

CI125544

Analyze and Devise in Subassembly Composer

Kati L. Mercier, P.E.

Nathan L. Jacobson & Associates, Inc.

Learning Objectives

- Learn how to create intuitive subassemblies that give you visual information at the conceptual stage
- Learn how to create informative subassemblies that help you make key design decisions as you progress into preliminary design
- Learn how to create decisive subassemblies as you make your corridor work for you—instead of you working for it
- Learn how to create complete subassemblies that bring you through final design

Description

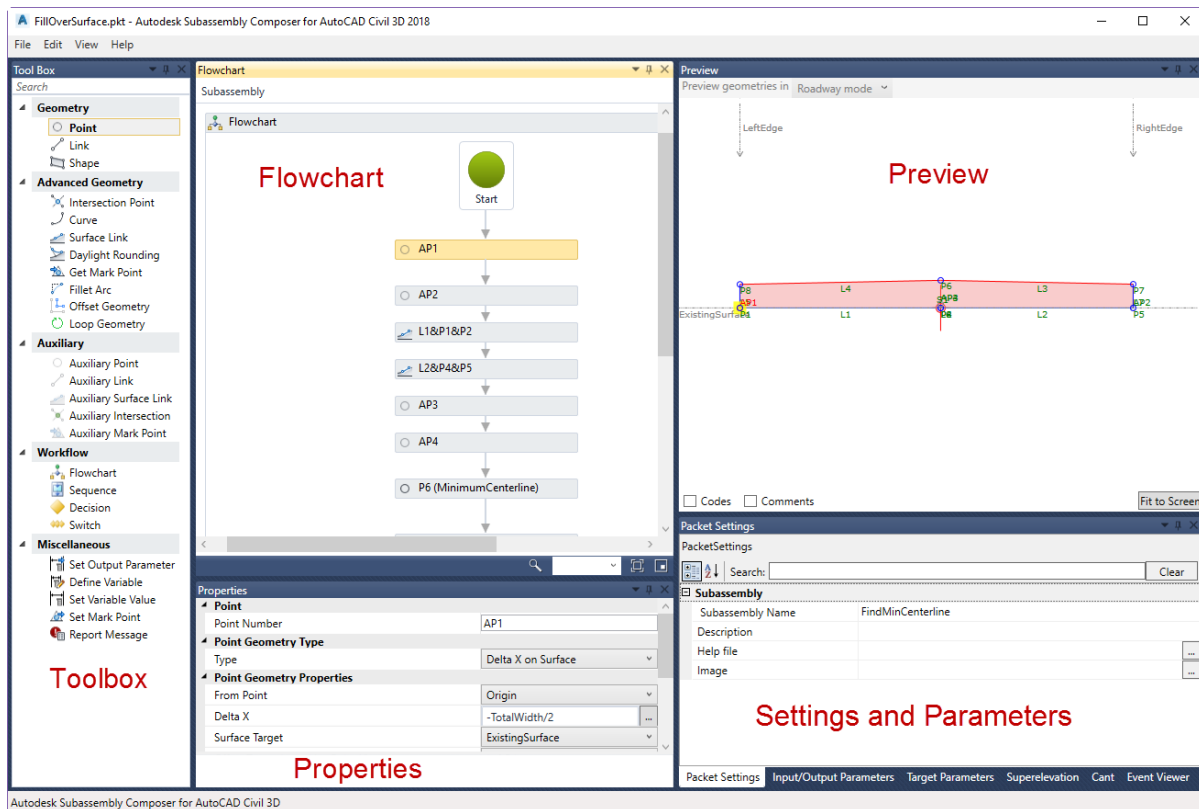
Subassembly Composer is a powerful tool for generating custom subassemblies for your corridors in AutoCAD Civil 3D software. By thinking about what your project needs in each phase of the project design you can make subassemblies that work for you. In conceptual design, you want to understand the site and visually see what you have to work with. As you progress into preliminary design, you want to gather information and start to make decisions to generate your design concept. As the design progresses, subassemblies can start to make more decisions for you. And with final design around the corner, you will finesse your design with a complete composite assembly with both your custom subassemblies and out-of-the-box subassemblies working together. With the right mindset, Subassembly Composer can generate subassemblies created especially for helping you through each of these key points in your civil designs.

Speaker

Kati Mercier is a Professional Engineer licensed in the states of Connecticut and New York. Since 2004 she has served as a project engineer at Nathan L. Jacobson & Associates, Inc., a civil and environmental engineering firm that provides engineering, review, and design consultation to local municipalities and other clients. Kati has been an associate with the firm since 2014. Kati coauthored Mastering AutoCAD® Civil 3D® 2013. She is an Autodesk® Expert Elite, AutoCAD Civil 3D Certified Professional, presented at Autodesk University 2011 and 2012, and has a special place in her heart for Autodesk® Subassembly Composer.

katimercier@gmail.com

Autodesk Subassembly Composer for AutoCAD Civil 3D 2018



The 'Subassembly Composer® for AutoCAD® Civil 3D®' (hereafter referred to as Subassembly Composer) program is used to create subassemblies to supplement the stock subassemblies that come with Civil 3D. This lecture assumes that you have a working knowledge of the program and have created basic subassembly PKT files and know how to import them into Civil 3D. All the examples in this lecture utilize Autodesk Subassembly Composer 2018.

Subassembly Composer can be a powerful tool and help provide the customization that many companies need that the stock subassemblies are unable to fully provide. That said, the stock subassemblies are also powerful tools and should not be overlooked. Subassembly Composer should not be used to recreate the wheel; if there is already a stock subassembly that does what you need for a part of your assembly use it! Remember, this is a **SUB**assembly Composer and is meant to give you the building blocks needed to build a full assembly and therefore you do not necessarily need to build the full assembly in Subassembly Composer.

In this lecture we are going to look at creating several subassemblies that you can use throughout different phases of the design process.

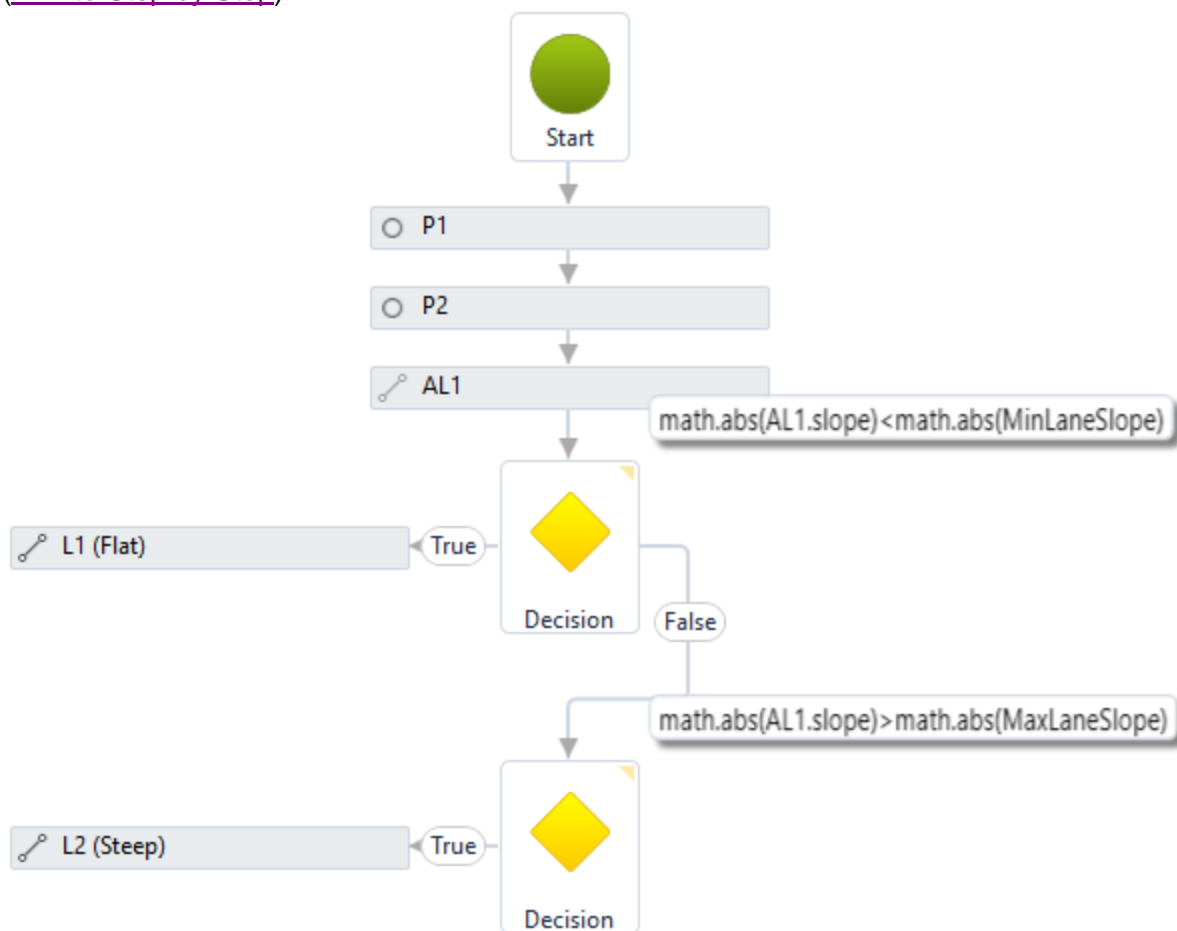
The following sections will list the logic and reasoning behind each of the examples, a step by step procedure for building each of the example subassemblies is included as bonus material at the end of this white paper (links provided throughout the discussions) along with a [four page 'cheat sheet'](#) of VB Expressions and API Functions that will be helpful to your future in subassembly creation.

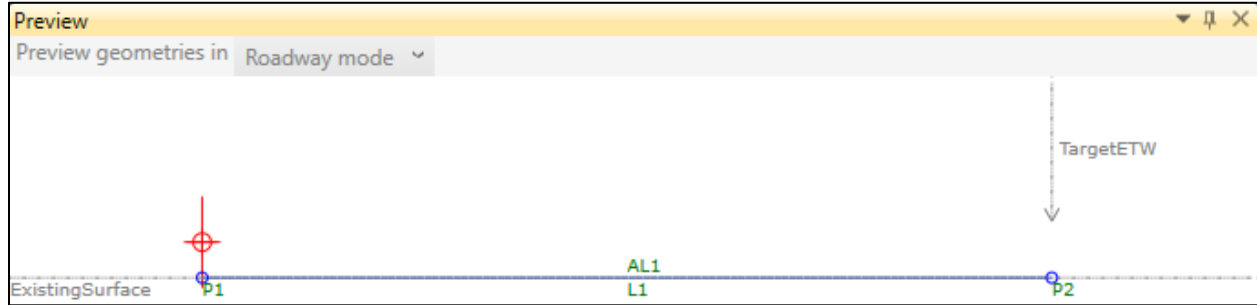
Conceptual Design: Visually Intuitive Subassemblies

Corridors do not always need to be used to create a surface, sometimes they can be used to give you information before you even start creating. In this first example we will create a subassembly that gives the user information on the existing cross slope of a road that you may want to focus more fully on as your design progresses. While a slope analysis of the surface can highlight areas of different slope ranges, it will incorporate the longitudinal slope of the road along with the cross slope of the road. In a mill and pave job, while a 6% longitudinal slope would be acceptable, you may want to remedy an area that has a 6%+ cross slope with a reconstruction to correct the cross slope.

Example 1: Colored Lane Slope Analysis

[\(Link to Step by Step\)](#)



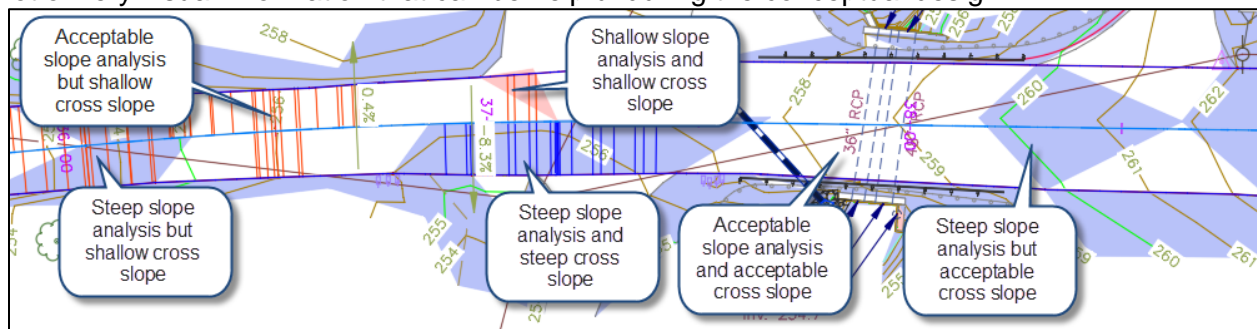


This subassembly will utilize a centerline alignment, an offset target at the edge of the travel way and a target existing surface to analyze.

Input/Output Parameters					
Name	Type	Direction	Default Value	DisplayName	Description
Side	Side	Input	Right		
MinLaneSlope	Grade	Input	1.50%		
MaxLaneSlope	Grade	Input	6.00%		
Create parameter					

The subassembly has input parameters that ask for the minimum and maximum acceptable cross slopes. The subassembly places a point at the alignment and at the edge of travel way on top of the existing surface. It then looks at the slope between the two points and depending on how the slope compares to the minimum and maximum acceptable cross slopes it will highlight the areas that fall out of the range. To highlight these zones, two existing Civil 3D stock codes are utilized; the link that is too flat will be assigned the "Daylight_Cut" code and the link that is too steep will be assigned the "Daylight_Fill" code.

As previously noted, a slope analysis highlighting areas that are shallower or steeper than the acceptable cross slope would not show the cross slope. This image shows the cross shallow cross slopes in red links and the steep cross slopes in blue links with the slope analysis areas shown in the background with light red and light blue for the same ranges of slopes. This gives a lot of very visual information that can be helpful during the conceptual design.

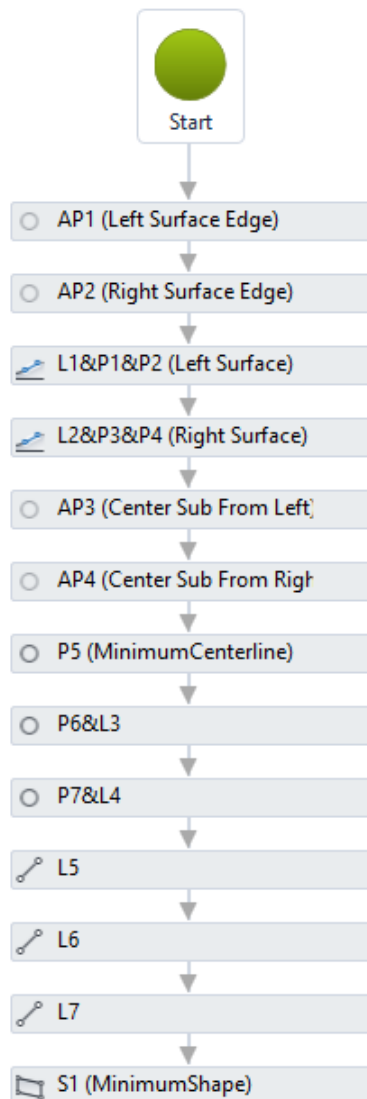


Preliminary Design: Informative Subassemblies

The next example is for a project site that had a constraint that due to an agreement with the property owner and the past use of the site, there was no cut allowed and therefore the proposed trail needed to be fully in a fill condition. To minimize the amount of material, and therefore the cost of the project, the proposed profile of the trail was critical. The subassembly for this project would use similar logic to one that you would make for an overlay road project. It will look at the existing surface in each cross section and determine based on the left slope and the right slope what the minimum elevation at the alignment would need to be. By generating a surface from this corridor and analyzing it under different scenarios an informative profile can be created for the alignment and then the user can create a proposed profile with clean vertical geometry that remains above the calculated minimum profile (thus knowing there will be no cut).

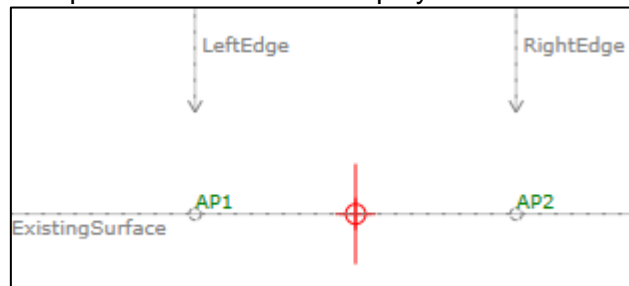
Example 2: Fill over Surface

[\(Link to Step by Step\)](#)

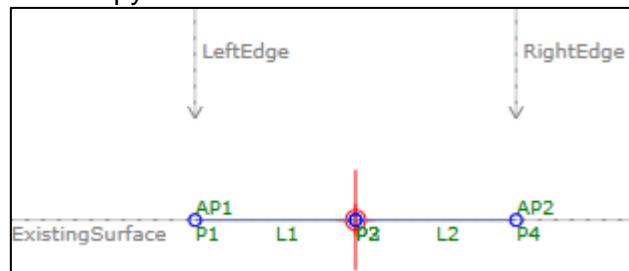


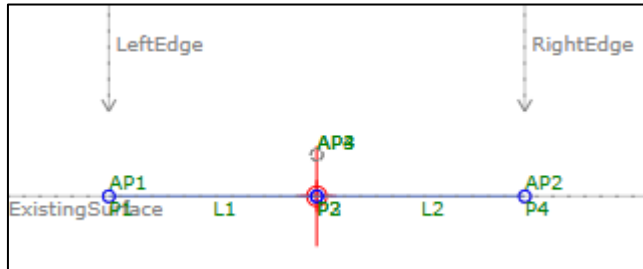
This subassembly will utilize a centerline alignment, an optional left and right edge of trail (or it will utilize the widths provided in the input parameters), and a target existing surface to analyze. The subassembly has input parameters that ask for the left and right widths, the minimum depth, and the left and right top slopes.

The subassembly places an auxiliary point at the left surface edge and an auxiliary point at the right surface edge. These auxiliary points are temporary test points used by Subassembly Composer but will not be displayed in Civil 3D.



The subassembly will then generate a surface link from the left edge to the center and another surface link from the center to the right edge. While the target surface in Subassembly Composer is visualized as a horizontal line, it will treat the cross section as the bumpy surface that it is in Civil 3D.

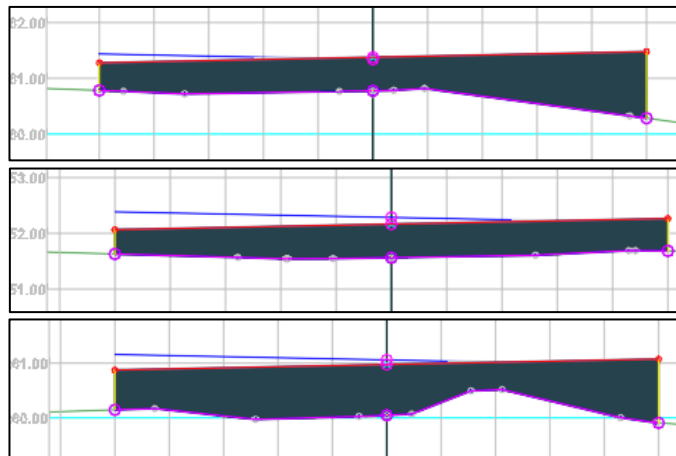
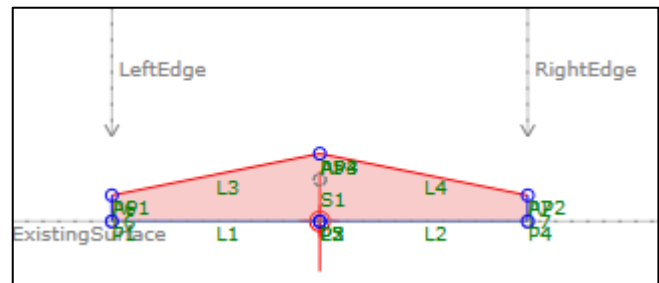




Next, it will temporarily calculate the center elevation based on the proposed slope on the right, the desired depth, and the existing surface on the right and place an auxiliary point at that centerline elevation, and will also calculate the center elevation the same way based on the left elements.

In the Subassembly Composer Preview window, it will show both auxiliary points on top of one another if the left and right slopes are the same since the target surface is treated as horizontal, but they will likely be different in an actual situation. It will then place the center profile point being the maximum of either the calculated center auxiliary point from the left or right analysis.

Once that point is placed the rest of the links for the shape of the base material can be generated utilizing the proposed slope on the left and right as well as the vertical links connecting down to the existing surface links. One last link will be needed in between the two surface links that do not share a common point, this will close the shape. Once this is done a shape can be generated.

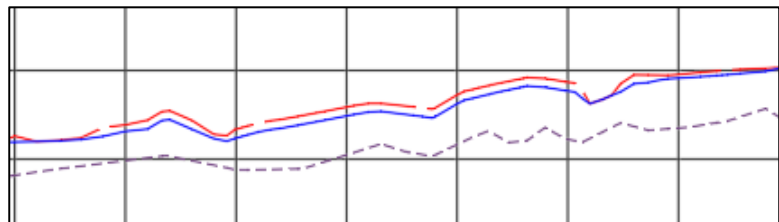


At left are three screenshots of cross sections that utilize a -2% slope on the left and a +2% on the right to create a cross sloped trail cross section that is high on the right-hand side

A similar analysis corridor was run with the high side on the left-hand side (seen in the background of the screenshots as the blue angled link). Using the surfaces generated from each of these analysis corridors, surface profiles for each were generated as shown in the image below with the right high side shown in the red long dash profile

and the left high side shown in the blue solid line profile with the existing surface centerline shown in the short dash below them both.

From this we can progress into preliminary design with this informative information to generate a proposed profile with clean vertical geometry that is above the appropriate calculated profile for whether that portion of the trail will have the high side on the left or the right.

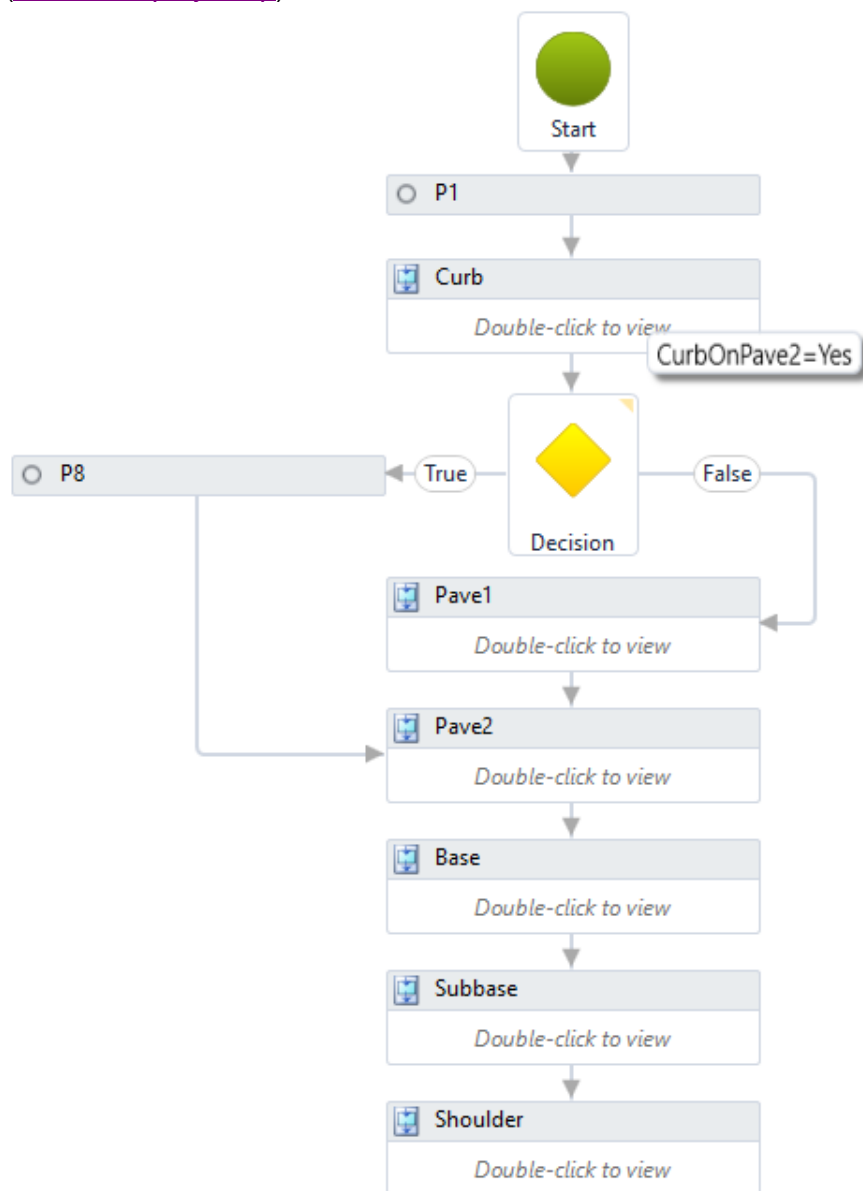


Progress: Decisive Subassemblies

Sometimes the stock subassemblies don't do exactly what you need but that doesn't mean that you shouldn't still use them when you can. Remember, don't recreate the wheel but make the pieces that you need. The next subassembly was made to solve a few of our office's curb problems. It is a bituminous curb that can either be set on the binder course of the pavement or the top course of the pavement. It also allows for the various subgrade, base, and pavement layers to have varying offsets underneath the curb creating cross sections that visually represent the construction detail. Lastly, it translates the datum link from the subgrade up to the shoulder for accurate datum surface volume computations compared to the existing surface.

Example 3: Curb Over Extended Subgrade

[\(Link to Step by Step\)](#)

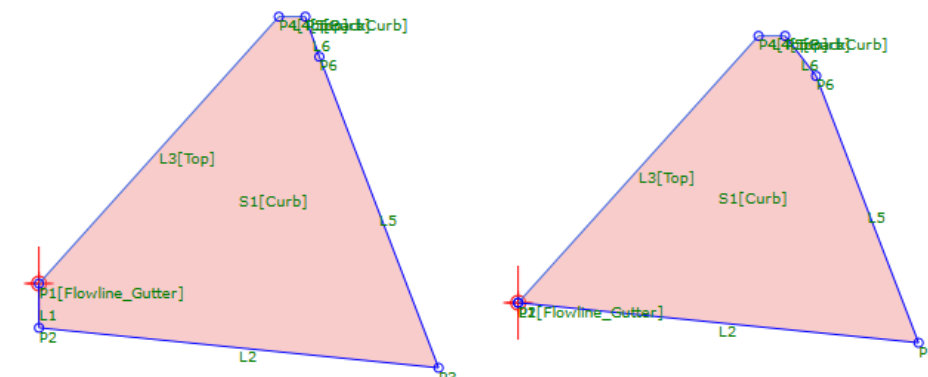


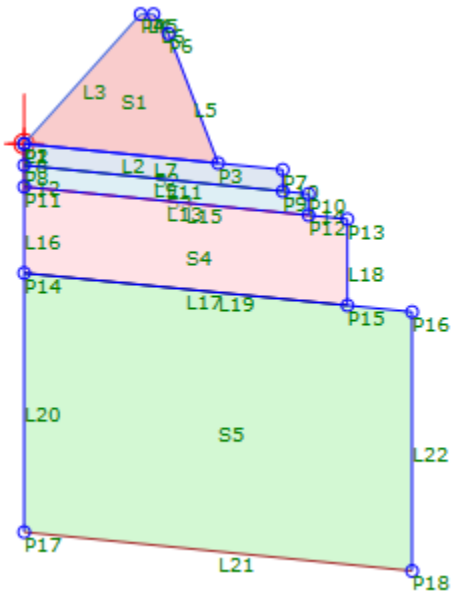
This subassembly is intended to be inserted at the gutter line after a lane stock subassembly. It utilizes similar input parameters with a few additional ones to further define the subassembly. A coworker asked why the subassembly does not also just include the lane, but it goes back to not recreating the wheel. By keeping the subassembly doing only the tasks at hand it saves time in creating the subassembly and keeps it simple. We do not need to include any superelevation calculations in this subassembly to match the subgrade slope of the lane, instead we utilize the parameter reference in Civil 3D to pass the subgrade slope from the stock LaneSuperelevationAOR subassembly to our curb subassembly, letting Civil 3D do the work.

Input/Output Parameters						
Name	Type	Direction	Default Value	DisplayName	Description	
Side	Side	Input	Right			
Pave1Depth	Double	Input	0.083			
Pave2Depth	Double	Input	0.083			
Pave1Width	Double	Input	1		Pave1 Width From Gutterline	
Pave2Width	Double	Input	1.1		Pave2 Width From Gutterline	
BaseWidth	Double	Input	1.25		Base Width From Gutterline	
SubbaseWidth	Double	Input	1.5		Pave1 Width From Gutterline	
ShoulderSlope	Grade	Input	2.00%			
CurbOnPave2	Yes\No	Input	Yes			
CurbHeight	Double	Input	0.5			
CurbBaseWidth	Double	Input	0.75			
CurbTopWidth	Double	Input	0.5			
CurbFrontAngle	Double	Input	48			
CurbBackAngle	Double	Input	21			
ShoulderWidth	Double	Input	4			
SubgradeSlope	Grade	Input	-10.00%			
BaseDepth	Double	Input	0.333			
SubbaseDepth	Double	Input	1			

Create parameter

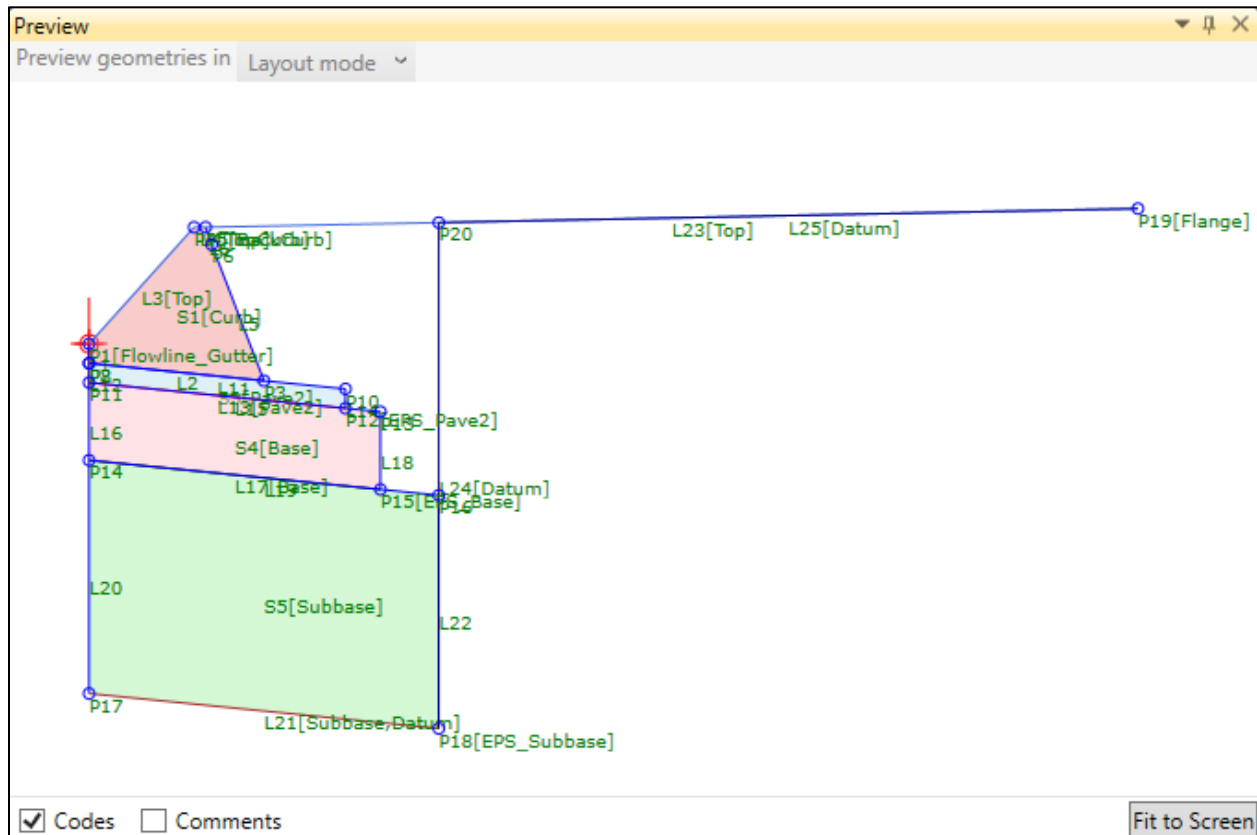
First, this subassembly generates the curb. In creating the curb, it looks at whether the user answered Yes or No to whether the Curb is on Pave2 (the binder course); if the user answers no, then it is assumed that the Curb is instead on Pave1 (the top course) and it shapes it accordingly. The curb on the left will be on Pave2 and the curb on the right will be on Pave1.



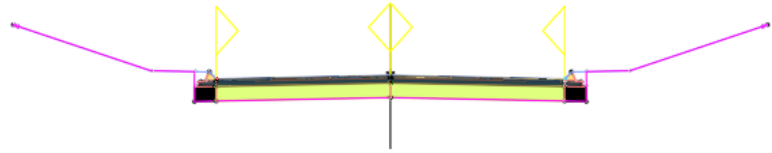


After the curb is generated the subassembly again looks at whether the curb is on Pave2 or not and generates the Pave1 layer or not. When Pave1 is generated the point at the bottom of Pave1 at the gutter line is point P8, this point will be where Pave2 is built off. If Pave1 is not generated (due to the curb being on Pave2) then point P8 is placed on its own in a second instance of the point. Point numbers cannot be duplicated twice in a subassembly unless it is on a different line of the flowchart such as this. After that Pave2, Base, and Subbase layers are generated.

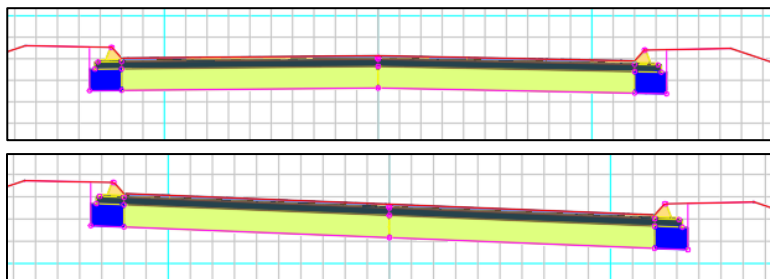
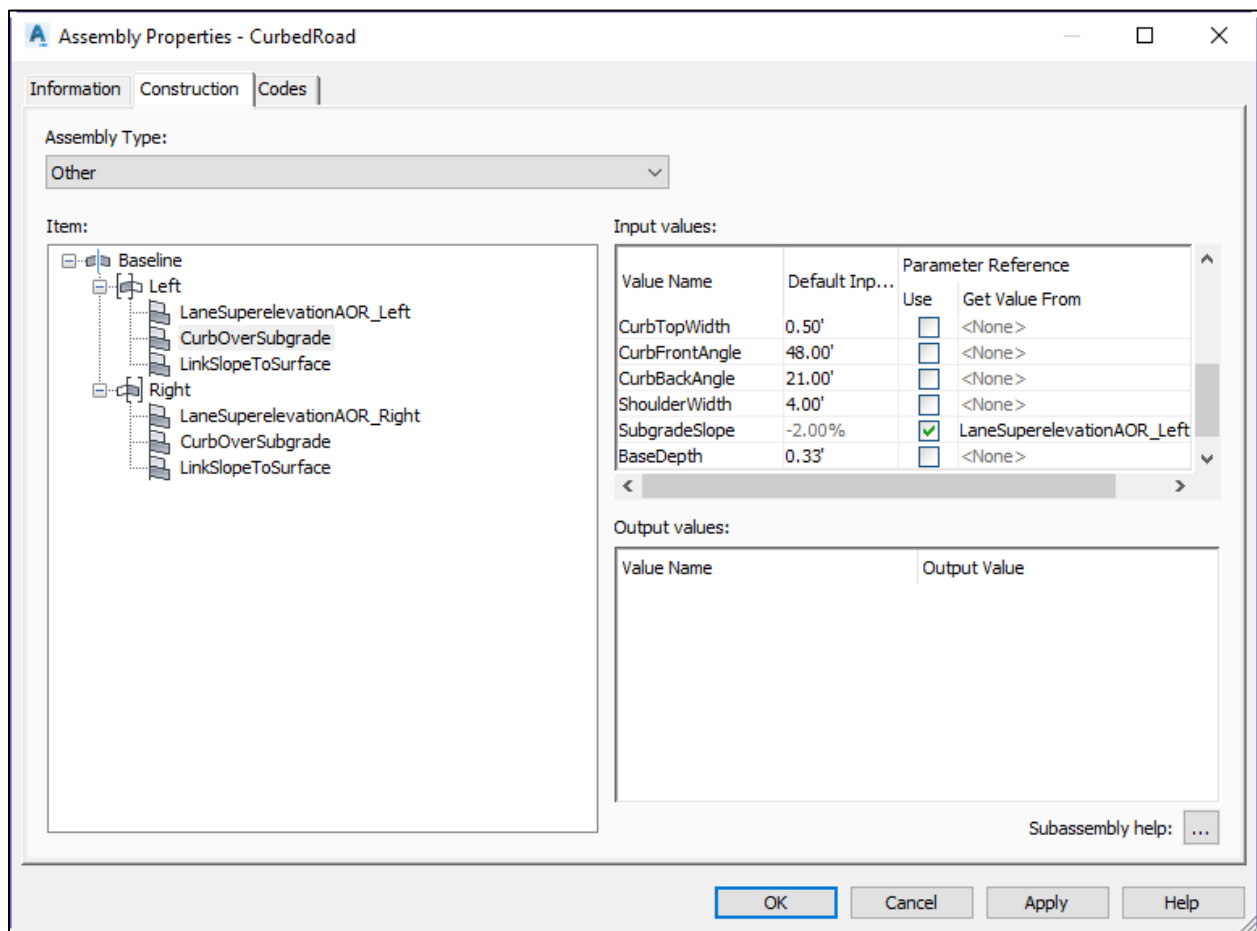
Finally, the shoulder link (top surface) is generated off the back of the curb. Once that is generated the link up from the Subbase datum link is generate, since a surface cannot have two points vertically over one another of different elevations, the intersection point generated off P18 is placed at a slope of 1000000% which will generate an X offset that will be undiscernible to the cross section. Then the datum surface portion of the shoulder link finishes the subassembly.



Once generated, the subassembly is imported in to Civil 3D and it can be used in the assembly. The assembly below utilizes the stock LaneSuperelevationAOR lane, the curb subassembly that we just created, and a generic LinkSlopeToSurface.



In the Assembly Properties window examine the Construction tab. For the SubgradeSlope in each of the Curb subassemblies select the checkbox under Use Parameter Reference and then using the dropdown list in the Get Value From column select the Lane Slope output variable.



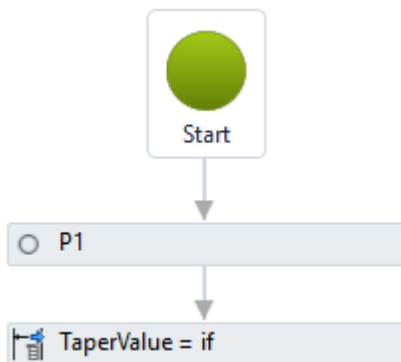
With the subgrade being passed on from the lane, the slope of the lane, including any superelevation will be continued through to the subgrade underneath the curb as shown in these sample cross sections.

Final Design: Complete Subassemblies

The last example helps create a smooth final design corridor. There will be times that you have to taper either a width or a grade or some other input parameter value. This is by far one of the simplest subassemblies but holds a lot of power when used in combination with the stock Civil 3D subassemblies (or your custom Subassembly Composer subassemblies). This assembly utilized two input parameters and ratios where the cross section is in a region and provides the ratioed output parameter to be used as a reference parameter for one of the other subassemblies in an assembly.

Example 4: Taper Input Parameter

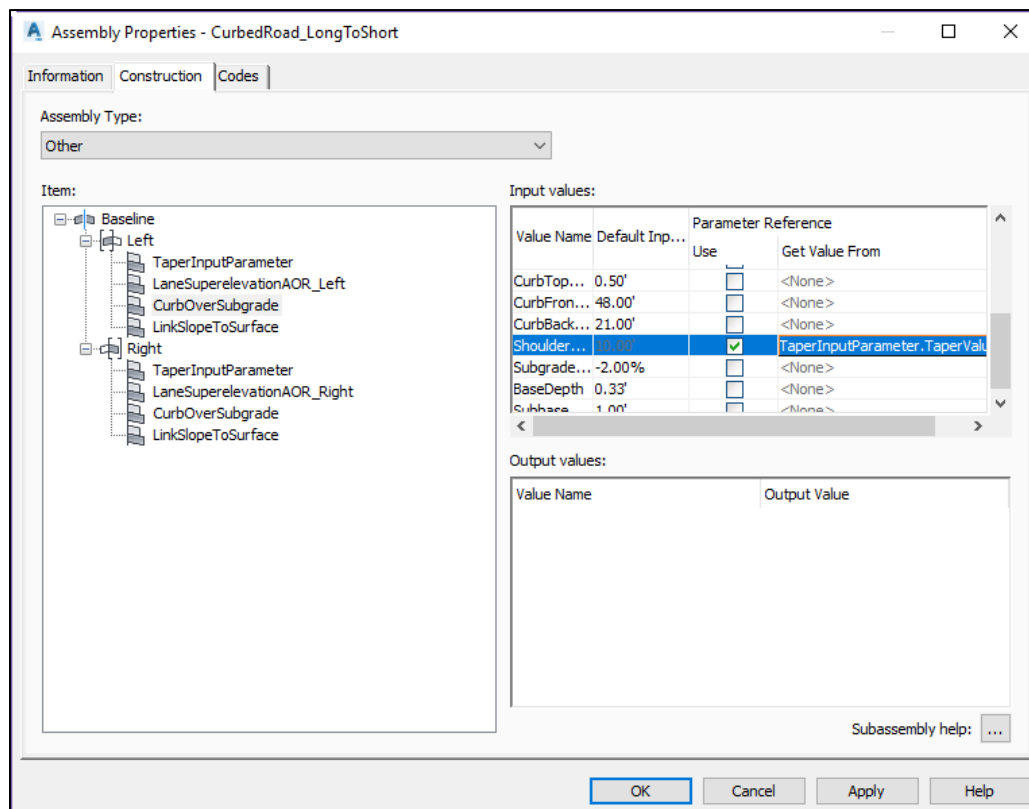
[\(Link to Step by Step\)](#)



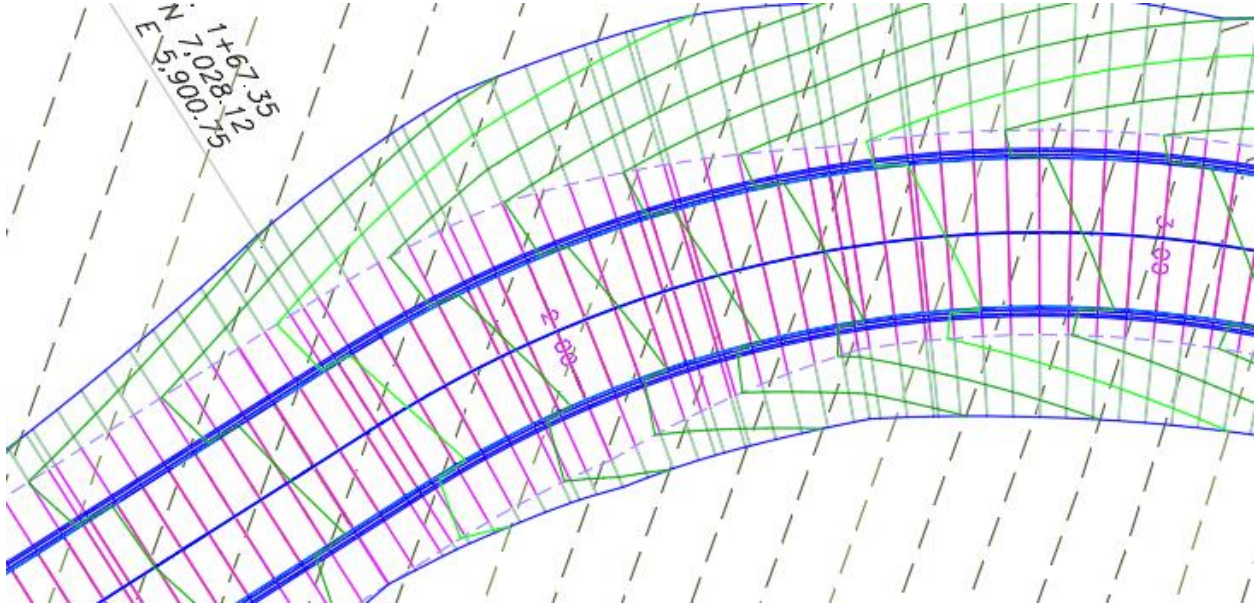
The output parameter is calculated as follows:

(SA.islayout=true, StartValue, StartValue+(Baseline.Station-Baseline.RegionStart)/(Baseline.RegionEnd-Baseline.RegionStart)(EndValue-StartValue))*

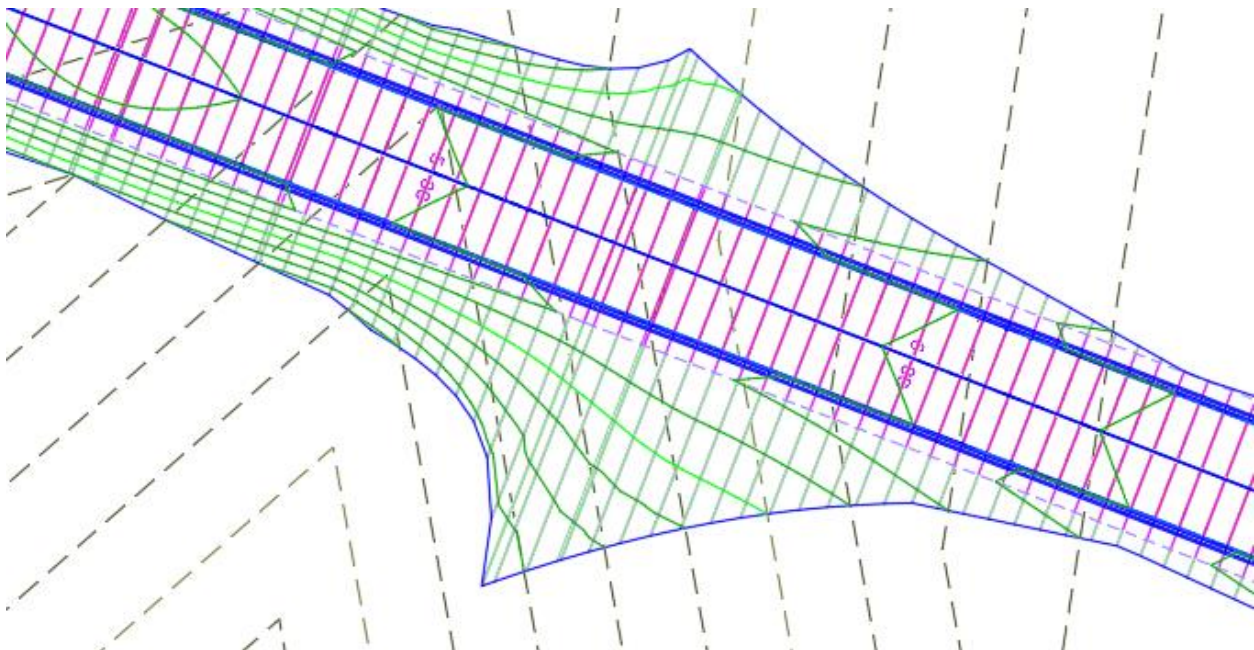
To use this subassembly place it in advance of the subassembly component that has an input parameter that you would like to taper and then in the Assembly Properties window in Civil 3D examine the Construction tab. For the parameter that you want to taper select the checkbox under Use Parameter Reference and then using the dropdown list in the Get Value column select the Taper Value output variable.



A shoulder can easily be tapered between two different widths using a target polyline when on a straight segment, but it can be complicated to taper along a curve such as in the following image (the dashed line is the shoulder hinge marker):



This same subassembly could also be used to taper a grade. Since the subassembly is set up to ratio between double (decimal) values, a 10% grade would be entered as 0.1 and a 50% grade would be entered as 0.5.



Endless Possibilities

Subassembly Composer is a versatile tool, it can help you gain intuitive visual information, gain informative information for further design, work hand in hand with stock subassemblies with output parameters, and create a smooth final design corridor. In conjunction with the stock subassemblies already available in the Civil 3D tool palette and the subassemblies you create with the tool box in the Subassembly Composer, your corridors will have endless possibilities.

The best way to learn a program is by using it, so go ahead and try the Subassembly Composer for Civil 3D 2018 to your computer if you haven't already. Before you know it, your mind will be opened to a whole different way of approaching corridors in your design.

For Further Assistance

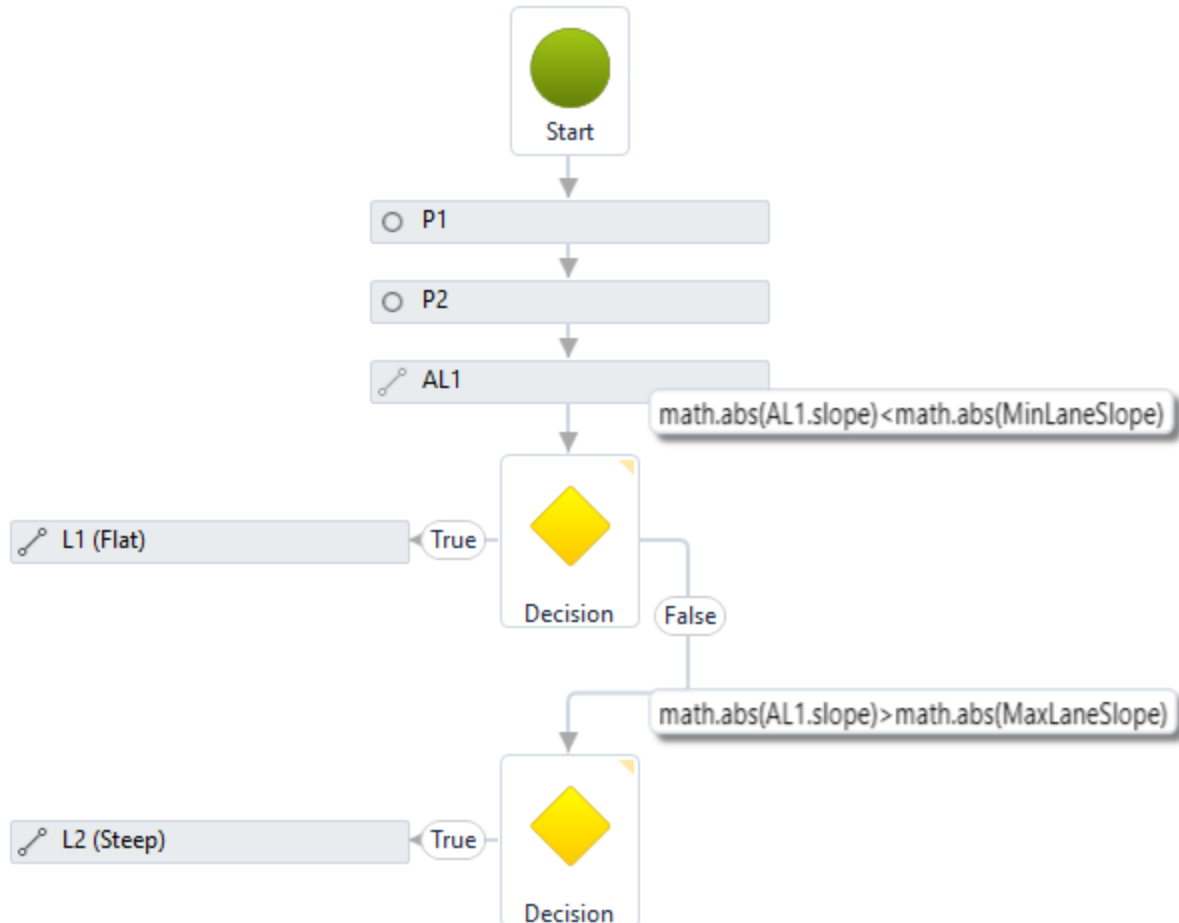
- 1 Autodesk® WikiHelp for Autodesk Subassembly Composer:
· <http://help.autodesk.com/view/CIV3D/2018/ENU/?guid=GUID-C569F4E7-D548-410E-B7D6-942A927FFD0B>
- 2 Autodesk® Discussion Groups for AutoCAD Civil 3D:
· <http://forums.autodesk.com/t5/AutoCAD-Civil-3D/bd-p/66>

(This page intentionally left blank.)

Class Subassemblies Step by Step

Subassembly: Colored Lane Slope Analysis

[\(Link back to discussion\)](#)



1) Set [Packet Settings]:

Subassembly	
Subassembly Name	ColoredLaneSlopeAnalysis
Description	
Help file	...
Image	...

2) Define the [Input/Output Parameters]:

Input/Output Parameters					
Name	Type	Direction	Default Value	DisplayName	Description
Side	Side	Input	Right		
MinLaneSlope	Grade	Input	1.50%		
MaxLaneSlope	Grade	Input	6.00%		
Create parameter					

3) Define the [Target Parameters]:

Target Parameters				
Name	Type	Preview Value	DisplayName	Enabled In Preview
ExistingSurface	Surface	-0.5		<input checked="" type="checkbox"/>
TargetETW	Offset	12		<input checked="" type="checkbox"/>
Create parameter				

4) Build the subassembly using the [Tool Box]:

a) **P1**: Drag and drop a **Point** element to the [Flowchart]. Define point **P1** as follows:

Point	
Point Number	P1
Point Codes	"Crown"
Point Geometry Type	
Type	Delta X on Surface
Point Geometry Properties	
From Point	Origin
Delta X	0
Surface Target	ExistingSurface
Offset Target (override)	None



- b) **P2**: Drag and drop a **Point** element to below P1. Define point **P2** as follows:

Point	
Point Number	P2
Point Codes	"ETW"
Point Geometry Type	
Type	Delta X on Surface
Point Geometry Properties	
From Point	P1
Delta X	12
Surface Target	ExistingSurface
Offset Target (overrid	TargetETW

- c) **AL1**: Drag and drop an **Auxiliary Link** element to below P2. Define auxiliary link **AL1** as follows:

Link	
Link Number	AL1
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P1
End Point	P2

- d) **Decision 1**: Drag and drop a **Decision** element to below AL1. Define decision as follows:



System.Activities.Statements.FlowDecision	
<div>   <input type="text" value="Search:"/> <input type="button" value="Clear"/> </div>	
Misc	
Condition	$\text{math.abs(AL1.slope)} < \text{math.abs(MinLaneSlope)}$
FalseLabel	False
TrueLabel	True

- e) **L1:** Drag and drop a **Link** element to the True side of Decision 1. Define link **L1** as follows:

Link	
Link Number	L1
Link Codes	"Daylight_Cut"
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P1
End Point	P2
Miscellaneous	
Comment	Flat

(Note: This isn't really a daylight cut but I purposefully picked a link code that I knew was already in my drawing template that would have a style to differentiate between the 'flat' and 'steep' analysis links.)

- f) **Decision 2:** Drag and drop a **Decision** element to the False side of Decision 1. Define decision as follows:

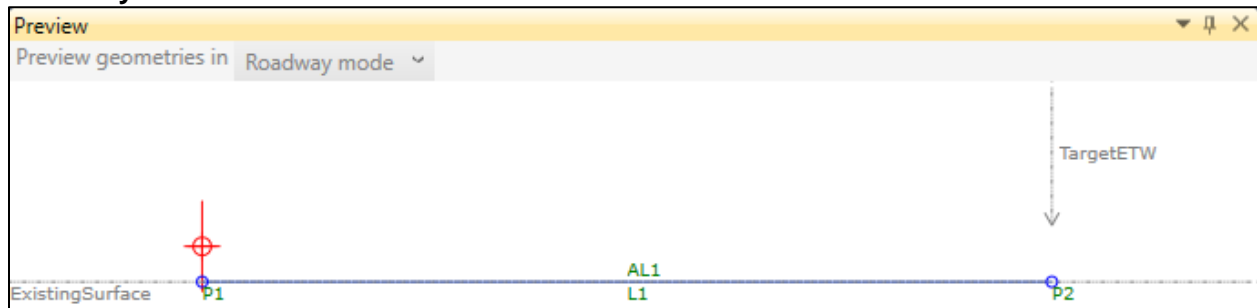
System.Activities.Statements.FlowDecision	
  Search: <input type="text"/> <input type="button" value="Clear"/>	
Misc	
Condition	$\text{math.abs(AL1.slope)} > \text{math.abs(MaxLaneSlope)}$
FalseLabel	False
TrueLabel	True

- g) **L2:** Drag and drop a **Link** element to the True side of Decision 2. Define link **L2** as follows:

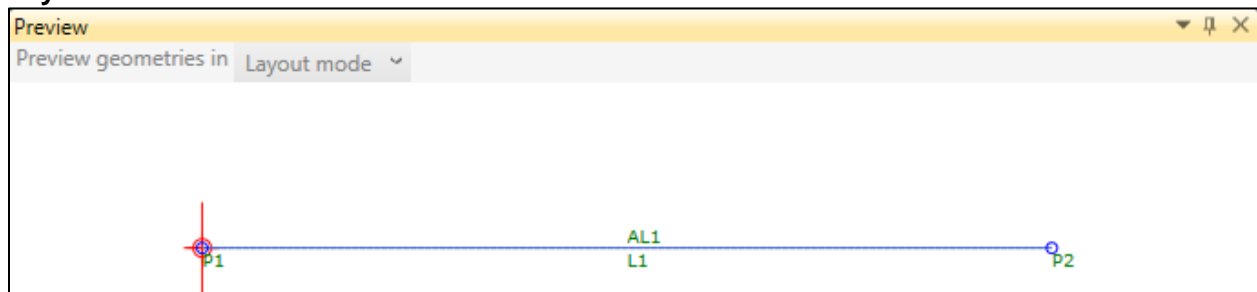
Link	
Link Number	L2
Link Codes	"Daylight_Fill"
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P1
End Point	P2
Miscellaneous	
Comment	Steep

(Note: This isn't really a daylight fill but I purposefully picked a link code that I knew was already in my drawing template that would have a style to differentiate between the 'flat' and 'steep' analysis links.)

Roadway Mode:

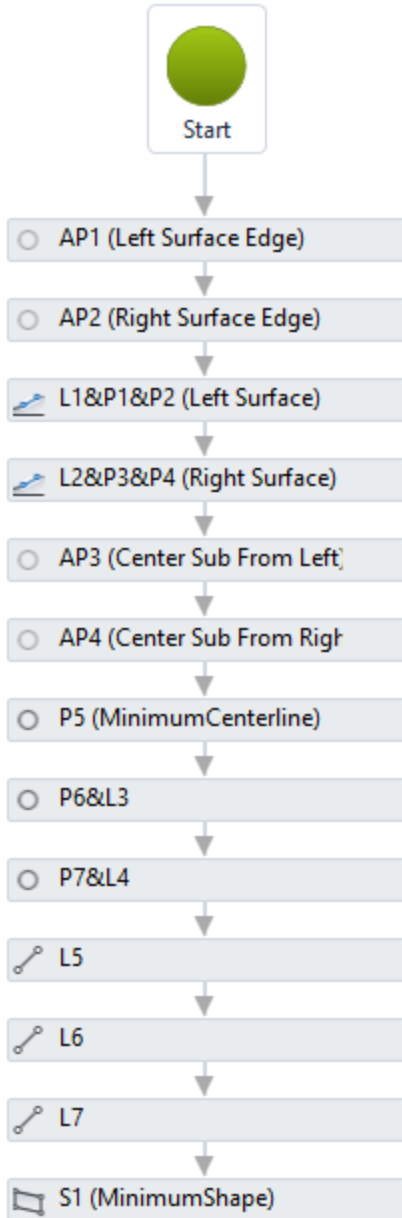


Layout Mode:



Subassembly: Fill Over Surface

([Link back to discussion](#))



1) Set [Packet Settings]:

Subassembly	
Subassembly Name	FillOverSurface
Description	
Help file	...
Image	...

2) Define the [Input/Output Parameters]:

Input/Output Parameters						
Name	Type	Direction	Default Value	DisplayName	Description	
Side	Side	Input	None			
LeftWidth	Double	Input	5			
RightWidth	Double	Input	5			
MinimumDepth	Double	Input	0.5			
LeftSlope	Grade	Input	-2.00%			
RightSlope	Grade	Input	-2.00%			
Create parameter						

3) Define the [Target Parameters]:

Target Parameters				
Name	Type	Preview Value	DisplayName	Enabled In Preview
ExistingSurface	Surface	0	Existing Surface	<input checked="" type="checkbox"/>
LeftEdge	Offset	-4	Left Edge (Optional)	<input checked="" type="checkbox"/>
RightEdge	Offset	4	Right Edge (Optional)	<input checked="" type="checkbox"/>
Create parameter				

4) Build the subassembly using the [Tool Box]:

- a) **AP1**: Drag and drop an **Auxiliary Point** element to the [Flowchart]. Define auxiliary point **AP1** as follows:

Point	
Point Number	AP1
Point Geometry Type	
Type	Delta X on Surface
Point Geometry Properties	
From Point	Origin
Delta X	-LeftWidth
Surface Target	ExistingSurface
Offset Target (overrides Delta X)	LeftEdge
Link	
Miscellaneous	
Comment	Left Surface Edge
DeltaY for Layout Mode	0

- b) **AP2:** Drag and drop an **Auxiliary Point** element to below AP1. Define auxiliary point **AP2** as follows:

Point	
Point Number	AP2
Point Geometry Type	
Type	Delta X on Surface
Point Geometry Properties	
From Point	Origin
Delta X	RightWidth
Surface Target	ExistingSurface
Offset Target (overrides Delta X)	RightEdge
Link	
Miscellaneous	
Comment	Right Surface Edge
DeltaY for Layout Mode	0

- c) **L1&P1&P2:** Drag and drop a **Surface Link** element to below AP2. Define the surface link **L1**, along with the start and end points **P1** & **P2**, as follows:

Link	
Link Number	L1
Link Codes	"Subbase", "Datum"
Geometry Properties	
Surface Target	ExistingSurface
Start X	AP1.X
End X	0
Start Offset Target (overrides Start Offset)	None
End Offset Target (overrides End Offset)	None
Depth	0
Points	
Start Point Name	P1
Start Point Codes	"ETW_Subbase"
End Point Name	P2
End Point Codes	"Crown_Subbase"
Miscellaneous	
Comment	Left Surface
Depth for Layout Mode(measured from Origin)	0

- d) **L2&P3&P4**: Drag and drop a **Surface Link** element to below L1&P1&P2. Define the surface link **L2**, along with the start and end points **P3** & **P4**, as follows:

Link	
Link Number	L2
Link Codes	"Subbase", "Datum"
Geometry Properties	
Surface Target	ExistingSurface
Start X	0
End X	AP2.x
Start Offset Target (overrides Start Offset)	None
End Offset Target (overrides End Offset)	None
Depth	0
Points	
Start Point Name	P3
Start Point Codes	
End Point Name	P4
End Point Codes	"ETW_Subbase"
Miscellaneous	
Comment	Right Surface
Depth for Layout Mode(measured from Origin)	0

- e) **AP3**: Drag and drop an **Auxiliary Point** element to below L2&P3&P4. Define auxiliary point **AP3** as follows:

Point	
Point Number	AP3
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	Origin
Delta X	0
Delta Y	L1.maxintercepty(Leftslope)
Link	
Miscellaneous	
Comment	Center Sub From Left

- f) **AP4**: Drag and drop an **Auxiliary Point** element to below AP3. Define auxiliary point **AP4** as follows:

Point	
Point Number	AP4
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	Origin
Delta X	0
Delta Y	L2.maxintercepty(Rightslope
Link	
Miscellaneous	
Comment	Center Sub From Right

- g) **P5**: Drag and drop a **Point** element to below AP4. Define point **P5** as follows:

Point	
Point Number	P5
Point Codes	"Crown"
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	Origin
Delta X	0
Delta Y	math.max(AP3.y,AP4.y)+Mir
Link	
Miscellaneous	
Comment	MinimumCenterline

- h) **P6&L3**: Drag and drop a **Point** element to below P5. Define point **P6**, along with its connecting link **L3**, as follows:

Point	
Point Number	P6
Point Codes	"ETW"
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P5
Slope	LeftSlope L ...
Delta X	-LeftWidth ...
Offset Target (overrides Delta X)	LeftEdge
Elevation Target (overrides Slope and Supereleva	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L3
Codes	"Top"
ApplyAOR	<input type="checkbox"/>

- i) **P7&L4**: Drag and drop a **Point** element to below P6&L3. Define point **P7**, along with its connecting link **L4**, as follows:

Point	
Point Number	P7
Point Codes	"ETW"
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P5
Slope	RightSlope F ...
Delta X	RightWidth ...
Offset Target (overrides Delta X)	RightEdge
Elevation Target (overrides Slope and Supereleva	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L4
Codes	"Top"
ApplyAOR	<input type="checkbox"/>

- j) **L5:** Drag and drop a **Link** element to below P7&L4. Define link **L5** as follows:

Link	
Link Number	L5
Link Codes	
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P2
End Point	P3

(Note: The two surface links both created their own start and end points. Although points P2 and P3 are on top of one another they will cause a gap in the shape which therefore will cause an error unless you close the shape with a zero-length link between them.)


- k) **L6:** Drag and drop a **Link** element to below L5. Define link **L6** as follows:

Link	
Link Number	L6
Link Codes	
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P1
End Point	P6

- l) **L7:** Drag and drop a **Link** element to below L6. Define link **L7** as follows:

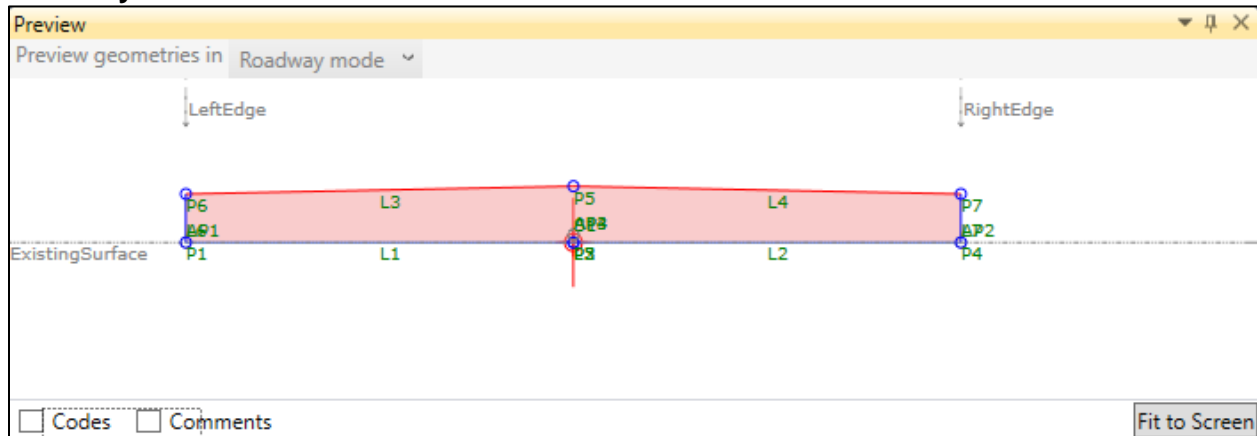
Link	
Link Number	L7
Link Codes	
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P7
End Point	P4

m) **S1**: Drag and drop a **Shape** element to below L7. Define shape **S1** as follows:

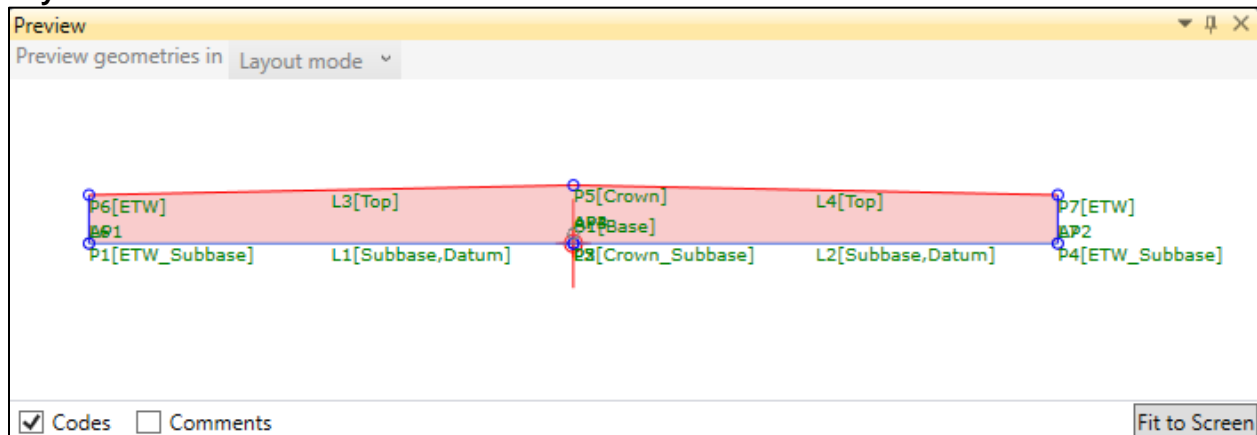
Shape	
Shape Number	S1
Shape Codes	"Base"
Component	
Links	L1 L5 L3 L4 L7 L2 L6 Add Link 
Miscellaneous	
Comment	MinimumShape

(Note: To add the link components you can click the button at the right and then click inside the shape in the preview window and the links will automatically fill in.)

Roadway Mode:

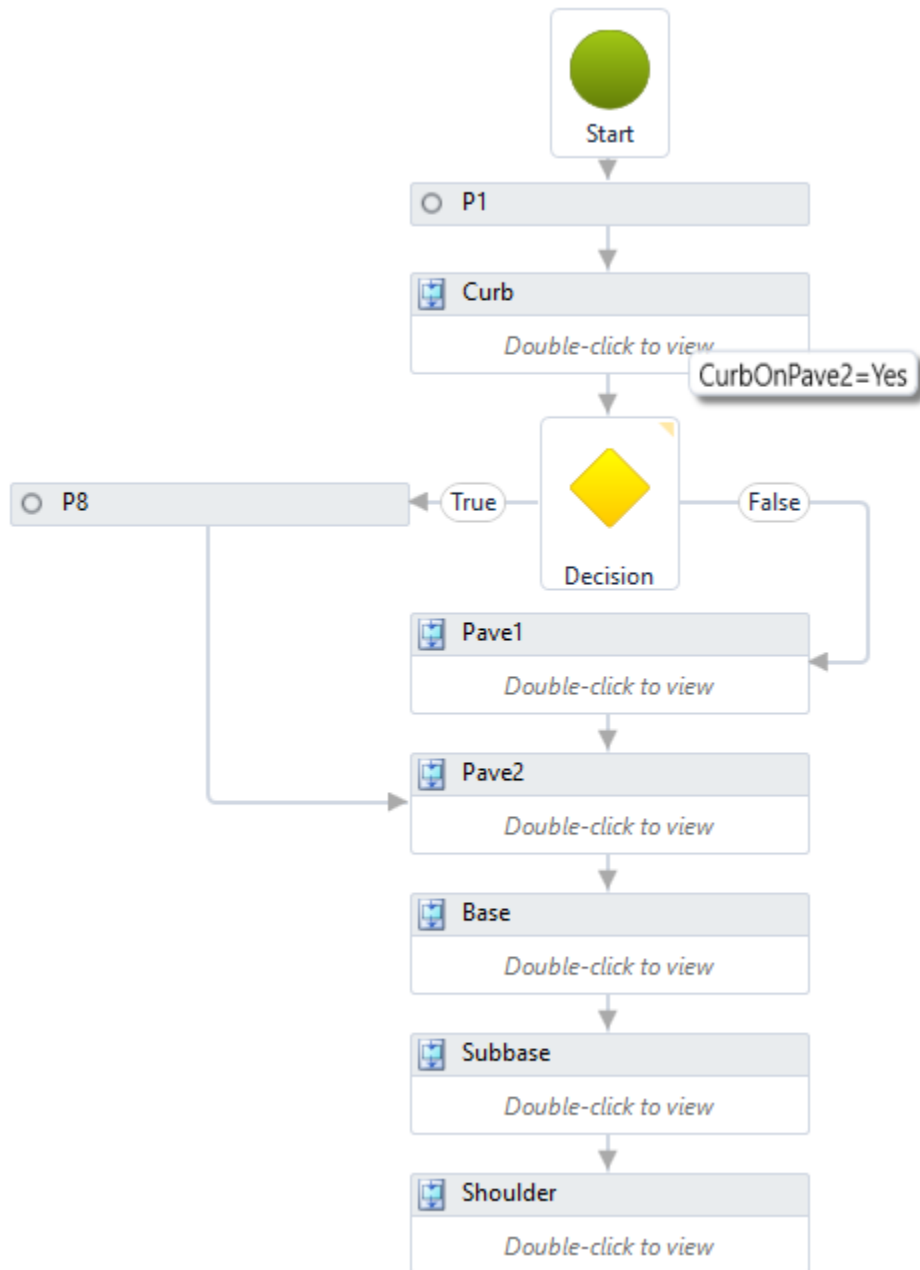


Layout Mode with Codes on:



Subassembly: Curb Over Extended Subgrade

([Link back to discussion](#))



1) Set [Packet Settings]:

Packet Settings ▼ 🔍 ✕

PacketSettings

🔍 Search: Clear

☒ **Subassembly**

Subassembly Name	CurbOverSubgrade
Description	
Help file	...
Image	...

2) Define the [Input/Output Parameters]:

Input/Output Parameters ▼ 🔍 ✕

Name	Type	Direction	Default Value	DisplayName	Description
Side	Side	Input	Right		
Pave1Depth	Double	Input	0.083		
Pave2Depth	Double	Input	0.083		
Pave1Width	Double	Input	1		Pave1 Width From Gutterline
Pave2Width	Double	Input	1.1		Pave2 Width From Gutterline
BaseWidth	Double	Input	1.25		Base Width From Gutterline
SubbaseWidth	Double	Input	1.5		Pave1 Width From Gutterline
ShoulderSlope	Grade	Input	2.00%		
CurbOnPave2	Yes\No	Input	Yes		
CurbHeight	Double	Input	0.5		
CurbBaseWidth	Double	Input	0.75		
CurbTopWidth	Double	Input	0.5		
CurbFrontAngle	Double	Input	48		
CurbBackAngle	Double	Input	21		
ShoulderWidth	Double	Input	4		
SubgradeSlope	Grade	Input	-10.00%		
BaseDepth	Double	Input	0.333		
SubbaseDepth	Double	Input	1		

[Create parameter](#)

3) Define the [Target Parameters]:

Target Parameters ▼ 🔍 ✕

Name	Type	Preview Value	DisplayName	Enabled In Preview
ShoulderHingeElevation	Elevation	0.75		<input checked="" type="checkbox"/>
ShoulderHingeOffset	Offset	4		<input checked="" type="checkbox"/>

[Create parameter](#)

4) Build the subassembly using the [Tool Box]:

a) **P1**: Drag and drop a **Point** element to the [Flowchart]. Define point **P1** as follows:

Point	
Point Number	P1
Point Codes	"Flowline_Gutter"
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	Origin
Delta X	0
Delta Y	0

b) **Curb Sequence**: Drag a **Sequence** element to the [Flowchart] below P1. Double click into the **Curb** sequence and define the following:

i) **P2&L1**: Drag and drop a **Point** element into the Curb sequence. Define point **P2**, along with its connecting link **L1**, as follows:

Point	
Point Number	P2
Point Codes	
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	P1
Delta X	0
Delta Y	if(CurbOnPave2=Yes,-Pave1Depth,0)
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L1
Codes	
ApplyAOR	<input type="checkbox"/>

- ii) **P3&L2:** Drag and drop a **Point** element below P2&L1. Define point **P3**, along with its connecting link **L2**, as follows:

Point	
Point Number	P3
Point Codes	
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P2
Slope	SubgradeSlope S ...
Delta X	CurbBaseWidth ...
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L2
Codes	
ApplyAOR	<input type="checkbox"/>

- iii) **P4&L3:** Drag and drop a **Point** element below P3&L2. Define point **P4**, along with its connecting link **L3**, as follows:

Point	
Point Number	P4
Point Codes	"TopCurb"
Point Geometry Type	
Type	Angle and Delta Y
Point Geometry Properties	
From Point	P1
Reference Point	None
Reverse Angle Reference	<input type="checkbox"/>
Angle	CurbFrontAngle ...
Delta Y	CurbHeight ...
Elevation Target (overrides Delta)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L3
Codes	"Top"
ApplyAOR	<input type="checkbox"/>

- iv) **P5&L4**: Drag and drop a **Point** element below P4&L3. Define point **P5**, along with its connecting link **L4**, as follows:

Point	
Point Number	P5
Point Codes	"BackCurb"
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	P4
Delta X	CurbTopWidth-L3.xlength
Delta Y	0
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L4
Codes	"Top"
ApplyAOR	<input type="checkbox"/>

- v) **P6&L5**: Drag and drop a **Point** element below P5&L4. Define point **P6**, along with its connecting link **L5**, as follows:

Point	
Point Number	P6
Point Codes	
Point Geometry Type	
Type	Angle and Delta Y
Point Geometry Properties	
From Point	P3
Reference Point	None
Reverse Angle Reference	<input checked="" type="checkbox"/>
Angle	270+CurbBackAngle
Delta Y	L1.ylength+L3.ylength
Elevation Target (overrides Delta	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L5
Codes	
ApplyAOR	<input type="checkbox"/>

vi) **L6**: Drag and drop a Link element below P6&L5. Define link **L6** as follows:

Link	
Link Number	L6
Link Codes	
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P5
End Point	P6

vii) **S1**: Drag and drop a **Shape** element to below L6. Define shape **S1** as follows:

Shape	
Shape Number	S1
Shape Codes	"Curb"
Component	
Links	L1 L3 L4 L6 L5 L2 <i>Add Link</i>

c) **Decision**: Drag and drop a **Decision** element to below the Curb sequence. Define decision as follows:

Misc	
Condition	CurbOnPave2=Yes
FalseLabel	False
TrueLabel	True

d) **Pave1** Sequence: Drag a **Sequence** element to the [Flowchart] to the False side of the Decision. Double click into the **Pave1** sequence and define the following:

- i) **P7&L7:** Drag and drop a **Point** element into the Pave1 sequence. Define point **P7**, along with its connecting link **L7**, as follows:

Point	
Point Number	P7
Point Codes	
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P2
Slope	SubgradeSlope S ...
Delta X	Pave1Width ...
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L7
Codes	
ApplyAOR	<input type="checkbox"/>

- ii) **P8&L8:** Drag and drop a **Point** element below P7&L7. Define point **P8**, along with its connecting link **L8**, as follows:

Point	
Point Number	P8
Point Codes	
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	P2
Delta X	0 ...
Delta Y	-math.abs(Pave1Depth) ...
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L8
Codes	
ApplyAOR	<input type="checkbox"/>


- iii) **P9&L9**: Drag and drop a **Point** element below P8&L8. Define point **P9**, along with its connecting link **L9**, as follows:

Point	
Point Number	P9
Point Codes	"EPS_Pave1"
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P8
Slope	L7.slope L ...
Delta X	L7.xlength ...
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L9
Codes	"Pave1"
ApplyAOR	<input type="checkbox"/>

- iv) **L10**: Drag and drop a Link element below P9&L9. Define link **L10** as follows:

Link	
Link Number	L10
Link Codes	
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P7
End Point	P9

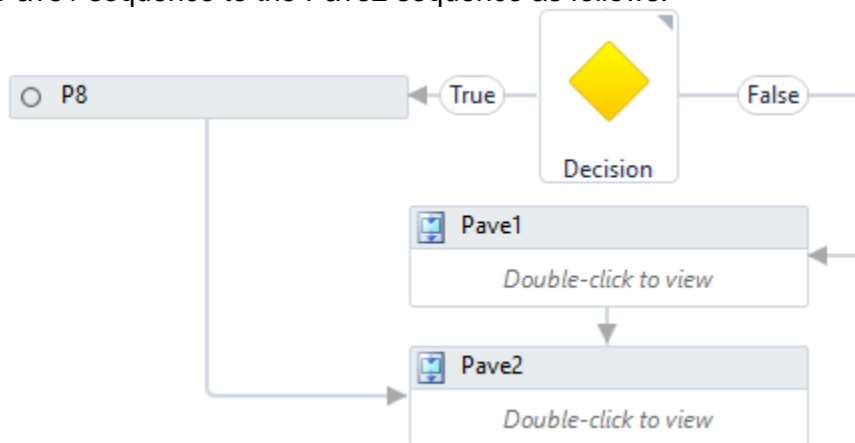
- v) **S2**: Drag and drop a **Shape** element to below L10. Define shape **S2** as follows:

Shape	
Shape Number	S2
Shape Codes	"Pave1"
Component	
Links	L7 L10 L9 L8 <i>Add Link</i> 

- e) **P8** (second instance): Drag a **Point** element to the [Flowchart] to the True side of the Decision. Rename the point as P8 and define point P8 as follows:

Point	
Point Number	P8
Point Codes	
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	P2
Delta X	0
Delta Y	0

- f) **Pave2** Sequence: Drag a **Sequence** element to the [Flowchart] below the Pave1 sequence. **Connect** the P8 (second instance) point to the Pave2 sequence and the Pave1 sequence to the Pave2 sequence as follows:



Double click into the **Pave2** sequence and define the following:

- i) **P10&L11:** Drag and drop a **Point** element into the Pave2 sequence. Define point **P10**, along with its connecting link **L11**, as follows:

Point	
Point Number	P10
Point Codes	
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P8
Slope	SubgradeSlope S ...
Delta X	Pave2Width ...
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L11
Codes	
ApplyAOR	<input type="checkbox"/>

- ii) **P11&L12:** Drag and drop a **Point** element below P10&L11. Define point **P11**, along with its connecting link **L12**, as follows:

Point	
Point Number	P11
Point Codes	
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	P8
Delta X	0 ...
Delta Y	-math.abs(Pave2Depth) ...
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L12
Codes	
ApplyAOR	<input type="checkbox"/>

- iii) **P12&L13:** Drag and drop a **Point** element below P11&L12. Define point **P12**, along with its connecting link **L13**, as follows:

Point	
Point Number	P12
Point Codes	"EPS_Pave2"
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P11
Slope	L11.slope
Delta X	L11.xlength
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L13
Codes	"Pave2"
ApplyAOR	<input type="checkbox"/>

- iv) **L14:** Drag and drop a Link element below P12&L13. Define link **L14** as follows:

Link	
Link Number	L14
Link Codes	
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P10
End Point	P12

- v) **S3:** Drag and drop a **Shape** element to below L14. Define shape **S3** as follows:

Shape	
Shape Number	S3
Shape Codes	"Pave2"
Component	
Links	L11 L14 L13 L12 Add Link

- g) **Base Sequence:** Drag a **Sequence** element to the [Flowchart] below the Pave2 sequence. Double click into the **Base** sequence and define the following:
- i) **P13&L15:** Drag and drop a **Point** element into the Base sequence. Define point **P13**, along with its connecting link **L15**, as follows:

Point	
Point Number	P13
Point Codes	
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P11
Slope	SubgradeSlope S
Delta X	BaseWidth
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L15
Codes	
ApplyAOR	<input type="checkbox"/>

- ii) **P14&L16:** Drag and drop a **Point** element below P13&L15. Define point **P14**, along with its connecting link **L16**, as follows:

Point	
Point Number	P14
Point Codes	
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	P11
Delta X	0
Delta Y	$-\text{math.abs}(\text{BaseDepth})$
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L16
Codes	
ApplyAOR	<input type="checkbox"/>

- iii) **P15&L17** Drag and drop a **Point** element below P14&L16. Define point **P15**, along with its connecting link **L17**, as follows:

Point	
Point Number	P15
Point Codes	"EPS_Base"
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P14
Slope	L15.slope L ...
Delta X	L15.xlength ...
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L17
Codes	"Base"
ApplyAOR	<input type="checkbox"/>

- iv) **L18**: Drag and drop a Link element below P15&L17. Define link **L18** as follows:

Link	
Link Number	L18
Link Codes	
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P13
End Point	P15

- v) **S4**: Drag and drop a **Shape** element to below L18. Define shape **S4** as follows:

Shape	
Shape Number	S4
Shape Codes	"Base"
Component	
Links	L15 L18 L17 L16 <i>Add Link</i> 

- h) **Subbase** Sequence: Drag a **Sequence** element to the [Flowchart] below the Base sequence. Double click into the **Subbase** sequence and define the following:
- i) **P16&L19**: Drag and drop a **Point** element into the Subbase sequence. Define point **P16**, along with its connecting link **L19**, as follows:

Point	
Point Number	P16
Point Codes	
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P14
Slope	SubgradeSlope S ...
Delta X	SubbaseWidth ...
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L19
Codes	
ApplyAOR	<input type="checkbox"/>

- ii) **P17&L20**: Drag and drop a **Point** element below P16&L19. Define point **P17**, along with its connecting link **L20**, as follows:

Point	
Point Number	P17
Point Codes	
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	P14
Delta X	0 ...
Delta Y	$-\text{math.abs}(\text{SubbaseDepth})$...
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L20
Codes	
ApplyAOR	<input type="checkbox"/>


- iii) **P18&L21**: Drag and drop a **Point** element below P17&L20. Define point **P18**, along with its connecting link **L21**, as follows:

Point	
Point Number	P18
Point Codes	"EPS_Subbase"
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P17
Slope	L19.slope L ...
Delta X	L19.xlength ...
Offset Target (overrides Delta X)	None
Elevation Target (overrides Slope)	None
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L21
Codes	"Subbase", "Datum"
ApplyAOR	<input type="checkbox"/>

- iv) **L22**: Drag and drop a Link element below P18&L21. Define link **L22** as follows:

Link	
Link Number	L22
Link Codes	
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P16
End Point	P18

- v) **S5**: Drag and drop a **Shape** element to below L22. Define shape **S5** as follows:

Shape	
Shape Number	S5
Shape Codes	"Subbase"
Component	
Links	L19 L22 L21 L20 <i>Add Link</i> 

- i) **Shoulder** Sequence: Drag a **Sequence** element to the [Flowchart] below the Subbase sequence. Double click into the **Shoulder** sequence and define the following:
- i) **P19&L23**: Drag and drop a **Point** element into the Shoulder sequence. Define point **P19**, along with its connecting link **L23**, as follows:

Point	
Point Number	P19
Point Codes	"Flange"
Point Geometry Type	
Type	Slope and Delta X
Point Geometry Properties	
From Point	P5
Slope	ShoulderSlope S ...
Delta X	$\text{math.max(ShoulderWidth,L21.xlength-CurbTopWidth+.001)}$...
Offset Target (overrides Delta X)	ShoulderHingeOffset
Elevation Target (overrides Slope)	ShoulderHingeElevation
Superelevation (overrides Slope)	None
Link	
Add Link to From Point	<input checked="" type="checkbox"/>
Name	L23
Codes	"Top"
ApplyAOR	<input type="checkbox"/>

- ii) **P20**: Drag and drop an Intersection Point element below P19&L23. Define point P20 as follows:

Point	
Point Number	P20
Point Codes	
Point Geometry Type	
Type	Intersection: LinkPointSlope
Geometry Properties	
Link	L23
Point	P18
Extend Link	<input type="checkbox"/>
Slope	1000000.00% 1 ...
Extend Slope	<input type="checkbox"/>
Reverse Slope	<input type="checkbox"/>

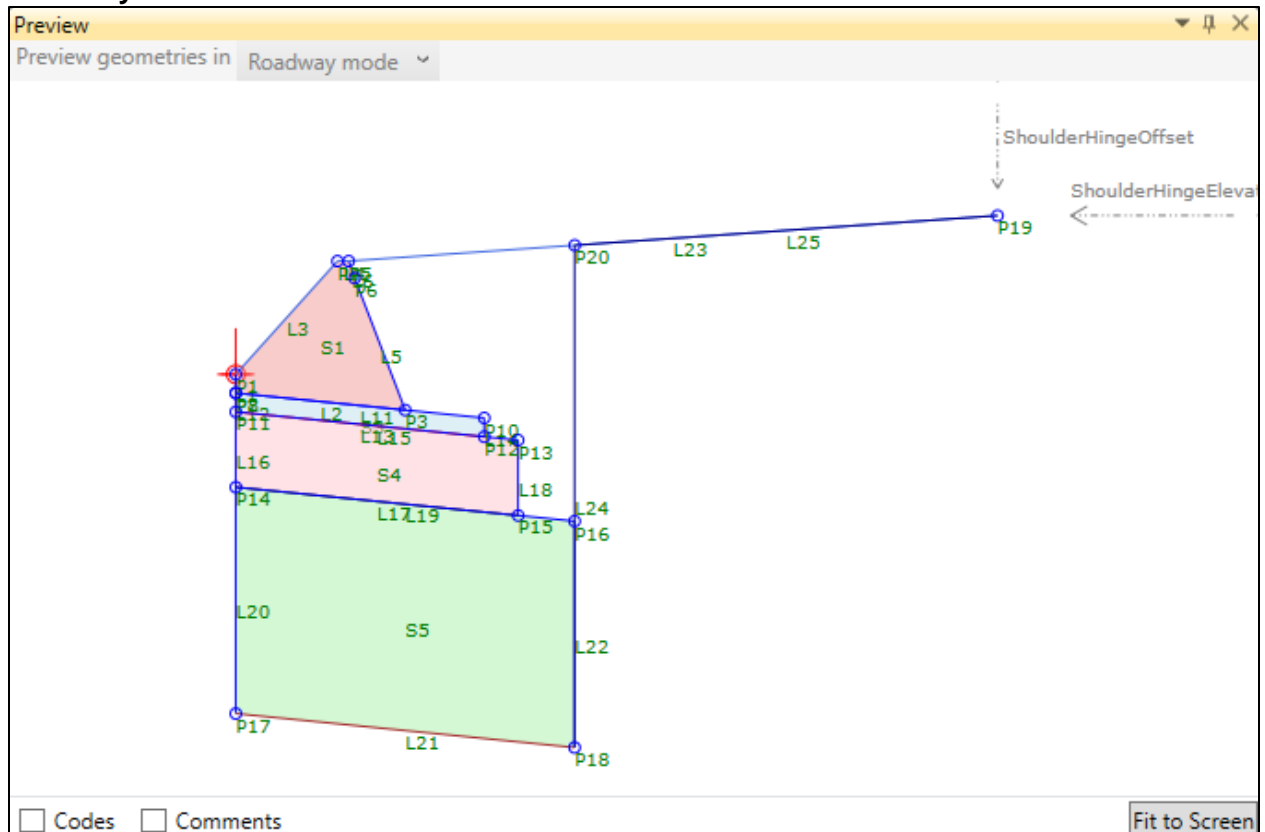
iii) **L24**: Drag and drop a Link element below P20. Define link **L24** as follows:

Link	
Link Number	L24
Link Codes	"Datum"
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P18
End Point	P20

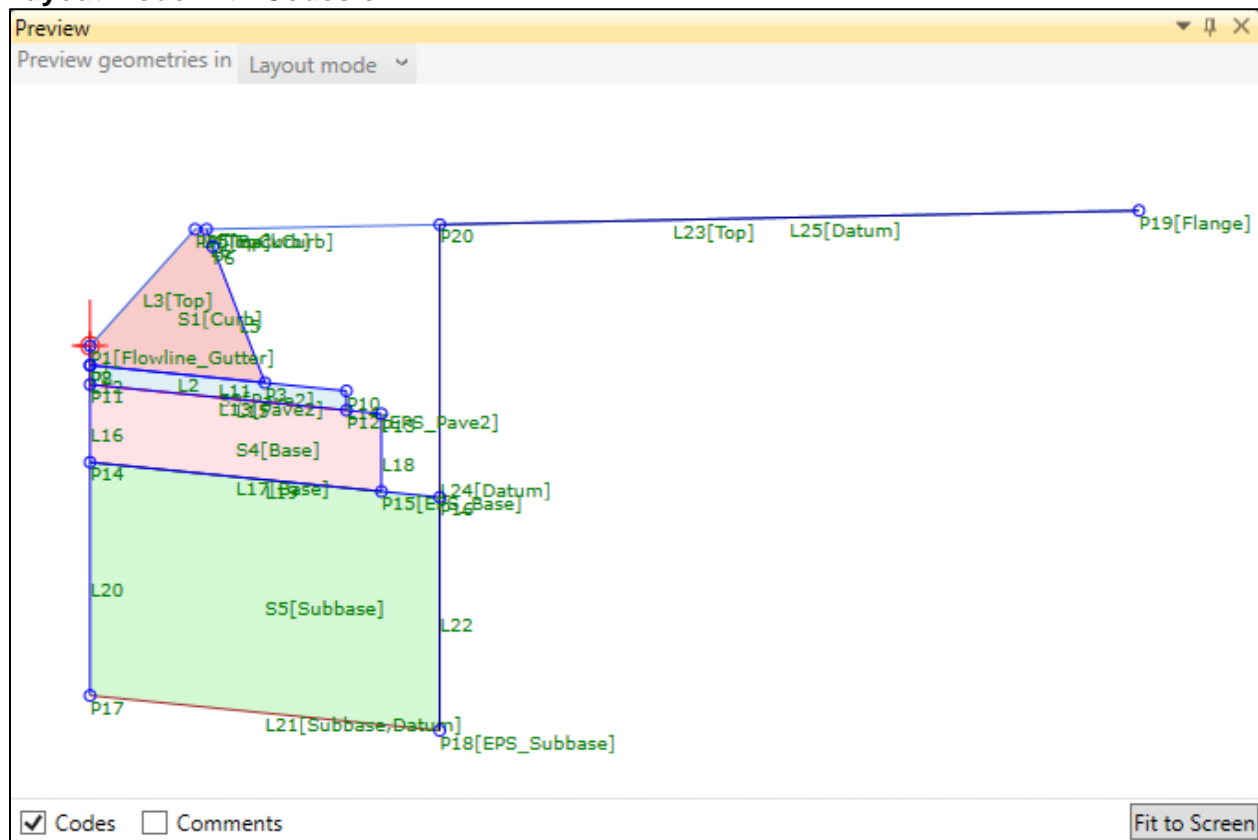
iv) **L25**: Drag and drop a Link element below L24. Define link **L25** as follows:

Link	
Link Number	L25
Link Codes	"Datum"
ApplyAOR	<input type="checkbox"/>
Position	
Start Point	P20
End Point	P19

Roadway Mode:

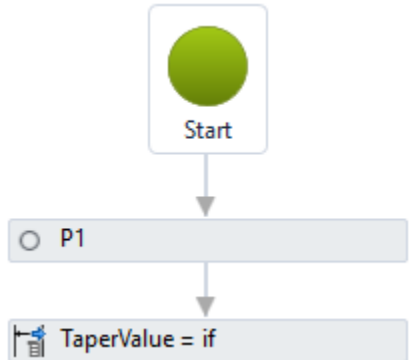


Layout Mode with Codes on:



Subassembly: Taper Input Parameter

([Link back to discussion](#))



- 1) Set [Packet Settings]:

Subassembly	
Subassembly Name	TaperInputParameter
Description	
Help file	...
Image	...

- 2) Define the [Input/Output Parameters]:

Input/Output Parameters						
Name	Type	Direction	Default Value	DisplayName	Description	
Side	Side	Input	None			
StartValue	Double	Input	2			
EndValue	Double	Input	3			
TaperValue	Double	Output	0			
Create parameter						

- 3) This subassembly has no [Target Parameters].

4) Build the subassembly using the [Tool Box]:

a) **P1**: Drag and drop a **Point** element to the [Flowchart]. Define point **P1** as follows:

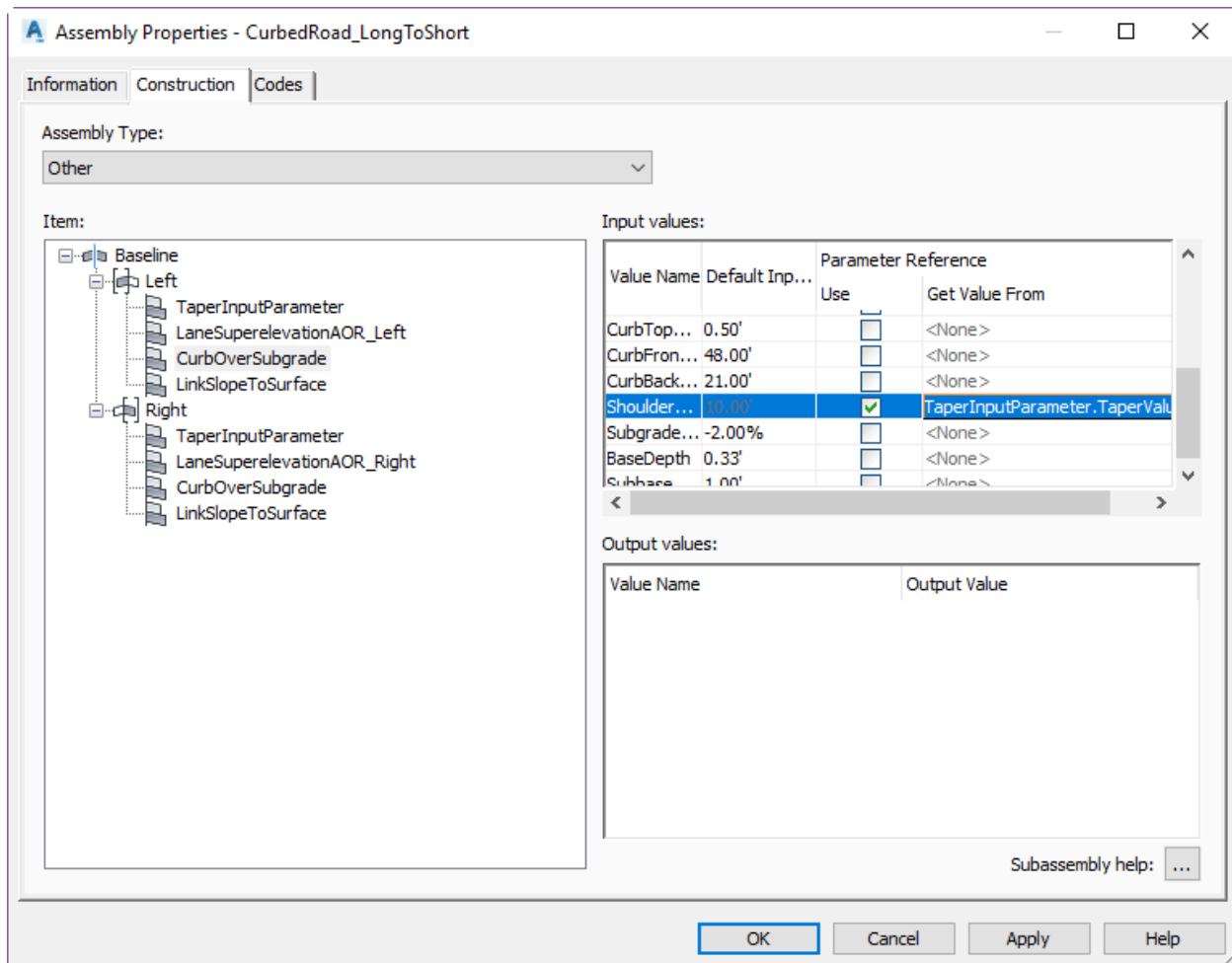
Point	
Point Number	P1
Point Codes	
Point Geometry Type	
Type	Delta X and Delta Y
Point Geometry Properties	
From Point	Origin
Delta X	0
Delta Y	0

b) **Output Parameter**: Drag and drop a **Set Output Parameter** element to below P1. Define the output parameter as follows:

General	
Output Parameter	TaperValue
Value	$\text{if}(\text{SA.islayout}=\text{true}, \text{StartValue}, \text{StartValue} + \frac{(\text{Baseline.Station} - \text{Baseline.RegionStart})}{(\text{Baseline.RegionEnd} - \text{Baseline.RegionStart})} * (\text{EndValue} - \text{StartValue}))$

The Value is: $\text{if}(\text{SA.islayout}=\text{true}, \text{StartValue}, \text{StartValue} + \frac{(\text{Baseline.Station} - \text{Baseline.RegionStart})}{(\text{Baseline.RegionEnd} - \text{Baseline.RegionStart})} * (\text{EndValue} - \text{StartValue}))$

To use this subassembly place it in advance of the subassembly component that has an input parameter that you would like to taper and then in the Assembly Properties window in Civil 3D examine the Construction tab. For the parameter that you want to taper select the checkbox under Use Parameter Reference and then using the dropdown list in the Get Value column select the Taper Value output variable.



Appendix: VB Expressions and API Functions

VB Expressions: Math

Emphasized values can be changed to reference the applicable value.

Math VB Expression	Output	Description
math.round(2.568,2)	2.57	Returns a value rounded to the nearest specified decimal places (ex. -2 = <i>hundreds</i> , -1 = <i>tens</i> , 0 = <i>whole number</i> , 1 = <i>tenths</i> , 2 = <i>hundredths</i> , etc.)
math.floor(2.568)	2	Returns the largest integer that is less than or equal to the specified value (i.e. rounds down)
math.ceiling(2.568)	3	Returns the smallest integer that is greater than or equal to the specified value (i.e. rounds up)
math.max(2.568,0.813)	2.568	Returns the larger of two specified values
math.min(2.568,0.813)	0.813	Returns the smaller of two specified values
math.abs(-2.568)	2.568	Returns the absolute value
math.pi	3.14159...	Returns the value of the constant pi
math.e	2.71828...	Returns the value of the constant e
math.sin(math.pi)	0	Returns the sine of a specified angle measured in radians
math.cos(math.pi)	-1	Returns the cosine of a specified angle measured in radians
math.tan(math.pi)	0	Returns the tangent of a specified angle measured in radians
math.asin(1)	1.57079...	Returns the angle measured in radians whose sine is the specified value
math.acos(1)	0	Returns the angle measured in radians whose cosine is the specified value
math.atan(1)	0.78539...	Returns the angle measured in radians whose tangent is the specified value
math.log(math.e)	1	Returns the natural (base e) logarithm of a specified value
math.log10(10)	1	Returns the base 10 logarithm of a specified value
math.exp(1)	2.71828...	Returns e raised to the specified power
math.pow(2,3)	8	Returns a value raised to the specified power
math.sqrt(81)	9	Returns the square root of a specified value
math.iieee remainder(7,2)	1	Returns the remainder of the first value divided by the second value
math.sign(-2.1)	-1	Returns an integer indicating the sign of the number

VB Expressions: Casting

Emphasized values can be changed to reference the applicable value.

Ctype VB Expression	Output	Description
CType(10%,double)	0.10	Converts the first value into the specified variable type (integer, double, string)

VB Expressions: Logic

Emphasized values can be changed to reference the applicable value.

Logic VB Expression	Description
IF(P1.Y>P2.Y,2,3)	Used in a VB Expression, returns a value depending on whether the condition ($P1.Y > P2.Y$) is true (<i>value of 2</i>) or false (<i>value of 3</i>)
$P1.Y > P2.Y$	Returns true if $P1.Y$ is greater than $P2.Y$
$P1.Y \geq P2.Y$	Returns true if $P1.Y$ is greater than or equal to $P2.Y$
$P1.Y < P2.Y$	Returns true if $P1.Y$ is less than $P2.Y$
$P1.Y \leq P2.Y$	Returns true if $P1.Y$ is less than or equal to $P2.Y$
$P1.Y = P2.Y$	Returns true if $P1.Y$ is equal to $P2.Y$
$P1.Y \neq P2.Y$	Returns true if $P1.Y$ is not equal to $P2.Y$
$(P1.Y > P2.Y) \text{AND} (P2.X > P3.X)$	Returns true if both the condition ($P1.Y > P2.Y$) AND the condition ($P2.X > P3.X$) are true
$(P1.Y > P2.Y) \text{OR} (P2.X > P3.X)$	Returns true as long as either the condition ($P1.Y > P2.Y$) OR the condition ($P2.X > P3.X$) is true
$(P1.Y > P2.Y) \text{XOR} (P2.X > P3.X)$	Returns true if only one of the two conditions ($P1.Y > P2.Y$), ($P2.X > P3.X$) is true (if both are true or both are false, then false is returned)

VB Expressions: Subassembly Composer Application Programming Interface (API) Functions

Emphasized values can be changed to reference the applicable element.

Points and Auxiliary Points Class

Point API Function	Description
$P1.X$	Horizontal distance from point $P1$ to Origin
$P1.Y$	Vertical distance from point $P1$ to Origin
$P1.Offset$	Horizontal distance from point $P1$ to assembly baseline
$P1.Elevation$	Elevation of point $P1$ relative to 0
$P1.DistanceTo("P2")$	Distance from point $P1$ to point $P2$ (<i>Always positive</i>)
$P1.SlopeTo("P2")$	Slope from point $P1$ to point $P2$ (Upward = positive, Downward = Negative)
$P1.IsValid$	Point $P1$ assigned & valid to use (<i>T/F</i>)
$P1.DistanceToSurface(SurfaceTarget)$	Vertical distance from point $P1$ to <i>SurfaceTarget</i> (point above = positive, point below = negative)

Links and Auxiliary Links Class

Link API Function	Description
$L1.Slope$	Slope of link $L1$
$L1.Length$	Length of link $L1$ (<i>Always positive</i>)
$L1.Xlength$	Horizontal distance between start and end of link $L1$ (<i>Always positive</i>)
$L1.Ylength$	Vertical distance between start and end of link $L1$ (<i>Always positive</i>)
$L1.StartPoint$	A point located at the start of link $L1$ (Can be used in API Functions for P1 Class)

Link API Function	Description
<i>L1.EndPoint</i>	A point located at the end of link <i>L1</i> (Can be used in API Functions for P1 Class)
<i>L1.MaxY</i>	Maximum Y elevation from a link's points
<i>L1.MinY</i>	Get the minimum Y elevation from a link's points
<i>L1.MaxInterceptY(slope)</i>	Apply the highest intercept of a given link's points to the start of another link
<i>L1.MinInterceptY(slope)</i>	Apply the lowest intercept of a given link's points to the start of another link
<i>L1.LinearRegressionSlope</i>	Slope calculated as a linear regression on the points in a link to find the best fit slope between all of them
<i>L1.LinearRegressionInterceptY</i>	The Y value of the linear regression link
<i>L1.IsValid</i>	Link <i>L1</i> is assigned & valid to use (T/F)
<i>L1.HasIntersection("L2")</i> <i>L1.HasIntersection("L2", true, true)</i>	<i>L1</i> and <i>L2</i> have an intersection, second input is a Boolean defining whether to extend <i>L1</i> with default of false, third input is a boolean defining whether to extend <i>L2</i> with default of false (T/F)

Offset Target Class

Offset API Function	Description
<i>OffsetTarget.IsValid</i>	<i>OffsetTarget</i> is assigned & valid to use (T/F)
<i>OffsetTarget.Offset</i>	Horizontal distance from <i>OffsetTarget</i> to assembly baseline

Elevation Target Class

Elevation API Function	Description
<i>ElevationTarget.IsValid</i>	<i>ElevationTarget</i> is assigned & valid to use (T/F)
<i>ElevationTarget.Elevation</i>	Vertical distance from <i>ElevationTarget</i> to assembly baseline

Surface Target Class

Offset API Function	Description
<i>SurfaceTarget.IsValid</i>	<i>SurfaceTarget</i> is assigned & valid to use (T/F)

Superelevation Class

Superelevation API Function	Description
<i>SE.HasLeftLI</i>	Left lane inside superelevation slope is present & valid to use (T/F)
<i>SE.HasLeftLO</i>	Left lane outside superelevation slope is present & valid to use (True/False)
<i>SE.HasLeftSI</i>	Left shoulder inside superelevation slope is present & valid to use (T/F)
<i>SE.HasLeftSO</i>	Left shoulder outside superelevation slope is present & valid to use (T/F)
<i>SE.HasRightLI</i>	Right lane inside superelevation slope is present & valid to use (T/F)
<i>SE.HasRightLO</i>	Right lane outside superelevation slope is present & valid to use (T/F)

Superelevation API Function	Description
SE.HasRightSI	Right shoulder inside superelevation slope is present & valid to use (T/F)
SE.HasRightSO	Right shoulder outside superelevation slope is present & valid to use (T/F)
SE.LeftLI	Left lane inside superelevation slope
SE.LeftLO	Left lane outside superelevation slope
SE.LeftSI	Left shoulder inside superelevation slope
SE.LeftSO	Left shoulder outside superelevation slope
SE.RightLI	Right lane inside superelevation slope
SE.RightLO	Right lane outside superelevation slope
SE.RightSI	Right shoulder inside superelevation slope
SE.RightSO	Right shoulder outside superelevation slope

Baseline Class (*Note assembly baseline may or may not be the subassembly origin)

Baseline API Function	Description
Baseline.Station	Station on assembly baseline
Baseline.Elevation	Elevation on assembly baseline
Baseline.RegionStart	Station at the start of the current corridor region
Baseline.RegionEnd	Station at the end of the current corridor region
Baseline.Grade	Grade of assembly baseline
Baseline.TurnDirection	Turn direction of assembly baseline (<i>Left</i> = -1, <i>Non-curve</i> = 0, <i>Right</i> = 1)

EnumerationType Class

Enumeration API Function	Description
<i>EnumerationType.Value</i>	The string value of the current enumeration item

Subassembly Class

Subassembly API Function	Description
SA.IsLayout	Current preview mode is Layout Mode (T/F)

Cant Class

Cant API Function	Description
Cant.PivotType	Pivot method assigned to the current curve: Low Side Rail (left rail) = -1 Center Baseline = 0 High Side Rail (right rail) = 1
Cant.LeftRailDeltaElevation	Differential elevation for the left rail
Cant.RightRailDeltaElevation	Differential elevation for the right rail
Cant.TrackWidth	Track Width assigned to the alignment
Cant.IsDefined	Cant has been calculated on the alignment (T/F)