worden geconfigureerd met behulp van de volgende opdrachten.
In alle commando's is de parameter **OR** de **oriëntatie** van het scherm en dit kan **LANDSCAPE**, **PORTRAIT**, **RLANDSCAPE** of **RPORTRAIT** zijn. Deze kan worden afgekort tot **L, P, RL** of **RP**. Het voorvoegsel **R** geeft de **omgekeerde** of **"ondersteboven"** oriëntatie aan.

```
OPTION LCDPANEL SSD1306I2C, OR [,offset]
```

Initialiseert een **OLED-scherm** met behulp van de **SSD1306-controller** met een **I2C-interface**.
Dit ondersteunt **128 * 64** resolutie. Er kan een extra parameteroffset worden gespecificeerd om de positie van het display te regelen. **0,96"** displays typisch hebben een waarde van **0** nodig. Displays van **1,3 inch** hebben meestal een waarde van **2** nodig. De standaardwaarde indien weggelaten is **0**.
NB veel goedkope **I2C-versies** van **SSD1306-schermen** implementeren **I2C** niet correct door een bedradingsfout. Deze lijkt vooral het geval te zijn bij **1.3"-varianten**.

Opm: Bij aanpassen **OR** eerst **OPTION LCDPANEL SSD1306I2C, ,DISABLE** voor aanpassing.

```
OPTION LCDPANEL SSD1306I2C32, OR
```

Initialiseert een **OLED-scherm** met behulp van de **SSD1306-controller** met een **I2C-interface**. Dit ondersteunt een resolutie van **128 * 32**.

Op SSD1963 gebaseerde lcd-schermen

Naast de op **SPI** en **I2C** gebaseerde controllers die de **PicoMite** ondersteunt LCD displays met behulp van de **SSD1963-controller** als rechts afgebeeld.

Deze gebruiken een parallelle interface, zijn verkrijgbaar in maten van **4,3"** tot **9"** en hebben betere specificaties dan de kleinere toont. Al deze displays hebben een SD kaartsocket die volledig wordt ondersteund door MMBasic op de PicoMite.

Op eBay kun je geschikte displays vinden door te zoeken naar de naam van de **controller SSD1963.**

Omdat de **SSD1963-controller** een **parallelle interface** de PicoMite kan overdracht van gegevens veel sneller dan een SPI interface wat resulteert in een zeer snel scherm bijwerken. Deze schermen zijn ook veel groter, hebben meer pixels en zijn helderder. MMBasic stuurt ze aan met 24- bit true colour voor een volledige kleurweergave (**16 miljoen kleuren**).

De kenmerken van deze displays zijn:

- Een **4.3, 5, 7, 8** of **9 inch** display.
- Resolutie van **480 x 272** pixels (**4.3"** versie) of **800 x 480** pixels (**5”, 7", 8"** of **9"** versies).
- Een **SSD1963**-displaycontroller met een parallelle interface (**8080** formaat).
- Een aanraakcontroller (**SPI-interface**).
- Een **SD-kaartsleuf** van volledige grootte.

Er zijn een aantal verschillende ontwerpen die de SSD1963-controller gebruiken maar gelukkig hebben de meeste Chinese leveranciers gestandaardiseerd op een enkele connector zoals rechts afgebeeld. Het wordt sterk aanbevolen dat elk gekocht display heeft een bijpassende connector - dit biedt wat vertrouwen dat de fabrikant de norm heeft gevolgd die de PicoMite is ontworpen om te gebruiken.

Een op SSD1963 gebaseerd LCD-paneel aansluiten

De SSD1963-controller gebruikt een parallelle interface terwijl de touch controller en SD-kaart gebruiken een SPI-interface. De aanraking en SD-kaart functies zijn optioneel, maar als ze worden gebruikt, gebruiken ze de tweede SPI poort (SPI2).
In de volgende tabel staan de benodigde verbindingen tussen het display board en de PicoMite ter ondersteuning van de SSD1963-interface en de LCD Scherm. De aanraakcontroller en SD-kaart-interfaces worden verder vermeld onderstaand.

SSD1963 Display	Beschrijving	PicoMite
DB0	Parallelle databus bit 0	Pin 1/GP0
DB1	Parallelle databus bit 1	Pin 2/GP1
DB2	Parallelle databus bit 2	Pin 4/GP2
DB3	Parallelle databus bit 3	Pin 5/GP3
DB4	Parallelle databus bit 4	Pin 6/GP4
DB5	Parallelle databus bit 5	Pin 7/GP5
DB6	Parallelle databus bit 6	Pin 9/GP6
DB7	Parallelle databus bit 7	Pin 10/GP7
CS	Chip Select (actief laag)	Aarde (dwz altijd sel)
WR	Schrijven (actief laag)	Pin 19/GP14
RD	Lezen (actief laag)	Pin 20/GP15
RS	Command/Data	Pin 17/GP13
RESET	Reset de SSD1963	Pin 21/GP16
LED_A	Achtergrondverlichting voor een ongewijzigd weergavepaneel.	Configureerbaar
5V 5V	Voeding voor de achtergrond verlichting op sommige schermen (de meeste schermen gebruiken hiervoor de 3,3V voeding).	
3.3V	Voeding.	
GND	Aarde	

De volgende tabel geeft een overzicht van de aansluitingen die nodig zijn om de interface van de touch-controller te ondersteunen:

SSD1963 Display	Beschrijving	PicoMite
T_CS	Touch Chip Select	Aanbevolen Pin 24/GP18
T_IRQ	Touch Interrupt	Aanbevolen Pin 25/GP19
T_DIN	Touch Data In (MOSI)	Aanbevolen Pin 15/GP11
T_CLK	Touch SPI Clock	Aanbevolen Pin 14/GP10
T_DO	Touch Data Out (MISO)	Aanbevolen Pin 16/GP12

De volgende tabel geeft een overzicht van de aansluitingen die nodig zijn om de SD-kaartconnector te ondersteunen:

SSD1963 Display	Beschrijving	PicoMite
SD_CS	SD-kaart Chip Selecteer	Aanbevolen Pin 29/GP22
SD_DIN	SD-kaartgegevens in (MOSI)	Aanbevolen pin 15/GP11
SD_CLK	SD-kaart SPI-klok	Aanbevolen pin 14/GP10
SD_DO	SD-kaartgegevens uit (MISO)	Aanbevolen pin 16/GP12

Waar een PicoMite-verbinding wordt vermeld als **"Aanbevolen"**, moet de specifieke pin worden gespecificeerd in de juiste **OPTION** commando.

Over het algemeen hebben 7-inch en grotere schermen een aparte pin op de connector (gemarkeerd met 5V) voor het voeden van de achtergrondverlichting van een 5V voeding. Als deze pin niet is voorzien, wordt de achtergrondverlichting van de 3.3V-pin gehaald. Merk op dat de stroomverbruik door de achtergrondverlichting kan aanzienlijk zijn. Een 7-inch scherm trekt bijvoorbeeld doorgaans 330 mA van de 5V-pin.
De stroom die door de achtergrondverlichting wordt getrokken, kan een spanningsval op de aardingspin van het LCD-scherm veroorzaken, wat kan leiden tot draai verschuif de logische niveaus zoals gezien door de displaycontroller, wat resulteert in beschadigde kleuren of tekst. Een gemakkelijke manier om de diagnose van dit effect is om de CPU-snelheid te verlagen tot (laten we zeggen) 48 MHz. Als dit het probleem oplost, is het een sterke indicatie dat dit de oorzaak is. Het solderen van stroom- en aardingsdraden rechtstreeks naar de PCB van het LCD-scherm is één tijdelijke oplossing.
Wees voorzichtig met beeldschermen die de SPI-poort delen tussen een aantal apparaten (SD-kaart, touch, enz.).
In dit geval moeten alle Chip Select-signalen geconfigureerd zijn in MMBasic of uitgeschakeld worden door een permanente verbinding met 3.3V. Als dit niet wordt gedaan, zal de pin zweven waardoor de verkeerde controller reageert op opdrachten op de SPI-bus.
Op de PicoMite kan elk SPI-kanaal worden gebruikt om te communiceren met de aanraakcontroller en de SD-kaartinterface zoals gedefinieerd door de **OPTION SYSTEM SPI**-instelling. Als dit is ingesteld, is dat SPI-kanaal niet beschikbaar voor BASIC programma's (die het andere SPI-kanaal kunnen gebruiken).

Een op SSD1963 gebaseerd LCD-paneel configureren

Om het display te gebruiken, moet MMBasic worden geconfigureerd met behulp van het **OPTION LCDPANEL** commando, dat normaal gesproken is ingevoerd bij de opdrachtprompt. Elke keer dat de PicoMite opnieuw wordt opgestart, zal MMBasic automatisch de Scherm.
De syntaxis is:

```
OPTION LCDPANEL controller, oriëntation
```

Waar:
'controller' kan zijn:

- **SSD1963_4** Voor een 4,3-inch scherm.
- **SSD1963_5** Voor een 5 inch beeldscherm.
- **SSD1963_5A** Voor een alternatieve versie van het 5 inch beeldscherm als SSD1963_5 niet werkt.
- **SSD1963_7** Voor een 7 inch beeldscherm.
- **SSD1963_7A** Voor een alternatieve versie van het 7 inch beeldscherm als SSD1963_7 niet werkt.
- **SSD1963_8** Voor 8 inch of 9 inch beeldschermen.

'oriëntatie' kan **LANDSCAPE, PORTRAIT, RLANDSCAPE** of **RPORTRAIT** zijn.
Deze kunnen worden afgekort tot: **L, P, RL** of **RP.** Het voorvoegsel R geeft de omgekeerde of "ondersteboven" oriëntatie aan.
Deze opdracht hoeft maar één keer te worden uitgevoerd. Vanaf dat moment zal MMBasic het display automatisch initialiseren op opstarten of resetten. In sommige omstandigheden kan het nodig zijn om de stroom naar het LCD-paneel te onderbreken terwijl de PicoMite is actief (bijv. om de batterij te sparen) en in dat geval kan het **GUI RESET LCDPANEL** commando worden gebruikt om het display opnieuw te initialiseren.

Als het LCD-paneel niet langer nodig is, kan het commando **OPTION LCDPANEL DISABLE** worden gebruikt, waardoor retourneer de I/O-pinnen voor algemeen gebruik.

Om de configuratie te verifiëren, kunt u het commando **OPTION LIST** gebruiken om alle opties weer te geven die zijn ingesteld inclusief de configuratie van het LCD-paneel.

Om het display te testen kunt u het commando **GUI TEST LCDPANEL** invoeren. U zou een geanimeerde weergave moeten zien van kleurcirkels worden snel over elkaar heen getrokken. Druk op de spatiebalk op het toetsenbord van de console voor de test.

8 en 9 inch beeldschermen

De controllerconfiguratie SSD1963_8 is alleen getest met de 8 en 9 inch schermen van EastRising (beschikbaar op www.buydisplay.com). Deze moeten worden gekocht als een TFT LCD-paneel met 8080-interface, 800x480 pixel LCD, SSD1963-displaycontroller en XPT2046-touchcontroller. Merk op dat de EastRising-panelen een niet-standaard interface connector pin-out, dus u zult hun gegevensbladen moeten raadplegen wanneer u deze aansluit op de PicoMite. Een geschikte adapter om te converteren naar de standaard 40-pins aansluiting is te koop bij:
https://www.rictech.nz/micromite-products

Achtergrondverlichting

Voor de **ILI9163, ILI9341, ST7735, ST7735S, SSD1331, ST7789, ILI9481, ILI9488, ILI9488W, ST7789_135** en **ST7789_320** geeft een optionele parameter weer ', achtergrondverlichting' kan worden toegevoegd aan het einde van de configuratieparameters die een pin specificeert die moet worden gebruikt om de helderheid van de achtergrondverlichting te regelen. Dit zal een PWM-uitgang instellen op die pin met een frequentie van 1 KHz en een initiële duty cycle van 99%.
U kunt dan het BACKLIGHT-commando gebruiken om de helderheid te wijzigen tussen
0 en 100%. Het PWM-kanaal is geblokkeerd voor normaal PWM-gebruik en mag niet conflicteren met het PWM-kanaal dat mogelijk is ingesteld voor audio.
Bijvoorbeeld:

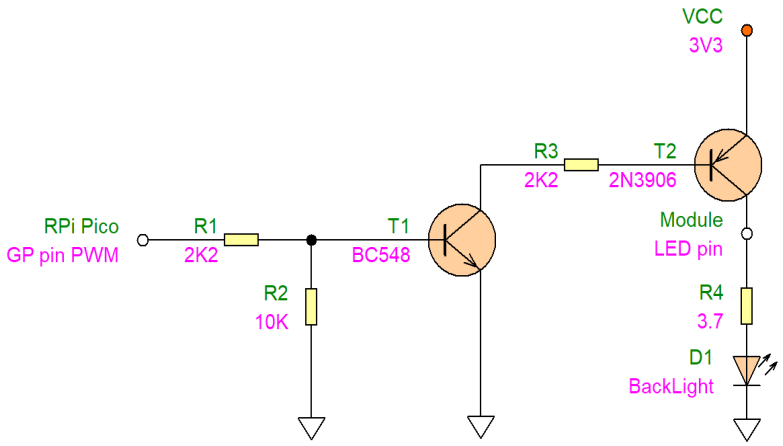
```
OPTION LCDPANEL ILI9341, OR, DC, RESET, CS, GP11
```

De achtergrondverlichting kan dan worden ingesteld op 40% met dit commando:

```
BACKLIGHT 40
```

De meeste op **SSD1963** gebaseerde LCD-panelen hebben drie paar soldeerpads op de print die zijn gegroepeerd onder de kopje **"Achtergrondverlichting"**. Normaal gesproken is het paar gemarkeerd met **"LED-A"** aan elkaar kortgesloten met een weerstand van **nul** ohm en dit maakt controle van de helderheid van de achtergrondverlichting met een **PWM** (pulsbreedte) gemoduleerd) signaal op de **LED-A** pin van het display pin van het display hoofdconnector van het paneel.

```
ILI9341 pinnen
-  1  VCC(3V3)  |
-  2  GND       T
-  3  CS        F
-  4  RESET     T
-  5  D/C       L
-  6  MOSI/DI   C
-  7  CLK/SCK   D
-  8  LED       |
-  9  MISO/DO   |
-  10 T_CLK/SCK T
-  11 T_CS      O
-  12 T_MOSI/DI U
-  13 T_MISO/DO C
-  14 T_IRQ     H
```



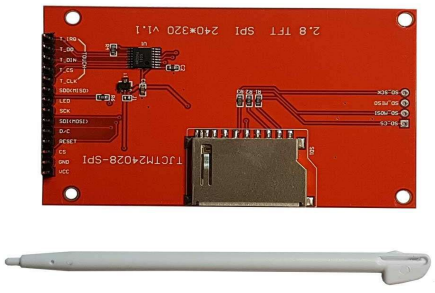
Schema voor ILI9341 versie 1.1 backlight voor waar geen driver op de module aanwezig is.

Het is echter beter om de SSD1963-controller te gebruiken om genereer dit signaal omdat het één I/O-pin vrijmaakt. om de te **gebruiken SSD1963** voor helderheidsregeling van de weerstand van nul ohm moet worden verwijderd uit het paar gemarkeerd met **"LED-A"** en gebruikt om het nabij-gelegen paar soldeerpads met de aanduiding **"1963-PWM"** kort te sluiten. De PicoMite kan dan de helderheid regelen via de **SSD1963-controller** met behulp van het **BACKLIGHT-commando**.

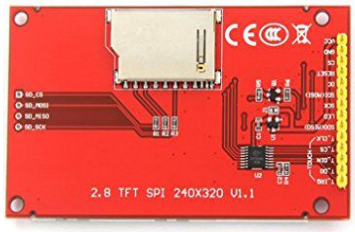
Voorbeeld LCD-paneelconfiguratie

Hieronder volgt een samenvatting van hoe een typisch **LCD-paneel** met een **ILI9341-controller** kan worden aangesloten. Deze voorbeeld ondersteunt de **SD-kaartaansluiting**, het LCD-scherm en de **aanraakinterface**.

Typische panelen zijn te vinden op ebay.com en vergelijkbare sites door te zoeken op het trefwoord **"ILI9341"**. Zorg ervoor dat dat de aansluitingen aan de achterkant van het paneel lijken op die hier onder:



ILI9341 zonder LED driver
IC is de touch controller



ILI9341 met LED driver
IC is de touch controller

MAKER PI PICO			PicoMite Help
VCC	=	3V3 *1	3V3 bij J1=dicht en 5V
GND	=	GND	GND bij J1=open.
CS	=	GP6	GP13(Zie datablad)
RESET	=	GP7	GP14
DC	=	GP8	GP15
MOSI/DI	=	GP11SPI	GP19 DO
SCK/CLK	=	GP10SPI	GP18 CLK
LED 56Ω	=	3V3 GP26	3V3 GP26/GP20/GP10=BL
MISO/DO	=	nc SPI	nc Zie opmerking
T T-CLK	=	GP10SPI	GP18 CLK
O T-CS	=	GP9	GP12
U T-DIN	=	GP11SPI	GP19 DO
C T-DO	=	GP12SPI	GP16 DI
H T-IRQ	=	GP18	GP11
SD-CLK	=	GP10SPI	GP18 CLK
SD-DOUT	=	GP12SPI	GP16 DI
SD-DIN	=	GP11SPI	GP19 DO
SD-D1	=	GP13	nc
SD-D2	=	GP14	nc
SD-CS	=	GP15	GP22

*1= J1 op achterkant VCC=3V3=DICHT 5V=OPEN

COM1:RX=GP17 TX=GP16	COM1:RX=GP17 TX=GP12
COM2:RX=GP21 TX=GP20	COM2:RX=GP09 TX=GP08
I2C:SDA=GP04 SCL=GP05	I2C:SDA=GP04 SCL=GP05

Voorbeeld aansluitingen.

Opmerking: Bij **ILI9341** scherm versie **1.1** is geen **LED driver** aanwezig en kan dan met weerstand of extra schakeling helderheid regelen zie schema hierboven. Bij versie **1.2** is wel een **LED driver** aanwezig door middel van **Q1, R5, R6** en kan direct met optie backlight worden aangestuurd op **LED** pin.

Om de bovenstaande verbindingen overeen te laten komen, moeten de volgende configuratie commando's worden ingevoerd, één voor één op de opdrachtprompt:

```
OPTION SYSTEM SPI GP18,GP19,GP16                ' CLK=GP18,MOSI=GP19,MISO=GP16
OPTION SDCARD GP22          ' CS=GP22,OR=L,DC=GP15,RESET=GP14,CS=GP13,BackLight
OPTION LCDPANEL ILI9341,L,GP15,GP14,GP13        ' GP26 of GP10 BackLight
OPTION TOUCH GP12,GP11                          ' CS=GP12,IRQ=GP11,Beep
```

Deze commando's worden onthouden en automatisch toegepast bij het opstarten. Merk op dat na elk commando dat is ingevoerd, wordt de PicoMite opnieuw opgestart en wordt de USB-verbinding verbroken en moet deze opnieuw worden aangesloten.

Vervolgens moet het aanraakscherm worden gekalibreerd:

```
GUI CALIBRATE
```

Vervolgens kunt u de verschillende onderdelen testen. Het volgende zal de bestanden op de SD-kaart weergeven, als het wordt uitgevoerd zonder fouten kunt u er zeker van zijn dat de SD kaart-interface goed is.

```
FILES
```

Het volgende zal meerdere kleurrijke overlappende cirkels op het LCD-scherm tekenen, wat zal bevestigen dat de LCD is correct aangesloten:

```
GUI TEST LCDPANEL
```

Ten slotte zal het volgende de aanraakinterface testen. Wanneer u het LCD-scherm aanraakt, verschijnt er een stip op de scherm op het exacte punt van aanraking.

```
GUI TEST TOUCH
```

Als dit niet juist is, moet u wellicht het **GUI CALIBRATE** commando een tweede keer uitvoeren, waarbij u voorzichtiger moet zijn.
Als u problemen ondervindt om het scherm te laten werken, is het de moeite waard om alles los te koppelen en de opties te wissen met het commando **OPTION CLEAR** zodat je met een schone lei kunt beginnen. Sluit het dan één stap tegelijk weer aan tijd en configureer en test elke nieuwe fase naarmate je vordert. Eerst de SD-kaartinterface, dan het LCD-scherm en tot slot de aanraakinterface.
Merk ook op dat de ILI9341-controller gevoelig is voor statische ontlading, dus als het paneel niet reageert, kan dit zijn beschadigd en het zou de moeite waard zijn om met een ander paneel te testen.

Aanraakschermen

Veel LCD-panelen worden geleverd met een resistief aanraakgevoelig paneel en bijbehorende controllerchip. MMBasic ondersteunt deze interface op de PicoMite volledig en hierdoor kunnen veel van de fysieke knoppen en schakelaars die in een project dat moet worden geïmplementeerd als bedieningselementen op het scherm die door aanraking worden geactiveerd.
Let daar op:

- De maximale spanning op alle PicoMite I/O-pinnen is **3V3**. Niveauverschuiving is vereist als een display gebruikt **5V**-niveaus voor signalering.
- Ondersteuning voor lcd-scherm en aanraakinvoer is niet beschikbaar op de PicoMiteVGA-versie.

Aanraking configureren

De aanraakcontroller op een LCD-paneel gebruikt het SPI-protocol voor communicatie en dit moet specifiek geconfigureerd voordat het paneel kan worden geconfigureerd. Dit is de "systeem" SPI-poort, de poort die wordt gebruikt voor systeemgebruik (SD-kaart, LCD-scherm en de aanraak-controller op een LCD-paneel). Deze SPI-poort is dan niet beschikbaar voor BASIC-programma's (d.w.z. het is gereserveerd).

Er zijn een aantal poorten en pinnen die kunnen worden gebruikt, maar deze zijn hetzelfde als de configuratie die wordt gebruikt voor de voorbeeld LCD-paneelinterface eerder in deze handleiding. Deze opdracht hoeft niet herhaald te worden als de systeem-SPI is al geconfigureerd:

```
OPTION SYSTEM SPI GP18, GP19, GP16    ' CLK, MOSI, MISO
```

Om de aanraakfunctie te gebruiken, moet MMBasic worden verteld dat deze beschikbaar is met behulp van het **OPTION TOUCH** commando. Deze moet worden gedaan nadat het LCD-scherm is geconfigureerd. Deze opdracht vertelt MMBasic waarvoor pinnen worden gebruikt de Chip Select- en Interrupt-signalen. Dit stelt bijvoorbeeld Chip Select in op de GP12-pin en Interrupt op GP11:

```
OPTION TOUCH GP12, GP11                ' CS, IRQ
```

Deze opdrachten moeten worden ingevoerd bij de opdrachtprompt en zorgen ervoor dat de Pico-Mite opnieuw wordt opgestart. Dit heeft de neveneffect van het loskoppelen van de USB-console-interface die opnieuw moet worden aangesloten. Wanneer de PicoMite opnieuw wordt opgestart, zal MMBasic de aanraakcontroller automatisch initialiseren. Om de configuratie te verifiëren kunt u het commando **OPTION LIST** gebruiken om alle opties weer te geven die zijn ingesteld, inclusief de configuratie van het displaypaneel en raak aan. Merk op dat u veel verschillende configuraties kunt gebruiken met verschillende pintoewijzingen - dit is slechts een voorbeeld op basis van de hierboven vermelde configuratieopdrachten. Wees voorzichtig wanneer de SPI-poort wordt ge-deeld tussen een aantal apparaten (SD-kaart, touch, enz.). In dit geval alle de Chip Select-signalen moeten geconfigureerd zijn in MMBasic of anders uitgeschakeld zijn.

Het touchscreen kalibreren

Voordat de aanraakfunctie kan worden gebruikt, moet deze worden gekalibreerd met het commando

```
GUI CALIBRATE   0, 262, 327, 929, 702
```
Deze staat in de option list.

De waarden zijn: **negeren, x nul, y nul, x helling, y helling**. De option lijst kopieeren en opslaan X en y zijn afhankelijk van oriëntatie en dan zijn het eenvoudige lineaire vergelijkingen. Met deze opdracht wordt een doel in de linkerbovenhoek van het scherm weergegeven. Een puntig maar stomp voorwerp gebruiken (zoals een tandenstoker) druk precies op het midden van het doel en houd het minstens een seconde ingedrukt. MMBasic wil noteer deze locatie en ga verder met de kalibratie door het doel achtereenvolgens weer te geven in de andere drie hoeken van het scherm voor aanraking en kalibratie.De kalibratieroutine kan waarschuwen dat de kalibratie niet nauw-keurig was. Dit is slechts een waarschuwing en u kunt nog steeds gebruik desgewenst de aanraak-functie, maar het is beter om de kalibratie met meer zorg te herhalen.
Na de kalibratie kunt u de aanraakfunctie testen met het **GUI TEST TOUCH** commando. Dit com-mando zal het scherm leegmaken en wachten op een aanraking. Wanneer het scherm wordt aan-geraakt, wordt er een witte stip op het display geplaatst het markeren van de positie op het scherm. Als de kalibratie met succes is uitgevoerd, moet de stip worden weergegeven precies onder de plaats van de stylus op het scherm. Om de testroutine te verlaten, kunt u op de spatie-balk op het toetsenbord van de console drukken.

Aanraakfuncties

Om te detecteren of en waar het scherm wordt aangeraakt, kun je de volgende functies gebruiken in een BASIC-programma:

- **TOUCH(X)**
  Retourneert de X-coördinaat van de momenteel aangeraakte locatie of -1 als het scherm niet wordt aangeraakt.

- **TOUCH(Y)**
  Retourneert de Y-coördinaat van de momenteel aangeraakte locatie of -1 als het scherm niet wordt aangeraakt.

- **TOUCH(DOWN)**
  Retourneert waar als het scherm momenteel wordt aangeraakt.
  (dit is veel sneller dan TOUCH(X of Y)).

- **TOUCH(UP)**
  Retourneert waar als het scherm momenteel NIET wordt aangeraakt
  (ook sneller dan TOUCH(X of Y))

- **TOUCH(LASTX)**
  Retourneert de X-coördinaat van de laatste locatie die is aangeraakt.

- **TOUCH(LASTY)**
  Retourneert de Y-coördinaat van de laatste locatie die is aangeraakt.

- **TOUCH(REF)**
  Retourneert het referentienummer van het besturingselement dat momenteel wordt aangeraakt of nul als er geen besturingselement wordt gebruikt aangeraakt.
  Zie de sectie Geavanceerde grafische afbeeldingen voor meer details.

- **TOUCH(LASTREF)**
  Retourneert het referentienummer van het besturingselement dat het laatst is aangeraakt.

Het GUI BEEP commando

De piëzo-zoemer gespecificeerd in het **OPTION TOUCH** commando kan ook worden aangestuurd door een BASIC-programma met behulp van het bevel:

```
GUI BEEP msec
```

Waarbij **'msec'** het aantal **milliseconden is dat de pieper** moet worden **aangestuurd**. Een tijd van **3 ms produceert een klik** terwijl **100ms een korte pieptoon produceert**.

Touch Interrupts zonder geavanceerde GUI-bediening

Er kan een **interrupt** worden ingesteld op het **IRQ-pinnummer** dat is opgegeven bij het configureren van de aanraakfunctie. Naar detect touch down de interrupt moet worden geconfigureerd als INTL (d.w.z. van hoog naar laag).

Als bijvoorbeeld het commando: `OPTION TOUCH 7, 15`
Werd gebruikt om touch te configureren, zal het volgende programma:
Print de X- en Y-coördinaten van elke aanraking op het scherm.

```
SETPIN 15, INTL, MyInt
DO : LOOP

SUB MyInt
  PRINT TOUCH(X) TOUCH(Y)
END SUB
```

De interrupt kan worden geannuleerd met het commando: `SETPIN pin, OFF`.

Raak Onderbrekingen aan met geavanceerde GUI bediening

Wanneer de geavanceerde GUI-bedieningselementen zijn geactiveerd.
(door het aantal `GUI` bedieningselementen in te stellen op een getal dat niet nul is) met `OPTION GUI` **CONTROLS**) wordt in plaats daarvan de `GUI INTERRUPT` opdracht gebruikt om een aanraak-onderbreking in te stellen.
De syntaxis is:

```
GUI INTERRUPT down [, up]
```

Waar **'down'** de **subroutine** is die moet worden opgeroepen wanneer een touchdown is gedetecteerd. En optioneel **'omhoog'** is de **subroutine** om te bellen wanneer de aanraking van het scherm is gehaald (**'up'** en **'down'** kunnen naar dezelfde subroutine indien nodig).

Als voorbeeld zal het volgende programma de **X**- en **Y**-coördinaten van elke aanraking op het scherm afdrukken:

```
GUI INTERRUPT MyInt
DO : LOOP

SUB MyInt
  PRINT TOUCH(X) TOUCH(Y)
END SUB
```

Door het getal nul (één cijfer) als argument op te geven, worden zowel opwaartse als neerwaartse onderbrekingen geannuleerd. d.w.z:

```
GUI INTERRUPT 0
```

Grafische opdrachten en functies

Ondersteunde hardware

LCD-panelen

De resolutie en het aantal kleuren dat door een LCD-paneel wordt ondersteund, worden bepaald door het paneel zelf en de driver. Zie het hoofdstuk Beeldschermpanelen voor meer informatie.

VGA-video (RP2040 en RP2350)

Er zijn een aantal modi die kunnen worden geselecteerd met behulp van het MODE commando:
MODE **1 640x480 zwart-wit** met **RGB121**-tegels.
MODE **2 320x240 4-bit kleur**
MODE **3 640x480 4-bit kleur** (alleen RP2350)

HDMI-video (alleen RP2350)

Elke **HDMI** resolutie kan in een aantal modi werken die worden ingesteld met behulp van het MODE commando:

OPTION **RESOLUTION 640x480**

MODE 1	640x480x	2	**kleuren** met **RGB555**-tegels (gebruik de opdracht TILE zoals normaal)
MODE 2	320x240x	16	**kleuren** en kleurtoewijzing naar **RGB555**-tegels.
MODE 3	640x480x	16	**kleuren** en kleurtoewijzing naar **RGB555**-tegels.
MODE 4	320x240x	32768	**kleuren**
MODE 5	320x240x	256	**kleuren** en kleurtoewijzing naar **RGB555**-tegels.

OPTION **RESOLUTION 1280x720**

MODE 1	1280x720x	2	**kleuren** met **RGB332**-tegels (gebruik de opdracht TILE zoals normaal)
MODE 2	320x180x	16	**kleuren** en kleurtoewijzing naar **RGB332**-tegels.
MODE 3	640x360x	16	**kleuren** en kleurtoewijzing naar **RGB332**-tegels.
MODE 5	320x180x	256	**kleuren**

OPTION **RESOLUTION 1024x768**

MODE 1	1024x768x	2	**kleuren** met **RGB332**-tegels (gebruik de opdracht TILE zoals normaal)
MODE 2	256x192x	16	**kleuren** en kleurtoewijzing naar **RGB332**-tegels.
MODE 3	512x384x	16	**kleuren** en kleurtoewijzing naar **RGB332**-tegels.
MODE 5	256x192x	256	**kleuren**

Kleuren

Kleur wordt gespecificeerd als een 24-bits nummer met ware kleuren, waarbij de bovenste acht bits de intensiteit van het rood vertegenwoordigen kleur, de middelste acht bits de groene intensiteit en de onderste acht bits het blauw. De gemakkelijkste manier om te genereren dit nummer is met de RGB()-functie die de vorm heeft :

```
RGB(red, green, blue)
```

De functie **RGB()** ondersteunt ook een snelkoppeling waarmee u algemene kleuren kunt specificeren door ze een naam te geven. Voor bijvoorbeeld `RGB(red)` of `RGB(cyan)`.
De kleuren die kunnen worden benoemd met behulp van het snelkoppelingsformulier zijn:
wit, zwart, blauw, groen, cyaan, rood, bruin, magenta, geel, oranje, roze, goud, zalm, beige, lichtgrijs en grijs. (of USA spelling grijs/lichtgrijs). Engelse namen invoeren.
Deze namen zijn geldig bij RGB(): white, black, blue, green, cyan, red, brown, magenta, yellow, orange, pink, gold, salmon, beige, light gray and gray.
MMBasic vertaalt automatisch alle kleuren naar het formaat dat vereist is voor de individuele displaycontroller.

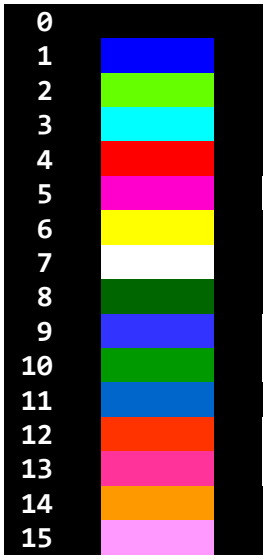
In het geval van de ILI9341-controller bijvoorbeeld, zijn 64K kleuren in het 565-formaat.
De standaard voor commando's die een kleurparameter vereisen, kan worden ingesteld met het **COLOR** commando (kan ook zijn gespelde **COLOR**). Dit is handig als je programma een consistent kleurenschema gebruikt, je kunt dan de standaardinstellingen instellen en gebruik de korte versie van de tekenopdrachten in uw hele programma.
Het COLOR commando heeft het formaat :

```
COLOUR foreground colour, background colour
```

De PicoMiteVGA-versie ondersteunt 2 "kleuren" in monochrome modus en 16 in kleurmodus. in zwart-wit modus zal elke kleur die geen nul is, resulteren in een verlichte pixel. In de kleurmodus wordt de **RGB** functie op één toegewezen van de 16 beschikbare kleuren.

Kleur No.	Kleur Naam	RGB Decimaal	RGB Waarde	Kleur
0	Black	0, 0, 0	000000	
1	Blue	0, 0,255	0000FF	
2	Green	0,255, 0	00FF00	
3	Cyan	0,255,255	00FFFF	
4	Red	255, 0, 0	FF0000	
5	Magenta	255, 0,255	FF00FF	
6	Yellow	255,255, 0	FFFF00	
7	White	255,255,255	FFFFFF	
8	Myrtle	0, 64, 0	004000	
9	Cobelt	0, 64,255	0040FF	
10	MidGreen	0,128, 0	008000	
11	Cerulean	0,128,255	0080FF	
12	Rust	255, 64, 0	FF4000	
13	Fuchsia	255, 64,255	FF40FF	
14	Brown	255,128, 0	FF8000	
15	Lilac	255,128,255	FF80FF	

Kleuren tabel



Kleuren tabel

Letter typen

Er zijn zeven ingebouwde lettertypen. Dit zijn:

Lettertype Nummer	Maatvoering (breedte x hoogte)	Karakter Set	Beschrijving instellen
1	**8 x 12**	Alle 95 ASCII-tekens plus 7F tot FF (hex)	Standaardlettertype (standaard bij opstarten)
2	**12 x 20**	Alle 95 ASCII-tekens	Middelgroot lettertype.
3	**16 x 16**	Alle 95 ASCII-tekens	Een groter lettertype voor VGA versies.
3	**16 x 24**	Alle 95 ASCII-tekens	Een groter lettertype voor HDMI versies en LCD panelen.
4	**10 x 16**	Alle 95 ASCII-tekens plus 7F tot FF (hex)	Een lettertype met uitgebreide grafische tekens. Geschikt voor schermen met hoge resolutie.
5	**24 x 32**	Alle 95 ASCII-tekens	Extra groot lettertype, zeer duidelijk.
6	**32 x 50**	0 tot 9 plus wat symbolen	Getallen plus decimale punt, positief, negatief, is gelijk aan, graden en dubbele punt symbolen. Heel duidelijk.
7	**6 x 8**	Alle 95 ASCII-tekens	Een klein lettertype dat handig is als lage resoluties worden gebruikt.
8	**4 x 6**	Alle 95 ASCII-tekens	Een nog kleiner lettertype.

In alle lettertypen (inclusief lettertype **#6**) is het achterste aanhalingsteken (**60 hex** of **96 decimaal**) is vervangen door het **gradensymbool** (**°**).
Lettertype **#1** (het standaardlettertype) en lettertype **#4** hebben een uitgebreide tekenset voor alle tekens van **CHR$(32)** tot **CHR$(255)** of **20** tot **FF (hex)**.
Indien nodig kunnen extra lettertypen worden ingesloten in een BASIC programma. Deze letter-typen werken precies hetzelfde als het ingebouwde lettertype.
(d.w.z. geselecteerd met het **FONT** commando of gespecificeerd in de **TEXT** opdracht).

De indeling van een ingesloten lettertype is:

```
DefineFont #Nbr
    hex [[ hex[…]
    hex [[ hex[…]
END DefineFont
```

Het moet beginnen met het trefwoord "DefineFont" gevolgd door het lettertypenummer (dat kan worden voorafgegaan door een optioneel **#** teken).

Elk lettertypenummer in het bereik van 2 tot 5 en 8 tot 16 kan worden opgegeven en als het hetzelfde is als een ingebouwd lettertype zal het dat letter-type vervangen. De hoofdtekst van het lettertype is een reeks van 8-cijferige hexadecimale woorden met elk woord gescheiden door een of meer spaties of een nieuwe regel. De lettertype-definitie wordt beëindigd door een **"End DefineFont"** trefwoord. Deze kunnen overal in een programma worden geplaatst en MMBasic zal het overslaan. Dit formaat is hetzelfde zoals die wordt gebruikt door de Micromite.

Aanvullende lettertypen en informatie zijn te vinden in de map **Embedded Fonts** in de PicoMite-firmware downloaden. Deze lettertypen dekken een breed scala aan tekensets, waaronder een sym-boollettertype (**Dingbats**), wat handig is voor het maken van pictogrammen op het scherm, enz.

Alleen lezen variabelen.

Alle coördinaten en metingen op het scherm zijn gedaan in termen van pixels, waarbij de **X-coördinaat** de is **horizontale positie** en **Y** de **verticale positie**. De **linkerbovenhoek** van het scherm heeft de **coördinaten X=0 en Y=0** en de waarden nemen toe naarmate u naar beneden en naar rechts van het scherm beweegt.

Er zijn vier alleen-lezen variabelen die nuttige informatie geven over het momenteel aangesloten display:

- `MM.HRES`
  Retourneert de breedte van het scherm (de X-as) in pixels.
- `MM.VRES`
  Retourneert de hoogte van het scherm (de Y-as) in pixels.
- `MM.FONTHEIGHT`
  Retourneert de hoogte van het huidige standaardlettertype (in pixels). Alle tekens in een lettertype hebben dezelfde hoogte.
- `MM.FONTWIDTH`
  Retourneert de breedte van een teken in het huidige lettertype (in pixels). Alle karakters hebben dezelfde breedte.

Tekenopdrachten

Er zijn negen basis tekenopdrachten die u kunt gebruiken in MMBasic-programma's op de PicoMite om interactie met een aangesloten LCD-scherm. Er is ook een reeks krachtigere GUI-commando's voor tekenen schakelaars, keuzerondjes, enz. Zie de volgende sectie Geavanceerde grafische afbeel-dingen voor meer details.
De meeste basistekenopdrachten hebben optionele parameters. Je kunt deze helemaal laten staan aan het einde van een commando of u kunt twee komma's achter elkaar gebruiken om een ontbrekende parameter aan te geven. Bijvoorbeeld de vijfde parameter van het **LINE** commando is optioneel, dus u kunt dit formaat gebruiken:

```
LINE 0, 0, 100, 100, , rgb(red)     ' Let op de twee comma's
```

Optionele parameters worden hieronder cursief weergegeven, bijvoorbeeld: *font*
In de volgende opdrachten is C de kleur van de tekening en standaard ingesteld op de huidige voorgrondkleur. **FILL** is de vulkleur die standaard op **-1** staat, wat aangeeft dat er geen opvulling moet worden gebruikt.

De tekenopdrachten zijn:

- **CLS C**
  Wist het scherm naar de kleur C. Als C niet is opgegeven, wordt de huidige standaard achtergrondkleur gebruikt.

- **PIXEL X, Y, C**
  Verlicht een pixel. Als C niet is opgegeven, wordt de huidige standaard voorgrondkleur gebruikt.

- **LINE X1, Y1, X2, Y2, LW, C**
  Tekent een lijn die begint bij X1 en Y1 en eindigt bij X2 en Y2.
  LW is de lijnbreedte en is alleen geldig voor horizontale of verticale lijnen. Het staat standaard op **1** indien niet gespecificeerd of als de lijn is een diagonaal. **C** is lijnkleur

- **BOX X, Y, B, H, LW, C, FILL**
  Tekent een kader dat begint bij X en Y en dat B pixels breed en H pixels hoog is.
  LW is de breedte van de zijkanten van de **doos** en kan **nul zijn**. Het staat **standaard op 1**.
  C is lijnkleur en **FILL** is vulkleur.

- **RBOX X, Y, B, H, R, C, FILL**
  Tekent een kader met afgeronde hoeken, beginnend bij X en Y, dat B pixels breed en H pixels hoog is.
  R is de **straal** van de **hoeken van de doos**. Het staat **standaard op 10**. **C** is lijnkleur, **Fill** is vulkleur.

- **CIRCLE X, Y, R, LW, A, C, FILL**
  Tekent een cirkel met X en Y als middelpunt en een straal R. LW is de breedte van de lijn die wordt **gebruikt voor de omtrek en kan nul zijn** (**standaard ingesteld op 1**). **A** is de beeldverhouding die een getal met drijvende komma is en standaard **ingesteld op 1**. **Een aspect van 0,5 tekent bijvoorbeeld een ovaal** waarbij de **breedte de helft van de hoogte is**.

- **TEXT X, Y, STRING, ALIGNMENT, FONT, SCALE, C, BC**
  Geeft een tekenreeks weer die begint bij X en Y. ALIGNMENT is **0, 1** of **2** tekens (een tekenreeksuitdrukking of variabele is ook toegestaan) waarbij de eerste letter de horizontale uitlijning rond **X** is en **L, C** of **R** kan zijn voor **LINKS**, **CENTER** of **RECHTS** uitgelijnde tekst en de tweede letter is de verticale uitlijning rond **Y** en kan **T, M** zijn of **B** voor **BOVEN**, **MIDDEN** of **ONDER** uitgelijnde tekst. De **standaarduitlijning** is **links/boven**. Een extra code letter kan worden gebruikt om de **tekst te roteren** (zie hieronder voor de details).
  LETTERTYPE en **SCHAAL** zijn optioneel en standaard aan die ingesteld door het **FONT** commando. **C** is de tekenkleur en **BC** is de achtergrondkleur. Zij zijn optioneel en standaard ingesteld op de **COLOR** opdracht.

- **GUI BITMAP X, Y, BITS, WIDTH, HEIGHT, SCALE, C, BC**
  Toont de bits in een bitmap beginnend bij X en Y. HOOGTE en **BREEDTE** zijn de **afmetingen** van de **bitmap** zoals weergegeven op het LCD-scherm en standaard ingesteld op **8x8**. **SCALE**, **C** en **BC** zijn hetzelfde als voor de **TEXT** opdracht. De **bitmap** kan een **geheel getal** of een **stringvariabele** of **constante** zijn en wordt getekend met de **eerste byte** als de **eerste bits** van de bovenste regel **(bit 7 eerst, dan bit 6, enz**.) gevolgd door de volgende byte, enz.
  Als de bovenste regel is gevuld, begint de volgende regel van de weergegeven bitmap met het volgende bit in het gehele getal of de tekenreeks.

- **POLYGON n, xarray%(), yarray%() [, bordercolour] [, fillcolour]**
  Tekent een gevulde of omlijnde veelhoek met n xy-coördinaatparen in **xarray%()** en **yarray%().**
  Als **'vulkleur'** is weggelaten, dan wordt alleen de omtrek van de veelhoek getekend. Als **'borderkleur'** wordt weggelaten, wordt standaard de huidige standaard voorgrondkleur.

- **ARC x, y, r1, [r2], a1, a2 [, c]**
  Tekent een boog van een cirkel met een bepaalde kleur en breedte tussen twee radialen **(gedefinieerd in graden).**
  Parameters voor het **ARC** commando zijn de **x**- en **y**-coördinaten van het midden van de boog, de binnen- en buitenstralen, het begin en eindhoeken van de boog en de kleur van de boog. De nul graden referentie bevindt zich op de 12 uur-positie.

Geroteerde tekst

Zoals hierboven beschreven, kan de uitlijning van de tekst in het **TEXT** commando worden gespecificeerd met een of twee tekens in een tekenreeksexpressie voor de derde parameter van de opdracht. In deze string kun je ook a specificeren derde teken om de rotatie van de tekst aan te geven. Dit karakter kan een van de volgende zijn:

N	Voor normale oriëntatie
V	Voor verticale tekst waarbij elk teken onder het vorige van boven naar beneden loopt.
I	De tekst wordt omgekeerd (d.w.z. ondersteboven).
U	De tekst wordt 90º tegen de klok in gedraaid.
D	De tekst wordt 90º met de klok mee gedraaid.

Deze extra functie is van toepassing op de commando's **TEXT** en **GUI CAPTION**.
Als voorbeeld wordt in het volgende de tekst **"LCD-display"** verticaal in de linkermarge van de weergegeven displaypaneel en verticaal gecentreerd:

```
TEXT 0, 250, "LCD-display", "LMV", 5
```

Positionering is relatief ten opzichte van de linkerbovenhoek van het personage wanneer het normaal wordt bekeken, dus omgekeerd **100, 100** zal de pixel linksboven van het eerste teken op **100, 100** hebben en de tekst zal dan boven **y=101** staan en links van **x=101**. Op dezelfde manier wordt **"R"** in de uitlijningsreeks bekeken vanuit het perspectief van het teken in wat dan ook oriëntatie waarin het zich bevindt (niet het scherm).

Transparante tekst

De **VGA**- of **HDMI**-video-uitgang of **LCD**-schermen met behulp van de **SSD1963, ILI9341, ST7789_320** of **ILI9488** met **MISO** aangesloten zijn in staat tot transparante tekst.
In dit geval staat het **TEXT** commando het gebruik van **-1** toe voor de achtergrondkleur. Dit betekent dat de tekst is over de achtergrond getekend, waarbij de achtergrondafbeelding door de gaten in de letters zichtbaar zijn.

Framebuffers en lagen

Alle varianten van de firmware kunnen één of twee framebuffers in het geheugen en buffers met één of twee lagen creëren (dit is geheugen afhankelijk). Dit zijn geheugengebieden met dezelfde breedte en hoogte als het hoofddisplay. Bij **HDMI**- en **VGA**-schermen hebben deze dezelfde kleurdiepte als de huidige modus. In het geval van **LCD** beeldschermen hebben ze **4 bits** per pixel (**16 kleuren**).

Afhankelijk van de versie van de firmware en de huidige weergavemodus, creatie van framebuffers of laagbuffers zal vooraf toegewezen geheugen gebruiken of geheugen uit het gebruikersgeheugen toewijzen.
Framebuffers kunnen worden gebruikt om beeldgegevens samen te stellen die naar het fysieke beeldscherm kunnen worden gekopieerd. Laagbuffers zijn dat wel meestal gebruikt om gedeeltelijke afbeeldingen te maken die bovenop een achtergrondafbeelding kunnen worden geplaatst en een efficiënte methode om weergave-elementen over een statische achtergrond te verplaatsen Alle standaard grafische tekencommando's kunnen op dezelfde manier op een framebuffer of laagbuffer worden gebruikt schrijven naar het fysieke beeldscherm.
Het **FRAMEBUFFER WRITE** commando wordt gebruikt om de bestemming van het grafische uitvoer met behulp van een code.

De code bestaat uit één teken dat kan zijn:

N	Het fysieke uitvoerapparaat.
F	De framebuffer.
2	Een tweede framebuffer (alleen **RP2350**)
L	De lagenbuffer
T	Een tweede laagbuffer (alleen **RP2350**)

De basisframebuffer-opdrachten zijn:

```
FRAMEBUFFER CREATE  code F
FRAMEBUFFER LAYER   code L
FRAMEBUFFER CLOSE
FRAMEBUFFER WRITE   code
FRAMEBUFFER COPY    code1, code2 [,B]
```

Zie de gedetailleerde opdrachtbeschrijvingen voor aanvullende framebufferopdrachten.
In het geval van **VGA**- en **HDMI**-versies van de firmware, afhankelijk van de weergavemodus en CPU-snelheid (**>=252MHz**), worden er automatisch lagen toegepast bovenop het hoofdbeeldscherm wanneer dit naar het scherm wordt weergegeven.
In het geval van **LCD**-schermen wordt het **FRAMEBUFFER MERGE** commando gebruikt om het uiteindelijke beeld te creëren van een framebuffer en een laagbuffer. Het spel **PETSCII-robots** laat zien hoe deze techniek met veel succes kan worden gebruikt.
De automatische toepassing van een laagbuffer is geïmplementeerd in de **VGA**-versies **mode 2** en **mode 3** (**RP2350 alleen**) en **HDMI mode 2, 3, 4** en **5**. Buffers met twee lagen zijn alleen beschikbaar op de **RP2350** en in de volgende modi: **VGA-modus 2, HDMI-modus 2 en 5.**

BLIT- en Sprite commando's

In eerdere versies van de firmware waren de **Blit**- en **Sprite** commando's synoniemen voor dezelfde functionaliteit. In versie **6.00.01** en later zijn het afzonderlijke opdrachten. Het verschil is dat **BLIT** een eenvoudig geheugen is bewerking kopiëren van en naar een beeldscherm of geheugen naar een beeldscherm of geheugen. Sprites zijn complexer en laat de programmeur elementen over een achtergrond weergeven en ze vervolgens zonder achtergrond over de achtergrond verplaatsen het beschadigen van de achtergrondafbeelding. Bovendien kan de programmeur de sprite-functio-naliteit gebruiken om te detecteren botsingen tussen sprites en tussen een sprite en de randen van het scherm.

Sprites kunnen niet worden gebruikt tenzij het display het lezen van de framebuffer en de blitfunctionaliteit ook ondersteunt beperkt tenzij dit het geval is. **Sprites** zijn ingeschakeld voor alle versies van de firmware bij gebruik in een in-memory framebuffer en **VGA**- en **HDMI**-versies van de firmware rechtstreeks op het scherm.
Sprites worden altijd opgeslagen als **RGB121**-nibbles met 2 pixels per byte. **BLIT**-buffers worden daarentegen opgeslagen als **RGB888**-waarden en dergelijke kunnen worden gebruikt met **LCD**-kleurenschermen. Uiteraard gaat dit ten koste van aanzienlijk groter geheugengebruik.
Zie de opdracht en functie **SPRITE** samen met **bijlage G** voor meer informatie over het gebruik van sprites.
Als het display transparante tekst kan weergeven, staat het **BLIT** commando een deel van het beeld toe dat momenteel wordt weergegeven op het beeldscherm om naar een geheugenbuffer te kopiëren en later terug naar het beeldscherm te kopiëren. Dit is handig wanneer er moet iets over de achtergrond worden getekend en later worden verwijderd zonder de afbeelding in de afbeelding te beschadigen achtergrond. Voorbeelden zijn onder meer een spel waarbij een personage zich voor een landschap beweegt of beweegt de naald van een draaispoelmeter.

De beschikbare commando's zijn:

```
BLIT READ  #b, x,  y, w, h
BLIT WRITE #b, x,  y, w, h
BLIT LOAD  #b, f$, x, y, w, h
BLIT CLOSE #b
```

#b is het buffergetal in het bereik van **1 tot 8**. **x en y** zijn de **coördinaten** van de **linkerbovenhoek** en **w en h** zijn de **breedte en hoogte** van de afbeelding. **READ** kopieert het weergegeven beeld naar de buffer, **WRITE** kopieert de buffer naar het display en **CLOSE** zal de buffer vrijmaken en het gebruikte geheugen terugwinnen. **LOAD** laadt een afbeeldingsbestand in de buffer.

BLIT LOAD en **BLIT WRITE** werken op elk scherm, terwijl **BLIT** en **BLIT READ** alleen werken op displays die transparante tekst kunnen weergeven (d.w.z. met behulp van de **SSD1963, ILI9341, ST7789_320** of **ILI9488** met **MISO** verbonden).
Deze opdrachten kunnen worden gebruikt om een deel van het scherm naar een andere locatie te kopiëren (door vervolgens naar een buffer te kopiëren) ergens anders schrijven), maar een eenvoudigere methode is om als volgt een alternatieve versie van het **BLIT** commando te gebruiken:

```
BLIT x1, y1, x2, y2, w, h
```

Dit kopieert een deel van de afbeelding op **x1/y1** naar de locatie **x2/y2**. **w en h** specificeren de **breedte en hoogte** van de te kopiëren afbeelding. De bron- en bestemmingsgebieden kunnen elkaar overlappen en het **BLIT** commando zal de correct kopiëren.

Deze vorm van het BLIT commando is vooral handig voor het maken van grafieken die horizontaal kunnen scrollen of verticaal naarmate er nieuwe gegevens worden toegevoegd.

Afbeelding laden

Zoals eerder beschreven in de sectie **"SD-kaartondersteuning"** kunnen de opdrachten **LOAD IMAGE** en **LOAD JPG** worden gebruikt om een afbeelding van de **SD-kaart** te laden en op het LCD-scherm weer te geven. Dit kan worden gebruikt om een logo te tekenen of voeg een sierlijke achtergrond toe aan de afbeeldingen die op het scherm zijn getekend.

Voorbeeld

Als voorbeeld zal het volgende programma een eenvoudige digitale klok tekenen op een op **ILI9341** gebaseerd LCD-scherm. De programma wordt beëindigd en keert terug naar de opdrachtprompt als het scherm wordt aangeraakt.
Eerst moeten de weergave- en aanraakopties worden geconfigureerd door de opdrachten in te voeren die aan het begin hiervan worden vermeld hoofdstuk. Het exacte formaat hiervan hangt af van hoe u het displaypaneel hebt aangesloten.
Voer vervolgens het programma in en voer het uit:

```
CONST DBlue = RGB(0, 0, 128)       ' Een donkerblauwe kleur
COLOR RGB(GREEN), RGB(BLACK)       ' De standaardkleuren instellen
FONT 1, 3                          ' Het standaardlettertype instellen

BOX 0, 0, MM.HRes -1, MM.VRes / 2, 3, RGB(RED), DBlue

DO
  TEXT MM.HRes / 2, MM.VRes / 4, TIME$, "CM", 1, 4, RGB(CYAAN), DBlue
  TEXT MM.HRes / 2, MM.VRes * 3 / 4, DATE$, "CM"
  IF TOUCH(X) <> -1 THEN END
LOOP
```

Dit programma begint met het definiëren van een constante met een waarde die overeenkomt met een donkerblauwe kleur en stelt vervolgens de standaardwaarden voor de kleuren en het lettertype.
Vervolgens tekent het een doos met rode muren en een donkerblauw interieur.

Hierna komt het programma in een continue lus waar het drie functies vervult:

1. Geeft de huidige tijd weer in het eerder getekende vak. De string wordt zowel horizontaal gecentreerd getekend en verticaal in het midden van de doos. Merk op dat het **TEXT** commando zowel het standaardlettertype als overschrijft kleuren om zijn eigen parameters in te stellen.
2. Tekent de datum gecentreerd in de onderste helft van het scherm. In dit geval gebruikt het **TEXT** commando de standaard lettertype en kleuren eerder ingesteld.
3. Controleert op aanraking op het scherm. Dit wordt aangegeven wanneer de **TOUCH(X)** functie iets teruggeeft anders dan **-1**. In dat geval wordt het programma beëindigd.

PicoMite geavanceerde grafische afbeeldingen

De PicoMite bevat een reeks geavanceerde grafische bedieningselementen die reageren op aanraking, waaronder op het scherm schakelaars, knoppen, indicatielampjes, toetsenbord, enz. MMBasic tekent de besturing en animeert deze (d.w.z. een schakelaar lijkt ingedrukt te zijn bij aanraking). Het enige dat het BASIC-programma hoeft te doen, is een enkele opdracht aanroepen om specificeer de basisdetails van de besturing. Merk op dat deze functie niet beschikbaar is op de PicoMiteVGA-versie.
Om de GUI-bedieningselementen in de PicoMite te gebruiken, moet het geheugen dat nodig is voor de GUI-bedieningselementen eerst worden toegewezen door met behulp van de opdracht **OPTION GUI CONTROLS**. Normaal gesproken zou u de opdracht als volgt gebruiken:

```
OPTION GUI CONTROLS 75
```

Hiermee wordt het maximale aantal bedieningselementen dat u kunt definiëren ingesteld op 75. Deze optie is permanent (d.w.z. het zal onthouden worden bij het uitschakelen). Standaard is het maximale aantal bedieningselementen ingesteld op nul en in dit geval is de GUI-functies zijn niet beschikbaar en er wordt geen geheugen gebruikt.

Besturingselementen definiëren

Dit zijn enkele van de geavanceerde **GUI** besturingselementen die u kunt gebruiken:

Elke controle heeft een referentienummer genaamd '#ref' in de beschrijving van de controle. Dit kan elk nummer zijn tussen 1 en de bovengrens ingesteld door het **OPTION CONTROL** commando. Dit referentienummer wordt gebruikt om een controle identificeren. Er kan bijvoorbeeld een selectievakje worden gemaakt met het referentienummer **#10**:

```
GUI CHECKBOX #10, "Test", 100, 100, 50, RGB(BLUE)
```

Eenmaal aangemaakt kan de gebruiker het vakje aan- en uitvinken met behulp van de aanraakfunctie van het LCD-paneel zonder het lopend BASIC-programma betrokken is.
Indien nodig kan het programma de waarde van het selectievakje bepalen door met behulp van het referentienummer in de **CtrlVal()** functie:

```
IF CtrlVal(#10) THEN
```

Het **#**-teken is optioneel, maar dient om de programmeur eraan te herinneren dat dit geen gewoon nummer is.
In de volgende opdrachten zijn alle argumenten die cursief zijn (bijv. Breedte, Hoogte) optioneel en zo niet opgegeven zal de waarde aannemen van de vorige opdracht die ze wel specificeerde. Dit betekent bijvoorbeeld dat het aantal keuzerondjes met dezelfde grootte en kleur kan worden opgegeven, waarbij alleen de eerste knop hoeft te worden weergegeven al de details. Merk op dat dit met de kleurspecificatie anders is dan de basistekenopdrachten die standaard ingesteld op de laatste **COLOR** opdracht.

Alle tekenreeksen die worden gebruikt in **GUI** besturingselementen en de **MsgBox** kunnen meerdere regels weergeven door het tilde-teken **(~)** te gebruiken om scheid elke regel in de tekenreeks. Het bijschrift van een drukknop kan bijvoorbeeld **"ALARM~TEST"** zijn en dit zou worden weergegeven als twee regels. Voor alle bedieningselementen is het lettertype dat voor het bijschrift wordt gebruikt, het lettertype dat is ingesteld met de FONT commando en de kleuren zullen zijn wat is ingesteld door het laatste **COLOR** commando.

Als het display transparante tekst kan weergeven, kunnen deze commando's het gebruik van -1 voor de achtergrondkleur toestaan.
Dit betekent dat de tekst over de achtergrond wordt getekend, waarbij de achtergrondafbeelding door de gaten in de brieven.

De geavanceerde grafische bedieningselementen zijn:

Kader

```
GUI FRAME #ref, caption$, CenterX, CenterY, Diameter, Colour
```

Hiermee wordt een kader getekend dat een doos is met ronde hoeken en een bijschrift. Een frame reageert niet op aanraking maar is handig wanneer een groep bedieningselementen visueel moet worden samengebracht. Het kan ook worden gebruikt om een groep te omringen van keuzerondjes en MMBasic zorgt ervoor dat de keuzerondjes omgeven door het frame exclusief zijn - dat wil zeg-gen, wanneer een keuzerondje wordt geselecteerd, wordt elke andere knop die was geselecteerd en binnen het frame is gedeselecteerd.

LED

```
GUI LED #ref, caption$, CenterX, CenterY, Diameter, Color
```

Dit zal een indicatielampje tekenen (het ziet eruit als een op een paneel gemonteerde LED). Wanneer de waarde is ingesteld op één, is het verlicht en wanneer het op nul is ingesteld, is het uit (een doffe versie van het kleurattribuut). De **LED** kan worden gemaakt knipperen door de waarde in te stellen op het **aantal milliseconden** dat het moet blijven branden voordat het wordt uitgeschakeld
Het **bijschrift** wordt rechts van de **LED** getekend en gebruikt de kleuren die zijn ingesteld met het **COLOR** commando. De **LED** bediening wordt niet geanimeerd bij aanraking, maar het referentie-nummer kan worden gevonden met **TOUCH(REF)** en **TOUCH(LASTREF)** in de touch-interrupts en elke gewenste animatie kan in MMBasic worden gedaan.

Selectievakje

```
GUI CHECKBOX #ref, caption$, StartX, StartY, Size, Color
```

Hiermee wordt een selectievakje getekend, een klein vakje met een bijschrift. Zowel de hoogte als de breedte worden gespecificeerd met de parameter **'Grootte'**. Bij aanraking wordt er een **X** in het vak getekend om aan te geven dat deze optie is ingeschakeld geselecteerd en de waarde van het besturingselement wordt ingesteld op **1**. Bij een tweede keer aanraken wordt het vinkje verwijderd en de waarde van het besturingselement zal nul zijn. Het bijschrift wordt rechts van het selectie-vakje getekend en gebruikt de kleuren ingesteld door het **COLOR** commando.

Druk op de knop

```
GUI BUTTON #ref, caption$, StartX, StartY, Width, Height, FColour, BColour
```

Hiermee wordt een tijdelijke knop getekend die een vierkante schakelaar is met het bijschrift op zijn gezicht. Wanneer aangeraakt de visuele afbeelding van de knop lijkt ingedrukt te zijn en de waarde van de besturing is **1**. Wanneer de aanraking is verwijderd, keert de waarde terug naar nul. Bijschrift kan een enkele tekenreeks zijn met twee bijschriften gescheiden door een verticaal bar **(|)** teken (bijv. **"UP|DOWN"**). Als de knop **UP** is, wordt de eerste string gebruikt en wanneer op de **DOWN** wordt gedrukt zal de tweede worden gebruikt.

Schakelaar

```
GUI SWITCH #ref, caption$, StartX, StartY, Width, Height, FColour, BColour
```

Hiermee wordt een vergrendelende schakelaar getekend met het bijschrift op zijn gezicht. Bij aanraking zal het visuele beeld van de knop lijken ingedrukt te zijn en de waarde van de regelaar zal **1** zijn. Bij een tweede keer aanraken zal de schakelaar losgelaten en de waarde keert terug naar nul. Bijschrift kan een enkele tekenreeks zijn met twee bijschriften gescheiden door een **|** teken (bijv. **"AAN|UIT"**). Wanneer dit wordt gebruikt, lijkt de schakelaar een tuimelschakelaar te zijn met elke helft van de bijschrift dat wordt gebruikt om elke helft van de tuimelschakelaar te labelen.

Radio knop

```
GUI RADIO #ref, caption$, CenterX, CenterY, Radius, Color
```

Hiermee wordt een keuzerondje met een bijschrift getekend. Bij aanraking wordt het midden van de knop verlicht om geven aan dat deze optie is geselecteerd en dat de waarde van het besturingselement 1 is. Wanneer een ander keuzerondje is geselecteerd, wordt de markering op deze knop verwijderd en is de waarde nul. Keuzerondjes zijn gegroepeerd wanneer omgeven door een kader en wanneer één knop in de groep is geselecteerd, worden alle andere in de groep gedeselecteerd. Als er geen frame wordt gebruikt, worden alle knoppen op het scherm gegroepeerd.
Het bijschrift wordt rechts van de knop getekend en gebruikt de kleuren die zijn ingesteld door het **COLOR** commando.

Displaydoos

```
GUI DISPLAYBOX #ref, StartX, StartY, Breedte, Hoogte, FColour, BColour
```

Hiermee teken je een doos met afgeronde hoeken. Elke tekenreeks kan in het vak worden weergegeven met behulp van de **CtrlVal(r) =** opdracht. Dit is handig voor het weergeven van tekst, cijfers en berichten. Dit besturingselement wordt niet geanimeerd wanneer aangeraakt, maar het referentie-nummer kan worden gevonden met **TOUCH(REF)** en **TOUCH(LASTREF)** in de aanraking interrupts en elke gewenste animatie kan in MMBasic worden gedaan.

Tekstvak

```
GUI TEXTBOX #ref, StartX, StartY, Breedte, Hoogte, FColour, BColour
```

Hiermee teken je een doos met afgeronde hoeken. Wanneer de doos wordt aangeraakt een QWERTY-toetsenbord zal verschijnen op het scherm zoals rechts weergegeven. Gebruik makend van op dit virtuele toetsenbord kan elke tekst worden ingevoerd het vak inclusief hoofdletters of kleine letters, cijfers en andere tekens in de ASCII karakterset. De nieuwe tekst vervangt elke tekst eerder in de doos.

Ent is de **enter-toets**, **Del** is de annuleertoets en zal sluit het tekstvak en zet het terug in de oor-spronkelijke staat, de driehoek is de **Shift-toets**, de **[ ]**-toets zal een spatie invoegen en de **&12**-toets selecteert een alternatieve toets selectie met cijfers en speciale tekens (er zijn twee sets speciale tekens en de shift-toets zal ertussen schakelen).
De weergegeven string kan worden ingesteld door een string toe te wijzen aan de box met behulp van de opdracht **CtrlVal(r) =**. De waarde van de besturing kan ook worden ingesteld op een string die begint met twee hekjes (**##**) en in dat geval de string (zonder de eerste twee hash-tekens) worden weergegeven in het vak met verminderde helderheid. Dit kan worden gebruikt om te geven
de gebruiker een hint over wat er moet worden ingevoerd (genaamd "spooktekst"). Het lezen van de waarde van de controleweergave spooktekst retourneert een lege tekenreeks. Wanneer een toets wordt ingedrukt, verdwijnt de spooktekst en wordt deze vervangen door de ingevoerde tekst.

MMBasic zal proberen het virtuele toetsenbord op het scherm te plaatsen om het tekstvak dat het veroorzaakte niet te verbergen tevoorschijn komen. Een onderbreking van de pen omlaag wordt gegenereerd net voordat het toetsenbord wordt ingezet en een onderbreking van de toets omhoog
worden gegenereerd wanneer de toetsen Enter of Cancel worden aangeraakt en het toetsenbord is verborgen.

Indien nodig kan het virtuele toetsenbord worden geforceerd om te verschijnen zonder dat de besturing wordt aangeraakt met het commando **GUI TEXTBOX ACTIVATE** en het kan worden beëindigd door het programma (hetzelfde als het aanraken van de annuleerknop) met het commando: **GUI TEXTBOX CANCEL**.

Nummervak

```
GUI NUMBERBOX #ref, StartX, StartY, Width, Height, FColour, BColour
```

Hiermee teken je een doos met afgeronde hoeken. Wanneer het vak wordt aangeraakt, er verschijnt een numeriek toetsenbord op het scherm zoals rechts weergegeven.

Dit gebruiken virtueel toetsen-bord elk nummer kan worden ingevoerd in de box met een getal met drijvende komma in exponen-tieel formaat. Het nieuwe nummer vervangt het nummer eerder in het vak.

De **Alt-toets** selecteert een alternatieve toetsselectie en de andere speciale toetsen zijn hetzelfde als bij het tekstvak.

Het weergegeven nummer kan ook worden ingesteld door toe te wijzen **een getal** (**float of integer**) naar het vak met behulp van de **CtrlVal(r) =** commando. gelijk aan het tekstvak, de waarde van het besturingselement kan worden ingesteld op een letterlijke tekenreeks met twee leidende hash-tekens (bijvoorbeeld **"##Hint"**) en daarin het geval dat de string (zonder de eerste twee tekens) wordt weergegeven in het vak met verminderde helderheid.

Als u dit leest, wordt nul teruggegeven en wanneer een toets wordt ingedrukt, verdwijnt de spooktekst.

MMBasic zal proberen het virtuele toetsenbord op het scherm te plaatsen om het nummervak dat het veroorzaakte niet te verbergen tevoorschijn komen. Een onderbreking van de pen omlaag wordt gegenereerd net voordat het toetsenbord wordt ingezet en een onderbreking van de toets omhoog
worden gegenereerd wanneer de Enter-toets wordt aangeraakt en het toetsenbord is verborgen. Wanneer de Enter-toets wordt aangeraakt, wordt ook de ingevoerde tekst wordt geëvalueerd als een getal en de **NUMBERBOX** besturing wordt opnieuw getekend om dit nummer weer te geven.

Indien nodig kan het virtuele toetsenbord worden geforceerd om te verschijnen zonder dat de besturing wordt aangeraakt met het commando **GUI NUMBERBOX ACTIVATE** en het kan worden verwijderd door het programma (hetzelfde als het aanraken van de annuleerknop) met het commando: **GUI NUMBERBOX CANCE**L.

Opgemaakt nummervak

GUI FORMATBOX #ref, Format, StartX, StartY, Width, Height, FColour, BColour

Hiermee teken je een doos met afgeronde hoeken. Wanneer het vakje wordt aangeraakt, verschijnt een numeriek toetsenblok dat lijkt op een Nummer vak. Het verschil is dat de gebruiker voor het opgemaakte nummervak cijfers moet invoeren volgens een specifiek formaat voor datums, tijd, enz. Ongeldige toetsen op het toetsenbord worden uitgeschakeld en de gebruiker zal begeleid in hun inzending met begeleidende tekst. Dit betekent dat de programmeur er zeker van kan zijn dat de invoer door de gebruiker heeft altijd een vast formaat.
Het type invoer wordt als volgt bepaald door het argument **'Formaat'**:

DATE1	Datum in UK/Aust/NZ formaat (dd/mm/jj)
DATE2	Datum in VS-formaat (mm/dd/jj)
DATE3	Datum in internationaal formaat (jjjj/mm/dd)
TIME1	Tijd in 24-uurs notatie (uu:mm)
TIME2	Tijd in 24-uurs notatie met seconden (uu:mm:ss)
TIME3	Tijd in 12-uurnotatie (uu:mm AM/PM)
TIME4	Tijd in 12 uur notatie met seconden (uu:mm:ss AM/PM)
DATETIME1	Zowel datum (UK fmt) als tijd (12 uur) (dd/mm/jj uu:mm AM/PM)
DATETIME2	Zowel datum (UK fmt) als tijd (24 uur) (dd/mm/jj uu:mm)
DATETIME3	Zowel datum (VS fmt) als tijd (12 uur) (mm/dd/jj uu:mm AM/PM)
DATETIME4	Zowel datum (VS fmt) als tijd (24 uur) (mm/dd/jj uu:mm)
LAT1	Breedtegraad in graden, minuten en seconden (d°` mm' ss" N/S)
LAT2	Breedtegraad met seconden tot één decimaal (dd° mm' ss.s" N/S)
LONG1	Lengtegraad in graden, minuten en seconden (ddd° mm' ss" E/W)
LONG2	Lengtegraad met seconden tot één decimaal (ddd° mm' ss.s" E/W)
ANGLE1	Hoek in graden en minuten (ddd° mm')

Bijvoorbeeld:

```
GUI FORMATBOX #1, DATE1, 300, 150, 200, 50
```

zou een gegevensinvoervak maken en wanneer het wordt aangeraakt, zal een toetsenbord verschijnen zoals rechts wordt weergegeven.
Let daar op:

- De displaydoos is gevuld met een geleidingskoord om de gebruiker over de benodigde gegevens
- Omdat de dag van de maand alleen met een cijfer kan beginnen van 0 tot 3 zijn alle andere toetsen uitgeschakeld. Dit gebeurt ook met andere nummers die een beperkt bereik hebben.
- De waarde van het besturingselement dat wordt opgehaald via **CtrlVal(#1)** is een string.
  Als de gebruiker bijvoorbeeld de datum voor de 8e van Mei 2020 zou de geretourneerde string **"08/05/20"** zijn (d.w.z. de UK/Aust/NZ-formaat zoals gespecificeerd door **DATE1**).

De waarde van het besturingselement kan uit elkaar worden gehaald met behulp van de tekenreeks-functies of, in sommige gevallen, de tekenreeks kan worden gebruikt direct. Als u bijvoorbeeld het bovenstaande formaatvak gebruikt om een datum te krijgen van de gebruiker, de interne klok van PicoMite kan dan direct als volgt worden ingesteld:

```
DATE$ = CtrlVal(#1)
```

Het **RTC SETTIME** commando accepteert een enkel stringargument in de notatie dd/mm/jj uu:mm dus op dezelfde manier kan de **RTC** tijd als volgt worden ingesteld als het geformatteerde vak **DATETIME2** gebruikt voor 'Format':

```
RTC SETTIME CtrlVal(#1)
```

U kunt de Amerikaanse stijl **DATETIME4** gebruiken om de datum/tijd te krijgen. In dat geval zou je dit gebruiken om de RTC in te stellen:

```
RTC SETTIME MID$(CtrlVal(#1),4,3)+LEFT$(CtrlVal(#1),2)+RIGHT$((CtrlVal(#1),9)
```

MMBasic zal proberen het virtuele toetsenbord op het scherm te plaatsen om het formaatvak dat het veroorzaakte niet te verbergen verschijnen. Er wordt een onderbreking van de pen naar beneden gegenereerd wanneer het toetsenbord wordt gebruikt en een onderbreking van de toets omhoog zal zijn gegenereerd wanneer alle vereiste gegevens zijn ingevoerd en het toetsenbord is verborgen.
Indien nodig kan het virtuele toetsenbord worden geforceerd om te verschijnen zonder dat de besturing wordt aangeraakt met het commando **GUI FORMATBOX ACTIVATE** en het kan door het programma worden gesloten (hetzelfde als het aanraken van de annuleerknop) met het commando: **GUI FORMATBOX CANCEL**.

Spin Box

```
GUI SPINBOX #ref, StartX,StartY,Width,Height,FColour,BColour, Step, Min, Max
```

Hiermee wordt een vak getekend met pictogrammen omhoog/omlaag aan beide uiteinden. Wanneer deze pictogrammen worden aangeraakt, het nummer in het vak wordt verhoogd of verlaagd met de **'StepValue'**, het ingedrukt houden van de aanraking herhaalt zich in een snel tempo. **'Minimum'** en **'Maximum'** stellen een grens aan de waarde die kan worden ingevoerd. **'StepValue'**, **'Minimum'** en **'Maximum'** zijn optioneel en indien niet gespecificeerd is **'StepValue' 1** en zal er geen limiet op het ingevoerde aantal. Elke keer dat omhoog/omlaag wordt aangeraakt of wanneer automatische herhaling plaatsvindt.

Ondertiteling

```
GUI CAPTION #ref, text$, StartX, StartY, Alignment, FColour, BColour
```

Hiermee wordt een tekstreeks op het scherm getekend. Het is vergelijkbaar met het basis-tekencommando **TEXT**, het verschil zijnde dat MMBasic deze controle automatisch zal dim-men als een toetsenbord of numeriek toetsenblok wordt weergegeven.
'Uitlijning' is nul tot drie tekens (een tekenreeksuitdrukking of variabele is ook toegestaan) waarbij de eerste letter is de horizontale uitlijning rond **X** en kan **L, C** of **R** zijn voor **LINKS, MIDDEN, RECHTS** en de tweede letter is de verticale uitlijning rond **Y** en kan **T, M** of **B** zijn voor **BOVEN**, **MIDDEN, ONDER**.
Een derde teken kan worden gebruikt om de rotatie van de tekst aan te geven. Dit kan **'N'** zijn voor normale **oriëntatie**, **'V'** voor **verticale tekst** waarbij elk teken onder het vorige van boven naar beneden loopt, **'I'** de tekst wordt **omgekeerd** (d.w.z. **ondersteboven**), '**U**' de tekst wordt **90° tegen de klok in gedraaid** en '**D**' de tekst wordt **met de klok mee gedraaid 90º**. De **standaarduitlijning** is **links/boven zonder rotatie**.
Als de **kleuren niet zijn opgegeven**, gebruikt dit besturingselement de kleuren die zijn ingesteld met het **COLOR** commando.

Ronde meter

```
GUI GAUGE #ref,StartX,StartY,Radius,FColour,BColour,min,max,nbrdec,units$,
c1,ta,c2,tb,c3,tc,c4
```

Dit definieert een grafische cirkelvormige analoge meter met een digitaal display in het midden dat de waarde en eenheden. Indien gespecificeerd, wordt de meter gekleurd om een grafische indicatie van het signaalniveau te geven (bijv. groen voor OK, geel voor waarschuwing, enz.).
'StartX' en **'StartY'** zijn de coördinaten van het midden van de meter terwijl **'Radius'** is de afstand van het midden tot de buitenrand.
'min' is de waarde die is gekoppeld aan de minimumwaarde van de meter en **'max'** is de maximale waarde. Wanneer **CtrlVal()** wordt gebruikt om een waarde toe te wijzen (zwevende punt of integer) naar de meter wordt het analoge deel van de meter naar een lengte getekend evenredig met het bereik tussen **'min'** en **'max'.**
Tegelijkertijd wordt de digitale waarde in het midden van de meter getekend met behulp van de huidige lettertype-instellingen (ingesteld met de opdracht **FONT**). **'nbrdec'** specificeert het nummer aantal decimalen dat in deze weergave moet worden gebruikt. Onder de digitale waarde zullen de **'eenheden$'** worden weergegeven (dit kan worden overgeslagen of een tekenreeks met een lengte van **nul** worden gebruikt als dit niet nodig is).

Normaal gesproken wordt de analoge grafiek getekend met de kleur die is opgegeven in **'Fcolour'** echter een meerkleurenmeter kan worden gemaakt met **'c1'** tot **'c4'** voor de kleuren en **'ta'** tot **'tc'** voor de drempels die worden gebruikt om te bepalen wanneer de kleur zal veranderen.

In het bijzonder is **'c1'** de kleur die moet worden gebruikt voor waarden tot **'ta'**. **'c2'** is de kleur die moet zijn gebruikt voor waarden tussen **'ta'** en **'tb', 'c3'** wordt gebruikt voor waarden tussen **'tb'** en **'tc'** en **c4** wordt gebruikt voor waarden boven **'tc'**. Niet benodigde kleuren en drempels kunnen weggelaten worden dan lijst. Voor een tweekleurenmeter hoeven bijvoorbeeld alleen **'c1'**, **'ta'** en **'c2'** te worden opgegeven.

Wanneer kleuren en drempels zijn opgegeven, wordt de achtergrond van de meter getekend met een saaie versie van de meetkleur op dat niveau ("spookkleuring") zodat de gebruiker kan inschatten hoe dicht de verschillende drempels zijn de werkelijke waarde is. Ook de digitale waarde die in het midden wordt weergegeven, verandert ook in de kleur die wordt gespecificeerd door de huidige waarde.

Als er slechts één kleur nodig is voor de hele analoge grafiek, kan deze worden gespecificeerd door **'c1'** te gebruiken en alles te laten de volgende parameters uit.

Staafmeter

```
GUI BARGAUGE #ref,StartX,StartY,breedte,hoogte,FColour,BColour,min,max,
c1,ta,c2,tb,c3,tc,c4
```

Dit definieert een horizontale of verticale staafmaat. de meter kan worden gekleurd om een grafische indicatie van het signaalniveau te geven (bijv. groen voor OK, geel voor waarschuwing, enz.) en veel staafdiagrammen kunnen dicht bij elkaar worden verpakt zodat een aantal waarden kan worden weergegeven tegelijkertijd een kleine hoeveelheid schermruimte gebruiken (zoals weergegeven in de afbeelding die uit tien staafmaten bestaat).

Als de breedte kleiner is dan de hoogte, wordt de staafmaat getekend verticaal met de analoge grafiek groeiend van onder naar de top. Anders, als de breedte meer is dan de hoogte, zal het zijn horizontaal getekend met de analoge grafiek groeiend van links naar rechts. In beide gevallen verwijzen **'StartX'** en **'StartY'** naar de coördinaat linksboven van de staafgrafiek terwijl **'breedte'** de horizontale is breedte en **'hoogte'** de verticale hoogte.

De staafgrafiek heeft geen digitale weergave van de waarde, maar andere dan dat de parameters hetzelfde zijn als voor de ronde meter (hierboven omschreven).

'min' en **'max'** specificeren het bereik van waarden voor de balk en, indien gespecificeerd, **'c1'** tot **'c4'** en **'ta'** tot **'tc'** specificeren de kleuren en drempels voor het analoge balkbeeld. Merk op dat in tegenstelling tot de ronde staafmeter een **"spookbeeld"** van de kleuren worden niet weergegeven op de achtergrond.

Net als bij de ronde meter, als er slechts één kleur nodig is voor de hele meter, kan deze worden gespecificeerd door gewoon te gebruiken **'c1'** en laat u alle volgende parameters uit.

Gebied

```
GUI AREA #ref, StartX, StartY, Width, Height
```

Dit definieert een onzichtbaar gebied van het scherm dat gevoelig is voor aanraking en stelt **TOUCH(REF)** en **TOUCH(LASTREF)** dienovereenkomstig wanneer aangeraakt of losgelaten. Het kan worden gebruikt als basis voor het maken van een custom controle die wordt gedefinieerd en beheerd door het BASIC-programma.

Interactie met bedieningselementen

Met behulp van de volgende opdrachten en functies kunnen de kenmerken van de bedieningselementen op het scherm worden gewijzigd en hun waarde opgehaald.

- **= CTRLVAL(#ref)**
  Dit is een functie die de huidige waarde van een besturingselement retourneert. Voor bedieningselementen zoals selectievakjes of schakelaars zal de nummer één (true) zijn om aan te geven dat het besturingselement is geselecteerd door de gebruiker of nul (false) als dat niet het geval is.
  Voor bedieningselementen die een nummer bevatten (bijv. een **SPINBOX**) is de waarde het nummer (normaal gesproken een drijvende komma nummer). Voor besturingselementen die een tekenreeks bevatten (bijv. **TEXTBOX**) is de waarde een tekenreeks. Bijvoorbeeld:

```
PRINT "Het nummer in de spinbox is: " CTRLVAL(#10)
```

- **CTRLVAL(#ref) =**
  Met deze opdracht wordt de waarde van een besturingselement ingesteld. Voor aan/uit bedieningen zoals selectievakjes zal het elke . overschrijven aanraakinvoer en kan worden gebruikt om schakelaars in/uit te schakelen, selectievakjes aan of uit te vinken, enz. Een waarde van nul is uitgeschakeld of niet aangevinkt en niet nul zal de besturing aanzetten. Voor een **LED** zal het ervoor zorgen dat de **LED** oplicht of uitgeschakeld. Het kan ook worden gebruikt om de beginwaarde van draaivakken, tekstvakken, enz. in te stellen. Bijvoorbeeld:

```
CTRLVAL(#10) = 12, 4
```

- **GUI FCOLOUR colour, #ref1 [, #ref2, #ref3, etc]**
  Hierdoor wordt de voorgrondkleur van de opgegeven bedieningselementen gewijzigd in **'kleur'**. Dit is vooral handig voor een **LED** die van kleur kan veranderen.

- **GUI BCOLOUR colour, #ref1 [, #ref2, #ref3, etc]**
  Hierdoor wordt de achtergrondkleur van de opgegeven bedieningselementen gewijzigd in **'kleur'.**

- **= TOUCH(REF)**
  Dit is een functie die het referentienummer teruggeeft van de knop die momenteel wordt aangeraakt. Als er geen controle is momenteel wordt aangeraakt, wordt nul geretourneerd.

- **= TOUCH(LASTREF)**
  Dit is een functie die het referentienummer retourneert van de knop die het laatst is aangeraakt.

- **GUI DISABLE #ref1 [, #ref2, #ref3, etc]**
  Hierdoor worden de bedieningselementen in de lijst uitgeschakeld. Uitgeschakelde bedieningselementen reageren niet op aanraking en worden weergegeven gedimd. Het sleutelwoord **ALL** kan als argument worden gebruikt en dat zal alle bedieningselementen op de huidigeweergegeven pagina. Bijvoorbeeld:

  ```
  GUI DISABLE ALL
  ```

- **GUI ENABLE #ref1 [, #ref2, #ref3, etc]**
  Dit maakt de effecten van **GUI DISABLE** ongedaan en herstelt de bedieningselementen in de lijst naar de normale werking. De trefwoord **ALL** kan worden gebruikt als argument voor alle bedieningselementen op de momenteel weergegeven pagina.

- **GUI HIDE #ref1 [, #ref2, #ref3, etc]**
  Hierdoor worden de bedieningselementen in de lijst verborgen. Verborgen bedieningselementen reageren niet op aanraking en worden vervangen op het scherm met de huidige achtergrondkleur. Het sleutelwoord **ALL** kan als argument worden gebruikt.

- **GUI SHOW #ref1 [, #ref2, #ref3, etc]**
  Hiermee worden de effecten van **GUI HIDE** ongedaan gemaakt en worden de bedieningselementen in de lijst weer zichtbaar en in staat om normale operatie. Het sleutelwoord **ALL** kan worden gebruikt als argument voor alle besturingselementen.

- **GUI DELTE #ref1 [, #ref2, #ref3, etc]**
  Hiermee worden de besturingselementen in de lijst verwijderd. Dit omvat het verwijderen van de afbeelding van de besturing van het scherm de huidige achtergrondkleur gebruiken en het geheugen vrijmaken dat door de besturing wordt gebruikt. Het sleutelwoord **ALL** kan worden gebruikt als het argument en dat zal ertoe leiden dat alle besturingselementen worden verwijderd.

MsgBox()

De functie **MsgBox()** zal een berichtvenster op het scherm weergeven en wachten op invoer van de gebruiker. Terwijl het bericht wordt weergegeven, worden alle bedieningselementen uitgeschakeld, zodat de berichtenbox de volledige focus heeft.

De syntaxis is:

```
r = MsgBox(message$, button1$ [, button2$ [, button3$ [, button4$]]])
```

Alle argumenten zijn strings. **'message$'** is het bericht dat moet worden weergegeven. Dit kan een of meer tilde-tekens bevatten **(~)** die een regeleinde aangeven. Er kunnen **maximaal 10 regels** in de doos worden weergegeven. **'button1$'** is het **bijschrift** voor de **eerste knop**, **'button2$'** is het bijschrift voor de tweede knop, enz. Er moet minimaal één knop worden opgegeven en vier is het maximum. Knoppen die niet in de argumentenlijst zijn opgenomen, worden niet weergegeven. Het gebruikte lettertype is het standaardlettertype dat is ingesteld met de opdracht **FONT** en de gebruikte kleuren zijn de standaardwaarden ingesteld door het **COLOR** commando. De doos wordt automatisch op maat gemaakt, rekening houdend met de afmetingen van de standaardlettertype, het aantal regels dat moet worden weergegeven en het aantal opgegeven knoppen.

Wanneer de gebruiker een knop aanraakt, wist het berichtvenster zichzelf, herstelt het scherm (bijv. schakel alle bedieningselementen opnieuw in) en retourneer het nummer van de knop die is aangeraakt (de eerste knop retourneert 1, de tweede 2, enz.).

Opmerking dat, in tegenstelling tot alle andere **GUI**-besturingselementen, het **BASIC**-programma stopt met draaien terwijl het bericht-venster wordt weergegeven, interrupts worden echter gehonoreerd en opgevolgd.

Om het gebruik van een berichtvenster te illustreren, zal het volgende programmafragment proberen een bestand te openen en als er een fout optreedt, geeft het programma een foutmelding weer met behulp van de functie **MsgBox()**. Het bericht heeft twee lijnen en het vak heeft twee knoppen voor opnieuw proberen en annuleren.

```
Do
  On Error Skip
  Open "file.txt" For Input As #1
  If MM.Errno <> 0 Then
    If MsgBox("Error~Opening file.txt","RETRY","CANCEL") = 2 Then Exit Sub
  EndIf
Loop While MM.Errno <> 0
```

Dit zou het resultaat zijn als het bestand **"file.txt"** niet zou bestaan:

Afbeelding zie pdf in orginele document: http://geoffg.net/picomite.html

WiFi- en internet functies
ALLEEN WEBMITE VERSIE

Dit hoofdstuk geeft een overzicht van de WiFi- en internetfuncties die in de **WebMite** versie zijn geïmplementeerd de **Raspberry Pi Pico W** en de **Raspberry Pi Pico 2 W**.
Deze functies zijn complex en daarom moeten een paar punten worden opgemerkt:

- Het implementeren van internetprotocollen vergt veel van de bronnen van de processor (vooral RAM) de **WebMite** firmware mag alleen worden gebruikt waar WiFi en internet-verbinding belangrijk zijn enige prestatie-impact kan worden getolereerd in vergelijking met de standaard Raspberry Pi Pico.
- Deze handleiding beschrijft hoe de WebMite met de WiFi- en internetprotocollen communiceert en biedt enkele eenvoudige voorbeelden, maar het beschrijft niet **HTML, TCP** en de vele andere protocollen en verdragen betrokken. Ze kunnen verwarrend zijn voor de nieuwkomer, die een aantal van de veel primers beschikbaar op internet.
- Tijdens het verwerken van complexe internetprotocollen kan de WebMite-firmware in de war raken en vastlopen of weggooien een fout en keer terug naar de opdrachtprompt Om dit mogelijk te maken, moeten programma's de **WATCHDOG** gebruiken functie om uit dergelijke situaties te herstellen. Het wordt ook aanbevolen om af en toe (bijvoorbeeld één keer per week) het programma zou een herstart moeten forceren met behulp van **CPU** **RESTART** om ervoor te zorgen dat de protocolstacks opnieuw worden ingesteld.

Verbinding maken met een WiFi-netwerk

Voordat u iets doet, moet u de WiFi-functie op de WebMite inschakelen en dit doet u met de volgende opdracht ingevoerd op de opdrachtprompt:

```
OPTION WIFI ssid, password
```

Waarbij **'ssid'** de naam is van het WiFi-netwerk en **'wachtwoord'** het beveiligingswachtwoord is dat wordt gebruikt om toegang te krijgen tot het netwerk. Beide zijn strings en als stringconstanten worden gebruikt, moeten deze tussen aanhalingstekens worden geplaatst, zoals in dit voorbeeld wordt getoond:

```
OPTION WIFI "MijnNetwerk", "geheim"
```

Wanneer dit wordt ingevoerd, wordt de **WebMite** opnieuw opgestart, wat betekent dat u de USB-verbinding en toegang verliest de console. Als u snel opnieuw verbinding maakt, ziet u het volgende bericht:

```
Connecting to WiFi...
Connecting 192.168.1.52
```

Het adres dat in de laatste regel wordt vermeld, is het IP-adres dat door de router aan de WebMite is gegeven en kan afhankelijk hiervan variëren op het netwerk. Als de WebMite geen verbinding kan maken, ziet u dit bericht:

```
Connecting to WiFi...
failed to connect.
```

Als de verbinding mislukt, controleer dan de configuratie van uw WiFi-router:

- De WiFi-beveiliging moet WPA-PSK zijn met TKIP- of AES-codering (of beide).
- Voor de WiFi-toegang moet een wachtwoord zijn ingesteld.
- DHCP moet worden geconfigureerd.

Deze opdracht hoeft slechts één keer te worden ingevoerd en wordt elke keer dat uw WebMite opnieuw wordt opgestart onthouden.
U kunt de instelling ervan controleren met de commando **OPTION LIST**.
Wanneer u verbonden bent, kunt u ook het IP-adres controleren adres als volgt toegewezen:

```
> PRINT MM.Info(ip-adres)
192.168.1.52
```

Bij sommige routers kan het enige tijd of een paar pogingen duren om verbinding te maken, dus als u **OPTION** gebruikt **AUTORUN ON**, het zou de moeite waard zijn om zoiets helemaal aan het begin van uw programma in te voegen:

```
DO WHILE MM.INFO(IP ADDRESS) = "0.0.0.0"
  IF TIMER > 5000 THEN CPU RESTART
LOOP
```

Dit zou 5 seconden wachten op een verbinding en opnieuw opstarten als er nog steeds geen verbinding is.

Toegang tot console op afstand

Via WiFi kunt u via Telnet op afstand verbinding maken met de console van de WebMite. Dit is handig als het apparaat is op een moeilijk bereikbare locatie. Eenmaal verbonden via Telnet kunt u alles doen wat u zou willen normaal gesproken via de USB-console, inclusief het uitvoeren van de editor.
Om deze functie in te schakelen, typt u het volgende achter de opdrachtprompt:

```
OPTION TELNET CONSOLE ON
```

Dit hoeft u slechts één keer in te voeren en wordt onthouden telkens wanneer uw WebMite opnieuw wordt opgestart. Dat zal ook zo zijn ervoor zorgen dat de WebMite opnieuw opstart, dus u moet opnieuw verbinding maken met de USB-console.
Tera Term (http://tera-term.en.lo4d.com) ondersteunt Telnet en wordt aanbevolen voor deze taak omdat het een goede VT100-emulatie en ondersteunt het XModem-protocol voor bestandsoverdracht.

Bestandsoverdracht

Bestanden kunnen via **XModem** of **TFTP** van en naar de **WebMite** worden overgebracht. XModem wordt ondersteund door Tera Term en zal via Telnet op dezelfde manier werken als via een directe seriële verbinding. TFTP is echter veel sneller en betrouwbaarder dan XModem, dus het is de aanbevolen methode voor het overbrengen van bestanden van en naar de WebMite.
Een TFTP-server op de WebMite wordt automatisch ingeschakeld wanneer u verbonden bent met een WiFi-netwerk, dus niets is daar nodig. U heeft wel een TFTP-client voor uw pc nodig en er zijn veel verschillende implementaties voor beschikbaar Windows, MacOS en Linux. Merk op dat in veel tutorials over TFTP het instellen van een TFTP-server wordt besproken; dit is niet het geval nodig, je hebt alleen een klant nodig.
Voor Windows is een TFTP-client opgenomen in het besturingssysteem, maar deze moet worden ingeschakeld via het Configuratiescherm.

Om dit te doen selecteert u:

Configuratiescherm -> Programma's en onderdelen -> Windows-functies in- of uitschakelen

Blader vervolgens door de lijst en vink **TFTP-client** aan.

Om een bestand van uw pc naar de WebMite te verzenden, voert u het volgende uit in een Command- of Power Shell-venster op uw computer Windows-pc (**'IP-Addr'** is het IP-adres van de WebMite):

TFTP -i IP-Addr PUT filenaam

En om een bestand van de WebMite naar uw pc te kopiëren:

TFTP -i IP-Addr GET filenaam

Gebruik het volgende om de functies van de TFTP-client weer te geven:

TFTP-h

Een alternatieve en eenvoudige grafische Windows TFTP-client is:

http://www.3iii.dk/linux/dd-wrt/tftp2.exe

De tijd krijgen

Een gebruikelijke eerste stap in een programma is het ophalen van de tijd/datum en het instellen van de klok in de WebMite. Deze actie ook bevestigt dat de WebMite het internet kan bereiken. Het verkrijgen van de tijd gebeurt als volgt met de **WEB NTP** commando:

```
WEB NTP [timeoffset [, NTPserver$]]
```

Waarbij **'timeoffset'** de tijdzone is waarin u zich bevindt en **'NTPserver$'** de naam of het IP-adres van de tijd is server te gebruiken. Deze laatste parameter is optioneel en als deze wordt weggelaten, zal de firmware een openbare tijdserverpool gebruiken. Als de parameter **'timeoffset'** wordt ook weggelaten. De klok van de **WebMite** wordt ingesteld op **UTC**.

Dit is een typisch voorbeeld voor een apparaat in de tijdzone van Los Angeles:

```
> WEB NTP -10
ntp address 27.124.125.251
kreeg ntp antwoord: 08/03/2023 05:34:57
```

Als de WebMite geen toegang heeft tot internet, krijgt u een foutmelding. Dit kan worden opgevangen met behulp van de **ON ERROR SKIP** opdracht en een passende actie die is ondernomen (dat wil zeggen opnieuw opstarten of een bericht weergeven voor de operator).

Bijvoorbeeld:

```
ON ERROR SKIP 3
WEB NTP -10
IF MM.ERRNO THEN WEB NTP -10
IF MM.ERRNO THEN PRINT "Failure to connect to the Internet" : CPU RESTART
```

De reden waarom we de **WEB NTP** opdracht twee keer proberen, is omdat de eerste poging vanwege een bepaalde timing mislukte fout (dit kan gebeuren) – het zou echter bij de tweede poging succesvol moeten zijn.

Een webserver implementeren

Een populaire vereiste is het opzetten van een webserver die gegevens weergeeft die door de WebMite over een gebruiker zijn verzameld vriendelijke webpagina.
De eerste stap is het configureren van de serverfunctie met behulp van deze opdracht bij de opdrachtprompt:

```
OPTION TCP SERVER PORT nn
```

Waarbij **'nn'** het te gebruiken poortnummer is (normaal gesproken **80** voor een webpagina).
Meestal zal het commando zijn:

```
OPTION TCP SERVER PORT 80
```

Net als bij de andere hierboven genoemde **OPTION** commando's hoeft dit slechts één keer te worden ingevoerd en wordt het onthouden elke keer dat de WebMite opnieuw wordt opgestart. Het zorgt er ook voor dat de WebMite opnieuw opstart en als u snel opnieuw verbinding maakt, wordt u ziet het volgende (met een ander IP-adres):

```
Starting server at 192.168.1.52 on port 80
```

Bij de bovenstaande stap werd de WebMite-firmware geconfigureerd om een TCP-server te ondersteunen. In je programma moet je beginnen de server draait met het volgende commando:

```
WEB TCP INTERRUPT InterruptSub
```

Waarbij **'InterruptSub'** de naam is van uw subroutine die wordt aangeroepen wanneer een verzoek wordt ontvangen door de **TCP**-server. Deze subroutine kan het commando: **WEB TCP READ** gebruiken om het binnenkomende verzoek van de externe client en het commando:
WEB TRANSMIT PAGE kunnen worden gebruikt om de gevraagde webpagina te verzenden.
Hieronder vindt u bijvoorbeeld het volledige programma om een eenvoudige webpagina te implementeren (vergeet niet het **OPTION TCP SERVER** commando eerst):

```
 1 DIM buff%(4096 / 8)
 2 WEB TCP INTERRUPT WebInterrupt
 3 DO
 4   '<do some processing here>'
 5 LOOP
 6
 7 SUB WebInterrupt
 8   LOCAL a%, p%, t%, s$
 9   FOR a% = 1 To MM.INFO(MAX CONNECTIONS)
10     WEB TCP READ a%, buff%()
11     p% = LINSTR(buff%(),"GET")
12     t% = LINSTR(buff%(),"HTTP")
13     If (p% <> 0) And (t% > p%) Then
14       WEB TRANSMIT PAGE a%,"index.html"
15     ENDIF
16   NEXT a%
17 END SUB
```

Hieronder wordt dit programma in detail beschreven:

Regel 1 Eerst wordt een **4K byte** buffer aangemaakt voor het binnenkomende verzoek van de client. In dit voorbeeld wordt gebruik gemaakt van de lange reeksopdrachten in MMBasic voor het verwerken van de gegevens en dit is de buffer hiervoor (zie het volgende hoofdstuk getiteld **LongStrings** voor een beschrijving van **Long Strings**). De grootte van deze buffer beperkt de hoeveelheid gegevens die van de klant worden ontvangen.

Regel 2 Hiermee start u de server en specificeert u de interrupt subroutine voor het afhandelen van inkomende verzoeken als **WebInterrupt**.

Regel 4 Dit is uw hoofdprogrammalus en normaal gesproken verzamelt u gegevens, schakelt u uitgangen, enz.

Regel 7 Dit is de subroutine die wordt aangeroepen telkens wanneer er een verzoek wordt gedaan door de externe clientbrowser.

Regel 9 Loop door alle inkomende verbindingen (er kunnen meerdere gelijktijdige verbindingen zijn).

Regel 10 Lees het binnenkomende bericht in de lange tekenreeksbuffer.

Regel 11 & 12 Ontvang de locaties van sleutelwoorden in het bericht.

Regel 13 Controleer of de trefwoorden aanwezig zijn en in de juiste volgorde staan.

Regel 14 Verstuur de pagina. Dit is een bestand met de naam index.html dat zich in de standaardmap op het intern flashbestandssysteem of **SD**-kaart. Het is opgemaakt in **HTML**, wat betekent dat het tags kan bevatten zoals **<h1>. Dit is een kop </h1>.** Zie Een typische **WEB pagina** hieronder voor meer informatie.

Gegevens in de webpagina invoegen

Meestal moet u de door het BASIC programma gegenereerde gegevens in de verzonden webpagina invoegen. Dit is eenvoudig gedaan door de naam van de BASIC variabele tussen accolades (dwz { en }) in de tekst in te voegen van de webpagina.
Als uw programma bijvoorbeeld een variabele had met de naam CurrentTemp, die de waarde 24 had en werd weergegeven de huidige temperatuur, de volgende webpagina:
De temperatuur is {CurrentTemp} zou worden weergegeven de browser van de klant als:

```
De temperatuur is 24.
```

De identificatie tussen de accolades kan een **Float**, **Integer** of **String**, een **Array** element en zelfs een expressie (dat wil zeggen, A + B). U kunt ook functies gebruiken, dus als u een drijvendekommagetal wilt opmaken met de correct aantal decimalen, enz. kunt u de opmaakfunctie **Str$()** gebruiken. Houd er rekening mee dat er een fout in de expressie zal een overeenkomstige fout veroorzaken wanneer de opdracht **WEB TRANSMIT PAGE** wordt uitgevoerd.
Als u op uw webpagina een accolade moet openen, kunt u er twee als paar gebruiken (dat wil zeggen, **{{}** en dat is gewijzigd in een enkele openingsbeugel. Een sluitende accolade zonder openingspartner blijft onaangeroerd.

Meerdere pagina's verzenden

Wanneer een externe client gegevens opvraagt zonder een pagina op te geven, verzendt deze het verzoek als **GET / HTTP** met de een schuine streep naar voren die de standaardpagina voor de server vertegenwoordigt (normaal gesproken **index.html**). Het voorbeeld hierboven nam niet de moeite om dit te controleren, het stuurde gewoon dezelfde pagina voor alle verzoeken.

Als uw pagina echter klikbare links bevat, zoals **Volgende pagina** en de gebruiker op deze link heeft geklikt, zal de externe client vervolgens een ander verzoek verzenden met **GET /page2.html HTTP**.
Dit kan eenvoudig worden gedaan door de gevraagde gegevens tussen de **GET**- en **HTML** trefwoorden te onderzoeken. Vervang bijvoorbeeld regel 15 in het bovenstaande voorbeeld door het volgende **(s$ is het opgevraagde bestand**):

```
s$ = LGetStr$(buff%(), p% + 4, t% - p% - 5)
IF s$ = "/" THEN
  WEB TRANSMIT PAGE a%,"index.html"
ELSE IF s$ = "/page2.html" THEN
  WEB TRANSMIT PAGE a%,"page2.html"
ELSE IF s$ = "/pix.jpg" THEN
  WEB TRANSMIT FILE a%,"pix.jpg","image/jpeg"
ENDIF
```

Dit kan worden uitgebreid tot zoveel pagina's als u nodig heeft.

Een afbeelding (image) verzenden

U kunt een afbeelding in uw webpagina invoegen met behulp van de volgende **html-code **. Wanneer de externe browser leest dit en stuurt het volgende verzoek **GET /pix.jpg HTTP**. Vervolgens kunt u de gevraagde afbeelding met behulp van de vetgedrukte code:

```
s$ = LGetStr$(buff%(), p% + 4, t% - p% - 5)
IF s$ = "/" THEN
  WEB TRANSMIT PAGE a%,"index.html"
ELSE IF s$ = "/page2.html" THEN
  WEB TRANSMIT PAGE a%,"page2.html"
ENDIF
```

Houd er rekening mee dat **pix.jpg** een **jpeg-image** moet zijn die zich in de standaardmap van het **interne flashbestandssysteem** of **SD bevindt** kaart. De WebMite is geen snelle server, dus kleine en eenvoudige afbeeldingen hebben de voorkeur.
De parameter **"image/jpeg"** staat bekend als een **MIME**-type en er zijn veel verschillende typen, waarvan andere gebruikelijk zijn afbeeldingstypen zijn **image/bmp**, **image/png** en **image/gif**.

Pagina niet gevonden (404) Reactie

Als een externe client een pagina of een bestand opvraagt dat niet door uw programma wordt ondersteund, kunt u **WEB** gebruiken **TRANSMIT CODE** commando om als volgt een **404** fout te verzenden:

```
s$ = LGetStr$(buff%(), p% + 4, t% - p% - 5)
IF s$ = "/" THEN
  WEB TRANSMIT PAGE a%,"index.html"
ELSE IF s$ = "/page2.html" THEN
  WEB TRANSMIT PAGE a%,"page2.html"
ELSE IF s$ = "/pix.jpg" THEN
  WEB TRANSMIT FILE a%,"pix.jpg","image/jpeg"
ELSE
  WEB TRANSMIT CODE a%, 404
ENDIF
```

Live grafische gegevens op een webpagina

Het weergeven van cijfers en tekst op een webpagina is handig, maar vaak wilt u ook grafische elementen opnemen zoals pi-diagrammen, lijngrafieken, historische trends, enz. afgeleid van de gegevens verzameld door het programma. Dit kan zijn gedaan op een omslachtige manier door de WebMite te configureren met een virtueel weergavepaneel en daar vervolgens op te tekenen weergeven met behulp van de standaard tekenopdrachten (pixel, lijn, cirkel, enz.) en dat opslaan als een **BMP-image**. Het bestand kan vervolgens als afbeelding in de webpagina worden opgenomen. Meer gedetailleerd is dit proces als volgt:
Eerst moet een virtueel display worden geconfigureerd. Er zijn er twee die je kunt gebruiken, **VIRTUAL_C** is een pixel van **320x240** afbeelding met **16 kleuren** en **VIRTUAL_M** is een **monochrome** afbeelding van **640x480** pixels. Bijvoorbeeld:

```
OPTION LCDPANEL VIRTUAL_C
```

Net als bij de andere **OPTION** commando's moet dit bij de opdrachtprompt worden ingevoerd, wat een herstart veroorzaakt hoeft slechts één keer ingevoerd te worden.
Vervolgens kunt u in uw programma afbeeldingen en tekst op dit **"display"** tekenen met behulp van de opdrachten die worden beschreven in het hoofdstuk Grafische opdrachten en functies.

Bijvoorbeeld:

```
CIRCLE 100, 100, 50, 1, 1, RGB(red), RGB(blue)
LINE 10, 10, 200, 200, 1, RGB(yellow)
```

Als u klaar bent, slaat u deze afbeelding op:

```
SAVE COMPRESSED IMAGE "graph.bmp"
```

Binnen de webpagina die u bedient, kunt u deze afbeelding invoegen met behulp van de volgende **html-code**:

```
<img src="graph.bmp"/>
```

Tenslotte moet u in uw BASIC-programma ervoor zorgen dat dit bestand wordt verzonden wanneer de webpagina door de externe browser zoals hierboven beschreven (zie Een afbeelding verzenden) Plaats dit bijvoorbeeld in de **ELSE IF** keten hierboven beschreven:

```
ELSE IF s$ = "/graph.bmp " THEN
  WEB TRANSMIT FILE a%, "graph.bmp", "image/bmp"
```

Uw BASIC-programma zal dit bestand moeten bijwerken als er nieuwe gegevens worden vastgelegd, maar dit hoeft alleen maar te gebeuren wanneer de externe browser de afbeelding opvraagt, mag dit geen overmatige slijtage van het flashgeheugen veroorzaken.

Een complete server voor algemeen gebruik

Op de bovenstaande pagina's zijn de afzonderlijke componenten beschreven waaruit een webserver bestaat. Hieronder ziet u een voorbeeld van een complete en geïntegreerde server voor algemeen gebruik die de meeste verzoeken van een browser kan verwerken. Als u de extensie van het aangevraagde bestand onderzoekt, wordt vervolgens het juiste **WEB TRANSMIT** commando gebruikt om het gevraagde gegevens. Het kan worden gebruikt als drop-in-module voor elk project waarvoor de WebMite als webserver nodig is.

```
WEB TCP INTERRUPT WebInterrupt
DO
  '<Do some processing here>'
LOOP
' Sub to handle all web server requests
SUB WebInterrupt
  LOCAL a%, p1%, p2%, file$, buff%(4096 / 8)
  FOR a% = 1 To MM.INFO(MAX CONNECTIONS)
    WEB TCP READ a%, buff%()
    P1% = LINSTR(buff%(),"GET")
    P2% = LINSTR(buff%(),"HTTP")
    If (p1% <> 0) And (p2% <> 0) And (p2% > p1%) Then
      file$ = LCASE$(LGetStr$(buff%(), p1% + 4, p2% - p1% - 5))
      IF file$ = "/" THEN file$ = "/index.html"
      ON ERROR SKIP
      OPEN file$ FOR INPUT AS #1 ' Check that the file exists
      IF MM.ERRNO THEN WEB TRANSMIT CODE a%, 404 : CONTINUE FOR
      CLOSE #1
      SELECT CASE RIGHT$(file$, 4)
        CASE "html", ".htm", ".txt"
          WEB TRANSMIT PAGE a%, file$
        CASE ".bmp", ".png", ".gif"
          WEB TRANSMIT FILE a%, file$, "image/" + RIGHT$(file$, 3)
        CASE ".jpg", "jpeg"
          WEB TRANSMIT FILE a%, file$, "image/jpeg"
        CASE ".ico"
          WEB TRANSMIT FILE a%, file$, "image/vnd.microsoft.icon"
      END SELECT
    ENDIF
  NEXT a%
END SUB
```

Houd er rekening mee dat alle bestanden moeten worden opgeslagen in de root van het flash-bestandssysteem of de SD-kaart en dat hun namen moeten worden gebruikt alleen kleine letters (het flashbestandssysteem is hoofdlettergevoelig).

Een typische webpagina

De WEB-pagina die via het **WEB TRANSMIT PAGE** commando moet worden verzonden, moet overeenkomstig zijn opgebouwd de html-standaard. Het kan zo simpel zijn als een enkele regel tekst zonder opmaak, dat wil zeggen:

```
De temperatuur is {CurrentTemp}
```

Of u kunt enkele eenvoudige opmaak toevoegen:

```
<title>WebMite</title>
<h2>Temperature Monitor</h2>
The temperature is {CurrentTemp}
```

Of u kunt een complexe pagina verzenden. Meestal hebben deze hoofd- en hoofdgedeelten en bakenen ze tekst af alinea's en gebruik de break-tag voor de afstand. Dit is het skelet van zo'n pagina:

```
<html>
<head>
<title>WebMite</title>
</head>
<body>
<h1>This is a heading</h1>
<br>
<p>The temperature is {CurrentTemp}</p>
</body>
</html>
```

Er zijn veel bronnen op internet die tutorials in **HTML** voor beginners aanbieden. Een typisch voorbeeld is http://www.simplehtmlguide.com/. Er zijn ook talloze **WISWIG HTML** editors beschikbaar. Bijvoorbeeld: https://onlinehtmleditor.dev/

Invoervelden en besturing

Met behulp van **HTML** invoervelden kunt u knoppen, selectievakjes, keuzerondjes, enz. op uw WEB-pagina plaatsen sta de gebruiker toe verzoeken naar de WebMite te sturen. Zo kan de gebruiker op afstand zaken aan/uit zetten, bediening instellen parameters en nog veel meer - allemaal vanaf de webpagina die door de WebMite wordt bediend.
Hiervoor moet u een HTML-formulier in de webpagina invoegen met daarin een of meer invoervelden. Er zijn veel soorten invoervelden om uit te kiezen.
(zie https://www.w3schools.com/tags/att_input_type.asp) maar in onze in een eenvoudig voorbeeld gebruiken we twee keuzerondjes om een fictief apparaat in en uit te schakelen.
De volgende code moet in de standaardwebpagina worden ingevoegd (dwz index.html):

```
<form method='get'>
   <input name='RB' type='radio' value='OFF' {offb$} onClick='this.form.submit()'> Off
   <input name='RB' type='radio' value='ON' {onb$} onClick='this.form.submit()'> On
</form>
```

Hierdoor worden twee keuzerondjes gemaakt die er in een browser als volgt uitzien:



Merk op dat de bovenstaande **HTML**-code twee BASIC-variabelen bevat (offb$ en onb$) die zullen worden vervangen door het **WEB TRANSMIT PAGE** commando. Deze variabelen bepalen hoe de knop wordt weergegeven. Als ze zijn ingesteld op een lege string, de knop wordt weergegeven als niet aangevinkt of, als deze is ingesteld op de string **"checked='checked'"**, de knop wordt weergegeven als aangevinkt. Deze variabelen moeten worden geïnitialiseerd wanneer het programma start.
Wanneer de gebruiker op de eerste knop klikt, verzendt de browser een verzoek met daarin:

```
GET /?RB=OFF HTML
```

en wanneer op de tweede wordt geklikt, is het verzoek: **GET /?RB=ON HTML**

In de **ELSE IF** keten in de **TCP**-interrupt-subroutine kunnen we op deze verzoeken reageren:

```
ELSE IF s$ = "/?RB=OFF"
    ' <insert code here to turn off the device>
    offb$ = "checked='checked'" : onb$ = ""
    WEB TRANSMIT PAGE a%, "index.html"
ELSE IF s$ = "/?RB=ON"
    ' <insert code here to turn on the device>
    offb$ = "" : onb$ = "checked='checked'"
    WEB TRANSMIT PAGE a%, "index.html"
```

Het enige dat deze code in wezen doet, is het apparaat in- of uitschakelen zoals gevraagd, de variabelen onb$ en offb$ instellen op geeft de nieuwe status van de knoppen weer en stuurt vervolgens de hele webpagina opnieuw terug naar de browser.
Dit voorbeeld heeft de vele details verdoezeld en je kunt het extreem ingewikkeld maken als je dat zou willen met meerdere invoermogelijkheden, waaronder knoppen, tekstinvoer, vervolgkeuzelijsten, wachtwoordvelden, verzoeken om bestandsuploads en nog veel meer. Om dit te doen, zul je echter dieper in HTML-codering moeten duiken. Voor een voorbeeld hiervan zie het Garden Watering Controller-project op: https://geoffg.net/retic.html

Een TCP-client implementeren

De WebMite kan ook fungeren als TCP-client om gegevens op te vragen bij een externe server. Dit wordt beheerd met drie opdrachten. De eerste is:

```
WEB OPEN TCP CLIENT Domein$, Poortnummer
```

Dit opent een TCP-verbinding met Domain$ (bijvoorbeeld "openweathermap.org") met behulp van het opgegeven Poortnummer (normaal gesproken **80** voor een webpagina).
Als de verbinding open is, kunt u met dit commando een of meer verzoeken verzenden:

```
WEB TCP CLIENT REQUEST query$, inbuf [, time-out]
```

Het te verzenden verzoek is **'query$'** en het antwoord wordt opgeslagen in **'inbuf'**, wat normaal gesproken een lange reeks is variabele zoals **buff%(4096/8)**. De grootte van deze buffer (in bytes) beperkt de hoeveelheid ontvangen gegevens van de server en moet worden verhoogd als er meer gegevens worden verwacht.
'time-out' is optioneel en is de time-out in milliseconden.
Als u een website bezoekt, kan **'query$'** zoiets eenvoudigs zijn als **'GET / HTTP'**, waarmee de standaardpagina voor die website. Uw programma is dan verantwoordelijk voor het uitkiezen van de gewenste gegevens vanuit de reactie.
Tenslotte sluit je de verbinding met:

```
WEB CLOSE TCP CLIENT
```

Het volgende voorbeeld is een compleet programma (met behulp van deze opdrachten) om de huidige temperatuur voor de stad Parijs van **openweathermap.com**. Om dit te laten werken heb je een (gratis) account nodig bij Open Weerkaart (https://openweathermap.org) die een API-sleutel bevat, dit is een 32-cijferig hexadecimaal nummer waarmee u maak de vraag. Dit nummer moet worden vervangen door de dummysleutel op de eerste regel.

```
CONST Key = "nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn"
CONST Query = "GET /data/2.5/weather?q=Paris,fra&APPID="+Key+Chr$(13)+Chr$(10)
DIM buff%(4096/8)

WEB OPEN TCP CLIENT "api.openweathermap.org", 80
WEB TCP CLIENT REQUEST Query, buff%()
WEB CLOSE TCP CLIENT
temp = VAL(JSON$(buff%(), "main.temp"))
PRINT "Current temperature in Paris is:" temp - 273
```

De functie **JSON()** wordt gebruikt om de gewenste waarde uit de **JSON (JavaScriptObjectNotation)** te extraheren geformatteerde reactie.
Wanneer dit programma wordt uitgevoerd, zou u iets moeten zien als:

```
Connected
Current temperature in Paris is: 13.54
```

UDP gebruiken

User Datagram Protocol (UDP) is een snel en lichtgewicht protocol dat wordt gebruikt om berichten tussen apparaten te verzenden. Het kan onbetrouwbaar zijn omdat het niet garandeert dat datapakketten in de juiste volgorde worden afgeleverd of aankomen, hoe dan ook vaak gebruikt voor tijdgevoelige toepassingen waarbij snelheid belangrijker is dan betrouwbaarheid. Het instellen van een UDP-server is vergelijkbaar met het instellen van een TCP-server. Ten eerste wordt UDP ingeschakeld met de volgende optie ingevoerd bij de opdrachtprompt (dit zal ook een herstart veroorzaken):

```
OPTION UDP SERVER PORT port_nbr
```

In uw programma moet u de server starten met het volgende commando:

```
WEB UDP INTERRUPT InterruptSub
```

Waarbij **'InterruptSub'** de naam is van uw subroutine die wordt aangeroepen wanneer een bericht wordt ontvangen door de UDP-server. Wanneer dit gebeurt, wordt het IP-adres van het verzendende apparaat in de alleen-lezen variabele bewaard **MM.ADDRESS$** en het ontvangen bericht in **MM.MESSAGE$**.
U kunt een UDP-bericht verzenden met:

```
WEB UDP Send ip_address$, port_nbr, message$
```

De volgende demonstratie gebruikt twee WebMites geconfigureerd met **OPTION WIFI** op hetzelfde netwerk.
Voordat u beide WebMites start, moeten ze worden geconfigureerd met:

```
OPTION UDP SERVER PORT 77
```

Noteer op de eerste WebMites het IP-adres en voer het volgende programma in:

```
WEB UDP Interrupt myint
WEB UDP Send "192.168.1.127", 77, "Starting UDP echo"
Do
  '<Do some processing here>'
Loop
```

```
' Interrupt subroutine.
Sub myint
  Print "Received " + MM.Message$ + " from " + MM.Address$
  Pause 100
  WEB UDP Send MM.Address$, 77, "WebMite #2 message"
End Sub
```

Voer vervolgens het programma uit, dat aanvankelijk in een lus blijft zitten en niets doet.
Op de tweede WebMite voert u het volgende programma in, waarbij u het IP-adres vervangt door dat van de eerste WebMite. Voer vervolgens het programma uit:

```
WEB UDP Interrupt myint
WEB UDP send "192.168.1.127", 77, "Starting UDP echo"
Do
  '<Do some processing here>'
Loop

'Interrupt subroutine.
Sub myint
  Print "Received " + MM.Message$ + " from " + MM.Address$
  Pause 100
  WEB UDP Send MM.Address$, 77, "WebMite #2 message"
End Sub
```

Het resultaat zal een zeer snelle pingpong van UDP-berichten tussen de twee WebMites zijn.

E-mails verzenden

Als u een extern apparaat zoals de WebMite heeft, is het handig dat u e-mails kunt verzenden om alarm te slaan over fouten, rapporteer over de status ervan, enzovoort. De WebMite kan dit doen door gebruik te maken van het SMTP-protocol om verbinding te maken met een server die de e-mail vervolgens naar de bestemming doorstuurt.
In het volgende voorbeeld wordt de gratis SMTP-relayservice gebruikt die wordt aangeboden door **SMTP2GO** en die **1000 e-mails** mogelijk maakt per maand (genoeg voor de WebMite).
Om te beginnen moet u een gratis login aanmaken op **SMTP2GO** (https://www.smtp2go.com/), een geverifieerde login registreren afzender en maak een bijbehorende gebruikersnaam en wachtwoord aan. Beide moeten vervolgens worden geconverteerd naar Base64-gecodeerd strings (de volgende website zal dit voor u doen: https://www.base64encode.org). Houd er rekening mee dat **SMTP2GO** accepteert **geen** registratieverzoeken van mensen met een algemeen gratis e-mail-adres (bijvoorbeeld xxx@gmail.com).

De Base64-gecodeerde gebruikersnaam moet worden gebruikt om de **nnnnnnnnnn-reeks** in de eerste regel van het volgende programma, terwijl het met Base64 gecodeerde wachtwoord moet worden gebruikt om **xxxxxxxxxxxx** in de tweede regel. De andere vier regels aan het begin van het programma moeten ook worden vervangen door uw gegevens:

```
CONST userBase64$  = "nnnnnnnnnnnn"
CONST paswdBase64$ = "xxxxxxxxxxxx"
CONST mailfrom$    = "from@server.com"
CONST mailto$      = "to@server.com"
CONST subject$     = "Test EMail"
CONST message$     = "Test of SMTP2GO"

CONST cr = Chr$(13) + Chr$(10)
DIM buff%(4096 / 8), body$

body$ = "From: " + mailfrom$ + cr + "To: " + mailto$ + cr
body$ = body$ + "Subject: " + subject$ + cr + cr
body$ = body$ + message$ + cr + "." + cr

WEB OPEN TCP CLIENT "mail.smtp2go.com", 2525
WEB TCP CLIENT REQUEST "EHLO" + cr, buff%()
WEB TCP CLIENT REQUEST "AUTH LOGIN" + cr, buff%()
WEB TCP CLIENT REQUEST userBase64$ + cr, buff%()
WEB TCP CLIENT REQUEST paswdBase64$ + cr, buff%()
WEB TCP CLIENT REQUEST "MAIL FROM: " + mailfrom$ + cr, buff%()
WEB TCP CLIENT REQUEST "RCPT TO: " + mailto$ + cr, buff%()
WEB TCP CLIENT REQUEST "DATA" + cr, buff%()
PAUSE 300
WEB TCP CLIENT REQUEST body$, buff%()
PAUSE 300
WEB CLOSE TCP CLIENT
IF LINSTR(buff%(), "250 OK") = 0 THEN
  PRINT "Email send failed"
Else
  Print "Email sent OK"
EndIf
```

Houd er rekening mee dat het mailfrom$-e-mailadres dat in het bovenstaande programma wordt gebruikt, hetzelfde **MOET** zijn als het e-mailadres dat werd gebruikt toen u registreerde de geverifieerde afzender met **SMTP2GO**. Als ze niet dezelfde zijn, zal **SMTP2GO** de e-mail weigeren (dit is een voorzorgsmaatregel tegen spam).
Tegenwoordig wordt het gebruik van een **SMTP**-relayservice bemoeilijkt door variaties in het SMTP-protocol dat door elke service wordt gebruikt leverancier en de verschillende beveiligingsmaatregelen die zijn getroffen om de hoeveelheid spam te verminderen. Dit voorbeeld is specifiek voor de SMTP-protocol zoals gebruikt door **SMTP2GO**, maar andere services kunnen ook worden gebruikt, maar het programma moet dat wel zijn aangepast om tegemoet te komen aan hun eigen versie van het SMTP-protocol.

Basis 64 codering

Base64 is een systeem voor het converteren van binaire gegevens naar een tekstreeks die alleen ASCII-tekens gebruikt (dat wil zeggen, er zijn geen controlekarakters). Het is ontworpen om het gemakkelijk te maken om binaire gegevens via internet te verzenden met behulp van protocollen die dat wel doen accepteren geen binaire gegevens en veel protocollen vereisen het gebruik ervan. De volgende functie converteert een MMBasic-reeks die binaire gegevens bevat naar een Base 64-gecodeerde reeks.

```
  Function base64Encode(si As String) As String
    Local String b64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
0123456789+/"
    Local Integer i, p, n, pad
    For p = 1 To Len(si) Step 3
      n = Asc(Mid$(si, p, 1)) << 8
      If p + 1 <= Len(si) Then n = n Or Asc(Mid$(si, p+1, 1)) Else pad = pad+1
      n = (n << 8)
      If p + 2 <= Len(si) Then n = n Or Asc(Mid$(si, p+2, 1)) Else pad = pad+1
      For i = 3 To 0 Step -1
        base64Encode = base64Encode+Mid$(b64, ((n >> (i*6)) And &b111111)+1, 1)
      Next i
    Next p
    base64Encode = Left$(base64Encode, Len(base64Encode)-pad)+Left$("==", pad)
  End Function
```

Een typisch gebruik is in het bovenstaande programma voor het verzenden van e-mails. In plaats van een externe service te gebruiken om de gebruikersnaam en wachtwoord naar Base64 kunt u dit in het programma doen met behulp van deze functie. Bijvoorbeeld:

```
  WEB TCP CLIENT REQUEST base64Encode("MyUserName") + cr, buff%()
  WEB TCP CLIENT REQUEST base64Encode("MyPassword") + cr, buff%()
```

MQTT client

MQTT is een protocol waarmee clients zoals de WebMite berichten kunnen plaatsen of ophalen op een server (ook wel een **MQTT** makelaar). Dit lijkt veel op een prikbord of een webgebaseerd forum waar mensen berichten plaatsen anderen kunnen het later op hun gemak lezen – het belangrijkste verschil is dat MQTT is ontworpen voor machine tot machine communicatie.
Een typisch voorbeeld zou een WebMite op batterijen kunnen zijn die het waterniveau in een dam bewaakt. Twee maal per dag. De volgende dag zou het opstarten, de meting uitvoeren, het resultaat op een MQTT-makelaar plaatsen en weer uitschakelen. Een clientprogramma (misschien op een pc) zou deze berichten later kunnen lezen, de resultaten kunnen weergeven en in een grafiek kunnen weergeven.
Er zijn veel gratis makelaars beschikbaar, gebruik Google om te zoeken naar "gratis MQTT broker".

De WebMite gebruikt vijf opdrachten om berichten te plaatsen of op te halen:

`WEB MQTT CONNECT`	Maak verbinding met een MQTT broker.
`WEB MQTT PUBLISH`	Publiceer inhoud in een MQTT-onderwerp (dwz plaats een bericht).
`WEB MQTT SUBSCRIBE`	Abonneer u op een MQTT-onderwerp (dwz berichten ophalen).
`WEB MQTT UNSUBSCRIBE`	Afmelden voor een MQTT-onderwerp.
`WEB MQTT CLOSE`	Sluit de MQTT-verbinding.

Ping

De WebMite reageert op een ping-bericht, zodat u kunt controleren of deze leeft en toegankelijk is. Als er verbinding mee is op het openbare internet kan een gratis dienst zoals: https://uptimerobot.com/ worden gebruikt om u te waarschuwen als deze niet meer actief is.

Audio streamen

De opdrachten **WEB OPEN TCP STREAM** en **WEB TCP CLIENT STREAM** kunnen samen met het **PLAY STREAM** commando implementeert heel eenvoudig een basisinternetradiomogelijkheid. Dit wordt gedemonstreerd in de code hieronder ontvangt het Britse programma ClassicFM.
Let op: Voor dit programma is een **VS1053** met audiocodec vereist.

```
Option Escape
Option Default None
' Create the request for the radio site (ClassicFM).
Dim a$ = "ice-the.musicradio.com"
Dim q$ = "GET "
Inc q$, "/ClassicFMMP3"
Inc q$, " HTTP/1.1\r\n"
Inc q$, "Host: "
Inc q$, a$
Inc q$, "\r\nConnection: close\r\n\r\n"
' Create a circular buffer for reading the internet stream and read and
' write pointers.
Dim buff%(4095), w%, r%
' Configure the VS1053 and tell it to play from the circular buffer.
Play Stream buff%(), r%, w%
' Open the internet radio site.
WEB Open TCP Stream a$, 80
' Send the request to start the stream using the circular buffer specified.
WEB TCP CLIENT STREAM q$, buff%(), r%, w%
' Sit back and listen.
Do : Pause 500 : Loop
```

Long Strings

Lange reeksen zijn een reeks opdrachten en functies waarmee MMBasic onbeperkte reeksen kan manipuleren lengte en zijn vooral handig bij het omgaan met gegevens die via WiFi en internet worden verzonden. Standaard Strings in MMBasic zijn beperkt tot een maximale lengte van 255 tekens. Lange tekenreeksen dupliceren deze functies, maar zullen dat wel doen werken met teken-reeksen van elke lengte, alleen beperkt door de hoeveelheid beschikbaar RAM.

Variabelen met Long String

Variabelen voor het vasthouden van lange tekenreeksen moeten worden gedefinieerd als arrays met gehele getallen. De lange reeksroutines houden zich niet aan getallen in deze arrays, maar gebruik ze gewoon als geheugenblokken voor het bewaren van lange reeksen.
Bij het maken van deze arrays moeten ze worden gedefinieerd als enkelvoudige integer-arrays met het aantal elementen ingesteld op het aantal tekens dat nodig is voor de maximale tekenreeks-lengte gedeeld door acht. De reden voor delen door acht betekent dat elk geheel getal in een MMBasic-array acht bytes in beslag neemt.

Het volgende is een voorbeeld van het declareren van drie lange reeksvariabelen die zullen worden gebruikt om **maximaal 2048** vast te houden karakters in elk:

```
CONST MaxLen = 2048
DIM INTEGER Str1(MaxLen / 8), Str2(MaxLen / 8), Str3(MaxLen / 8)
```

Deze zullen bij het aanmaken lege strings bevatten (dwz hun lengte zal nul zijn). Wanneer deze variabelen worden doorgegeven bij de lange reeksfuncties moeten ze worden ingevoerd als de naam van de variabele, gevolgd door lege haakjes.

Voorbeeld:

```
LONGSTRING COPY Str1(), Str2()
```

Lange reeksvariabelen kunnen als argumenten worden doorgegeven aan door de gebruiker gedefinieerde subroutines en functies. Bijvoorbeeld:

```
Sub MySub longarg() AS INTEGER
  PRINT "Lange tekenreekslengte is" LLEN(longarg())
END SUB
```

En het zou zo kunnen heten:

```
MySub str1()
```

Samenvatting van de Commando's en Functies

Deze worden gedetailleerd gedocumenteerd in de secties Commando's en Functies van deze handleiding. De commando's zijn:

```
LONGSTRING AES128 ENCRYPT/DECRYPT    Codeert of decodeert een lange reeks.
LONGSTRING APPEND array%(), String$ Voeg een gewone string toe aan een lange string.
LONGSTRING BASE64 ENCODE/DECODE      Codeert of decodeert een lange string met basis 64.
LONGSTRING CLEAR array%()            Wis een lange tekenreeks (dwz ingesteld op leeg).
LONGSTRING COPY dest%(), src%()      Kopieer een lange string.
LONGSTRING CONCAT dest%(), src%()    Voeg twee lange strings samen.
LONGSTRING LCASE array%()            Converteert een lange tekenreeks naar kleine letters.
LONGSTRING LEFT dest%(),src%(),nbr   Haal het linker aantal karakters op uit een lange string.
LONGSTRING LOAD array%(),nbr,string$ Kopieer tekens naar een lange tekenreeks.
LONGSTRING MID dest%(),src%(),start,nbr  Haal tekens op uit het midden van een tekenreeks.
LONGSTRING PRINT [#n,] src%()        Druk een lange string af.
LONGSTRING REPLACE array%(),string$,start Vervang tekens in een lange tekenreeks.
LONGSTRING RESIZE addr%(),nbr        Stel de lengte van een lange string in.
LONGSTRING RIGHT dest%(),src%(),nbr Haal het juiste aantal tekens uit een lange reeks.
LONGSTRING SETBYTE addr%(),nbr,data Stel een byte in een lange string in.
LONGSTRING TRIM array%(), nbr        Knip tekens vanaf de linkerkant van een lange reeks af.
LONGSTRING UCASE array%()            Converteert een lange tekenreeks naar hoofdletters.
```

De functies zijn:

```
 r = LGETBYTE(array%(), n)     Geeft de waarde van een byte in een lange tekenreeks terug.
r$ = LGETSTR$(array%(), start, lengte)          Geeft een deel van een lange tekenreeks
                                                terug als een normale tekenreeks.
 r = LINSTR(array%(), search$ [,start][,size]) Geeft de positie van een string in een
                                                lange string terug.
 r = LLEN(array%())            Geeft de lengte van een lange tekenreeks terug.
MATH(BASE64 ENCODE/DECODE)     Codeert of decodeert gegevens met basis 64.
```

Geavanceerde grafische programmeertechnieken

Bij het programmeren met behulp van de geavanceerde **GUI** commando's zijn er een aantal hints en technieken waarmee u rekening moet houden dat maakt het gemakkelijker om uw programma te ontwikkelen en te onderhouden.

De gebruiker moet de controle hebben

Traditionele, op karakters gebaseerde programma's hebben normaal gesproken de controle over de interactie met de gebruiker. Bijvoorbeeld de programma kan een menu weergeven en de gebruiker vragen om een actie te selecteren. Als de gebruiker een ongeldige optie selecteert, programma zou een foutmelding weergeven en de menu-opties opnieuw weergeven.

Grafische programma's zoals die gemaakt met behulp van de geavanceerde GUI-commando's zijn echter anders.
Gewoonlijk begint het programma gewoon te draaien wat het normaal doet (bijv. temperatuur, snelheid, enz. regelen) en het is de taak van de gebruiker om parameters te selecteren en te wijzigen zonder dat daarom wordt gevraagd. Dit is een andere manier van programmeren en is het voor de traditionele programmeur vaak moeilijk om aan deze andere techniek te wennen.

Beschouw als voorbeeld een programma dat een snij-inrichting moet aansturen. Het traditionele programma zou vragen de gebruiker voor de snelheid en snijtijd. Als beide zijn ingevoerd, vraagt het programma om de snij cyclus. Een grafisch programma zou echter twee nummervakken weergeven waar de gebruiker de snelheid en tijd samen met een run-knop invoerd. De nummervakken kunnen worden gevuld met standaardwaarden en de run-knop zou worden uitgeschakeld als de ge-bruiker een ongeldige snelheid of tijd invoerde. Wanneer de run-knop wordt aangeraakt, snijcyclus zou beginnen.

Een goed voorbeeld van dit type grafische interface is het dialoogvenster dat wordt gebruikt op een Windows/IOS/Android computer om de tijd en datum in te stellen. Het toont een aantal vakjes waar de gebruiker de datum/tijd kan invoeren met een OK-knop die het programma vertelt om de inge-voerde gegevens te accepteren.

De gebruiker wordt op geen enkel moment gedwongen om een keuze uit een menu. Ook wordt de huidige tijd/datum al weergegeven in de invoervakken, zodat de gebruiker ze kan accepteren als standaard als ze dat willen.

Als je wat inspiratie nodig hebt over hoe je grafische programma eruit moet zien en aanvoelt, kijk dan in de dichtstbijzijnde GUI gebaseerd besturingssysteem om te zien hoe ze werken.

Programmastructuur

Normaal gesproken zou een programma beginnen met het definiëren van de bedieningselementen (die MMBasic op het scherm zal tekenen), en dan zou de standaardinstellingen instellen en uiteindelijk zou het in een continue lus vallen waar het zou doen waarvoor het ook is ontworpen om te doen. Neem bijvoorbeeld het geval van een eenvoudige controller voor de motor snelheid waarbij de gebruiker de motor laat lopen door op een schermknop te drukken.

Om deze functie te implementeren ziet het programma er ongeveer zo uit:

```
GUI CAPTION #1, "Snelheid (rpm)", 100, 50 ' Label het nummervak.
GUI NUMBERBOX #2, 100, 100, 150, 40   ' Definieer en teken het nummervak.
CtrlVal(#2) = 100                     ' Standaardwaarde voor de snelheid.
GUI BUTTON #3, "RUN", 100, 100, 30, 30, RGB(red)
                                      ' Definieer en teken de RUN-knop.
DO                                    ' Dit loopt voor altijd in een lus.
  IF CtrlVal(#3) < 10 OR CtrlVal(#3) > 100 THEN
                                      ' Controleer de snelheidsinstelling.
    GUI DISABLE #3                    ' RUN uitschakelen als het ongeldig is.
  ELSE                                ' Anders.
    GUI ENABLE #3                     ' Activeer de RUN-knop.
  ENDIF
  IF CtrlVal(#3) = 1 THEN             ' Als de knop wordt ingedrukt.
    SetMotorSpeed CtrlVal(#2)         ' Laat de motor draaien.
  ELSE                                ' Anders is de knop omhoog.
    SetMotorSpeed 0                   ' Zet daarom het motortoerental op nul.
  ENDIF
LOOP
```

Merk op dat de gebruiker niet wordt gevraagd om iets te doen; het programma zit gewoon in een lus en reageert op de veranderingen die de gebruiker heeft gedaan aan de bedieningselementen (d.w.z. de gebruiker heeft de controle).

Ongeldige bedieningselementen uitschakelen

Zoals in het bovenstaande voorbeeld, zal het uitschakelen van een besturingselement voorkomen dat een gebruiker het gebruikt en zal MMBasic het opnieuw tekenen in een doffe kleur om aan te geven dat het niet beschikbaar is. Dit is het equivalent van een foutmelding in een traditionele tekst
gebaseerd programma en is gebruiksvriendelijker dan het verschijnen van een berichtvenster dat eerder moet worden gesloten iets anders kan worden gedaan.

Het komt vaak voor dat een controle ongeldig kan zijn, bijvoorbeeld wanneer een invoer niet gereed is of gewoon wanneer een optie of actie is niet van toepassing. Later, wanneer het besturings-element geldig wordt, kunt u de **GUI ENABLE** gebruiken commando om het weer in gebruik te nemen. Een ander voorbeeld is wanneer een **GUI NUMBERBOX** toetsenbord wordt weergegeven MMBasic schakelt automatisch alle andere bedieningselementen op het scherm uit, zodat het voor de gebruiker duidelijk is waar zijn invoer is vereist.

Als u een bedieningselement uitschakelt, blijft het op het scherm staan, zodat de gebruiker weet dat het er is, maar het wordt gedimd en reageert niet op aanraking. Niet reageren op aanraking betekent ook dat de gebruiker het niet kan veranderen en een interrupt wordt niet gegenereerd wanneer deze wordt aangeraakt. Dit is handig voor jou als programmeur, want dat hoeft niet controleer of de con-trole geldig is voordat u ernaar handelt.

Constanten gebruiken voor controlereferentienummers

De geavanceerde besturingen gebruiken een referentienummer om de besturing te identificeren. Om het gemakkelijk te lezen en te onderhouden uw programma moet u deze getallen definiëren als constanten met gemakkelijk te herkennen namen.

In het volgende programmafragment is **MAIN_SWITCH** bijvoorbeeld gedefinieerd als een constante en deze constante is gebruikt waar het referentienummer voor die controle vereist is:

```
CONST MAIN_SWITCH = 5
CONST ALARM_LED   = 6
'...
GUI SWITCH MAIN_SWITCH,"ON|OFF",330, 50, 140, 50, RGB(white), RGB(blue)
GUI LED ALARM_LED, 215, 220,30, RGB(red)
'...
IF CtrlVal(MAIN_SWITCH) = 0 THEN
'... Zet bijvoorbeeld de pomp uit.
IF ALARM THEN CtrlVal(ALARM_LED) = 1
```

Het is veel gemakkelijker om te onthouden wat **MAIN_SWITCH** doet dan te onthouden naar welk besturingselement het getal 5 verwijst. Als je veel bedieningselementen hebt, is het ook veel gemak-kelijker om de bedieningselementen opnieuw te nummeren als al hun nummers zijn gedefinieerd op de ene plaats aan het begin van het programma.

Het referentienummer moet een getal zijn tussen 1 en de waarde die is ingesteld met het **OPTION CONTROL** commando.
Als u dit aantal verhoogt, verbruikt u meer RAM en als u dit verlaagt, wordt wat RAM hersteld.

Het hoofdprogramma loopt nog

Het is belangrijk om te beseffen dat uw belangrijkste BASIC-programma nog steeds actief is terwijl de gebruiker interactie heeft met de **GUI** bedieningen. Het blijft bijvoorbeeld draaien, zelfs als een gebruiker een schakelaar op het scherm ingedrukt houdt en het blijft draaien terwijl het virtuele toetsenbord wordt weergegeven als gevolg van het aanraken van een **TEXTBOX** bediening.

Om deze reden zou uw hoofdprogramma niet willekeurig de aanraakgevoelige schermbedieningen moeten bijwerken, omdat ze kan het beeld op het scherm veranderen terwijl de gebruiker ze gebruikt (met ongedefinieerde resultaten). Normaal gesproken wanneer een BASIC-programma dat **GUI** besturingselementen gebruikt, start het initialiseert besturingselementen zoals een **SPINBOX**, **NUMBERBOX** en **TEXTBOX** naar een initiële waarde, maar vanaf dat moment zou het hoofdprogramma alleen de waarde hiervan moeten lezen bedieningselementen - het is de verantwoordelijk-heid van de gebruiker om deze te wijzigen, niet uw programma.

Als u echter de waarde van zo'n bedieningselement op het scherm wilt wijzigen, heeft u een mecha-nisme nodig om te voorkomen dat zowel het programma als de gebruiker tegelijkertijd een wijzi-ging doorvoeren. Een methode is om een vlag in te stellen binnen de toets omlaag onderbreken om aan te geven dat de besturing gedurende deze tijd niet moet worden bijgewerkt. Deze vlag kan dan worden gewist in de toetsomhoog-onderbreking, zodat het hoofdprogramma het updaten van de besturing kan hervatten.
Merk op dat deze discussie alleen van toepassing is op bedieningselementen die reageren op aanraking. Bedieningselementen zoals **CAPTION** en **LED** kunnen op elk moment worden gewijzigd door het hoofdprogramma en zijn dat vaak ook.

Interrupts en SELECT CASE statements gebruiken

Alles wat er op een scherm gebeurt met behulp van de geavanceerde bedieningselementen, wordt gesignaleerd door een onderbreking, ofwel tik op naar beneden of naar boven halen. Dus als u direct iets wilt doen wanneer een besturingselement wordt gewijzigd, moet u dat doen in een onder- breking. Meestal zult u geïnteresseerd zijn in wanneer de aanraking (of pen) is uitgeschakeld, maar in sommige gevallen zou u ook willen weten wanneer het uitkomt.

Omdat de onderbreking wordt geactiveerd wanneer de pen een bedieningselement of een deel van het scherm aanraakt dat u moet ontdekken welke controle werd aangeraakt. Dit kunt u het beste doen met de **TOUCH(REF)** functie en de **SELECT CASE** verklaring.

In het volgende fragment wordt bijvoorbeeld de subroutine PenDown aangeroepen als er een aanraking is en de functie **TOUCH(REF)** retourneert het referentienummer van de regelaar die wordt aangeraakt. De **SELECT** gebruiken GEVAL zal de alarm-LED aan of uit gaan, afhankelijk van welke knop wordt aangeraakt. De actie kan elke zijn een aantal dingen zoals het verhogen van een I/O-pin om een fysieke sirene in te schakelen of het afdrukken van een bericht op de console.

```
CONST ALARM_ON  = 15
CONST ALARM_OFF = 16
CONST ALARM_LED = 33
GUI INTERRUPT PenDown
'...
GUI BUTTON ALARM_ON, "ALARM AAN",330,50,140,50, RGB(white),RGB(blue)
GUI BUTTON ALARM_OFF, "ALARM UIT",330,150,140,50, RGB(white),RGB(blue)
GUI LED ALARM_LED, 215, 220, 30, RGB(red)
'...
DO : LOOP ' Het hoofdprogramma doet iets.

' Deze sub wordt gebeld wanneer aanraking wordt gedetecteerd.
SUB PenDown
  SELECT CASE TOUCH(REF)
    CASE ALARM_ON
      CtrlVal(ALARM_LED) = 1
    CASE ALARM_OFF
      CtrlVal(ALARM_LED) = 0
  END SELECT
END SUB
```

De **SELECT CASE** kan ook testen op andere controles en de acties uitvoeren die daarvoor nodig zijn in hun eigen gedeelte van de **CASE** verklaring.
Het belangrijkste punt is dat het onderhoud van de bedieningselementen (bijvoorbeeld reageren op de knoppen en het werken van het alarm, LED uit of aan) gebeurt automatisch zonder dat het hoofdprogramma erbij betrokken is – het kan blijven doen iets nuttigs, zoals het berekenen van een besturingsreactie, enz.

Touch-up onderbreken

In de meeste gevallen kunt u alle gebruikersinvoer verwerken in de touch-down-interrupt. Maar er zijn uitzonderingen en een typische een voorbeeld is wanneer u de kenmerken van het bedienings-element dat wordt aangeraakt, moet wijzigen.

Bijvoorbeeld, als je wilde de voorgrondkleur van een knop veranderen van wit naar rood als deze omlaag is. Wanneer het wordt geretourneerd naar de bovenste staat moet de kleur terugkeren naar wit.
Het instellen van de kleur bij het aanraken is eenvoudig. Definieer gewoon een touch-down interrupt en verander de kleur in de onderbreken wanneer dat besturingselement wordt aangeraakt. Om de kleur echter weer wit te maken, moet u detecteren wanneer de touch is verwijderd van de besturing (d.w.z. touch up). Dit kan met een touch-up interrupt.
Om een retoucheeronderbreking te specificeren, voegt u de naam van de subroutine voor deze onderbreking toe aan het einde van de **GUI INTERRUPT** commando. Bijvoorbeeld:

```
GUI INTERRUPT IntTouchDown, IntTouchUp
```

Binnen de touch-up-subroutine kun je dezelfde structuur gebruiken als in de touch-down-sub, maar je moet zoeken naar het referentienummer van de laatste controle die werd aangeraakt. Dit komt omdat de aanraking het scherm al heeft verlaten en er wordt momenteel geen bediening aangeraakt. Om het nummer van de laatst aangeraakte knop te krijgen, moet u gebruiken de functie:

```
TOUCH(LASTREF)
```

Het volgende voorbeeld laat zien hoe u aan de bovenstaande eis kunt voldoen en zowel een touch-down kunt implementeren en een retoucheeronderbreking:

```
SUB IntTouchDown
  SELECT CASE TOUCH(REF)
    CASE ButtonRef
      GUI FCOLOUR RGB(RED), ButtonRef
  END SELECT
END SUB

SUB IntTouchUp
  SELECT CASE TOUCH(LASTREF)
    CASE ButtonRef
      GUI FCOLOUR RGB(WIT), ButtonRef
  END SELECT
END SUB
```

Houd onderbrekingen erg kort

Omdat een touch-interrupt een verzoek van de gebruiker aangeeft, is het verleidelijk om uitgebreid te programmeren binnen een onderbreking. Als de aanraking bijvoorbeeld aangeeft dat de gebruiker een bericht naar een ander wil sturen, controller klinkt het logisch om al die code in de interrupt te plaatsen. Maar dit is geen goed idee omdat MMBasic kan niets anders doen terwijl uw programma de onderbreking verwerkt en het verzenden van een bericht kan vele milliseconden duren.

In plaats daarvan zou je programma een globale variabele moeten updaten om aan te geven wat er gevraagd wordt en de werkelijke uitvoering naar het hoofdprogramma. Als de gebruiker bijvoorbeeld de knop "een bericht verzenden" heeft aangeraakt, is uw programma kan eenvoudig een globale variabele op true zetten. Dan kan het hoofdprogramma deze variabele monitoren en als deze verandert voer de logica en communicatie uit die nodig zijn om aan het verzoek te voldoen. Denk aan het gebod "Gij zult niet onderbreken".

Meerdere schermen

Uw programma heeft mogelijk een aantal schermen nodig met verschillende bedienings-elementen op elk scherm. Dit zou kunnen zijn geïmplementeerd door de oude bedieningselementen te verwijderen en nieuwe te maken wanneer het scherm wordt omgeschakeld. Maar op een andere manier om dit te doen, gebruikt u de opdrachten **GUI SETUP** en **PAGE**. Hiermee kunt u de bedienings-elementen op pagina's ordenen en met één simpele opdracht kun je van pagina wisselen. Alle bedie-ningselementen op de oude pagina worden automatisch verborgen en bedieningselemen-ten op de nieuwe pagina worden automatisch weergegeven.
Om besturingselementen aan een pagina toe te wijzen, gebruikt u de **GUI SETUP nn** opdracht waarbij nn verwijst naar de pagina in het bereik van **1** t/m **32**. Wanneer u deze opdracht heeft gebruikt, worden alle nieuw gemaakte bedieningselementen aan die pagina toegewezen. Jij kan gebruik **GUI SETUP** zo vaak als u wilt. Bijvoorbeeld in het programmafragment onder de eerste twee bedieningselementen worden toegewezen aan pagina 1, de tweede aan pagina 2, enz.

```
GUI SETUP 1
GUI Caption #1, "Flow Rate", 20, 170,, RGB(brown), 0
GUI Displaybox #2, 20, 200, 150, 45

GUI SETUP 2
GUI Caption #3, "Hoog:", 232, 260, LT, RGB(yellow)
GUI Nummerbox #4, 318, 6, 90, 12, RGB(yellow), RGB(64, 64, 64)

GUI SETUP 3
GUI Checkbox #5, "Alarmen", 500, 285, 25
GUI Checkbox #6, "Waarschuwingen", 500, 325, 25
```

Standaard worden alleen de instellingen van de bedieningselementen zoals pagina 1 weergegeven en de andere worden verborgen.
Om het scherm naar pagina 3 te schakelen, hoeft u alleen maar het commando **PAGE 3** te gebruiken. Dit zorgt ervoor dat bedieningselementen **#1** en **#2** automatisch worden verborgen en bedieningselementen **#5** en **#6** worden weergegeven. Evenzo zal **PAGE 2** alles verbergen behalve **#3** en **#4** die worden weergegeven.

U kunt meerdere pagina's specificeren om tegelijkertijd weer te geven, **PAGE 1, 3** zal bijvoorbeeld zowel pagina's 1 als 3 weergeven terwijl u pagina 2 verbergt. Dit kan handig zijn als u een set bedieningselementen hebt die altijd zichtbaar moeten zijn. Bijvoorbeeld, **PAGE 1, 2** en **PAGE 1, 3** laten de bedieningselementen op pagina 1 zichtbaar terwijl de andere worden in- en uitgeschakeld.

Het is volkomen legaal voor een programma om bedieningselementen op andere pagina's te wijzigen, ook al worden ze niet weergegeven op de tijd. Dit omvat het wijzigen van de waarde en kleuren en het uitschakelen of verbergen ervan. Wanneer het display is overgeschakeld naar hun pagina zullen de bedieningselementen worden weergegeven met hun nieuwe attributen.

Het is mogelijk om de **PAGE** commando's in de touch-down-interrupt te plaatsen, zodat het indrukken van een bepaalde knop of onderdeel van het scherm zal overschakelen naar een andere pagina.

Merk op dat wanneer **ALL** wordt gebruikt voor de lijst met bedieningselementen in opdrachten zoals **GUI ENABLE ALL** dit alleen verwijst naar de bedieningselementen op de pagina's die momenteel zijn geselecteerd voor weergave. De bedieningselementen op andere pagina's blijven ongewijzigd.

Alle programma's starten met het equivalent van de commando's **GUI SETUP 1** en **PAGE 1** die van kracht zijn. Dit betekent dat als de **GUI SETUP**- en **PAGE** commando's niet worden gebruikt, zal het programma werken zoals je zou verwachten bij alles bedieningselementen weergegeven.
Hieronder wordt een typisch gebruik van het **PAGE** commando weergegeven.

Met twee knoppen (die altijd worden weergegeven) kan de gebruiker kiezen tussen de eerste pagina en de tweede pagina.
De schakelaar wordt gedaan in de touch-down-interrupt.

```
GUI SETUP 1
GUI Button #10, "SELECT PAGE EEN",50,100,150,30, RGB(yellow), RGB(blue)
GUI Button #11, "SELECT PAGE TWEE",50,140,150,30, RGB(yellow),RGB(blue)

GUI SETUP 2
GUI Caption #1, "Eerste pagina weergeven", 20, 20

GUI SETUP 3
GUI Caption #2, "Weergeven tweede pagina", 20, 50

Page 1, 2
GUI INTERRUPT TouchDown
Do

Loop

Sub TouchDown
  If Touch(REF) = 10 Then Page 1, 2
  If Touch(REF) = 11 Then Page 1, 3
End sub
```

Meerdere Interrupts

Met veel schermpagina's kan de interrupt-subroutine lang en ingewikkeld worden. Om daar om heen te werken is het mogelijk om meerdere interrupt-subroutines te hebben en er dynamisch tussen te schakelen zoals u wilt (normaal gesproken) na het wisselen van pagina). Dit wordt gedaan door de huidige interruptroutines opnieuw te definiëren met behulp van de **GUI INTERRUPT** opdracht.

Dit programmafragment gebruikt bijvoorbeeld verschillende interruptroutines voor pagina's 4 en 5 en deze zijn gespecificeerd onmiddellijk na het wisselen van pagina's.

```
PAGE 4
GUI INTERRUPT P4keydown, P4keyup
...
PAGE 5
GUI INTERRUPT P5keydown, P5keyup
...
```

Basic tekenopdrachten gebruiken

Er zijn twee soorten objecten die op het scherm kunnen staan. Dit zijn de **GUI** bedieningselementen en de basis tekening objecten (**PIXEL**, **LINE**, **TEXT**, enz.). Het mixen van de twee op het scherm is geen goed idee, omdat MMBasic dat wel doet de positie van de basistekenobjecten niet volgen en ze kunnen botsen met de **GUI** besturingselementen.

Als gevolg hiervan, tenzij u bereid bent om wat extra programmering te doen, moet u ofwel de **GUI** bedieningselementen gebruiken of de basistekenobjecten – maar u moet niet beide gebruiken. Gebruik dus bijvoorbeeld geen **TEXT** maar gebruik **GUI CAPTION** in plaats daarvan. Als u alleen **GUI** besturingselementen gebruikt, zal MMBasic het scherm voor u beheren, inclusief wissen en indien nodig opnieuw tekenen, bijvoorbeeld wanneer een virtueel toetsenbord wordt weergegeven.

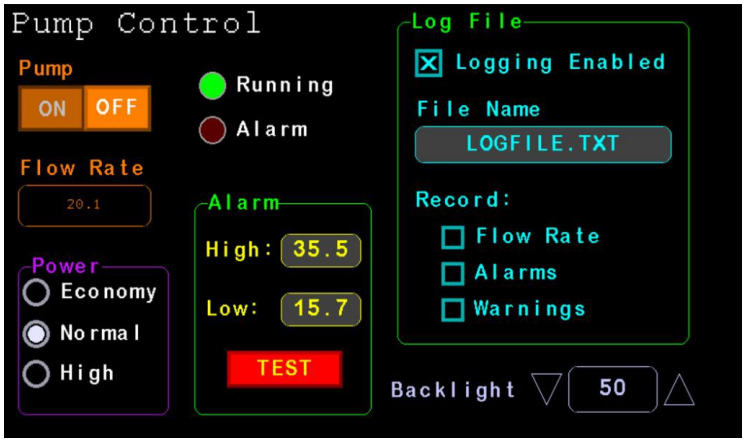
Merk op dat het **CLS**-commando (gebruikt om het scherm leeg te maken) automatisch alle **GUI**-bedieningselementen op het scherm instelt op verborgen (d.w.z. het doet een **GUI HIDE ALL** voordat het scherm wordt gewist).

Het grootste probleem met het mixen van basisafbeeldingen en **GUI** besturingselementen doet zich voor met het tekstvak, het geformatteerde vak en Number Box-besturingselementen die een virtueel toetsenbord weergeven. Dit kan alle basisafbeeldingen wissen en MMBasic zal dat doen weet niet om ze te herstellen wanneer het toetsenbord wordt verwijderd. Als u basisafbeeldingen wilt mixen met GUI-besturingselementen je zou moeten:

- Onderschep de touch-down interrupt voor de Text Box, Formatted Box en Number Box controls als dat geeft aan dat er een virtueel toetsenbord wordt weergegeven en dat geeft u de mogelijkheid om opnieuw te tekenen uw niet-GUI-basisafbeeldingen in afwachting van deze gebeurtenis (teken ze bijvoorbeeld gedimd om verschijnen alsof ze zijn uitgeschakeld).
- Onderschep de touch-up-interrupt voor dezelfde bedieningselementen als die aangeeft dat het virtuele toetsenbord is geweest verwijderd en u kunt vervolgens alle niet-GUI-afbeeldingen in hun oorspronkelijke staat opnieuw tekenen.

Overlappende bedieningselementen

Bedieningselementen kunnen zo worden gedefinieerd dat ze elkaar overlappen op het display, dit gebeurt meestal met **GUI AREA** die, als voorbeeld, misschien wil je een aanraking vastleggen die bedoeld was voor (bijvoorbeeld) een **GUI BUTTON**. Hiermee creëer je uw eigen animatie voor de knop in plaats van die van MMBasic. In dit geval de controle die u wenst om op aanraking te reageren (d.w.z. **GUI AREA**) moet een lager referentienummer (d.w.z. #ref) hebben dan het besturingselement dat het dekt (d.w.z. de **GUI BUTTON**). Dit komt omdat wanneer het scherm wordt aangeraakt, MMBasic de huidige lijst van actieve bedieningselementen beginnend met controlenummer 1 en werkend naar boven. Wanneer een match is gemaakt MMBasic zal de juiste actie ondernemen en de zoekopdracht beëindigen. Dit resulteert in de lager genummerde controle effectief maskeren van een hoger genummerd besturingselement dat hetzelfde schermgebied bestrijkt als de aangeraakte locatie.



MMBasiskenmerken

Naamgevingsconventies

Commandonamen, functienamen, labels, variabelenamen, enz. zijn niet hoofdlettergevoelig, zodat **"Run" en "RUN"** zijn equivalent en **"dOO" en "Doo"** verwijzen naar dezelfde variabele**.**
Het type van een variabele kan worden gespecificeerd in het **DIM** commando of door een achtervoegsel toe te voegen aan het einde van de variabele naam. Het achtervoegsel voor een geheel getal is bijvoorbeeld '%', dus als er automatisch een variabele met de naam nbr% wordt gemaakt, is dit
een geheel getal. Er zijn drie soorten variabelen:

1. **Drijvende punt**. Deze kunnen een getal met een decimaalteken en een breuk opslaan (bijv. **45.386**) en ook erg grote getallen. Ze verliezen echter hun nauwkeurigheid wanneer er **meer** dan **14** significante **cijfers** worden opgeslagen of gemanipuleerd. Het achtervoegsel is '**!**' en drijvende komma is de standaard wanneer een variabele wordt gemaakt zonder achtervoegsel.

2. **64-bits geheel getal**. Deze kunnen nummers met maximaal 19 decimale cijfers opslaan zonder de nauwkeurigheid te verliezen, maar ze kan geen breuken opslaan (d.w.z. het deel na de komma). Het achtervoegsel voor een geheel getal is **'%**'.

3. **strings**. Deze zullen een reeks karakters opslaan (bijvoorbeeld "Tom"). Het achtervoegsel voor een string is het **'$**'-symbool (bijv. **naam$**, **s$**, enz.) Tekenreeksen kunnen tot 255 tekens lang zijn.

Namen en labels van variabelen kunnen beginnen met een alfabetisch teken of onderstrepingsteken en kunnen elk alfabetisch . bevatten of numeriek teken, de punt (.) en het onderstrepingsteken (_). Ze mogen maximaal 32 tekens lang zijn. Een variabele naam of een label mag niet hetzelfde zijn als een commando of een functie of een van de volgende trefwoorden:
`THEN`, `ELSE`, `TO`, `STEP`, `FOR`, `NEXT`, `WHILE`, `UNTIL`, `MOD`, `NOT`, `AND`, `OR`, `XOR`, `AS`.
Bijv. `Step = 5` is illegaal.

Constanten

Numerieke constanten kunnen beginnen met een numeriek cijfer (**0-9**) voor een decimale constante, **&H** voor een hexadecimaal constante, **&O** voor een octale constante of **&B** voor een binaire constante. Bijvoorbeeld **&B1000** is hetzelfde als de decimale constante **8**. Constanten die beginnen met **&H**, **&O** of **&B** worden altijd behandeld als **64-bits** integerconstanten.

Decimale constanten kunnen worden voorafgegaan door een min **(-)** of plus **(+)** en kunnen worden afgesloten met **'E'** gevolgd door een exponentnummer om exponentiële notatie aan te duiden. Bijvoorbeeld **1.6E+4** is hetzelfde als **16000**.

Als de decimale constante een decimaalteken of een exponent bevat, wordt deze behandeld als een drijvende-kommaconstante anders wordt het behandeld als een **64-bits integer** constante.
Stringconstanten worden omgeven door dubbele aanhalingstekens (**"**).
Bijvoorbeeld: `"Hallo wereld"`**.**

Uitvoeringskenmerken

- Maximaal programma - zie tabel. Houd er rekening mee dat MMBasic het programma tokeniseert wanneer het in flash wordt opgeslagen, waardoor de uiteindelijke grootte in flash kan afwijken van de grootte van de platte tekst.
- De maximale lengte van een **opdrachtregel** is **255** tekens.
- De maximale lengte van een **variabelenaam** of een label is **31** tekens.
- Het maximale aantal dimensies voor een **array** is **5**.
- Het maximum aantal **argumenten voor opdrachten** die een variabel aantal arg accepteren, is - zie tabel.
- Het maximum aantal geneste **FOR…NEXT** lussen is **20**.
- Het maximum aantal geneste **DO . . . LOOP** commando's is **20**.
- Het maximum aantal geneste **GOSUB**'s, **subroutines** en **functies** (gecombineerd) is **320**.
- Het maximum aantal geneste multiline **IF...ELSE...ENDIF** opdrachten is **20**.
- Maximaal aantal door de gebruiker gedefinieerde **labels**, **sub's** en **functies** (gecombineerd) - zie tabel.
- Maximaal aantal **Interrupt-pinnen** dat kan worden geconfigureerd: **10**
- Getallen worden opgeslagen en gemanipuleerd als **Floating** met dubbele precisie of **64-bits Integers** met teken. De bereik van getallen met drijvende komma is **1.797693134862316e+308** tot **2.225073858507201e-308**.
- Het bereik van **64-bits** gehele getallen (hele getallen) dat kan worden gemanipuleerd is **± 9223372036854775807.**
- De maximale **tekenreekslengte** is **255** tekens.
- Het maximale **regelnummer** is **65000**.
- Het maximum aantal **achtergrondpulsen** dat door het **PULSE** commando wordt gelanceerd, is **5**.
- Maximum aantal **globale variabelen en constanten** is - zie tabel.
- Maximaal aantal **lokale variabelen** is - zie tabel.
- Het **maximale aantal bestanden** dat kan worden weergegeven met de opdracht **FILES** is **1000.**
- De **maximale lengte** van de ondersteunde **bestandsnaam** is **63** tekens.

- TABEL - Editie: V6.00.02RC19	Maximale Programma	Maximale Frequetie	Max. aantal Subs, Funct.	Max. Array Omvang	Max. aantal Globale Variabelen	Max. aantal Locale Varia.
PicoMite RP2040	128Kb	420MHz	256	6	256	256
PicoMite USB RP2040	128Kb	420MHz	256	6	256	256
PicoMite VGA RP2040	100Kb	378MHz	256	6	256	256
PicoMite VGA USB RP2040	100Kb	378MHz	256	6	256	256
WebMite RP2040	88Kb	252MHz	256	6	240	240
PicoMite RP2350	288Kb	396MHz	512	5	384	384
PicoMite USB RP2350	288Kb	396MHz	512	5	384	384
PicoMite VGA RP2350	184Kb	378MHz	512	5	384	384
PicoMite VGA USB RP2350	184Kb	378MHz	512	5	384	384
PicoMite HDMI RP2350	184Kb	372MHz	512	5	384	384
PicoMite HDMI USB RP2350	184Kb	372MHz	512	5	384	384
WebMite RP2350	208Kb	252MHz	512	5	384	384

Compatibiliteit

MMBasic implementeert een grote subset van Microsoft's GW-BASIC. Er zijn tal van verschillen vanwege fysieke en praktische overwegingen, maar de meeste standaard BASIC-commando's en -functies zijn in wezen de zelfde. Een online handleiding voor GW-BASIC is beschikbaar op http://www.antonis.de/qbebooks/gwbasman/index.html en dit geeft een meer gedetailleerde beschrijving van de commando's en functies.
MMBasic implementeert ook een aantal moderne programmeerstructuren die zijn gedocumenteerd in de ANSI-standaard voor: Volledig BASIC (X3.113-1987) of ISO/IEC 10279:1991. Deze omvatten **SUB**/**END SUB**, de **DO WHILE** … **LOOP**, de **SELECT**…**CASE** -statements en gestructureerde **IF** .. **THEN** … **ELSE** … **ENDIF** -statements.

Vooraf gedefinieerde Alleen Lezen Variabelen

Deze variabelen worden ingesteld door MMBasic en kunnen niet worden gewijzigd door het lopende programma.

MM.XXX	Beschrijving 139
MM.ADDRESS$	**ALLEEN WEBMITE VERSIE** **Deze variabele retourneert het IP-adres van de afzender van het laatst ontvangen UDP-datagram.**
MM.CMDLINE$	**Ondersteuning voor MM.CMDLINE$** in alle versies in alle versies.
MM.DEVICE$	**Een tekenreeks die het apparaat of platform vertegenwoordigt waarop MMBasic wordt uitgevoerd.** Momenteel bevat deze variabele een van de volgende: "Maximite" Op de standaard Maximite en compatibles. "Colour Maximite" Op de Color Maximite en UBW32. "Colour Maximite 2" Op de Color Maximite 2 "DuinoMite" Wanneer u draait op een van de DuinoMite "Generic PIC32" Voor de MMBasic op een PIC32. "Micromiet" Op de PIC32MX150/250 "Micromite MkII" Op de PIC32MX170/270 "Micromite Plus" Op de PIC32MX470 "Micromite Extreme" Op de PIC32MZ-serie "ARMmite H7" Op de ArmmiteH7 "ARMmite F407" Op de ArmmiteF4 "ARMmite L4" Met chipnr. aan ArmmiteL4 toegevoegd. "PicoMite" Op de Raspberry Pi Pico. "PicoMiteVGA" Op de Raspberry Pi Pico VGA editie. "PicoMiteHDMI" Op de Raspberry Pi Pico HDMI editie. "WebMite" Op de Raspberry Pi Pico W editie. Voorbeeld : Print MM.Device$ Retouneert: **PicoMite**
MM.ERRNO MM.ERRMSG$	**Als een instructie een fout veroorzaakte die werd genegeerd, worden deze variabelen ingesteld.** MM.ERRNO is een getal waarbij niet nul betekent dat er was een fout en MM.ERRMSG$ is een tekenreeks die het foutbericht vertegenwoordigt dat normaal gesproken op de console zou zijn weergegeven. Ze worden op nul gezet en een lege string door RUN, ON ERROR **IGNORE** of ON ERROR **SKIP.**
MM.FONTHEIGHT MM.FONTWIDTH	**Gehele getallen die de hoogte en breedte van het huidige lettertype (in pixels) vertegenwoordigen, behouden voor compatibiliteit.** Let op: deze worden automatisch omgezet in MM.INFO**(FONTHEIGHT)** en MM.INFO**(FONTWIDTH)** door MMBasic.
MM.I2C	**Na een I2C schrijf- of leescommando wordt deze integervariabele ingesteld op geef het resultaat van de bewerking als volgt aan:** **0** = De opdracht is foutloos voltooid. **1** = Een **NACK** antwoord ontvangen. **2** = Opdracht time-out.

MM.XXX	Beschrijving
MM.INFO() MM.INFO$()	**Deze twee versies kunnen door elkaar worden gebruikt, maar goede programmering de praktijk zou vereisen dat u degene gebruikt die overeenkomt met de geretourneerde data type.**
MM.INFO(ADC)	**Retourneert het nummer van de buffer die momenteel klaar is om te lezen bij gebruik van ADC.** RUN (1 of 2). Retourneert 0 als er niets gereed is.
MM.INFO(ADC DMA)	**Retourneert waar (1) als de ADC DMA actief is.**
MM.INFO$(AUTORUN)	**Retourneert de instelling van het OPTION AUTORUN commando.**
MM.INFO(BOOT COUNT)	**Retourneert het aantal keren dat de Pico opnieuw is opgestart sinds de flashdrive voor het laatst werd geformatteerd.**
MM.INFO$(CPUSPEED)	**Retourneert de CPU-snelheid als een tekenreeks.**
MM.INFO$(CURRENT)	**Geeft de naam van het huidige programma terug als het vanuit een bestand wordt geladen, of NONE als het wordt aangeroepen na een NEW-, AUTOSAVE-, XMODEM- of EDIT-commando.**
MM.INFO$(LCDPANEL)	**Retourneert de naam van het geconfigureerde LCD-paneel of een lege tekenreeks.**
MM.INFO(LCD320)	**Retourneert true als het beeldscherm geschikt is voor 320x240-weergave met behulp van de OPTION LCD320 commando.**
MM.INFO$(SDCARD)	**Geeft de status van de SD-kaart terug.** Geldige retourzendingen zijn: **DISABLED**, **NOT PRESENT, READY** en **UNUSED**.
MM.INFO(DEVICE)	Het IS mogelijk om de informatie over het apparaat **MM.INFO (DEVICE)** direct bij het opstarten van het systeem weer te geven.
MM.INFO(DISK SIZE)	**Retourneert de capaciteit van de SD-kaart in bytes.**
MM.INFO$(DRIVE)	**Retourneert de actieve schijf “A:” of “B:”**
MM.INFO(EXISTS FILE fname$)	**Retourneert 1 als het bestand filename$ bestaat, anders wordt 0 geretourneerd.**
MM.INFO(EXISTS DIR dname$)	**Retourneert 1 als de directory dirname$ bestaat, anders wordt 0 geretourneerd.**
MM.INFO$(FREE SPACE)	**Retourneert de vrije ruimte op de SD kaart.**
MM.INFO(FILESIZE file$)	**Retourneert de grootte van file$ in bytes of -1 als het niet wordt gevonden, -2 als het een map is.**
MM.INFO(ID)	**Retourneert de unieke ID van de Pico PCB.**
MM.INFO$(MODIFIED file$)	**Geeft de datum/tijd terug waarop bestand$ is gewijzigd, lege tekenreeks indien niet gevonden.**
MM.INFO(FCOLOUR)	**Geeft de huidige voorgrondkleur terug.**
MM.INFO(BCOLOUR)	**Geeft de huidige achtergrondkleur terug.**
MM.INFO(FLASH)	**De waarde die wordt geretourneerd door MM.Info(FLASH)** Identificeert vanuit welk slot het programma wordt uitgevoerd en kan in de bibliotheek worden gebruikt om bibliotheekcode/routines voorwaardelijk op te nemen of uit te sluiten.

MM.XXX	Beschrijving 141
MM.INFO(FONT)	**Retourneert het nummer van het momenteel actieve lettertype.**
MM.INFO(FONT ADDRESS n)	**Retourneert het adres van de geheugenlocatie** met het adres van **FONT n.**
MM.INFO(FONT POINTER n)	**Geeft een POINTER terug naar het begin van FONT n in het geheugen.**
MM.INFO(FONTHIGHT) MM.INFO(FONTWIDTH)	**Gehele getallen die de hoogte en breedte van het huidige lettertype vertegenwoordigen (in pixels).**
MM.INFO$(FLASH)	**Rapporteert uit welke flash-sleuf het programma is geladen,indien van toepassing**.
MM.INFO(FLASH ADDRESS n)	**Retourneert het adres van flash slot n.**
MM.INFO(HEAP)	**Geeft de hoeveelheid MMbasic Heap geheugen vrij**. MMBasic heap wordt gebruikt voor **strings**, **arrays** en diverse andere tijdelijke en permanente **buffers** (bijv. **audio**).
MM.INFO(ID)	**Retourneert de unieke ID van de Pico**.
MM.INFO$(IP ADDRESS)	**Retourneert het IP-adres van de WebMite**.
MM.INFO(HPOS) MM.INFO(VPOS)	**De huidige horizontale en verticale positie (in pixels) na de laatste afbeeldingen of printopdracht.**
MM.INFO(OPTION option) MM.INFO(OPTION ANGLE)	**Retourneert de huidige waarde van een reeks opties die van invloed zijn op hoe een programma zal runnen.** **"optie"** kan een van **AUTORUN, BASE, BREAK, DEFAULT, EXPLICIT.**
MM.INFO$(PATH)	**Geeft het pad van het huidige programma terug, of NONE als het wordt aangeroepen na een NEW- of EDIT commando.**
MM.INFO$(OPTION FLASH SIZE)	**Zou de grootte van het flash-geheugen moeten geven**.
MM.INFO$(DISK SIZE)	**Zou de grootte van de flash drive A: moeten geven of de B: drive indien aanwezig.** Werken op de standaard drive.
MM.INFO$(FREE SPACE)	**Zou de grootte van de flash drive A: moeten geven** of de **B:** drive indien aanwezig. Werken op de standaard drive.
MM.INFO$(PIN pinno) MM.INFO(PINNO)	**Geeft de status van I/O pin 'pinno'.** Geldige retourzendingen zijn: **INVALID, RESERVED, IN USE** en **OFF, DIN, DOUT, AIN, UNUSED**. Waar **OPTION EXPLICIT** gespecificeerd.
MM.INFO$(PINNO GPnn)	**Geeft het fysieke pinnummer voor een bepaald GP-nummer.** **> ? mm.info(pinno GP28) of > ? mm.info(pinno "GP28")** **> a$="GP28":? mm.info(pinno a$)** v.a. V5.07.05RC2. **MM.INFO(PINNO ...)** waar **OPTION EXPLICIT** gespecificeerd**.**
MM.INFO(PIO RX DMA)	**Geeft aan of het PIO RX DMA kanaal bezet is.**
MM.INFO(PIO TX DMA)	**Geeft aan of het PIO TX DMA kanaal bezet is.**
MM.INFO$(PLATFORM)	**Retourneert de tekenreeks die eerder is ingesteld met :** **OPTION PLATFORM.**

MM.XXX	Beschrijving
MM.INFO(PS2)	**Rapporteert het laatste onbewerkte bericht dat op de PS2-interface is ontvangen, indien ingeschakeld.**
MM.INFO(SDCARD)	**Retourneert "Niet geconfigureerd" als de SD-kaart niet is ingesteld**
MM.INFO$(SOUND)	**Retourneert de huidige activiteit op de audio-uitvoer (UIT, PAUSED, TONE, WAV, FLAC, SOUND).**
MM.INFO(STACK)	**Geeft de C stackpointer terug.** Complexe of recursieve Basiscode kan resulteren in de fout: **"Stack overflow, expression too complex at depth %".** Dit gebeurt wanneer de stack lager is dan **&H2003f800**. Door de stapel te bewaken, kan de programmeur vereenvoudigingen in de basiscode identificeren om de fout te voorkomen.
MM.INFO$(SYSTEM I2C)	**Retourneert I2C, I2C2 of NOT SET, zoals van toepassing.**
MM.INFO(SYSTEM HEAP)	**Retourneert de vrije ruimte op de systeem heap.**
MM.INFO(TCP PORT)	**Retourneert de TCP PORT die is ingesteld als server of 0 als deze niet is ingesteld.**
MM.INFO(TILE HEIGHT)	**ALLEEN VGA EN HDMI VERSIES** **Geeft de huidige instelling van de tegel hoogte terug.**
MM.INFO(TRACK)	**Retourneert de naam van het FLAC, MP3, WAV of MIDI bestand dat momenteel wordt afgespeeld op de audio uitvoer.**
MM.INFO$(TOUCH)	**Retourneert de status van de Touch-controller.** Geldige retour zendingen zijn: **"Uitgeschakeld"**, **"Niet gekalibreerd"** en **"Gereed"**.
MM.INFO(UDP PORT)	**Retourneert de UDP PORT die is ingesteld als server of 0 als deze niet is ingesteld.**
MM.INFO(USB n)	**Retourneer de apparaatcode voor elk apparaat dat is aangesloten op kanaal 'n', wat een getal is van 1 tot 4.** De geretourneerde apparaatcode kan zijn: **0** = niet in gebruik **1** = toetsenbord **2** = muis **128** = ps4 **129** = ps3 **130** = SNES/ Algemeen Standaard wordt een aangesloten toetsenbord toegewezen aan kanaal 1, een muis aan kanaal 2 en gamepads aan kanaal 3 en vervolgens aan kanaal 4. Als er 2 of meer toetsenborden of muizen zijn aangesloten of 3 of meer gamepads, worden de extra apparaten toegewezen aan het hoogst beschikbare kanaal.
MM.INFO(VARCNT)	**Retourneert het aantal variabelen dat in het MMBasic programma wordt gebruikt.**
MM.INFO$(LINE)	**Retourneert het huidige regelnummer als een tekenreeks.** **LIBRARY** wordt geretourneerd als het in de bibliotheek staat en **UNKNOWN** als het niet in een programma staat. Helpt bij diagnostiek tijdens het testen van eenheden.

MM.XXX	Beschrijving 143
MM.INFO(UPTIME)	**Retourneert de tijd in seconden sinds het opstarten als een drijvende komma getal.**
MM.INFO(VERSION)	**Retourneert het versienummer als een getal met drijvende komma.**
MM.INFO(WRITEBUFF)	**Retourneert het adres in het geheugen van de huidige buffer die wordt gebruikt voor teken commando's.**
MM.INFO(TCP PORT)	**Retourneert de TCP-poort die is ingesteld als server of 0 als deze niet is ingesteld. ALLEEN voor de WEBMITE.**
MM.INFO(UDP PORT)	**Retourneert de UDP-poort die is ingesteld als server of 0 als deze niet is ingesteld. ALLEEN voor de WEBMITE.**
MM.INFO(TCPIP STATUS)	**Retourneert de TCPIP-status van de verbinding.** **ALLEEN voor de WEBMITE.**
MM.INFO(WIFI STATUS)	**Geeft de WIFI-status van de verbinding terug.** Geldige retour zendingen zijn: **0** = Verbinding is uitgeschakeld. **1** = Verbonden met WIFI. **2** = Verbonden met WIFI, maar geen IP-adres. **3** = Maak verbinding met WIFI met een IP-adres. **-1** = Verbinding mislukt. **-2** = Geen overeenkomende SSID gevonden. (mogelijk buiten bereik of niet beschikbaar). **-3** = Authenticatie fout.
MM.HRES MM.VRES	**Gehele getallen die de horizontale en verticale resolutie van het LCD-scherm vertegenwoordigen weergavepaneel** (indien gecon-figureerd) in pixels.
MM.MESSAGE$	**Deze alleen lezen variabele retourneert de inhoud van het laatst ontvangen UDP-datagram. ALLEEN voor de WEBMITE.**
MM.ONEWIRE	**Na een 1-Wire reset-functie wordt deze integer-variabele ingesteld om aan te geven het resultaat van de bewerking:** **0 = apparaat niet gevonden** **1 = apparaat gevonden**.
MM.VER	**Het versienummer van de firmware als een getal met drijvende komma in de vorm aa.bbcc waarbij aa het hoofdversienummer is, bb de secundaire en cc is het revisienummer.** Voorbeeld : **Print MM.Ver retouneert 6.0001 : A = MM.VER**
MM.HPOS MM.VPOS	**De huidige horizontale en verticale positie (in pixels) volgend op de laatste afbeeldingen of afdrukopdracht.**
MM.WATCHDOG	**Een geheel getal dat waar is als MMBasic opnieuw is gestart als resultaat van eenWatchdog time-out.** (zie de opdracht : **WATCHDOG**) anders false.

Opties

Deze tabel geeft een overzicht van de verschillende optiecommando's die kunnen worden gebruikt om MMBasic te configureren en de manier waarop het kan worden gewijzigd werkt. Opties die als permanent zijn gemarkeerd, worden automatisch opgeslagen in het niet-vluchtige geheugen hersteld wanneer de PicoMite opnieuw wordt opgestart. Opties die niet permanent zijn, worden bij het opstarten gereset.

Veel OPTION commando's dwingen een herstart van de PicoMite af en dat zal ervoor zorgen dat de interface van de USB-console wordt resetten. Het programma dat in het geheugen wordt bewaard, gaat niet verloren omdat de firmware automatisch een reservekopie herstelt op herstarten.

OPTION	Beschrijving 144
`OPTION ANGLE RADIANS | DEGREES`	**Met dit commando schakelt u functies tussen graden en radialen.** Werkt op `SIN`,`COS`,`TAN`,`ATN`,`ATAN2`,`MATH ATAN3`,`ACOS`,`ASIN`
`OPTION AUDIO PWMnApin, PWMnBpin` of `OPTION AUDIO DISABLE`	**Configureert een van de PWM-kanalen als audiouitgang.** Voorbeeld: `OPTION AUDIO PWM` **GP18, GP19** zou **PWM1A** gebruiken en **PWM1B** op respectievelijk **pinnen 24 en 25**. Deze optie voorkomt het gebruik van deze pinnen in het BASIC programma. De audiouitvoer wordt gegenereerd met behulp van **PWM**, dus een laagdoorlaatfilter is nodig op de uitgang. De audiouitvoer van de PicoMite is ergluidruchtig. `OPTION` **POWER** gebruiken en/of stroom leveren via een aparte **3V3** lineaire regelaar kan dit verminderen.
`OPTION AUDIO SPI CSpin, LDACpin, SCKpin, SDIpin` of `OPTION AUDIO DISABLE`	**Ondersteuning voor MCP4822 DAC(8-pins versie) voor audiouitvoer (MCP4802=8-bit, MCP4812=10-bit en MCP4822=12-bit dual DAC).** Voordelen van de **DAC**, Verkrijgbaar als **PDIP**, Geen ingewikkelde filters nodig. Minder ruis op audio, Interne **2.048V-referentie**, dus spannings- zwaai is tussen **0 en 2.048 volt**. Om dit mogelijk te maken. NB: **SCKpin** en **SDIpin** moeten geldige **SPI TX**- en **CLK-pinnen** zijn volgens de handleiding. Het **SPI-kanaal** is toegewezen aan de **DAC** en kan niet worden gedeeld. Deze modus gebruikt ook het **PWM-kanaal** dat is gekoppeld aan de gekozen **CLK-pin.** bijv. `OPTION AUDIO SPI` **GP0, GP1, GP2, GP3** Gebruikt: **SPI** en **PWM 1** **GEBRUIK** standaard syntaxis voor **PWM AUDIO**-uitvoer.
`OPTION AUDIO SPI CSpin,CLKpin,MOSIpin` of `OPTION AUDIO DISABLE`	**Schakel over naar het gebruik van een MCP48n2 DAC voor audiouitvoer.** Koppel **LDAC** aan **GND**.
`OPTION AUDIO VS1053 CLKpin, MOSIpin, MISOpin, XCSpin, XDCSpin, DREQpin, XRSTpin` of `OPTION AUDIO DISABLE`	**Configureert de audio-uitvoer zodat deze naar een VS1053 CODEC wordt gestuurd.** Dit maakt **MP3**- en **MIDI**-weergave mogelijk naast de andere formaten ondersteund en ondersteunt ook realtime **MIDI**-uitvoer. Zie **PLAY** opdracht voor meer details.

OPTION	Beschrijving 145
OPTION AUTOREFRESH OFF | ON	**Zwart-witschermen kunnen slechts een volledig scherm tegelijk worden bijgewerkt.** Door OPTION **AUTOREFRESH OFF/ON** te gebruiken, kunt u of een schrijfopdracht het scherm onmiddellijk bijwerkt of niet. Als **AUTO-REFRESH** is **OFF**. Het **REFRESH** commando kan worden gebruikt om het schrijven activeren. Dit geldt voor de volgende displays: **N5110**, **SSD1306I2C**,**SSD1306I2C32**,**SSD1306SPI**,**ST7920**,**GDEH029A1**
OPTION AUTORUN ON [,NORESET] of OPTION AUTORUN n [,NORESET] of OPTION AUTORUN OFF	**Instrueert MMBasic om automatisch een programma uit te voeren bij het opstarten of herstarten.** **AAN** zorgt voor een automatische backup van het huidige programma in het RAM worden opgehaald en uitgevoerd. Als u **'n'** opgeeft, wordt die locatie in flashgeheugen dat moet worden geladen en uitgevoerd. **'n'** moet tussen 1en 3 liggen. Door de optionele parameter **"NORESET"** op te geven, wordt **AUTORUN** gehandhaafd, zelfs als het programma een systeemfout veroorzaakt (standaard zorgt dit ervoor dat de firmware elke OPTION **AUTORUN** instelling annuleert) **UIT** schakelt de autorun optie uit en is de standaard voor een nieuwe programma. Het invoeren van de break-toets (standaard **CTRL-C**) op de console zal onderbreken het lopende programma en keer terug naar de opdrachtprompt. Zet een **Pauze** van **2** seconde in begin programma.
OPTION BASE 0 | 1	**Stel de laagste waarde voor matrixsubscripts in op 0 of 1.** Dit moet worden gebruikt voordat arrays worden gedeclareerd en wordt gereset naar de **standaard 0** bij opstarten.
OPTION BAUDRATE nn	**Stel de baudrate van de seriële console in (indien geconfigureerd).**
OPTION BREAK nn	**Stel de waarde van de break key in op de ASCII-waarde 'nn'.** Deze sleutel is gebruikt om een lopend programma te onderbreken. De waarde van de break-toets is ingesteld op **CTRL-C**-toets bij het opstarten, maar het kan met deze opdracht worden gewijzigd in elke klaviertoets (voor OPTION **BREAK 4** zal bijvoorbeeld de break-toets instellen op **CTRL-D** sleutel). Als u deze optie op nul zet, wordt de pauzefunctie volledig uitgeschakeld.
OPTION CASE LOWER | UPPER | TITLE	**Wijzig de hoofdlettergebruik voor het weergeven van opdracht- en functienamen wanneer met behulp van het LIST commando**. De standaard is **TITLE** maar de oude standaard van MMBasic kan worden hersteld met OPTION CASE **UPPER**.
OPTION COLOUR VGA ON | OFF	**Alleen beschikbaar op de PicoMiteVGA.** Schakelt tussen **640 x 480 zwart-wit** en **320 x 240 4-bits kleurmodi**.
OPTION COLOURCODE ON | OFF	**Kleurcodering in- of uitschakelen voor de uitvoer van de editor.** Trefwoorden zullen in cyaan zijn, opmerkingen in geel, enz. De instelling is stanaard **UIT**. Het trefwoord **COLORCODE** (**USA** spelling) kan ook worden gebruikt. Dit vereist een terminalemulator die de juiste ontsnappingscodes (bijv. **Tera Term**).

OPTION	Beschrijving 146
OPTION CONSOLE output	**Specificeert waar printinstructies worden uitgevoerd.** Geldige instellingen zijn **BOTH** (d.w.z. **SCREEN** en **SERIAL**), **SERIAL**, **SCREEN**, **NONE**. Dit is een tijdelijke optie die wordt gereset als er een programma bestaat.
OPTION CPUSPEED speed	**NIET BIJ HDMI VERSIES** **Wijzig de CPU-klok.** **'snelheid'** Is de **CPU-klok in KHz** in het bereik van **48000** tot **378000 KHz** (alleen **126000** en **252000 KHz** toegestaan op de Pico-MiteVGA). Snelheden **boven 133 MHz** (**150 MHz RP2350** en **304 MHz** maximaal) worden beschouwd als overklokken, aangezien dat het gespecificeerde maximum is snelheid van de standaard **Raspberry Pi Pico RP2040**. Als er geen optie is ingesteld, wordt de CPU snelheid standaard ingesteld op **133000** (**126000** aan de **PicoMiteVGA**) of (**315000** voor **PicoMiteHDMI USB RP2350A** is **niet** te **wijzigen**).
OPTION COUNT pin1, pin2, pin3, pin4	**Specificeert welke pinnen moeten worden gebruikt als telingangen** Standaard zijn dit **GP6**, **GP7**, **GP8** en **GP9**. Het **SETPIN**-commando definieert de Tellermodus. Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt. (niet in een programma)
OPTION DEFAULT FLOAT | INTEGER | STRING | NONE	**Wordt gebruikt om het standaardtype in te stellen voor een variabele die niet expliciet is bepaald**. Als **OPTION DEFAULT NONE** wordt gebruikt, moeten alle variabelen hun type expliciet gedefinieerd of de fout "Variabele type niet gespecificeerd" zal voorkomen. Wanneer een programma wordt uitgevoerd, is de standaard ingesteld op **FLOAT** voor compatibiliteit met **Microsoft BASIC** en eerdere versies van **MMBasic**.
OPTION DEFAULT COLOURS foreground [,background]	**ALLEEN VGA- EN HDMI-VERSIES** **Stel de standaard voorgrond- en achtergrondkleuren in voor beide monochrome en kleurmodi.** De kleur moet een van de volgende zijn: **Nederlands: wit, geel, lila, bruin, fuchsia, roest, magenta, rood, cyaan, groen, ceruleum, middengroen, kobalt, mirte, blauw, zwart.** **Engels: white, yellow, lilac, brown, fuchsia, rust, magenta, red, cyan, green, cerulean, midgreen, cobalt, myrtle, blue, black.** Er kan geen numerieke waarde worden gebruikt. De standaard-instelling is **wit, zwart**. Namen zijn in het **Engels.** Als de achtergrond wordt weggelaten, is deze standaard zwart.
OPTION DEFAULT MODE n	**Hiermee wordt de standaardweergavemodus bij het opstarten ingesteld.** Deze opdracht moet worden uitgevoerd vanaf de opdracht-prompt (niet in een programma)

OPTION	Beschrijving 147
OPTION DISK SAVE fname$ OPTION DISK LOAD fname$	**Met deze opdrachten kan de gebruiker de volledige set opties die van en naar een schijfbestand zijn gedefinieerd, opslaan en herstellen.** Voorbeeld: OPTION DISK SAVE "RP2350A-V6.00.01.opt" Het bestand kan vervolgens met behulp van XMODEM naar een hostcomputer worden overgebracht, waardoor extra apparaten eenvoudig kunnen worden geconfigureerd of opties kunnen worden hersteld na een firmware-upgrade.
OPTION DISPLAY lines [,chars]	**Stel de kenmerken in van de weergaveterminal die voor de console wordt gebruikt.** Zowel de LIST- als de EDIT-commando's moeten deze informatie kennen om de tekst correct opmaken voor weergave. **'lines'** is het aantal regels op het display en **'chars'** is de breedte van de weergave in tekens. De standaard is 24 regels x 80 tekens en wanneer gewijzigd, wordt deze optie onthouden, zelfs als de stroom uitvalt **VERWIJDERD**. Maximale waarden zijn **100 regels** en **240 tekens**. Hierdoor wordt een **ESC-reeks** verzonden om de **VT100-terminal** op de overeenkomende grootte in te stellen. **TerraTerm, Putty** reageren op deze reeks en stellen de terminalbreedte in (als de optie is ingeschakeld in de terminalinstellingen) **Dit commando is niet beschikbaar als het scherm als console wordt gebruikt**.
OPTION ESCAPE	**Dit maakt escape-reeksen in strings mogelijk en moet worden gebruikt in elk programma waar u ze wilt gebruiken.** U kunt het ook gebruiken bij de opdrachtprompt, maar net als OPTION EXPLICIT wordt het weer uitgeschakeld door het RUN commando en moet het dus in het programma worden opgenomen Deze wijziging zorgt ervoor dat de nieuwe functionaliteit geen bestaande programma's kapot maakt.
OPTION EXPLICIT	**Als u dit commando aan het begin van een programma plaatst, moet elke variabele expliciet worden gedeclareerd met behulp van de DIM, LOCAL of STATIC commando's voordat het in het programma kan worden gebruikt.** Deze optie is standaard uitgeschakeld wanneer een programma wordt uitgevoerd. Als het wordt gebruikt, moet worden opgegeven voordat variabelen worden gebruikt.
OPTION FAST AUDIO ON | OFFF	**Wanneer u de opdracht PLAY SOUND gebruikt, kunnen wijzigingen in geluiden, volumes of frequenties hoorbare klikken in de uitvoer veroorzaken.** De firmware probeert dit te verzachten door het volume van de vorige uitvoer van het kanaal te verlagen voordat de uitvoer wordt gewijzigd en weer omhoog wordt gebracht. Dit verbetert de audio-uitvoer aanzienlijk, maar gaat ten koste van een korte vertraging in het PLAY **SOUND** commando (in het slechtste geval **3 msec**). Deze vertraging kan worden vermeden door **OPTION FAST AUDIO ON** in een programma te gebruiken. De hoorbare klikken kunnen dan opnieuw verschijnen, maar dit is ter beoordeling van de programmeur. Dit is een tijdelijke optie die telkens op **UIT** wordt gezet wanneer een programma wordt uitgevoerd.

OPTION	Beschrijving
`OPTION FNKey string$`	**Definieer de string die wordt gegenereerd wanneer een functie-toets wordt ingedrukt bij de opdrachtprompt.** '**FNKey**' kan **F1** zijn en **F5** tot en met **F9**. Voorbeeld: `OPTION F8 "RUN "+Chr$(34)+"myprog"+Chr$(34)+ Chr$(13)+Chr$(10)` `OPTION List -> OPTION F8 RUN "myprog"` Uitschakelen: `OPTION F8 ""`
`OPTION HDMI PINS clockpositivepin, d0positivepin, d1positivepin, d2positivepin`	**ALLEEN HDMI-VERSIE** **Stel de I/O-pinnen in die worden gebruikt voor de HDMI-videouit-voer.** Dit is alleen nodig voor niet-standaard **PCB**-indelingen. De positieve **HDMI**-signaalpinnen worden ingesteld volgens **'nbr'** hieronder. Geldige waarden zijn **0-7** en de pinnen mogen elkaar niet overlap-pen voor elk kanaal. Als **'nbr'** een **even getal (0,2,4,6)** is, bevindt de negatieve uitvoer zich op fysieke **pin + 1**, als **'nbr' oneven (1,3,5,7)** is, bevindt deze zich op fysieke **pin - 1**. `nr.  HSTX-nr. Fysieke pin` `0  HSTX  0  GP12` `1  HSTX  1  GP13` `2  HSTX  2  GP14` `3  HSTX  3  GP15` `4  HSTX  4  GP16` `5  HSTX  5  GP17` `6  HSTX  6  GP18` `7  HSTX  7  GP19` De standaardinstelling is: `OPTION HDMI PINS 2, 0, 6, 4` Wat betekent dat: `CK+ en CK- worden toegewezen aan GP14 en GP15` `D0+ en D0- worden toegewezen aan GP12 en GP13` `D1+ en D1- worden toegewezen aan GP18 en GP19` `D2+ en D2- worden toegewezen aan GP16 en GP17`
`OPTION GUI CONTROLS NbrOfGUIControls`	**Specificeert het maximum aantal GUI-besturingselementen dat kan worden gedefinieerd.** Elk besturingselement gebruikt **52 bytes** en het totale gebruikte geheugen moet worden afgerond tot het volgende veelvoud van **2048 bytes**. Bijvoorbeeld **70 specificeren** bedieningselementen ge-bruiken **4 KB RAM**. Standaard is het aantal GUI-besturingselementen ingesteld op nul, dus deze optie moet worden gebruikt voordat GUI-besturingsele-menten worden gedefinieerd. **Niet op de PicoMiteVGA**.
`OPTION HEARTBEAT ON | OFF`	**NIET WEBMITE VERSIE** **Hiermee kan men de Led op het bord (GP25) uitschakelen en deze pin voor andere doeleinden gebruiken.** (V5.07.05b15 en later)

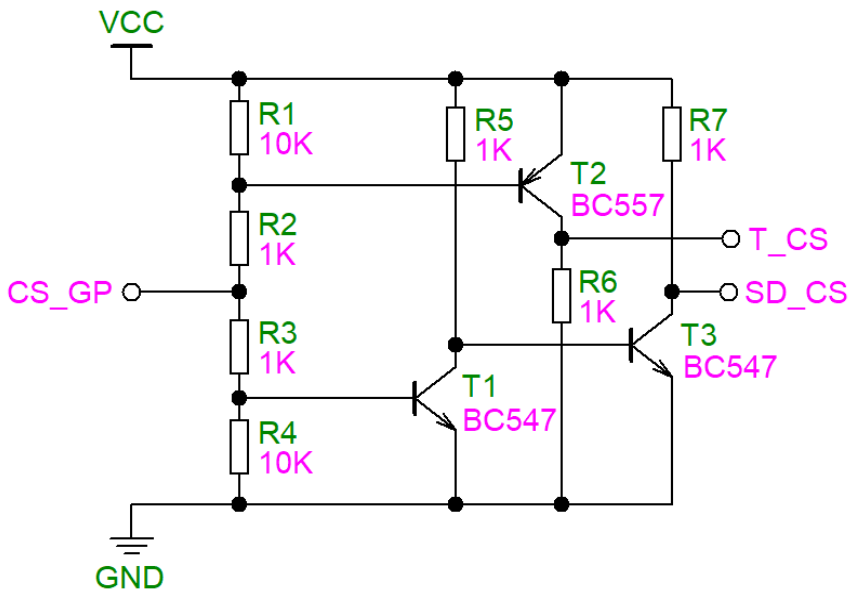
OPTION	Beschrijving
OPTION KEYBOARD nn [,capslock] [,numlock] [repeatstart] [repeatrate] of **OPTION KEYBOARD DISABLE**	**Configureer een PS2-toetsenbord.** Dit kan worden gebruikt voor consoleinvoer en elke getypte tekens zijn beschikbaar via alle opdrachten die worden gelezen uit de console (serieel via USB). **'nn'** is een code van twee tekens die de toetsenbordindeling definieert. De keuzes zijn **VS** voor de standaard toetsenbordindeling **in de VS**, **Australië** en **Nieuw Zeeland** en **VK** voor het **Verenigd Koninkrijk**, **DE** voor **Duitsland**, **FR** voor **Frankrijk** en **ES** voor **Spanje**. Het toetsenbord moet als volgt worden aangesloten: • Sluit de **GP8** aan op de **PS2**-aansluiting als **KLOK**-pin. • Sluit de **GP9** aan op de **PS2**-aansluiting als **DATA**-pin. • Sluit **VBUS** (**5V**) of **3V3** (**3,3V**) aan op **PS2** aansluiting **+5V.** (**Level shifter gebruiken**) • Sluit **GND** aan op **PS2-aansluiting GND** Sommige toetsenborden werken op **3,3V** en in dat geval de klok en data pinnen kunnen direct worden aangesloten. Als een toetsenbord niet beschikbaar is met **3V3** moet voor deze verbindingen een levelshifter worden gebruikt, zodat op de Pico-Mite I/O-pinnen niet meer dan **3,6 V** komt. Deze opdracht kan worden uitgevoerd vanaf de opdrachtregel en zal leiden tot het opnieuw opstarten. Deze instelling kan worden gereset met het commando : **OPTION KEYBOARD NONE** De waarden parameters capslock en numlock stellen de beginstatus van het toetsenbord (standaard **0, 1**). De herhalingstart hoe lang voordat een teken de eerste keer wordt herhaald. (geldig **0-3 = 250mSec**, **500mSec**, **750mSec**, **1S**: Standa. **1=500mSec**) De herhalingssnelheid definieert hoe snel een teken wordt herhaald na de eerste herhaling. (geldig **0-31 = 33mSec** tot **500mSec** standaard **12=100mSec**)
OPTION KEYBOARD I2C	**Stel OPTION SYSTEM I2C in als dit nog niet is ingesteld.** Stel vervolgens **OPTION KEYBOARD I2C** in. Toetsen bovenaan zijn geprogrammeerd als **ESC, F1, F2, F3, CTRL**. Om de editor te gebruiken, zie de handleiding voor de bedieningstoetsen die gebruikt worden voor vooruit, achteruit etc.
OPTION KEYBOARD PINS clockpin, datapin	**Hiermee kan de gebruiker de pinnen selecteren die moeten worden gebruikt voor het aansluiten van een PS2 toetsenbord**. De standaardinstelling is **pin 11** (**GP8**) en **pin 12** (**GP9**). Het **PS2-toetsenbord** moet zijn uitgeschakeld (**OPTION KEYBOARD DISABLE**)
OPTION KEYBOARD REPEAT repeatstart,repeatrate	**ALLEEN USB-TOETSENBORD** **Stelt de tijd in voor de eerste herhaling van een ingedrukte toets** (**100-2000**) en daaropvolgende herhalingen (**25-2000**) in milliseconden.

OPTION	Beschrijving 150
OPTION LCDPANEL	**GEEN VGA- OF HDMI-VERSIES** **Configureert een LCD-paneel op versies die een aangesloten LCD accepteren.**
OPTION LCDPANEL VIRTUAL_C of **OPTION LCDPANEL VIRTUAL_M**	**Configureert een virtueel LCD-paneel zonder een fysiek aangesloten paneel.** **VIRTUEEL _C = Kleur, 4 bit, 320 x 240** **VIRTUEEL _M = Monochroom, 640 x 480** Met behulp van deze functie kan een programma grafische afbeeldingen op deze virtuele afbeelding tekenen paneel en sla ze vervolgens op als een BMP-bestand. Handig voor het maken van een afbeelding voor export zonder aangesloten beeldscherm.
OPTION LCDPANEL options of **OPTION LCDPANEL DISABLE**	**Configureert de PicoMite-firmware om te werken met een aangesloten LCD-paneel.** Zie het hoofdstuk LCD-schermen voor de details. Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt **(niet in een programma).**
OPTION LCDPANEL CONSOLE [font [, fc [, bc[, blight]]] [,NOSCROLL] of	**Configureert het LCD-scherm voor gebruik als console uitgang.** Het LCD-scherm moet transparante tekst ondersteunen (d.w.z. de **SSD1963_x, ILI9341** of **ST7789_320** controllers). **'font'** is het standaardlettertype **'fc'** is de standaard voorgrondkleur **'bc'** is de standaard achtergrondkleur.
OPTION LCDPANEL NOCONSOLE	**Deze parameters zijn optioneel en zijn standaard ingesteld op lettertype 1, wit, zwart en 100%.** Deze instellingen worden toegepast bij het opstarten. Het optionele **NOSCROLL** commando verandert de firmware zodanig dat bij uitvoer naar de laatste regel van het display in plaats van naar het display scrollen wordt gewist en de uitvoer gaat verder bovenaan het scherm. Dit staat toe dat beeldschermen die geen leesondersteuning bieden, als consoleapparaat worden gebruikt. Houd er rekening mee dat voor andere beeldschermen dan de SSD1963 scrollen voor elke console mogelijk is de uitvoer is erg langzaam, dus het wordt aanbevolen om de **NOSCROLL** optie te gebruiken voor deze displays. Deze instelling wordt in flash opgeslagen en wordt automatisch opgeslagen toegepast bij het opstarten. Om het uit te schakelen gebruikt u het **OPTION LCDPANEL NOCONSOLE**-opdracht. Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt.
OPTION LCDPANEL USER hres, vres	**Configureert een door de gebruiker geschreven beeldschermstuurprogramma in MMBasic.** Zie het bestand **"Gebruiker Display Driver.txt"** in de PicoMite firmwaredistributie voor een beschrijving hoe u het stuurprogramma **schrijft.**

OPTION	Beschrijving 151
OPTION LCDPANEL CONSOLE [font [,fc [,bc]]] of **OPTION LCDPANEL NOCONSOLE**	**ALLEEN VGA- EN HDMI-VERSIES** **Wijzigt het standaardlettertype dat op het VGA- of HDMI-scherm wordt gebruikt.** **'fc'** is de voorgrondkleur en **'bc'** is de achtergrondkleur. **Schakelt de console-uitvoer naar het VGA/HDMI-scherm uit.** Deze optie is permanent, zowel de afdrukuitvoer als de console uitvoer zijn dat uitgeschakeld en alleen grafische opdrachten worden naar het VGA-scherm uitgevoerd. Als de uitvoer tijdelijk moet worden uitgeschakeld in een programma, gebruikt u de optie **OPTION CONSOLE** commando.
OPTION LCD320 ON | OFF	**GEEN VGA- OF HDMI-VERSIES** **Hiermee worden 16-bits LCD-schermen in de 320x240-modus in- of uitgeschakeld, waardoor zaken als games op deze grotere LCD-schermen mogelijk zijn.** In het geval van **800 x 480-schermen** wordt het **320 x 240 beeld** geschaald met **2** en beslaat het schermgebied **80, 0** tot **719, 479**. In het geval van **480 x 272 weergaven** wordt de afbeelding van **320 x 240** in een venster weergegeven beslaat het schermgebied **80, 16** tot **399, 255**.
OPTION LEGACY ON of **OPTION LEGACY OFF**	**Hiermee wordt de compatibiliteitsmodus met de grafische opdrachten in- of uitgeschakeld gebruikt in de originele Color Maximite.** De commando's **COLOR**, **LINE**, **CIRCLE** en **PIXEL** gebruiken de oude syntaxis en alle tekeningen commando's accepteren kleuren in het bereik van **0** tot **7.** Opmerkingen: • Trefwoorden zoals ROOD, BLAUW, enz. zijn niet geïmplementeerd, dus ze moeten indien nodig als constanten worden gedefinieerd. • Raadpleeg de Color Maximite MMBasic Language Manual voor de syntaxis van de legacy-opdrachten. Dit kan worden gedownload van: https://geoffg.net/OriginalColourMaximite.html.
OPTION LIST	**Dit toont de instellingen van alle opties die zijn gewijzigd van hun standaardinstelling en zijn van het permanente type.** Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt (niet in een programma)
OPTION MILLISECONDS ON | OFF	**Hiermee wordt een milliseconde-uitvoer in de TIME$ functie in- of uitgeschakeld.** Dat wil zeggen, **UU:MM:SS.mmm** De millisecondenteller wordt op nul gezet telkens wanneer de tijd wordt bijgewerkt met behulp van het **TIME**-commando, **WEB NTP**-commando of **RTC GETTIME**-commando. Standaard is **UIT**.

OPTION	Beschrijving 152
OPTION MODBUFF ENABLE | DISABLE [sizeinK]	**Creëert of verwijdert een gebied met flashgeheugen dat wordt gebruikt voor het laden en verwijderen van bestanden .MOD bestanden afspelen.** Indien ingeschakeld, wordt er een **mod-buffer** gemaakt met een grootte van **128 Kbytes**. Dit kan worden over-schreven met **"sizeinK"** Merk op dat deze optie een deel van het **Flash**-bestandssysteem reserveert (dat wil zeggen, het verkleint het **Flash**-bestandssysteem) De standaardinstelling is uitgeschakeld.
OPTION MOUSE CLKpin, DATApin	**ALLEEN GEEN USB-FIRMWARE** **Stel de pinnen in die moeten worden gebruikt om een PS2-muis aan te sluiten.** Met dit commando wordt de muis automatisch geconfigureerd tijdens het opstarten en kun je interrupts instellen en waarden lezen zonder extra commando's. Dit is anders dan de **MUIS OPEN**, die alleen verbinding maakt met een muis terwijl het programma actief is. De **PS2-muis MOET** uitgeschakeld zijn.
OPTION MOUSE DISABLE	**Schakelt de automatische verbinding met een PS2-muis uit en maakt de pinnen vrij voor normaal gebruik.**
OPTION PICO ON | OFF	**ALLE VERSIES BEHALVE WEBMITE** **Wanneer ingesteld op OFF, zijn de pinnen GP23, GP24 en GP29 niet ingesteld voor normaal Pico gebruik en zijn ze onmiddellijk beschikbaar voor gebruik.** Standaard **AAN** voor **RP2350A** en **RP2040** , **UIT** voor **RP2350B.**
OPTION PIN nbr	**Stel 'nbr' in als de pincode (persoonlijk identificatienummer) voor toegang tot de consoleprompt. 'nbr' kan elk getal zijn dat niet nul is, met maximaal acht cijfers.** Telkens wanneer een actief programma om welke reden dan ook probeert af te sluiten via de opdrachtprompt, zal MMBasic om dit nummer vragen voordat de prompt wordt weergegeven. Dit is een beveiligingsfunctie, omdat een indringer zonder toegang tot de opdrachtprompt het programma in het geheugen niet kan weergeven of wijzigen, of de werking van MMBasic op geen enkele manier kan wijzigen. Om deze functie uit te schakelen, voert u een nul in voor de pincode (d.w.z. **OPTION PIN 0**). Een permanente vergrendeling kan worden toegepast door **99999999** als pincode te gebruiken nummer. Als er een permanente vergrendeling wordt toegepast of als de pincode verloren gaat, is de enige manier om te herstellen is door de PicoMite-firmware opnieuw te laden. Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt (niet in een programma).

OPTION	Beschrijving
`OPTION PLATFORM name$`	**Hiermee kan een gebruiker een bepaalde hardwareconfiguratie identificeren die vervolgens in programma's kan worden gebruikt om de werking van het programma te besturen.** **'naam$'** mag **maximaal 31 tekens** lang zijn. Dit is een permanente optie **MM.INFO$(PLATFORM)** retourneert deze tekenreeks. Dit kan bijvoorbeeld worden gebruikt voor een bepaalde hardware-configuratie: **OPTION PLATFORM "GameMite"** Dan kunnen programma's die op dit of andere platforms kunnen draaien het volgende gebruiken: `IF MM.INFO$(PLATFORM) = "GameMite" THEN`
`OPTION POWER PFM | PWM`	**Verandert de werking van de 3,3V voeding met schakelmodus.** Standaard draait dit in de **PFM**-modus. **PWM** levert betere geluidsprestaties, maar is minder energiezuinig. Houd er rekening mee dat het systeem bij zware belasting in de **PWM**-modus draait, ongeacht deze instelling.
`OPTION PS2 PINS clockpin, datapin`	**Standaard GP8, GP9 zoals voorheen.**
`OPTION PSRAM PIN n` of `OPTION PSRAM DISABLE`	**PSRAM-ondersteuning in-/uitschakelen.** **'n'** is de **PSRAM** chip select (**CS**) pin en kan **0, 8, 19** of **47** zijn. Meestal wordt **GP47** gebruikt voor Pimoroni-borden. Standaard is uitgeschakeld.
`OPTION RESET`	**Alle opties wissen**. **OPTION RESET zet alle opties terug naar standaard**, dus zal ook de bibliotheek verwijderen en **Flash Slot 3** terugbrengen voor normaal gebruik.
`OPTION RESET cfg` of `OPTION RESET LIST` `OPTION RESET OLIMEX`	**Reset alle opties naar de standaardwaarden voor de configuratie 'cfg'.** **OPTION RESET LIST** geeft een overzicht van alle beschikbare configuraties. Zie tabel onderaan de lijst. Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt (niet in een programma).
`OPTION RESOLUTION nn [,cpuspeedinkHz]` `OPTION RESOLUTION` `640x480` `720x400` `800x600` `848x480` `1024x768` `1280x720` `OPTION RESOLUTION` `640` `720` `800` `848` `1024` `1280`	**ALLEEN HDMI-VERSIES** **Voor firmware met HDMI-video stelt u de videoresolutie in op 'nn'.** Waar **'nn'** is: * **= Versie 6.00.02** `RESOLUTION CPU_FREQ BEELD(Hz) MODE` `640 x 480 of 640, 252000kHz (60Hz) Mode 1,2,3,5` `640 x 480 of 640, 315000kHz (75Hz) Mode 1,2,3,5` `* 720 x 400 of 720, 283200kHz (60Hz) Mode 1,2,3,5` `* 800 x 600 of 800, 360000kHz (56Hz) Mode 1,2,3,5` `* 848 x 480 of 848, 336000kHz (60Hz) Mode 1,2,3,5` `1024 x 768 of 1024, 324000kHz (60Hz) Mode 1,2,3,5` `1280 x 720 of 1280, 372000kHz (60Hz) Mode 1,2,3,5` Voor **640 x 480** kan de weergavefrequentie worden ingesteld op **60 Hz (252 MHz)** of **75Hz (315MHz)** door **'cpuspeedinkHz'** aan het commando toe te voegen (dwz **252000** of **315000**). Elke **HDMI** resolutie kan in een aantal modi werken die worden ingesteld met behulp van de **MODE** opdracht. Zie ook: **Option List**

OPTION	Beschrijving
OPTION RTC AUTO ENABLE | DISABLE	**Schakel automatisch laden van Time$ en Date$ van RTC in tijdens het opstarten en elk uur.** Als ingeschakeld en de **RTC** reageert niet, dan wordt elk lopend programma afgebroken met een foutmelding. Bij de opdrachtprompt wordt een informatiebericht weergegeven. Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt (niet in een programma).
OPTION SDCARD CSpin [,CLK-pin, MOSI-pin, MISO-pin] of **OPTION SDCARD DISABLE**	**Specificeer de I/O-pinnen die moeten worden gebruikt voor de SD-kaartinterface.** Als de uitgaven pinnen niet zijn gebruikt, gebruikt de **SD-kaart** de pinnen weergegeven door **OPTION SYSTEM SPI.** Opmerking: de door **OPTION SYSTEM SPI** gespecificeerde pinnen moeten geldig zijn set hardware **SPI** pinnen (**SPI** of **SPI2**), echter de pinnen door **OPTION SDCARD** kan alle pinnen zijn. De pincode door **OPTION SYSTEM SPI** en **OPTION SDCARD** kunnen niet hetzelfde zijn. Op de **PicoMiteVGA** zijn de **CLKpin, MOSI pin** en **MISO pin** verplicht en het **OPTION SYSTEM SPI** commando niet beschikbaar. Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt (niet in een programma).
OPTION SDCARD COMBINED CS	**Als dit is gespecificeerd, wordt de touch-chip-selectiepin ook gebruikt voor de SD-kaart.** In dit geval zijn externe schakelingen nodig om de SD-chipselectie als volgt te implementeren.  De firmware gebruikt de aanraakpin als volgt: **TOUCH_CS laag: TOUCH_CS laag, SD_CS hoog** **TOUCH CS hoog: SD_CS laag: TOUCH_CS hoog** **TOUCH CS** ingesteld als ingang (hoge impedantie) **TOUCH_CS en SD_CS hoog**

OPTION	Beschrijving
OPTION SYSTEM SPI CLKpin,MOSIpin,MISOpin of **OPTION SYSTEM SPI DISABLE**	**Specificeer de SPI-poort en pinnen voor gebruik door systeemapparaten (SD-kaart, LCD-paneel, enz.).** De PicoMite gebruikt een specifieke hardware **SPI-poort** voor systeemapparaten, de ander voor de gebruiker overlatend. Dit commando specificeert welke pinnen zijn worden gebruikt, en dus welke van de **SPI-poorten** moet worden gebruikt. de pinnen toege-wezen aan de **SYSTEM SPI** zal niet beschikbaar zijn voor andere MMBasic commando's. Deze commando's moeten worden gebruikt vanuit de pompt. **Niet op de PicoMiteVGA.**
OPTION SERIAL CONSOLE uartapin, uartbpin **OPTION SERIAL CONSOLE TXpin, RXpin [,B]** **OPTION SERIAL CONSOLE DISABLE**	**Geef op dat de console toegankelijk moet zijn via een hardware-poort (in plaats van virtuele serieel via USB).** **'uartapin'** en **'uartbpin'** kunnen elk geldig paar **rx-** en **tx-** pinnen zijn voor of **COM1** of **COM2**. Het toevoegen van de **"B"**-parameter betekent dat de uitvoer naar **"B"** gaat, naast de seriële poort en de USB. De waarin ze zijn begonnen, is niet belangrijk. De snelheid is standaard ingesteld op **115200 baud**, maar kan worden gewijzigd bekijk de **OPTION BAUDRATE**. Let op: **OPTION SERIAL CONSOLE COM1: GP0, GP1** zal de usb uitschakelen en is bediening alleen via **externe com1 module** mogelijk. Om weer toegang te krijgen tot de usb zal men het geheugen moeten wissen met **Clear_flash.uf2** en daarna weer met normale firmware moeten laden waardoor alles zal worden gewist. **Keer terug naar de normale USB-console.** Deze opdrachten moeten worden uitgevoerd vanaf de opdrachtprompt (niet in een programma).
OPTION SYSTEM I2C sdapin, sclpin, SLOW / FAST	**Specificeer de I2C poort en pinnen voor gebruik door systeemapparaten (LCD-paneel, en RTC)**. De PicoMite gebruikt een specifieke I2C-poort voor systeemapparaten, laat de andere voor de program-meur. Dit commando specificeert welke pinnen moeten worden ge-bruikt, en dus welke van de I2C poorten moeten worden gebruikt. De pinnen die zijn toegewezen aan de **SYSTEM I2C** zijn niet beschikbaar voor andere MMBasic **SETPIN** instellingen, maar kan worden gebruikt voor extra **I2C** apparaten met behulp van het standaard **I2C** commando. Opmerking: **I2C(2) OPEN** en **I2C(2) CLOSE** zijn in dit geval niet beschikbaar. Standaard wordt de **I2C**-poort geopend met een snelheid van **400 KHz** en met een time-out van **100 mSec**. De **I2C**-frequentie kan worden ingesteld met behulp van de optionele derde parameter die de waarden **FAST = 400 KHz** of **SLOW = 100 KHz** kan aannemen.
OPTION TAB 2 | 3 | 4 | 8	**Stel de afstand in voor de tab-toets. Standaard is dit 2**.

OPTION	Beschrijving
`OPTION TCP SERVER PORT n`	**ALLEEN WEBMITE** **Start een TCP-server op poort 'n' tijdens elke herstart van de WebMite.** Normaal gesproken gebruiken **HTTP-servers poort 80**. GEBRUIK **"OPTION TCP SERVER PORT 0"** om uit te schakelen. Wanneer de server actief is, kan deze reageren op maximaal **MM.INFO(MAX CONNECTIONS)**
`OPTION TELNET CONSOLE OFF | ONLY | ON`	**ALLEEN WEBMITE** **Configureert de afhandeling van de console via Telnet.** **AAN** = Console-uitvoer verzonden naar **USB** en **Telnet.** **ALLEEN** = Console-uitvoer wordt alleen naar **Telnet** verzonden. **UIT** = Console-uitvoer wordt alleen naar **USB** verzonden.
`OPTION TFTP OFF | ON`	**ALLEEN WEBMITE** **Schakelt de TFTP-server in of uit**. Standaard staat aan.
`OPTION TOUCH T_CSpin, T_IRQpin [, Beep]` of `OPTION TOUCH DISABLE`	**Configureert MMBasic voor de aanraakgevoelige functie van een aangesloten LCD-scherm.** **'T_CS pin'** en **'T_IRQ pin'** zijn de PicoMite **I/O-pinnen** die worden gebruikt voor chip select en touch interrupt respectievelijk (elke vrije pinnen kunnen worden gebruikt). De overige pinnen zijn verbonden met de pinnen die zijn opgegeven met behulp van de: **OPTION SYSTEM SPI** opdracht. **'Beep'** is een optionele pin die kan worden aangesloten op een kleine zoemer/pieper om een **"klik"** of **pieptoon** te genereren wanneer een geavanceerde bediening wordt aangeraakt. (dwz keuzerondje, schakelaar, enz.). Dit is beschreven in **Advanced Graphics Functions.pdf.** Deze opdracht moet worden uitgevoerd vanaf de opdrachtprompt (niet in een programma).
`OPTION UDP SERVER PORT n`	**ALLEEN WEBMITEVERSIE** **Stelt een luisteraansluiting in op de opgegeven poort.** Alle UDP-data-grammen ontvangen op die poort zal worden verwerkt en de inhoud zal worden opgeslagen **MM.MESSAGE$**. Het **IP-adres** van de afzender wordt opgeslagen in **MM.ADDRESS$**. Opmerking: als het UDP-datagram langer is dan 255 tekens, waarna eventuele extra tekens worden weggegooid. **GEBRUIK "OPTION UDP SERVER PORT 0"** om uit te schakelen.
`OPTION VGA PINS HSYNCpin, BLUEpin`	**ALLEEN VGA-VERSIE** **Verandert de pinnen die worden gebruikt voor VGA-weergaveuitvoer, waardoor meer flexibiliteit in PCB-ontwerp of bedrading mogelijk wordt.** **“HSYNCpin”** definieert het begin van een paar aaneengesloten GP-genummerde pinnen die zijn verbonden met **HSYNC** en **VSYNC.** **“BLUEpin”** definieert het begin van vier aaneengesloten GP-genummerde pinnen die zijn verbonden met: **BLUE, GREEN_LSB, GREEN_MSB** en **RED**.

OPTION	Beschrijving 157
OPTION VCC voltage	**Specificeert de spanning (Vcc) die aan de PicoMite wordt geleverd.** Bij gebruik van de **ADC**-pinnen om de spanning te meten, gebruikt de PicoMite de spanning op de pin gemarkeerd **VREF** als referentie. Deze spanning kan nauwkeurig gemeten met een **DMM** en ingesteld met deze opdracht voor nauwkeuriger meten. De parameter wordt niet opgeslagen en moet worden geïnitialiseerd op de opdrachtregel of in een programma. De standaardwaarde, indien niet ingesteld, is **3,3 V**.
OPTION WEB MESSAGES ON | OFF	**ALLEEN WEBMITEVERSIE** **Schakel informatieve webberichten uit als deze is ingesteld op UIT.** Standaard is **AAN**
OPTION WIFI ssid$, passwd$, [name$] [,ipaddress$, mask$, gateway$]	**ALLEEN WEBMITEVERSIE** **Configureert de firmware om automatisch verbinding te maken met een WiFi-netwerk bij het opnieuw opstarten.** **'ssid$'** is de naam van het netwerk en **'password$'** is de toegang wachtwoord. Beide zijn strings en als stringconstanten worden gebruikt, moeten deze tussen aanhalingstekens worden geplaatst. Optioneel kan een naam voor het apparaat worden opgegeven **'name$'**, anders wordt er een naam gemaakt op basis van de unieke **apparaat-ID**. Optioneel kan dit een statisch **IP-adres**, **IP-masker** en **gateway-adres** zijn gespecificeerd als **'ipadres$', 'mask$', 'gateway$'** bijv. **OPTION WIFI "mysid", "mijnwachtwoord", "myPico", "192.168.1.111", "255.255.255.0", "192.168.1.1"**

OPTION RESET LIST

VGAUSB	VGA	HDMIUSB	OLIMEX	Game	WEBMITE	PICOMITEUSB
PICOGAME 4-PWM	PICOGAME 4-PWM	HDMI	HDMIBasic	*Mite	Game*Mite	USB Edition V1.0
PICOGAME 4	PICOGAME 4		PICOMITE	Pico-ResTouch-LCD-3.5	Pico-ResTouch-LCD-3.5	
CMM1.5	PICOMITEVGA V1.1			Pico-ResTouch-LCD-2.8	Pico-ResTouch-LCD-2.8	
	PICOMITEVGA V1.0			PICO RUGZAK	PICO RUGZAK	
	VGA ontwerp 1			RP2040-LCD-1.28		
	VGA ontwerp 2			RP2040 LCD-0.96		
	SWEETIEPI			RP2040-GEEK		
	VGA Basic					

Operatoren

OPERATOREN	Beschrijving 158
Rekenkundige OPERATOR	
^	Machtsverheffen (bijv. **b^n** betekent bn).
* / \ MOD	Vermenigvuldiging. Deling. Gehele deling. Modulus (rest).
+ -	Optellen. Aftrekken.
Shift OPERATOR	
x << y x >> y	Deze werken op een speciale manier. **<<** betekent dat de geretourneerde waarde is de waarde van **x** verschoven met **y** bits naar links terwijl **>>** betekent de hetzelfde alleen rechts verschoven. Het zijn integerfuncties en alle bits verschoven worden weggegooid. Voor een verschuiving naar rechts worden alle ingevoerde bits ingesteld naar de waarde van het bovenste bit (bit 63). Voor een linker shift zijn alle bits geïntroduceerd zijn op nul gezet.
Logische OPERATOR	
NOT INV	Keer de logische waarde aan de rechterkant om (bijv. **NOT a = b** is **a <> b**) of bitsgewijze inversie van de waarde aan de rechterkant (bijv. **a = INV b**).
<> < > <= =< >= => =	Ongelijkheid. Kleiner dan. Groter dan. Kleiner dan of gelijk aan. Groter dan of gelijk aan. Gelijkheid
AND OR XOR	Conjunctie, Disjunctie, e**X**clusief **OFF**.
Tekenreeks operator	
+	Voeg twee strings samen
<> < > <= =< => <= =	Ongelijkheid. Kleiner dan. Groter dan. Kleiner dan of gelijk aan. Gelijk aan of kleiner dan. Gelijk aan of groter dan. Kleiner dan of gelijk aan. Gelijkheid.

Commando's

Vierkante haken geven aan dat de parameter of tekens optioneel zijn.

COMMANDO	Beschrijving 159
' enkel aanhalingsteken	**Start een opmerking en alle tekst die erop volgt, wordt genegeerd.** Opmerkingen kunnen ergens op een lijn worden geplaatst. In handleiding ' Licht Groen en bij de EDIT uitvoer ' Geel.
*file	**Het sterretje/sterretje-commando is een sneltoets voor RUN die alleen gebruikt mag worden bij de MMBasic-prompt.** bijvoorbeeld, `* LOOP` `*foe RUN "foe"` `*"foo-bar" RUN "foo-bar"` `*foo -wombat RUN "foo","--wombat"` `*foo "wom" RUN "foo",CHR$(34)+"wom"+CHR$(34)` `*foo --wom="bat" RUN "foo","--wom="+CHR$(34)+"bat"+CHR$(34)` Tekenreeksexpressies worden niet ondersteund/geëvalueerd door deze opdracht; alle opgegeven argumenten worden als een letterlijke tekenreeks doorgegeven aan de opdracht RUN.
? (vraagteken)	**Sneltoets voor het PRINT commando.**
/* */	**Begin en einde van opmerkingen met meerdere regels.** **/*** en ***/** moeten de eerste niet-spatie zijn tekens aan het begin van een regel en een spatie of einde van de regel erna (d.w.z. het zijn MMBasic-opdrachten). Commentaar met meerdere regels kan binnen niet worden gebruikt subroutines en functies. Alle tekens na ***/** op een regel worden ook behandeld als een **opmerking**.
A: of B: of Drive "A:" of Drive "B:"	**Snelkoppeling voor DRIVE "A:" en DRIVE "B:" bij de opdrachtprompt.**
ADC OPEN freq, n_channels[,interrupt]	**Dit wijst maximaal 4 ADC-kanalen toe voor gebruik GP26, GP27, GP28 en GP29 en stelt in dat ze worden geconverteerd met de opgegeven frequentie.** De maximale frequentie is **500KHz**. (bijv. **125KHz** als alle vier de kanalen moeten worden gesampled). Als het aantal kanalen is één, dan zal het altijd **GP26** worden gebruikt, indien twee dan **GP26** en **GP27** enz.. Sampling van meerdere kanalen is sequentieel (er is een slechts één ADC). De opgegeven pinnen zijn vergrendeld voor de functie wanneer ADC OPEN is actief De optionele interrupt parameter specificeert een interrupt die moet worden aangeroepen wanneer de conversie voltooid. Indien niet gespecificeerd, wordt de conversie geblokkeerd.
ADC FREQUENCY freq	**Dit verandert de bemonsteringsfrequentie van de ADC conversie zonder te sluiten en opnieuw te openen.**
ADC CLOSE	**Maakt de pinnen vrij voor normaal gebruik.**

OPTION	Beschrijving
ADC START array1!() [,array2!()] [,array3!()] [,array4!()]	**Dit start de conversie naar de opgegeven arrays.** De arrays moeten zwevend zijn punt en dezelfde grootte. De grootte van de arrays definieert het aantal conversies. De resultaten worden geretourneerd als een spanning tussen 0 en OPTION **VCC** . (standaard ingesteld op 3,3 V). RP2040 alleen zuivere **8bits ADC** door fout in dy. Start kan herhaaldelijk worden opgeroepen zodra de ADC OPEN is.
ADC START array1!() [,array2!()] [,array3!()] [,array4!()] [,Chan4arr!()] [,C1min] [,C1max] [,C2min] [,C2max] [,C3min] [,C3max] [,C4min] [,C4max]	**Hiermee wordt de conversie naar de opgegeven arrays gestart.** De arrays moeten een drijvende komma hebben en dezelfde grootte hebben. De grootte van de arrays bepaalt het aantal conversies. Start kan herhaaldelijk worden opgeroepen zodra de ADC OPEN is. **'Cxmin'** en **'Cxmax'** zullen de meetwaarden schalen. **C1min=200** en **C1max=100** creëren bijvoorbeeld waarden variërend van **200** tot **100** voor equivalente spanningen van 0 - 3,3 V. Als de schaling niet wordt gebruikt, worden de resultaten geretourneerd als een spanning tussen 0 en OPTION **VCC** (standaard ingesteld op 3,3 V).
ADC RUN array1%(), array2%)	**Voert de ADC continu uit in dubbel gebufferde modus.** De **ADC** vult eerst array1% en vervolgens array2% en dan terug naar array1% enz. Als er meer dan één **ADC-kanaal** is opgegeven in de opdracht ADC OPEN, worden de gegevens verweven. De gegevens worden geretourneerd als verpakte 8-bits waarden. (gebruik MEMORY **UNPACK** om naar een normale array te converteren). MM.INFO(ADC) retourneert het nummer van de buffer die momenteel beschikbaar is voor lezen (1 of 2).
ARC x, y, r1, [r2], a1, a2 [, c]	**Tekent een cirkelboog met een bepaalde kleur en breedte tussen twee radialen (gedefinieerd in graden)**. Parameters voor het ARC commando zijn: **x** : X-coördinaat van boogmidden **y** : Y-coördinaat van boogmidden **r1**: Binnenstraal van boog **r2**: Buitenste boogstraal kan worden weggelaten als 1 pixel breed **a1**: Beginhoek van boog in graden **a2**: Eindhoek van boog in graden **c** : Kleur van de boog (indien weggelaten zal deze standaard de voorgrondkleur hebben) Nul graden is op de 9 uur positie.
AUTOSAVE APPEND	**Voegt de inkomende seriële datastroom toe aan het bestaande programma.** Perfect voor het toevoegen van bibliotheekroutines aan nieuwe code.

COMMANDO	Beschrijving 161
AUTOSAVE of **AUTOSAVE CRUNCH**	**Ga naar de automatische programma invoermodus**. Deze opdracht zal tekstregels in beslag nemen van de seriële ingang van de console en sla ze op in het programmageheugen. **Deze modus** wordt beëindigd door **Control-Z** of **F1** in te voeren, wat dan zal leiden tot de ontvangen gegevens worden overgebracht naar het programmageheugen en overschrijven de vorig programma. **Gebruik F2** om af te sluiten en het programma onmiddellijk uit te voeren. **De CRUNCH-optie geeft MMBasic de opdracht om alle opmerkingen te verwijderen, blanco regels en onnodige spaties uit het programma voordat u het opslaat.** Dit kan zijn gebruikt op grote program-ma's zodat ze in een beperkt geheugen passen. **CRUNCH** kan wor-den afgekort tot de enkele **letter C**. Dit commando kan op elk mo-ment worden afgebroken door **Control-C** die zal verlaten program-mageheugen onaangeroerd. Dit is een manier om een BASIC-programma over te zetten naar de PicoMite. Het programma dat moet worden overgedragen, kan in een terminalemulator worden geplakt en dit commando zal de tekst-stroom vastleggen en opslaan in het programmageheugen. Het kan ook worden gebruikt om een klein programma direct bij de con-soleingang in te voeren.
BACKLIGHT n [,DEFAULT]	**NIET VGA- OF HDMI-VERSIES** **Stelt de achtergrondverlichting van het display in.** Geldige waarden zijn **0** tot **100**. Als **DEFAULT** is opgegeven dan zal de firmware bij het inschakelen automatisch de achtergrondverlichting op dat niveau omhoog instellen. Dit is met name handig bij gebruik op batterijen, waarbij de batterijduur wordt verminderd. Het achtergrondverlichtingsniveau kan de levensduur van de batterij aanzienlijk verlengen.
BITBANG	**Vervangen door het commando DEVICE** of door alleen **BITBANG** weg te laten in het programma. Voor compatibiliteit kan **BITBANG** nog steeds zijn gebruikt in pro-gramma's en wordt automatisch geconverteerd naar **DEVICE.**
BLIT	**BLIT is een eenvoudige geheugenbewerking waarbij van en naar een display of geheugen naar een display of geheugen wordt ge-kopieerd.** Opmerkingen: • Er zijn **32 buffers** beschikbaar, variërend van **#1** tot **#32**. • Bij het opgeven van het buffernummer is het **#** -symbool optioneel. • Alle andere argumenten zijn in pixels.

COMMANDO	Beschrijving 162
BLIT READ **[#] b, x, y, w, h**	**BLIT READ kopieert een deel van het display naar de geheugen-buffer '#b'.** De broncoördinaat is **'x'** en **'y'** en de breedte van het te kopiëren weergavegebied is **'w'** en de hoogte is **'h'**. Wanneer deze opdracht wordt gebruikt, wordt de geheugenbuffer automatisch aangemaakt en wordt er voldoende geheugen toege-wezen. Deze buffer kan worden vrijgegeven en het geheugen kan worden hersteld met het **BLIT CLOSE** commando.
BLIT WRITE **[#]b, x, y [, mode]%, x, y [,col]**	**BLIT WRITE kopieert de geheugenbuffer '#b' naar het display.** De bestemmingscoördinaat is **'x'** en **'y'.** De optionele parameter **'mode'** is standaard ingesteld op **0** en specificeert hoe de opgeslagen afbeeldingsgegevens worden ge-wijzigd terwijl deze worden weggeschreven. Het is de bitsgewijze **EN** van de volgende waarden: **&B001 = van links naar rechts gespiegeld.** **&B010 = van boven naar beneden gespiegeld.** **&B100 = transparante pixels niet kopiëren.**
BLIT LOAD [BMP] **[#]b, fname$** **[,x] [,y]** **[,w] [,h]**	**BLIT LOAD laadt een blitbuffer uit een 24-bits bmp-beeldbestand.** **x, y** definieert de startpositie in de afbeelding om te beginnen met laden en **w, h** specificeert de breedte en hoogte van het te laden gebied. Deze opdracht werkt op de meeste displaypanelen (niet alleen op panelen die de ILI9341-controller gebruiken). **BLIT LOAD #1, "image1", 50, 50, 100, 100** laadt bijvoorbeeld een gebied van **100** vierkante pixels met de linkerbovenhoek op **50, 50** van de afbeelding **image1.bmp**
BLIT CLOSE [#]b	**BLIT CLOSE sluit de geheugenbuffer '#b' zodat deze kan worden ge-bruikt voor nog een BLIT READ bewerking en herstel het gebruikte geheugen**. **BLIT nummer is geheugenafhankelijk.** Opmerkingen: • Er zijn max. **32** buffers beschikbaar, variërend van **#1** tot **#32**. • Bij het specificeren van het buffernummer is het **#** symbool op-tioneel. Alle andere argumenten zijn in pixels. **#8 => #32**
BLIT MERGE colour, x, y, w, h	**GEEN VGA OF HDMI VERSIES** **Kopieert een gebied van de framebuffer gedefinieerd door de pixelcoördinaten 'x' en 'y' linksboven en met een breedte van 'w' en hoogte 'h' naar het LCD-scherm.** Als onderdeel van de kopie wordt het **LCD-scherm** bedekt met **pixels** uit de laagbuffer die niet zijn ingesteld op de opgegeven **'kleur'**. De **kleur** wordt gespecificeerd als een **getal** tussen **0** en **15** dat staat voor: **Ned: Zwart, Blauw, Mirte, Kobalt, Middengroen, Cerulean, groen, cyaan, rood, magenta, roest, fuschia, bruin, lila, geel** en **wit.** Vereist dat zowel een framebuffer als een laagbuffer zijn gemaakt om te kunnen werken. Wacht automatisch tot het frame leeg is voordat het kopiëren wordt gestart op **ILI9341-, ST7789_320-** en **ILI9488** beeldschermen.

COMMANDO	Beschrijving 163
BLIT FRAMEBUFFER from, to, xin, yin, xout, yout, width, height [,colour]	**Kopieert een gebied van een specifieke 'van' framebuffer N, F of L naar een andere 'naar' framebuffer N, F of L.** **'xin'** en **'yin'** definiëren de linkerbovenhoek van het gebied van **'width'** en **' height'** op de bronframebuffer die moet worden gekopieerd. **'xout'** en **'yout'** definiëren de linkerbovenhoek van het gebied op de doelframebuffer waar de kopie moet worden ontvangen. De optionele parameter kleur definieert een pixelkleur op de bron die niet wordt gekopieerd. Als u dit weglaat, worden alle pixels gekopieerd. De kleur wordt opgegeven als een getal tussen **0** en **15** vertegenwoordigt: **Ned: Zwart, Blauw, Mirte, Kobalt, Middengroen, Cerulean, groen, cyaan, rood, magenta, roest, fuschia, bruin, lila, geel** en **wit.** Vereist dat zowel een framebuffer als een laagbuffer zijn gemaakt om te kunnen werken. Wacht automatisch tot het frame leeg is voordat het kopiëren wordt gestart op **ILI9341-, ST7789_320-** en **ILI9488** beeldschermen
BLIT MEMORY address, x, y [,col]	**Kopieert een geheugengebied dat wordt behandeld als een reeks kleurnibbles naar de huidige grafische uitvoer zoals gespecificeerd door FRAMEBUFFER WRITE.** De kleur wordt gespecificeerd als een getal tussen **0** en **15** dat staat voor: **Zwart, Blauw, Mirte, Kobalt, Middengroen, Cerulean, groen, cyaan, rood, magenta, roest, fuschia, bruin, lila, geel** en **wit.** Het eerste woord van het geheugengebied dat begint bij **'adres%'** moet de breedte en hoogte van het te kopiëren gebied bevatten als **16-bit** gehele getallen, met de **breedte** als de **onderste 16 bits**. Het adres moet worden uitgelijnd met een woordgrens (deelbaar door **4**). Als de optionele parameter 'col' is opgegeven, wordt die specifieke kleur niet gekopieerd. Als de **bovenste bit** van de **breedte** of **hoogte** is ingesteld op **1**, worden de kleurgegevens behandeld als gecomprimeerd (de resterende **15 bits** worden gebruikt als **breedte en/of hoogte**). Het compressie-algoritme is eenvoudig: elke byte bevat een telling in de onderste nibble (**1-15**) en een kleur in de bovenste nibble (**0-15**). In het geval dat meer dan **15 pixels** dezelfde kleur hebben, worden extra bytes voor die kleur gebruikt.
BLIT COMPRESSED address%, x, y [,col]	**Werkt hetzelfde als BLIT MEMORY, maar gaat ervan uit dat de gegevens zijn gecomprimeerd en negeert het bovenste bit in de breedte en hoogte.**

COMMANDO	Beschrijving 164
BLIT x1,y1,x2,y2,w,h	**Kopieer een gedeelte van het beeldscherm naar een ander deel van het beeldscherm.** De broncoördinaat is **'x1'** en **'y1'**. De bestemmingscoördinaat is **'x2'** en **'y2'**. De breedte van het te kopiëren schermgebied is **'w'** en de hoogte is **'h'**. Alle argumenten zijn in pixels. Als de uitvoer naar een LCD-paneel gaat, moeten dit de **SSD19863, ILI9341_8, ILI9341, ILI9488** (indien MISO aangesloten) of **ST7789_320** controllers zijn.
BOX x, y, w, h [,lw] [,c] [,fill]	**Tekent een vak op het bijgevoegde LCD-paneel met de linkerboven hoek op 'x' en 'y' met een breedte van 'w' pixels en een hoogte van 'h' pixels**. **'lw'** is de breedte van de zijkanten van de doos en kan **nul** zijn. Het staat standaard op **1**. **'c'** specificeert de kleur en wordt **standaard ingesteld** op de standaard voorgrondkleur als dat niet het geval is gespecificeerd. **'fill'** is de vulkleur. Het kan worden weggelaten of ingesteld op **-1**, in welk geval de doos wordt niet gevuld. Alle parameters kunnen worden uitgedrukt als **arrays** en de software **plot de aantal dozen** zoals bepaald door de afmetingen van de **kleinste array**. **'x', 'y', 'w'** en **'h'** moeten **allemaal arrays** zijn of **allemaal enkele variabelen /constanten** anders wordt er een **fout** gegenereerd. **'lw', 'c'** en **fill** kunnen ofwel **arrays** ofwel zijn enkele **variabelen / constanten.** Zie het hoofdstuk **"Basis Tekenopdrachten"** voor een definitie van de **kleuren** en **grafische coördinaten**.
CALL usersubname$ [,usersubparameters,.]	**Dit is een efficiënte manier om door de gebruiker gedefinieerde subroutines programmatisch aan te roepen (zie ook de CALL( ) functie).** In veel gevallen kan het u toelaten om zich te ontdoen van complexe SELECT- en IF THEN ELSEIF ENDIF clausules en wordt verwerkt in een veel efficiëntere manier. De **"usersubname$"** kan elke tekenreeks of variabele of functie zijn die wordt opgelost naar de naam van een normale gebruikers subroutine (geen ingebouwde opdracht). De **"usersubparameters"** zijn dezelfde parameters die zouden worden gebruikt om de subroutine direct aan te roepen. Een typisch gebruik zou het schrijven van elke soort emulator kunnen zijn waar een van een groot aantal subroutines moet worden aangeroepen, afhankelijk van een of ander variabele. Het maakt het ook mogelijk om de naam van een subroutine door te geven aan een ander subroutine of functioneren als een variabele.

COMMANDO	Beschrijving 165
CAMERA	**NIET VGA OF HDMI** **Commando ter ondersteuning van de OV7670 cameramodule.**
CAMERA OPEN XLKpin, PLKpin, HSpin, VSCpin, RETpin, D0pin	**Hierdoor wordt de camera geïnitialiseerd en wordt een klok van 12 MHz op XLK (PWM) uitgevoerd controleert of de signalen op PLK, VS en HS correct worden ontvangen.** De camera is ingesteld op een resolutie van **160 x 120 (QQVGA)**, wat het maximaal haalbare is binnen de grenzen van het beschikbare geheugen. Schakel OPTION **SYSTEM** I2C in de PicoMite-firmware in en sluit **SCL** en **SDA** aan op de relevante pinnen (mogelijk voorzien van het label **SIOC** en **SIOD** op de cameramodule). Deze aansluitingen moeten een **pullup** naar 3,3V hebben - **2K7 aanbevolen**). Andere pinnen zijn bedraad volgens het OPEN commando. (NB: **VS** kan het label **VSYNC** hebben, **HS** kan het label **HREF** hebben, **PLK** kan het label **PCLK** hebben, **RET** kan het label **RESET** hebben en **XLK** kan **XCLK** zijn op uw module). D0 pin definieert het begin van een reeks van 8 aaneengesloten pinnen (bijvoorbeeld GP0 -GP7).
CAMERA CAPTURE [scale, [x , y]]	**Hiermee wordt een foto van de camera (RGB565) gemaakt en weergegeven op een LCD-scherm.** Er moet een SPI **LCD** zijn aangesloten en ingeschakeld om de opdracht te laten werken. (**ILI9341** en **ST7789_320** aanbevolen). De **schaal** is standaard ingesteld op **1** en **x, y** elk op **0.** Standaard wordt een afbeelding van **160 x 120** weergegeven op het **LCD-scherm**, met de linkerbovenhoek op **0, 0** op het LCD-scherm. Als u de **schaal** instelt op **2**, wordt een scherm van **320 x 240** met de afbeelding gevuld. Als u de **x-** en **y-**parameters instelt, wordt de linkerbovenhoek van het beeld op het LCD-scherm verschoven. De updatesnelheid in een continue lus is **7FPS** op het scherm op **schaal 1:1** en **5FPS** geschaald naar **320 x 240**. Ervan uitgaande dat het beeldscherm is voorzien van **MISO**-bekabeling, is het dan mogelijk om de afbeelding op schijf op te slaan met behulp van de opdracht SAVE **IMAGE**, zoals gebruikt om de bovenstaande voorbeeldafbeelding te maken.
CAMERA CLOSE	**Sluit het camera subsysteem aan en maakt alle pinnen vrij die zijn toegewezen in het OPEN commando.**
CAMERA CHANGE image%(), change! [,scale [, x, y]]	**De camerafirmware kan met behulp van de opdracht ook beweging in het gezichtsveld van de camera detecteren.** Dit gebeurt door de camera in de **YUV-modus** te gebruiken in plaats van in **RGB**. Dit heeft het voordeel dat de intensiteitsinformatie en kleurinformatie gescheiden zijn en dat er slechts één byte nodig is voor een grijswaardenbeeld met **256 niveaus** dat ideaal is voor het detecteren van beweging. **image%** is een array van **160 x 120 bytes.** (DIM afbeelding **%(160, 120 / 8 - 1)**

COMMANDO	Beschrijving 166
	Bij het aanroepen van de opdracht bevat het een ingepakte **8-bits** grijswaarden afbeelding. De verandering! variabele retourneert het percentage dat de afbeelding is veranderd ten opzichte van de vorige keer dat de opdracht werd aangeroepen. Optioneel, als **"schaal"** is ingesteld, wordt de beelddelta naar het scherm weergegeven, d.w.z. het verschil tussen het vorige beeld en dit beeld. Net als bij het **CAPTURE** commando kan het deltabeeld naar wens worden geschaald en gepositioneerd. Als de schaalparameter wordt weggelaten, wordt het **LCD-scherm** niet bijgewerkt door deze opdracht.
`CAMERA TEST tnum`	**Schakelt een testsignaal van de camera in of uit.** **tnum = 2** genereert kleurenbalken. **tnum = 0** keert terug naar de visuele invoer. **tnum = 1** en **tnum = 3** doen iets, maar wat?
`CAMERA REGISTER reg%, data%`	**Stelt het register "reg%" in de camera in op de waarde "data%".** Bij gebruik rapporteert de opdracht de vorige waarde aan de console en bevestigt automatisch dat de nieuwe waarde is ingesteld zoals gevraagd. De kleurweergave van de camera zoals geïnitialiseerd is redelijk, maar kan waarschijnlijk nog verder worden verbeterd door de verschillende cameraregisters af te stemmen.
`CAT S$, N$`	**Voegt de strings samen door N$ toe te voegen aan S$**. Dit is functioneel de zelfde a **S$** = **S$** + **N$** maar werkt sneller.
`CHDIR dir$`	**Wijzig de huidige werkmap op de SD-kaart in 'dir$'** Het speciale item **".."** vertegenwoordigt het bovenliggende item van de huidige map en **"."** staat voor de huidige directory. **"/"** is de hoofdmap.
`CIRCLE x, y, r[, lw] [, a][, c][, fill]`	**Teken een cirkel op de video-uitgang gecentreerd op 'x' en 'y' met een straal van 'r' op het bijgevoegde LCD-paneel.** **'lw'** is optioneel en is de lijnbreedte (standaard **1**). **'c'** is de optionele kleur en wordt standaard ingesteld op de huidige voorgrondkleur als dat niet het geval is gespecificeerd. **'a'** is een optionelegetal met drijvende komma dat de beeldverhouding definieerd. Als het aspect niet is gespecificeerd, is het standaard **1.0** wat standaard een cirkel geeft. **'fill'** is de vulkleur. Het kan worden weggelaten of ingesteld op **-1** waarin in het geval dat de doos niet wordt gevuld. Alle parameters kunnen worden uitgedrukt als arrays en de software plot een aantal cirkels zoals bepaald door de afmetingen van de kleinste array. **'x'**, **'y'** en **'r'** moeten allemaal arrays zijn of allemaal enkele variabelen/constanten, anders zal een fout worden gegenereerd. **'lw'**, **'a'**, **'c'** en fill kunnen zowel arrays als single zijn variabelen / constanten. Zie het hoofdstuk "Basis Tekenopdrachten" voor een definitie van de kleuren en grafische coördinaten.

COMMANDO	Beschrijving 167
CLEAR	**Verwijder alle variabelen en herstel het geheugen dat ze gebruiken.** Zie ERASE voor het verwijderen van specifieke arrayvariabelen.
CLOSE [#]fnbr [,[#] fnbr] ...	**Sluit de eerder geopende bestanden met het bestandsnummer '#fnbr'. De # is optioneel**. Zie ook het commando OPEN.
CLS [colour]	**Wist het scherm van het LCD-paneel**. Optioneel kan **'kleur'** worden opgegeven welke wordt gebruikt voor de achtergrond bij het wissen van het scherm.
CMM2 LOAD fname$ [,R] of CMM2 RUN fname$ #DEFINE #INCLUDE #COMMENT START/END #MMDEBUG ON/OFF	**Laadt en/of voert een programma uit vanaf schijf met behulp van het CMM2 programmalaadmechanisme.** Dit omvat een agressieve aanpak van het programma en ondersteunt **#INCLUDE** bestanden en **#DEFINE** tekstvervanging. Dit kan worden gebruikt voor compatibiliteit met **CMM2** programma's of om het structureren van programma's in afzonderlijke modules mogelijk te maken. Het is belangrijk op te merken dat, indien gebruikt, alle bewerkingen van programma's offline of rechtstreeks van en naar schijf moeten plaatsvinden, aangezien de bronbestanden niet kunnen worden gereconstrueerd op basis van de versie die door deze opdrachten wordt geladen.
COLOUR fore [, back] of COLOR fore [, back]	**Stelt de standaardkleur in voor opdrachten (PRINT, enz.) die worden weergegeven op de op de aangesloten LCD-paneel**. **'fore'** is de voorgrondkleur, **'back'** is de achtergrond kleur. De achtergrond is optioneel en als deze niet is opgegeven, wordt deze standaard zwart.
COLOUR MAP inarray%(), outarray%()[, colourmap%()]	**Deze opdracht genereert RGB888-kleuren in outarray% uit kleurcodes (0 - 15) in inarray%.** Als de optionele parameter **colourmap%** wordt gebruikt, moet deze **16 elementen** lang zijn). In dit geval worden de waarden in **array%** toegewezen aan de kleuren voor die indexwaarde in **colourmap%**
CONFIGURE cfg	**Configureert een bord volgens het door "cfg" gespecificeerde equivalent van OPTION RESET).**
CONFIGURE LIST	**Geeft een overzicht van alle verschillende configuraties die beschikbaar zijn voor de firmwareversie.**
CONST id = expression [, id = expression]... etc	**Maak een constante id aan die niet kan worden gewijzigd nadat deze is gemaakt**. **'id'** is de identifier die dezelfde regels volgt als voor variabelen. De identifier kan een typeachtervoegsel hebben (**!, %** of **$**), maar dit is niet vereist. Als het is opgegeven, moet het overeenkomen met het type **'expressie'**. **'expressie'** is de waarde van de identifier en het kan een normale uitdrukking zijn (inclusief door de gebruiker gedefinieerde) functies) die worden geëvalueerd wanneer de constante wordt gemaakt. Een constante gedefinieerd buiten een sub of functie is globaal en kan worden gezien het hele programma. Een constante gedefinieerd binnen een sub of functie is lokaal aan die routine en zal een globale constante met dezelfde naam verbergen.

COMMANDO	Beschrijving 168
CONTINUE	**Hervat het uitvoeren van een programma dat is gestopt door een END instructie, en fout of CTRL-C**. Het programma zal herstarten met de volgende instructie volgend op het vorige stoppunt. Merk op dat het niet altijd mogelijk is om het programma correct te hervatten. Dit is met name van toepassing op complexe programma's met afbeeldingen, geneste lussen en/of geneste subroutines en functies.
CONTINUE DO of CONTINUE FOR	**Ga naar het einde van een DO/LOOP of een FOR/NEXT lus**. De lus voorwaarde wordt dan getest en indien nog steeds geldig, gaat de lus verder met de volgende iteratie.
COPY fname1$ To fname2$	**Kopieer een bestand van 'fname1$' naar 'fname2$'.** Beide zijn strings. Een directorypad kan zowel in **'fname$'** als in **'fname$'** worden gebruikt. Als de paden anders wordt het bestand gespecificeerd in **'fname$'** gekopieerd naar het pad gespecificeerd in **'fname2$'** met de bestandsnaam zoals gespecificeerd. v.b.1 `Copy "a:/*.*" To "b:/backup"` v.b.2 `Copy "b:/backuprp2040/*.*" To "a:"`
COPY fname$ TO dirname$	**Kopie met jokertekens.** De bulkkopie wordt geactiveerd als **fname$** een **'*'** of een **'?'** karakter. **dirname$** moet een geldige mapnaam zijn en mag **NIET** eindigen op een schuine streep.
COPY A2B fname1$ To fname2$	**Kopieert bestand van flash naar sd-kaart.** v.b. `Copy fname$ To dirname$` De bulkkopie wordt geactiveerd als **fname$** een **'*'** of een **'?'** bevat karakter **dirname$** moet een geldige mapnaam zijn en mag **NIET** eindigen op een schuine streep.
COPY B2A fname1$ To fname2$	**Kopieert bestand van sd-kaart naar flash.** v.b. `COPY fname$ To dirname$` De bulkkopie wordt geactiveerd als **fname$** een **'*'** of een **'?'** bevat karakter **dirname$** moet een geldige mapnaam zijn en mag **NIET** eindigen op een schuine streep.
CPU RESTART	**Forceert een herstart van de processor.** Hiermee worden alle variabelen gewist en alles opnieuw ingesteld (bijv. **timers**, **COM poorten**, I2C, enz.) **vergelijkbaar** met een opstartsituatie, maar zonder de opstartbanner. Als OPTION AUTORUN is ingesteld, is het programma in de **gespecificeerde flits locatie** wordt opnieuw gestart.
CPU SLEEP n	**Zorgt ervoor dat de processor 'n' seconden slaapt.** Merk op dat de CPU geen echte slaapstand voor laag energieverbruik heeft, dus de energiebesparing is gelimiteerd.

COMMANDO	Beschrijving
CSUB **name [type** **[, type] ...]** **hex [[hex[...]** **hex [[hex[...]** **END CSUB**	**Definieert de binaire code voor een embedded machinecode programmamodule geschreven in C of ARM assembler**. De module verschijnt in **MMBasic** als de commando **'naam'** en kan op dezelfde manier worden gebruikt als een ingebouwd commando. Meerdere ingebedde routines kunnen in een programma worden gebruikt, waarbij elk een andere module met een andere **'naam'**. Het eerste **'hex'**-woord is **een 32-bits** woord dat vanaf het begin de offset in bytes is van de **CSUB** naar het ingangspunt van de ingebedde routine (meestal de functie voornaamst()). De volgende hexadecimale woorden zijn de gecompileerde binaire code voor de module Deze worden automatisch in MMBasic geprogrammeerd wanneer het programma wordt opgeslagen. Elke **'hex'** moet exact acht hexadecimale cijfers zijn die staan voor de bits in een **32-bits** woord en worden gescheiden door een of meer spaties of nieuw lijnen. De opdracht moet worden beëindigd met een overeenkomende **END CSUB**. Alle fouten in het gegevensformaat worden gerapporteerd wanneer het programma wordt uitgevoerd. **Tijdens de uitvoering** zal **MMBasic alle CSUB** commando's overslaan, zodat ze kunnen overal in het programma worden geplaatst. Het type van elke parameter kan worden gespecificeerd in de definitie. Bijvoorbeeld: **CSUB MySub Integer, Integer, String** Dit specificeert dat er drie parameters zullen zijn, de eerste twee zijn gehele getallen en de derde een string. Opmerking: • Er kunnen maximaal tien argumenten worden opgegeven (**'arg1', 'arg2'**, enz.). • Als een variabele of array als argument is opgegeven, zal de **C routine** ontvang een verwijzing naar het geheugen dat is toegewezen aan de **variabele** of **array** en de **C routine** kan dit geheugen wijzigen om een waarde terug te sturen naar de beller In het geval van **arrays** moeten ze worden doorgegeven met lege haakjes, b.v. **arg()**. In de **CSUB** wordt het **argument geleverd** als een verwijzing naar de eerste element van de array. • **Constanten** en **uitdrukkingen** worden doorgegeven aan de ingebedde C-routine als verwijzingen naar een tijdelijke geheugen ruimte die de waarde bevat.
CTRLVAL(#ref) =	**Deze opdracht stelt de waarde van een geavanceerdbesturingselement in**. **'#ref'** is het referentienummer van de besturing. Voor aan/uit-bedieningen zoals selectievakjes zal het elke aanraakinvoer overschrijven en kan worden gebruikt om schakelaars in/uit te schakelen, selectievakjes in/uit te schakelen, enz. Een waarde van nul is uit of niet aangevinkt en niet-nul zal de besturing aanzetten. Voor een LED het zal ervoor zorgen dat de LED oplicht of uitgaat. Het kan ook worden gebruikt om in te stellen de beginwaarde van spinboxen, tekstvakken, enz.

	Bijvoorbeeld: `CTRLVAL(#10) = 12.4` Alle besturingselementen verwachten een nummer toegewezen te krijgen (**Float** of **Integer**) behalve **Frame**, **Bijschrift**, **Display Box**, **Text Box** en **Format Box** die een string verwachten. **Niet op de PicoMiteVGA**.
`DATA constant[, constant]...`	**Slaat numerieke en string constanten op die toegankelijk zijn voor `READ`.** In het algemeen moeten stringconstanten worden omgeven door dubbele aanhalingstekens (**"**). Uitzondering is wanneer de tekenreeks alleen uit alfanumerieke tekens bestaat die dat wel doen geen MMBasic-trefwoorden vertegenwoordigen (zoals `THEN`, `WHILE`,enz) In dat case quotes zijn niet nodig. Numerieke constanten kunnen ook uitdrukkingen zijn zoals **5 * 60.**
`DATE$ = "DD-MM-YY[YY]"` of `DATE$ = "DD/MM/YY[YY]"` of `DATE$ = "YYYY-MM-DD"` of `DATE$ = "YYYY/MM/DD"`	**Stel de datum van de interne klok/kalender in.** DD, MM en YY zijn getallen, bijvoorbeeld: `DATE$ = "28-7-2014"` De datum is ingesteld op **"01-01-2000"** bij het opstarten.
`DEFINEFONT #Nbr` `  hex [[ hex[…]` `  hex [[ hex[…]` `END DEFINEFONT`	**Dit definieert een ingesloten lettertype dat kan worden gebruikt naast of om te vervangen de ingebouwde font(s) die op een aangesloten LCD paneel worden gebruikt.** Deze werken precies hetzelfde als de ingebouwde lettertypen (d.w.z. geselecteerd met het `FONT` commando of gespecificeerd in het `TEXT` commando). Zie de map Embedded Fonts in het PicoMite-distributiezipbestand voor een selectie van ingesloten lettertypen en een volledige beschrijving van hoe u ze kunt maken. **'#Nbr'** is het referentienummer van het lettertype (**van 1 tot 16**). Het kan hetzelfde zijn nummer als een ingebouwd lettertype en in dat geval zal het het ingebouwde lettertype vervangen. Elke 'hex' moet exact acht hexadecimale cijfers zijn en worden gescheiden door spaties of nieuwe regels van de volgende. • Meerdere regels met 'hexadecimale' woorden kunnen worden gebruikt met het commando beëindigd door een bijpassende `END DEFINEFONT`. • Meerdere ingesloten lettertypen kunnen in een programma worden gebruikt, waarbij elk een ander lettertype met een ander lettertypenummer. • Tijdens uitvoering zal MMBasic elke `DEFINEFONT` overslaan commando's zodat ze overal in het programma kunnen worden geplaatst. • Eventuele fouten in het gegevensformaat worden gerapporteerd wanneer het programma is opgeslagen.

COMMANDO	Beschrijving
DEVICE BITSTREAM pinno, n_transitions, array%()	**Dit commando wordt gebruikt om een uiterst nauwkeurige bitreeks op de opgegeven pin te genereren.** De pin moet vooraf als uitgang zijn ingesteld en op het gewenste startniveau zijn ingesteld. Opmerkingen: • De array bevat de lengte van elk niveau in de bitstream microseconden. De maximaal toegestane periode is 65,5 mSec • De eerste transitie zal onmiddellijk plaatsvinden bij het uitvoeren van de commando. • De laatste periode in de array wordt genegeerd, afgezien van het definiëren van de tijd voordat de besturing terugkeert naar het programma of de opdrachtregel. • De pin blijft in de startstatus als het aantal overgangen gelijk is even en de tegenovergestelde toestand als het aantal overgangen oneven is.
DEVICE CAMERA	**Zie CAMERA commando**
DEVICE GAMEPAD	**Zie GAMEPAD commando**
DEVICE HUMID	**Zie HUMID commando**
DEVICE KEYPAD	**Zie KEYPAD commando**
DEVICE MOUSE	**Zie MOUSE commando**
DEVICE LCD	**Zie LCD commando**
DEVICE SERIALTX pinno, baudrate, ostring$	**Voert 'ostring$' uit als een seriële datastroom op 'pinno'.** **'baudrate'** kan tussen **110** en **230400** liggen (bij **230400** moet de **CPU** mogelijk overgeklokt worden). Houd er rekening mee dat het programma tijdens de verzending stopt en onderbrekingen negeert.
DEVICE SERIALRX pinno, baudrate, istring$, timeout_in_ms, status% [,nbr] [,terminators$]	**Voert seriële gegevens in op 'pinno'. 'baudrate' kan tussen 110 en 230400 liggen (Voor de 230400 moet de CPU mogelijk worden over-geklokt).** **'status%'** retourneert: **-1** = time-out (Opmerking: gebruik **LEN(istring$)** om het aantal tekens te zien ontvangen) **2** = aantal gevraagde tekens tevreden **3** = eindteken tevreden **'nbr'** specificeert het aantal tekens dat moet worden ontvangen voordat het commando terugkeert. **'terminators$'** specificeert een of meer tekentekens die kunnen worden gebruikt om de ontvangst te beëindigen. Het programma stopt en interrupts worden genegeerd terwijl deze opdracht wordt uitgevoerd.
DEVICE WII	**Zie WII commando**
DEVICE WS2812	**Zie WS2812 commando**

COMMANDO	Beschrijving 172
DIM [type] decl[,decl] ... waar **'decl'** is **var [length] [type] [Init]** **'Var'** is een variabele naam met optionele afmetingen	**Declareert een of meer variabelen (d.w.z. maakt de naam van de variabele en zijn kenmerken bekend bij de tolk)**. Wanneer **OPTION EXPLICIT** wordt gebruikt (zoals aanbevolen) worden de **DIM**, **LOCAL** of **STATIC** opdrachten zijn de enige manier waarop een variabele kan worden gemaakt. Als dit optie niet wordt gebruikt, is het gebruik van het **DIM** commando optioneel en indien niet gebruikt de variabele wordt automatisch gemaakt wanneer er voor het eerst naar wordt verwezen.
'Length' wordt gebruikt om de maximale grootte in te stellen van de tekenreeks naar **'n'** zoals in **LENGTE** nr	Het type van de variabele (d.w.z. **String**, **Float** of **Integer**) kan in één worden gespecificeerd op drie manieren.Door een type achtervoegsel te gebruiken (d.w.z. **!, %** of **$** voor **Float**,**Integer** of **String**). Voor voorbeeld: **DIM nbr%, amount!, name$** **' nbr% = Integer, amount! = Fload, name$ = String**
'type' is een van **FLOAT** of **INTEGER** of **STRING** (het type kan worden vooraf-gegaan door het trefwoord **AS** - zoals in **AS FLOAT**) **'Init'** is de waarde om de variabel te initialiseren en bestaat uit: **= <expressie>**	Door onmiddellijk een van de trefwoorden **FLOAT**, **INTEGER** of **STRING** te gebruiken na het commando **DIM** en voordat de variabele(n) worden vermeld. de opgegeven type is dan van toepassing op alle vermelde variabelen (d.w.z. het hoeft niet herhaald te worden). Bijvoorbeeld: **DIM STRING voornaam, achternaam, stad** Door de Microsoft-conventie te gebruiken voor het gebruik van het trefwoord **"AS"** en het type trefwoord (d.w.z. **FLOAT**, **INTEGER** of **STRING**) na elke variabele als jij gebruik deze methode het type moet worden opgegeven voor elke variabele en kan worden veranderd van variabel naar variabel.
Voor een eenvoudige variabele uitdrukking wordt gebruikt, voor een array lijst met door komma's gescheiden uitdrukkingen omgeven door haakjes wordt gebruikt.	Bijvoorbeeld: **DIM bedrag AS FLOAT, naam AS STRING** Variabelen met drijvende komma of integer worden bij het maken op nul gezet en strings worden ingesteld op een lege String (d.w.z. ""). U kunt de waarde van de variabele initialiseren met iets anders door een gelijkteken (=) en een uitdrukking te gebruiken die de variabele definitie volgt. Bijvoorbeeld: **DIM STRING city= "Perth", house= "Brick"**
Voorbeelden: **DIM nbr(50)** **DIM INTEGER nbr%(50)** **DIM name AS STRING** **DIM a, b$, nbr(100), Strn$(20)** **DIM a(5,5,5),b(1000)** **DIM Strn$(200)** **LENGTH 20** **DIM STRING Strn(200)** **LENGTE 20** **DIM a=1234, b=345** **DIM STRING Strn="tekst"** **DIM %(3)=(11,22,33,44)**	De initialisatiewaarde kan een uitdrukking zijn (inclusief andere variabelen) en wordt geëvalueerd wanneer het **DIM** commando wordt uitgevoerd. Zie het hoofdstuk "Variabelen definiëren en gebruiken" voor meer voorbeelden van de syntaxis. Naast het declareren van eenvoudige variabelen, zal het **DIM** commando ook declareren array-variabelen (d.w.z. een geïndexeerde variabele met een aantal dimensies). Na de naam van de variabele worden de afmetingen gespecificeerd door een lijst van getallen gescheiden door komma's en tussen haakjes geplaatst. Bijvoorbeeld: **DIM array(10, 20)** Elk getal geeft het aantal elementen in elke dimensie aan.

Normaal gesproken de nummering van elke dimensie begint bij **0**, maar de **OPTION BASE** commando kan worden gebruikt om dit te veranderen commando kan worden gebruikt om dit te veranderen in **1**. Het bovenstaande voorbeeld specificeert een tweedimensionale array met **11** elementen (**0** tot **10**) in de eerste dimensie en **21** (**0** tot **20**) in de tweede dimensie. De totaal aantal elementen is **231** en omdat elk getal met drijvende komma aan de PicoMite heeft **8 bytes nodig**, een totaal van **1848 bytes geheugen** zal zijn toegewezen.
Strings zullen **standaard 255 bytes** (d.w.z. tekens) geheugen toewijzen voor elk element en dit kan snel geheugen verbruiken bij het definiëren van arrays van strings. In dat geval kan het sleutelwoord **LENGTH** worden gebruikt om de hoeveelheid geheugen die aan elk element moet worden toegewezen en daarom de maximale lengte van de string die kan worden opgeslagen. Deze toewijzing ('n') kan zijn van **1** tot **255** tekens.
Bijvoorbeeld: `DIM STRING s(5, 10)` declareert een stringarray met **66** elementen die **16.896 bytes geheugen** verbruiken terwijl:
`DIM STRING s(5, 10) LENGTH 20` **6 x 11 x 21 = 1386**
Zal slechts **1.386 bytes geheugen** verbruiken. Merk op dat het bedrag van geheugen toegewezen voor elk element is **n + 1** omdat de extra byte wordt gebruikt om te volgen de werkelijke lengte van de string die in elk element is opgeslagen.
Als een string langer dan **'n'** wordt toegewezen aan een element van de **array**, zal er een fout optreden geproduceerd worden. Anders dan dit, string-arrays gemaakt met de **LENGTH** trefwoord werkt precies hetzelfde als andere tekenreeksarrays.
Dit trefwoord kan ook worden gebruikt met nietarray tekenreeksvariabelen, maar het zal geen geheugen besparen.
In het bovenstaande voorbeeld kunt u ook de Microsoft-syntaxis van specificeren gebruiken het type na de lengte-kwalificatie.
Bijvoorbeeld: `DIM s(5, 10) LENGTH 20 AS STRING`
Arrays kunnen ook worden geïnitialiseerd wanneer ze worden gedeclareerd door een gelijk aan toe te voegen symbool (=) gevolgd door een lijst met waarden tussen haakjes aan het einde van de verklaring
Bijvoorbeeld: `DIM INTEGER nbr(4) = (22,44,55,66,88)` of
`DIM s$(3) = ("foo", "boo", "doo", "zoo")`
Merk op dat het aantal initialisatiewaarden moet overeenkomen met het aantal elementen in de array inclusief de basiswaarde ingesteld door **OPTION BASE**. Als een multidimensionale array is geïnitialiseerd, dan zal de eerste dimensie zijn eerst geïnitialiseerd gevolgd door de tweede, enz.
Merk ook op dat de initialisatiewaarden na de **LENGTH** kwalificatie moeten staan (als gebruikt) en na de typeverklaring.
(indien gebruikt)

COMMANDO	Beschrijving 174
DO <statements> LOOP	**Deze structuur loopt voor altijd door, het EXIT DO commando kan worden gebruikt om de lus beëindigen of de besturing moet expliciet worden overgedragen buiten de loop door commando's zoals GOTO of EXIT SUB (indien in een subroutine)**.
DO WHILE expression <statements> LOOP	**Loopt terwijl 'expression' waar is (dit is gelijk aan de oudere WHILE WEND lus)**. Als in het begin de uitdrukking onwaar is,worden de uitspraken in de lus wordt niet uitgevoerd, zelfs niet één keer.
DO <statements> LOOP UNTIL expression	**Loopt door totdat de uitdrukking die volgt op UNTIL** waar is. **Omdat de test** is gemaakt aan het **einde van de lus**, worden de **instructies binnen de lus** uitgevoerd op minstens **één keer**, **zelfs** als de **uitdrukking waar is**.
DRAW3D	**GEEN WEBMITEVERSIE** **De 3D-engine bevat opdrachten voor het manipuleren van 3D-beelden, inclusief het instellen van de camera, maken, verbergen, roteren, enz.** Zie het document **"The CMM2 3D engine.pdf"** in de PicoMite-firmwaredownload voor een volledige beschrijving.
DRIVE drive$	**DRIVE drive$ kan "a:" of "b:" zijn, waarbij a: de flashdrive is en b: beschikbaar als er een SD-kaart is ingeschakeld.** Dit stelt het stan-daardstation in waar commando's zoals: **LOAD, SAVE, FILE, XMODEM** etc. worden uitgevoerd. NB vergelijkbaar met originele Maximite. Bij het opstarten is de standaardschijf de **SD-kaart**, indien ingeschakeld, anders de **flashdrive**. Gebruik **DRIVE "A:/FORMAT"** om de flashdrive opnieuw te initialiseren.
EDIT of EDIT FILE EDIT fname$	**Roep de editor op volledig scherm op.** **Als er een bestandsnaam is opgegeven, laadt de editor het bestand vanaf de huidige schijf (A: of B:) om bewerking mogelijk te maken en bij het afsluiten met F1 of F2 wordt het bestand op de schijf opgeslagen.** Als het bestand niet bestaat, wordt het bij het afsluiten aangemaakt. Het huidige programma dat in het flashgeheugen is opgeslagen, wordt niet beïnvloed. Als u een bestaand bestand bewerkt, wordt er bij het afsluiten ook een back-up met .bak aan de bestandsnaam gemaakt. **Als fname$ een andere extensie dan .bas bevat, wordt de kleurcodering tijdelijk uitgeschakeld tijdens de bewerking.** Als er geen extensie is opgegeven, gaat de firmware uit van **.bas** Door een bestand vanaf schijf te bewerken, kunnen nietbasisbestanden zoals **HTML**- of **sprite**-bestanden zonder corruptie worden bewerkt tijdens het tokeniseringsproces dat plaatsvindt wanneer het in flash wordt opgeslagen. **EDIT** en **EDIT fname$** kunnen alleen worden aangeroepen vanaf de opdrachtprompt.

COMMANDO	Beschrijving 175
EDIT FILE fname$	**Als u echter een bestand in een programma wilt bewerken, kunt u de opdracht EDIT FILE fname$ gebruiken.** Dit verschilt van **EDIT fname$** doordat het geen variabelen wist en alleen vrij geheugen kan gebruiken voor de bewerkingsbuffer. Dit zal grotere beperkingen stellen aan de grootte van het bestand dat kan worden bewerkt als het geheugengebruik van het aanroepende programma zelf groot is. Zie het hoofdstuk Volledig scherm-editor voor details over het gebruik van de editor.
ELSE	**Introduceert een optionele standaardvoorwaarde in een IF instructie met meerdere regels**. Zie de meerregelige **IF** verklaring voor meer details.
ELSEIF expression THEN of ELSE IF expression THEN	**Introduceert een optionele secundaire voorwaarde in een IF instructie met meerdere regels**. Zie de meerregelige **IF** verklaring voor meer details.
END	**Beëindig het lopende programma en keer terug naar de opdrachtprompt.**
END [noend] of END cmd$	**Beëindig het actieve programma en keer terug naar de opdrachtprompt.** **Als er een subroutine met de naam MM.END in het programma bestaat, wordt deze uitgevoerd telkens wanneer het programma eindigt met een feitelijk of impliciet END commando.** Het wordt niet uitgevoerd als het programma eindigt met het breakteken (dwz **Ctrl-C**). De optionele parameter **'noend'** kan worden gebruikt om de uitvoering van de **MM.END** subroutine te blokkeren, bijvoorbeeld **"END noend"** als **'cmd$'** is opgegeven, wordt deze uitgevoerd alsof het vanaf de opdrachtprompt is nadat het programma is voltooid. Let op: als **"END cmd$"** maar een subroutine wordt gebruikt **MM.END** bestaat, het wordt uitgevoerd en **cmd$** wordt genegeerd.
END CSUB	**Markeert het einde van een C-subroutine.** Zie het **CSUB** commando. Elke **CSUB** moet één en slechts één overeenkomende **END CSUB** instructie hebben.
END FUNCTION	**Markeert het einde van een door de gebruiker gedefinieerde functie**. Zie de opdracht **FUNCTION**. Elke functie moet één en slechts één passende **END FUNCTION** hebben stelling. Gebruik **EXIT FUNCTION** als u van een functie moet terugkeren vanuit zijn lichaam.
ENDIF of END IF	**Beëindigt een IF instructie met meerdere regels**. Zie de meerregelige **IF** verklaring voor meer details.

COMMANDO	Beschrijving 176
END SUB	**Markeert het einde van een door de gebruiker gedefinieerde sub-routine**. Zie het SUB commando. Elke sub moet één en slechts één overeenkomende END SUB instructie hebben. Gebruiken EXIT SUB als je vanuit een subroutine moet terugkeren vanuit zijn lichaam.
ERASE variabele [,variabele]...	**Wist variabelen en maakt het toegewezen geheugen vrij**. Dit zal werken met array-variabelen en normale (niet-array) variabelen. Arrays kunnen zijn gespecificeerd met lege haakjes(bijv.**dat()**) of gewoon door de naam op te geven van de variabele (bijv. **dat**). Gebruik CLEAR om alle variabelen tegelijk te verwijderen (inclusief arrays).
ERROR [error_msg$]	**Forceert een fout en beëindigt het programma.** Dit wordt normaal gesproken gebruikt in foutopsporing of om gebeurtenissen vast te leggen die niet zouden moeten plaatsvinden.
EXECUTE command$	**Hiermee wordt het basiscommando "command$" uitgevoerd.** Het gebruik moet worden beperkt tot basisopdrachten die opeenvolgend worden uitgevoerd. De GOTO-instructie zal bijvoorbeeld niet correct werken. Dingen die zijn getest en goed werken, zijn onder meer GOSUB, subroutineaanroepen en andere eenvoudige instructies (zoals PRINT en eenvoudige opdrachten) Meerdere uitspraken gescheiden door **:** zijn niet toegestaan en geven fouten. Het commando stelt een interne watchdog in voordat het gevraagde commando wordt uitgevoerd en als de controle niet terugkeert naar het commando, zoals in een GOTO instructie, loopt de timer af. In dit geval krijgt u de melding **"Command timeout"**. RUN is een speciaal geval en annuleert de timer, zodat u de opdracht kunt gebruiken om indien nodig programma's aan elkaar te koppelen
EXIT DO	**EXIT DO zorgt voor een vroege exit uit een DO...LOOP**.
EXIT FOR	**EXIT FOR zorgt voor een vroege exit uit een FOR...NEXT lus**.
EXIT FUNCTION	**EXIT FUNCTION zorgt voor een vroege exit uit een gedefinieerde functie.**
EXIT SUB	**EXIT SUB biedt een vroege exit uit een gedefinieerde subroutine**. De oude standaard van EXIT op zichzelf (Exit Do Loop) wordt ook ondersteund.
FILES [fspec$] [,sort]	**Geeft een overzicht van bestanden in elke map op de SD-kaart**. **'fspec$'** (indien gespecificeerd) kan een pad en zoekjokertekens bevatten in de bestandsnaam. Vraagtekens **(?)** komen overeen met elk teken en een asterisk **(*)** komt overeen met een willekeurig aantal tekens. Indien weggelaten, worden alle bestanden weergegeven.

COMMANDO	Beschrijving
	Bijvoorbeeld: `"*"` Vind alle inzendingen. `"*.TXT"` Vind alle vermeldingen met de extensie `TXT`. `"E*.*"` Vind alle vermeldingen die beginnen met `E`. `"X?X.*"` Vind alle drieletterige bestandsnamen die beginnen en eindigen met : `X?X.*` `"X mydir/*"` Vind alle vermeldingen in de **map** `mydir`. NB: het plaatsen van jokertekens in de padnaam resulteert in een fout. **'sort'** Specificeert de sorteervolgorde als volgt maat door oplopende maat tijd door aflopende tijd / datum naam op bestandsnaam (standaard indien niet gespecificeerd) typ op bestandsextensie.
`FLASH`	**Beheert de opslag van programma's in het flashgeheugen.** Er kunnen maximaal drie programma's worden opgeslagen in het flashgeheugen en worden opgeroepen als vereist. Houd er rekening mee dat deze opgeslagen programma's worden gewist met een firmware upgrade. Een van deze flashgeheugenlocaties kan automatisch worden geladen en uitgevoerd wanneer de stroom wordt ingeschakeld met behulp van de opdracht `OPTION AUTORUN n`. In de volgende **'n'** is een nummer `1` tot `3` (`1` tot `3` voor PicoMiteVGA)
`FLASH LIST n [, all]`	**Maak een lijst van het hele programma dat is opgeslagen in slot n**.
`FLASH DISK LOAD n, fname$ [,O[VERWRITE]]`	**Laadt de inhoud van bestand fname$ in flash-slot n als een binaire afbeelding.** Het bestand kan worden gemaakt met `LIBRARY DISK SAVE`. Bovendien kan elk bestand dat extern is gemaakt met de gegevens die een programma nodig heeft, worden geladen en geopend met behulp van opdrachten als `PEEK` en `MEMORY COPY` met behulp van het adres van het flashslot. Als de optionele parameter **OVERWRITE** (of **O**) is opgegeven, wordt de inhoud van het flashslot overschreven zonder dat er een fout optreedt.
`FLASH ERASE nr`	**Wis een flash-programmalocatie.**
`FLASH ERASE ALL`	**Wis alle locaties van het flashprogramma.**
`FLASH SAVE nr`	**Sla het huidige programma op de opgegeven flash-locatie op.**
`FLASH LOAD n [1-3]`	**Laadt het programma in het hoofdprogrammageheugen en heeft toegang tot elke bestaande bibliotheekcode.**
`FLASH RUN n`	**Voert het programma uit in flash-locatie n, wis alle variabelen**. Verandert niet het programmageheugen.
`FLASH CHAIN n`	**Voert het programma uit in flash-locatie n, waarbij alle variabelen intact blijven (staat een programma dat veel groter is dan het programmageheugen)**. Verandert niet het programmageheugen.

COMMANDO	Beschrijving 178
FLASH CHAIN [1-3]	**Door een in flash opgeslagen programma met FLASH CHAIN [1-3] aan elkaar te koppelen, wordt GEEN normale code uitgevoerd die in de bibliotheek is opgeslagen vóór het hoofdprogramma, maar het programma heeft wel toegang tot alle SUB 's, FUNCTION, FONT 's of CSUB 's in de bibliotheek.** De waarde die wordt gere-tourneerd door MM.Info(FLASH) identificeert vanaf welk slot het programma is geketend en kan in de bibliotheek worden gebruikt om bibliotheek- code/routines voorwaardelijk op te nemen of uit te sluiten.
FLASH DISK LOAD n, fname$ [,O[VERWRITE]]	**Laadt de inhoud van bestand fname$ in flash-slot n als een binaire afbeelding.** Het bestand kan worden gemaakt met LIBRARY **DISK** SAVE. Ook elk extern gemaakt bestand met gegevens die door een programma nodig zijn, kunnen worden geladen en geopend met behulp van opdrachten zoals PEEK en MEMORY COPY met behulp van het adres van het flashslot. Als de optionele parameter **OVERWRITE** (of **O**) is opgegeven, wordt de inhoud van het flash slot overschreven zonder dat er een fout optreedt.
FLASH OVERWRITE n	**Wis een flash-programmalocatie en sla het huidige programma vervolgens op in de flash locatie opgegeven.**
FLUSH [#]fnbr	**Veroorzaakt gebufferde schrijfacties naar een bestand dat eerder is geopend met het bestandsnummer '#fnbr' moet naar schijf worden geschreven.** De **#** is optioneel. Het gebruik van deze opdracht zorgt ervoor dat er geen gegevens verloren gaan als de stroom uitvalt na een schrijfopdracht.
FONT [#]font-number, scaling	**Dit stelt het standaardlettertype in voor het weergeven van tekst op een LCD-paneel**. Lettertypen worden opgegeven als een getal. Bijvoorbeeld **#2** (de **#** is optioneel). Zien het hoofdstuk "Basis tekenopdrachten" voor details over de beschikbare lettertypen. **'schalen'** kan variëren van **1** tot **15** en vermenigvuldigt de grootte van de pixels waardoor het weergegeven teken dienovereenkom-stig breder en hoger wordt. bijv. een schaal van 2 verdubbelt de hoogte en breedte.
FOR counter = start TO finish [STEP increment]	**Start een FOR - NEXT lus met de 'teller' aanvankelijk ingesteld op 'start' en ophogen in 'oplopende' stappen (standaard is 1) totdat 'teller' groter is dan 'finish'.** De **'toename'** kan een geheel getal of een getal met drijvende komma zijn. Let daar op het gebruik van een breukgetal met drijvende komma voor **'toename'** kan accumuleren afrondingsfouten in **'counter'** waardoor de lus voortijdig zou kunnen eindigen of later. **'increment'** kan negatief zijn, in welk geval **'finish'** kleiner moet zijn dan **'start'** en de lus telt naar beneden. Zie ook het NEXT commando.

COMMANDO	Beschrijving
NIET MET HDMI EN VGA	NIET MET HDMI EN VGA VERSIES
FRAMEBUFFER	**Met de opdracht Framebuffer kunt u een deel van het variabele geheugen toewijzen naar een framebuffer, een tweede weergavelaag of beide en gebruik deze vervolgens interessante manieren om te voorkomen dat artefacten worden gescheurd en/of grafische objecten worden afgespeeld over het achtergronddisplay.**
FRAMEBUFFER CREATE	**Creëert een framebuffer "F" met een RGB121-kleurruimte en bijpassende resolutie het geconfigureerde SPI-kleurendisplay.**
FRAMEBUFFER LAYER	**Creëert een framebuffer "L" met een RGB121-kleurruimte en resolutie overeenkomen met het geconfigureerde SPI-kleurendisplay.**
FRAMEBUFFER WRITE where/where$	**Specificeert het doel voor volgende grafische opdrachten.** **"waar"** kan **N, F** of **L** zijn, waarbij **N** de daadwerkelijke weergave is. **AA** string variabele kan worden gebruikt of letterlijk.
FRAMEBUFFER COPY from, to [,b]	**Maakt een sterk geoptimaliseerde kopie op volledig scherm van de ene framebuffer naar de andere.** **"van"** en **"naar"** kunnen **N, F** of **L** zijn, waarbij **N** het fysieke beeldscherm is. U kunt alleen kopiëren vanaf **N** op beeldschermen die **BLIT** en transparante tekst ondersteunen. De firmware zal de **RGB**-resolutie automatisch comprimeren of uitbreiden wanneer kopiëren van en naar ongeëvenaarde framebuffers Natuurlijk gaat bij het kopiëren van **RGB565** naar **RGB121** informatie verloren, maar voor velen toepassingen (bijvoorbeeld **games**) **16 kleurniveaus** is meer dan genoeg. Bij het kopiëren naar een **LCD-display** kan de optionele parameter **"b"** worden gebruikt (**FRAMEBUFFER COPY F/L, N, B**). Dit instrueert de firmware om actie te ondernemen kopiëren met behulp van de tweede **CPU in de Raspberry Pi Pico** en de controle keert terug onmiddellijk naar het Basic programma.
FRAMEBUFFER CLOSE [which]	**Sluit een framebuffer en geeft het geheugen vrij.** De optionele parameter **"welke"** kan **F** of **L** zijn. Indien weggelaten, worden beide gesloten.
FRAMEBUFFER WAIT	**Pauzeert de verwerking totdat het LCD-scherm overgaat in frame blanking. Geïmplementeerd voor ILI9341-, ST7789_320- en ILI9488-displays.** Wordt gebruikt om artefacten te verminderen wanneer schrijven naar het scherm.
FRAMEBUFFER MERGE [colour] [,mode] [,updaterate]	**Kopieert de inhoud van de laagbuffer en framebuffer naar het LCD-scherm het weglaten van alle pixels van een bepaalde kleur.** Randvoorwaarden voor de opdracht zijn dat **FRAMEBUFFER** en **LAYERBUFFER** beide worden gemaakt.

COMMANDO	Beschrijving 180
FRAMEBUFFER MERGE [colour] [,mode] [,updaterate]	**FRAMEBUFFER MERGE** Schrijft de inhoud van de **framebuffer** naar het fysieke weergave overschrijft alle pixels in de framebuffer die zijn ingesteld in de laag-buffer (niet nul). **FRAMEBUFER MERGE col** Schrijft de inhoud van de framebuffer naar het fysieke weergave overschrijft alle pixels in de framebuffer die zich in de laagbuffer is niet ingesteld op de transparante kleur **"col"**. De kleur wordt gespecificeerd als een getal tussen **0** en **15** dat staat voor: **0:ZWART, 1:BLAUW, 2:MIRTE, 3:KOBALT, 4:MIDDENGROEN, 5:CERULEAN, 6:GROEN, 7:CYAAN, 8:ROOD, 9:MAGENTA, 10:ROEST, 11:FUCHSIA, 12:BRUIN, 13:LILA, 14:GEEL, 15:WIT** **FRAMEBUFFER MERGE col, B** Zoals hierboven, behalve dat de overdracht naar de fysieke weer-gave vindt plaats op de **tweede CPU** en de besturing keert onmid-dellijk terug naar Basic. **FRAMEBUFFER MERGE col, R [,updaterate]** Stelt de **tweede CPU** in update het fysieke display voortdurend met de samenvoeging van de twee buffers. Stelt **FRAMEBUFFER WRITE F** automatisch in als **F** of **L** nog niet is ingesteld. Door standaard wordt het scherm zo snel mogelijk bijge-werkt. Op **133 MHz** wordt een **ILI9341** in de **SPI**-modus wordt onge-veer 13 keer per seconde bijgewerkt, in 8-bit parallelle modus de **ILI9341** behaalt **27 FPS**. Als "updaterate" is ingesteld, wordt het scherm bijgewerkt naar de snelheid die is opgegeven in milliseconden (tenzij dat minder is dan het snelst haalbare op het display). **NB: FRAMEBUFFER WRITE** kan niet worden ingesteld op **N** tijdens continu samenvoegen bijwerken is actief. **FRAMEBUFFER MERGE col, A** Breekt de voortdurende updates af. Bovendien zal het verwijderen van de **layerbuf** of **framebuffer**, **ctrl-C** of **END** dat wel doen en ook de automatische update afbreken.
FRAMEBUFFER SYNC	**Wacht tot de laatste update op de tweede CPU is voltooid voordat tekenen mogelijk is zonder te verminken.**

COMMANDO	Beschrijving 181
ALLEEN HDMI EN VGA	ALLEEN HDMI EN VGA VERSIES
FRAMEBUFFER	**Met de opdracht Framebuffer kunt u een deel van het variabele geheugen toewijzen naar framebuffers, laagbuffers of beide en gebruik deze vervolgens op interessante manieren beide vermijden het verminken van artefacten en/of spelen grafische objecten over de achtergrond weergave.**
FRAMEBUFFER CREATE	**Creëert een framebuffer “F” met een kleurruimte en resolutie die overeenkomen met de huidige weergavemodus.**
FRAMEBUFFER CREATE 2	**Alleen RP2350: Creëert een tweede framebuffer “2” met een kleurruimte en resolutie die overeenkomt met de huidige weergavemodus.**
FRAMEBUFFER LAYER [colour]	**Creëert een laagbuffer “L” met een kleurruimte en resolutie die overeenkomen met de huidige weergavemodus.** De optionele parameter kleur wordt opgegeven als een **getal 0 - 15 (modi 2 en 3)**, **RGB888 kleur (modus 4)** of **0-255 (modus 5)** en specificeert een kleur die wordt genegeerd wanneer de laag op het scherm wordt toegepast. In weergavemodi waarin automatische laagtoepassing geen laagbuffer ondersteunt fungeert als een andere framebuffer.
FRAMEBUFFER LAYER TOP [colour]	**Alleen RP2350: Creëert een tweede laagbuffer “T” met een kleurruimte en resolutie die overeenkomt met de huidige weergavemodus.** De optionele parameterkleur is gespecificeerd als een **getal 0-15 (modi 2 en 3), 0-255 (modus 5)** en specificeert een kleur die wordt genegeerd wanneer de laag op het scherm wordt toegepast. In weergave modi waarbij automatische toepassing van de tweede laag niet wordt ondersteund, fungeren als een andere framebuffer.
FRAMEBUFFER WRITE where/where$	**Specificeert het doel voor volgende grafische opdrachten.** **"waar"** kan **N, F, 2, T** of **L** zijn, waarbij **N** de daadwerkelijke weergave is. Een string variabele kan worden gebruikt.
FRAMEBUFFER CLOSE [which]	**Sluit een framebuffer en geeft het geheugen vrij.** De optionele parameter **"welke"** kan **F, 2, T of L** zijn. Als dit wordt weggelaten, wordt alles gesloten.
FRAMEBUFFER COPY from, to [,b]	**Maakt een sterk geoptimaliseerde kopie op volledig scherm van de ene framebuffer naar de andere.** **"van"** en **"naar"** kunnen **N, F, 2, T** of **L** zijn, waarbij **N** het fysieke beeldscherm is. Als de optionele parameter **‘b’** is opgegeven, wordt de verwerking gepauzeerd totdat de Monitor voert frame-blanking in.
FRAMEBUFFER WAIT	**Pauzeert de verwerking totdat de volgende frameonderdrukking begint.**

COMMANDO	Beschrijving 182
`FUNCTION xxx (arg1[, arg2,..])[AS <type>}` `  <statements>` `  <statements>` `xxx = <return value>` `END FUNCTION`	**Definieert een oproepbare functie**. Dit is hetzelfde als het toevoegen van een nieuwe functie aan **MMBasic** terwijl uw programma wordt uitgevoerd. **'xxx'** is de functienaam en deze moet voldoen aan de specificaties voor het benoemen van een variabele. Het type van de functie kan worden gespecificeerd met behulp van een typeachtervoegsel (d.w.z. **xxx$**) of door het type op te geven met **AS <type>** aan het einde van de functie definitie. Bijvoorbeeld: `FUNCTION xxx (arg1, arg2) AS STRING` **'arg1'**, **'arg2'**, etc zijn de argumenten of parameters voor de functie (de haakjes zijn altijd vereist, ook als er geen argumenten zijn). Een array is aangegeven door lege haakjes te gebruiken. d.w.z. **arg3()**. Het type argument kan worden gespecificeerd door een typeachtervoegsel te gebruiken (d.w.z. **arg1$**) of door het type op te geven met `AS <type> (d.w.z. arg1 AS STRING)`. Het argument kan ook een andere gedefinieerde functie zijn of dezelfde functie als recursie moet worden gebruikt (de recursiestapel is beperkt tot **50 geneste oproepen**). Om de retourwaarde van de functie in te stellen, wijst u de waarde toe aan de functie's naam. Bijvoorbeeld: `FUNCTION SQR(a)` `  SCUARE = a * a` `END FUNCTION` Elke definitie moet één `END FUNCTION` statement hebben. Wanneer dit is de functie heeft bereikt, retourneert de waarde naar de uitdrukking waaruit deze was genaamd. Het commando: `EXIT FUNCTION` Kan gebruikt worden voor een vroege exit. U gebruikt de functie door de naam en argumenten in een programma te gebruiken, net zoals je zou een normale MMBasic-functie hebben. Bijvoorbeeld: `PRINT SQR(56.8)` Wanneer de functie wordt aangeroepen, komt elk argument in de aanroeper overeen met het argument in de functiedefinitie. Deze argumenten zijn alleen beschikbaar binnen de functie. Functies kunnen worden aangeroepen met een variabel aantal argumenten. Enkele weggelaten argumenten in de lijst van de functie worden ingesteld op nul of een null-tekenreeks. Argumenten in de lijst van de beller die een variabele zijn en van het juiste type zijn wordt doorgegeven met verwijzing naar de functie. Dit betekent dat eventuele wijzigingen in het corresponderende argument in de functie wordt ook gekopieerd naar de aangeroepen variabele en is daarom toegankelijk nadat de functie is beëindigd. Arrays worden doorgegeven door de arraynaam op te geven met lege haakjes (bijv. **arg()**) en worden altijd als referentie doorgegeven en moeten van het juiste type zijn. U mag niet in of uit een functie springen met commando's zoals `GOTO`, `GOSUB`, enz. Dit zal ongedefinieerde bijwerkingen hebben, waaronder de mogelijkheid om je dag te verpesten.

COMMANDO	Beschrijving
GAMEPAD COLOUR channel, colour	**Verandert de kleur van het weergavepaneel op een PS4-controller op USB-kanaal 'kanaal'.** **'kleur'** is ingesteld als een standaard **RGB888** waarde, b.v. **RGB(RED)**
GAMEPAD HAPTIC channel left, right	**Zorgt ervoor dat de linker en rechter vibratiemotoren werken op een PS4-controller op USB-kanaal 'kanaal'.** **'left'** en **'right'** moeten een getal zijn tussen **0(uit)** en **255(maximaal)**
GAMEPAD INTERRUPT ENABLE channel, int [,mask]	**Maakt onderbrekingen mogelijk bij het indrukken van een knop op een USB-gamecontroller.** De optionele parameter **'mask'** definieert welke van de schakelaars de **interrupt** zal activeren (standaard is alles). **'mask'** is een bitmap die overeenkomt met de uitvoer van de functie **DEVICE (GAMEPAD channel, B).**
GAMEPAD INTERRUPT DISABLE channel	**Schakelt interrupts van de gamepad uit op het opgegeven kanaal.**
GOTO target	**Vertakt de uitvoering van het programma naar het doel, wat een regelnummer of een label kan zijn.**
GUI AREA #ref, startX, startY, width, height	**Dit definieert een onzichtbaar gebied van het scherm dat gevoelig is voor aanraking en zal touch-down en touch-up interrupts genereren.** Het kan worden gebruikt als een basis voor het creëren van aangepaste controles die worden gedefinieerd en beheerd door het programma. **'#ref'** is het referentie nummer van de besturing. **'startX'** en **'startY'** zijn de linkerbovenhoek coördinaten terwijl **'breedte'** en **'hoogte'** de afmetingen instellen. **Niet op de PicoMiteVGA.**
GUI BCOLOUR colour, #ref1 [, #ref2, #ref3, enz.]	**Hierdoor wordt de achtergrondkleur van de opgegeven bedieningselementengewijzigd in 'kleur' dat is een RGB-waarde voor de tekenkleur**. **'#ref'** is het referentienummer van de besturing. **Niet op de PicoMiteVGA**.
GUI BARGAUGE #ref, StartX, StartY, width, height, FColour, BColour, min, max, c1, ta, c2, tb, c3, tc, c4	**Definieer een horizontale of verticale analoge staafgrafiek**. **'#ref' is het referentienummer van de besturing.** **'StartX'** en **'StartY'** zijn de coördinaten linksboven van de balk, terwijl **'breedte'** de horizontale breedte en **'hoogte'** de verticale hoogte. Als de breedte kleiner is dan de hoogte zal de staafmeter verticaal worden getekend terwijl de grafiek groeit vanaf de onderkant naar de bovenkant. Anders wordt het horizontaal getekend met de grafiek groeit van links naar rechts. **'Fcolour'** is de kleur die wordt gebruikt voor de meter, terwijl **'Bcolour'** de achtergrond is kleur. **'min'** is de minimumwaarde van de meter en **'max'** is het maximum waarde (beide drijvende komma).

COMMANDO	Beschrijving
	Er kan een meerkleurenmeter worden gemaakt met **'c1'** tot **'c4'** voor de kleuren en **'ta'** naar **'tc'** voor de drempels die worden gebruikt om te bepalen wanneer de kleur zal veranderen. **'width'**, **'height'**, **'FColour'**, **'BColour'**, **'min'** en **'max'** zijn optioneel en zullen standaard de waarden gebruikt in de vorige definitie van een **GUI BARGAUGE**. **'c1', 'ta', 'c2', 'tb', 'c3', 'tc' en 'c4'** zijn optioneel en indien niet gespecificeerd de meter zal minder kleuren gebruiken. Als ze allemaal worden weggelaten, wordt de meter getekend met **'Fkleur'**. De sectie Geavanceerde grafische afbeeldingen heeft een meergedetailleerde beschrijving. **Niet op de PicoMiteVGA.**
`GUI BITMAP x, y, bits [, width] [, height] [, scale] [, c] [, bc]`	**Geeft de bits in een bitmap weer op een VGA/HDMI-monitor of LCD-paneel, beginnend bij 'x' en 'y' op een aangesloten apparaat.** **'hoogte'** en **'breedte'** zijn de afmetingen van de bitmap zoals weergegeven op het apparaat en zijn standaard 8 x 8. **'scale'** is optioneel en is standaard ingesteld op de opdracht **FONT**. **'c'** is de tekenkleur en **'bc'** is de achtergrondkleur. Ze zijn optioneel en hebben standaard de huidige voor- en achtergrondkleuren. De bitmap kan een geheel getal of een stringvariabele of constante zijn en wordt getekend met de eerste byte als de eerste bits van de bovenste regel (eerst bit 7, dan bit 6, enz.), gevolgd door de volgende byte, enz. Wanneer de bovenste regel is gevuld, begint de volgende regel van de weergegeven bitmap met de volgende bit in het gehele getal of de tekenreeks. Zie het hoofdstuk Grafische opdrachten en functies voor een definitie van de kleuren en grafische coördinaten.
`GUI BUTTON #ref, caption$, startX, startY, width, height [, FColour] [, BColour]`	**Dit zal een tijdelijke knop tekenen die een vierkante schakelaar is met het bijschrift op zijn gezicht.** Bij aanraking lijkt het visuele beeld van de knop ingedrukt en de waarde van het besturingselement is 1. Wanneer de aanraking wordt verwijderd, zal de waarde terug naar nul. **#ref'** is het referentienummer van de controle. **'caption$'** is de tekenreeks die op de voorkant van de knop moet worden weergegeven. Het kan een single zijn string met twee bijschriften gescheiden door een | teken (bijv. **"UP | DOWN"**). Als de knop omhoog is, wordt de eerste string gebruikt en wanneer op de wordt gedrukt tweede zal worden gebruikt. **'startX'** en **'startY'** zijn de coördinaten linksboven terwijl **'width'** en **'height'** zijn ingesteld de dimensies. **'FColour'** en **'BColor'** zijn **RGB** waarden voor de voorgrond en achtergrondkleuren. **'width'**, **'height'**, **FColour** en **'BColour'** zijn optioneel en worden standaard gebruikt in vorige bedieningselementen of ingesteld met het **COLOR** commando. **Niet op de PicoMiteVGA**.

COMMANDO	Beschrijving 185
GUI CALIBRATE of GUI CALIBRATE a, b, c, d, e	**NIET VGA- EN HDMI-VERSIES** **Deze opdracht wordt gebruikt om de aanraakfunctie op een LCD-paneel te kalibreren.** Het zal een reeks doelen op het scherm weergeven en wachten tot ze allemaal precies zijn aangeraakt. Het commando kan ook worden gebruikt met vijf argumenten die de kalibratiewaarden specificeren en in dit geval zal de kalibratie worden uitgevoerd zonder dat er doelen worden weergegeven of dat er input van de gebruiker nodig is. Om de waarden te ontdekken, gebruikt u de OPTION LIST nadat u het display normaal heeft gekalibreerd. Houd er rekening mee dat deze waarden specifiek zijn voor dat scherm en aanzienlijk kunnen variëren.
GUI CAPTION #ref, text$, startX, startY [,align$] [, FColour] [, BColour]	**Hiermee wordt een tekstreeks op het scherm getekend.** **'#ref'** is het referentienummer van de besturing. **'text$'** is de tekenreeks die moet worden weergegeven. **'startX'** en **'startY'** zijn de linkerbovenhoek coördinaten. **'align$'** is nul tot drie tekens (een tekenreeksuitdrukking of variabele is ook toegestaan) waarbij de eerste letter de horizontale uitlijning rond **X** is en kan wees **L, C** of **R** voor **LEFT, CENTER, RIGHT** en de tweede letter is de verticale uitlijning rond **Y** en kan **T, M** of **B** zijn voor **BOVEN, MIDDEN, ONDERKANT**. Een derde teken kan in de tekenreeks worden gebruikt om aan te geven rotatie van de tekst. Dit kan **'N'** zijn voor normale oriëntatie, **'V'** voor verticaal tekst met elk teken onder het vorige van boven naar beneden, **'I'** de tekst wordt omgekeerd (d.w.z. ondersteboven), **'U'** de tekst wordt gedraaid tegen de klok in met **90º** en **'D'** wordt de tekst **90º** met de klok mee gedraaid. De standaarduitlijning is links/boven zonder rotatie. **'FColour** en **'BColor'**zijn RGB waarden voor de voorgrond en achtergrond kleuren. Op een display dat transparante tekst ondersteunt, kan **Bkleur -1** zijn, wat betekent dat de achtergrond door de gaten in de karakters heen zichtbaar is. **FColour** en **'BColor'** zijn optioneel en standaard ingesteld op de kleuren die zijn ingesteld door het COLOR commando. **Niet op de PicoMiteVGA.**
GUI CHECKBOX #ref, caption$, startX, startY [, size] [, colour]	**Hiermee wordt een selectievakje getekend, een klein vakje met een bijschrift**. Wanneer aanraakt, wordt er een **X** in het vak getekend om aan te geven dat deze optie heeft is geselecteerd en de waarde van de regelaar wordt ingesteld op **1**. Wanneer een punt wordt aangeraakt voor de tweede keer wordt het vinkje verwijderd en wordt de waarde van het besturingselement nul. **'#ref'** is het referentienummer van de besturing. De tekenreeks **'caption$'** wordt rechts van het besturingselement getekend met de kleuren ingesteld door het COLOR commando. **'startX'** en **'startY'** zijn de coördinaten linksboven terwijl **'size'** de hoogte instelt en breedte (de bix is vierkant). **'kleur'** is een **RGB** waarde voor de tekeningkleur.

	'grootte' en **'kleur'** zijn optioneel en standaard naar die gebruikt in eerdere controles. **Niet op de PicoMiteVGA**.
GUI DELETE #ref1 [,#ref2, #ref3, enz] of **GUI DELETE ALL**	**Hiermee worden de besturingselementen in de lijst verwijderd**. Dit omvat het verwijderen van de afbeelding van de besturing vanaf het scherm met de huidige achtergrondkleur en bevrijding het geheugen dat door de besturing wordt gebruikt. **'#ref'** is het referentienummer van de besturing. Het sleutelwoord **ALL** kan worden gebruikt als het argument en dat zal alle besturingselementen verwijderen. **Niet op de PicoMiteVGA**.
GUI DISABLE #ref1 [,#ref2, #ref3, enz] of **GUI DISABLE ALL**	**Hierdoor worden de bedieningselementen in de lijst uitgeschakeld** Uitgeschakelde bedieningselementen reageren niet op aanraken en wordt gedimd weergegeven. **'#ref'** is het referentienummer van de besturing. Het sleutelwoord **ALL** kan worden gebruikt als het argument en dat zal alle bedieningselementen uitschakelen. **GUI ENABLE** kan worden gebruikt om de bedieningselementen te herstellen. **Niet op de PicoMiteVGA**.
GUI DISPLAYBOX #ref, startX, startY, width, height, FColour, BColour	**Hiermee wordt een vak met afgeronde hoeken getekend dat kan worden gebruikt om een string**. **'#ref'** is het referentienummer van de besturing. **'startX'** en **'startY'** zijn de coördinaten linksboven terwijl **'width'** en **'height'** zijn ingesteld de dimensies. **' FColour'** en **'BColor'** zijn **RGB** waarden voor de voorgrond en achtergrondkleuren. **'breedte'**, **'hoogte', FColour** en **'BColour'** zijn optioneel en standaard naar die gebruikt in eerdere controles. Elke tekst kan in het vak worden weergegeven met behulp van de opdracht **CtrlVal(r) =**. Dit is handig voor het weergeven van tekst, cijfers en berichten. Deze bediening reageert niet op aanraking. **Niet op de PicoMiteVGA**.
GUI ENABLE #ref1 [,#ref2, #ref3, enz] of **GUI ENABLE ALL**	**Dit zal de effecten van GUI DISABLE ongedaan maken en de besturing(en) herstellen naar normale operatie**. **'#ref'** is het referentienummer van de besturing. Het sleutelwoord **ALL** kan worden gebruikt als het argument en dat zal alle bedieningselementen uitschakelen. **Niet op de PicoMiteVGA**.
GUI FCOLOUR colour, #ref1 [, #ref2, #ref3, enz.]	**Hierdoor wordt de voorgrondkleur van de opgegeven bedieningselementen gewijzigd in 'kleur' dat is een RGB waarde voor de tekenkleur**. **'#ref'** is het referentienummer van de besturing. **Niet op de PicoMiteVGA**.

COMMANDO	Beschrijving
`GUI FRAME #ref, caption$, startX, startY, width, height, colour`	**Hiermee wordt een kader getekend dat een doos is met ronde hoeken en een bijschrift.** **'#ref'** is het referentienummer van de besturing. **'caption$'** is een tekenreeks die als bijschrift moet worden weergegeven. **'startX'** en **'startY'** zijn de top linker coördinaten terwijl **'breedte'** en **'hoogte'** de afmetingen instellen. **'kleur'** is een RGB waarde voor de kleur van de tekening. **'breedte'**, **'hoogte'** en **'kleur'** zijn optioneel en standaard naar die gebruikt in eerdere controles. Een frame is handig wanneer een groep bedieningselementen visueel moet worden gebracht samen. Het wordt ook gebruikt om een groep keuzerondjes en MMBasic zorgt ervoor dat de keuzerondjes die door het frame worden omgeven exclusief zijn, d.w.z. wanneer een keuzerondje is geselecteerd, een andere knop die is geselecteerd en binnen het frame wordt automatisch gedeselecteerd. Een frame reageert niet op aanraking. **Niet op de PicoMiteVGA**.
`GUI FORMATBOX #ref, Format, startX, startY, width, height, FColour, BColour`	**Dit tekent een doos met afgeronde hoeken die kan worden gebruikt om een virtueel toetsenbord voor het invoeren van gegevens in een specifiek formaat.** **'#ref'** is het referentienummer van de besturing. **'startX'** en **'startY'** zijn de coördinaten linksboven terwijl **'width'** en **'height'** zijn ingesteld de dimensies. **'FColour'** en **'BColor'** zijn RGB waarden voor de voorgrond en achtergrondkleuren. **'breedte', 'hoogte', FColour** en '**BColour'** zijn optioneel en standaard naar die gebruikt in eerdere controles. Het argument **'Formaat'** specificeert het formaat van het item als volgt: **DATE1** Datum in UK/Aust/NZ formaat (dd/mm/jj) **DATE2** Datum in VS formaat (mm/dd/jj) **DATE3** Datum in internationaal formaat (jjjj/mm/dd) **TIME1** Tijd in 24 uur notatie (uu:mm) **TIME2** Tijd in 24 uur notatie met seconden (uu:mm:ss) **TIME3** Tijd in 12 uur notatie (uu:mm AM/PM) **TIME4** Tijd in 12 uur notatie met seconden uu:mm:ss AM/PM **DATETIME1** Datum UK fmt en tijd 12 uur dd/mm/jj uu:mm AM/PM **DATETIME2** Datum UK fmt en tijd 24 uur dd/mm/jj uu:mm **DATETIME3** Datum VS fmt en tijd 12 uur mm/dd/jj uu:mm AM PM **DATETIME4** Datum VS fmt en tijd 24 uur mm/dd/jj uu:mm **LAT1** Breedtegraad in graden, minuten en seconden (dd° mm' ss" N/S) **LAT2** Breedtegraad met seconden tot één decimaal (dd° mm' ss.s" N/S) **LONG1** Lengtegraad in graden, minuten en seconden (ddd° mm' ss" E/W) **LONG2** Lengtegraad seconden tot één decimaal (ddd° mm' ss.s" E/W)

COMMANDO	Beschrijving 188
	ANGLE1 Hoek in graden en minuten (ddd° mm') Bijvoorbeeld dit commando: `GUI FORMATBOX #1, LAT1, 50, 50, 300, 50` Zou een formaatvak creëren dat de invoer van breedtegraad in het formaat zou accepteren van dd° mm' ss" N/S. De waarde van **CtrlVal(#1)** zou een string zijn PicoMite die de cijfers en scheidingstekens bevat. Bijvoorbeeld een vermelding van **17 graden**, **32 minuten** en **1 seconde zuid** zou resulteren in de string **17° 32' 01" S**. MMBasic zal proberen het virtuele toetsenbord op het scherm te plaatsen om niet verdoezel het formaatvak dat ervoor zorgde dat het verscheen. Een onderbreking van de pen naar beneden zal worden gegenereerd net voordat het toetsenbord wordt ingezet en een onderbreking van de toetsaanslag zal worden gegenereerd wanneer de invoer is voltooid en het toetsenbord is verborgen. **Niet op de PicoMiteVGA.**
`GUI FORMATBOX ACTIVATE #ref`	**Hierdoor wordt het virtuele toetsenbord voor de besturing '#ref' weergegeven onder programmabesturing zonder dat de besturing wordt aangeraakt.** Het is hetzelfde alsof de gebruiker de besturing heeft aangeraakt, behalve dat de touch-down onderbreking niet is gegenereerd. **Niet op de PicoMiteVGA**.
`GUI FORMATBOX CANCEL`	**Dit zal een virtueel toetsenbord verwijderen als het op het scherm wordt weergegeven.** Het is de hetzelfde als wanneer de gebruiker de annuleertoets aanraakt, behalve dat de touch-up interrupt wordt niet gegenereerd. Als er geen toetsenbord wordt weergegeven, zal deze opdracht niets doen. **Niet op de PicoMiteVGA**.
`GUI GAUGE #ref, StartX, StartY, Radius, FColour, BColour, min, max, nbrdec, units$, c1, ta, c2, tb, c3, tc, c4`	**Definieer een grafische ronde analoge meter met een digitaal display in de centrum**. **'#ref'** is het referentienummer van de besturing. **'StartX'** en **'StartY'** zijn de coördinaten van het midden van de meter, **'Radius'** is de afstand van het midden tot de buitenrand. **'min'** is de minimale waarde van de meter en **'max'** is de maximale waarde (beide zwevende komma). **'nbrdec'** specificeert het aantal decimalen dat moet worden gebruikt bij het tekenen van de digitale waarde in het midden van de meter. Daaronder wordt **'units$'** weergegeven. **'Fcolour'** is de kleur die wordt gebruikt voor de meter, terwijl **'Bcolour'** de achtergrond is kleur. Er kan een meerkleurenmeter worden gemaakt met **'c1'** tot **'c4'** voor de kleuren en **'ta'** tot **'tc'** voor de drempels die worden gebruikt om te bepalen wanneer de kleur zal verandering. Wanneer kleuren en drempels zijn opgegeven, wordt de achtergrond van de meter zal worden getekend met een saaie versie van de kleur op dat niveau. Ook de digitale waarde zal veranderen in de kleur gespecificeerd door de huidige waarde.

	'Radius', 'FColour', 'BColour', 'min', 'max', 'nbrdec' en 'units$' zijn optioneel en zal standaard de waarden gebruiken die in de vorige definitie van een **GUI** zijn gebruikt **GRAADMETER**. **'c1', 'ta', 'c2', 'tb', 'c3', 'tc'** en **'c4'** zijn optioneel en indien niet gespecificeerd de meter zal minder kleuren gebruiken. Als ze allemaal worden weggelaten, wordt de meter getekend met **'Fkleur'** De sectie Geavanceerde grafische afbeeldingen heeft een meergedetailleerde beschrijving. **Niet op de PicoMiteVGA**.
`GUI HIDE #ref1 [,#ref2, #ref3, enz]` of `GUI HIDE ALL`	**Hierdoor worden de bedieningselementen in de lijst verborgen**. Verborgen bedieningselementen reageren niet opaanraken en zal niet zichtbaar zijn. **'#ref'** is het referentienummer van de besturing. Het sleutelwoord **ALL** kan worden gebruikt als het argument en dat zal alle besturingselementen verbergen. **GUI SHOW** kan worden gebruikt om de besturing te herstellen. **Niet op de PicoMiteVGA**.
`GUI INTERRUPT down [, up]`	**Met deze opdracht wordt een interrupt ingesteld die wordt geactiveerd bij een druk op de LCD-paneel en optioneel als de aanraking wordt losgelaten**. **'down'** is de subroutine die moet worden opgeroepen wanneer een touchdown is gedetecteerd. 'omhoog' is de subroutine die moet worden opgeroepen wanneer de aanraking van het scherm is gehaald (**'omhoog'** en **'down'** kan indien nodig naar dezelfde subroutine verwijzen). Door het getal nul (één cijfer) als argument op te geven, worden beide deze onderbrekingen. d.w.z: `GUI INTERRUPT 0` **Niet op de PicoMiteVGA**.
`GUI LED #ref, caption$, centerX, centerY, radius, colour`	**Dit zal een indicatielampje tekenen dat eruitziet als een op een paneel gemonteerde LED**. **Een LED** reageert **niet** op **aanraking**. **'#ref'** is het referentienummer van de besturing. De tekenreeks **'caption$'** wordt rechts van het besturingselement getekend met de kleuren ingesteld door het **COLOR**-commando. **'centerX'** en **'centerY'** zijn de coördinaten van het midden van de **LED** en **'radius'** is de straal van de **LED**. **'kleur'** is een **RGB**-waarde voor de tekening kleur. **'straal'** en **'kleur'** zijn optioneel en standaard gebruikt in vorige controles. Wanneer de waarde van een **LED** is ingesteld op een waarde van één, gaat deze branden en wanneer het is ingesteld op **nul**, het zal uit zijn (een saaie versie van het kleurattribuut). De LED kan worden gemaakt om aan en uit te knipperen door de waarde van de **LED** in te stellen op een getal groter is dan één, wat de tijd in milliseconden is dat het aan moet blijven. De kleur kan worden gewijzigd met het **GUI FCOLOUR** commando. **Niet op de PicoMiteVGA**.

COMMANDO	Beschrijving 190
`GUI NUMBERBOX #ref, startX, startY, width, height, FColour, BColour`	**Dit tekent een doos met afgeronde hoeken die kan worden gebruikt om een virtueel numeriek toetsenbord voor gegevensinvoer.** **'#ref'** is het referentienummer van de besturing**.** **'startX'** en **'startY'** zijn de coördinaten linksboven terwijl **'width'** en **'height'** zijn ingesteld de dimensies. **'FColour'** en **'BColor'** zijn **RGB** waarden voor de voorgrond en achtergrondkleuren. **'breedte'**, **'hoogte', 'FColour'** en **'BColour'** zijn optioneel en standaard naar die gebruikt in eerdere controles. Wanneer het vakje wordt aangeraakt, verschijnt er een numeriek toetsenbord op het scherm. Gebruik makend van dit virtuel toetsenbord kan elk nummer in het vak worden ingevoerd, inclusief een getal met drijvende komma in exponentieel formaat. Het nieuwe nummer zal vervangen het nummer dat eerder in het vak stond. De waarde van het besturingselement kan worden ingesteld op een letterlijke tekenreeks (geen uitdrukking) vanaf met twee hashtekens. Bijvoorbeeld: `CtrlVal(nnn) = "##Voer nummer in"` En in dat geval zal de string (zonder de eerste twee hash-tekens) zijn weergegeven in de doos met verminderde helderheid. Dit kan worden gebruikt om de gebruiker een hint geven over wat er moet worden ingevoerd (genaamd **"spooktekst"**). Het lezen van waarde van het besturingselement dat spooktekst weergeeft, zal nul opleveren. Wanneer de controle normaal wordt gebruikt, zal de spooktekst verdwijnen. MMBasic zal proberen het virtuele toetsenbord op het scherm te plaatsen om niet verdoezel het nummervak dat ervoor zorgde dat het verscheen. Een onderbreking van de pen naar beneden zal wor-den gegenereerd net voordat het toetsenbord wordt ingezet en een onderbreking van de toetsaanslag zal worden gegenereerd wanneer de Enter-toets wordt aangeraakt en het toetsenbord is verborgen wanneer de Enter-toets wordt aangeraakt, wordt het ingevoerde nummer geëvalueerd als een nummer en de **NUMBERBOX** besturing opnieuw getekend om dit nummer weer te geven. **Niet op de PicoMiteVGA.**
`GUI NUMBERBOX ACTIVATE #ref`	**Hierdoor wordt het virtuele toetsenbord voor de besturing '#ref' weergegeven onder programmabesturing zonder dat de besturing wordt aangeraakt.** Het is hetzelfde alsof de gebruiker de besturing heeft aangeraakt, behalve dat de touch-down-onderbreking niet is gegenereerd. **Niet op de PicoMiteVGA**.
`GUI NUMBERBOX CANCEL`	**Dit zal een virtueel toetsenbord verwijderen als het op het scherm wordt weergegeven.** Het is de hetzelfde als wanneer de gebruiker de annuleertoets aanraakt, behalve dat de touch-up interrupt wordt niet gegenereerd. Als er geen toetsenbord wordt weergegeven, zal deze opdracht niets doen. **Niet op de PicoMiteVGA**.

COMMANDO	Beschrijving 191
`GUI PAGE #n [, #n2, #n3, etc]`	**Dit zal de display omschakelen naar de regelaars die zijn toegewezen (via de GUI SETUP opdracht) naar de paginanummers gespecificeerd op de opdracht regel (#n, #n2, enz.).** Alle bedieningselementen die werden weergegeven maar niet op de huidige lijst met pagina's wordt automatisch verborgen. Alle bedieningselemen-ten op een pagina die op het oude scherm werd weergegeven en ook wordt gespecificeerd in de **nieuwe GUI PAGE commando** blijft onaangetast. De standaardinstelling wanneer een programma begint te lopen is **GUI PAGE 1** en **GUI SETUP 1**. Dit betekent dat als deze commando's niet worden gebruikt, het programma zal werken als **normaal met alle GUI besturingselementen die zijn gedefinieerd**. Zie ook de **GUI SETUP** opdracht.
`GUI RADIO #ref, caption$, centerX, centerY, radius, colour`	**Hiermee wordt een keuzerondje met een bijschrift getekend**. **'#ref'** is het referentienummer van de besturing. De tekenreeks **'caption$'** wordt rechts van het besturingselement getekend met de kleuren ingesteld door het **COLOR** commando. **'centerX'** en **'centerY'** zijn de coördinaten van het midden van de knop en **'radius'** is de straal van de knop. **'kleur'** is een **RGB** waarde voor de tekening kleur. **'straal'** en **'kleur'** zijn optioneel en standaard gebruikt in vorige controles. Bij aanraking wordt het midden van de knop verlicht om aan te geven dat deze optie is geselecteerd en de waarde van het besturingselement is **1**. Wanneer een ander keuzerondje is geselecteerd, de markering op deze knop wordt verwijderd en de waarde ervan nul zal zijn. Keuzerondjes zijn gegroepeerd wanneer ze worden omringd door een frame en wanneer één knop in de groep is geselecteerd, alle andere in de groep zal worden gedeselecteerd. Als er geen frame wordt gebruikt, worden alle knoppen op het scherm samengevoegd. **Niet op de PicoMiteVGA**.
`GUI REDRAW #ref1 [,#ref2, #ref3, enz]` of `GUI REDRAW ALL`	**Hiermee worden de bedieningselementen op het scherm opnieuw getekend**. Het is handig als het schermbeeld is op de een of andere manier beschadigd. **'#ref'** is het referentienummer van de besturing. Het sleutelwoord **ALL** kan worden gebruikt als het argument en dat zal eerst het scherm wissen en vervolgens alle bedieningselementen opnieuw tekenen. Dit is handig als het hele scherm ververst moet worden. **Niet op de PicoMiteVGA**.

COMMANDO	Beschrijving 192
`GUI RESET LCDPANEL`	**NIET VGA- EN HDMI-VERSIES** **Zal het geconfigureerde LCD-paneel opnieuw initialiseren.** Initialisatie wordt automatisch uitgevoerd wanneer de PicoMite-firmware opstart, maar in sommige omstandigheden kan het nodig zijn om de stroom naar het LCD-paneel te onderbreken (bijvoorbeeld om de batterij te sparen) en deze opdracht kan vervolgens worden gebruikt om het display opnieuw te initialiseren.
`GUI TEST LCDPANEL` of `GUI TEST TOUCH`	**NIET VGA- EN HDMI-VERSIES** **Test de display- of aanraakfunctie op een LCD-paneel.** Met **GUI TEST LCDPANEL** wordt snel een geanimeerde weergave van kleurencirkels over elkaar getekend. **Met GUI TEST TOUCH zal het scherm leeg zijn en wachten op een aanraking, waardoor een witte stip op het scherm wordt geplaatst die de aanraakpositie op het scherm markeert.** Elk personage dat op de console wordt ingevoerd, beëindigt de test
`GUI SETUP #n`	**Hiermee worden alle nieuwe besturingselementen die zijn gemaakt, toegewezen aan de pagina '#n'.** Deze opdracht kan zo vaak als nodig worden gebruikt terwijl **GUI besturingselementen** zijn gedefinieerd worden. De standaardinstelling wanneer een programma wordt gestart, is **GUI SETUP 1.** Zie ook het **GUI PAGE commando**. **Niet op de PicoMiteVGA**.
`GUI SHOW #ref1 [,#ref2, #ref3, enz]` of `GUI SHOW ALL`	**Dit maakt de effecten van GUI HIDE ongedaan en herstelt de controle(s) naar zijn zichtbaar en in staat om normaal te werken.** **'#ref'** is het referentienummer van de besturing. Het sleutelwoord **ALL** kan worden gebruikt als het argument en dat zal alle bedieningselementen uitschakelen. **Niet op de PicoMiteVGA**.
`GUI SPINBOX #ref, startX, startY, width, height, FColour, BColour, Step, Minimum, Maximum`	**Hiermee wordt een vak getekend met pictogrammen omhoog/omlaag aan beide uiteinden.** Wanneer deze pictogrammen worden aangeraakt, wordt het nummer in het vak verhoogd of verlaagd. Door de omhoog/omlaag-pictogrammen ingedrukt te houden,wordt de stap in een snel tempo herhaald. **'#ref'** is het referentienummer van de besturing. **'startX'** en **'startY'** zijn de coördinaten linksboven terwijl **'width'** en **'height'** zijn ingesteld de dimensies. **'FColour'** en **'BColor'** zijn **RGB** waarden voor de voorgrond en achtergrondkleuren. **'width'**, **'height', FColour** en **'BColour'** zijn optioneel en worden standaard gebruikt bij eerdere controles. **'Stap'** stelt de hoeveelheid in waarmee het nummer bij elke aanraking moet worden verhoogd/verlaagd. **'Minimum'** en **'Maximum'** stellen grenzen aan het aantal dat kan worden ingevoerd. Allemaal drie parameters kunnen getallen met drijvende komma zijn en zijn optioneel. De standaard voor **'Stap'** is **1** en **'Minimum'** en **'Maximum'**, indien weggelaten, worden standaard ingesteld op nee begrenzing. **Niet op de PicoMiteVGA**.

COMMANDO	Beschrijving
`GUI SWITCH #ref, caption$, startX, startY, width, height, FColour, BColour`	**Dit zal een vergrendelingsschakelaar tekenen die een vierkante schakelaar is die vergrendelt wanneer aangeraakt.** **'#ref'** is het referentienummer van de besturing. **'caption$'** is een tekenreeks die moet worden weergegeven als bijschrift op de voorkant van de schakelaar. **'startX'** en **'startY'** zijn de coördinaten linksboven terwijl **'width'** en **'height'** zijn ingesteld de dimensies. **'FColour'** en **'BColor'** zijn **RGB** waarden voor de voorgrond en achtergrondkleuren. **'breedte'**, **'hoogte'**, **FColour** en **'BColour'** zijn optioneel en standaard naar die gebruikt in eerdere controles. Bij aanraking lijkt het visuele beeld van de knop ingedrukt en de waarde van de regelaar is **1**. Bij een tweede keer aanraken zal de schakelaar losgelaten en de waarde keert terug naar nul. Bijschrift kan uit twee bestaan bijschriften gescheiden door een **| teken** (bijv. **"AAN | UIT"**). Wanneer dit wordt gebruikt, schakelaar lijkt een tuimelschakelaar te zijn waarbij elke helft van het bijschrift wordt gebruikt om label elke helft van de tuimelschakelaar. **Niet op de PicoMiteVGA**.
`GUI TEXTBOX #ref, startX, startY, width, height, FColour, BColour`	**Dit tekent een doos met afgeronde hoeken die kan worden gebruikt om een virtueel toetsenbord voor gegevensinvoer.** **'#ref'** is het referentienummer van de besturing. **'startX'** en **'startY'** zijn de coördinaten linksboven terwijl **'width'** en **'height'** zijn ingesteld de dimensies. **'FColour** en **'BColor'** zijn **RGB** waarden voor de voorgrond en achtergrondkleuren. **'breedte', 'hoogte', FColour** en **'BColour'** zijn optioneel en standaard naar die gebruikt in eerdere controles. Op een scherm dat ondersteunt transparante tekst **BKleur** kan **-1** zijn, wat betekent dat de achtergrond zal door de gaten in de karakters heen zichtbaar zijn. Voorbeeld 1 op een 320x240 TFT scherm: `OPTION GUI CONTROLS 75` `GUI TEXTBOX #1,0,0,318,100, RGB(black), RGB(cyan)` `' #ref, startX, startY, Breed, Hoog, FC, BC` Wanneer de box wordt aangeraakt, verschijnt er een **QWERTY toetsenbord** op het scherm. Met behulp van dit virtuele toetsenbord kan elke tekst in het vak worden ingevoerd, inclusief hoofdletters/kleine letters, cijfers en andere tekens in de **ASCII** karakterset. De nieuwe tekst vervangt alle tekst die eerder in het vak stond. De waarde van het besturingselement kan worden ingesteld op een tekenreeks die begint met twee hash-tekens. Voorbeeld 2: `CtrlVal(nnn) = "##Voer bestandsnaam in"` En in dat geval zal de string (zonder de eerste twee hash-tekens) zijn weergegeven in de doos met verminderde helderheid. Dit kan worden gebruikt om de gebruiker een hint geven over wat er moet worden ingevoerd (genaamd **"spooktekst"**).

COMMANDO	Beschrijving 194
	Het lezen van waarde van het besturingselement dat spooktekst weergeeft, retourneert een lege tekenreeks. Wanneer de besturing wordt normaal gebruikt, de spooktekst zal verdwijnen. MMBasic zal proberen het virtuele toetsenbord op het scherm te plaatsen om niet verduister het tekstvak waardoor het werd weergegeven. Een onderbreking van de pen naar beneden zal zijn gegenereerd net voordat het toetsenbord wordt ingezet en een toetsomhoogonderbreking zal zijn gegenereerd wanneer de Enter-toets wordt aangeraakt en het toetsenbord is verborgen. **Niet op de PicoMiteVGA**.
`GUI TEXTBOX CANCEL`	**Dit zal een virtueel toetsenbord sluiten als het op het scherm wordt weergegeven.** Het is de hetzelfde als wanneer de gebruiker de annuleertoets aanraakt, behalve dat de touch-up interrupt wordt niet gegenereerd. Als er geen toetsenbord wordt weergegeven, is deze opdracht voldoende niets. **Niet op de PicoMiteVGA**.
`GUI BITMAP x,y,bits [, width][, height] [, scale][, c][, bc]`	**Toont de bits in een bitmap op een LCD-scherm beginnend bij 'x' en 'y' op een aangesloten LCD-paneel.** **'hoogte'** en **'breedte'** zijn de afmetingen van de bitmap zoals weergegeven op de **LCD-paneel** en standaard ingesteld op **8x8**. **'scale'** is optioneel en wordt standaard ingesteld door het **FONT** commando. **'c'** is de kleur van de tekening en **'bc'** is de achtergrondkleur. Ze zijn optioneel en standaard de huidige voor- en achtergrondkleuren. De bitmap kan een geheel getal of een tekenreeksvariabele of constante zijn en wordt getekend de eerste byte gebruiken als de eerste bits van de bovenste regel (bit 7 eerst, dan bit 6, enz.) gevolgd door de volgende byte, enz. Als de bovenste regel is gevuld, volgt de volgende regel van de weergegeven bitmap begint met het volgende bit in het gehele getal of string. Zie de sectie Basis tekencomman-do's voor een definitie van de kleuren en grafische coördinaten.
`GUI CALIBRATE`	**Deze opdracht wordt gebruikt om de aanraakfunctie op een LCD paneel te kalibreren.** Het zal een reeks doelen op het scherm weergeven en wachten tot ze allemaal zijn precies aangeraakt. **Niet op de PicoMiteVGA**.
`GUI RESET LCDPANEL`	**Hiermee wordt het geconfigureerde LCD paneel opnieuw geïnitialiseerd.** Initialisatie is automatisch gedaan wanneer de PicoMite op-start, maar in sommige omstandigheden kan het zijn nodig om de stroom naar het LCD-paneel te onderbreken (bijvoorbeeld om batterijvermogen te sparen) en deze opdracht kan vervolgens worden gebruikt om de weergave opnieuw te initialiseren. **Niet op de PicoMiteVGA**.
`GUI TEST LCDPANEL`	**Zal het display of de aanraakfunctie op een LCD-paneel testen.** Met **GUI TEST LCDPANEL** wordt een geanimeerde weergave van en cirkels snel over elkaar heen getrokken.

COMMANDO	Beschrijving
GUI TEST TOUCH	**Met GUI TEST TOUCH (niet beschikbaar op de PicoMiteVGA) het scherm wordt leeg en wacht op een aanraking waardoor een witte stip wordt geplaatst op het display markeert de aanraakpositie op het scherm.** Elk teken dat op de console wordt ingevoerd, beëindigt de test.
HUMID pin, tvar, hvar [,DHT11]	**Retourneert de temperatuur en vochtigheid met behulp van de DHT22-sensor.** Alternatieve versies van de **DHT22** zijn de **AM2303** of de **RHT03** (allemaal compatibel). **'pin'** is de I/O-pin die op de sensor is aangesloten. Elke I/O-pin kan worden gebruikt. **'tvar'** is de variabele die de gemeten temperatuur vasthoudt en **'hvar'** is hetzelfde voor de vochtigheid. Beide moeten aanwezig zijn en beide moeten drijvende-kommavariabelen zijn. Bijvoorbeeld: **HUMID 3, TEMP!, HUMIT!** De temperatuur wordt gemeten in **ºC** en de luchtvochtigheid is het percentage relatieve vochtigheid. Beide worden gemeten met een resolutie van **0,1**. Als er een fout optreedt (sensor niet aangesloten of signaal beschadigd), zijn beide waarden **1000,0**. Normaal gesproken zou de **DHT22** gevoed moeten worden door **3,3V** om zijn output onder de **3,6V** te houden voor de **Raspberry Pi Pico** (de Pico 2 heeft dit probleem niet) en de signaalpin moet omhoog worden getrokken door een weerstand van 1K tot 10K (4,7K aanbevolen) tot **3,3V.** De optionele **DHT11** parameter wijzigt de timing zodat deze met de **DHT11** werkt. Stel in op **1** voor **DHT11** en **0** of laat het weg voor **DHT22**.
I2C	**Meer details vindt u in bijlage B**
I2C OPEN speed, timeout	**Activeert de eerste I2C module in mastermodus. 'snelheid' is de kloksnelheid (in KHz) om te gebruiken en moet een van 100, 400 of 1000 zijn.** **'time-out'** is een waarde in milliseconden waarna de master verzenden en ontvangen opdrachten worden onderbroken als ze niet zijn voltooid. Het minimum waarde is **100**. Een waarde van **nul** zal de timeout uitschakelen (hoewel dit niet wordt aanbevolen).
I2C WRITE addr, option, sendlen, senddata [,senddata..]	**Stuur gegevens naar de I2C slave-apparaat**. **'addr'** is het **I2C addr van de slaaf.** **'optie'** kan **0** zijn voor normaal bedrijf of **1** om **controle over de bus** te houden na de opdracht (er wordt geen stop-voorwaarde verzonden aan het einde van de opdracht). **'sendlen'** is het aantal bytes dat moet worden verzonden. **'senddata'** zijn de te verzenden gegevens dit kan op verschillende manieren worden gespecificeerd (alle verzonden waarden zullen tussen **0** en liggen **255**).

COMMANDO	Beschrijving
	Opmerkingen: • De gegevens kunnen als afzonderlijke bytes op de opdrachtregel worden aangeleverd. Voorbeeld: `I2C WRITE &H6F, 0, 3, &H23, &H43, &H25` • De gegevens kunnen zich in een eendimensionale array bevinden die is opgegeven met leeg beugels (dus geen afmetingen). **'sendlen'** bytes van de array worden verzonden beginnend met het eerste element. Voorbeeld: `I2C WRITE &H6F, 0, 3, ARRAY()` De gegevens kunnen een tekenreeksvariabele zijn (geen constante). Voorbeeld: `I2C WRITE &H6F, 0, 3, STRING$`
`I2C READ addr, option, rcvlen, rcvbuf`	**Haal gegevens op van het I2C-slave-apparaat.** **'addr' is het I2C adres van de slaaf.** **'optie'** kan **0** zijn voor normaal bedrijf of **1** om controle over de bus te houden na de opdracht (er wordt geen stop-voorwaarde verzonden aan het einde van de opdracht) **'rcvlen'** is het aantal te ontvangen bytes. **'rcvbuf'** is de variabele of array die wordt gebruikt om de ontvangen gegevens op te slaan - dit kan zijn: • Een stringvariabele. Bytes worden opgeslagen als opeenvolgende tekens. • Een eendimensionale reeks getallen gespecificeerd met lege haakjes. Ontvangen bytes worden opgeslagen in opeenvolgende elementen van de array beginnend met de eerste. Voorbeeld: `I2C READ &H6F, 0, 3, ARRAY()` Een normale numerieke variabele (in dit geval moet **rcvlen 1**) zijn.
`I2C CLOSE`	**Schakelt de master I2C-module uit.** Deze opdracht stuurt ook een stop als de bus wordt nog gehouden.
`I2C SLAVE`	**Zie bijlage B.**
`I2C2`	**Dezelfde reeks opdrachten als voor I2C (hierboven) maar van toepassing op de tweede I2C kanaal.**
`IF expr THEN stmt [: stmt]` of `IF expr THEN stmt ELSE stmt`	**Evalueert de uitdrukking 'expr' en voert de instructie uit die volgt op de THEN trefwoord als het waar is of springt naar de volgende regel als het niet waar is.** Als er meer zijn uitspraken op de regel (gescheiden door dubbele punten **(:)** ze zullen ook worden uitgevoerd als waar of overgeslagen als onwaar. Het **ELSE** trefwoord is optioneel en indien aanwezig de instructie(s) die erop volgen, worden uitgevoerd als **'expr'** is opgelost als onwaar. De constructie **'THEN statement'** kan ook worden vervangen door: **GOTO regelnummer | label'**. Dit type **IF statement** staat allemaal op één regel.

COMMANDO	Beschrijving
`IF expression THEN` `  <statements>` `[ELSEIF expression THEN` `  <statements>]` `[ELSE` `  <statements>]` `ENDIF`	**Multiline IF-instructie met optionele ELSE- en ELSEIF-cases en END met ENDIF.** Elk onderdeel staat op een aparte regel. Evalueert **'uitdrukking'** en voert de instructie(s) uit die volgen op **THEN** als de expressie is waar of optioneel de instructie(s) die volgen op de **ELSE** verklaring indien onwaar. De **ELSEIF** instructie (indien aanwezig) wordt uitgevoerd als de vorige voorwaarde is onwaar en het begint een nieuwe **IF** keten met verder **ELSE** en / of **ELSEIF** verklaringen zoals vereist. Een **ENDIF** wordt gebruikt om de **multiline IF** te beëindigen.
`INC var [,increment]`	**Verhoogt de variabele "var" met 1 of, indien gespecificeerd, de waarde in verhogen.** **"toename"** kan een negatief hebben. Dit is functioneel hetzelfde als **var = var + verhoging** maar wordt veel sneller verwerkt.
`INPUT ["prompt$";] var1 [,var2 [, var3 [, enz.]]]`	**Zal een lijst met waarden nemen, gescheiden door komma's (,) ingevoerd op de console en zal ze toewijzen aan een sequentiële lijst met variabelen.** Als het commando bijvoorbeeld is: `INPUT a, b, c` En het volgende wordt op het toetsenbord getypt: `23, 87, 66` Dan is **a = 23** en **b = 87** en **c = 66** De lijst met variabelen kan een mix zijn van **float-**, **integer-** of **string-**variabelen. De waarden die op de console worden ingevoerd, moeten overeenkomen met het type variabele. Als een enkele waarde wordt ingevoerd, is een komma niet vereist (maar die waarde mag geen komma bevatten). **'prompt$'** is een tekenreeksconstante(geen variabele of uitdrukking) en indien opgegeven het wordt eerst afgedrukt. Normaal gesproken wordt de prompt afgesloten met een puntkomma **(;)** en in dat geval wordt er een vraagteken afgedrukt na de prompt. Als de prompt wordt afgesloten met een komma **(,)** in plaats van de puntkomma **(;)** de vraagteken wordt onderdrukt.
`INPUT #nbr, list of variables`	**Hetzelfde als hierboven, behalve dat de invoer wordt gelezen vanaf een seriële poort of bestand eerder geopend voor INPUT als 'nbr'.** Zie het commando **OPEN**.
`INTERRUPT [myint]`	**Deze opdracht activeert een software-interrupt.** De interrupt wordt ingesteld met `INTERRUPT 'myint'` waarbij **'myint'** de naam is van een subroutine die wordt uitgevoerd wanneer de interrupt wordt geactiveerd. Gebruik `INTERRUPT 0` om de interrupt uit te schakelen. Gebruik **INTERRUPT** zonder parameters om de interrupt te activeren NB: de interrupt kan ook worden geactiveerd vanuit een **CSUB**.

COMMANDO	Beschrijving
`IR dev, key, int` of `IR CLOSE`	**Decodeert NEC of Sony infrarood afstandsbedieningssignalen.** Een **IR-ontvangermodule** wordt gebruikt om het **IR-licht** te detecterenen de signaal. Het kan op elke pin worden aangesloten, maar deze pin moet worden geconfigureerd geavanceerd met het commando: `SETPIN n, IR` Het decoderen van het IR signaal gebeurt op de achtergrond en het programma zal na deze opdracht zonder onderbreking verder gaan. **'dev'** en **'key'** moeten zijn numerieke variabelen en hun waarden worden bijgewerkt wanneer er een nieuw signaal is ontvangen (**'dev'** is de apparaatcode die door de afstandsbediening wordt ver-zonden en **'sleutel'** is de toets die is ingedrukt). **'int'** is een door de gebruiker gedefinieerde subroutine die wordt aangeroepen als er een nieuwe toets wordt ingedrukt ontvangen of wanneer de bestaande toets wordt ingedrukt voor automatisch herhalen. In de interrupt subroutine het programma kan de variabelen **'dev'** en **'key'** onderzoeken en passende maatregelen nemen.
`IR CLOSE`	**Het IR CLOSE commando beëindigt de IR-decoder.** Merk op dat voor het NEC-protocol de bits in **'dev'** en **'key'** zijn omgekeerd. Voor in **'sleutel'** moet **bit 0** bijvoorbeeld **bit 7** zijn, **bit 1** moet **bit 6** zijn, enz. Dit heeft geen invloed op normaal gebruik,maar als u op zoek bent naar een specifieke numerieke code geleverd door een fabrikant, moet u de bits omkeren. Dit beschrijft hoe om het te doen: http://www.thebackshed.com/forum/forum_posts.asp?TID=8367 Zie de sectie Speciale hardwareapparaten voor meer details.
`IR SEND pin, dev, key`	**Genereer een 12-bits Sony Remote Control protocol infrarood signaal 'pin' is de te gebruiken I/O-pin.** Dit kan elke I/O-pin zijn die automatisch wordt geconfigureerd als uitgang en moet worden aangesloten op een infrarood-LED. Inactief is laag met hoge niveaus die aangeven wanneer de LED moet worden ingeschakeld. **'dev'** is het apparaat dat wordt bestuurd en is een getal van **0 tot 31**, **'key'** is de gesimuleerde toetsaanslag en is een getal van **0 tot 127**. Het **IR-signaal** wordt gemoduleerd op ongeveer **38 KHz** en het verzenden van het signaal duurt ongeveer **25 mS**.
`KEYPAD var, int,` `r1, r2, r3, r4,` `c1, c2, c3 [, c4]`	**Bewaak en decodeer toetsaanslagen op een 4x3 of 4x4 toetsenbord**. Bewaking van het toetsenbord gebeurt op de achtergrond en het programma zal ga na deze opdracht zonder onderbreking verder. **'var'** moet een numeriek zijn variabele en de waarde ervan wordt bijgewerkt wanneer een toetsaanslag wordt gedetecteerd. **'int'** is een door de gebruiker gedefinieerde subroutine die wordt aangeroepen als er een nieuwe toets wordt ingedrukt hebben ontvangen. In de interrupt-subroutine kan het programma de variabele onderzoeken **'var'** en neem passende maatregelen.

	r1, r2, r3 en r4 zijn pinnummers die worden gebruikt voor de vier rij- en verbindingen met de toetsenbord en **c1, c2, c3 en c4** zijn de ko-lomaansluitingen. **c4** is optioneel en wordt alleen gebruikt met **4x4** bedieningspanelen. Deze opdracht zal automatisch configureren deze pinnen indien nodig. Als u op een toets drukt, is de waarde die is toegewezen aan **'var'** het nummer van een numerieke toets (bijv. **'6'** levert **6**) of **10** op voor de ***-toets** en **11** voor de **#-toets**. Op **4x4** toetsenborden wordt het getal **20** teruggegeven voor **A**, **21** voor **B**, **22** voor **C** en **23** voor **D**.
`KEYPAD CLOSE`	**Het KEYPAD CLOSE commando beëindigt de bediendeelfunctie en zet de I/O-pin terug in een niet-geconfigureerde staat.** Zie de sectie Speciale hardwareapparaten voor meer details.
`KILL file$ [,all]`	**Verwijdert het bestand gespecificeerd door 'file$'. Als er een exten-sie is, moet deze zijn gespecificeerd.** Bulk wissen wordt geactiveerd als **fname$** een **'*'** of een **'?'** bevat karakter. Als de optionele parameter **'all'** wordt gebruikt, wordt u om een enkele bevestiging gevraagd. Als **'alles'** niet is opgegeven, wordt u bij elk bestand om een vraag gevraagd.
`LCD INIT` `d4, d5, d6, d7, rs, en` of `LCD line, pos, text$` of `LCD CLEAR` of `LCD CLOSE`	**Geef tekst weer op een LCD-tekendisplaymodule.** Deze opdracht werkt met de meeste **1-regelige, 2-regelige of 4-rege-lige LCD-modules** die de **KS0066-, HD44780-** of **SPLC780** controller gebruiken (dit kan echter niet worden gegarandeerd). Het **LCD INIT** commando wordt gebruikt om de LCD-module te initia-liseren voor gebruik. **'d4'** tot **'d7'** zijn de I/O-pinnen die verbinding maken met ingangen **D4** tot **D7** op de LCD-module (ingangen **D0** tot **D3** moeten met aarde worden verbonden). **'rs'** is de pin die is aangesloten op de registerselectieingang op de module (ook wel **CMD** genoemd). **'en'** is de pin die is aangesloten op de enable- of chip-select-ingang op de module. De **R/W** ingang op de module moet **altijd geaard zijn**. De bovenstaande I/O-pinnen zijn automatisch ingesteld op uitgang-en met deze opdracht. Wanneer de module is geïnitialiseerd, kunnen er gegevens naar wor-den geschreven met behulp van het LCD-commando. **'line'** is de regel op het display (**1 t/m 4**) en **'pos'** is de tekenlocatie op de regel (de eerste locatie is **1**). **'text$'** is een string die de tekst bevat die naar het LCD-scherm moet worden geschreven. **'pos'** kan ook **C8, C16, C20 of C40** zijn. In dat geval wordt de regel ge-wist en wordt de tekst gecentreerd op een display met **8, 16, 20** of **40** regels.

COMMANDO	Beschrijving 200
	Bijvoorbeeld: `LCD 1, C16, "Hallo"` **LCD CLEAR** wist alle gegevens die op het LCD-scherm worden weergeven en **LCD CLOSE** beëindigt de LCD-functie en zet alle I/O-pinnen terug naar de niet-geconfigureerde status. Zie het hoofdstuk Speciale hardwareapparaten voor meer details.
LCD CMD d1 [, d2 [, etc]] of **LCD DATA d1 [, d2 [, etc]]**	Deze commando's sturen een of meer bytes naar een LCD-scherm als commando (**LCD CMD**) of als data (**LCD DATA**). Elke byte is een getal tussen 0 en 255 en moet worden gescheiden door komma's. Het LCD-scherm moet eerder zijn geïnitialiseerd met het **LCD INIT**-commando (zie hierboven). Deze opdrachten kunnen worden gebruikt om een niet-standaard LCD-scherm in **"onbewerkte modus"** aan te sturen, of ze kunnen worden gebruikt om gespecialiseerde functies in te schakelen, zoals scrollen, cursors en aangepaste tekensets. U moet het gegevensblad van uw LCD raadplegen om de benodigde opdracht- en gegevenswaarden te vinden.
LET variable = expression	**Wijst de waarde van 'expressie' toe aan de variabele.** **LET** is automatisch aangenomen als een statement niet begint met een com-mando. Bijvoorbeeld: `Var = 56`
LIBRARY SAVE, **LIBRARY DELETE** en **LIBRARY LIST**	**De bibliotheekopdrachten kunnen alleen worden uitgevoerd vanaf de opdrachtprompt.** Het **LIBRARY** commando kan worden gebruikt in samen gesteld instructies, waardoor het beter kan worden gebruikt in programmeerbare functies. In de Micromites voegt de bibliotheek zichzelf in voor het einde van het hoofdprogrammagebied, waardoor het beschikbare normale programmagebied wordt verkleind, maar wat er nodig is efficiënter wordt gebruikt, wat resulteert in een beter gebruik van het programmagebied en ook de bibliotheekfuncties. In de Picomite-implementatie gebruikt de bibliotheek de flitser van de laatste Flash-sleuf (momenteel 3), dus voegt de bibliotheek toe zonder het programmagebied te verkleinen. Het gebruik van de Bibliotheek en de laatste Flash Slot sluiten elkaar wederzijds uit. Het eerste **LIBRARY SAVE** commando claimt het Flash-slot als het niet in gebruik is. Het **FLASH** commando kan slot 3 claimen als het nog niet door de bibliotheek is gebruikt. Wanneer een programma wordt uitgevoerd vanuit het hoofdprogrammagebied, wordt elke normale code in de bibliotheek uitgevoerd vóór het hoofdprogramma. Het hoofdprogramma wordt dan uitgevoerd en heeft toegang tot alle routines in de bibliotheek. Het uitvoeren van een programma dat is opgeslagen in flash met **FLASH RUN [1-3]** zal elke normale code die is opgeslagen in de bibliotheek uitvoeren vóór het hoofdprogramma, en het programma heeft toegang tot alle **SUB's**, **FUNCTION**, **FONT's** of **CSUB's** in de bibliotheek.

COMMANDO	Beschrijving 201
LIBRARY DELETE	**LIBRARY DELETE verwijdert de bibliotheek en keert Flash Slot 3 terug voor normaal gebruik.** (**OPTION RESET** will doet het zelfde).Het wist ook het programma-gebied.
LIBRARY DISK SAVE fname$ LIBRARY DISK LOAD fname$	**LIBRARY DISK SAVE fname$ slaat de huidige bibliotheek op als een binair bestand, waardoor een volgende aanroep van LIBRARY DISK LOAD fname$ mogelijk is om de bibliotheek te herstellen.** Samen zorgen deze ervoor dat bibliotheken die specifiek zijn voor individuele programma's eenvoudig kunnen worden opgeslagen, hersteld en gedistribueerd. Naast het gebruik van versiespecifieke functionaliteit in de bibliotheek (**WEB, VGA, GUI**) kunnen bibliotheken tussen versies worden gedeeld.
LIBRARY LIST	**LIBRARY LIST geeft de inhoud van de bibliotheek weer.** De bibliotheek slaat geen opmerkingen of de HEX-code voor FONT's of CSUB's op. Een kopie van de bibliotheekbron moet extern of in een van de **Flash-slots** worden bewaard als u deze moet wijzigen of herstellen. bijv. Schrijf uw bibliotheekcode in het hoofdprogrammagebied. Maak een back-up naar een flash-slot, FLASH SAVE 3
LIBRARY LIST ALL	**LIBRARY LIST ALL geïmplementeerd.**
LIBRARY SAVE	**Voer vervolgens LIBRARY SAVE uit, de code in het normale programmageheugen wordt nu naar de bibliotheek verplaatst (opmerkingen verwijderd, HEX voor lettertypen en CSUB's verwijderd) en het hoofdprogramma-gebied gewist.** Om de bibliotheek te wijzigen, **LIBRARY DELETE,** FLASH LOAD 3 om de bron terug te krijgen, EDIT commando om te wijzigen. FLASH **OVERWRITE 3** om er een back-up van te maken, LIBRARY SAVE om het in de bibliotheek te plaatsen of bewaar alles gewoon in **MMEdit**. LIBRARY SAVE is cumulatief. d.w.z. volgende LIBRARY SAVEs zullen alles wat zich momenteel in het hoofdprogrammagebied bevindt toevoegen aan de bibliotheek.
LINE AA x1, y1, x2, y2 [, LW [, C]]	**Tekent een lijn met anti-aliasing.** De parameters zijn zoals die van het LINE commando. Deze versie gebruikt variabeleintensiteitswaarden van de opgegeven kleur om de **"verspringende"** kwaliteit van diagonale lijnen te verminderen. **NB:** Bij lage kleurresoluties (PicoMiteVGA en framebuffers op de PicoMite) hebben andere kleuren dan groen mogelijk geen voordeel van antialiasing en kunnen kleuren zoals groen (egyellow) een andere tint laten zien in de pixels met lagere intensiteit. Voorbeeld: `ON ERROR SKIP : MODE 2` `For i = 0 To 360 Step 10` `x=100*Cos(Rad(i)) : y=100*Sin(Rad(i))` `Line aa MM.HRes\2,MM.VRes\2,MM.HRes\2-x, MM.VRes\2-y,3,RGB(green)` `Next` `Do : Loop`

COMMANDO	Beschrijving 202
LINE GRAPH x(), y(), colour	**Deze opdracht genereert een lijngrafiek van de coördinatenparen gespecificeerd in "x()" en "y()".** De grafiek heeft **n-1** segmenten met **n** elementen in de **x** en **y** arrays.
LINE x1, y1, x2, y2 [, LW [, C]]	**Op een aangesloten LCD scherm tekent deze opdracht een lijn die begint bij de coördinaten 'x1' en 'y1' en eindigend op 'x2' en 'y2'.** **'LW'** is de lijnbreedte en is alleen geldig voor horizontale of verticale lijnen. Het standaard ingesteld op 1 indien niet gespecificeerd of als de lijn een diagonaal is. **'C'** is een geheel getal vertegenwoordigt de **kleur** en is standaard ingesteld op de huidige voorgrondkleur. Alle parameters kunnen worden uitgedrukt als arrays en de software plot de aantal lijnen zoals bepaald door de afmetingen van de kleinste array. **'x1', 'y1'**, **'x2'** en **'y2'** moeten allemaal arrays zijn of allemaal enkele variabelen /constanten anders wordt er een fout gegenereerd. **'lw'** en **'c'** kunnen ofwel **arrays** zijn of enkele variabelen/constanten
LINE INPUT [prompt$,] string-variabele$	**Leest een hele regel van de console-invoer in 'string-variable$'.** **'prompt$'** is een tekenreeksconstante (geen variabele of uitdrukking) en indien opgegeven het wordt eerst afgedrukt. In tegenstelling tot **INPUT** leest dit commando een hele regel en stopt het niet voor door komma's gescheiden gegevensitems. Een vraagteken wordt niet afgedrukt tenzij het onderdeel is van een **'prompt$'**.
LINE INPUT #nbr, string-variabele$	**Hetzelfde als hierboven, behalve dat de invoer wordt gelezen van een seriële communicatie port of een bestand dat eerder is geopend voor INPUT als 'nbr'.** Zie de **OPEN** opdracht.
LINE PLOT ydata([,nbr] [,xstart] [,xinc] [,ystart] [,yinc] [,colour]	**Tekent een lijngrafiek uit een reeks gegevenspunten op de y-as.** **'ydata'** is een array van floats of gehele getallen die moeten worden geplot. **'nbr'** is het aantal lijnsegmenten dat moet worden geplot. Standaard ingesteld op de kleinste van de arraygrootte en **MM.HRES 2** indien weggelaten. **'xstart'** is de **x-coördinaat** om te beginnen met plotten. Standaard ingesteld op **0.** **'xinc'** is de stapgrootte langs de **x-as** om elke coördinaat uit te zetten Standaard ingesteld op **1**. **'ystart'** is de locatie in ydata waar de plot wordt gestart. Standaard is dit de start van de array. **'yinc'** is de toename in **ydata** die moet worden toegevoegd voor elk uit te zetten punt. **'kleur' is de kleur om de lijn te trekken.**
LIST [fname$]	**Lijst van een programma op de console.** **LIST** op zichzelf zal het programma weergeven met een pauze bij elk scherm tot het vol is.

COMMANDO	Beschrijving
LIST ALL [fname$]	**LIST ALL zal het programma zonder pauzes weergeven.** Dit is handig als u dat wilt breng het programma over naar een terminal emulator op een pc die de mogelijkheid heeft om de invoerstroom vastleggen in een bestand. Als de optionele **'fname$'** is opgegeven, dan dat bestand op de SD-kaart wordt weergegeven.
LIST COMMANDS	**Geeft een overzicht van alle geldige opdrachten of functies.** Geeft na **> LIST COMMANDS** een lijst van totaal **152** commands. De lijst kan afwijken bij **PicoMite**, **PicoMiteVGA**, **WebMite**. Versie: **PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.01** Voorbeeld: `> list commands` `*/ /* ADC Arc` `Autosave Blit Blit Memory Box` `Call Case Case Else Cat` `Chdir Circle Clear Close` `CLS CMM2 Load CMM2 Run Color` `Colour Colour Map Configure Const` `Continue Copy CPU CSub` `Data Date$ DefineFont Device` `Dim Do Draw3D Drive` `Edit Edit File Else Else If` `ElseIf End End CSub End DefineFont` `End Function End If End Select End Sub` `EndIf Erase Error Execute` `Exit Exit Do Exit For Exit Function` `Exit Sub Files Flash Flush` `Font For FRAMEBUFFER Function` `Gamepad GoSub GoTo GUI` `Humid I2C I2C2 If` `Inc Input Interrupt IR` `IReturn Keypad Kill LCD` `Let Library Line Line Input` `List Load Local LongString` `Loop Map Map( Math` `Memory MID$( Mkdir MODE` `Mouse New Next On` `OneWire Open Option Pause` `Pin( PIO Pixel Play` `Poke Polygon Port( Print` `Pulse PWM Randomize RBox` `Read Rem Rename Restore` `Return Rmdir RTC Run` `Save Seek Select Case Servo` `SetPin SetTick Sort SPI` `SPI2 Sprite Static Sub` `SYNC TEMPR START Text TILE` `Time$ Timer Trace Triangle` `VAR WatchDog While Wii` `Wii Classic Wii Nunchuck WS2812 XModem` `Total of 152 commands`

COMMANDO	Beschrijving
`LIST FUNCTIONS`	**Geeft na `> LIST FUNCTIONS` een lijst van totaal 134 functions en operators.** De lijst kan afwijken bij PicoMite, PicoMiteVGA, WebMite. Versie: **PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.01** Voorbeeld: `> list functions` `*  +  -  /` `<  <<  <=  <>` `=  =<  =>  >` `>=  >>  @(  \` `^  Abs(  ACos(  And` `As  Asc(  ASin(  Atan2(` `Atn(  Bin$(  Bin2str$(  Bound(` `Call(  Choice(  Chr$(  Cint(` `Cos(  Cwd$  Date$  DateTime$(` `Day$(  Deg(  DEVICE(  Dir$(` `Distance(  DRAW3D(  Else  Eof(` `Epoch(  Eval(  Exp(  Field$(` `Fix(  For  Format$(  GetScanLine` `GoSub  GoTo  GPS(  Hex$(` `Inkey$  Input$(  Instr(  Int(` `INV  KeyDown(  LCase$(  LCompare(` `Left$(  Len(  LGetByte(  LGetStr$(` `LInStr(  LLen(  Loc(  Lof(` `Log(  Map(  Math(  Max(` `Mid$(  Min(  MM.Device$  MM.ErrMsg$` `MM.Errno  MM.Fontheight  MM.Fontwidth  MM.HPOS` `MM.HRes  MM.I2C  MM.Info$(  MM.Info(` `MM.OneWire  MM.PS2  MM.VER  MM.VPOS` `MM.VRes  MM.Watchdog  Mod  Not` `Oct$(  Or  Peek(  Pi` `Pin(  Pio(  Pixel(  Port(` `Pos  Pulsin(  Rad(  RGB(` `Right$(  Rnd  Rnd(  Sgn(` `Sin(  Space$(  SPI(  SPI2(` `sprite(  Sqr(  Step  Str$(` `Str2bin(  String$(  Tab(  Tan(` `TEMPR(  Then  Time$  Timer` `To  UCase$(  Until  Val(` `While  Xor` `Total of 134 functions and operators`
`LOAD file$ [,R]`	**Laadt een programma genaamd "file$" van een SD-kaart in het programmageheugen.** Als het optionele achtervoegsel ,**R** is toegevoegd, het programma wordt onmiddellijk uitgevoerd zonder prompt (in dit geval moet "**file$**" een tekenreeksconstante zijn). Als er geen extensie is opgegeven, wordt `".BAS"` aan de bestandsnaam toegevoegd. b.v. `LOAD "A:/mijnapp"`
`LOAD IMAGE file$ [, x] [, y]`	**Laad een bitmapafbeelding van de SD-kaart en geef deze weer op het LCD-scherm.** **"file$"** is de naam van het bestand en **'x'** en **'y'** zijn de schermcoördinaten voor de linker bovenhoek van de afbeelding. Als de coördinaten niet zijn opgegeven, wordt de afbeelding wordt in de linkerbovenhoek op het scherm getekend.

	Als er geen extensie is opgegeven, wordt **".BMP"** aan de bestandsnaam toegevoegd. Alle typen van het BMP-formaat worden ondersteund, inclusief zwart-wit en 24-bits afbeeldingen in ware kleuren.
LOAD JPG file$ [, x] [, y]	**Laad een jpg-afbeelding van de SD kaart en geef deze weer op het LCD-paneel.** "**bestand$**" is de naam van het bestand en '**x**' en '**y**' zijn de scherm-coördinaten voor de top linkerhoek van de afbeelding. Als de coördinaten niet zijn opgegeven, wordt de afbeelding wordt linksboven op het scherm getekend. Als er geen extensie is opgegeven, wordt **".JPG"** toegevoegd aan de bestandsnaam. Progressieve jpg afbeeldingen worden niet ondersteund.
LOAD PNG fname$ [, x] [, y] [,transparent] [,alphacut]	**Laadt en toont een png-bestand 'fname'.** Als er geen extensie is opgegeven, wordt **.png** automatisch aan de bestandsnaam toegevoegd. Het bestand moet de **RGBA8888** indeling hebben, wat de normale standaard is. Indien gespecificeerd geven '**x**' en '**y**' aan waar op het scherm of de framebuffer de afbeelding zal verschijnen. De optionele parameter '**transparant**' (standaard ingesteld op **0**) specificeert een van de kleurcodes (**0-15**) die wordt toegewezen aan pixels in het **png** bestand met een alfawaarde kleiner dan '**alphacut**' (standaard ingesteld op **20**). Als '**transparant**' is ingesteld op **-1**, wordt de **png** afbeelding geschreven met pixels met een alfawaarde kleiner dan '**alphacut**' die volledig zijn gemist.
LOCAL variabele [, variabeles] **Zie DIM voor de volledige syntaxis.**	**Definieert een lijst met variabelenamen als lokaal voor de subroutine of functie.** Deze opdracht gebruikt exact dezelfde syntaxis als **DIM** en maakt variabelen die alleen zichtbaar zijn binnen de subroutine of functie. Zij wordt automatisch verwijderd wanneer de subroutine of functie wordt afgesloten.
LONGSTRING	**De LONGSTRING commando's maken het mogelijk om strings langer te manipuleren dan de normale MMBasic limiet van 255 tekens**. Variabelen voor het vasthouden van lange strings moeten worden gedefinieerd als enkeldimensionaal integer-arrays met het aantal elementen ingesteld op het aantal tekens vereist voor de maximale string lengte gedeeld door acht. De reden voor delen door acht is dat elk geheel getal in een MMBasic-array acht bytes inneemt. Merk op dat de lange tekenreeksroutines niet controleren op overloop in de lengte van de strings. Als een poging wordt gedaan om een string te maken die langer is dan de grootte van de lange tekenreeksvariabele zal de uitkomst ongedefinieerd zijn.

COMMANDO	Beschrijving 206
LONGSTRING AES128 ENCRYPT/DECRYPT CBC/ECB/CTR key$/key[!/%](), in%(), out%() [,iv$/iv[!/%]()]	**Codeert of decodeert de longstring in in%() en plaatst het antwoord in out%().** Voor **CBC-** en **CTR-modi** genereert de codering een willekeurige initialisatievector en wordt **out%()** voorafgegaan door de **IV**. Als er een expliciete **IV** is gespecificeerd, wordt deze gebruikt in plaats van de willekeurige vector en wordt deze toegevoegd aan **out%()** Voor **CBC-** en **CTR-decodering** gaat de firmware ervan uit dat de eerste **16 bytes** van **in%()** de initialisatievector zijn. In het geval dat u een bericht zonder **IV** wilt verzenden, kunt u **LONGSTRING RIGHT** gebruiken om de **IV** te verwijderen voordat u het bericht verzendt. In dit geval moet de ontvanger zowel de IV als de sleutel kennen en een volledige longstring aanmaken voordat hij **DECRYPT** gebruikt. Dit kan worden gedaan door **LONGSTRING CONCAT** te gebruiken om het binnenkomende bericht toe te voegen aan een longstring die de **IV** bevat.
LONGSTRING APPEND array%(), string$	**Voeg een normale MMBasic-tekenreeks toe aan een lange tekenreeksvariabele.** **array%()** is een lange tekenreeks-variabele terwijl **string$** een normale MMBasic-tekenreeksexpressie is.
LONGSTRING BASE64 ENCODE | DECODE in%(), out%()	**Deze BASE64 codeert of decodeert de longstring in in%() en plaatst het antwoord in out%().** De array die als uitvoer wordt gebruikt, moet groot genoeg zijn in verhouding tot de invoer en de richting. Bij het **code**ren wordt de lengte met **4/3 vergroot** en bij het **decoderen** wordt de lengte met **3/4 verkort.**
LONGSTRING CLEAR array%()	**Wist de lange string variabele array%().** d.w.z. het wordt ingesteld op een lege string.
LONGSTRING COPY dest%(), src%()	**Kopieer de ene lange string naar de andere.** **dest%()** is de bestemmingsvariabele en **src%()** is de bronvariabele. Wat in **dest%()** stond, wordt overschreven.
LONGSTRING CONCAT dest%(), src%()	**Verbind de ene lange string met de andere.** **dest%()** is de bestemmingsvariabele en **src%()** is de bronvariabele. **src%()** wordt toegevoegd aan het einde van **dest%()** (de bestemming wordt niet overschreven).
LONGSTRING LCASE array%()	**Zal alle hoofdletters in array%() naar kleine letters converteren.** **array%()** moet een lange tekenreeksvariabele zijn.
LONGSTRING LEFT dest%(), src%(), nbr	**Kopieer de linker 'nbr' tekens van src%() naar dest%() overschrijven wat er ook in dest%() zat.** d.w.z. kopiëren vanaf het begin van **src%()** **dest%()** moeten lange tekenreeksvariabelen zijn. **'nbr'** moet een geheel getal constant zijn of uitdrukking.
LONGSTRING LOAD array%(),nbr,string$	**Kopieer 'nbr'-tekens van string$ naar de lange string-variabele array%() overschrijven wat zich in array%() bevond.**

COMMANDO	Beschrijving 207
LONGSTRING MID dest%(), src%(), start, nbr	**Kopieer 'nbr'-tekens van src%() naar dest%() beginnend bij teken positie 'start' overschrijven wat zich in dest%() bevond.** d.w.z. kopiëren van de midden van **src%()**. **'nbr'** is optioneel en indien weggelaten de karakters van **'start'** tot het einde van de string wordt gekopieerd **src%()** en **dest%()** moet een lange string zijn variabelen. **'start'** en **'nbr'** moeten een geheel getal constanten of uitdrukkingen zijn.
LONGSTRING PRINT [#n,] src%()	**Drukt de lange string die is opgeslagen in 'src%()' af naar het bestand of de COM-poort die is geopend als '#N'.** Als **'#n'** niet is opgegeven, wordt de uitvoer naar de console verzonden.
LONGSTRING REPLACE array%(), string$, start	**Zal karakters in de normale MMBasic string string$ vervangen door een bestaande lange string array%() beginnend op positie 'start' in de lange string**.
LONGSTRING RESIZE addr%(), nbr	**Stelt de grootte van de longstring in op nbr.** Dit heft de grootte op die door andere is ingesteld **longstring** commando's moeten dus met de nodige voorzichtigheid worden gebruikt. Typisch gebruik zou zijn bij het gebruik van een lange string als byte-array.
LONGSTRING RIGHT dest%(), src%(), nbr	**Zal de rechterhand 'nbr'-tekens kopiëren van src%() naar dest%() overschrijven wat er ook in dest%() zat.** d.w.z. kopiëren vanaf het einde van **src%()**. **src%()** en **dest%()** moeten lange tekenreeksvariabelen zijn. **'nbr'** moet een geheel getal constant zijn of uitdrukking.
LONGSTRING SETBYTE addr%(), nbr, data	**Stelt byte nbr in op de waarde "data", nbr respecteert OPTION BASE.**
LONGSTRING TRIM array%(), nbr	**Knipt 'nbr'-tekens vanaf de linkerkant van een lange reeks.** **array%()** moet een lange tekenreeksvariabelen zijn. '**nbr**' moet een geheel getal constante of uitdrukking zijn.
LONGSTRING UCASE array%()	**Zet alle kleine letters in array%() om in hoofdletters**. **Reeks%()** moet een lange tekenreeksvariabele zijn.
LOOP [UNTIL expression]	**Beëindigt een programmaloop: zie DO.**
MAP	**ALLEEN HDMI VERSIE** Met de **MAP** opdrachten kan de programmeur de kleuren instellen die worden gebruikt in **4- of 8-bits** kleurmodi. Elke waarde in het **4-** of **8-bits** kleurenpalet kan worden ingesteld op een onafhankelijke **24-bits kleur** (dwz **RGB555**-indeling). Zie de **MAP** functie voor meer informatie.
MAP(n) = rgb%	**Hierdoor wordt de 24-bits kleur 'rgb%' toegewezen aan alle pixels met de 4-bits kleurwaarde 'n'.** De **RGB** waarde wordt omgezet naar een van de beschikbare **16 VGA RGB121** kleuren zoals ingesteld door het weerstandsnetwerk. De wijziging wordt geactiveerd na het **MAP SET** commando.

<table>
<tr><th>COMMANDO</th><th>Beschrijving</th></tr>
<tr><td>MAP MAXIMITE</td><td>Hierdoor wordt de kleurenkaart ingesteld op de kleuren die zijn geïmplementeerd in de originele Color Maximite.</td></tr>
<tr><td>MAP GREYSCALE</td><td>Hiermee wordt de kleurenkaart ingesteld op 16 of 32 grijstinten (afhankelijk van de MODUS).
MAP GREYSCALE is ook geldig.</td></tr>
<tr><td>MAP SET</td><td>Dit zorgt ervoor dat MMBasic de kleurenkaart (ingesteld met MAP(n) = rgb%) bijwerkt tijdens het volgende frameblankings-interval.</td></tr>
<tr><td>MAP RESET</td><td>Hiermee wordt de kleurenkaart teruggezet naar de standaard-kleuren.</td></tr>
<tr><td>MAP</td><td><u>ALLEEN VGA VERSIE</u>
Met de MAP opdrachten kan de programmeur de kleuren selec-teren die in 4-bits kleurmodi worden gebruikt.
Elke waarde in het 4-bits kleurenpalet kan worden ingesteld op een van de 16 beschikbare kleuren. Zie de MAP functie voor informatie.</td></tr>
<tr><td>MAP(n) = rgb%</td><td>Hiermee wordt de 24-bits kleur 'rgb%' toegewezen aan alle pixels met de 4-bits kleurwaarde 'n'.
De RGB waarde wordt omgezet naar een van de 16 beschikbare VGA RGB121 kleuren, zoals ingesteld door het weerstandsnetwerk. De wijziging wordt geactiveerd na de MAP SET opdracht.</td></tr>
<tr><td>MAP MAXIMITE</td><td>Hiermee wordt de kleurenkaart ingesteld op de kleuren die in de originele Colour Maximite zijn geïmplementeerd.</td></tr>
<tr><td>MAP SET</td><td>Hierdoor werkt MMBasic de kleurenkaart bij tijdens het volgende frame-blankinginterval. Ingesteld met MAP(n) =rgb%</td></tr>
<tr><td>MAP RESET</td><td>Hij zal de kleurenkaart resetten naar de standaardkleuren die in de 4-bits modus zijn:
<table>
<tr><th>'n'</th><th>Kleur</th><th>Kleurwaarde</th></tr>
<tr><td>15</td><td>WIT</td><td>RGB(255,255,255)</td></tr>
<tr><td>14</td><td>GEEL</td><td>RGB(255,255,000)</td></tr>
<tr><td>13</td><td>LILA</td><td>RGB(255,128,255)</td></tr>
<tr><td>12</td><td>BRUIN</td><td>RGB(255,128,000)</td></tr>
<tr><td>11</td><td>FUCHSIA</td><td>RGB(255,064,255)</td></tr>
<tr><td>10</td><td>ROEST</td><td>RGB(255,064,000)</td></tr>
<tr><td>9</td><td>MAGENTA</td><td>RGB(255,000,255)</td></tr>
<tr><td>8</td><td>ROOD</td><td>RGB(255,000,000)</td></tr>
<tr><td>7</td><td>CYAAN</td><td>RGB(000,255,255)</td></tr>
<tr><td>6</td><td>GROEN</td><td>RGB(000,255,000)</td></tr>
<tr><td>5</td><td>CERULEA</td><td>RGB(000,128,255)</td></tr>
<tr><td>4</td><td>MIDDENGROEN</td><td>RGB(000,128,000)</td></tr>
<tr><td>3</td><td>KOBALT</td><td>RGB(000,064,255)</td></tr>
<tr><td>2</td><td>MIRTE</td><td>RGB(000,064,000)</td></tr>
<tr><td>1</td><td>BLAUW</td><td>RGB(000,000,255)</td></tr>
<tr><td>0</td><td>ZWART</td><td>RGB(000,000,000)</td></tr>
</table></td></tr>
</table>

COMMANDO	Beschrijving
MATH (WISKUNDE)	**Het wiskundige commando voert veel eenvoudige wiskundige berekeningen uit die in BASIC kan worden geprogrammeerd, maar coderen heeft snelheidsvoordelen lusstructuren in de firmware en er is het voordeel dat zodra debuggen zijn ze er voor iedereen zonder het wiel opnieuw uit te vinden.** Opmerking: 2-dimensionale wiskundematrices worden altijd gespecificeerd **DIM array(n_kolommen, n_rows**) en natuurlijk respecteren de afmetingen **OPTION BASE**. Quaternions worden opgeslagen als een **array** met **5 elementen w, x, y, z, magnitude**.
MATH RANDOMIZE [n]	**Zaadt het Mersenne Twister-algoritme.** Als **n** niet is opgegeven, is het zaad de tijd in microseconden sinds het opstarten. Het Mersenne Twister-algoritme geeft een veel beter willekeurig getal dan de ingebouwde functie in de **C-bibliotheek.**
MATH(RAND)	**Genereert een willekeurig getal 0.0 >= n < 1.0 met behulp van de "Mersenne Twister:** A 623-dimensionaal * Equidistributed uniforme generator van pseudowillekeurige getallen. (Alleen **RP2040**)
	Eenvoudige array-berekeningen
MATH SET nbr, array()	**Stelt alle elementen in array() in op de waarde nbr.** Let op: dit is de **snelste manier** om een **array wissen** door deze **op nul te zetten**.
MATH SCALE in(), scale, out()	**Dit schaalt de matrix in() met de scalaire schaal en plaatst het antwoord in out().** Werkt voor arrays van elke dimensionaliteit van zowel integer als float en can tussen omzetten. Het instellen van b op 1 is geoptimaliseerd en is de snelste manier om het kopiëren van een hele array.
MATH ADD in(), num, out()	**Dit voegt de waarde 'num' toe aan elk element van de matrix in() en plaatst de antwoord in uit().** Werkt voor arrays van elke dimensionaliteit van zowel integer als zweven en kan converteren tussen. Het instellen van **num op 0** is geoptimaliseerd en is een snelle manier om een hele array te kopiëren. **in()** en **out()** kunnen dezelfde array zijn.
MATH INTERPOLATE in1(), in2(), ratio, out()	**Deze opdracht implementeert de volgende vergelijking op elk array element: `out = (in2 - in1) * verhouding + in1`** Arrays kunnen een willekeurig aantal dimensies hebben en moeten verschillend zijn en: hetzelfde aantal totale elementen. De opdracht werkt met bot **integer** en drijvende-komma-arrays in elk mengsel.

COMMANDO	Beschrijving
`MATH WINDOW in(), minout, maxout, out() [,minin, maxin]`	**Deze opdracht neemt de "in"-array en schaalt deze tussen "minout" en "maxout" en retourneert het antwoord in "out".** Optioneel kan het ook de minimum- en maximumwaarden retourneren die in de originele gegevens zijn gevonden. (**"minin"** en **"minout"**) Let op: **"minout"** kan groter zijn dan **"maxout"** en in dit geval worden de gegevens zowel geschaald als omgekeerd. Bijv. `DIM IN(2) = (1, 2, 3)` `DIM OUT(2)` `MATH WINDOW IN(), 7, 3, OUT(), LOW, HIGH` Geeft **OUT(0) = 7, OUT(1) = 5, OUT(2) = 3, LOW = 1, HIGH = 3** terug Deze opdracht kan het schalen van gegevens voor het plotten etc. enorm vereenvoudigen.
`MATH SLICE sourcearray(), [d1] [,d2][,d3][,d4][,d5], destinationarray()`	**Deze opdracht kopieert een gespecificeerde set waarden van een multidimensionaal array in een eendimensionale array.** Het is veel sneller dan het gebruik van een **FOR lus**. Het segment wordt gespecificeerd door een waarde te geven voor **alle bronnen** behalve **één array indexen** en er moeten zoveel indices in de opdracht zijn, inclusief de lege, omdat er dimensies zijn in de bronarray. **bijv.:** `OPTION BASE 1` `DIM a(3, 4, 5)` `DIM b(4)` `MATH SLICE a(), 2, , 3, b()` Zal de elementen **2, 1, 3** en **2, 2, 3** en **2, 3, 3** en **2, 4, 3** kopiëren naar `array b()`
`MATH INSERT targetarray(),[d1] [,d2][,d3][,d4][,d5], sourcearray()`	**Dit is het tegenovergestelde van MATH SLICE, heeft een zeer vergelijkbare syntaxis en staat toe: u,bijvoorbeeld, om een enkele vector in een array van vectoren te vervangen door een enkele instructie bijv.** `OPTION BASE 1` `DIM targetarray (3, 4, 5)` `DIM sourcearray(4) = (1, 2, 3, 4)` `MATH INSERT targetarray(), 2, , 3, sourcearray()` Zal elementen **2,1,3 = 1** en **2,2,3 = 2** en **2,3,3= 3** en **2,4,3 = 4** instellen
`MATH POWER inarray(), power, outarray()`	**Verhoogt elk element in 'inarray()' tot de gedefinieerde 'macht' en plaatst de uitvoer in 'outarray()'**
`MATH SHIFT inarray%(), nbr, outarray%() [,U]`	**Deze opdracht voert een bitverschuiving uit op alle elementen van inarray%() en plaatst het resultaat in outarray%() (kan hetzelfde zijn als inarray%()).** **nbr** kan tussen -63 en 63 liggen. Positieve getallen zijn een verschuiving naar links (vermenigvuldigen met de macht van 2). Negatieve getallen zijn een verschuiving naar rechts. De optionele parameter: **,U** forceert een unsigned shift.

COMMANDO	Beschrijving
	Matrixberekening
`MATH M_INVERSE array!(), inversearray!()`	**Dit retourneert de inverse van array!() in inversearray!().** De array moet zijn vierkant en je krijgt een **foutmelding** als de array niet kan worden omgekeerd (**determinant = 0**). **array!()** en **inversearray!()** kunnen niet hetzelfde zijn.
`MATH M_PRINT array()`	**Snel mechanisme om een 2D-matrix één rij per regel af te drukken.**
`MATH M_TRANSPOSE in(), out()`	**Transponeer matrix in() en plaats het antwoord in matrix out(), beide arrays moeten 2D maar hoeft niet vierkant te zijn. Als ze niet vierkant zijn, moeten de arrays zijn gedimensioneerd:** `in(m,n), out(n,m)`
`MATH M_MULT in1(), in2(), out()`	**Vermenigvuldig de arrays in1() en in2() en zet het antwoord in out()c**. Alle **arrays** moet **2D** zijn, maar hoeft niet vierkant te zijn. Als ze niet vierkant zijn, moeten de **arrays** zijn gedimensioneerd `in1(m,n), in2(p,m), out(p,n)`
	Vector rekenkunde
`MATH V_PRINT array() [,hex]`	**Snel mechanisme om een kleine array op een enkele regel af te drukken. 'hex' will print in hex.**
`MATH V_NORMALISE inV(), outV()`	**Converteert een vector inV() naar eenheidsschaal en plaatst het antwoord in outV(). (sqr(x * x + y * y + ...) = 1** Er is geen limiet op het aantal elementen in de vector.
`MATH V_MULT matrix(), inV(), outV()`	**Vermenigvuldigt matrix() en vector inV() en geeft vector outV() terug.** De **vectoren** en **de 2D-matrix** kan elke grootte hebben, maar moet dezelfde kardinaliteit hebben.
`MATH V_CROSS inV1(), inV2(), outV()`	**Berekent het uitwendige product van twee vectoren met drie elementen inV1() en inV2() en zet het antwoord in outV()**.
`MATH V_ROTATE x, y, a, xin(), yin(), xout(), yout()`	**Deze opdracht roteert de coördinatenparen in "xin()" en "yin()" rond het middelpunt gedefinieerd door "x" en "y" door de hoek "a" en plaatst de resultaten in "xout()" en "yout()".** NB: de invoer- en uitvoerarrays kunnen hetzelfde zijn en de rotatie-hoek is standaard in radialen, maar dit kan worden gewijzigd met de opdracht: `OPTION ANGLE`.
	Quaternion rekenkunde
`MATH Q_INVERT inQ(), outQ()`	**Keer het quaternion om in inQ() en zet het antwoord in outQ().**
`MATH Q_VECTOR x, y, z, outVQ()`	**Converteert een vector gespecificeerd door x , y en z naar een ge-normaliseerde quaternionvector outVQ() met de oorspronkelijke grootte opgeslagen.**
`MATH Q_CREATE theta, x, y, z, outRQ()`	**Genereert een genormaliseerd rotatie-quaternion outRQ() om quaternion te roteren vectoren rond as x,y,z door een hoek van theta**. **Theta** wordt uitgedrukt in **radialen.**

COMMANDO	Beschrijving
`MATH Q_EULER yaw, pitch, roll, outRQ()`	**Genereert een genormaliseerd rotatie-quaternion outRQ() om quaternion te roteren vectoren zoals gedefinieerd door de yaw-, pitch- en roll-hoeken.** Met de vector voor de **"kijker"** kijkt **gieren** vanaf de **bovenkant** van de **vector** en draait met de klok mee, pitch draait de bovenkant weg van **de camera** en rol draait met de klok mee rond de **z-as**. De **yaw-, pitch-** en **roll-hoeken** zijn **standaard radialen**, maar respecteren de instelling van **OPTION ANGLE**.
`MATH Q_MULT inQ1(), inQ2(), outQ()`	**Vermenigvuldigt twee quaternionen inQ1() en inQ2() en zet het antwoord in outQ().**
`MATH Q_ROTATE RQ(), inVQ(), outVQ()`	**Roteert de bron-quaternionvector inVQ() met het rotatiequaternion RQ() en zet het antwoord in outVQ().**
`MATH C_ADD array1%(), array2%(), array3%()` `MATH C_SUB array1%(), array2%(), array3%()` `MATH C_MUL array1%(), array2%(), array3%()` `MATH C_DIV array1%(), array2%(), array3%()` `MATH C_ADD array1!(), array2!(), array3!()` `MATH C_SUB array1!(), array2!(), array3!()` `MATH C_MUL array1!(), array2!(), array3!()` `MATH C_DIV array1!(), array2!(), array3!()`	**Deze opdrachten voeren cel-voor-cel-bewerkingen uit (vandaar C_) op arrays van identieke grootte.** Er zijn geen beperkingen op het aantal dimensies en geen beperkingen op het tweemaal of zelfs driemaal gebruiken van dezelfde array in de parameters. Het gegevenstype moet voor alle arrays hetzelfde zijn. `MATH C_MULT a%(),a%(),a%()` zal bijvoorbeeld alle waarden in de **array a%()** kwadrateren. `MATH C_ADD array1%(), array2%(), array3%()` `MATH C_SUB array1%(), array2%(), array3%()` `MATH C_MUL array1%(), array2%(), array3%()` `MATH C_DIV array1%(), array2%(), array3%()` `MATH C_ADD array1!(), array2!(), array3!()` `MATH C_SUB array1!(), array2!(), array3!()` `MATH C_MUL array1!(), array2!(), array3!()` `MATH C_DIV array1!(), array2!(), array3!()`
`MATH FFT signalarray!(), FFTarray!()`	**Voert een snelle Fourier-transformatie uit van de gegevens in "signalarray!".** **"signaalarray"** moet een drijvende komma zijn en de grootte moet een macht van **2** zijn. (bijv. **s(1023)** ervan uitgaande dat **OPTION BASE** nul is). **"FFArray"** moet een drijvende komma zijn en afmeting **2 * N** hebben, waarbij **N** is hetzelfde als de signaalarray. (bijv. **f(1.1023)** ervan uitgaande dat **OPTION BASE** nul is). De opdracht retourneert de **FFT** als complexe getallen met het reële deel in **f(0,n)** en het imaginaire deel in **f(1,n)**.
`MATH FFT INVERSE FFTarray!(), signalarray!()`	**Voert een inverse snelle Fourier-transformatie uit van de gegevens in "FFTarray!".** **"FFTarray"** moet een drijvende komma zijn en afmeting **2 * N** hebben, waarbij **N** moet een macht van **2** zijn (bijv. **f(1.1023)** ervan uitgaande dat **OPTION BASE nul** is) met de **reële dee**l in **f(0,n)** en het imaginaire deel in **f(1,n)**.**"signalarray"** moet een drijvende komma zijn en de enkele dimen-sie moet hetzelfde als de **FFT-array** zijn. De opdracht retourneert het reële deel van de inverse transformatie in **"signaalarray"**.

COMMANDO	Beschrijving
`MATH FFT MAGNITUDE signalarray!(), magnitudearray!()`	**Genereert grootheden voor frequenties voor de gegevens in "signaalarray!"**. **"signalarray"** moet een drijvende komma zijn en de grootte moet een **macht van 2** zijn. (bijv. **s(1023)** ervan uitgaande dat **OPTION BASE nul** is). **"magnitudearray"** moet een drijvende komma zijn en de grootte moet hetzelfde zijn als designaal array. De opdracht retourneert de grootte van het signaal op verschillende frequenties volgens de formule: frequentie op arraypositie **N = N * sample_frequency/number_of_ samples.**
`MATH FFT PHASE signalarray!(), phasearray!()`	**Genereert fasen voor frequenties voor de gegevens in : "signaalarray!".** **"signalarray"** moet een drijvende komma zijn en de grootte moet een **macht van 2** zijn. (bijv. **s(1023)** ervan uitgaande dat **OPTION BASE** nul is). **"phasearray"** moet een drijvendekomma zijn en de grootte moet hetzelfde zijn als de **signaal array**. **De opdracht** retourneert de **fasehoek** van het signaal op verschillende frequenties volgens bovenstaande formule.
`MATH SENSORFUSION type ax, ay, az, gx, gy, gz, mx, my, mz, pitch, roll, yaw [,p1] [,p2]`	**Type kan MAHONY of MADGWICK zijn:** **Ax, ay en az zijn: De versnellingen in de drie richtingen en zouden moeten zijn gespecificeerd in eenheden van standaard zwaarte-krachtversnelling.** **Gx, gy** en **gz** zijn: De **momentane** waarden van de **rotatiesnelheid** die zouden moeten worden opgegeven in **radialen per seconde**. **Mx, my** en **mz** zijn: De **magnetische velden** in de **drie** richtingen en zouden moeten zijn gespecificeerd in **nano-Tesla (nT)**. Er moet voor worden gezorgd dat de **x-, y-** en **z**-componenten consistent zijn tussen de drie ingangen. Dus, bijvoorbeeld, gebruik de MPU-9250 de juiste invoer zal zijn **ax, ay,az, gx, gy, gz, my, mx, -mz** gebaseerd op de lezing van de voeler. **Pitch, roll** en **yaw** moeten drijvendekommavariabelen zijn en bevatten de uitgangen van de sensorfusie. De **SENSORFUSIE** routine meet automatisch de tijd tussen opeenvolgende gesprekken en zal dit gebruiken in zijn interne berekeningen. Het **Madwick**-algoritme neemt een optionele **parameter p1**. Dit wordt gebruikt als **bèta** bij de berekening. Het is standaard **0.5** indien niet gespecificeerd. Het **Mahony**-algoritme heeft twee optionele **parameters p1 en p2**. Dit zijn gebruikt als **Kp** en **Ki** in de berekening. Indien niet gespecificeerd, zijn deze standaard ingesteld op **10.0** en **0.0** respectievelijk. Een volledig uitgewerkt voorbeeld van het gebruik van de code wordt gegeven op het **BackShed-forum**. Bij: https://www.thebackshed.com/forum/ViewTopic.php?TID=13459&PID=166962 #166962

COMMANDO	Beschrijving
`MATH PID INIT channel, pid_params!(),callback`	**Met deze opdracht wordt een PID-regelaar ingesteld die automatisch op de achtergrond kan werken.** **Er kunnen maximaal 8 PID-regelaars gelijktijdig werken (kanalen 1 tot 8).** **'callback'** is een MMbasic-subroutine die wordt aangeroepen met de snelheid die wordt bepaald door de bemonsteringstijd. Zie de functie **MATH(PID ...)** voor details over wat er in de subroutine moet worden opgenomen. De **pid_params!() array** moet gedimensioneerd zijn voor alle vermelde elementen, inclusief de geheugenparameters van de controller (**DIM pid_params!(13)**) en geïnitialiseerd worden met de vereiste instellingen. **PID configuratie** **Element 0 = Kp** **Element 1 = Ki** **Element 2 = Kd** **Element 3 = tau 'Afgeleide tijdconstante van het laagdoorlaatfilter** **Element 4 = limMin ' Uitgangslimieten** **Element 5 = limMax** **Element 6 = limMinInt ' Integratorlimieten** **Element 7 = limMaxInt** **Element 8 = T ' Bemonsteringstijd (in seconden)** **Controller "geheugen"** **Element 9 = integrator** **Element 10 = vorigeError** **Element 11 = differentiator** **Element 12 = vorigeMeting** **Element 13 = uit**
`MATH PID START channel`	**Start een eerder geïnitialiseerde PID-regelaar op het opgegeven kanaal.**
`MATH PID STOP channel`	**Stopt een eerder geïnitialiseerde PID-regelaar op het opgegeven kanaal en verwijdert de interne datastructuren.** Zie https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=17263 Voor een voorbeeld van het instellen en gebruiken van een PID-regelaar.
`MATH AES128 ENCRYPT/DECRYPT CBC/ECB/CTR key$/key(),in$/in(), out$/out()[,iv$/iv()]`	**Deze opdracht codeert of decodeert de gegevens in 'in' en plaatst het antwoord in 'out' met behulp van de opgegeven AES128-coderingsmethode.** De parameters kunnen elk een string, een integer-array of een float-array met elk zijn mix mogelijk. De sleutel moet **16** elementen lang zijn (**16 * 8 = 128 bits**), in en out moeten een veelvoud zijn van **16** elementen lang. In het geval dat out wordt gespecificeerd als een string (bijv. **out$**), moet de stringvariabele bestaan en ingesteld zijn op leeg (**DIM out$ = ""**). Het maximale aantal elementen in **'in'** en **'out'** wordt beperkt door het geheugen wanneer gedefinieerd als arrays.

	Tekenreeksen voor codering zijn beperkt tot **240 bytes (EBR)** en **224 bytes (CTR** en **CBC)**. Voor **CBC-** en **CTR-codering** kunt u optio-neel een initialisatie op-geven vector **'iv'**. **'iv'** moet **16 elementen lang zijn** (**16*8=128bits**). Als een initialisatie vector is niet gespeci-ficeerd encryptie genereert een willekeurige initialisatievector. In beide gevallen wordt de uitvoer voorafgegaan door de **IV.** Voor **CBC** en **CTR** vereist decodering dat de eerste 16 elementen van de invoer worden ingevoerd zijn de initialisatievector. In het geval dat u een bericht wilt verzenden zonder **IV**, moet u dat doen verwijder de IV voordat u het bericht verzendt met standaard MMBasic manipulaties. In dit geval moet de ontvanger zowel de IV als de sleutel kennen en maak een compleet bericht voordat u **DECRYPT** gebruikt door de IV naar het binnenkomende bericht.
`MEMORY`	**Lijst met de hoeveelheid geheugen die momenteel in gebruik is.** Bijvoorbeeld: `Program:` `0K (0%) Program (0 lines)` `128K (100%) Free  ' RP2350 180K (100%) Free` `Saved Variables:` `16K (100%) Free` `RAM:` `0K ( 0%) 0 Variables` `0K ( 0%) General` `156K (100%) Free  ' RP2350 228K (100%) Free` Opmerkingen: • Het geheugengebruik wordt afgerond op de dichtstbijzijnde 1K byte en kan per versie in omvang verschillen. • Algemeen geheugen wordt gebruikt door seriële I/O-buffers, enz.
`MEMORY SET address, byte, numberofbytes`	**Met deze opdracht wordt een geheugengebied ingesteld op een waarde.** `BYTE` = Eén byte per geheugenadres. `KORT` = Twee bytes per geheugenadres.
`MEMORY SET BYTE address, byte, numberofbytes`	`BYTE` = Eén byte per geheugenadres. `WORD` = Vier bytes per geheugenadres. `FLOAT`= Acht bytes per geheugenadres. **'increment'** is optioneel en regelt de toename van de **'adres'**-aanwijzer als de operatie wordt uitgevoerd. Als bijvoorbeeld staps-gewijs = **3**, dan slechts elke derde onderdeel van het doel is ingesteld De standaardwaarde is **1**
`MEMORY SET SHORT address, short, numberofshorts`	`WORD` = Vier bytes per geheugenadres.
`MEMORY SET INTEGER address, integervalue, numberofintegers [,increment]`	**'increment' is optioneel en regelt de toename van de 'adres' aan-wijzer als de operatie wordt uitgevoerd.** Bijvoorbeeld: `If increment = 3 Then` alleen elke derde element van het doel is ingesteld. De standaardwaarde is **1.**

COMMANDO	Beschrijving 216
MEMORY SET FLOAT address, floatingvalue, numberoffloats [,increment] MEMORY SET WORD address, word, numberofwords	FLOAT = Acht bytes per geheugenadres. WORD = Vier bytes per geheugenadres.
MEMORY COPY sourceaddress, destinationaddress, numberofbytes [,sourceincrement] [,destination increment]	**Met deze opdracht wordt het ene geheugengebied naar het andere gekopieerd.** COPY INTEGER en FLOAT kopiëren **acht bytes** per bewerking. **'sourceincrement'** is optioneel en regelt de increment van de **'sourceaddress'** aanwijzer terwijl de bewerking wordt uitgevoerd.
MEMORY COPY INTEGER sourceaddress, destinationaddress, numberofintegers [,sourceincrement] [,destination increment] of MEMORY COPY FLOAT bron adres, bestemmingsadres, aantal gehele getallen [,bronincrement] [,bestemmingin verhoging]	**Bijvoorbeeld, if sourceincrement = 3, then is alleen elk derde element van de bron gekopieerd.** De standaardwaarde is **1**. **'destinationincrement'** is vergelijkbaar en werkt op het **'destinationaddress'** wijzer.
MEMORY PRINT #]fnbr, nbr, address%/ array() MEMORY INPUT[#]fnbr, nbr, address% |array()	**Met deze opdrachten wordt een 'aantal' databytes opgeslagen of gelezen van of naar het geheugen van of naar een geopend schijfbestand.** **Het op te slaan geheugen kan worden gespecificeerd als een integer-array. In dat geval wordt het aantal bytes dat moet worden opgeslagen of gelezen, vergeleken met de arraygrootte.** Als alternatief kan een geheugenadres worden gebruikt. In dat geval kan er geen controle plaatsvinden en kunnen gebruikersfouten leiden tot een crash van de firmware.
MEMORY PACK source%() | sourceaddress%, dest%() | destaddress% , number, size	**Met Memory Pack en Unpack kunnen gehele waarden uit één array worden gebruikt gecomprimeerd in een ander of ongecom-primeerd van de een naar de ander.** De twee arrays zijn altijd normale **integer-arrays**, maar de **ingepakte array** kan **2**, **4**, **8**, **16** of **64** waarden bevatten. Een enkel integer array-element zou dus 2 woorden van 32 bits, 4 waarden van 16 bits, 8 bytes, 16 nibbles of 64 booleans (bits) kunnen opslaan.

COMMANDO	Beschrijving
`MEMORY UNPACK source() | sourceaddress%, dest() | destaddress%, number, size`	**"aantal" specificeert het aantal waarden dat moet worden inge-pakt of uitgepakt en "grootte" specificeert het aantal bits (1,4,8,16 of 32)** Als alternatief kunnen geheugenadressen worden gebruikt. In dat geval kan er geen controle plaatsvinden en kunnen gebruikers-fouten resulteren in een crash van de firmware.
`MKDIR dir$` `MKDIR "dir$"` `CHDIR "dir$"` `NAME "dir$"` `RMDIR "dir$"`	**Maak of creëer de directory 'dir$' op de SD-kaart.** **MKDIR, CHDIR** en **RMDIR** nu voor **FLASH.** De sleutelcode is geïmporteerd uit de CMM2 die directorytracking in mijn firmware moet doen vanwege beperkingen in **FATFS** bij ge-bruik van **exFAT.** b.v. `CHDIR "/peter/test" LOAD "../fred/myapp"` `DRIVE "A:"` `MKDIR "flashdir" 'Maak dir. Let op dat er geen A: is` `PLAY WAV "B:/wavfiles/wavsample"` `COPY "B:/mydir/myprog.bas" To "a:/flashdir/myprog.bas"` `DRIVE "B:"` `FILES "A:/flashdir/"` `OPEN "A:/testfile.dat" For Input As #1` `OPEN "B:/anotherfile.dat" For Output As #2` `a$ = Input$(100, 1)` `Print #2, a$` `Close #1` `Close #2` b.v. `Drive "a:" of a: 'Ga naar drive a:` `Kill "test.bas" 'Wis test.bas` `MKdir "backup-rp2350" 'Maak een dir backup-rp2350` `Chdir "peter" 'Ga naar dir peter` `Drive "b:" of b: 'Ga naar drive b:` `Chdir "all" 'Ga naar dir all` `Copy "file.dat2" To "newfile.dat" 'Naam veranderen` `Copy "a:/*.*" to "b:/backup-rp2350/"'Backup maken.` `Copy "b:/backup-rp2350/*.* to "a:/"'Zet Backup terug` `Drive "A:/format" 'Wis drive A:/ Eerst een backup!`
`MID$( str$, start [, num]) = str2$`	**De tekens in 'str$', beginnend op positie 'start', worden vervangen door de tekens in 'str2$'.** Het optionele **'num'** verwijst naar het aantal karakters van **'str2'** die worden gebruikt in de vervanging. Als **'num'** wordt weggelaten, is alles van **'str2'** gebruikt. Of **'num'** nu wordt weggelaten of opgenomen, de vervanging van tekens gaat nooit verder dan de oorspronkelijke lengte van **'str$'.**
`MODE 1` of `MODE 2` of `MODE 3 (RP2350 only)`	**ALLEEN VGA-VERSIE** **Schakelt tussen 640 x 480 zwart-wit (MODE 1) en 320 x 240 4-bit kleur (MODE 2) en 640 x 480 4-bit** kleur (**MODE 3 –** alleen **RP2350**). Bij het opnieuw opstarten wordt de modus gereset naar de modus die is gedefinieerd met de opdracht **OPTION DEFAULT MODE**.

	MODE 1 **640 x 480 x 2 kleuren** (**monochroom**). Standaard bij opstarten. De breedte van de tegels is vastgesteld op **8 pixels.** De tegelhoogte is standaard ingesteld op **12 pixels,** maar kan variëren van **8** tot **MM.HRES**. Tegelkleuren zijn gespecificeerd met behulp van de standaard **RGB888** notatie. Dit is omgezet naar **RGB121**. Een **framebuffer** (**F**) en een **laagbuffer (L)** kan worden aangemaakt. Deze hebben geen invloed op de kan worden gebruikt voor het maken van afbeeldingen en het kopiëren naar het weergavescherm **(N)**
	MODE 2 **320 x 240 x 16 kleuren.** **RGB121** formaat (d.w.z. **1 bit voor rood**, **2 bits voor groen** en **1 bit voor blauw**). Er kan een **framebuffer (F)** worden gemaakt. Dit heeft geen invloed op het display en gebruikt geen gebruikersgeheugen maar kan worden gebruikt voor het maken van afbeeldingen en het kopiëren naar het weergavescherm **(N)**. Daarnaast kan er een laagbuffer worden gemaakt. Ook hier wordt geen gebruikersgeheugen gebruikt. De eventuele pixels geschreven naar de laagbuffer verschijnt automatisch op het display zit bovenop wat er ook maar in de hoofdrol zit weergavebuffer. Er kan een kleur worden opgegeven. (**0-15**: standaard ingesteld op **0**), wat niet aangeeft dat de hoofddisplaybuffer om door te laten zien. Kaartfunctionaliteit is beschikbaar om de standaardkleuren van de **16** te overschrijven beschikbaar In het geval van **VGA** is de hardware beperkt tot de **16 kleuren** gedefinieerd door het weerstandsnetwerk.
	MODE 3 **640 x 480 x 16 kleuren.** (RP2350) **RGB121** formaat (d.w.z. **1 bit voor rood**, **2 bits voor groen** en **1 bit voor blauw**). Er kan een **framebuffer (F)** worden gemaakt. Dit heeft geen invloed op het display en gebruikt geen gebruikersgeheugen maar kan worden gebruikt voor het maken van afbeeldingen en het kopiëren naar het weergavescherm **(N)**. Daarnaast kan er een laagbuffer worden gemaakt. Ook hier wordt geen gebruikersgeheugen gebruikt. De eventuele pixels geschreven naar de laagbuffer verschijnt automatisch op het display zit bovenop wat er ook maar in de hoofdrol zit weergavebuffer. Er kan een kleur worden opgegeven (**0-15**: standaard ingesteld op **0**), wat niet aangeeft dat de hoofddisplaybuffer om door te laten zien. Kaartfunctionaliteit is beschikbaar om de standaardkleuren van de **16** te overschrijven beschikbaar. In het geval van **VGA** is de hardware beperkt tot de **16 kleuren** gedefinieerd door het weerstandsnetwerk.

COMMANDO	Beschrijving 219
MODE n	**ALLEEN HDMI-VERSIES** **HDMI-video ondersteunt een aantal resoluties.** **(zie OPTION RESOLUTION)** Met dit commando wordt de modus **'n'** geselecteerd, afhankelijk van de resolutie: **OPTION RESOLUTION 640 x 480 CPUSPEED: 283200 of 315000 KHz** **MODE 1** **640 x 480 x 2 kleuren (monochroom).** Standaard bij opstarten. Gebruik de opdracht **TILE** zoals normaal. De tegelbreedte is vastgesteld op **8 pixels**. De tegelhoogte is standaard **12 pixels**, maar kan variëren van **8** tot **MM.HRES**. Tegelkleuren worden gespecificeerd volgens de standaard **RGB888**-notatie. Dit wordt omgezet naar **RGB555**. Een **framebuffer (F)** en een **laagbuffer (L)** kunnen worden gemaakt. Deze kan worden gebruikt voor het maken van afbeeldingen en het kopiëren naar het **display scherm (N)** Niet in **Option List** i.v.m standaard: **Option Display 40, 80** **MODE 2** **320 x 240 x 16 kleuren.** Er kan een **framebuffer (F)** worden gemaakt. Dit kan worden gebruikt voor het maken beelden en kopiëren naar het **weergavescherm (N)**. Daarnaast een laagbuffer kan worden gemaakt. Alle pixels die naar de laag zijn geschreven buffer verschijnt automatisch op het display bovenaan van wat zich ook in de hoofddisplaybuffer bevindt. Een kleur kan worden opgegeven (**0-15:** standaard ingesteld op **0**), wat niet wordt weergegeven waardoor de hoofddisplaybuffer zichtbaar wordt. Kaart functionaliteit is beschikbaar om de standaardkleuren te overschrijven. Niet in **Option List** i.v.m standaard: **Option Display 30, 53** **MODE 3** **640 x 480 x 16 kleuren.** Kleurtoewijzing naar **RGB555** palet. Een **framebuffer (F)** kan dat wel zijn gemaakt. Het kan worden gebruikt voor het maken van afbeeldingen en het kopiëren naar het **weergavescherm (N)**. Daarnaast kan er een laagbuffer worden gemaakt. Alle pixels die naar de laagbuffer worden geschreven, worden automatisch weergegeven verschijnen op het display, bovenop wat er zich ook in bevindt hoofddisplaybuffer. Er kan een kleur worden opgegeven (**0-15:** standaardwaarden naar **0**), wat niet aangeeft dat de buffer van het hoofddisplay dit mag doen door het te laten zien. Niet in **Option List** i.v.m standaard: **Option Display 40, 80**

MODE 4 **320 x 240 x 32768 kleuren.**
Dit is volledige **RGB555** waardoor kleurenafbeeldingen van goede kwaliteit mogelijk zijn worden weergegeven. Een **framebuffer** **(F)** en een **laagbuffer (L)** kunnen dat wel worden gecreëerd. Deze hebben geen invloed op het display en kunnen dat wel worden gebruikt voor het maken van afbeeldingen en het kopiëren naar het displayscherm **(N)**. Er kan er maar één worden gemaakt.
Option Display 30, 53

MODE 5 **320 x 240 x 256 kleuren.**
Er kan een **framebuffer (F)** worden gemaakt. Dit heeft geen invloed op het scherm. Het kan worden gebruikt voor het maken van afbeeldingen en kopiëren naar het weergavescherm **(N)**. Daarnaast een laag buffer kan worden aangemaakt.
Dit maakt geen gebruik van het gebruikersgeheugen. Alle pixels die naar het laagbuffer verschijnt automatisch op het scherm bovenaan van wat zich ook in de hoofddisplaybuffer bevindt. Een kleur kan worden opgegeven (**0-255:** standaard ingesteld op **0**), wat niet wordt weergegeven waardoor de hoofddisplaybuffer zichtbaar wordt.
De kaart functionaliteit is beschikbaar om de standaardkleuren te overschrijven en er zijn **256 kleuren** beschikbaar. Elk van de **256 kleuren** kan aan elke kleur worden toegewezen **RGB555 kleur**. **Option Display 30, 53**

OPTION RESOLUTION 720 x 400 CPUSPEED: 283200 KHz

MODE 1 **720 x 400 x 2 kleuren met RGB332-tegels (gebruik de TILE commando zoals normaal). Option Display 33, 90**
MODE 2 **320 x 180 x 16 kleuren met 2 optionele lagen en kleur mapping naar RGB332-palet. Option Display 25, 60**
MODE 3 **640 x 360 x 16 kleuren met optionele laag- en kleurtoewijzing naar het RGB332-palet. Option Display 33, 90**
MODE 5 **320 x 180 x 256 kleuren met optionele laag (geen geheugen gebruik) Option Display 25, 60**

OPTION RESOLUTION 800 x 600 CPUSPEED: 360000 KHz

MODE 1 **800 x 600 x 2 kleuren met RGB332-tegels (gebruik de TILE commando zoals normaal). Option Display 50, 100**
MODE 2 **320 x 180 x 16 kleuren met 2 optionele lagen en kleur mapping naar RGB332-palet. Option Display 37, 64**
MODE 3 **640 x 360 x 16 kleuren met optionele laag- en kleurtoewijzing naar het RGB332-palet. Option Display 50, 100**
MODE 5 **320 x 180 x 256 kleuren met optionele laag (geen geheugen gebruik) Option Display 37, 66**

	OPTION RESOLUTION 848 x 480 CPUSPEED: 336000 KHz **MODE 1 848 x 480 x 2 kleuren met RGB332-tegels (gebruik de TILE commando zoals normaal). Option Display 40, 106** **MODE 2 320 x 180 x 16 kleuren met 2 optionele lagen en kleur mapping naar RGB332-palet. Option Display 30, 70** **MODE 3 640 x 360 x 16 kleuren met optionele laag- en kleurtoe-wijzing naar het RGB332-palet. Option Display 40, 106** **MODE 5 320 x 180 x 256 kleuren met optionele laag (geen geheugen gebruik) Option Display 30, 70** **OPTION RESOLUTION 1024 x 768 CPUSPEED: 324000 KHz** **MODE 1 1024 x 768 x 2 kleuren met RGB332-tegels (gebruik de TILE commando zoals normaal). Option Display 32, 64** **MODE 2 256 x 192 x 16 kleuren met 2 optionele lagen en kleur mapping naar RGB332-palet. Option Display 24, 42** **MODE 3 512 x 384 x 16 kleuren met optionele laag- en kleurtoe-wijzing naar het RGB332-palet. Option Display 32, 64** **MODE 5 256 x 192 x 256 kleuren met optionele laag (geen geheugen gebruik). Option Display 24, 42** **OPTION RESOLUTION 1280 x 720 CPUSPEED: 372000 KHz** **MODE 1 1280 x 720 x 2 kleuren met RGB332-tegels (gebruik de TILE commando zoals normaal) Option Display 30, 80** **MODE 2 320 x 180 x 16 kleuren met 2 optionele lagen en kleur mapping naar RGB332-palet. Option Display 22, 53** **MODE 3 640 x 360 x 16 kleuren met optionele laag- en kleurtoe-wijzing naar het RGB332-palet. Option Display 30, 80** **MODE 5 320 x 180 x 256 kleuren met optionele laag (geen geheugen gebruik) Option Display 22, 53**
`MOUSE`	**Voor alle varianten van de opdracht.** In het geval van USB-firmware is **'kanaal'** de USB-poort waarop de muis is aangesloten **(1-4)**. Zie **MM.INFO(USB n)** voor meer informatie. Voor **PS2-firmware** is **'kanaal'** vast ingesteld op de **waarde 2.**
`MOUSE INTERRUPT ENABLE channel, int`	**'int' is een door de gebruiker gedefinieerde subroutine die wordt aangeroepen wanneer de linkermuisknop wordt ingedrukt.**
`MOUSE INTERRUPT DISABLE channel`	**Schakelt een onderbreking met de linkermuisknop uit.**
`MOUSE SET channel, x-coord, y-coord [, wheel-count]`	**Stelt de huidige positie in die wordt geretourneerd door de muis x, y en optioneel wielposities.**

COMMANDO	Beschrijving 222
MOUSE OPEN channel, CLKpin, DATApin	**NIET-USB-VERSIES - ALLEEN VOOR EEN PS2 MUIS** **Opent een verbinding met een PS2-muis die is aangesloten op de twee opgegeven pinnen.** Deze opdracht kan in een programma worden gebruikt om de muis te configureren terwijl het programma actief is, in tegenstelling tot OPTION MOUSE die de muis permanent configureert. Kanaal is inbegrepen voor compatibiliteit met **USB-muis** functionaliteit en moet worden ingesteld op 2. Als er geen muis is aangesloten, krijgt u een foutmelding en kan de opdracht opnieuw worden opgeroepen zodra de muis is aangesloten.
MOUSE CLOSE channel	**Sluit de toegang tot de muis af en herstelt de pinnen voor normaal gebruik.** De opdracht geeft een foutmelding als OPTION MOUSE is ingesteld.
NEW	**Wist het programma in RAM en alle variabelen inclusief opgeslagen variabelen.** Deze opdracht wist ook de reservekopie van het programma dat wordt bewaard in Flash-geheugen.
NEXT [tellervariabele] [,tellervariabele], enz	NEXT **komt aan het einde van een** FOR - NEXT **lus. zie** FOR**.** De **'counter-variable'** geeft precies aan op welke lus wordt geopereerd. Als er is geen **'tellervariabele'** opgegeven, de NEXT zal standaard de binnenste lus zijn. Het is ook mogelijk om meerdere tellervariabelen te specificeren zoals in: NEXT x, y, z
ON ERROR ABORT of	**Dit regelt de actie die wordt ondernomen als er een fout optreedt tijdens het uitvoeren van een programma en is van toepassing op alle fouten die door MMBasic zijn ontdekt, inclusief syntaxisfouten, foutgegevens, ontbrekende hardware, enz.** ON ERROR **ABORT zal ervoor zorgen dat MMBasic een foutmelding weergeeft, abort het programma en keer terug naar de opdrachtprompt.** Dit is het normale gedrag en is de standaardinstelling wanneer een programma begint te lopen.
ON ERROR IGNORE of	ON ERROR **IGNORE zorgt ervoor dat elke fout wordt genegeerd.** ON ERROR **IGNORE** kan het erg moeilijk maken om een programma te debuggen, dus het is sterk aanbevolen om alleen ON ERROR **SKIP** te gebruiken.
ON ERROR SKIP [nn] of	ON ERROR **SKIP negeert een fout in een aantal commando's (opgegeven door het nummer 'nn') uitgevoerd na dit commando.** **'nn'** is optioneel, de standaard indien niet gespecificeerd is één. Nadat het aantal opdrachten is voltooid (met een fout of niet) het gedrag van **MMBasic** zal terugkeren naar ON ERROR **ABORT.** Als er een fout optreedt en wordt genegeerd/overgeslagen, wordt de alleen-lezen variabele MM.ERRNO wordt ingesteld op niet nul en MM.ERRMSG$ wordt ingesteld op de foutmelding die normaal zou worden gegenereerd.

COMMANDO	Beschrijving
ON ERROR CLEAR	**Deze worden teruggezet naar nul en een lege string door ON ERROR CLEAR.** Ze worden ook gewist wanneer het programma wordt uitgevoerd en wanneer **ON ERROR IGNORE** en **ON ERROR SKIP** worden gebruikt.
ON KEY target of **ON KEY ASCIIcode, target**	**De eerste versie van de opdracht stelt een interrupt in die de 'doel' gebruiker zal aanroepen gedefinieerde subroutine wanneer er een of meer tekens wachten in de seriële console-invoerbuffer.** Merk op dat alle tekens die in de invoerbuffer wachten, gelezen moeten worden in de subroutine onderbreken, anders wordt er automatisch een andere onderbreking uitgevoerd gegenereerd zodra het programma terugkeert van de interrupt. Met de tweede versie kunt u een interruptroutine koppelen aan een specifieke toets indrukken. Dit werkt op een laag niveau voor de seriële console en indien geactiveerd de sleutel wordt niet in de invoerbuffer geplaatst, maar activeert alleen de interrupt. Het gebruikt een aparte interrupt van de eenvoudige **ON KEY** opdracht, dus kan worden gebruikt indien nodig tegelijkertijd. In beide varianten, om de onderbreking uit te schakelen, gebruikt u de numerieke nul voor het doel, d.w.z.: **ON KEY 0.** of **ON KEY ASCIIcode, 0**
ON PS2 target	**Dit activeert een interrupt wanneer de PicoMite-firmware een bericht van de PS2-interface ziet.** Gebruik **MM.info(PS2)** om het ontvangen onbewerkte bericht te rapporteren. Hierdoor kan de programmeur zowel het indrukken als het loslaten van een toets onderscheppen. Zie https://wiki.osdev.org/PS/2_Keyboard voor de scancodes (Set 2).
1-Wire	**Commando's voor communicatie met 1-Wire-apparaten.**
ONEWIRE RESET pin	**ONEWIRE RESET reset de 1-Wire bus.**
ONEWIRE WRITE pin, flag, lengte, data [, data ...]	**ONEWIRE WRITE stuurt een aantal bytes.**
ONEWIRE READ pin, vlag, lengte, data [, data ...]	**ONEWIRE READ leest een aantal bytes.** **'pin'** is de I/O-pin (in de achterste connector) die moet worden gebruikt. Het kan elke pin zijn geschikt voor digitale I/O. **'vlag'** is een combinatie van de volgende opties: **1** - Stuur reset voor commando. **2** - Stuur reset na opdracht. **4** - Alleen een bit verzenden/ontvangen in plaats van een byte aan gegevens. **8** - Roep een sterke pull-up op na het commando (de pin wordt hoog gezet en open afvoer uitgeschakeld). **'lengte'** is de lengte van de gegevens die moeten worden verzonden of ontvangen. **'data'** zijn de gegevens die moeten worden verzonden of de variabele die moet worden ontvangen. Het aantal gegevensitems moet overeenkomen met de lengteparameter. Zie ook Bijlage C.

COMMANDO	Beschrijving 224
`OPEN fname$ FOR MODE AS [#]fnbr`	**Opent een bestand om te lezen of te schrijven.** **'fname'** is de bestandsnaam met een optionele extensie gescheiden door een **punt (.).** Lange bestandsnamen met hoofdletters en kleine letters worden ondersteund. Een directorypad kan worden opgegeven met de backslash als directoryscheidingsteken. De ouder van de huidige map kan worden opgegeven met behulp van een map naam van **..** (twee puntjes) en de huidige map met **.** (een enkele stip). Als voorbeeld: `OPEN "..\dir1\dir2\filename.txt" FOR INPUT AS #1` **'modus'** is **INPUT**, **OUTPUT**, **APPEND** of **RANDOM**. **INPUT** zal het bestand openen om te lezen en een foutmelding geven als het bestand dat niet doet bestaan. **OUTPUT** zal het bestand openen om te schrijven en zal automatisch een bestaand bestand met dezelfde naam overschrijven. **APPEND** zal het bestand ook openen om te schrijven, maar het zal een bestand niet overschrijven bestaand bestand, in plaats daarvan worden alle schrijfbewerkingen aan het einde van het bestand toegevoegd. Als er is geen bestaand bestand de **APPEND modus** zal hetzelfde werken als de **OUTPUT modus** (d.w.z. het bestand wordt gemaakt en vervolgens geopend om te schrijven). **RANDOM** zal het bestand openen voor zowel lezen als schrijven en zal willekeurig toestaan toegang met behulp van het **SEEK** commando In geopende toestand is de lees-/schrijfaanwijzer, aan het einde van het bestand geplaatst. **'fnbr'** is het dossiernummer (**1 t/m 10**). De **#** is optioneel. Er kunnen maximaal **10** bestanden tegelijk worden geopenend en kan alleen of beide drives **A:** en **B:** openen. De **INPUT**, **LINE INPUT**, **PRINT**, **WRITE** en **CLOSE** commando's en de functies **EOF()** en **INPUT$()** gebruiken allemaal **'fnbr'** om het be-stand te identificeren waarop wordt geopereerd. Zie ook **ON ERROR** en **MM.ERRNO** voor foutafhandeling.
`OPEN comspec$ AS [#]fnbr`	**Opent een seriële communicatiepoort voor lezen en schrijven.** **Twee poorten zijn beschikbaar (COM1: en COM2:) en beide kunnen tegelijkertijd open zijn.** v.b `Open "com2: 38400, inv" As #2` Voor een volledige beschrijving met voorbeelden **zie *bijlage A***. Met behulp van **'fnbr'** kan naar de poort worden geschreven en gelezen met elk commando of functie die een bestandsnummer gebruikt. **'fnbr'** is het dossier nummer (**1** t/m **10**). De **#** is optioneel. Er kunnen maximaal **10** bestanden tegelijkertijd worden geopenend en kunnen op een of beide schijven **A:** en **B:** staan.

COMMANDO	Beschrijving
`OPEN compspec$ AS GPS [,timezone_offset] [,monitor]`	**Opent een seriële communicatiepoort voor het lezen van een GPS-ontvanger**. Zien de **GPS-functie** voor details. De geïnterpreteerde zinnen zijn **GPRMC, GNRMC, GPCGA** en **GNCGA**. De parameter **timezone_offset** wordt gebruikt om **UTC** te **converteren** zoals **ontvangen** van de **GPS** naar de lokale tijdzone. Indien weggelaten, wordt de tijdzone standaard ingesteld op UTC. De **timezone_ offset** kan een willekeurig getal tussen **-12** en **14** zijn, de tijd toestaan correct worden ingesteld, zelfs voor de Chatham-eilanden in Nieuw-Zeeland (**UTC +12:45**). Als de monitorparameter is ingesteld op **1**, wordt alle GPS-invoer naar de console gestuurd. Dit kan worden gestopt door het GPS-kanaal te sluiten.
`OPTION`	**Zie de sectie Opties eerder in deze handleiding.**
`PAUSE delay`	**Stop de uitvoering van het lopende programma voor 'vertraging'ms** Dit kan een zijn fractie. **0.2** is bijvoorbeeld gelijk aan **200 µs**. De **maximale vertraging** is **2147483647 ms** (**ongeveer 24 dagen**). Houd er rekening mee dat onderbrekingen tijdens een pauze worden herkend en verwerkt.
`PIN(pin) = value`	**Voor een 'pin' geconfigureerd als digitale uitgang zal dit de uitgang op laag zetten ('waarde' is nul) of hoog ('waarde' niet nul).** U kunt een output hoog of laag voordat het wordt geconfigureerd als een uitgang en die instelling wordt de standaard uitvoer wanneer het `SETPIN` commando van kracht wordt. Zie de **function** `PIN( )` voor het **lezen** van **een pin** en het commando `SETPIN` voor het configureren. Raadpleeg het hoofdstuk **"Gebruik van de I/O-pinnen"** voor een algemene beschrijving van de input/output mogelijkheden van de PicoMite.
`PIO`	**De RP2040 chip die in de PicoMite wordt gebruikt, bevat een programmeerbaar I/O systeem met twee identieke PIO apparaten (pio%=0 of pio%=1) die werken als gespecialiseerd CPU-kernen.** De **Raspberry Pi Pico 2** met de **RP2350**-microcontroller heeft drie PIO-apparaten. **Zie bijlage F** voor een meer gedetailleerde beschrijving van het programmeren van PIO's.
`PIO assemble pio, linedata$`	**Deze opdracht assembleert en laadt op tekst gebaseerde PIO-assembler-code inclusief labels voor sprongen.** Gebruik: `PIO assemble pio,".program Anything"` Om de assembler te initialiseren. Gebruik: `PIO assemble pio,".side_set n[opt][pindirs]"` Als je side set gebruikt. Dit is verplicht om de opcodes correct te constru-eren als er een of meer zijpennen worden gebruikt. Het laadt het **pinctrl-register** niet, aangezien dit specifiek is voor de state-machine. Merk ook op dat de parameter **"opt"** de opcode verandert op instructies die een zijparameter hebben. Gebruik: `PIO assemble pio,".line n"`

	Om te assembleren vanaf een andere regel dan 1 - dit is optioneel. Gebruik: `PIO assemble pio,".end program [list]"` Om de montage te beëindigen en de pio te programmeren. De optionele parameter **LIST** veroorzaakt een hex dump van de opcodes naar de terminal. Gebruik: `PIO assemble pio,"label:"` Om een label te definiëren. Dit moet verschijnen als een aparte opdracht. Gebruik: `PIO assemble pio,"instructie [parameters]"` Om de daadwerkelijke PIO instructies te definiëren die zullen worden geconverteerd naar machine code.
`PIO CLEAR pio`	**Dit stopt de pio gespecificeerd op alle statemachines** en **wist de controle registers voor de statemachines PINCTRL, EXECTRL en SHIFTCTRL naar standaard instellingen.**
`PIO DMA RX pio, sm, nbr, data%()[, completioninterrupt] [,transfersize] [,loopbackcount]` en `PIO DMA TX pio, sm, nbr, data%()[, completioninterrupt] [,transfersize] [,loopbackcount]`	**Stelt DMA-overdrachten in van PIO naar MMBasic-geheugen** **pio specificeert welke van de twee pio-instanties moet worden gebruikt (0 of 1 sm specificeert welke van de statusmachines moet worden gebruikt (0-3).** **nbr** specificeert hoeveel **32-bits woorden** er moeten worden overgedragen. Zie hieronder voor het speciale geval van het instellen van **nbr op nul**. **data%()** is de array die de **PIO-Data** levert of ontvangt. De optionele parameter completeringinterrupt is de naam van een MMBasic subroutine rdie wordt aangeroepen wanneer de DMA is voltooid en in het geval van **DMA_OUT** de **FIFO** is geleegd. Als de optionele interrupt niet wordt gebruikt, kan de status van de DMA worden gecontroleerd gebruik van de functies `MM.INFO(PIO RX DMA)` `MM.INFO(PIO TX DMA)` Met de optionele parameteroverdrachtgrootte kan de gebruiker de normale **32-bit** overdrachten en selecteer **8, 16** of **32**. De optionele parameter loopbackcount specificeert hoeveel gegevensitems er moeten worden gelezen of geschreven voordat de DMA opnieuw begint aan het begin van de buffer. De parameter moet een **macht** van **2** zijn tussen **2** en **32768**. Vanwege een beperking in de **RP2040 / RP2350** als loopbackcount wordt gebruikt van de MMBasic-array moet in het geheugen worden uitgelijnd op het aantal bytes in de lus. Dus als de array 64 gehele getallen lang is, wat 512 bytes is, dan moet de array dat ook zijn uitgelijnd met een grens van 512 bytes in het geheugen. Alle MMBasic-arrays zijn uitgelijnd op een grens van 256 bytes, maar om een array te maken die gegarandeerd wordt uitgelijnd met een grens van 512 bytes of groter dan de **PIO**. Het commando **MAKE RING BUFFER** moet worden gebruikt als loopbackcount is ingesteld, kan **"nbr"** worden ingesteld op 0. In dit geval vindt de overdracht plaats continu uitvoeren en herhaaldelijk de buffer vullen totdat deze expliciet wordt gestopt.

COMMANDO	Beschrijving 227
PIO DMA RX OFF PIO DMA TX OFF	**Breekt een actieve DMA af.**
PIO EXECUTE pio, state_machine, instructie%	**Voert de instructie onmiddellijk uit op de opgegeven pio** en **state-machine.**
PIO INIT MACHINE pio%, statemachine%, clockspeed [,pinctrl] [,execctrl] [,shiftctrl] [,startinstruction]	**Initialiseert PIO 'pio%' met state machine 'statemachine%'.** **'kloksnelheid'** is de kloksnelheid van de toestandsmachine in kHz. **De vier optionele argumenten zijn**, **variabelen** met **initialisatie-waarden** van de **toestandsmachineregisters** en de **adres** van de **eerste instructie** die moet worden uitgevoerd (**standaard nul**). Deze **beslissen** hoe de **PIO zal werken**. De verwachting is dat de PIO-assembler uiteindelijk in staat zal zijn om de registerwaarden voor de gebruiker samen met de program-mareeks op basis van de gedefinieerde assembler-richtlijnen.
PIO INTERRUPT pio, sm [,RXinterrupt] [,TXinterrupt]	**Gebruik de waarde 0 voor RXinterrupt of TXinterrupt om een interrupt uit te schakelen.** Laat waarden weg die niet nodig zijn. De **RX**-onderbreking wordt geactiveerd wanneer een woord door de **PIO**-code in de gespecificeerde **FIFO** is **"geduwd"**. De gegevens **MOETEN** in de interrupt worden gelezen om deze te wissen. De **TX**-onderbreking wordt geactiveerd wanneer de opgegeven **FIFO FULL** is en de **PIO** code deze nu heeft "getrokken". **PIO**-onderbrekingen hebben voorrang op toetsenbordonderbreking-en, maar boven alles. Ook 0 accepteert voor TX-interrupt.
PIO PROGRAM pio, array%()	Programmeert het volledige pio-programmageheugen met de gege-vens in **array%()**. Zie **Bijlage F**.
PIO PROGRAM LINE pio, line, instruction	**Programmeert alleen de opgegeven regel in een PIO programma.**
PIO READ pio, state_machine, count, data%[()]	**Haalt de data-elementen uit de opgegeven pio en statemachine.** **Het gelezen is niet-blokkerend**, dus de **status machine** moet de **ge-gevens kunnen leveren** die zijn **aangevraagd**. NB: dit commando heeft waarschijnlijk extra mogelijkheden nodig in toekomstige releases.
PIO START pio, statemachine	**Start een gegeven state machine op pio.**
PIO STOP pio, statemachine	**Stop een gegeven state machine op pio.**
PIO WRITE pio, state_machine, telling, data0 [,data1....]	**Schrijft de data elementen naar de opgegeven pio- en state machine. Het schrijven is blokkeren, zodat de statusmachine de verstrekte gegevens moet kunnen opnemen.** NB: dit commando heeft waarschijnlijk extra mogelijkheden nodig in toekomstige releases.
PIO WRITEFIFO a,b,v,d	**Schrijft naar een van de 4 individuele FIFO-registers.** **'a'** is de **pio (0 of 1), 'b' id** de toestandsmachine **(0...3), 'c'** is het **FIFO-register *0...3), 'd'** is het **data% (32 bit** gehele waarde)

COMMANDO	Beschrijving 228
PIXEL x, y [,c]	**Stel een pixel op een aangesloten LCD paneel in op een kleur.** **'x'** is de **horizontale coördinaat** en **'y'** is de **verticale coördinaat** van de pixel. **'c' is een 24 bit** nummer dat de **kleur** aangeeft. **'c'** is optioneel en indien weggelaten de huidige voorgrondkleur wordtgebruikt van de **kleinste array**. **'x'** en **'y'** moeten **beide arrays** zijn of **beide** zijn **enkele variabelen / constanten** anders wordt er een fout gegenereerd. **'c'** kan zijn ofwel **een array** of een **enkele variabele** of **constante**. Zie de sectie Basis tekencommando's voor een definitie van de kleuren en grafische coördinaten. Alle **parameters** kunnen worden uitgedrukt **als arrays** en de software zal het aantal **pixels plotten** zoals bepaald door de afmetingen van de **kleinste array**. **'x'** en **'y'** moeten **beide arrays** zijn of **beide** zijn **enkele variabelen / constanten** anders wordt er een fout gegenereerd. **'c'** kan zijn ofwel **een array** of een **enkele variabele** of **constante**. Zie de sectie Basis tekencommando's voor een definitie van de kleuren en grafische coördinaten.
PLAY	**Deze opdracht genereert een verscheidenheid aan audiouitgangen.** Zie het **OPTION AUDIO** commando voor het instellen van de I/O-pinnen die moeten worden gebruikt voor het resultaat. De audio is een pulsbreedte-gemoduleerd signaal, dus een laagdoorlaatfilter is vereist om de draaggolffrequentie te verwijderen.
PLAY TONE left [, right [, dur] [,interrupt]]]	**Genereert twee afzonderlijke sinusgolven op de geluidsuitvoer links en rechts kanalen.** **'links'** en **'rechts'** zijn de frequenties in **Hz** voor **links** en **rechts** kanalen. De toon speelt op de achtergrond (het programma gaat verder uitgevoerd na dit commando) en **'dur'** specificeert het aantal milliseconden waarvoor de toon zal klinken. Als de duur niet is gespecificeerd, zal de toon doorgaan totdat het programma expliciet wordt gestopt of het programma wordt beëindigd. **'interrupt'** is optioneel en is de naam van een subroutine die wordt aangeroepen als het bestand is afgespeeld. De **frequentie** kan van **1 Hz** tot **20 KHz** zijn en is zeer nauwkeurig (het is gebaseerd op op een kristaloscillator). De frequentie kan op elk moment worden gewijzigd door een nieuw **PLAY TONE** commando te geven.
PLAY FLAC file$ [, interrupt]	**PLAY FLAC- hetzelfde als PLAY WAV maar voor FLAC-bestanden.** Speelt een **FLAC bestand** af op de geluidsuitvoer. **'file$'** is het **FLAC bestand** dat moet worden afgespeeld (de extensie **.flac** wordt toegevoegd als het ontbreekt). De samplefrequentie kan oplopen tot **48 kHz** in stereo (**96 kHz** als de Pico dat is overgeklokt) Het **FLAC-bestand** wordt op de achtergrond afgespeeld. **'interrupt'** is optioneel en is de naam van een subroutine die wordt aangeroepen als het bestand klaar is met afspelen. Als **file$ een map** is, zal de Pico alle bestanden in die map achter-eenvolgens afspelen.

COMMANDO	Beschrijving 229
PLAY WAV file$ [, interrupt]	**Speelt een WAV-bestand af op de geluidsuitvoer.** **'file$'** is het **WAV-bestand** dat moet worden afgespeeld (de extensie van **.wav** wordt toegevoegd als niet aanwezig). Het **WAV-bestand** moet **PCM-gecodeerd** zijn in **stereo met 8-bit bemonstering**. De samplefrequentie kan oplopen tot **48kHz** in stereo. Het **WAV-bestand** wordt op de achtergrond afgespeeld. **'interrupt'** is optioneel en is de naam van een subroutine die wordt aangeroepen als het bestand is afgespeeld.
PLAY MP3 file$ [, interrupt]	**Speelt een MP3-bestand af via de geluidsuitvoer (ALLEEN RP2350).** **'file$'** is het **MP3-bestand** dat moet worden afgespeeld (de extensie **.mp3** wordt toegevoegd als deze ontbreekt). De samplefrequentie kan oplopen tot **48 kHz**. Het **MP3-bestand** wordt op de achtergrond afgespeeld. **'interrupt'** is optioneel en is de naam van een subroutine die wordt aangeroepen als het bestand klaar is met afspelen. Als **file$ een map** is, zal de Pico alle bestanden in die map achtereenvolgens afspelen.
PLAY MODFILE file$ [,interrupt]	**Speelt een MOD-bestand af op de geluidsuitvoer.** **'file$'** is het **MOD-bestand** dat moet worden afgespeeld. (de extensie **.mod** wordt toegevoegd als het ontbreekt). Het **MOD-bestand** wordt op de achtergrond afgespeeld en continu afgespeeld in een lus. Als de optionele **'interrupt'** is opgegeven, wordt dit aangeroepen als het bestand is voltooid wordt één keer afgespeeld in de reeks en het afspelen wordt dan beëindigd.
PLAY MODSAMPLE Samplenum, channel [,volume]	**Speelt een specifieke sample af in het mod-bestand op het opgegeven kanaal.** Het volume is optioneel en kan tussen **0** en **64** liggen. Deze opdracht kan alleen worden gebruikt wanneer er wordt al een mod-bestand afgespeeld en tijdens het afspelen kunnen geluidseffecten worden uitgevoerd de achtergrondmuziek speelt nog steeds.
PLAY LOAD SOUND array%()	**Laadt een array met 1024 elementen, bestaande uit 4096 16-bits waarden tussen 0 en 4095.** Dit levert de gegevens voor elke willekeurige golfvorm waarmee gespeeld kan worden het **PLAY SOUND** commando. U kunt de opdracht **MEMORY PACK** gebruiken om maak de arrays op basis van een normale integer-array met **40956 elementen**.

COMMANDO	Beschrijving 230
PLAY SOUND soundno, channelno, type [,frequency] [,volume]	**Speel een reeks geluiden tegelijkertijd af op de audio-uitgang.** **'soundno'** is het geluidsnummer en kan van **1** tot en met **4** zijn, waardoor er vier mogelijk zijn gelijktijdige geluiden op elk kanaal. **'channelno'** specificeert het uitgangskanaal. Dit kan: **L** (**links**) **R** (**rechts**) **B** (**beide luidsprekers**) **M** (**mono** uitvoer met het rechterkanaal omgekeerd vergeleken met links). **'type'** specificeert de te gebruiken golfvorm. Het kan een: **S** (**sinus golf**) **Q** (**vierkant golf**) **T** (**driehoekige golf**) **W** (**zaagtand golf)** **O** (**nuluitgang**) **P** (**periodiek ruis**) **N** (**willekeurige ruis**) **U** (**door gebruiker gedefinieerd** met **PLAY LOAD SOUND**). **'frequentie'** is de **frequentie van 1 tot 20.000 (Hz)** en moet worden gespecificeerd behalve wanneer het **type O** is. **'volume'** is optioneel en moet tussen **1** en **25** liggen. De standaard-waarde is **25**. Wanneer **PLAY SOUND** wordt aangeroepen, wordt al het andere audiogebruik geblokkeerd, blijven geblokkeerd totdat **PLAY STOP** wordt geroepen. De uitvoer kan worden gestopt tijdelijk gebruik van **PLAY PAUSE** en **PLAY RESUME**. Als u **SOUND** aanroept op een al actief **'soundno'**, wordt onmiddellijk het vorige uitvoer. Individuele geluiden worden uitgeschakeld met behulp van het type **“O”** Er worden 4 geluiden tegelijkertijd weergegeven op beide kanalen van de audio-uitgang verbruikt ongeveer 23% van de CPU.
PLAY PAUSE	**PLAY PAUSE stopt tijdelijk het momenteel afgespeelde bestand of de toon.**
PLAY RESUME	**PLAY RESUME hervat het afspelen van een geluid dat was gepauzeerd.**
PLAY STOP	**PLAY STOP beëindigt het afspelen van het bestand of de toon.** Wanneer het programma om welke reden dan ook wordt beëindigd, wordt de geluidsuitvoer ook automatisch uitgevoerd gestopt.
PLAY NEXT	**Stopt het afspelen van het huidige audiobestand en start het volgende in de directory.**
PLAY PREVIOUS	**Stopt het afspelen van het huidige audiobestand en start het vorige in de directory.**

COMMANDO	Beschrijving 231
PLAY VOLUME left, right	**Past het volume van de audio-uitgang aan.** **'links'** en **'rechts'** zijn de niveaus die moeten worden gebruikt voor de **linker-** en **rechter kanalen** en kunnen worden tussen **0** en **100**, waarbij **100** het maximale volume is. Er is een lineaire relatie tussen het gespecificeerde niveau en de output. Het volume staat standaard op maximum wanneer een programma wordt uitgevoerd.
VS1053 PLAY commando 's	**VS1053 specifieke PLAY commando's**
PLAY MP3 file$ [, interrupt]	**Speelt een MP3-bestand af op de geluidsuitvoer.** **'file$'** is het **MP3-bestand** dat moet worden afgespeeld (de extensie **.mp3** wordt toegevoegd als ontbreekt). De samplefrequentie moet **44100 Hz** stereo zijn. Het **MP3-bestand** wordt op de achtergrond afgespeeld. **'interrupt'** is optioneel en is de naam van een subroutine die wordt aangeroepen als het bestand klaar is met afspelen. Als **file$** een **map is**, zal de **Pico** alle bestanden in die map achtereenvolgens afspelen.
PLAY HALT	**Deze opdracht werkt wanneer een MP3-bestand wordt afgespeeld.** Het stopt het afspelen en registreert de huidige bestandspositie, zodat het afspelen vanaf hetzelfde punt kan worden hervat. Deze opdracht is speciaal ontworpen om mp3-audioboeken te ondersteunen
PLAY CONTINUE track$	**Hervat het afspelen van het opgegeven MP3-nummer.** **"track$"** zal de naam zijn van het bestand dat werd afgespeeld toen het werd gestopt en alle bestandskenmerken waren verwijderd. **PLAY** bijvoorbeeld: **MP3** **"B:/mp3/mymp3.mp3"** enige tijd later af. **PLAY HALT** later opnieuw. **PLAY RESUME** **"mymp3"**
PLAY MIDIFILE file$ [, interrupt]	**Speelt een MIDI-bestand af op de geluidsuitvoer.** **'file$'** is het **MIDI-bestand** dat moet worden afgespeeld (de extensie **.mid** wordt toegevoegd als deze ontbreekt). Het **MIDI-bestand** wordt op de achtergrond afgespeeld. **'interrupt'** is optioneel en is de naam van een subroutine die wordt aangeroepen als het bestand klaar is met afspelen. Als file$ een map is, zal de Pico alle bestanden in die map achtereenvolgens afspelen.
PLAY MIDI	**Start de realtime midi-modus.** In deze modus kunnen midiinstructies naar de **VS1053** worden gestuurd om te selecteren welke instrumenten op welke kanalen moeten worden bespeeld, noten aan te zetten en ze uit te zetten in real timer.
PLAY MIDI CMD cmd%, data1%, data2%	**Stuurt een midi-commando in realtime midi-modus.** Een voorbeeld is het toewijzen van een instrument aan een kanaal. Bijvoorbeeld: `PLAY MIDI CMD &B11000001, 4  ' Zet kanaal 1 op instrument 4.`

COMMANDO	Beschrijving 232
PLAY MIDI TEST n	**Speelt een MIDI-testsequentie af, n = 0 tot 3, 0 = normale realtime, de anderen spelen noot- en instrumentsamples.**
PLAY NOTE ON channel%, note%, velocity%	**Schakelt de noot in op het opgegeven kanaal in de realtime MIDI-modus.**
PLAY NOTE OFF channel%, note% [,velocity%]	**Schakelt de noot uit op het opgegeven kanaal in de realtime MIDI-modus.**
PLAY STREAM buffer%(), readpointer%, writepointer%	**Verzendt gegevens naar de VS1053 CODEC vanuit de circulaire buffer "buffer%".** Deze opdracht initieert een achtergronduitvoerstroom waarbij de **VS1053** alles in de buffer tussen de leespointer en de schrijfpointer verzendt, waarbij de leespointer gaandeweg wordt bijgewerkt. Kan worden gebruikt voor willekeurige golfvormuitvoer.
POKE BYTE addr%, byte	**Stelt een byte of een woord in binnen de PicMite virtuele geheugen ruimte**. **POKE BYTE** zal de byte (d.w.z. **8 bits**) op de geheugenlocatie **'addr%'** instellen op **'byte'**. **'addr%'** moet een geheel getal zijn.
POKE SHORT addr%, short%	**POKE SHORT stelt het korte gehele getal (d.w.z. 16 bits) in op de geheugenlocatie 'addr%' naar 'woord%'.** **'addr%'** en **short%'** moeten gehele getallen zijn.
POKE WORD addr%, word%	**POKE WORD zet het woord (d.w.z. 32 bits) op de geheugenlocatie 'addr%' naar 'woord%'**. **'addr%' en 'word%'** moeten gehele getallen zijn.
POKE INTEGER addr%, int%	**POKE INTEGER zal het MMBasic integer (d.w.z. 64 bits) in het geheugen plaatsen locatie 'addr%' naar int%'.** **'addr%' en int%'** moeten gehele getallen zijn.
POKE FLOAT addr%, float!	**POKE FLOAT zet het woord (d.w.z. 32 bits) op de geheugenlocatie 'addr%' drijven!'.** **'addr%'** moet een geheel getal zijn en **'float!'**
POKE VAR var, offset, byte	**POKE VAR zal een byte instellen in het geheugenadres van 'var'.** **'offset'** is de **±offset** van het **adres van de variabele**. Een **array** wordt **gespecificeerd als var()**.
POKE VARTBL, offset, byte	**POKE VARTBL zal een byte instellen in de variabele tabel van MMBasic**. **'offset'** is de **±offset** vanaf het begin van de variabele tabel. **Let op:** een komma is vereist na het trefwoord **VARTBL**.
POKE DISPLAY commando [,data1] [,data2] [,datan]	**Deze opdracht stuurt opdrachten en bijbehorende gegevens naar de beeldschermcontroller voor een aangesloten beeldscherm.** Hierdoor kan de programmeur parameters wijzigen hoe het display is geconfigureerd. bijv. **POKE DISPLAY &H28** wordt uitgeschakeld een **SSD1963 display** en **POKE DISPLAY &H29** zullen het weer inschakelen. Werkt voor alle schermen behalve **VGA, ST7790** en **GDEH029A1.**

COMMANDO	Beschrijving
`POKE DISPLAY HRES n` `POKE DISPLAY VRES n`	**Deze commando's veranderen de opgeslagen waarde van MM.HRES en MM.VRES waardoor de programmeur niet standaard displays kan configureren.**
`POLYGON n, xarray%(), yarray%()` `[, borderkleur]` `[, vulkleur]`	**Tekent een gevulde of omlijnde veelhoek met n xy coördinaatparen in xarray%() en yarray%().** Als **'vulkleur'** wordt weggelaten, is alleen de polygoonomtrek getrokken. Als **'borderkleur'** wordt weg-gelaten, wordt deze standaard ingesteld op de huidige standaard voorgrond kleur.
`POLYGON n(), xarray%(), yarray%()` `[, randkleur()]` `[, vulkleur()]`	**Als het laatste xy-coördinatenpaar niet hetzelfde is als het eerste, zal de firmware automatisch een extra xy-coördinatenpaar aan maken om de veelhoek.** De grootte van de arrays moet minstens zo groot zijn als het aantal x,y coördinaat paren.
`POLYGON n(), xarray%(), yarray%()` `[, randkleur]` `[, vulkleur()]`	**'n' kan een array zijn en de kleuren kunnen optioneel ook als volgt arrays zijn**: `POLYGON n(),xarray%(), yarray%()[,randkleur()][,vulkleur]` `POLYGON n(),xarray%(), yarray%()[,randkleur][,vulkleur]` De elementen van **array n()** definiëren het aantal **xy-coördinaat** paren in elk van de veelhoeken. bijv. **DIM n(1) = (3, 3)** zou definiëren dat **2 polygonen** moeten zijn getekend met elk drie hoekpunten. De grootte van de n array bepaalt de aantal polygonen dat zal worden getekend tenzij er een element wordt gevonden met de waarde nul in welk geval de firmware alleen polygonen tot die tijd verwerkt punt. De x,y-coördinatenparen voor alle polygonen worden opgeslagen in **xarray%()** en **yarray%()**. De parameters **xarray%()** en **yarray%()** moeten minimaal evenveel elementen hebben als het totaal van de waarden in de **n-array**. Elke polygoon kan worden gesloten met hetzelfde eerste en laatste element. Als het laatste element is niet hetzelfde als het eerste, de firmware zal automatisch een extra x, y-coördinatenpaar aanmaken om de veelhoek te voltooien. Als vulkleur wordt weggelaten, dan worden alleen de polygooncontouren getekend. De kleurparameters kunnen een enkele waarde zijn. In dat geval zijn alle polygonen getekend in dezelfde kleur of het kunnen arrays zijn met dezelfde kardinaliteit als **N**. In dit geval kan elke getekende polygoon een andere kleur van beide randen hebben en of vullen. Dit tekent bijvoorbeeld 3 driehoeken in geel, groen, rood: `DIM c%(2)=(3,3,3)` `DIM x%(8)=(100,50,150,100,50,150,100,50,150)` `DIM y%(8)=(50,100,100,150,200,200,250,300,300)` `DIM fc%(2)=(rgb(yellow),rgb(green),rgb(red))` `POLYGON c%(),x%(),y%(),fc%(),fc%()`

COMMANDO	Beschrijving 234
PORT(start, nbr [, start, nbr]..) = waarde	**Stel een aantal I/O-pinnen tegelijk in (dus met één commando).** **'start'** is een I/O-pincode en het laagste bit in **'waarde'** (**bit 0**) wordt gebruikt om **zet die pin**. **Bit 1** wordt gebruikt om de pin **'start'** plus 1 in te stellen, **bit 2** zal pin instellen **'start'+2** enzovoort voor **'nbr'** aantal bits. Gebruikte I/O-pinnen moeten worden genummerd op-eenvolgend en elke I/O-pin die ongeldig is of niet is geconfigureerd als uitgang zal een fout veroorzaken. Het start/nbr-paar kan worden herhaald als een extra groep van output pinnen moet worden toe-gevoegd. Bijvoorbeeld: **PORT(15, 4, 23, 4) = &B10000011** Zal acht I/O-pinnen instellen. **Pins 15 en 16** worden **hoog** gezet terwijl **17, 18, 23, 24** en **25** worden op een **laag** gezet en ten slotte worden **26 hoog** ingesteld. Deze opdracht kan worden gebruikt om gemakkelijk te communiceren met parallel apparaten zoals **LCD-schermen**. Een willekeurig aantal **I/O-pinnen** (en dus **bits**) kan worden gebruikt van **1** tot het aantal **I/O-pinnen** op de chip. Zie de **PORT** functie om gelijktijdig van een aantal pinnen te lezen.
PRINT uitdrukking [[,;]uitdrukking].enz of **?** uitdrukking [[,;]uitdrukking].enz	**Voert tekst uit naar de seriële console, gevolgd door een regelterugloop / nieuwe regel.** Er kunnen meerdere uitdrukkingen worden gebruikt en deze moeten worden gescheiden door een: • Komma **(,)** die het tab-teken zal weergeven • Puntkomma **(;)** die niets zal uitvoeren (het wordt alleen gebruikt om uitdrukkingen te scheiden). • **Niets** of een spatie die hetzelfde werkt als een puntkomma. Een puntkomma **(;)** of een komma **(,)** aan het einde van de lijst met uitdrukkingen wordt onderdrukt de uitvoer van het naar regelterugloop **/** nieuwe regel aan het einde van een afdrukinstructie. Bij het afdrukken wordt een getal voorafgegaan door een spatie indien positief of een minteken **(-)** indien negatief, maar wordt niet gevolgd door een spatie. Gehele getallen (hele getallen) zijn afgedrukt zonder decimale punt terwijl breuken worden afgedrukt met decimaal punt en de significante decimale cijfers. Grote of kleine drijvende komma getallen worden automatisch afgedrukt in wetenschappelijk getalformaat. De functie **TAB()** kan worden gebruikt om een spatie naar een bepaalde kolom te plaatsen en de **STR$()** functie kan worden gebruikt om tekenreeksen te rechtvaardigen of anderszins op te maken.
PRINT #nbr, expressie [[,;]uitdrukking].enz of **?** #nbr, expressie [[,;]uitdrukking].enz	**Hetzelfde als hierboven, behalve dat de uitvoer is gericht op een seriële communicatie port of een bestand geopend voor OUTPUT of APPEND met het bestandsnummer 'nbr'.** Zie het commando **OPEN**.

COMMANDO	Beschrijving 235
`PRINT #GPS,expressie [[,; ]uitdrukking].enz` of `? #GPS, expressie [[,; ]uitdrukking].enz`	**Voert een NMEA string uit naar een geopend GPS apparaat**. De string moet beginnen met een **$ teken** en eindigen met een ***** **teken.** De controlesom wordt automatisch berekend en toegevoegd aan de string samen met de **CR/LF** tekens.
`PRINT @(x [, y]) uitdrukking` of `? @(x, [y], m) uitdrukking`	**Werkt op terminalconsole op een aangesloten computer of het display als: `OPTION LCDPANEL CONSOLE` is ingeschakeld.** **Hetzelfde als het standaard `PRINT` commando, behalve dat de cursor op coördinaten staat de x, y uitgedrukt in pixels.** Als y wordt weggelaten, wordt de cursor geplaatst op **"x"** op de huidige regel. Voorbeeld: `PRINT @(150,45) "Hallo wereld"` De **@** functie kan overal in een printopdracht worden gebruikt. Voorbeeld: `PRINT @(150,45) "Hallo" @(150,55) "Wereld"` De functie **@(x,y)** kan worden gebruikt om de cursor overal aan of uit te zetten op het scherm. `PRINT @(-10, 0) "Hallo"` toont bijvoorbeeld alleen **"llo"** als de eerste twee tekens konden niet worden weergegeven omdat ze niet op het scherm stonden. De functie **@(x,y)** onderdrukt automatisch de automatische regelterugloop normaal uitgevoerd wanneer de cursor buiten de rechterschermmarge komt. Als **'m'** is opgegeven, is de modus van de videobewerking als volgt: **m = 0** Normale tekst (witte letters, zwarte achtergrond) **m = 1** De achtergrond wordt niet getekend (dwz transparant) **m = 2** De video wordt omgekeerd (zwarte letters, witte achtergrond) **m = 5** Huidige pixels worden omgekeerd(transparante achtergrond)
`PULSE pin, breedte`	**Zal een puls op 'pin' genereren met een duur van 'breedte' ms.** **'breedte'** kan een fractie zijn. Bijvoorbeeld, **0,01** is gelijk aan **10 µs** en dit maakt de generatie van zeer smalle pulsen. Het minimum is **5 µs** bij **40 MHz** tot **40 µs** bij **5 MHz**. De gegenereerde puls heeft de tegenovergestelde polariteit van de toestand van de I/O-pin wanneer de opdracht wordt uitgevoerd. Als de output bijvoorbeeld hoog is ingesteld. Het **PULSE** commando genereert een negatief gaande puls. Opmerkingen: • **'pin'** moet als uitgang worden geconfigureerd. • Voor een **puls** van minder dan **3 ms** is de nauwkeurigheid **± 1 µs**. • Voor een **puls** van **3 ms** of meer is de nauwkeurigheid **± 0,5 ms**. • Op de achtergrond loopt een **puls** van **3 ms** of meer. Tot vijf verschillende en gelijktijdige pulsen kunnen op de achtergrond worden uitgevoerd en elk kan laat de tijd veranderen door een nieuw **PULSE** commando te geven of het kan beëindigd door het geven van een **PULSE** commando met nul voor **'breedte'**.

COMMANDO	Beschrijving 236
PWM kanaal, OFF	Met PWM **kanaal, OFF** stopt men de PWM uitvoer.
PWM kanaal,frequentie, [dutyA] [,dutyB] [,phase] [,defer]	**Er zijn 8 afzonderlijke PWM-frequenties beschikbaar (kanalen 0 t/m 7) en maximaal 16 uitgangen met individueel geregelde inschakelduur.** U kunt op beide uitvoeren **PWMnA** of **PWMnB** of beide voor elk kanaal - geen beperking. Inschakelduur is dat wel opgegeven als een percentage en u kunt een negatieve waarde gebruiken om de uitvoer om te keren (**-100, 0 <= dienst <= 100, 0**). **Minimale frequentie = (cpusnelheid + 1) / (2^24) Hz. (15Hz)** **Maximale snelheid bedraagt OPTION CPUSPEED / 4.** Bij zeer hoge snelheden zullen de duty-cycles hetzelfde zijn steeds beperkter. **Fase** is een parameter die ervoor zorgt dat de golfvormen zodanig worden gecentreerd dat er een golf ontstaat. Een formulier met een kortere inschakelduur begint en eindigt gelijke tijden vanaf een langere. Gebruik 1 om deze modus in te schakelen en 0 (of weg te laten) om normaal te werken. Als de parameter **'deferredstart'** is ingesteld op **1**, worden de PWM-kanalen geconfigureerd als maar start de uitvoer niet. Ze kunnen worden gestart met behulp van de PWM **SYNC**-opdracht. Dit kan worden gebruikt om ongewenste opstartartefacten te voorkomen Het PWM-commando kan ook servo's als volgt aansturen: **PWM 1, 50, (positie_als_percentage * 0.05 + 5)**
PWM SYNC s0 [,s1][,s2][,s3][,s4] [,s5][,s6][,s7]	**Dit initieert de PWM op kanalen waar een uitgestelde start is gedefinieerd of synchroniseert alleen bestaande lopende kanalen.** De kracht zit echter in de mogelijkheid om de kanalen ten opzichte van elkaar te compenseren (gedefinieerd als een percentage van de tijdsperiode volgens de **duty cycle** kan een **float** zijn). Een **offset** kan ook **-1** zijn en **-1** kan men gebruiken om een kanaal uit de synchronisatie weg te laten.
RANDOMIZE nbr	**Seed de random number generator met 'nbr'.** Bij het opstarten wordt de generator voor willekeurige getallen gezaaid met nul en zal elke keer dezelfde reeks willekeurige getallen genereren. Om een te genereren andere willekeurige volgorde elke keer moet je een andere waarde gebruiken voor **'nbr'** (daarvoor is de TIMER functie handig). Deze opdracht doet niets op de **RP2350,** die een hardware willekeurige generator heeft waarvoor geen seeden vereist is.
REFRESH	**Start een update van het scherm voor bepaalde zwartwitschermen.** Deze kunnen alleen een volledig scherm tegelijk worden bijgewerkt en als OPTION **AUTOREFRESH** is **OFF**. Deze opdracht kan worden gebruikt om het schrijven te activeren. Dit geldt voor de volgende beeldschermen: **N5110, SSD1306I2C, SSD1306I2C32, SSD1306SPI, ST7920, GDEH029A1**.

COMMANDO	Beschrijving 237
RBOX x, y, w, h [, r] [,c] [, fill]	**Tekent een doos met afgeronde hoeken op een aangesloten LCD paneel beginnend bij 'x' en 'y' die 'w' pixels breed en 'h' pixels hoog is.** **'r'** is de straal van de hoeken van de doos. Het staat standaard op **10** **'c'** specificeert de kleur en wordt standaard ingesteld op de standaard voorgrondkleur als dat niet het geval is gespecificeerd. **'fill'** is de vulkleur. Het kan worden weggelaten of ingesteld op **-1**, in dat geval zal de doos niet worden gevuld. Alle parameters kunnen worden uitgedrukt als arrays en de software plot de aantal dozen zoals bepaald door de afmetingen van de kleinste array. **'x', 'y', 'w'** en **'h'** moeten allemaal arrays zijn of allemaal enkele variabelen / constanten anders wordt er een fout gegenereerd. **'r', 'c'** en **'fill'** kunnen ofwel arrays zijn of enkele variabelen / constanten. Zie het hoofdstuk "Basis Tekenopdrachten" voor een definitie van de kleuren en grafische coördinaten.
READ variabele [, variabele]...	**Leest waarden uit DATA instructies en wijst deze waarden toe aan de benoemde variabelen.** Variabeletypen in een READ instructie moeten overeenkomen met de gegevenstypen in DATA statements zoals ze worden gelezen. Arrays kunnen worden gebruikt als variabelen (aangegeven met lege haakjes, bijv. **a()**) en in dat geval wordt de grootte van de array gebruikt om te bepalen hoeveel elementen er zijn om gelezen te worden. Als de array multidimensionaal is, zal de meest linkse dimensie de snelst bewegende zijn. Zie ook DATA en RESTORE.
READ SAVE or READ RESTORE	**READ SAVE bewaart de virtuele aanwijzer die door het READ-commando wordt gebruikt om naar de volgende DATA te wijzen die moet worden gelezen.** READ RESTORE herstelt de aanwijzer die eerder was opgeslagen. Hierdoor kunnen subroutines gegevens READ en vervolgens de leeswijzer herstellen om andere delen van het programma die mogelijk dezelfde gegevensinstructies lezen, niet te verstoren. Deze opdrachten kunnen worden genest.
REM string	**Met REM kunnen opmerkingen in een programma worden opgenomen.** Let op het gebruik in Microsoft stijl van het enkele aanhalingsteken **(')** om aan te duiden opmerkingen wordt ook ondersteund en heeft de voorkeur.
RENAME old$ AS new$	**Hernoem een bestand of een map van 'old$' naar 'new$'. Beide zijn strings. Dus " " gebruiken.** Een directorypad kan zowel in **'old$'** als in **'new$'** worden gebruikt. Als de paden verschillen, bestand gespecificeerd in **'old$'** wordt verplaatst naar het pad gespecificeerd in **'new$'** met de bestandsnaam zoals opgegeven. Voorbeeld: `RENAME "Ledje.bas AS "Led.bas"`

COMMANDO	Beschrijving
`RESTORE [line]`	**Stelt de regel- en positietellers voor de READ instructie opnieuw in.** Als **'lijn'** is opgegeven, worden de tellers teruggezet naar het begin van de opgegeven lijn. **'regel'** kan een regelnummer of label zijn of een variabele hiermee waarden. Als **'lijn'** niet is opgegeven, worden de tellers teruggezet naar de start van het programma.
`RMDIR dir$`	**Verwijder of verwijder de map 'dir$' op de SD-kaart.** Voorbeeld: `RMDIR "Hobby"`
`RTC GETTIME`	**RTC GETTIME krijgt de huidige datum/tijd van een PCF8563, DS1307 of DS3231 real-time klok en stel de interne MMBasic klok dienovereenkomstig in.** De **datum/tijd** kan dan worden opgevraagd met de functies DATE$ en TIME$.
`RTC SETTIME jaar, mnd, dag, uur, min, sec`	**RTC SETTIME stelt de tijd in op de klokchip.** **'uur'** moet 24 uur gebruiken notatie. De opdracht **RTC SETTIME** accepteert ook een enkele tekenreeks argument in de notatie **dd/mm/jj uu:mm**. Dit betekent dat de **datum/tijd** kan worden ingevoerd door de gebruiker met behulp van een GUI **FORMATBOX** met het **DATETIME2** formaat.
`RTC SETREG reg, waarde` `RTC GETREG reg, var`	**De opdrachten RTC SETREG en GETREG kunnen worden gebruikt om de inhoud van registers in de chip.** **'reg'** is het nummer van het register, **'value'** is het nummer dat in het register moet worden opgeslagen en **'var'** is een variabele die het gelezen nummer zal ontvangen uit het register. Deze commando's zijn niet nodig voor normale werking, maar ze kunnen worden gebruikt om speciale functies van de chip (alarmen, uitgangssignalen, enz.). Ze zijn ook handig voor tijdelijke opslag informatie in het door de batterij ondersteunde **RAM**-geheugen van de chip. Deze chips zijn I2C apparaten en moeten worden aangesloten op de twee **I2C-pinnen** met geschikte **pullup-weerstanden**. Als de **I2C-bus** al open is, is de RTC commando zal de huidige instellingen gebruiken anders zal het tijdelijk de verbinding met een snelheid van 100 kHz. Zie ook commando OPTION RTC **AUTO ENABLE.**
`RUN` of `RUN[file$][, cmdline$]`	**Voer een programma uit.** Als **'file$'** niet wordt opgegeven, voer dan het programma uit dat momenteel in het programma staat geheugen. Als **'file$'** wordt opgegeven, voert u het genoemde bestand uit vanaf de **Flash-** of **SD-kaart** bestandssysteem. Als **'file$'** geen extensie **'.BAS'** bevat, dan wordt er wel een automatisch toegevoegd. Als **'cmdline$'** wordt opgegeven, geef dan de waarde ervan door aan de constante MM.CMDLINE$ van het programma wanneer het wordt uitgevoerd. Als **'cmdline$'** niet wordt opgegeven, dan is er een lege string waarde wordt doorgegeven aan MM.CMDLINE$..

COMMANDO	Beschrijving 239
	Opmerkingen: • Zowel **'file$'** als **'cmdline$'** kunnen als tekenreeksexpressies worden opgegeven. • Gebruik **FLASH RUN n** om een programma uit te voeren dat is opgeslagen in een **Flash Slot**.
SAVE file$	**Slaat het programma op in de huidige werkdirectory van de SD-kaart als 'file$'.** Voorbeeld: **SAVE "TEST.BAS"** Als er geen extensie is opgegeven, wordt **".BAS"** aan de bestandsnaam toegevoegd. Zie ook **FLASH SAVE nr for saving to a Flash Slot**.
SAVE IMAGE file$ [,x, y, w, h] **SAVE COMPRESSED IMAGE file$ [,x, y, w, h]**	**Sla het huidige beeld op de video-uitgang of het LCD-paneel op als een BMP-bestand.** Elk LCD-paneel moet kunnen worden gelezen, bijvoorbeeld een op **ILI9341** gebaseerd paneel of een **VIRTUAL_M-** of **VIRTUAL_C**-paneel. **'file$'** is de naam van het bestand. Als er geen extensie is opgegeven, zal **".BMP"** dat zijn toegevoegd aan de bestandsnaam. De afbeelding wordt opgeslagen als een **24-bits afbeelding** met ware kleuren. **'x', 'y', 'w' en 'h'** zijn optioneel en zijn de coördinaten (**'x' en 'y'** zijn de bovenste linkercoördinaat) en afmetingen (breedte en hoogte) van het op te slaan gebied. Als niet gespecificeerd, wordt het hele scherm opgeslagen.
SEEK [#]fnbr, pos	**Plaatst de lees-/schrijfaanwijzer in een bestand dat op de SD is geopend kaart voor SEEK (WILLEKEURIGE) toegang tot de 'pos' byte.** De eerste byte in een bestand is genummerd met één, dus **SEEK #5, 1** plaatst de lees-/schrijfwijzer naar het begin van de file die is geopendals **#5**.
SELECT CASE waarde **CASE testexp** **[[, testexp]...]** **<statements>** **<statements>** **CASE ELSE** **<statements>** **<statements>** **END SELECT**	**Voert een van meerdere groepen instructies uit, afhankelijk van de waarde van een uitdrukking. 'waarde' is de uitdrukking die moet worden getest.** Het kan een nummer zijn of tekenreeksvariabele of een complexe uitdrukking. **'testexp'** is de waarde waarmee moet worden vergeleken. Het kan zijn: • Een enkele uitdrukking (d.w.z. **34, "string" or PIN(4) * 5**) waaraan deze gelijk kan zijn. • Een bereik van waarden in de vorm van twee enkele expressies gescheiden door de trefwoord **"TO"** (d.w.z. **5 TO 9** of **"aa" TO "cc"**). • Een vergelijking die begint met het trefwoord **"ELSE"** (dat optioneel is). Voor voorbeeld: **ELSE > 5, ELSE <= 10**. Wanneer een aantal testuitdrukkingen (gescheiden door komma's) worden gebruikt, wordt de **CASE**-instructie is waar als een van deze tests als waar wordt beoordeeld.

	Als **'waarde'** niet kan worden gekoppeld aan een **'testexp'**, wordt deze automatisch gekoppeld aan het **CASE ELSE**. Als **CASE ELSE** niet aanwezig is, wordt het programma niet uitgevoerd eventuele **<statements>** en ga verder met de code die volgt op de **END SELECT** Wanneer een overeenkomst is gemaakt, zullen de **<statements>** na het **CASE**-statement worden uitgevoerd totdat **END SELECT** of een andere **CASE** wordt aangetroffen wanneer de programma gaat dan verder met de code die volgt op de **END SELECT.** Er kan een onbeperkt aantal **CASE**-statements worden gebruikt, maar er moet slechts één **CASE ELSE** en dat zou de laatste moeten zijn voor de **END SELECT**. Voorbeeld: `SELECT CASE nbr%` `  CASE 4, 9, 22, 33 TO 88` `  uitspraken` `  CASE IS < 4, IS > 88, 5 TO 8` `  uitspraken` `  CASE ELSE` `  uitspraken` `END SELECT` Elke **SELECT CASE** moet één en één hebben die overeenkomt met **END SELECT** stelling. Elk aantal **SELECT...CASE**-instructies kan worden genest binnen de **CASE**-instructies van andere **SELECT.. CASE**-instructies.
SERVO channel [positionA [,positionB] **SERVO channel,, positionB**	**Bedien een standaard servo.** **'positionA'** en **'positionB'** kunnen tussen **-20** en **120** liggen en genereren een **50Hz-signaal** tussen **800uSec** en **2,2 mSec.** Net als bij het **PWM**-commando moeten de pinnen worden ingesteld met **SETPIN n, PWM** Om alleen **kanaal B** te gebruiken, gebruikt u de syntaxis: **SERVO kanaal, , positieB** Raadpleeg de pin-out om het kanaal en subkanaal (**A of B**) voor elke pin weer te geven. Voorbeeld: `> SetPin GP6, PWM` `> Servo 3, -20` `> Servo 3, 90` `> Servo 3, 120` `> PWM 3, OFF`

COMMANDO	Beschrijving
`SETPIN pin, cfg [, optie]`	**Configureert een externe I/O-pin. Raadpleeg het hoofdstuk "Gebruik van de I/O-pinnen" voor een algemene beschrijving van de input/output-mogelijkheden van de PicoMite.** **'pin'** is de **I/O-pin** die moet worden geconfigureerd, **'cfg'** is de modus waarop de pin moet worden ingesteld en **'optie'** is een optionele parameter. **'cfg'** is een trefwoord en kan elk zijn van de volgende: **OFF** **Niet** geconfigureerd of inactief. **AIN** **Analoge ingang** (d.w.z. meet de spanning op de ingang). **ARAW** **SNELLE analoge ingang** geeft een waarde terug van **0 tot 1023**. Voorbeeld: **> SETPIN Pinno/GPno, ARAW** **> ? Pin(Pinno/GPno)** **DIN** **Digitale ingang.** Als **'optie'** wordt weggelaten, heeft de ingang een hoge impedantie. Als **'optie'** het sleutelwoord **"PULLUP"** is, zal een gesimuleerde weerstand worden gebruikt om de invoerpin naar **3,3 V** te trekken. Als het sleutelwoord **"PULLDOWN"** wordt gebruikt, de pin wordt naar **nul volt** getrokken. De **pull up/down** is een constante stroom van ongeveer **50 µA.** **FIN** **Frequentie-ingang.** **'optie'** kan worden gebruikt om de poorttijd (de tijdsduur gebruikt om de invoercycli te tellen). Het kan elk getal tussen **10 ms** en **100000 ms** zijn. Merk op dat de functie **PIN()** altijd retourneer de **frequentie** correct geschaald in **Hz,** ongeacht de poort tijd gebruikt. Als **'optie'** wordt weggelaten, is de gate tijd **1 seconde.** **PIN** **Periode invoer.** **'optie'** kan worden gebruikt om het aantal invoercycli te specificeren het gemiddelde van de periodemeting over. Het kan elk nummer zijn tussen **1** en **10000**. Merk op dat de functie **PIN()** retourneer altijd de gemiddelde periode van één cyclus correct geschaald in **ms,** ongeacht het aantal cycli dat voor het gemiddelde is gebruikt. Als **'optie'** wordt weggelaten, wordt de periode van slechts één cyclus gebruikt. De pinnen kunnen **GP6, GP7, GP8** of **GP9** zijn (kan worden **gewijzigd met OPTION COUNT).**

CIN **Tellende invoer.**
'optie' kan worden gebruikt om aan te geven welke flank de telling activeert.
Aantal voor **SETPIN**-pin, **CIN** verhoogd naar **64**-bits.
1 specificeert een stijgende flank
2 een dalende flank en
3 specificeert dat beide flanken moeten worden geteld.
Als **'optie'** wordt weggelaten, zal een stijgende flank activeren de telling.
Vanwege een bug in de **RP2350** chips wordt **pulldown** niet aanbevolen.
De pinnen kunnen **GP6, GP7, GP8** of **GP9** kunnen worden gewijzigd met **OPTION COUNT**.

DOUT **Digitale uitgang.**
'optie' kan **"OC"** zijn voor MMBasic, maar de **RP2040** heeft geen **"OC"** dus deze optie niet gebruiken.
De functies **PIN()** en **PORT()** kunnen ook worden gebruikt om de waarde op één of meeruitgangspinnen te retourneren.
Eerdere versies van MMBasic gebruikten nummers voor **'cfg'** en de modus **OOUT**.

Voor achterwaartse compatibiliteit worden ze nog steeds herkend.
Zie de functie **PIN()** voor het lezen van invoer en het statement **PIN() =** voor een uitgang instellen. Zie de onderstaande opdracht als een interrupt is geconfigureerd.

SETPIN pin, cfg, target [, optie]

Zal 'pin' configureren om een interrupt te genereren volgens 'cfg'.
Elke **I/O pin** geschikt voor digitale invoer kan worden geconfigureerd om een onderbreking te genereren met een maximum van tien interrupts tegelijk geconfigureerd.

'cfg' is een trefwoord en kan een van de volgende zijn:
OFF Niet geconfigureerd of inactief.
INTH Interrupt op lage naar hoge input.
INTL Interrupt op hoge naar lage input.
INTB Interrupt op beide (d.w.z. elke wijziging aan de invoer).

'target' is een door de gebruiker gedefinieerde subroutine die wordt aangeroepen wanneer de gebeurtenis gebeurt. Terugkeer van de interrupt is via de **END SUB** of **EXIT SUB** commando's.

'optie' kunnen de trefwoorden **"PULLUP"** of **"PULLDOWN"** zijn als gespecificeerd voor een normale invoerpin (**SETPIN** pin **DIN**).
Als **'optie'** wordt weggelaten, wordt zal de ingang een hoge impedantie hebben. **RP2350 PULLDOWN** niet gebruiken ivm bug.
Deze mode configureert de pin ook als een digitale ingang, zodat de waarde van de pin kan worden opgevraagd met de functie **PIN()**
Raadpleeg het hoofdstuk **"Gebruik van de I/O-pinnen"** voor een algemene beschrijving van de **PicoMite's input/output** mogelijkheden

COMMANDO	Beschrijving 243
SETPIN GP25, DOUT | HEARTBEAT Zie ook: **OPTION HEARTBEAT ON | OFF**	**Deze versie van SETPIN stuurt de on-board LED aan.** Als het is geconfigureerd als **DOUT**, kan het worden in- en uitgeschakeld onder programma controle. Indien geconfigureerd als **HEARTBEAT**, **knippert** het continu **1s aan, 1s uit** terwijl aangedreven. Dit is de standaardstatus en wordt hiernaar hersteld wanneer de gebruikersprogramma stopt met werken.
SETPIN p1[, p2[, p3]], device	**Deze commando's worden gebruikt voor de pintoewijzing voor speciale apparaten.** Pinnen moeten worden gekozen uit het pin-aanduidingsdiagram en moeten worden toegewezen voordat de apparaten kunnen worden gebruikt. Merk op dat de pinnen (bijv. **rx, tx,** enz.) in willekeurige volgorde gedeclareerd en dat naar de pinnen kan worden verwezen met hun pin nummer (bijv. **2, 4**) of GP-nummer (bijv. **GP0, GP1**). Houd er rekening mee dat in de WebMite-versie: • SPI1 en SPI2 zijn niet beschikbaar op GP20 tot GP28 • COM1 en COM2 zijn niet beschikbaar op P20 tot GP28 • I2C is niet beschikbaar op pin 34 (GP28) • Het volgende is niet beschikbaar; GP29, GP25, GP24 en GP2
SETPIN rx, tx, COM1	**Wijs de pinnen toe die moeten worden gebruikt voor seriële poort COM1.** Geldige pinnen zijn: **RX:GP01,GP13,GP17** **TX:GP00,GP12,GP16,GP28**
COMMANDO	**Beschrijving** 243
SETPIN rx, tx, COM2	**Wijs de pinnen toe die moeten worden gebruikt voor seriële poort COM2**. Geldige pinnen zijn: **RX:GP05,GP09,GP21** **TX:GP04,GP08,GP20**
SETPIN rx, tx, clk, SPI	**Wijs de pinnen toe die moeten worden gebruikt voor SPI poort SPI** Geldige pinnen zijn: **RX:GP00,GP04,GP16,GP20** **TX:GP03,GP07,GP19** **CLK:GP02,GP06,GP18**
SETPIN rx, tx, clk, SPI2	**Wijs de pinnen toe die moeten worden gebruikt voor SPI poort SPI2.** Geldige pinnen zijn: **RX:GP08,GP12,GP28** **TX:GP11,GP15,GP27** **CLK:GP10,GP14,GP26**
SETPIN sda, scl, I2C	**Wijs de pinnen toe die moeten worden gebruikt voor de I2C poort I2C**. Geldige pinnen zijn: **SDA:GP00,GP04,GP08,GP12,GP16,GP20,GP28** **SCL:GP01,GP05,GP09,GP13,GP17,GP21**
SETPIN sda, scl, I2C2	**Wijs de pinnen toe die moeten worden gebruikt voor de I2C poort I2C2**. **Geldige pinnen** zijn: **SDA:GP02,GP06,GP10,GP14,GP18,GP22,GP26** **SCL:GP03,GP07,GP11,GP15,GP19,GP27**

COMMANDO	Beschrijving 244
SETPIN pin, PWM [nx]	**Pin toewijzen aan PWMnx.** **'n'** is het PWM nummer (**0** tot **7**) en **'x'** en is het kanaal (**A** of **B**) **De setpin** kan worden gewijzigd totdat het PWM commando wordt gegeven. Op dat punt de pin wordt vergrendeld op PWM totdat **PWMn**, **OFF** wordt uitgegeven.
SETPIN pin, IR	**Wijs pinnen toe voor infraroodcommunicatie (IR) (kan elke pin zijn)**
SETPIN pin, PIOn	**Reserve-pin voor gebruik door een PIO0 of PIO1.** (zie **Bijlage E** voor PIO details).
SETPIN GP1, FFIN [,gate]	**ALLEEN RP2350** **Stelt GP1 in als snelle frequentie-ingang.** **Invoer** tot **CPU-snelheid /2** kan worden opgenomen. **'gate'** kan worden gebruikt om de gate-tijd te specificeren (de tijdsduur die wordt gebruikt om de invoercycli te tellen). Dit kan elk getal tussen **10 ms** en **100.000 ms** zijn. De **PIN()**-functie retourneert altijd de frequentie correct geschaald in **Hz**, ongeacht de gebruikte gate-tijd. Als **'optie'** wordt weggelaten, bedraagt de poorttijd **1 sec**. De functie gebruikt PWM-kanaal **0** om te tellen en is dus incompatibel met enig ander gebruik van dat PWM-kanaal.
SETTICK period, target [, nbr]	**Dit zal een periodieke interrupt (of "tick") instellen.** Er zijn **vier tick timers** beschikbaar (**'nbr' = 1, 2, 3 of 4**). **'nbr'** is optioneel en zo niet gespecificeerde timer nummer 1 zal worden gebruikt. De tijd tussen onderbrekingen is **'periode'** milliseconden en **'doel'** is de interrupt subroutine die wordt aangeroepen wanneer de getimede gebeurtenis plaatsvindt. De periode kan variëren van **1** tot **2147483647 ms**. (ongeveer **24 dagen**). Deze onderbrekingen kunnen worden uitgeschakeld door **'periode'** op nul te zetten. (d.w.z. `SETTICK 0, 0, 3` schakelt **tick timer nummer 3 uit**).
SETTICK PAUSE, doel [, nbr] of SETTICK RESUME, doel [, nbr]	**PAUSE of RESUME de opgegeven timer.** Bij **pauze** is de interrupt vertraagd maar de huidige telling blijft behouden.
SORT array() [, indexarray()] [, vlaggen] [, startpositie] [, elemententosort]	**Deze opdracht neemt een array van elk type (integer, float of string) en sorteert deze in oplopende volgorde geplaatst.** Het heeft een optionele parameter **'indexarray%()'**. Indien gebruikt, moet dit een geheel getal zijn array van dezelfde grootte als de array die moet worden gesorteerd. Na de sortering zal deze array bevatten de originele indexpositie van elk element in de array die wordt gesorteerd voordat het was gesorteerd. Alle gegevens in de array worden overschreven. Dit maakt het mogelijk verbonden arrays die moeten worden gesorteerd. Zie de sectie Gegevens sorteren in de tutorial.

De parameter **'vlag'** is optioneel en geldige vlagwaarden zijn:

bit0: **0** (standaard indien weggelaten) normale sortering
1 omgekeerde sortering.
bit1: **0** (standaard) hoofdletterafhankelijk
1 sortering is hoofdletteronafhankelijk (alleen strings).
bit2: **0** (standaard) normale sortering
1 lege tekenreeksen gaan naar het einde van de array

De optionele **'startposition'** definieert welk element in de array de soort.
Standaard is **0 (OPTION BASE 0)** of **1 (OPTION BASE 1)**

De optionele **'elementstosort'** bepaalt hoeveel elementen in de array moet worden gesorteerd. De standaard zijn alle elementen na de startpositie.
Elk van de optionele parameters kan worden weggelaten, bijvoorbeeld om gewoon te sorteren de eerste 50 elementen van een array die je zou kunnen gebruiken: `SORT array(),,,, 50`
Voorbeeld:
De **array city$()** kan de namen van wereldsteden bevatten en dat kan eenvoudig gesorteerd in oplopende alfabetische volgorde met het commando: **SORT city$()**

Het **SORT** commando werkt met **Strings**, **Floats** en **Integer** getallen, maar de te sorteren array moet enkeldimensionaal zijn.
Vaak worden gegevens in meerdere arrays bewaard, bijvoorbeeld de naam van elke stad in de **array city$()**, de populatie in de **array pop %()** en de grootte van de stad in **gebied!()**. Dezelfde index zou verwijzen naar de naam, bevolking en het gebied van de stad.
Het sorteren en openen van deze gegevens is iets ingewikkelder, maar het kan wel relatief eenvoudig met behulp van een optionele parameter voor de sorteeropdracht als volgt:
SORT array(), indexarray%()
indexarray%() moet een integer-array met één dimensie zijn van dezelfde grootte als de array wordt gesorteerd. Na het sorteren zal **indexarray%()** de corresponderende index met de originele gegevens voordat deze werden gesorteerd. (iets die eerder in **indexarray%()** stond, wordt overschreven).
Om toegang te krijgen tot de gesorteerde gegevens kopieert u eerst de array met de hoofdsleutel naar een tijdelijke array en sorteer die terwijl u indexarray%() opgeeft. Na de **Sort** **indexarray%()** kan worden gebruikt om de originele arrays te indexeren.

	Bijvoorbeeld: `DIM city$(100), pop%(100), area!(100), sindex%(100), t$(100)` `FOR i = 0 to 100` `  t$(i) = city$(i) ' Tijdelijke kopie van de sleutels` `NEXT i` `SORT t$(), sindex%() ' Sorteer de tijdelijke array,` `FOR i = 0 to 100` `  k = sindex%(i)  ' Index naar de originele array.` `  PRINT city$(k), pop%(k), area!(k)` `  ' In gesorteerde volgorde afdrukken.` `NEXT i`
`SPI OPEN snelheid, mode, bits` of `SPI READ nbr,array()` of `SPI WRITE nbr, data1, data2, data3, ... enz` of `SPI WRITE nbr, string$` of `SPI WRITE nbr, array()` `SPI CLOSE`	**Communicatie via een SPI kanaal. Zie Bijlage D voor de details.** **'nbr'** is het aantal gegevensitems dat moet worden verzonden of ontvangen **'data1', 'data2'**, etc kunnen **float** of **integer** zijn en in het geval van **WRITE** kan het een constante of uitdrukking zijn. Als **'string$'** wordt gebruikt, worden **'nbr'** tekens verzonden. **'array'** moet een float- of integer array met één dimensie zijn en **'nbr'** elementen worden verzonden of ontvangen.
`SPI2`	**Dezelfde reeks opdrachten als voor SPI (hierboven) maar van toepassing op de tweede SPI kanaal..**
`SPRITE`	**ALLEEN VGA- EN HDMI VERSIES** **De SPRITE-opdrachten worden gebruikt om kleine grafische afbeeldingen op het VGA- of HDMI-scherm te manipuleren en zijn handig bij het schrijven van games.** Sprites werken alleen in framebuffers in **MODE 2** en **3**. Sprites worden altijd opgeslagen als RGB121 **'nibbles'** voor efficiëntie. De maximale grootte van een sprite is **MM.HRES - 1** en **MM.VRES - 1** Zie ook de **BLIT** opdracht en **SPRITE()** functies.
`SPRITE CLOSE [#]n`	**Sluit sprite "n" en geeft zijn geheugenbronnen vrij, zodat het sprite nummer hergebruikt kan worden.** De opdracht geeft een foutmelding als andere sprites van deze worden gekopieerd, tenzij ze eerst worden gesloten.
`SPRITE CLOSE ALL`	**Sluit alle sprites en geeft al het spritegeheugen vrij. Het scherm wordt niet gewijzigd.**
`SPRITE COPY [#]n, [#]m, nbr`	**Maakt een kopie van sprite "n" naar "nbr" van nieuwe sprites die beginnen met een nummer "m".** Gekopieerde sprites delen dezelfde geladen afbeelding als het origineel om geheugen te besparen.

COMMANDO	Beschrijving
`SPRITE HIDE [#]n`	**Verwijdert sprite n van het scherm en vervangt de opgeslagen achtergrond.** Om een scherm te herstellen naar een eerdere staat, moeten sprites worden verborgen in de omgekeerde volgorde waarin ze zijn geschreven **"LIFO"**
`SPRITE HIDE ALL`	**Verbergt alle sprites, zodat de achtergrond kan worden gemanipuleerd.** De volgende opdrachten kunnen niet worden gebruikt wanneer alle sprites zijn verborgen: `SPRITE SHOW (SAFE)` `SPRITE HIDE (SAFE, ALL)` `SPRITE SWAP` `SPRITE MOVE` `SPRITE SCROLLR` `SPRITE SCROLL`
`SPRITE RESTORE`	**Herstelt de sprites die eerder waren verborgen met:** `SPRITE HIDE ALL`
`SPRITE HIDE SAFE [#]n`	**Verwijdert sprite n van de weergave en vervangt de opgeslagen achtergrond.** Verbergt automatisch alle recente sprites en de gevraagde sprite en vervangt ze daarna. Dit zorgt ervoor dat sprites die worden bedekt door andere sprites kunnen worden verwijderd zonder dat de gebruiker de schrijfvolgorde bijhoudt. Natuurlijk is deze versie minder performant dan de eenvoudige versie en mag alleen worden gebruikt als er een risico bestaat dat de sprite gedeeltelijk wordt bedekt.
`SPRITE INTERRUPT Sub`	**Geeft de naam op van de subroutine die wordt aangeroepen wanneer er een sprite botsing plaatsvindt**. Zie **Bijlage H** voor het gebruik van de functie `SPRITE` om details te ondervragen van wat er is gebotst.
`SPRITE READ [#]b, x, y, w, h`	**Hiermee wordt een deel van de weergave gekopieerd naar de geheugenbuffer '#b'.** De broncoördinaat is **'x'** en **'y'** en de breedte van het te kopiëren weergavegebied is **'w'** en de hoogte is **'h'**. Wanneer deze opdracht wordt gebruikt, wordt de geheugenbuffer automatisch gemaakt en wordt er voldoende geheugen toegewezen. Deze buffer kan worden vrijgegeven en het geheugen kan worden hersteld met de opdracht `SPRITE CLOSE`.
`SPRITE WRITE [#]b, x, y`	**Kopieert sprite '#b' naar het scherm.** De bestemmingscoördinaat is **'x'** en **'y'** en de breedte/hoogte van de te kopiëren buffer is **'w'** en **'h'**. De optionele **'mode'** parameter is standaard **4** en specificeert hoe de opgeslagen afbeeldingsgegevens worden gewijzigd terwijl deze worden weggeschreven. Het is de bitgewijze `AND` van de volgende waarden: `&B001 = Gespiegeld van links naar rechts.` `&B010 = Gespiegeld van boven naar beneden.` `&B100 = Transparante pixels niet kopiëren.`

COMMANDO	Beschrijving 248
SPRITE LOAD fname$ [,start_sprite_number] [,mode]	**Laadt het bestand 'fname$' dat moet worden geformatteerd als een origineel Colour Maximite spritebestand.** Zie de originele Colour Maximite MMBasic Language Handleiding voor het bestandsformaat Meerdere spritebestanden kunnen worden geladen door voor elk bestand een ander **'start_sprite_number'** op te geven. De programmeur is verantwoordelijk voor het ervoor zorgen dat de sprites elkaar niet overlappen. De standaardinstelling voor de modus is nul, in welk geval de **CMM1**, **CMM2** kleurcodes worden gebruikt: (Zwart, Blauw, Groen, Cyaan, Rood, Magenta, Geel, Wit, Mirte, Kobalt, Middengroen, Cerulean, Roest, Fuchsia, Bruin, Lila): (ENG) Als de modus is opgegeven als 1, worden de RGB121 kleurcodes gebruikt: (Zwart, Blauw, Mirte, Kobalt, Middengroen, Cerulean, Groen, Cyaan, Rood, Magenta, Roest, Fuchsia, Bruin, Lila, Geel, Wit) (ENG)
SPRITE LOADARRAY [#]n, w, h, array%()	**Maakt de sprite 'n' met breedte 'w' en hoogte 'h' door w*h RGB888 waarden te lezen uit 'array%()'.** De **RGB888** waarden moeten worden opgeslagen in de volgorde van kolommen over en vervolgens rijen naar beneden, beginnend linksboven. Hierdoor kan de programmeur eenvoudige sprites in een programma maken zonder ze van de schijf te hoeven laden of van het scherm te hoeven lezen. De firmware genereert een fout als **'array%()'** niet groot genoeg is om het vereiste aantal waarden te bevatten.
SPRITE LOADBMP [#]b, fname$ [,x] [,y] [,w] [,h]	**Laadt een blit buffer van een 24-bits bmp imagebestand.** **'x'** en **'y'** definiëren de startpositie in de afbeelding om te beginnen met laden en **'w'** en **'h'** specificeren de breedte en hoogte van het te laden gebied. Bijvoorbeeld: `BLIT LOAD #1,"image1", 50, 50, 100, 100` Laadt een gebied van **100** pixels in het vierkant met de linkerbovenhoek op **50, 50** van de afbeelding **image1.bmp**
SPRITE LOADPNG [#]b, fname$ [,transparent] [,alphacut]	**Laadt SPRITE nummer 'b' uit het png bestand 'fname$'.** Als er geen extensie is opgegeven, wordt .png automatisch toegevoegd aan de bestandsnaam. Het bestand moet in RGBA8888 formaat zijn, wat de normale standaard is. De optionele parameter **'transparent'** (standaard **0**) specificeert een van de kleurcodes (**0**-**15**) die worden toegewezen aan pixels in het png bestand met een alfawaarde lager dan **'alphacut'** (standaard **20**). De variabele transparency kan vervolgens worden gebruikt met de opdracht: **SPRITE SET TRANSPARENT n** of **FRAMEBUFFER LAYER n** om de sprite weer te geven met het transparante gebied verborgen.
SPRITE MOVE	**Acties een enkele atomaire transactie die alle sprites verplaatst die eerder een locatiewijziging hebben gehad die is ingesteld met de op-dracht SPRITE NEXT.** Botsingen worden gedetecteerd zodra alle sprites zijn verplaatst en op dezelfde manier gerapporteerd als bij een scroll.

COMMANDO	Beschrijving 249
SPRITE NEXT [#]n, x, y	**Stelt de X- en Y-coördinaat van de sprite in die moet worden gebruikt wanneer het scherm de volgende keer wordt gescrolld of de opdracht SPRITE MOVE wordt uitgevoerd.** Met SPRITE NEXT in plaats van SPRITE **SHOW** kunnen meerdere sprites worden verplaatst als onderdeel van dezelfde atomische transactie.
SPRITE SCROLL x, y [,col]	**Scrollt de achtergrond en alle sprites op de actieve framebuffer (L of N) 'x' pixels naar rechts en 'y' pixels omhoog.** **'x'** kan elk getal zijn tussen MM.HRES - 1 en MM.HRES - 1, **'y'** kan elk getal zijn tussen MM.VRES - 1 en MM.VRES - 1. Sprites op elke laag anders dan nul blijven op hun vaste positie op het scherm. Standaard wikkelt de scroll de afbeelding rond. Als **'col'** is opgegeven, zal de kleur het gebied achter de gescrolde afbeelding vervangen. Als **'col'** is ingesteld op **-1**, zal het gescrolde gebied onaangetast blijven.
SPRITE SET TRANSPARENT n	**Stelt de kleurcode (0-15) in** die transparant wordt gebruikt wanneer sprites worden weergegeven over een achtergrond (standaard **0**).
SPRITE SHOW [#]n, x,y, layer [,options]	**Geeft sprite 'n' weer op het scherm met de linkerbovenhoek op de coördinaten 'x', 'y'.** Sprites botsen alleen met andere sprites op dezelfde laag, laag nul of met de rand van het scherm. Als een sprite al op het scherm wordt weergegeven, verplaatst de opdracht SPRITE **SHOW** de sprite naar de nieuwe locatie. De weergaveachtergrond wordt opgeslagen als onderdeel van de opdracht en wordt vervangen wanneer de sprite wordt verborgen of verder wordt verplaatst. De parameter **'options'** is optioneel en kan als volgt worden ingesteld: `bit 0 ingesteld - Gespiegeld van links naar rechts.` `bit 1 ingesteld - Gespiegeld van boven naar beneden.` `bit 2 ingesteld - Zwarte pixels worden niet als transparant behandeld standaard is 0.`
SPRITE SHOW SAFE [#]n, x,y, layer [,orientation][,ontop]	**Toont een sprite en compenseert automatisch voor andere sprites die deze overlappen.** Als de sprite nog niet wordt weergegeven, werkt de opdracht precies hetzelfde als SPRITE **SHOW**. Als de sprite al wordt weergegeven, wordt deze verplaatst en blijft deze in zijn positie ten opzichte van andere sprites op basis van de oorspronkelijke schrijfvolgorde. d.w.z. als sprite 1 is geschreven vóór sprite 2 en wordt verplaatst om sprite 2 te overlappen, wordt deze onder sprite 2 weergegeven. Als de optionele parameter **"ontop"** is ingesteld op 1, wordt de verplaatste sprite de nieuwste sprite en komt deze bovenop elke andere sprite te staan waarmee deze overlapt. Raadpleeg SPRITE **SHOW** voor meer informatie over de oriëntatie-parameter.

SPRITE SWAP [#]n1, [#]n2 [,orientation]	**Vervangt de sprite 'n1' door de sprite 'n2'.** De sprites moeten hetzelfde hebben breedte en hoogte en **'n1'** moeten worden weergegeven, anders wordt er een foutgegenereerd. Raadpleeg **SPRITE SHOW** voor details over de oriëntatieparameter. De vervangende sprite erft zowel de achtergrond van het origineel als de achtergrond ervan positie in de lijst met getrokken bestellingen.
STATIC variabele [, variabelen] Zie **DIM** voor de volledige syntaxis.	**Definieert een lijst met variabelenamen die lokaal zijn voor de subroutine of functie.** Deze **variabelen** behouden hun waarde tussen oproepen naar de subroutine of functie (in tegenstelling tot variabelen die zijn gemaakt met de opdracht **LOCAL**). Deze opdracht gebruikt exact dezelfde syntaxis als **DIM**. Het enige verschil is dat de lengte van de variabelenaam gecreëerd door **STATIC** en de lengte van de subroutine of functienaam bij elkaar opgeteld mag niet langer zijn dan **32 tekens**. Statische variabelen kunnen worden geïnitialiseerd op een waarde. Deze initialisatie duurt alleen effect bij de eerste oproep naar de subroutine (niet bij volgende oproepen).
SUB xxx (arg1 [, arg2, ...]) **<statements>** **<statements>** **END SUB**	**Definieert een oproepbare subroutine.** Dit is hetzelfde als het toevoegen van een nieuw commando aan MMBasic terwijl uw programma wordt uitgevoerd. **'xxx'** is de naam van de subroutine en deze moet voldoen aan de specificaties voor het benoemen van een variabel. **'arg1', 'arg2'**, etc zijn de argumenten of parameters voor de subroutine. Een array wordt gespecificeerd door lege haakjes te gebruiken. d.w.z. **arg3()**. Het type argument kan worden gespecificeerd door een typeachtervoegsel te gebruiken (d.w.z. **arg1$**) of door het type op te geven met **AS <type> (**d.w.z. **arg1 AS STRING)**. Elke definitie moet één **END SUB** statement hebben. Wanneer dit is bereikt het programma keert terug naar de volgende instructie na de aanroep van de subroutine. Het commando **EXIT SUB** kan gebruikt worden voor een vroege exit. Je gebruikt de subroutine door zijn naam en argumenten in een programma te gebruiken, net zoals bij een normaal commando. Bijvoorbeeld: **MySub a1, a2** Wanneer de subroutine wordt aangeroepen, komt elk argument in de aanroeper overeen met het argument in de subroutinedefinitie. Deze argumenten zijn alleen beschikbaar binnen de subroutine. Subroutines kunnen worden aangeroepen met een variabel aantal argumenten. Alle weggelaten argumenten in de lijst van de subroutine worden op nul gezet of een null-string.

	Argumenten in de lijst van de beller die een variabele zijn en van het juiste type zijn wordt doorgegeven met verwijzing naar de subroutine. Dit betekent dat eventuele wijzigingen naar het corresponderende argument in de subroutine wordt ook gekopieerd naar de variabele van de aanroeper en kan daarom worden geopend nadat de subroutine heeft eindigde. Arrays worden doorgegeven door de naam van de array op te geven met lege haakjes (bijv. arg()) en worden altijd als referentie doorgegeven. Beugels rond de argumentenlijst in zowel de aanroeper als de definitie zijn optioneel.
SYNC time% [,periode] of	**Het SYNC commando stelt de gebruiker in staat zeer nauwkeurig getimede herhaalde acties uit te voeren(1-2 microseconden nauwkeurigheid).** Om dit mogelijk te maken wordt eerst het commando aangeroepen met de parameter **time%**. Dit stelt een herhalende klok in voor **tijd%** microseconden. De optionele parameter **'periode'** wijzigt de tijd en kan **"U"** zijn voor microseconden, **"M"** voor milli-seconden of **"S"** voor seconden.
SYNC	Zodra de klok is ingesteld, stelt u het programma in wordt ermee gesynchroniseerd met behulp van het SYNC commando zonder parameters. Hiermee wordt gewacht tot de klokperiode is verstreken. Voor perioden van minder dan 2 milliseconden is dit niet onderbreekbaar. Boven de twee milliseconden reageert het programma op **Ctrl-C**, maar niet op MMBasic interrupts. Typisch gebruik is om de klok buiten een loop in te stellen en vervolgens bovenaan de lus het SYNC commando zonder parameters aan te roepen. Dit betekent dat de inhoud van de loop precies één keer wordt uitgevoerd voor elke ingestelde klokperiode. Bijvoorbeeld, zou een **Servo** aansturen met de vereiste nauwkeurige 50Hz timing: `SYNC 20, M` `DO` `  SYNC` `  PULSE GP0, n` `LOOP`
TEMPR START pin [, precisie]	**Dit commando kan worden gebruikt om een conversie te starten die wordt uitgevoerd op een DS18B20 temperatuursensor aangesloten op 'pin'**. Normaal gesproken is de functie TEMPR**()** alleen voldoende om een temperatuur meting te maken, dus het gebruik van deze opdracht is optioneel. Dit commando start de meting op de temperatuursensor. Het programma kan dan andere taken uitvoeren terwijl de meting loopt en gebruik later de functie TEMPR**()** om de meting te krijgen. Als de TEMPR**()** functie wordt gebruikt voordat de conversietijd is verstreken, zal de functie wacht op de resterende conversietijd voor-dat u de waarde retourneert.

	Een willekeurig aantal van deze conversies (op verschillende pinnen) kan worden gestart en gelijktijdig lopen. **'precisie'** is de resolutie van de meting en is **optioneel**. Het is een getal tussen **0** en **3** betekenis: **0** = **0.5 °C** Resolutie, **100ms** conversietijd. **1** = **0.25 °C** Resolutie, **200ms** conversietijd (dit is de standaard) **2** = **0.125 °C** Resolutie, **400ms** conversietijd. **3** = **0.0625 °C** Resolutie, **800ms** conversietijd.
TEXT x, y, string$ [,alignment$] [, font][, scale] Numerieke [, c] [, bc]	**Geeft een tekenreeks weer op een aangesloten LCD-paneel, beginnend bij 'x' en 'y'.** **'string$'** of **"Test"** is de tekenreeks die moet worden weergegeven. Numerieke gegevens moeten worden geconverteerd naar een string en geformatteerd met behulp van de **Str$()** functie. **'alignment$'** is een tekenreeksuitdrukking of tekenreeksvariabele bestaande uit 0, 1 of 2 letters waarbij de eerste letter de horizontale uitlijning rond **'x'** is en kan zijn **L, C** of **R** voor **LEFT, CENTER, RIGHT** en de tweede letter is de verticale uitlijning rond **'y'** en kan **T, M** of **B** zijn voor **BOVEN, MIDDEN, ONDER**. De standaarduitlijning is links/boven. Bijvoorbeeld. **"CM"** centreert de tekst verticaal en horizontaal. De tekenreeks **'alignment$'** kan een constante zijn (bijv. **"CM"**) of het kan een tekenreeks zijn variabel. Voor achterwaartse compatibiliteit met eerdere versies van MMBasic de string kan ook zonder aanhalingstekens zijn (bijv. CM). In de PicoMite kan een derde letter in de uitlijnreeks worden gebruikt om aan te geven de rotatie van de tekst. Dit kan **'N'** zijn voor normale oriëntatie, **'V'** voor verticale tekst waarbij elk teken onder het vorige van boven naar onderaan loopt, **'I'** de tekst wordt omgekeerd (d.w.z. ondersteboven), **'U'** de tekst wordt **90º** tegen de klok in gedraaid en met **'D'** wordt de tekst **90º** met de klok mee gedraaid. **'font'** en **'scale'** zijn optioneel en standaard ingesteld op het **FONT** commando. **'c'** is de kleur van de tekening en **'bc'** is de achtergrondkleur. Ze zijn optioneel en standaard de huidige voor- en achtergrondkleuren. Zie het hoofdstuk **"Basis Tekenopdrachten"** voor een definitie van de kleuren en grafische coördinaten.

COMMANDO	Beschrijving
TILE x, y [,foreground] [,background] [,nbr_tiles_wide] [,nbr_tiles_high]	**VGA- OF HDMI-VERSIES ALLEEN MODE 1** **Stelt de kleur in voor een of meer tegels op het scherm.** In de monochrome modus wordt het videoscherm standaard opgesplitst in 80 x 40 tegels van elk 8 x 12 pixels. Dit komt overeen met lettertype 1 en maakt volledige kleurcodering mogelijk in de editor in monochrome modus. Elke tegel kan een andere voorgrond- en achtergrondkleur hebben toegewezen uit de volgende: wit, geel, lila, bruin, fuchsia, roest, magenta, rood, cyaan, groen, cerulean, middengroen, kobalt, mirte, blauw en zwart. **'x'** en **'y'** zijn de coördinaten van het startblok (**0**-**79**, **0**-**39**) **'voorgrond'** en **'achtergrond'** zijn de nieuwe geselecteerde kleuren. **'nbr_tiles_wide'** en **'nbr_tiles_high'** zijn het aantal tegels dat moet worden gewijzigd. De wijziging is onmiddellijk van kracht en heeft momenteel geen invloed op de tekst of afbeeldingen weergegeven in de tegels (alleen de kleuren).
TILE HEIGHT n	**Stelt de hoogte van de tegels in. 'n' kan tussen 12 en 480 (RP2040) of tussen 8 en 480 (RP2350) liggen** Bijvoorbeeld: `TILE HEIGHT 50` `TEXT 8, 50, "12:20:40",,,, rgb(red), rgb(green)` Kleur bewerking is nu beschikbaar in **Mode 1** met : **OPTION COLOURCODE ON** ingesteld.
TIME$ = "UU:MM:SS" of TIME$ = "UU:MM" of TIME$ = "HH"	**Stelt de tijd van de interne klok in. MM en SS zijn optioneel en worden standaard tot nul indien niet gespecificeerd.** Bijvoorbeeld `TIME$ = "14:30"` zal de klok instellen op **14:30** met nul seconden. De tijd is ingesteld op "00:00:00" bij het opstarten. Met **OPTION RTC AUTO ENABLE** start de PicoMite firmware met de **TIME$** die is geprogrammeerd in **RTC**. Zonder **OPTION RTC AUTO ENABLE** start de firmware met `TIME$ = "00:00:00"`
TIMER = msec	**Zet de timer terug op een aantal milliseconden.** Normaal gesproken wordt dit gewoon gebruikt om reset de timer naar nul, maar u kunt deze instellen op elk positief geheel getal. Zie de **TIMER** functie voor meer details.
TRACE ON of TRACE OFF	**TRACE ON/OFF zet de traceerfunctie aan/uit.** Deze faciliteit zal afdrukken het nummer van elke regel (gerekend vanaf het begin van het programma) in vierkante haken terwijl het programma wordt uitgevoerd. Dit is handig bij het debuggen programma's. Het **TRACE** commando is nu op de hoogte van de bibliotheek. De test voor de bibliotheek wordt gecorrigeerd wanneer `TRACE ON` wordt toegepast, voor **TRACE LIST** wordt **[Lib]** weergegeven voor het regelnummer in de bibliotheek. (dwz het hoeft niet ingeschakeld te zijn).

COMMANDO	Beschrijving 254
TRACE LIST nn	**TRACE LIST toont de laatste 'nn' regels die zijn uitgevoerd in het beschreven formaat boven.** MMBasic logt altijd de uitgevoerde regels, dus deze faciliteit is altijd beschikbaar (d.w.z. het hoeft niet aan te staan). Voor TRACE LIST wordt [Lib] weergegeven voor het regelnummer in de bibliotheek.
TRIANGLE X1, Y1, X2, Y2, X3, Y3 [, C [, FILL]]	**Tekent een driehoek op het LCD scherm met de hoeken op X1, Y1 en X2, Y2 en X3, Y3.** **'C'** is de kleur van de driehoek en is standaard de huidige voorgrond-kleur. **'FILL'** is de vulkleur en is standaard geen vulling (het kan ook worden ingesteld op **-1** voor geen vulling). Alle parameters kunnen worden uitgedrukt als arrays en de software plot de aantal driehoeken zoals bepaald door de afmetingen van de kleinste array tenzij **X1 = Y1 = X2 = Y2 = X3 = Y3 = -1** In dit geval zal de verwerking stoppen op het punt **'x1', 'y1', 'x2', 'y2', 'x3', 'y3'** en moeten allemaal arrays zijn of allemaal enkele varia-belen / constanten anders wordt er een fout gegenereerd **'c'** en **'fill'** kunnen arrays of enkele variabelen/constanten zijn.
TRIANGLE RESTORE [#]n	**Herstelt een opgeslagen driehoekig gebied van het scherm en ver-wijdert de opgeslagen buffer.**
TRIANGLE SAVE [#]n, x1, y1, x2, y2, x3, y3	**Slaat een driehoekig gebied van het scherm op in buffer #n.** Werkt alleen op **VGA** en beeldschermen die BLIT en transparante tekst ondersteunen.
UPDATE FIRMWARE	**Zorgt ervoor dat de PicoMite naar de firmware updatemodus gaat (hetzelfde als het inschakelen van de stroom terwijl u de BOOTSEL knop ingedrukt houdt).** Het laden van de PicoMite firmware zal het flashgeheugen wissen, inclusief de huidige programma, alle programma's die zijn opgesla-gen in flash geheugenslots en alle opgeslagen variabelen. Zorg er dus voor dat u een backup maakt van deze gegevens voordat u de firm-ware upgrade uitvoert. Bij het laden van de firmware worden ook alle opties teruggezet naar hun standaardwaarden. Dit is hetzelfde als het omzetten van de **BOOTSEL** drukknop op het Raspberry Pi Pico CPU-bord naar **SYSTEM** en maakt snelle firmware-updates mogelijk zonder de behuizing van de PicoMite te openen. U dient de stroom uit en weer aan te zetten na een firmware update of als deze opdracht per ongeluk is gebruikt. Hierdoor keert de RP2040 of RP2350 CPU terug naar de normale modus.
VAR SAVE var [, var]	**VAR SAVE slaat een of meer variabelen op in niet vluchtig flashge-heugen waar ze later kunnen worden hersteld (normaal gesproken na een stroomonderbreking).** **'var'** kan een willekeurig aantal numerieke of tekenreeksvariabelen en/of arrays zijn. Arrays worden gespecificeerd door lege haakjes te gebruiken. Bijvoorbeeld: Var()

COMMANDO	Beschrijving 255
VAR RESTORE	**VAR RESTORE haalt de eerder opgeslagen variabelen op en voegt ze in (en hun waarden) in de variabelentabel.** De opdracht VAR SAVE kan herhaaldelijk worden gebruikt. Variabelen die waren eerder opgeslagen worden bijgewerkt met hun nieuwe waarde en eventuele nieuwe variabelen(niet eerder opgeslagen) worden toegevoegd aan de opgeslagen lijst voor later herstel.
VAR CLEAR	**VAR CLEAR wist alle opgeslagen variabelen.** Ook zullen de opgeslagen variabelen automatisch worden gewist door een firmware upgrade, door het NEW commando of wanneer een nieuw programma wordt geladen via AUTOSAVE, XMODEM, enz Deze opdracht wordt normaal gesproken gebruikt om kalibratiegegevens, opties en andere gegevens die niet vaak veranderen, maar over een machtsgebied moeten worden bewaard onderbreking. Normaal gesproken wordt het VAR RESTORE commando aan het begin geplaatst van het programma zodat eerder opgeslagen variabelen worden hersteld en onmiddellijk beschikbaar voor het programma wanneer het start. Opmerkingen: • De beschikbare opslagruimte voor deze opdracht is 16 KB. • Het gebruik van VAR RESTORE zonder een eerdere opslag heeft geen effect en zal geen fout genereren. • Als er bij gebruik van RESTORE al een variabele met dezelfde naam bestaat de waarde ervan wordt overschreven. • Opgeslagen arrays moeten worden gedeclareerd (met DIM) voordat ze kunnen worden hersteld. • Houd er rekening mee dat stringarrays snel al het toegewezen geheugen kunnen gebruiken aan dit commando. De LENGTH-kwalificatie kan worden gebruikt wanneer een string array wordt gedeclareerd om de grootte van de array te verkleinen (zie de opdracht DIM). Dit is niet nodig voor gewone stringvariabelen. • De opgeslagen variabelen worden automatisch gewist door een firmware upgrade, door het NEW commando of wanneer er een **nieuw programma is geladen via AUTOSAVE, XMODEM, enz.**
WATCHDOG timeout	**Start de watchdog-timer die de processor automatisch opnieuw start wanneer het is verstreken**. Dit kan worden gebruikt om te herstellen van een gebeurtenis die het lopende programma uitge-schakeld (zoals een eindeloze lus of een programmeer- of andere fout die een lopend programma stopt). Dit kan belangrijk zijn in een onbeheerde controlesituatie.

COMMANDO	Beschrijving 256
	'timeout' is de tijd in **milliseconden (ms)** voordat een herstart wordt geforceerd. Deze commando moet op strategische locaties worden geplaatst in de draaiende BASIC programma om de watchdog timer constant te resetten (naar 'timeout') en dus voorkomen dat het aftelt tot nul. Als de timertelling nul bereikt (misschien omdat het BASIC programma is gestopt met werken) wordt de PicoMite automatisch opnieuw gestart en de automatische variabele **MM.WATCHDOG** wordt ingesteld op **true (d.w.z. 1)** wat aangeeft: dat er een fout is opgetreden. Bij normaal opstarten wordt **MM.WATCHDOG** ingesteld op **onwaar (d.w.z. 0)**. Merk op dat **OPTION AUTORUN** moet worden gespecificeerd voor het programma opnieuw te starten.
WATCHDOG OFF	**WATCHDOG OFF kan worden gebruikt om de watchdog-timer uit te schakelen (dit is de standaard op een reset of opstarten).** De timer wordt ook **uitgeschakeld wanneer de pauze** teken(**CTRL-C**) wordt op de console gebruikt om een lopend programma te onderbreken.
WATCHDOG HW timeout	**De time-out kan worden verwerkt in de systeemtimer-interrupt (WATCHDOG-opdracht) of als een echte CPU/hardware-waakhond (WATCHDOG HW-opdracht).** Bij gebruik van de hardware watchdog bedraagt de timer maximaal **8,3 seconden**. Nee een dergelijke beperking bestaat voor de softwarewaakhond. **'time-out'** is de tijd in **milliseconden (ms)** voordat een herstart wordt dit geforceerd. Het commando moet op strategische locaties in de lopende BASIC programma worden geplaatst om de watchdog-timer voortdurend te resetten (naar **'time-out'**) en daarom te voorkomen dat het aftelt naar nul. Als de timertelling nul bereikt (misschien omdat het BASIC-programma dat heeft gedaan) wordt de PicoMite-firmware automatisch opnieuw opgestart en wordt de automatische variabele **MM.WATCHDOG** wordt ingesteld op **true (d.w.z. 1)**, wat aangeeft dat er is een fout opgetreden. Bij een normale start wordt **MM.WATCHDOG** ingesteld op **false (dat wil zeggen 0)**. Houd er rekening mee dat **OPTION AUTORUN** moet worden opgegeven voordat het programma dit opnieuw kan opstarten.
WATCHDOG HW OFF	**WATCHDOG OFF kan worden gebruikt om de watchdog-timer uit te schakelen (dit is de standaard bij een reset of opstarten).** De timer wordt ook uitgeschakeld tijdens het pauze teken **(CTRL-C)** wordt op de console gebruikt om een actief programma te onderbreken.

COMMANDO	Beschrijving 257
WII [CLASSIC] OPEN [,interrupt]	**Opent een WII Classic-controller en implementeert achtergrond-onderzoek van het apparaat.** De Wii Classic moet worden aangesloten op de door **OPTION** aangegeven pinnen **SYSTEM I2C**, wat een vereiste is. Open pogingen om met de **Wii Classic** te praten en er verschijnt een foutmelding als deze niet wordt gevonden. Als ontdekte dat de firmware de **Wii-gegevens** op de achtergrond zal samplen met een snelheid van **50 Hz**. Als er een optionele gebruikersonderbreking is opgegeven, wordt deze geactiveerd als een van de volgende wordt uitgevoerd en de knoppen veranderen (zowel aan als uit) Zie de **DEVICE**-functie voor het lezen van gegevens uit de **Wii Classic**.
WII [CLASSIC] CLOSE	**CLOSE stopt de polling op de achtergrond en schakelt elke gespecificeerde interrupt uit.**
WII NUNCHUCK OPEN [,interrupt][myint]	**Opent een WII Nunchuck-controller en implementeert achtergrondonderzoek van het apparaat.** De **Wii Nunchuck** moet worden aangesloten op de door **OPTION** aangegeven pinnen **SYSTEM I2C**, wat een vereiste is. Open pogingen om met de **Wii Nunchuck** te praten en als dat niet het geval is, krijg je een gevonden foutmelding. Indien gevonden, zal de firmware de Wii-gegevens op de achtergrond bemonsteren met een snelheid van **50 Hz**. Als er een optionele gebruikersonderbreking is opgegeven, wordt deze geactiveerd als een van de knoppen ver-andert (zowel aan als uit). Zie de **DEVICE** functie voor het lezen van gegevens van de **Wii Nunchuck.**
WII NUNCHUCK CLOSE	**CLOSE stopt de polling op de achtergrond en schakelt elke gespecificeerde interrupt uit.**
WEB	**ALLEEN WEBMITE** **Het WEB commando worden gebruikt om de internetmogelijkheden van de WebMite.**
WEB CONNECT [ssid$, passwd$, [name$][,ipaddress$, mask$, gateway$]]	**Deze opdracht, zonder optionele parameters, maakt verbinding met het standaard commando netwerk indien mogelijk (zoals eerder ingesteld met OPTION WIFI) of met de optionele parameters maken verbinding met het opgegeven netwerk en stellen ook de OPTION in WIFI voor toekomstig gebruik.**
WEB MQTT CONNECT addr$, port, user$, passwd$ [, interrupt]	**Maak verbinding met een MQTT-broker.** **'addr$'** is het **IP-adres.** **'port'** is het te gebruiken **poortnummer.** **'user$'** is de **gebruiker naam.** **'passwd$'** is het **wachtwoord van het account**

COMMANDO	Beschrijving
	'interrupt' is optioneel en als opgegeven is de subroutine die moet worden aangeroepen wanneer een bericht wordt ontvangen. **WEB CONNECT** verbreekt dus niet de verbinding met een eerder verbonden netwerk en mag alleen worden gebruikt als er nog niets eerder is ingesteld of als een eerder geconfigureerd netwerk niet actief is of een eerder geconfigureerd netwerk kan geen verbinding maken tijdens het opstarten (geen parameters)
WEB MQTT PUBLISH topic$, msg$, [,qos] [,retain]	**Publiceer inhoud naar een MQTT-brokeronderwerp.** **'topic$'** is de onderwerpnaam. **'msg$'** is het bericht. **'qos'** is de optionele servicekwaliteit met waarden van **0, 1** of **2** (**standaard is 1**).
WEB MQTT SUBSCRIBE topic$ [,qos]	**Abonneer u op een MQTT-brokeronderwerp.** **'topic$'** is de onderwerpnaam en **'qos'** is de optionele servicekwaliteit met waarden van **0, 1** of **2** (**standaard is 1**)
WEB MQTT UNSUBSCRIBE topic$	**Afmelden voor een MQTT-brokeronderwerp.** **'topic$'** is de onderwerpnaam.
WEB MQTT CLOSE	**Sluit een permanente MQTT-verbinding.**
WEB NTP [timeoffset [, NTPserver$]] [, timeout]]]	**Haal de datum/tijd op van een NTP-server en stel de interne Web-Mite datum/-tijd klok in.** **'timeoffset'** is de lokale tijdzone en als dit wordt weggelaten, wordt de datum/tijd ingesteld **GMT.** **'NTPserver$'** is de tijdserver die moet worden gebruikt en als deze wordt weggelaten, wordt deze standaard ingesteld op een internationale tijdserverpool. **'time-out'** is de optionele time-out in **milliseconden** en standaard ingesteld op **5000**.
WEB OPEN TCP CLIENT address$, port	**Opent een TCP-clientverbinding met een WEB-server.** **'address$'** is een string en is het adres van de server waarmee verbinding moet worden gemaakt. Het kan zijn ofwel een : **URL** (bijv. **"api.openweathermap.org"**) of **IP-adres** (bijv. **"192.168.1.111"**). **'poort'** is het nummer van de te gebruiken poort. Wordt gebruikt met **WEB TCP CLIENT REQUEST** om de server te ondervragen. Houd er rekening mee dat één **CLIENT** verbinding is toegestaan.

COMMANDO	Beschrijving
WEB OPEN TCP STREAM address$, port	**Opent een TCP-clientverbinding met een WEB-server.** Zoals **WEB OPEN TCP CLIENT** maar verbindt in plaats daarvan de **WEB TCP CLIENT STREAM** ontvangerlogica dan de logica voor **WEB TCP CLIENT REQUEST.** **'address$'** is een string en is het adres van de server waarmee verbinding moet worden gemaakt. Het kan zijn ofwel een: **URL (bijv. "api.openweathermap.org")** of **IP-adres (bijv. "192.168.1.111")**. **'poort'** is het nummer van de te gebruiken poort. Houd er rekening mee dat één **CLIENT** verbinding is toegestaan.
WEB SCAN [array%()]	**Scant naar alle beschikbare wifi-verbindingen.** Als **'array%()'** is opgegeven, wordt de uitvoer opgegeven wordt opgeslagen in een **Longstring**, anders wordt de uitvoer naar de console gestuurd. Het commando kan worden gebruikt, ongeacht of er al een netwerkverbinding actief is.
WEB TCP CLIENT REQUEST request$, buff%() [,timeout]	**Stuur een verzoek naar de externe server die is geopend met WEB OPEN TCP CLIENT en wacht op een antwoord.** **'request$'** is een string en is het verzoek dat naar de server moet worden verzonden. **'buff%()'** is een integer-array die het antwoord ontvangt als een **LONGSTRING.** De grootte van deze buffer beperkt de hoeveelheid gegevens die van de server wordt ontvangen. **'time-out'** is de optionele **time-out in milliseconden** en is standaard ingesteld op **5000**. Als er een time-out optreedt bij het verzoek, zal er een fout optreden, anders zullen de ontvangen gegevens dat wel zijn opgeslagen in de **LONGSTRING 'buff%()'**. Als de ontvangen gegevens een **JSON**-tekenreeks zijn dan kan de functie **JSON$()** worden gebruikt om het te parseren.
WEB TCP CLIENT STREAM command$, buffer%(), readpointer%, writepointer%	**Maakt verbinding met een server die eerder is geopend met WEB OPEN TCP STREAM.** **'command$'** is een string en is het verzoek dat naar de server moet worden verzonden. **'buffer%()'** is een integer-array die de lopende reacties zal ontvangen en fungeert als een circulaire buffer van ontvangen bytes. De firmware onderhoudt de parameter **'writepointer%'** als de gegevens van de server arriveert. **'readpointer%'** moet door het Basic-programma worden onderhouden terwijl het gegevens verwijdert uit de circulaire buffer. Als **'writepointer%' 'readpointer%'** inhaalt, dan zal **'readpointer%'** dat zijn verhoogd om één byte voorop te blijven en inkomende gegevens gaan verloren. Deze opdracht is ontworpen om compatibel te zijn met de **PLAY STREAM** opdracht om de implementatie van streaming internet-audio mogelijk te maken.

COMMANDO	Beschrijving 260
WEB CLOSE TCP CLIENT	**Sluit de verbinding met de externe server die is geopend met WEB OPEN TCP CLIENT.** Dit moet worden gedaan voordat er opnieuw wordt geprobeerd te openen.
WEB TCP INTERRUPT InterruptSub	**Start de TCP-server actief.** **'InterruptSub'** is de subroutine die moet worden aangeroepen wanneer er een verzoek wordt gedaan aan de **TCP-server** (dwz een interrupt). Houd er rekening mee dat het **OPTION WIFI** commando eerst moet zijn gebruikt, gevolgd door het **OPTION TCP SERVER PORT**-commando om de **TCP-server** in te schakelen.
WEB TCP READ cb%, buff%()	**Lees de gegevens van een potentiële TCP-verbinding 'cb%'.** **' buff%()'** is een array daarvoor ontvang alle gegevens van die verbinding als een longstring. De grootte van deze buffer beperkt de hoeveelheid gegevens die wordt ontvangen van de externe client. Als dat zo is niets ontvangen op die verbinding, dit zal een lege string retourneren (dat wil zeggen, **LLEN(verbetering%())=0)**. Als er gegevens zijn ontvangen, moet het BASIC-programma reageren met een van de **WEB TRANSMIT** commando om te reageren en de verbinding.
WEB TCP SEND cb%, data%() WEB TCP CLOSE cb%	**Deze twee opdrachten bieden meer flexibiliteit bij het gebruik van de TCP-server.** In tegenstelling tot **WEB TRANSMIT PAGINA** of **WEB TRANSMIT FILE**, **WEB TCP SEND** doet dat er wordt geen enkele header gemaakt en de **TCP-verbinding** wordt daarna ook niet afgesloten overdragen. Het verzendt gewoon precies wat er in de **LONGSTRING data%()** zit en ook is het aan de Basisprogrammeur om de verbinding te verbreken wanneer dat nodig is.
WEB TRANSMIT CODE cb%, nnn%	**Stuur een numeriek antwoord naar de open TCP-verbinding 'cb%' en sluit vervolgens de verbinding.** Typisch gebruik is **TRANSMIT CODE cb%, 404** om aan te geven dat pagina niet bestaat of gevonden is.
WEB TRANSMIT FILE cb%, filename$, content-type$	**Creëert een HTTP 1.1-header met het 'content-type$' .** Zoals gespecificeerd, en verzendt deze en verzendt vervolgens de inhoud van het bestand naar de open **TCP**-verbinding **cb%** en verder voltooiing, sluit de verbinding. **'content-type$'** is een **MIME**-type uitgedrukt als een string. Bijvoorbeeld **"afbeelding/jpeg"**
WEB TRANSMIT PAGE cb%, filename$[,buffersize]	**Creëert een HTTP 1.1-header, verzendt deze en verzendt vervolgens de inhoud van het bestand naar de open TCP-verbinding cb% en sluit na voltooiing de verbinding.** **MMBasic** vervangt de huidige waarden voor alle MMBasic-variabelen of expressies gedefinieerd in het bestand tussen accolades, bijvoorbeeld **{myvar%}**. Variabelen kan eenvoudig zijn, array-elementen of expressies.

	Een accolade openen kan in de uitvoer worden opgenomen met behulp van **{{**. Standaard wijst de opdracht een buffer toe ter grootte van het bestand **+ 4096 bytes** bouw de pagina om te verzenden. Als de pagina echter complex is en er veel bevat MMBasic-variabelen die tekst opleveren die groter is dan de naam van de variabele is mogelijk dat de buffer niet groot genoeg zal zijn. In dit geval kan de ge-bruiker de extra specificeren benodigde ruimte (standaard 4096 indien niet gespecificeerd)
`WEB UDP INTERRUPT intname`	**Stelt een BASIC-interruptroutine in die wordt geactiveerd wanneer er een UDP plaatsvindt datagram wordt ontvangen.** De inhoud wordt opgeslagen in **MM.MESSAGE$**. Het **IP adres** van de afzender wordt opgeslagen in **MM.ADDRESS$**.
`WEB UDP SEND addr$, port, data$`	**Wordt gebruikt om een datagram naar een externe ontvanger te sturen. In dit geval moet het IP-adres worden gespecificeerd en kan een numeriek adres zijn (bijvoorbeeld "192.168.1.147") of een normaal tekstadres (bijvoorbeeld 'google.com').** Het poortnummer van de ontvanger moet ook worden opgegeven en het bericht zelf. Het **SEND** commando kan worden gebruikt als een reactie op een binnenkomend bericht of standalone.
`WS2812 type, pin, nbr, value%[()]`	**Dit commando stuurt een of meer WS2812 LED-chips aan die op 'pin' zijn aangesloten.** Houd er rekening mee dat de pin moet worden ingesteld op een digitale uitgang voordat deze opdracht wordt gebruikt. **'type'** is een enkel teken dat het type chip specificeert dat wordt aangedreven: **O = origineel WS2812** **B = WS2812B** **S = SK6812** **W = SK6812W (RGBW)** **'nbr'** is het aantal **LED's** in de keten (**1 tot 256**). De array **'waarde%()'** moet een array met gehele getallen zijn, zo groot dat hij exact hetzelfde aantal elementen bevat als het aantal aan te sturen **LED's**. Voor de eerste drie varianten moet elk element in de array de kleur bevatten in het normale **RGB888** formaat (d.w.z. **0 tot &HFFFFFF**). Gebruik voor **type W** een **RGBW**-waarde (**0 - &HFFFFFFFF**). Als er slechts één LED is aangesloten, moet voor 'waarde%' een enkel geheel getal worden gebruikt (dat wil zeggen, geen array).
`XMODEM SEND`	**Brengt een BASIC programma over van of naar een externe computer met behulp van het XModem protocol.** De overdracht gebeurt via de USB consoleverbinding.
`XMODEM SEND file$`	**XMODEM SEND stuurt het huidige programma in de PicoMite's programmageheugen naar het externe apparaat.**

COMMANDO	Beschrijving 262
`XMODEM RECEIVE`	**XMODEM RECEIVE accepteert een programma verzonden door het externe apparaat apparaat en sla het op in het RAM-geheugen van de PicoMite overschrijven van het programma dat daar momenteel wordt gehouden.**
`XMODEM RECEIVE file$` `XMODEM R "test.jpg"`	**U kunt ook 'file$' opgeven die de gegevens van/naar een bestand op de SD-kaart zal overbrengen.** Als het bestand al bestaat, wordt het overschreven bij ontvangst van een bestand. Merk op dat de gegevens worden gebufferd in **RAM**, wat het maximale programma beperkt maakt. Deze opdracht maakt ook een backup van het programma in het flashgeheugen die automatisch wordt opgehaald als de CPU wordt gereset of de stroom uitvalt.
`XMODEM CRUNCH`	**De CRUNCH optie werkt als RECEIVE maar verwijdert alle opmerkingen, lege regels en onnodige spaties uit het programma voordat u het opslaat.** Deze kan worden gebruikt op grote programma's zodat ze in een beperkt geheugen passen. **SEND, RECEIVE** en **CRUNCH** kunnen worden afgekort tot **S, R** en **C.** Het **XModem** protocol vereist een samenwerkend softwareprogramma dat draait op de externe computer en aangesloten op de seriële poort. Het is getest op Tera Term draait op Windows en het wordt aanbevolen dit te gebruiken. Na het uitvoeren van de **XMODEM** opdracht in MMBasic selecteert u: **Bestand -> Overdragen -> XMODEM -> Ontvangen/verzenden** uit het Tera Term-menu om de overdracht te starten. Het kan tot 15 seconden duren voordat de overdracht start en als de **XMODEM** commando kan geen communicatie tot stand brengen, het keert terug naar de MMBasic prompt na 60 seconden en laat het programmageheugen ongemoeid. Download Tera Term van: http://ttssh2.sourceforge.jp **Voorbeeld1: Bestand naar PicoMite** (Basic programma versturen) Op de **PicoMite** zorg er voor de juiste **Com poort** is geselecteerd in **TeraTerm: Setup/Serieel** en sla dit op (Save Setup). Voer in op de PicoMite: `> XMODEM RECEIVE` Daarna het lijkt te hangen, maar wacht de **PicoMite** op de data. In Tera Term: **File menu/Transfer/XMODEM/Send.**.en maak in pop-upvenster de bestandsnaam keuze in de juiste map en kies voor Openen. Na even wachten verschijnt: Saved nnn bytes en is uw bestand geladen, en daarna **>** controle met **list** of **edit**. **Voorbeeld2:** Bestand van PicoMite naar Pc (Backup) Voer in op de PicoMite: `> XMODEM SEND` en het lijkt te hangen. In Tera Term: **File menu/Transfer/XMODEM/Receive..** en krijgt u een pop-upvenster waarin u kunt selecteren waar u het bestand wilt opslaan. Tera Term sneltoetsen voor bovenstaande : Verzenden: **Alt-F TXS** Ontvangen: **Alt-F TXR**

Functies

Merk op dat de functies met betrekking tot communicatiefuncties (**I2C**, **ONEWire** en **SPI**) hier niet worden vermeld, maar zijn beschreven in de **bijlagen aan het einde van dit document**.
Vierkante haken geven aan dat de parameter of tekens optioneel zijn.

FUNCTIES	Beschrijving 263
ABS(getal)	**Geeft als resultaat de absolute waarde van het argument 'getal'.** (d.w.z. elk negatief tekenwordt verwijderd en een positief getal wordt geretourneerd).
ACOS(getal)	**Geeft als resultaat de inverse cosinus van het argument 'getal' in radialen.**
ASC(string$)	**Retourneert de ASCII-code (d.w.z. bytewaarde) voor de eerste letter in 'string$'.**
ASIN(getal)	**Geeft de inverse sinuswaarde van het argument 'getal' in radialen.**
ATN(getal)	**Geeft als resultaat de arctangens van het argument 'getal' in radialen.**
ATAN2(y, x)	**Geeft als resultaat de boogtangens van de twee getallen x en y als een hoek uitgedrukt in radialen.** Het is vergelijkbaar met het berekenen van de **boogtangens** van **y / x**, behalve dat de tekens van beide argumenten worden gebruikt om het kwadrant van het resultaat te bepalen.
BIN$(getal [,chars])	**Geeft een tekenreeks terug die de binaire (grondtal 2) waarde voor het 'getal' geeft.** **'chars'** is optioneel en specificeert het aantal karakters in de string met nul als het leidende opvulteken(s).
BIN2STR$(type,waarde [,BIG])	**Geeft een tekenreeks terug die de binaire representatie van 'waarde' bevat.** **'type'** kan zijn: **INT64** Ondertekend 64-bits geheel getal geconverteerd naar een 8 byte string. **UINT64** Niet-ondertekend 64-bits geheel getal geconverteerd naar een tekenreeks van 8 bytes. **INT32** Ondertekend 32-bits geheel getal geconverteerd naar een 4-byte string. **UINT32** Unsigned 32-bit integer geconverteerd naar een 4-byte string. **INT16** Ondertekend 16-bits geheel getal geconverteerd naar een string van 2 bytes. **UINT16** Niet-ondertekend 16-bits geheel getal geconverteerd naar een tekenreeks van 2 bytes. **INT8** Ondertekend 8-bits geheel getal geconverteerd naar een string van 1 byte. **UINT8** Niet-ondertekend 8-bits geheel getal geconverteerd naar een string van 1 byte.

	SINGLE Enkele precisie drijvende-kommagetal geconverteerd naar een 4 byte string. **DOUBLE** Dubbele precisie drijvende-kommagetal geconverteerd naar een 8 byte tekenreeks. Standaard bevat de string het getal in little-endian-formaat (d.w.z. de minst significante byte is de eerste in de string). De derde instellen parameter naar **'BIG'** retourneert de string in big-endian-formaat (d.w.z. de meest significante byte is de eerste in de string) In het geval van het gehele getal conversies, wordt er een fout gegenereerd als de **'waarde'** niet past in de 'type' (bijvoorbeeld een poging om de **waarde 400 op te slaan in een INT8**). Deze functie maakt het gemakkelijk om gegevens voor te bereiden voor efficiënte binaire bestands-I/O of voor het voorbereiden van nummers voor uitvoer naar sensoren en opslaan in flashgeheugen. Zie ook de functie **STR2BIN**
BOUND(array() [,dimensie]	**Dit retourneert de bovengrens van de array voor de gevraagde dimensie.** De dimensie is standaard één als deze niet is opgegeven. Een dimensie opgeven waarde van **0** retourneert de huidige waarde van **OPTION BASE**. Ongebruikte dimensies retourneren de **waarde nul**. Bijvoorbeeld: `DIM myarray(44, 45)` `BOUND(myarray(), 2)  ' Retourneert 45`
CALL(userfunname$, [,userfunparameters,...])	**Dit is een efficiënte manier om door de gebruiker gedefinieerde functies programmatisch aan te roepen.** (Zie ook het **CALL** commando). In veel gevallen kan het worden gebruikt om te elimineren complexe **SELECT** en **IF THEN ELSEIF ENDIF** clausules en wordt verwerkt op een veel efficiëntere manier. **'userfunname$'** kan elke tekenreeks of variabele of functie zijn die wordt omgezet in de naam van een normale gebruikersfunctie (geen ingebouwde opdracht). **'userfunparameters'** zijn dezelfde parameters die zouden worden gebruikt om de . aan te roepen direct functioneren. Een typisch gebruik voor dit commando zou het schrijven van elke soort emulator kunnen zijn waarbij een van een groot aantal functies moet worden aangeroepen, afhankelijk van een variabel. Het biedt ook een methode om een functienaam door te geven aan een ander subroutine of functioneren als een variabele.

FUNCTIES	Beschrijving
CHOICE(voorwaarde, ExpressieIfTrue, ExpressieIfFalse)	**Met deze functie kunt u eenvoudige of/of-selecties efficiënter uitvoeren en sneller dan het gebruik van IF THEN ELSE ENDIF clausules.** De voorwaarde is alles dat wordt omgezet in **niet-nul** (**true**) of **nul** (**false**). De uitdrukkingen zijn alles wat je normaal gesproken zou kunnen toewijzen aan een variabele of gebruiken in een commando en kunnen gehele getallen, floats of strings zijn. Voorbeelden: `PRINT CHOICE (1, "hallo", "tot ziens")` `' Zal "Hallo" afdrukken.` `PRINT CHOICE (0, "hallo", "tot ziens")` `' Zal "Tot ziens" afdrukken.` `a = 1 : b = 1 : PRINT CHOICE (a = b, 4, 5)` `' Zal 4 afdrukken.`
CHR$(number)	**Geeft een tekenreeks van één teken terug die bestaat uit het teken dat overeenkomt met de ASCII-code (d.w.z. bytewaarde) aangegeven door argument 'getal'.**
CINT(getal)	**Ronde getallen met breuken omhoog of omlaag naar het volgende geheel getal of geheel getal.** Voorbeeld: `45.47` wordt afgerond op `45` `45.57` wordt afgerond op `46` `-34.45` wordt afgerond op `-34` `-34.55` wordt afgerond op `-35` Zie ook **INT()** en **FIX()**.
COS(getal)	**Geeft als resultaat de cosinus van het argument 'getal' in radialen.**
CTRLVAL(#ref)	**Geeft de huidige waarde van een geavanceerd besturingselement.** **'#ref'** is de referentie van het besturingselement. Voor bedieningselementen zoals selectievakjes of schakelaars zal de **nummer één zijn** (**waar**) om aan te geven dat het besturingselement is geselecteerd door de gebruiker **of nul** (**false**) **indien niet**. Voor bedieningselementen die een nummer bevatten (bijv. een **SPINBOX)** is de waarde het getal (normaal gesproken **een drijvende komma** nummer). Voor bedieningselementen die een tekenreeks bevatten (bijv. **TEXTBOX**) is de waarde een string. **Deze functie is niet beschikbaar op de PicoMiteVGA**.
CWD$	**De huidige werkmap op de SD-kaart.** **Ongeldig** voor **exFAT** indeling. Het formaat is: `A:/dir1/dir2.`
DATE$	**Retourneert de huidige datum op basis van de interne klok van MMBasic als een tekenreeks in het formulier "DD-MM-JJJJ".** Bijvoorbeeld `"28-07-2012"`. De **interne klok/kalender** houdt de **tijd en datum** bij, **inclusief schrikkeljaren**. Gebruik het commando **DATE$ =** om de datum in te stellen.

FUNCTIES	Beschrijving 266
DATETIME$(n)	**Geeft als resultaat de datum en tijd die overeenkomen met het epochenummer n (aantal seconden die zijn verstreken sinds middernacht GMT op 1 januari 1970).** Het formaat van de geretourneerde tekenreeks is: "dd-mm-jjjj uu:mm:ss". Gebruik de tekst **NOW** om de huidige datetime string te krijgen, d.w.z. ? DATETIME$(NOW)
DAY$(date$)	**Retourneert de dag van de week voor een bepaalde datum als een tekenreeks "Maandag", "dinsdag" enz.** Het formaat voor **datum$** is "DD-MM-YY", "DD-MM-YYYY", of "JJJJ-MM-DD". Gebruik **NOW** om **de dag** voor de **huidige datum te krijgen**, b.v. PRINT DAY$(NOW)
DEG(radialen)	**Converteert 'radialen' naar graden.**
DEVICE(GAMEPAD channel, funct)	**Retourneert gegevens van een USB PS3- of PS4-controller.** **'funct'** is een code van 1 of 2 letters die als volgt aangeeft welke informatie moet worden geretourneerd: **LX de positie van de x-as van de analoge linker joystick** **LY de positie van de y-as van de analoge linker joystick** **RX de positie van de x-as van de analoge rechter joystick** **RY de positie van de y-as van de analoge rechter joystick** **GX de aflezing van de X-as-gyro (waar ondersteund)** **GY de aflezing van de Y-as-gyro (waar ondersteund)** **GZ de aflezing van de Z-as-gyro (waar ondersteund)** **AX de waarde van de versnellingsmeter op de X-as (waar ondersteund)** **AY de waarde van de Y-as-versnellingsmeter (waar ondersteund)** **AZ de waarde van de Z-as-versnellingsmeter (waar ondersteund)** **L de positie van de analoge linkerknop** **R de positie van de analoge rechterknop** **B een bitmap van de status van alle knoppen. Een bit wordt op 1 gezet als de knop wordt ingedrukt.** **T de ID-code van de controller. De bitmap van de knop is als volgt:** **BIT 0 Knop R/R1** **BIT 1 Knop start/opties** **BIT 2 Knop naar huis** **BIT 3 Knop selecteren/delen** **BIT 4 Knop L/L1** **BIT 5 Knop omlaag-cursor** **BIT 6 Knop rechtercursor** **BIT 7 Knop omhoog-cursor** **BIT 8 Knop linkercursor** **BIT 9 Rechterschouderknop 2/R2** **BIT 10 Knop x/driehoek** **BIT 11 Knop a/cirkel** **BIT 12 Knop y/vierkant** **BIT 13 Knop z/kruis** **BIT 14 Links moet knop 2/L2 zijn** **BIT 15 Touchpad**

DEVICE(MOUSE channel, funct)	**Retourneert gegevens van een muis die is aangesloten via 'kanaal'.** Een PS2-muis krijgt altijd kanaal 2 toegewezen. Normaal gesproken krijgt een USB-muis dat ook toegewezen aan kanaal 2, maar dit kan variëren. Zie **MM.INFO(USB n)** voor meer informatie. **'funct'** is een code van 1 letter die als volgt aangeeft welke informatie moet worden geretourneerd: **X de X-coördinaat (0 tot MM.HRES - 1)** **Y de Y-coördinaat (0 tot MM.VRES - 1)** **L de status van de linkermuisknop.** **R de status van de rechtermuisknop.** **M de status van de middelste muisknop (wielklik)** **D 1 als er een dubbelklik met de linkermuisknop is geweest.**
DEVICE(WII [CLASSIC] funct)	**Retourneert gegevens van een Wii Classic-controller.** **'funct'** is een code van 1 of 2 letters die als volgt aangeeft welke informatie moet worden geretourneerd: **LX de positie van de x-as van de analoge linker joystick** **LY de positie van de y-as van de analoge linker joystick** **RX de positie van de x-as van de analoge rechter joystick** **RY de positie van de y-as van de analoge rechter joystick** **L de positie van de analoge linkerknop** **R de positie van de analoge rechterknop** **B een bitmap van de status van alle knoppen. Een bit wordt op 1 gezet als de knop wordt ingedrukt.** **T de ID-code van de controller - moet hex &HA4200101 zijn.** **De bitmap van de knop is als volgt:** **BIT 0 Knop R** **BIT 1 Knopstart** **BIT 2 Knop naar huis** **BIT 3 Knop selecteren** **BIT 4 knop L** **BIT 5 Knop omlaag-cursor** **BIT 6 Knop rechtercursor** **BIT 7 Knop omhoog-cursor** **BIT 8 Knop linkercursor** **BIT 9 Knop ZR** **BIT 10 Knop x** **BIT 11 Knop a** **BIT 12 Knop y** **BIT 13 knop b** **BIT 14 Knop ZL**

FUNCTIES	Beschrijving 268
DEVICE(WII NUNCHUCK funct) DEVICE(NUNCHUCK x)	**Retourneert gegevens van een Wii Classic-controller.** **'funct'** is een code van 1 of 2 letters die als volgt aangeeft welke informatie moet worden geretourneerd: **AX de versnelling op de x-as** **AY de versnelling van de y-as** **AZ de z-asversnelling** **JX de positie van de x-as van de joystick** **JY de positie van de y-as van de joystick** **C de status van de C-knop** **Z de status van de Z-knop** **T de ID-code van de controller, moet hex &HA4200000 zijn.**
DIR$(fspec, type) of DIR$(fspec) of DIR$( )	**Zal op een SD-kaart naar bestanden zoeken en de namen van gevonden items retourneren.** **'fspec'** is een bestandsspecificatie die dezelfde jokertekens gebruikt als gebruikt door de FILES opdracht. bijv. **"*.*"** retourneert alle items, **"*.TXT"** retourneert tekstbestanden. Merk op dat het jokerteken ***.*** geen bestanden of mappen vindt zonder een verlenging. **'type'** is het type invoer dat moet worden geretourneerd en kan een van de volgende zijn: **VOL** Alleen zoeken naar het volumelabel **DIR** Alleen zoeken naar mappen **FILE** Zoek alleen naar bestanden (de standaardinstelling als **'type'** niet is opgegeven) De functie retourneert de **eerste gevonden invoer**. Volgende items ophalen gebruiken de functie zonder argumenten. d.w.z. **DIR$( )**. De terugkeer van een lege string geeft aan dat er geen items meer zijn om op te halen. In dit voorbeeld worden alle bestanden in een map afgedrukt: **f$ = DIR$("*.*", FILE)** **DO WHILE f$ <> ""** **PRINT f$** **f$ = DIR$()** **LOOP** U moet naar de vereiste directory gaan voordat u deze opdracht uitvoert.
DISTANCE (trigger; echo) of DISTANCE (trigger-echo)	**Meet de afstand tot een doel met behulp van de HC-SR04 ultrasone afstand sensor.** **Vierpolige sensoren** hebben aparte trigger- en echo-aansluitingen. **'trigger'** is de I/O-pin aangesloten op de **"trig"**-ingang van de sensor en **'echo'** is de pin aangesloten op de **"echo"**-uitgang van de sensor. **Driepolige sensoren** hebben een gecombineerde **trigger-** en **echo-**aansluiting en daarbij in het geval dat u slechts één I/O-pin hoeft te specificeren voor de interface naar de sensor.

FUNCTIES	Beschrijving 269
	Merk op dat de **HC-SR04** een **5V** apparaat is, dus **niveauverschuiving** is vereist op de **Pico RP2040** processors, maar niet op de **Pico 2 RP2350** processors. De **I/O-pinnen** worden hierdoor automatisch geconfigureerd functie en **meerdere sensoren** kunnen op verschillende **I/O-pinnen worden gebruikt**. **De geretourneerde waarde is de afstand in centimeters tot het doel** of **-1** indien geen **doel** werd gedetecteerd of **-2 als er een fout was** (d.w.z. sensor niet aangesloten).
`EOF( [#]fnbr )`	**Geeft true terug als het bestand eerder op de SD-kaart is geopend voor INPUT met het bestandsnummer '#fnbr' staat aan het einde van het bestand.** De # is optioneel. Zie ook de **OPEN**, **INPUT** en **LINE INPUT** commando's en de **INPUT$** functie.
`EPOCH(DATETIME$)`	**Retourneert het epochenummer (aantal seconden dat is verstreken sinds middernacht GMT op 1 januari 1970) voor de opgegeven DATETIME$ tekenreeks.** Het formaat voor **DATETIME$** is: `"dd-mm-jjjj uu:mm:ss"` `"dd-mm-jj uu:mm:ss"` `"jjjj-mm-dd uu:mm:ss"` Gebruik **NOW** om het tijdperk te krijgen nummer voor de huidige datum en tijd, d.w.z. `PRINT EPOCH(NOW)`
`EVAL(string$)`	**Zal 'string$' evalueren alsof het een BASIC-expressie is en het resultaat retourneren.** **'string$'** kan een **constante**, een **variabele** of een **tekenreeksuitdrukking** zijn. De **uitdrukking** kan **alle bekende operators**, **functies, variabelen, subroutines**, enz. **gebruiken** op het **moment** van **executie**. De geretourneerde waarde is **een geheel getal, float** of **string** afhankelijk van het **resultaat van de evaluatie**. Bijvoorbeeld: `S$ = "COS(RAD(30)) * 100" : PRINT EVAL(S$)` Wordt weergegeven: `86.6025`
`EXP(getal)`	**Geeft als resultaat de exponentiële waarde van 'getal', d.w.z. exp waarbij x 'getal' is.**
`FIX(getal)`	**Kap een getal af tot een geheel getal door de komma te elimineren en alle tekens rechts van de komma.** `9.89` retourneert bijvoorbeeld `9` en `-2.11` retourneert `-2`. Het belangrijkste verschil tussen **FIX()** en **INT()** is dat **FIX()** een true integer functie (d.w.z. retourneert niet het eerstvolgende lagere getal voor negatief getallen zoals **INT()** doet). Dit gedrag is bedoeld voor Microsoft-compatibiliteit. Zie ook **CINT()**.

FUNCTIES	Beschrijving
`FIELD$(string1, nbr, string2 [, string3])`	**Retourneert een bepaald veld in een tekenreeks, waarbij de velden worden gescheiden door scheidingstekens.** **'nbr'** is het veld dat moet worden geretourneerd (de eerste is nr. 1). **'string1'** is de string waarnaar moet worden gezocht en **'string2'** is een string die de scheidingstekens bevat (er kunnen er meer dan één worden gebruikt). **'string3'** is optioneel en bevat, indien gespecificeerd, tekens die eraan gewend zijn citeer tekst in **'string1'** (dat wil zeggen, geciteerde tekst wordt niet doorzocht op een scheidingsteken). Bijvoorbeeld: `S$ = "foe, boe, dierentuin, doo"` `r$ = FIELD$(s$, 2, ",")` Zal resulteren in: `r$ = "boe"` Terwijl: `s$ = "foe, 'boe, dierentuin', doo"` `r$ = FIELD$(s$, 2, ",", "'")` Zal resulteren in: `r$ = "boe, dierentuin"`
`FORMAT$(nbr [, fmt$])`	**Retourneert een tekenreeks die 'nbr' vertegenwoordigt, geformatteerd volgens de specificaties in de string 'fmt$'.** De formaatspecificatie begint met een **%** teken en eindigt met een letter. Alles buiten deze constructie wordt ongewijzigd naar de uitvoer gekopieerd. De structuur van een formaatspecificatie is: `%[vlaggen] [breedte] [.precisie] type` Waar **'vlaggen'** kunnen staan: **-** Links de waarde uitvullen binnen een gegeven veldbreedte. **0** Gebruik 0 voor het pad-teken in plaats van spatie. **+** Dwingt dat het **+** teken wordt weergegeven voor positieve getallen. **spatie** Zorgt ervoor dat een positieve waarde een spatie voor het teken weergeeft. Negatieve waarden tonen nog steeds het **–** teken. **'breedte'** is het **minimum** aantal **tekens** dat moet worden uitgevoerd, minder dan dit is de nummer zal worden opgevuld, meer dan dit zal de breedte worden vergroot. **'precisie'** specificeert het aantal **breukcijfers** dat moet worden gegenereerd met een **e** of **f** type of het maximale aantal significante cijfers dat moet worden gegenereerd met een **g** type. Indien gespecificeerd, moet de precisie worden voorafgegaan door een punt **(.)**.

FUNCTIES	Beschrijving 271
	'type' kan een van de volgende zijn: **g** Automatisch het nummer opmaken voor de beste presentatie **f** Maak het getal op met de komma en de volgende cijfers. **e** Maak het getal op in exponentieel formaat. Als **hoofdletter G** of **F** wordt gebruikt, gebruikt de exponentiële uitvoer een hoofdletter **E**. Als de formaatspecificatie niet is gespecificeerd, wordt "**%g**" aangenomen. Voorbeelden: `Format$(4)  ' Retourneert 45` `Format$(45, "%g") ' Zal 45 retourneren`
GETSCANLINE	**ALLEEN VGA-VERSIE** **Dit rapporteert over de lijn die momenteel op de VGA-monitor wordt getekend in het bereik van 0 tot 525.** Dit is ongeacht de huidige **MODUS**. Als u dit gebruikt om updates op het scherm te timen, kunt u timing-effecten vermijden die worden veroorzaakt door updates terwijl het scherm wordt bijgewerkt. De eerste zichtbare regel retourneert de **waarde 0**. Elk regelnummer boven **479** is aanwezig de frameonderdrukkingsperiode.
GPS()	**De GPS functies worden gebruikt om gegevens van seriële communicatie terug te sturen kanaal geopend als GPS.** De functie **GPS(VALID)** moet worden gecontroleerd voordat een van deze functies wordt uitgevoerd worden gebruikt om ervoor te zorgen dat de geretourneerde waarde geldig is.
GPS(ALTITUDE)	**Geeft de huidige hoogte terug (als zin GGA is ingeschakeld).**
GPS(DATE)	**Geeft de normale datumreeks terug, gecorrigeerd voor lokale tijd, b.v. "12-01-2020".**
GPS(DOP)	**Retourneert DOP waarde (verdunning van precisie) (als zin GGA is ingeschakeld).**
GPS(FIX)	**Retourneert niet-nul (true) als de GPS een fix heeft op voldoende satellieten en is het produceren van geldige gegevens.**
GPS(GEOID)	**Retourneert de geoïde-ellipsoïde scheiding (als zin GGA is ingeschakeld).**
GPS(LATITUDE)	**Retourneert de breedtegraad in graden als een getal met drijvende komma, waarden zijn negatief voor ten zuiden van de evenaar.**
GPS(LONGITUDE)	**Geeft de lengtegraad in graden als een getal met drijvende komma, waarden zijn negatief voor ten westen van de meridiaan.**
GPS(SATELLITES)	**Retourneert het aantal satellieten in zicht (als zin GGA is ingeschakeld).**
GPS(SPEED)	**Retourneert de grondsnelheid in knopen als een getal met drijvende komma.**

FUNCTIES	Beschrijving 272
GPS(TIME)	**Geeft de normale tijdreeks terug, gecorrigeerd voor lokale tijd, b.v. "12:09:33".**
GPS(TRACK)	**Retourneert de track over de grond (graden waar) als een getal met drijvende komma.**
GPS(VALID)	**Retourneert: 0 = ongeldige gegevens, 1 = geldige gegevens.**
HEX$(number [, chars])	**Geeft een string terug die de hexadecimale (grondtal 16) waarde voor het 'getal' geeft.** **'chars'** is optioneel en specificeert het **aantal karakters** in de **string** met **nul** als het **leidende opvulteken(s)**.
INKEY$	**Controleert de invoerbuffer van de console en, als er een of meer tekens zijn, wacht in de wachtrij, verwijdert het eerste teken en retourneert het als een single teken in een string.** Als de invoerbuffer leeg is, keert deze functie onmiddellijk terug met een lege tekenreeks (d.w.z. "").
INPUT$(nbr, [#]fnbr)	**Geeft een tekenreeks terug die is samengesteld uit 'nbr'-tekens die uit een serienummer zijn gelezen communicatiepoort geopend als 'fnbr'.** Deze functie zal zoveel teruggeven tekens die in de ontvangstbuffer wachten tot **'nbr'**. Als er geen zijn tekens wachten, zal het onmiddellijk terugkeren met een lege tekenreeks. **#0** kan worden gebruikt, wat verwijst naar de invoerbuffer van de console. De **#** is optioneel. Zie ook het commando **OPEN**.
INSTR([startpositie,]string gezocht$, stringpatroon$) [,size])	**Retourneert de positie waarop 'string-pattern$' voorkomt in 'string-searched$', beginnend bij 'startpositie'.** Als **'startpositie'** niet is opgegeven, wordt deze standaard naar **1.** Zowel **de positie die wordt geretourneerd** als de **'startpositie'** gebruiken **1** voor het eerste teken, **2** voor de tweede enz. De functie **retourneert nul** als **'string-pattern$'** niet wordt gevonden Als de optionele parameter **"size"** is opgegeven, wordt het **"string-pattern"** behandeld als een reguliere expressie. **Zie Bijlage E voor de details**.
INT(getal)	**Een uitdrukking afkappen tot het volgende gehele getal kleiner dan of gelijk aan de argument. 9.89 retourneert** bijvoorbeeld **9** en **-2.11 retourneert -3.** Dit gedrag is voor Microsoft-compatibiliteit, de **FIX()** functie biedt een echte integer functie. Zie ook **CINT()**.

FUNCTIES	Beschrijving 273
JSON$(array%(), string$)	**Retourneert een tekenreeks die een specifiek item vertegenwoordigt uit de JSON-invoer die is opgeslagen in de lange reeksarray%().** Houd er rekening mee dat veel **JSON-gegevenssets** behoorlijk groot zijn en dat kunnen ook zijn te groot om te parseren met het beschikbare geheugen. Voorbeelden (overgenomen van api.openweathermap.org): `JSON$(a%(), "naam")` `JSON$(a%(), "coord.lat")` `JSON$(a%(), "weer[0].description")` `JSON$(a%(), "list[4].weather[0].description")`
KEYDOWN(n)	**Retourneert de decimale ASCII-waarde van de USB-toetsenbordtoets die momenteel wordt vastgehouden omlaag of nul als er geen toets omlaag is.** De decimale waarden voor de functie en de pijl sleutels vindt u in **bijlage F**. Deze functie rapporteert meerdere gelijktijdige toetsaanslagen en de parameter **'n'** is het nummer van de toetsaanslag die moet worden gerapporteerd. **KEYDOWN(0)** retourneert de aantal toetsen dat wordt ingedrukt. Als bijvoorbeeld **"c", "g"** en **"p"** tegelijkertijd worden ingedrukt **KEYDOWN(0)** geeft **3** terug, **KEYDOWN(1)** geeft **99** terug, **KEYDOWN(2)** geeft **103** terug, etc. De toetsen hoeven niet tegelijkertijd ingedrukt te worden en zullen zich melden in de bestelling ingedrukt. Als u een vinger van een toets haalt, wordt de volgende ingedrukte toets verplaatst naar **#1**. De eerste sleutel (**'n' = 1**) wordt ingevoerd in de toetsenbordbuffer (toegankelijk via **INKEY$**), terwijl de toetsen **2 t/m 6** alleen toegankelijk zijn via deze functie. Dit gebruiken functie zal de invoerbuffer van de console leegmaken. **KEYDOWN(7)** geeft alle wijzigingstoetsen weer die worden ingedrukt. Deze sleutels niet optellen bij de telling in **keydown(0)**. De retourwaarde is een bitmasker als volgt: **laat? 1, lctrl? 2, lgui? 4, verschuiving? 8, ralt? 16, rctrl? 32, rgui? 64, rverschuiving? 128** **KEYDOWN(8)** geeft de huidige status van de slotsleutels. Deze sleutels niet optellen bij de telling in **keydown(0)**. De retourwaarde is een bitmasker als volgt: **caps_lock? 1, num_lock? 2, scroll_lock? 4** Houd er rekening mee dat sommige toetsenborden het aantal actieve toetsen beperken dat ze kunnen gebruiken rapporteer over.
LCASE$(string$)	**Geeft als resultaat 'string$' geconverteerd naar kleine letters.**

FUNCTIES	Beschrijving 274
LCOMPARE(array1%(), array2%())	**Vergelijk de inhoud van twee lange tekenreeksvariabelen array1%() en array2%().** De geretourneerde waarde is een geheel getal en zal **-1** zijn als **array1%() kleiner** is dan **array2%()**. Het zal **nul** zijn als ze **gelijk zijn** in lengte en inhoud en **+1** als **array1%()** is **groter dan array2%()**. De vergelijking gebruikt de **ASCII** tekenset en is **hoofdlettergevoelig**.
LEFT$(string$, nbr)	**Retourneert een substring van 'string$' met 'nbr' tekens van links (begin) van de tekenreeks.**
LEN(string$)	**Geeft het aantal tekens in 'string$' terug.**
LGETBYTE(array%(),n)	**Retourneert de numerieke waarde van de 'n'-de byte in de LONGSTRING die wordt vastgehouden in 'matrix%()'.** Deze functie **respecteert** de instelling van **OPTION BASE** in bepalen welke **byte moet worden geretourneerd**.
LGETSTR$(matrix%(), start, lengte)	**Retourneert een deel van een lange tekenreeks die is opgeslagen in array%() als een normale MMBasic string.** De parameters **start** en **lengte** definiëren het deel van de string dat moet worden teruggekeerd.
LINSTR(matrix%(), zoek$ [, start] [,size]))	**Retourneert de positie van een zoekreeks in een lange tekenreeks.** De **geretourneerde waarde** is een **geheel getal** en **is nul** als **de subtekenreeks niet kan worden gevonden**. **array%()** is de **string waarnaar** moet worden **gezocht** en moet een **lange stringvariabele** zijn. **Zoeken$** is de **subtekenreeks** waarnaar moet worden **gezocht** en het moet een normale MMBasic-tekenreeks of uitdrukking zijn (**geen lange string**). **De zoekopdracht is hoofdlettergevoelig**. **Normaal gesproken begint** het **zoeken** bij het **eerste** teken in **'str'**, maar de **optionele derde parameter** maakt het mogelijk om de **startpositie van de zoekopdracht te specificeren**. Als de optionele parameter 'size' is opgegeven, wordt **'search$'** behandeld als een reguliere parameter uitdrukking. Zie **Bijlage E** voor de details.
LLEN(array%())	**Geeft de lengte van een lange tekenreeks die is opgeslagen in array%()**
LOC([#]fnbr)	**Voor een seriële communicatiepoort geopend als 'fnbr' zal deze functie terugkeren het aantal ontvangen bytes dat in de ontvangstbuffer wacht om gelezen te worden.** **#0** kan worden gebruikt, wat verwijst naar de **invoerbuffer van de console**. De **#** is optioneel.
LOF([#]fnbr)	**Voor een seriële communicatiepoort geopend als 'fnbr' zal deze functie terugkeren de resterende ruimte (in tekens) in de verzendbuffer.** Merk op dat wanneer **de buffer vol is**, **MMBasic pauzeert** bij het toevoegen van een nieuwe karakter en wacht tot er wat ruimte vrijkomt. De **#** is optioneel.

FUNCTIES	Beschrijving
LOG(getal)	**Geeft als resultaat de natuurlijke logaritme van het argument 'getal'.**
MAP(n)	**ALLEEN HDMI EN VGA** **Retourneert de 24-bits RGB-waarde voor de index 'n' in de kleuren-kaarttabel.** Zie het **MAP commando**. Hierdoor kan de Basisprogrammeur een kleur gebruiken gespecificeerd door het **MAP** commando bijv: `MAP(8), RGB(100, 100, 100)` `MAP INST` `Pixel x, y, MAP(8)` **Let op:** voor **VGA** worden alle kleuren die door het kaartcommando zijn ingesteld, geconverteerd naar de dichtstbijzijnde **RGB121** kleur zoals bepaald door het **VGA**-weerstandsnetwerk. Voor HDMI beeldschermkleuren worden geconverteerd naar de dichtstbijzijnde **RGB555** kleur (**640x480** resolutie) of **RGB332** kleur (**1024x768** of **1280x720** resolutie)
MATH (WISKUNDE)	**De wiskundige functie voert veel eenvoudige wiskundige berekeningen uit die kan in Basic worden geprogrammeerd, maar coderen heeft snelheidsvoordelen lusstructuren in C en er is het voordeel dat ze, eenmaal debuggen, zijn er voor iedereen zonder het wiel opnieuw uit te vinden.**
Eenvoudige functies	**Eenvoudige functies**
MATH(ATAN3 x, y)	**Retourneert ATAN3 van x en y.**
MATH(COSH a)	**Geeft als resultaat de cosinus hyperbolicus van a.**
MATH(LOG10 a)	**Geeft als resultaat de logaritme met grondtal 10 van a.**
MATH(RAND)	**Genereert een willekeurig getal 0.0 >= n < 1.0 met behulp van de "Mersenne Twister: A 623-dimensionaal * Equidistributed uniforme generator van pseudowillekeurige getallen.** Indien niet gezaaid met **MATH RANDIMIZE** het eerste gebruik zaden met de tijd in microseconden sinds het opstarten.
MATH(SINH a)	**Geeft als resultaat de sinus hyperbolicus van a.**
MATH(TANH a)	**Retourneert de hyperbolische tan van a.**
MATH(CRCn data [,length] [,polynome] [,startmask] [,endmask] [,reverseIn] [,reverseOut]	**Berekent de CRC op basis van n bits (8, 12, 16, 32) "data".** **"data"** kan een integer of drijvendekommaarray of een stringvariabele zijn **"Lengte"** is optioneel en als dit niet is opgegeven, wordt de grootte van de array of de tekenreekslengte gebruikt. De standaardwaarden voor **startmask, endmask reverseIn** en **reversOut** zijn allemaal **nul**. **reverseIn** en **reversOut** zijn beide Booleans en hebben de waarde **1** of **0**. De standaardwaarden voor polynomen zijn: **CRC8=&H07, CRC12=&H80D, CRC16=&H1021, CRC32=&H04C11DB7** Gebruik bijvoorbeeld voor: **crc16_CCITT MATH(CRC16 array(), n,, &HFFFF)**

FUNCTIES	Beschrijving
`MATH(CRC8 array(), length, [polynoom,] [startmask,] [endmask,] [reverseIn,] [reverseOut]`	**Standaardwaarden voor startmask, endmask reverseIn en reverseOut zijn allemaal nul.** Standaardwaarden voor **Polynomen** zijn **CRC8 = &H07**
`MATH(CRC12 array(), lengte, [polynoom,] [startmask,] [endmask,] [reverseIn,] [reverseOut]`	**Standaardwaarden voor startmask, endmask reverseIn en reverseOut zijn allemaal nul.** Standaardwaarden voor **Polynomen** zijn **CRC12 = &H80D**
`MATH(CRC16 array(), lengte, [polynoom,] [startmask,] [endmask,] [reverseIn,] [reverseOut]`	**Standaardwaarden voor startmask, endmask reverseIn en reverseOut zijn allemaal nul.** Standaardwaarden voor **Polynomen** zijn **CRC16 = &H1021**
`MATH(CRC32 array(), lengte, [polynoom,] [startmask,] [endmask,] [reverseIn,] [reverseOut]`	**Standaardwaarden voor startmask, endmask reverseIn en reverseOut zijn allemaal nul.** Standaardwaarden voor **Polynomen** zijn **CRC8 = &H04C11DB7** bijv. Voor **crc16_CCITT** gebruik: `MATH(CRC16 array(),n,,&HFFFF)` dus: `DIM a%(8) = (49,50,51,52,53,54,55,56,57)` `PRINT HEX$(MATH(CRC16 a%(),9,,&HFFFF))` geeft `&H29B1`
statistieken	**Eenvoudige statistieken**
`MATH(CHI a())`	**Retourneert de chi-kwadraatwaarde van Pearson van de tweedimensionale matrix a()).**
`MATH(CHI_p a())`	**Retourneert de bijbehorende waarschijnlijkheid in % van de chi-kwadraatwaarde van Pearson van de tweedimensionale array a())**
`MATH(CROSSING array() [,level] [,direction]`	**Dit retourneert de array-index waarbij de waarden in de array het "niveau" passeren de opgegeven richting.** **Niveau** is standaard ingesteld op **0**. **Richting** is standaard ingesteld op **1 (geldige waarden zijn -1 of 1)**
`MATH(CORREL a(),b())`	**Retourneert de correlatiecoëfficiënt van Pearson tussen arrays a() en b().**
`MATH(MAX a() [, index%])`	**Retourneert het maximum van alle waarden in de a() array, a() kan elke hebben aantal afmetingen**. Als de **integervariabele** is opgegeven, is het **bijgewerkt met de index** van de **maximale waarde in de array**. Dit is **alleen beschikbaar** op **eendimensionale arrays**.
`MATH(MEAN a())`	**Retourneert het gemiddelde van alle waarden in de a() array, a() kan elk willekeurig getal hebben van afmetingen.**
`MATH(MEDIAN a())`	**Retourneert de mediaan van alle waarden in de a() array, a() kan een willekeurig aantal dimensies hebben.**

FUNCTIES	Beschrijving
MATH(MIN a(), [index%])	**Retourneert het minimum van alle waarden in de a() array, a() kan elk willekeurig getal hebben van afmetingen.** Als de **integervariabele is opgegeven**, wordt deze **bijgewerkt met de index** van de **minimale waarde in de array**. Dit is alleen beschikbaar op **eendimensionale arrays.**
MATH(SD a())	**Retourneert de standaarddeviatie van alle waarden in de a() array, a() kan een willekeurig aantal afmetingen hebben.**
MATH(SUM a())	**Retourneert de som van alle waarden in de a() array, a() kan een willekeurig aantal dimensies hebben.**
Vector rekenkunde	**Vector rekenkunde**
MATH(MAGNITUDE v())	**Retourneert de grootte van de vector v(). De vector kan een willekeurig aantal elementen hebben.**
MATH(DOTPRODUCT v1(), v2())	**Retourneert het puntproduct van twee vectoren v1() en v2().** De vectoren kunnen hebben een willekeurig aantal elementen maar moet dezelfde kardinaliteit hebben.
Matrix Rekenen	**Matrix Rekenen**
MATH(M_DETERMINANT reeks!())	**Retourneert de determinant van de array.** De array moet vierkant zijn. **MMBasic** ondersteunt een volledig scala aan functies maken de manipulatie van complexe getallen mogelijk. In deze implementatie complexe getallen hebben een 32-bits echt en 32-bits denkbeeldig deel en om te maken dit werkt in **MMBasic**, het gebruikt gehele getallen (64-bit) om houd deze vast.
Floating returns	**Floating returns**
	`real! = MATH(C_REAL  complex%)` `imag! = MATH(C_IMAG  complex%)` `arg! = MATH(C_ARG  complex%)` `mod! = MATH(C_MOD  complex%)` `phase! = MATH(C_PHASE complex%)`
Unary functions	**Unary functions**
	`complex1% = MATH(C_CONJ  complex2%)` `complex1% = MATH(C_SIN  complex2%)` `complex1% = MATH(C_COS  complex2%)` `complex1% = MATH(C_TAN  complex2%)` `complex1% = MATH(C_ASIN  complex2%)` `complex1% = MATH(C_ACOS  complex2%)` `complex1% = MATH(C_ATAN  complex2%)` `complex1% = MATH(C_SINH  complex2%)` `complex1% = MATH(C_COSH  complex2%)` `complex1% = MATH(C_TANH  complex2%)` `complex1% = MATH(C_ASINH complex2%)` `complex1% = MATH(C_ACOSH complex2%)` `complex1% = MATH(C_ATANH complex2%)` `complex1% = MATH(C_PROJ  complex2%)`

FUNCTIES	Beschrijving
Creation	**Creation**
	`complex% = MATH(C_CPLX r!, i!)` `complex% = MATH(C_POLAR radius!, angle!)`
Basic Arithmetic	**Basic Arithmetic**
	`complex1% = MATH(C_ADD complex2%, complex3%)` `complex1% = MATH(C_SUB complex2%, complex3%)` `complex1% = MATH(C_MUL complex2%, complex3%)` `complex1% = MATH(C_DIV complex2%, complex3%)` `complex1% = MATH(C_POW complex2%, complex3%)` `complex1% = MATH(C_AND complex2%, complex3%)` `complex1% = MATH(C_OR  complex2%, complex3%)` `complex1% = MATH(C_XOR complex2%, complex3%)`
`MATH(PID channel, setpoint!, measurement))`	**Deze functie moet worden aangeroepen in de PID callbacksub-routine voor het 'kanaal' opgegeven en retourneert de uitvoer van de controllerfunctie.** De **'setpoint'** waarde is de gewenste toestand die de controller pro-beert te bereiken. De **'meting'** is de huidige waarde van de echte wereld. https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=17263 Voor een voorbeeld van het instellen en gebruiken van een **PID-**regelaar.
`MATH(BASE64 ENCODE | DECODE in$ | in(), out$ | out())`	**Retourneert de lengte van out$/out(). Deze base64 codeert of decodeert de gegevens in 'in' en zet het resultaat op 'out'.** Wanneer arrays als uitvoer worden gebruikt, moeten ze groot zijn voldoende ten opzichte van de invoer en de richting. **Encryptie ver-groot** de **lengte met 4/3** en **decodering verkleint het met 3/4**.
`MAX(arg1 [,arg2 [,...]])` of `MIN(arg1 [,arg2 [,...]])`	**Retourneert het maximum of minimum aantal in de lijst met argu-menten.** Merk op dat de vergelijking een drijvende-kommavergelijking is (integer-argumenten worden geconverteerd naar **floats**) en een **float** wordt geretourneerd.
`MID$(string$, start)` of `MID$(string$, start, nbr)`	**Retourneert een subtekenreeks van 'string$' beginnend bij 'start' en doorgaand voor 'nbr' karakters.** Het eerste teken in de string is nummer 1. Als **'nbr'** wordt **weggelaten**, wordt de geretourneerde tekenreeks **verlengd tot het einde van 'string$'.**
`MSGBOX(msg$, b1$ [, b2$ ... b4$])`	**Deze functie geeft een berichtvenster op het scherm weer met één tot vier aanraakgevoelige knoppen.** Alle andere bedieningselementen worden uitgeschakeld totdat de gebruiker raakt een van de knoppen aan. Het berichtvenster wordt dan gewist, de vorige be-dieningselementen worden hersteld en de functie retourneert het aantal de knop aangeraakt (de eerste knop is nummer één) **'msg$'** is het bericht dat moet worden weergegeven.

FUNCTIES	Beschrijving 279
	Dit kan een of meer tilde bevatten tekens **(~)** die een regeleinde aangeven. Er kunnen maximaal **10** regels worden weergegeven in de box. **'b1$'** is het on-derschrift voor de **eerste knop**, **'b2$'** is het onderschrift voor de **tweede knop**, enz. Er moet minimaal één knop worden opgegeven en vier is het maximum. Knoppen die niet in de argumentenlijst zijn opgenomen, **worden niet weergegeven.**
`OCT$(number [, chars])`	**Geeft een string terug die de octale (grondtal 8) representatie van 'getal' geeft.** **'chars'** is optioneel en specificeert het aantal karakters in de string met **nul** als het leidende **opvulteken(s)**.
`PEEK(BP n%)`	**peek(bp n%) ' Retourneert de byte op adres n% en verhoogt n% om naar de volgende byte te verwijzen.**
`PEEK(SP n%)`	**peek(sp n%) ' Retourneert de kortsluiting op adres n% en verhoogt n% om naar de volgende kortsluiting te verwijzen.**
`PEEK(WP n%)`	**peek(wp n%) ' Retourneert het woord op adres n% en verhoogt n% om naar het volgende woord te verwijzen.**
`PEEK(BYTE addr%)`	**Retourneert een byte of een woord binnen de PicoMite virtuele geheugenruimte.** **BYTE** retourneert de **byte** (**8-bits**) die zich op **'addr%'** bevindt.
`PEEK(SHORT addr%)`	**SHORT retourneert het korte gehele getal (16-bits) op 'addr%'.**
`PEEK(WORD addr%)`	**WORD retourneert het woord (32-bits) dat zich op 'addr%' bevindt.**
`PEEK(INTEGER addr%)`	**INTEGER retourneert het gehele getal (64-bits) op 'addr%'.**
`PEEK(FLOAT addr%)`	**FLOAT retourneert het drijvende-kommagetal (32-bits) op 'addr%'.**
`PEEK(VARADDR var)`	**VARADDR retourneert het adres (32-bits) van de variabele 'var' in geheugen. Een array wordt gespecificeerd als var().**
`PEEK(CFUNADDR cfun)`	**CFUNADDR retourneert het adres (32-bits) van de CFunction 'cfun' in geheugen.** Dit adres kan worden doorgegeven aan een andere **CFunctie** die dan roep het om een **gemeenschappelijk** proces **uit te voeren**.
`PEEK(VAR var, ±offset)`	**VAR, retourneert een byte in het geheugen dat is toegewezen aan 'var'. Een array is opgegeven als var().**
`PEEK( VARTBL, ±offset)`	**VARTBL, retourneert een byte in het geheugen dat is toegewezen aan de variabele tabel onderhouden door MMBasic.** **Let op:** Er staat een **komma** achter het trefwoord **VARTBL.**
`PEEK( PROGMEM, ±offset)`	**PROGMEM, retourneert een byte in het geheugen dat aan het programma is toegewezen.** **Merk op** dat er een **komma** staat achter het trefwoord **PROGMEM**. **Merk op dat 'addr%' een geheel getal moet zijn.**
`PI`	**Retourneert de waarde van pi.**

FUNCTIES	Beschrijving 280
PIN(pin) SETPIN Pinno/GPno,ARAW ? Pin(Pinno/GPno)	**Geeft de waarde op de externe I/O 'pin'.** **Nul** betekent **digitaal laag**, **1** betekent **digitaal hoog** en voor **analoge ingangen** zal het de **geme-ten spanning** retourneren als een getal met drijvende komma. **Frequentie-ingangen** retourneren de **frequentie in Hz**. **Een periode-invoer** zal terugkeren **de periode in milliseconden** terwijl **een tel-linginvoer** de **telling sinds reset** zal retourneren (**tellen gebeurt op de positieve stijgende flank**). **De telinvoer** kan zijn **reset naar nul** door **de pin te resetten naar telingang** (**zelfs als dit al zo is geconfigureerd**). Deze **functie retour-neert** ook de **status** van **een pin** die als **uitgang** is **geconfigureerd**. **Zie ook de SETPIN en PIN() = commando's.** Raadpleeg het hoofdstuk **"Gebruiken" de I/O-pinnen"** voor een al-gemene beschrijving van de invoer/uitvoer van de **PicoMite** moge-lijkheden.
PIN(BOOTSEL)	**Retourneert de status van de opstartkeuzeschakelaar, zodat deze kan worden gebruikt als gebruikersinvoer in een programma.**
PIN(TEMP)	**Geeft de temperatuur van de RP2040 / RP2350 chip terug.** **(zie het RP2040/RP2350 gegevensblad voor) de details)**
PIO(DMA RX POINTER) PIO(DMA TX POINTER)	**Retourneert het huidige gegevensitem dat door de PIO wordt geschreven of gelezen.**
PIO(SHIFTCTRL push_threshold [,pull_threshold] [,autopush][,autopull] [,in_shiftdir] [,out_shiftdir] [,fjoin_rx] [,fjoin_tx])	**Help functie om de waarde van shiftctrl voor de INIT MACHINE commando te berekenen.**
PIO(PINCTRL no_side_set_pins [,no_set_pins] [,no_out_pins] [,IN basis] [,side_set_base] [,set_base] [,out_base])	**Help functie om de waarde van pinctrl voor de INIT MACHINE commando te berekenen.** **Note: The pin parameters must be formatted as GPn.**
PIO(EXECCTRL jmp_pin ,wrap_target, wrap [,side_pindir] [,side_en])	**De EXECCTRL waarde voor PIO INIT correct berekenen.** **Vereenvoudigt het gebruik van wrappen in PIO assemble.**
PIO(READFIFO a, b, c)	**Lezen van een PIO FIFO** **'a' is de pio (0 of 1),** **'b' id de toestandsmachine (0...3),** **'c' is het FIFO-register * 0...3)**
PIO(FDEBUG pio)	**Retourneert de waarde van het FSDEBUG-register voor de opgege-ven pio.**

FUNCTIES	Beschrijving 281
`PIO(FSTAT pio)`	**Retourneert de waarde van het FSTAT-register voor de opgegeven pio.**
`PIO(FLEVEL pio)`	**Geeft de waarde van het FLEVEL-register voor de gespecificeerde pio) PIO(FLEVEL pio)**
`PIO(FLEVEL pio,sm,DIR`	**DIR kan RX of TX zijn. Retourneert het niveau van de specifieke fifo**
`PIO(.WRAP)`	**Retourneert de locatie van de .wrap-richtlijn in PIO ASSEMBLE.**
`PIO(.WRAP TARGET)`	**Retourneert de locatie van de .wrap-doelrichtlijn in PIO ASSEMBLE.** Deze kunnen als volgt in de **PIO(EXECCTRL-functie)** worden gebruikt: `PIO(EXECCTRL jmp_pin PIO(.WRAP TARGET), PIO(.WRAP [,side_pindir] [,side_en])`
`PORT(start, nbr [, start, nbr]...)`	**Geeft de waarde van een aantal I/O-pinnen in één bewerking.** **'start'** is een **I/O-pincode** en de waarde ervan wordt geretourneerd als bit 0. **'start' +1** zal worden geretourneerd als **bit 1**, **'start' + 2** wordt geretourneerd als **bit 2**, enzovoort voor **'nbr' aantal bits**. Gebruikte I/O pinnen moeten doorlopend worden genummerd en elke I/O pin die ongeldig is of niet als invoer is geconfigureerd, zal een fout veroorzaken. Het start/nbr paar kan tot **25 keer** worden herhaald als extra groepen invoerpinnen hoeft te worden toegevoegd**.** Deze functie retourneert ook de status van een pin die als uitgang is geconfigureerd. Het kan worden gebruikt om gemakkelijk te communiceren met parallelle apparaten zoals geheugenchips. Elk aantal I/O-pinnen (en dus bits) kan worden gebruikt van 1 tot het aantal I/O-pinnen op de chip. Zie het PORT-commando om gelijktijdig uit te voeren naar een aantal pinnen.
`PIXEL(x, y [, paginanummer])`	**Retourneert de kleur van een pixel op een LCD-scherm.** **'x'** is de **horizontale coördinaat** en **'y'** is de **verticale coördinaat** van **de pixel**. **Het scherm** moet een van de **SSD1963-, ILI9341-, ILI9488- of ST7789_320** controllers **gebruiken**.
`PULSIN(pin, polariteit)` of `PULSIN(pin, polariteit, t1)` of	**Meet de breedte van een ingangspuls van 1 µs tot 1 seconde met 0,1 µs oplossing.** **'pin'** is de **I/O-pin** die moet worden **gebruikt voor de meting**, deze moet eerder zijn **geconfigureerd als een digitale ingang**. **'polariteit'** is het **type puls** dat moet **worden gemeten**, als **nul** zal de functie **de breedte** van de **volgende negatieve puls teruggeven**, indien **niet nul** zal het de **volgende positieve puls meten**. **'t1'** is **de timeout** die wordt toegepast tijdens het wachten tot **de puls arriveert**, **'t2'** is **de timeout** gebruikt **tijdens het meten van de puls.** Beide zijn in **microseconden (µs)** en zijn **optioneel**. Als **'t2'** wordt **weggelaten**, wordt de waarde van **'t1'** voor **beide gebruikt timeouts**

FUNCTIES	Beschrijving 282
PULSIN(pin, polariteit, t1, t2)	**Als zowel 't1' als 't2' worden weggelaten, worden de timeouts ingesteld op 100000 (d.w.z. 100 ms).** Deze **functie** retourneert de **breedte** van **de puls in microseconden (µs)** of **-1** als een **timeout is opgetreden**. De **meting** is **nauwkeurig** tot **± 0,5%** en **±0.5µs**. **Merk op** dat deze **functie** ervoor zorgt dat het **lopende programma pauzeert** terwijl **de meting wordt gedaan** en **interrupts** worden **tijdens deze periode genegeerd.**
RAD(graden)	**Converteert 'graden' naar radialen.**
RGB(red,green,blue) of RGB(shortcut)	**Genereert een echte RGB kleurwaarde.** **'rood'**, **'blauw'** en **'groen'** vertegenwoordigen de intensiteit van elke kleur. Een waarde van **RGB(snelkoppeling) nul** staat voor **zwart** en **255** staat voor **volledige intensiteit**. **'shortcut' maakt het mogelijk om algemene kleuren te specificeren door ze een naam te geven.** De kleuren die genoemd kunnen wor-den zijn wit, zwart, blauw, groen, cyaan, rood, magenta, geel, bruin, wit, oranje, roze, goud, zalm, beige, lichtgrijs en grijs (of USA spelling grijs/lichtgrijs). Bijvoorbeeld RGB(red) of RGB(cyan). Merk op dat de geretourneerde waarde een geheel getal is en, als het moet worden opgeslagen, de variabele moet worden gedeclareerd als **een geheel getal om de nauwkeurigheid van de te behouden nummer.**
RIGHT$(string$, aantal tekens)	**Retourneert een subtekenreeks van 'string$' met 'aantal tekens' van rechts (einde) van de string.**
RND(getal) of RND	**Retourneert een pseudo willekeurig getal in het bereik van 0 tot 0.999999.** De **'nummer'** waarde wordt **genegeerd indien opgegeven.** De opdracht **RANDOMIZE** zal opnieuw de **generator starten** voor **willekeurige getallen.** **See also the RANDOMIZE command (RP2040 only).**
SGN(getal)	**Geeft het teken van het argument 'getal', +1 voor positieve getallen, 0 voor 0, en -1 voor negatieve getallen.**
SIN(getal)	**Geeft als resultaat de sinus van het argument 'getal' in radialen.**
SPACE$(number)	**Retourneert een reeks lege spaties 'getal'-tekens lang.**
SPI(data) of SPI2(data)	**Gegevens verzenden en ontvangen via een SPI-kanaal.** Een enkele SPI transactie verzendt gegevens en ontvangt tegelijkertijd gegevens van de slaaf. 'data' zijn de gegevens die moeten worden verzonden en de functie retourneert de gegevens ontvangen tijdens de transactie. **'data'** kan een geheel getal of een floating zijn puntvariabele of een constante.

FUNCTIES	Beschrijving 283
SPRITE()	**ALLEEN VGA- EN HDMI-VERSIES** **De SPRITE functies retourneren informatie over sprites, dit zijn kleine grafische afbeeldingen op het VGA/HDMI-scherm.** Deze zijn handig bij het schrijven van games. **Zie ook de SPRITE commando's.**
SPRITE(C, [#]n)	**Geeft het aantal momenteel actieve botsingen voor sprite n terug.** Als **n=0** retourneert het aantal sprites dat momenteel een actieve botsing heeft na een **SPRITE SCROLL** commando.
SPRITE(C, [#]n, m)	**Geeft het nummer terug van de sprite die de "m"-de botsing van sprite n veroorzaakte.** Als **n=0** retourneert het **sprite-nummer** van **"m"-de sprite** die momenteel een actieve botsing heeft na een **SPRITE SCROLL** commando. Als de botsing met de rand van het scherm plaatsvond, is de geretourneerde waarde: `&HF1 botsing met de linkerkant van het scherm` `&HF2 botsing met de bovenkant van het scherm` `&HF4 botsing met de rechterkant van het scherm` `&HF8 botsing met de onderkant van het scherm`
SPRITE(D, [#]s1, [#]s2)	**Geeft de afstand terug tussen de middelpunten van sprites 's1' en 's2' (retourneert -1 als een van beide sprite niet actief is)**
SPRITE(E, [#]n	**Retourneert een bitmap die de randen van het scherm aangeeft waarmee de sprite in contact is:** `1 = linkerkant van scherm,` `2 = bovenkant van scherm,` `4 = rechts van scherm,` `8 = onderkant van scherm`
SPRITE(H, [#]n)	**Geeft de hoogte van sprite n terug. Deze functie is actief, ongeacht of de sprite momenteel wordt weergegeven (actief).**
SPRITE(L, [#]n)	**Geeft het laagnummer van actieve sprites nummer n terug.**
SPRITE(N)	**Geeft het aantal weergegeven (actieve) sprites terug.**
SPRITE(N, n)	**Geeft het aantal weergegeven (actieve) sprites op laag n terug.**
SPRITE(S)	**Geeft het nummer terug van de sprite die het laatst een botsing veroorzaakte.** **NB:** Als het geretourneerde getal nul is, is de botsing het resultaat van een **SPRITE SCROLL** commando en moet de **SPRITE(C…)** functie worden gebruikt om te bepalen hoeveel en welke sprites met elkaar in botsing zijn gekomen.

FUNCTIES	Beschrijving 284
SPRITE(V, spriteno1, spriteno 2)	**Retourneert de vector van 'spriteno1' naar 'spriteno2' in radialen.** De hoek is gebaseerd op de klok, dus als **'spriteno2'** boven **'spriteno1'** op het scherm staat, is het antwoord nul. Dit kan worden gebruikt op elk paar sprites dat zichtbaar is. Als een van beide sprite niet zichtbaar is, retourneert de **functie -1**. Dit is vooral handig na een botsing als de programmeur een differentiële beslissing wil nemen op basis van waar de botsing heeft plaatsgevonden. De hoek wordt berekend tussen het midden van elk van de sprites, die uiteraard verschillende afmetingen kunnen hebben.
SPRITE(T, [#]n)	**Geeft een bitmap terug die alle sprites toont die momenteel de gevraagde sprite raken.** **Bits 0-63** in het geretourneerde gehele getal vertegenwoordigen een huidige botsing met respectievelijk **sprites 1 tot 64**
SPRITE(V, [#]so1, [#]s2)	**Retourneert de vector van sprite 's1' naar 's2' in radialen.** De hoek is gebaseerd op de klok, dus als **'s2'** hoger is dan **'s1'** op het scherm, is het antwoord **nul**. Dit kan worden gebruikt op elk paar sprites dat zichtbaar is. Als een van beide sprite niet zichtbaar is, retourneert de **functie -1**. Dit is vooral handig na een botsing als de programmeur een differentiële beslissing wil nemen op basis van waar de botsing heeft plaatsgevonden. De hoek wordt berekend tussen het midden van elk van de sprites, die uiteraard verschillende afmetingen kunnen hebben.
SPRITE(W, [#]n)	**Geeft de breedte van sprite n terug. Deze functie is actief, ongeacht of de sprite momenteel wordt weergegeven (actief).**
SPRITE(X, [#]n)	**Geeft de X-coördinaat van sprite n terug.** Deze functie is alleen actief als de sprite momenteel wordt weergegeven (**actief**). **Retourneert anders 10000**.
SPRITE(Y, [#]n)	**Geeft de Y-coördinaat van sprite n terug.** Deze functie is alleen actief als de sprite momenteel wordt weergegeven (**actief**). **Retourneert anders 10000**.
SQR(getal)	**Geeft als resultaat de vierkantswortel van het argument 'getal'.**
STR$(getal)	**Retourneert een tekenreeks in de decimale (grondtal 10) representatie van 'getal'.**
STR$(getal, m)	**Als 'm' is opgegeven, worden er voldoende spaties toegevoegd aan het begin van het nummer om ervoor te zorgen dat het aantal tekens vóór de komma (inclusief het negatieve of positieve teken) bestaat uit ten minste 'm' tekens.** Als **'m' nul** is of het **nummer** heeft meer dan **'m'** significante cijfers er zijn geen opvulspaties toegevoegd. Als **'m' negatief** is, worden **positieve** getallen voorafgegaan door het plusteken en negatieve getallen met het negatieve symbool. Als **'m' positief** is, dan alleen het **negatieve symbool** wordt gebruikt.

FUNCTIES	Beschrijving 285
`STR$(getal, m, n)`	**'n' is het aantal cijfers dat nodig is om de decimale plaats te volgen.** Als het **nul** is de **tekenreeks** wordt **geretourneerd zonder** de **komma. Als het negatief is, output zal altijd het exponentiële formaat gebruiken met 'n' cijfers resolutie. Als 'n' is niet gespecificeerd, het aantal decimalen en het uitvoerformaat zullen variëren automatisch volgens het nummer.**
`STR$(getal, m, n, c$)`	**'c$' is een tekenreeks en indien opgegeven wordt het eerste teken van deze tekenreeks gebruikt als het opvulteken in plaats van een spatie (zie het 'm'-argument).** Voorbeelden: `STR$( 123.456)` retourneert `" 123.456"` `STR$(-123.456)` retourneert `"-123.456"` `STR$( 123.456, 1)` retourneert `" 123.456"` `STR$( 123.456,-1)` retourneert `"+123.456"` `STR$( 123.456, 6)` retourneert `" 123.456"` `STR$( 123.456,-6)` retourneert `"+123.456"` `STR$(-123.456, 6)` retourneert `"-123.456"` `STR$(-123.456, 6, 5)` retourneert `"-123.45600"` `STR$(-123.456, 6,-5)` retourneert `"-1.23456e+02"` `STR$( 53, 6)` retourneert `" 53 "` `STR$( 53, 6, 2)` retourneert `" 53.00 "` `STR$( 53, 6, 2,"*")` retourneert `"****53.00"`
`STR2BIN(type,string$ [,BIG])`	**Geeft een getal terug dat gelijk is aan de binaire representatie in 'string$'.** **'type'** kan zijn: **INT64** converteert een string van 8 bytes die een 64 bits geheel getal met teken vertegenwoordigt naar een geheel getal. **UINT64** converteert een tekenreeks van 8 bytes die een niet ondertekend 64 bits geheel getal vertegenwoordigt naar een geheel getal. **INT32** converteert een string van 4 bytes die een 32 bits geheel getal met teken vertegenwoordigt naar een geheel getal. **UINT32** converteert een tekenreeks van 4 bytes die een niet ondertekend 32 bits geheel getal vertegenwoordigt naar een geheel getal. **INT16** converteert een tekenreeks van 2 bytes die een 16 bits geheel getal met teken vertegenwoordigt naar een geheel getal. **UINT16** converteert een tekenreeks van 2 bytes die een niet ondertekend 16 bits geheel getal vertegenwoordigt naar een geheel getal. **INT8** converteert een tekenreeks van 1 byte die een 8 bits geheel getal met teken vertegenwoordigt naar een geheel getal. **UINT8** converteert een tekenreeks van 1 byte die een niet ondertekend 8 bits geheel getal vertegenwoordigt naar een geheel getal.

FUNCTIES	Beschrijving 286
	SINGLE converteert een string van 4 bytes die een enkele precisie-float voorstelt naar een float. **DOUBLE** converteert een string van 8 bytes die een enkele precisiefloat voorstelt naar een float. Standaard moet de string het getal in little-endian-formaat bevatten (d.w.z. de minst significante byte is de eerste in de string). De derde parameter instellen to **'BIG'** interpreteert de string in bigendian formaat (d.w.z. de meest significante byte is de eerste in de string). Deze functie maakt het gemakkelijk om gegevens uit binaire gegevensbestanden te lezen, te interpreteren getallen van sensoren of efficiënt binaire gegevens uit het flashgeheugen lezen chips. Er wordt een fout gegenereerd als de string de verkeerde lengte heeft voor de conversie gevraagd. **Zie ook de functie BIN2STR$.**
STRING$(nbr, ascii) of **STRING$(nbr, tekenreeks$)**	**Retourneert een tekenreeks 'nbr' bytes lang bestaande uit ofwel het eerste teken van string$ of het teken dat de ASCII-waarde 'ascii' vertegenwoordigt, wat een integer is of float-getal in het bereik van 0 tot 255.**
TAB(getal)	**Voert spaties uit totdat de kolom aangegeven met 'getal' is bereikt op de console-uitgang.**
TAN(getal)	**Geeft als resultaat de tangens van het argument 'getal' in radialen.**
TEMPR(pin)	**Retourneert de temperatuur gemeten door een DS18B20 temperatuursensor aangesloten op 'pin' (die niet geconfigureerd hoeft te worden).** De geretourneerde waarde is **graden C** met een standaardresolutie van **0,25ºC**. Als er **een fout is** tijdens **de meting**, is de geretourneerde **waarde.** De **tijd** die **nodig** is voor de algehele **meting is 200 ms** en **interrups** zullen gedurende deze **periode worden genegeerd**. Als alternatief het **TEMPR START** commando kan worden gebruikt **om de meting te starten** en **uw programma kan andere dingen doen terwijl de conversie vordert**. Wanneer deze functie **de waarde** wordt **genoemd** wordt dan **onmiddellijk geretourneerd**, ervan uitgaande dat de **conversieperiode is verstreken**. Als dit **niet het geval** is, **wacht** deze functie de rest van de conversietijd af **voordat u de waarde retourneert**. De **DS18B20** kan afzonderlijk worden gevoed door een voeding van **3V** tot **5V** of het kan werken op parasitaire kracht van de **PicoMite**. Zie de sectie Speciale hardwareapparaten voor meer details.
TIME$	**Retourneert de huidige tijd op basis van de interne klok van MMBasic als een tekenreeks in de vorm "HH:MM:SS" in 24 uurs notatie.** Bijvoorbeeld **"14:30:00"**. **Gebruik het commando TIME$ = om de huidige tijd in te stellen.**

FUNCTIES	Beschrijving 287
TIMER	**Geeft de verstreken tijd in milliseconden (bijv. 1/1000 van een seconde) sinds resetten.** De **timer** wordt op **nul** gezet bij het **opstarten** of bij het **opnieuw opstarten van de CPU** en u kunt ook **reset** het door **TIMER** als commando te gebruiken. Als het **niet specifiek wordt gereset**, zal het voor **altijd blijven optellen** (**het is een 64-bits getal** en zal daarom **alleen roll over naar nul na 200 miljoen jaar).**
TOUCH(X) of TOUCE(Y))	**Retourneert de X- of Y-coördinaat van de locatie die momenteel is aangeraakt op een LCD-scherm.** Als het **scherm niet** wordt **aangeraakt**, keert de functie **terug** naar **-1** **Deze functie is niet beschikbaar op de PicoMiteVGA.**
UCASE$(string$)	**Geeft als resultaat 'string$' geconverteerd naar hoofdletters.**
VAL(string$)	**Retourneert de numerieke waarde van de 'string$'.** Als **'string$'** ongeldig is nummer geeft de functie nul terug. Deze functie **herkent** het **&H** voorvoegsel **voor** een **hexadecimaal getal, &O voor octaal en &B voor binair.**

Verouderde opdrachten en functies

Deze commando's en functies zijn meestal opgenomen om te helpen bij het converteren van programma's die zijn geschreven voor Microsoft.

BASIC.

Voor nieuwe programma's moeten de overeenkomstige moderne commando's in MMBasic worden gebruikt.
Houd er rekening mee dat deze opdrachten in de toekomst kunnen worden verwijderd om geheugen voor andere functies te herstellen.

COMMANDO / FUNCTIES	Beschrijving 288
BITBANG	**Vervangen door het commando DEVICE.** Voor compatibiliteit kan **BITBANG** nog steeds in programma's worden gebruikt en wordt automatisch geconverteerd naar **DEVICE**
DEVICE CAMERA	**Gewijzigd in het CAMERA commando.**
DEVICE GAMEPAD	**Gewijzigd in het GAMEPAD commando.**
DEVICE HUMID	**Gewijzigd in het HUMID commando.**
DEVICE KEYPAD	**Gewijzigd in het KEYPAD commando.**
DEVICE MOUSE	**Gewijzigd in het MOUSE commando.**
DEVICE LCD	**Gewijzigd in het LCD commando.**
DEVICE WII	**Gewijzigd in het WII commando.**
DEVICE WS2812	**Gewijzigd in het WS2812 commando.**
GOSUB doel	**Start een subroutine-oproep naar het doel, wat een regelnummer of een label kan zijn.** De subroutine moet eindigen op **RETURN**. Nieuwe programma's moeten gedefinieerde subroutines gebruiken (d.w.z. **SUB...END SUB**).
IF voorwaarde THEN linnenbr	**Voor Microsoft-compatibiliteit wordt een GOTO aangenomen als de THEN instructie is gevolgd door een nummer.** Een label is ongeldig in deze constructie. Nieuwe programma's moeten gebruiken: `IF condition THEN GOTO linnenbr | label`
IRETURN	**Retourneert van een onderbreking wanneer de onderbrekingsbestemming een regelnummer was of een etiket.** Nieuwe programma's moeten een door de gebruiker gedefinieerde subroutine gebruiken als een **interrupt bestemming**. In dat geval zal **END SUB** of **EXIT SUB** een **return veroorzaken** van de onderbreking.
ON nbr GOTO | GOSUB doel[, doel, doel,.]	**ON ofwel vertakt (GOTO) of roept een subroutine (GOSUB) aan op basis van de afgeronde waarde van 'nbr', als het 1 is, wordt het eerste doelwit genoemd, als het 2 is, het tweede doel wordt gebeld, enz.** Het **doel** kan een **regelnummer** of een **label** zijn. **Nieuwe programma's** moeten **SELECT CASE gebruiken**.

COMMANDO / FUNCTIES	Beschrijving 289
MM.HPOS MM.VPOS	**Vervangen door MM.INFO(HPOS) en MM.INFO(VPOS).**
MM.VER	**MMBasic versienummer.** **Vervangen door MM.INFO(VER).**
MM.FONTHEIGHT MM.FONTWIDTH	**Gehele getallen die de hoogte en breedte van het huidige lettertype vertegenwoordigen (in pixels), worden behouden voor compatibiliteit.** **Let op:** Deze worden automatisch omgezet in: MM.INFO**(FONTHEIGHT)** en MM.INFO**(FONTWIDTH)** wanneer het programma wordt geladen.
POS	**Retourneert voor de console de huidige cursorpositie in de regel in tekens.**
RETURN	**RETURN sluit een door GOSUB aangeroepen subroutine af en keert terug naar de verklaring na de GOSUB.**

Bijlage A
Seriële communicatie

Er zijn twee seriële interfaces beschikbaar voor asynchrone seriële communicatie. Ze zijn gelabeld met **COM1:** en **COM2:**.

I/O pinnen

Voordat een seriële interface kan worden gebruikt, moeten de I/O-pinnen worden gedefinieerd met de volgende opdracht voor het eerste kanaal (aangeduid als **COM1**):

```
SETPIN rx, tx, COM1
```
Geldige pinnen zijn:
```
RX: GP01, GP13, GP17
TX: GP00, GP12, GP16, GP28
```

En de volgende opdracht voor het tweede kanaal (aangeduid als **COM2**):

```
SETPIN rx, tx, COM2
```
Geldige pinnen zijn:
```
RX: GP05, GP09, GP21
TX: GP04, GP08, GP20
```

TX zijn gegevens van de PicoMite en **RX** zijn gegevens ernaartoe.

Merk op dat **COM1** en **COM2** op de **WebMite** versie niet beschikbaar zijn op **GP20** tot **GP28.**

De signaalpolariteit is standaard voor apparaten die op **TTL** spanningen werken. Inactief is de spanning hoog, het startbit is de spanning laag, data gebruikt een hoge spanning voor logische 1 en het stopbit is de spanning hoog. Met deze signaalniveaus kunt u rechtstreeks verbinding maken met ap-paraten zoals GPS-modules (die over het algemeen **TTL** spanningsniveaus gebruiken).

Option instellingen

```
OPTION BAUDRATE nn
```

Stel de baudrate van de seriële console in (indien geconfigureerd).

```
OPTION SERIAL CONSOLE  uartapin, uartbpin
```

Geef op dat de console toegankelijk moet zijn via een hardwarepoort (in plaats van virtuele serieel via USB).
'uartapin' en **'uartbpin'** kunnen elk geldig paar **rx- en tx-** pinnen zijn **voor of COM1 of COM2.** De waarin ze zijn begonnen, is niet belangrijk. De snelheid is standaard ingesteld op **115200 baud**, maar kan worden gewijzigd bekijk de **OPTION BAUDRATE**.

```
OPTION SERIAL CONSOLE DISABLE
```

Keer terug naar de normale USB-console.

Commando's

Nadat de seriële poort is geopend, heeft deze een bijbehorend bestandsnummer en kunt u alle opdrachten gebruiken die werken met een bestandsnummer om naar/vanuit te lezen en te schrijven. Een seriële poort kan worden gesloten met het commando **CLOSE**.
Het volgende is een voorbeeld:

```
SETPIN GP13, GP16, COM1 ' Wijs de I/O-pinnen toe aan de eerste seriële poort.
OPEN "COM1:4800" AS #5  ' Open de eerste seriële poort met een snelheid 4800B
PRINT #5, "Hallo"       ' Stuur de string "Hallo" uit de seriële poort.
dat$ = INPUT$(20, #5)   ' Haal maximaal 20 tekens uit de seriële poort.
CLOSE #5                ' Sluit de seriële poort.
```

Het OPEN commando

Een seriële poort wordt geopend met het commando:

```
OPEN comspec$ AS #fnbr
```

'**fnbr**'	Is het te gebruiken dossiernummer. Het moet tussen **1** en **10** liggen. De **#** is optioneel.
'**comspec$**'	Is de communicatiespecificatie en is een string (het kan een stringvariabele zijn) die de seriële specificeert poort die moet worden geopend en optionele parameters. De standaardwaarde is **9600 baud, 8 databits, geen pariteit** en **1 stopbit.**

Het heeft de vorm: `"COMn: baud, buf, int, int-trigger, EVEN, ODD, S2, 7BIT"`
Waarbij:

- '**n**' Is het seriële poortnummer voor **COM1:** of **COM2:**.
- '**baud**' Is de **baudrate**. Dit kan elk getal zijn van **1200** tot ruim boven de **1000000**. De standaard waarde is **9600**.
- '**buf**' Is de grootte van de ontvangstbuffer in **bytes** (standaardgrootte is **256**). De verzendbuffer is vastgesteld op **256** bytes.
- '**int**' Is een **interrup**t subroutine die moet worden aangeroepen wanneer de seriële poort gegevens heeft ontvangen.
- '**int-trigger**' Is het aantal ontvangen tekens dat een interrupt zal activeren.

Alle parameters behalve de seriële poortnaam (**COMn:**) zijn optioneel. Als een parameter wordt weggelaten, dan zijn alle volgende parameters moeten ook worden weggelaten en de standaard-instellingen worden gebruikt.
Aan het einde van 'comspec$' kunnen vier opties worden toegevoegd. Dit zijn:

- '**S2**' Geeft aan dat er **2 stopbits** worden verzonden na elk verzonden teken.
- **EVEN** **EVEN** geeft aan dat een even pariteitsbit wordt toegepast, dit zal resulteren in een **9-bits** overdracht tenzij **7BIT is ingesteld**.
- **ODD** Geeft aan dat een **oneven pariteitsbit** wordt toegepast, dit resulteert in een overdracht van **9 bits** tenzij **7BIT is ingesteld**.
- **7BIT** Geeft aan dat er **7 bits aan gegevens zijn**. Dit wordt normaal gesproken gebruikt met **EVEN** of **ODD**.
- **INV** Specifies that the output signals will be inverted and input assumed to be inverted

Voorbeelden

Een seriële poort openen met alle standaardinstellingen:

```
OPEN "COM1:" AS #2
```

Een seriële poort openen die alleen de baudrate specificeert (4800 bits per seconde):

```
OPEN "COM1:4800" AS #1
```

Een seriële poort openen die de baudrate (9600 bits per seconde) specificeert en de grootte van de ontvangstbuffer (1KB):

```
OPEN "COM2:9600, 1024" AS #8
```

Hetzelfde als hierboven, maar met twee stopbits ingeschakeld:

```
OPEN "COM2:9600, 1024, S2" AS #8
```

Een voorbeeld dat alles specificeert, inclusief een interrupt, een interruptniveau en twee stopbits:

```
OPEN "COM2:19200, 1024, ComIntLabel, 256, S2" AS #5
```

Lezen en schrijven

Zodra een seriële poort is geopend, kunt u elke opdracht of functie gebruiken die een bestandsnummer gebruikt om uit te lezen en schrijven naar de poort. Gegevens die door de seriële poort worden ontvangen, worden automatisch in het geheugen gebufferd door MMBasic totdat het door het programma wordt gelezen en de functie **INPUT$()** de handigste manier is om dat te doen. Wanneer met behulp van de **INPUT$()** functie zal het aantal gespecificeerde karakters het maximum aantal karakters zijn geretourneerd, maar het kan minder zijn als er minder tekens in de ontvangstbuffer zijn. In feite zal de **INPUT$()** functie retourneer onmiddellijk een lege teken-reeks als er geen tekens beschikbaar zijn in de ontvangstbuffer.

De **LOC()** functie is ook handig, het geeft het aantal tekens terug dat wacht in de ontvangst-buffer (d.w.z. de maximum aantal tekens dat kan worden opgehaald door de functie **INPUT$()**). Merk op dat als de ontvangstbuffer overloopt met inkomende gegevens, zal de seriële poort auto-matisch de oudste gegevens weggooien om ruimte te maken voor de nieuwe data.

Het **PRINT** commando wordt gebruikt voor uitvoer naar een seriële poort en alle te verzenden gegevens worden in een geheugen bewaard buffer terwijl de seriële poort het verzendt. Dit betekent dat MMBasic doorgaat met het uitvoeren van de opdrachten na het **PRINT** commando terwijl de gegevens worden verzonden. De enige uitzondering is als de uitvoerbuffer vol is en in dat geval zal MMBasic pauzeren en wachten tot er voldoende ruimte is alvorens verder te gaan. De **LOF()** functie retourneert de hoeveelheid resterende ruimte in de verzendbuffer en u kunt dit gebruiken om te voorkomen dat het programma wacht tot er ruimte in de buffer vrijkomt.

Als je er zeker van wilt zijn dat alle gegevens zijn verzonden (misschien omdat je de reactie van de apparaat op afstand) moet u wachten tot de functie **LOF() 256** retourneert (de grootte van de verzendbuffer), wat aangeeft dat er niets meer valt te verzenden.

Seriële poorten kunnen worden gesloten met het **CLOSE** commando. Dit zal wachten tot de verzendbuffer leeg is, dan maak het geheugen vrij dat door de buffers wordt gebruikt en annuleer de onderbreking (indien ingesteld). Een seriële poort wordt ook automatisch gesloten wanneer commando's zoals **RUN** en **NEW** worden gegeven.

Onderbrekingen (Interrupts)

De interrupt subroutine (indien gespecificeerd) werkt hetzelfde als een algemene interrupt op een externe I/O-pin (zie het hoofdstuk **"Gebruik van de I/O-pinnen"** voor een beschrijving).
Als u interrupts gebruikt, moet u er rekening mee houden dat het enige tijd duurt voordat MMBasic op de interrupt reageert en er zouden in de tussentijd meer karakters kunnen zijn aangekomen, vooral bij hoge baudrates. Als je bijvoorbeeld hebben het onderbrekingsniveau gespecificeerd als **200** karakters en een buffer van **256** karakters dan vrij gemakkelijk de buffer zal zijn overgelopen tegen de tijd dat de interrupt subroutine de gegevens kan lezen. In dit geval zou de buffer moeten zijn verhoogd tot **512** tekens of meer.

Bijlage B
I2C communicatie

Er zijn twee **I2C kanalen**. Ze kunnen in **master**- of **slave-modus** werken.

I/O pinnen

Voordat de I2C-interface kan worden gebruikt, de I/O-pinnen moeten worden gedefinieerd met behulp van de volgende opdracht voor het eerste kanaal (aangeduid als **I2C**):

```
SETPIN sda, scl, I2C    of    OPTION SYSTEM I2C SDA, SCL (werkt alleen voor I2C)
```
Geldige pinnen zijn: **SDA: GP00, GP04, GP08, GP12, GP16, GP20, GP28**
SCL: GP01, GP05, GP09, GP13, GP17, GP21

Let op dat in de **WebMite** versie **I2C SDA** niet beschikbaar is op **GP28**.
En de volgende opdracht voor het tweede kanaal (aangeduid als **I2C2**):

```
SETPIN sda, scl, I2C2
```
Geldige pinnen zijn: **SDA: GP02, GP06, GP10, GP14, GP18, GP22, GP26**
SCL: GP03, GP07, GP11, GP15, GP19, GP27

Bij gebruik van de **I2C-bus boven 100 kHz** wordt de bekabeling tussen de apparaten belangrijk. Ideaal moeten de kabels zo kort mogelijk zijn (om de capaciteit te verminderen) en de data- en kloklijnen mogen niet naast elkaar lopen, maar hebben een aarddraad ertussen (om overspraak te verminderen).

Als de datalijn niet stabiel is wanneer de klok hoog is, of de kloklijn zenuwachtig is, kunnen de **I2C** randapparatuur **"verward"** raken en uiteindelijk de bus vergrendelen (normaal gesproken door de kloklijn laag te houden). Als u de hogere niet nodig hebt snelheden dan is werken op **100 kHz** de veiligste keuze.
De opdracht **I2C CHECK addr** kan worden gebruikt om te controleren of een apparaat aanwezig is op het adres **'addr'**. Dit stelt de alleen lezenvariabele **MM.I2C** in op **0** als een apparaat reageert of op **1** als er geen reactie is.

Errors

Na een **I2C** schrijf- of leesbewerking wordt de automatische variabele **MM.I2C** ingesteld om het resultaat als volgt aan te geven:

- **0 =** De opdracht is voltooid zonder fouten.
- **1 =** Een **NACK** antwoord ontvangen.
- **2 =** Opdracht is verlopen.

7-bits adressering

De standaardadressen die in deze opdrachten worden gebruikt, zijn **7-bits** adressen (zonder het lees-/schrijfbit). MMBasic voegt het lees-/schrijfbit toe en manipuleer het dienovereenkomstig tijdens overdrachten. Sommige leveranciers bieden 8-bits adressen die de lees-/schrijfbit bevatten. U kunt bepalen of dit het geval is omdat ze één adres zullen geven om naar het slave-apparaat te schrijven en een ander om van de slave te lezen. In deze situaties moet u alleen de bovenste zeven bits van het adres gebruiken. Bijvoorbeeld: Als het gelezen adres **9B (hex)** en het schrijfadres is **9A (hex)**, dan krijg je door alleen de bovenste zeven bits te gebruiken een adres van **4D (hex)**.
Een andere indicator dat een leverancier 8 bits adressen gebruikt in plaats van 7 bits adressen, is het controleren van het adresbereik.
Alle 7-bits adressen moeten tussen **08** en **77 (hex)** liggen. Als uw slave-adres groter is dan dit bereik, dan waarschijnlijk heeft uw leverancier een 8-bits adres opgegeven.

I2C Master Commando's

COMMANDO	Beschrijving 295
`I2C OPEN speed, timeout`	**Hiermee schakelt u de I2C module in master mode in.** Het I2C commando verwijst naar **kanaal 1** terwijl het commando I2C2 verwijst naar kanaal 2 met dezelfde syntaxis. **'speed'** is de klok-snelheid (in **KHz**) die gebruikt moet worden en moet **100**, **400** of **1000** zijn. **'timeout'** is een waarde in milliseconden waarna de mas-ter send en receive commands onderbroken worden als ze niet vol-tooid zijn. De minimumwaarde is **100**. Een waarde van **nul** schakelt de timeout uit (hoewel dit niet aanbevolen wordt).
`I2C WRITE addr, option, sendlen, senddata [,senddata]`	**Verstuur gegevens naar het I2C slave apparaat.** De I2C opdracht verwijst naar **kanaal 1**, terwijl de opdracht I2C2 verwijst naar **kanaal 2** met dezelfde syntaxis. **'addr'** is het I2C-adres van de **slave**. **'option'** kan **0** zijn voor normale werking of **1** om de controle over de bus te behouden na de opdracht (er wordt geen stopconditie verzonden bij voltooiing van de opdracht) **'sendlen'** is het aantal bytes dat moet worden verzonden. **'senddata'** is de te verzenden data. Dit kan op verschillende manieren worden gespeci-ficeerd (alle verzonden data wordt verzonden als bytes met een waarde tussen **0** en **255**): • De data kan worden geleverd als individuele bytes op de opdracht-regel. Voorbeeld: **I2C WRITE &H6F, 0, 3, &H23, &H43, &H25** • De data kan in een eendimensionale array staan die is gespecifi-ceerd met lege haakjes (d.w.z. geen dimensies). **'sendlen'** bytes van de array worden verzonden, beginnend met het eerste element. Voorbeeld: **I2C WRITE &H6F, 0, 3, ARRAY()** • De data kan een stringvariabele zijn (geen constante). Voorbeeld: **I2C WRITE &H6F, 0, 3, STRING$**
`I2C READ addr, option, rcvlen, rcvbuf`	**Haal gegevens op van het I2C slave apparaat.** De I2C opdracht verwijst naar **kanaal 1**, terwijl de opdracht I2C2 ver-wijst naar **kanaal 2** met dezelfde syntaxis. **'addr'** is het I2C-adres van de slave. **'option'** kan **0** zijn voor normale werking of **1** om de bus te behouden na de opdracht (er wordt geen stopconditie verzonden bij voltooiing van de opdracht) **'rcvlen'** is het aantal bytes dat moet worden ontvangen. **'rcvbuf'** is de variabele of array die wordt ge-bruikt om de ontvangen gegevens op te slaan. Dit kan zijn: • Een stringvariabele. Bytes worden opgeslagen als opeenvolgende tekens in de string. • Een eendimensionale array met getallen die worden opgegeven met lege haakjes. Ontvangen bytes worden opgeslagen in sequen-tiële elementen van de array, beginnend met de eerste. Voorbeeld: **I2C READ &H6F, 0, 3, ARRAY()** • Een normale numerieke variabele (in dit geval moet **'rcvlen' 1** zijn).
`I2C CLOSE`	**Schakelt de master I2C-module uit en zet de I/O-pinnen terug naar een "niet geconfigureerde" status.** Deze opdracht stuurt ook een stop als de bus nog steeds vastzit.

I2C Slave Commando's

COMMANDO	Beschrijving 296
`I2C SLAVE OPEN addr, send_int, rev_int`	**Schakelt de I2C module in slave-modus in.** De I2C opdracht verwijst naar **kanaal 1** terwijl de opdracht I2C2 verwijst naar **kanaal 2** met dezelfde syntaxis. **'addr'** is het slave I2C-adres. **'send_int'** is de subroutine die moet worden aangeroepen wanneer de module heeft gedetecteerd dat de master gegevens verwacht. **'rcv_int** is de subroutine die moet worden aangeroepen wanneer de module gegevens van de master heeft ontvangen. Let op dat dit wordt geactiveerd bij de eerste byte die wordt ontvangen, dus uw programma moet mogelijk wachten totdat alle gegevens zijn ontvangen.
`I2C SLAVE WRITE sendlen, senddata [,senddata]`	**Stuur de gegevens naar de I2C master.** De I2C opdracht verwijst naar **kanaal 1**, terwijl de opdracht I2C2 verwijst naar **kanaal 2** met dezelfde syntaxis. Deze opdracht moet worden gebruikt in de verzendinterrupt (d.w.z. in de subroutine **'send_int'** wanneer de master gegevens heeft opgevraagd). Als alternatief kan een vlag worden ingesteld in de interrupt subroutine en kan de opdracht worden aangeroepen vanuit de hoofdprogramma lus wanneer de vlag is ingesteld. **'sendlen'** is het aantal bytes dat moet worden verzonden. **'senddata'** is de te verzenden gegevens. Dit kan op verschillende manieren worden opgegeven, zie de **I2C Write** opdrachten voor meer informatie.
`I2C SLAVE READ rcvlen, rcvbuf, revd`	**Ontvang gegevens van het I2C masterapparaat.** De I2C opdracht verwijst naar **kanaal 1** terwijl de opdracht I2C2 verwijst naar **kanaal 2** met dezelfde syntaxis. Deze opdracht moet worden gebruikt in de ontvangstinterrupt (d.w.z. in de subroutine **'rcv_int'** wanneer de master gegevens heeft verzonden). Als alternatief kan een vlag worden ingesteld in de ontvangst-interrupt-subroutine en kan de opdracht worden aangeroepen vanuit de hoofdprogramma-lus wanneer de vlag is ingesteld. **'rcvlen'** is het maximale aantal bytes dat moet worden ontvangen. **'rcvbuf'** is de variabele om de gegevens te ontvangen. Dit kan op verschillende manieren worden opgegeven, zie de **I2C READ** opdrachten voor meer informatie. **'rcvd'** is een variabele die, na voltooiing van de opdracht, het werkelijke aantal ontvangen bytes bevat (wat kan verschillen van **'rcvlen'**).
`I2C SLAVE CLOSE`	**Schakelt de slave I2C module uit en zet de externe I/O-pinnen terug naar een "niet geconfigureerde" status.** Ze kunnen vervolgens worden geconfigureerd met **SETPIN**.

Voorbeelden

Als voorbeeld van een eenvoudige communicatie waarbij de PicoMite de master is, volgt het volgende programma zal de huidige tijd (uren en minuten) lezen en weergeven, bijgehouden door een **PCF8563 realtime klok** chip aangesloten op het **tweede I2C** kanaal:

```
DIM AS INTEGER RData(2)        ' Dit bevat ontvangen gegevens.
SETPIN GP6, GP5, I2C2          ' Deze regel niet als OPTION I2C ingevuld
I2C2 OPEN 100, 1000            ' Deze regel niet als OPTION I2C ingevuld
I2C2 WRITE &H51, 0, 1, 3       ' Zet het eerste register op 3.
I2C2 READ &H51, 0, 2, RData()  ' Lees twee registers.
I2C2 CLOSE                     ' Deze regel niet als OPTION I2C ingevuld
PRINT "Tijd is " RData(1) ":" RData(0)
```

Dit is een voorbeeld van communicatie tussen twee PicoMites waarbij de ene de master is en de andere is de slaaf.

Eerst de master:

```
SETPIN GP2, GP3, I2C2          ' Deze regel niet als OPTION I2C ingevuld
I2C2 OPEN 100, 1000            ' Deze regel niet als OPTION I2C ingevuld
i = 10
DO
  i = i + 1
  a$ = STR$(i)
  I2C2 WRITE &H50, 0, LEN(a$), a$
  PAUSE 200
  I2C2 READ &H50, 0, 8, a$
  PRINT a$
  PAUSE 200
LOOP
```

Dan de slaaf:

```
SETPIN GP2, GP3, I2C2          ' Deze regel niet als OPTION I2C ingevuld
I2C2 SLAVE OPEN &H50, tint, rint
DO : LOOP

SUB rint
  LOCAL count, a$
  I2C2 SLAVE READ 10, a$, count
  PRINT LEFT$(a$, count)
END SUB

SUB tint
  LOCAL a$ = Time$
  I2C2 SLAVE WRITE LEN(a$), a$
END SUB
```

Bijlage C
OneWire communicatie

Het **1-Wire protocol** is ontwikkeld door **Dallas Semiconductor** om met chips te communiceren via een enkele signaal lijn. Deze implementatie is voor MMBasic geschreven door Gerard Sexton.

Er zijn vier commando's die u kunt gebruiken:

```
ONEWIRE RESET pin                                    Reset de 1-draads bus.
ONEWIRE WRITE pin, vlag, lengte, data [, data...] Stuur een aantal bytes.
ONEWIRE READ  pin, vlag, lengte, data [, data...] Verkrijg een aantal bytes.
MM.ONEWIRE                Retourneert waar (1) als een apparaat is gevonden.
```

Waar:

pin - **De PicoMite I/O-pin die moet worden gebruikt**. Het kan elke pin zijn die in staat is tot digitale I/O.

vlag - **Een combinatie van de volgende opties**:

1 - Stuur reset voor commando

2 - Stuur reset na opdracht

4 - Alleen een bit verzenden/ontvangen in plaats van een byte aan gegevens

8 - Roep een sterke pull-up op na het commando (de pin wordt hoog gezet en open drain wordt uitgeschakeld)

lengte - **Lengte van de gegevens die moeten worden verzonden of ontvangen**.

data - **Gegevens om te verzenden of variabel om te ontvangen.**

Het aantal gegevensitems moet overeenkomen met de lengteparameter.

De automatische variabele **MM.ONEWIRE** retourneert waar als een apparaat is gevonden.

Nadat de opdracht is uitgevoerd, wordt de I/O-pin ingesteld op de niet geconfigureerde status, tenzij **vlagoptie 8** wordt gebruikt.
Wanneer een reset wordt aangevraagd, zal de automatische variabele **MM.ONEWIRE** de waarde **true** retourneren als er een apparaat is gevonden. Deze gebeurt met het **ONEWIRE RESET** commando en de **ONEWIRE READ** en **ONEWIRE WRITE** commando's als een reset is aangevraagd (**vlag = 1 of 2**).
Het **1-Wire protocol** wordt vaak gebruikt bij de communicatie met de **DS18B20** temperatuurmeet-sensor en om hulp in dat opzicht MMBasic bevat de functie **TEMPR()** die een handige methode biedt om direct het lezen van de temperatuur van een **DS18B20** zonder deze functies te gebruiken.

Voorbeeld 1:

DS18B20 met functie TEMPR(pin) en met OneWire routine functie TEMPW(pin)

```
' Functie TEMPR() om de temperatuur te krijgen van een Dallas DS18B20.
' De sensor wordt aangeloten op pin 9 = GP6. Bij OneWire geen GP nummer
Do
  Print
  Print "De DS18B20 met TEMPW(9): ";Str$(TEMPW(9),2,2);" ";Chr$(176);"C"
  ' Str$(TEMPW(pin), cijfers voor, cijfers na komma), Chr$(176)=graden.
  Print "De DS18B20 met TEMPR(GP6): ";Str$(TEMPR(GP6),2,2);" ";Chr$(176);"C"
  ' Str$(TEMPR(pin), cijfers voor, cijfers na komma), Chr$(176)=graden.
  Pause 5000
Loop

Function TEMPW(PinNbr)
  Local T1, T2, b, t
  OneWire Reset PinNbr                        ' Reset.
  OneWire Write PinNbr, 1, 2, &hCC, &h44      ' Start conversie.
  Pause 80                                    ' Dit geeft de toegangstijd weer.
  t = Timer
  Do
    If Timer - t > 1000 Then Error "Sensor reageert niet"
    OneWire Read PinNbr, 4 , 1 , b            ' Conversie gedaan?
  Loop Until b = 1
  OneWire Write PinNbr, 1, 2, &hCC, &hBE      ' Commando lees data
  OneWire Read  PinNbr, 2, 2, T1, T2          ' Verkrijg de data
  TEMPW = ((T2 And &b111) * 256 + T1) / 16
  If T2 And &b1000 Then TEMPW = -TEMPW        ' Aanpassen indien negatief
End Function
```

De uitvoer op Tera Term terminal:

```
De DS18B20 met TEMPW(4): 20.35 °C
De DS18B20 met TEMPR(4): 20.35 °C
```

Bijlage D
SPI-communicatie

Het communicatieprotocol Serial Peripheral Interface (SPI) wordt gebruikt voor het verzen-den en ontvangen van gegevens tussen geïntegreerde schakelingen. De PicoMite fungeert als de master (d.w.z. hij genereert de klok).

I/O-pinnen

Voordat een SPI-interface kan worden gebruikt, moeten de I/O-pinnen voor het kanaal als volgt worden toegewezen commando's. Voor het eerste kanaal (aangeduid als **SPI**) is dit:

```
SETPIN rx, tx, clk, SPI of SETPIN MISO, MOSI, CLK, SPI
        Geldige SPI pinnen zijn:  DI, MISO, RX: GP00, GP04, GP16, GP20
                                  DO, MOSI, TX: GP03, GP07, GP19, GP23
                                       CLK: GP02, GP06, GP18, GP22
                                       CSn: GP01, GP05, GP17, GP21
```

En het volgende commando voor het tweede kanaal (aangeduid als **SPI2**) is:

```
SETPIN rx, tx, clk, SPI2 of SETPIN MISO, MOSI, CLK, SPI2
        Geldige SPI pinnen zijn:  DI, MISO, RX: GP08, GP12, GP24, GP28
                                  DO, MOSI, TX: GP11, GP15, GP27
                                       CLK: GP10, GP14, GP26
                                       CSn: GP09, GP13, GP25, GP29
```

TX zijn gegevens van de PicoMite en **RX** zijn gegevens er naar toe.

```
Master Picomite SPI0 pin GP16 = Rx(MISO) naar Slave SDCARD SPI pin Tx(MISO)
Master PicoMite SPI0 pin GP19 = Tx(MOSI) naar Slave SDCARD SPI pin Rx(MOSI)
Master PicoMite SPI0 pin GP18 =   CLK    naar Slave SDCARD SPI pin   CLK
Master PicoMite SPI0 pin GP17 =   CSn    naar Slave SDCARD SPI pin   CSn
```

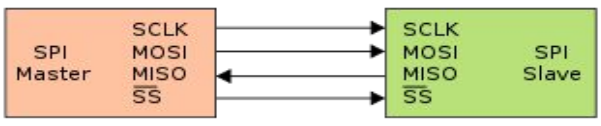
SPI openen

Om de SPI-functie te gebruiken, moet eerst het SPI-kanaal worden geopend.
De syntaxis voor het openen van het eerste SPI-kanaal is (gebruik SPI2 voor het tweede kanaal):

```
SPI OPEN speed, mode, bits
```

Waar:

- **'speed'** **Is de snelheid van de klok**. Het is een getal dat de kloksnelheid in **Hz** weergeeft.
- **'mode'** **Is een enkel numeriek cijfer dat de transmissiemodus vertegenwoordigt** – zie Transmissieformaat hieronder.
- **'bits'** **Is het aantal bits dat moet worden verzonden/ontvangen**. Dit kan elk getal zijn in het bereik van **4 tot 16 bits**.
- **Het is de verantwoordelijkheid van het programma om de CS** (chip select) pin indien nodig apart te manipuleren.



SPI: Enkele master naar enkele slave.

Verzendformaat

Het meest significante bit wordt als eerste verzonden en ontvangen. Het formaat van de verzending kan worden gespecificeerd door de **'mode'** zoals hieronder weergegeven. **Mode 0** is het meest gebruikte formaat.

Mode	Beschrijving	CPOL	CPHA
0	Klok is actief hoog, data reageerd op een stijgende flank en de output op een dalende flank.	0	0
1	Klok is actief hoog, data reageerd op een dalende flank en de output op een stijgende flank.	0	1
2	Klok is actief laag, data reageerd op een dalende flank en de output op een stijgende flank.	1	0
3	Klok is actief laag, data reageerd op een stijgende flank en de output op een dalende flank.	1	1

Voor een meer volledige uitleg zie: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Standaard verzenden/ontvangen

Wanneer het eerste SPI-kanaal open is, kunnen gegevens worden verzonden en ontvangen met behulp van de SPI-functie (gebruik SPI2 voor de tweede kanaal). De syntaxis is:

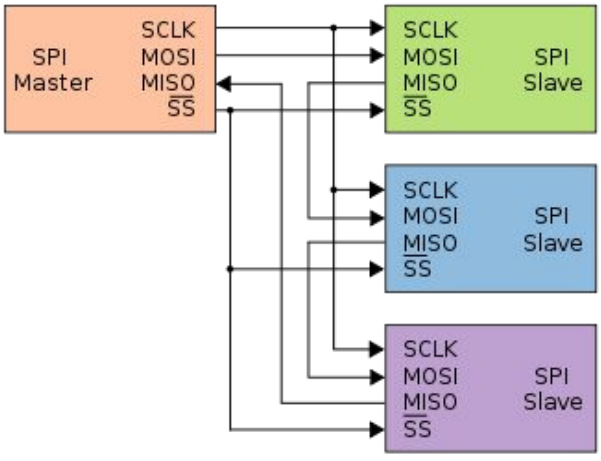
```
received_data = SPI(data_to_send)
```

Houd er rekening mee dat een enkele **SPI** transactie gegevens verzendt en tegelijkertijd gegevens van de slave ontvangt.

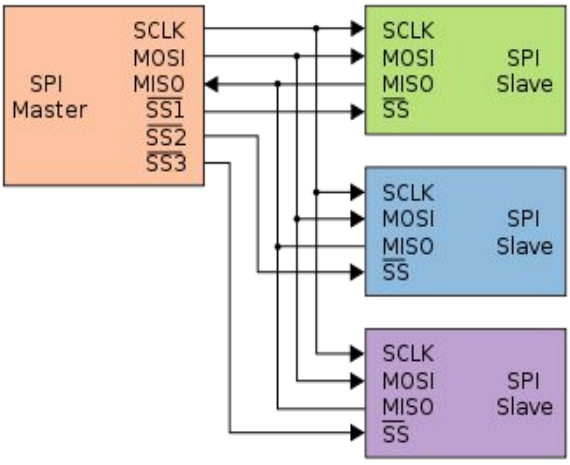
'**data_to_send**' **Zijn de gegevens die moeten worden verzonden en de functie retourneert de gegevens die tijdens de transactie zijn ontvangen.**

'**data_to_send**' **Kan een geheel getal of een variabele met drijvende komma of een constante zijn**.

Als u geen gegevens wilt verzenden (d.w.z. u wilt alleen ontvangen), kan elk nummer (bijv. nul) worden gebruikt voor de gegevens te verzenden. Evenzo, als u de ontvangen gegevens niet wilt gebruiken, kunt u deze aan een variabele toewijzen en negeren.



SPI-bus: master en drie onafhankelijke slaves



SPI-bus: master en coöperatieve slaves

Bulk verzenden/ontvangen

Gegevens kunnen ook in bulk worden verzonden (gebruik SPI2 voor het tweede kanaal):

```
  SPI WRITE nbr, data1, data2, data3, ... etc
```

of

```
  SPI WRITE nbr, string$
```

of

```
  SPI WRITE nbr, array()
```

In de eerste methode is '**nbr**' het aantal gegevensitems dat moet worden verzonden en de gegevens zijn de uitdrukkingen in de argumentenlijst (d.w.z. **'data1'**, **data2'** enz.). De gegevens kunnen een geheel getal of een variabele met drijvende komma of een constante zijn.
Bij de tweede of derde methode die hierboven wordt vermeld, bevinden de te verzenden gegevens zich in de '**string$**' of de inhoud van '**array()**' (**dit moet een array met één dimensie zijn** van gehele getallen of getallen met drijvende komma). De snaarlengte, of de grootte van de array moet gelijk zijn aan of groter zijn dan **nbr**. Alle gegevens die door de slave worden geretourneerd, worden weggegooid.

Gegevens kunnen ook in **bulk** worden ontvangen (gebruik SPI2 voor het tweede kanaal):

```
  SPI READ nbr, array()
```

Waar '**nbr**' het aantal te ontvangen gegevensitems is en **array()** een integer-array met één dimensie is waarbij de ontvangen gegevensitems worden opgeslagen. Dit commando verzendt nullen tijdens het lezen van de gegevens van de slave.

SPI Sluiten

Indien nodig kan het eerste SPI-kanaal als volgt worden gesloten (de I/O-pinnen worden op inactief gezet):

```
  SPI CLOSE
```

Gebruik SPI2 voor het tweede kanaal.

OPTION SYSTEM SPI

```
  OPTION SYSTEM SPI CLKpin, MOSIpin, MISOpin
```

Specificeer de SPI-poort en pinnen voor gebruik door systeemapparaten (SD-kaart, LCD-paneel, enz.). De PicoMite gebruikt een specifieke hardware **SPI poort** voor systeemapparaten, de ander voor de gebruiker overlatend. Dit commando specificeert welke pinnen zijn worden gebruikt, en dus welke van de **SPI-poorten** moet worden gebruikt. de pinnen toegewezen aan de **SYSTEM SPI** zal niet beschikbaar zijn voor andere MMBasic commando's.

```
SPI: RX  = MISO  MISO = Master Input Slave Output
SPI: TX  = MOSI  MOSI = Master Output Slave Input
SPI: CLK = SCLK   CLK = Klok signaal
SPI: SS  = CSn    CSn = Chip Select
```

Voorbeeld 1:

Het volgende voorbeeld laat zien hoe u de SPI-poort gebruikt voor algemene I/O. Het stuurt een commando 80 (hex) en ontvang twee bytes van het slave SPI-apparaat met behulp van de standaard verzend-/ontvangstfunctie:

```
PIN(10) = 1: SETPIN 10, DOUT          ' Pin 10 wordt gebruikt als het
                                      ' activeringssignaal.
SETPIN GP20, GP4, GP1, SPI            ' Wijs de I/O-pinnen toe.
SPI OPEN 5000000, 3, 8                ' Snelheid is 5 MHz en de gegevens-
                                      ' grootte is 8 bits.
PIN(10) = 0                           ' Bevestig de activeringsregel
                                      ' (actief laag).
junk = SPI(&H80)                      ' Stuur de opdracht en negeer de
                                      ' return.
byte1 = SPI(0)                        ' Haal de eerste byte van de slave.
byte2 = SPI(0)                        ' Haal de tweede byte van de slave.
PIN(10) = 1                           ' Deselecteer de slave.
SPI CLOSE                             ' En sluit het kanaal.
```

Het volgende is vergelijkbaar met het bovenstaande voorbeeld, maar deze keer wordt de overdracht gedaan met behulp van de bulk commando's verzenden/ontvangen:

```
OPTIE BASIS 1                         ' Onze array begint met de index 1.
DIM data%(2)                          ' Definieer de array voor het ont-
                                      ' vangen van de data.
SETPIN GP20, GP4, GP1, SPI            ' Wijs de I/O-pinnen toe.
PIN(10) = 1: SETPIN 10, DOUT          ' Pin 10 wordt gebruikt als het
                                      ' activeringssignaal.
SPI OPEN 5000000, 3, 8                ' Snelheid is 5 MHz, 8 bits data.
PIN(10) = 0                           ' Bevestig de activeringsregel
                                      ' (actief laag).
SPI WRITE 1, &H80                     ' Stuur de opdracht.
SPI READ 2, data%()                   ' Haal twee bytes van de slave.
PIN(10) = 1                           ' Deselecteer de slave.
SPI CLOSE                             ' En sluit het kanaal.
```

Bijlage E
Regex syntaxis

De alternatieve vormen van de functies **INSTR()** en **LINSTR()** kunnen een reguliere expressie als zoekpatroon gebruiken.
De alternatieve vorm van de opdrachten zijn:

```
INSTR([start],tekst$, zoek$ [,grootte])
LINSTR(tekst%(),zoek$ [,start] [,grootte]
```

In beide gevallen zorgt het specificeren van de parameter size ervoor dat de firmware de zoekreeks als normaal interpreteert uitdrukking. De parameter size is een drijvende-kommavariabele die door de firmware wordt gebruikt om de grootte van a terug te geven bijpassende string. Als de variabele niet bestaat, wordt deze gemaakt. Zoals geïmplementeerd in MMBasic moet u de start- en grootte waarden geretourneerd aan de functie **MID$** om de overeenkomende tekenreeks te extraheren. bijvoorbeeld,

```
IF start THEN match$ = MID$(tekst$, start, grootte) ELSE match$ = "" ENDIF
```

De bibliotheek die voor de reguliere expressies wordt gebruikt, implementeert POSIX-concept P1003.2/D11.2, met uitzondering van enkele van de internationalisering kenmerken.
Zie: http://mirror.math.princeton.edu/pub/oldlinux/Linux.old/Ref-docs/POSIX/all.pdf sectie 2.8 voor details over het maken van reguliere expressies of andere online tutorials.
De syntaxis van reguliere expressies kan enigszins variëren afhankelijk van de verschillende implementaties. Dit document is een samenvatting van de syntaxis en ondersteunde bewerkingen die worden gebruikt in de MMBasic-implementatie.

Haakjes

^	Begin van de tekenreeks
$	Einde van de reeks
\b	Woordgrens
\B	Geen woordgrens
\<	Begin van het woord
\>	Einde van het woord

Kwalificaties

*	0 of meer (niet ontsnapt)
\+	1 of meer
\?	0 of 1
\{3\}	Precies 3
\{3,\}	3 of meer
\{3,5\}	3, 4 of 5

Groepen en bereiken

(a|b)	a of b
\(…\)	Groep
[abc]	Bereik (a of b of c)
[^abc]	Niet (a of b of c]
[a-q]	Kleine letters a tot en met q
[A-Q]	Hoofdletters A tot en met Q
[0-7]	Cijfers van 0 tot 7

Ontsnappingen vereist om normale karakters te matchen

```
\^          Passend bij ^ (caret)
\.          Matchen. (punt)
\*          Passend bij * (asterix)
\$          Passend bij $ (dollar)
\[          Passend bij [ (linker haakje)
\\          Passend bij \ (backslash)
```

Ontsnapt met speciale functies

```
\+          Zie Kwantificatoren
\?          Zie Kwantificatoren
\{          Zie Kwantificatoren
\}          Zie Kwantificatoren
\|          Zie Groepen en bereiken
\(          Zie Groepen en bereiken
\)          Zie Groepen en bereiken
\w          Zie Tekenklassen
```

Karakter klassen

```
\w          Cijfers,letters en _
[:word:]    Cijfers,letters en _
[:upper:]   Hoofdletters_
[:lower:]   Kleine letters_
[:alpha:]   Alle letters
[:alnum:]   Cijfers en letters
[:digit:]   Cijfers
[:xdigit:]  Hexidecimale cijfers
[:punct:]   Puntuatie
[:blank:]   Spatie en tab
[:space:]   Lege tekens
[:cntrl:]   Beheer karakters
[:graph:]   Gedrukte tekens
[:print:]   Afgedrukte tekens en spaties
```

Voorbeeldexpressie die overeenkomt met een IP-adres dat zich binnen een woordgrens bevindt.

```
"\<[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9] \{1,3\}\>"
```

Bijlage F
Het PIO programmeer pakket

Inleiding tot de PIO

De **RP2040** en **RP2350** hebben veel ingebouwde randapparatuur zoals **PWM, UART, ADC, SPI**. Met behulp van PIO's wel is het mogelijk om gespecialiseerde functies/randapparatuur toe te voegen, zoals zeer nauwkeurige seriële data-interfaces en bitstreams.
PIO's kunnen worden gezien als ingekorte, zeer gespecialiseerde CPU-kernen. De **RP2040** bevat **twee PIO blokken** terwijl de **RP2350 drie blokken** heeft. MMBasic verwijst naar hen als **PIO0, PIO1** en **PIO2** in overeenstemming met de **Raspberry Pi documentatie**. De PIO's werken volledig onafhankelijk van het hoofdsysteem en van elkaar werken extreem snel, met een doorvoer van **maximaal 32 bits tijdens elke klokcyclus.**

PIO's implementeren state machines. Voordat een state machine zijn programma kan uitvoeren, moet het programma dat zijn naar het PIO-geheugen geschreven en de state machine moet worden geconfigureerd.

In deze bijlage wordt beschreven welke ondersteuning MMBasic kan geven bij het gebruik van PIO. Er staat geen uitleg hoe dat moet schrijf PIO statemachine programma's. Voor een beter begrip van hoe de PIO statemachines werken, raadpleegt u de volgende draad :

"PIO uitgelegd PICOMITE" op het thebackshed.com forum:

https://www.thebackshed.com/forum/ViewTopic.php?FID=16&TID=15385

Overzicht van PIO

Eén enkel PIO-blok heeft vier onafhankelijke state machines.
Alle vier de state machines delen één enkele 32 instructie programmagebied van flash geheugen. Dit geheugen heeft alleen schrijftoegang vanuit het hoofdsysteem, maar heeft vier leestoegang poorten, één voor elke statemachine, zodat iedereen er onafhankelijk en op zijn eigen snelheid toegang toe heeft.
Elke statemachine heeft een eigen **programma teller**.
Elke statemachine heeft ook **twee 32-bits "kladblok" registers, X en Y,** die kunnen worden gebruikt als tijdelijke gegevens opslag.
I/O-pinnen zijn toegankelijk via een **input/output mapping module** die toegang heeft tot 32 pinnen (maar beperkt tot 30 voor de **RP2040**). Alle statemachines hebben onafhankelijk en gelijktijdig toegang tot alle pinnen.
Het systeem kan gegevens **schrijven naar** het **invoereinde van een TX FIFO-buffer** van **4 woorden** en **32 bits breed**. De statemachine kan dat wel gebruik vervolgens **pull** om het uitgangswoord van de **FIFO** naar de **OSR** (**Output Shift Register**) te verplaatsen. Het kan ook nuttig zijn **verschuif 1-32 bits tegelijk** van de **OSR** naar de **uitvoer**toewijzings module of andere bestemmingen.
AUTOPULL kan worden gebruikt om automatisch gegevens **op te halen** totdat de **TX FIFO leeg is of** een vooraf ingesteld niveau bereikt.
Het systeem kan gegevens **lezen** van het **uitgangseinde** van een **4-woord 32-bit brede RX FIFO**-buffer. De statemachine kan dat wel gebruik vervolgens in om 1-32 bits aan gegevens tegelijk van de **input mapping module** naar de **ISR** te verschuiven (**Input Shift Register**). Het kan ook **push** gebruiken om de inhoud van de **ISR** naar de **FIFO** te verplaatsen.
AUTOPUSH kan hiervoor worden gebruikt automatisch gegevens **pushen** totdat de **RX FIFO vol is of** een vooraf ingesteld **niveau** bereikt.

De FIFO-buffers kunnen opnieuw worden geconfigureerd om een FIFO van 8 woorden en 32 bits in een enkele richting in een enkele richting te vormen. De buffers maken het mogelijk dat gegevens van en naar de statemachines worden doorgegeven zonder het systeem of de statemachine moeten wachten op de ander.
Aan elk van de vier statemachines in de **PIO** zijn vier registers gekoppeld:

- **CLKDIV** Is de klokdeler, die een 16-bits gehele deler en een 8-bits fractionele deler heeft. Dit bepaalt hoe snel de statemachine draait. Het splitst zich af van de hoofdsysteemklok.
- **EXECCTRL** Bevat informatie die de vertaling en uitvoering van het programmageheugen regelt.
- **SHIFTCTRL** Regelt de indeling en het gebruik van de schuifregisters.
- **PINCTRL** Bepaalt welke en hoe de GPIO-pinnen worden gebruikt.

De vier statemachines van een **PIO** hebben gedeelde toegang tot het blok van 8 interrupt vlaggen. Elke statemachine kan dit gebruiken welke vlag dan ook. Ze kunnen instellen, resetten of wachten tot ze veranderen. Op deze manier kunnen ze synchroon lopen als vereist. De onderste vier vlaggen zijn ook toegankelijk van en naar het hoofdsysteem, zodat de **PIO** kan worden bestuurd pass onderbreekt terug.
DMA kan worden gebruikt om informatie van en naar het PIO-blok door te geven via de FIFO van-uit het geheugen van de **RP2040**.
Een **PIO** heeft negen mogelijke programmeerinstructies, maar er kunnen veel variaties op elke instructie bestaan. Bijvoorbeeld, **Mov** kan **maximaal 8 bronnen**, **8 bestemmingen** en **3 procesbe-werkingen** hebben tijdens het kopiëren, met optionele vertraging en/of side-set operaties!

- **Jmp** Naar een absoluut adres in het programmageheugen springen als een voorwaarde waar is (of direct).
- **Wait** Wacht tot de werking van de statusmachine wordt geblokkeerd totdat een voor-waarde waar is.
- **In** **Verschuift een aantal bits** van een bron naar de **ISR**. (Input Shift Register)
- **Out** **Verschuift een aantal bits** uit de **OSR** naar een bestemming. (Output Shift Register)
- **Push** Duw de inhoud van de **ISR** naar de **RX FIFO** als een enkel **32-bits woord**.
- **Pull** Laad een **32-bits woord** van de **TX FIFO** naar de **OSR**.
- **Mov** Kopieer de data van een bron naar een side.
- **Irq** Een interruptvlag instellen of wissen.
- **Set** Schrijf onmiddellijk gegevens naar een side.

Instructies zijn allemaal 16-bits en bevatten zowel de instructie als alle bijbehorende gegevens. Alle instructies werken in 1 klokcyclus, maar het is mogelijk om een vertraging van verschillende inac-tieve klokcycli in te voeren tussen een instructie en de volgende.
Bovendien is er een faciliteit genaamd **"side set"**, waarmee een waarde naar een vooraf gedefi-nieerde uitvoer kan worden geschreven pinnen terwijl een instructie uit het geheugen wordt ge-lezen. Dit is transparant voor het programma.

PIO programmeren

PicoMite-firmware programmeert het PIO-statusmachinegeheugen met behulp van een van de drie methoden. Elke methode zal zijn uitgelegd met een voorbeeld van exact hetzelfde programma dat een van de GPIO-pinnen van de Raspberry Pi schakelt Pico. Welke GPIO-pin wordt omge-schakeld, wordt bepaald door de configuratie, niet door het programma zelf.

PIO ASSEMBLE

Dit commando wordt gebruikt om de ingebouwde assembler te gebruiken om het programma op basis van geheugensteuntjes te genereren en het vervolgens te schrijven rechtstreeks in het PIO-geheugen.

```
PIO ASSEMBLE 1,".program test"      ' Een programma moet een naam hebben.
PIO ASSEMBLE 1,".line 0"            ' Start het programma op regel 0.
PIO ASSEMBLE 1,"SET PINDIRS, 1"     ' SET de GPIO regel op uitvoer.
PIO ASSEMBLE 1,"label:"             ' Definieer een label met de naam "label"
PIO ASSEMBLE 1,"SET PINS, 1"        ' ZET de GPIO pin hoog.
PIO ASSEMBLE 1,"SET PINS, 0"        ' ZET de GPIO pin laag.
PIO ASSEMBLE 1,"JMP label"          ' Spring naar "label"
PIO ASSEMBLE 1,".end program LIST" ' Einde programma, List = toon resultaat.
```

PIO PROGRAMMA REGELS

Met dit commando kunnen **16-bits** waarden op individuele regels (locaties) in het PIO-geheugen worden geprogrammeerd.

```
PIO PROGRAM LINE 1,0,&hE081         ' SET pin uitgang.
PIO PROGRAM LINE 1,1,&hE001         ' SET pin hoog.
PIO PROGRAM LINE 1,2,&hE000         ' SET pin laag.
PIO PROGRAM LINE 1,3,&h0001         ' JMP naar regel 1.
```

Met deze opdracht worden alle **32 regels** in **één PIO** vanuit een array geschreven. Dit is handig als er fouten in een PIO-programma zijn opgelost. Het is uiterst compact.

```
DIM a%(7)=(&h0001E0000E001E081,0,0,0,0,0,0,0)
PIO PROGRAM 1,a%()
```

PIO configureren

De PicoMite-firmware kan elke statusmachine afzonderlijk configureren. Configuratie staat 2 statusmachines toe om exact dezelfde programmalijnen uit te voeren (bijvoorbeeld een SPI-interface) maar te werken met verschillende **GPIO**-pinnen en op verschillende snelheden. Er zijn verschillende configuratievelden.

FREQUENTIE

PicoMite-firmware bevat een standaardconfiguratie voor elk configuratieveld, behalve de frequentie. De frequentie wordt ingesteld door een 16 bit **CLKDIV**-deler van de ARM-klok. Voorbeeld: wanneer **OPTION CPUSPEED 126000** is ingesteld, kan de PIO werken op snelheden tussen **126 MHz** en **1,922 kHz** (**126000000 / 65536**). Houd daar rekening mee hogere CPU-snelheden (overklokken) hebben een directe invloed op de frequentie van de statusmachine.

PIN CONTROL

PicoMite-firmware stelt de **GPIO**-pinnen standaard in voor gebruik door MMBasic. Dat de **PIO** eigenaar wordt van een **GPIO**-pin MMBasic moet het aan PIO toewijzen, zoals hieronder.

```
SETPIN GPxx, PIOx     (bijv. SETPIN gp0, pio1)
```

Een statusmachine kan de status van een pin-**SET** (**SET** is een statusmachine-instructie), maar kan ook seriële gegevens uitvoeren naar een of meer **GPIO**-pinnen met behulp van de **OUT**-instructie. Of lees seriële gegevens met behulp van de **IN**-instructie. En GPIO pinnen kunnen worden ingesteld als neveneffect van elke staatsmachine-instructie (**SIDE SET**). Voor elke interfacemethode, verschillende pinnen kunnen worden toegewezen aan de statusmachine.

Het is belangrijk om te begrijpen dat deze instructies op opeenvolgende pinnen werken. Dit betekent dat er een bereik is aantal pinnen dat kan worden beheerd, beginnend bij het laagste GPx-pinnummer (bijv. GP0), en pinnen ernaast kunnen worden inbegrepen (tot 5 pinnen in totaal). **GP0,GP1,GP2** zijn dus een geldige set IO-pinnen. **GP0,GP1,GP6** niet. Overweeg dit bij het ontwerpen van een **PIO**-applicatie.
Het toewijzen van GPIO-pinnen aan een statusmachine maakt gebruik van de PIO-helperfunctie **PINCTRL**:

PIO(PINCTRL a,b,c,d,e,f,g)

a/	Het aantal **SIDE SET** pins	(0.....4),	**SIDE SET** kan 5 pinnen tegelijk schrijven.
b/	Het aantal **SET** pins	(0.....4),	**SET** kan 5 pinnen tegelijk schrijven.
c/	Het aantal **OUT** pins	(0....31),	**OUT** kan 32 pinnen tegelijk schrijven.
d/	De laagste pin voor **IN** pins	(GP0..GP31)	**IN** kan maximaal 32 pinnen tegelijk lezen.
e/	De laagste pin voor **SIDE SET**	(GP0..GP31)	
f/	De laagste pin voor **SET**	(GP0..GP31)	
g/	De laagste pin voor **OUT**	(GP0..GP31)	

Bereiken voor de verschillende functies kunnen elkaar overlappen, identiek zijn of aangrenzend zijn.

EXECUTE CONTROL

Het uitvoeringscontroleregister **EXECCTRL** configureert de programmastroom. Er is een veld dat een **GPIO** verbindt vastmaken aan een voorwaardelijke sprong (**JMP**-instructie) en velden die het regeladres van de hoofdprogrammalus bevatten begin (**.WRAP TARGET**) en einde (**.WRAP**).
Als we willen dat de programmastroom verandert als reactie op een **GPIO**-pinstatus, wordt een **JMP**-pincode gebruikt. De **JMP**-pin is toegewezen in de execute control-configuratie (er kan slechts 1 pin per statusmachine zijn) en de **JMP** gebeurt alleen als de pin hoog is).
Het staatsmachineprogramma begint bij het begin en loopt door tot het einde. In het bovenstaande demo programma is de programma loopt van het einde naar het begin met behulp van een (onvoorwaardelijke) **JMP**-instructie. Een alternatieve manier om te gebruiken de **JMP**-instructie definieert het begin van de lus (**WRAP TARGET** = regel 1) en het einde van de lus (**WRAP** = regel 2) en configureer de statusmachine om deze instructies alleen tussendoor uit te voeren.
De **JMP** instructie in regel 3 is verouderd als **WRAP**/**WRAP TARGET** wordt gebruikt.

PIO(EXECCTRL a,b,c)

a/	De **GPIO** pin voor voorwaardelijke **JMP**	(bijv. GP0)
b/	Het **WRAP TARGET** regelnummer	(bijvoorbeeld 1)
c/	Het **WRAP** regelnummer	(bijvoorbeeld 2)

SHIFT CONTROL

De **IN**- en **OUT**-instructies verschuiven gegevens van het **FIFO**-register naar de **GPIO**-pinnen.
Tussen MMBasic en de **PIO, 32-bits woorden** kunnen worden gecommuniceerd. Omdat zowel de ARM-kernen als de PIO-microcontrollers werken zelfstandig worden de gegevens uitgewisseld via FIFO's. De ARM (MMBasic) plaatst gegevens in de FIFO, PIO leest het. Hierbij wordt gebruik gemaakt van de **TX FIFO**. Andersom wordt de **RX FIFO** gebruikt. De FIFO's zijn normaal gesproken 4 woorden diep, maar kan worden geconfigureerd tot een enkele 8-woord diepe RX of TX FIFO.
De PIO kan gegevens **IN** de **RX FIFO** "verschuiven" vanaf de MSB-zijde of vanaf de LSB-zijde.
Dat wordt ingesteld met de **IN SHIFTDIR-bit**. Vergelijkbaar met de **OUT SHIFTDIR-bit** voor **OUT**-gegevens. De autopull- en autopush-vlaggen in combinatie met de **pull**- en **down** niveaus bepalen wanneer FIFO wordt bijgevuld.

In de **RP2350** kunnen de **FIFO's** ook als individuele registers worden gebruikt, waardoor een flexibelere communicatie tussen de twee mogelijk wordt MMBasic en de staatsmachines.
Dit wordt bereikt door **FJOIN_RX_GET** en **FJOIN_RX_PUT** in het **SHIFTCTRL-register**.

PIO(SHIFTCTRLa,b,c,d,e,f,g,h)

a/	**Push threshold**	(laat voorlopig 0 staan)
b/	**Pull threshold**	(laat voorlopig 0 staan)
c/	**Autopush**	(laat voorlopig 0 staan)
d/	**Autopull**	(laat voorlopig 0 staan)
e/	**IN shiftdir**	(1 = shift MSB, 0 = shift LSB)
f/	**OUT shiftdir**	(1 = MSB verschuiven, 0 = LSB verschuiven)
g/	**Fjoin_tx**	(voeg TX- en RX-fifo samen tot 1 RX-fifo)
h/	**Fjoin_rx**	(voeg TX- en RX-fifo samen tot 1 TX-fifo)
i/	**Fjoin_rx_get**	(1=ARM schrijft individuele FIFO-registers, alleen 2350)
j/	**Fjoin_rx_put**	(1=ARM leest individuele FIFO-registers, alleen 2350)

DE CONFIGURATIE VAN DE STATE MACHINE SCHRIJVEN

Een statusmachineconfiguratie wordt geschreven met behulp van de opdracht:

PIO INIT MACHINE a,b,c,d,e,f,g

a/	De **PIO**	(0 of 1)
b/	Het **status machine nr.**	(0...3)
c/	**Frequentie**	(CPUSPEED/65536...CPUSPEED in Hz)
d/	**Pincontrol** waarde	(PIO(PINCTRL .....))
e/	**Execture control** waarde	(PIO(EXECCTRL.....))
f/	**Shiftcontrol** waarde	(PIO(SHIFCTRL.....))
g/	**Start** adres	(0....31, de regel waarop de statusmachine begint met uitvoeren, kan een etiket zijn)

EEN STATE MACHINE STARTEN EN STOPPEN

Zodra de PIO is geconfigureerd, kunt u de statusmachine starten en stoppen met behulp van:

PIO START a,b
PIO STOP a,b

a/	Het **PIO** nummer	(0 of 1)
b/	De **state machine**	(0...3)

Merk op dat wanneer een toestandsmachine wordt gestopt, deze precies daar stopt waar hij is. Om de statusmachine opnieuw op te starten, is het raadzaam om dit te doen **PIO INIT MACHINE** eerst.

PIO ASSEMBLE V6.00.02b8

PIO assembler list commands

.end program	.label	.line	.program
.side set	.wrap	.wrap target	In
IRQ	IRQ CLEAR	IRQ NOWAIT	IRQ SET
IRQ WAIT	Jmp	List	Mov
Nop	Out	PIO	Pull
Push	Set	Static	Wait

VOORBEELD PROGRAMMA 1 (Frequentie = 2029Hz uitgangs signaal)

Een volledige **PIO**-implementatie die **GPIO**-pinnen omschakelt, kan in **MMBasic** worden geïmplementeerd, zoals hieronder weergegeven.
Sluit een zoemer aan op **GP0** en hoor de audiotoon die door de **PIO** wordt gegenereerd.

```
'Ontkoppel ARM van GP0
Setpin gp0,pio1             'Gebruik GP0 als uitvoerpin voor PIO 1

'Pio programma gebruikt
'0 E081     'SET pin output
'1 E001     'SET pin hoog
'2 E000     'SET pin laag
'3 0001     'JMP 1

'Programmeer pio 1 met behulp van een array om het programma in het PIO
'geheugen te schrijven en start.
Dim a%(7)=(&h0001E000E001E081,0,0,0,0,0,0,0)
PIO program 1,a%()

'Configureer pio 1 state machine 0
p=Pio(pinctrl 0,1,,,,gp0,) 'Define SET gebruikt 1 pin, en dat is GP0.
f=2029                      '2029Hz is CPUSPEED met laagste frequentie 133000
PIO init machine 1,0,f,p    'Gebruik standaard voor execctrl, shiftctrl, start
                            'adres(=0)

'Start de PIO 1 state machine 0
PIO start 1,0
```

Houd er rekening mee dat het **MMBasic**-programma eindigt, maar dat het geluid van de zoemer doorgaat. **PIO** is onafhankelijk van de **ARM CPU** en gaat door totdat deze wordt gestopt.
Als u de **MMBasic editor opent, stopt de PIO**.

FIFO's

MMBasic en de PIO wisselen informatie uit via FIFO's. De PIO's **PUSH** gegevens in de **RX FIFO** (**MMBasic** is de **ontvanger**), of **PULL** gegevens van de **TX FIFO** (**MMBasic** is de **zender**).
Wanneer **PIO gegevens** ophaalt van de **FIFO**, worden de gegevens van daaruit overgebracht naar de **OSR** (Output Shift Register) en het kan worden verwerkt. De PIO kan de gegevens van de **ISR** (Input shift register) naar de **FIFO pushen**. Aanvullend, de PIO heeft 2 registers X en Y die kunnen worden gebruikt voor opslag of tellen. PIO kan niet optellen of aftrekken vergelijken.
Gegevensstroom:

```
MMBasic -> FIFO -> OSR -> PIO (of pinnen)
PIO (of pinnen) -> ISR -> FIFO -> MMBasic
```

MMBasic kan gegevens naar de **TX FIFO schrijven** en gegevens van de **RX FIFO lezen** met behulp van:

```
PIO READ  a,b,c,d
PIO WRITE a,b,c,d
```

a/	**PIO** nummer	(0 of 1)
b/	**State Machine** nummer	(0...3)
c/	**Aantal** 32 bit woorden	(1...4)
d/	**Integer** getal variabelenaam	(d.w.z. variabele% of array%())

PIO CLEAR wist alle **PIO FIFO's**, net als **PIO START** en **PIO INIT MACHINE**.

Het MMBAsic-programma hoeft niet te wachten tot gegevens in de FIFO verschijnen, aangezien de **RX FIFO** kan worden toegewezen een **interrupt**. De MMBasic-interruptroutine kan de gegevens van de **FIFO** ophalen.
Hetzelfde geldt voor **TX-interrupt**, in welk geval MMBasic een interrupt krijgt wanneer gegevens nodig zijn voor de **TX FIFO**.

PIO INTERRUPT a,b,c,d

a/	**PIO** nummer	(0 of 1)
b/	**State Machine** nummer	(0...3)
c/	Naam van de **RX-interrupt handler**	(d.w.z. "myRX_Interrupt" of 0 om uit te schakelen)
d/	Naam van de **TX-interrupt handler**	(d.w.z. "myTX_Interrupt" of 0 om uit te schakelen)

VOORBEELD PROGRAMMA 2

PIO ASSEMBLE V6.00.02b8 (NES controller HEF4021, CD4021 met 8 drukknoppen)

Met de nieuwste PicoMite versie kan men **PIO ASSEMBLE** standaard invoeren. **RP2350A Edition**
Het onderstaande programma legt veel van de hierboven gepresenteerde MMbasic functies en opdrachten uit. Het programma leest een NES-controller (SPI) aangesloten op de Raspberry Pi Pico. De NES controller bestaat uit een **HEF4021** schuif register aangesloten op 8 drukknopschakelaars.
Programmagebruik: **wrap** en **wrap target**, **IN**, **side set** en **delay**, **PUSH**, **PIO READ**.
GP0 en **GP1** zijn in **SET** voor pinrichting en aan de zijkant ingesteld voor compacte code.

Zo was het eerst:

```
'Ontkoppel ARM van GP0/1/2
  Setpin gp0,pio1                       'Klok  OUT SPI:CLK
  Setpin gp1,pio1                       'Data  OUT SPI:TX
  Setpin gp2,pio1                       'Data  IN  SPI:RX

'PIO programma
  PIO assemble 1,".program NES"         'Een programma heeft een naam nodig.
  PIO assemble 1,".side_set 2"          'Use 2 bits voor side set, 3 voor delay
  PIO assemble 1,".line 0"              'Startcode op regel 0.
  PIO assemble 1,"SET pindirs,&b11"     'Set GP0,GP1 uitgang, side GP0,GP1 Low.
  PIO assemble 1,".wrap target"         'Wrap target = Bovenkant van de Loop
  PIO assemble 1,"IN null,32 side 2"    'Stel ISR op 0, GP1 hoog (load),GP0 Low
  PIO assemble 1,"SET X,7 side 0"       'Set X teller op 7, GP0,GP1 laag.
  PIO assemble 1,"loop:"                'Binnenste loop label.
  PIO assemble 1,"IN pins, 1 side 0"    'Schuif 1 databit naar binnen, houd GP0,GP1 laag
  PIO assemble 1,"JMP X-- loop side 1"'JMP naar loop, dec. X, GP0 hoog(klok)
  PIO assemble 1,"PUSH side 0 [7]"      'Nu X=0, PUSH resultaat in FIFO, delay7
  PIO assemble 1,".wrap"                'Einde buiten Loop, herhaal.
  PIO assemble 1,".end program list"    'Einde programma, list resultaat.

'Configureer pio1
  p=PIO(pinctrl 2,2,,gp2,gp0,gp0,)      'GP0,GP1 OUT (SET en SIDE SET), GP2 IN.
  f=1e5                                 '100 kHz.
  s=PIO(shiftctrl 0,0,0,0,0,0)          'Shift in van LSB voor IN (en OUT).
  e=PIO(execctrl gp0,PIO(.wrap target),PIO(.wrap)) 'Wrap en Wrap Target.

'Schrijf de configuratie.
  PIO init machine 1,0,f,p,e,s,0

'Start de pio1 code.
  PIO start 1,0
```

```
'Controleer de gelezen gegevens in MMBasic en druk af.
   Dim d%
   Do
     PIO Read 1,0,1,d%
     Print Bin$(d%)
     Pause 200
   Loop
   End
```

Kopieer bovenstaande uit de handleiding naar **MMEdit** en stuur het naar de **PicoMite**.
Pas in **EDIT** het nodige aan. Plaats **PIO assemble 1** op de eerste regel en verwijder het op de overige regels en ook alle **"** op elke regel. Verander de regel **.side_set 2** naar **.side set 2** en regel **loop:** naar **.label nes:** en de regel **JMP X-- loop side 1** naar **JMP X-- nes side 1**
Het **assembler** programma kan nu worden opgeslagen met **F1** of **F2**.
Het programma leest een **NES** controller (**SPI**) aangesloten op de **Raspberry Pi Pico.**
De **NES** controller bestaat uit een **HEF4021 schuif register** aangesloten op **8 drukknopschakelaars**.
Nu kan men het zo invoeren:

```
> edit

'Ontkoppel ARM van GP0/1/2
Setpin gp0,pio1           'Klok  OUT SPI:CLK Klok op scoop.
Setpin gp1,pio1           'Data  OUT SPI:TX  Puls op scoop.
Setpin gp2,pio1           'Data  IN  SPI:RX

PIO assemble 1            'Bij PicoMite PIO assemble 0,1,WebMite PIO assemble 0
  .program NES            'Een programma heeft een naam nodig
  .side set 2             'Gebruikt 2 bits voor side set, 3 voor delay
  .line 0                 'Startcode op regel 0.
    SET pindirs, &b11     'Set GP0,GP1 uitgang, side GP0,GP1 laag
  .wrap target            'Wrap target = Bovenkant van de nes loop
    IN null, 32 side 2    'Stel ISR op 0, GP1 hoog (load),GP0 laag
    SET X, 7 side 0       'Set X teller op 7, GP0,GP1 laag
.label nes:               'Binnenste nes label
    IN pins, 1 side 0     'Schuif 1 databit naar binnen, houd GP0,GP1 laag
    JMP X-- nes side 1    'JMP naar nes, dec. X, GP0 hoog(klok)
    PUSH side 0 [7]       'Nu X=0, PUSH resultaat in FIFO, delay 7
  .wrap                   'Einde buiten nes, herhaal.
.end program list         'Einde programma, List=programma, Run=start programma

> run
0: E083
1: 5060
2: E027
3: 4001
4: 084
5: 8700
```

DMA van en naar de FIFO's

De manier waarop **DMA** werkt is als volgt:
Bij het **lezen** van de **FIFO** wacht de **DMA**-controller tot de gegevens zich in de FIFO bevinden en wanneer deze worden overgedragen die gegevens in het processorgeheugen. Elke keer dat het leest, wordt de pointer in het processorgeheugen verhoogd, zodat het kan bijvoorbeeld stapsgewijs een array vullen naarmate elk gegevensitem beschikbaar wordt gemaakt.
Bij het **schrijven** naar de **FIFO** schrijft de **DMA**-controller automatisch gegevens uit het processorgeheugen naar de **FIFO wachten** wanneer de **FIFO vol** is. Zo kunnen gegevens in een array worden voorbereid en zal de **DMA**-controller streamen die gegevens naar de **PIO FIFO** zo snel als het PIO-programma dit vereist.

DMA kan een 32-bits woord, een 16-bits korte of een 8-bits byte overbrengen en bij het instellen van DMA moet u dit vertellen de omvang van de overboeking en het aantal overboekingen dat moet worden gedaan. Omdat elke overdracht het geheugen vergroot pointer van 1,2 of 4 bytes MMBasic moet omgaan met de gegevens die in het geheugen zijn verpakt in plaats van de 64-bits die worden gebruikt voor MMbasic gehele getallen en floats. Gelukkig implementeert MMBasic twee commando's **MEMORY PACK** en **MEMORY UNPACK** om dit zeer efficiënt te doen, maar het kan evengoed worden gedaan met behulp van standaard BASIC-rekenkunde.
De DMA kan worden geconfigureerd om herhaaldelijk gegevens in of uit een geheugengedeelte (een ringbuffer) te lussen
De commando's zijn:

```
PIO DMA RX a, b, c, d, e, f, g
PIO DMA TX a, b, c, d, e, f, g
```

Waar:

- **a =** **Pio** (0 of 1)
- **b =** **State** machine (0...3)
- **c =** **Nbr** (aantal over te dragen woorden)
- **d =** **Data%()** (interger-arraynaam)
- **e =** Voltooiingsonderbreking (waar naartoe als u klaar bent, optioneel)
- **f =** Transfergrootte (**8 =16 =32**, optioneel)
- **g =** Loopbackcount (gebruikte **data%()** als ringbuffer, optioneel, loopbackcount = **2^n**)

De **DMA** start de status machine automatisch en er is geen **PIO START** commando nodig. Maar, voordat u met de overdracht begint, moet u ervoor zorgen dat er een nieuwe **PIO INIT MACHINE** is uitgevoerd, zodat de statusmachine start op de verplicht startadres.
Wanneer een ringbuffer wordt gebruikt, vereist deze een speciale voorbereiding:

```
PIO MAKE RINGBUFFER a, b
```

Waar:

- **a =** Naam van geheeltallige buffer
- **b =** Grootte van de array in bytes

Voorbeeld :

```
DIM packed%
PIO MAKE RING BUFFER packed%, 4096
```

packed% zal dan een **array** met gehele getallen zijn met **4096/8 = 512** gehele getallen.
Dit kan vervolgens door de DMA worden gebruikt voor een loopbackcounter met **DMA** van **1024 32-bits** woorden, **2048 16-bits** shorts of **4096 8-bits** bytes.

VOORBEELD PROGRAMMA 3 (6 kanaals Logic Analyser)

Dit programma brengt alles samen en gebruikt **DMA** om **128 samples** uit de **PIO RX FIFO** te lezen. Voor de demonstratie, **GP0** tot **GP5** zijn uitgangen van **3 PWM**S, en worden tegelijkertijd door de **PIO** bemonsterd als een **6 kanaallogica-analysator of oscilloscoop**. De 128 samples worden als golf-vormen naar de seriële poort gestuurd.
Dit Logic Analyzer-programma demonstreert ook **PIO DMA RX, MEMORY UNPACK,** het gebruik van buffers. Het gebruikt **PWM**'s om een testsignaal te genereren op **gp0..gp5** voor demodoeleinden. Deze zelfde pinnen worden door de logica gelezen analysator en uitvoer naar de console.
Om deze Logic Analyzer te gebruiken, moet u normaal gesproken de eerste 14 regels becommentariëren.

```
'Genereer een 50Hz 3 fase testsignaal om de DMA op 6 GPIO pinnen te demon-
'streren.
SetPin gp0, pwm       'CH 0a
SetPin gp1, pwm       'CH 0b
SetPin gp2, pwm       'CH 1a
SetPin gp3, pwm       'CH 1b
SetPin gp4, pwm       'CH 2a
SetPin gp5, pwm       'CH 2b

Fpwm = 50: PW = 100 / 3
PWM 0, Fpwm, PW, PW - 100, 1, 1
PWM 1, Fpwm, PW, PW - 100, 1, 1
PWM 2, Fpwm, PW, PW - 100, 1, 1
PWM sync 0, 100 / 3, 200 / 3

'------------------------------- LA-code PIO -------------------------------
'PIO code voor het bemonsteren van GP0..GP6 als elementaire logica-analysator
PIO clear 1

'In dit programma leest de PIO GP0..GP5 brute force
'en duwt gegevens naar FIFO. De kloksnelheid bepaalt de
'bemonsteringssnelheid. Er zijn 2 instructies per cyclus
'het nemen van 10000 / 2 / 50 = 100 monsters per cyclus van 50 Hz.

PIO assemble 1,".program push"
PIO assemble 1,".line 0"
PIO assemble 1,".wrap target"

PIO assemble 1,"IN pins,6"          'Krijg 6 bits van GP0..GP5
PIO assemble 1,"PUSH block"         'Pushgegevens wanneer FIFO ruimte heeft

PIO assemble 1,".wrap"
PIO assemble 1,".end program list"

'Configuratie
f = 1e4                             'PIO draait op 10 kHz
p = Pio(pinctrl 0,0,0,gp0,,,)       'IN basis = GP0
e = Pio(execctrl gp0,PIO(.wrap target),PIO(.wrap)) 'Wrap 1 naar 0, gp0 is
                                                   'standaard
s = Pio(shiftctrl 0,0,0,0,0,0) 'Shift in via LSB, out wordt niet gebruikt
'Schrijf de configuratie, snelheid 10 kHz (gegevens in FIFO 10us na rand GP0)
PIO init machine 1,0,f,p,e,s,0 'Startadres = 0
```

```
'----------------------- LA-code MMBasic ---------------------------
'geheugenbuffers definiëren
Dim a$(1) = ("_","-")         'Tekens voor de afdruk.
length%   = 64                 'Grootte van de ingepakte array.
Dim data%(2 * length% - 1)    'Array om de 32-bits samples in te plaatsen.
'FIFO formaat.
Dim packed%(length%-1)        'DMA array om 32-bits samples in 64 bits integer
'Laat de DMA-machine draaien en herhaal wanneer je maar wilt.
Do
 PIO DMA RX 1, 0, 2 * length%, packed%(), ReadyInt
 Print "Druk op een willekeurige toets om de bemonstering opnieuw te starten"
 Do:Loop While Inkey$ = ""
Loop
End
'-----------------------------SUBS MMBasic -------------------------
Sub ReadyInt
  'Stop de PIO en start opnieuw voor de volgende run
  PIO stop 1,0
  PIO init machine 1,0,f,p,e,s,0 'Startadres = 0
```

Haal de gegevens uit de ingepakte DMA-buffer en pak deze uit naar het originele 32-bits formaat

```
 Memory unpack packed%(), data%(), 2 * length%, 32
'Seriële uitvoer alsof de logica-analysator traceert
For j = 0 To 5
  mask% = 2^j
For i = 0 To 2 * length% - 1
  If i < 106 Then Print a$(((data%(i) And mask%) = mask%));
Next i
Print: Print
Next j
End Sub
```

Voorbeeld 4 (diverse uitgangen met frequentie = 200MHz)

```
' Toggle multiple GPIO with set and out of phase with Side Set.
SetPin GP0, PIO1
SetPin GP1, PIO1
SetPin GP2, PIO1
SetPin GP3, PIO1
SetPin GP4, PIO1
SetPin GP5, PIO1
SetPin GP6, PIO1
SetPin GP7, PIO1
SetPin GP8, PIO1
SetPin GP9, PIO1
PIO ASSEMBLE 1
 .Program multi_toggle
 .Side Set 5                        ' Pinsdir, &b11111 ' 5 side set pins
.Line 0                             ' Start line 0
  Set PinDirs, &b1111               ' Set GP0 to GP9 to output. GP0-GP3=OUT
 .Wrap Target
  Set Pins, &b00000 Side &b11111    ' Set low, side set high
  Set Pins, &b11111 Side &b00000    ' Set high, side set low
 .Wrap                              ' Wrap back to wrap_target
.End Program List
```

```
' Configure pio1 StateMachine 0.     ' Option CPUSPEED 400000 KHz. (RP2040)
f0 = 200e6                           ' 200MHz, change this to meet your need.
'                a b c d e   f   g   ' a,e=Side Set c,g=OUT b,f=Set. Zie einde
p0 = Pio(PINCTRL 5,5, , ,GP5,GP0, )
e0 = Pio(EXECCTRL GP0, Pio(.Wrap Target), Pio(.Wrap))
' Write the configuration
PIO Init Machine 1,0,f0,p0,e0,,0     ' SM0 start at address 0
PIO START 1, 0
Timer = 0
Dim Integer count = 0
Print "Frequency =" f0 / 2000000 "MHz"
Do
  If Timer > 999 Then Timer = 0 : Inc count
  Select Case count
    Case 30
      Poke word &H4001c004, &H00
      Inc count
      Print "2mA, normaal"
    Case 40
      Poke word &H4001c004, &H01
      Inc count
      Print "2mA, Hoge snelheid"
    Case 50
      Poke word &H4001c004, &H10
      Inc count
      Print "4mA, normaal"
    Case 60
      Poke word &H4001c004, &H11
      Inc count
      Print "4mA, Hoge snelheid"
    Case 70
      Poke word &H4001c004, &H20
      Inc count
      Print "8mA, normaal"
    Case 80
      Poke word &H4001c004, &H21
      Inc count
      Print "8mA, Hoge snelheid"
    Case 90
      Poke word &H4001c004, &H30
      Inc count
      Print "12mA, normaal"
    Case 100
      Poke word &H4001c004, &H31
      Inc count
      Print "12mA, Hoge snelheid"
    Case 110
      Poke word &H4001c004, &H21
      Inc count
      Print "Standaard, 8mA, Hoge snelheid"
  End Select
Loop
```

VOORBEELD PROGRAMMA 5 (Alleen voor RP2350)

Dit programma draait alleen op de **RP2350** en demonstreert het gebruik van de **FIFO**'s als individuele registers. De **PIO** van de **RP2350** heeft specifieke instructies om deze `MOV RXFIFO[y], ISR` en `MOV OSR, RXFIFO[y]` te ondersteunen.
MMBasic kan de **4 individuele FIFO registers lezen/schrijven** met behulp van:

```
PIO WRITE FIFO a, b, c, d
PIO READ FIFO( a, b, c)
```

Waar:

- **a/** **Pio** (0 of 1)
- **b/** **StateMachine** (0...3)
- **c/** **nr** (FIFO-register 0...3)
- **d/** **data%** (32 bit gehele waarde)

Het programma configureert de FIFO voor individuele leesbewerkingen en schrijft vervolgens enkele waarden in dit register. Het begint de **PIO** die **2** van de **4** individuele **FIFO** registers bijwerkt. Vervolgens leest MMBasic de waarden om slechts 2 registers weer te geven hebben wijzigingen ondergaan en er zijn geen gegevens verschoven (zoals zou gebeuren in **RP2040** **FIFO**).

```
' Alleen voor RP2350 assembler. Test de rxfifo. Dit werkt niet op RP2040.
pio assemble 1
pio assemble 1,".program test"
pio assemble 1,".line 0"
pio assemble 1,"set y,4"           ' Slechts een waarde 4 in Y.
pio assemble 1,"mov isr,y"         ' Kopieer Y naar isr
pio assemble 1,"jmp y--,next"      ' y=y-1 altijd naar volgende.
pio assemble 1,"next:"             ' Label
pio assemble 1,"mov rxfifo[y],isr" ' Moet 4 programmeren in fifo [3]
pio assemble 1,"mov isr,y"         ' Kopieer Y naar isr
pio assemble 1,"mov rxfifo[2],isr" ' Moet 3 programmeren in fifo [2]
pio assemble 1,"jmp 0"             ' Herhalen.
pio assemble 1,".end program"
f  = 1e6                           ' 1MHz
' PIO(EXECCTRL a,b,c)
e  = pio(execctrl gp0,0,31)        ' Standaardwaarde, feitelijk niet gewijzigd.
' PIO(SHIFTCTRL a,b,c,d,e,f,g,h,i,j)
sr = Pio(shiftctrl 0,0,0,0,0,0,0,0,0,1) ' Lees individuele RX, RX=4 diep.
sw = Pio(shiftctrl 0,0,0,0,0,0,0,0,1,0) ' Schrijf individuele RX, RX=4 diep.
' PIO(PINCTRL a,b,c,d,e,f,g)
p  = Pio(pinctrl 0,0,0,gp0,gp0,gp0,gp0)
' Standaardwaarde, feitelijk niet gewijzigd.
' Test het gebruik van FIFO als individuele registers.
' Vul de fifo met vooraf bepaalde waarden.
pio init machine 1,0,f,p,e,sw,0    ' Init machine voor schrijven naar RX fifo.
For i = 0 To &h3                   ' Schrijf de 4 RXFIFO-registers.
  Pio writefifo 1,0,i,&h100*(i+1)  ' Schrijf waarden &h100, &h200, &h300, &h400
Next

' Controleer of de waarden correct zijn geschreven
Print "3 RXFIFO registers voordat het programma wordt uitgevoerd."
Pio init machine 1,0,f,p,e,sr,0    ' Init machine voor het lezen van de RX fifo
For i = 0 To 3                     ' Lees de 4 RXFIFO registers
  Print i,Hex$(Pio(readfifo 1,0,i))' Controleer of ze correct zijn geschreven.
Next
```

```
Print
' Voer het PIO programma uit. Dat zou (continu) naar reg 2 en 3 in de FIFO
' moeten schrijven, maar verander register 0 en 1 niet pio begin 1,0 laat de
' bijgewerkte FIFO registers zien.
Print "3 RXFIFO registers na het uitvoeren van het programma."
For i = 0 To 3 ' Lees de 4 FIFO registers om te zien of het programma werkt.
  Print i, Hex$(Pio(readfifo 1,0,i))
Next
```

De verwachte output is:

```
3 RXFIFO registreert voordat het programma wordt uitgevoerd.
0 100
1 200
2 300
3 400

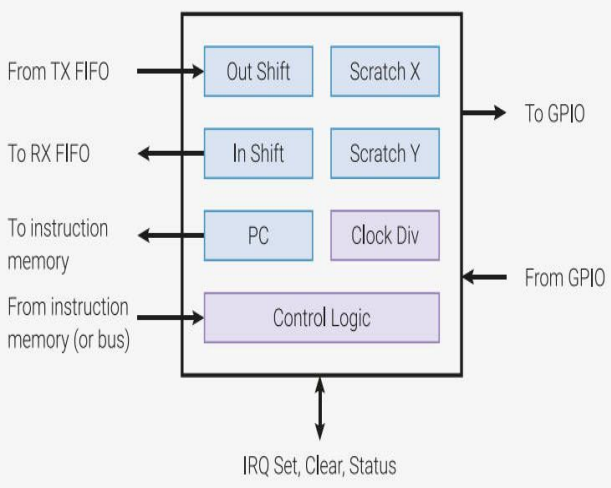
3 RXFIFO registreert na het uitvoeren van het programma.
0 100
1 200
2 300
3 400
```

PIO assembler met MMBasic Ver 6.00.02(beta)

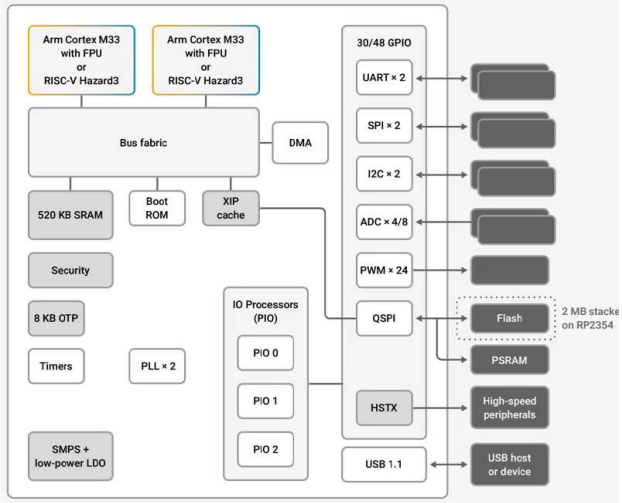
Elke State Machine heeft:

2x 32bit Schuif registers, in beide richtingen, elke schuif telling.
2x 32bit Scratch registers.
4x 32bit bus verzendt FIFO.
4x 32bit bus verzendt FIFO.
4x 32bit bus verzendt FIFO plus.
4x 32bit bus ontvangt FIFO (kan opnieuw worden geconfigureerd als 8×32 in één richting)
Fractionele (16 gehele getallen, 8 fractionele bits) klok deler.
GPIO toewijzing.
DMA interface, maximaal één woord per klok vanaf systeem DMA.
IRQ vlag ingesteld/wissen/status.

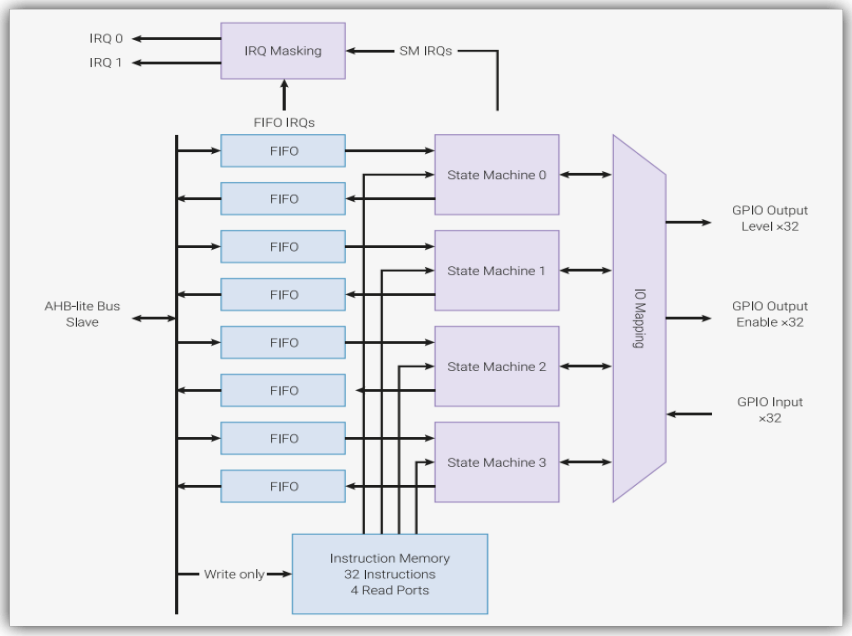
Om ze te manipuleren zijn er **negen** instructies: **JMP**, **WAIT**, **IN**, **OUT**, **PUSH**, **PULL**, **MOV**, **IRQ** en **SET** met bibliotheken beschikbaar voor algemene functies zoals seriële businterfaces.



State Machine 0 - 3



Zo is de opbouw van de RP2350



2x PIO RP2040 en 3x PIO RP2350

PIO assembler list commands en wat uitleg en voorbeelden

.end program	.label	.line	.program
.side set	.wrap	.wrap target	In
IRQ	IRQ CLEAR	IRQ NOWAIT	IRQ SET
IRQ WAIT	Jmp	List	Mov
Nop	Out	PIO	Pull
Push	Set	Static	Wait

Dit zijn de 9 instructies voor de PIO

Jmp	Naar een absoluut adres in het programmageheugen springen als een voorwaarde waar is (of direct).
Wait	Wacht tot de werking van de statusmachine wordt geblokkeerd totdat een voorwaarde waar is.
In	Shift een aantal bits van een bron naar de **ISR**.
Out	Verschuift een aantal bits uit de **OSR** naar een bestemming. v.b. `OUT PC,n`
Push	Duw de inhoud van de **ISR** naar de **RX FIFO** als een enkel 32-bits woord.
Pull	Laad een 32-bits woord van de **TX FIFO** naar de **OSR**.
Mov	Kopieer de data van een bron naar een bestemming. `MOV PINDIRS,src/prev/next`
Irq	Een interruptvlag instellen of wissen.
Set	Schrijf onmiddellijk gegevens naar een bestemming.

Nop	Is geen PIO instructie, maar het vertaalt zich naar **MOV y, y** het doet niets.
Irq Clear	Wis de **IRQ**.
Irq NoWait	Stel de **IRQ** in zonder te wachten.
Irq Set	Stel de **IRQ** in zonder te wachten.
Irq Wait	Stel de **IRQ** in en wacht tot deze is gewist voordat u verdergaat.

Aan elk van de 4 state machines in de PIO zijn 4 registers gekoppeld:

CLKDIV	Is de klok deler, die een 16-bits gehele deler en een 8-bits fractionele deler heeft. Dit bepaalt hoe snel de state machine draait. Het splitst zich af van de hoofdsysteemklok.
EXECCTRL	Bevat informatie die de vertaling en uitvoering van het programmageheugen regelt.
SHIFTCTRL	Regelt de indeling en het gebruik van de schuifregisters.
PINCTRL	Bepaalt welke en hoe de GPIO pinnen worden gebruikt.
PINCTRL	**Parameter** van `PIO INIT MACHINE` configureert die pinnen nu als uitgangen. Dit overwint een eerdere beperking dat **OUT** pinnen aaneengesloten moesten zijn met **SIDE** pinnen om ze te kunnen gebruiken.

PIO ASSEMBLE

`PIO assemble nummer`	**RP2040 0** of **1** en **RP2350 0** of **1** of **2**
`.program naam`	Programma moet een naam hebben.
`.side set nummer`	**.side set** in **RP2040** documentatie genaamd **.side_set**
`.line nummer`	Begin loop. **.line** in **RP2040** documentatie dit wordt **.origin** genoemd.
`.wrap target`	Begin van loop. **.wrap target** in **RP2040** documentatie **.wrap_target.**
`.label naam:`	Binnenste label met de naam:
`.wrap`	Einde van loop. `.line` **nbr** of `.wrap target`.
`.end program list`	Einde programma. **.end program list** speciaal geval van **.end program.**
`.line next`	Is in essentie een **.line** met **"next"** als speciaal geval.
`.word`	Documentatie ondersteunt ook **.word** Zie ook **.define** en **.lang_opt**

Wat voorbeeld code nieuwe PIO assembler

Nes controller met HEF4021 schuifregister aangesloten op 8 drukknopschakelaars.

```
PIO assemble 1          'Bij PicoMite PIO assemble 0,1, WebMite PIO assemble 0
 .program NES           'Een programma heeft een naam nodig. NES controller.
 .side set 2            'Gebruikt 2 bits voor side set, 3 voor delay
 .line 0                'Startcode op regel 0.
  SET pindirs, &b11     'Maak GP0,GP1 uitgangen, side GP0,GP1 laag
 .wrap target           'Wrap target = Bovenkant van de nes loop
  IN null,32 side 2     'Stel ISR op 0, GP1 hoog (load),GP0 laag
  SET X,7 side 0        'Zet de X teller op 7, GP0,GP1 laag.
.label nes:             'Binnenste nes label
  IN pins, 1 side 0     'Schuif 1 databit naar binnen, houd GP0,GP1 laag.
  JMP X-- nes side 1    'JMP naar nes, dec. X, GP0 hoog(klok).
  PUSH side 0 [7]       'Nu X=0, PUSH resultaat in FIFO, delay 7
 .wrap                  'Einde buiten nes, herhaal.
.end program list       'Einde programma, List = Print resultaat programma.

PIO assemble 1
 .program test          'Een programma moet een naam hebben.
 .line 0                'Start het programma op regel 0.
  SET PINDIRS, 1        'Maak de GPIO GP0 als uitgang of &b1.
.label test:            'Definieer een label met de naam "test"
  SET PINS, 1           'ZET de GPIO GP0 pin hoog.
  SET PINS, 0           'ZET de GPIO GP0 pin laag.
  JMP test              'Spring naar label met de naam test.
.end program list       'Einde programma, List = toon resultaat.

PIO assemble 1
 .program push          'Een programma moet een naam hebben "push".
 .line 0                'Start het programma op regel 0.
 .wrap target           'Wrap target = Bovenkant van de push loop
  IN pins, 6            'Krijgt 6 bits van GP0..GP5.
  PUSH block            'Push gegevens wanneer FIFO ruimte heeft.
 .wrap                  'Einde buitenste loop, herhaal.
.end program list       'Einde programma, List = toon resultaat.(print)

'Alleen voor RP2350 assembler. Dit werkt niet op RP2040
PIO assemble 1
 .program test          'Een programma moet een naam hebben.
 .line 0                'Start het programma op regel 0.
  SET y, 4              'Plaats een waarde 4 in Y register.
  MOV isr, y            'Verplaats inhoud Y register naar isr.
  JMP y--, next         'y = y - 1 spring altijd naar label met de naam next.
.label next:            'Naam Label is next.
  MOV rxfifo[y], isr    'Moet 4 programmeren in fifo [3]
  MOV isr, y            'Verplaats Y naar isr (isr=interrupt service routine)
  MOV rxfifo[2], isr    'Moet 3 programmeren in fifo [2]
  JMP 0                 'Spring naar regel 0. (line 0)
.end program list       'Einde programma, List = toon resultaat.
```

Nop werkt vanaf V6.00.02b10

```
Print "compiling first"
PIO assemble 1
 .program first
 .side set 1
 .line 0
  Pull block
 .wrap target
  Mov x, osr side 1
.label loop                  'Naam label is loop.
  Nop side 0 [4]
  Jmp x--, loop side 1 [4]   'Naam label is loop. Dus spring naar label loop.
  IRQ 0 side 0
 .wrap
.end program list
Print

Print "compiling second"
PIO assemble 1
 .program second
 .side set 1
 .line next
 .wrap target
  Wait 1 irq 0 side 0
  Wait 1 irq 0 side 1
 .wrap
.end program list

> run
compiling first
0: 80A0
1: B027
2: A442
3: 1442
4: C000

compiling second
5: 20C0
6: 30C0
>
```

11.4. PIO Instructieset

De nummers komen overeen met de RP2350-datasheet.pdf handleiding.

11.4.1. Samenvatting

PIO-instructies zijn **16 bits** lang en gebruiken de volgende codering:

PIO instructiecodering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JMP	0	0	0	Vertraging/side-set					Conditie			Adres				
WAIT	0	0	1	Vertraging/side-set					Pol	Bron		Index				
IN	0	1	0	Vertraging/side-set					Bron			Bit Telling				
OUT	0	1	1	Vertraging/side-set					Bestemming			Bit Telling				
PUSH	1	0	0	Vertraging/side-set					0	IfF	Blk	0	0	0	0	0
MOV	1	0	0	Vertraging/side-set					0	0	0	1	Idxl	0	Index	
PULL	1	0	0	Vertraging/side-set					1	IfF	Blk	1	0	0	0	0
MOV	1	0	0	Vertraging/side-set					1	0	0	1	Idxl	0	Index	
MOV	1	0	1	Vertraging/side-set					Bestemming			Op		Bron		
IRQ	1	1	0	Vertraging/side-set					0	Clr	WAIT	IdxMode		Index		
SET	1	1	1	Vertraging/side-set					Bestemming			Data				

Alle **PIO**-instructies worden in één klokcyclus uitgevoerd.
De functie van het 5-bits **Delay/side-set**-veld hangt af van de **SIDESET_COUNT**-configuratie van de statusmachine:

- Maximaal **5 LSB**'s (5 min **SIDESET_COUNT**) coderen een aantal inactieve cycli die tussen deze instructie en de volgende zijn ingevoegd.

- Maximaal **5 MSB**'s, ingesteld door **SIDESET_COUNT**, coderen een **side set** (paragraaf 11.5.1) die op sommige GPIO's, gelijktijdig met de uitvoering van de hoofdinstructie.

11.4.2. JMP

11.4.2.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JMP	0	0	0	Vertraging/side-set					Conditie			Adres				

11.4.2.2. Operatie

Stel de programmateller in op **Adres** als de **Conditie** waar is, anders geen bewerking.
Vertragingscycli op een **JMP** worden altijd van kracht, ongeacht of **Conditie** waar of onwaar is, en vinden plaats nadat **Conditie** dat is geëvalueerd en de programmateller wordt bijgewerkt.

- Voorwaarde:
  - **000: (geen voorwaarde):** Altijd
  - **001: !X : Scratch X nul** (Kladblok register **X** of **Y**)
  - **010: X-- : Scratch X niet nul**, vóór verlaging
  - **011: !Y : Scratch Y nul**
  - **100: Y-- : Scratch Y niet nul**, vóór verlaging
  - **101: X!=Y : Scratch X is niet gelijk aan scratch Y**
  - **110: PIN : Koppeling met invoer pin**
  - **111: !OSRE : Uitgangs Schuif Register niet Leeg** (**OSRE**=**O**utput**S**hift**R**egisterN**E**mpty)
- Adres: Instructieadres waar naartoe gesprongen moet worden. In de instructiecodering is dit een absoluut adres binnen de PIO instructie geheugen.

JMP PIN vertakt zich op de GPIO geselecteerd door **EXECCTRL_JMP_PIN**, een configuratieveld dat één uit het maximum selecteert van **32 GPIO-ingangen** zichtbaar voor een toestandsmachine, onafhankelijk van de andere invoertoewijzingen van de toestandsmachine. De tak is gemaakt als de GPIO hoog is.

!OSRE vergelijkt de bits die zijn verschoven sinds de laatste **PULL** met de drempelwaarde voor het aantal verschuivingen die is geconfigureerd door **SHIFTCTRL_PULL_THRESH**.
Dit is dezelfde drempelwaarde die wordt gebruikt door autopull (paragraaf 11.5.4).

JMP X-- en **JMP Y--** verlagen altijd respectievelijk kladregister X of Y. De verlaging is niet afhankelijk van de huidige waarde van het scratch-register. De vertakking is afhankelijk van de initiële waarde van het register, d.w.z. vóór de er heeft een afname plaatsgevonden: als het register aanvankelijk niet nul is, wordt de vertakking genomen.

11.4.2.3. Assembler syntaxis

JMP (<cond>) <doel>

Waar:

<cond> Een optionele voorwaarde die hierboven wordt vermeld (bijvoorbeeld **!x** voor **scratch X nul**). Als er geen voorwaardecode is opgegeven, wordt de tak wordt altijd genomen.

<doel> Een programmalabel of -waarde (zie Paragraaf 11.3.2) die de instructie-offset binnen het programma vertegenwoordigt (de eerste instructie is **offset 0**). Omdat de PIO **JMP** instructie absolute adressen in de PIO gebruikt instructiegeheugen, moeten JMP's worden aangepast op basis van de programma belastings offset tijdens run-time. Dit is wordt voor u afgehandeld bij het laden van een programma met de SDK, maar u moet voorzichtig zijn bij het coderen van **JMP** instructies voor gebruik door **OUT EXEC**.

11.4.3. WAIT

11.4.3.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAIT	0	0	1	Vertraging/side-set					Pol	Bron		Index				

11.4.3.2. Operatie

Wacht totdat aan een bepaalde voorwaarde is voldaan.
Zoals bij alle vertragingsinstructies (paragraaf 11.2.5) beginnen vertragingscycli nadat de instructie is voltooid. Dat wil zeggen, als er enige vertraging is cycli aanwezig zijn, beginnen ze pas te tellen nadat aan de wachtvoorwaarde is voldaan.

- Polariteit:
  - **1:** Wacht op een 1.
  - **0:** Wacht op een 0.
- Bron: waar u op moet wachten. Waarden zijn:
  - **00: GPIO:** Systeem GPIO-invoer geselecteerd door Index. Dit is een absolute GPIO-index en wordt niet beïnvloed door de staat invoer-IO-toewijzing van de machine.
  - **01: PIN:** Invoerpin geselecteerd door Index. De invoer-IO-toewijzing van deze statusmachine wordt eerst toegepast en vervolgens Index selecteert op welke van de toegewezen bits moet worden gewacht. Met andere woorden, de pin wordt geselecteerd door Index toe te voegen aan de **PINCTRL_IN_BASE** configuratie, modulo 32.
  - **10: IRQ:** **PIO IRQ-vlag** geselecteerd door Index
  - **11: JMPPIN:** Wacht op de pin geïndexeerd door de **PINCTRL_JMP_PIN** configuratie, plus een index in het bereik 0-3, allemaal modulo 32. Andere waarden van Index zijn voorbehouden.
- Index: welke pin of bit moet worden gecontroleerd.
  WAIT x IRQ gedraagt zich enigszins anders dan andere **WAIT**-bronnen:
- Als de polariteit **1** is, wordt de geselecteerde IRQ-vlag gewist door de statusmachine zodra aan de wachtvoorwaarde is voldaan.
- De vlagindex wordt op dezelfde manier gedecodeerd als het **IRQ**-indexveld, waarbij vanaf de twee MSB's naar beneden wordt gedecodeerd (uitgelijnd met het veld **IRQ**-instructie IdxMode):
  - **00:** De drie LSB's worden rechtstreeks gebruikt om de IRQ-vlaggen in dit PIO-blok te indexeren.
  - **01 (PREV),** Verwijst de instructie naar een **IRQ** van de **PIO** met het volgende lagere nummer in het systeem, doorlopend naar de PIO met het hoogste nummer als dit PIO0 is.
  - **10 (REL),** De statusmachine-ID (0…3) wordt toegevoegd aan de **IRQ**-index, door middel van modulo-4-optelling op de twee LSB's. State machine 2 met een vlagwaarde van 0x11 wacht bijvoorbeeld op vlag 3, en een vlagwaarde van 0x13 wacht op vlag 1. Hierdoor kunnen meerdere statusmachines waarop hetzelfde program-ma draait, met elkaar synchroniseren.
  - **11 (NEXT),** De instructie verwijst naar een **IRQ** van de **PIO** met het volgende hogere nummer in het systeem, doorlopend naar PIO0 als dit de **PIO** met het hoogste nummer is.

VOORZICHTIG

WAIT 1 IRQ x Mag niet worden gebruikt met **IRQ** vlaggen die aan de interruptcontroller worden gepresenteerd, om een raceconditie met een systeem-interrupt-handler.

```
WAIT <polariteit> gpio <gpio_num>
WAIT <polariteit> pin <pin_num>
WAIT <polariteit> irq <irq_num> (rel, next, vorige)
WAIT <polariteit> jmppin (+ <pin_offset>)
```

Waar:

<polariteit> Een waarde (zie Paragraaf 11.3.2) die de polariteit specificeert (0 of 1).

<pin_num> Een waarde (zie Paragraaf 11.3.2) die het invoerpinnummer specificeert (zoals toegewezen door de SM-invoerpin in kaart brengen).

<gpio_num> Een waarde (zie Paragraaf 11.3.2) die het feitelijke GPIO-pincode specificeert.

<irq_num> (rel) Een waarde (zie Paragraaf 11.3.2) die het **IRQ**-nummer specificeert waarop moet worden gewacht (0-7). Als rel aanwezig is, dan is de het daadwerkelijk gebruikte **IRQ**-nummer wordt berekend door de onderste twee bits van het IRQ-nummer te vervangen (**irq_num10**) met de onderste twee bits van de som (**irq_num10** + **sm_num10**) waarbij sm_num10 het statusmachinenummer is.

<pin_offset> Een waarde (zie Paragraaf 11.3.2) toegevoegd aan de **jmp_pin** om het daadwerkelijke pinnummer te verkrijgen.

11.4.4. IN

11.4.4.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IN	0	1	0	Vertraging/side-set					Bron			Bit Telling				

11.4.4.2. Operatie

Shift Bit telt bits van de bron naar het **Input Shift Register (ISR)**. De schakelrichting wordt voor elke statusmachine geconfigureerd door **SHIFTCTRL_IN_SHIFTDIR**. Verhoog bovendien het aantal ingangsverschuivingen met het aantal bits, tot een verzadiging van 32.

- Bron:
  - **000: PIN** codes
  - **001: X** (scratch register **X**) (Kladblok register **X**)
  - **010: Y** (scratch register **Y**) (Kladblok register **Y**)
  - **011: NULL** (allemaal nullen) (Alles nul)
  - **100: Gereserveerd**
  - **101: Gereserveerd**
  - **110: ISR** (**I**nput **S**hift **R**egister)
  - **111: OSR** (**O**utput **S**hift **R**egister)
- Aantal bits: hoeveel bits er naar de **ISR** moeten worden verschoven. **1…32 bits**, 32 is gecodeerd als **00000**.

Als automatische push is ingeschakeld, zal **IN** de **ISR**-inhoud ook naar de **RX FIFO** pushen als de push-drempel wordt bereikt (**SHIFTCTRL_PUSH_THRESH**). **IN** wordt nog steeds in één cyclus uitgevoerd, ongeacht of er een automatische push plaatsvindt of niet. De staatsmachine zal afslaan als de **RX FIFO** vol is wanneer er automatisch wordt geduwd. Een automatische push wist de **ISR**-inhoud naar allemaal nullen, en wist de telling van de invoerverschuiving. Zie paragraaf 11.5.4.
IN gebruikt altijd de minst significante bittellingsbits van de brongegevens. Als **PINCTRL_IN_BASE** bijvoorbeeld is ingesteld op 5, wordt de instructie **IN PINS, 3** neemt de waarden van **pin 5, 6** en **7,** en verschuift deze naar de **ISR**. Eerst wordt de **ISR** naar links verschoven of rechts om ruimte te maken voor de nieuwe invoergegevens, dan worden de invoergegevens gekopieerd naar het gat dat hierdoor ontstaat. De bitvolgorde van de invoergegevens zijn niet afhankelijk van de verschuivingsrichting.
NULL kan worden gebruikt voor het verschuiven van de inhoud van de **ISR**. UART's ontvangen bijvoorbeeld eerst de **LSB** en moeten dus naar **rechts verschuiven**.
Na **8 IN PINS, 1** instructies zullen de ingevoerde seriële gegevens **bits 31…24** van de **ISR** bezetten. Een **IN NULL, 24** instructie zal verschuiven in **24 nulbits**, waarbij de invoergegevens worden uitgelijnd op **ISR-bits 7…0**. Als alternatief zou de processor of **DMA** een byte-lezing kunnen uitvoeren van **FIFO**-adres **+ 3**, waarvoor **bits 31…24** van de **FIFO**-inhoud nodig zijn.

11.4.4.3. Assembler syntaxis

```
IN <bron>, <bit_count>
```

Waar:

<bron> Een van de hierboven gespecificeerde bronnen.

<bit_count> Een waarde (zie Paragraaf 11.3.2) die het aantal bits specificeert dat moet worden verschoven (geldig bereik 1-32).

11.4.5. OUT

11.4.5.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT	0	1	1	Vertraging/side-set					Bestemming			Bit Telling				

11.4.5.2. Operatie

Shift Bit telt bits uit het **Output Shift Register** (**OSR**) en schrijft die bits naar Destination. Verhoog bovendien de uitgangsverschuivingtelling per bittelling, verzadigend op 32.

- Bestemming:
  - **000: PIN** codes
  - **001: X** (scratch register **X**) (Kladblok register **X**)
  - **010: Y** (scratch register **Y**) (Kladblok register **Y**)
  - **011: NULL** (gegevens negeren)
  - **100: PINDIRS** (PINDIRS is OUT)
  - **101: PC** (**P**rogram **C**ounter)
  - **110: ISR** (zet ook de **ISR** shiftteller op Bittelling) (**I**nput **S**hift **R**egister)
  - **111: EXEC** (Voer **OSR** shiftgegevens uit als instructie) (**O**utput **S**hift **R**egister)
- Aantal bits: hoeveel bits er uit de **OSR** moeten worden verschoven. **1…32 bits**, **32** is gecodeerd als **00000**.

Er wordt een waarde van 32 bits naar Destination geschreven: de bits met het laagste aantal bits zijn afkomstig van de **OSR** en de rest bestaat uit nullen. Deze waarde is het minst significante aantal bits van de OSR als **SHIFTCTRL_OUT_SHIFTDIR** aan de rechterkant is, anders is dit het meest aan-zienlijke stukjes.
PINS en **PINDIRS** gebruiken de **OUT** pintoewijzing, zoals beschreven in Paragraaf 11.5.6.
Als automatisch ophalen is ingeschakeld, wordt de **OSR** automatisch bijgevuld vanaf de **TX FIFO** als de trekdrempel, **SHIFTCTRL_PULL_THRESH**, wordt bereikt. De telling van de uitgangsverschuiving wordt tegelijkertijd teruggezet naar **0**. In dit geval zal de **OUT** blokkeren als de **TX FIFO** leeg is, maar wordt verder nog steeds in één cyclus uitgevoerd. De details worden gegeven in paragraaf 11.5.4.
Met **OUT EXEC** kunnen instructies inline in de **FIFO**-datastroom worden opgenomen. De **OUT** zelf wordt in één cyclus uitgevoerd, en de instructie van de OSR wordt uitgevoerd in de volgende cyclus. Er zijn geen beperkingen op de soorten instructies die dat kunnen door dit mechanisme worden uitgevoerd. Vertragingscycli op de initiële OUT worden genegeerd, maar de uitvoerende kan vertragingscycli invoegen als normaal.
OUT PC gedraagt zich als een onvoorwaardelijke sprong naar een adres dat uit de **OSR** is verschoven.

11.4.5.3. Assembler syntaxis

```
OUT <bestemming>, <bit_count>
```

Waar:

<bestemming> Eén van de hierboven gespecificeerde bestemmingen.
<bit_count> Een waarde (zie Paragraaf 11.3.2) die het aantal bits specificeert dat moet worden verschoven (geldig bereik 1-32).

11.4.6. PUSH

11.4.6.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUSH	1	0	0	Vertraging/side-set					0	IfF	Blk	0	0	0	0	0

11.4.6.2. Operatie

Duw de inhoud van de ISR naar de RX FIFO, als een enkel 32-bits woord. Wis ISR tot geheel nullen.

- **IfFull:** Indien 1, niets doen tenzij het totale aantal invoerverschuivingen de drempel heeft bereikt, **SHIFTCTRL_PUSH_THRESH** (dezelfde wat betreft autopush; zie paragraaf 11.5.4).
- **Block:** Indien 1: uitvoering blokkeren als **RX FIFO** vol is.

PUSH IFFULL helpt programma's compacter te maken, zoals autopush. Het is handig in gevallen waarin de **IN** vastloopt op een ongepast tijdstip als autopush is ingeschakeld, b.v. als de statemachine hierbij een extern stuursignaal afgeeft punt.
De **PIO**-assembler stelt standaard de **Block**-bit in. Als het blokbit niet is ingesteld, blijft de **PUSH** niet hangen op een volledige **RX FIFO** ga onmiddellijk door naar de volgende instructie. De **FIFO**-status en -inhoud blijven ongewijzigd wanneer dit gebeurt. De **ISR** is nog steeds gewist naar allemaal nullen, en de **FDEBUG_RXSTALL**-vlag is ingesteld (hetzelfde als een blokkerende **PUSH** of autopush naar een volledige **RX FIFO**) om aan te geven dat gegevens verloren zijn gegaan.

OPMERKING

De werking van de **PUSH**-instructie is niet gedefinieerd wanneer **SM0_SHIFTCTRL.FJOIN_RX_PUT** of **FJOIN_RX_GET** is set — zie Paragraaf 11.4.8 en Paragraaf 11.4.9 voor details over de **PUT**- en **GET**-instructie die in deze status kunnen worden gebruikt.

11.4.6.3. Assembler syntaxis

```
PUSH (IfFull)
PUSH (IfFull) Blok
PUSH (IfFull) NoBlock
```

Waar:

IfFull Equivalent aan **IfFull == 1** hierboven. d.w.z. de standaardwaarde als dit niet is opgegeven, is **IfFull == 0**.

Block Equivalent aan **Block == 1** hierboven. Dit is de standaardwaarde als noch **Block** noch **NoBlock** is opgegeven.

NoBlock Equivalent aan **Block == 0** hierboven.

11.4.7. PULL

11.4.7.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PULL	1	0	0	Vertraging/side-set					1	IfF	Blk	1	0	0	0	0

11.4.7.2. Operatie

Laad een **32-bits woord** van de **TX FIFO** in de **OSR**.

- **IfEmpty:** Indien 1, niets doen tenzij het totale aantal uitvoerverschuivingen de drempel heeft bereikt, **SHIFTCTRL_PULL_THRESH** (de hetzelfde als voor autopull; zie paragraaf 11.5.4).
- **Block:** Indien 1: blokkeren als **TX FIFO** leeg is. Indien 0, wordt bij het ophalen uit een lege **FIFO scratch X** naar **OSR** gekopieerd.

Sommige randapparatuur (**UART, SPI**, enz.) zou moeten stoppen als er geen gegevens beschikbaar zijn, en deze moeten oppikken zodra deze binnenkomen; anderen (**I2S**) moet continu klokken, en het is beter om tijdelijke of herhaalde gegevens uit te voeren dan om te stoppen met klokken. Dit kan zijn bereikt met de parameter **Block**.
Een niet-blokkerende **PULL** op een lege **FIFO** heeft hetzelfde effect als **MOV OSR, X**. Het programma kan scratch vooraf laden registreer **X** met een geschikte standaard, of voer een **MOV X, OSR** uit na elke **PULL NOBLOCK**, zodat het laatste geldige **FIFO**-woord wordt gerecycled totdat er nieuwe gegevens beschikbaar zijn.
PULL IFEMPTY is handig als een **OUT** met autopull op een ongepaste locatie blijft hangen als de **TX FIFO** leeg is. Indien leeg maakt enkele van dezelfde programmavereenvoudigingen mogelijk als autopull: bijvoorbeeld de eliminatie van een buitenste lusteller.
De blokkering treedt echter op een gecontroleerd punt in het programma op.

OPMERKING

Wanneer autopull is ingeschakeld, is elke **PULL**-instructie no-op als de **OSR** vol is, zodat de **PULL**-instructie zich gedraagt als een barrière. **OUT NULL, 32** kan worden gebruikt om de **OSR**-inhoud expliciet te verwijderen. Zie paragraaf 11.5.4.2 voor meer details.

11.4.7.3. Assembler syntaxis

```
PULL (indien leeg)
PULL (indien leeg) Blok
PULL (indien leeg) NoBlock
```

Waar:

IfEmpty Equivalent aan **IfEmpty == 1** hierboven. d.w.z. de standaardwaarde als dit niet is opgegeven, is **IfEmpty == 0**.

Block Equivalent aan **Block == 1** hierboven. Dit is de standaardwaarde als noch **Block** noch N**oBlock** is opgegeven.

NoBlock Equivalent aan **Block == 0** hierboven.

11.4.8. MOV (naar RX)

11.4.8.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOV	1	0	0	Vertraging/side-set					0	0	0	1	IdxI	0	Index	

11.4.8.2. Operatie

Schrijf de **ISR** naar een geselecteerd **RX FIFO**-item. De toestandsmachine kan de **RX FIFO**-gegevens in elke volgorde schrijven, ook geïndexeerd door het **Y**-register, of een onmiddellijke index in de instructie. Vereist dat het configuratieveld **SHIFTCTRL_FJOIN_RX_PUT** aanwezig is ingesteld, anders is de werking ervan ongedefinieerd. De **FIFO**-configuratie kan voor het programma worden opgegeven via de **.fifo**-richtlijn (zie pioasm_fifo).

Als **IdxI** (index per onmiddellijk) is ingesteld, worden de registers van de **RX FIFO** geïndexeerd door de twee minst significante bits van de Index operand. Anders worden ze geïndexeerd door de twee minst significante bits van het **Y**-register. Wanneer **IdxI** helder is, zijn alle waarden niet nul waarden van Index zijn gereserveerde coderingen en hun werking is niet gedefinieerd.

Wanneer alleen **SHIFTCTRL_FJOIN_RX_PUT** is ingesteld (in **SM0_SHIFTCTRL** tot en met **SM3_SHIFTCTRL**), kan het systeem ook de **RX** lezen **FIFO** registreert met willekeurige toegang via **RXF0_PUTGET0** tot en met **RXF0_PUTGET3** (waarbij **RXFx** aangeeft welke status de **FIFO** van de machine wordt benaderd). In deze toestand wordt de **FIFO**-registeropslag opnieuw gebruikt als statusregisters, waardoor de state machine kan op elk moment worden bijgewerkt en het systeem kan op elk moment lezen. Bijvoorbeeld een kwadratuurdecoderprogramma zou de huidige stappentelling te allen tijde in een statusregister kunnen bijhouden, in plaats van naar de **RX FIFO** te pushen en mogelijk blokkeren.

Wanneer zowel **SHIFTCTRL_FJOIN_RX_PUT** als **SHIFTCTRL_FJOIN_RX_GET** zijn ingesteld, heeft het systeem geen toegang meer tot de **RX FIFO** opslagregisters, maar de toestandsmachine kan de registers nu in willekeurige volgorde plaatsen/ophalen, waardoor ze kunnen worden gebruikt als extra krasopslag.

OPMERKING

De **RX FIFO**-opslagregisters hebben slechts één enkele leespoort en schrijfpoort, en toegang via elke poort wordt toegewezen naar slechts één van (systeem, toestandsmachine) tegelijk.

11.4.8.3. Assembler syntaxis

```
MOV rxfifo[y], isr
MOV rxfifo[<index>], isr
```

waar:

y Het letterlijke token **"y"**, dat de **RX FIFO**-invoer aangeeft, wordt geïndexeerd door het **Y** register.

<index> Een waarde (zie Paragraaf 11.3.2) die de **RX FIFO**-invoer specificeert die moet worden geschreven (geldig bereik 0-3).

11.4.9. MOV (van RX)

11.4.9.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOV	1	0	0	Vertraging/side-set					1	0	0	1	IdxI	0	Index	

11.4.9.2. Operatie

Lees de geselecteerde **RX FIFO-invoer** in de **OSR**. De **PIO**-statusmachine kan de **FIFO**-data in elke volgorde lezen, geïndexeerd hetzij door het **Y**-register, hetzij door een onmiddellijke index in de instructie. Vereist het configuratieveld **SHIFTCTRL_FJOIN_RX_GET** moet worden ingesteld, anders is de werking ervan ongedefinieerd.

Als **IdxI** (index per onmiddellijk) is ingesteld, worden de registers van de **RX FIFO** geïndexeerd door de twee minst significante bits van de Index operand. Anders worden ze geïndexeerd door de twee minst significante bits van het **Y**-register. Wanneer **IdxI** helder is, zijn alle waarden niet nul waarden van Index zijn gereserveerde coderingen en hun werking is niet gedefinieerd.

Wanneer alleen **SHIFTCTRL_FJOIN_RX_GET** is ingesteld, kan het systeem ook de **RX FIFO**-registers schrijven met willekeurige toegang via **RXF0_PUTGET0** tot en met **RXF0_PUTGET3** (waarbij **RXFx** aangeeft van welke statusmachine de **FIFO** wordt benaderd). Hierin status wordt de **RX FIFO**-registeropslag opnieuw gebruikt als extra configuratieregisters, die het systeem kan bijwerken op elk moment en de toestandsmachine kan op elk moment lezen. Een **UART TX**-programma kan deze registers bijvoorbeeld gebruiken configureer het aantal databits, of de aanwezigheid van een extra stopbit.

Wanneer zowel **SHIFTCTRL_FJOIN_RX_PUT** als **SHIFTCTRL_FJOIN_RX_GET** zijn ingesteld, heeft het systeem geen toegang meer tot de **RX FIFO** opslagregisters, maar de toestandsmachine kan de registers nu in willekeurige volgorde plaatsen/ophalen, waardoor ze kunnen worden gebruikt als extra krasopslag.

OPMERKING

De **RX FIFO**-opslagregisters hebben slechts één enkele leespoort en schrijfpoort, en toegang via elke poort wordt toegewezen naar slechts één van (systeem, toestandsmachine) tegelijk.

11.4.9.3. Assembler syntaxis

```
MOV osr, rxfifo[y]
MOV osr, rxfifo[<index>]
```

Waar:

y Het letterlijke token **"y"**, dat de **RX FIFO** invoer aangeeft, wordt geïndexeerd door het **Y**-register.

<index> Een waarde (zie Paragraaf 11.3.2) die de te lezen **RX FIFO** invoer specificeert (geldig bereik **0-3**).

11.4.10. MOV

11.4.10.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOV	1	0	1	Vertraging/side-set					Bestemming			Op		Bron		

Kopieer gegevens van bron naar bestemming.

- Bestemming:
  - **000:** **PINS** (gebruikt dezelfde pintoewijzing als **OUT**) (PINDIRS is **OUT**)
  - **001:** **X** (scratch register **X**) (Kladblok Register **X**)
  - **010:** **Y** (scratch register **Y**) (Kladblok Register **Y**)
  - **011:** **PINDIRS** (gebruikt dezelfde pintoewijzing als **OUT**) (PINDIRS is **OUT**)
  - **100:** **EXEC** (gegevens uitvoeren als instructie)
  - **101:** **PC** (**P**rogam **C**ounter)
  - **110:** **ISR** (Ingangsverschuivingsteller wordt door deze handeling op **0** gereset, d.w.z. **leeg**)
  - **111:** **OSR** (Uitgangsverschuivingsteller wordt door deze handeling op **0** teruggezet, is **vol**)
- Bediening:
  - **00:** **Geen**
  - **01:** **Omkeren** (bitsgewijs complement)
  - **10:** **Bit omgekeerd**
  - **11:** **Gereserveerd**
- Bron:
  - **000:** **PINS** (gebruikt dezelfde pintoewijzing als **IN**)
  - **001:** **X**
  - **010:** **Y**
  - **011:** **NULL**
  - **100:** **Gereserveerd**
  - **101:** **STATUS**
  - **110:** **ISR** (**I**nput **S**hift **R**egister)
  - **111:** **OSR** (**O**utput **S**hift **R**egister)

MOV **PC** veroorzaakt een onvoorwaardelijke sprong. **MOV** **EXEC** gedraagt zich hetzelfde als **OUT** **EXEC** (paragraaf 11.4.5) en maakt register mogelijk inhoud die als instructie moet worden uitgevoerd. De **MOV** zelf wordt in de ene cyclus uitgevoerd, en de instructie in Source in de volgende cyclus. Vertragingscycli op **MOV** **EXEC** worden genegeerd, maar de uitvoerende kan zoals normaal vertragingscycli invoegen.

De **STATUS** bron heeft een waarde van **allemaal enen** of **allemaal nullen**, afhankelijk van de status van een bepaalde statusmachine, zoals **FIFO** vol/leeg, geconfigureerd door **EXECCTRL_STATUS_SEL**.

MOV kan de overgedragen gegevens op beperkte manieren manipuleren, gespecificeerd door het argument Operatie. Omkeren stelt elke bit in Bestemming naar de logische **NOT** van de corresponderende bit in Source, d.w.z. 1 bits worden 0 bits, en omgekeerd. Beetje omgekeerde sets elke bit n in Destination tot bit 31 - n in Source, ervan uitgaande dat de bits genummerd zijn van 0 tot 31.

MOV **dst, PINS** leest pinnen met behulp van de **IN** pintoewijzing, gemaskeerd tot het aantal bits gespecificeerd door **SHIFTCTRL_IN_COUNT**. De LSB van de gelezen waarde is de pin aangegeven door **PINCTRL_IN_BASE**, en elke opeenvolgende bit is afkomstig van een pin met een hoger nummer, terugloop na 31. Resultaatbits die groter zijn dan de breedte die is opgegeven door de configuratie **SHIFTCTRL_IN_COUNT** zijn **0**.

MOV **PINDIRS, src** wordt niet ondersteund op **PIO** versie **0**.

11.4.10.3. Assembler syntaxis

```
MOV <bestemming>, (op) <bron>
```

Waar:

<bestemming>	Eén van de hierboven gespecificeerde bestemmingen.	
op	Indien aanwezig, is:	
	! of **~**	Voor **NOT** (Let op: dit is altijd een bitsgewijze **NOT**)
	::	Voor bit omkering.
<bron>	Een van de hierboven gespecificeerde bronnen.	

11.4.11. IRQ

11.4.11.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ	1	1	0	Vertraging/side-set					0	Clr	WAIT	IdxMode		Index		

11.4.11.2. Operatie

Stel de IRQ vlag in of wis deze die is geselecteerd door het Index-argument.

- **Wissen:** Indien 1, wis dan de door Index geselecteerde vlag, in plaats van deze te hijsen. Als **Clear** is ingesteld, heeft de **Wait**-bit geen effect.
- **Wachten:** Indien 1, stop totdat de gehesen vlag weer wordt gestreken, b.v. als een systeeminterrupthandler de vlag heeft bevestigd.
- **Index:** Specificeert een **IRQ-index** van **0-7**. Deze **IRQ**-vlag wordt ingesteld/gewist, afhankelijk van de **Clear**-bit.
- **IdxMode:** Wijzig het gedrag als het **Index**-veld, door de index aan te passen of door **IRQ**-vlaggen van een andere index te indexeren PIO-blok:
  - **00:** De drie **LSB**'s worden rechtstreeks gebruikt om de **IRQ**-vlaggen in dit PIO-blok te indexeren.
  - **01 (PREV):** De instructie verwijst naar een **IRQ**-vlag van de PIO met het volgende lagere nummer in het systeem, doorlopend naar het hoogst genummerde **PIO** als dit **PIO0** is.
  - **10 (REL):** De statusmachine-ID (0…3) wordt toegevoegd aan de IRQ-vlagindex, door middel van modulo-4-toevoeging op de twee LSB's. Statusmachine 2 met een vlagwaarde van '0x11' zal bijvoorbeeld wachten op vlag 3, en een vlag-waarde van '0x13' zal wacht op vlag 1. Hierdoor kunnen meerdere state machines met hetzelfde programma met elkaar synchroniseren.
  - **11 (NEXT):** De instructie verwijst naar een IRQ-vlag van de PIO met het volgende hogere nummer in het systeem, doorlopend naar PIO0 als dit de PIO met het hoogste nummer is.

Alle **IRQ**-vlaggen **0-7** kunnen worden doorgestuurd naar interrupts op systeemniveau, op een van de twee externe interruptverzoeklijnen van de PIO, geconfigureerd door **IRQ0_INTE** en **IRQ1_INTE**.
De modulo-toevoegmodus **(REL)** maakt relatieve adressering van **'IRQ'**- en**' WAIT'**-instructies mogelijk, voor het synchroniseren van de state machines waarop hetzelfde programma draait. Bit 2 (de derde LSB) wordt niet beïnvloed door deze toevoeging.
De **NEXT/PREV**-modi kunnen worden gebruikt om te synchroniseren tussen statusmachines in verschillende **PIO** blokken als deze staat.
De klokken van machines zijn verdeeld, hun klokdelers moeten hetzelfde zijn en moeten gesynchroniseerd zijn door te schrijven **CTRL.NEXTPREV_CLKDIV_RESTART** naast de relevante **NEXT_PIO_MASK/PREV_PIO_MASK** bits. Merk op dat de de **cross-PIO-verbinding** wordt verbroken tussen PIO's met verschillende toegankelijkheid tot niet-beveiligde code, volgens **ACCESSCTRL**.
Als Wacht is ingesteld, beginnen de vertragingscycli pas nadat de wachtperiode is verstreken.

11.4.11.3. Assembler syntaxis

```
IRQ <irq_num>         (vorige, rel, volgende)
IRQ Set <irq_num>     (vorige, rel, volgende)
IRQ NoWait <irq_num>  (vorige, rel, volgende)
IRQ Wait  <irq_num>   (vorige, rel, volgende)
IRQ Clear <irq_num>   (vorige, rel, volgende)
```

Waar:

<irq_num> (rel)	Een waarde (zie Paragraaf 11.3.2) die het IRQ-nummer specificeert waarop moet worden gewacht (**0-7**). Als rel aanwezig is, dan is de het daadwerkelijk gebruikte IRQ-nummer wordt berekend door de onderste twee bits van het IRQ-nummer te vervangen (**irq_num10**) met de onderste twee bits van de som (**irq_num10 + sm_num10**) waarbij **sm_num10** het state machine nummer is.
IRQ	Stel de IRQ in zonder te wachten.
IRQ Set	Stel de IRQ in zonder te wachten.
IRQ NoWait	Stel de IRQ in zonder te wachten.
IRQ Wait	Stel de IRQ in en wacht tot deze is gewist voordat u verdergaat.
IRQ Clear	Wis de IRQ.

11.4.12. SET

11.4.12.1. Codering

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET	1	1	1	Vertraging/side-set					Bestemming			Data				

11.4.12.2. Operatie

Schrijf onmiddellijke waardegegevens naar bestemming.

- Bestemming:
  - **000: PIN-codes**
  - **001: X** (scratch register **X**) 5 LSB's zijn ingesteld op Data, alle andere zijn gewist op 0.
  - **010: Y** (scratch register **Y**) 5 LSB's zijn ingesteld op Data, alle andere zijn gewist op 0.
  - **011: Gereserveerd**
  - **100: PINDIRS**
  - **101: Gereserveerd**
  - **110: Gereserveerd**
  - **111: Gereserveerd**
- Data: 5-bit onmiddellijke waarde om naar pinnen te sturen of te registreren.

Dit kan worden gebruikt om besturingssignalen zoals een klok- of chipselectie af te dwingen, of om lustellers te initialiseren. Omdat gegevens 5 bits bevatten grootte, kunnen scratch-registers worden ingesteld op waarden van **0-31**, wat voldoende is voor een lus van **32** iteraties.
De mapping van **SET** en **OUT** op pinnen wordt onafhankelijk geconfigureerd. Ze kunnen worden toegewezen aan verschillende locaties, bijvoorbeeld als de ene pin moet worden gebruikt als klok-signaal en de andere voor data. Het kunnen ook overlappende reeksen pinnen zijn:
UART-zender kan **SET** gebruiken om start- en stopbits te bevestigen, en **OUT**-instructies om **FIFO**-gegevens naar dezelfde pinnen te verschuiven.

11.4.12.3. Assembler syntaxis

```
SET <bestemming>, <waarde> in
```

Waar:

<bestemming> Is een van de hierboven gespecificeerde bestemmingen.
<waarde> De waarde (zie paragraaf 11.3.2) die moet worden ingesteld (geldig bereik **0-31**)

11.5. Functionele details

11.5.1. Site Set

Side-set is een functie waarmee staatsmachines het niveau of de richting van maximaal 5 pinnen kunnen wijzigen, gelijktijdig met de hoofduitvoering van de instructie.
Een voorbeeld waarbij dit nodig is, is een snelle **SPI**-interface: hier moet een klokovergang (tussen **1 → 0** of **0 → 1**) plaatsvinden gelijktijdig met een datatransitie, waarbij een nieuw databit wordt verschoven van de **OSR** naar een **GPIO**. In dit geval een **OUT** met een side-set zou deze beide tegelijk bereiken.
Dit maakt de timing van de interface nauwkeuriger en verkleint de totale programmagrootte (zoals een afzonderlijke **SET** instructie is het is niet nodig om de klokpin te wisselen) en verhoogt ook de maximale frequentie waarop de **SPI** kan werken.
Side-set maakt **GPIO**-mapping ook veel flexibeler, omdat de mapping onafhankelijk is van **SET**. De voorbeeld-**I2C**-code maakt het mogelijk dat **SDA** en **SCL** worden toegewezen aan twee willekeurige pinnen, als klokuitrekken is uitgeschakeld. Normaal gesproken schakelt **SCL** over naar synchroniseer de gegevensoverdracht en **SDA** bevat de gegevensbits die worden uitgeschoven. Er zijn echter enkele specifieke **I2C**-sequenties zoals start- en stoplijnomstandigheden, hebben een vast patroon nodig om zowel op **SDA** als op **SCL** te besturen. De mapping die **I2C** gebruikt dit te bereiken is:

- `Site Set` → **SCL**
- `Out` → **SDA**
- `Set` → **SDA**

Hierdoor kan de toestandsmachine de twee gebruiksscenario's van gegevens op **SDA** en klok op **SCL** bedienen, of vaste overgangen op beide **SDA** en **SCL**, terwijl **SDA** en **SCL** nog steeds kunnen worden toegewezen aan twee willekeurige GPIO's naar keuze.
De **side set**-gegevens worden gecodeerd in het Delay/**side set**-veld van elke instructie. Elke instructie kan worden gecombineerd met **side set**, inclusief instructies die naar de pinnen schrijven, zoals **OUT PINS** of **SET PINS**. De pin-mapping van Side-set is onafhankelijk van **OUT**- en **SET**-toewijzingen, hoewel deze elkaar kunnen overlappen. Als **side set** en een **OUT** of **SET** tegelijkertijd naar dezelfde pin schrijven, wordt er gebruik gemaakt van **side set**-gegevens.

Bijlage G
Sprites
ALLEEN VGA EN HDMI VERSIES

Je kunt op verschillende manieren een sprite maken, maar feitelijk sla je alleen een afbeelding op in een buffer. Het verschil komt wanneer je de sprite **TOONT**. In dit geval slaat de firmware de eerste keer het geheugengebied (of display real-estate) dat zal worden vervangen door de sprite en vervolgens de sprite op zijn plaats trekt.
Daaropvolgende **SHOW**-opdrachten vervangen de sprite door de opgeslagen achtergrond en slaan de achtergrond op voor de nieuwe locatie en teken ten slotte de sprite. Op deze manier kun je de sprite zonder extra's over de achtergrond verplaatsen code.
Botsingsdetectie zit hier bovenop en zoekt naar de rechthoekige grenzen van sprites die elkaar raken om te creëren een interrupt of een sprite die de rand van het frame raakt.
Sprites zijn zo geordend dat de tekenvolgorde in een lifo wordt bewaard. Stel dat je **sprite 1** hebt overlapt door **sprite 2** en vervolgens door **sprite 3**. Als je sprite 1 eenvoudigweg verplaatste, zou de achtergrond de bits van 2 en 3 overschrijven, niet wat wij willen. **SPRITE SHOW SAFE** windt de **LIFO** af door elke sprite in omgekeerde volgorde te verwijderen en verplaatst sprite 1 en herstelt vervolgens eerst 2 en vervolgens 3 erbovenop. Tenslotte is er het concept van lagen (dit is de 4e parameter in **SPRITE SHOW**).
Het concept van de sprite-implementatie is als volgt:

- Sprites zijn full colour en van elke grootte. De botsingsgrens is de omsluitende rechthoek.
- Sprites worden geladen tot een specifiek nummer (**1** tot **64**).
- Sprites worden weergegeven met de opdracht **SPRITE SHOW**.
- Voor elke opdracht **SHOW** moet de gebruiker een **"laag"** selecteren. Dit kan tussen **0** en **10** zijn
- **Sprites botsen** met sprites op **dezelfde laag**, **laag 0** of de **rand van het scherm**.
- **Laag 0** is een speciaal geval en sprites op alle andere lagen botsen ermee.
- Het **SCROLL** commando laten sprites op alle lagen behalve **laag 0** onbeweeglijk.
- Sprites van laag 0 scrollen met de achtergrond en dit kan botsingen veroorzaken.
- Er is geen praktische limiet aan het aantal botsingen dat wordt veroorzaakt door de opdrachten **SHOW** of **SCROLL**.
- Met de **SPRITE()** functie kan de gebruiker de details van een botsing volledig opvragen.
- Een **SHOW** commando overschrijft de details van eerdere botsingen voor die sprite.
- Een **SCROLL** commando overschrijft de details van eerdere botsingen voor **ALLE** sprites.
- Om een scherm te herstellen naar een eerdere staat, moeten sprites worden verwijderd in de omgekeerde volgorde waarin ze zijn geschreven (d.w.z. **last in first out**).

Omdat het verplaatsen van een sprite of, met name, het scrollen van de achtergrond, meerdere sprite-botsingen kan veroorzaken, is het belangrijk om te begrijpen hoe ze kunnen worden ondervraagd.
De beste manier om met een sprite-botsing om te gaan, is door de interrupt-faciliteit te gebruiken. Een collision interrupt-routine wordt ingesteld met het **SPRITE INTERRUPT** commando.

Bijvoorbeeld:
SPRITE INTERRUPT botsing.
Hieronder volgt een voorbeeldprogramma voor het identificeren van alle botsingen die het gevolg zijn van een **SPRITE SHOW** commando of een **SCROLL** commando.

```
'
' This routine demonstrates a complete interrogation of collisions
'
SUB Botsing
  LOCAL INTEGER i
  ' Gebruik eerst de functie SPRITE(S) om te zien wat de interrupt veroorzaakte
  IF SPRITE(S) <> 0 THEN ' Botsing van specifieke individuele sprite.
    ' SPRITE(S) retourneert de sprite die bewoog om de botsing te veroorzaken.
    PRINT "Botsing op sprite", SPRITE(S)
    proces_botsing(SPRITE(S))
    PRINT
  ELSE
    ' 0 betekent botsing van een of meer sprites veroorzaakt door bewegingen op
    ' de achtergrond.
    ' SPRITE(C, 0) vertelt ons hoeveel sprites een botsing hebben gehad.
    PRINT "Scroll zorgde ervoor dat in totaal ", SPRITE(C, 0),
    PRINT " sprites botsingen hadden."
    FOR i = 1 TO SPRITE(C, 0)
      ' SPRITE(C, 0, i) vertelt ons het spritenummer van de “I”-de sprite
      PRINT "Sprite", SPRITE(C, 0, i)
      proces_botsing(SPRITE(C, 0, i))
    NEXT i
    PRINT
  ENDIF
END SUB

' Krijg details van de specifieke botsingen voor een bepaalde sprite.
SUB process_botsing(S AS INTEGER)
  LOCAL INTEGER i, j
  ' SPRITE(C, #n) retourneert het aantal huidige botsingen voor sprite n.
  PRINT "Totaal van " SPRITE(C, S) " botsingen"
  FOR i = 1 TO SPRITE(C, S)
  ' SPRITE(C, S, i) vertelt ons het spritenummer van de “I”de sprite.
    j = SPRITE(C, S, i)
    IF j = &HF1 THEN
      PRINT "Botsing met linkerkant van scherm"
    ELSE IF j = &HF2 THEN
      PRINT "Botsing met bovenkant scherm"
    ELSE IF j = &HF4 THEN
      PRINT "Botsing met rechterkant van scherm"
    ELSE IF j = &HF8 THEN
      PRINT "Botsing met onderkant scherm"
    ELSE
      ' SPRITE(C, #n, #m) retourneert details van de maand botsing.
      PRINT "Botsing met sprite ", SPRITE(C, S, i)
    ENDIF
  NEXT i
END SUB
```

Bijlage H
Special Keyboard Keys

MMBasic genereert één uniek teken voor de functietoetsen en andere speciale toetsen op het toetsenbord.
Deze worden in deze tabel weergegeven als hexadecimale en decimale getallen:

Toetsenbord Toets	Toets Code (Hexadecimaal)	Toets Code (Decimaal)		Toetsenbord Toets	Toets Code (Hexadecimaal)	Toets Code (Decimaal)
Delete	**7F**	**127**		**F5**	**95**	**149**
Pijl Omhoog	**80**	**128**		**F6**	**96**	**150**
Pijl Omlaag	**81**	**129**		**F7**	**97**	**151**
Pijl naar Links	**82**	**130**		**F8**	**98**	**152**
Pijl naar Rechts	**83**	**131**		**F9**	**99**	**153**
Insert	**84**	**132**		**F10**	**9A**	**154**
Home	**86**	**134**		**F11**	**9B**	**155**
End	**87**	**135**		**F12**	**9C**	**156**
Pagina Omhoog	**88**	**136**		**PrtScr/SysRq**	**9D**	**157**
Pagina Omlaag	**89**	**137**		**PAUSE/BREAK**	**9E**	**158**
Alt	**8B**	**139**		**SHIFT_TAB**	**9F**	**159**
F1	**91**	**145**		**SHIFT_DEL**	**A0**	**160**
F2	**92**	**146**		**SHIFT_DOWN_ARROW**	**A1**	**161**
F3	**93**	**147**		**SHIFT_RIGHT_ARROW**	**A3**	**163**
F4	**94**	**148**				

Als bij **PS2**- en **USB**-toetsenborden de **Shift**-toets tegelijkertijd wordt ingedrukt met de functie-toetsen **F1** tot en met **F12**, wordt **40 (hex)** aan de code toegevoegd (dit is het equivalent van instel-lingsbit 6). **Shift-F10** genereert bijvoorbeeld **DA (hex)**.

De **shift-modifier** (werkt met de functietoetsen **F1** tot **F12**; deze wordt genegeerd voor de andere toetsen behalve **TAB, DEL, DOWN_ARROW** en **RIGHT_ARROW**, zoals hierboven aangegeven.
Behalve **SHIFT-F3** werkt de sift-modifier niet met de seriële console.
MMBasic vertaalt de meeste VT100-escape-codes gegenereerd door terminalemulators zoals Tera Term en Putty naar deze codes (exclusief de shift- en control-modifiers). Dit betekent dat een terminalemulator die werkt via een USB- of een seriële poort die als console is geopend, dezelfde sleutelcodes genereert als een direct aangesloten toetsenbord.

Bijlage I
Programmeren in BASIC. Een zelfstudie.

De BASIC-taal werd in 1964 geïntroduceerd door Dartmouth College in de VS als een computer taal voor het aanleren van programmeren en dienovereenkomstig is het gemakkelijk te gebruiken en te leren. Tegelijkertijd heeft het bewezen een competente en krachtige programmeertaal te zijn en als gevolg daarvan werd het erg populair eind jaren 70 en begin jaren 80. Zelfs vandaag de dag worden er nog steeds enkele grote commerciële datasystemen geschreven in de BASIC-taal (voornamelijk Pick Basic).

Voor de PicoMite is het grootste voordeel van BASIC het gebruiksgemak. Wat meer moderne talen zoals C en C++ kunnen echt geestverruimend zijn, maar met BASIC kun je beginnen met een éénregelig programma en er iets zinnigs uit halen. MMBasic is ook krachtig omdat je verfijnd kunt tekenen grafische afbeeldingen, de externe I/O-pinnen manipuleren om andere apparaten te bedien-en en met anderen te communiceren apparaten die gebruikmaken van een reeks inge-bouwde communicatie protocollen.

Commando prompt

Interactie met MMBasic gebeurt via de console bij de opdrachtprompt (d.w.z. het groter dan-symbool **(>)** op de console). Bij het opstarten geeft MMBasic de opdrachtprompt weer en wacht tot er een opdracht wordt ingevoerd. Het keert ook terug naar de opdrachtprompt als uw pro-gramma eindigt of als het een foutmelding genereert.
Wanneer de opdrachtprompt wordt weergegeven, hebt u een breed scala aan opdrachten die u kunt invoeren en uitvoeren. Meestal geven ze een lijst met het programma dat in het geheugen is opgeslagen (**LIST**) of bewerken ze het (**EDIT**) of stellen ze misschien enkele opties in (de opdracht **OPTION**). Meestal is de opdracht gewoon **RUN**, wat MMBasic de opdracht geeft om het program-ma in het programmageheugen uit te voeren.
Bijna elke opdracht kan worden ingevoerd bij de opdrachtprompt en dit wordt vaak gebruikt om een opdracht te testen om te zien hoe deze werkt. Een eenvoudig voorbeeld is de opdracht **PRINT** (hier later meer over), die u kunt testen door het volgende in te voeren bij de opdrachtprompt:
`PRINT 2 + 2` en het is niet verrassend dat MMBasic het getal **4** zal afdrukken voordat het terugkeert naar de opdrachtprompt.(commando prompt)
Deze mogelijkheid om een opdracht te testen bij de opdrachtprompt is handig wanneer u leert programmeren in BASIC, dus het zou de moeite waard zijn om een Raspberry Pi Pico geladen met de PicoMite-firmware bij de hand te hebben voor de incidentele test terwijl u deze tutorial door-loopt.
De **CYAAN** gekleurde opdrachten (commando's) zijn de kleuren in **EDIT** als **Option ColourCode On** is ingesteld. De kleur **GROEN** kan men instellen in **Tera Term** wat standaard een achtergrond kleur **ZWART** en voorgrond **WIT** heeft in **Tera Term**. (**Setup, Windows, Color, Text of Background, RGB**)

Structuur van een BASIC programma

Een BASIC-programma begint bij de eerste regel en gaat door totdat het aan het einde afloopt of op een **END** eindigd.
Command op dat moment zal MMBasic de commando prompt **(>)** op de console van **Tera Term** weergeven en wachten op iets om in te voeren.

Een programma bestaat uit een aantal statements of commando's, elk die de BASIC tolk vraagt om iets te doen (de woorden statement en command betekenen over het algemeen hetzelfde en worden gebruikt) uitwisselbaar in deze handleiding).
Normaal gesproken staat elke instructie op zijn eigen regel, maar u kunt meerdere instructies op één regel hebben gescheiden door de dubbele punt (**:**).
Bijvoorbeeld:

```
A = 24.6 : PRINT A
```

Elke regel kan beginnen met een regelnummer. Regelnummers waren verplicht in de vroege BASIC-tolken moderne implementaties (zoals MMBasic) hebben ze echter niet nodig. Je kunt ze nog steeds gebruiken als je wilt, maar ze hebben geen voordeel en over het algemeen maken ze je programma's rommelig.
Dit is een voorbeeld van een programma dat regelnummers gebruikt:

```
> 50 A = 24.6
> 60 PRINT A
 24.6
```

Bij gebruik van **EDIT** kunnen de commando's en functies in kleur worden weergegeven met de opdracht:

```
OPTION COLOURCODE ON of OPTION COLOURCODE OFF
```

Een regel kan ook beginnen met een label dat kan worden gebruikt als doel voor een programma-sprong met behulp van de **GOTO** opdracht. Dit zal in meer detail worden uitgelegd wanneer we het **GOTO** commando behandelen, maar dit is een voorbeeld (de labelnaam is **JmpBack:**):
Hier bij mag na **(:)** geen nieuwe opdracht volgen anders een foutmelding.

```
JmpTerug:          ' Begin van loop
A = A + 1          ' A = 0 en tel er telkens 1 bij elke ronde
PRINT A            ' Print A
PAUSE 100          ' Wacht even 100 mS
GOTO JmpTerug      ' Spring terug naar JmpTerug
```

of

```
a = 20 : b = 30 : c = 40
Gosub LABEL1
End

LABEL1:
Print "Label1"
Return
```

Opmerkingen

Een opmerking is elke tekst die volgt op het enkele aanhalingsteken (**'**). Een reactie kan overal geplaatst worden en strekt zich uit tot het einde van de lijn. Als MMBasic een opmerking tegenkomt, springt deze gewoon naar het einde ervan (dwz het onderneemt geen actie met betrekking tot een opmerking). In **EDIT** zal bij **Option Colourcode On** **GEEL** zijn.

Opmerkingen moeten worden gebruikt om niet voor de hand liggende delen van het programma uit te leggen en in het algemeen iemand te informeren die niet bekend is met het programma, hoe het werkt en wat het probeert te doen. Onthoud dat na slechts een paar maanden zal een programma dat je hebt geschreven uit je hoofd zijn verdwenen en eruit zien vreemd als je het weer oppakt. Om deze reden zul je jezelf later dankbaar zijn als je er veel van gebruikt opmerkingen.

Hieronder volgen enkele voorbeelden van opmerkingen:

```
  '  Bereken de hypotenusa
  PRINT SQR(a * a + b * b)
```

of

Oudere BASIC-programma's gebruikten het commando **REM** om een opmerking te starten en je kunt dat ook gebruiken als je wens, maar het enkele aanhalingsteken is gemakkelijker te gebruiken

```
  INPUT var    ' Krijg de temperatuur
```

en handiger.

Het PRINT commando

Er zijn een aantal veelvoorkomende commando's die fundamenteel zijn en we zullen ze hierin behandelen zelfstudie, maar misschien wel de meest bruikbare is de opdracht PRINT. Zijn taak is eenvoudig; iets afdrukken op de console. Dit wordt meestal gebruikt om gegevens uit te voeren die u kunt zien (zoals het resultaat van berekeningen) of informatieve berichten geven.

PRINT is ook handig wanneer u een fout in uw programma opspoort; je kunt het gebruiken om de waardes uit te printen van variabelen en weergaveberichten in belangrijke fasen in de uitvoering van het programma.
In zijn eenvoudigste vorm zal de opdracht gewoon afdrukken wat er op de opdrachtregel staat. Dus bijvoorbeeld:

```
  PRINT 54
```

Zal op de console het nummer **54** weergeven, gevolgd door een nieuwe regel.

De gegevens die moeten worden afgedrukt, kunnen zoiets eenvoudigs zijn als dit of een uitdrukking, wat betekent dat worden berekend. We zullen uitdrukkingen later in meer detail behandelen, maar als voorbeeld het volgende:

```
> PRINT 3/21
 0.1428571429
```

Dit is het zelfde resultaat, wat sneller werkt.

```
> ? 3/21
 0.1428571429
>
```

Berekent het resultaat van drie gedeeld door eenentwintig en geeft het weer. Merk op dat de groter dan symbool **(>)** is de opdrachtprompt geproduceerd door MMBasic - u typt dat niet in. Andere voorbeelden van de **PRINT** opdracht zijn:

```
> PRINT "Prachtige Wereld"
Prachtige Wereld
> PRINT (999 + 1) / 5
 200
>
```

U kunt deze uitproberen bij de opdrachtprompt.
De opdracht **PRINT** werkt ook met meerdere waarden tegelijk, bijvoorbeeld:

```
> PRINT "Het bedrag is" 345 " en het tweede bedrag is" 456
Het bedrag is 345 en het tweede bedrag is 456
>
```

Normaal gesproken wordt elke waarde gescheiden door een spatie, zoals in het vorige voorbeeld, maar u kunt scheid waarden ook met een komma **(,)**. **De komma** zorgt ervoor dat er **een tab** tussen de twee wordt ingevoegd waarden. In MMBasic-tabbladen in het **PRINT commando staan acht tekens uit elkaar**. Ter illustratie van het **tabblad** volgende opdracht drukt een lijst met nummers af met tabbladen:

```
> PRINT 12, 34, 9.4, 1000
 12          34          9.4         1000
>
```

Merk op dat er voor elk nummer een spatie is afgedrukt. Deze ruimte is een plaatshouder voor de min symbool (-) als de waarde negatief is. **Let op** het verschil met het **getal 12** in dit voorbeeld:

```
> PRINT -12, 34, -9.4, 1000
-12       34      -9.4      1000
>
```

Het printstatement kan worden afgesloten met een puntkomma **(;)**. Dit voorkomt het **PRINT** commando van het verplaatsen naar een nieuwe regel wanneer alle tekst is afgedrukt.
Bijvoorbeeld:

```
PRINT "Dit wordt ";
PRINT "geprint op een enkele regel."
```

Zal resulteren in deze uitvoer:

```
Dit wordt geprint op een enkele regel.
```

Het bericht zou er als volgt uitzien zonder de puntkomma aan het einde van de eerste regel:

Variabelen

Voordat we veel verder gaan, moeten we definiëren wat een "variabele" is, omdat ze fundamenteel zijn voor de werking van de BASIC-taal (in feite de meeste programmeertalen). Een variabele is gewoon een plaats om een gegevensitem op te slaan (dwz de "waarde"). Deze waarde kan worden gewijzigd terwijl het programma draait, wat daarom het een "variabele" wordt genoemd.
Variabelen in MMBasic kunnen een van de drie typen zijn. De meest voorkomende is drijvende komma (float). Dit wordt geprint op een enkele regel en dit is automatisch aangenomen als het type van de variabele niet is opgegeven. De andere twee typen zijn integer en string en we zullen ze later behandelen. Een getal met drijvende komma is een gewoon getal dat kan bevatten een decimale punt. **3.45** of **-0.023** of **100.00** zijn bijvoorbeeld allemaal getallen met drijvende komma. (**In Nederland gebruiken we een komma tussen de getallen maar in Basic een punt!**).

Een **variabele** kan worden **gebruikt om een getal op te slaan** en het kan dan op **dezelfde manier worden gebruikt als het getal zelf**, in welk geval het de waarde vertegenwoordigt van het **laatste nummer dat eraan is toegewezen**.

Als een eenvoudig voorbeeld:

```
A = 3
B = 4
PRINT A + B
```

Geeft het getal **7** weer. In dit geval zijn zowel **A** als **B** variabelen en gebruikt MMBasic hun huidige waarden in de **PRINT** instructie. **MMBasic** maakt automatisch een variabele aan wanneer deze voor het eerst wordt aangetroffen het is zo dat de instructie **A = 3** beide een drijvendekomma-variabele (het standaardtype) creëerden met de naam **A** en vervolgens heeft het de waarde **3** eraan toegekend.
De **naam van een variabele** moet beginnen met **een letter**, terwijl de rest van de naam letters mag gebruiken, cijfers, het onderstrepingsteken of de punt (of punt) tekens. De **naam** kan **maximaal 32** tekens lang zijn en de naamval (dwz hoofdletters of niet) is niet belangrijk.
Hier zijn enkele voorbeelden:

```
Tota_Count
ForeColour
temp3
x
ThisIsALongVariableName
increment.value
```

U kunt de waarde van een variabele overal in uw programma wijzigen met behulp van het toewijzingscommando, d.w.z:

```
variabele = expression
```

Bijvoorbeeld:

```
temp3 = 24.6
count = 5
CTemp = (FTemp - 32) * 0.5556
```

In het laatste voorbeeld zijn zowel **CTemp** als **FTemp variabelen** en deze regel **converteert** de waarde van **FTemp** (**in graden Fahrenheit**) naar **graden Celsius** en slaat het resultaat op in de variabele **CTemp**.

Uitdrukkingen

We zijn de term 'expressie' eerder tegengekomen in deze tutorial en bij het programmeren heeft het een specifieke betekenis. Het is een formule die door de BASIC-interpreter kan worden omgezet in een enkel getal of enkele waarde.

MMBasic evalueert numerieke uitdrukkingen met dezelfde regels die we op school hebben geleerd Voor **vermenigvuldigen** en **delen** worden bijvoorbeeld **eerst** uitgevoerd, gevolgd door **optellen** en **aftrekken**.
Deze worden de **voorrangsregels** genoemd en worden volledig beschreven in deze handleiding.

> Van links naar rechts: **HMVDWOA** = **H**eer **M**eneer **V**an **D**alen **W**acht **O**p **A**ntwoord
> **H**aakjes, **M**achtverheffen, **V**ermenigvuldigen of **D**elen, **W**ortels, **O**ptellen of **A**ftrekken
> **Bij twijfel gebruik haakjes voor de volgorde.**

Dit betekent dat MMBasic **2 + 3 * 6** zal oplossen door eerst **3 met 6** te **vermenigvuldigen**, wat **18** oplevert en **vervolgens 2 bij op te tellen** wat resulteert in een uiteindelijke **waarde van 20**. Op dezelfde manier zullen zowel **5 * 4** als **10 + 4 * 3 – 2** ook oplossen tot **20**.

Als u de interpreter wilt dwingen om eerst delen van de uitdrukking te evalueren, kunt u dat deel omringen van de uitdrukking tussen haakjes. Bijvoorbeeld, **(10 + 4) * (3 – 2)** zal oplossen tot **14**, niet **20** zoals zou het geval zou zijn geweest als de haakjes niet waren gebruikt. Het gebruik van haakjes vertraagt het programma, dus u moet ze royaal gebruiken als de kans bestaat dat MMBasic uw verkeerd interpreteert voornemen.

Zoals eerder aangegeven, kunt u variabelen in een uitdrukking precies hetzelfde gebruiken als rechte getallen.
Dit verhoogt bijvoorbeeld de waarde van de variabele temp met één:

```
temperatuur = temperatuur + 1
```

U kunt ook functies in uitdrukkingen gebruiken. Dit zijn speciale bewerkingen die worden aangeboden door MMBasic, voor bijvoorbeeld om goniometrische waarden te berekenen. Als voorbeeld zal het volgende de lengte van de af te drukken hypotenusa van een rechthoekige driehoek met behulp van de functie **SQR()** die de vierkantswortel van een getal retourneerd (a en b zijn variabelen die de lengtes van de andere zijden bevatten):

```
PRINT SQR(a * a + b * b)
```

MMBasic evalueert deze uitdrukking eerst door a te vermenigvuldigen met a, dan b te vermenigvuldigen met b, dan de resultaten bij elkaar optellen. Het resulterende getal wordt vervolgens doorgegeven aan de functie **SQR()** die zal bereken de vierkantswortel van dat getal (dwz de hypotenusa) en retourneer het voor het **PRINT** commando op het scherm.

Enkele andere wiskundige functies van MMBasic zijn onder meer:

- **SIN(r)** – de sinus van r
- **COS(r)** – de cosinus van r
- **TAN(r)** – de tangens van r

Er zijn nog veel meer functies voor u beschikbaar en ze zijn allemaal eerder in deze handleiding genoemd.

Merk op dat in de bovenstaande trigonometrische functies de waarde die aan de functie wordt doorgegeven (dwz 'r') de hoek is in radialen. In MMBasic kun je de functie **RAD(d)** gebruiken om een hoek om te rekenen van graden naar radialen (**'d'** is de **hoek in graden**).

Een ander kenmerk van BASIC is dat je functieaanroepen in elkaar kunt nesten. Bijvoorbeeld, gegeven de hoek in graden (dwz **'d'**) de sinus van die hoek kan worden gevonden met deze uitdrukking:

```
PRINT SIN(RAD(d))
```

In dit geval zal MMBasic eerst de waarde van d nemen en deze converteren naar radialen met behulp van de **RAD()** functie.
De uitvoer van deze functie wordt dan de invoer voor de **SIN()** functie.

De IF verklaring

Beslissingen nemen vormt de kern van de meeste computerprogramma's en in BASIC wordt dit meestal gedaan met de **IF** verklaring. Dit is bijna als een Engelse zin geschreven:

```
IF voorwaarde THEN actie
```

De voorwaarde is meestal een vergelijking zoals is gelijk aan, kleiner dan, meer dan, enz. Bijvoorbeeld:

```
IF Temp < 25 THEN PRINT "Koud"
```

Temp zou een variabele zijn die de huidige temperatuur vasthoudt (in**°C**) en `PRINT "Koud"` de actie om klaar te zijn.

Er zijn verschillende tests die u kunt uitvoeren:

```
= is gelijk aan        <>  niet gelijk aan
< kleiner   dan        <=  kleiner dan of gelijk aan
> groter    dan        >=  groter dan of gelijk aan
```

U kunt ook een **ELSE** clausule toevoegen die wordt uitgevoerd als de initiële voorwaarde onwaar is getest:

```
IF voorwaarde THEN waar-actie ELSE onwaar-actie
```

Dit zal bijvoorbeeld verschillende acties uitvoeren wanneer de temperatuur lager is dan 25 of 25 of hoger:

```
IF Temp < 25 THEN PRINT "Koud" ELSE PRINT "Warm"
```

De vorige voorbeelden gebruikten allemaal enkelregelige **IF** instructies, maar u kunt ook meerregelige **IF** verklaringen hebben. Ze zien er zo uit:

```
IF voorwaarde THEN
  TrueActies
  TrueActies
ENDIF
```

of

```
IF voorwaarde THEN
  TrueActies
  TrueActies
ELSE
  OnwaarActies
  OnwaarActies
ENDIF
```

In tegenstelling tot de **IF** instructie met één regel, kunt u veel echte acties hebben met elk op hun eigen regel en evenzo veel valse acties. Over het algemeen is de **IF** instructie met één regel handig als u een eenvoudige actie die moet worden ondernomen, terwijl de versie met meerdere regels veel gemakkelijker te begrijpen is als de acties zijn talrijk en ingewikkelder.
Een voorbeeld van een **IF** statement met meerdere regels met meer dan één actie is:

```
IF Bedrag < 25 THEN
    PRINT "Te laag"
    PRINT “Minimale waarde is 25”
ELSE
    PRINT "Invoer geaccepteerd"
    SaveToSDCard
    PRINT "Vul tweede bedrag in"
ENDIF
```

Merk op dat in het bovenstaande voorbeeld elke actie naar is ingesprongen om te laten zien tot welk deel van de **IF** structuur het behoort. Inspringen is niet verplicht, maar het maakt een programma veel gemakkelijker te begrijpen voor iemand die dat wel is ben er niet bekend mee en is daarom een echte aanrader.

In een meerregelig **IF** statement kunt u aanvullende tests uitvoeren met het **ELSE IF** commando. Dit is het beste uitgelegd aan de hand van een voorbeeld (de temperaturen zijn allemaal in ºC):

```
IF Temp < 0 THEN
  PRINT "Vorst"
ELSE IF Temp < 20 THEN
  PRINT "Koud"
ELSE IF Temp < 35 THEN
  PRINT "Warm"
ELSE
  PRINT "Heet"
ENDIF
```

De **ELSE IF** kan dezelfde tests gebruiken als een gewone **IF** (dwz **<, <=,** enz.), maar die test wordt alleen gedaan als de voorgaande test was vals. U krijgt dus bijvoorbeeld alleen de melding Warm als **Temp < 0** mislukt en **Temp < 20** is mislukt, maar **Temp < 35** was waar. De laatste **ELSE** zal de zaak opvangen waarin alle tests waren vals.

Een uitdrukking zoals **Temp < 20** wordt door MMBasic geëvalueerd als waar of onwaar, waarbij heeft een waarde van één en valse nul. Dit kun je zien als je bij de console het volgende hebt ingevuld:

```
PRINT 30 > 20
```

MMBasic zal **1** afdrukken, wat betekent dat de waarde waar is en op dezelfde manier zal het volgende de betekenis van **0** afdrukken dat de expressie is geëvalueerd als onwaar.

```
PRINT 30 < 20
```

De **IF** verklaring geeft niet echt om wat de aandoening eigenlijk is, het evalueert alleen de voorwaarde en als het resultaat nul is, wordt dat als onwaar beschouwd en als het niet nul is, wordt het als waar beschouwd. Deze biedt enkele handige snelkoppelingen. Als BalanceCorrect bijvoor-beeld een variabele is die waar is (niet- nul) wanneer een functie van het programma correct is, kan het volgende worden gebruikt om een beslissing te nemen op basis van die waarde:

```
IF BalansCorrect THEN ... doe iets ...
```

FOR lussen

Een andere veelvoorkomende vereiste bij het programmeren is het herhalen van een reeks acties. Je zou bijvoorbeeld kunnen alle zeven dagen van de week wilt doorlopen en voor elke dag dezelfde functie wilt uitvoeren. BASIS biedt de **FOR** lusconstructie voor dit type taak en het werkt als volgt:

```
FOR  dag = 1 TO 7
  Doe iets vanuit de waarde van ‘dag’
NEXT dag
```

Dit begint door de variabele dag aan te maken en hieraan de waarde **1** toe te kennen. Het program-ma zal dan de volgende instructies uitvoeren totdat het bij de **NEXT** instructie aankomt. Dit vertelt de BASIC interpreter om de waarde van **dag** te verhogen, ga terug naar de vorige **FOR** instructie en voer het opnieuw uit volgende uitspraken een tweede keer.
Dit blijft ronddraaien totdat de waarde van de dag groter is dan **7** en het programma zal dan de lus verlaten en doorgaan met de instructies die volgen op de **NEXT** uitspraak.
Als eenvoudig voorbeeld kunt u de getallen van één tot tien als volgt afdrukken:

```
FOR nummer = 1 TO 10
  PRINT nummer,;
NEXT nummer
```

De komma aan het einde van de **PRINT** instructie vertelt de interpreter om naar de volgende tab-kolom te gaan na door het nummer en de puntkomma af te drukken, blijft de cursor op deze regel staan in plaats van automatisch naar de volgende regel te gaan. Als gevolg hiervan worden de nummers in nette kolommen over de pagina afgedrukt.

Dit is wat je zou zien:

```
1       2       3       4       5       6       7       8       9       10
```

De **FOR** lus heeft ook een paar extra trucjes in petto. U kunt het bedrag dat de variabele wordt verhoogd met behulp van het sleutelwoord **STEP**. Dus, bijvoorbeeld, het volgende zal alleen de oneven nummers afdrukken:

```
FOR aantal = 1 TO 10 STEP 2
  PRINT aantal,;
NEXT aantal
```

De waarde van de stap (of incrementwaarde) is standaard één als het **STEP** sleutelwoord niet wordt gebruikt, maar u kunt het instellen op het gewenste nummer.

Wanneer MMBasic de variabele verhoogt, zal het controleren of de variabele de **TO** heeft overschreden waarde en, als dit het geval is, zal het de lus verlaten. Dus in het bovenstaande voorbeeld zal de waarde van **nbr** bereiken negen en het zal worden afgedrukt, maar in de volgende lus zal het nummer elf zijn en op dat moment zal de uitvoering verlaat de lus. Deze test wordt ook toegepast aan het begin van de lus (dwz als in het begin de waarde van de variabele de **TO** waarde over-schrijdt, zal de lus nooit worden uitgevoerd, zelfs niet één keer).

Door de **STEP** waarde op een negatief getal in te stellen, kunt u de **FOR** lus gebruiken om af te stappen van een hoge aantal te laag.

Het volgende zal bijvoorbeeld de getallen van **1** tot **10** in omgekeerde volgorde afdrukken:

```
FOR aantal = 10 TO 1 STEP -1
  PRINT aantal,;
NEXT aantal
```

Na achter **(>)** **LIST** in te typen kan men programma nogmaals opvragen.

Na achter **(>)** **NEW** in te typen wist men het programma op uitvoerscherm en in de editor.

Na achter **(>)** De **pijl omhoog** in te drukken wordt de laatste opdracht herhaald.

Tafel van vermenigvuldiging

Om verder te illustreren hoe loops werken en hoe nuttig ze kunnen zijn, zal het volgende korte programma gebruik maken van twee **FOR** lussen om de tafel van vermenigvuldiging uit te printen die we allemaal op school hebben geleerd. Het programma hiervoor is niet ingewikkeld:

```
FOR nbr1 = 1 To 10          ' Kleuren vanuit Edit.
  FOR nbr2 = 1 To 10
    PRINT nbr1 * nbr2,;
  NEXT nbr2
  PRINT
NEXT nbr1
```

```
> list                      ' Kleuren vanuit de commando regel.
For NBR1 = 1 To 10
  For nbr2 = 1 To 10
    Print nbr1 * nbr2,;
  Next nbr2
  Print
Next nbr1
> run
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      28      32      36      40
5       10      15      20      25      30      35      40      45      50
6       12      18      24      30      36      42      48      54      60
7       14      21      28      35      42      49      56      63      70
8       16      24      32      40      48      56      64      72      80
9       18      27      36      45      54      63      72      81      90
10      20      30      40      50      60      70      80      90      100
>
```

De uitvoer wordt weergegeven op het scherm boven, wat ook laat zien een list van het programma.

U moet de logica van dit voorbeeld regel voor regel doornemen om te begrijpen wat het doet.
In wezen bestaat het uit de ene lus in de andere. De binnenste lus, die de variabele verhoogt **nbr2** drukt één horizontale lijn van de tabel af. Wanneer deze lus is voltooid, wordt het volgende uitgevoerd: **PRINT** opdracht die niets heeft om af te drukken - dus het zal gewoon een nieuwe regel uitvoeren (dwz de . beëindigen lijn gedrukt door de binnenste lus).
Het programma zal dan nog een iteratie van de buitenste lus uitvoeren door **nbr1** en te verhogen opnieuw uitvoeren van de binnenste lus. Ten slotte, wanneer de buitenste lus is uitgeput (wanneer **nbr1** groter is dan **10**) het programma zal het einde bereiken en eindigen.
Een laatste punt, je kunt de naam van de variabele weglaten uit de **NEXT** instructie en MMBasic zal het raden welke variabele je bedoelt. Het is echter een goede gewoonte om de naam op te nemen om het gemakkelijker te maken voor iemand anders die het programma leest om het te begrijpen.
U kunt ook meerdere lussen beëindigen met behulp van een door komma's gescheiden lijst van variabelen in de **NEXT** instructie. Bijvoorbeeld:

```
FOR var1 = 1 TO 5
  FOR var2 = 10 TO 13
    PRINT var1 * var2
NEXT var1, var2
```

DO LOOP

Een andere methode van loopen is de **DO...LOOP** structuur die er als volgt uitziet:

```
DO WHILE voorwaarde
  uitspraak
  uitspraak
LOOP
```

Dit begint met het testen van de voorwaarde en als deze waar is, worden de instructies uitgevoerd totdat de **LOOP** commando is bereikt, waarna de voorwaarde opnieuw wordt getest en als het nog steeds waar is, zal de lus opnieuw uitvoeren. De '**voorwaarde**' is hetzelfde als in het **IF** commando (dwz **X < Y**).

Het volgende zal bijvoorbeeld het woord **"Hallo"** gedurende **4 seconden** op de console blijven afdrukken en dan stoppen:

```
Timer = 0
DO WHILE Timer < 4000  ' Tijd is 4000 mS is 4 sec.
  PRINT "Hallo"
LOOP
```

Merk op dat Timer een functie is binnen MMBasic die de tijd in milliseconden teruggeeft sinds de timer is gereset. Een reset wordt gedaan door nul toe te wijzen aan Timer (zoals hierboven gedaan) of bij het opstarten van de **PicoMite**.

Een variatie op de **DO - LOOP** structuur is de volgende:

```
DO
  uitspraak
  uitspraak
LOOP UNTIL conditie
```

In deze opstelling wordt de lus eerst één keer uitgevoerd, vervolgens wordt de voorwaarde getest en als de voorwaarde is false, wordt de lus herhaaldelijk uitgevoerd totdat de voorwaarde waar wordt. Merk op dat de test in **LOOP UNTIL** is het omgekeerde van **DO WHILE**.

Bijvoorbeeld, vergelijkbaar met het vorige voorbeeld, zal het volgende ook vier seconden lang **"Hallo"** afdrukken:

```
Timer = 0
DO
  PRINT "Hallo"
LOOP UNTIL Timer >= 4000
```

Beide vormen van de **DO** - **LOOP** doen in wezen hetzelfde, dus je kunt elke structuur gebruiken die past met de logica die u wilt implementeren.

Ten slotte is het mogelijk om een **DO** - **LOOP** te hebben die helemaal geen voorwaarden heeft - dat wil zeggen,

```
DO
  uitspraak
  uitspraak
LOOP
```

Deze constructie blijft voor altijd in een lus lopen en jij, als programmeur, moet een manier bieden om de lus expliciet te verlaten (de opdracht **EXIT DO** doet dit). Bijvoorbeeld:

```
Timer = 0
DO
  PRINT "Hallo"
  IF Timer >= 4000 THEN EXIT DO
LOOP
```

Console invoer

Naast het afdrukken van gegevens zodat de gebruiker uw programma's kan zien, willen ze ook input van de gebruiker.
Om dat te laten werken, moet je toetsaanslagen vastleggen vanaf de console en dit kan worden gedaan met de **INPUT** opdracht. In zijn eenvoudigste vorm is het commando:

```
INPUT var
```

Met deze opdracht wordt een vraagteken op het scherm van de console afgedrukt en wordt gewacht tot er een getal wordt ingevoerd gevolgd door de **Enter-toets**. Dat nummer wordt dan toegewezen aan de variabele **var**.
Het volgende programma breidt bijvoorbeeld de uitdrukking voor het vinden van de schuine zijde van een driehoek uit waardoor de gebruiker de lengtes van de andere zijden van de console kan invoeren.

```
PRINT "Lengte van zijde 1"
INPUT a
PRINT "Lengte van zijde 2"
INPUT b
PRINT "Lengte van de schuine zijde is" SQR(a * a + b * b)
```

Dit is een schermafbeelding van een typische sessie:

```
> run
Lengte van zijde 1
? 12
Lengte van zijde 2
? 15
Lengte van de schuine zijde is 19.20937271
```

Het **INPUT** commando kan uw prompt ook voor u afdrukken, zodat u geen aparte **PRINT** opdracht nodig heeft. Dit werkt bijvoorbeeld hetzelfde als het bovenstaande programma:

```
INPUT "Lengte van zijde 1"; a
INPUT "Lengte van zijde 2"; b
PRINT "Lengte van de schuine zijde is" SQR(a * a + b * b)
```

Ten slotte kunt u met het **INPUT** commando een reeks getallen invoeren, gescheiden door komma's met elk nummer wordt opgeslagen in verschillende variabelen.

Bijvoorbeeld:

```
INPUT "Voer de lengte van de twee zijden in: ", a, b
PRINT "Lengte van de schuine zijde is" SQR(a * a + b * b)
```

Als de gebruiker **12, 15** invoert, wordt het getal **12** opgeslagen in de variabele **a** en **15** in **b**.

Een andere methode om invoer van de console te krijgen, is het **LINE INPUT** commando. Dit krijgt de hele regel zoals getypt door de gebruiker en wijs deze toe aan een stringvariabele. Net als het **INPUT** commando kunt u ook een prompt opgeven. Dit is een eenvoudig voorbeeld:

```
LINE INPUT "Wat is uw naam?", s$
PRINT "Hallo" s$
```

We zullen stringvariabelen later in deze handleiding behandelen, maar voorlopig kun je ze zien als een variabele die een reeks van een of meer tekens bevat. Als je het bovenstaande programma hebt uitgevoerd en hebt ingetypt: **John** wanneer daarom wordt gevraagd, zou het programma reageren met **Hallo John.**

Soms wil je niet wachten tot de gebruiker op de enter-toets drukt, maar wil je het na elk teken krijgen nadat het wordt ingetypt. Dit kan gedaan worden met de **INKEY$** functie die de waarde van het karakter teruggeeft als een tekenreeks die uit slechts één teken bestaat of een lege tekenreeks als er niets is ingevoerd.

GOTO en Labels

Een methode om de stroom van het programma te regelen is het **GOTO** commando. Dit vertelt in wezen MMBasic om naar een ander deel van het programma te springen en van daaruit verder te gaan. Het doelwit van de **GOTO** is een label en dit moet eerst worden uitgelegd.

Een label is een identificatie die een deel van het programma markeert. Het moet het eerste aan de lijn zijn en het moet worden afgesloten met de dubbele punt **(:)**. De naam die u gebruikt mag max-imaal 32 tekens lang zijn en moet dezelfde regels volgen als voor de naam van een variabele. Bijvoorbeeld in de volgende programmaregel:
LoopBack is een label:

```
LoopBack: a = a + 1
```

Wanneer u het **GOTO** commando gebruikt om naar dat specifieke deel van het programma te gaan, gebruikt u het commando als volgt:

```
GOTO LoopBack
```

Om dit alles in context te plaatsen, zal het volgende programma alle getallen van **1** tot **10** afdrukken:

```
z = 0
LoopBack:
z = z + 1
PRINT z
IF z < 10 THEN GOTO LoopBack
```

Het programma begint met het instellen van de variabele z op nul en verhoogt deze vervolgens naar 1 op de volgende regel. De waarde van z wordt afgedrukt en vervolgens getest om te zien of het minder dan 10 is. Als het minder dan 10 is, zal de programma uitvoering terug springen naar het label **LoopBack** waar het proces wordt herhaald. uiteindelijk de waarde van z zal meer dan 10 zijn en het programma zal aan het einde lopen en eindigen.

Merk op dat een **FOR** lus hetzelfde kan doen (en eenvoudiger is), dus dit voorbeeld is puur bedoeld om: illustreren wat het commando **GOTO** kan doen.

In het verleden kreeg het **GOTO** commando een slechte reputatie. Dit komt omdat het gebruik van GOTO's mogelijk is om een programma te maken dat continu van het ene punt naar het andere springt (vaak aangeduid als "spaghetti code") en dat type programma is bijna onmogelijk voor een andere programmeur om te begrijpen. Met constructies zoals de meerregelige **IF** instructies is de behoefte aan de **GOTO** instructie verminderd en het mag alleen worden gebruikt als er geen andere manier is om de programmastroom te wijzigen.

Testen op priemgetallen

Het volgende is een eenvoudig programma dat veel van de programmeerfuncties samenbrengt eerder besproken.

```
DO
  InpErr:
  PRINT
  INPUT "Voer een getal in: "; a
  IF a < 2 THEN
    PRINT "Getal moet gelijk zijn aan of groter zijn dan 2"
    GOTO InpErr
  ENDIF

  Divs = 0
  FOR x = 2 TO SQR(a)
    r = a / x
    IF r = FIX(r) THEN Divs = Divs + 1
  NEXT x

  PRINT a " is";
  IF Divs > 0 THEN PRINT " niet";
  PRINT " een priemgetal."
LOOP
```

Dit zal eerst (op de console) om een nummer vragen en, wanneer het is ingevoerd, zal het testen of dat nummer is een priemgetal of niet en geef een geschikt bericht weer.

Het begint met een **DO** lus die geen voorwaarde heeft - dus het zal voor altijd doorgaan. Dit is wat we willen. Het betekent dat wanneer de gebruiker een getal heeft ingevoerd, hij zal melden of het een priemgetal is of niet en loop dan rond en vraag om een ander nummer. De manier waarop de gebruiker het programma kan verlaten (als ze dat wilden) is door het break-teken te typen (normaal gesproken **CTRL-C**).

Het programma drukt vervolgens een prompt voor de gebruiker af die wordt afgesloten met een puntkomma. Deze betekent dat de cursor links staat aan het einde van de prompt voor het **INPUT** commando, dat de krijgt nummer en sla het op in de variabele **a**.

Hierna wordt het nummer getest. Als het minder dan **2** is, wordt een foutbericht afgedrukt en het programma zal achteruit springen en opnieuw om het nummer vragen.

We zijn nu klaar om te testen of het getal een priemgetal is. Het programma gebruikt een **FOR** lus om te stappen door de mogelijke delers te testen of ze allemaal gelijk kunnen worden verdeeld in het ingevoerde getal. Elke keer het doet het programma zal de variabele **Divs** verhogen.

Merk op dat de test wordt gedaan met de functie **FIX(r)** die eenvoudig alle cijfers achter de komma verwijdert punt. Dus de voorwaarde **r = FIX(r)** zal waar zijn als **r** een geheel getal is (dwz geen cijfers heeft na de decimale punt).

Ten slotte zal het programma het bericht voor de gebruiker samenstellen. Het belangrijkste is dat als de variabele **Divs** groter is dan **nul**, betekent dit dat er een of meer getallen zijn gevonden die gelijkmatig in de kunnen worden verdeeld testnummer. In dat geval voegt het **IF** statement het woord **"not"** in het uitvoerbericht in.

Als het ingevoerde nummer bijvoorbeeld 21 was, ziet de gebruiker dit antwoord:

```
21 is geen priemgetal.
```

Dit is het resultaat van het uitvoeren van het programma en een deel van de uitvoer:

```
> RUN

Voer een getal in: ? 64
64 is niet een priemgetal.

Voer een getal in: ? 67
67 is een priemgetal.

Voer een getal in: ? 0
Getal moet gelijk zijn aan of groter zijn dan 2

Voer een getal in: ?
```

U kunt dit programma testen door de editor (het **EDIT** commando) te gebruiken om het in te voeren. Dan is het ook duidelijker met de kleuren en overzichtelijker. De kleuren zijn in deze handleiding ook gebruikt.

Met behulp van je nieuw geleerde vaardigheden zou je dan een kans kunnen maken om het efficiën-ter te maken. Bijvoorbeeld, omdat het programma telt hoe vaak een getal kan worden verdeeld in het testgetal duurt het a veel langer dan nodig is om een niet-priemgetal te detecteren. Het program-ma zou veel efficiënter werken als het uit de **FOR** lus is gesprongen bij het eerste getal dat gelijk-matig is verdeeld. Je zou de **GOTO** commando kunnen gebruiken om dit te doen of je zou het commando **EXIT FOR** kunnen gebruiken – dat zou ervoor zorgen dat de **FOR** lus onmiddellijk beëindigen.

Andere efficiëntieverbeteringen zijn onder meer het testen van de deling met oneven getallen (door een eerste test te gebruiken voor een even getal en start de **FOR** lus bij 3 en gebruik STAP 2) of door alleen priemgetallen te gebruiken voor de test (dat zou veel ingewikkelder zijn).

Arrays

Arrays zijn iets waar je waarschijnlijk niet aan denkt handig op het eerste gezicht, maar als je ze nodig hebt, zul je vind ze inderdaad erg handig.



Een array kun je het beste zien als een rij brievenbussen voor een blok van eenheden of apparte-menten zoals rechts getoond. De brievenbussen zijn allemaal zich op hetzelfde adres bevinden en elk vakje staat voor een eenheid of appartement op dat adres. U kunt een brief in de doos voor een-heid plaatsen één, of eenheid twee, enz.

Evenzo is een array in BASIC een enkele variabele met meerdere subeenheden (elementen genoemd in BASIC) die genummerd zijn. Jij kan gegevens in element één of element twee plaatsen, enz.
Array van 3

In BASIC wordt een array gemaakt door het **DIM** commando, bijvoorbeeld:

```
DIM numarr(300)
```

Dit creëert een array met de naam numarr die **301** elementen bevat (beschouw ze als brievenbussen) variërend van **0** tot **300**. Standaard begint een array vanaf nul, daarom is er een extra element maakt het totaal **301**. Om een specifiek element in de array te specificeren (dwz een specifieke brievenbus) u gebruikt een index die eenvoudigweg het nummer is van het array-element dat u wilt openen. Voor als u bijvoorbeeld element **nummer 100** in deze array wilt instellen op (zeg) het **nummer 876**, zou u dat doen op deze manier:

```
numarr(100) = 876
```

Normaal gesproken is de index naar een array geen constant getal zoals in dit voorbeeld (dwz 100) maar een variabele die kan worden gewijzigd om toegang te krijgen tot verschillende array-elemen-ten.

Als een voorbeeld van hoe u een array zou kunnen gebruiken, kunt u het geval overwegen waarin u de maximumtemperatuur voor elke dag van het jaar en bereken aan het einde van het jaar de totale
gemiddeld. Je zou gewone variabelen kunnen gebruiken om de temperatuur voor elke dag vast te leggen, maar je zou nodig hebben 365 van hen en dat zou je programma inderdaad onpraktisch maken. In plaats daarvan zou je een array kunnen definiëren om de waarden als volgt vast te houden:

```
DIM days(365)
```

Elke dag zou u de temperatuur op de juiste plaats in de array moeten opslaan. Als het aantal de dag in het jaar werd gehouden in de variabele doy en de maximale temperatuur werd gehouden in de
variabele maxtemp zou je de meting als volgt opslaan:

```
days(doy) = maxtemp
```

Aan het einde van het jaar zou het eenvoudig zijn om het gemiddelde van het jaar te berekenen:

```
totaal = 0
FOR i = 1 to 365
  totaal = totaal + days(i)
NEXT i
PRINT "Gemiddeld is:" totaal / 365
```

Dit is veel eenvoudiger dan **365** individuele variabelen optellen en middelen.



De bovenstaande array was enkeldimensionaal, maar u kunt er meerdere hebben dimensies. Terug-komend op onze analogie van brievenbussen, een array met twee dimensies kunnen worden gezien als een flatgebouw met meerdere vloeren. Een blok kan een rij van vier brievenbussen hebben voor niveau één, nog een rij van vier dozen voor niveau twee, enzovoort. Een brief plaatsen in een brievenbus moet u het verdiepingsnummer en de eenheid opgeven nummer op die verdieping.
In BASIC wordt zo'n array gespecificeerd met behulp van twee indices gescheiden door een komma. Bijvoorbeeld:

```
Brievenbus(verdieping, unit)
```

Neem als praktisch voorbeeld aan dat u de maximale temperatuur voor elke dag over vijf jaar. Om dit te doen, kunt u de array als volgt dimensioneren:

```
DIM days(365, 5)
```

De eerste index is de dag van het jaar en de tweede is een getal dat het jaar vertegenwoordigt. Als je wilt om **dag 100** in **jaar 3** in te stellen op **24 graden** doe je het als volgt:

```
days(100, 3) = 24
```

In MMBasic voor de PicoMite kun je tot vijf dimensies hebben (dit is anders dan bij sommige andere) versies van MMBasic die acht dimensies ondersteunen). De maximale grootte van een array is slechts beperkt door de hoeveelheid vrij RAM die beschikbaar is.

Integers

Tot nu toe zijn alle getallen en variabelen die we hebben gebruikt zwevende komma's. Zoals uitgelegd voorheen is drijvende komma handig omdat het cijfers achter de komma volgt en wanneer u gebruikt verdeling zal het een verstandig resultaat opleveren. Dus, als je gewoon dingen voor elkaar wilt krijgen en je geen zorgen maakt met de details moet u zich aan de drijvende komma houden.

De beperking van drijvende komma is echter dat het getallen opslaat als een benadering met een nauwkeurigheid van **14 cijfers** op de **PicoMite**. Meestal is dit kenmerk van drijvendekom-magetallen niet een probleem, maar er zijn gevallen waarin u grotere aantallen nauwkeurig moet opslaan.

Laten we als voorbeeld stellen dat u de tijd tot op de microseconde nauwkeurig wilt manipuleren, dus dat je twee verschillende datums/tijden kunt vergelijken om erachter te komen welke eerder is. De gemakkelijke manier om te doen dit is om de datum/tijd om te zetten in het aantal microseconden sinds een bepaalde datum (zeg **1 januari** in het **jaar nul**) - dan is het vinden van de vroegste van de twee een kwestie van een rekenkundige vergelijking gebruiken in een **IF** uitspraak.

Het probleem is dat het aantal microseconden sinds die datum het nauwkeurigheidsbereik van zal overschrijden drijvende-kommavariabelen en dit is waar integer-variabelen binnenkomen. Een integer variabele kan nauwkeurig bevatten zeer grote aantallen tot **negen miljoen miljoen miljoen** (of **±9223372036854775807** om precies te zijn).

Het nadeel van het gebruik van een geheel getal is dat het geen breuken kan opslaan (dwz getallen achter de komma).
Elke berekening die een fractioneel resultaat oplevert, wordt naar boven of naar beneden afgerond op het dichtstbijzijnde geheel nummer wanneer toegewezen aan een geheel getal. Deze eigen-schap kan echter handig zijn bij het omgaan met geld - u wilt iemand bijvoorbeeld geen rekening sturen van **$100,13333333333**.

Het is gemakkelijk om een integer variabele te maken, voeg gewoon het procentteken **(%)** toe als achtervoegsel aan een variabelenaam.
sec% is bijvoorbeeld een integer-variabele. Binnen een programma kun je gehele getallen en drijvende komma mixen en MMBasic zal de nodige conversies maken, maar als u de volledige nauwkeurigheid van wilt behouden integers moet u voorkomen dat u de twee vermengt.

Net als met drijvende komma kun je arrays van gehele getallen hebben met maximaal vijf dimen-sies, alles wat je hoeft te doen is voeg het procentteken toe als achtervoegsel aan de arraynaam. Bijvoorbeeld:

```
days%(365, 5)
```

Beginners raken vaak in de war over wanneer ze drijvende komma of gehele getallen moeten gebruiken en het antwoord is eenvoudig ... gebruik altijd een drijvende komma, tenzij u een extreem hoog niveau van nauw-keurigheid nodig hebt in de resulterend nummers. Dit gebeurt niet vaak, maar als je ze nodig hebt, zul je merken dat gehele getallen zijn een redder in nood.

Strings

Strings zijn een ander type variabele (zoals drijvende komma en gehele getallen). Strings worden gebruikt om een reeks vast te houden van karakters. Bijvoorbeeld in het commando:

```
PRINT "Hallo"
```

De tekenreeks **"Hallo"** is een tekenreeksconstante. Merk op dat een constante iets is dat niet veran-dert (als tegen een variabele, die kan) en die **stringconstanten** worden altijd omringd door dubbele aanhalingstekens.

Namen van stringvariabelen gebruiken het dollarteken **($)** als achtervoegsel om ze te identificeren als een string in plaats van een normale variabele met drijvende komma en u kunt gewone toewijzing gebruiken om hun waarde in te stellen. Het volgende zijn voorbeelden (merk op dat het tweede voorbeeld een reeks tekenreeksen gebruikt):

```
     Auto$ = "Houd vast"
Land$(12) = "India"
     Naam$ = "Fred"
```

U kunt strings ook samenvoegen met de plus-operator:

```
    Word1$ = "Hallo"
    Word2$ = "Wereld"
Greeting$ = Word1$ + " " + Word2$
```

In dat geval is de waarde van **Greeting$** = **"Hallo Wereld"**.
Strings kunnen ook worden vergeleken met operatoren zoals:
= (is gelijk aan), **<> (niet gelijk aan)**, **< (kleiner dan)**, enz.

Bijvoorbeeld:

```
IF Car$ = "Holden" THEN PRINT "Was een Aussie made car"
```

De vergelijking wordt gemaakt met behulp van de volledige **ASCII**-tekenset, dus er komt een spatie vóór een afdrukbare karakter. Ook is de vergelijking hoofdlettergevoelig, dus **'holden'** is niet gelijk aan **'Holden'**. De ... gebruiken functie **UCASE()** om de tekenreeks naar hoofdletters te converteren, kunt u een hoofdletterongevoelige vergelijking maken.

Bijvoorbeeld:

```
IF UCASE$(Car$) = "HOLDEN" THEN PRINT "Was an Aussie made car"
```

Je kunt arrays met strings hebben, maar je moet voorzichtig zijn als je ze declareert, want dat kan snel geen RAM meer heeft (algemeen geheugen dat wordt gebruikt voor het opslaan van variabelen, enz.). Dit komt omdat MMBasic zal door wijs standaard **255 bytes RAM** toe aan elk element van de array. Bijvoorbeeld een stringarray met **100** elementen gebruiken standaard **25K RAM**. Om dit te verlichten, kunt u de **LENGTH**-kwalificatie gebruiken om beperk de maximale grootte van elk element. Als u bijvoorbeeld weet dat de maximale lengte van elke tekenreeks die in de array wordt opgeslagen, zal minder dan **20** tekens bevatten, u kunt het volgende gebruiken declaratie om slechts **20 bytes** toe te wijzen aan elk element:

```
DIM MyArray$(100) LENGTH 20
```

De resulterende array gebruikt slechts **2K RAM**.

Strings manipuleren

String handling is een van de sterke punten van MMBasic en met een paar eenvoudige func-ties die je uit elkaar kunt halen en in het algemeen snaren manipuleren. De basis stringfuncties zijn:

`LEFT$(string$, nbr)`	Retourneert een substring van string$ met een aantal tekens van links (begin) van de tekenreeks.
`RIGHT$(string$, nbr)`	Hetzelfde als hierboven, maar retourneer het aantal tekens vanaf de rechterkant (einde) van de string.
`MID$(string$,pos,nbr)`	Retourneert een substring van string$ met een aantal tekens vanaf dekarakter pos in de string (dwz het midden van de string).

Bijvoorbeeld als `S$ = "Dit is een string"`

dan: `R$ = LEFT$(S$,7)`	zou ertoe leiden dat de waarde van R$ wordt ingesteld op: "This is"
en: `R$ = RIGHT$(S$,8)`	zou ertoe leiden dat de waarde van R$ wordt ingesteld op: "a string"
tenslotte: `R$ = MID$(S$,6,2)`	zou ertoe leiden dat de waarde van R$ wordt ingesteld op: "is"

Merk op dat in **MID$()** de **eerste tekenpositie** in een tekenreeks **nummer 1** is, **de tweede nummer 2** en zo aan. Dus, het **eerste teken** als **één tellend**, is de **zesde positie het begin van het woord "is"**

Een andere handige functie is:

`INSTR(string$, patroon$)`	Geeft een getal terug dat de positie vertegenwoordigt waarop patroon$ voorkomt in string$.

Dit kan worden gebruikt om te zoeken naar een string binnen een andere string. Het geretourneerde nummer is de positie van de substring in de hoofdstring. Net als bij MID$() is het begin van de string positie 1.

Bijvoorbeeld als `S$ = "Dit is een string"`

Dan: `pos = INSTR(S$, " ")`

zou resulteren in het instellen van **pos** op de positie van de eerste spatie in S$ (dwz 5).

INSTR() kan worden gecombineerd met andere functies, dus dit zou het eerste woord in S$ opleveren:

```
R$ = LEFT$(S$, INSTR(S$, " ") - 1)
```

Er is ook een uitgebreide versie van **INSTR():**

`INSTR(pos, string$, pattern$)`	Geeft een getal terug dat de positie vertegenwoordigt waarop patroon$ komt voor in string$ bij het starten van de zoekopdracht op de tekenpositie pos.

Dus we kunnen het tweede woord in **S$** vinden met behulp van het volgende:

```
pos = INSTR(S$, " ")
 R$ = LEFT$(S$, INSTR(pos + 1, S$, " ") - 1)
```

Dit laatste voorbeeld is nogal ingewikkeld, dus het kan de moeite waard zijn om het in detail uit te werken, zodat u kan begrijpen hoe het werkt.

Merk op dat **INSTR()** het getal nul retourneert als de subtekenreeks niet wordt gevonden en dat elke tekenreeks functie zal een fout genereren (en het programma stoppen) als dat wordt gebruikt als een tekenpositie. Dus, in een praktisch programma zou u eerst controleren of nul wordt geretour-neerd door **INSTR()** voordat u die waarde gebruikt.

Bijvoorbeeld:

```
pos = INSTR(S$, " ")
IF pos > 0 THEN R$ = LEFT$(S$, INSTR(pos + 1, S$, " ") - 1)
```

Wetenschappelijke notatie

Voordat we klaar zijn met het bespreken van gegevenstypen, moeten we het onderwerp drijvendekommagetallen behandelen en wetenschappelijke notatie.

De meeste getallen kunnen normaal worden geschreven, bijvoorbeeld **11** of **24.5**, maar zeer grote of kleine getallen zijn moeilijker. Er is bijvoorbeeld geschat dat het aantal zandkorrels op planeet Aarde is **7500000000000000000**. Het probleem met dit nummer is dat je gemakkelijk uit het oog kunt verliezen hoeveel er zitten nullen in het getal en daarom is het moeilijk om dit te vergelijken met een vergelijkbare maat nummer.

Een wetenschapper zou dit getal schrijven als **7.5 x 10^18**, wat wetenschappelijke notatie wordt genoemd en veel gemakkelijker is te begrijpen.

MMBasic schakelt automatisch over naar wetenschappelijke notatie bij zeer grote of kleine zwevende bewegingen punt nummers. Als het bovenstaande nummer bijvoorbeeld is opgeslagen in een variabele met drijvende komma, wordt de PRINT commando zou het weergeven als **7.5E+18** (dit is BASIC's manier om 7.5 x 10^18 weer te geven). Als een ander het nummer **0.0000000456** wordt bijvoorbeeld weergegeven als **4.56E-8**, wat hetzelfde is als **4.56 x 10^-8**.

U kunt ook wetenschappelijke notatie gebruiken bij het invoeren van constante getallen in MMBasic. Bijvoorbeeld:

```
Zandkorrels = 7.5E+18
```

MMBasic gebruikt alleen wetenschappelijke notatie voor het weergeven van getallen met drijvende komma (geen gehele getallen). Voor als u bijvoorbeeld het aantal zandkorrels aan een integer-variabele zou toewijzen, zou het worden afgedrukt als een normaal getal (met veel nullen).

DIM opdracht

We hebben eerder het DIM-commando gebruikt voor het definiëren van arrays, maar het kan ook worden gebruikt om gewone variabelen te maken. U kunt bijvoorbeeld tegelijkertijd vier tekenreeksvariabelen maken, zoals deze:

```
DIM STRING Auto, Naam, Straat, Stad
```

Merk op dat omdat deze variabelen zijn gedefinieerd als **strings** met het **DIM** commando, we dat niet doen het achtervoegsel **$** nodig hebben, is de definitie alleen voldoende voor MMBasic om hun type te identificeren. Evenzo, wanneer u deze variabelen gebruikt in een uitdrukking waarvoor u het achtervoegsel van het type niet nodig hebt: Bijvoorbeeld:

```
Stad = "Sydney"
```

U kunt ook het trefwoord **INTEGER** gebruiken om een aantal integer-variabelen te definiëren en **FLOAT** om te doen hetzelfde voor variabelen met drijvende komma. Dit type notatie kan op dezelfde manier worden gebruikt om arrays te definiëren.

Bijvoorbeeld:

```
DIM INTEGER seconden(200)
```

Een andere methode om het type variabelen te definiëren, is door het trefwoord **AS** te gebruiken.

Bijvoorbeeld:

```
DIM Auto AS STRING, Naam AS STRING, Straat AS STRING
```

Dit is de methode die wordt gebruikt door Microsoft (MMBasic probeert de compatibiliteit met Microsoft te behouden) en het is handig als de variabelen verschillende typen hebben.
Bijvoorbeeld:

```
DIM Auto$ AS STRING, Leeftijd% AS INTEGER, Waarde! AS FLOAT
```

U kunt elk van deze methoden gebruiken om het type van een variabele te definiëren, ze werken allemaal hetzelfde.

Het voordeel van het definiëren van variabelen met het **DIM** commando is dat ze duidelijk gedefinieerd zijn (bij voorkeur aan het begin van het programma) en hun type (float, integer of string) is niet onderhevig aan verkeerde interpretatie.

Je kunt dit versterken door de volgende commando's helemaal bovenaan je programma te gebruiken:

```
OPTION EXPLICIT
OPTION DEFAULT NONE
```

De eerste specificeert voor MMBasic dat alle variabelen expliciet moeten worden gedefinieerd met **DIM** voordat ze kunnen worden gebruikt. De tweede specificeert dat het type van alle variabelen moet worden opgegeven wanneer ze worden gemaakt.

Waarom zijn deze twee commando's belangrijk?

De eerste kan een veelvoorkomende programmeerfout helpen voorkomen, waarbij u per ongeluk een verkeerd gespelde naam van de variabele. Uw programma kan bijvoorbeeld de huidige temperatuur hebben opgeslagen in een variabele genaamd Temp, maar op een gege-ven moment spel je het per ongeluk verkeerd als Tmp. Hierdoor zal MMBasic maak automatisch een variabele aan met de naam **Tmp** en stel de waarde in op **nul**.

Dit is duidelijk niet wat je wilt en het zal een subtiele fout introduceren die moeilijk te vinden zou kunnen zijn - zelfs als je wist dat er iets niet klopte. Aan de andere kant, als u de **OPTION EXPLICIT** commando aan het begin van uw programma MMBasic zou weigeren om automatisch te maken variabele en zou in plaats daarvan een fout weergeven, waardoor u waar-schijnlijk hoofdpijn krijgt.

Het commando **OPTION** **DEFAULT NONE** helpt verder omdat het MMBasic vertelt dat de prog-rammeur moet specifiek het type van elke variabele specificeren wanneer ze worden gedeclareerd. Het is makkelijk om vergeet het type op te geven en MMBasic automatisch te laten aannemen dat het type kan leiden tot onverwachte gevolgen.

Voor kleine, snelle en vuile programma's is het prima om MMBasic automatisch variabelen te laten maken, maar in grotere programma's moet je deze functie altijd uitschakelen met **OPTION EXPLICIET** en versterken met **OPTION** **DEFAULT NONE**.

Wanneer een variabele wordt gemaakt, wordt deze op nul gezet voor float en integers en een lege string (dwz bevat no tekens) voor een tekenreeksvariabele. U kunt de beginwaarde instellen op iets anders wanneer het is gemaakt met **DIM**. Bijvoorbeeld:

```
DIM FLOAT nbr! = 12.56
DIM STRING Auto$ = "Ford", Stad$ = "Perth"
```

U kunt arrays ook initialiseren door de initialiserende waarden als volgt tussen haakjes te plaatsen:

```
DIM s$(2) = ("nul", "één", "twee")
```

Merk op dat omdat arrays standaard vanaf nul beginnen, deze array eigenlijk drie elementen heeft met de indexnummers van **0, 1** en **2**. Daarom hadden we drie stringconstanten nodig om het te initialiseren.

DIM LOCAL STATIC

Syntaxis:

```
DIM  [type] decl [,decl]...
LOCAL  variabele [, variabelen]...
STATIC variabele [, variabelen]...
```

Beschrijving:

```
DIM [type] decl [,decl]...
```

Declareert een of meer variabelen (dwz maakt de naam van de variabele en zijn kenmerken bekend aan de interpreter).
Wanneer **OPTION EXPLICIT** wordt gebruikt (zoals aanbevolen), zijn de opdrachten **DIM**, **LOCAL** of **STATIC** de enige manier waarop een variabele kan worden gemaakt.
Als deze optie **niet** wordt gebruikt, is het gebruik van de **DIM** opdracht optioneel en als deze optie niet wordt gebruikt, wordt de variabele automatisch gemaakt wanneer er voor het eerst naar wordt verwezen.
Het type variabele (dwz **string**, **float** of **integer**) kan op drie manieren worden opgegeven:

Door een typeachtervoegsel te gebruiken (dwz**!**, **%** of **$** voor **float** = **!**, **integer** = **%** of **string** = **$**).
Bijvoorbeeld: `DIM nbr%, bedrag!, naam$`

Door een van de trefwoorden **FLOAT**, **INTEGER** of **STRING** te gebruiken direct na het commando **DIM** en voordat de variabele(n) worden weergegeven.
Het gespecificeerde type is dan van toepassing op alle vermelde variabelen (dwz het hoeft niet herhaald te worden).
Bijvoorbeeld: `DIM STRING voornaam, achternaam, stad`

Door gebruik te maken van de Microsoft-conventie van het gebruik van het trefwoord "**AS**" en het type trefwoord (dwz **FLOAT**, **INTEGER** of **STRING**) na elke variabele.
Als u deze methode gebruikt, moet het type voor elke variabele worden opgegeven en kan het van variabele naar variabele worden gewijzigd.
Bijvoorbeeld: `DIM bedrag AS FLOAT, naam AS STRING`

Variabelen met drijvende komma of integer worden op nul gezet wanneer ze worden gemaakt en tekenreeksen worden ingesteld op een lege tekenreeks (dwz "").
U kunt de waarde van de variabele met iets anders initialiseren door een gelijkteken (**=**) en een uitdrukking na de variabeledefinitie te gebruiken.
Bijvoorbeeld: `DIM STRING stad = "Perth", huis = "Brick"`
De initialisatiewaarde kan een uitdrukking zijn (inclusief andere variabelen) en wordt geëvalueerd wanneer het **DIM** commando wordt uitgevoerd.
Zie het hoofdstuk in de handleiding "Variabelen definiëren en gebruiken" voor meer voorbeelden van de syntaxis.

Naast het declareren van eenvoudige variabelen, declareert het **DIM** commando ook **arrayvariabelen** (dwz een geïndexeerde variabele met **maximaal vijf dimensies** op de CMM2).
Merk op dat dit verschilt van de originele Color Maximite- en Micromite-versies van MMBasic die tot acht dimensies ondersteunden.
Na de naam van de variabele worden de afmetingen gespecificeerd door een lijst met getallen, gescheiden door komma's en tussen haakjes.

Bijvoorbeeld: `DIM array (10, 20)`

Elk getal geeft het aantal elementen in elke dimensie aan. Normaal begint de nummering van elke dimensie bij 0, maar de **OPTION BASE** commando kan worden gebruikt om dit te wijzigen in **1**.
Het bovenstaande voorbeeld specificeert een tweedimensionale array met **11** elementen (**0** tot **10**) in de eerste dimensie en **21** (**0** tot **20**) in de tweede dimensie.
Het totale aantal elementen is **231** en omdat elk getal met drijvende komma op de Color Maximite 2 **8 bytes** vereist, zal in totaal **1848** bytes geheugen nodig zijn en worden toegewezen.
Strings zullen standaard **255 bytes** (dwz karakters) geheugen toewijzen aan elk element en dit kan snel geheugen verbruiken bij het definiëren van arrays van strings.
In dat geval kan het sleutelwoord **LENGTH** worden gebruikt om de hoeveelheid geheugen te specificeren die aan elk element moet worden toegewezen en dus de maximale lengte van de string die kan worden opgeslagen.
Deze toewijzing **('n')** kan **1** tot **255** tekens lang zijn.
Bijvoorbeeld: `DIM STRING s(5,10)` declareert een stringarray met **66** elementen die **16.896 bytes** geheugen verbruiken, terwijl: `DIM STRING s(5,10) LENGTH 20` slechts **1.386 bytes** geheugen verbruikt.
Merk op dat de hoeveelheid geheugen die aan elk element is toegewezen **n + 1** is, aangezien de extra byte wordt gebruikt om de werkelijke lengte van de string die in elk element is opgeslagen bij te houden.
Als een string **langer dan 'n'** wordt toegewezen aan een element van de array, wordt er een fout geproduceerd.
Anders dan dit, werken **string arrays** die zijn gemaakt met het **LENGTH sleutelwoord** precies hetzelfde als andere string-arrays.
Dit sleutelwoord kan ook worden gebruikt met niet-array-tekenreeksvariabelen, maar het zal geen geheugen besparen tenzij de lengte kleiner is dan 16, wanneer het zowel geheugen bespaart als de prestaties verbetert.
In het bovenstaande voorbeeld kunt u ook de Microsoft-syntaxis gebruiken om het type op te geven na de lengte-kwalificatie.
Bijvoorbeeld: `DIM s(5, 10) LENGTH 20 AS STRING`
Arrays kunnen ook worden geïnitialiseerd wanneer ze worden gedeclareerd door een gelijkteken (=) toe te voegen, gevolgd door een lijst met waarden tussen haakjes aan het einde van de declaratie.

Bijvoorbeeld:
```
DIM INTEGER nbr%(4) = (22, 44, 55, 66, 88) ' Integer is geheel getal
DIM s$(3) = ("foo", "boo", "doo", "zoo")
```

Merk op dat het aantal initialisatiewaarden moet overeenkomen met het aantal elementen in de array, inclusief de basiswaarde die is ingesteld door **OPTION BASE**.
Als een array met meerdere dimensies wordt geïnitialiseerd, wordt eerst de eerste dimensie geïnitialiseerd, gevolgd door de tweede, enz.
Merk ook op dat de initialisatiewaarden achter de **LENGTH kwalificatie** moeten staan (indien gebruikt) en na de typedeclaratie (indien gebruikt).

```
' Het toewijzen van een float aan een DIM ed integer variabele is OK en
' de float wordt geconver-teerd naar integer.
' De interne conversie van float naar Integer is gelijk aan CINT(float).
 DIM c%
 c% =  2.4 : PRINT c%, CINT( 2.4), INT( 2.4)
 c% =  2.4 : PRINT c%, CINT( 2.6), INT( 2.6)
 c% = -2.4 : PRINT c%, CINT(-2.4), INT(-2.4)
 c% = -2.6 : PRINT c%, CINT(-2.6), INT(-2.6)
' Het toewijzen van een strikt geheel getal aan een float variabele is OK
 DIM e!
 e! = &hFFFF: PRINT e!
 e! = &hFFFFFFFFFFFFFFFF: PRINT e!

CINT(getal)   ' Ronde getallen met breuken omhoog of omlaag naar het
              ' volgende geheel getal of geheel getal.

           Voorbeeld: 45.47 wordt afgerond op  45
                      45.57 wordt afgerond op  46
                     -34.45 wordt afgerond op -34
                     -34.55 wordt afgerond op -35

FIX(getal)    ' Kap een getal af tot een geheel getal door de komma te
              ' elimineren en alle tekens rechts van de komma.

           Voorbeeld:  9.89 retourneert  9
                      -2.11 retourneert -2
```

Het belangrijkste verschil tussen **FIX()** en **INT()** is dat **FIX()** een true integer functie (d.w.z. retourneert niet het eerstvolgende lagere getal voor negatief getallen zoals **INT()** doet).

```
INT(getal)    ' Een uitdrukking afkappen tot het volgende gehele getal
              ' kleiner dan of gelijk aan de argument.
           Voorbeeld:  9.89 retourneert  9
                      -2.11 retourneert -3
```

Dit gedrag is voor Microsoft-compatibiliteit, de **FIX()** functie biedt een echte **integer functie**.

LOCAL variabele [, variabelen]
Definieert een lijst met variabelenamen als lokaal voor de subroutine of functie.
Dit commando gebruikt exact dezelfde syntax als DIM en zal variabelen creëren die alleen zichtbaar zijn binnen de subroutine of functie.
Ze worden automatisch verwijderd wanneer de subroutine of functie wordt afgesloten.

Als een variabele wordt gedeclareerd als een parameter in een **SUB** of **FUNCTION** declaratie, mag deze **NIET** worden gedeclareerd als **LOCAL** of **STATIC** en zal een fout produceren als u dat doet.

```
FUNCTION epochX(dt$) AS INTEGER
LOCAL dt$
'''
Fout in regel 16: DT reeds gedeclareerd
```

```
STATIC variabele [, variabelen]
```

Definieert een lijst met variabelenamen die lokaal zijn voor de subroutine of functie.
Deze variabelen behouden hun waarde tussen aanroepen van de subroutine of functie (in tegenstelling tot variabelen die zijn gemaakt met het **LOCAL** commando).
Deze opdracht gebruikt exact dezelfde syntaxis als **DIM**.
Het enige verschil is dat de lengte van de variabelenaam gemaakt door **STATIC** en de lengte van de subroutine of functienaam bij elkaar opgeteld niet meer dan **32 tekens** mag bedragen.
Statische variabelen kunnen worden geïnitialiseerd op een waarde. Deze initialisatie wordt alleen van kracht bij de eerste oproep naar de subroutine (niet bij volgende oproepen).

Voorbeeld van het gebruik van **STATIC**

```
Print pseudo(1245) ' zaai de generator voor willekeurige getallen
For n = 1 To 10
  Print pseudo()
Next nr

Function pseudo(s als geheel getal) As Float ' Float is zwevende komma.
  Static Integer x = 7 ' Integer is geheel getal en Static max 32 tekens.
  Static Integer a = 1103515245
  Static Integer c = 12345
  Static Integer m = 2^31
  If s <> 0 Then x = s
  x = (a * x + c) Mod m
  pseudo = x / m
End Function
```

```
STRING$()
```

Syntaxis:
```
STRING$(aantal, ascii | string$) ' Nummers of tekenreeks in bereik van 32 - 126
```

Beschrijving:
Retourneert een tekenreeks **'nbr' bytes** lang bestaande uit ofwel het eerste teken van **tekenreeks$** of het teken dat de **ASCII**-waarde **'ascii'** vertegenwoordigt, wat een decimaal getal is in het bereik van **32 tot 126.**

```
  PRINT tekenreeks$(10,"Maximum")
```

Constanten

Een veelvoorkomende vereiste bij het programmeren is het definiëren van een identifier die een waarde vertegenwoordigt zonder de risico dat de waarde per ongeluk wordt gewijzigd - wat kan gebeuren als variabelen voor dit doel worden gebruikt.
Dit worden constanten genoemd en ze kunnen I/O-pinnummers, signaallimieten, wiskundige constanten enzovoort.

U kunt een constante maken met de opdracht **CONST**. Dit definieert een identifier die werkt als a variabele maar is ingesteld op een waarde die niet kan worden gewijzigd.

Als u bijvoorbeeld de spanning wilt controleren van een batterij die is aangesloten op **pin 31**, kunt u de batterij definiëren met relevante waarden dus:

```
CONST AccuspanningPin = 31
CONST BatterijMinimum = 1.5
```

Deze constanten kunnen vervolgens in het programma worden gebruikt, waar ze logischer zijn voor de gewone lezer dan eenvoudige getallen.

Bijvoorbeeld:

```
SETPIN BatteryVoltagePin, AIN
IF PIN(BatteryVoltagePin) < BatterijMinimum THEN GeluidAlarm
```

Het is een goede programmeerpraktijk om constanten te gebruiken voor elk vast getal dat een belangrijke waarde heeft. Normaal gesproken worden ze aan het begin van een programma gedefinieerd, waar ze gemakkelijk te zien en handig zijn geplaatst voor een andere programmeur om aan te passen (indien nodig).

Subroutines

Een subroutine is een blok programmeercode dat op zichzelf staat (zoals een module) en kan worden overal in uw programma aangeroepen. Voor jouw programma lijkt het op een ingebouwde MMBasic commando en kan hetzelfde worden gebruikt. Neem bijvoorbeeld aan dat u een commando nodig hebt dat signaleer een fout door een bericht op de console af te drukken. Je zou de subroutine als volgt kunnen definiëren:

```
SUB ErrMsg
  PRINT "Fout gedetecteerd"
END SUB
```

Met deze subroutine ingebed in je programma hoef je alleen maar het commando **ErrMsg** te gebruiken wanneer u het bericht wilt weergeven. Bijvoorbeeld:

```
IF A < B THEN ErrMsg
```

De definitie van een subroutine kan overal in het programma staan, maar staat meestal aan het einde. Als MMBasic komt de definitie tegen tijdens het uitvoeren van uw programma, het zal deze gewoon overslaan.

Het bovenstaande voorbeeld is goed genoeg, maar het zou beter zijn als er een nuttiger bericht zou kunnen worden weergegeven, een die kon worden aangepast elke keer dat de subroutine werd aangeroepen. Dit kan door een string toe aan de subroutine als argument (soms een parameter ge-noemd).

In dit geval ziet de definitie van de subroutine er als volgt uit:

```
SUB ErrMsg Msg$
  PRINT "Fout: " + Msg$
END SUB
```

Wanneer u vervolgens de subroutine oproept, kunt u de tekenreeks opgeven die moet worden afge-drukt op de opdrachtregel van de subroutine. Bijvoorbeeld:

```
IF A < B THEN ErrMsg "Nummer te klein"
```

Als de subroutine zo wordt aangeroepen, wordt de melding **"Fout: Nummer te klein"** weergegeven afgedrukt op de console. Binnen de subroutine heeft **Msg$** de waarde **"Nummer te klein"** wanneer zo genoemd en het zal worden samengevoegd in de **PRINT** instructie om het volledige foutbericht te maken.

Een subroutine kan een willekeurig aantal argumenten hebben die float, integer of string kunnen zijn met elk argument gescheiden door een komma. Binnen de subroutine gedragen de argumenten zich als gewone variabelen, maar ze bestaan alleen binnen de subroutine en zullen verdwijnen wanneer de subroutine eindigt. Je kan hebben variabelen met dezelfde naam in het hoofdprogramma en ze worden verborgen in de subroutine en verschillen van de argumenten die voor de subroutine zijn gedefinieerd.

Het type argument dat moet worden opgegeven, kan worden opgegeven met een achtervoegsel van het type (dwz **$, %** of **! Voor letters$ = String**, **Geheel getal% = Integer** en **Zwevend!** = **Fload**). In het volgende moet het eerste argument bijvoorbeeld een string zijn en het tweede een geheel getal:

```
SUB MySub Msg$, Nbr%
  ...
END SUB
```

MMBasic zal de opgegeven waarden converteren als dit mogelijk is, dus als uw programma een drijvende-kommawaarde heeft opgegeven als tweede argument zal MMBasic het naar een geheel getal converteren. Als MMBasic de waarde niet kan converteren het geeft een fout weer en keert terug naar de opdrachtprompt. Als u bijvoorbeeld een tekenreeks hebt opgegeven voor het tweede argument dat uw programma zal stoppen met een fout.

U hoeft de type-achtervoegsels niet te gebruiken, u kunt in plaats daarvan het type van de argumen-ten definiëren met behulp van de AS-sleutelwoord vergelijkbaar met de manier waarop het wordt gebruikt in de **DIM** opdracht.

Het volgende is bijvoorbeeld identiek aan het bovenstaande voorbeeld:

```
SUB MySub Msg AS STRING, Nbr AS INTEGER
  ...
END SUB
```

Natuurlijk, als je maar één type variabele in het hele programma hebt gebruikt en **OPTION DEFAULT** hebt gebruikt om dat type in te stellen, zou je de kwestie van variabele typen volledig kunnen negeren.

Wanneer een subroutine wordt aangeroepen met een argument dat een variabele is (dus geen constante of uitdrukking).
MMBasic zal een corresponderende variabele creëren binnen de subroutine die terugverwijst naar deze variabele.
Elke wijziging aan de variabele die het argument in de subroutine vertegenwoordigt, zal ook de variabele die in de oproep wordt gebruikt. Dit wordt het doorgeven van argumenten door middel van verwijzing genoemd.

Dit kan het beste worden uitgelegd aan de hand van een voorbeeld:

```
DIM MyNumber = 5            ' Zet de variabele op 5.
CalcSquare MyNumber         ' De subroutine zal zijn waarde kwadrateren.
PRINT MyNumber              ' Dit zal het nummer 25 afdrukken.
END

SUB CalcSquare nbr
  nbr = nbr * nbr           ' Vierkant het argument en geef het terug.
END SUB
```

De subroutine CalcSquare neemt zijn argument, kwadrateert het en schrijft het terug naar de variabele die het argument vertegenwoordigen **(nbr)**. Omdat de subroutine werd aangeroepen met een variabele (**MyNumber**) de variabele **nbr** zal terugverwijzen naar **MyNumber** en elke wijziging in **nbr** zal ook **MyNumber** veranderen overeenkomstig. Als resultaat zal de **PRINT** instructie 25 uitvoeren.
Argumenten doorgeven door middel van verwijzing is handig omdat het een subroutine in staat stelt waarden terug te geven aan de code die het noemde. Het kan echter tot problemen leiden als een subroutine de variabele gebruikt die een voorstelt argument als een variabele voor algemene doel-einden en heeft de waarde ervan gewijzigd. Als het dan zou worden aangeroepen met een variabele als een argument, zou die variabele per ongeluk worden gewijzigd. Om deze reden moet je ver-mijden variabelen manipuleren die argumenten binnen een subroutine vertegenwoordigen, in plaats daarvan de waarde toewijzen aan een lokale variabele (zie hieronder) en manipuleer die in plaats daarvan.
Wanneer u een subroutine aanroept, kunt u enkele (of alle) parameters weglaten en deze zullen de waarde aannemen van nul (voor floats of integers) of een lege string. Dit is handig omdat uw subroutine kan zien of een parameter ontbreekt en handel dienovereenkomstig.

Hier is bijvoorbeeld onze subroutine om een foutmelding te genereren, maar deze versie kan worden gebruikt zonder een foutmelding als parameter specificeren:

```
SUB ErrMsg Msg$
  IF Msg$ = "" THEN
    PRINT "Fout gedetecteerd"
  ELSE
    PRINT "Fout: " + Msg$
  ENDIF
END SUB
```

Binnen een subroutine kunt u de meeste functies van MMBasic gebruiken, inclusief het aanroepen van andere subroutines, **IF...THEN** commando's, **FOR...NEXT** lussen enzovoort. Een ding dat u echter niet kunt doen, is: spring uit een subroutine met behulp van **GOTO** (als u dit doet, is het resultaat niet gedefinieerd). Normaal gesproken de subroutine wordt afgesloten wanneer het **END SUB** commando is bereikt, maar u kunt ook de beëindigen subroutine vroeg door het **EXIT SUB** commando te gebruiken.

Functies

Functies zijn vergelijkbaar met subroutines, met als belangrijkste verschil dat een functie wordt gebruikt om een waarde terug te geven in een uitdrukking. Als u bijvoorbeeld een functie wilt om een temperatuur om te zetten in graden Celsius naar Fahrenheit zou je kunnen definiëren:

```
FUNCTION Fahrenheit(C)
  Fahrenheit = C * 1.8 + 32
END FUNCTION
```

Dan zou je het in een expressie kunnen gebruiken:

```
Input "Voer een temperatuur in Celsius in: ", t
PRINT "Dat is hetzelfde als" Fahrenheit(t) "F"
```

Of als een ander voorbeeld:

```
IF Fahrenheit(temp) <= 32 THEN PRINT "Bevriezen"
```

Je zou ook het omgekeerde kunnen definiëren:

```
FUNCTION Celsius(F)
  Celsius = (F - 32) * 0.5556
END FUNCTION
```

Zoals je kunt zien, wordt de functienaam gebruikt als een gewone lokale variabele binnen de subroutine. Het is alleen wanneer de functie retourneert dat de waarde beschikbaar is gemaakt voor de expressie die deze heeft aangeroepen.

De regels voor de lijst met argumenten in een functie zijn vergelijkbaar met subroutines. Het enige verschil is dat haakjes zijn altijd vereist rond de lijst met argumenten wanneer u een functie aanroept, zelfs als er zijn geen argumenten (haakjes zijn optioneel bij het aanroepen van een subroutine).

Om een waarde uit de functie terug te geven, wijst u een waarde toe aan de naam van de functie binnen de functie. Als de naam van de functie wordt afgesloten met een achtervoegsel van het type (dwz **$,** een **%** of een **!**) de functie zal dat type teruggeven (string, integer of float), anders retourneert het ongeacht de **OPTION DEFAULT** is ingesteld. Voor voorbeeld, de volgende functie retourneert een tekenreeks:

```
FUNCTION LVal$(nbr)
  IF nbr = 0 THEN LVal$ = "False" ELSE LVal$ = "True"
END FUNCTION
```

U kunt het type functie expliciet specificeren door het **AS** sleutelwoord te gebruiken en dan hoeft u niet om een typeachtervoegsel te gebruiken (vergelijkbaar met het definiëren van een variabele met **DIM**).

Dit is het bovenstaande voorbeeld herschreven om te profiteren van deze functie:

```
FUNCTION LVal(nbr) AS STRING
  IF nbr = 0 THEN LVal = "False" ELSE LVal = "True"
END FUNCTION
```

In dit geval zal het type dat wordt geretourneerd door de functie **LVal** een string zijn.

Wat betreft subroutines kun je de meeste functies van MMBasic gebruiken binnen functies. Dit bevat **FOR…NEXT** lussen, andere functies en subroutines aanroepen, enz. De functie keert ook terug naar de expressie die het aanriep toen het **END FUNCTION** commando is bereikt, maar je kunt ook vroeg terugkeren door het **EXIT FUNCTION** commando te gebruiken.

Lokale variabelen

Variabelen die zijn gemaakt met **DIM** of die gewoon automatisch worden gemaakt, worden globale variabelen genoemd. Dit betekent dat ze overal in het programma kunnen worden bekeken en gebruikt, ook binnen subroutines en functies. Binnen een subroutine of functie moet u echter vaak variabelen gebruiken voor verschillende taken die intern zijn voor de subroutine/functie. In draagbare code wil je niet de naam die je hebt ervoor gekozen om zo'n variabele te laten botsen met een globale variabele met dezelfde naam. Hiertoe kunt u definiëren: een variabele die het **LOCAL** commando gebruikt binnen de subroutine/functie.

De syntaxis voor **LOCAL** is identiek aan het **DIM** commando, dit betekent dat de variabele een array kan zijn, u kunt het type van de variabele instellen en u kunt deze initialiseren op een bepaalde waarde.

Dit is bijvoorbeeld onze **ErrMsg** subroutine, maar deze keer is deze uitgebreid om een lokale variabele te gebruiken voor het samenvoegen van de foutbericht string.

```
SUB ErrMsg Msg$
  LOCAL STRING tstr
  tstr = "Fout: " + Msg$
  PRINT tstr
END SUB
```

De variabele **tstr** wordt gedeclareerd als **LOCAL** binnen de subroutine, wat betekent dat (zoals het argument lijst) zal het alleen binnen de subroutine bestaan en zal het verdwijnen wanneer de subroutine wordt afgesloten. Je kunt een globale variabele genaamd **tstr** in je hoofdprogramma en het zal anders zijn dan de variabele **tstr** in de subroutine (in dit geval wordt de globale **tstr** verborgen in de subroutine).

U moet altijd lokale variabelen gebruiken voor bewerkingen binnen uw subroutine of functie, omdat ze helpen de module veel meer op zichzelf staand en draagbaar te maken.

Statische variabelen

LOCAL variabelen worden teruggezet naar hun beginwaarden (normaal nul of een lege tekenreeks) telkens wanneer de subroutine of functie start, maar er zijn momenten waarop u wilt dat de variabele zijn waarde behoudt tussen de gesprekken. Dit type variabele wordt gedefinieerd met het **STATIC** commando.

We kunnen aantonen hoe **STATIC** variabelen nuttig zijn door de **ErrMsg** subroutine uit te breiden om te voorkomen dat dubbele oproepen naar de subroutine die herhaaldelijk hetzelfde bericht weer-geven. Bijvoorbeeld ons programma kan deze subroutine vanuit meerdere plaatsen aanroepen, maar als het bericht hetzelfde is in een aantal volgende oproepen willen we het bericht graag een keer zien. Dit is onze nieuwe subroutine:

```
SUB ErrMsg Msg$
  STATIC STRING lastmsg
  LOCAL STRING tstr
  IF Msg$ <> lastmsg THEN
    tstr = "Fout: " + Msg$
    PRINT tstr
    lastmsg = Msg$
  END IF
END SUB
```

Om het laatst weergegeven bericht bij te houden, gebruiken we een statische variabele genaamd **lastmsg**. Dit houdt stand de tekst van het laatste bericht en we kunnen het vergelijken met de huidige berichttekst om te bepalen of dit het geval is verschillend en moet daarom worden afgedrukt. Dit zou slechts één bericht geven elke keer dat er wordt gebeld met een dubbele berichttekst.

De opdracht **STATIC** gebruikt exact dezelfde syntaxis als **DIM**. Dit betekent dat u kunt definiëren dat verschillende soorten statische variabelen, inclusief arrays, en u kunt ze ook initialiseren naar een bepaalde waarde.

De statische variabele wordt gemaakt bij de eerste keer dat het **STATIC** commando wordt aangetroffen en het is automatisch ingesteld op nul (als een float of integer) of een lege string. Bij volgen-de oproepen naar de subroutine of functie MMBasic zal herkennen dat de variabele al is gemaakt en het zal laat de waarde onaangeroerd (dat wil zeggen, wat het ook was in de vorige oproep). Net als bij **DIM** kunt u ook initialiseer een statische variabele naar een bepaalde waarde. Bijvoorbeeld:

```
STATIC INTEGER var = 123
```

Bij de eerste aanroep (wanneer de variabele wordt gemaakt) wordt deze geïnitialiseerd op **123**, maar bij volgende oproepen wordt deze behouden waarmee de waarde eerder was ingesteld.

Meestal worden statische variabelen gebruikt om de toestand van iets bij te houden terwijl het zich in een subroutine bevindt of functie. Een staat is een verslag van iets dat eerder is gebeurd.

Voorbeelden zijn:

- Is de COM-poort al geopend?
- Welke stappen in een reeks hebben we voltooid?
- Welke tekst is er al weergegeven?

Normaal gesproken gebruik je globale variabelen (gemaakt met **DIM**) om een status bij te houden, maar soms wil je dat dit in een module moet worden opgenomen en dit is waar statische variabelen waardevol zijn. Net als **LOCAL** het gebruik van **STATIC** helpt om uw subroutines en functies meer op zichzelf staand en draagbaar te maken.

Bereken dagen

We hebben tot nu toe veel programmeeropdrachten en -technieken behandeld in deze handleiding en voordat we het zou de moeite waard zijn om een voorbeeld te geven van hoe ze samenwerken. Het volgende is een voorbeeld die veel functies van de BASIC-taal gebruikt om het aantal dagen tussen twee datums te berekenen:

```
  'Voorbeeldprogramma om het aantal dagen tussen twee datums te berekenen.
  OPTION EXPLICIT
  OPTION DEFAULT NONE

  DIM STRING s
  DIM FLOAT d1, d2

  DO
  ' Hoofdprogramma lus
    PRINT : PRINT "Vul de datum in als dd mnd jjjj"
    PRINT "Eerste datum";
    INPUT s
    d1 = GetDays(s)
    IF d1 = 0 THEN PRINT "Ongeldige datum!" : CONTINUE DO
    PRINT "Tweede datum";
    INPUT s
    d2 = GetDays(s)
    IF d2 = 0 THEN PRINT "Ongeldige datum!" : CONTINUE DO
    PRINT "Verschil is" ABS(d2 - d1) " dagen"
  LOOP

  ' Bereken het aantal dagen sinds 1/1/1900.
  FUNCTION GetDays(d$) AS FLOAT
    LOCAL STRING Month(11) = ("jan","feb","maart","apr","mei","jun","jul",
"aug","sep","okt","nov","dec")
    LOCAL FLOAT Days(11) = (0,31,59,90,120,151,181,212,243,273,304,334)
    LOCAL FLOAT day, mth, yr, s1, s2

   ' Zoek het scheidingsteken in een datum.
    s1 = INSTR(d$, " ")
    IF s1 = 0 THEN EXIT FUNCTION
    s2 = INSTR(s1 + 1, d$, " ")
    IF s2 = 0 THEN EXIT FUNCTION
   ' Verkrijg de dag, maand en jaar als cijfers.
    day = VAL(MID$(d$, 1, s2 - 1)) - 1
    IF day < 0 OR day > 30 THEN EXIT FUNCTION
    FOR mth = 0 TO 11
      IF LCASE$(MID$(d$, s1 + 1, 3)) = Month(mth) THEN EXIT FOR
    NEXT mth
    IF mth > 11 THEN EXIT FUNCTION
    yr = VAL(MID$(d$, s2 + 1)) - 1900
    IF yr < 1 OR yr >= 200 THEN EXIT FUNCTION
  ' Bereken het aantal dagen inclusief correctie voor schrikkeljaren.
    GetDays = (yr * 365) + FIX((yr - 1) / 4)
    IF yr MOD 4 = 0 AND mth >= 2 THEN GetDays = GetDays + 1
    GetDays = GetDays + Days(mth) + day
  END FUNCTION
```

Houd er rekening mee dat de regel die begint met **LOCAL STRING Month(11)** omwikkeld is vanwege de beperkte paginabreedte. Het is één regel als volgt:

```
  LOCAL STRING Month(11) = ("jan","feb","mar","apr","may","jun","jul","aug",
"sep","oct","nov","dec")
```

Dit programma werkt door twee datums van de gebruiker op de console te krijgen en deze vervolgens te converteren naar gehele getallen die het aantal dagen sinds 1900 vertegenwoordigen. Met deze twee getallen een eenvoudige aftrekking geeft het aantal dagen ertussen.

Wanneer dit programma wordt uitgevoerd, zal het vragen om de twee datums in te voeren en moet u het formulier gebruiken van: **dd mmm jjjj**.

Deze schermopname laat zien wat de lopend programma eruit zal zien.

Het belangrijkste kenmerk van het programma is de door de gebruiker gedefinieerde functie **GetDays()** die een string nodig heeft die is ingevoerd bij de console, splitst het in een dag, maand en jaarcomponenten berekent vervolgens het aantal dagen sinds 1 januari 1900.

```
> run

Vul de datum in als dd mnd jjjj
Eerste datum? 10 nov 1959
Tweede datum? 21 feb 2022
Verschil is 22749 dagen

Vul de datum in als dd mnd jjjj
Eerste datum?
```

Deze functie wordt twee keer aangeroepen, één keer voor de eerste date en dan weer voor de tweede datum. Het is dan gewoon een kwestie van één datum (in dagen) aftrekken van de andere om het verschil in dagen te krijgen.

We zullen niet ingaan op de details van hoe de berekeningen worden gemaakt (dwz het omgaan met schrikkeljaren), want dat kan zijn achtergelaten als oefening voor de lezer. Het is echter gepast om te wijzen op enkele kenmerken van MMBasic die door het programma worden gebruikt.

Het laat zien hoe lokale variabelen kunnen worden gebruikt en hoe ze kunnen worden geïnitialiseerd. In de functie **GetDays()** twee arrays worden tegelijkertijd gedeclareerd en geïnitialiseerd. Deze zijn gewoon een handig methode om de namen van de maanden en het cumulatieve aantal dagen voor elke maand op te zoeken.
Later in de functie (de **FOR LOOP**) kun je zien hoe ze omgaan met twaalf verschillende maanden behoorlijk efficiënt.

Een ander kenmerk dat door dit programma wordt benadrukt, zijn de stringverwerkingsfuncties van MMBasic. De functie **INSTR()** wordt gebruikt om de twee spatietekens in de datumreeks te loka-liseren en later de functie **MID$()** gebruikt deze om de dag-, maand- en jaarcomponenten van de datum te extraheren. De **VAL()** functie wordt gebruikt om een reeks cijfers (zoals het jaar) om te zetten in een getal dat kan worden opgeslagen in een numeriek variabel.

Merk op dat de waarde van een functie elke keer dat de functie wordt uitgevoerd wordt geïnitia-liseerd op nul en tenzij deze is ingesteld naar een bepaalde waarde zal het een nulwaarde terug-geven. Dit maakt het afhandelen van fouten eenvoudig omdat we gewoon de functie kunnen verlaten als er een fout wordt ontdekt. Het is dan de verantwoordelijkheid van de aanroepende programmacode om te controleren op een retourwaarde van nul die een fout aangeeft.

Dit programma illustreert een van de voordelen van het gebruik van subroutines en functies, name-lijk dat wanneer geschreven en volledig getest kunnen ze worden behandeld als een vertrouwde "zwarte doos" die niet geopend hoeft te worden.
Om deze reden moeten functies als deze goed worden getest en vervolgens, indien mogelijk, onaan-geroerd worden gelaten (in als u een fout toevoegt).

Er zijn een paar functies van dit programma die we nog niet eerder hebben behandeld. De eerste is de **MOD** operator die de rest van het delen van het ene getal in het andere berekent. Als je bijvoor-beeld **4/15** zou je een rest van **3** hebben, wat precies is wat de uitdrukking **15 MOD 4** zal terug-keren. De functie **ABS()** is ook nieuw en retourneert het argument als een positief getal. (bijv. **ABS(-15)** zal **+15** retourneren, evenals **ABS(15)**).

Het **EXIT FOR** commando verlaat een **FOR LOOP**, ook al heeft het het einde van zijn lus nog niet bereikt, **EXIT FUNCTION** zal een functie onmiddellijk verlaten, ook al heeft de uitvoering het einde van nog niet bereikt de functie en **CONTINUE DO** zullen er onmiddellijk voor zorgen dat het prog-ramma naar het einde van een **DO LOOP** springt en voer het opnieuw uit.

Waarom zou dit programma nuttig zijn? Nou, sommige mensen houden ervan om hun leeftijd in dagen te tellen, op die manier elke dag is een verjaardag! U kunt uw leeftijd in dagen berekenen, voer gewoon de datum in waarop u bent geboren en de datum van vandaag. Dat is niet bijzonder nuttig, maar het programma zelf is waardevol omdat het veel demonstreert van de kenmerken van programmeren in MMBasic. Dus, werk je een weg door het programma en bekijk elke sectie totdat u het begrijpt - het zou een lonende ervaring moeten zijn.

Enkele Hints & Tips voor het schrijven van efficiënte code.

Gewijzigd op 2021/12/12 10:24 door CaptainBoing Gecategoriseerd als programmastroom, programmabeheer. » Enkele Hints & Tips voor het schrijven van efficiënte code. **(MMBasic)**

Hoewel er een verdienste is in het argument dat als je snelheid nodig hebt, geen geïnterpreteerde taal moet gebruiken, is elke code die je schrijft bedoeld om redelijk lang mee te gaan (dat wil zeggen dat je hem daadwerkelijk opslaat met de bedoeling hem ooit weer te gebruiken) moet worden aangescherpt om efficiënt te zijn.

Veel van wat volgt is idiosyncratisch, vrij zwaar pedant en niet erg belangrijk, maar het krabt mijn efficiëntie jeuk. Ik kom uit een Z80/68000-achtergrond en oudere CPU's hadden weinig geheugen en kracht en het leerde me zuinig te zijn. Niet veel van het volgende is echt van toepassing op gecompileerde code en moderne CPU's, maar er zijn elementen van "sanitaire" code ontwikkeling waardoor je je overal warm en wazig voelt. Het is vermeldenswaard dat moderne code lijdt aan een fenomeen dat in de volksmond bekend staat als "bloat" - dit is grotendeels het effect van ontwikkelaars die niet compact (of zelfs slordig) schrijven omdat ze zich er geen zorgen over hoeven te maken in deze dagen van immens snelle multi-core CPU's en gigabytes aan geheugen. Ben ik te streng voor ze? Misschien. In de dagen van tragere systemen kon je iets schrijven en zien dat het gewoon niet t goed genoeg en dus zou je er meerdere keren naar moeten streven om het elke keer beter te maken (hopelijk). Moderne machines zijn zo snel dat zelfs afval in een oogwenk doorloopt, dus misschien krijgen ontwikkelaars niet die schop in hun broek als iets dat ze hebben geschreven duidelijk werkt als een driepotige ezel - de impuls om uitstekende code te schrijven is er nu gewoon niet. Goed voor de accountants. Persoonlijk krijg ik een kick van het maken van code zo goed als ik denk dat ik het kan krijgen, misschien maanden later opnieuw naar code te gaan, alleen voor de mentale oefening... Oude gewoonten en zo. geen schop onder je broek krijgen als iets dat ze hebben geschreven duidelijk loopt als een driepotige ezel - de impuls om uitstekende code te schrijven is er nu gewoon niet. Goed voor de accountants. Persoonlijk krijg ik een kick van het maken van code zo goed als ik denk dat ik het kan krijgen, misschien maanden later opnieuw naar code te gaan, alleen voor de mentale oefening... Oude gewoonten en zo. geen schop onder je broek krijgen als iets dat ze hebben geschreven duidelijk loopt als een driepotige ezel - de impuls om uitstekende code te schrijven is er nu gewoon niet. Goed voor de accountants. Persoonlijk krijg ik een kick van het maken van code zo goed als ik denk dat ik het kan krijgen, misschien maanden later opnieuw naar code te gaan, alleen voor de mentale oefening... Oude gewoonten en zo.

Er zijn tal van methoden - sommige staan open voor discussie, die kunnen helpen. Deze variëren van eenvoudig tot ronduit cryptisch, en het is geen nieuw concept. Eenvoudige aanpassingen in de manier waarop je dingen doet, kunnen de cyclustijden voor je code behoorlijk verkorten - en dat zal het overkoepelende thema van deze hints zijn; om code zo snel en compact mogelijk te krijgen, rekening houdend met het feit dat klein niet altijd snel is... daar komen we op terug.

Dit artikel is gericht op het MicroMite-assortiment microcontrollers, maar de ideeën kunnen volledig worden gemigreerd naar alle MMBasic-platforms en daarbuiten. Op de kleinere micromieten kunnen sommige taken een grote vraag zijn voor een tolk op 48MHz (of langzamer). Sommige van de volgende tips hebben een RC4-coderingsalgoritme bruikbaar gemaakt - het was moeilijk om te doen un-encrypt >> process >> encryptin minder dan 1,5 seconden. Het is nog steeds strak voor lange snaren, maar het is nu bruikbaar. Dus deze tips kunnen helpen en zijn in sommige gevallen zelfs van vitaal belang gebleken - doe niet te lichtvaardig af als louter pedanterie.

Dit is geen uitputtende lijst; meer een verzameling hints en suggesties.

"Genees uzelf, dokter"

Veel voorbeelden van mijn eigen code die in deze bibliotheek zijn gepubliceerd, voldoen mogelijk niet al te nauw aan deze richtlijnen. Dit heeft waarschijnlijk twee belangrijke redenen:

Deze gewoontes zijn geëvolueerd sinds ik begon met publiceren en maken deel uit van mijn eigen ontwikkeling.

Ik heb de code niet te veel aangescherpt om het gemakkelijk begrijpelijk te houden.

Vaak als ik de code in woede gebruik, zal deze slechts een vluchtige gelijkenis vertonen met een van de artikelen in een wiki-artikel hier.

Bespreek, bekritiseer, draag bij.
Houd uw variabelenamen relevant en pittig. Zelfdocumenterende code wordt enorm geholpen door variabelenamen te hebben die beschrijven waarvoor ze worden gebruikt en het is een geweldig idee ... maar ... het kan te veel worden gebruikt, niemand zal het erg vinden als je belt een teller voor algemeen gebruik **"x"**. Stel je de volgende situatie voor:

```
For MyBigLoopCountVariable = 1 To 10
  Print MyBigLoopCountVariable
Next
```

Verbetert de duidelijkheid niet

```
For x = 1 To 10
  Print x
Next
```

Om nog maar te zwijgen over de kleinere codevoetafdruk.
Korte variabelenamen zijn prima waar van toepassing (en een klein beetje sneller om te ontleden, dus zal een paar mS scheren in een strakke lus met duizenden iteraties), maar het volgende schreeuwt om goede variabelenamen.

```
a(m, j) = c / d + 1
```

Ziet er een stuk beter uit als:

```
ColorMap(current, pointer) = OldColVal / daynum + 1
```

Je hebt een goede kans om te begrijpen wat die regel doet door er alleen maar naar te kijken.

Enig inzicht in de namen van variabelen

In strakke/repetitieve loops waar timing belangrijk is, helpt het om te begrijpen hoe MMBasic variabelen definieert en benadert - het kan helpen om elke keer een paar microseconden te besparen: MMBasic slaat alle variabelen op "de hoop" op zoals ze zijn gedefinieerd. Er is geen ordening of gereserveerde variabele ruimte - elke keer dat u deze gebruikt, moet er naar gezocht worden (BBC Basic had 26 gereserveerde spaties voor de 26 integer-variabelen a-zwat betekende dat het niet naar hen hoefde te zoeken, het "instinctief wist" waar elk was - dit was een slimme versneller voor de tolk). Met MMBasic kun je het interne mechanisme een beetje gebruiken door veelgebruikte variabelen te definiëren vóór obscure / zelden gebruikte variabelen, de meest gebruikte eerst, in de volgorde waarin ze worden gebruikt als je kunt. Als je bijvoorbeeld een variabele hebt ndie overal in je code in tellussen wordt gebruikt, definieer deze dan vroeg, zodat de interpreter het niet moeilijk heeft om deze te vinden wanneer dat nodig is. Omgekeerd, een variabele die je twee keer gebruikt in een stukje code dat eens in de 10 minuten wordt uitgevoerd, definieer het als laatste, zodat MMBasic het niet hoeft over te slaan bij het zoeken naar naltijd. MMBasic ontleedt variabelenamen 4 tekens tegelijk, dus er is geen reden om willekeurig variabelenamen te verminderen, tenzij u dat wilt (om coderuimte te besparen?), bijvoorbeeld om te gebruiken tin plaats van trev, maar het is een grote reden om het niet te gebruiken trevor(het heeft twee rondes nodig van parsing), tenzij de langere naam iets toevoegt aan uw code.

Bereken dingen niet twee keer - gebruik een variabele

Als u de processor wat werk laat doen dat u later nodig zult hebben, sla dan het resultaat op en herhaal de inspanning niet. Het volgende dwingt het snijden van strings en evaluatie twee keer uit te voeren.

```
If Val(Mid$(a$, 2, 2)) = 17 Then
  Print "The value is "; Val(Mid$(a$, 2, 2)); " in that position "
EndIf
```

Variabelen zijn veel sneller dan evaluaties (ish... zie hieronder) Beter om te hebben

```
x = Val(Mid$(a$, 2, 2))
If x = 17 Then
  Print "The value is "; x; " in that position "
EndIf
```

Wacht! Gebruik geen variabele
Beschouw dit in een Sub of Functie:

```
Local Integer x
x = t << b
```

... en dan x gebruiken op de plaatsen waar je de berekening had, is zeker langzamer dan gewoon gebruiken:

```
t << b
```

al was het maar een paar keer in de onderwerpcode omdat de overhead van variabelenbeheer niet echt goed afsteekt tegen de snelheid van eenvoudige integere wiskunde die een of twee keer wordt gedaan.

Het punt dat ik probeer over te brengen is; kweek de mentaliteit om met de opties te spelen en te zien welke het meeste voordeel oplevert, accepteer niet noodzakelijk het eerste waar je aan denkt voor meer dan 'snel krabbelen'-code. Bekijk de nieuwste versie van de Bit Manipulation Functions om te zien wat ik hier bedoel.

Over Absoluten en CONST's

Als we het toch over variabelen hebben, vergeet ook CONSTants niet - ze worden op dezelfde manier behandeld als variabelen, maar u kunt de inhoud natuurlijk niet wijzigen. Het kost veel minder moeite om een waarde uit een CONST op te zoeken dan voor de parser om het uit je code te halen, dus probeer geen absolute waarden te gebruiken als je dit kunt vermijden. Nog een raaklijn! Houd bij het specificeren van absolute waarden rekening met de parser... Wij mensen werken heel goed in base 10, maar processors niet. Elke absolute waarde moet elke keer dat deze wordt aangetroffen, worden omgezet in CPU-binair. Kijk eens naar deze domme code:

```
Dim Integer n
Timer = 0
Do
  n = n + 1
  If n = 100000, Then Exit Do
Loop
Print Timer
```

Overweeg nu waar n wordt opgehoogd. 1 is hetzelfde in alle basen, maar je kunt een klontje van de toename afschaven door een computervriendelijke basis op te geven (zelfs gezien de extra twee bytes die worden geïnterpreteerd). Hier zijn enkele timings voor verschillende basen in dat ene kleine regeltje:

```
18268mS met n = n + 1      ' Decimaal
17884mS met n = n + &h1    ' Hexadicimaal
17841mS met n = n + &b1    ' Binair
17834mS met n = n + &o1    ' Octaal
```

al het bovenstaande betekent hetzelfde, maar je kunt bijna een seconde van die lus scheren met een radix (waarom octaal marginaal de snelste is, kan ik niet zeggen, maar je begrijpt wat ik bedoel).

Dit is echter niet het einde; die vergelijkingsregel verdoezelt ook de parser. In plaats van een absoluut (zelfs met een radix) te gebruiken, hoe zit het met het gebruik van een **CONST** voor de vergelijking? Ze zijn al binair opgeslagen en zijn veel sneller te gebruiken. Terugkerend naar ons **n = n + 1** ergste geval hierboven, door een **CONST** te gebruiken, scheren we bijna 5 seconden van dezelfde lus! Slechts **13839 mS** met **CONST c = 100000** en later **if n =** congeveer 25% sneller. Je zou nog verder kunnen gaan en een andere **CONST** gebruiken voor de incrementwaarde - ik laat je daarmee experimenteren.

Samengevat:Probeer geen absolute waarden te gebruiken - gebruik CONST's tenzij je een goede reden hebt en als je absolute waarden wilt/moet gebruiken, overweeg dan om ze in verschillende radices te specificeren om waardevolle mS af te scheren. De sleutel is balans: al het bovenstaande is grotendeels zinloos buiten stukjes code die keer op keer worden uitgevoerd (of worden gebeld). En het heeft niet veel zin om een radix te gebruiken voor de CONST (tenzij het iets toevoegt). Het wordt maar één keer uitgevoerd in je code en ik betwijfel of je een paar mS zult opmerken die tijdens het opstarten worden opgeslagen.

Meerdere tests in IF statements

Het If-statement is misschien wel de meest gebruikte constructie. Het neemt een beslissing en zorgt ervoor dat code er intelligent uitziet door te reageren op verschillende situaties (de veel geroemde AI is grotendeels slechts een massa door machines gegenereerde **IF** statements). De verklaring evalueert een argument (slechts wat gegevens doorgegeven aan iets anders) en komt neer op een **1** of **0**... ja of nee. Vaak worden de **AND**- en **OR**-operators gebruikt om de tests dieper te maken, maar ze kunnen een eenvoudige test bemoeilijken en de besluitvorming vertragen. Probeer de argumen-ten "bewust" te maken van waar ze naartoe werken. Beschouw het voorbeeld dat we hebben om de toestand van een schakelaar te testen, maar alleen als het overdag is. We zouden iets kunnen schrijven als:

```
If DayTime = 1 And SwitchPressed = 1 Then...
```

maar onthoud dat het hele argument moet worden geëvalueerd. De schakelaar wordt getest en er wordt gekeken of het dag is. In werkelijkheid, als het geen dag is, geven we niets om de schakelaar, zodat je de parser kunt helpen door dingen op te splitsen. Beschouwen:

```
If DayTime Then
  If SwitchBusy Then
```

Ja, het kost meer ruimte in je code, maar het zal iets sneller zijn. Als dit honderden keren per minuut wordt uitgevoerd, tellen al die besparingen op. Met deze methode kan de parser het hele testen van de schakelstatus onmiddellijk overslaan als het nacht is. Merk ook op dat we niet hoeven te testen voor =1. De parser is een vertrouwende ziel en hij gelooft dat alles waar is, tenzij je iets anders zegt, dus **" If DayTime Then"** is functioneel identiek aan **" If DayTime <> 0 Then"**, maar een beetje sneller omdat er geen vergelijking (**met 1**) wordt gemaakt. Vermeldenswaard is dat elk getal behal-ve nul waar is, dus je kunt deze truc gebruiken voor elke voorwaarde die niet nul is.

Terwijl we toch bezig zijn, doe de snelste, eenvoudigste, meest onwaarschijnlijke dingen eerst. Wat kost meer tijd om uit te voeren? Wat is het minst waarschijnlijke dat er gebeurt dat de rest van de test snel kan diskwalificeren? enz...

Even terzijde, dit laatste aspect is het onderwerp van een hele IT-inspanning bij het stroomlijnen van databases; Overweeg een hypothetische database met wereldbevolking met de namen en natio-naliteiten van alle mensen erin (enorme veronderstellingen gemaakt om de cijfers eenvoudig te houden). Stel nu dat we een lijst willen van alle Chinese mannen. Welke van de twee onderstaande "query's" is het beste? Met beter bedoel ik, snelste - de laagste "kosten" voor de database-engine.

```
Select NAME From POPULATION Where SEX = 'M' And NATIONALITY = 'CH';
```

of

```
Select NAME From POPULATION Where NATIONALITY = 'CH' And SEX = 'M';
```

Beide resultaten zijn hetzelfde, maar de manier waarop de meeste databases werken, is door sets te verkleinen. Het begint met een groot gegevensblok, verwijdert vervolgens de dingen waar het niet in geïnteresseerd is om te resulteren in een ander blok ... enzovoort, totdat je eindigt met de subset waarin je geïnteresseerd bent. De eerste doorzoekt onze DB van 7 Miljard records voor alle mannen (we nemen aan dat het 50% is) en moet dan 3,5 miljard records doorzoeken om de Chinese leden te bepalen. 10,5 miljard records onderzocht.

Het tweede voorbeeld doorzoekt 7 miljard records, dat is waar, maar daarna hoeft er nog maar 1 miljard records te worden doorzocht - in totaal 8 miljard records. Het eerste deel van de zoektocht is onvermijdelijk, maar het tweede deel van 70% van het werk - gewoon door na te denken over de taak die voorhanden is en de **"IF"** dienovereenkomstig vorm te geven; vertegenwoordigt een besparing van bijna 25% over de gehele werklast. Dit is een heel bot voorbeeld, maar ik hoop dat het mijn punt illustreert - verwijder eerst de meest voor de hand liggende dingen en behandel daarna de fijne kneepjes.

Testen op toegestane tekens in een string

Dus je vraagt de gebruiker om een keuze te maken... Ja, Nee of Annuleren... je zou hem kunnen vragen om alleen **Y, N** of **C** in te voeren. De volgende code is een beetje gekunsteld maar bewijst het punt. Gebruik **INPUT** trouwens niet voor dit soort dingen. Het zijn twee toetsaanslagen in plaats van één en is **"blokkerend"** - uw programma verliest de controle tijdens het wachten in de Input-instructie. **INPUT** is prima voor krabbelcode, maar het is veelzeggend wanneer het wordt gebruikt in een voltooid item.

```
Do
  a$ = Inkey$
Loop Until a$ ="Y" Or a$="y" Or a$="N" Or a$="n" Or a$="C" Or a$="c"
```

OK, dat is erg gekunsteld en we weten allemaal hoe we **UCase$** moeten gebruiken (toch?) Dus je zou het kunnen schrijven als:

```
Do
  a$ = UCase$(Inkey$)
Loop Until a$ ="Y" Or a$ ="N" Or a$ ="C"
```

Daar blij mee? Onthoud wat we hierboven zeiden over alle factoren in een argument dat moet worden geëvalueerd. Het is waar dat als je op de gebruiker wacht, de processor waarschijnlijk met zijn duimen aan het draaien is, dus de vertraging in de verwerking is onbelangrijk. Maar we zijn op zoek naar efficiëntie. Wat dacht je van:

```
Do
  a$ = UCase$(Inkey$)
Loop Until Instr(" YNC", a$) > 1
```

Dit is ultra-efficiënte, kleine code en slechts een enkele evaluatie. Het is ook doodeenvoudig om andere opties toe te voegen... je plaatst ze gewoon in de teststring in de **Instr()**-functie. Het werkt door elk geretourneerd teken te testen tegen een lijst met toegestane tekens. De lege string, "" (dwz geen toets ingedrukt) is altijd positie één en dat geldt ook voor het eerste niet-lege teken. Dus plaatsen we een dummy-teken op die positie van onze testreeks - de spatie, het echte vlees begint op positie twee. Dus als we een waarde groter dan 1 retourneren, hebben we ten minste teken twee op het toetsenbord getypt. Het gebruik van Instr om het nummer van het teken terug te geven is erg snel omdat we de "primitieven" van de taal gebruiken om het te ontleden in plaats van het zelf te schrijven - je gaat bijna achter de tolken aan...

Een kort woordje over **Inkey$** Het krijgt de eerste toets die uit de buffer wordt ingedrukt. Het is belangrijk om te onthouden dat de sleutel weg is zodra **Inkey$** deze heeft ingenomen. Merk je op dat we de waarde aan een variabele hebben toegewezen? waarom zou dit niet goed zijn?

```
If UCase$(Inkey$) = "Y" Or UCase$(Inkey$) = "N"
```

Plaats geen opmerkingen en witruimte in het uitvoeringspad, tenzij het moet.
Ik ben een groot voorstander van het becommentariëren en inspringen van uw code om het gemakkelijk te volgen te maken. In een gecompileerde/geassembleerde taal maakt het echt niet uit waar dat gebeurt, maar in een geïnterpreteerde taal als MMBasic moeten al die dingen worden overgeslagen om bij de code te komen - dit kost tijd (daardoor, in het verdeeldheidsargument van **Spaties versus TAB's**; Ik ben stevig in het **TAB's-kamp** omdat het de codegrootte en het aantal over te slaan bytes verkleint). Stel je de volgende situatie voor

```
For n = 1 To 1000
  ' Does something
  a = a + 1
Next
```

ziet er onschuldig genoeg uit, maar elke keer dat je door die lus gaat, moet de opmerking worden overgeslagen en dat kost tijd. Dit is beter maar functioneel identiek

```
' Does something
For n = 1 To 1000
  a = a + 1
Next
```

Wat betreft opmerkingen, houd het dan beknopt. Alleen waar nodig - ze nemen ruimte in beslag en coderen traag, waar ze ook zijn. probeer witruimte voor of na een ' te vermijden, tenzij het op de een of andere manier helpt.

Maak geen subs of functies voor dingen die je maar één keer doet als je het kunt vermijden.
Elke keer dat je naar een subprogramma of functie vertakt, kost het enige tijd om de werkomgeving in te stellen - ondanks het ontleden van de code om daarheen te springen, wat werk doen, opruimen en terugkomen.

Als je een stuk code maar één keer gebruikt, plaats het dan in de hoofdstroom van de code, tenzij het echt helpt, dat wil zeggen dat het maar af en toe wordt uitgevoerd of de code de leesbaarheid van de hoofdstroom vermindert - het is gemakkelijk om het te isoleren en maak er later een Sub van als je het nodig hebt, bijvoorbeeld als je het ook ergens anders vandaan roept of het begint gewoon de programmastroom te "beslaan".

Overweeg of u een sorteeralgoritme schrijft dat twee variabelen moet verwisselen. Dit zal waarschijnlijk duizenden, zo niet tienduizenden keren worden genoemd op een redelijk grote lijst. De swap wordt slechts op één plaats uitgevoerd. Ja, het ziet er leuk uit om een Swap-functie in je code te hebben (MMBasic heeft geen native Swap), als je elke keer dat je het aanroept, je slechts 500uS verspilt met al het huishouden, meer dan 2000 swaps die hele soort routine draait nu een seconde langzamer dan het zou kunnen. Denk je dat 2000 een onrealistisch getal is om een punt en veel swaps te bewijzen? Bij een bubbelsortering zijn er voor een lijst van 100 items in het slechtste geval 5000+ swaps nodig om te sorteren - slechts 100 items! Bij 500uS per swap heb je zojuist 2 1 / 2 . toegevoegdseconden tot de sorteertijd door te vertakken naar een mooie Swap-functie - en dat omvat geen *elke* berekening voor de taak.

Een moment van hypocrisie

Als ik mijn werk publiceer (in ieder geval alleen fragmenten, functies enz.), probeer ik echt geen cryptische code te gebruiken, maar er zijn momenten waarop een simpele Peek of Poke echt een groot verschil kan maken.

De volgende zijn twee zeer specifieke alternatieven die een groot deel van je string-slicing zullen scheren.

Wanneer u de ASCII-codes progressief zoekt via een string. Overweeg dit:

```
For n = 1 To Len(a$)
  x = Asc(Mid$(a$, n, 1))
Next
```

vrij duidelijk wat we doen, het verkrijgen van de tekencode voor elk afzonderlijk teken in de tekenreeks. Begrijpen dat strings in de eerste plaats gewoon lange lijsten van deze ASCII-codes zijn, helpt om een snellere manier af te leiden om hiermee om te gaan:

```
For n = 1 To Len(a$)
  x = Peek(VAR a$, n)
Next
```

Ik weet dat ik zei dat ik cryptisch wilde vermijden, maar deze simpele Peek retourneert direct het nummer van het puntige karakter - precies wat je wilt en precies dezelfde functionaliteit als het eerste voorbeeld, maar slechts een enkele functie en verminderde complexiteit voor de evaluator. In feite is het ongeveer 16% sneller; Met een reeks van 255 tekens duurt een MicroMite van 48 MHz respectievelijk 36 mS en 30 mS. Het verlagen van de CPU naar 5MHz overdrijft het verschil - 376mS en 300mS. Een zeer gezonde acceleratie, maar wacht er is meer...

Bij het inkorten van een variabele van links, zou het heel redelijk zijn om:

```
a$ = Left$(a$, 128)
```

Maar als je de structuur van strings kent (en dat offset 0 de lengtebyte is), zou je gemakkelijk kunnen hebben

```
Poke VAR a$, 0, 128
```

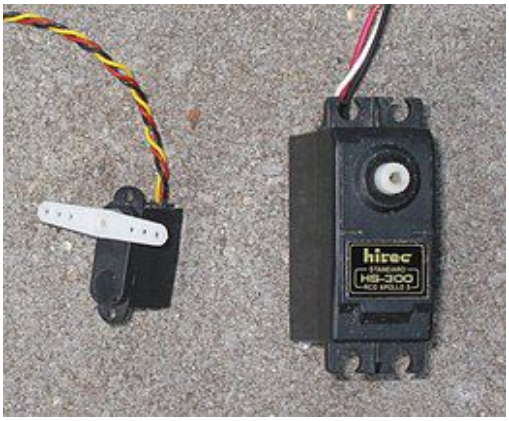
Deze eenvoudige aanpassing is een verbazingwekkende 50% sneller omdat het de string-evaluator volledig elimineert. In een lus van 1000 iteraties op een 48MHz MicroMite waren de timings respectievelijk 146mS en 98mS.
De bovenstaande twee voorbeelden zijn redelijk hardcore en zeer specifiek voor MMBasic, maar ze werken in alle huidige smaken van MMBasic (dus compatibiliteit is goed) en betalen onmiddellijk dividend. U raakt ook gewend aan enkele van de meer "low-level" functionaliteit. De leesbaarheid kan in dergelijke omstandigheden moeilijk te handhaven zijn en ik zou u willen aanmoedigen om het te vermijden waar snelheid geen probleem is, maar als het u aan het denken zet, dan is mijn werk gedaan.

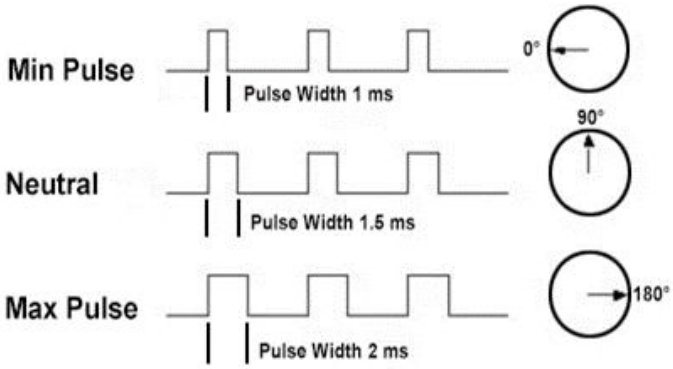
Er zijn nog veel meer "super-tweaks" in deze geest, hopelijk inspireren deze twee je om een beetje te verkennen. Speel wat met de low-level dingen en veel plezier met het leren van de innerlijke werking en het krijgen van gelikte code in het proces - ook al is het misschien niet erg leesbaar. Sla uw programma op voordat u het uitvoert, want dit kan processoruitzonderingen veroorzaken . Staar naar je code Herlees hem, polijst hem, stroomlijn hem. Wees er trots op dat het zo efficiënt is als je het kunt krijgen, maar wees niet boos als iemand je een betere manier laat zien. Dat is gratis leren! Onthoud: efficiëntie is wat je bepaalt, niet alleen kleine code, niet alleen snel - het moet een balans zijn tussen de twee die je leuk vindt en waar je een goedkeurende knipoog naar kunt geven.

Overweeg een Cosinus-berekening. Je zou een kleine routine kunnen schrijven die Chebyshev-polynomen gebruikt om de Cos van elke hoek te berekenen. Het zal klein zijn - maar het zal slooooow zijn (relatief gezien). Als je de behoefte analyseert, zou je kunnen ontdekken dat je maar tot een tiende van een graad hoeft te rekenen... Nu zou je een tabel kunnen maken van 3600 (10 stappen voor elk van 360 graden) met alle waarden voor de relevante Cosinus en zoek eenvoudig het resultaat op door naar het juiste punt in de tabel te springen: de ingevoerde graden, vermenig-vuldigd met 40 (10 voor elke graad en 4 voor elk drijvende-kommagetal) en opgeteld bij de plaats waar uw tabel begint en u heeft uw antwoord . Helemaal geen berekening. Supersnel... maar ook supergroot; 14,4 K geheugen in beslag genomen.

Dit is overigens het eeuwige raadsel van computergebruik; je kunt het groot en snel hebben of klein en langzaam.

PWM regelingen





SERVO

Deze functionaliteit maakt het mogelijk om RC(hobby) servomotoren aan te sturen.
Er zijn geen speciale commando's toegewezen, omdat de servo eenvoudig kan worden gebruikt door een **PWM-pin** in te stellen met een frequentie van **50 Hz.**
Opm. Er is nu ook een commando: **Servo.**

OPTION POWER PFM
PWM

Wijzigt de werking van de **3.3V** voeding met schakelmodus.
Standaard draait dit in de **PFM**-modus. **PWM** geeft betere ruis prestaties, maar is minder energiezuinig. Merk op dat onder zware belasting het systeem werkt in **PWM**-modus ongeacht deze instelling.

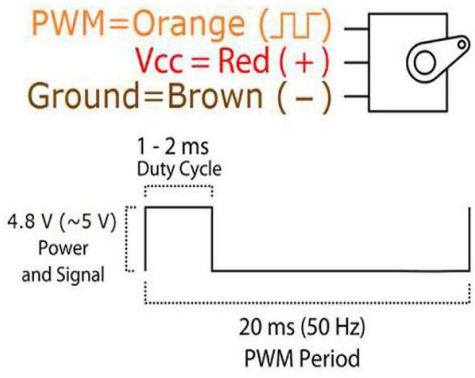
SETPIN pin, PWMnx

Pin toewijzen aan **PWMnx.**
'n' is het **PWM** nummer (**0** tot **7**) en **'x'** en is het kanaal (**A** of **B**)
De **setpin** kan worden gewijzigd totdat het **PWM** commando wordt gegeven. Op dat punt de pin wordt vergrendeld op **PWM** totdat **PWMn, OFF** wordt uitgegeven.

PWM kanaal, frequentie, [dutyA] [,dutyB]

PWM kanaal, OFF

Er zijn 8 aparte PWM frequenties beschikbaar **(kanalen 0 t/m 7)** en maximaal 16 uitgangen met individueel geregelde inschakelduur. **(duty cycle)**
U kunt op beide uitvoeren: **PWMnA** of **PWMnB** of **beide voor elk kanaal** - geen beperking.
Minimale frequentie is **15 Hz**. Maximale snelheid is:
OPTION CPUSPEED / 4.
Bij zeer hoge snelheden zullen de duty-cycles steeds beperkter worden.



Voorbeeld1:

Uitleg berekeningen Duty Cycle:

```
FREQ     = 50                       Hz  1 / 50 = 20mS
PERIODE  = 1E6 / FREQ               µS  1000000 / 50 = 20000 µS = 20 mS
PWmin    = 800                      µS  PW=PulsBreedte 1250 Hz = 1.25 kHz
PWmax    = 2200                     µS  PW=PulsBreedte  455 Hz
DCmin    = 100 * PWmin / PERIODE     %  100* 800/(1^6/50)  DC=Duty Cycle
DCmax    = 100 * PWmax / PERIODE     %  100*2200/(1^6/50)  DC=Duty Cycle
ANmin    = -90                    -90°  = 1   mS  DC   0°
ANnul    =   0                      0°  = 1.5 mS  DC  90°
ANmax    = +90                    +90°  = 2   mS  DC 180°
DCbereik = DCmax - DCmin            Bereik is 11% -  4% = 7%  Duty Cycle
ANbereik = ANmax - ANmin            Hoek is -90° - 90°
Min -> Max zwaai
For Hoek = ANmin To ANmax Step 1
DC = DCmin + ((Hoek - ANmin) / ANbereik) * DCbereik
Max -> Min zwaai
For Hoek = ANmax To ANmin Step -1

' Dubbele Servo zwaai programma

FREQ = 50                       ' Frequentie voor een servo = 50Hz
DC   =  4                       ' DCmin = 4, DCmax = 11. Berekende waarde.
                                ' DC =  4.0 = 1.0 mS =   0°
                                ' DC =  7.5 = 1.5 mS =  90°
' IO-configuratie               ' DC = 11.0 = 2.0 mS = 180°
SetPin  9, PWM 3A               ' Pin 9  = PB6 = PWM3A.
SetPin 10, PWM 3B               ' Pin 10 = PB7 = PWM3B.

' Zwaai Servo
Do                              ' Start programma loop
  ' Min -> Max zwaai
  For DC = 4 To 11 Step 0.117   ' Zwaai = -90 gr = 4 tot 90 gr = 11.
                                ' Verschil = 7 voor 0 - 180 graden.
'                               ' Step = 7 / 60 = 60 stapjes van 0-180 gr.
    PWM 3, FREQ, DC, DC         ' Freq = 50 Hz voor een servo.
    Pause 1000                  ' Duty cycle (DC) =  1 mS = -90 tot 1.5 mS
                                ' = neutraal tot 2 mS = 90 graden.
  Next DC
  ' Max -> Min zwaai
  For DC = 11 To 4 Step -0.117
    PWM 3, FREQ, DC, DC
    Pause 1000                  ' Wacht 1000 mS = 1 Sec.
  Next DC
Loop
End
```

Voorbeeld2:

In plaats van **PWM** te gebruiken, zou je **SETTICK** kunnen gebruiken om de start van alle servo's te activeren en vervolgens **PULSE** gebruiken voor elke servopin die je wilt? De periode van de **SETTICK** geeft u het begin van elk **"veld"** en de pulslengte geeft u de positie. Op die manier kun je op elke **DOUT** een servo hebben als je wilt, er is geen beperking op het aantal **PWM**-poorten. Het lijkt mij een logische manier om dingen te doen, aangezien **AFAIK** de servo's niet erg lastig vallen met frequentie, het is de mark-space-verhouding.

```
Setpin GP25 DOUT
Settick 20, servo
k = 90   ' Stel de hoek in graden in van 0-180
p = 1 + (k / 180)

Do
Loop

Sub servo
  Pulse GP25, p
End Sub
```

Het lijkt aardig te werken. Ik heb echter geen servo om mee te spelen.
Iets viel me op - als je Input k in de lus gebruikt, werkt het niet omdat de Settick-interrupt niet wordt aangeroepen terwijl Input wacht. Dat was ik vergeten.
EDIT - formule vast. Geeft nu **1 ms-2 ms** met intervallen van **20 ms**, met **1,5 ms** bij **90 graden**

SETPIN pin, cfg [, optie]
Configureert een externe I/O-pin. Raadpleeg het hoofdstuk **"Gebruik van de I/O-pinnen"** voor een algemene beschrijving van de input/output-mogelijkheden van de PicoMite.
'pin' is de I/O-pin die moet worden geconfigureerd, **'cfg'** is de modus waarop de pin moet worden ingesteld en **'optie'** is een optionele parameter.
DOUT Digitale uitgang.

SETTICK period, target [, nbr]
Dit zal een periodieke interrupt (of **"tick"**) instellen.
Er zijn **vier tick timers** beschikbaar (**'nbr' = 1, 2, 3 of 4**). **'nbr'** is optioneel en zo niet gespecificeerde **timer nummer 1** zal worden gebruikt. De tijd tussen onderbrekingen is **'periode'** milliseconden en **'doel'** is de interrupt subroutine die wordt aangeroepen wanneer de getimede gebeurtenis plaatsvindt. De periode kan variëren van **1** tot **2147483647 ms**. (ongeveer **24 dagen**). Deze onderbrekingen kunnen worden uitgeschakeld door **'periode'** op **nul** te zetten. (d.w.z. **SETTICK 0, 0, 3** schakelt tick timer nummer **3** uit).

PULSE pin, breedte
Zal een puls op **'pin'** genereren met een duur van **'breedte'** ms.
'breedte' kan een fractie zijn. Bijvoorbeeld, **0,01** is gelijk aan **10 µs** en dit maakt de generatie van zeer smalle pulsen. Het minimum is **5 µs** bij **40 MHz** tot **40 µs** bij **5 MHz**.
De gegenereerde puls heeft de tegenovergestelde polariteit van de toestand van de I/O-pin wanneer de opdracht wordt uitgevoerd. Als de output bijvoorbeeld hoog is ingesteld.
Het **PULSE** commando genereert een negatief gaande puls.
Opmerkingen:

- **'pin'** moet als uitgang worden geconfigureerd.
- Voor een puls van minder dan **3 ms** is de nauwkeurigheid **± 1 µs**.
- Voor een puls van **3 ms** of meer is de nauwkeurigheid **± 0,5 ms**.

De PWM-opdracht wordt als volgt gewijzigd:

```
PWM kanaal, frequentie, [dutyA] [,dutyB] [,deferredstart]
```

De nieuwe parameter **"deferredstart"** bereidt, indien ingesteld op **1, de PWM kanalen** voor zoals voorheen, maar start de uitvoer niet. Dit is om ongewenste opstartartefacten te voorkomen.

Er is een nieuw subcommando:

```
PWM SYNC [kanaal0offset] [,kanaal1offset] .......[kanaal7offset]
```

Dit initieert de PWM op kanalen waar een uitgestelde start is gedefinieerd of synchroniseert alleen bestaande lopende kanalen. De kracht zit echter in de mogelijkheid om de kanalen ten opzichte van elkaar te compenseren (gedefinieerd als een percentage van de tijdsperiode volgens de duty cycle - kan een float zijn)

In deze thread wordt een methode beschreven om twee signalen te creëren met één die daarna start de andere en eindigt ervoor.

Met de nieuwe functionaliteit is dit eenvoudig zonder pokes of wisselende frequentie.

```
Setpin GP0, PWM
Setpin GP2, PWM
PWM 0, 10000, 50, , 1
PWM 1, 10000, 40, , 1
PWM SYNC 0, 5
```

Dezelfde methode kan een kwadratuurklok maken

```
Setpin GP0, PWM
Setpin GP2, PWM
Setpin GP4, PWM
Setpin GP6, PWM
PWM 0, 10000, 25, , 1
PWM 1, 10000, 25, , 1
PWM 2, 10000, 25, , 1
PWM 3, 10000, 25, , 1
PWM SYNC 0, 25, 50, 75
```

Helaas heb ik geen 4-kanaals scope.

Het SYNC-commando kan ook worden gebruikt als de kanalen al actief zijn,

b.v.

```
Setpin GP0, PWM
Setpin GP2, PWM
PWM 0, 10000, 50
PWM 1, 10000, 50
Do
  For i = 0 To 99
    PWM SYNC 0, i
    Pause 50
  Next
Loop
```

In de marge ben je beperkt door het aantal klokpulsen in de tijdsperiode **@400KHz** met **1CPU @ 378MHz** heb je **945** klokperioden per cyclus dus de helft daarvan is **472** en daarom moet een "kortere" puls **470** zijn. dezelfde beperking.

```
Setpin GP0, PWM
Setpin GP2, PWM
PWM 0, 400000, 50, , 1
PWM 1, 400000, 49, 8, , 1
PWM SYNC 0, 0, 1
```

Geeft je een symmetrisch paar pulsen met exact 1 teloffset (2,6ns) aan elk uiteinde wanneer de CPU-klok 378MHz is.

Hallo Peter,

Kan de duty-cycle worden gewijzigd terwijl de PWM's actief zijn en dezelfde faserelatie behouden. Is dit geldig?

```
Setpin GP0, PWM
Setpin GP2, PWM
Setpin GP4, PWM
Setpin GP6, PWM
PWM 0, 10000, 25, , 1
PWM 1, 10000, 25, , 1
PWM 2, 10000, 25, , 1
PWM 3, 10000, 25, , 1
PWM SYNC 0, 25, 50, 75

For x = 25 To 50 Step 5
  PWM 0, 10000, x
  PWM 1, 10000, x
  PWM 2, 10000, x
  PWM 3, 10000, x
  Pause 100
Next x
```

Vergeten te vermelden dat je een **offset van -1** kunt gebruiken om een kanaal uit de synchronisatie weg te laten,

b.v.

```
Setpin GP0, PWM
Setpin GP1, PWM
PWM 0, 5000, 33, 66
Setpin GP2, PWM
Setpin GP4, PWM
PWM 1, 10000, 25, , 1
PWM 2, 10000, 25
PWM SYNC -1, 0, 25    ' Synchroniseert kanalen 1 en 2 maar laat 0 ongemoeid
```

PWM SYNC aan het werk zetten.

```
' Stappenmotor continue aandrijving.

SetPin gp0, pwm  ' CH 0a
SetPin gp1, pwm  ' CH 0b
SetPin gp2, pwm  ' CH 1a
SetPin gp3, pwm  ' CH 1b

PW = 25          ' Pulse width %.
F2 = 50 * 2      ' Frequency Hz.

Poke word &h40050000, 3
Poke word &h40050014, 3

PWM 0, F2, PW, PW - 100, 1
PWM 1, F2, PW - 100, PW, 1

PWM sync 0, 50

' Driefasige motoraandrijving.

SetPin gp0, pwm  ' CH 0a
SetPin gp1, pwm  ' CH 0b
SetPin gp2, pwm  ' CH 1a
SetPin gp3, pwm  ' CH 1b
SetPin gp4, pwm  ' CH 2a
SetPin gp5, pwm  ' CH 2b

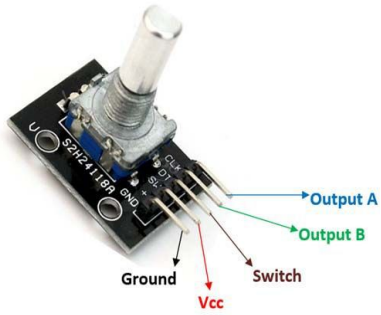
PW = 100 / 3     ' Pulse width %.
F2 = 50 * 2      ' Frequency Hz.

Poke word &h40050000, 3
Poke word &h40050014, 3
Poke word &h40050028, 3

PWM 0, F2, PW, PW - 100, 1
PWM 1, F2, PW - 100, PW, 1
PWM 2, F2, PW, PW - 100, 1
PWM sync 0, 100 / 3, 200 / 3
```

Hier wat programma's voor een rotary encoder:

```
Option Default Integer
Const false= 0
Const true = 1
Const in0  = 3 ' Encoder pinnen zijn hoog getrokken, dus zouden hoog
       ' moeten zijn in posities, middelste pen verbonden met massa.
Const in1  = 2 ' Verwissel de pindefinities zoals vereist om de
               ' juiste draairichting te krijgen.
init:
Setpin in0,intB,rint ' Onderbreekt aan beide randen van beide pinnen
Setpin in1, intB, rint
rotatedright = false
rotatedleft  = false
resetenc     = true
value        = 0
lastvalue    = 0
Print value
'
MAIN:
Do
  If value <> lastvalue Then
    Print value
    lastvalue = value
  EndIf
Loop
End
Sub rint:
  pin0 = NOT Pin(in0)              ' Reverse sense om positieve logica te krijgen.
  pin1 = NOT Pin(in1)
  If pin0 Then
    If pin1 And Not resetenc Then  ' In0 en in1 beide actief.
      If rotatedleft And Not rotatedright Then
        value = value - 1
        rotatedleft = false
      EndIf
      If rotatedright And Not rotatedleft Then
        value = value + 1
        rotatedright = false
      EndIf
    Else                           ' Alleen in0 actief.
      If resetenc And Not rotatedleft Then
        rotatedright = true
        resetenc     = false
      EndIf
    EndIf
  Else
    If pin1 Then                   ' Alleen in1 actief.
      If resetenc And Not rotatedright Then
        rotatedleft = true
        resetenc    = false
      EndIf
    Else                           ' Beide uit dus reset.
      rotatedleft  = false
      rotatedright = false
      resetenc     = true
    EndIf                          ' Opnieuw invoeren als er bijna onmiddellijk een
  EndIf                            ' andere wijziging heeft plaatsgevonden.
  If Pin(in0) = pin0 Or Pin(in1) = pin1 Then Goto rint
  IReturn
End Sub            '
```

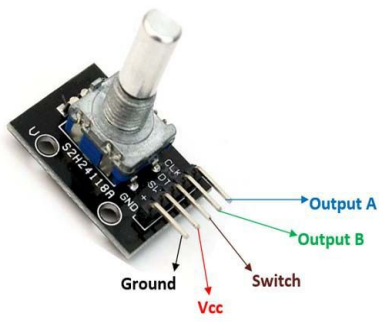


Nieuwe versie van Sub rint:

```
Sub rint
  rint2:
  pin0=NOT Pin(in0)                ' Reverse sense om positieve logica te krijgen.
  pin1=NOT Pin(in1)
  If pin0 Then
    If pin1 And Not resetenc Then  ' In0 en in1 beide actief.
      If rotatedleft And Not rotatedright Then
        value=value - 1
        rotatedleft = false
      EndIf
      If rotatedright And Not rotatedleft Then
        value = value+1
        rotatedright = false
      EndIf
    Else                           ' Alleen in0 actief.
      If resetenc And Not rotatedleft Then
        rotatedright = true
        resetenc     = false
      EndIf
    EndIf
  Else
    If pin1 Then                   ' Alleen in1 actief.
      If resetenc And Not rotatedright Then
        rotatedleft = true
        resetenc    = false
      EndIf
    Else                           ' Beide uit dus reset.
      rotatedleft  = false
      rotatedright = false
      resetenc     = true
    EndIf
  EndIf
  If Pin(in0) = pin0 Or Pin(in1) = pin1 Then Goto rint2
 ' Voer opnieuw in als er bijna onmiddellijk een andere wijziging heeft plaatsgevonden.
End Sub
```

Test programma voor een rotary encoder

```
SetPin GP10, DIN, PULLUP           ' GP10 instellen als ingang.          OutA
SetPin GP11, INTH, RInt, PULLUP    ' Interrupt instellen indien GP11=H. OutB
Do                                 ' Over contacten 10nF plaatsen. _____
 ' < Hoofdtekst van het programma > '    OutA=CLK=GP10        GND--| TOP |--OutB
Loop                               '    OutB=DT =GP11       OutC--|  O  |--GND
                                   '    OutC=SW =GP12             |_____|--OutA
Sub RInt                           ' Interrupt om de encoderuitgang te decoderen.
  If Pin(gp10) = 1 Then
    Waarde = Waarde + 1            ' Rechtsom draaien.
  Else
    Waarde = Waarde - 1            ' Linksom draaien.
  EndIf
  Pause 30                         ' Antie dender waarde.
  Print "   waarde", Waarde
End Sub
```



Uitvoer op een terminal

```
> run
  waarde    1     (Rechts om)
  waarde    2
  waarde    3
  waarde    4
  waarde    3     (Links om)
```

Digitale ingangen.

Ik gebruik een anti-rebound procedure mijn programma als voorbeeld programma in Picomite handleiding.

```
Option Default Integer        ' Standaard geheel getal.
Const false = 0
Const true  = 1
Const in0   = 49
Const in1   = 50
'
Dim Integer protb             ' Lezing pin 3 avec antirebond.
Dim Integer valrot            ' Valeur codeur rotatif.
Dim Integer lastvalue
Dim Integer vrot
Dim Integer vrotsens          ' Sens de rotatie 0=positief, 1=negatief.

init:
  Setpin in0, intH, rint3       ' Inschakelen interrupts op high of rot A pin.
  Setpin in1, DIN
  Protb = (Pin(in1) Xor 1)      ' Om de eerste status van pin "B" te definiëren.
  value = 0
  Print value
  '
MAIN:
  Do
    ' --- Minimaal 2 pass nodig met dezelfde waarde om te bevestigen.
    Protb = ((Protb * 2) And 3) Or (Pin(in1) Xor 1)
    ' ----
    If valrot <> lastvalue Then
      Print valrot
      lastvalue = value
    EndIf
  Loop
End
'
 ' ------- Proces onderbreken (interrupt)----
rint3:
  If Protb = 3 Then             ' Status 1 bevestigen
    valrot = valrot + 1
    ' Vrotsens = 0
  End If
  If protb = 0 Then             ' Status nul bevestigen
    valrot = valrot - 1
    ' Vrotsens = 2
  End If
    valrot
    ' Valrot afdrukken'
IReturn
```

OneWire Voorbeeld:

Subroutine om de temperatuur te krijgen van een **Dallas DS18B20/DS18S22** of **DSS20/DS1820** extern of parasiet gevoed via 1-draads protocol.
Rapporteert temperatuur, apparaatfamilie, ROM-code en stroomaansluiting.
Slechts 1 apparaat toegestaan op de kabel.

```
PinNbr = 20

GetTemp PinNbr, Temp, Power$, device$, code$
Print "The temperature is:" format$(Temp, "% 1.3f"); " degrees C"
Print device$; code$; Power$

Sub GetTemp (PinNbr, Value, Power$, device$, code$)
    Local T1,T2,t,a1,a2,a3,a4,a5,a6,a7,a8, device, presence, Power
    Local Countrem, T1C, TConv, T1t, T1t16, Value12, a, b, c, d, e,f,g,h

    Tconv = 750             ' Max tijd mS voor tijdelijke conversie in
                            ' parasitaire modus voor langzaamste apparaat.
'*********************************************
'* Controleer of er een apparaat aanwezig is *
'*********************************************
OneWire Reset PinNbr        ' Reset.
If presence = 0 Then        ' Geen apparaat.
   temp     = 999.99
   device$  = "Geen apparaat gevonden!"
   code$    = ""
   Power$   = ""
   Exit Sub
EndIf

'************************************************
'* Controleer of externe of parasitaire voeding *
'************************************************
OneWire Reset PinNbr, presence
OneWire Write PinNbr, 1, 2, &hcc, &hb4
OneWire Read  PinNbr, 4, 1, Power
If Power = 1 Then
   Power$ = " met behulp van externe voeding."
Else
   Power$ = " met behulp van parasitaire kracht."
EndIf

'*************************************************************************
'*Haal ROM-code op handig als u meer dan 1 apparaat op draad wilt hebben*
'*************************************************************************
OneWire Write PinNbr, 1, 1, &h33 ' Lees ROM-code.
OneWire Read  PinNbr, 0, 8, a1, a2, a3, a4, a5, a6, a7, a8
Code$ = " met ROM-code " + Str$(a1) + " " + Str$(a2) + " " + Str$(a3) +
" " + Str$(a4) + " "
Code$ = Code$ + Str$(a5) + " " + Str$(a6) + " "+Str$(a7) + " " + Str$(a8)
```

```
'************************************
'* Apparaatfamilie bepalen uit code *
'************************************
If a1       = 16 Then
  Apparaat$ = "op DS1820/DS18S20"
ElseIf a1   = 34 Then
  Apparaat$ = "op DS18S22"
ElseIf a1   = 40 Then
  Apparaat$ = "op DS18B20"
Else
  Device$   = "Niet bekend"
EndIf

'*********************
'* Lees temperatuur *
'*********************
OneWire Reset PinNbr          ' Reset voor commando (of gebruik flag=9).
OneWire Write PinNbr, 8, 2, &hcc, &h44    ' Start conversie.

' Lees extern wanneer bit gaat hi, voor parasitaire wacht gewoon.
If Power = 0 Then
  Pause Tconv
Else
  t = Timer
  Do
    If Timer - t > 1000 Then Error "Sensor reageert niet"
    OneWire Read PinNbr, 4, 1, b             ' Conversie gedaan?
  Loop To b = 1
EndIf

' Lees de temperatuur van het kladblok en converteer naar graden C + of -
OneWire Write PinNbr, 1, 2, &hcc, &hbe    ' Commando lees data.
OneWire Read  PinNbr, 2, 2, T1, T2        ' Verkrijg de gegevens.
OneWire Reset PinNbr

'*********************************************************************
'* bereken de temperatuur afhankelijk van het gebruikte apparaat *
'*********************************************************************
  'Moet type chip analyseren om temperatuur te berekenen uit T1 en T2.
  'T2 is MSB met teken, T1 is LSB.
  If a1 = 34 Or a1 = 40 Then ' DS18S22 of DS18B20 12 bit.
    If T2 And &b1000 Then    ' Negatieve temp.
      ' Maak 2s complement (1s complement+1).
      T2 = (T2 Xor &b11111111)
      T1 = (T1 Xor &b11111111) + 1
      If T1 = 1 Then T2 = T2 + 1  ' Voeg de carry toe indien nodig.
        Value = ((T2 And &b111) * 256 + T1) / 16
        Value = -Value
      Else                       ' Positieve temp.
        Value = ((T2 And &b111) * 256 + T1) / 16
      EndIf
    ElseIf a1 = 16 Then      ' DS18S20 of DS1820 9bit of calc to 12bit.
      If T2 And &b10000000 Then ' Als MSB van T2=1 dan negatief.
```

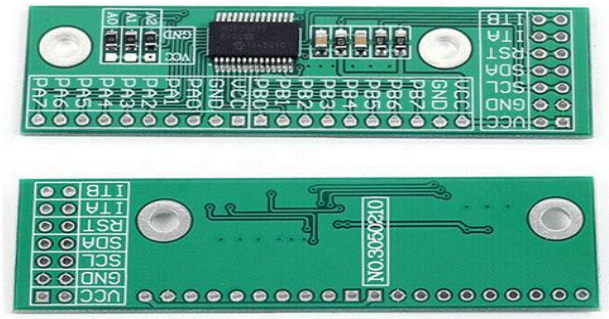
```
      ' Lees 12bit resolutie en pas aan.
      ' Lees kladblok met matchrom om Count Remaining @byte 7 te krijgen.
         OneWire Write PinNbr, 1, 9, &h55, a1, a2, a3, a4, a5, a6, a7, a8
                                          ' Lezen vanaf kladblok.
         OneWire Write PinNbr, 0, 1, &hbe
         OneWire Read  PinNbr, 0, 8, a, b, c, d, e, f, g, h
         COUNTREM = g
         ' Kort 0,5deg waarde af (of gebruik integer delen \).
         T1t = T1 And &b11111110
         T1t = T1t / 2           ' Maak hele graden.
         ' Voeg compensatiewaarden toe die uit het kladblok zijn gelezen.
         T1t16   = T1t * 16      ' Maak lsb 1/16 graden.
         Value12 = T1t16 - 4 + (16 - COUNTREM) ' 12 bit waarde toevoegen.
         ' Neem 2s-complement.
         T1C = (Value12 XOR &b111111111111) + 1
         Value = T1C / 16        ' Maak decimale waarde in graden.
         Value = -Value
      Else                       ' Positieve temp.
        Value = T1 / 2           ' 9bit-waarde.
        ' Lees 12bit resolutie en pas aan.
        ' Kladblok lezen met matchrom.
        OneWire Write PinNbr, 1, 9, &h55, a1, a2, a3, a4, a5, a6, a7, a8
                                 ' Lezen vanaf kladblok.
        OneWire Write PinNbr, 0, 1, &hbe
        OneWire Read  PinNbr, 0, 8, a, b, c, d, e, f, g, h
        COUNTREM = g
        T1t = T1 AND &b11111110'Afkappen 0,5deg(of gebruik integer delen)
        Value = T1t / 2
        Value = Value - 0.25 + (16 - COUNTREM) / 16 ' 12 bit waarde.
      EndIf
    EndIf
End Sub
```

COMMANDO	Beschrijving 400
`I2C OPEN speed, timeout` of `OPTION SYSTEM I2C SDA, SCL` `SDA:GP0,4,8,12,16,20` `SCL:GP1,5,9,13,17,21`	**Activeert de eerste I2C module in mastermodus. 'snelheid' is de kloksnelheid (in KHz) om te gebruiken en moet een van 100, 400 of 1000 zijn.** **'time-out'** is een waarde in milliseconden waarna de master verzenden en ontvangen opdrachten worden onderbroken als ze niet zijn voltooid. Het minimum waarde is **100**. Een waarde van **nul** zal de **timeout uitschakelen** (hoewel dit niet wordt aanbevolen).
`I2C WRITE addr, option, sendlen, senddata [,senddata]`	**Stuur gegevens naar de I2C slave-apparaat**. **'addr'** is het **I2C addr van de slaaf.** **'optie'** kan **0** zijn voor normaal bedrijf of **1** om **controle over de bus** te houden na de opdracht (er wordt geen stop-voorwaarde verzonden aan het einde van de opdracht). **'sendlen'** is het aantal bytes dat moet worden verzonden. **'senddata'** zijn de te verzenden gegevens dit kan op verschillende manieren worden gespecificeerd (alle verzonden waarden zullen tussen **0** en liggen **255**). Opmerkingen: • De gegevens kunnen als afzonderlijke bytes op de opdrachtregel worden aangeleverd. Voorbeeld: `I2C WRITE &H6F, 0, 3, &H23, &H43, &H25` • De gegevens kunnen zich in een eendimensionale array bevinden die is opgegeven met leeg beugels (dus geen afmetingen). **'sendlen'** bytes van de array worden verzonden beginnend met het eerste element. Voorbeeld: `I2C WRITE &H6F, 0, 3, ARRAY()` De gegevens kunnen een tekenreeksvariabele zijn (geen constante). Voorbeeld: `I2C WRITE &H6F, 0, 3, STRING$`
`I2C READ addr, option, rcvlen, rcvbuf`	**Haal gegevens op van het I2C-slave-apparaat.** **'addr' is het I2C adres van de slaaf.** **'optie'** kan **0** zijn voor normaal bedrijf of **1** om **controle over de bus** te houden na de opdracht (er wordt geen stop-voorwaarde verzonden aan het einde van de opdracht) **'rcvlen'** is het aantal te ontvangen bytes. **'rcvbuf'** is de variabele of array die wordt gebruikt om de ontvangen gegevens op te slaan - dit kan zijn: • Een stringvariabele. Bytes worden opgeslagen als opeenvolgende tekens. • Een eendimensionale reeks getallen gespecificeerd met lege haakjes. Ontvangen bytes worden opgeslagen in opeenvolgende elementen van de array beginnend met de eerste. Voorbeeld: `I2C READ &H6F, 0, 3, ARRAY()` Een normale numerieke variabele (in dit geval moet **rcvlen 1**) zijn.
`I2C CLOSE`	**Schakelt de master I2C-module uit.** Deze opdracht stuurt ook een stop als de bus wordt nog gehouden.
`I2C SLAVE`	**Zie bijlage B.**

I2C programma

MCP23017 module

Een uitbreiding van het aantal uitgangen of ingangen met een **I2C MCP23017** met **16** stuks.



Dit werkt voor mij om LED's op een MCP23017 te laten knipperen:

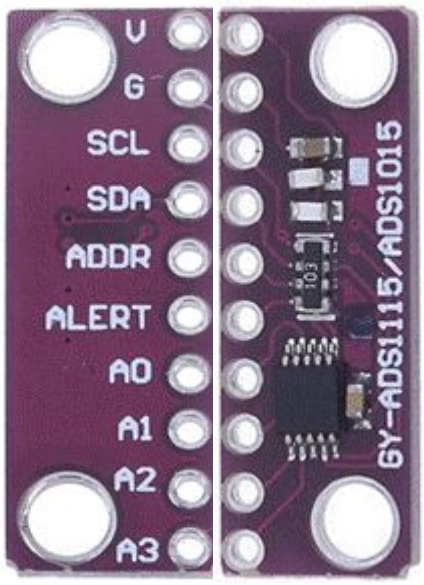
```
' MCP23017.bas

Const MCP23017 = &h20                       ' A2, A1, A0, R/W allemaal aangesloten op 0V.
Const I2Caddr  = MCP23017
Const IODIRA   = &h00                       ' Poort A IO Richtingsregister DEFAULT = I/P.
Const IODIRB   = &h01                       ' Poort B IO Richtingregister  DEFAULT = I/P.
Const IOCON    = &h0A                       ' IO Expander config register - adres &h0B
                                            ' heeft toegang tot het zelfde register.
Const GPIOA    = &h12                       ' Poort A Register voor algemeen gebruik.
Const GPIOB    = &h13                       ' Poort B General Purpose register.
Const OLATA    = &h14                       ' Poort A latch register.
Const OLATB    = &h15                       ' Poort B vergrendelingsregister.
Const GPUPB    = &h0D                       ' Poort B pull-up register.

I2C Write MCP23017, 0, 2, IODIRA, 0         ' Zet richting naar uitgang.
I2C Write MCP23017, 0, 2, IODIRB, 0         ' Zet richting naar uitgang.
' I2C WRITE MCP23017, 0, 2, GPUPB, &b001111111 ' Stel zwakke pullups in op alle behalve
                                            ' bits 6&7.  Bit76543210    Hex    Dec
                                            '             &b00111111 =  &h3F =  63
Do
  MCP17
Loop

Sub MCP17
' I2C Write MCP23017, 0, 2, IODIRA, 0       ' Stel richting naar uitgang in.
' I2C Write MCP23017, 0, 2, IODIRB, 0       ' Stel de richting in voor de uitvoer.
  Print "schrijf naar MCP23017"
  For i = 1 To 6
    I2C Write MCP23017, 0, 2, OLATA, &b10010010
    I2C Write MCP23017, 0, 2, OLATB, &b01010101
    PAUSE 1000
    I2C Write MCP23017, 0, 2, OLATA, &b01010101
    I2C Write MCP23017, 0, 2, OLATB, &b10010010
    PAUSE 1000
  Next i
  I2C Write MCP23017, 0, 2, OLATA, 0        ' Alles uitzetten.
  I2C Write MCP23017, 0, 2, OLATB, 0        ' Alles uitzetten.
  Pause 1000
End Sub
```

ADS1115 module met I2C



Dit is een ander voorbeeld van verbinding van een module met **ADS1115** die op Ebay is gekocht voor minder dan 2 €.

Dit is een **16 bit ADC 4-kanaals** module met programmeerbare versterkingsversterker. Omdat de module al twee 10K I2C pullups bevat, zijn er geen externe weerstanden nodig.

Omdat dit apparaat vrij eenvoudig te bedienen is, kan het direct worden bestuurd met behulp van een **"stuurprogramma"** dat in **MMBasic** is geschreven.

Wat uitleg over de ADS1115.

```
' Testprogramma voor ADS1115 module
  i2caddr = &h48  adres
  cnv     = &h00  conversie register
  cfreg   = &h01  config register
  cfhi    = &hC3  mux en versterking
  cflo    = &h81  data snelheid

' Stel de ADS1115 in met :
'  AINp = AIN0 en AINn = GND
'   FSR = ±4,096 V
' Speed = 128 SPS
'-------------------------------------------------------
' INVOER MULTIPLEX (inp_mux)
' AINp is de positieve invoer
' AINn is de negatieve invoer   VERSTERKING(gain)       DATA(speed)
' cfhi                          cfhi                    cflo
' &h8x, 0: AINp = AIN0 en AIN1  &hx1, 0: FSR = ±6.144V  &h01, 0:   8 SPS
' &h9x, 1: AINp = AIN0 en AIN3  &hx3, 1: FSR = ±4,096V  &h21, 1:  16 SPS
' &hAx, 2: AINp = AIN1 en AIN3  &hx5, 2: FSR = ±2.048V  &h41, 2:  32 SPS
' &hBx, 3: AINp = AIN2 en AIN3  &hx7, 3: FSR = ±1,024V  &h61, 3:  64 SPS
' &hCx, 4: AINp = AIN0 en GND   &hx9, 4: FSR = ±0,512V  &h81, 4: 128 SPS
' &hDx, 5: AINp = AIN1 en GND   &hxB, 5: FSR = ±0,256V  &hA1, 5: 250 SPS
' &hEx, 6: AINp = AIN2 en GND   &hxD, 6: FSR = ±0,256V  &hC1, 6: 475 SPS
' &hFx, 7: AINp = AIN3 en GND   &hxF, 7: FSR = ±0,256V  &hE1, 7: 860 SPS

'                 1MUX  FSR1                    SPS0 0001
' cfhi = &hC3= &b1100  0011     cflo = &h81= &b1000 0001


'
```

Voorbeeld met ADS1115 met I2C-Module:

```
' Met de tabel simpel cfhi en cflo te bepalen.
' INVOER MULTIPLEX (inp_mux)
' AINp = pos invoer,Ain0-3
' AINn = neg invoer,Ain1,3,GND  VERSTERKING(gain)       DATA(speed)
' cfhi                         cfhi                     cflo
' &h8x, 0: AINp = AIN0 en AIN1  &hx1, 0: FSR = ±6.144V  &h01, 0:   8 SPS
' &h9x, 1: AINp = AIN0 en AIN3  &hx3, 1: FSR = ±4,096V  &h21, 1:  16 SPS
' &hAx, 2: AINp = AIN1 en AIN3  &hx5, 2: FSR = ±2.048V  &h41, 2:  32 SPS
' &hBx, 3: AINp = AIN2 en AIN3  &hx7, 3: FSR = ±1,024V  &h61, 3:  64 SPS
' &hCx, 4: AINp = AIN0 en GND   &hx9, 4: FSR = ±0,512V  &h81, 4: 128 SPS
' &hDx, 5: AINp = AIN1 en GND   &hxB, 5: FSR = ±0,256V  &hA1, 5: 250 SPS
' &hEx, 6: AINp = AIN2 en GND   &hxD, 6: FSR = ±0,256V  &hC1, 6: 475 SPS
' &hFx, 7: AINp = AIN3 en GND   &hxF, 7: FSR = ±0,256V  &hE1, 7: 860 SPS
' ____________________________________________________________________
'        1MUX  FSR1            SPS0 0001
' cfhi = 1000  0001     cflo = 0000 0001
'
Option Explicit
Dim Float adc0, getvolts, wait = 100
SetTick 1 * wait, ADC, 4

DO
  Print "Volts A0  is "; Str$(adc0, 0, 3)
LOOP

Sub ADC               ' ADC uitlezen
  Local cfreg, cnv, cfhi, cflo

  addr    = &h48      ' I2C adres.
  cfreg   = &h01      ' Datasnelheid 128 sps geen comp bewerkingen.
  cnv     = &h00      ' Enkele Ain0, Versterking=+/-4.096, Datasp=128sps.
  cfhi    = &hC3      ' Config reg hi = Cx = AIN0-GND  x3 = FRS 4.096V
  cflo    = &h81      ' Config reg lo = 8x = 128sps    x1 = Geen comp bew
  I2C Write addr, 0, 3, cfreg, cfhi, cflo 'Addr 48 wijst naar config reg.
  I2C Write addr, 0, 1, cnv           ' Addr 48 wijst naar conversie reg.
  Getadc
  adc0 = (getvolts + adc0) / 2
End Sub

Sub  Getadc
  Local adcv
  Local Integer adcrd(2)              ' Om 2 bytes te ontvangen
  Pause wait / 2
  I2C Read addr, 0, 2, adcrd()
  adcv = (adcrd(0) << 8) Or adcrd(1)
  If adcv > &h7fff Then adcv = adcv - &hFFFF
  getvolts = adcv * (4.096 / 32768.0)
End Sub
```

PicoMite Basic programma ADS1115 om PH en temperatuur te meten:

```
' RTC and  ADS1115 modules on I2C pins 18,17,pin 16 DS18B20 temperature
' sensor. A0 & A1 PH input,A2 temp from probe,A3 0 to 5 volt raw inputs.
Option Explicit

Dim Float adc01,adc23,adc0,adc1,adc2,adc3,getvolts,tmp,ofs=-2.67,wait=100
Dim startime$
RTC Gettime
' RTC SETREG &h07, &h92
startime$ = Time$
Dim fs = 2, p = 0
I2C Open 100, 5000
Box 0, 0, MM.HRes - 1, MM.VRes - 1, 8, , RGB(yellow)
PWM 2, 1000, 50
' SetTick 3600000, settime, 1  ' update time from RTC hourly
SetTick 7 * wait, ADC, 4       ' periodical reading
Clr

Do
 Locate 5, 1
 Print startime$; "   "; Time$; "   "
 Print "Volts A01 is "; Str$(adc01, 0, 3),
 Print Str$((adc01 + .414.12) / .05916, 0, 3); " pH"
 Print "Volts A23 is "; Str$(adc23, 0, 3)
 Print "----------------------"
 Print "Volts A0  is "; Str$(adc0, 0, 3)
 Print
 Print "Volts A1  is "; Str$(adc1, 0, 3)
 Print
 Print "Volts A2  is "; Str$(adc2, 0, 3)
 Print
 Print "Volts A3  is "; Str$(adc3, 0, 3)
 Print
 Print "Temp  "; Str$((adc2 - 2.7315) * 100, 0, 3); " C   "
 Print "Temp  "; Str$((adc2 - 2.7315) * 100 * (9 / 5) + 32,0,3); " F   "
 Print "Temp  "; Str$((adc2 - 2.7315)*100*(9/5) + 32 + ofs, 0, 3); " F "
 Print "--------------------"
 tmp = TEMPR(16)
 Print "The temp  "; Str$(tmp,0,3); " C ", Str$(tmp * 1.8+32,0,3); " F "
 ' Print "24.8 C is "; Str$(24.8 * (9 / 5) + 32, 0, 3);" F  "

 Text MM.HRes / 2,30,Date$ + " " + Time$,cm,1,fs,RGB(red), RGB(yellow)
 Text 10, 40,  "A01 is " + Str$(adc01,2,3),,1,fs,RGB(blue),RGB(yellow)
 Text 10, 65,  "A23 is " + Str$(adc23,2,3),,1,fs,RGB(blue),RGB(yellow)
 Text 10, 90,  "A0  is " + Str$(adc0, 2,3),,1,fs,RGB(blue),RGB(yellow)
 Text 10, 115, "A1  is " + Str$(adc1, 2,3),,1,fs,RGB(blue),RGB(yellow)
 Text 10, 140, "A2  is " + Str$(adc2, 2,3),,1,fs,RGB(blue),RGB(yellow)
 Text 10, 165, "A3  is " + Str$(adc3, 2,3),,1,fs,RGB(blue),RGB(yellow)
 Text 10, 190, "Tmp is " + Str$(tmp*1.8 + 32),,1,fs,RGB(blue),RGB(yellow)
Loop
```

```
Sub ADC                                 ' ADC reading
  Local cfreg, cnv, cfhi, cflo
  cfreg = &h01: cnv  = &h00             ' rate 128sps no comp operations
  cfhi  = &h83: cflo = &h8    ' reading Dif Ain0,Ain1,FS=+/-4.096,128sps
  I2C Write &H48, 0, 3, cfreg, cfhi, cflo ' adr 48 point to config reg
  I2C Write &h48, 0, 1, cnv             ' adr 48 point to conversion reg
  getadc
  adc01 = (getvolts + adc01) / 2
  cfhi  = &hB3: cflo = &h81   ' reading Dif Ain2,Ain3,FS=+/-4.096,128sps
  I2C Write &H48, 0, 3, cfreg, cfhi, cflo ' adr 48 point to config reg
  I2C Write &h48, 0, 1, cnv             ' adr 48 point to conversion reg
  getadc
  adc23 = (getvolts + adc23) / 2
  '
  cfhi = &hC3: cflo = &h81      ' reading single Ain0,FS= +/-4.096,128sps
  I2C Write &H48, 0, 3, cfreg, cfhi, cflo ' adr 48 point to config reg
  I2C Write &h48, 0, 1, cnv             ' adr 48 point to conversion reg
  getadc
  adc0 = (getvolts + adc0) / 2
  '
  cfhi = &hD3: cflo = &h81      ' reading single Ain1,FS=+/-4.096,128sps
  I2C Write &H48, 0, 3, cfreg, cfhi, cflo ' adr 48 point to config reg
  I2C Write &h48, 0, 1, cnv             ' adr 48 point to conversion reg
  getadc
  adc1 = (getvolts + adc1) / 2
  '
  cfhi = &hE3: cflo = &h81      ' reading single Ain2,FS=+/-4.096,128sps
  I2C Write &H48, 0, 3, cfreg, cfhi, cflo ' adr 48 point to config reg
  I2C Write &h48, 0, 1, cnv             ' adr 48 point to conversion reg
  getadc
  adc2 = (getvolts + adc2) / 2
  '
  cfhi = &hF3: cflo = &h81      ' reading single Ain1,FS=+/-4.096,128sps
  I2C Write &H48, 0, 3, cfreg, cfhi, cflo ' adr 48 point to config reg
  I2C Write &h48, 0, 1, cnv             ' adr 48 point to conversion reg
  getadc
  adc3 = (getvolts + adc3) / 2
End Sub

Sub getadc
  Local adcv
  Local Integer adcrd(2)                ' to receive 2 bytes
  Pause wait / 2
  I2C Read &h48, 0, 2, adcrd()
  adcv = (adcrd(0) << 8) Or adcrd(1)
  If adcv > &H7fff Then adcv = adcv - &HFFFF
  getvolts = adcv * (4.096 / 32768.0)
End Sub


Sub settime
  RTC gettime
  Pause 50
End Sub
```

```
Sub I2Cscan
  Local found, i, h$, temp
  found = 0
  Print
  Print "I2C-Scanner from Adr. 8-119": Print
  For i = &h08 To &h77              ' gueltige I2C-Adressen von 8-119
    I2C Open 100, 1000              ' i2c enable 100kHz, 1000ms timeout
    I2C Read i, 0, 1, temp
    If MM.I2C = 0 Then
      Print "Found I2C-Address at ";  i; " (&H"; Hex$(i); ")"
      found = 1
    EndIf
    I2C Close                       ' i2c disable
  Next i
  If found = 0 Then Print "NO I2C-Address found!"
End Sub

Sub Clr
  Print Chr$(27) + "[f": Print Chr$(27) + "[2J"
End Sub

Sub Locate x, y
  Print Chr$(27) + "[" + Str$(X) + ";" + Str$(Y) + "f";
End Sub
```

I2C scanner

```
' i2c_scanner.bas Scannen van aangesloten modules. Versie 1.1
 Print " ****************** I2C SCANNER ********************"
 Print " Input/Output  PCF8574     Adres: Dec: 32   Hex:0x20"
 Print " Input/Output  MCP23017    Adres: Dec: 32   Hex:0x20"
 Print " LCD 20x4 5V0  HD44780     Adres: Dec: 32   Hex:0x20"
 Print " Licht  Lux    BH1750      Adres: Dec: 35   Hex:0x23"
 Print " Temp/%RV      DHT20       Adres: Dec: 56   Hex:0x38"
 Print " LCD 16x2 5V0  KS0066      Adres: Dec: 56   Hex:0x38"
 Print " Oled  128x64  SSD1306     Adres: Dec: 60   Hex:0x3C"
 Print " Temp/%RV      HTU21D      Adres: Dec: 64   Hex:0x40"
 Print " ADC/DAC 8Bit  PCF8591     Adres: Dec: 72   Hex:0x48"
 Print " ADC 4x 16Bit  ADS1115     Adres: Dec: 72   Hex:0x48"
 Print " EEPROM 32K    AT24C32     Adres: Dec: 80   Hex:0x50"
 Print " EEPROM 32K    AT24C32     Adres: Dec: 87   Hex:0x57"
 Print " Klok RTC 3V3  DS3231      Adres: Dec:104   Hex:0x68"
 Print " Klok RTC 5V0  DS1307      Adres: Dec:104   Hex:0x68"
 Print " Temp/%RV/Druk BMP280      Adres: Dec:118   Hex:0x76"
 Print " Opm:Indien module is 5V dan pullups I2C verwijderen"
 Print "     Alleen externe SDA en SCL pullups aan de 3V3!  "
 Print "     Of voor SDA en SCL een level shifter gebruiken."
 Print
 Print "                    Dec     Hex      Hex"
 For i = 0 To 127           ' Te testen adressen decimaal.
   I2C Write i, 0, 1, &H00 ' I2C Write addr,option,sendlen,senddata.
   If MM.I2C = 0 Then       ' MM.I2C 0=Adres gevonden. 1=Geen Adres.
     Print " Gevonden adres    "; Str$(i, 3), "   &h";
     Print Hex$(i, 2); "     0x"; Hex$(i, 2)
   EndIf
 Next i
 Print
 Print " HEX  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F"
 For y = 0 To 7
   Print " "; Hex$(y, 1); "0: ";
   For x = 0 To 15
     addr = y * 16 + x                ' Bereken adres.
     I2C Write addr, 0, 1, &H00       ' Schrijf nul naar dat adres.
     If MM.I2C = 0 Then               ' Controleer op fouten.
       If addr = 0 Then Print "-- ";  ' Addr is 00 dan "-- "
       If addr > 0 Then Print Hex$(addr, 2); " "; ' Een gevonden!
     Else
       Print "-- ";                   ' Niets hier..
     EndIf
   Next x
   Print
 Next y
 End
```

In nieuwste firmware **V6.00.02** is **LIST SYSTEM I2C** mogelijk.

De uitvoer op een terminal:

```
> run
 ****************** I2C SCANNER ********************
 Input/Output  PCF8574     Adres: Dec: 32   Hex:0x20
 Input/Output  MCP23017    Adres: Dec: 32   Hex:0x20
 LCD 20x4 5V0  HD44780     Adres: Dec: 32   Hex:0x20
 Licht  Lux    BH1750      Adres: Dec: 35   Hex:0x23
 Temp/%RV      DHT20       Adres: Dec: 56   Hex:0x38
 LCD 16x2 5V0  KS0066      Adres: Dec: 56   Hex:0x38
 Oled  128x64  SSD1306     Adres: Dec: 60   Hex:0x3C
 Temp/%RV      HTU21D      Adres: Dec: 64   Hex:0x40
 ADC/DAC 8Bit  PCF8591     Adres: Dec: 72   Hex:0x48
 ADC 4x 16Bit  ADS1115     Adres: Dec: 72   Hex:0x48
 EEPROM 32K    AT24C32     Adres: Dec: 80   Hex:0x50
 EEPROM 32K    AT24C32     Adres: Dec: 87   Hex:0x57
 Klok RTC 3V3  DS3231      Adres: Dec:104   Hex:0x68
 Klok RTC 5V0  DS1307      Adres: Dec:104   Hex:0x68
 Temp/%RV/Druk BMP280      Adres: Dec:118   Hex:0x76
 Opm:Indien module is 5V dan pullups I2C verwijderen
     Alleen externe SDA en SCL pullups aan de 3V3!
     Of voor SDA en SCL een level shifter gebruiken.

                    Dec     Hex      Hex
 Gevonden adres      56     &h38     0x38
 Gevonden adres      60     &h3C     0x3C
 Gevonden adres      64     &h40     0x40
 Gevonden adres      87     &h57     0x57
 Gevonden adres     104     &h68     0x68

 HEX  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 30: -- -- -- -- -- -- -- -- 38 -- -- -- 3C -- -- --
 40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 50: -- -- -- -- -- -- -- 57 -- -- -- -- -- -- -- --
 60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
 70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
>
```

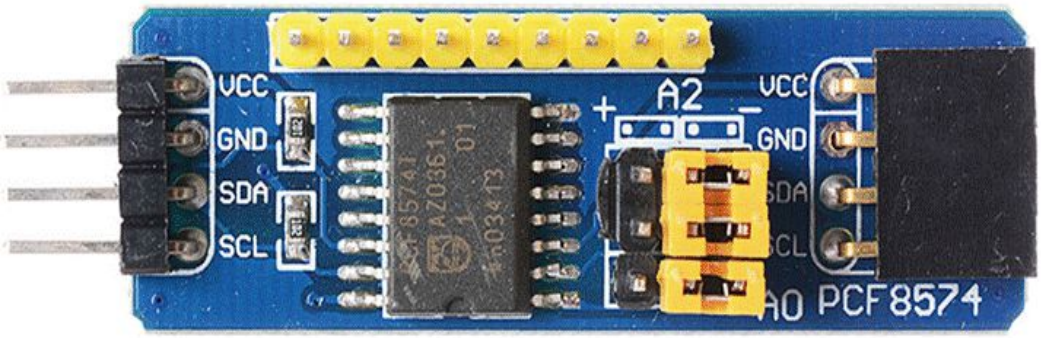
In nieuwste firmware **V6.00.02** is **LIST SYSTEM I2C** mogelijk.

```
' VL53LOX afstandsmeter code met I2C en Oled scherm met PicoMite MMBasic
' V53LOX
Dim d$
' SetPin gp4, gp5, i2c  ' Of in de Option list opnemen.
' I2C Open 100, 1000
    Const i2caddr = &H29
I2C Write i2caddr, 0, 2, &H89, &H1
Font 5
Pause 200               ' Wacht 200 mS
CLS

Do
  getdistance
  Pause 100
Loop
'
Sub getdistance
  I2C Write i2caddr, 0, 2, 0, 1
  I2C Write i2caddr, 1, 1, &H1E
  I2C Read  i2caddr, 0, 2, d$


  Text MM.HRes/2, MM.VRes/2, Str$(Str2bin(uint16,d$,BIG),3,0) + "mm", cm
  Print "Afstand :"; Str2bin(uint16,d$,BIG); " mm"
End Sub
```

PCF8574-module



Het biedt 8 digitale ingangen of uitgangen.

```
DIM AS INTEGER RData(2)        ' Dit bevat ontvangen gegevens.
' SETPIN GP4, GP5, I2C2        ' Deze regel niet als OPTION I2C ingevuld
' I2C2 OPEN 100, 1000          ' Deze regel niet als OPTION I2C ingevuld
Const I2CAddr = 32             ' Ingesteld op module i2c-adres 32=&h20
' Schrijf van 0 tot 255 op de module. Elke pin knippert op een andere
For i = 0 To 255                ' frequentie.
  PCF8574_write i
Next i
' Lees alle invoer.
' Het resultaat wordt afgedrukt in de seriële console
' zet alle ingangen in de pullup-status.
PCF8574_write 255

r = 0
For i = 0 To 1000
  PCF8574_read r
  Print r
Next i
End

Sub PCF8574_write(x)
  I2C Write I2CAddr, 0, i
  I2C Write write x
End Sub

Sub PCF8574_read(x)
  I2C Write I2CAddr, 0, i
  I2C Read  I2CAddr, 0, i
End Sub
```

DHT20 of AHT20 I2C Temperatuur en Vochtigheid sensor

```
' Voorbeeld van interface met de sensor DHT20 zonder een bibliotheek te
' gebruiken, maar alleen bestaande I2C commando's / functies met MMBasic.
' http://www.aosong.com/userfiles/files/media/Data%20Sheet%20DHT20.pdf
' Deze subroutine gevonden op het Annex Basic forum en aangepast voor MMBasic.

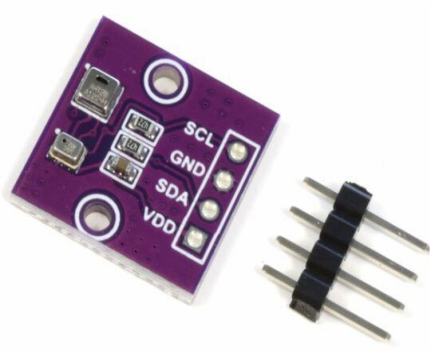
OPTION SYSTEM I2C GP4,GP5             ' I2C bus SDA=GP4, SCL=GP5
OPTION RTC AUTO ENABLE

te = 0
hu = 0
Do
  dht20_init                          ' Start de sensor
  dht20_read te,hu                    ' Lees temperatuur en vochtigheid.
  Print "Vocht:";Str$(hu,2,1);" %","Temp:";Str$(te,2,2);Chr$(176);"C",
  Print "Tijd:";Left$(Time$,5)
  Pause 60000                         ' Wacht 60 Sec.
Loop
End

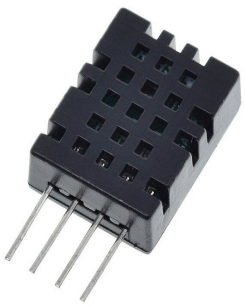
Sub dht20_init
  I2C Write &h38,0,1,&hbe             ' Begin (resetten).
  Pause 100                           ' Pauze na reset 100mS.
End Sub

Sub dht20_read(temp,hum)
  Local Integer arr(6)                ' Array gebruikt voor het lezen.
  I2C Write &h38,0,3,&hac,&h33,&h00   ' Activeer de meting.
  Pause 80                            ' Wachttijd gevraagd voor het lezen 80mS.
  I2C READ &h38,0,6, arr()            ' Lees de gegevens.
  If (arr(0) And 128) Then            ' Arr(0) is de status.
    Print "Gegevens niet beschikbaar, verhoog de wachttijd"
  Else
     hum = ((arr(1) << 12) + (arr(2) << 4) + (arr(3) >> 4)) / 10485.76
    temp = (((arr(3) And &h0f) << 16) + (arr(4) << 8) + arr(5)) / 1048576
    temp = temp * 200 - 50
  End If
End Sub
```

```
  > run
  Vocht:59.7 %    Temp:17.74°C    Tijd:15:41
```



AHT20 + BMP280 sensor



DHT20 sensor

```
' Option SYSTEM SPI GP18,GP19,GP16
' Option SYSTEM I2C GP4,GP5           ' GP4=SDA , GP5=SCL
' Option LCDPANEL SSD1306I2C, LANDSCAPE
' Option SDCARD GP17
' Option RTC AUTO ENABLE
' RTC SETTIME 2023,3,26,21,28,0 jaar,maand,dag,uur,min,sec RTC tijd instellen.
' Zonder RTC tijd: Time$="hh:mm:ss" en Date$="dd-mm-jjjj" Bij power on opnieuw.
' Code dht20 komt van AnnexBasic en is geschikt gemaakt voor MMBasic.
te = 0
hu = 0
' CLS                               ' Wis Oled scherm.
' Backlight 20                      ' Backlight 0 - 100.
Do
  dht20_init                        ' Start de sensor.
  dht20_read te,hu                  ' Lees temperatuur en vochtigheid.
  Print "DHT20: ", "Vocht:";Str$(hu,2,1);" %","Temp:";Str$(te,2,1);Chr$(176);
  Print "C","Tijd:";Left$(Time$,5);"   ";dagw$(Date$);" "; Date$
 ' Text 0 , 0, dagw$(Date$)         ' Dagw$(Date$) is dag van de week.
 ' Text 20, 0, Left$(Date$,5)
 ' Text 80, 0, Left$(Time$,5)
 ' Text 0 ,20, "Temp.:     C"
 ' Text 81,20, Chr$(96)             ' Let op Chr$(96) is graden teken.
 ' Text 50,20, Str$(te,2,1)         ' Str$(te,2,1) is temperatuur xx.x C
 ' Text 0 ,40, "Vocht:     %"
 ' Text 50,40, Str$(hu,2,1)         ' Str$(hu,2,1) is vochtigheid xx.x %
  Pause 60000                       ' Wacht 20 Sec.
Loop
End
Sub dht20_init                      ' DHT20 sub routine.
  I2C Write &h38,0,1,&hbe           ' Begin (resetten).
  Pause 100                         ' Pauze na reset 100mS.
End Sub
Sub dht20_read(temp,hum)
  Local integer arr(6)              ' Array gebruikt voor het lezen.
  I2C Write &h38,0,3,&hac,&h33,&h00 ' Activeer de meting.
  Pause 80                          ' Wachttijd gevraagd voor het lezen 80mS.
  I2C READ &h38,0,6, arr()          ' Lees de gegevens.
  If (arr(0) And 128) Then          ' Arr(0) is de status.
    Print "Gegevens niet beschikbaar, verhoog de wachttijd"
  Else
     hum = ((arr(1) << 12) + (arr(2) << 4) + (arr(3) >> 4)) / 10485.76
    temp = (((arr(3) And &h0f) << 16) + (arr(4) << 8) + arr(5)) / 1048576
    temp = temp * 200 - 50
  EndIf
End Sub
Function dagw$(dt$)                 ' Functie dag van de week bepalen NL Ma-Zo
  Local INTEGER d,m,y,jd,weekday
  d = Val(Mid$(dt$,1,2))            ' dd-xx-xxxx.
  m = Val(Mid$(dt$,4,2))            ' xx-mm-xxxx.
  y = Val(Mid$(dt$,7,4))            ' xx-xx-jjjj.
  If m < 3 Then m = m + 12 : y = y - 1
  jd = d + Fix(365.25 * (y+4716)) + Fix(30.6001 * (m + 1)) - y\100 + y\400 + 4
  weekday = (jd Mod 7) + 1
  dagw$ = Mid$("  Ma Di Wo Do Vr Za Zo ", weekday * 3, 3)
End Function
```

Heb geen probleem gehad met AHT10 die System I2C deelt met DS3231 RTC, 24C32 EEPROM en VL53LOX laserafstandsapparaat tegelijkertijd.

Werkt met alle **PicoMite** **firmware.**

```
' AHT10 op systeem I2C

Dim Float humid, temp
Const AHaddr = &H38                 ' AHT10 I2C adres
I2C Write AHaddr, 0, 3, 172, 51, 0 ' Trigger AHT10 meting om data te vernieuwen
Pause 45                            ' Wacht op data

Do
 T% = Timer + 999
 AHT10
 Print "Temp "; temp; "`C", "Humidity "; humid; "%"
 Do : Loop Until Timer > T%
Loop

Sub AHT10
  I2C Read AHaddr, 1, 5, dat() ' Lees eerste 5 databytes, de laatste niet nodig
  humid = Cint(dat(1) / 2.56)
  temp  = Cint((((dat(3) And 15) << 8) + dat(4)) / 2.048 - 500) / 10
  I2C Write AHaddr, 0, 3, 172, 51, 0    ' Trigger meting voor volgende lezing.
End Sub
```

Edit

Het AHT10 adres kan ook &H39 zijn als een link overbrugd is. Dit zal het vinden.

```
' Vindt adres
I2C Write &h38, 0, 1, 0                ' Schrijf nul naar dat adres
If MM.I2C = 0 Then                     ' Controleer op fouten
  AHaddr  = &H38
Else
  AHaddr  = &H39
EndIf
Print ; " AHT10 adres "; Hex$(AHaddr, 2); " Hex"
```

Wat aanpassingen voor bovenstaande code.

```
' 25/01/2025
' MM2 v5.05.05
' Read AHT10, DS3231 and VL53LOX using I2C.
RTC GETTIME : Print Date$, Time$
' Pin 17 = I2C clock, Pin 18 = I2C data
I2C OPEN 100, 100, PU
Dim Integer  dat(4), dst(1), dist
Dim Float humid, temp, Trtc
Const VLaddr = &H29, AHaddr = &H38    ' VL53LOX & AHT10 device addresses.
I2C Write VLaddr, 0, 2, 0, 1          ' Trigger first VL53LOX measurement.
I2C Write AHaddr, 0, 3, 172, 51, 0    ' Trigger first AHT10 measurement.
Print "rel. humidity    temp                                    distance"
Pause 75                              ' Wait 75 mS for data.
```

```
Sub AHT10
  I2C Read AHaddr, 1, 5, dat() ' Read first 5 data bytes - last one not needed.
  humid = Cint(dat(1) * 0.390625)
  temp = Cint((((dat(3) And 15) << 8) + dat(4)) * 0.48828125 - 500) * 0.1
  I2C write AHaddr, 0, 3, 172, 51, 0  ' Trigger measurement for next read.
End Sub

Sub VL53LOX
  I2C Write VLaddr, 1, 1, 30
  I2C Read VLaddr, 0, 2, dst()
  dist = (dst(0) << 8) + dst(1)
  I2C Write VLaddr, 0, 2, 0, 1        ' Trigger measurement for next read.
End Sub

Sub DS3231                            ' Read DS3231 RTC temp, if installed.
  Local Integer Tdeg, Tdec
  On Error skip 3 : RTC GetReg 17, Tdeg : RTC GetReg 18, Tdec
  Trtc = 0.1 * Cint((Tdeg + Tdec / 256) * 10) And &hFFF)
End Sub

Do
  AHT10
  DS3231
  VL53LOX                             ' Comment this out if you don't have it.
  Print  humid; " %",, temp; " C",, "RTC Temp  "; Trtc; " C",, dist " mm"
  Pause 2000                          ' Read every 2S.
Loop
End
```

```
> RUN
25-01-2025      15:33:49
rel. humidity    temp                                          distance
52 %            23.2 C         RTC Temp   23.3 C                89 mm
52 %            23.3 C         RTC Temp   23.3 C                91 mm
52 %            23.2 C         RTC Temp   23.3 C                88 mm
52 %            23.2 C         RTC Temp   23.3 C                89 mm
52 %            23.2 C         RTC Temp   23.3 C                88 mm
52 %            23.2 C         RTC Temp   23.3 C                88 mm
>
```

LCDI2CLib test voor bijna elk type PCF8574 I2C naar LCD-module.

```
PicoMite MMBasic Version 5.07.07b5
OPTION SYSTEM SPI GP18,GP19,GP16
OPTION SYSTEM I2C GP4,GP5
OPTION COLOURCODE ON
OPTION DISPLAY 40, 90
OPTION LCDPANEL SSD1306I2C, LANDSCAPE    ' 128x64 Oled scherm
OPTION SDCARD GP22
OPTION RTC AUTO ENABLE
' Deze subroutine gevonden op het MMBasic forum en geschikt gemaakt.
' Gebruikt PCF8574 - 8 bit I2C Port Expander en zonder 4K7 pullups aan 5V
' I2CAddress is &H20=20x4 LCD, &H38=16x2 LCD of zie: I2C_Scanner.bas
' PCF8574-poort is als volgt bedraad:
' P0 - RS
' P1 - RW
' P2 - EN
' P3 - Achtergrondverlichting aan/uit
' P4 - D4 LCD
' P5 - D5 LCD
' P6 - D6 LCD
' P7 - D7 LCD-scherm
' LCDI2C_INIT &H38,1,0,2,3 ' I2CAddr, D0~D3=0 D4~D7=1, RS=0, EN=2, BL=3.
' LCDI2C_BackLightOn       ' Zet de LCD-achtergrondverlichting AAN.
' LCDI2C_BackLightOff      ' Zet de LCD-achtergrondverlichting UIT.
' LCDI2C(LineNum, CharPos, Text$) ' Line, Char, Text$=Teken of Commando.
' LCDI2C_CLEAR()           ' Wist het LCD.
' LCDI2C_CMD(Byte)         ' Stuur een COMMAND byte naar de LCD, dwz RS=0
' LCDI2C_DATA(Byte)        ' Stuur een DATA    byte naar de LCD, dwz RS=1
' LCDI2C_DirectSend(Byte)  ' Stuur een BYTE    naar het LCD.
' * Intern gebruikt door de lib *
' LCDI2C_WireWrite(Byte)   ' Schrijf een byte op de I2C-bus naar de LCD.
' WAITFORAKEY(MSG$)        ' Wacht op een toets input data.
' LCDI2C_Init &H38,1,0,2,3 ' I2CAddr, D0~D3=0 D4~D7=1, RS=0, EN=2, BL=3

LCDI2C_Init &H3F, 1, 0, 2, 3      ' I2CAddr=&H20=20x4, &H38=16x2, &3F=16x2.
' Hoofd programma.
Do                                ' Met een 4x20 2004 LCD scherm.
  LCDI2C_BackLightOn              ' Achtergrond LED aan.
  LCDI2C_CMD(64)                  ' Aangepast teken Chr$(0) is CMD Reg. 64-71.
  LCDI2C_DATA(7): LCDI2C_DATA(5): LCDI2C_DATA(7): LCDI2C_DATA(0)   ' Byte 1-4.
  LCDI2C_DATA(0): LCDI2C_DATA(0): LCDI2C_DATA(0): LCDI2C_DATA(0)   ' Byte 5-8.
  ' De character editor:   https://maxpromer.github.io/LCD-Character-Creator/
  LCDI2C 1, 1, "BuitenT:R 18.3 C" ' LCDI2C Line, Pos, Data$
  LCDI2C 1,15, Chr$(0)            ' Zet graden teken op Line=1, Pos=15, Chr$(0)
  LCDI2C 2, 1, "BinnenT:  20.6 C" ' LCDI2C Line, Pos, Data$
  LCDI2C 2,15, Chr$(0)            ' Zet graden teken op Line=2, Pos=15, Chr$(0)
  Pause 10000                     ' Wacht 10 seconde.
Loop
End
```

```
' **** Alle subroutines voor lcd 16x2 en 20x4 ****
' Wacht op een toets input data.
Sub WaitForAKey(Msg$)
  If Msg$ = "" Then Msg$ = "Press any key to continue"
  Print Msg$;
  Do
  Loop While Inkey$ = ""
  Print
End Sub
' Line=1-4, Char=1-16, Text$=CHR$ + ALLE CHARACTERS
Sub LCDI2C(LineNum, CharPos, Text$)
  Local I
  If LineNum = 1 Then I = (&H80 + CharPos - 1)
  If LineNum = 2 Then I = (&HC0 + CharPos - 1)
  If LineNum = 3 Then I = (&H94 + CharPos - 1)
  If LineNum = 4 Then I = (&HD4 + CharPos - 1)
  LCDI2C_CMD(I)
  For I = 1 To Len(Text$)
    LCDI2C_DATA(Asc(Mid$(Text$, I, 1)))
  Next I
End Sub
' Wis het LCD scherm
Sub LCDI2C_Clear()
  LCDI2C_CMD(&H01)
End Sub
'
Sub LCDI2C_Init I2CAddr, NibPos, RSBitNo, ENBitNo, BLBitNo
  Dim LCDI2C_BackLight
  LCDI2C_BackLight = 2 ^ BLBitNo
  Dim LCDI2C_LCDBackLight
  LCDI2C_LCDBackLight = LCDI2C_BackLight
  Dim LCDI2C_I2CAddr
  LCDI2C_I2CAddr = I2CAddr
  Dim LCDI2C_RSDataMask
  LCDI2C_RSDataMask = 2 ^ RSBitNo
  Dim LCDI2C_EMask
  LCDI2C_EMask = 2 ^ ENBitNo
  Dim LCDI2C_NibPos
  LCDI2C_NibPos = NiBPos
  LCDI2C_CMD(&H33)
  LCDI2C_CMD(&H32)
  LCDI2C_CMD(&H28)
  LCDI2C_CMD(&H0C)
  LCDI2C_CMD(&H06)
  LCDI2C_CMD(&H01)
  LCDI2C_BackLightOff()
End Sub
'
Sub LCDI2C_Close()
  I2C CLOSE
End Sub
'
Sub LCDI2C_BacklightOn
  LCDI2C_LCDBacklight = LCDI2C_Backlight
  LCDI2C_WireWrite(0)
End Sub
```

```
'
Sub LCDI2C_BacklightOff
  LCDI2C_LCDBackLight = &H00
  LCDI2C_WireWrite(0)
End Sub
'
Sub LCDI2C_CMD(Byte)
  LCDI2C_DirectSend(Byte And &HF0)
  LCDI2C_DirectSend((Byte And &H0F) * 16)
End Sub
'
Sub LCDI2C_DATA(Byte)
  LCDI2C_DirectSend((Byte And &HF0) Or LCDI2C_RSDataMask)
  LCDI2C_DirectSend(((Byte And &H0F) * 16) Or LCDI2C_RSDataMask)
End Sub
'
Sub LCDI2C_DirectSend(Byte)
  Local B, B1
  If LCDI2C_NibPos = 0 Then
    B1 = Byte \ 16
  Else
    B1 = Byte
  EndIf
  B = B1 Or LCDI2C_EMask Or LCDI2C_LCDBacklight
  I2C WRITE LCDI2C_I2CAddr, 0, 1, B
  I2C WRITE LCDI2C_I2CAddr, 0, 1, B1 Or LCDI2C_LCDBacklight
End Sub
'
Sub LCDI2C_WireWrite(Byte)
  I2C WRITE LCDI2C_I2CAddr, 0, 1, Byte Or LCDI2C_LCDBacklight
End Sub
```

Aangepaste grafische afbeeldingen maken op 1602/2004 LCD

Dit werkt op HD44780 KS0066 SPLC780-controllerchips met 1,2 of 4 regels tekst.
De karakters zijn gemaakt op een 5X8 raster.
Hier is een aangepast karakter van een bel.

```
   16 8 4 2 1
 1      *      =  4
 2    * * *    = 14(8+4+2=14)
 3    * * *    = 14
 4    * * *    = 14
 5  * * * * *  = 31(16+8+4+2+1=31)
 6             =  0
 7      *      =  4
 8             =  0
```

Er zijn 8 aangepaste tekens:

```
CHR$(0) heeft het commando LCD CMD  64   ' Byte 1= 64, 2= 65, ... 8= 71
CHR$(1) heeft het commando LCD CMD  72   ' Byte 1= 72, 2= 73, ... 8= 79
CHR$(2) heeft het commando LCD CMD  80   ' Byte 1= 80, 2= 81, ... 8= 87
CHR$(3) heeft het commando LCD CMD  88   ' Byte 1= 88, 2= 89, ... 8= 95
CHR$(4) heeft het commando LCD CMD  96   ' Byte 1= 96, 2= 97, ... 8=103
CHR$(5) heeft het commando LCD CMD 104   ' Byte 1=104, 2=105, ... 8=111
CHR$(6) heeft het commando LCD CMD 112   ' Byte 1=112, 2=113, ... 8=119
CHR$(7) heeft het commando LCD CMD 120   ' Byte 1=120, 2=121, ... 8=127
```

Dus om het Bell-teken hierboven af te drukken, gebruikt u

```
LCD CMD 64 : LCD DATA 4, 14, 14, 14, 31, 0, 4, 0
LCD 1, 1, CHR$(0)
```

Om een ander aangepast teken af te drukken, gebruikt u de volgende **CHR$**-

```
LCD CMD 72 : LCD DATA X,X,X,X,X,X,X,X
LCD 1, 2, CHR$(1)
```

Denk eraan als u de PicoMite gebruikt om **BITBANG LCD** te gebruiken.
De moeite waard om toe te voegen, je hoeft het slechts één keer te doen :

```
LCD CMD 64
LCD DATA 4, 14, 14, 14, 31, 0, 4, 0
```

(bijv. in de preambule) niet elke keer dat je de :

```
LCD 1, 1, CHR$(0)
```

Voor LCDI2C 16x2 of 20x4 scherm als volgt:

```
LCDI2C_Init &H3F, 1, 0, 2, 3    ' I2CAddr=&H20=20x4, &H38=16x2, &3F=16x2.
LCDI2C_CMD(64)                  ' Aangepast teken Chr$(0) is CMD(64) Reg(64-71)
LCDI2C_DATA(7) : LCDI2C_DATA(5) : LCDI2C_DATA(7) : LCDI2C_DATA(0) ' Byte 1 - 4
LCDI2C_DATA(0) : LCDI2C_DATA(0) : LCDI2C_DATA(0) : LCDI2C_DATA(0) ' Byte 5 - 8
LCDI2C 1, 1, "BuitenT:R 18.3 C" ' LCDI2C Line, Pos, Data$
LCDI2C 1,15, Chr$(0)            ' Zet graden teken op Line 1, Pos=15, Chr$(0)
```

De character editor: https://maxpromer.github.io/LCD-Character-Creator/
Zie ook subroutines **LCDI2C.**

De PicoMite ondersteunt de **SSD1963-schermen**, maar deze zijn nu duur en gebruiken bijna alle I/O-pinnen. Er zijn echter veel SPI-schermen beschikbaar.
Deze hebben twee nadelen. Ten eerste: ze zijn relatief traag omdat ze afhankelijk zijn van seriële gegevensoverdracht. Ten tweede: veel van hen ondersteunen het lezen van de framebuffer van de controller niet en zelfs degenen die dat wel doen, zijn zelfs langzamer in leesmodus dan in schrijf-modus.

Daarom is besloten om wat extra functionaliteit toe te voegen om dit te omzeilen. De beperking is dat het geheugen in de RP2040 bij lange na niet groot genoeg is om een complete framebuffer vast te houden voor zelfs een 240x240 scherm zoals de Waveshare RP2040-LCD-1.28 in zijn normale resolutie (RGB565 = 2 bytes per pixel). Dus ik heb een deel van de code van de PicoMiteVGA gebruikt om een geheugenframebuffer te creëren op RGB121 zoals de PicoMiteVGA. Hierdoor kan ik **twee** geheugenframebuffers maken en heb ik nog steeds voldoende ruimte voor variabelen.
De demo gebruikt deze nieuwe functionaliteit om een artefactvrij bewegend display te creëren. Ik zal de nieuwe firmware posten zodra een beetje opruimen is voltooid.

Het grote probleem met al deze beeldschermen is iets **"ongedaan maken"**. Beschouw de demo maar zonder de framebuffers.
Denk aan de wijzers van de klok. Om de secondewijzer te verplaatsen moet ik:

1: De achtergrond die eronder zat herstellen.
2: De achtergrond opslaan waar ik hem nu ga tekenen.
3: De hand in de nieuwe positie tekenen.

Dit alles kost tijd en het resultaat is dat de hand flikkert van de ene locatie naar de andere.
Dit is verbeterd als de achtergrond een egale kleur heeft, aangezien ik stap 2 kan missen, maar ik krijg nog steeds flikkering en zelfs met moeilijke) zoals de demoweergave.
De framebuffer lost dit op omdat ik de afbeelding in een framebuffer kan assembleren waar al deze bewegingen onzichtbaar zijn en dan gewoon de voltooide afbeelding (**COPY**) of het deel dat is veranderd (**BLIT**) naar de daadwerkelijke weergave schilderen.
Het kopiëren van framebuffer naar framebuffer duurt minder dan een milliseconde. Het kopiëren van het volledige beeld naar het display in de demo duurt **50 mSec @** standaard CPU snelheid en **46mSec @ 378 MHz.**

```
Option Explicit
Option Default None
'
Const Y_top = 0
' Y-coördinaat van de bovenkant van het afbeeldingsbestand op het scherm
Const X_left = 0
' X-coördinaat van de linkerkant van het afbeeldingsbestand op het scherm Const
s_Y_point = 172
' Y-coördinaat van het midden van de secondewijzer ten opzichte van de
' afbeelding boven.
Const s_X_point = 119
' X-coördinaat van het midden van de secondewijzer ten opzichte van de
' afbeelding links.
Const h_Y_point = 119
' Y-coördinaat van het midden van de uren- en minutenwijzers ten opzichte
' van de afbeelding boven.
Const h_X_point = 120
' X-coördinaat van het midden van de uren- en minutenwijzers ten opzichte
' van de afbeelding boven.
```

```
Const sl = -30
Const ml = -100
Const hl = -75
'
' Nu definiëren we de wijzers als de X,Y coördinaten van een aantal drie-
' hoeken met het rotatiepunt op 0,0 -coördinaten bovenaan de aanwijzer '
' zijn negatief.
' De volgorde van gegevenselementen in de array is:
' t1x1,t1y1,t1x2,t1y2,t1x3,t1y3,t2x1,t2y1,t2x2,t2y2,t2x3,t2y3,...
' tntx1,tnty1, tntx2,tnty2,tntx3,tnty3
'
Dim Integer nt = 6
' Aantal driehoeken gebruikt om de aanwijzer te definiëren.
Dim Integer x1(nt - 1) = (-1, -1, -2, -2, -3, -3)
Dim Integer y1(nt - 1) = (0, 0, 0, 0, 0, 0)
Dim Integer x2(nt - 1) = (-1, 1, -2, 2, -3, 3)
Dim Integer y2(nt - 1) = (sl, sl, ml, ml, hl, hl)
Dim Integer x3(nt - 1) = (1, 1, 2, 2, 3, 3)
Dim Integer y3(nt - 1) = (sl, 0, ml, 0, hl, 0)
Dim Integer pcolour = RGB(Green) ' Definieer de kleur van de aanwijzer.

Const s_pivot = 3, m_pivot = 5, h_pivot = 3
' Diameter in pixels van het draaipunt van de aanwijzer.
'
Dim Integer xx0(nt - 1), yy0(nt - 1), xx1(nt - 1), yy1(nt - 1),xx2(nt-1),
yy2(nt - 1), tcol(nt - 1)
Dim Integer first = 1
' Marker om alternatieve verwerking te besturen bij de eerste keer door  ' de
lus.
'
FRAMEBUFFER Create
FRAMEBUFFER Layer
FRAMEBUFFER Write l
CLS
Load Image "gezicht"
'
' Democode de aanwijzer verplaatsen.
Dim Float secondangle = 0, minuteangle = 0, hourangle = 0

Do
  FRAMEBUFFER Copy l,f
  If secondangle Mod 60 = 0 Then Inc minuteangle, 1
  If (minuteangle Mod 72=0) And (secondangle Mod 60=0) Then Inc hourangle,6

  drawpointer
  FRAMEBUFFER Copy f, n
  Inc secondangle, 6
Loop
```

```
'
Sub drawpointer
  Local Integer i, x, y
  Local Float angle
  For i = 0 To nt - 1
    If i < 2 Then
      x = s_X_point
      y = s_Y_point
      pcolour = RGB(green)
      angle = secondangle
    Else
      x = h_X_point
      y = h_Y_point
      pkleur = RGB(white)
      If i < 4 Then
        angle = minuteangle
      Else
        angle = hourangle
      EndIf
    EndIf
    rotateangle(i, pcolour, angle, x + X_left, y + Y_top, x1(i), y1(i), x2(i),
y2(i), x3(i), y3(i)) ' Draai de aanwijzer in de tekenreeks.
  Next i
  FRAMEBUFFER Write f
  Triangle xx0(), yy0(), xx1( ), yy1(), xx2(), yy2(),tcol(),tcol()
drawpivot
End Sub

Sub rotatetriangle(n As Integer, col As Integer, angle As Float, x As
Integer, y As Integer, x0 As Integer, y0 As Integer, x1 As Integer, y1 As
Integer, x2 As Integer, y2 As Integer)
  Local Float sine = Sin(Rad(angle)), cosine = Cos(Rad(angle))
  Local Integer x0a, y0a, x1a, y1a, x2a, y2a
  x0a = x0 * cosine - y0 * sine + x
  y0a = y0 * cosine + x0 * sine + y
  x1a = x1 * cosine - y1 * sine + x
  y1a = y1 * cosine + x1 * sine + y
  x2a = x2 * cosine - y2 * sine + x
  y2a = y2 * cosine + x2 * sine + y
  xx0(n) = x0a
  yy0(n) = y0a
  xx1(n) = x1a
  yy1(n) = y1a
  xx2(n) = x2a
  yy2(n) = y2a
  tcol(n) = pcolour
End Sub
'
Sub drawpivot()
' Zet hier alles wat je wilt om het draaipunt te tekenen zoals je wilt.
  Circle s_X_point + X_left, s_Y_point + Y_top, s_pivot, 0,,, RGB(green )
  Circle h_X_point + X_left, h_Y_point + Y_top, m_pivot, 0,,, RGB(white)
  Circle h_X_point + X_left, h_Y_point + Y_top, h_pivot, 0,,, RGB(black)
End Sub
```

COMMANDO's en BASIC TREFWOORDEN

```
 /*
 */
#DEFINE                        Met CMM2 LOAD fname$ [,R] en CMM2 RUN fname$
#INCLUDE
#COMMENT START/END
#MMDEBUG ON/OFF

A
A: of B: of Drive "A:" of Drive "B:"
ADC CLOSE
ADC FREQUENCY freq
ADC OPEN freq, n_channels [, interrupt]
ADC START array1!() [, array2!()] [, array3!()] [, array4!()]
ADC START array1!()[,array2!()] [,array3!()] [,array4!()][,Chan4arr!()][,C1min]
                   [,C1max][,C2min][,C2max][,C3min] [,C3max] [,C4min] [,C4max]
ADC RUN array1%(), array2%)
ARC x, y, r1, [r2], a1, a2[, c]
AUTOSAVE
AUTOSAVE APPEND
AUTOSAVE CRUNCH

B
BACKLIGHT n [,DEFAULT]
BITBANG      Vervangen door het commando DEVICE
BLIT
BLIT x1, y1, x2, y2, w, h
BLIT CLOSE [#] b
BLIT COMPRESSED address%, x, y [,col]
BLIT FRAMEBUFFER from, to, xin, yin, xout, yout, width, height [,colour]
BLIT LOAD [BMP] [#]b, fname$ [,x] [,y] [,w] [,h]
BLIT LOAD JPG [#] b, fname$ [, x] [, y] [, w] [, h]
BLIT LOADBMP [#] b, fname$ [, x] [, y] [, w] [, h]
BLIT MEMORY address, x, y [,col]
BLIT MERGE colour, x, y, w, h
BLIT READ [#] b, x, y, w, h
BLIT WRITE [#]b, x, y [, mode]%, x, y [,col]
BOX x, y, w, h [, lw] [, c] [, fill]

C
CALL usersubname$ [, usersubparameters, ....]
CAMERA
CAMERA OPEN XLKpin, PLKpin, HSpin, VSCpin, RETpin, D0pin
CAMERA CAPTURE [scale, [x , y]]
CAMERA CLOSE
CAMERA CHANGE image%(), change! [,scale [, x, y]]
CAMERA TEST tnum
CAMERA REGISTER reg%, data%
CAT S$, N$
CASE testexp [[, testexp]?]   zie ook SELECT CASE
CASE ELSE
CHDIR dir$
```

```
CIRCLE x, y, r [, lw] [, a] [, c] [, fill]
CLEAR
CLOSE [#] fnbr [, [#] fnbr] …
CLS [colour]
CMM2 LOAD fname$ [,R]
CMM2 RUN fname$
COLOR fore [, back]
COLOUR fore [, back]
CONST id = expression [, id = expression] … etc
CONTINUE
CONTINUE DO
CONTINUE FOR
COPY fname1$ TO fname2$
COPY fname$ TO dirname$
COPY "a:/fname" To "b:/fname"
COPY A2B fname1$ To fname2$
COPY B2A fname1$ To fname2$
CPU RESTART
CPU SLEEP n
CSUB name [type [, type] …] hex[[ hex […] hex [[ hex[…]   END CSUB
CTRLVAL(#ref) = nbr

D
DATA constant [, constant] ...   zie ook READ
DATE$ = "DD-MM-YY [YY]"
DATE$ = "DD/MM/YY [YY]"
DATE$ = "YYYY-MM-DD"
DATE$ = "YYYY/MM/DD"
DEFINEFONT #Nbr hex [[ hex[…] hex[[ hex[…]    END DEFINEFONT
DEVICE BITSTREAM pinno, n_transitions, array%()
DEVICE CAMERA
DEVICE GAMEPAD
DEVICE HUMID
DEVICE KEYPAD
DEVICE MOUSE
DEVICE LCD
DEVICE SERIALTX pinno, baudrate, ostring$
DEVICE SERIALRX pinno, baudrate, istring$, timeout_in_ms, status%[,nbr][,terminators$]
DEVICE WII
DEVICE WS2812
DIM [type] decl [,decl]...'decl' var[length] [type] [init] 'var'variabele naam
DIM nbr(50)
DIM INTEGER nbr%(50)
DIM FLOAT   nbr!(50)
DIM STRING  nbr$(50)
DIM name AS STRING
DIM a, b$, nbr(100), strn$(20)
DIM a(5, 5, 5), b(1000)
DIM strn$(200) LENGTH 20
DIM STRING strn(200) LENGTE 20
DIM a = 1234, b = 345
DIM STRING strn = "tekst"
DIM x%(3) = (11, 22, 33, 44)
```

```
DO <statements> LOOP
DO WHILE expression <statements> LOOP
DO <statements> LOOP UNTIL expression
DRAW3D
DRIVE drive$    drive$ kan "A:" of "B:"  DRIVE "A:/FORMAT" Alleen eerste keer.

E
EDIT
EDIT FILE
EDIT fname$
EDIT FILE fname$
ELSE
ELSE IF expression THEN
ELSEIF expression THEN
END
END [noend]
END cmd$
END CSUB
End DefineFont
END FUNCTION
ENDIF
END SELECT
END SUB
END IF
ERASE variabele [, variabele]...
ERROR [error_msg$]
Execute command$
EXIT DO
EXIT FOR
EXIT FUNCTION
EXIT SUB

F
FILES [ fspec$] [, sort]
FLASH ALL
FLASH CHAIN n
FLASH CHAIN nr [1-3]
FLASH DISK LOAD n, fname$ [,O[VERWRITE]]
FLASH ERASE ALL
FLASH ERASE nr
FLASH LIST n [, all]
FLASH LOAD n  [1-3]
FLASH OVERWRITE nr
FLASH RUN nr
FLASH SAVE nr
FLASH OVERWRITE n
FLUSH [#] fnbr
FONT [#] font - number, scaling
FOR counter = start TO finish [ STEP increment]
```

```
FRAMEBUFFER CLOSE [which]
FRAMEBUFFER CREATE
FRAMEBUFFER CREATE 2
FRAMEBUFFER COPY from, to [,b]
FRAMEBUFFER CREATE
FRAMEBUFFER LAYER
FRAMEBUFFER LAYER [colour]
FRAMEBUFFER LAYER TOP [colour]
FRAMEBUFFER MERGE [colour] [,mode] [,updaterate]
FRAMEBUFFER SYNC
FRAMEBUFFER WRITE where / where$
FRAMEBUFFER WAIT
FUNCTION xxx (arg1[,arg2, …])[AS <type>}<statements> xxx = <return value>
                              END FUNCTION
G
GAMEPAD COLOUR channel, colour
GAMEPAD HAPTIC channel left, right
GAMEPAD INTERRUPT ENABLE channel, int [,mask]
GAMEPAD INTERRUPT DISABLE channel
GOSUB          U mag niet in of uit een functie springen met commando's zoals  GOSUB,GOTO
GOTO target    Dit zal ongedefinieerde bijwerkingen hebben,waaronder je dag te verpesten.
GUI AREA #ref, startX, startY, width, height
GUI BCOLOUR colour, #ref1 [, #ref2, #ref3, enz.]
GUI BARGAUGE #ref, StartX, StartY, width, height, FColour, BColour, min, max,
                   c1, ta, c2, tb, c3, tc, c4
GUI BITMAP x, y, bits [, width] [, height] [, scale] [, c] [, bc]
GUI BUTTON #ref, caption$, startX, startY, width, height [, FColour] [,BColour]
GUI CALIBRATE
GUI CALIBRATE a, b, c, d, e
GUI CAPTION #ref, text$, startX, startY [, align$] [, FColour][, BColour]
GUI CHECKBOX #ref, caption$, startX, startY [, size] [, colour]
GUI DELETE #ref1 [, #ref2, #ref3, enz.]
GUI DELETE ALL
GUI DISABLE #ref1[,#ref2, #ref3, enz.]
GUI DISABLE ALL
GUI DISPLAYBOX #ref, startX, startY, width, height, FColour, BColour
GUI ENABLE #ref1 [, #ref2, #ref3, enz.]
GUI ENABLE ALL
GUI FCOLOUR colour, #ref1 [, #ref2, #ref3, enz.]
GUI FRAME #ref, caption$, startX, startY, width, height, colour
GUI FORMATBOX #ref, Format, startX, startY, width, height,FColour,BColour
GUI FORMATBOX ACTIVATE #ref
GUI FORMATBOX CANCEL
GUI GAUGE #ref, StartX, StartY, Radius, FColour, BColour, min, max, nbrdec,
                units$, c1, ta, c2, tb, c3, tc, c4
GUI HIDE #ref1 [, #ref2, #ref3, enz.]
GUI HIDE ALL
GUI INTERRUPT down [, up]
GUI LED #ref, caption$, centerX, centerY, radius, colour
GUI NUMBERBOX #ref, startX, startY, width, height, FColour, BColour
GUI NUMBERBOX ACTIVATE #ref
GUI NUMBERBOX CANCEL
```

```
GUI PAGE #n [, #n2, #n3, etc]
GUI RADIO #ref, caption$, centerX, centerY, radius, colour
GUI REDRAW #ref1 [, #ref2, #ref3, enz.]
GUI REDRAW ALL
GUI RESET LCDPANEL
GUI SETUP #n
GUI SHOW #ref1 [, #ref2, #ref3, enz.]
GUI SHOW ALL
GUI SPINBOX #ref, startX, startY, width, height, FColour, BColour, Step,
                  Minimum, Maximum
GUI SWITCH #ref, caption$, startX, startY, width, height, FColour,BColour
GUI TEXTBOX #ref, startX, startY, width, height, FColour, BColour
GUI TEXTBOX ACTIVATE #ref
GUI TEXTBOX CANCEL
GUI BITMAP x, y, bits [, width] [, height] [, scale] [, c] [, bc]
GUI CALIBRATE
GUI RESET LCDPANEL
GUI TEST LCDPANEL
GUI TEST TOUCH

H
HUMID pin, tvar, hvar [,DHT11]

I
I2C CLOSE
I2C OPEN speed, timeout
I2C READ addr, option, rcvlen, rcvbuf
I2C SLAVE
I2C WRITE addr, option, sendlen, senddata [, senddata ....]
I2C2 CLOSE
I2C2 OPEN speed, timeout
I2C2 READ addr, option, rcvlen, rcvbuf
I2C2 SLAVE
I2C2 WRITE addr, option, sendlen, senddata [, senddata ....]
IF expr THEN stmt [: stmt]
IF expr THEN stmt ELSE stmt
IF expression THEN <statements> [ELSEIF expression THEN <statements>]
IF expression THEN <statements> [ELSE <statements>] ENDIF
INC var [, increment]
INPUT ["prompt$";] var1 [, var2[, var3[, enz.]]]
INPUT #nbr, list of variables
INTERRUPT [myint]
IR dev, key, int
IR CLOSE
IR SEND pin, dev, key
IRETURN (Retourneert van een onderbreking)

K
KEYPAD var, int, r1, r2, r3, r4, c1, c2, c3 [, c4]
KEYPAD CLOSE
KILL file$ [,all]
```

```
L
LCD INIT d4, d5, d6, d7, rs, en
LCD line, pos, text$
LCD CLEAR
LCD CLOSE
LCD CMD d1 [, d2 [, etc]]
LCD DATA d1 [, d2 [, etc]]
LET variable = expression
LIBRARY DELETE
LIBRARY DISK SAVE fname$
LIBRARY DISK LOAD fname$
LIBRARY LIST
LIBRARY LIST ALL
LIBRARY SAVE
LINE  AA x1, y1, x2, y2 [, LW [, C]]
LINE GRAPH x(), y(), colour
LINE INPUT [prompt$,] string-variabele$
LINE INPUT #nbr, string-variabele$
LINE PLOT ydata([,nbr] [,xstart] [,xinc] [,ystart] [,yinc] [,colour])
LINE x1, y1, x2, y2 [, LW [, C]]
LIST ALL [fname$]
LIST COMMANDS
LIST [fnaam$]
LIST FUNCTIONS
LOAD file$ [, R] LOAD "FILE$" of LOAD "A:/FILE$" of LOAD "B:/PROG/FILE$"
LOAD IMAGE file$ [, x] [, y]
LOAD JPG file$ [, x] [, y]
LOAD PNG fname$ [, x] [, y] [,transparent] [,alphacut]
LOCAL variabele [, variabeles]
LONGSTRING
LONGSTRING AES128 ENCRYPT/DECRYPT CBC/ECB/CTR key$/key[!/%](),in%(),out%()[,iv$/iv[!/%]()]
LONGSTRING APPEND array%(), string$
LONGSTRING BASE64 ENCODE | DECODE in%(), out%()
LONGSTRING CLEAR array%()
LONGSTRING COPY dest%(), src%()
LONGSTRING CONCAT dest%(), src%()
LONGSTRING LCASE array%()
LONGSTRING LEFT dest%(), src%(), nbr
LONGSTRING LOAD array%(), nbr, string$
LONGSTRING MID dest%(), src%(), start, nbr
LONGSTRING PRINT [#n,] src%()
LONGSTRING REPLACE array%(), string$, start
LONGSTRING RESIZE addr%(), nbr
LONGSTRING RIGHT dest%(), src%(), nbr
LONGSTRING SETBYTE addr%(), nbr, data
LONGSTRING TRIM array%(), nbr
LONGSTRING UCASE array%()
LOOP [UNTIL expression]
```

```
M
MAP                                          HDMI VERSIE RP2350
MAP(n) = rgb%
MAP MAXIMITE
MAP GREYSCALE
MAP SET
MAP RESET
MAP                                          VGA VERSIE
MAP(n) = rgb%
MAP MAXIMITE
MAP SET
MAP RESET
MATH ADD in(), num, out()
MATH AES128 ENCRYPT/DECRYPT CBC/ECB/CTR key$/key(), in$/in(), out$/out([,iv$/iv()]
MATH C_MULT a%(),a%(),a%()
MATH C_ADD array1%(), array2%(), array3%()
MATH C_SUB array1%(), array2%(), array3%()
MATH C_MUL array1%(), array2%(), array3%()
MATH C_DIV array1%(), array2%(), array3%()
MATH C_ADD array1!(), array2!(), array3!()
MATH C_SUB array1!(), array2!(), array3!()
MATH C_MUL array1!(), array2!(), array3!()
MATH C_DIV array1!(), array2!(), array3!()
MATH FFT INVERSE FFTarray!(), signalarray!()
MATH FFT MAGNITUDE signalarray!(), magnitudearray!()
MATH FFT PHASE  signalarray!(), phasearray!()
MATH FFT signalarray!(), FFTarray!()
MATH INSERT targetarray(), [d1] [, d2] [, d3] [, d4][, d5], sourcearray()
MATH INTERPOLATE in1(), in2(), ratio, out()
MATH M_INVERSE array!(), inversearray!()
MATH M_MULT in1(), in2(), out()
MATH M_PRINT array()
MATH M_TRANSPOSE in(), out()
MATH PID INIT channel, pid_params!(), callback
MATH PID START channel
MATH PID STOP channel
MATH POWER inarray(), power, outarray()
MATH Q_CREATE theta, x, y, z, outRQ()
MATH Q_EULER yaw, pitch, roll, outRQ()
MATH Q_INVERT inQ(), outQ()
MATH Q_MULT nQ1(), iinQ2(), outQ()
MATH Q_ROTATE, RQ(), inVQ(), outVQ()
MATH Q_VECTOR x, y, z, outVQ()
MATH(RAND)
MATH RANDOMIZE [n]
MATH SCALE in(), scale, out()
MATH SENSORFUSION type ax,ay,az,gx,gy,gz,mx,my,mz,pitch,roll,yaw [,p1][,p2]
MATH SET nbr, array()
MATH SHIFT inarray%(), nbr, outarray%() [,U]
MATH SLICE sourcearray(), [d1][, d2][, d3][, d4][,d5], destinationarray()
MATH V_CROSS inV1(), inV2(), outV()
MATH V_MULT matrix(), inV(), outV()
MATH V_NORMALISE inV(), outV()
MATH V_PRINT array()
MATH V_ROTATE x, y, a, xin(), yin(), xout(), yout()
```

```
MATH WINDOW in(), minout, maxout, out() [,minin, maxin]
MEMORY
MEMORY COPY FLOAT bron adres, bestemmingsadres, aantal plunderingen
 [, bronincrement][, bestemminginverhoging]
MEMORY COPY INTEGER bron adres, bestemmingsadres, aantal gehele getallen
 [, bronincrement][, bestemminginverhoging]
MEMORY COPY sourceaddress, destinationaddress, numberofbytes [,sourceincrement]
 [, destination increment]
MEMORY COPY INTEGER sourceaddress, destinationaddress, numberofintegers
 [, sourceincrement] [,destination increment]
MEMORY COPY FLOAT bron adres, bestemmingsadres, aantal gehele getallen
 [, bronincrement] [,bestemmingin verhoging]
MEMORY INPUT[#]fnbr, nbr, address% |array()
MEMORY PACK
MEMORY PACK inarray%(),array%(), 4096, 16
MEMORY PACK source%() | sourceaddress%, dest%() | destaddress%, number, size
MEMORY PRINT #]fnbr, nbr, address%/ array()
MEMORY SET address, byte, numberofbytes
MEMORY SET BYTE address, byte, numberofbytes
MEMORY SET FLOAT address, floatingvalue, numberoffloats [, increment]
MEMORY SET INTEGER address, integervalue, numberofintegers [, increment]
MEMORY SET SHORT address, short, numberofshorts
MEMORY SET WORD address, word, numberofwords
MEMORY UNPACK
MEMORY UNPACK source() | sourceaddress%, dest() | destaddress%, number, size
MID$(str$, start [, num]) = str2$
MKDIR dir$  MKDIR "dir$"  CHDIR "dir$"  NAME "dir$"  RMDIR "dir$"
MODE 1   640 x 480 x 2     kleuren (monochroom). ALLEEN VGA    VERSIE
MODE 2   320 x 240 x 16    kleuren.
MODE 3   640 x 480 x 16    kleuren.              ALLEEN RP2350 VERSIE
MODE n  OPTION RESOLUTION 640 x 480              ALLEEN HDMI   VERSIE
MODE 1   640 x 480 x 2     kleuren (monochroom).
MODE 2   320 x 240 x 16    kleuren.
MODE 3   640 x 480 x 16    kleuren.
MODE 4   320 x 240 x 32768 kleuren.
MODE 5   320 x 240 x 256   kleuren.
MODE n  OPTION RESOLUTION 1280x720               ALLEEN HDMI   VERSIE
MODE 1  1280 x 720 x 2     kleuren (monochroom).
MODE 2   320 x 180 x 16    kleuren.
MODE 3   640 x 360 x 16    kleuren.
MODE 5   320 x 180 x 256   kleuren.
MODE n  OPTION RESOLUTION 1024x768               ALLEEN HDMI   VERSIE
MODE 1  1024 x 768 x 2     kleuren (monochroom).
MODE 2   256 x 192 x 16    kleuren.
MODE 3   512 x 384 x 16    kleuren.
MODE 5   256 x 192 x 256   kleuren.
MOUSE
MOUSE INTERRUPT ENABLE channel, int
MOUSE INTERRUPT DISABLE channel
MOUSE SET channel, x-coord, y-coord [, wheel-count]
MOUSE OPEN channel, CLKpin, DATApin
MOUSE CLOSE channel

N
NEW
NEXT [tegenvariabele] [, tegenvariabele], enz
```

```
O
ON ERROR ABORT
ON ERROR CLEAR
ON ERROR IGNORE
ON ERROR SKIP [nn]
ON KEY ASCIIcode, target
ON KEY target
ON PS2 target
ONEWIRE READ pin, flag, lengte, data [, data ...]
ONEWIRE RESET pin
ONEWIRE WRITE pin, flag, lengte, data [, data ...]
OPEN comspec$ AS [#] fnbr
OPEN compspec$ AS GPS [, timezone_offset][, monitor]
OPEN fname$ FOR MODE AS [#] fnbr  OPEN "B:/mydir/myfile.dat"
OPTION ANGLE RADIANS | DEGREES
OPTION FAST AUDIO ON/OFF
OPTION AUDIO PWMnApin, PWMnBpin
OPTION AUDIO DISABLE
OPTION AUDIO SPI CSpin, LDACpin, SCKpin, SDIpin
OPTION AUDIO DISABLE
OPTION AUDIO SPI CSpin,CLKpin,MOSIpin
OPTION AUDIO VS1053 CLKpin, MOSIpin, MISOpin, XCSpin, XDCSpin, DREQpin, XRSTpin
OPTION AUDIO SPI GP0, GP1, GP2, GP3
OPTION AUDIO SPI CSpin, CLKpin, SDIpin
OPTION AUTOREFRESH ON | OFF
OPTION AUTORUN ON [,NORESET]
OPTION AUTORUN n [,NORESET]
OPTION AUTORUN OFF
OPTION BASE 0 | 1
OPTION BAUDRATE nn
OPTION BREAK nn
OPTION CASE LOWER | UPPER | TITLE
OPTION COLOUR VGA ON | OFF
OPTION COLOURCODE ON | OFF
OPTION CONSOLE output
OPTION CPUSPEED speed (kHz)
OPTION COUNT pin1, pin2, pin3, pin4
OPTION DEFAULT FLOAT | INTEGER | STRING | NONE
OPTION DEFAULT COLOURS foreground [,background]
OPTION DEFAULT MODE n
OPTION DISK SAVE fname$   OPTION DISK SAVE "RP2350A-V6.00.01.opt"
OPTION DISK LOAD fname$
OPTION DISPLAY lines [, chars]
OPTION ESCAPE
OPTION EXPLICIT
OPTION FAST AUDIO ON | OFF
OPTION FNKey string$
OPTION HDMI PINS clockpositivepin, d0positivepin, d1positivepin, d2positivepin
OPTION GUI CONTROLS NbrOfGUIControls
OPTION HEARTBEAT ON | OFF
OPTION KEYBOARD nn [, capslock] [, numlock] [repeatstart] [repeatrate]
OPTION KEYBOARD DISABLE
OPTION KEYBOARD I2C
```

```
OPTION KEYBOARD PINS clockpin, datapin
OPTION KEYBOARD REPEAT repeatstart,repeatrate      ALLEEN USB-TOETSENBORD
OPTION LCDPANEL                                    GEEN VGA- OF HDMI-VERSIES
OPTION LCDPANEL CONSOLE [font[, fc[, bc backlight]]] [,NOSCROLL]
OPTION LCDPANEL VIRTUAL_C
OPTION LCDPANEL VIRTUAL_M
OPTION LCDPANEL DISABLE
OPTION LCDPANEL NOCONSOLE
OPTION LCDPANEL options
OPTION LCDPANEL USER hres, vres
OPTION LCD320 ON | OFF
OPTION LEGACY ON | OFF
OPTION LIST
OPTION MILLISECONDS ON | OFF
OPTION MODBUFF ENABLE | DISABLE [sizeinK]
OPTION MOUSE CLKpin, DATApin
OPTION PICO ON | OFF
OPTION PIN nbr
OPTION PLATFORM name$
OPTION PS2 PINS clockpin, datapin
OPTION PSRAM PIN n
OPTION PSRAM DISABLE
OPTION POWER PFM | PWM
OPTION RESET               Ook bibliotheek en Flash Slot 4 terug naar normaal.
OPTION RESET cfg
OPTION RESET LIST
OPTION RESET OLIMEX
OPTION RESOLUTION nn [,cpuspeedinKhz]
OPTION RESOLUTION 640x480  1280x720  1024x768  1024x600  1024x768
OPTION RESOLUTION 640  1280  1024
OPTION RTC AUTO ENABLE | DISABLE
OPTION SDCARD CSpin [, CLK-pin, MOSI-pin, MISO-pin]
OPTION SDCARD CSpin, CLKpin, MOSIpin, MISOpin
OPTION SDCARD DISABLE
OPTION SDCARD COMBINED CS
OPTION SERIAL CONSOLE DISABLE
OPTION SERIAL CONSOLE TXpin, RXpin [,B]
OPTION SERIAL CONSOLE uartapin, uartbpin,
OPTION SYSTEM I2C sdapin, sclpin [,SLOW / FAST]
OPTION SYSTEM I2C DISABLE
OPTION SYSTEM SPI CLKpin, MOSIpin, MISOpin
OPTION SYSTEM SPI DISABLE
OPTION TAB 2 | 3 | 4 | 8
OPTION TCP SERVER PORT n
OPTION TELNET CONSOLE OFF | ONLY | ON
OPTION TFTP OFF | ON
OPTION TOUCH T_CS pin, T_IRQ pin [, Beep]
OPTION TOUCH DISABLE
OPTION UDP SERVER PORT n
OPTION VGA PINS HSYNCpin, BLUEpin
OPTION VCC Voltage
OPTION WEB MESSAGES ON | OFF
OPTION WIFI ssid$, passwd$, [name$] [,ipaddress$, mask$, gateway$]
```

```
P
PAUSE delay
PIN(pin) = value
PIO ASSEMBLE pio, linedata$
PIO ASSEMBLE pionum,".wrap"
PIO ASSEMBLE pionum,".wrap target"
PIO CLEAR pio nr (nr 0 of 1)
PIO DMA RX pio, sm, nbr, data%()[,completioninterrupt][,transfersize [,loopbackcount]
PIO DMA RX OFF
PIO DMA TX pio, sm, nbr, data%()[,completioninterrupt][,transfersize][,loopbackcount]
PIO DMA TX OFF
PIO (EXECCTRL jmp_pin ,wrap_target , wrap [,side_pindir] [,side_en])
PIO EXECUTE pio, statestate_machine, instructie%
PIO INIT
PIO INIT MACHINE pio%, statemachine%, kloksnelheid [, pinctrl] [, execctrl]
                       [, shiftctrl] [, startinstruc]
PIO INTERRUPT pio, sm [,RXinterrupt] [,TXinterrupt]
PIO MAKE RING BUFFER ivar%, size
PIO PROGRAM LINE pio, line, instruction
PIO PROGRAM pio, array%()
PIO READ pio, state_machine, aantal, data%[()]
PIO START pio, statemachine
PIO STOP pio, statemachine
PIO WRITE pio, state_machine, telling, data0[, data1....]
PIO WRITEFIFO a, b, v, d
PIXEL x, y [, c]
PLAY
PLAY FLAC file$ [, interrupt]
PLAY LOAD SOUND array%()
PLAY MODFILE file$ [, interrupt]
PLAY MODSAMPLE Samplenum, channel [,volume]
PLAY MP3  file$ [, interrupt]
PLAY NEXT
PLAY PAUSE
PLAY PREVIOUS
PLAY RESUME
PLAY SOUND
PLAY SOUND chan, M, golfvorm, freq [,vol]
PLAY SOUND chan, L/R/B, U, freq
PLAY SOUND LOAD array%()
PLAY SOUND soundno, channelno, type [, frequentie] [, volume]
PLAY STOP
PLAY TONE left [, right[, dur]] [,interrupt]]]
PLAY VOLUME left, right
PLAY WAV  file$ [, interrupt]
PLAY WAV "A:mywav.wav"
PLAY MP3 file$ [, interrupt]     ***** VS1053 PLAY commando's ****
PLAY HALT
PLAY CONTINUE track$
PLAY MIDIFILE file$ [, interrupt]
PLAY MIDI
PLAY MIDI CMD cmd%, data1%, data2%
```

```
PLAY MIDI TEST n
PLAY NOTE ON channel%, note%, velocity%
PLAY NOTE OFF channel%, note% [,velocity%]
PLAY STREAM buffer%(), readpointer%, writepointer%
POKE BYTE addr%, byte
POKE DISPLAY commando [, data1] [, data2] [, datan]
POKE DISPLAY HRES n
POKE DISPLAY VRES n
POKE FLOAT addr%, float!
POKE INTEGER addr%, int%
POKE SHORT addr%, short%
POKE WORD addr%, word%
POKE VAR var, offset, byte
POKE VARTBL, offset, byte
POLYGON n, xarray%(), yarray%() [, borderkleur] [, vulkleur]
POLYGON n(), xarray%(), yarray%() [, randkleur()] [, vulkleur()]
POLYGON n(), xarray%(), yarray%() [, randkleur] [, vulkleur()]
PORT(start, nbr [, start, nbr]...) = waarde
PRINT #nbr, expressie [[,;] uitdrukking] … enz of ?
PRINT #GPS, expressie [[,;] uitdrukking] … enz
PRINT @(x [, y]) uitdrukking
PRINT @(x, [y], m) uitdrukking
PRINT uitdrukking [[,;] uitdrukking] … enz
PULSE pin, breedte
PWM AUDIO uitvoer
PWM kanaal,frequentie, [dutyA] [,dutyB] [,phase] [,defer]
PWM kanaal, OFF
PWM SYNC s0 [,s1] [,s2] [,s3][,s4] [,s5] [,s6] [,s7]

R
RANDOMIZE nbr
RBOX x, y, w, h [, r] [, c] [, fill]
READ variabele [, variabele]...
READ SAVE
READ RESTORE
REFRESH
REM String
RENAME old$ AS new$
RESTORE [line]
RETURN
RMDIR dir$
RTC GETTIME
RTC SETTIME jaar, maand, dag, uur, minuut, seconde
RTC SETREG reg, waarde
RTC GETREG reg, var
RUN
RUN [file$][, cmdline$]
RUN [bestand$]
```

```
S
SAVE COMPRESSED IMAGE file$ [,x, y, w, h]
SAVE file$
SAVE "FILE$" of SAVE "A:/FILE$" of SAVE "B:/PROGRAMMA/FILE$"  B: = SD kaart
SAVE IMAGE
SAVE IMAGE file$ [, x, y, w, h]
SEEK [#] fnbr, pos
SELECT CASE nbr% CASE testexp [[, testexp] …] <statements> <statements>
       CASE ELSE <statements> <statements> END SELECT
SERVO channel [positionA [,positionB]
SERVO channel,, positionB
SETPIN GP25, DOUT | HEARTBEAT      OPTION HEARTBEAT ON | OFF
SETTICK PAUSE, doel [, nbr]
SETTICK period, target [, nbr]
SETPIN pin, cfg [, optie]        cfg: OFF, AIN, DIN, FIN, PIN, CIN, DOUT
SETPIN pin, cfg, doel [, optie]  cfg: OFF, INTH, INTL, INTB
SETPIN GP1, FFIN [,gate]          ALLEEN RP2350 snelle frequentie ingang
SETPIN pin, IR
SETPIN pin, PIOn
SETPIN pin, PWM [nx ]
SETPIN pin, cfg, target [, optie]
SETPIN p1 [, p2[, p3]], device
SETTICK period, target [, nbr]
SETTICK PAUSE, doel [, nbr]
SETTICK RESUME, doel [, nbr]
SETPIN rx, tx, COM1
SETPIN rx, tx, COM2
SETPIN rx, tx, clk, SPI
SETPIN rx, tx, clk, SPI2
SETPIN sda, scl, I2C
SETPIN sda, scl, I2C2
SORT array() [, indexarray()] [, vlaggen] [, startpositie]
             [, elemententosort]
SPI CLOSE
SPI OPEN snelheid, mode, bits   SPI 1 OPEN 1000000, 3
SPI READ  nbr, array()
SPI WRITE nbr, array()
SPI WRITE nbr, data1, data2, data3, … enz
SPI WRITE nbr, string$
SPI2 CLOSE
SPI2 OPEN snelheid, mode, bits   SPI 1 OPEN 1000000, 3
SPI2 READ  nbr, array()
SPI2 WRITE nbr, array()
SPI2 WRITE nbr, data1, data2, data3, … enz
SPI2 WRITE nbr, string$
SPRITE CLOSE [#]n
SPRITE CLOSE ALL
SPRITE COPY [#]n, [#]m, nbr
SPRITE HIDE [#]n
SPRITE HIDE ALL
SPRITE RESTORE
SPRITE HIDE SAFE [#]n
SPRITE INTERRUPT Sub
```

```
SPRITE READ [#]b, x, y, w, h
SPRITE WRITE [#]b, x, y
SPRITE LOAD fname$ [,start_sprite_number] [,mode]
SPRITE LOADARRAY [#]n, w, h, array%()
SPRITE LOADBMP [#]b, fname$ [,x] [,y] [,w] [,h]
SPRITE LOADPNG [#]b, fname$ [,transparent] [,alphacut]
SPRITE MOVE
SPRITE NEXT [#]n, x, y
SPRITE SCROLL x, y [,col]
SPRITE SET TRANSPARENT n
SPRITE SHOW SAFE [#]n, x,y, layer [,orientation] [,ontop]
SPRITE SWAP [#]n1, [#]n2 [,orientation]
STATIC variabele [, variabelen]
SUB xxx (arg1 [, arg2,…]) <statements> <statements> END SUB
SYNC time% [,periode]
SYNC

T
TEMPR START pin [, precisie]
TEXT x, y, string$ [, alignment$] [, font] [, scale] Numerieke [, c][,bc]
TILE x, y [,foreground] [,background] [,nbr_tiles_wide] [,nbr_tiles_high]
TILE HEIGHT n
TIME$ = "UU:MM:SS"
TIME$ = "UU:MM"
TIME$ = "HH"
TIMER = msec
TRACE ON | OFF
TRACE LIST nn
TRIANGLE RESTORE [#]n
TRIANGLE SAVE [#]n, x1,y1,x2,y2,x3,y3
TRIANGLE X1, Y1, X2, Y2, X3, Y3 [, C [, FILL]]

U
UPDATE FIRMWARE

V
VAR
VAR CLEAR
VAR RESTORE
VAR SAVE var [, var] ...
```

```
W
WATCHDOG HW timeout
WATCHDOG HW OFF
WATCHDOG OFF
WATCHDOG timeout
WII [CLASSIC] OPEN [,interrupt]
WII [CLASSIC] CLOSE
WII NUNCHUCK OPEN [,interrupt] [myint]
WII NUNCHUCK CLOSE
WEB
WEB CONNECT [ssid$, passwd$, [name$] [,ipaddress$, mask$, gateway$]]
WEB MQTT CONNECT addr$, port, user$, passwd$ [, interrupt]
WEB MQTT PUBLISH topic$, msg$, [,qos] [,retain]
WEB MQTT SUBSCRIBE topic$ [,qos]
WEB MQTT UNSUBSCRIBE topic$
WEB MQTT CLOSE
WEB NTP [timeoffset [, NTPserver$]] [, timeout]]]
WEB OPEN TCP CLIENT address$, port
WEB OPEN TCP STREAM address$, port
WEB SCAN [array%()]
WEB TCP CLIENT REQUEST request$, buff%() [,timeout]
WEB TCP CLIENT STREAM command$, buffer%(), readpointer%, writepointer%
WEB CLOSE TCP CLIENT
WEB TCP INTERRUPT InterruptSub
WEB TCP READ cb%, buff%()
WEB TCP SEND cb%, data%()
WEB TCP CLOSE cb%
WEB TRANSMIT CODE cb%, nnn%
WEB TRANSMIT FILE cb%, filename$, content-type$
WEB TRANSMIT PAGE cb%, filename$[,buffersize]
WEB UDP INTERRUPT intname
WEB UDP SEND addr$, port, data$
WS2812 type, pin, nbr, value%[()]
WHILE

X
XMODEM CRUNCH
XMODEM RECEIVE
XMODEM RECEIVE file$
XMODEM R "test.jpg"
XMODEM SEND
XMODEM SEND file$
```

FUNCTIES, STRINGS en OPERATORS

Rekenkundige operator
^
*
+
-
/
\
MOD

Logische operator en Tekenreeks
<
>
<<
<=
=<
>=
=>
<>
=
>>
<>
NOT
INV
AND
OR
XOR
+

@(

A
ABS(getal)
ACOS(getal)
AND
AS
ASC(string$)
ASIN(getal)
ATAN2(y, x)
ATN(getal)

B
BIN$(getal [, chars])
BIN2STR$(type, waarde [, BIG])
BOUND(array() [, dimensie]

C
CALL(userfunname$, [, userfunparameters,....])
CHOICE(voorwaarde, ExpressieIfTrue, ExpressieIfFalse)
CHR$(number)
CINT(getal)
COS(getal)
CTRLVAL(#ref)
CWD$

```
D
DATE$
DATETIME$(n)
DAY$(date$)
DEG(radialen)
DEVICE(GAMEPAD channel, funct)
DEVICE(MOUSE channel, funct)
DEVICE(WII [CLASSIC] funct)
DEVICE(WII NUNCHUCK funct)
DEVICE(NUNCHUCK x)
DIR$(fspec, type)
DIR$(fspec)
DIR$( )
DISTANCE(trigger; echo)
DISTANCE(trigger-echo)

E
ELSE
EOF( [#] fnbr)
EPOCH(DATETIME$)
EVAL(string$)
EXP(getal)

F
Field$(
FIX(getal)
FOR
FORMAT$(nbr [, fmt$])

G
GETSCANLINE
GOSUB
GOTO
GPS()
GPS(ALTITUDE)
GPS(DATE)
GPS(DOP)
GPS(FIX)
GPS(GEOID)
GPS(LATITUDE)
GPS(LONGITUDE)
GPS(SATELLITES)
GPS(SPEED)
GPS(TIME)
GPS(TRACK)
GPS(VALID)

H
HEX$(number [, chars])

I
INKEY$
INPUT$(nbr, [#]fnbr)
INSTR( [startpositie,] string gezocht$, stringpatroon$ [,size])
INT(getal)
INV
```

```
J
JSON$(array%(), string$)

K
KEYDOWN(n)

L
LCASE$(string$)
LCOMPARE(matrix1%(), matrix2%())
LCOMPARE(array1%(), array2%())
LEFT$(string$, nbr)
LEN(string$)
LGETBYTE(array%(), n)
LGETSTR$(matrix%(), start, lengte)
LINSTR(matrix%(), zoek$ [, start])
LLEN(array%())
LOC([#]fnbr)
LOF([#]fnbr)
LOG(getal)

M
MAP( n )

- Eenvoudige functies -
MATH(ATAN3 x, y)
MATH(COSH a)
MATH(LOG10 a)
MATH(RAND)
MATH(SINH a)
MATH(TANH a)
MATH(CRCn data [,length] [,polynome] [,startmask] [,endmask][,reverseIn][,reverseOut]
MATH(CRC8  array(), length, [polynoom,][startmask,][endmask,][reverseIn,][reverseOut]
MATH(CRC12 array(), lengte, [polynoom,][startmask,][endmask,][reverseIn,][reverseOut]
MATH(CRC16 array(), lengte, [polynoom,][startmask,][endmask,][reverseIn,][reverseOut]
MATH(CRC32 array(), lengte, [polynoom,][startmask,][endmask,][reverseIn,][reverseOut]

- Eenvoudige statistieken -
MATH(CHI a())
MATH(CHI_p a())
MATH(CROSSING array() [,level] [,direction]
MATH(CORREL a(), b())
MATH(MAX a() [, index%])
MATH(MEAN a())
MATH(MEDIAN a())
MATH(MIN a(), [index%])
MATH(SD a())
MATH(SUM a())

- Vector rekenkunde -
MATH(MAGNITUDE v())
MATH(DOTPRODUCT v1(), v2())

- Matrix Rekenen -
MATH(M_DETERMINANT reeks!())
```

```
- Floating returns
real!   = MATH(C_REAL  complex%)
imag!   = MATH(C_IMAG  complex%)
arg!    = MATH(C_ARG   complex%)
mod!    = MATH(C_MOD   complex%)
phase!  = MATH(C_PHASE complex%)

- Unary functions -
complex1% = MATH(C_CONJ  complex2%)
complex1% = MATH(C_SIN   complex2%)
complex1% = MATH(C_COS   complex2%)
complex1% = MATH(C_TAN   complex2%)
complex1% = MATH(C_ASIN  complex2%)
complex1% = MATH(C_ACOS  complex2%)
complex1% = MATH(C_ATAN  complex2%)
complex1% = MATH(C_SINH  complex2%)
complex1% = MATH(C_COSH  complex2%)
complex1% = MATH(C_TANH  complex2%)
complex1% = MATH(C_ASINH complex2%)
complex1% = MATH(C_ACOSH complex2%)
complex1% = MATH(C_ATANH complex2%)
complex1% = MATH(C_PROJ  complex2%)

- Creation -
complex%  = MATH(C_CPLX r!, i!)
complex%  = MATH(C_POLAR radius!, angle!)

- Basic Arithmetic -
complex1% = MATH(C_ADD complex2%, complex3%)
complex1% = MATH(C_SUB complex2%, complex3%)
complex1% = MATH(C_MUL complex2%, complex3%)
complex1% = MATH(C_DIV complex2%, complex3%)
complex1% = MATH(C_POW complex2%, complex3%)
complex1% = MATH(C_AND complex2%, complex3%)
complex1% = MATH(C_OR  complex2%, complex3%)
complex1% = MATH(C_XOR complex2%, complex3%)

MATH(PID channel, setpoint!, measurement))
MATH(BASE64 ENCODE | DECODE in$ | in(), out$ | out())
MAX(arg1 [,arg2 [,...]])
MIN(arg1 [,arg2 [,...]])
MID$(string$, start)
MID$(string$, start, nbr)
MSGBOX(msg$, b1$ [, b2$ ... b4$])
MAX(arg1 [, arg2 [, …]])
MID$(string$, start)
MID$(string$, start, nbr)
MIN(arg1 [, arg2 [, …]])
```

```
MM.ADDRESS$
MM.CMDLINE$
MM.DEVICE$                      opm. Voor opvragen MM.xxx na > eerst ? of Print
MM.ERRMSG$
MM.ERRNO
MM.FONTHEIGHT
MM.FONTWIDTH
MM.HPos
MM.HRES
MM.I2C
MM.INFO$()
MM.INFO$(AUTORUN)
MM.INFO$(CPUSPEED)
MM.INFO$(CURRENT)
MM.INFO$(DISK SIZE)
MM.INFO$(DRIVE)                  Geeft A: of B: terug.
MM.INFO$(FLASH)
MM.INFO$(FREE SPACE)
MM.INFO$(IP ADDRESS)
MM.INFO$(LCDPANEL)
MM.INFO$(LINE)
MM.INFO$(MODIFIED file$)
MM.INFO$(OPTION FLASH SIZE)
MM.INFO$(PATH)
MM.INFO$(PIN pinno)
MM.INFO$(PINNO GPnn)
MM.INFO$(PLATFORM)
MM.INFO$(SDCARD)
MM.INFO$(SOUND)
MM.INFO$(SYSTEM I2C)
MM.INFO$(TOUCH)
MM.INFO()
MM.INFO(ADC)
MM.INFO(ADC DMA)
MM.INFO(BCOLOUR)
MM.INFO(BOOT COUNT)
MM.INFO(DEVICE)
MM.INFO(DISK SIZE)
MM.INFO(EXISTS FILE fname$)
MM.INFO(EXISTS DIR dname$)
MM.INFO(FCOLOUR)
MM.INFO(FILESIZE file$)
MM.INFO(FLASH)
MM.INFO(FLASH ADDRESS n)
MM.INFO(FONT)
MM.INFO(FONT ADDRESS n)
MM.INFO(FONT POINTER n)
MM.INFO(FONTHIGHT)
MM.INFO(FONTWIDTH)
MM.INFO(HEAP)
MM.INFO(HPOS)
MM.INFO(ID)
MM.INFO(LCD320)
```

```
MM.INFO(OPTION ANGLE)
MM.INFO(OPTION option)
MM.INFO(PINNO)
MM.INFO(PIO RX DMA)
MM.INFO(PIO TX DMA)
MM.INFO(PS2)
MM.INFO(SDCARD)
MM.INFO(STACK)
MM.INFO(SYSTEM HEAP)
MM.INFO(TCP PORT)
MM.INFO(TCPIP STATUS)
MM.INFO(TILE HEIGHT)
MM.INFO(TRACK)
MM.INFO(UDP PORT)
MM.INFO(UPTIME)
MM.INFO(USB n)
MM.INFO(VARCNT)
MM.INFO(VERSION)
MM.INFO(VPOS)
MM.INFO(WIFI STATUS)
MM.INFO(WRITEBUFF)
MM.HRES
MM.HPOS
MM.MESSAGE$
MM.ONEWIRE
MM.VER
MM.VPos
MM.VRes
MM.WATCHDOG
MOD
MSGBOX(msg$, b1$ [, b2$ … b4$])

N
NOT

O
OCT$(number [,chars])
OR

P
PEEK(bp n%)
PEEK(BYTE addr%)
PEEK(CFUNADDR cfun)
PEEK(FLOAT addr%)
PEEK(INTEGER addr%)
PEEK(PROGMEM, ±offset)
PEEK(SHORT addr%)
PEEK(sp n%)
PEEK(VARADDR var)
PEEK(VAR var, ±offset)
PEEK(VARTBL, ±offset)
PEEK(WORD addr%)
PEEK(wp n%)
PI
PIN(BOOTSEL)
```

```
PIN(pin)
PIN(Pinno/GPno)
PIN(TEMP)
PIO(DMA RX POINTER)
PIO(DMA TX POINTER)
PIO(EXECCTRL jmp_pin, wrap_target, wrap)
PIO(EXECCTRL jmp_pin, wrap_target, wrap [,side_pindir] [,side_en])
PIO(FDEBUG pio)
PIO(FLEVEL pio)
PIO(FLEVEL pio, sm, DIR
PIO(FSTAT pio)
PIO(PINCTRL no_side_set_pins[,no_set_pins][,no_out_pins][,IN basis]
                            [,side_set_base][,set_base][,out_base])
PIO(SHIFTCTRL push_threshold[,pull_threshold][,autopush][,autopull]
                           [,in_shiftdir][,out_shiftdir][,fjoin_rx][,fjoin_tx])
PIO(READFIFO a, b, c)
PIO(.WRAP)
PIO(.WRAP TARGET)
PIXEL(x, y [, paginanummer])
PORT(start, nbr [, start, nbr] ...)
POS
PULSIN(pin, polariteit)
PULSIN(pin, polariteit, t1)
PULSIN(pin, polariteit, t1, t2)

R
RAD(graden)
RGB(red, green, blue)
RGB(shortcut)
RIGHT$(string$, aantal tekens)
RND
RND(getal)

S
SGN(getal)
SIN(getal)
SPACE$(number)
SPI(data)
SPI2(data)
SPRITE()
SPRITE(C, [#]n)
SPRITE(C, [#]n, m)
SPRITE(D, [#]s1,[#]s2)
SPRITE(E, [#]n
SPRITE(H, [#]n)
SPRITE(L, [#]n)
SPRITE(N)
SPRITE(N, n)
SPRITE(S)
SPRITE(V, spriteno1, spriteno 2)
SPRITE(T, [#]n)
SPRITE(V, [#]so1, [#]s2)
SPRITE(W, [#]n)
SPRITE(X, [#]n)
SPRITE(Y, [#]n)
```

SQR(getal)
STEP
STR$(getal)
STR$(getal, m)
STR$(getal, m, n)
STR$(getal, m, n, c$)
STR2BIN(type, string$ [, BIG])
STRING$(nbr, ascii)
STRING$(nbr, tekenreeks$)

T
TAB(getal)
TAN(getal)
TEMPR(pin)
THEN
TIME$
TIMER
TO
TOUCH(X)
TOUCH(Y)

U
UCASE$(string$)
UNTIL

V
VAL(string$)

W
WHILE

X
XOR

ASCII tabel voor VT100 072 - 255 Font: Terminal en rest Font: Lettertype

Graden   FONT Lettertype = 176 = 100°, Terminal = 248 = 100° Chr$(248)
Kwadraat FONT Lettertype = 178 = 10², Terminal = 253 = 10² Chr$(253)
Half   FONT Lettertype = 189 = 10½, Terminal = 171 = 10½ Chr$(171)

-- Tera Term FONT: Terminal ----------				-- FONT: Lettertype ---	
Alt-255	Alt-209 ╤	Alt-163 ú	Alt-117 u	Alt-0128 €	Alt-0170 ª
Alt-254 ■	Alt-208 ╨	Alt-162 ó	Alt-116 t	Alt-0129	Alt-0171 «
Alt-253 ²	Alt-207 ╧	Alt-161 í	Alt-115 s	Alt-0130 ‚	Alt-0172 ¬
Alt-252 ⁿ	Alt-206 ╬	Alt-160 á	Alt-114 r	Alt-0131 ƒ	Alt-0173
Alt-251 √	Alt-205 ═	Alt-159 ƒ	Alt-113 q	Alt-0132 „	Alt-0174 ®
Alt-250 ·	Alt-204 ╠	Alt-158 ₧	Alt-112 p	Alt-0133 …	Alt-0175 ¯
Alt-249 ·	Alt-203 ╦	Alt-157 ¥	Alt-111 o	Alt-0134 †	Alt-0176 °
Alt-248 °	Alt-202 ╩	Alt-156 £	Alt-110 n	Alt-0135 ‡	Alt-0177 ±
Alt-247 ≈	Alt-201 ╔	Alt-155 ¢	Alt-109 m	Alt-0136 ˆ	Alt-0178 ²
Alt-246 ÷	Alt-200 ╚	Alt-154 Ü	Alt-108 l	Alt-0137 ‰	Alt-0179 ³
Alt-245 ⌡	Alt-199 ╟	Alt-153 Ö	Alt-107 k	Alt-0138 Š	Alt-0180 ´
Alt-244 ⌠	Alt-198 ╞	Alt-152 ÿ	Alt-106 j	Alt-0139 ‹	Alt-0181 »
Alt-243 ≤	Alt-197 ┼	Alt-151 ù	Alt-105 i	Alt-0140 Œ	Alt-0182 ¶
Alt-242 ≥	Alt-196 ─	Alt-150 û	Alt-104 h	Alt-0141	Alt-0183 ·
Alt-241 ±	Alt-195 ├	Alt-149 ò	Alt-103 g	Alt-0142 Ž	Alt-0184 ¸
Alt-240 ≡	Alt-194 ┬	Alt-148 ö	Alt-102 f	Alt-0143	Alt-0185 ¹
Alt-239 ∩	Alt-193 ┴	Alt-147 ô	Alt-101 e	Alt-0144	Alt-0186 º
Alt-238 ε	Alt-192 └	Alt-146 Æ	Alt-100 d	Alt-0145 '	Alt-0187 »
Alt-237 ϕ	Alt-191 ┐	Alt-145 æ	Alt-099 c	Alt-0146 '	Alt-0188 ¼
Alt-236 ∞	Alt-190 ╛	Alt-144 É	Alt-098 b	Alt-0147 “	Alt-0189 ½
Alt-235 δ	Alt-189 ╜	Alt-143 Å	Alt-097 a	Alt-0148 ”	Alt-0190 ¾
Alt-234 Ω	Alt-188 ╝	Alt-142 Ä	Alt-096 `	Alt-0149 •	Alt-0191 ¿
Alt-233 Θ	Alt-187 ╗	Alt-141 ì	Alt-095 _	Alt-0150 –	Alt-0192 À
Alt-232 Φ	Alt-186 ║	Alt-140 î	Alt-094 ^	Alt-0151 —	Alt-0193 Á
Alt-231 τ	Alt-185 ╣	Alt-139 ï	Alt-093]	Alt-0152 ˜	Alt-0194 Â
Alt-230 µ	Alt-184 ╕	Alt-138 è	Alt-092 \	Alt-0153 ™	Alt-0195 Ã
Alt-229 σ	Alt-183 ╖	Alt-137 ë	Alt-091 [	Alt-0154 š	Alt-0196 Ä
Alt-228 Σ	Alt-182 ╢	Alt-136 ê	Alt-090 Z	Alt-0155 ›	Alt-0197 Å
Alt-227 π	Alt-181 c	Alt-135 ç	Alt-089 Y	Alt-0156 œ	Alt-0198 Æ
Alt-226 Γ	Alt-180 ┤	Alt-134 å	Alt-088 X	Alt-0157 ﾘ	Alt-0199 Ç
Alt-225 ß	Alt-179 │	Alt-133 à	Alt-087 W	Alt-0158 ž	Alt-0200 È
Alt-224 α	Alt-178 ▓	Alt-132 ä	Alt-086 V	Alt-0159 Ÿ	Alt-0201 É
Alt-223 ▀	Alt-177 ▒	Alt-131 â	Alt-085 U	Alt-0160	Alt-0202 Ê
Alt-222 ▐	Alt-176 ░	Alt-130 é	Alt-084 T	Alt-0161 ¡	Alt-0203 Ë
Alt-221 ▌	Alt-175 »	Alt-129 Ç	Alt-083 S	Alt-0162 ¢	Alt-0204 Ì
Alt-220 ▄	Alt-174 «	Alt-128 Ç	Alt-082 R	Alt-0163 £	Alt-0205 Í
Alt-219 ▉	Alt-173 ¡	Alt-127 ⌂	Alt-081 Q	Alt-0164 ¤	Alt-0206 Î
Alt-218 ┌	Alt-172 ¼	Alt-126 ~	Alt-080 P	Alt-0165 ¥	Alt-0207 Ï
Alt-217 ┘	Alt-171 ½	Alt-125 }	Alt-079 O	Alt-0166 ¦	Alt-0208 Ð
Alt-216 ╪	Alt-170 ¬	Alt-124 |	Alt-078 N	Alt-0167	Alt-0209 Ñ
Alt-215 ╫	Alt-169 ⌐	Alt-123 {	Alt-077 M	Alt-0168 ¨	Alt-0210 Ò
Alt-214 ╓	Alt-168 ¿	Alt-122 z	Alt-076 L	Alt-0169 ©	
Alt-213 ╒	Alt-167 º	Alt-121 y	Alt-075 K		
Alt-212 ╘	Alt-166 ª	Alt-120 x	Alt-074 J		
Alt-211 ╙	Alt-165 Ñ	Alt-119 w	Alt-073 I		
Alt-210 ╥	Alt-164 ñ	Alt-118 v	Alt-072 H		

ASCII tabel voor TERA TERM VT100 0 - 47 Font: Terminal Chr$(42) = *

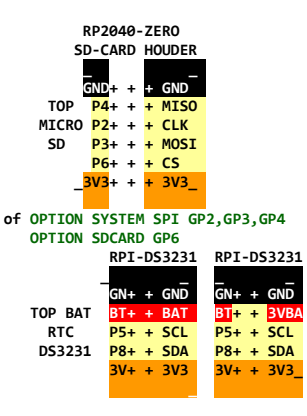
ASCII tabel

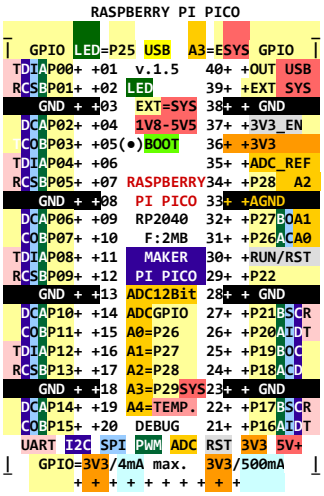
--- Tera Term FONT: Terminal ---

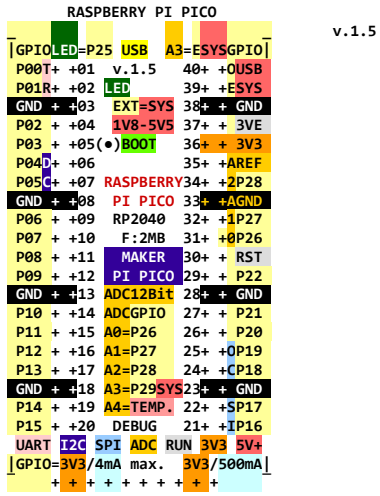
ASCII	HEX	OCT	HTML	FUNCTIE
0	00	000	NULL	
1	01	001	SOH	(Start of Header)
2	02	002	STX	(Start of Text)
3	03	003	ETX	(End of Text)
4	04	004	EOT	(End of Transmission)
5	05	005	ENQ	(Enquiry)
6	06	006	ACK	(Acknowledgement)
7	07	007	BEL	(Bell)
8	08	010	BS	(Backspace)
9	09	011	HT	(Horizontal Tab)
10	0a	012	LF	(Line Feed)
11	0b	013	VT	(Vertical Tab)
12	0c	014	FF	(Form Feed)
13	0d	015	CR	(Carriage Return)
14	0e	016	SO	(Shift Out)
15	0f	017	SI	(Shift In)
16	10	020	DLE	(Data Link Escape)
17	11	021	DC1	(Device Control 1)
18	12	022	DC2	(Device Control 2)
19	13	023	DC3	(Device Control 3)
20	14	024	DC4	(Device Control 4)
21	15	025	NAK	(Negative Acknowledgement)
22	16	026	SYN	(Synchronous Idle)
23	17	027	ETB	(End of Transmission Block)
24	18	030	CAN	(Cancel)
25	19	031	EM	(End of Medium)
26	1a	032	SUB	(Substitute)
27	1b	033	ESC	(Escape)
28	1c	034	FS	(File Separator)
29	1d	035	GS	(Group Separator)
30	1e	036	RS	(Record Separator)
31	1f	037	US	(Unit Separator)
32	20	040	SPACE	
33	21	041	!	
34	22	042	"	
35	23	043	#	
36	24	044	$	
37	25	045	%	
38	26	046	&	
39	27	047	'	
40	28	050	(	
41	29	051	)	
42	2a	052	*	
43	2b	053	+	
44	2c	054	,	
45	2d	055	-	
46	2e	056	.	
47	2f	057	/	

Een aantal sjablonen voor een Raspberry Pi Pico module voor op een breadboard of een MAKER PI PICO module.

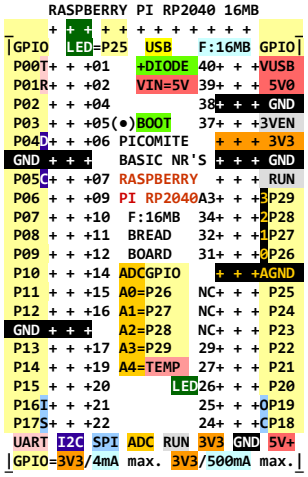
Print het vanuit **PDF** met **Adobe Acrobat** op **102%** i.v.m. de maatvoering en beplak ze aan beide kanten met blanco tape of lamineer het geheel met stugge 80 -100 micro hoezen. Snij ze met een scherp mes uit en prik de gaatjes met een priem of ruimer op een gaatjes print voor de juiste 0.1" maat en gebruik print pinnen om het sjabloon op zijn plek te houden bij het prikken of boren.

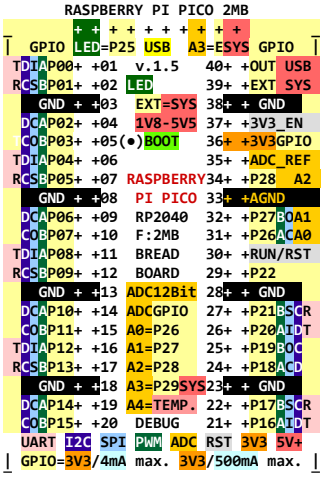




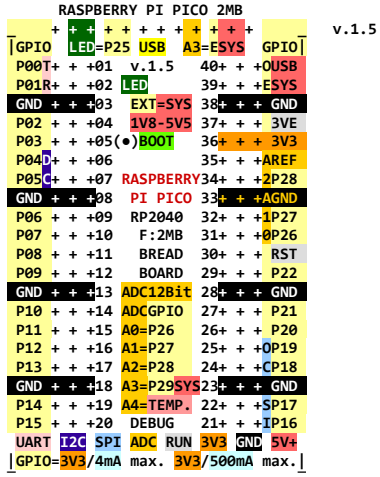


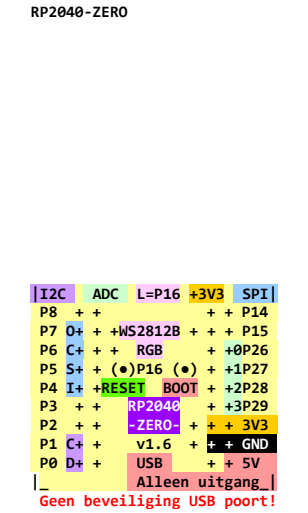
pin aantal klopt, <- afstand -> niet!

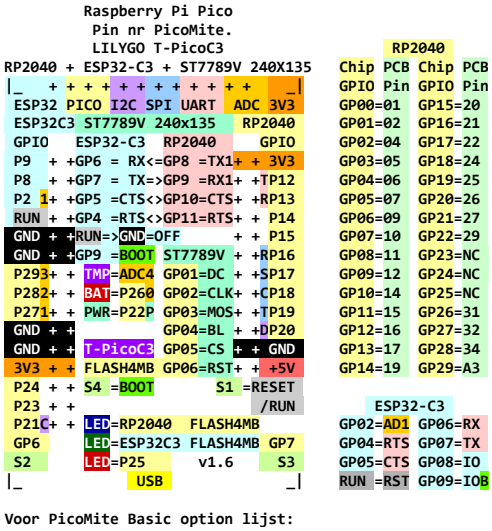


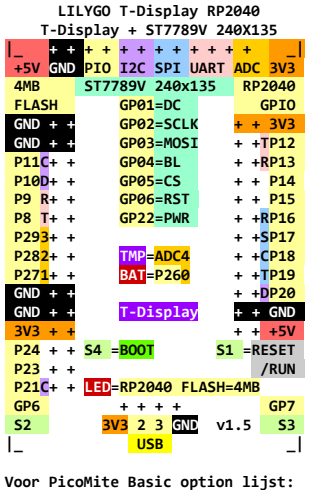


pin aantal klopt, <- afstand -> niet!









Alle hardware I/O functies RP2040 en RP2350 Chip pin nummers

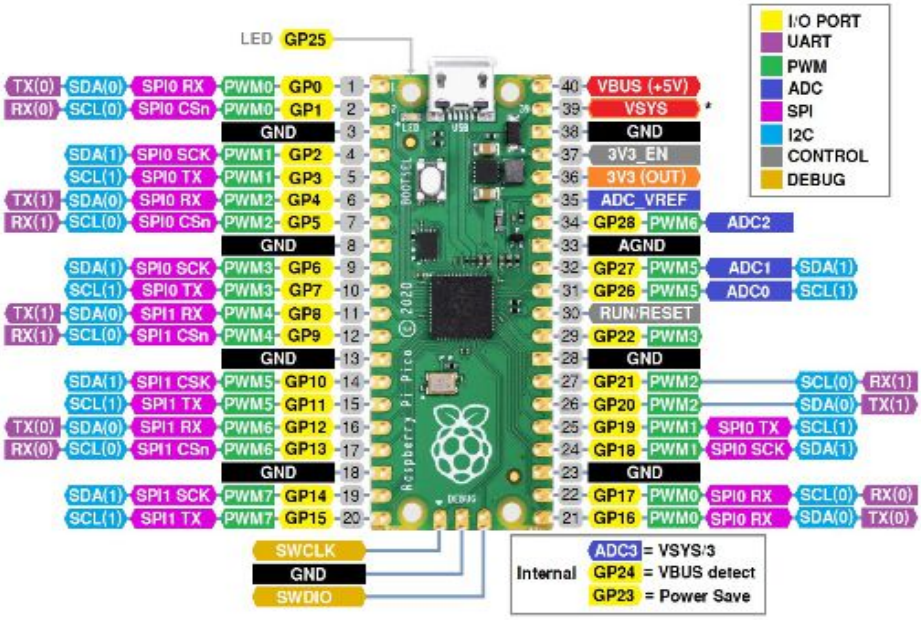
```
Serial1: RX,TX Com1
RX :GP01,GP13,GP17,GP29
TX :GP00,GP12,GP16,GP28
CTS:GP02,GP14,GP18
RTS:GP03,GP15,GP19
Serial2: RX,TX COM2
RX :GP05,GP09,GP21,GP25
TX :GP04,GP08,GP20,GP24
CTS:GP06,GP10,GP22,GP26
RTS:GP07,GP11,GP23,GP27
ADC: 12 Bit  CH:0-4
AD0:GP26  AD3:GP29=SYS
AD1:GP27  AD4:     TEMP
AD2:GP28
```

```
SPI : MISO,MOSI,CS,CLOCK
MISO:GP00,GP04,GP16,GP20
MOSI:GP03,GP07,GP19,GP23
CSn :GP01,GP05,GP17,GP21
SCK :GP02,GP06,GP18,GP22
SPI2: MISO,MOSI,CS,CLOCK
MISO:GP08,GP12,GP24,GP28
MOSI:GP11,GP15,GP27
CSn :GP09,GP13,GP25,GP29
SCK :GP10,GP14,GP26
GPIO:GP00-GP29
PIO0:GP00-GP29
PIO1:GP00-GP29
MISO:MasterInpuSlaveOutp
MOSI:MasterOutpSlaveInpu
```

```
I2C  : SDA,SCL
SDA  :GP0,GP4,GP08,GP12,GP16,GP20,GP28
SCL  :GP1,GP5,GP09,GP13,GP17,GP21,GP29
I2C2 : SDA,SCL
SDA  :GP2,GP6,GP10,GP14,GP18,GP22,GP26
SCL  :GP3,GP7,GP11,GP15,GP19,GP23,GP27
PWM  : PWM0A-PWM7A en PWM0B-PWM7B
PWM0A:GP00,GP16    PWM0B:GP01,GP17
PWM1A:GP02,GP18    PWM1B:GP03,GP19
PWM2A:GP04,GP20    PWM2B:GP05,GP21
PWM3A:GP06,GP22    PWM3B:GP07,GP23
PWM4A:GP08,GP24    PWM4B:GP09,GP25
PWM5A:GP10,GP26    PWM5B:GP11,GP27
PWM6A:GP12,GP28    PWM6B:GP13,GP29
PWM7A:GP14         PWM7B:GP15
```

Pin nummers Raspberry Pi Pico en Chip pinnen

Chip Pin	PCB Pin	Chip Pin	PCB Pin	
GP00 =	01	GP15 =	20	
GP01 =	02	GP16 =	21	
GP02 =	04	GP17 =	22	
GP03 =	05	GP18 =	24	
GP04 =	06	GP19 =	25	
GP05 =	07	GP20 =	26	
GP06 =	09	GP21 =	27	
GP07 =	10	GP22 =	29	
GP08 =	11	GP23 =	41	DOUT
GP09 =	12	GP24 =	42	DIN
GP10 =	14	GP25 =	43	HEATBEAT
GP11 =	15	GP26 =	31	
GP12 =	16	GP27 =	32	
GP13 =	17	GP28 =	34	
GP14 =	19	GP29 =	44	AIN



```
PicoMite MMBasic RP2040 Edition V6.00.01
Copyright 2011-2024 Geoff Graham
Copyright 2016-2024 Peter Mather

> list commands
*/                  /*                  ADC                 Arc                 Autosave
Backlight           Blit                Blit Memory         Box                 Call
Camera              Case                Case Else           Cat                 Chdir
Circle              Clear               Close               CLS                 Color
Colour              Configure           Const               Continue            Copy
CPU                 CSub                CtrlVal(            Data                Date$
DefineFont          Device              Dim                 Do                  Draw3D
Drive               Edit                Edit File           Else                Else If
ElseIf              End                 End CSub            End DefineFont      End Function
End If              End Select          End Sub             EndIf               Erase
Error               Execute             Exit                Exit Do             Exit For
Exit Function       Exit Sub            Files               Flash               Flush
Font                For                 FRAMEBUFFER         Function            GoSub
GoTo                GUI                 Humid               I2C                 I2C2
If                  Inc                 Input               Interrupt           IR
IReturn             Keypad              Kill                LCD                 Let
Library             Line                Line Input          List                Load
Local               LongString          Loop                Math                Memory
MID$(               Mkdir               Mouse               New                 Next
On                  OneWire             Open                Option              Pause
Pin(                PIO                 Pixel               Play                Poke
Polygon             Port(               Print               Pulse               PWM
Randomize           RBox                Read                Refresh             Rem
Rename              Restore             Return              Rmdir               RTC
Run                 Save                Seek                Select Case         Servo
SetPin              SetTick             Sort                SPI                 SPI2
Sprite              Static              Sub                 SYNC                TEMPR START
Text                Time$               Timer               Trace               Triangle
Update Firmware     VAR                 WatchDog            While               Wii
Wii Classic         Wii Nunchuck        WS2812              XModem
Total of 149 commands

Dim .. Float   a! = Zwevende komma getallen.
Dim .. Integer a% = Gehele getallen 64-bits.
Dim .. String  a$ = Namen en getallen 0 - 255.

> memory
Program:
   0K ( 0%) Program (0 lines)
 128K (100%) Free

Saved Variables:
  16K (100%) Free

RAM:
   0K ( 0%) 0 Variables
   0K ( 0%) General
 156K (100%) Free
> flash list
 Slot 1 in use
 Slot 2 available
 Slot 3 available
```

```
PicoMite MMBasic RP2040 Edition V6.00.01
Copyright 2011-2024 Geoff Graham
Copyright 2016-2024 Peter Mather

> list functions
*               +               -               /               <
<<              <=              <>              =               =<
=>              >               >=              >>              @(
\               ^               Abs(            ACos(           And
As              Asc(            ASin(           Atan2(          Atn(
Bin$(           Bin2str$(       Bound(          Call(           Choice(
Chr$(           Cint(           Cos(            CtrlVal(        Cwd$
Date$           DateTime$(      Day$(           Deg(            DEVICE(
Dir$(           Distance(       Else            Eof(            Epoch(
Eval(           Exp(            Field$(         Fix(            For
Format$(        GoSub           GoTo            GPS(            Hex$(
Inkey$          Input$(         Instr(          Int(            INV
LCase$(         LCompare(       Left$(          Len(            LGetByte(
LGetStr$(       LInStr(         LLen(           Loc(            Lof(
Log(            Math(           Max(            Mid$(           Min(
MM.Device$      MM.ErrMsg$      MM.Errno        MM.Fontheight   MM.Fontwidth
MM.HPOS         MM.HRes         MM.I2C          MM.Info$(       MM.Info(
MM.OneWire      MM.PS2          MM.VER          MM.VPOS         MM.VRes
MM.Watchdog     Mod             MsgBox(         Not             Oct$(
Or              Peek(           Pi              Pin(            Pio(
Pixel(          Port(           Pos             Pulsin(         Rad(
RGB(            Right$(         Rnd             Rnd(            Sgn(
Sin(            Space$(         SPI(            SPI2(           sprite(
Sqr(            Step            Str$(           Str2bin(        String$(
Tab(            Tan(            TEMPR(          Then            Time$
Timer           To              Touch(          UCase$(         Until
Val(            While           Xor
Total of 133 functions and operators
```

ABS(getal)	Geeft als resultaat de absolute waarde van het argument. **-9 -> 9**
ASC(string$)	Retourneert de **ASCII-code** (d.w.z. **bytewaarde**)
BIN$(getal[,chars])	Geeft een string terug die **binaire** (**grondtal 2**) voor het **'getal'** geeft.
CINT(getal)	Ronde getallen met breuken omhoog of omlaag naar het volgende geheel getal of geheel getal. **45.47 -> 45, 45.57 -> 46**
EXP(getal)	Geeft als resultaat de exponentiële waarde van **'getal'**.
INT(getal)	Een uitdrukking afkappen tot het volgende gehele getal kleiner dan of gelijk aan de argument. **9.89 -> 9, -2.11 -> -3**
FIX(getal)	Verwijder alle cijfers achter de komma. **9.89 -> 9, -2.11 -> -2**
HEX$(getal[,chars])	Geeft een string terug die de **hex (grondtal 16)** voor het **'getal'** geeft.
OCT$(getal[,chars])	Geeft een string terug die de **bin (grondtal 2)** voor het **'getal'** geeft.
TAB(getal)	Voert **spaties** uit totdat de kolom aangegeven met **'getal'** is bereikt op de console uitgang.
VAL(string$)	Retourneert de numerieke waarde van de **'string$'**.

COMMANDO's en BASIC TREFWOORDEN

```
PicoMite MMBasic RP2040 Edition V6.00.02RC23
Copyright 2011-2025 Geoff Graham
Copyright 2016-2025 Peter Mather

> list commands
*/                .end program    .label          .line          .program
.side set         .wrap           .wrap target    /*             ADC
Arc               Array Add       Array Insert    Array Set      Array Slice
Autosave          Backlight       Bit(            Blit           Blit Memory
Box               Byte(           Call            Camera         Case
Case Else         Cat             Chain           Chdir          Circle
Clear             Close           CLS             Color          Colour
Configure         Const           Continue        Copy           CPU
CSub              CtrlVal(        Data            Date$          DefineFont
Device            Dim             Do              Draw3D         Drive
Edit              Edit File       Else            Else If        ElseIf
End               End CSub        End DefineFont  End Function   End If
End Select        End Sub         EndIf           Erase          Error
Execute           Exit            Exit Do         Exit For       Exit Function
Exit Sub          Files           Flag(           Flags          Flash
Flush             Font            For             FRAMEBUFFER    Function
GoSub             GoTo            GUI             Help           Humid
I2C               I2C2            If              In             Inc
Input             Interrupt       IR              IReturn        IRQ
IRQ CLEAR         IRQ NOWAIT      IRQ SET         IRQ WAIT       Jmp
Keypad            Kill            LCD             Let            Library
Line              Line Input      List            Load           Local
LongString        Loop            Math            Memory         MID$(
Mkdir             Mouse           Mov             New            Next
Nop               On              OneWire         Open           Option
Out               Pause           Pin(            PIO            Pixel
Play              Poke            Polygon         Port(          Print
Pull              Pulse           Push            PWM            Randomize
RBox              Read            Refresh         Rem            Rename
Restore           Return          Rmdir           RTC            Run
Save              Seek            Select Case     Servo          Set
SetPin            SetTick         Sort            SPI            SPI2
Sprite            Static          Sub             SYNC           TEMPR START
Text              Time$           Timer           Trace          Triangle
Update Firmware   VAR             Wait            WatchDog       While
Wii               Wii Classic     Wii Nunchuck    WS2812         XModem
Total of 180 commands

PIO assembler list commands
.end program    .label          .line           .program        .side set
.wrap           .wrap target    In              IRQ             IRQ CLEAR
IRQ NOWAIT      IRQ SET         IRQ WAIT        Jmp             List
Mov             Nop             Out             PIO             Pull
Push            Set             Static          Wait

Dim .. Float   a! = Zwevende komma getallen.
Dim .. Integer a% = Gehele getallen 64-bits.
Dim .. String  a$ = Namen en getallen 0 - 255.
```

FUNCTIES, STRINGS en OPERATORS

```
PicoMite MMBasic RP2040 Edition V6.00.02RC23
Copyright 2011-2025 Geoff Graham
Copyright 2016-2025 Peter Mather
> list functions
*               +               -               /               <
<<              <=              <>              =               =<
=>              >               >=              >>              @(
\               ^               Abs(            ACos(           And
As              Asc(            ASin(           Atan2(          Atn(
Bin$(           Bin2str$(       Bit(            Bound(          Byte(
Call(           Choice(         Chr$(           Cint(           Cos(
CtrlVal(        Cwd$            Date$           DateTime$(      Day$(
Deg(            DEVICE(         Dir$(           Distance(       Else
Eof(            Epoch(          Eval(           Exp(            Field$(
Fix(            Flag(           For             Format$(        GoSub
GoTo            GPS(            Hex$(           Inkey$          Input$(
Instr(          Int(            INV             LCase$(         LCompare(
Left$(          Len(            LGetByte(       LGetStr$(       LInStr(
LLen(           Loc(            Lof(            Log(            Math(
Max(            Mid$(           Min(            MM.CMDLINE$     MM.DEVICE$
MM.DISPLAY      MM.ErrMsg$      MM.Errno        MM.FLAGS        MM.FONTHEIGHT
MM.FONTWIDTH    MM.HEIGHT       MM.HPOS         MM.HRES         MM.I2C
MM.Info$(       MM.Info(        MM.ONEWIRE      MM.PS2          MM.VER
MM.VPOS         MM.VRES         MM.WATCHDOG     MM.WIDTH        Mod
MsgBox(         Not             Oct$(           Or              Peek(
Pi              Pin(            Pio(            Pixel(          Port(
Pos             Pulsin(         Rad(            RGB(            Right$(
Rnd             Rnd(            Sgn(            Sin(            Space$(
SPI(            SPI2(           sprite(         Sqr(            Step
Str$(           Str2bin(        String$(        Tab(            Tan(
TEMPR(          Then            Time$           Timer           To
Touch(          UCase$(         Until           Val(            While
Xor
Total of 142 functions and operators
```

ABS(getal)	Geeft als resultaat de absolute waarde van het argument. **-9 -> 9**
ASC(string$)	Retourneert de **ASCII-code** (d.w.z. **bytewaarde**)
BIN$(getal[,chars])	Geeft een string terug die **binaire** (**grondtal 2**) voor het **'getal'** geeft.
BIT(var$,byteno)	Geeft waarde van specifieke **bit (0-63)** in gehele variabele **(0/1)** terug.
BYTE(var$,byteno)	Geeft gehele waarde van specifieke **byte** in een string **(0-255)** terug.
CINT(getal)	Ronde getallen met breuken omhoog of omlaag naar het volgende geheel getal of geheel getal. **45.47 -> 45, 45.57 -> 46**
EXP(getal)	Geeft als resultaat de exponentiële waarde van **'getal'**.
INT(getal)	Een uitdrukking afkappen tot het volgende gehele getal kleiner dan of gelijk aan de argument. **9.89 -> 9, -2.11 -> -3**
FIX(getal)	Verwijder alle cijfers achter de komma. **9.89 -> 9, -2.11 -> -2**
HEX$(getal[,chars])	Geeft een string terug die de **hex (grondtal 16)** voor het **'getal'** geeft.
OCT$(getal[,chars])	Geeft een string terug die de **bin (grondtal 2)** voor het **'getal'** geeft.
TAB(getal)	Voert **spaties** uit totdat de kolom aangegeven met **'getal'** is bereikt op de console uitgang.
VAL(string$)	Retourneert de numerieke waarde van de **'string$'**.

LILYGO T-PicoC3. RP2040 + ESP32-C3 + ST7789V 1.14 inch IPS kleuren LCD 240x135

```
OPTION SYSTEM SPI GP2,GP3,GP0                    ' GP2=CLK, GP3=MOSI, GP0=MISO
OPTION FLASH SIZE 4194304    ' USB-C uart: RP2040 LED=Blauw. ESP32-C3 LED=Groen
OPTION COLOURCODE ON                                ' OR = L,P,RL,RP
OPTION LCDPANEL ST7789_135, LANDSCAPE,GP1,GP6,GP5,GP4    ' OR,DC,RST,CS,BKL
OPTION SDCARD GP17,GP18,GP19,GP20 'Ext.SD: GP17=CS,GP18=CLK,GP19=MOSI,GP20=MISO
```

LILYGO T-Display. RP2040 + ST7789V 1.14 inch IPS kleuren LCD 240x135

```
OPTION SYSTEM SPI GP2,GP3,GP0                   ' GP2=CLK, GP3=MOSI, GP0=MISO
OPTION FLASH SIZE 4194304
OPTION COLOURCODE ON                                ' OR = L, P, RL, RP
OPTION LCDPANEL ST7789_135, LANDSCAPE,GP1,GP6,GP5,GP4    ' OR, DC, RST, CS, BKL
OPTION SDCARD GP17,GP18,GP19,GP20 'Ext.SD: GP17=CS,GP18=CLK,GP19=MOSI,GP20=MISO
SetPin GP22, Dout : Pin(GP22) = 1 'Power on lcd. In het programma opnemen.
```

MAKER PI PICO bord. Voor Raspberry Pi Pico of Raspberry Pi Pico W

```
OPTION SYSTEM SPI GP10,GP11,GP12              ' GP10=CLK, GP11=MOSI, GP12=MISO
OPTION SYSTEM I2C GP4,GP5
OPTION COLOURCODE ON
OPTION HEARTBEAT OFF      ' Nu kan men GP25 zelf als knipper led programmeren.
OPTION DISPLAY 40, 80
OPTION SDCARD GP15
OPTION RTC AUTO ENABLE    ' RTC SETTIME Jaar, Mnd, Dag, Uur, Min, Sec
```

RASPBERRY PI PICO met ILI9341 LCD en SD

```
OPTION SYSTEM SPI GP18,GP19,GP16          ' CLK=GP18,MOSI=GP19,MISO=GP16
OPTION SDCARD GP22 ' CS=GP22              '  OR=L,DC=GP15,RESET=GP14,CS=GP13,BL
OPTION LCDPANEL ILI9341,L,GP15,GP14,GP13 ' GP26 of GP10 = BackLight
OPTION TOUCH GP12,GP11                    ' CS=GP12,IRQ=GP11,Beep
```

RP2040-GEEK RP2040 + ST7789 1.14 inch IPS kleuren LCD 240x135 en SD

```
PicoMite MMBasic Version 5.09.01
OPTION SYSTEM SPI GP10,GP11,GP24 ' GP10=CLK, GP11=MOSI, GP24=MISO LCD Scherm.
OPTION SYSTEM I2C GP4,GP5        ' I2C: GP4=SDA, GP5=SCL
OPTION SERIAL CONSOLE GP4,GP5    ' USB->SERIAL CONSOLE TX=GP4=BL, RX=GP5=GR
OPTION AUTORUN  ON               ' SETPIN RX,TX,COMX    GP5,GP4,COM2 Serieel
OPTION FLASH SIZE 4194304        ' I2C,UART,IO: GP4=BL, GND=ZW, GP05=GR
OPTION COLOURCODE ON             ' DEBUG,I/O  : GP2=BL, GND=ZW, GP03=GR
OPTION HEARTBEAT OFF             ' I2C,ADC,COM: GND=ZW, 3V3=RD, GP28=OR,GP29=GE
OPTION PICO OFF
OPTION CPUSPEED 252000           ' KHz
OPTION DISPLAY 40, 80                                ' OR=L, P, RL, RP
OPTION LCDPANEL ST7789_135, RLANDSCAPE,GP8,GP12,GP9,GP25 ' OR, DC, RST, CS, BKL
OPTION SDCARD GP23,GP18,GP19,GP20 ' GP23=CS, GP18=CLK, GP19=MOSI, GP20=MISO
OPTION PLATFORM RP2040-GEEK
```

PicoMiteVGA met PS-2 Keyboard en Audio en SD kaart en RTC

```
PicoMiteVGA MMBasic Version 5.09.01
OPTION SDCARD GP13,GP10,GP11,GP12 'OPTION SDCARD CS,CLK,MOSI,MISO)
OPTION SYSTEM I2C GP4,GP5 '(SDA,SCL)  Audio(L-R): GP06,GP07 (Links,Rechts)
OPTION COLOURCODE ON                  ' PS2(3V3): GP08,GP09 (Klok,Data)(5V-3V3)
OPTION KEYBOARD US 'CLK=GP8,DAT=GP9) VGA(RGGBHV): GP16,GP17,GP18,GP19,GP20,GP21
OPTION AUDIO GP6,GP7 '(L=PWMA,R=PWMB)       Vrij: GP00,GP01,GP02,GP03,GP14
OPTION RTC AUTO ENABLE                    ' Vrij: GP15,GP22,GP23,GP24,GP25
OPTION VGA PINS GP14,GP10 'HSYNC(2),BLUE(4) Vrij: GP26,GP27,GP28,GP29 (A0-A3)
```

```
PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.02b16
Copyright 2011-2024 Geoff Graham
Copyright 2016-2024 Peter Mather

> memory
Program:
   0K ( 0%) Program (0 lines)
 180K (100%) Free

Saved Variables:
  16K (100%) Free

RAM:
   0K ( 0%) 0 Variables
   0K ( 0%) General
 228K (100%) Free

> flash list
 Slot 1 available
 Slot 2 available
 Slot 3 available

> files
A:/
   <DIR>  .
   <DIR>  ..
00:00 01-01-2024          4  bootcount
2 directories, 1 file, 2543616 bytes free

> option list  ' Option lijst voor Olimex met HDMI en SD-KAART en AUDIO.
PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.02b16
OPTION SERIAL CONSOLE COM2,GP8,GP9
OPTION FLASH SIZE 4194304
OPTION COLOURCODE ON
OPTION KEYBOARD US
OPTION CPUSPEED (KHz) 315000
OPTION HDMI PINS  1, 3, 7, 5
OPTION SDCARD GP22, GP6, GP7, GP4             ' OPTION SDCARD CS,CLK,MOSI,MISO
' OPTION AUDIO GP26,GP27', ON PWM CHANNEL 5.  ' Met audio plug op de Olimex.
' OPTION AUDIO I2S GP1,GP5', ON PWM CHANNEL 0 ' Met externe PCM5102A aan Olimex
```

Opmerking: De Olimex print is een losse print zonder Raspberry Pi Pico en geschikt voor standaard RP2040 en RP2350 modules en W versies. HDMI ondersteuning aleen voor RP2350 met MMBasic. "Olimex RP2040-PICO-PC" voor Raspberry Pi Pico. De site: Tiny Tronics Eindhoven. €15,- RaspBerry Pi Pico RP2040 of RP2350 of de W versie. De site: KIWI Electronics Den Haag. v.a. ≈ €5,- De overige componenten bij AliExpress in China of Europa.
De verzendkosten vanuit het buitenland en btw maakt het vaak extra duur, dus eerst vergelijken.

```
> OPTION LIST      ' Korte demo breadboard met RP2040 en micro SD kaart houder.
PicoMite MMBasic RP2040 Edition V6.00.02b16 ' PicoMite RP2040 versie 6.00.02b14
OPTION SYSTEM I2C GP4,GP5              ' OPTION SYSTEM I2C SDA=GP4, SCL=GP5
OPTION COLOURCODE ON                   ' OPTION COLOURCODE ON is EDIT in kleur.
OPTION DISPLAY 44, 89                  ' OPTION DISPLAY REGELS=44, TEKENS=89.
OPTION SDCARD GP17, GP18, GP19, GP16   ' OPTION SDCARD CS, CLK, MOSI, MISO
```

Vanaf versie V6.00.02 (beta) enkele extra mogelijkheden.

```
> Option SYSTEM I2C GP4, GP5  ' GP4=SDA, GP5=SCL
> LIST SYSTEM I2C             ' Geeft de gebruikte I2C adressen in een tabel.
 HEX  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

> list pins    ' Lijst van standaard PicoMite pinnen op een Raspberry Pi pico.
GP0      1      OFF
GP1      2      OFF
GP2      4      OFF
GP3      5      OFF
GP4      6      OFF
GP5      7      OFF
GP6      9      OFF
GP7      10     OFF
GP8      11     OFF
GP9      12     OFF
GP10     14     OFF
GP11     15     OFF
GP12     16     OFF
GP13     17     OFF
GP14     19     OFF
GP15     20     OFF
GP16     21     OFF
GP17     22     OFF
GP18     24     OFF
GP19     25     OFF
GP20     26     OFF
GP21     27     OFF
GP22     29     OFF
GP23     41     DOUT
GP24     42     DIN
GP25     43     HEARTBEAT
GP26     31     OFF
GP27     32     OFF
GP28     34     OFF
GP29     44     AIN
```

Nu zijn alle pinnen beschikbaar:

```
Option HEARTBEAT OFF  ' Schakeld GP25 uit (knipper led) en is nu beschikbaar.
Option PICO OFF       ' Maakt alle overige pinnen beschikbaar voor gebruik.
```

I2S gebruik is mogelijk:

```
OPTION AUDIO I2S GP1, GP5  ' OPTION AUDIO I2S BCKpin, DATApin (LCKpin=BCK+1)
AUDIO PCM5102A I2S DAC: BCK=GP1, LCK=BCK+1=GP2, DIN=GP5, SCK=GND, VIN=5V, GND=-
```

Firmware Olimex bord met HDMI en USB keyboard configureren methode 1 of 2

Voor een Olimex: Laad van de site Geoff Graham de versie PicoMiteRP2350HDMIUSBV6.00.02RC23.uf2
1. Plaats HDMI scherm aan RPI en usb keyboard en usb voeding op Olimex. Type: option reset olimex usb
2. Standaard: OPTION SERIAL CONSOLE COM2, GP8, GP9 en geen Olimex pinnen. (Let op kruis links Rx-Tx)
Kies in Tera Term: Setup, Serial port, Com xx. (Welke te voorschijn komt bij port)
Maak verbinding met een externe Uart via pin GP8=Tx en GP9=Rx en GND. (standaard) Ook op 10 pin con
Doe nu: > OPTION RESET LIST of > OPTION RESET OLIMEX USB en doe dan een reset.
Nu is het standaard Com1: OPTION SERIAL CONSOLE COM1, GP0, GP1. Ook 3 pinnen op print naast usb.
Maak verbinding met een externe uart via pin GP0=Tx en GP1=Rx en GND. Zie ook alleen HDMI versie.

Na methode **1** of **2** en **Option List** kan dit op het scherm komen.

```
PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.02RC23
OPTION SERIAL CONSOLE COM1, GP0, GP1
OPTION SYSTEM I2C GP20,GP21       'Deze handmatig invoeren voor RTC klok en
OPTION FLASH SIZE 4194304         'Option RTC AUTO ENABLE voor juiste tijd.
OPTION COLOURCODE ON              'Tijd: > rtc settime jaar,mnd,dag,uur,min,sec
OPTION KEYBOARD US
OPTION CPUSPEED (KHz) 252000
OPTION RESOLUTION 640x480         'OPTION RESOLUTION 1280 geeft een beter beeld
OPTION HDMI PINS  1, 3, 7, 5
OPTION SDCARD GP22, GP6, GP7, GP4
OPTION AUDIO GP26,GP27            ' Of OPTION AUDIO I2S GP2,GP5 voor een PCM5102A I2S DAC
OPTION MODBUFF ENABLE  192        ' Eerst AUDIO uitschakelen met OPTION AUDIO DISABLE
OPTION PLATFORM OLIMEX USB
> LIST PINS
GP0      1      Boot Reserved : CONSOLE TX   ' COM1 ook naast de usb.  USB  GND-Rx-Tx
GP1      2      Boot Reserved : CONSOLE RX
GP2      4      OFF
GP3      5      OFF
GP4      6      Boot Reserved : SD MISO
GP5      7      OFF
GP6      9      Boot Reserved : SD CLK
GP7      10     Boot Reserved : SD MOSI
GP8      11     OFF
GP9      12     OFF
GP10     14     OFF
GP11     15     OFF
GP12     16     Boot Reserved : HDMI
GP13     17     Boot Reserved : HDMI
GP14     19     Boot Reserved : HDMI
GP15     20     Boot Reserved : HDMI
GP16     21     Boot Reserved : HDMI
GP17     22     Boot Reserved : HDMI
GP18     24     Boot Reserved : HDMI
GP19     25     Boot Reserved : HDMI
GP20     26     Boot Reserved : SYSTEM I2C SDA
GP21     27     Boot Reserved : SYSTEM I2C SCL
GP22     29     Boot Reserved : SD CS
GP23     41     DOUT
GP24     42     DIN
GP25     43     HEARTBEAT
GP26     31     Boot Reserved : AUDIO L
GP27     32     Boot Reserved : AUDIO R
GP28     34     OFF
GP29     44     AIN
```

```
PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.02RC23
Copyright 2011-2025 Geoff Graham
Copyright 2016-2025 Peter Mather

> list commands
*/              .end program    .label          .line
.program        .side set       .wrap           .wrap target
/*              ADC             Arc             Array Add
Array Insert    Array Set       Array Slice     Autosave
Bit(            Blit            Blit Memory     Box
Byte(           Call            Case            Case Else
Cat             Chain           Chdir           Circle
Clear           Close           CLS             CMM2 Load
CMM2 Run        Color           Colour          Colour Map
Configure       Const           Continue        Copy
CPU             CSub            Data            Date$
DefineFont      Device          Dim             Do
Draw3D          Drive           Edit            Edit File
Else            Else If         ElseIf          End
End CSub        End DefineFont  End Function    End If
End Select      End Sub         EndIf           Erase
Error           Execute         Exit            Exit Do
Exit For        Exit Function   Exit Sub        Files
Flag(           Flags           Flash           Flush
Font            For             FRAMEBUFFER     Function
Gamepad         GoSub           GoTo            GUI
Help            Humid           I2C             I2C2
If              In              Inc             Input
Interrupt       IR              IReturn         IRQ
IRQ CLEAR       IRQ NOWAIT      IRQ SET         IRQ WAIT
Jmp             Keypad          Kill            LCD
Let             Library         Line            Line Input
List            Load            Local           LongString
Loop            Map             Map(            Math
Memory          MID$(           Mkdir           MODE
Mouse           Mov             New             Next
Nop             On              OneWire         Open
Option          Out             Pause           Pin(
PIO             Pixel           Play            Poke
Polygon         Port(           Print           Pull
Pulse           Push            PWM             Ram
Randomize       RBox            Read            Rem
Rename          Restore         Return          Rmdir
RTC             Run             Save            Seek
Select Case     Servo           Set             SetPin
SetTick         Sort            SPI             SPI2
Sprite          Static          Sub             SYNC
TEMPR START     Text            TILE            Time$
Timer           Trace           Triangle        VAR
Wait            WatchDog        While           Wii
Wii Classic     Wii Nunchuck    WS2812          XModem
Total of 184 commands
```

```
PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.02RC23
Copyright 2011-2025 Geoff Graham
Copyright 2016-2025 Peter Mather

> list functions
*               +               -               /
<               <<              <=              <>
=               =<              =>              >
>=              >>              @(              \
^               Abs(            ACos(           And
As              Asc(            ASin(           Atan2(
Atn(            Bin$(           Bin2str$(       Bit(
Bound(          Byte(           Call(           Choice(
Chr$(           Cint(           Cos(            Cwd$
Date$           DateTime$(      Day$(           Deg(
DEVICE(         Dir$(           Distance(       DRAW3D(
Else            Eof(            Epoch(          Eval(
Exp(            Field$(         Fix(            Flag(
For             Format$(        GetScanLine     GoSub
GoTo            GPS(            Hex$(           Inkey$
Input$(         Instr(          Int(            INV
KeyDown(        LCase$(         LCompare(       Left$(
Len(            LGetByte(       LGetStr$(       LInStr(
LLen(           Loc(            Lof(            Log(
Map(            Math(           Max(            Mid$(
Min(            MM.CMDLINE$     MM.DEVICE$      MM.DISPLAY
MM.ErrMsg$      MM.Errno        MM.FLAGS        MM.FONTHEIGHT
MM.FONTWIDTH    MM.HEIGHT       MM.HPOS         MM.HRES
MM.I2C          MM.Info$(       MM.Info(        MM.ONEWIRE
MM.VER          MM.VPOS         MM.VRES         MM.WATCHDOG
MM.WIDTH        Mod             Not             Oct$(
Or              Peek(           Pi              Pin(
Pio(            Pixel(          Port(           Pos
Pulsin(         Rad(            RGB(            Right$(
Rnd             Rnd(            Sgn(            Sin(
Space$(         SPI(            SPI2(           sprite(
Sqr(            Step            Str$(           Str2bin(
String$(        Tab(            Tan(            TEMPR(
Then            Time$           Timer           To
UCase$(         Until           Val(            While
Xor
Total of 142 functions and operators

> option list
PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.02RC23
OPTION SERIAL CONSOLE COM1,GP0,GP1
OPTION SYSTEM I2C GP20,GP21
OPTION FLASH SIZE 4194304
OPTION COLOURCODE ON
OPTION KEYBOARD US
OPTION CPUSPEED (KHz) 252000
OPTION RESOLUTION 640x480
OPTION HDMI PINS  1, 3, 7, 5
OPTION SDCARD GP22, GP6, GP7, GP4
OPTION RTC AUTO ENABLE
OPTION MODBUFF ENABLE  192
OPTION PLATFORM OLIMEX USB
```

```
PicoMiteVGA MMBasic USB RP2040 Edition V6.00.02RC23
Copyright 2011-2025 Geoff Graham
Copyright 2016-2025 Peter Mather

> US USB Keyboard Connected on channel 1
> USB Mouse Connected on channel 2

> list commands
*/              .end program    .label          .line
.program        .side set       .wrap           .wrap target
/*              ADC             Arc             Array Add
Array Insert    Array Set       Array Slice     Autosave
Bit(            Blit            Blit Memory     Box
Byte(           Call            Case            Case Else
Cat             Chain           Chdir           Circle
Clear           Close           CLS             Color
Colour          Colour Map      Configure       Const
Continue        Copy            CPU             CSub
Data            Date$           DefineFont      Device
Dim             Do              Draw3D          Drive
Edit            Edit File       Else            Else If
ElseIf          End             End CSub        End DefineFont
End Function    End If          End Select      End Sub
EndIf           Erase           Error           Execute
Exit            Exit Do         Exit For        Exit Function
Exit Sub        Files           Flag(           Flags
Flash           Flush           Font            For
FRAMEBUFFER     Function        Gamepad         GoSub
GoTo            GUI             Help            Humid
I2C             I2C2            If              In
Inc             Input           Interrupt       IR
IReturn         IRQ             IRQ CLEAR       IRQ NOWAIT
IRQ SET         IRQ WAIT        Jmp             Keypad
Kill            LCD             Let             Library
Line            Line Input      List            Load
Local           LongString      Loop            Map
Map(            Math            Memory          MID$(
Mkdir           MODE            Mouse           Mov
New             Next            Nop             On
OneWire         Open            Option          Out
Pause           Pin(            PIO             Pixel
Play            Poke            Polygon         Port(
Print           Pull            Pulse           Push
PWM             Randomize       RBox            Read
Rem             Rename          Restore         Return
Rmdir           RTC             Run             Save
Seek            Select Case     Servo           Set
SetPin          SetTick         Sort            SPI
SPI2            Sprite          Static          Sub
SYNC            TEMPR START     Text            TILE
Time$           Timer           Trace           Triangle
VAR             Wait            WatchDog        While
Wii             Wii Classic     Wii Nunchuck    WS2812
XModem
Total of 181 commands
```

```
PicoMiteVGA MMBasic USB RP2040 Edition V6.00.02RC23
Copyright 2011-2025 Geoff Graham
Copyright 2016-2025 Peter Mather

> US USB Keyboard Connected on channel 1
> USB Mouse Connected on channel 2

> list functions
*               +               -               /
<               <<              <=              <>
=               =<              =>              >
>=              >>              @(              \
^               Abs(            ACos(           And
As              Asc(            ASin(           Atan2(
Atn(            Bin$(           Bin2str$(       Bit(
Bound(          Byte(           Call(           Choice(
Chr$(           Cint(           Cos(            Cwd$
Date$           DateTime$(      Day$(           Deg(
DEVICE(         Dir$(           Distance(       DRAW3D(
Else            Eof(            Epoch(          Eval(
Exp(            Field$(         Fix(            Flag(
For             Format$(        GetScanLine     GoSub
GoTo            GPS(            Hex$(           Inkey$
Input$(         Instr(          Int(            INV
KeyDown(        LCase$(         LCompare(       Left$(
Len(            LGetByte(       LGetStr$(       LInStr(
LLen(           Loc(            Lof(            Log(
Map(            Math(           Max(            Mid$(
Min(            MM.CMDLINE$     MM.DEVICE$      MM.DISPLAY
MM.ErrMsg$      MM.Errno        MM.FLAGS        MM.FONTHEIGHT
MM.FONTWIDTH    MM.HEIGHT       MM.HPOS         MM.HRES
MM.I2C          MM.Info$(       MM.Info(        MM.ONEWIRE
MM.PERSISTENT   MM.VER          MM.VPOS         MM.VRES
MM.WATCHDOG     MM.WIDTH        Mod             Not
Oct$(           Or              Peek(           Pi
Pin(            Pio(            Pixel(          Port(
Pos             Pulsin(         Rad(            RGB(
Right$(         Rnd             Rnd(            Sgn(
Sin(            Space$(         SPI(            SPI2(
sprite(         Sqr(            Step            Str$(
Str2bin(        String$(        Tab(            Tan(
TEMPR(          Then            Time$           Timer
To              UCase$(         Until           Val(
While           Xor
Total of 143 functions and operators

> option list
PicoMiteVGA MMBasic USB RP2040 Edition V6.00.02RC23
OPTION SERIAL CONSOLE COM2,GP8,GP9
OPTION SYSTEM I2C GP16,GP17
OPTION COLOURCODE ON
OPTION KEYBOARD US
OPTION HEARTBEAT OFF
OPTION PICO OFF
OPTION RESOLUTION 640x480 @ 252000KHz
OPTION SDCARD GP21, GP18, GP19, GP20
OPTION VGA PINS GP11,GP22
OPTION AUDIO I2S GP13,GP15', ON PWM CHANNEL 6
```

```
PicoMiteVGA MMBasic USB RP2040 Edition V6.00.02RC23
Copyright 2011-2025 Geoff Graham
Copyright 2016-2025 Peter Mather

> US USB Keyboard Connected on channel 1
> USB Mouse Connected on channel 2

> memory
Program:
   0K ( 0%) Program (0 lines)
 100K (100%) Free

Saved Variables:
  16K (100%) Free

RAM:
   0K ( 0%) 0 Variables
   0K ( 0%) General
 126K (100%) Free

> list pins
GP0      1      OFF
GP1      2      OFF
GP2      4      OFF
GP3      5      OFF
GP4      6      OFF
GP5      7      OFF
GP6      9      OFF
GP7      10     OFF
GP8      11     Boot Reserved : CONSOLE TX
GP9      12     Boot Reserved : CONSOLE RX
GP10     14     OFF
GP11     15     Boot Reserved : VGA HSYNC
GP12     16     Boot Reserved : VGA VSYNC
GP13     17     Boot Reserved : I2S BCLK
GP14     19     Boot Reserved : I2S LRCK
GP15     20     Boot Reserved : I2S DATA
GP16     21     Boot Reserved : SYSTEM I2C SDA
GP17     22     Boot Reserved : SYSTEM I2C SCL
GP18     24     Boot Reserved : SPI SYSTEM CLK
GP19     25     Boot Reserved : SPI SYSTEM MOSI
GP20     26     Boot Reserved : SPI SYSTEM MISO
GP21     27     Boot Reserved : SD CS
GP22     29     Boot Reserved : VGA BLUE
GP23     41     Boot Reserved : VGA GREEN L
GP24     42     Boot Reserved : VGA GREEN H
GP25     43     Boot Reserved : VGA RED
GP26     31     OFF
GP27     32     OFF
GP28     34     OFF
GP29     44     OFF
>
```

Beschrijving van connectorpinnen voor een Olimex print

Men kan versie **PicoMiteHDMI MMBasic USB RP2350A Edition V6.00.02RC23** home computer of versie **PicoMiteHDMI MMBasic RP2350A Edition V6.00.02RC23** met standaard usb en Tera Term. Bij de alleen **HDMI** kan men zonder **externe uart** en **usb-mini -> usb-A** dus een standaard **usb** gewoon met **Tera Term** werken dus uitvoer naar laptop en een los **HDMI** scherm voor ontwikkeling programma's. Updaten gaat ook via de standaard methode.

10 polige connectorpinnen

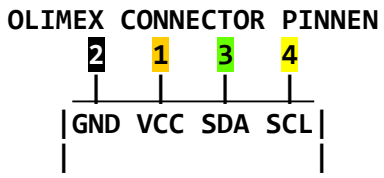
GP Pin	Pin #	Signaalnaam
3V3	1	3V3
GND	2	GND
GP20	3	I2C SDA RTC
GP21	4	I2C SCL RTC
GP9	5	SPI DC LCD
GP8	6	SPI RESET LCD
GP4	7	SPI MISO SD
GP7	8	SPI MOSI SD
GP6	9	SPI KLOK SD
GP5	10	SPI CS LCD

UEXT1 10 Polig. LCD Backlight met weerstand of externe pin

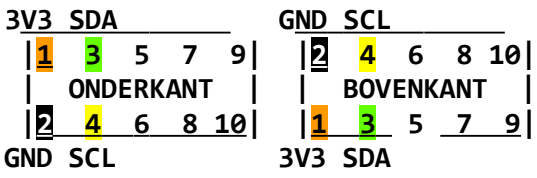
3 pins aansluiting naast USB

GP Pin	Pin #	Signaalnaam
GP0	1	COM1 TX
GP1	2	COM1 RX
GND	3	GND

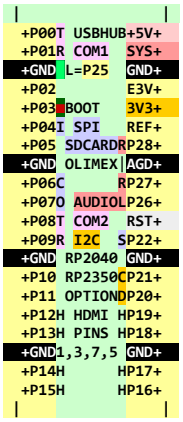
DBG1 pinnen na Option Reset Olimex USB



DS3231 CONNECTOR



OLIMEX CONNECTOR



Sjabloon RP2040 of RP2350 Olimex

Voor printen zie blz. 447 bovenaan.