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
```
Geldige pinnen zijn: **SDA: GP00, GP04, GP08, GP12, GP16, GP20, GP28**
SCL: GP01, GP05, GP09, GP13, GP17, GP21, GP29

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, GP23, 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.**

I2C Master opdrachten

Er zijn vier commando's die als volgt kunnen worden gebruikt voor het eerste kanaal (**I2C**) in de mastermodus.
De commando's voor het tweede kanaal (**I2C2**) zijn identiek, behalve dat het commando **I2C2** is.

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.

COMMANDO	Beschrijving 2

`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.**

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
```

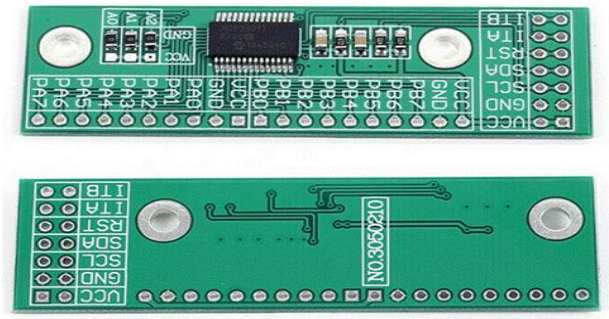
Dan de slaaf:

```
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
```

I2C programma

MCP23017 module

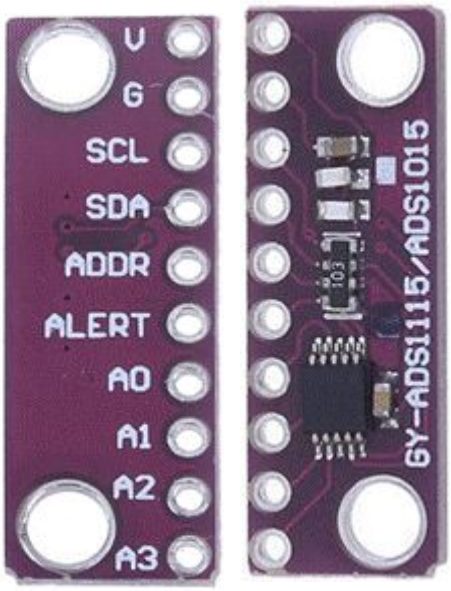
Een uitbreiding van het aantal uitgangen of ingangen met een **I2C MCP23017** met **16** stuks.



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 pull-ups 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 ADS11115 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

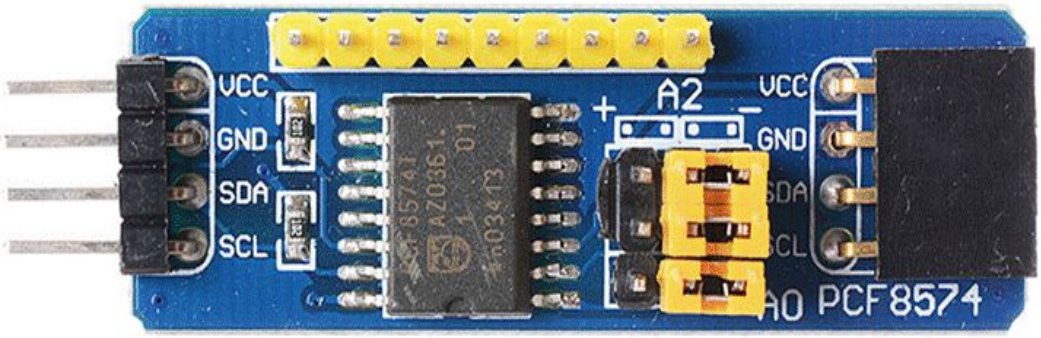
```
' i2c_scanner.bas Scannen van aangesloten modules. Versie 1.1
 Print " ****************** I2C SCANNER ********************"
 Print " Input/Output  PCF8574     Adres: Dec: 32   Hex:0x20"
 Print " Input/Output  MCP23017    Adres: Dec: 32   Hex:0x20"
 Print " LCD 20x4 5V0  HD44780     Adres: Dec: 32   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 3-5V DS3231      Adres: Dec:104   Hex:0x68"
 Print " Klok RTC 4-5V 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
```

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 3-5V DS3231      Adres: Dec:104   Hex:0x68
 Klok RTC 4-5V 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: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
>
```

```
' 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



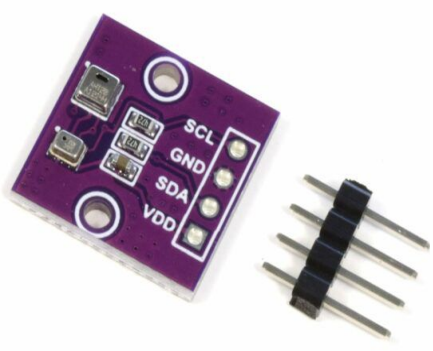
Het biedt 8 digitale ingangen of uitgangen.

```
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
```

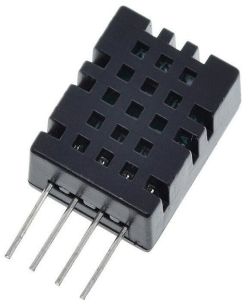
DHT20 of AHT20 I2C Temperatuur en Vochtigheid sensor

```
' 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

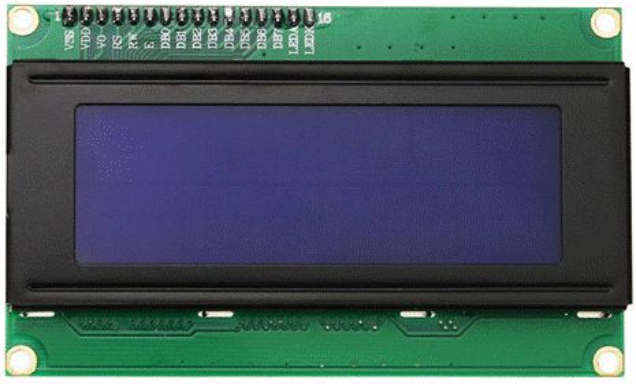
```
' Option SYSTEM SPI GP18,GP19,GP16
' Option SYSTEM I2C GP4,GP5           ' GP4=SDA , GP5=SCL
' Option LCDPANEL SSD1306I2C, LANDSCAPE
' Option SDCARD GP17
' Option RTC AUTO ENABLE
' RTC SETTIME 2023,3,26,21,28,0 jaar,maand,dag,uur,min,sec RTC tijd instellen.
' Zonder RTC tijd: Time$="hh:mm:ss" en Date$="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
```

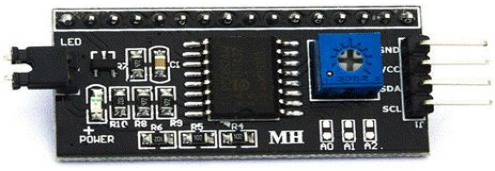
LCDI2CLib test voor bijna elk type PCF8574 I2C naar LCD-module

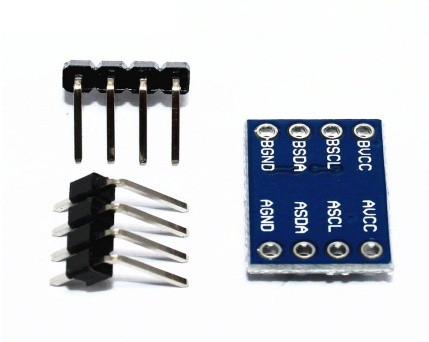
```
PicoMite MMBasic Version 5.07.07b5
OPTION SYSTEM SPI GP18,GP19,GP16
OPTION SYSTEM I2C GP4,GP5
OPTION COLOURCODE ON
OPTION DISPLAY 40, 90
OPTION LCDPANEL SSD1306I2C, LANDSCAPE    ' 128x64 Oled scherm
OPTION SDCARD GP22
OPTION RTC AUTO ENABLE
' Deze subroutine gevonden op het MMBasic forum en geschikt 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$(0) is CMD Reg. 64-71.
  LCDI2C_DATA(7): LCDI2C_DATA(5): LCDI2C_DATA(7): LCDI2C_DATA(0)   ' Byte 1-4.
  LCDI2C_DATA(0): LCDI2C_DATA(0): LCDI2C_DATA(0): LCDI2C_DATA(0)   ' Byte 5-8.
  LCDI2C 1, 1, "BuitenT:R 18.3 C" ' LCDI2C Line, Pos, Data$
  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)
```

Zie ook subroutines **LCDI2C.**

Alle hardware I/O functies RP2040 en 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 : RX,TX,SCK,CSn
RX  :GP00,GP04,GP16,GP20
TX  :GP03,GP07,GP19,GP23
CSn :GP01,GP05,GP17,GP21
SCK :GP02,GP06,GP18,GP22
SPI2: RX,TX,SCK,CSn
RX  :GP08,GP12,GP24,GP28
TX  :GP11,GP15,GP27
CSn :GP09,GP13,GP25,GP29
SCK :GP10,GP14,GP26
GPIO:GP00-GP29
PIO0:GP00-GP29
PIO1:GP00-GP29
RX :MasterInputSlaveOutp
TX :MasterOutpSlaveInput
```

```
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 =	PCB
GP09 =	12	GP24 =	PCB
GP10 =	14	GP25 =	PCB
GP11 =	15	GP26 =	31
GP12 =	16	GP27 =	32
GP13 =	17	GP28 =	34
GP14 =	19	GP29 =	PCB