PDHonline Course G320 (4 PDH)

Designing with Parameters in Autodesk Inventor

Instructor: Chad A. Thompson, P.E.

2020

PDH Online | PDH Center

5272 Meadow Estates Drive
Fairfax, VA 22030-6658
Phone: 703-988-0088
www.PDHonline.com

An Approved Continuing Education Provider

Designing with Parameters in Autodesk Inventor

Chad A. Thompson, P.E.

Table of Contents

Introduction

Autodesk Inventor is a parametric CAD application, meaning it makes use of editable parameters to build 3D models. As you create a model, parameters are created behind the scenes which store necessary values for building the model. These parameters can be changed after initial modeling to modify the model, they can be used in equations to create relationships between parameters, or they can be shared between files.

This course will discuss many topics involving Inventor parameters, including using parameters in equations, giving parameters meaningful names, creating user parameters, and sharing parameters with other Inventor files as well as with spreadsheets.

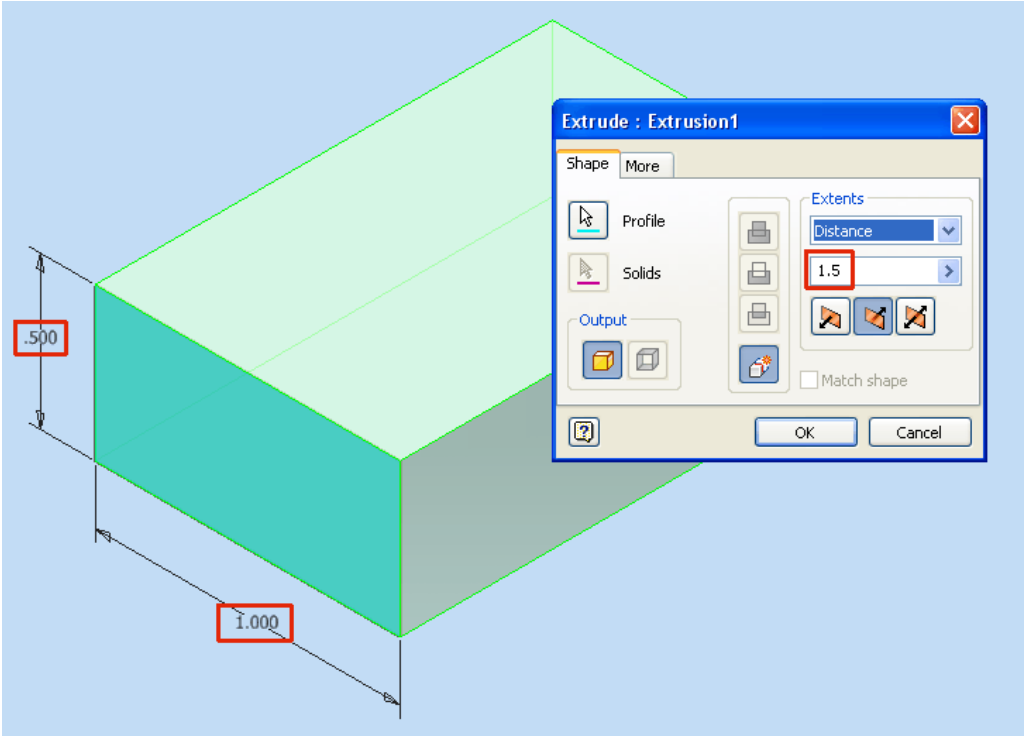
A basic level of proficiency with Inventor is presumed for students taking this course, especially an understanding of how to create parts and assemblies. Access to the software while taking the course, while not required, is highly recommended in order to follow along with the concepts discussed. Any version of Inventor is fine for this purpose.

Inventor 2010 is used in this course to illustrate examples, but the concepts discussed here are applicable to any previous version of Inventor, and presumably future versions as well. For the purposes of this course, differences between Inventor 2010 and previous versions are minor, and will be noted where appropriate.

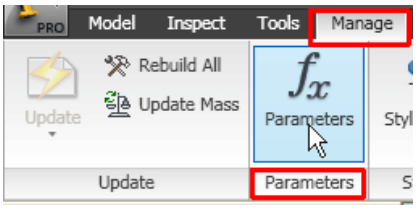
At the end of this document are several case studies where many of the principles discussed here are applied in simulated real-world scenarios.

Basic Parameter Concepts

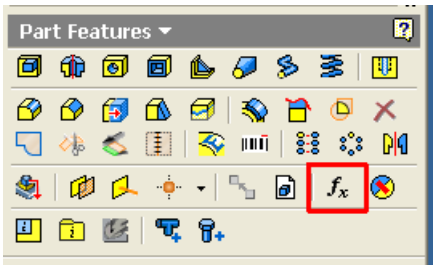
In the part below, a rectangular block has been created by extruding a sketched rectangle.



As this feature is created, each of the defining dimensions is set up automatically by Inventor as a parameter. We can see these parameter definitions by opening the *Parameters* dialog box. To access this dialog box, go to the *Manage* tab, and in the *Parameters* panel, click the *Parameters* tool button.

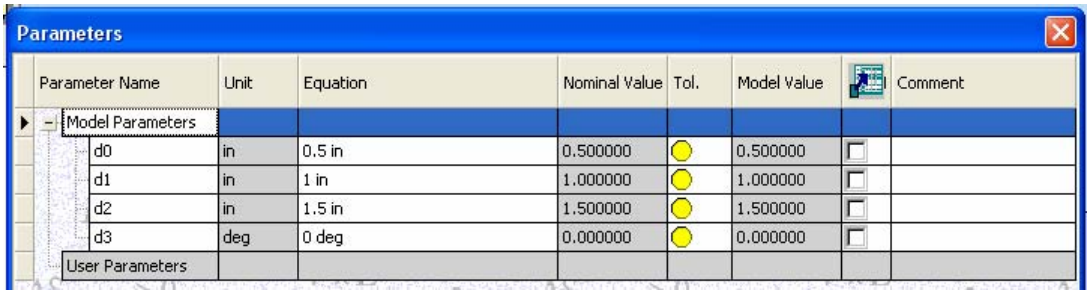


In versions of Inventor prior to 2010, the *Parameters* dialog box is accessed by clicking the *Parameters* tool button in the panel bar.



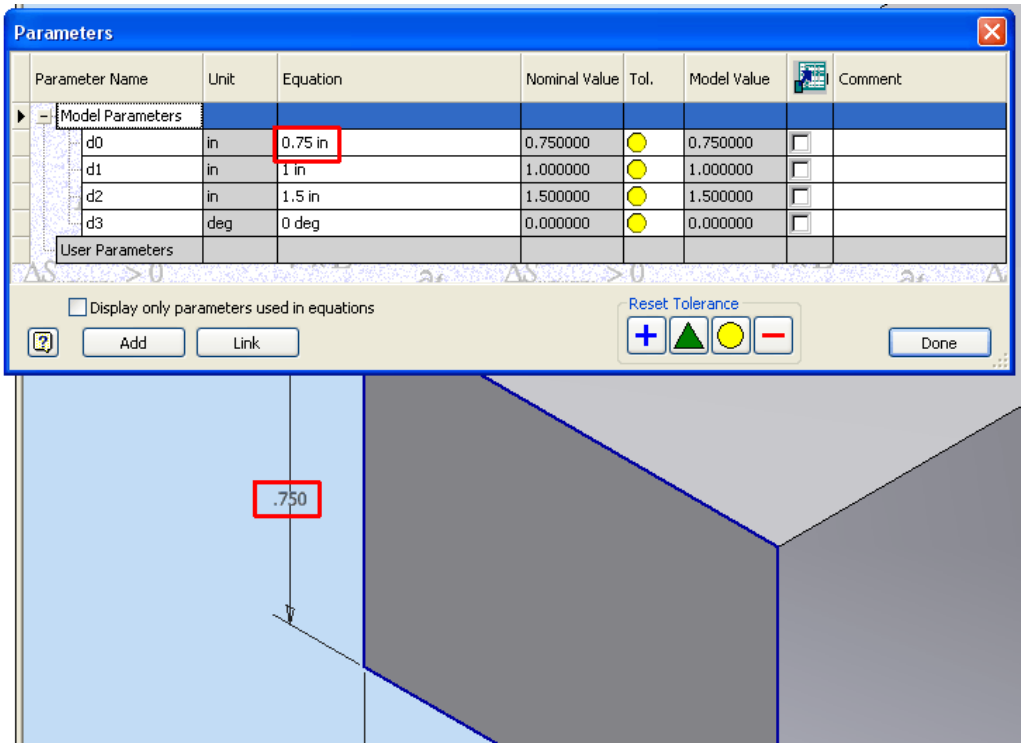
Note that, in a part file, you must be out of sketch mode to access the *Parameters* tool.

Looking at the parameters in the *Parameters* dialog box, we can see the model parameters that Inventor has automatically created for the rectangular block above.

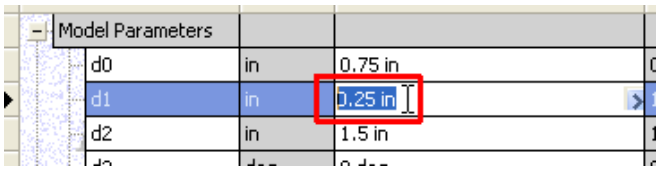


Each of the parameters has been assigned an arbitrary name of the format "d*n*". We can see each of the three values we used to create the rectangular block assigned to a parameter name. (The *d3* parameter was set up for the taper angle of the *Extrude* feature that was created.)

If we return to the model and change a dimensional value, we see this change reflected in the *Parameters* dialog box.

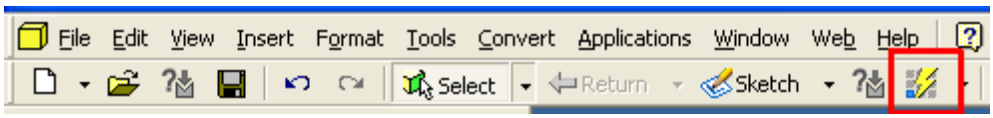


We can also change a parameter right in the *Parameters* dialog box, then update the model to reflect the change. Simply click in the *Equations* field for the appropriate parameter and type in the new value.



When a parameter is modified in this manner, a manual update must be performed on the model to reflect the change. To do this, click the *Update* tool in the *Quick Access* toolbar at the top of the Inventor window.

In versions of Inventor prior to 2010, the *Update* tool is found in the *Inventor Standard* toolbar.

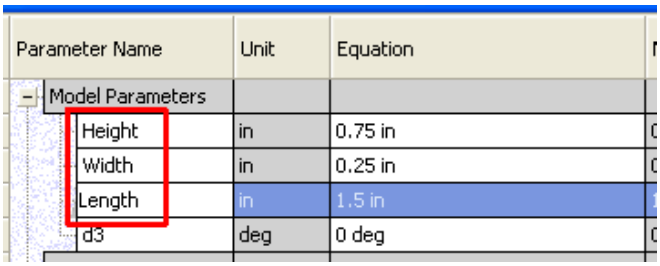


Renaming Parameters

The arbitrary "d*n*" convention used by Inventor when assigning parameter names can become confusing. The parameter list can grow quickly as additional features are added to an Inventor part, and keeping track of critical parameters can become unnecessarily burdensome if this naming convention is maintained.

Inventor gives us the ability to assign more meaningful names to our parameters, thus allowing us to reference the parameters more readily later on. This can be very useful when changing parameter values in the *Parameters* dialog box, or when using parameter names as variables in equations.

Continuing with our rectangular box from the previous section, let's open the *Parameters* dialog box to give the critical parameters in our model more meaningful names. To rename a parameter, simply click in the field in the *Parameter Names* column and type the new name. We can rename all three parameters which define the critical dimensions of our rectangular block.



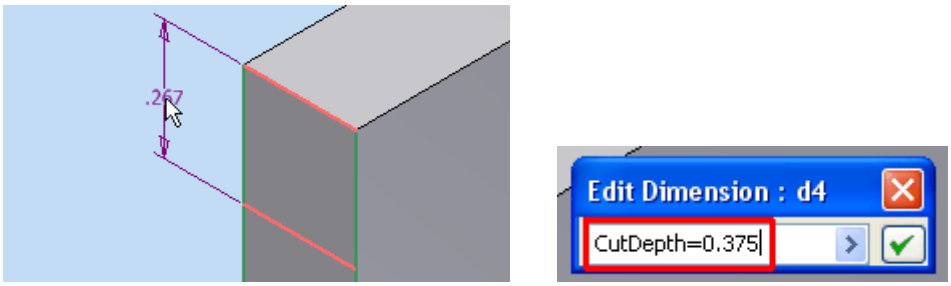
There are some limitations on what we can name our parameters. Parameter names cannot have special characters in them, including spaces. There are also a few other expressions that are reserved by Inventor for other purposes which cannot be used as parameter names. Inventor will give you an error message if you attempt to give a parameter an invalid name.

Parameter names are also case-sensitive. For example, using the expression "height" or "HEIGHT" in an equation would not appropriately reference the first parameter on our list above.

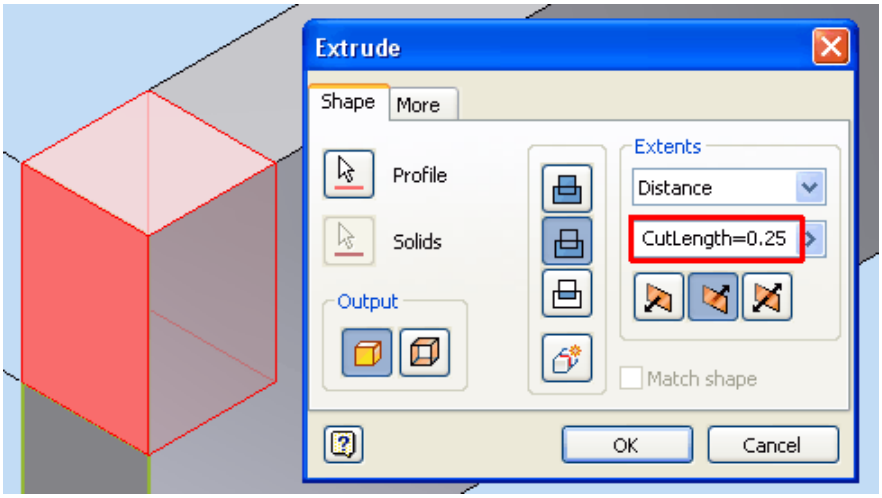
Prior to Inventor 2010, the method above was the only way to rename a parameter. You would create the parameter in the model, Inventor would assign an arbitrary name to the parameter, and then you would rename the parameter through the *Parameters* dialog box.

There is a quicker method in Inventor 2010 which allows you to give the parameter a meaningful name as you are creating it. When you are entering a dimension value, simply use the format *<parameter name>=<value>*.

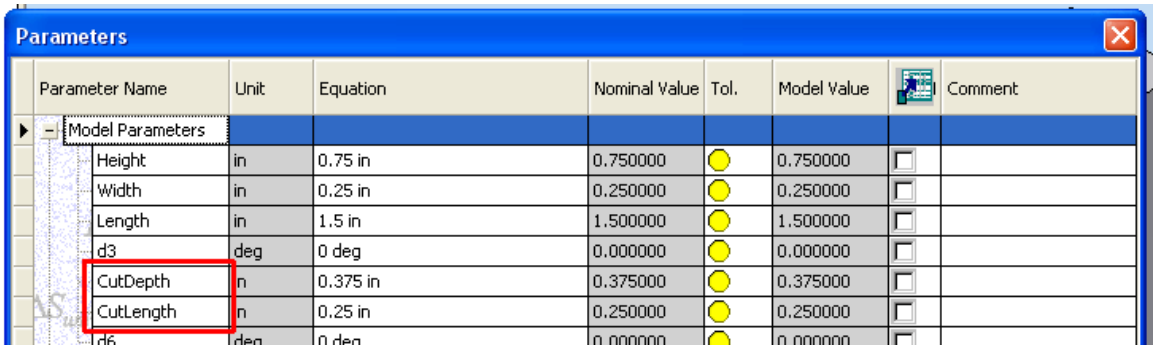
For example, in our rectangular block example, if we create another sketch and add a dimension to that sketch, we can type the new dimension value as shown.



This functionality works in feature creation dialog boxes as well. For instance, we can create a cut extrude feature on our part using the above sketch, and use the same technique to rename the parameter defining our extrude depth, as shown below.

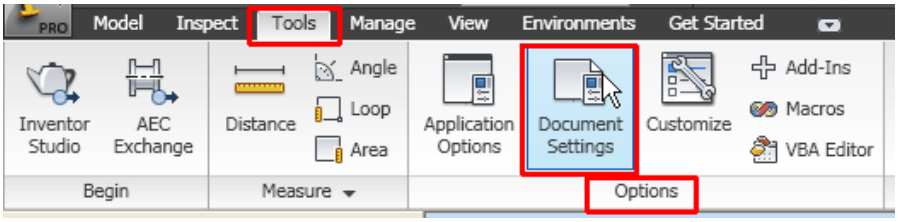


If we open the *Parameters* dialog box, we see the new parameters with the designated names we assigned to them as we created the new feature.



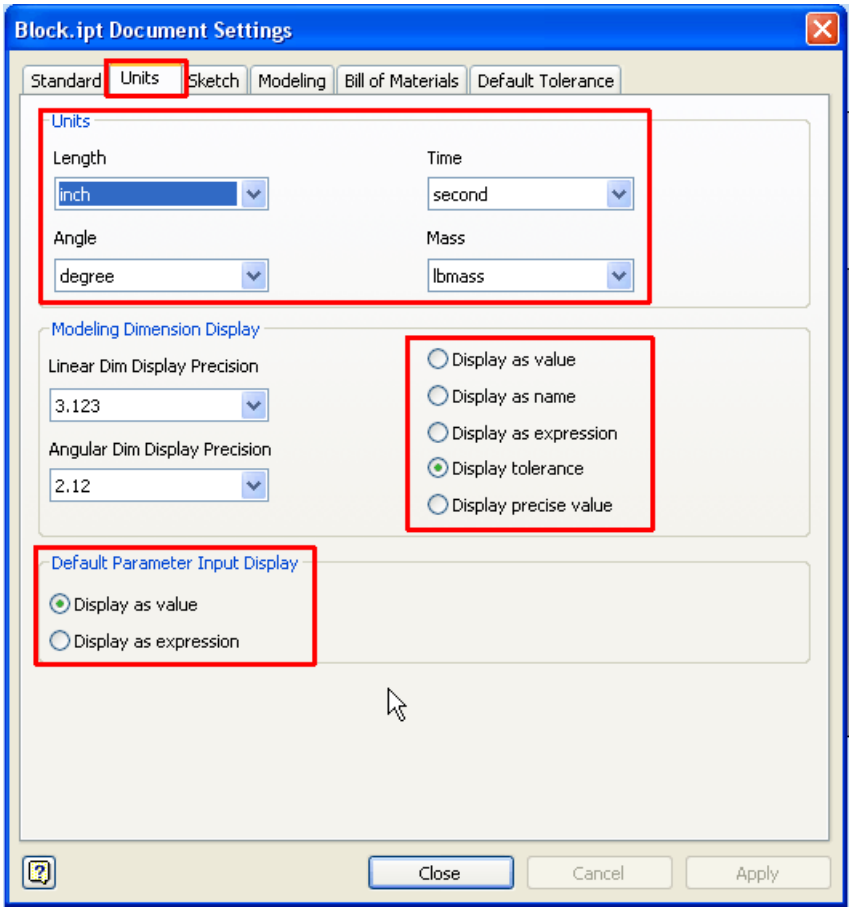
Using Document Settings with Parameters

There are a few settings in Inventor you should be aware of when working with parameters. To access these settings, on the *Tools* tab, in the *Options* panel, click *Document Settings*.



In versions of Inventor prior to 2010, access *Document Settings* from the *Tools* menu.

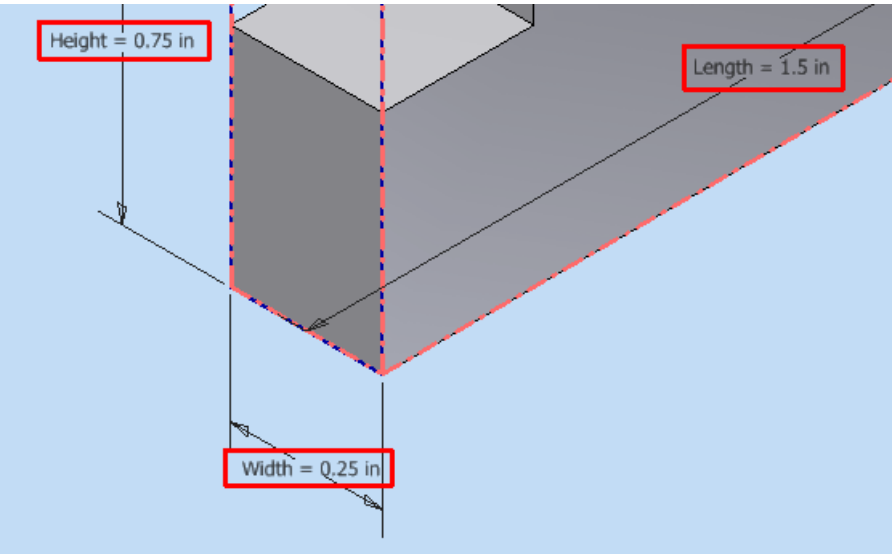
In the *Document Settings* dialog box, under the *Units* tab, are the default unit settings, the dimension display setting, and the input display setting (not found in versions prior to 2010).



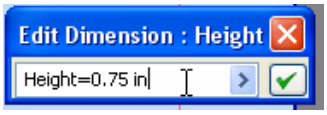
The *Units* settings determine the default units for parameters. These are typically set up in your Inventor template files and are brought into a new file when the desired template

for the new file is selected. However, these settings can be changed at any point after the file is created as well. We will discuss units further later in the course.

The *Dimension Display* setting can be set to show model dimensions as values, parameter names, or both. For instance, setting the dimension display for our rectangular block file to *Display as expression* results in the model dimensions being displayed as shown.



Setting the *Input Display* to *Display as expression* results in an expression being displayed in the *Edit Dimension* dialog box as shown. Again, this option is not available in versions of Inventor prior to 2010.



When overwriting an existing expression for a dimension value while in this display mode, it is not necessary to type the parameter name unless you are changing it.

Using Parameters in Equations

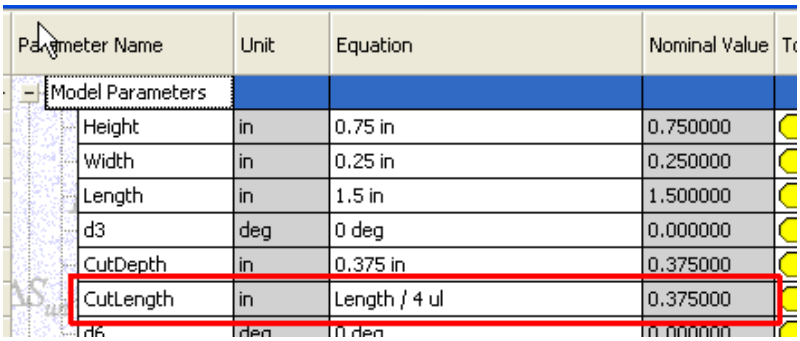
A huge advantage in parametric modeling is the ability to use equations to drive the values of our parameters. These equations can make use of mathematical operators and functions, and can also use other parameters as variables.

Mathematical functions that Inventor recognizes are listed below.

cos(expr)	acosh(expr)	abs(expr)
sin(expr)	asinh(expr)	max(expr1;expr2)
tan(expr)	atanh(expr)	min(expr1;expr2)
acos(expr)	sqrt(expr)	ln(expr)
asin(expr)	sign(expr)	log(expr)
atan(expr)	exp(expr)	pow(expr1; expr2)
cosh(expr)	floor(expr)	random()
sinh(expr)	ceil(expr)	isolate(expr;unit;unit)
tanh(expr)	round(expr)	

For more details on these functions, consult the Help menu in the Inventor application.

As a simple example of using an equation, let's make the length of the cut in our rectangular block equal to one-fourth of the total length of the block. We can do this by entering an equation for the *CutLength* parameter. One way to do this is to open the *Parameters* dialog box and edit the *Equation* field for *CutLength* as shown.

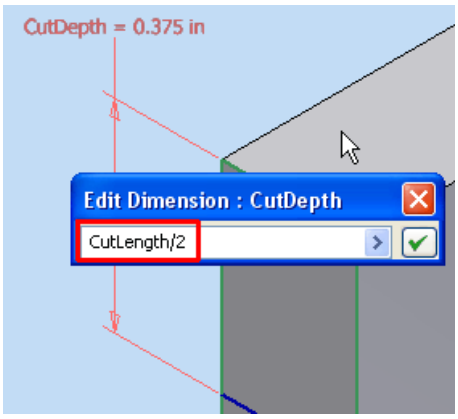


Parameter Name	Unit	Equation	Nominal Value
Model Parameters			
Height	in	0.75 in	0.750000
Width	in	0.25 in	0.250000
Length	in	1.5 in	1.500000
d3	deg	0 deg	0.000000
CutDepth	in	0.375 in	0.375000
CutLength	in	Length / 4 ul	0.375000

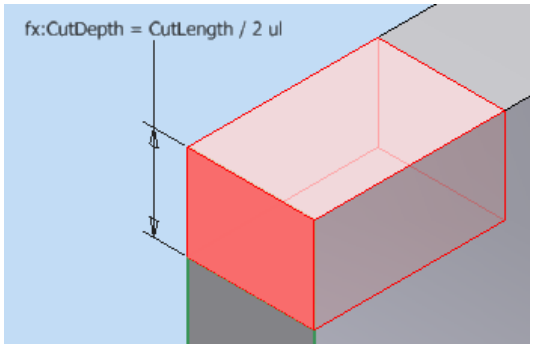
Note the results of the equation shown in the N*ominal Value* column. (The '*ul*' at the end of the equation stands for *unitless*, and indicates that the '*4*' term in the equation has no units associated with it. The '*ul*' doesn't have to be typed in. Inventor fills this into equations automatically where it is needed. We will discuss units further a little later.)

We might set up this scenario if, for instance, we're pretty sure we want the length of the cut to be one-quarter of the total block length, but we're not sure what the final block length will be. If we want to experiment with different block lengths as our design progresses, we only have to change the *Length* parameter and the *CutLength* parameter changes automatically to accommodate it.

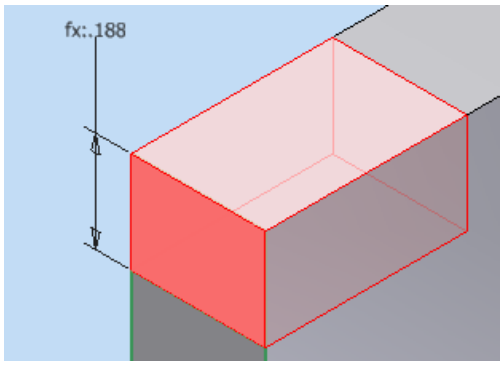
An equation can also be assigned to a parameter by entering the equation as the value for a sketch dimension or as the distance in, say, an extrude feature. For instance, if I want the parameter *CutDepth* to always be one-half of *CutLength*, I can edit the sketch for the extrude feature and enter the appropriate equation for the dimension value, as shown.



Below, note again the automatic addition of the '*ul*' term. Also note the '*fx:*' symbol in front of the parameter name, indicating that the value of the parameter is calculated from an equation.



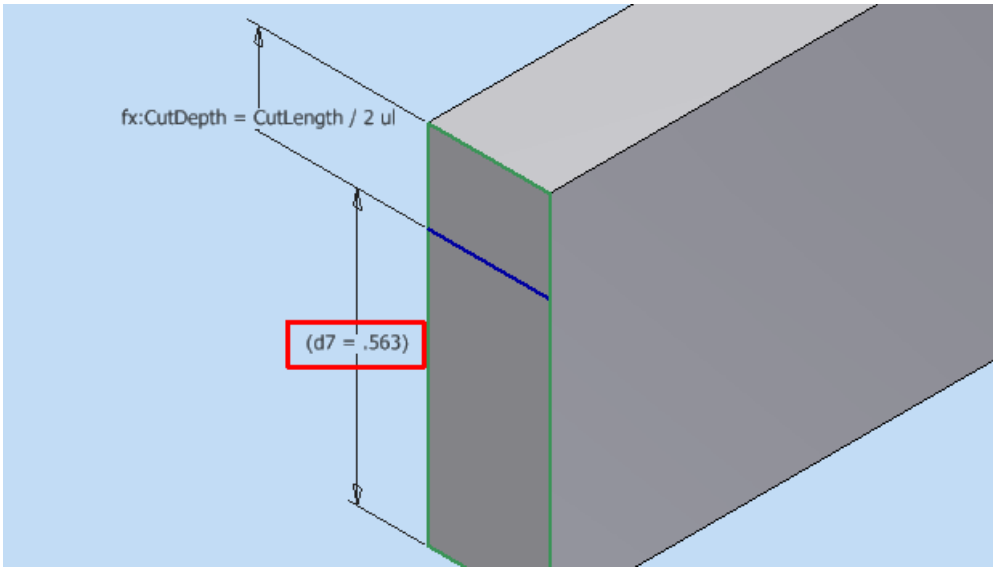
The usefulness of the '*fx:*' symbol becomes more obvious if we change our dimension display setting to *Display as value*. We can see below that if the symbol were not there, it would not be obvious that the value of the dimension was the result of an equation.



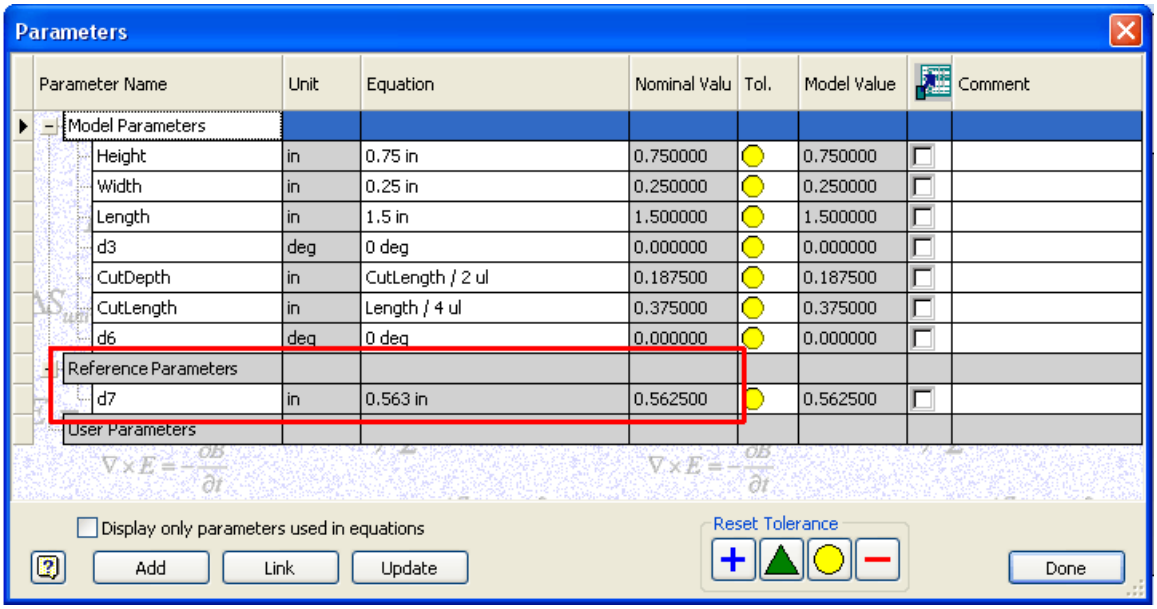
We will look at more examples of parameter equations later in the case studies.

Reference Parameters

Reference parameters are created when driven dimensions are placed on a sketch. For instance, if we open the sketch for the extruded cut in our rectangular block, we can place a driven dimension (indicated with parentheses) as shown.



If we open the *Parameters* dialog box, we can see the driven dimension represented as a reference parameter.

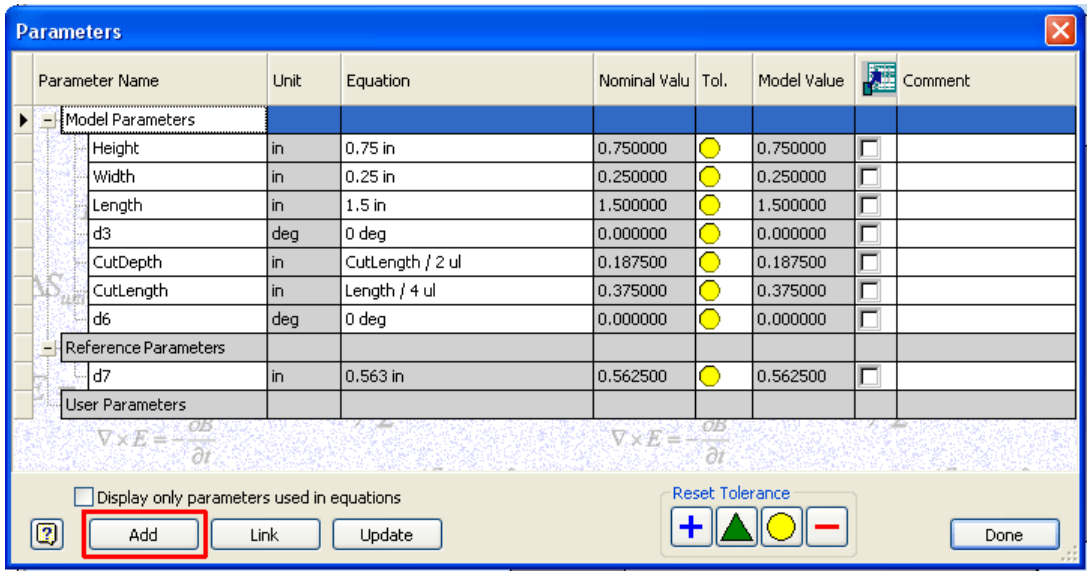


This parameter can be renamed or referenced in an equation just like the model parameters we have looked at. Note, however, that the equation field is in gray, indicating that it is not directly editable.

User Parameters

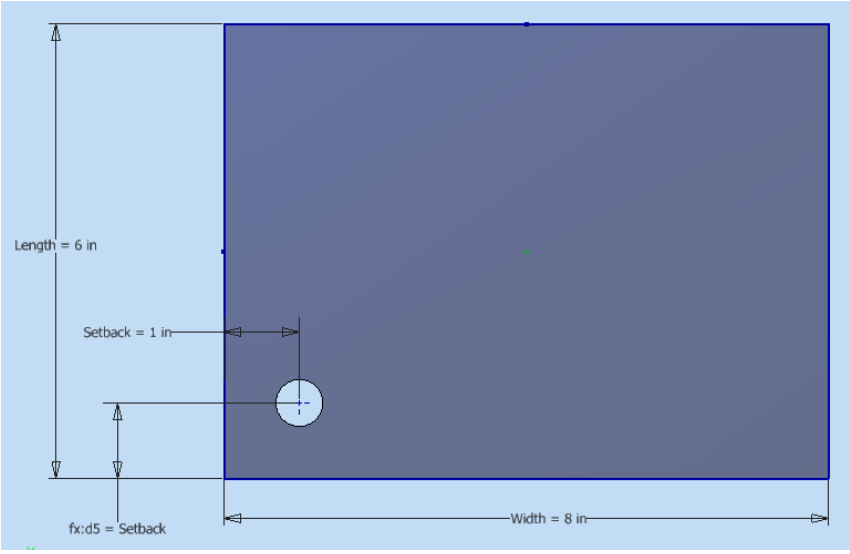
User parameters are not created by placing dimensions in a sketch or creating features. Instead, they are created directly in the *Parameters* dialog box. They can be used in equations or referenced in sketches and features just like other parameters we have looked at.

To create a user parameter, open the *Parameters* dialog box and click the *Add* button.

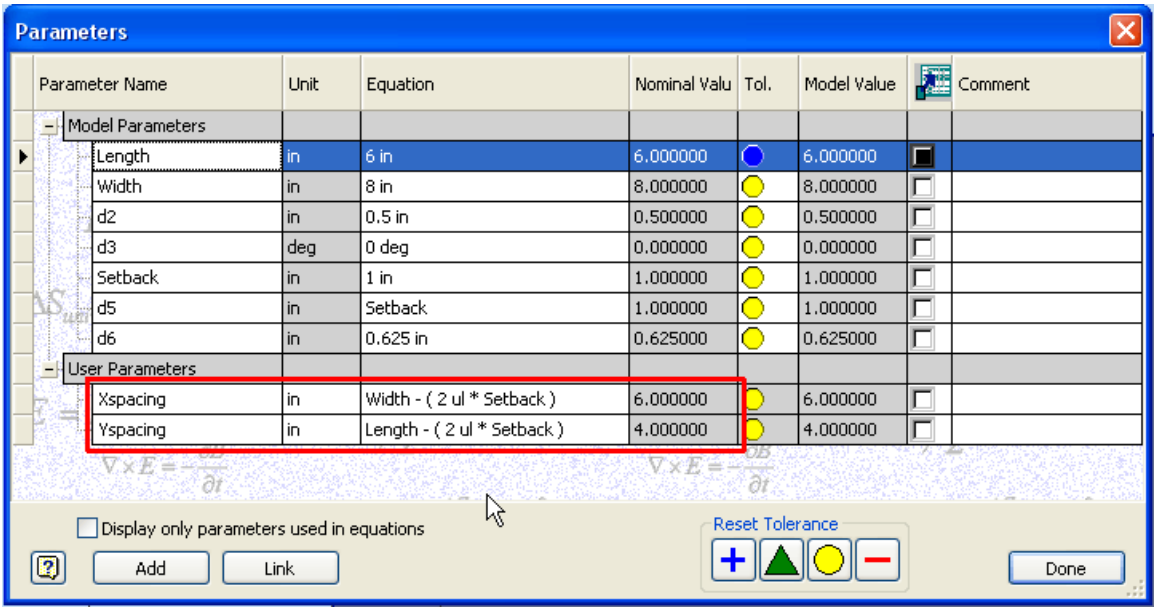


This will create a new line in the *User Parameters* area, where the name and the equation for the user parameter can be entered.

To show user parameters in action, see below a rectangular plate with a single mounting hole in one corner. We would like to use the parameters we have set up to create a rectangular pattern so that each corner has an identically placed hole.

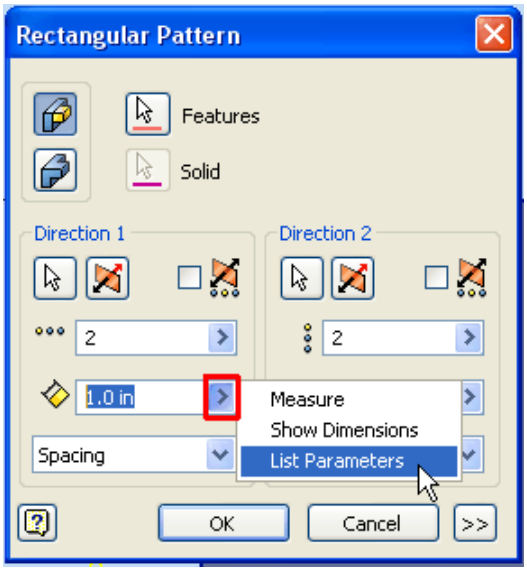


We will begin by opening the *Parameters* dialog box and creating two user parameters as shown.



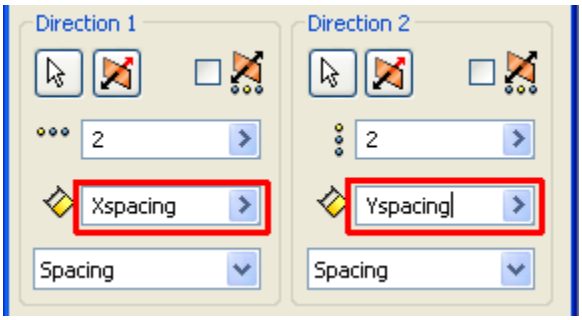
These user parameters will be used to set the spacing for our rectangular pattern. As the *Length* and *Width* parameters vary, the pattern spacing will change accordingly so that the holes always remain in the same place relative to the edges.

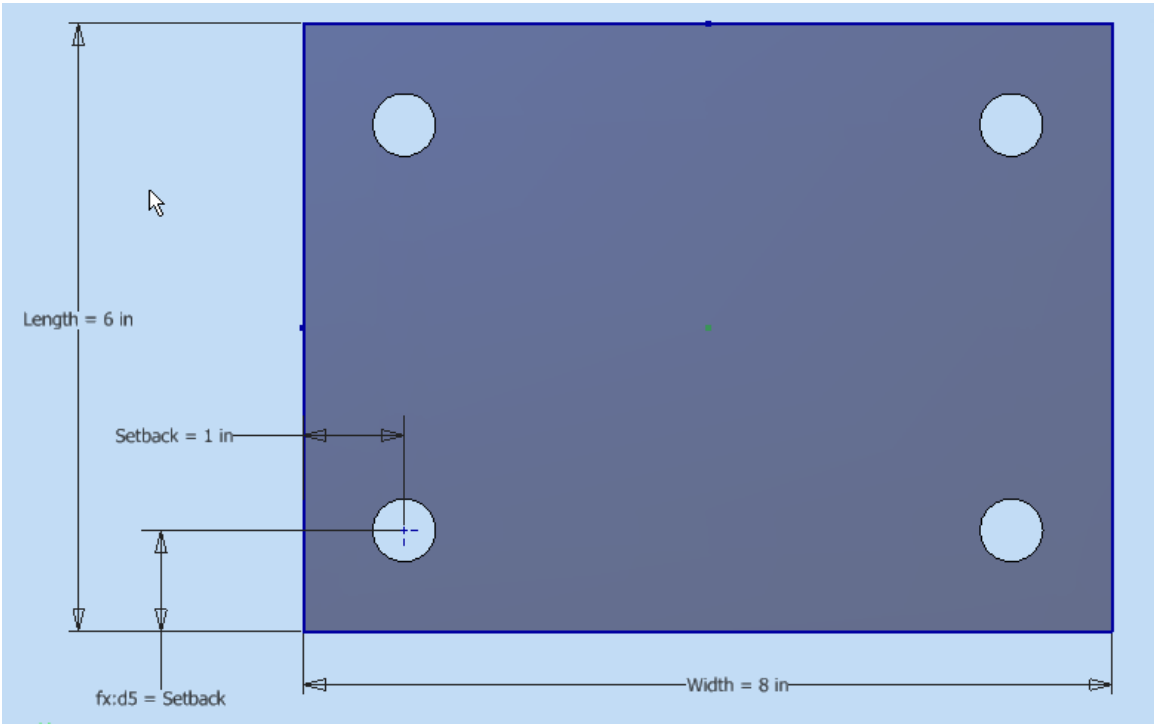
In the rectangular pattern dialog box, we simply need to fill in our user parameter names in the pattern spacing fields. We could type in the names manually, but to save some keystrokes, we can use another technique. Highlight the value in the field, then click the flyout arrow and select *List Parameters* as shown.



This will display a list of all the parameters that have user-created names. We simply need to select the appropriate parameter name from the list and it will be filled in automatically.

Repeat for the spacing in the other direction, and we have linked our pattern spacing to the user parameters.





At this point, perhaps we might want to save this part as a template or an iPart. For each new plate size, we need only change the Length and Width, and the hole pattern updates automatically.

Parameter Units

Each Inventor parameter has either a type of unit, such as inches or degrees or pounds, assigned to it, or it is unitless, such as a parameter specifying the number of occurrences in a feature pattern. Also, each term in a parameter equation must have the units specified.

The units for model parameters are determined by the default units settings, which we looked at previously in the *Document Settings*. If no units are specified for a parameter value, the default units are assumed. For instance, if a file has inches set up as the default length units, and we enter a value of '*14*' as a parameter value, Inventor will assume we mean 14 inches, and '*14 in*' will be displayed in the equation field.

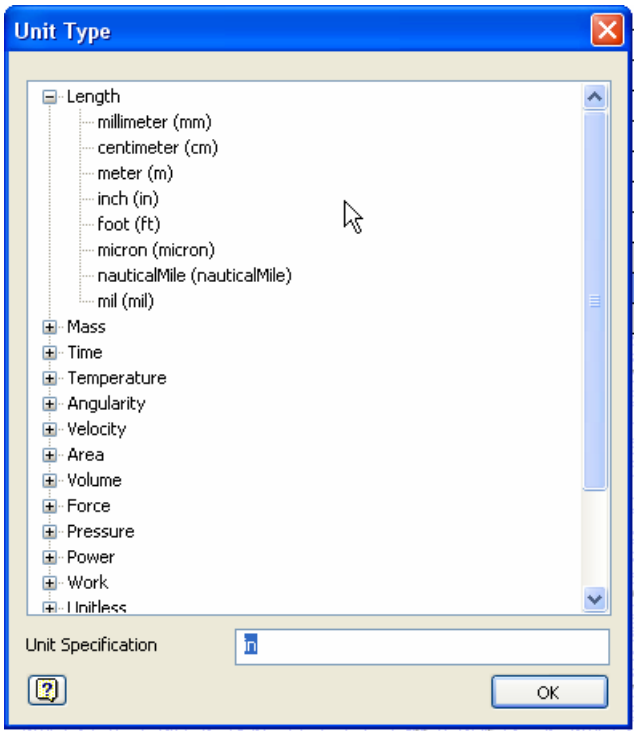
We can, however, still explicitly indicate alternate units for the value of a parameter. For instance, even if a parameter is set up with inches as its units, we can enter '*150 mm*', and Inventor will understand that we mean 150 millimeters. Of course, the alternate units must still be the same type of unit. For instance, in the previous example, inches and millimeters are both units of length.

Looking again at the *Parameters* dialog box for our rectangular plate, we can see that the Units fields for the Model Parameters are gray, while those for the User Parameters are white.

Parameter Name	Unit	Equation	Nominal V	Tol.	Model Va
Model Parameters					
Length	in	6 in	6.000000		6.000000
Width	in	8 in	8.000000		8.000000
d2	in	0.5 in	0.500000		0.500000
d3	deg	0 deg	0.000000		0.000000
Setback	in	1 in	1.000000		1.000000
d5	in	Setback	1.000000		1.000000
d6	in	0.625 in	0.625000		0.625000
d13	ul	2 ul	2.000000		2.000000
d15	in	Xspacing	6.000000		6.000000
d16	ul	2 ul	2.000000		2.000000
d18	in	Yspacing	4.000000		4.000000
User Parameters					
Xspacing	in	Width - (2 ul * Setback)	6.000000		6.000000
Yspacing	in	Length - (2 ul * Setback)	4.000000		4.000000

The Model Parameters units cannot be edited from the *Parameters* dialog box. As we discussed, these units are set by the default unit settings in the *Document Settings* dialog box.

We can, however, edit the units for our user parameters. By simply clicking in the Unit field for a user parameter, we bring up the *Unit Type* dialog box, where we can select alternate units for the parameter.



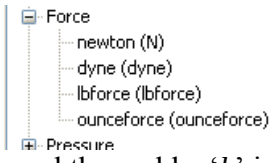
If we change the units for our two user parameters to millimeters, note below that the value for each of the parameters is now displayed in millimeters.

User Parameters			
Xspacing	mm	Width - (2 ul * Setback)	152.400000
Yspacing	mm	Length - (2 ul * Setback)	101.600000

Inventor also understands unit prefixes. Shown below are the prefixes which can be used when specifying parameter units.

"exa"	"E"	= 1.0e18	"deci"	"d"	= 1.0e-1
"peta"	"P"	= 1.0e15	"centi"	"c"	= 1.0e-2
"tera"	"T"	= 1.0e12	"milli"	"m"	= 1.0e-3
"giga"	"G"	= 1.0e9	"micro"	"micro"	= 1.0e-6
"mega"	"M"	= 1.0e6	"nano"	"n"	= 1.0e-9
"kilo"	"k"	= 1.0e3	"pico"	"p"	= 1.0e-12
"hecto"	"h"	= 1.0e2	"femto"	"f"	= 1.0e-15
"deca"	"da"	= 1.0e1	"atto"	"a"	= 1.0e-18

For instance, say we want a user parameter with units of kilo-pounds, or kips. Looking at our default choices for force in the *Unit Type* dialog box, this is not one of the options.



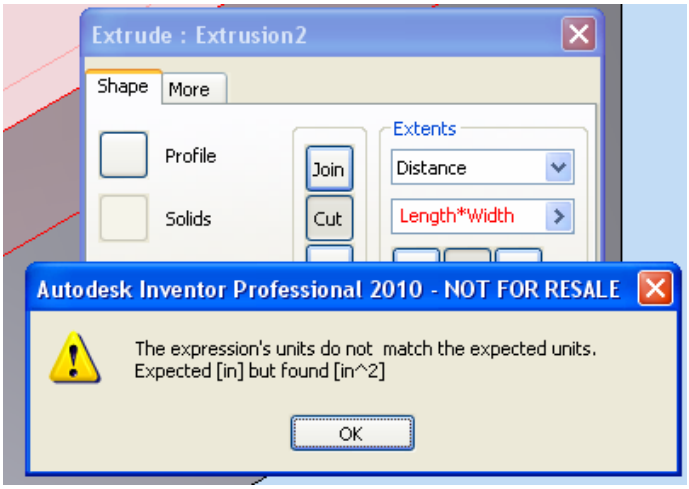
We can, however, select *lbforce*, and then add a '*k*' in front as shown to create our desired units.

We can also specify combinations of recognized units to create custom units, such as *rad/s* (radians per second) or *N m* (newton-meters). Many of the units recognized in Inventor are unrelated to the actual physical dimensions of the model. As we shall see in the case studies later, though, they can still be quite useful.

The syntax for a unit must be entered exactly as it is shown in the *Unit Type* dialog box, including the case. With a few exceptions, most units in Inventor are lower case. For instance, for units of inches, Inventor recognizes '*in*', but not '*IN*' or '*inches*'.

Because Inventor understands and uses units, it also looks for agreement between the units of a parameter and the resulting units of the equation for the parameter.

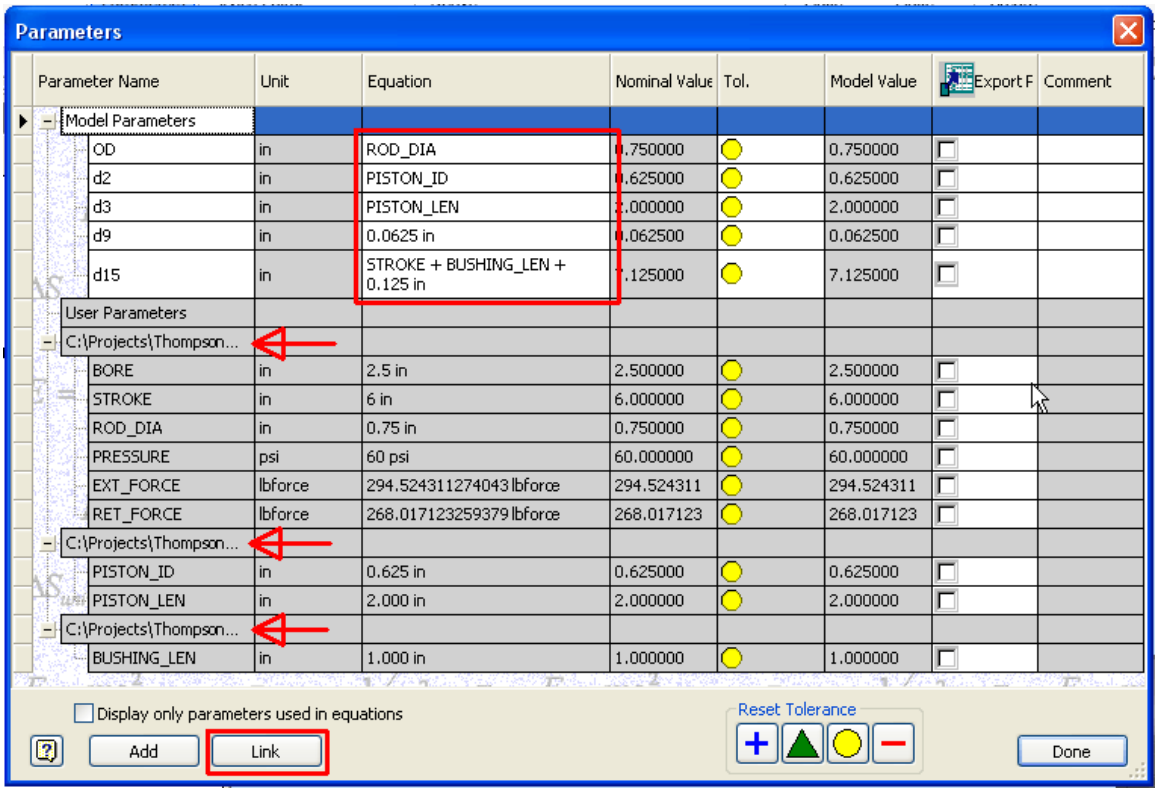
For instance, if I have a part with two parameters *Length* and *Width*, both with units of inches, and I attempt to specify the distance for an extrusion with the equation *Length*Width*, I get an error message as shown.



Note also that the equation entered in the field for extrusion distance is in red. As I am typing an equation in either a feature dialog box, a dimension edition box, or the Parameters dialog box, the text will show as red until the units of the equation are in agreement with the units of the parameter.

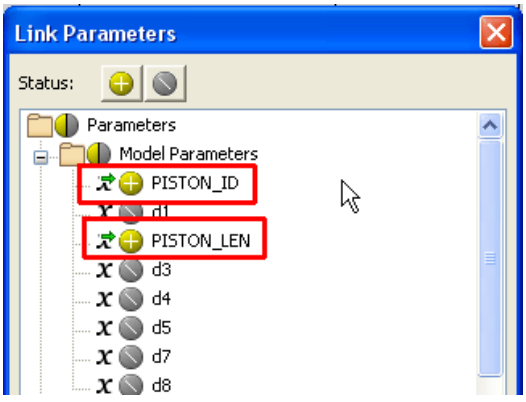
Linking Parameters from Other Inventor Files

An Inventor file can make use of parameters from other Inventor files. To set up this link, open the file which will make use of the linked parameters, open the *Parameters* dialog box, and click the *Link* button.



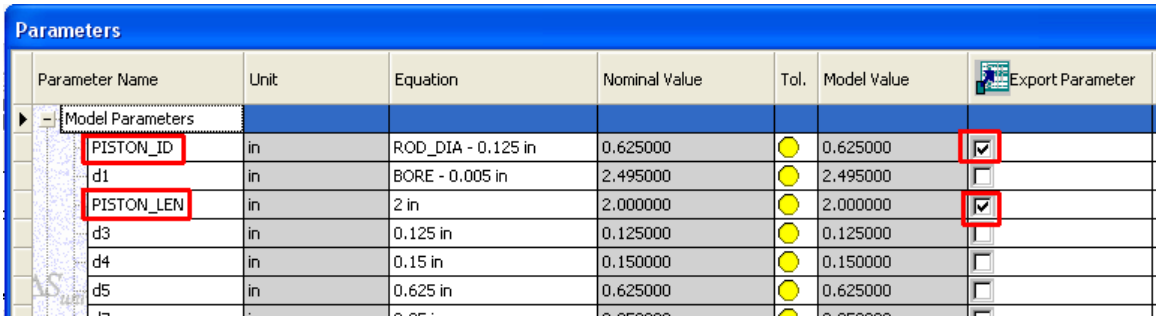
In the example above, a link has been established between the open file and three other files. Parameters from each of the files have then been used in equations for the model parameters of the open file.

When the second link above was established, the needed parameters were specified by selecting them from a list, as shown.



The icon before each parameter indicates whether the parameter is included or excluded from the link. Clicking the icon for a parameter changes the parameter's status.

Also, if we open up the linked file and look at its parameters, we see that in the *Export Parameter* column there is a check placed in the boxes for the two parameters selected.



This indicates that these parameters are available to be referenced by another file. These check boxes can be selected or unselected manually from within the *Parameters* dialog box.

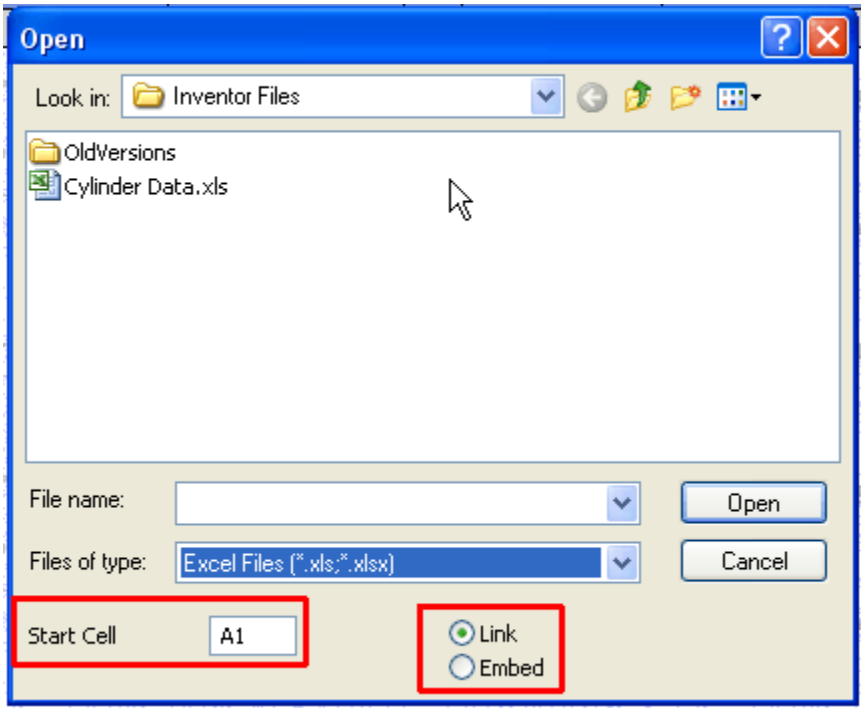
If a parameter is chosen to be included during the linking process, and the *Export Parameter* check box is not already selected for that parameter, the box will be selected automatically, and a message will be displayed indicating that this has been done.

Linking Parameters From Spreadsheets

In some cases it may be preferable to use a spreadsheet to maintain parameters for your Inventor files. This may be due to the ease with which data can be manipulated within a spreadsheet, the need for formulas which may be cumbersome or even impossible to do within Inventor, the need to pass data to multiple Inventor files from a single source, or the need for non-Inventor users to view or edit the design parameters.

The spreadsheet itself must be formatted properly so that Inventor can recognize the contents as parameters. The data can start on any cell of the spreadsheet, and it can be in rows or columns, but the rows or columns must be in a specific order, which is:
1) parameter name, 2) value or equation, 3) unit of measurement, 4) comment.

To link a spreadsheet, again click the *Link* button in the *Parameters* dialog box. With the file type set to *Excel Files*, browse to and select the desired Excel file, specify the cell in which the data starts, and choose whether you want to link or embed the file.



Choosing *Link* maintains a link to the external Excel file, and is typically used whenever there is a need to maintain a separate file, such as if multiple Inventor files will access the spreadsheet data, or if non-Inventor users need to access the data in the spreadsheet.

Choosing *Embed* copies and embeds the spreadsheet within the Inventor file, and no link is maintained to the original external Excel file. Any changes to the data in an embedded spreadsheet only affect the Inventor file in which the spreadsheet is embedded.

Parameters in Assemblies

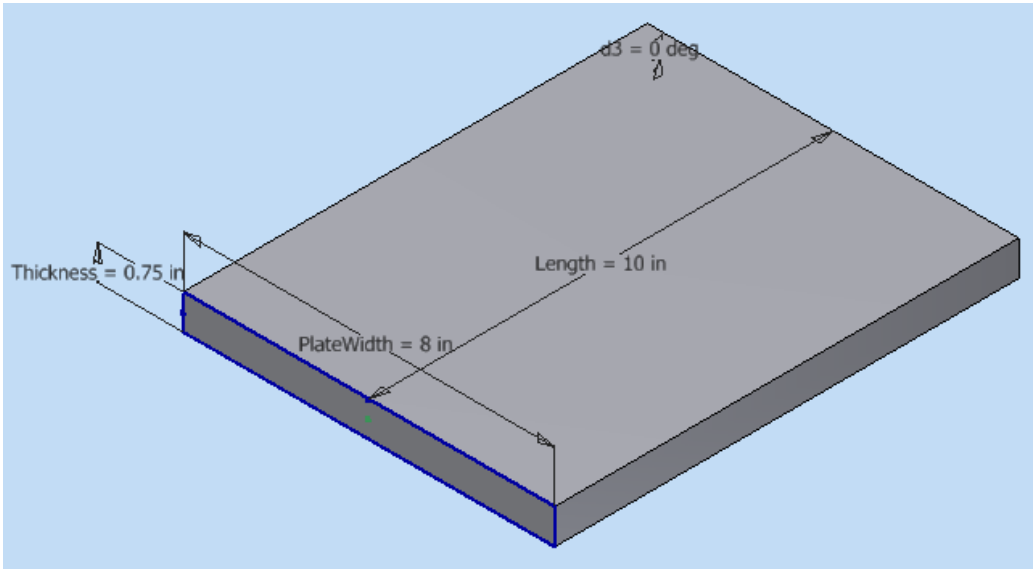
Thus far, we have only looked at parameters in part files, but Inventor assembly files make use of parameters as well. Things like offset values for assembly constraints and numbers of occurrences for component patterns are maintained as parameters in assembly files.

Parameters in assemblies are accessed in the same way as in part files, through the *Parameters* dialog box, and the same principles apply for manipulating the parameters. Assembly parameters have units, can be populated with equations, and can be linked from other files.

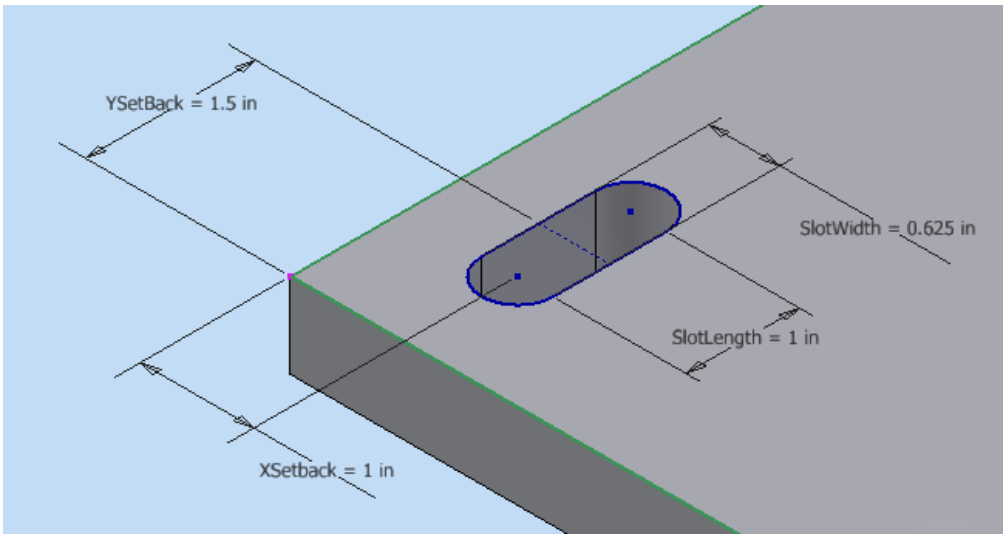
Case Study #1: Determining the Safety Factor of a Plate in Tension

While more advanced versions of Inventor have some sophisticated stress analysis tools, here we will make use of the parametric functionality in basic Inventor to determine the stress and safety factor for a part. We will also use parameters set up some guidelines for how many mounting holes the plate needs based on the width of the plate. Since this is a relatively simple model, it may be helpful to create your own Inventor part and follow along with the steps below.

Let's create a rectangular steel plate with its dimensions determined by parameters named *PlateWidth*, *Thickness*, and *Length* as shown.



Let's then create a sketch for a slotted hole in the plate using parameters named *SlotLength*, *SlotWidth*, *XSetback*, and *YSetback* as shown, then make a cut extrude through the plate using the sketch.



Now we want to create a pattern of slotted holes using this initial hole as the basis, but let's first set up a parameter to automatically calculate the number of holes based on the plate width using the following rules:

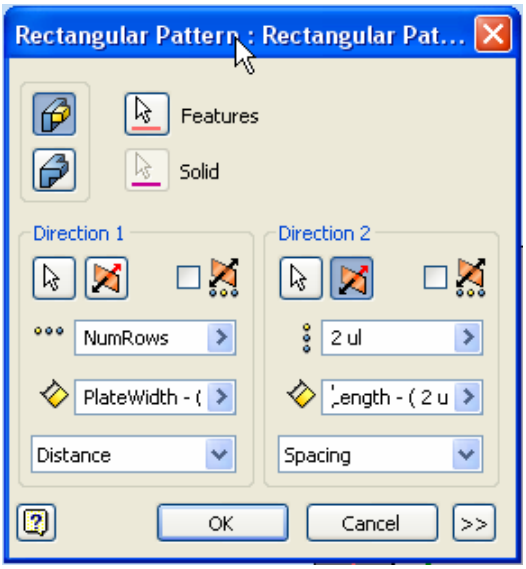
- If the width is less that 8", there should be two rows of holes across the width.
- For a plate width of 8", there should be three rows of holes
- For every additional 4" of width beyond 8", there should be another row of holes.

Let's create a user parameter called *NumRows*, which will be used when patterning the slotted hole to create the desired number of rows. Examine the formula below to see how the results meet the above requirements. (The *floor* function rounds the argument down to the next whole number, and the *max* function returns the largest value from the list of arguments.)

NumRows	ul	max(floor(PlateWidth / 4 in) + 1 ul;2 ul)

Think about the units for each term in the equation as well to see how the results are unitless, as required by the parameter.

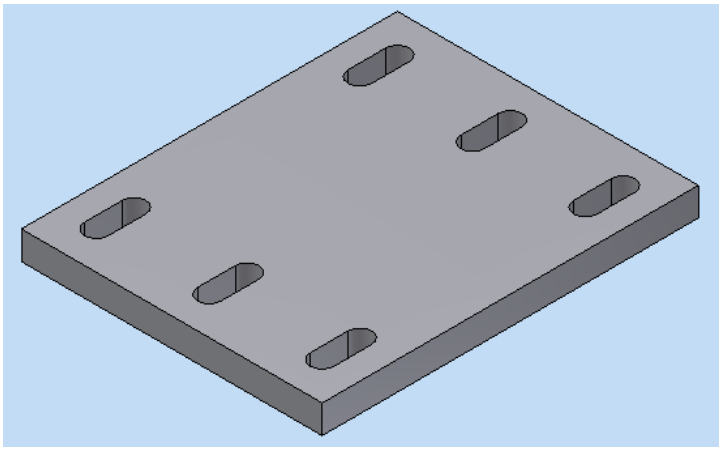
We can now create the pattern of slotted holes using the parameters we have set up. Examine the *Rectangular Pattern* dialog box below to see how the parameters have been used.



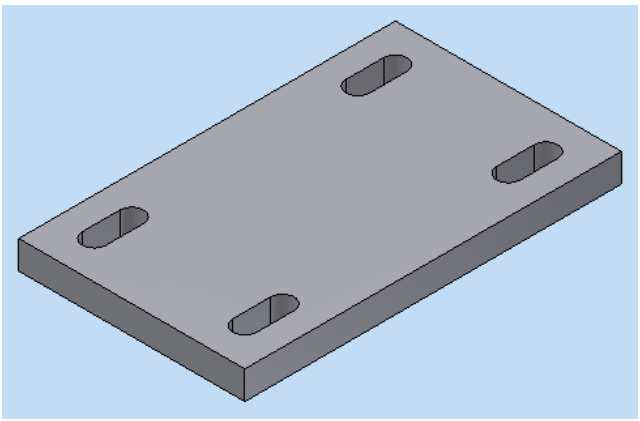
Note the use of the *NumRows* parameter to set the number of occurrences for *Direction 1*. Note also that the distance fields for both directions are too small to see the entire formulas that have been entered here. In a situation like this, the *Parameters* dialog box can be useful for seeing the entire formula, as shown below.

d13	in	PlateWidth - (2 ul * XSetBack)
d14	ul	2 ul
d16	in	Length - (2 ul * YSetback)

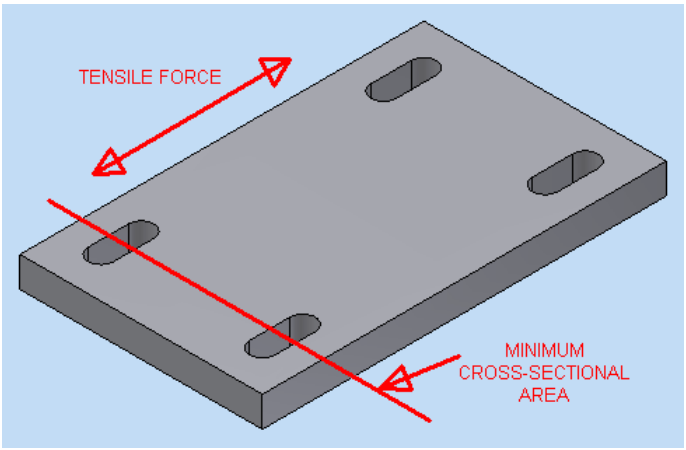
Here are the results of the feature pattern with the *PlateWidth* parameter set to 8 inches.



If *PlateWidth* is changed to 6 inches, the pattern is recalculated automatically as shown.



Let us now say that that a tensile force is applied to the plate in the direction of its length. To find the tensile stress in the plate, we need to first determine the minimum cross-sectional area across the plate, which is the area through the slots. See below.



We will create a user parameter called *CrossSection*, with the equation shown below to determine the cross-sectional area across the slots. Note again the units defined for the parameter, in this case square inches, and how this agrees with the results of the formula.

CrossSection	in^2	(PlateWidth - (NumRows * SlotWidth)) * Thickness

The value for the tensile force must be specified, and for this we will simply set up another user parameter called *TensileForce*, with units of lbforce, and enter the desired value.

TensileForce	lbforce	35000 lbforce

While advanced versions of Inventor capable of stress analysis can determine the yield strength of an assigned part material, here we are assuming only basic Inventor functionality. Therefore, we will set up the yield strength of the material as a manually entered user parameter as shown. Note the use of *psi* (pounds per square inch) as the units for the parameter. This is a standard unit of measure in Inventor.

YieldStrength	psi	48000 psi

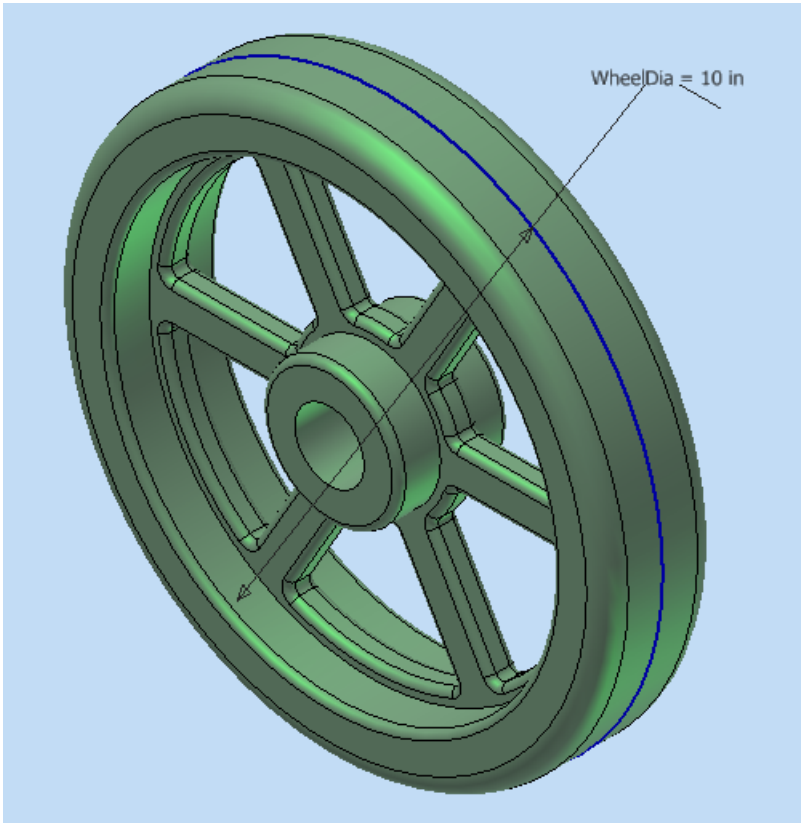
Finally, we will set up one more user parameter as shown to calculate the safety factor for the part.

SafetyFactor	ul	YieldStrength / (TensileForce / CrossSection)

We now have a parameter for the safety factor that is tied to the part geometry and to two values which are entered manually, the material yield strength and the applied tensile force. If any of these variables changes, the *SafetyFactor* parameter will be instantly updated to reflect the change, eliminating the need to pull out a calculator to check the safety factor every time any of these conditions change during the design process.

Case Study #2: Calculating the Velocity of a Vehicle

Here we have a model of a wheel whose outside diameter has been set up using a parameter called *WheelDia*, as shown below. (To create your own model and follow along with this case study, there in no need to create any complex geometry. A simple cylinder will do.)



Let's say we are still in the design stage for our wheel, and the diameter may still change. We would like to be able to quickly determine what the maximum velocity will be for the vehicle which uses this wheel whenever the wheel diameter or the angular velocity of the wheel changes.

Shown below is a user parameter for the maximum angular velocity of the wheel. Note the use of *rpm* (revolutions per minute) for the units, which is a standard unit of measurement in Inventor.

MaxRPMs	rpm	500 rpm

As it is shown here, *MaxRPMs* is simply a manually input value, but it could just as easily be linked from another file, say the model of a drive motor.

Let us now say we want to calculate the maximum vehicle speed in miles-per-hour based on our two values for the wheel diameter and the angular velocity. Even though we have

only two variables to deal with, the manual calculation is quite laborious simply because of all the conversion factors we need to employ to get from inches and RPM's to mph. The manual calculation would look something like this:

$[(500\ \text{rev/min})*(10\pi\ \text{in/rev})*(60\ \text{min/hr})] / [(12\ \text{in/ft})*(5280\ \text{ft/mile})] \approx 14.9\ \text{mph}$

And, of course, if either our angular velocity or wheel diameter changed, we would need to run through this calculation again to determine the new vehicle speed.

Let's now set up a user parameter in Inventor called *VehicleSpeed* to calculate our vehicle speed automatically. Here is what the formula looks like.

VehicleSpeed	mph	WheelDia * PI * MaxRPMs / 360 deg	14.874965

Something interesting is going on here. As we can see, the resulting value for our parameter is about 14.9 mph, which is the same value we would get if we ran through the manual calculation above, but the formula for our parameter doesn't look much like the formula for the manual calculation. (Note: the term '*PI*' is a recognized mathematical term in Inventor representing the unitless value 3.14159...)

To understand what's happening, let's think, not about units, but about the unit types. Our specified units for the parameter, *mph*, could be though of in terms of unit type as

$$\frac{\text{Length}}{\text{Time}}$$

If we look at the terms in our formula in this same way, we get this:

$$\textit{WheelDia} \quad * \quad \textit{PI} \quad * \quad \textit{MaxRPMs} \quad / \quad \frac{\textit{360}}{\textit{deg}}$$

$$\text{Length} \quad * \quad (\text{unitless}) \quad * \quad \frac{\cancel{\text{Angle}}}{\text{Time}} \quad * \quad \frac{1}{\cancel{\text{Angle}}} \quad = \quad \frac{\text{Length}}{\text{Time}}$$

Our formula works because its unit type is the same as that of the specified units for our parameter, even though the units themselves are different. It's the same concept as when, say, we have a parameter with units set to millimeters, and we enter '*1 in*' for the equation. The value for the parameter is *25.4* (there are 25.4 mm in an inch) because Inventor understands the relationship between millimeters and inches.

In this same way, Inventor understands the relationships between the different units in our calculation for the vehicle speed. Indeed, if we want our vehicle speed expressed in some other units, say meters-per-second, we simply need to change the units of the parameter, and because the type of unit is still the same, the same formula is still valid.

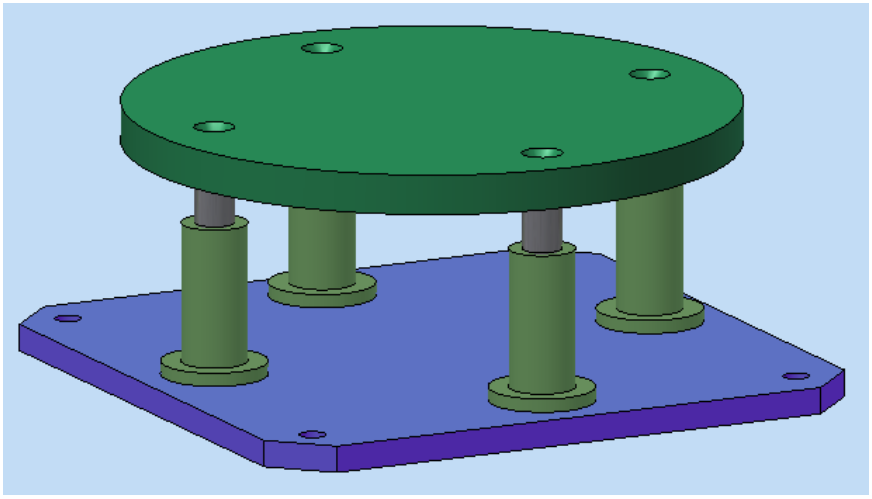
VehicleSpeed	m/s	WheelDia * PI * MaxRPMs / 360 deg	6.649704

Note here also that there is no standard Inventor unit for meters-per-second, but we were able to combine two standard units to create a custom unit.

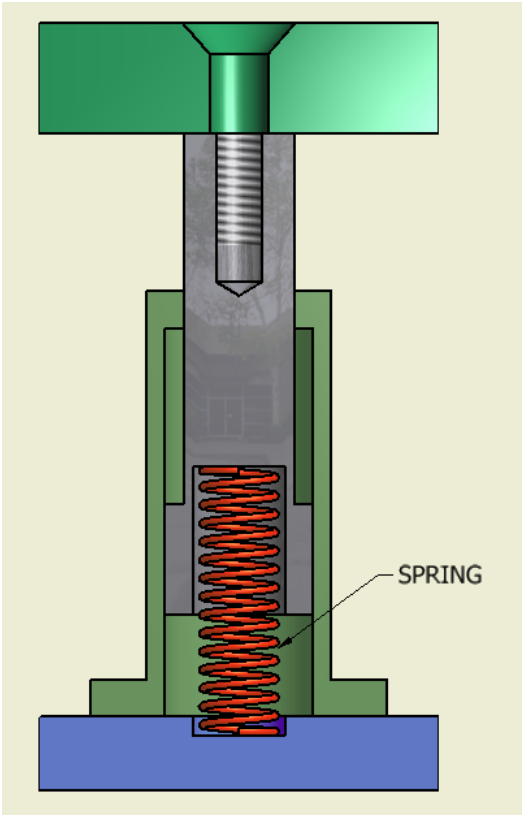
Case Study #3: Determining the Equilibrium Position of a Spring-Loaded Assembly

Here we will simulate a spring-loaded assembly with an external force applied to it. We will set up parameters to determine the equilibrium position of the assembly based on the magnitude of the force and the properties of the springs.

Our assembly looks like this...



...and a cross-section of one of the cylinders looks like this.



We wish to apply a force on the top surface of the upper plate, and based on the number and properties of the springs, we want to know the resting position of the assembly.

Let's first examine the part file for the spring. There are two pertinent parameters in this file, the free length of the spring (i.e. the length of the spring with no applied force), named *FreeLen*, and the spring constant (i.e. the force required per inch of spring compression), named *SpringConstant*.

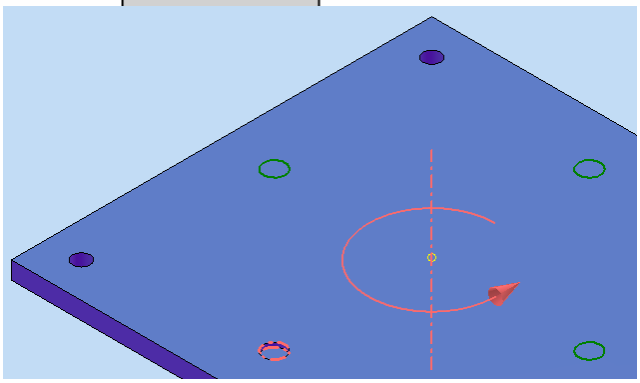
FreeLen	in	4 in
SpringConstant	lbforce/in	20 lbforce/in

Note the combination of two standard parameter units, *lbforce* and *in*, to create a custom unit for the spring constant.

(In this example, I have actually created a physical representation of the spring as a part file, which adjusts in length based on the conditions in the assembly. This requires some Inventor functionality that is beyond the scope of this course, namely *adaptivity*. I have done this primarily for illustrative purposes. One could easily set this assembly up using imaginary springs with no physical representation. In this case, the parameters above could simply be created in the assembly file rather than linked from another file.)

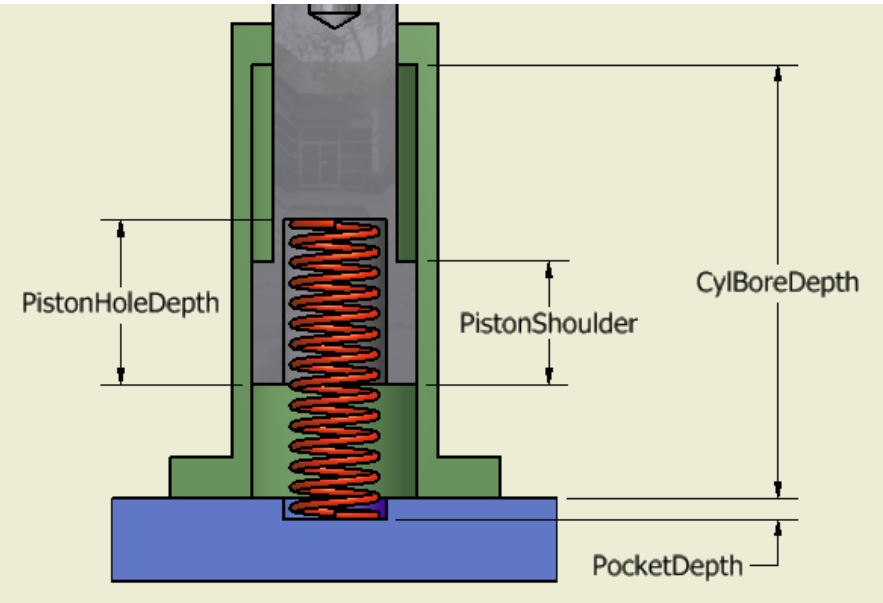
The number of cylinders used in the assembly, and therefore the number of springs, is determined by a parameter set up in the base plate file called *NumCyls*. This parameter has been used to set the number of occurrences in a circular pattern of holes which are used as pockets for the springs in the assembly.

NumCyls	ul	4 ul



The pattern of cylinders in the assembly is based on the hole pattern in the base plate.

There are several physical dimensions in our assembly which influence the springs, and parameters have been set up in the part files which we can use in determining our assembly position. Below is a cross-section of the cylinder with representations of the pertinent parameters.



Returning to the assembly, we can create links to all of the above parameters so they may be referenced by equations in our assembly.

C:\Projects\Thompson...		
PocketDepth	in	0.125 in
NumCyls	ul	4.000 ul
C:\Projects\Thompson...		
FreeLen	in	4.000 in
SpringConstant	lbforce/in	20.000 lbforce/in
C:\Projects\Thompson...		
PistonShoulder	in	0.750 in
PistonHoleDepth	in	1.000 in
C:\Projects\Thompson...		
CylBoreDepth	in	2.625 in

Looking at the cylinder cross-section, we can see that there are physical limits to how far the cylinder can extend or retract, and therefore limits to our maximum and minimum spring length. Let's set up user parameters in our assembly to determine these lengths. Examine the equations below to see how they make use of the parameters from the part files to determine these values.

MaxSpringLen	in	CylBoreDepth - PistonShoulder + PocketDepth + PistonHoleDepth	3.000000
MinSpringLen	in	PistonHoleDepth + PocketDepth	1.125000

Next, let's set up a couple of user parameters in the assembly to represent the forces applied to the springs. The total force is made up of two components, the weight of the

moving parts in the assembly, and the force applied to the top plate. Here are the parameters for those forces.

AppliedForce	lbforce	150 lbforce
TareWeight	lbforce	25 lbforce

(Even though Inventor calculates the weight of parts and assemblies based on assigned materials, these weights cannot be utilized directly as parameters when using only basic Inventor functionality.)

We now have enough information to determine the compressed length of our spring. We are assuming our springs are basic linear compression springs, which follow Hooke's law

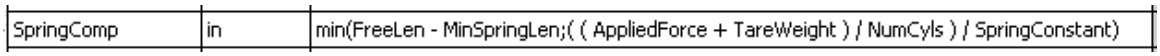
$$F = k * x \quad \text{or} \quad x = F / k,$$

where *F* is the force applied to the spring, *k* is the spring constant, and *x* is the compression of the spring.

The force on each spring can be expressed as (*AppliedForce* + *TareWeight*) / *NumCyls*, so Hooke's law can be restated in terms of our parameters as

$$x = ((AppliedForce + TareWeight) / NumCyls) / SpringConstant$$

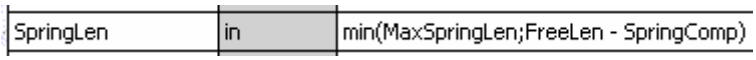
As we determined previously, there is a physical limit to how short the spring can be, for which we created the parameter *MinSpringLen*, so our maximum spring compression is found with the formula *FreeLen – MinSpringLen*. We can now set up a parameter for the compression of each spring as shown. (The *min* function returns the lowest value from the list of arguments. Each argument is separated by a semi-colon)



SpringComp	in	min(FreeLen - MinSpringLen;((AppliedForce + TareWeight) / NumCyls) / SpringConstant)

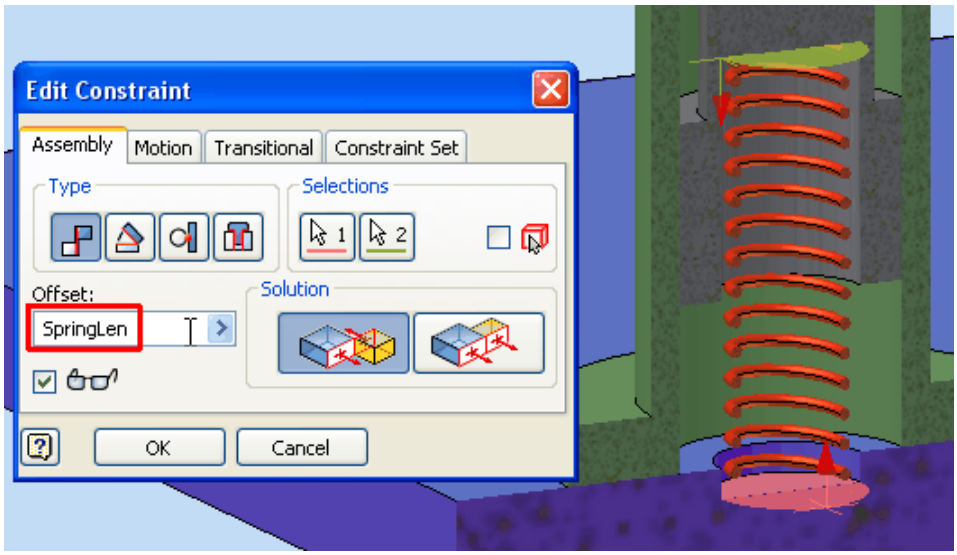
The length of each spring under load can then be determined with the formula *FreeLen – SpringComp*.

Again, however, there is a physical limitation to the maximum length of the spring, which we set up as the parameter *MaxSpringLen*, so we can set up a parameter for the actual spring length as shown.



SpringLen	in	min(MaxSpringLen;FreeLen - SpringComp)

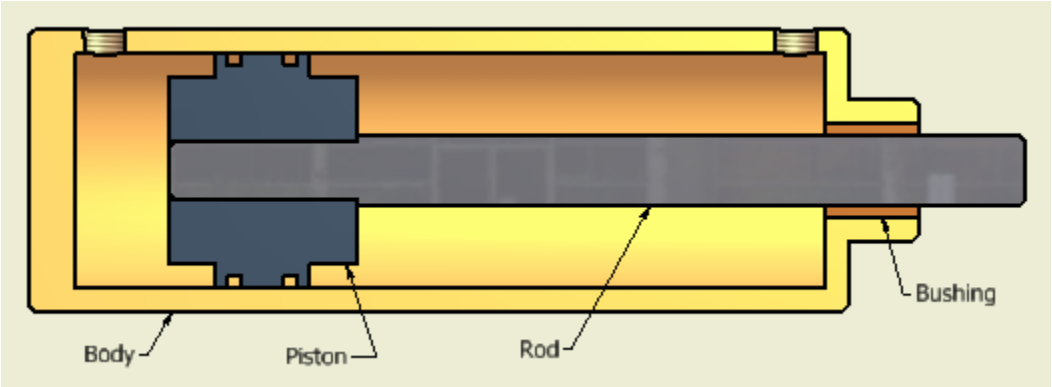
We now need simply to create an assembly constraint which makes use of this parameter to correctly position our assembly. Again, the physical representation of the spring, as shown here, is not really necessary. An imaginary spring could be used, and the parameters representing the spring properties could simply reside in the assembly file.



Though it takes a little time to create the initial setup for a design like this, we now have the ability to change a number of variables, including the properties of the spring, the force on the assembly, the number of cylinders, and/or the geometry of the cylinders, and see instantly how those changes affect the position of our assembly.

Case Study #4: Driving a Cylinder Design with a Spreadsheet

Here we have a simplified representation of a pneumatic or hydraulic cylinder consisting of four parts, a body, a piston, a rod, and a bushing



In an effort to minimize effort when experimenting with various bores, strokes, and rod diameters, we would like to make use of our ability to link parameters between files.

We might at first think the best approach would be to set up our critical parameters in the assembly file, then link to those parameters in each part file to define our geometry. In fact, this is not allowed in Inventor. This creates a circular reference whereby the geometry of a part would be defined by the assembly file, and in turn, the assembly file would be defined by the geometry of the part.

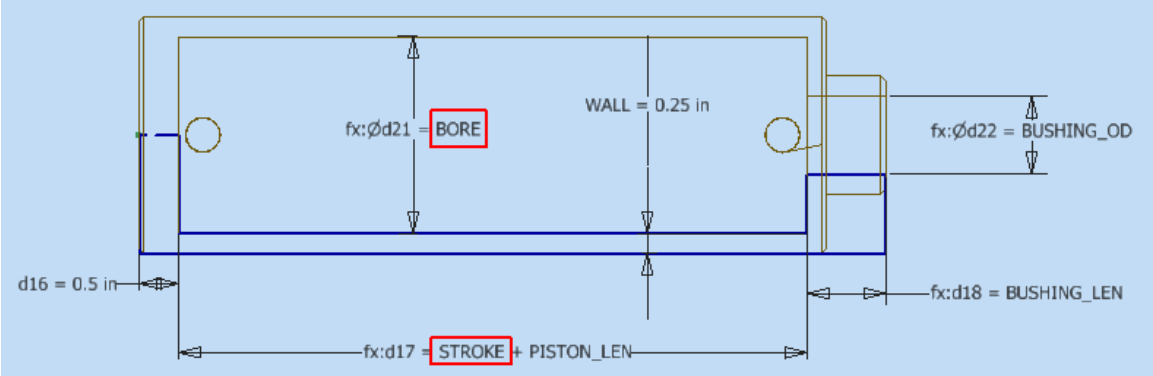
Another option would be for our critical parameters to reside in the part files, and then the parameters can be referenced between the part files as needed. This approach would work, but it could quickly become confusing to keep track of which parameters reside in which part files. For instance, in the case of our cylinder, is the parameter defining the cylinder bore in our body file or in our piston file? And, of course, the more complex the assembly becomes, the more confusing this situation could be.

This is an example of where a spreadsheet can come in handy. Below is a portion of a spreadsheet worksheet where the critical dimensions of our cylinder have been defined, as well as a few other values.

	A	B	C	D
1	Parameter Name	Equation	U/M	Comment
2	BORE	2.5	in	
3	STROKE	6	in	
4	ROD_DIA	0.75	in	
5	PRESSURE	60	psi	
6	FLOWRATE	2	ft^3/min	
7	EXT_FORCE	294.5	lbforce	
8	RET_FORCE	268.0	lbforce	
9	EXT_TIME	0.51	s	
10	RET_TIME	0.47	s	

For clarity, the shaded cells are values input by the user. Note the formatting of the spreadsheet, as defined earlier in the section *Linking Parameters from Spreadsheets*.

Each part in the assembly is linked to this spreadsheet and makes use of the three parameters which define our physical dimensions, *BORE*, *STROKE*, and *ROD_DIA*. For instance, if we look at the main sketch defining the dimensions for the cylinder body, we see the *BORE* and *STROKE* parameters in use.



The other two input values in the spreadsheet, *PRESSURE* and *FLOWRATE* are not used to define physical dimensions in our model, but are used in combination with our physical dimension parameters to calculate the values for *EXT_FORCE*, *RET_FORCE*, *EXT_TIME*, and *RET_TIME*. Note also the combination of two standard Inventor units, *ft* and *min*, for the *FLOWRATE* parameter to create a custom unit of measurement for cubic feet per minute, since *cfm* is not a recognized Inventor unit.

A link has been created to the spreadsheet for each of the part files in the cylinder assembly, as well as the assembly itself.

C:\Documents and Settings...		
BORE	in	2.5 in
STROKE	in	6 in
ROD_DIA	in	0.75 in
PRESSURE	psi	60 psi
EXT_FORCE	lbforce	294.524311274043 lbforce
RET_FORCE	lbforce	268.017123259379 lbforce
EXT_TIME	s	0.511326929295214 s
FLOWRATE	ft^3/min	2 ft^3/min
RET_TIME	s	0.465307505658644 s

While no physical dimensions from the spreadsheet are used directly in the assembly file, the link can still be handy so that the other calculated values can be viewed in the *Parameters* dialog box without having to open the spreadsheet.

Another potential advantage to using a spreadsheet for Inventor designs is that it gives non-Inventor users access to view or edit design parameters.