SIEMENS

NX Expressions Your Way

(Making Custom DesignLogic Functions)

Taylor Anderson
NX Product Manager
(DesignLogic, Knowledge Fusion and Validation)
Wednesday, June 4th, 2008
UGS Connection Conference

NATIONAL COUNCIL OF EXAMINERS FOR ENGINEERING AND SURVEYING

FUNDAMENTALS OF ENGINEERING

(FE)

REFERENCE HANDBOOK

The relationship between the load (w), shear (V), and moment (M) equations are:

$$w = dV(x)/dx; \quad V = dM(x)/dx$$

$$V_2 - V_1 = \int_{x_1}^{x_2} w(x)\,dx$$

$$M_2 - M_1 = \int_{x_1}^{x_2} V(x)\,dx$$

Stresses in Beams

$$\varepsilon_x = -y/\rho, \text{ where}$$

ρ = the radius of curvature of the deflected axis of the beam and
y = the distance from the neutral axis to the longitudinal fiber in question.

Using the stress-strain relationship $\sigma = E\varepsilon$,

Axial Stress: $\sigma_x = -Ey/\rho$, where

σ_x = the normal stress of the fiber located y-distance from the neutral axis.

$1/\rho = M/(EI)$, where

M = the moment at the section and
I = the *moment of inertia* of the cross-section.

$$\sigma_x = -My/I, \text{ where}$$

y = the distance from the neutral axis to the fiber location above or below the axis. Let $y = c$, where c = distance from the neutral axis to the outermost fiber of a symmetrical beam section.

$$\sigma_x = \pm Mc/I$$

Let $S = I/c$: then

$$\sigma_x = \pm M/S, \text{ where}$$

S = the *elastic section modulus* of the beam member.

Shear stress: $\tau_{xy} = VQ/(Ib)$, where

τ_{xy} = shear stress on the surface,
V = shear force at the section,
b = width or thickness of the cross-section, and
$Q = A'y$, where
A' = area above the layer (or plane) upon which the desired shear stress acts and
y = distance from neutral axis to area centroid.

Deflection of Beams

Using $1/\rho = M/(EI)$,

$$EI\frac{d^2y}{dx^2} = M, \text{ differential equation of deflection curve}$$

$$EI\frac{d^3y}{dx^3} = dM(x)/dx = V$$

$$EI\frac{d^4y}{dx^4} = dV(x)/dx = -w$$

Determine the deflection curve equation by double integration (apply boundary conditions applicable to the deflection and/or slope).

I'_x, I'_y = the moment of inertia about the new axis.

Radius of Gyration

The *radius of gyration* r_x, r_x, r_z is the distance from a reference axis at which all of the area can be considered to be concentrated to produce the moment of inertia.

$$r_x = \sqrt{I_x/A}\ ;\quad r_y = \sqrt{I_y/A}\ ;\quad r_p = \sqrt{J/A}$$

Product of Inertia

The *product of inertia* (I_{xy}, etc.) is defined as:

$I_{xy} = \int xydA$, with respect to the xy-coordinate system,

$I_{xz} = \int xzdA$, with respect to the xz-coordinate system, and

$I_{yz} = \int yzdA$, with respect to the yz-coordinate system.

The *transfer theorem* also applies:

$I'_{xy} = I_{x_cy_c} + d_x d_y A$, for the xy-coordinate system, etc., where

d_x = x-axis distance between the two axes in question and
d_y = y-axis distance between the two axes in question.

FRICTION

The largest frictional force that is possible to develop is called the *limiting friction*. Any further increase in applied forces would cause motion.

$$F = \mu N, \text{ where}$$

F = friction force,
μ = *coefficient of static friction*, and
N = normal force between surfaces in contact.

SCREW THREAD

For a *screw-jack, square thread*,

$$M = Pr\tan(\alpha \pm \phi), \text{ where}$$

+ is for screw tightening,
− is for screw loosening,
M = external moment applied to axis of screw,
P = load on jack applied along and on the line of the axis,
r = the mean thread radius,
α = the *pitch angle* of the thread, and
$\mu = \tan\phi$ = the appropriate coefficient of friction.

BRAKE-BAND OR BELT FRICTION

$$F_1 = F_2 e^{\mu\theta}, \text{ where}$$

F_1 = force being applied in the direction of impending motion,

COROLLARY to Kelvin-Planck: No heat engine can have a higher efficiency than a Carnot cycle operating between the same reservoirs.

Clausius Statement of Second Law

No refrigeration or heat pump cycle can operate without a net work input.

COROLLARY: No refrigerator or heat pump can have a higher COP than a Carnot cycle refrigerator or heat pump.

VAPOR-LIQUID MIXTURES

Henry's Law at Constant Temperature At equilibrium, the partial pressure of a gas is proportional to its concentration in a liquid. Henry's Law is valid for low concentrations; i.e., $x \approx 0$.

$$P_i = Py_i = hx_i, \text{ where}$$

h = Henry's Law constant,
P_i = partial pressure of a gas in contact with a liquid,
x_i = mol fraction of the gas in the liquid,
y_i = mol fraction of the gas in the vapor, and
P = total pressure.

Raoult's Law for Vapor-Liquid Equilibrium

Valid for concentrations near 1; i.e., $x_i \approx 1$.

$$P_i = x_i P_i^*, \text{ where}$$

P_i = partial pressure of component i,
x_i = mol fraction of component i in the liquid, and
P_i^* = vapor pressure of pure component i at the temperature of the mixture.

ENTROPY

$$dS = (1/T)\,\delta Q_{rev}$$

$$S_2 - S_1 = \int_1^2 (1/T)\,\delta Q_{rev}$$

Inequality of Clausius

$$\oint (1/T)\,\delta Q \le 0$$

$$\int_1^2 (1/T)\,\delta Q \le S_2 - S_1$$

Isothermal, reversible process

$$\Delta S = S_2 - S_1 = Q/T$$

Isentropic process

$$\Delta S = 0; \quad dS = 0$$

A reversible adiabatic process is isentropic.

Adiabatic Process $\Delta S \ge 0$

Increase of Entropy Principle

$$\Delta S_{total} = \Delta S_{system} + \Delta S_{surroundings} \ge 0$$

$$\Delta \dot{S}_{total} = \Sigma\, \dot{m}_{out} s_{out} - \Sigma\, \dot{m}_{in} s_{in} - \Sigma\,(\dot{Q}_{external}/T_{external}) \ge 0$$

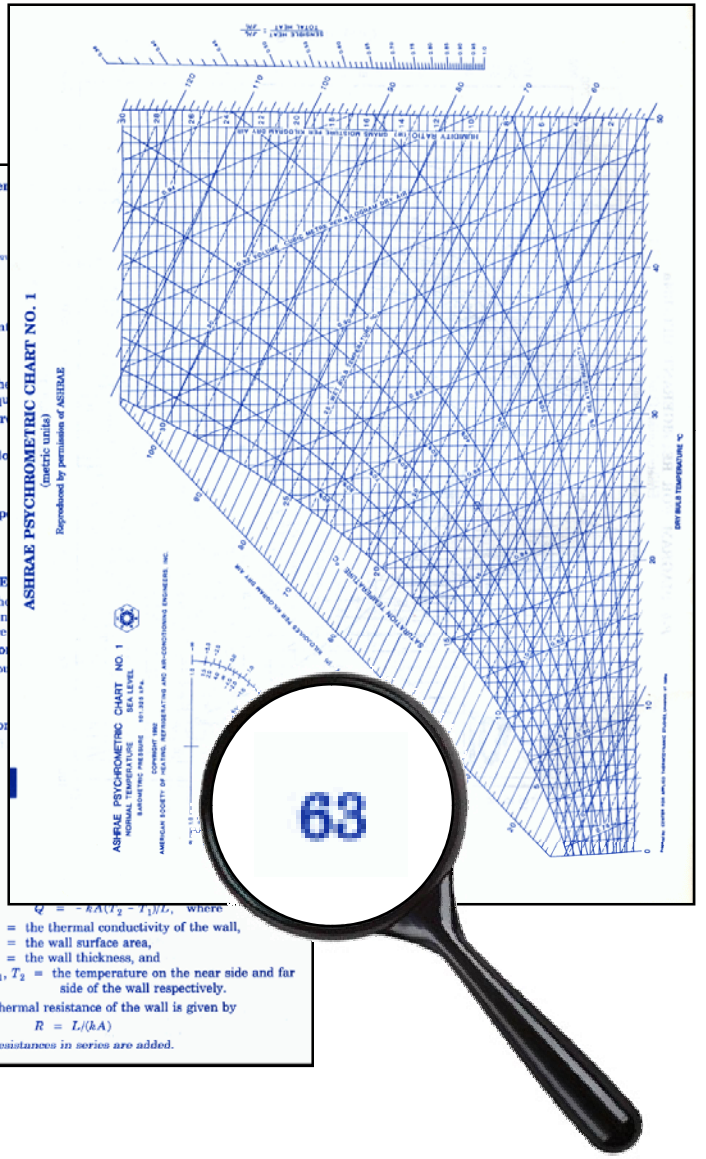
$$Q = -kA(T_2 - T_1)/L, \text{ where}$$

k = the thermal conductivity of the wall,
A = the wall surface area,
L = the wall thickness, and
T_1, T_2 = the temperature on the near side and far side of the wall respectively.

Thermal resistance of the wall is given by

$$R = L/(kA)$$

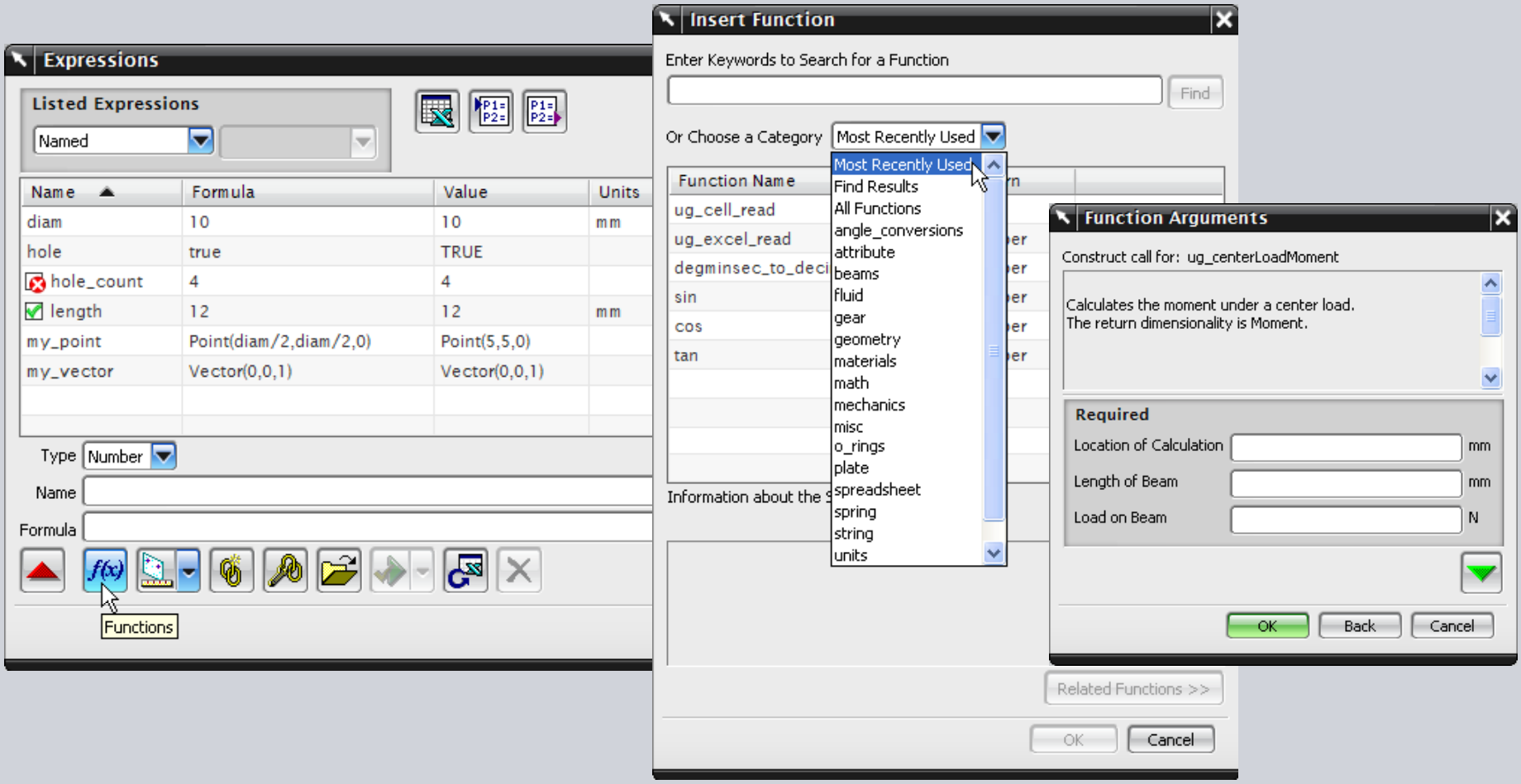
Resistances in series are added.



The Expressions Dialog:

"Where are the equations, anyway?"

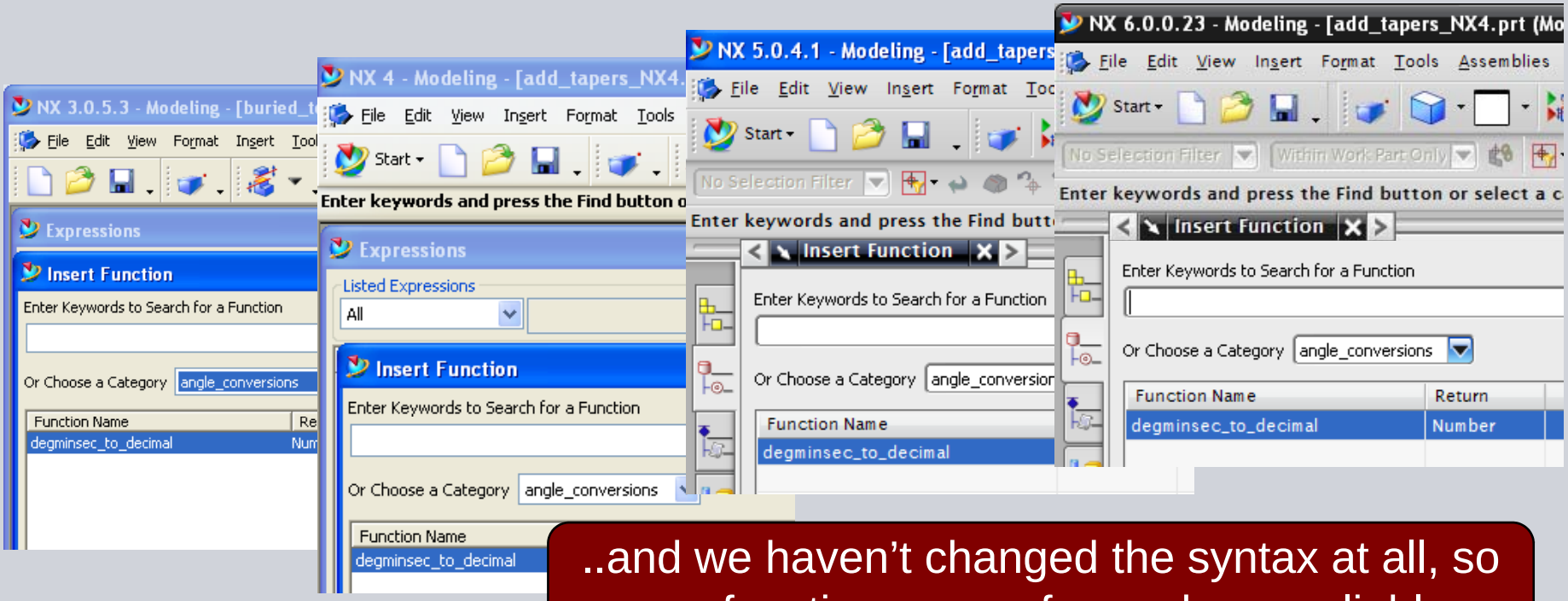
SIEMENS



SIEMENS

This is actually not a new NX 6 thing…

This became possible at the same time as the new Expressions dialog (and more importantly, the introduction of Knowledge Fusion in the background behind model expressions) in NX 3.



..and we haven't changed the syntax at all, so new functions move forward very reliably.

Demonstration

Contents of:
angle_conversions.dfa

SIEMENS

```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number  $deg;     # constant -  #.Angle Degrees Component.#
number  $min;     # constant -  #.Angle Minutes Component.#
number  $sec      # constant -  #.Angle Seconds Component.#
)
@{
   $value << $deg + ($min*(1/60)) + ($sec*(1/3600));
   ug_units_setMeasure($value, "Angle" );
} number;

#+
-------------------------------------------------------------------------------
Returns:
angle - #.Returns the decimal angle.#

See Also:
-------------------------------------------------------------------------------
#-
```

Contents of:
angle_conversions.dfa

SIEMENS

First, this filename determines the category name in the Insert Function dialog.

(If you put multiple functions in one file, they will appear under the same Category.)

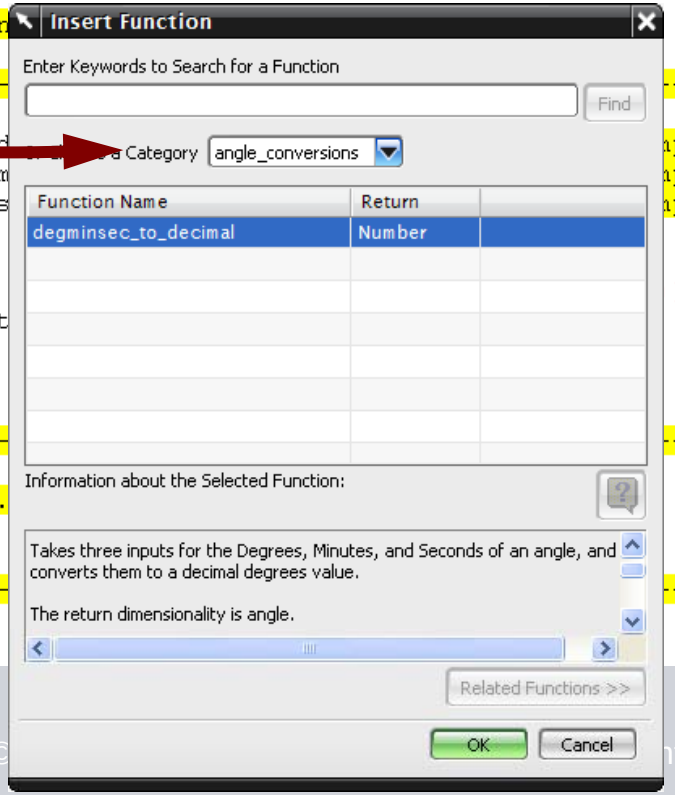
```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return
 #
-------
#-
number   $d
number   $m
number   $s
)
@(
    $value
    ug_unit
) number;

#+
----------
Returns:
angle - #.

See Also:
----------
#-
```



Contents of:
angle_conversions.dfa

SIEMENS

Knowledge Fusion version statement (This needs to appear only once, and must be in the first line of the file.)

```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number   $deg;     # constant -   #.Angle Degrees Component.#
number   $min;     # constant -   #.Angle Minutes Component.#
number   $sec      # constant -   #.Angle Seconds Component.#
)
@(
   $value << $deg + ($min*(1/60)) + ($sec*(1/3600));
   ug_units_setMeasure($value, "Angle" );
) number;

#+
-------------------------------------------------------------------------------
Returns:
angle - #.Returns the decimal angle.#

See Also:
-------------------------------------------------------------------------------
#-
```

Contents of:
angle_conversions.dfa

SIEMENS

Name of your new function (This is OUTSIDE the comments.)

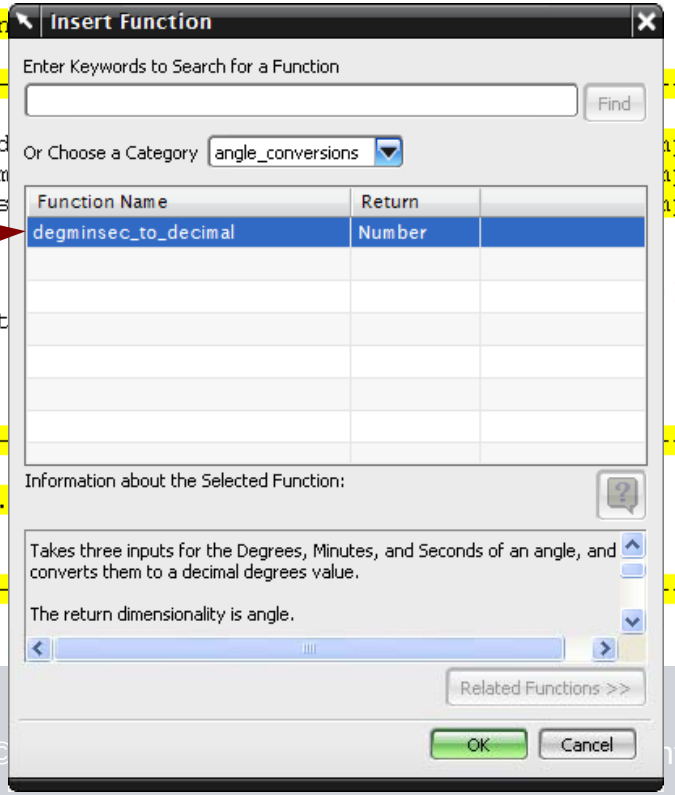
```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return
.#
----------                                                    ----------
#-
number   $d                                                  ponent.#
number   $m                                                  ponent.#
number   $s                                                  ponent.#
)
@(
    $value                                                    );
    ug_unit
) number;

#+
----------                                                    ----------
Returns:
angle - #.

See Also:
----------                                                    ----------
#-
```



Contents of:
angle_conversions.dfa

SIEMENS

States that this function SHOULD be visible to DesignLogic (…should show up in expressions.)

If this equals “no”, then the function will only be visible to Knowledge Fusion.

```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number   $deg;     # constant -   #.Angle Degrees Component.#
number   $min;     # constant -   #.Angle Minutes Component.#
number   $sec      # constant -   #.Angle Seconds Component.#
)
@(
   $value << $deg + ($min*(1/60)) + ($sec*(1/3600));
   ug_units_setMeasure($value, "Angle" );
) number;

#+
-------------------------------------------------------------------------------
Returns:
angle - #.Returns the decimal angle.#

See Also:
-------------------------------------------------------------------------------
#-
```

Contents of:
angle_conversions.dfa

SIEMENS

The #. .# syntax identifies the Description here.

This description will be visible in the Insert Function dialog…

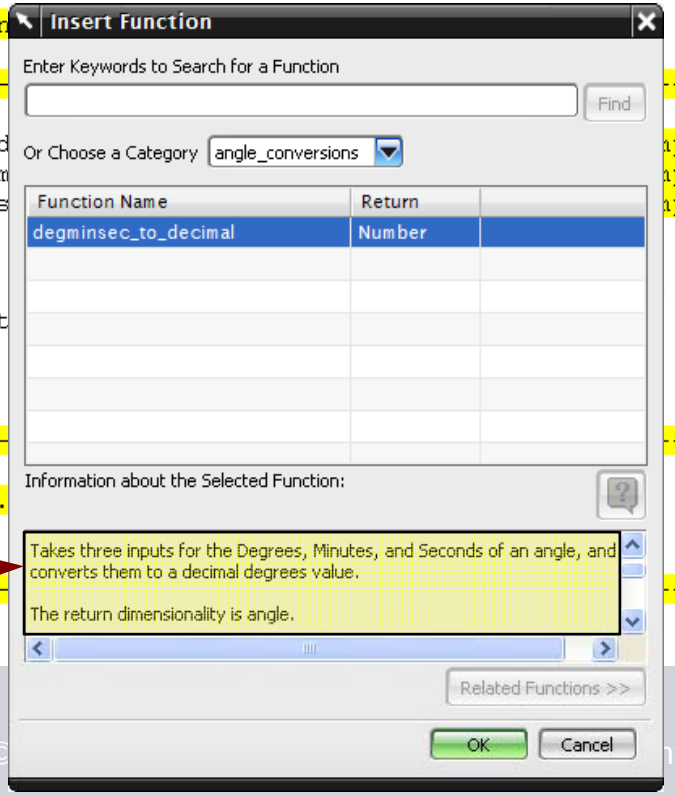
```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return
.#
----------
#-
number   $d
number   $m
number   $s
)
@(
    $value
    ug_unit
) number;

#+
----------
Returns:
angle - #.

See Als
----------
#-
```



Contents of:
angle_conversions.dfa

SIEMENS

The #. .# syntax identifies the Description here.

..and also in the Function Arguments dialog.

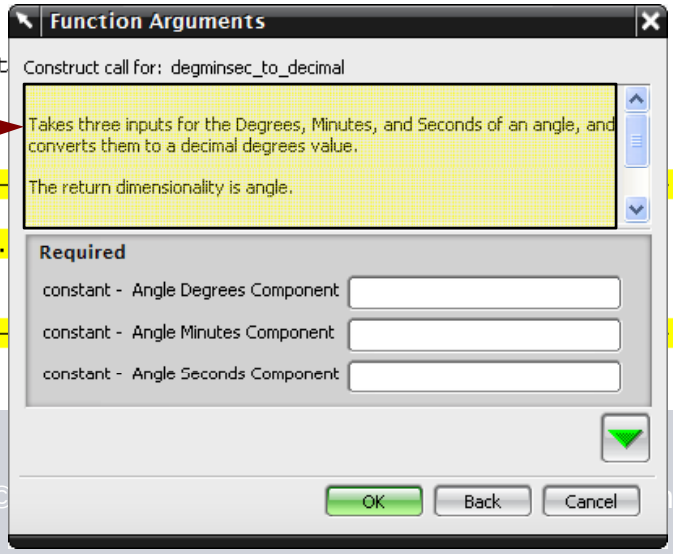
```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number   $deg;     # constant -  #.Angle Degrees Component.#
number   $min;     # constant -  #.Angle Minutes Component.#
number   $sec      # constant -  #.Angle Seconds Component.#
)
@{
    $value
    ug_unit
} number;

#+
----------                                  ----------
Returns:
angle - #.

See Also:
----------                                  ----------
#-
```



Contents of:
angle_conversions.dfa

SIEMENS

These three lines are identifying the inputs to this function.

Again, the #. .# syntax identifies text that will be displayed at the bottom of the Insert Function dialog.

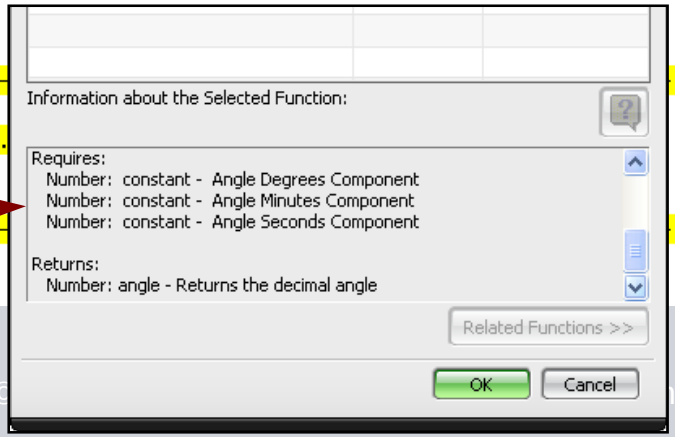
```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number   $deg;     # constant -   #.Angle Degrees Component.#
number   $min;     # constant -   #.Angle Minutes Component.#
number   $sec      # constant -   #.Angle Seconds Component.#
)
@{
   $value << $deg + ($min*(1/60)) + ($sec*(1/3600));
   ug_units_setMeasure($value, "Angle" );
} number;

#+
-------------------------------------------------------------------------------
Returns:
Angle - #.

See Also:
-------------------------------------------------------------------------------
#-
```



Contents of:
angle_conversions.dfa

SIEMENS

These three lines are identifying the inputs to this function.

…and in the detailed Function Arguments dialog.

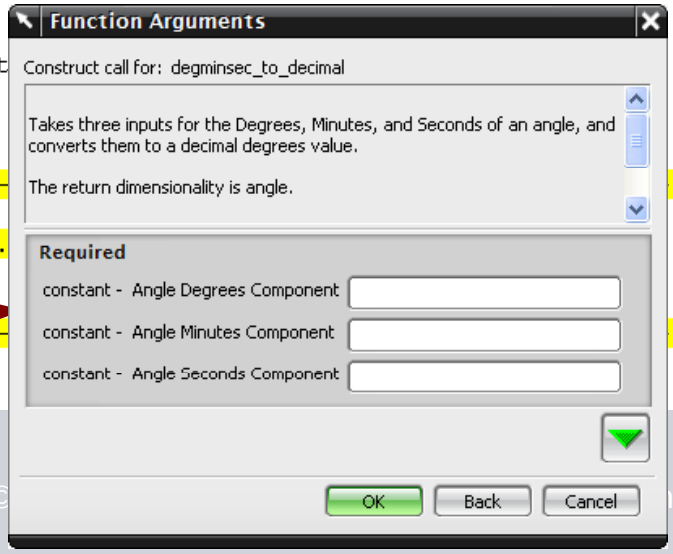
```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number   $deg;     # constant -  #.Angle Degrees Component.#
number   $min;     # constant -  #.Angle Minutes Component.#
number   $sec      # constant -  #.Angle Seconds Component.#
)
@(
    $value
    ug_unit
) number;

#+
----------                                    ----------
Returns:
Angle - #.

See Also:
----------                                    ----------
#-
```



Contents of:
angle_conversions.dfa

SIEMENS

The code inside the procedural block @{ } here is where the rubber hits the road.

The three inputs are combined to create an output value…

```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number   $deg;     # constant -   #.Angle Degrees Component.#
number   $min;     # constant -   #.Angle Minutes Component.#
number   $sec      # constant -   #.Angle Seconds Component.#
)
@{
    $value << $deg + ($min*(1/60)) + ($sec*(1/3600));
    ug_units_setMeasure($value, "Angle" );
} number;

#+
-------------------------------------------------------------------------------
Returns:
angle - #.Returns the decimal angle.#

See Also:
-------------------------------------------------------------------------------
#-
```

Contents of:
angle_conversions.dfa

SIEMENS

We set the return units of the value to be an Angle…

```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number  $deg;     # constant -  #.Angle Degrees Component.#
number  $min;     # constant -  #.Angle Minutes Component.#
number  $sec      # constant -  #.Angle Seconds Component.#
)
@(
   $value << $deg + ($min*(1/60)) + ($sec*(1/3600));
   ug_units_setMeasure($value, "Angle" );
) number;

#+
-------------------------------------------------------------------------------
Returns:
angle - #.Returns the decimal angle.#

See Also:
-------------------------------------------------------------------------------
#-
```

Contents of:
angle_conversions.dfa

SIEMENS

```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-------------------------------------------------------------------------------
#-
number   $deg;     # constant -   #.Angle Degrees Component.#
number   $min;     # constant -   #.Angle Minutes Component.#
number   $sec      # constant -   #.Angle Seconds Component.#
)
@(
    $value << $deg + ($min*(1/60)) + ($sec*(1/3600));
    ug_units_setMeasure($value, "Angle" );
) number;

#+
-------------------------------------------------------------------------------
Returns:
angle - #.Returns the decimal angle.#

See Also:
-------------------------------------------------------------------------------
#-
```

…and tell Knowledge Fusion to send back a floating point number.

(…as opposed to a string or a vector, for instance.)

Contents of:
angle_conversions.dfa

SIEMENS

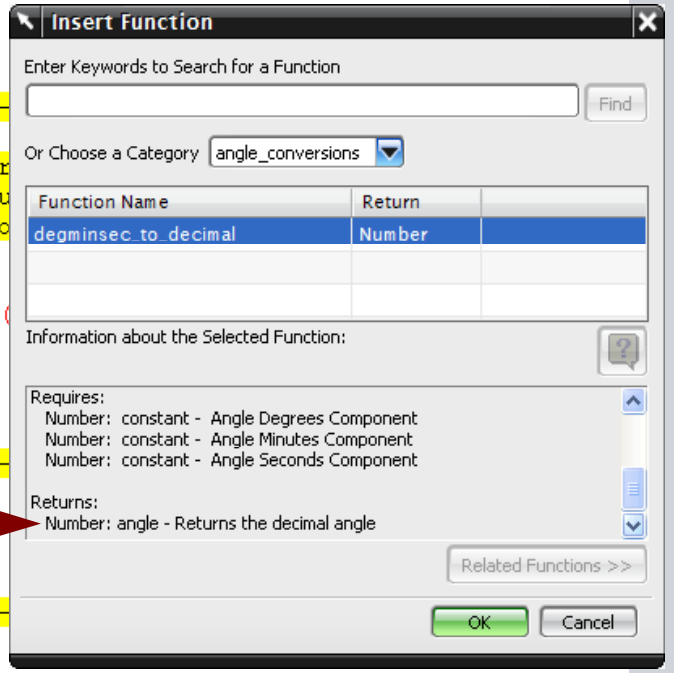
```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-----------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dimensionality is angle.
.#
-----------------------------------------------
#-
number   $deg;     # constant -  #.Angle Degr
number   $min;     # constant -  #.Angle Minu
number   $sec      # constant -  #.Angle Seco
)
@(
   $value << $deg + ($min*(1/60)) + ($sec*(
   ug_units_setMeasure($value, "Angle" );
) number;

#+
-----------------------------------------------
Returns:
angle - #.Returns the decimal angle.#

See Also:
-----------------------------------------------
#-
```



This comment is also automatically displayed in the Insert Function dialog, describing the return value.

Contents of:
angle_conversions.dfa

SIEMENS

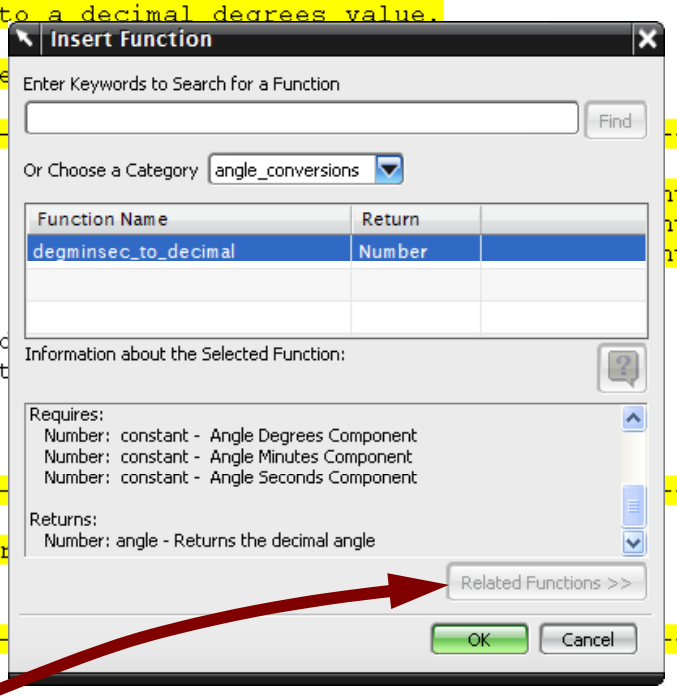
```
#! NX/KF 3.0

Defun: degminsec_to_decimal(
#+
DesignLogic=yes
-------------------------------------------------------------------------------
Description:
#.
Takes three inputs for the Degrees, Minutes, and Seconds of an angle, and
converts them to a decimal degrees value.

The return dime                                                    nt.#
.#                                                                 nt.#
-------------------------------------------------------------------nt.#
#-
number   $deg;
number   $min;
number   $sec
)
@(
    $value << $c
    ug_units_set
) number;

#+
-------------------------------------------------------------------------------
Returns:
angle - #.Retur

See Also:
-------------------------------------------------------------------------------
#-
```



…and functions listed under the "See Also:" heading will populate the "Related Functions" button.

Another Demonstration

SIEMENS

Thank you!

www.siemens.com/plm