```
'******************************************

'BME280 sample code

'(C) 2015 by Michael Lehmann
'Mlehmann (a) mgkulm.ch

'Translated from German to Engish by Google Translate


'******************************************

'*** Changes ***************************

'Current version
Const Test_version = "V1.1"


'V1.1 16-01-16
'Error corrected with t_fine. Thus, the air pressure was compensated wrong
'-t_fine, P_fine, h_fine, pressure_old in T, P, H, h_old renamed
'-in Setup_bme280 () is a waiting time of 5ms inserted so that the sensor has
completed the startup procedure safely


'V1.0 12-12-15
'-All Basic functions available



'**** Description

'The BME280 sensor from Bosch can temperature, pressure and humidity measuring.
'It is cheap and very well suited for applications such as weather stations,
altimeter, etc.
'Thanks to the high accuracy of the air pressure sensor in altimeters resolutions of
a few cm.

'Since the sensor itself provides only uncompensated values, they must be compensated
for with the calibration.
'Here are a few, relatively complicated formulas are needed. Accordingly, a sample
code listed for the test,
'Or for integration is in their own projects.

'Two functions are listed to calculate the air pressure value. Depending on the
application only one is needed.
'Pressure_64 () calculates the air pressure with 64bit, it results in a resolution of
0.001hPa. This is for
'Precision altimeter / Varios needed.

'Pressure_32 () calculates the air pressure with 32bit, it results in a resolution of
0.01hPa.
'This is quite sufficient for a weather station.

'For your own applications, please see the datasheet of BME280 sure to read !!

'** THE CODE SHALL NOT USED COMMERCIALLY !!! **




'**** Program:

'The sample code list values ??from the sensor and outputs them via UART with 9600
baud from.
'The sensor for communication is by i2c bus.

'Only at the beginning needed:
```

```
'- Initialize sensor: Write settings Read, calibration

'In the main program:
Read uncompensated values ??from the sensor - '
'- Calculate the compensated values









'MCU config
$ Regfile = "m328def.dat"
$ Crystal = 8000000
$ Hwstack = 256
$ Swstack = 256
$ Frame size = 256



'I2C config
Config = Sda Portc.0
Config Scl = Portc.1


'Config UART
Config Com1 = 9600, Synchronous = 0, Parity = None, Stop bits = 1, data bits = 8,
Clockpol = 0
Open "com1:" For Binary As # 1





'******************************************
'Settings for the sensor BME280

'Please read the datasheet

'Sensor address
Const Bme280_adress = HEC

'Oversampling
Const Osrs_t = & B101
Const Osrs_p = & B101
Const Osrs_h = & B101

'Mode Register
Const Mode_reg = B11 'Normal Mode

't_sb
Const T_sb = B000 't_sb = 0.5ms

'IIR filter
Const filter = & B100 'filter = 16


'******************************************





'Dimension variables
Dim txt As String * 10

'Data array for i2c communication
```

```
Dim Wert_array (24) As Byte

'Calibration values ??from the sensor
Dim Dig_t1 As Word
Dim Dig_t2 As Integer
Dim Dig_t3 As Integer
Dim Dig_p1 As Word
Dim Dig_p2 As Integer
Dim Dig_p3 As Integer
Dim Dig_p4 As Integer
Dim Dig_p5 As Integer
Dim Dig_p6 As Integer
Dim Dig_p7 As Integer
Dim Dig_p8 As Integer
Dim Dig_p9 As Integer
Dim Dig_h1 As Byte
Dim Dig_h2 As Integer
Dim Dig_h3 As Byte
Dim Dig_h4 As Integer
Dim Dig_h5 As Integer
Dim Dig_h6 As Byte

'Uncompensated sensor values
Dim Ut As Long
Dim Up As Long
Dim Uh As Long

'Cross-functional auxiliary variables
Dim T_fine As Long
Dim P_old As Dword

'Compensated sensor values
Dim T As Long
Dim P As Dword
Dim H As Dword

'Declarieren SUBs and Functons
Declare Sub I2c_write (byval Device_adres As Byte, byval Reg_adres As Byte, byval
value As Byte)
Declare Sub I2c_read (byval Device_adres As Byte, byval Reg_adres As Byte, byval
Wert_count As Byte)
Declare Sub Setup_bme280 ()
Declare Sub Read_bme280_value ()
Declare Function Temp () As Long
Declare Function Pressure_64 () As Dword
Declare Function Pressure_32 () As Dword
Declare Function Humidity () As Dword




'Initialize the sensor BME280
Call Setup_bme280 ()
'Calibration values ??from the sensor reading, writing filter values




'*** StartupInfo
Print # 1, "BME280 test program"; Trial version
Print # 1, "(c) 2015 by Michael Lehmann"
Print # 1, ""


Show '*** calibration
Print # 1, "calibration"
Print # 1, "Dig_t1:"; Dig_t1
Print # 1, "Dig_t2:"; Dig_t2
Print # 1, "Dig_t3:"; Dig_t3
Print # 1, "Dig_p1:"; Dig_p1
Print # 1, "Dig_p2:"; Dig_p2
Print # 1, "Dig_p3:"; Dig_p3
Print # 1, "Dig_p4:"; Dig_p4
Print # 1, "Dig_p5:"; Dig_p5
Print # 1, "Dig_p6:"; Dig_p6
Print # 1, "Dig_p7:"; Dig_p7
Print # 1, "Dig_p8:"; Dig_p8
Print # 1, "Dig_p9:"; Dig_p9
Print # 1, "Dig_h1:"; Dig_h1
Print # 1, "Dig_h2:"; Dig_h2
Print # 1, "Dig_h3:"; Dig_h3
Print # 1, "Dig_h4:"; Dig_h4
Print # 1, "Dig_h5:"; Dig_h5
Print # 1, "Dig_h6:"; Dig_h6
Print # 1, ""







'*** Main program

do


  read '** BME280
  Call Read_bme280_value ()
  'List the uncompensated value from the sensor,
  'These are stored in Ut, Up, Uh
  Print # 1, "Ut"; Ut; "Up"; Up; "Uh,"; Uh


  '** Temperature
  T = Temp ()
  'Is the temperature in ° C back, the resolution is 0.01 ° C
  'The value of 2415 corresponds to 24.15 ° C
  Txt = Str (t)
  Txt = format (txt, "00.00")
  Print # 1, "temperature"; Txt; "° C"


  '** Barometer with 64bit calculation
  P = Pressure_64 ()
  'Is the air pressure in hPa back, the resolution is 0.001hPa
  'The value corresponds to 963 861 963.861hPa
  Txt = Str (p)
  Txt = format (txt, "00,000")
  Print # 1, "Pressure 64bit:"; Txt; "HPa"


  '** Barometer with 32bit calculation
  P = Pressure_32 ()
  'Is the air pressure in hPa back, the resolution is 0.01hPa
  'The value of 96386 corresponds 963.86hPa
  Txt = Str (p)
  Txt = format (txt, "00.00")
  Print # 1, "Pressure 32bit:"; Txt; "HPa"


  '** Humidity
  H = Humidity ()
  'Is the humidity returns in%, the resolution 0.001hPa
  'The value of 46333 corresponds to 46 333%
  Txt = Str (h)
  Txt = format (txt, "00,000")
  Print # 1, "humidity"; Txt; "%"


  Print # 1, ""

  Wait 1
loop


End








Sub I2c_write (device_adres As Byte, Reg_adres As Byte, Value As Byte)
  I2cstart
  I2cwbyte Device_adres
  I2cwbyte Reg_adres
  I2cwbyte value
  I2cstop
  Waitms 10
End Sub


Sub I2c_read (device_adres As Byte, Reg_adres As Byte, Wert_count As Byte)
  Local X As Byte
  Local Y As Byte
  Y = Wert_count - 1

  I2cstart 'Start I2C
  I2cwbyte Device_adres' transmitting slave address
  I2cwbyte Reg_adres' register address
  I2cstart
  Incr Device_adres
  I2cwbyte Device_adres' send slave address +1 reading
  If Wert_count> 1 Then
    For X = 1 To Y
      I2crbyte Wert_array (x), Ack 'read value
    Next
  End If
  I2crbyte Wert_array (wert_count), Nack 'read value
  I2cstop
  Waitms 10
End Sub


Sub Read_bme280_value ()
  Call I2c_read (bme280_adress, & HF7, 8)

  'Uncompensated pressure value
  Up = Wert_array (1)
  Shift Up, Left, 8
  Up = Up + Wert_array (2)
  Shift Up, Left, 8
  Up = Up + Wert_array (3)
  Shift Up, Right, 4

  'Uncompensated temperature value
  Ut = Wert_array (4)
  Shift Ut, Left, 8
  Ut = Ut + Wert_array (5)
  Shift Ut, Left, 8
  Ut = Ut + Wert_array (6)
  Shift Ut, Right, 4
  'Uncompensated humidity value
  Uh = Wert_array (7)
  Shift Uh, Left, 8
  Uh = Uh + Wert_array (8)
End Sub


Function Temp () As Long
  Local Var1 As Long
  Local Var2 As Long
  Local X As Long

  Var1 = Ut
  Shift Var1, Right, 3, Signed
  X = Dig_t1
  Shift X, Left, 1, Signed
  Var1 = Var1 - X
  Var1 = Var1 * Dig_t2
  Shift Var1, Right, 11, Signed

  Var2 = Ut
  Shift Var2, Right, 4, Signed
  Var2 = Var2 - Dig_t1
  Var2 = Var2 * Var2
  Shift Var2, Right, 12, Signed
  Var2 = Var2 * Dig_t3
  Shift Var2, Right, 14, Signed

  T_fine = Var1 + Var2
  Temp = T_fine * 5
  Temp Temp = + 128
  Shift Temp, Right, 8, Signed
End Function



Function Pressure_64 () As Dword
  Local Var1 As Double
  Local Var2 As Double
  Local X As Double
  Local Y As Double
  Local Z As Double
  Local S1 As Single
  Local L1 As Long

  'Var1 = t_fine - 128000
  S1 = T_fine
  Y = S1
  X = 128000
  Var1 = Y - X

  'Var2 = var1 var1 * * dig_P6
  Var2 = Var1 Var1 *
  S1 = Dig_p6
  X = S1
  Var2 = Var2 * X

  'Var2 = var2 + ((var1 * dig_P5) << 17);
  S1 = Dig_p5
  X = S1
  X = X Var1 *
  Y = 2 ^ 17
  X = X * Y
  Var2 = Var2 + X

  'Var2 = var2 + (dig_P4 << 35);
  S1 = Dig_p4
```

```
X = S1
Y = 2 ^ 35
X = X * Y
Var2 = Var2 + X

'Var1 = ((var1 var1 * * dig_P3) >> 8) + ((var1 * dig_P2) << 12);
X = Var1 Var1 *
S1 = Dig_p3
Z = S1
X = X * Z
Y = 2 ^ 8
X = X / Y

S1 = Dig_p2
Z = S1
Y = Var1 * Z
Z = 2 ^ 12
Y = Y * Z
Var1 = X + Y

'Var1 = ((1 << 47) + var1) * dig_P1 >> 33;
X = H800000000000
X = X + Var1
S1 = Dig_p1
Z = S1
Var1 = X * Z
Y = 2 ^ 33
Var1 = Var1 / Y

L1 = Var1
If L1 = 0 Then
  Pressure_64 = P_old
  Exit Function
End If

'X = 1048576-up;
X = 1048576
S1 = Up
Z = S1
X = X - Z

'X = (((x << 31) -var2) * 3125) / var1;
Y = 2 ^ 31
X = X * Y
X = X - Var2
Y = 3125
X = X * Y
X = X / Var1

'Var1 = (dig_P9 * (x >> 13) * (x >> 13)) >> 25;
Z = X
Y = 2 ^ 13
Z = Z / Y
S1 = Dig_p9
Y = S1
Var1 = Y * Z
Var1 = Var1 * Z
Y = 2 ^ 25
Var1 = Var1 / Y

'Var2 = (dig_P8 * x) >> 19;
S1 = Dig_p8
Y = S1
Var2 = Y * X
Y = 2 ^ 19
Var2 = Var2 / Y

'X = ((x + var1 + var2) >> 8) + (dig_P7 << 4);
```

```
  X = X + Var1
  X = X + Var2
  Y = 2 ^ 8
  X = X / Y
  S1 = Dig_p7
  Y = S1
  Z = 2 ^ 4
  Y = Y * Z
  X = X + Y

  Y = 25.6
  X = X / Y

  Pressure_64 = X
  P_old = Pressure_64
End Function


Function Pressure_32 () As Dword
  Local Var1 As Long
  Local Var2 As Long
  Local X As Long
  Local Y As Long
  Local Z As Dword

  Var1 = T_fine
  Shift Var1, Right, 1, Signed
  Var1 = Var1 - 64000

  X = Var1
  Shift X, Right, 2
  Var2 = X * X
  Shift Var2, Right, 11, Signed
  Var2 = Var2 * Dig_p6

  X = Var1 * Dig_p5
  Shift X, Left, 1, Signed
  Var2 = Var2 + X

  X = Var2
  Shift X, Right, 2, Signed
  Y = Dig_p4
  Shift Y, Left, 16, Signed
  Var2 = X + Y

  X = Var1
  Shift X, Right, 2, Signed
  X = X * X
  Shift X, Right, 13, Signed
  X = X * Dig_p3
  Shift X, Right, 3, Signed
  Y = Dig_p2 * Var1
  Shift Y, Right, 1, Signed
  Var1 = X + Y
  Shift Var1, Right, 18, Signed

  Var1 = 32756 + Var1
  Var1 = Var1 * Dig_p1
  Shift Var1, Right, 15, Signed

  If Var1 = 0 Then
    Pressure_32 = P_old
    Exit Function
  End If

  X = 1048576 - Up
  Y = Var2
  Shift Y, Right, 12, Signed
  X = X - Y
  Pressure_32 = X * 3125

  If Pressure_32 <& h80000000 Then
    Shift Pressure_32, Left, 1, Signed
    Pressure_32 = Pressure_32 / Var1
  else
    Pressure_32 = Pressure_32 / Var1
    Pressure_32 = Pressure_32 * 2
  End If

  Z = Pressure_32
  Shift Z, Right, 3, Signed
  Z = Z * Z
  Shift Z, Right, 13, Signed
  Z = Z * Dig_p9
  Shift Z, Right, 12, Signed
  Var1 = Z

  Z = Pressure_32
  Shift Z, Right, 2, Signed
  Z = Z * Dig_p8
  Shift Z, Right, 13, Signed
  Var2 = Z

  Z = Var1 + Var2
  Z = Z + Dig_p7
  Shift Z, Right, 4, Signed
  Pressure_32 = Pressure_32 + Z
  P_old = Pressure_32
End Function


Function Humidity () As Dword
  Local Var1 As Long
  Local X As Long
  Local Y As Long
  Local Z As Long

  Var1 = T_fine - 76800

  X = Uh
  Shift X, Left, 14, Signed
  Y = Dig_h4
  Shift Y, Left, 20, Signed
  X = X - Y
  Y = Dig_h5 * Var1
  X = X - Y
  X = X + 16384
  Shift X, Right, 15, Signed

  Y = Var1 * Dig_h6
  Shift Y, Right, 10, Signed

  Z = Var1 * Dig_h3
  Shift Z, Right, 11, Signed
  Z = Z + 32768

  Y = Y * Z
  Shift Y, Right, 10, Signed
  Y = Y + 2097152
  Y = Y * Dig_h2
  Y = Y + 8192
  Shift Y, Right, 14, Signed

  Var1 = X * Y

  X = Var1
  Shift X, Right, 15, Signed
  X = X * X
  Shift X, Right, 7, Signed
  X = X * Dig_h1
  Shift X, Right, 4, Signed
  Var1 = Var1 - X

  If Var1 <0 Then
    Var1 = 0
  End If

  If Var1> 419430400 Then
    Var1 = 419430400
  End If

  Shift Var1, Right, 12, Signed
  Humidity = Var1 / 1,024
End Function




Sub Setup_bme280 ()
  Local X As Byte

  'Wait until the sensor completed the startup procedure
  Waitms 5

  'BME280 write filter value and setting t_sb
  X = T_sb
  Shift X, Left, 3
  X = X + Filter
  Shift X, Left, 2
  Call I2c_write (bme280_adress, & HF5, X)

  'BME280 write OSRS_h value (ctrl_hum register)
  Call I2c_write (bme280_adress, & HF2 Osrs_h)

  'BME280 write OSRS_t, OSRS_p, Fashion value (ctrl_meas register)
  X = Osrs_t
  Shift X, Left, 3
  X = X + Osrs_p
  Shift X, Left, 2
  X = X + Mode_reg

  Call I2c_write (bme280_adress, & HF4, X)

  'BME280 read calibration
  Call I2c_read (bme280_adress, & H88, 24)

  Dig_t1 = Wert_array (2)
  Shift Dig_t1, Left, 8
  Dig_t1 = Dig_t1 + Wert_array (1)

  Dig_t2 = Wert_array (4)
  Shift Dig_t2, Left, 8
  Dig_t2 = Dig_t2 + Wert_array (3)

  Dig_t3 = Wert_array (6)
  Shift Dig_t3, Left, 8
  Dig_t3 = Dig_t3 + Wert_array (5)

  Dig_p1 = Wert_array (8)
  Shift Dig_p1, Left, 8
  Dig_p1 = Dig_p1 + Wert_array (7)

  Dig_p2 = Wert_array (10)
  Shift Dig_p2, Left, 8
  Dig_p2 = Dig_p2 + Wert_array (9)

  Dig_p3 = Wert_array (12)
  Shift Dig_p3, Left, 8
  Dig_p3 = Dig_p3 + Wert_array (11)

  Dig_p4 = Wert_array (14)
  Shift Dig_p4, Left, 8
  Dig_p4 = Dig_p4 + Wert_array (13)

  Dig_p5 = Wert_array (16)
  Shift Dig_p5, Left, 8
  Dig_p5 = Dig_p5 + Wert_array (15)

  Dig_p6 = Wert_array (18)
  Shift Dig_p6, Left, 8
  Dig_p6 = Dig_p6 + Wert_array (17)

  Dig_p7 = Wert_array (20)
  Shift Dig_p7, Left, 8
  Dig_p7 = Dig_p7 + Wert_array (19)

  Dig_p8 = Wert_array (22)
  Shift Dig_p8, Left, 8
  Dig_p8 = Dig_p8 + Wert_array (21)

  Dig_p9 = Wert_array (24)
  Shift Dig_p9, Left, 8
  Dig_p9 = Dig_p9 + Wert_array (23)

  Call I2c_read (bme280_adress, & HA1, 1)

  Dig_h1 = Wert_array (1)

  Call I2c_read (bme280_adress, & HE1, 8)

  Dig_h2 = Wert_array (2)
  Shift Dig_h2, Left, 8
  Dig_h2 = Dig_h2 + Wert_array (1)

  Dig_h3 = Wert_array (3)

  Dig_h4 = Wert_array (4)
  Shift Dig_h4, Left, 4
  X = Wert_array (5)
  Shift Dig_h4, Left, 4
  Shift Dig_h4, Right, 4
  Dig_h4 = Dig_h4 + X

  Dig_h5 = Wert_array (6)
  Shift Dig_h5, Left, 4
  X = Wert_array (5)
  Shift Dig_h5, Right, 4
  Dig_h5 = Dig_h5 + X

  Dig_h6 = Wert_array (7)
End Sub
```