2-point temperature adjustment of the AirChip 3000 using a programmable external device

Table of Contents

1. Foreword

The resistance [Ohm] of a Pt100 RTD is a non-linear function of temperature [˚C] and can be expressed as a 4th degree polynomial (see IEC 751 and ASTM E1137 standards):

$$R = R_0(1 + At + Bt^2 + C(t - 100)t^3)$$

R_0 = 100 Ω
A = $3.9083\ 10^{-3}$
B = $-5.7750\ 10^{-7}$
C = $-4.1830\ 10^{-12}$
t = temperature in ˚C

Notes:
- At temperature values ≥ 0˚C, the coefficient C is set to zero.
- Setting C = 0 at temperatures below freezing results in an error of less than 0.05°C down to -65°C

The AirChip 3000 uses this polynomial to convert the data from the RTD into a linear data. The default factory values used by the AirChip 3000 for the polynomial coefficients are as indicated above.

The default factory values are retained in the AirChip 3000 memory. This allows returning the AirChip 3000 to its original factory settings at any time.

Any specific Pt100 RTD can deviate from the “standard” RTD within tolerances that are defined in both the IEC 751 and ASTM E1137 standards. In addition, the linear circuit used to measure the Pt100 RTD and digitize the temperature signal can introduce an offset, which has essentially the same effect as changing the value R_0.

For a specific RTD connected to a specific measuring circuit, determining the actual values R_0, A, B and C requires measurement data at 4 different temperature reference values. In most cases this is not practical.

For adjusting the temperature measured by the AirChip 3000, ROTRONIC uses an approximation that requires measurement data at only two temperature reference values.

The approximation relies on the fact that coefficient A of the RTD polynomial has a much larger influence on the general slope of the polynomial than coefficients B and C. A new value for the digital offset and a new value for the coefficient A are calculated so as to make the measurement data provided by the AirChip 3000 agree with the corresponding temperature reference values. Coefficients B and C are left unchanged.

2. Commands used for a 2-point temperature adjustment

Document E-M-AC3000-CP describes the RO-ASCII communication protocol which is the standard protocol used by all devices based on the AirChip3000.

The calculation of both the digital offset and coefficient A required by the approximation is beyond the capabilities of the AirChip 3000 and must be carried out with the help of an external device such as a PC.

The HCA command is part of the RO-ASCII protocol and can be used to adjust the temperature measured by the AirChip 3000. Due to the limitations of the AirChip 3000, this command is limited to a 1-point temperature adjustment.

The following RO-ASCII commands are used for a 2-point temperature adjustment:

- **ERD**: read data from the AirChip 3000 Eeprom
- **EWR**: write data to the AirChip 3000 Eeprom

3. Code for a 2-point temperature adjustment

The following lines of code are written in the Visual Basic programming language (Visual Studio) and can be used for the following:

- Convert the AirChip 3000 data required to calculate a new value for both the digital temperature offset and polynomial coefficient A.
- Calculate new values for both the digital temperature offset and polynomial coefficient A so as to make the measurement data provided by the AirChip 3000 agree with the corresponding temperature reference values.
- Generate the commands that will write the new values to the AirChip.

The AirChip 3000 UART interface is used for the 2-way communication between the external device and the AirChip 3000. When available any of the following interfaces may also be used: RS-232, RS-485 or TCP/IP.

Communication between the AirChip 3000 device and the external device is not included in the lines of code and must be programmed separately by the user. Similarly the capture of the two temperature measurements and of the corresponding temperature reference values is also not included.

Comment lines are used to describe the commands and answers required to perform a 2-point temperature adjustment.

Please note the following:

- All temperature values should be in °C
- TM1 is the temperature data read at the reference temperature TR1. TM2 is read at the reference temperature TR2
- TR1 is the lowest of the two temperature reference values (TR1 < TR2) and both temperatures are assumed to be ≥ 0 °C
- Coefficients B and C of the Pt100 RTD polynomial are left unchanged.
- The RO-ASCII communication protocol is the default protocol used by all AirChip 3000 devices and is described in document E-M-AC3000-CP

Variables		**Notes**
TR1, TR2	Input	Temperature reference values
TM1, TM2	Input	Temperature measured by the AirChip 3000 at TR1 and TR2
PT100CoeffA	Input / Output	PT100 Coefficient A [float32, Eeprom Address 1295]
PT100CoeffB	Input	PT100 Coefficient B [float32, Eeprom Address 1299]
TempOffset	Input / Output	Temperature Digital Offset [Counts / Ohm] [float32, Eeprom Address 1278]
TempConv	Input	Conversion coefficient [Counts / Ohm] [float32, Eeprom Address 1287]

```
' Declaration of the variables:

Dim TR1 As Single
Dim TR2 As Single
Dim TM1 As Single
Dim TM2 As Single
Dim R2 As Single
Dim PT100CoeffA As Single
Dim PT100CoeffB As Single
Dim TempOffset As Single
Dim TempConv As Single
Dim Count1 As Integer
Dim Count2 As Integer
Dim ResponseString As String
Dim CommandString As String

' 1) Read and capture internal data from the AirChip 3000 Eeprom:

' 1.1) Read PT100 Coefficient A:

' Send the following command string to the AirChip 3000: { 99ERD 0;1295;004}
' Write the answer string to the variable ResponseString
  example: { 99erd 050;017;128;059;Z
' Convert the string to a Single variable

Dim Data(3) as Byte
Data(0) = CByte(ResponseString.Substring(8))
Data(1) = CByte(ResponseString.Substring(12))
Data(2) = CByte(ResponseString.Substring(16))
Data(3) = CByte(ResponseString.Substring(20))

PT100CoeffA = BitConverter.ToSingle(Data, 0)

' 1.2) Read PT100 Coefficient B:

' Send the following command string to the AirChip 3000: { 99ERD 0;1299;004}
' Write the answer string to the variable ResponseString
' example: { 99erd 127;005;027;181;V
' Convert the string to a Single variable

Dim Data(3) as Byte
Data(0) = CByte(ResponseString.Substring(8))
Data(1) = CByte(ResponseString.Substring(12))
Data(2) = CByte(ResponseString.Substring(16))
Data(3) = CByte(ResponseString.Substring(20))

PT100CoeffB = BitConverter.ToSingle(Data, 0)

' 1.3) Read the temperature digital offset:

' Send the following command string to the AirChip 3000:{ 99ERD 0;1278;004}
' Write the answer string to the variable ResponseString
' example: { 99erd 000;000;000;000;4
' Convert the string to a Single variable

Dim Data(3) as Byte
Data(0) = CByte(ResponseString.Substring(8))
Data(1) = CByte(ResponseString.Substring(12))
Data(2) = CByte(ResponseString.Substring(16))
Data(3) = CByte(ResponseString.Substring(20))

TempOffset  = BitConverter.ToSingle(Data, 0)
```

' 1.4) Read the temperature conversion coefficient:

' Send the following command string to the AirChip 3000: { 99ERD 0;1287;004}
' Write the answer string to the variable ResponseString
' example: { 99erd 052;254;181;067;]
' Convert the string to a Single variable

```
Dim Data(3) as Byte
Data(0) = CByte(ResponseString.Substring(8))
Data(1) = CByte(ResponseString.Substring(12))
Data(2) = CByte(ResponseString.Substring(16))
Data(3) = CByte(ResponseString.Substring(20))

TempConv = BitConverter.ToSingle(Data, 0)
```

' 2) Compute the new temperature digital offset and polynomial coefficient A

```
Count1 = CalcCount(TM1)
Count2 = CalcCount(TM2)

TempOffset = 0

For i = 0 To 8

    ' Compute new offset for temperature

    TempOffset = CalcOffset(TR1, Count1)

    ' Compute resistance of PT100 at measured temperature TM2

    R2 = Count2 / (TempConv + TempOffset)
    ' Compute new value of PT100CoeffA

    PT100CoeffA = (R2 - 100 - 100 * PT100CoeffB * TR2 ^ 2) / (100 * TR2)

Next

Private Function CalcCount(ByVal Temp As Single) As Integer

    Dim R As Single

    R = 100 * (1 + PT100CoeffA * Temp + PT100CoeffB * Temp * Temp)
    CalcCount = R * (TempConv + TempOffset)

End Function

Private Function CalcOffset(ByVal Temp As Single, ByVal Count As Integer) As Single

    Dim R As Single

    R = 100 * (1 + PT100CoeffA * Temp + PT100CoeffB * Temp * Temp)
    CalcOffset = Count / R - TempConv

End Function
```

```
' 3) Convert the new value of PT100CoeffA and TempOffset and write both values to the Eeprom

Dim Data() as Byte
Data = BitConverter.GetBytes(PT100CoeffA)

CommandString = "{ 99EWR 0;1295;"
CommandString = CommandString & Format(Data(0),"000") & ";"
CommandString = CommandString & Format(Data(1),"000") & ";"
CommandString = CommandString & Format(Data(2),"000") & ";"
CommandString = CommandString & Format(Data(3),"000") & ";"

' Send the command string to the AirChip 3000
' Answer string (example): { 99ewr OK

Dim Data() as Byte
Data = BitConverter.GetBytes(TempOffset)

CommandString = "{ 99EWR 0;1278;"
CommandString = CommandString & Format(Data(0),"000") & ";"
CommandString = CommandString & Format(Data(1),"000") & ";"
CommandString = CommandString & Format(Data(2),"000") & ";"
CommandString = CommandString & Format(Data(3),"000") & ";"

' Send the command string to the AirChip 3000
' Answer string (example): { 99ewr OK
```

4. Document Releases

Doc. Release	Date	Notes
_10	Nov. 21, 2008	Original release