

ALGORITHMS USED BY ELECTRONIC LOGBOOKS FOR THE COMPUTATION OF DEW POINT TEMPERATURE

Name of e-logbook	SEAS
Agency	NOAA National Data Buoy Center (NDBC)
Country	USA
Contact point (name, e-mail)	John Wasserman <John.Wasserman@noaa.gov>
Web site for the software	http://www.aoml.noaa.gov/phod/goos/seas/amverseas_software.php
Version number of e-logbook	9.1
Date of version of e-logbook	May 2013
Version number of algorithm	all used formulas are taken from WMO-No.8, 7th edition, 2008, Part I, Annex 4.B
Date of version of algorithm	February 2013
Name of algorithm	CalculateDewPointWMO

Variables used as input for the computation of Dew Point Temperature		
Name	Units	Precision required
t_dry	deg C	0.1
t_wet	deg C	0.1
pressure	hPa	

Variables returned by the algorithm		
Name	Units	Resulted precision
dewPoint	deg C	0.1

Pseudocode of the algorithm¹
<pre> ; function to calculate the dewpoint from the given values for ; pressure, dry-bulb and wet-bulb temperature ; ; all used formulas are taken from ; WMO-No.8, 2008 edition, updated 2010, Part I, Annex 4.B ; ; IN: pressure - pressure of moist air in hPa ; t_dry - dry-bulb temperature in °C ; t_wet - wet-bulb temperature in °C ; iced - measurements over ice - boolean value: yes / no ; OUT: Dewpoint - dewpoint in °C ; Function Dewpoint(pressure, t_dry, t_wet, iced) ; interim values, for better readability of code ; ; pressure dependency f_p = 1.0016 + 0.00000315 * pressure - 0.074 / pressure ; psychrometric coefficients (for Assmann psychrometer) if (NOT iced)then </pre>

¹ : Possibly based on C++ alike syntax whenever possible; otherwise using original source language that was used

```

    psychrometricCoeff = 0.000653
else
    psychrometricCoeff = 0.000575
endif

If (NOT iced) then
; saturation and actual vapour pressure
    SatVapourPressure = f_p * 6.112 * Exp(17.62 * t_wet / (243.12 + t_wet))
    VapourPressure = SatVapourPressure -
        psychrometricCoeff * (1 + 0.000944 * t_wet) * pressure * (t_dry - t_wet)
; calculation of dewpoint
    Dewpoint = (243.12 * Log(VapourPressure / (6.112 * f_p))) /
        (17.62 - Log(VapourPressure / (6.112 * f_p)))
else
    SatVapourPressure = f_p * 6.112 * Exp(22.46 * t_wet / (272.62 + t_wet))
    VapourPressure = SatVapourPressure - psychrometricCoeff * pressure *
        (t_dry - t_wet)
    Dewpoint = (272.62 * Log(VapourPressure / (6.112 * f_p))) /
        (22.46 - Log(VapourPressure / (6.112 * f_p)))
endif

; return result
return Dewpoint

End Function

```

Implementation:

```

// function calculates the pressure dependency of the
// saturation vapour pressure
//
// IN:  pressure - given pressure for the calculation
// OUT: f_p      - pressure dependency
double CMetDataLoggerDoc::PressureDependency(const double& pressure)
{
    return 1.0016 + 0.00000315 * pressure - 0.074 / pressure;
};

// function calculates and returns the saturation vapour pressure
// in the pure phase with regard to water (e_w) or
// with regard to ice (e_i) at the given temperature
//
// IN:  pressure      - given pressure for the calculation
//      temperature   - given temperature for the calculation
//      of the saturation vapour pressure
//      iced          - measurements over ice yes / no
// OUT: satVapourPressure - saturation vapour pressure
double CMetDataLoggerDoc::SatVapourPressure(const double& pressure, const double& temperature, const bool& iced)
{
    double satVapourPressure;
    double f_p = PressureDependency(pressure); // pressure dependency

    if (iced == false)
        satVapourPressure = f_p * 6.112 * exp(17.62 * temperature / (243.12 + temperature));
    else
        satVapourPressure = f_p * 6.112 * exp(22.46 * temperature / (272.62 + temperature));

    return satVapourPressure;
};

// function calculates the actual vapour pressure (e_prime)
//
// IN:  pressure      - pressure of moist air in hPa
//      t_dry         - dry-bulb temperature in °C
//      t_wet         - wet-bulb temperature in °C
//      iced          - measurements over ice yes / no
// OUT: vapourPressure - actual vapour pressure
double CMetDataLoggerDoc::VapourPressure(const double& pressure, const double& t_dry, const double& t_wet, const bool& iced)
{
    double vapourPressure;
    double satVapourPressure = SatVapourPressure(pressure, t_wet, iced);

    if (iced == false)
        vapourPressure = satVapourPressure - 0.000653 * (1 + 0.000944 * t_wet) * pressure * (t_dry - t_wet);
    else
        vapourPressure = satVapourPressure - 0.000575 * pressure * (t_dry - t_wet);

    return vapourPressure;
};

```

```

// function to calculate the dewpoint from the given values for
// pressure, dry-bulb and wet-bulb temperature
//
// all used formulas are taken from
// WMO-No.8, 7th edition, 2008, Part I, Annex 4.B
//
// IN:  pressure - pressure of moist air in hPa (pressure at the height of the temperature measurements.)
//      t_dry    - dry-bulb temperature in °C
//      t_wet    - wet-bulb temperature in °C
// OUT: dewpoint - dewpoint in °C
// Returns FALSE if the dew point can not be computed.
//
|BOOL CMetDataLoggerDoc::CalculateDewPointWMO(const double& t_dry, const double& t_wet, const double& pressure, double& dewPoint)
{
    BOOL bReturn = TRUE;

    try
    {
        // iced - measurements over ice yes / no
        bool iced;
        if(t_wet <= 0)
            iced = true;
        else
            iced = false;

        double ln_value = VapourPressure(pressure, t_dry, t_wet, iced) / (6.112 * PressureDependency(pressure)); // interim value,
        if (ln_value <= 0)
            ln_value = 0.00000001;

        if (iced == false)
            dewPoint = (243.12 * log(ln_value)) / (17.62 - log(ln_value));
        else
            dewPoint = (272.62 * log(ln_value)) / (22.46 - log(ln_value));

        // Checks a given double-precision floating-point value for not a number (NaN).
        if (_isnan(dewPoint) != 0)
            bReturn = FALSE; // Dewpoint NaN;
    }
    catch (...)
    {
        bReturn = FALSE;
    };

    return bReturn;
};

```