UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Report

Analysis and Development of an Object-Oriented Library Structure for Solar Thermal Power Plants – Documentation of a Sun Position and Weather Data Script for Flownex

Georg F. Schneider[1], Paul Gauché[2]

October 27, 2012

Table of Contents

1. A Sun Position and Weather Data Script for Flownex .... - 2 -
2. Solar equations .... - 3 -
3. Excel spread sheet to access external data .... - 4 -
4. Literature .... - 5 -
Appendix .... - 6 -
A. C# Source code of Solar Position Script .... - 6 -

[1] Bsc (Mech), Visiting research student at Stellenbosch University, RWTH Aachen University, Templergraben 55, 52062 Aachen, Germany, Phone: +49 177 88 96 77 0, Email: Georg.Schneider@rwth-aachen.de
[2] Meng (Mech), Sr. Researcher and Director STERG, Dept. Mechanical and Mechatronic Engineering, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa, Phone: +27 21 808 4242, Email: paulgauche@sun.ac.za

Solar Thermal Energy Research
Mechanical & Mechatronic Engineering
in association with CRSES

Solar Thermal Energy Research Group

Nomenclature

Latin letters	
$\boldsymbol{B}$	Value in **Equation 2-3** [degree]
$\boldsymbol{E}$	Rotation speed correction according to **Equation 2-3** [min]
$\boldsymbol{N}$	Day of the year [d]
$\boldsymbol{t}$	Time [h]
Greek letters	
$\boldsymbol{\alpha}$	Azimuth angle of the sun [degree]
$\boldsymbol{\gamma}$	Elevation angle of the sun [degree]

$\boldsymbol{\delta}$	Declination of the sun [degree]
$\boldsymbol{\lambda}$	Longitude [degree]
$\boldsymbol{\varphi}$	Latitude of plant position [degree]
$\boldsymbol{\omega}$	Hour angle of the sun [degree]
Suffices	
$\boldsymbol{loc}$	Legal (watch), local
$\boldsymbol{sol}$	Solar
$\boldsymbol{std}$	Standard of time zone
$\boldsymbol{s}$	sun

1. A Sun Position and Weather Data Script for Flownex

Due to the fact that Flownex SE lacks a possibility to calculate the sun position an own script has to be written. An implementation was made and this document aims to explain the underlying math. For the use in other components and models the script calculates as an output hour angle, elevation angle, azimuth angle of the sun and solar time. Figure 2-1 shows the used coordinate system to derive the equations in the next section. Please note that the Flownex library owns two scripting components, one “Iterative script” solved at steady state and one “Script” used in transient simulations.

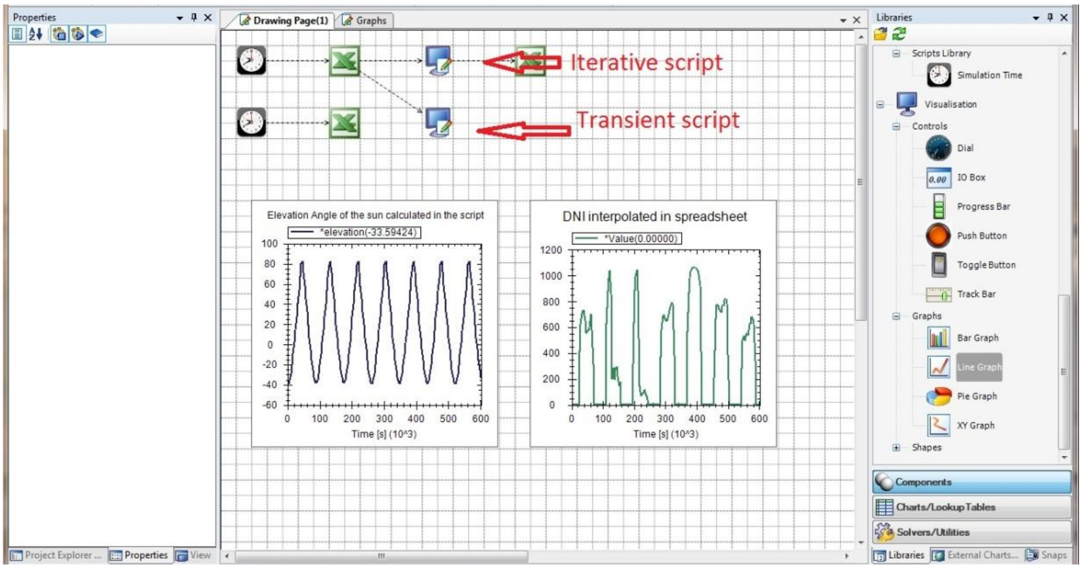

Figure 1-1 Screenshot of Flownex showing the necessary data links

2. Solar equations

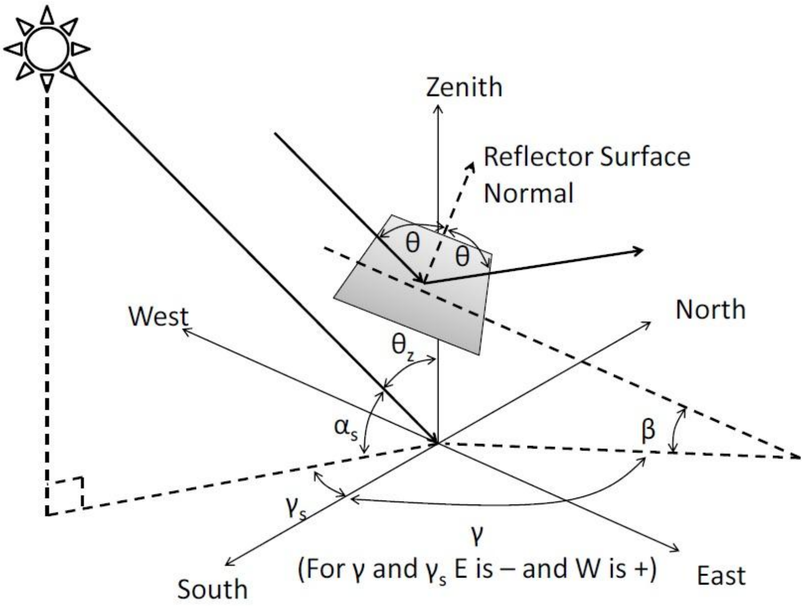

Figure 2-1 Horizontal coordinate system (Gauché, Pfenninger, Meyer, von Backström, & Brent, 2012)

The following equations are used to calculate the azimuth and elevation angle of the sun in a horizontal coordinate system (As seen in Figure 2-1).

The declination, δ, is calculated using the following equation (Cooper, 1969).

$$\delta = 23.45 \times \sin\left(\frac{360\ deg}{365\ d} \times (284 + N)\right) \qquad \textbf{Equation 2-1}$$

The unit of declination angle δ is degrees and the unit of Day N is days. The minimum and maximum value of the declination is -23.45 degree and 23.45 degree, respectively. Conventions are defined for a southern hemisphere winter, where the declination is bigger than zero and for a northern hemisphere winter declination is smaller than zero.

The apparent solar time T_{sol} is derived by the following equations (Duffie & Beckman, 2006; Iqbal, 1983; Spencer, 1972).

$$t_{sol} = t_{loc} + 4\ \frac{min}{deg} \times (\lambda_{loc} - \lambda_{std}) + E \qquad \textbf{Equation 2-2}$$

$$E = 9.87 \times \sin(2B) - 7.53 \times \cos(B) - 1.5 \times \sin(B)\ , E\ in\ [min] \qquad \textbf{Equation 2-3}$$

$$B = \left((N - 81) \times 0.989\right), \qquad B\ in\ [deg] \qquad \textbf{Equation 2-4}$$

t_{loc} is the legal (watch) time in hours, $\boldsymbol{\lambda_{loc}}$ is the local longitude in degrees, $\boldsymbol{\lambda_{std}}$ is the standard longitude of the corresponding time zone in degrees, *E* is the rotation speed correction in minutes. Unit of *B* is degrees, Unit of Day of the year ***N*** is in days.

The hour angle of the sun is calculated using the following equation (Duffie & Beckman, 2006).

$$\boldsymbol{\omega = \frac{360}{24}\frac{deg}{h} * \left(12 - t_{sol}\right)}$$ **Equation 2-5**

The hour angle ω is in degrees t_{sol} is in hours. The following definitions are made: At zenith the hour angle equals zero, in the morning the value is bigger than zero, in the afternoon the hour angle is smaller than zero.

The elevation angle of the sun is calculated using the following equation (Duffie & Beckman, 2006).

$$\boldsymbol{\gamma_s = arcsin\{sin(\varphi) * sin(\delta) + sin(\varphi) * sin(\delta) * sin(\omega)\}}$$ **Equation 2-6**

γ_s is the elevation angle in degrees, φ is the latitude of the plant position in degrees, δ is the declination of the sun in degrees. ω is the hour angle in degrees.

The azimuth angle α_s of the sun is calculated using the following equation (Duffie & Beckman, 2006).

$$\boldsymbol{\alpha_s = arccos\left\{\frac{[sin(\varphi) * sin(\gamma) - sin(\delta)]}{[cos(\varphi) * cos(\gamma)]}\right\}}$$

Equation 2-7

φ is the latitude of the plant location in degrees, δ is the declination in degrees, γ is the elevation angle of the sun in degrees and ω is the hour angle in degrees. Note that the value is zero to the south, positive towards east and negative towards west.

3. Excel spread sheet to access external data

To store the measured data and the necessary parameters the Excel component is used.

It needs as an input the simulation time of the current simulation. The local longitude, the standard longitude of the current time zone and the local latitude have to be specified. The experimental data can be copied in the spread sheet.

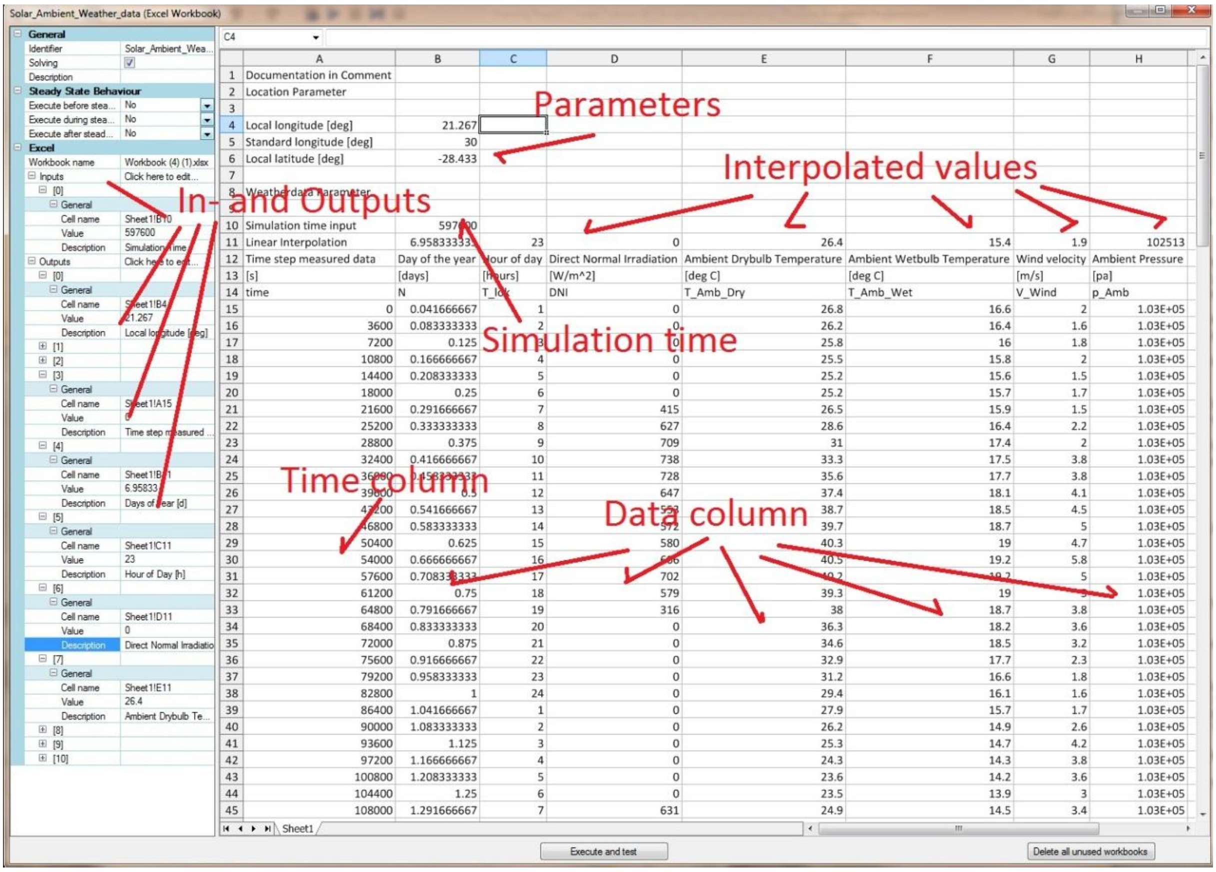

Figure 3-1 Screenshot of the Spreadsheet

The time dependent data is read by using linear interpolation, interpolating the measured value depending on the current simulation time step.

The methodology is as follows. The Excel function FORECAST and OFFSET are used.

InterpolatedValue=FORECAST(*NewX*,OFFSET(*KnownY*,MATCH(*NewX*,*KnownX*,1)-1,0,2), OFFSET(*KnownX*,MATCH(*NewX*,*KnownX*,1)-1,0,2))

InterpolatedValue is the aimed value. NewX is the current simulation time step, KnownY is the column containing the measured values and KnownX is the column containing the time.

The function was derived by this source:

http://www.blueleafsoftware.com/Products/Dagra/LinearInterpolationExcel.php

4. Literature

Cooper, P. I. (1969). Solar Energy. *Solar Energy*, *12*(3), 333-346. doi:10.1016/0038-092X(69)90047-4

Duffie, J. A., & Beckman, W. A. (2006). *Solar Engineering of Thermal Processes* (3rd ed.). John Wiley and Sons.

Gauché, P., Pfenninger, S., Meyer, A. J., von Backström, T. W., & Brent, A. C. (2012). MODELING DISPATCHABILITY POTENTIAL OF CSP IN SOUTH AFRICA. *Proceedings of the 1th Southern African Solar Energy Conference* (Vol. 2011). 21-23 May, Stellenbosch, South Africa.

Iqbal, M. (1983). *An Introduction to Solar Radiation.* Toronto: Academic.

Spencer, J. W. (1972). Fourier Series Representation of the Position of the Sun. *Search*, *2*(5), 172.

Appendix

A. C# Source code of Solar Position Script

This source code can be directly copied and paced in the script components of Flownex.

-------------- start copy below here -----------------

```
//script using directives

//css_ref IPS.Core.dll;

//css_ref IPS.PluginInterface.dll;

//css_ref IPS.Units.dll;

// Copyright (c) 2012 for this script by Georg Ferdinand Schneider

// All rights reserved.


using System;

using IPS.Properties;

using IPS.Scripting;


//script must be derived from IComponentScript

public class Script: IPS.Scripting.IComponentScript
```

```
{
        IPS.Properties.Double PI;
        IPS.Properties.Double _N;
        IPS.Properties.Double _Declination;
        IPS.Properties.Double _B;
        IPS.Properties.Double _E;
        IPS.Properties.Double _T_S;
        IPS.Properties.Double _t_solar;
        IPS.Properties.Double _T_lok;
        IPS.Properties.Double _longitude_lok;
        IPS.Properties.Double _longitude_std;
        IPS.Properties.Double _latitude;
        IPS.Properties.Double _hourangle;
        IPS.Properties.Double _elevation;
        IPS.Properties.Double _azimuth;

        //do pre simulation initialisation here
        public override void Initialise()
        {
 }

        //do post simulation cleanup here
        public override void Cleanup()
        {
 }

 //script main execution function - called every cycle
        public override void Execute(double Time)
        {
```

/*************************** Solar equations ***************************/

/*

The angles and values are calculated assuming a horizontal coordinate system.

Reference point is south and reference plane is the horizontal plane.

*/

// Equation to calculate the DECLINATION of the sun

_Declination.Value = 23.45 * Math.Sin(PI.Value / 180 * 360 / 365 * (284+_N.Value));

/* Explanation:

Equation to calculate the declination angle as a function of day of the year as proposed by Cooper (1969) - The Absorption of Radiation in Solar Stills, Solar Energy, Vol. 12, pp 333-346.

Unit of declination angle: [degree], Unit of Day of the year N: [d]

Minimum value -23.45 degree, Maximum value: 23.45 degree

Conventions: Southern Hemisphere Winter decl > 0 Northern Hemisphere Winter decl < 0

*/

// Equation to calculate the B VALUE

_B.Value = (_N.Value - 81) * 0.989 * PI.Value / 180;

/* Explanation

Function to calculate the B Value for Function E

Unit of B: [rad], Unit of Day: [d]

Source: Duffie, J. A.; Beckman W. A. - Solar Engineering of Thermal Processes, 3rd Edition, 2006, John Wiley and Sons, Eq. (1.4.2)

*/

// Equation to calculate the E VALUE

_E.Value = 9.87 * Math.Sin(2*_B.Value) - 7.53 * Math.Cos(_B.Value) - 1.5 * Math.Sin(_B.Value);

/* Explanation

Equation to calculate the changing of the earth rotation velocity

Unit E [min], Unit B [rad]

From: Spencer, J. W. - Fourier Series Representation of the Position of the Sun., Search, 2 (5), p. 172, 1972

as cited by Iqbal, M. - An Introduction to Solar Radiation., Academic, Toronto, 1983

as cited by Duffie, J. A.; Beckman W. A. - Solar Engineering of Thermal Processes, 3rd Edition, 2006, John Wiley and Sons, Eq. (1.5.3)

*/

// Equation to calculate the APPARENT SOLAR TIME and SOLAR TIME

_T_S.Value = _T_lok.Value + (4 * (_longitude_lok.Value - _longitude_std.Value))/60 + _E.Value/60;

_t_solar.Value = 12 - _T_S.Value;

/*

Equation that calculates the apparent solar time

T_lok is the legal time in [h]

longitude_lok is the lokal longitude in [degree]

longitude_std is the standard longitude of the corresponing time zone [degree]

E is the rotation speed correction [min]

Source: Duffie, J. A.; Beckman W. A. - Solar Engineering of Thermal Processes, 3rd Edition, 2006, John Wiley and Sons, Eq. (1.5.2)

*/

// Equation to calculate the HOURANGLE

_hourangle.Value =(12 - _T_S.Value) * 360/24;

/*

Equation calculates the hour angle as a function of solar time

Unit hourangle [degree], Unit T_S [h]

Definitions: At zenith hourangle = 0, morning hourangle > 0, afternoon hourangle < 0

*/

// Equation to calculate the ELEVATION ANGLE

_elevation.Value = 180/PI.Value * Math.Asin(Math.Sin(PI.Value/180 * _latitude.Value) * Math.Sin(PI.Value/180 * _Declination.Value) + Math.Cos(PI.Value/180 * _latitude.Value) * Math.Cos (PI.Value/180 * _Declination.Value) * Math.Cos(PI.Value/180 * _hourangle.Value));

```
/*

Equation that calculates the elevation angle of the sun in a horizontal coordinate system

elevation in [degree]

latitude in in [degree]

declination in [degree]

hourangle in [degree]

Source: Duffie, J. A.; Beckman W. A. - Solar Engineering of Thermal Processes, 3rd Edition, 2006, John Wiley and Sons

*/

        // Equation to calculate the AZIMUTH ANGLE

// If clause necessary to provide negative values in the afternoon

if ( _hourangle.Value > 0)

_azimuth.Value = 180 / PI.Value * Math.Acos((Math.Sin(PI.Value/180 * _latitude.Value)*Math.Sin(PI.Value/180 * _elevation.Value) - Math.Sin(PI.Value/180 * _Declination.Value))/(Math.Cos(PI.Value/180 * _latitude.Value)*Math.Cos(PI.Value/180 * _elevation.Value)));

else

_azimuth.Value = -1 * 180 / PI.Value * Math.Acos((Math.Sin(PI.Value/180 * _latitude.Value)*Math.Sin(PI.Value/180 * _elevation.Value) - Math.Sin(PI.Value/180 * _Declination.Value))/(Math.Cos(PI.Value/180 * _latitude.Value)*Math.Cos(PI.Value/180 * _elevation.Value)));

//              if ( _hourangle.Value > 0)

// _azimuth.Value = (180/PI.Value * Math.Acos((Math.Sin(PI.Value/180 * _latitude.Value)*Math.Sin(PI.Value/180 * _hourangle.Value)-Math.Sin(PI.Value/180 * _Declination.Value))/(Math.Cos(PI.Value/180 * _latitude.Value)*Math.Cos(PI.Value/180 * _hourangle.Value))));

//_azimuth.Value = 1;

//else

// _azimuth.Value = -1*(180/PI.Value * Math.Acos((Math.Sin(PI.Value/180 * _latitude.Value)*Math.Sin(PI.Value/180 * _hourangle.Value)-Math.Sin(PI.Value/180 * _Declination.Value))/(Math.Cos(PI.Value/180 * _latitude.Value)*Math.Cos(PI.Value/180 * _hourangle.Value))));

//_azimuth.Value = 2;
```

```
/*

Equation that calculates the azimuth angle of the sun in a horizontal coordinate system

azimuth in [degree]

latitude in [degree]

declination in [degree]

elevation angle in [degree]

hourangle in [degree]

Source: Duffie, J. A.; Beckman W. A. - Solar Engineering of Thermal Processes, 3rd Edition, 2006, John Wiley and Sons

*/

        }


  //any processing you want to do before steady state

        public override void ExecuteBeforeSteadyState()

        {

   Execute(0.0);

        }

   //any processing you want to do while solving steady state

        public override void ExecuteSteadyState()

        {

   Execute(0.0);

        }


  //any processing you want to do after steady state

        public override void ExecuteAfterSteadyState()

        {

   Execute(0.0);

        }
```

```
//constructer initialises parameters
public Script()
{
    PI = new IPS.Properties.Double();
    _N = new IPS.Properties.Double();
    _Declination = new IPS.Properties.Double();
    _B = new IPS.Properties.Double();
    _E = new IPS.Properties.Double();
    _T_S = new IPS.Properties.Double();
    _T_lok = new IPS.Properties.Double();
    _t_solar = new IPS.Properties.Double();
    _longitude_lok = new IPS.Properties.Double();
    _longitude_std = new IPS.Properties.Double();
    _latitude = new IPS.Properties.Double();
    _hourangle = new IPS.Properties.Double();
    _elevation = new IPS.Properties.Double();
    _azimuth = new IPS.Properties.Double();


// Start values of variables
    PI.Value = 3.14159265358979323846264338327950288419716939;
//From Pi to one Million places Website
    _N.Value = 172;
    _Declination.Value = 0;
    _B.Value = 0;
    _E.Value = 0;
    _T_S.Value = 0;
    _t_solar.Value = 0;
    _T_lok.Value = 15;
    _longitude_lok.Value = 6.08;
```

```
            _longitude_std.Value = 15;
            _latitude.Value = 50.78;
            _hourangle.Value = 0;
            _elevation.Value = 0;
            _azimuth.Value = 0;
        }

        //property declarations to make
        //parameters visible to outside world
        // Declination, B, E are not visible to the outside
        [PropertyUsage(UseProperty.DYNAMIC)]
        public IPS.Properties.Double N
        {
            get
            {
                return _N;
            }
        }
/*
        [PropertyUsage(UseProperty.DYNAMIC)]
        public IPS.Properties.Double Declination
        {
            get
            {
                return _Declination;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        public IPS.Properties.Double B
```

```
        {
                get
                {
                        return _B;
                }
        }

        [PropertyUsage(UseProperty.DYNAMIC)]
        public IPS.Properties.Double E
        {
                get
                {
                        return _E;
                }
        }

                [PropertyUsage(UseProperty.DYNAMIC)]
        public IPS.Properties.Double T_S
        {
                get
                {
                        return _T_S;
                }
        }
*/
        [PropertyUsage(UseProperty.DYNAMIC)]
        public IPS.Properties.Double t_solar
        {
                get
```

```
        {
            return _t_solar;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    public IPS.Properties.Double T_lok
    {
        get
        {
            return _T_lok;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    public IPS.Properties.Double longitude_lok
    {
        get
        {
            return _longitude_lok;
        }
    }

    [PropertyUsage(UseProperty.DYNAMIC)]
    public IPS.Properties.Double longitude_std
    {
        get
        {
            return _longitude_std;
```

```
        }
}

[PropertyUsage(UseProperty.DYNAMIC)]
public IPS.Properties.Double latitude
{
        get
        {
                return _latitude;
        }
}

[PropertyUsage(UseProperty.DYNAMIC)]
public IPS.Properties.Double hourangle
{
        get
        {
                return _hourangle;
        }
}

[PropertyUsage(UseProperty.DYNAMIC)]
public IPS.Properties.Double elevation
{
        get
        {
                return _elevation;
        }
}
```

```
        [PropertyUsage(UseProperty.DYNAMIC)]
        public IPS.Properties.Double azimuth
        {
                get
                {
                        return _azimuth;
                }
        }
}
```