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://seas.amverseas.noaa.gov/seas/ (soon to be reformed)
Version number of e-logbook	8.0 (9.0 version is currently available for beta testing by U.S. VOS personnel, Date of version 9.0 March 2011)
Date of version of e-logbook	September 2008
Version number of algorithm	-(All versions of SEAS use the same algorithm)
Date of version of algorithm	Jan 1977
Name of algorithm	DEWPT

Variables used as input for the computation of Dew Point Temperature		
Name	**Units**	**Precision required**
wet-bulb temperature	deg C	0.1
dry-bulb temperature	deg C	0.1

Variables returned by the algorithm		
Name	**Units**	**Resulted precision**
Dew point temperature	deg C	0.1

Pseudocode of the algorithm[1]

SEAS uses the DEWPT routine benchmarked at ERL in the early 1980s. The function calculates the dew point given the water vapor pressure. The routines ESILO and ESLO compute the water vapor over ice and liquid respectively. Original FORTRAN code can be found at http://wahiduddin.net/calc/density_algorithms.htm .
Below is the C version of the calculations performed by SEAS.

```
#include <stdio.h>
#include <io.h>
#include <math.h>
#include <windows.h>
#include <winbase.h>
int true = 1;
int false = 0;
FILE *dpt;

float dewpt( float , float );
 void main (argv,argc)
        {
        int i, j, k1,  array[47] [23];
        float W, D,dp, jj;

        dpt = fopen("d:\\dewptc.txt", "w");
        for (i=0; i < 47; i++)
                {
                for (j = 0; j< 23; j++)
                        array[i][j]= 909;
```

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

```
                }

        for (i=-16; i<31; i++)
                        {
                        for (jj=0.0; jj < 11.5; jj +=0.5)
                                {
                                W = (float) i;
                                D = (float) i + jj;
                                dp = dewpt(D, W);
                                k1 = (int)(jj*2.0);
                                if(dp < 0)
                                        array[i +16] [k1] =  (int) (dp-0.5);
                                else
                                        array[i +16] [k1] =  (int) (dp+0.5);
                                }
                        }

// Check WB > DB

        fprintf(dpt,"%6.2f\n", dewpt( 10.0, 12.0));

                        for(i=0;i<47; i++)
                                {
                                fprintf(dpt,"%4d",i-16);
                                for (j=0;j<23;j++)
                                        {
                                        fprintf(dpt,"%4d",array[i][j]);
                                        }
                                fprintf(dpt,"\n");
                                }

        }


float dewpt( float DryBulb, float WetBulb)
        {

//    Paul R. Lowe, "An Approximating Polynomial for the Computation of
//    Saturation Vapor Pressure", Journal of Applied Meterology, Vol 16, No 1
//    January 1977, pp 100-103.

//        Will return a (float) dew point when passed a drybulb and a wetbulb temperature in Degrees C.
//  Returns 999 if the dew point can not be computed

//    dew point based on casio program  code
//       coefficients for saturation vapor pressure
//      with respect to ice,    with respect to water

                double coef[7][2] = {6.109177956,       6.107799961,
            5.03469897E-1,     4.436518521E-1,
            1.886013408E-2,    1.428945805E-2,
            4.176223716E-4,    2.650648471E-4,
            5.824720280E-6,    3.031240396E-6,
            4.838803174E-8,    2.034080948E-8,
            1.838826904E-10,   6.136820929E-11};

                double p,q,w,dif,deweq;
                int i;
                int ice;

            if(WetBulb <= 0)
                    ice = true;
            else
```

```
                    ice = false;


    w=WetBulb;
    dif=DryBulb - WetBulb;
        if (dif < 0.0)
                    {
                    return 999.0;
                    }
    i=1;
    if(ice)i=0;
    p=coef[0][i]+w*(coef[1][i]+w*(coef[2][i]+w*(coef[3][i]+w*(coef[4][i]+
                w*(coef[5][i]+w*coef[6][i])))));

//   use standard atmosphere of 29.92 inches of mercury

    p=p-(1013.20789*dif*(0.00066*(1.+w* 0.00115)));
    if(p >= 0.0)
                    {
                        q=log(p);
                        deweq= (243.5*q-440.8)/(19.48-q);
                        return (float) deweq;
                    }
    else
      return 999.0;

        }
```

This is the algorithm that is implemented:

```
BOOL CMetDataLoggerDoc::CalculateDewPoint(const double& dDryBulb,                  // in, Celcius degrees
                                          const double& dWetBulb,                  // in, Celcius degrees
                                          double& dDewPoint)                       // out, Celcius degrees
{
        BOOL bResult = FALSE;
        double dDifference;
        dDifference = dDryBulb - dWetBulb;

        if (dDifference < 0.0)
                // 999 if the dew point can not be computed
                dDewPoint = 999.;
        else{
                // dew point based on casio program code coefficients for saturation vapor pressure
                // with respect to ice, with respect to water
                //                          ice                 water
                double coef[7][2] = {  6.109177956,       6.107799961,
                                       5.034698970E-1,    4.436518521E-1,
                                       1.886013408E-2,    1.428945805E-2,
                                       4.176223716E-4,    2.650648471E-4,
                                       5.824720280E-6,    3.031240396E-6,
                                       4.838803174E-8,    2.034080948E-8,
                                       1.838826904E-10,   6.136820929E-11};
                int i;
                if(dWetBulb <= 0)
                        i = 0;
                else
                        i = 1;

                double p;
                p = coef[0][i] +
                        dWetBulb*(coef[1][i] + dWetBulb*(coef[2][i] +
                        dWetBulb*(coef[3][i] + dWetBulb*(coef[4][i] +
                        dWetBulb*(coef[5][i] + dWetBulb*coef[6][i])))));

                // use standard atmosphere of 29.92 inches of mercury
                p = p - (1013.20789 * dDifference * (0.00066 * (1. + dWetBulb * 0.00115)));
                if(p >= 0.0){
                        double q = log(p);
                        dDewPoint = (243.5 * q - 440.8) / (19.48 - q);
                        bResult = TRUE;
                }
                else
                        dDewPoint = 999.;
        };

        return bResult;
};
```