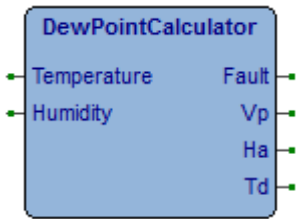
### 1.1.1 DewPointCalculator, Dew point calculator

| Type | Library |
|---|---|
| FB | |




Questo blocco funzione esegue lil calcolo della temperatura del punto di rugiada, questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 15 Min.

Fornendo il valore di temperatura ed umidità il blocco funzione calcolerà il valore di pressione di vapore saturo ***Vp***, l'umidità assoluta ***Ha*** e la temperatura del punto di rugiada ***Td***.

| | |
|---|---|
| **Temperature** (REAL) | Valore temperatura (°C) |
| **Humidity** (REAL) | Valore umidità (%) |
| **Fault** (BOOL) | Attivo in caso di errore nella gestione. |
| **Vp** (REAL) | Valore pressione di vapore saturo (mbar) |
| **Ha** (REAL) | Valore umidità assoluta (g/m3) |
| **Td** (REAL) | Valore temperatura punto di rugiada (°C) |

## Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10084020 FB protetta, terminato tempo funzionamento in modo demo.

## Esempi

Viene eseguito il calcolo del punto di rugiada acquisendo i valori di temperatura ed umidità da un sensore Modbus Sensit STH 102.

### Definizione variabili

```
VAR
  STHData : ARRAY[ 0..1 ] OF INT;    (* STH 102 data *)
  Mdb : ModbusMaster; (* Modbus master FB *)
  Serial : SysSerialPort; (* Serial port *)
  DwP : DewPointCalculator; (* Dew point calculator *)
END_VAR
```

### Esempio ST

```
  (* ---------------------------------------------------------------------- *)
  (* INITIALIZATION                                                         *)
  (* ---------------------------------------------------------------------- *)
  (* Program initialization. *)

  IF (SysFirstLoop) THEN

    (* Serial port initialization. *)

    Serial.COM:=ADR('COM2'); (* COM port definition *)
    Serial.Baudrate:=9600; (* Baudrate *)
    Serial.Parity:='N'; (* Parity *)
    Serial.DataBits:=8; (* Data bits *)
    Serial.StopBits:=1; (* Stop bits *)
    Serial.DTRManagement:=DTR_AUTO_WO_TIMES; (* DTR management *)
    Serial.DTRComplement:=FALSE; (* DTR complement *)
    Serial.EchoFlush:=FALSE; (* Received echo flush *)
    Serial.DTROnTime:=0; (* DTR On time delay (mS) *)
    Serial.DTROffTime:=0; (* DTR Off time delay (mS) *)
    Serial.FlushTm:=0; (* Flush time (mS) *)
    Serial.RxSize:=0; (* Rx buffer size *)
    Serial.TxSize:=0; (* Tx buffer size *)

    (* Modbus master initialization. *)

    Mdb.SpyOn:=TRUE; (* Spy On *)
    Mdb.Type:=0; (* Modbus type *)
    Mdb.Node:=1; (* Device modbus node *)
    Mdb.FCode:=16#03; (* Modbus function code *)
    Mdb.Address:=10; (* Modbus register address *)
    Mdb.Points:=SIZEOF(STHData)/2; (* Modbus register points *)
    Mdb.Buffer:=ADR(STHData); (* Memory buffer address *)
    Mdb.IFTime:=3430; (* Interframe time (uS) *)
    Mdb.Timeout:=100; (* Timeout time (mS) *)
    Mdb.Delay:=10; (* Delay time (mS) *)
  END_IF;

  (* ---------------------------------------------------------------------- *)
  (* MODBUS MANAGEMENT                                                      *)
  (* ---------------------------------------------------------------------- *)
  (* Manage the modbus master communication. *)

  Serial(Open:=TRUE); (* Serial port management *)
  Mdb.File:=Serial.File; (* File pointer *)
  Mdb(); (* Modbus master *)
  Mdb.Enable:=Serial.Opened AND NOT(Mdb.Done); (* Modbus enable *)

  (* ---------------------------------------------------------------------- *)
  (* DEW POINT CALCULATION                                                  *)
  (* ---------------------------------------------------------------------- *)
  (* Calculate the dew point. *)

  DwP.Temperature:=TO_REAL(STHData[0])/100.0; (* Temperature (°C) *)
  DwP.Humidity:=TO_REAL(STHData[1])/100.0; (* Humidity (%) *)
  DwP(); (* Dew point calculation *)

(* [End of file] *)
```