CES501594

Automation opportunities hiding inside Civil3D since the early 2000's

Ian Philpott – BIM Development Lead
Atkins / SNC Lavalin

Learning Objectives

- Understand the automation opportunities hiding within Civil3D
- Understand the changes needed to your subassemblies to leverage them for automation
- How to modify your code set styles to speed up drawing output and annotation
- How to use subassemblies and code sets to drive data deep into your quantification process

Description

You say the word automation, immediately people start talking about Dynamo or API coding. What if I said there are a host of automation opportunities hiding inside Civil3D and most users only scratch the surface of what is right at their fingertips. This class looks at these opportunities to automate repetitive tasks, by leveraging code set styles to speed up drawing outputs and quantification of corridor materials

Speaker

An Engineer with 25+ years experience in the highways and linear infrastructure realm. I specialise in the development, enablement and support of digital workflows across a wide variety of software platforms and technologies. These areas include modelling applications such as Civil3D and Infraworks, Common Data Environment configuration and full asset lifecycle data management. One of my specialist areas is Civil3D. I have been using and supporting this software since 2010 and have trained numerous users. I have also presented at Autodesk University on 5 occasions and have attended the by invitation only Inside the Factory events with the Civil3D development team. I continue to provide feedback through the Civil3D Feedback Community.

Introduction

The word “Automation” always triggers many people to think Dynamo or using the API and one of many coding languages. I’m not saying Dynamo and coding won’t give good solutions but what if there were automation opportunities that are hiding in plain sight?

So, why do we want to automate?

- Reduce the time taken to perform a task
- Ensuring output achieves requirements
- Improve consistency
- Minimise the human inconsistency variables
- Remove risks through connections between the 3D models and 2D drawings
- Mitigate impact and risks of design changes

Throughout this guide we will refer to different types of models so below is a quick reference guide to them

- **Design Model** – Source drawing used for undertaking the design. Civil3D objects stylised to aid design process. Where Civil3D objects can be datashortcutted they would not be consumed from the design models into the drawings.
- **Presentation Model** – Consuming models that datashortcut in relevant Civil3D objects and are stylised for drawing output and used as an xref into drawings.
- **Co-ordination Model** – Consuming models that datashortcut in relevant Civil3D objects and are stylised to support the production of a federated model.
- **Setting Out Model** – What is says on the tin. Consumes Civil3D objects and stylised for setting out needs, whether the information is to be presented in drawings or for export to machine control or instruments.

Learning Objective 1 – Understanding the automation opportunities hiding within Civil3D

Before we can start the process of automating, we need to have a solid understanding of what we are trying to achieve, only then can we start to look at what the tools inside Civil3D will allow us to automate. So, what are the possible target areas?

- Stylise model output to improve consistency and ease drawing output, moving from a manual annotation or hatching process to an automated solution.

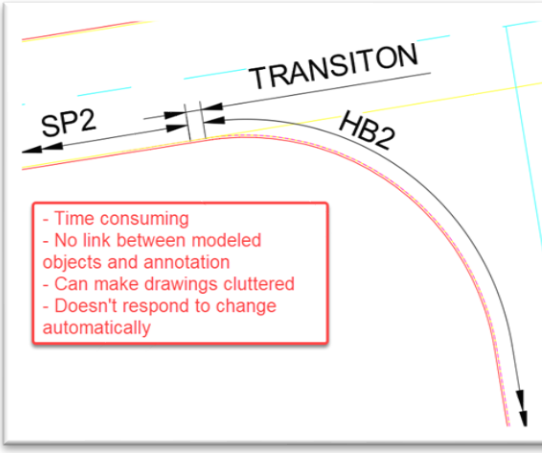


Figure 1- Manual Kerb Labelling

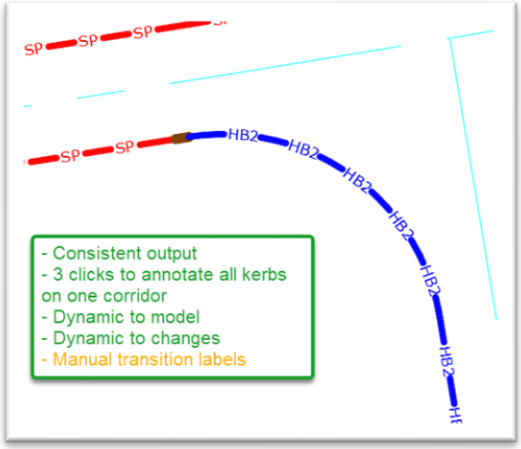


Figure 2 - Automated Kerb Labelling

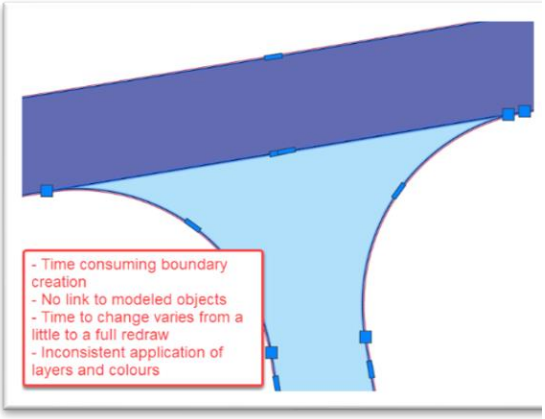


Figure 3 - Manual Pavement Type Hatching

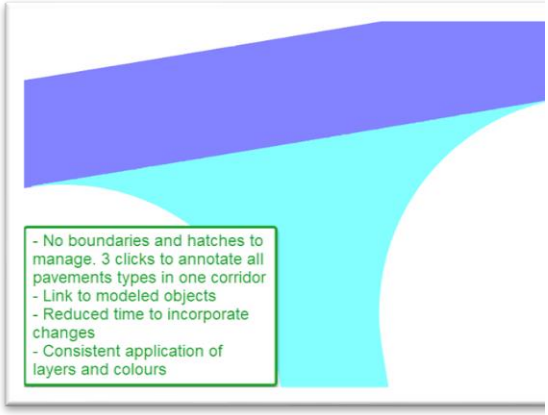


Figure 4 - Automated Pavement Type Hatching

- Embed data that can be used in downstream uses such as
  - Quantification
  - Construction sequencing
  - Asset Management

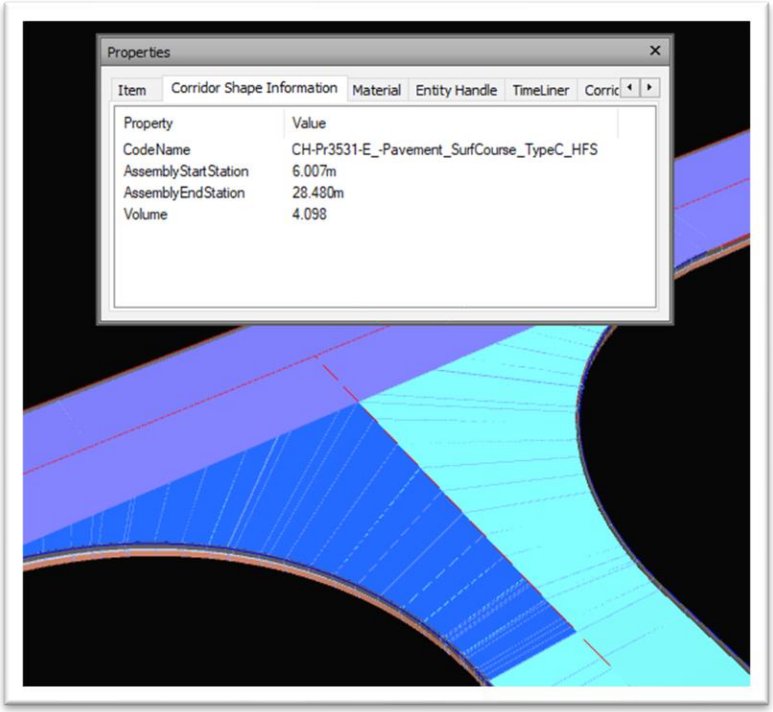


Figure 5 - Embedded Data

- Design rationalisation to GIS – Set to display only the linework to be pushed to GIS

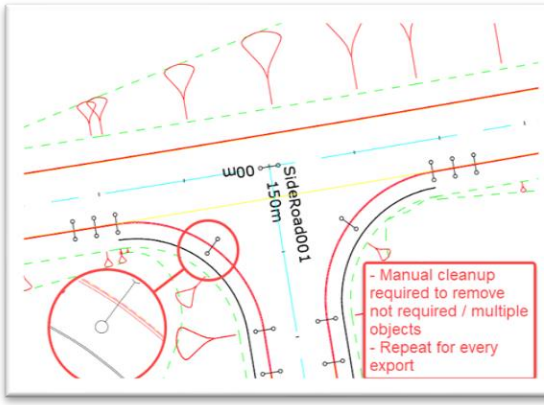


Figure 6 - Manual GIS Export

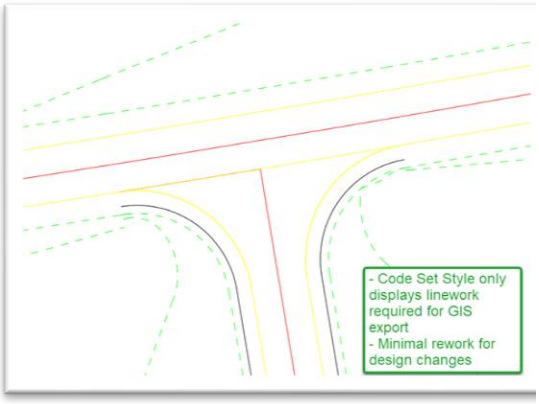
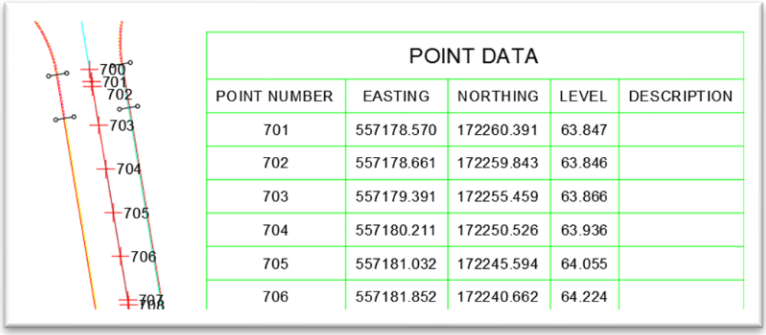


Figure 7 - Code Set Style Controlled Export

- Setting out - Set to display only the linework needed to deliver the setting out strings.



POINT DATA

POINT NUMBER	EASTING	NORTHING	LEVEL	DESCRIPTION
701	557178.570	172260.391	63.847	
702	557178.661	172259.843	63.846	
703	557179.391	172255.459	63.866	
704	557180.211	172250.526	63.936	
705	557181.032	172245.594	64.055	
706	557181.852	172240.662	64.224	

Figure 8 - Control Setting Out Strings Via Code Set Styles

What else do you need?

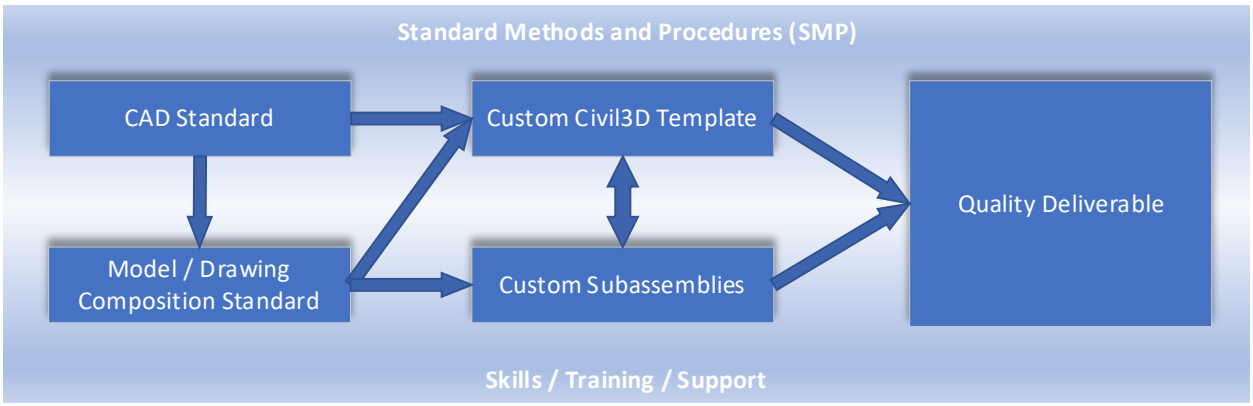


Figure 9 - Core Components of Success

So what does that mean in the real world?

1. Management Support – **YOU NEED THAT**. You are about to make big changes to the way people work and some won't like it. People change is the biggest challenges we face in the transformation programmes.
2. Time – **YES, PLENTY.** These improvements need a methodical approach and testing programme. These take time to do right.
3. CAD & Model / Drawing Composition Standards – These define some of the outputs you are trying to achieve.
4. Civil3D Template – Alot of customisation is needed. Out of the box (OTB) won't give you the benefits.
5. Custom subassemblies – Out of the box subassemblies are not configured to support these processes. This class will give you the skills to manage the point, link and shape codes but you will need someone who can build the geometric behaviours you require.
6. Standard Methods and Procedures, skills, training and support – You have spent time and effort developing these time saving tools and processes. If the team don't adopt them then all the time spent is wasted. Guidance and support during deployment and use is essential to success.

Learning Objective 2 - Understand the changes needed to your subassemblies to leverage them for automation

Before we start looking at the changes to subassemblies there are some important fundamentals of subassemblies and how they work with Civil3D that need to be understood, and that's points, links and shapes

What are points, links and shapes?

Points, links and shapes are the fundamental building blocks of Civil3D subassemblies.

Point

A primary node that controls the geometric shape of a subassembly (Subassembly Composer Help - Point)

Link

A line that joins 2 points (Subassembly Composer Help - Link)

Shape

A closed polygon formed from a number of links (Subassembly Composer Help - Shape)

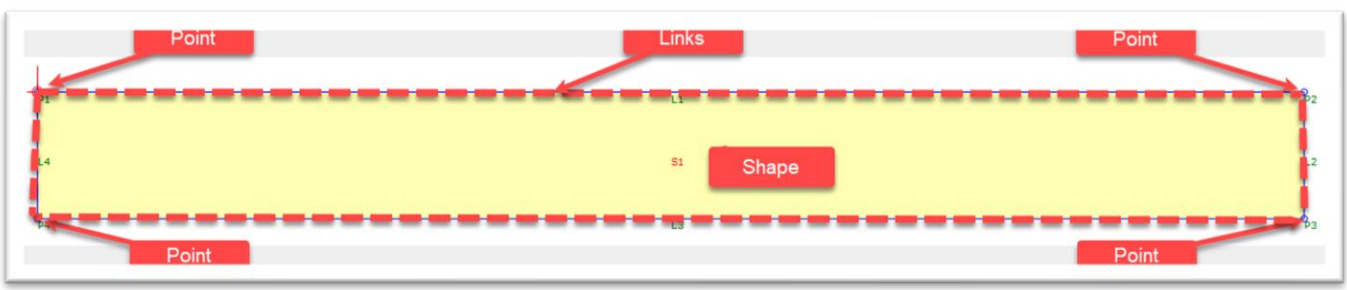


Figure 10 - What are Points, Links and Shapes

What are point, link and shape codes?

Each point, link or shape in your subassembly can have no code applied or from one to multiple codes depending on how you want to use them and how granular you want to get. Each point, link or shape code gives you the ability to leverage the code to create objects or stylise the way things look in plan, profile, cross section, assembly or extracted solid.

Each code type has a variety of uses within Civil3D Code Sets (See Figure 29). For this class we are focusing on the uses shown in **bold** below

- Point codes – Point markers and point labels in subassemblies and cross sections, **feature lines in corridors,** adding specific breaklines in surfaces.

- Link codes – Line styles in subassemblies and cross sections, link labels in subassemblies, profiles and cross sections, render materials, display of frequency insertions in corridors, **hatching of the corridor.**
- Shape codes – Shape labels in subassemblies and cross sections, **controlling extracted solids**.

The next step is to map outputs required in Civil3D back to the subassemblies. For this class we are going to focus on a kerb (Figure 11) and a carriageway element (Figure 12). Once you have mastered the techniques for kerbs these principles can be applied to other long thin linear elements such as fences, edgings, linear channels, barriers etc. Similarly, principles from the carriageway element can be applied to footpaths/footways, verges and earthworks.

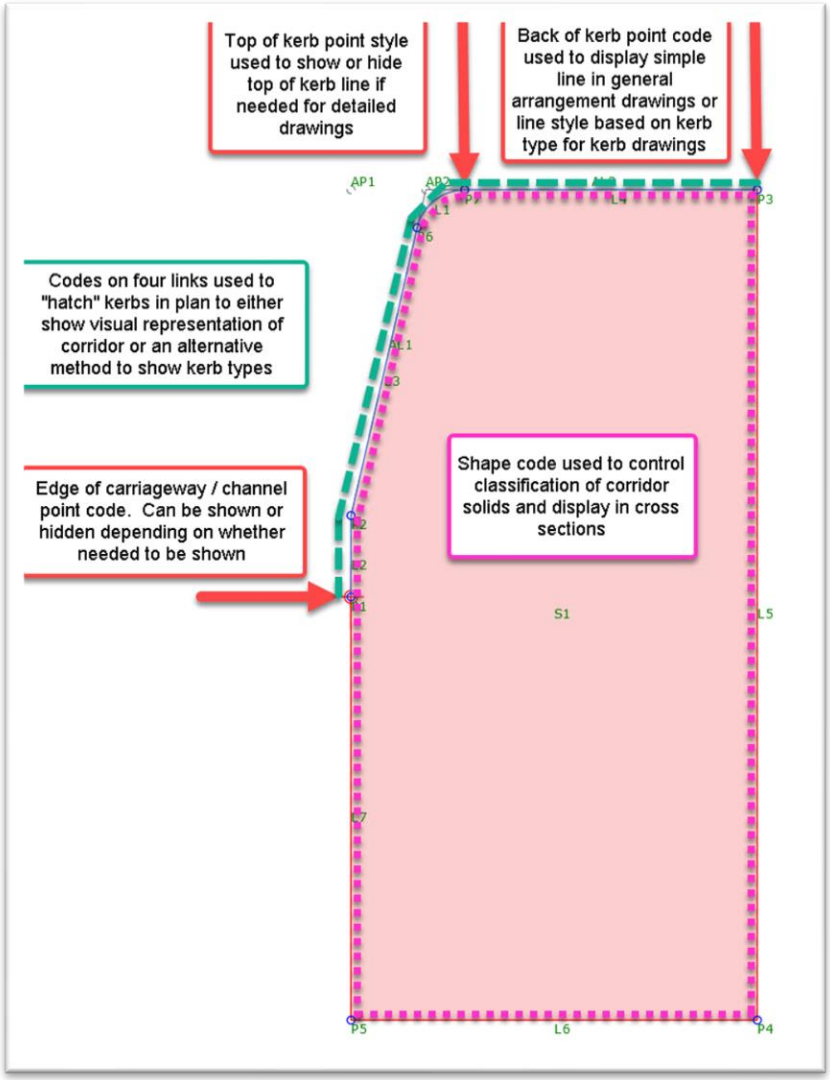


Figure 11 - Code behaviour for kerbs

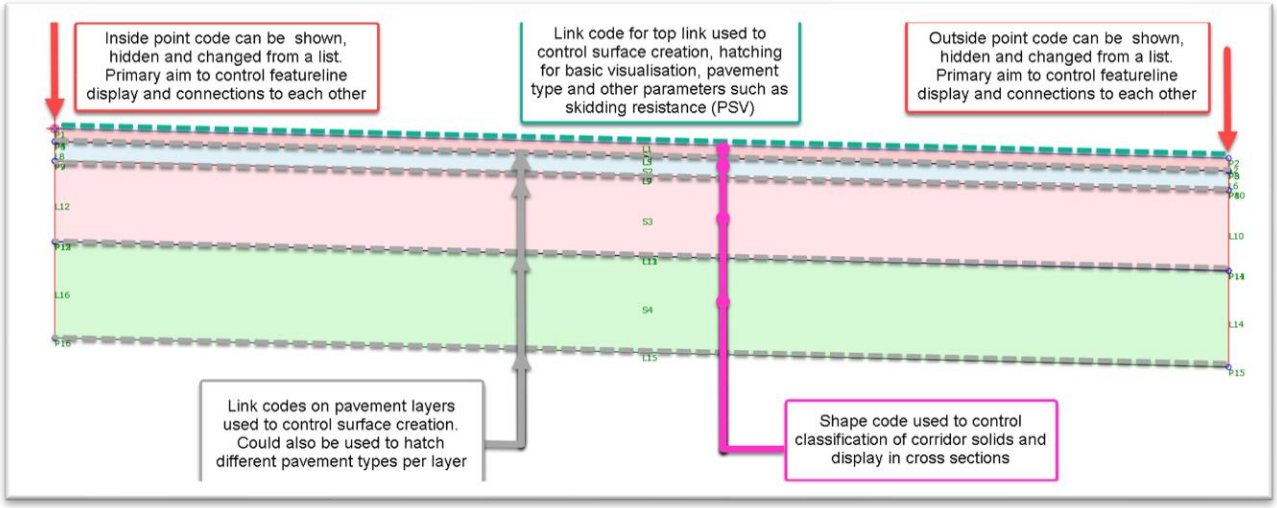


Figure 12 - Code behaviour for carriageway elements

Next, we need to plan the changes needed to the subassemblies. But before we look at the actual changes to our two sample subassemblies we need to look at some techniques in subassembly composer to allow us to drive this data deep in to our objects but also make it easy for us to manage.

We are not going to teach you how to build a subassembly from scratch or how to define complex geometric behaviours. There are lots of resources out there, some of the ones I've found useful are listed in the Reference Section at the end of this document.

The kerb example provided in IP_Kerb_Example_P04.pkt is based on UK standard kerb shapes and dimensions. It has also been simplified with just enough geometric functionality to enable a demonstration of how the point, link and shape codes change depending on certain parameter changes.

The kerb subassembly has three types (other kerb options omitted and replaced with error messages)

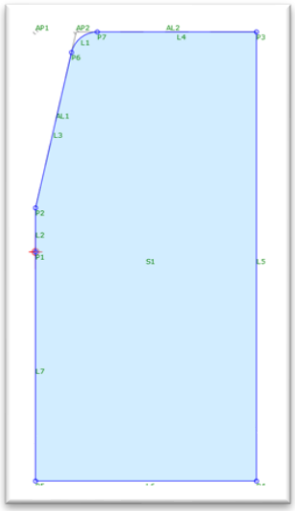


Figure 13 - HB2 Kerb

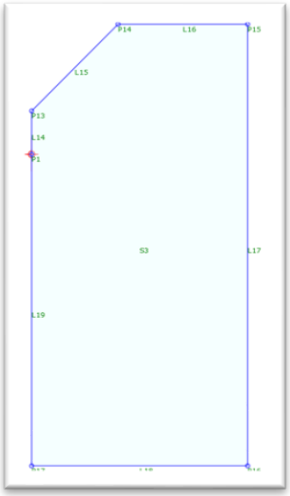


Figure 14 - SP Kerb

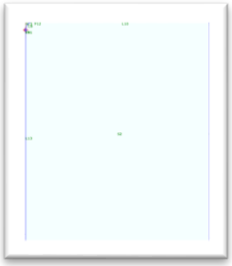


Figure 15 - BN Kerb

The kerb selection is controlled by two parameters:, KerbType and KerbUpstand.

Input/Output Parameters

Name	Type	Direction	Default Value	DisplayName	Description
Side	Side	Input	Right		
FrontPoint	En_PointFront	Input	TOK	Top of Kerb Point	Sets top of kerb p
KerbType	EnKerbType	Input	HB2	Kerb Type	Sets kerb type
KerbUpstand	EnKerbUpstanc	Input	Us125mm	Kerb Upstand	Sets kerb upstand
CustomUpstand	Double	Input	0.05	Custom Upstand	Sets height of cus
BackPoint	En_PointBack	Input	BOK	Back of Kerb Point	Sets back of kerb
ChannelPoint	En_PointChann	Input	Channel	Channel line Point	Sets channel line p
KerbShape	EnShapeKerb	Input	Pr2593	Uniclass Code	Sets Uniclass code

Figure 16 - Kerb Parameters

Both of these parameters are controlled by enumeration lists.

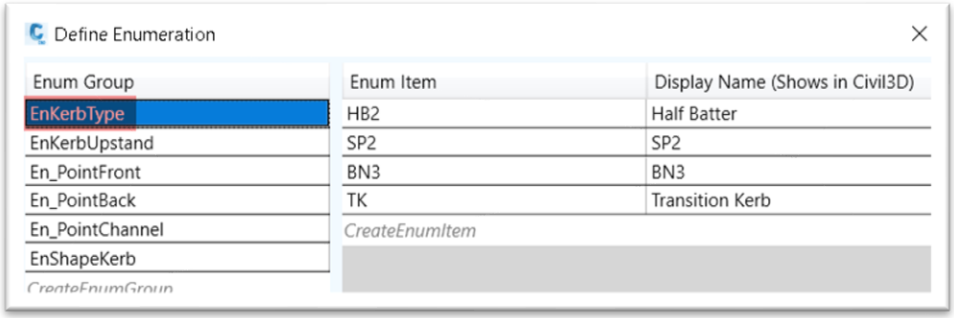


Figure 17 - Kerb Type Ennumeration List

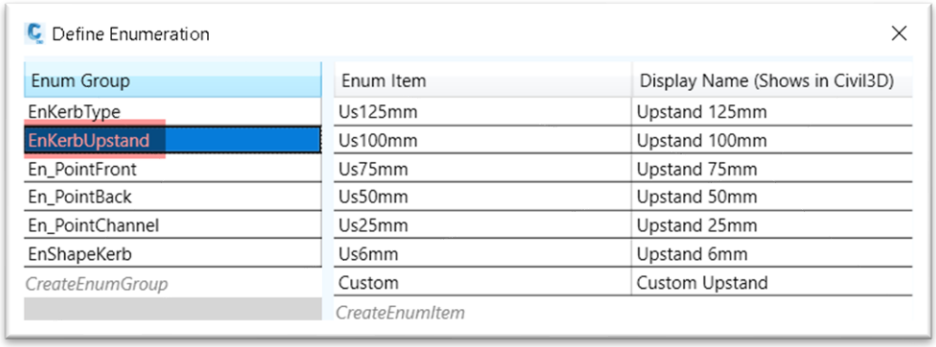


Figure 18 - Kerb Upstand Enumeration List

In Figure 17 you can see the kerb types. This drives both a decision in the graphical construction of the subassembly and is used as part of the code naming process. In Figure 18, the enumeration list for the kerb upstand control is displayed. The geometric construction of the kerb the upstand (channel line to top of kerb) needs to be a numeric value. However, enumeration items cannot start with a numeric value therefore can't control the upstand directly. We can work round this by using the process below:

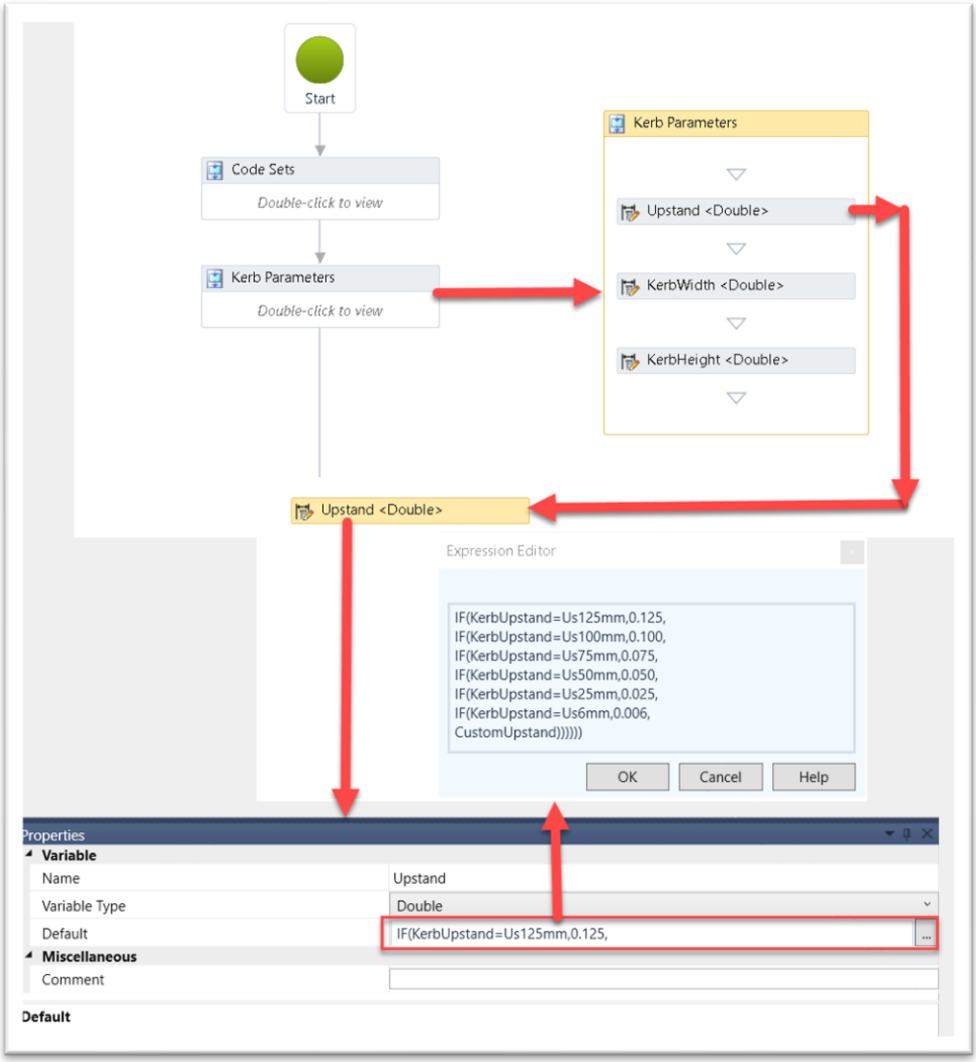


Figure 19 – Enumeration list to numeric value

Figure 19 is a pasted series of images that show the creation of sequence to house various kerb parameters. The defined parameter for the kerb upstand uses an “IF” statement to translate the enumeration list, which as stated above can’t only be numeric values, from an alphanumeric value to a numeric value.

The point code being used to control the kerb type annotation automation is the back of kerb point as shown in Figure 20.

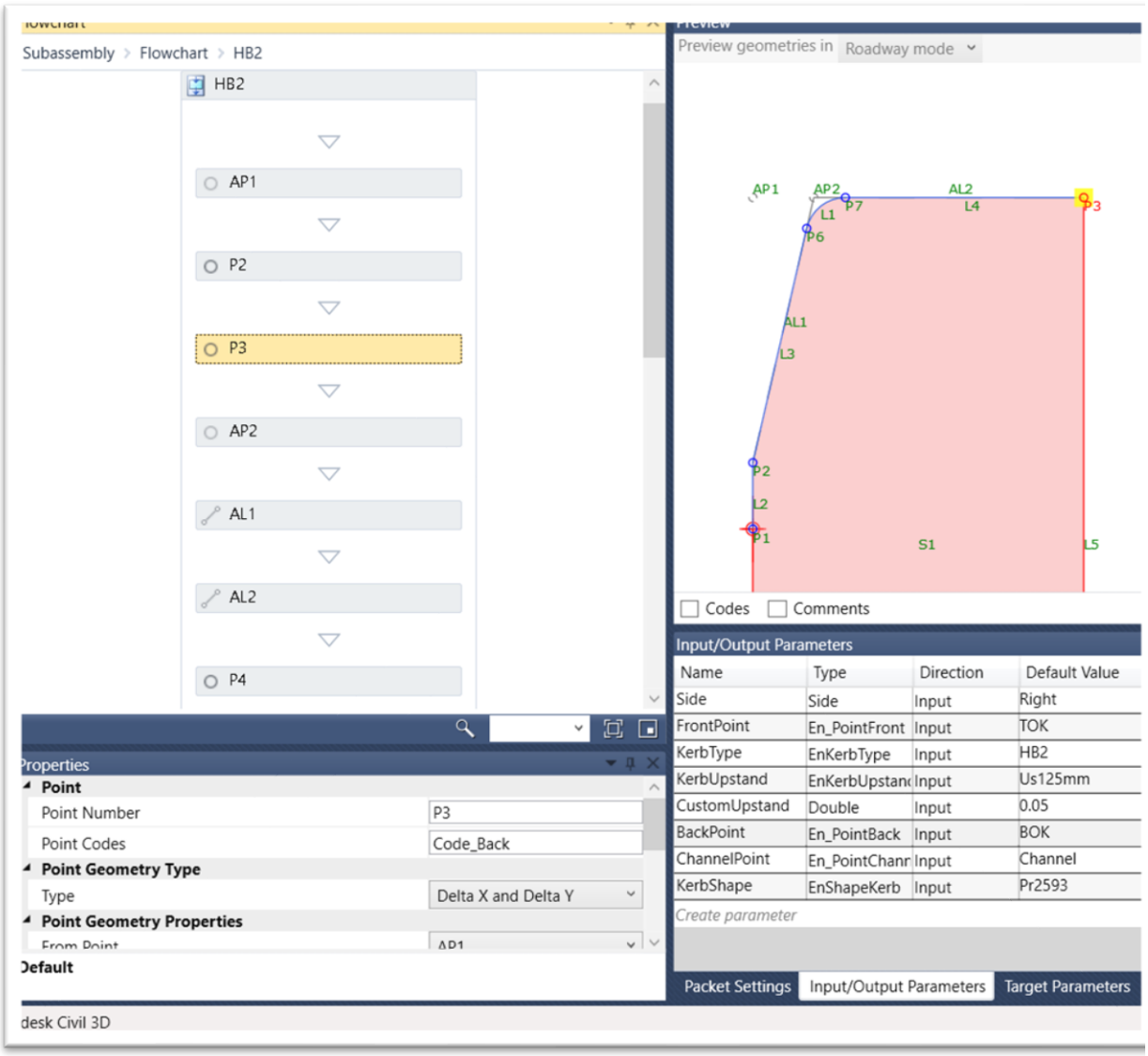


Figure 20 - Back of Kerb Point Defined Variable

Each variant of the kerb geometry (See Figure 13, Figure 14 and Figure 15) uses the same defined variable of “Code_Back” (See Figure 20) for the back of kerb point code. This enables codes to be managed in the “Code Sets” sequence rather than having to make edits in each geometric variant if changes are needed,

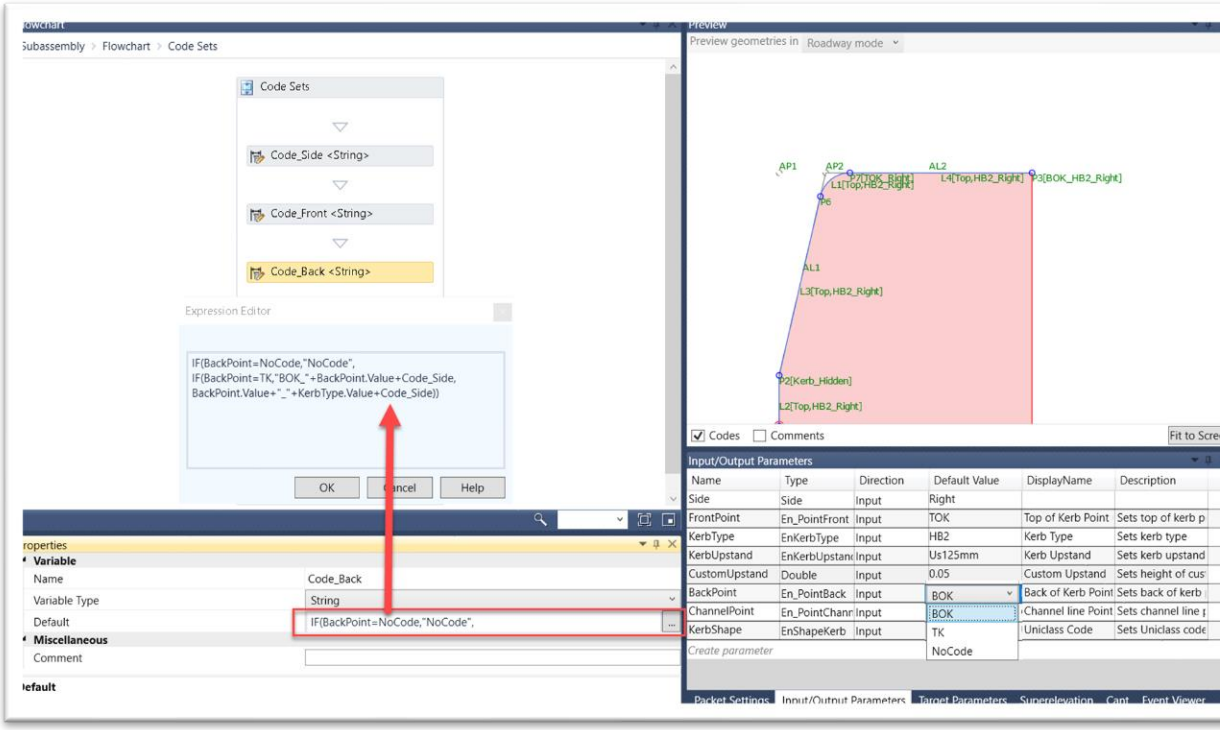


Figure 21 - Code_Back Expression

If we delve into the expression, we start to see the intelligence being derived to drive the point code changes.

IF(BackPoint=NoCode,"NoCode",

This section of the expression allows the back of kerb point code to be switched off if needed either at a subassembly level or using the overrides in the corridor section editor. I've used the "*BackPoint*" parameter with the "*NoCode*" option and set that as "No Display" in the code set styles but equally replacing "*NoCode*" with " " to blank out the value would also work.

IF(BackPoint=TK,"BOK_"+BackPoint.Value+Code_Side,

This section of the expression controls the point code if the "*BackPoint*" parameter is changed to "*TK*" (for transition kerb). This allows the point code to be overridden in the corridor section editor to annotate the kerb transition without changing the geometric kerb shape. So in practice this works as follows

If the "*BackPoint*" parameter is set to TK then the Back Point Code (*Code_Back*) will be formed from

BOK (Static value) plus

the value to the "BackPoint" parameter (ParameterName.Value extracts that value) plus

Code_Side which extracts the side parameter

Example BOK_TK_Left

The final section of the expression is the default case. This uses the value of the *BackPoint* parameter and adds the value of the *KerbType* parameter and the side code.

BackPoint.Value+"_"+KerbType.Value+Code_Side))

Example BOK_HB2_Left

Looking at the link codes, focusing codes used for the links between the back of kerb point to the channel line point (See green line on Figure 11)

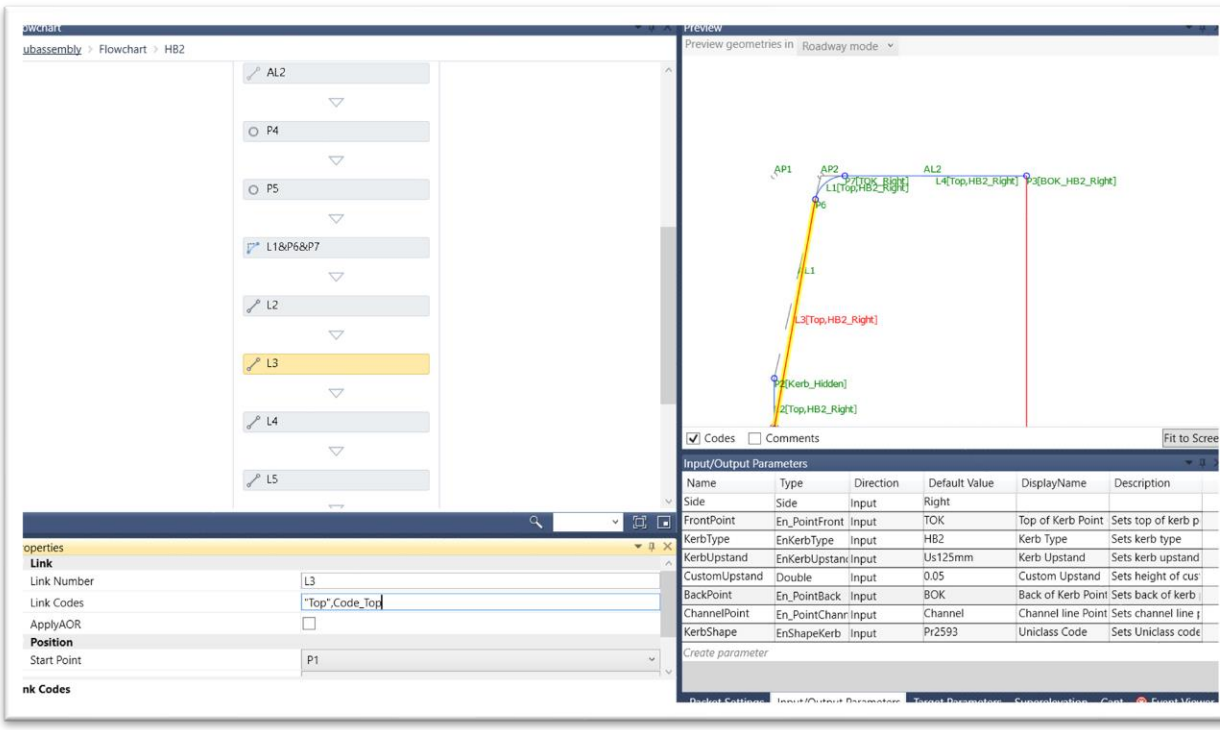


Figure 22 - Kerb Link Codes

As shown in Figure 22 the link code has two codes. The code “Top” is hard coded so that when a corridor surface is being created the top surface can be created by selecting “Top Links” in the corridor surface creation dialogue. “Code_Top” is the variable that creates the dynamic element of the link code.

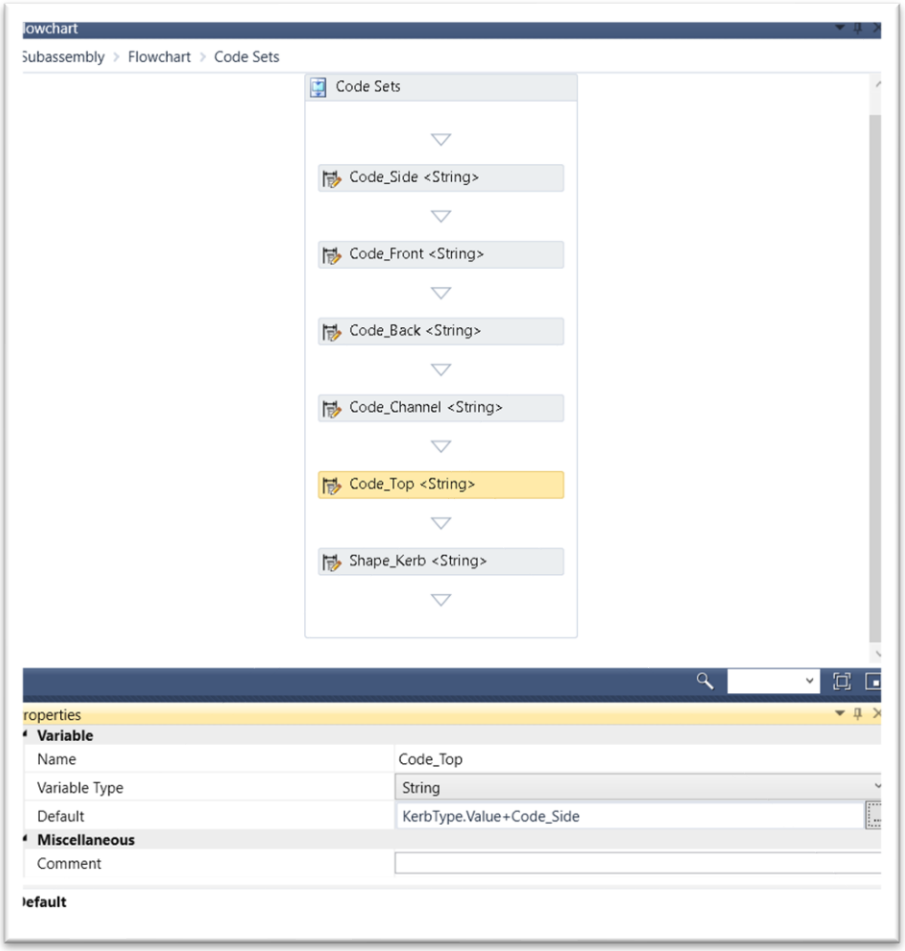


Figure 23 - Top Link Code

In a very similar way to the back of kerb point code, the "*Code_Top*" grabs the "*KerbType*" parameter value and adds the "*Code_Side*" parameter which reports the value from the "*Side*" parameter.

The shape code has been built up from a layer name to meet the requirements of the UK standard BSENISO 13567 Part 2 (as shown below). This also picks up dynamic elements, as used in the point and link codes, these change the various parameters as the subassembly change.

Discipline-Classification-Presentation-Description

So why do this?

1. When extracting corridor solids in Civil3D no additional input is needed to determine the layer for each solid. The default can be set to “Codes” (See Figure 24) and then the layer names will always be correct as they are built in to this shape code.

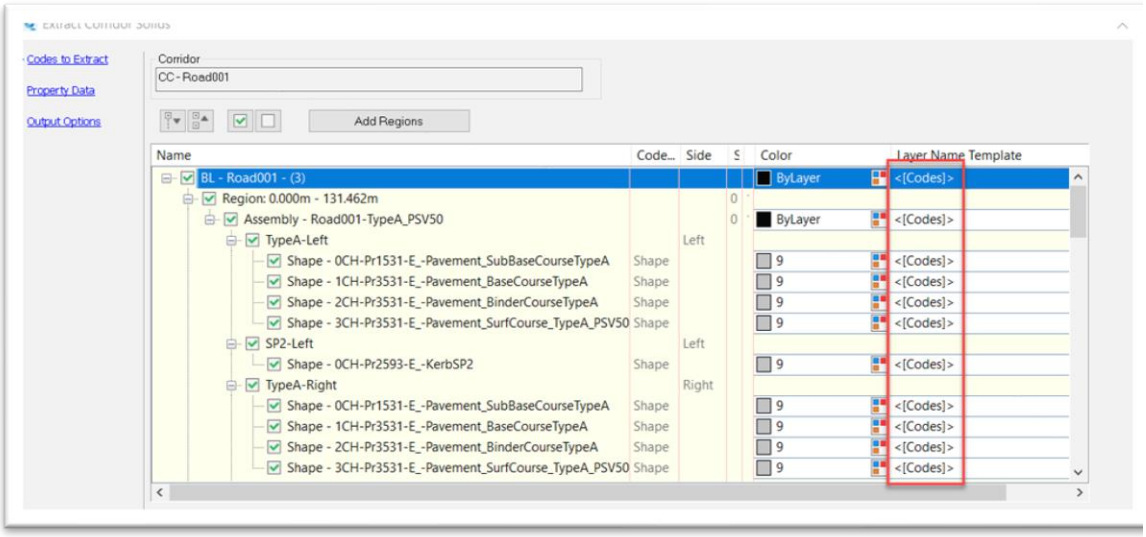


Figure 24 - Solids Extraction Layer Template based on "Codes"

2. Because the layer name is correct and more importantly consistent, it improves the flexibility in downstream uses such as
    a. Search sets in Navisworks
    b. Appearance profiler in Navisworks
    c. Construction sequencing
    d. Quantity extraction
    e. Application of asset data

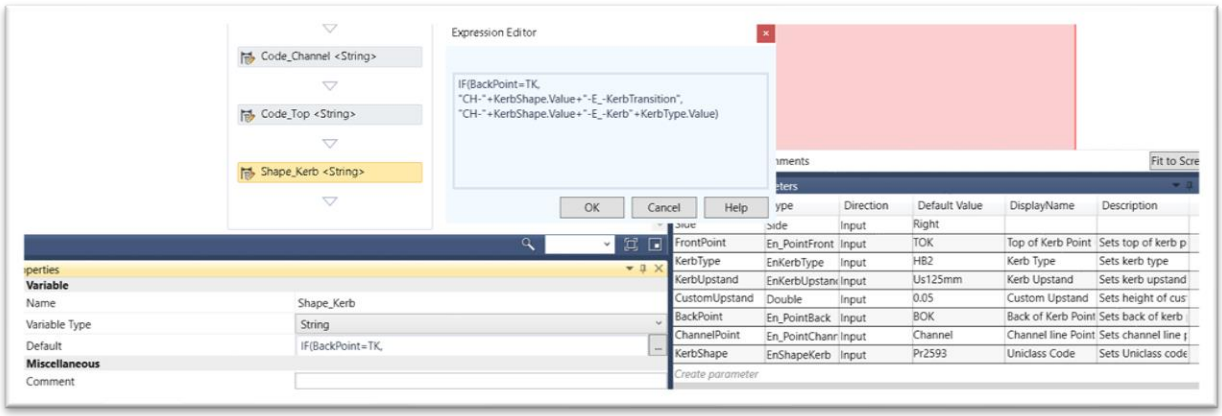


Figure 25 - Kerb Shape Code

Expression

IF(BackPoint=TK,
"CH-"+KerbShape.Value+"-E_-KerbTransition",
"CH-"+KerbShape.Value+"-E_-Kerb"+KerbType.Value)

The expression uses similar principles to the point code, utilising an IF statement to set the code to one thing if the *BackPoint* code is set to *TK* for Transitions and a different configuration for all other cases.

In both cases *KerbShape.Value* is used to grab the classification value. In this example the enumeration list only has one value but I still use an enumeration list for consistency and the ease of adding additional classifications.

In all other cases except where a transition exists, *KerbType.Value* is used to embed the kerb type as part of the code.

The carriageway example provided in IP_Carriageway_Example_P02.pkt is a simple four layer pavement construction with just enough geometric functionality to enable a demonstration of how the point, link and shape codes change depending on certain parameter changes. This subassembly doesn't respond to superelevation or targets etc.

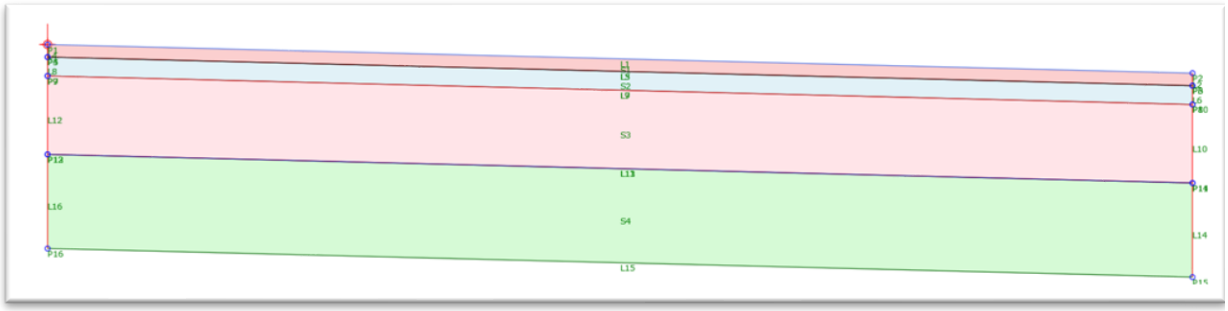


Figure 26 - Example Carriageway Subassembly

The point codes on the inside and outside edges of the pavement follow very similar principles to the kerb subassembly i.e.

- Enumeration lists for each that give a defined list of point name options for consistency.
- The "Code Sets" sequence creates attributes that are placed on the point codes so if changes are needed it can be done in the "Code Sets" sequence rather than having to find the points in the geometry creation.
- If the "*InsidePoint*" parameter is set to "NoCode" then an IF statement within "*Code_Inside*" uses the alternative method to that of the kerb to clear the point code; in other cases it takes the "*InsidePoint*" value and adds the value from the "*Side*" parameter (as shown below). The same happens with the "*OutsidePoint*" parameter

  IF(InsidePoint=NoCode," ",
  InsidePoint.Value+Code_Side)

- The point codes are used as part of the plan display, GIS output and setting out workflows for the purposed of only displaying the featurelines required

The link code strategy is slightly different depending on whether it is the top layer, bottom layer or the intermediate layers.

Top Layer

Figure 27 - Pavement Top Link Code

The top link of the top pavement layer has four link codes assigned to it. They are used as follows: -

- **Top** – This is a fixed code so that when a corridor surface is created comprising many components, a top surface can be created by selecting "*Top Links*" in the corridor surface creation dialogue and it will pick up the top links of many components in one go.
- **Pavement type e.g. Type A** – This is the part of the code that will be leveraged in the code set style to apply a *material area fill style* to hatch the pavement in different colours or patterns based on the pavement type parameter set in the subassembly.
- **Pavement friction value e.g. PSV50 or HFS** – This functions in the same way as the pavement type above, enabling a second hatch style to be used.
- **Pavement** - This is a fixed code that is used to make the "simple" visualisation hatching easy, allow all pavements to be hatched a single colour by setting the material area fill style against one code.

Intermediate Layers

The top link of each intermediate layer has two link codes assigned to it. They are used as follows: -

- **XX_Top e.g. BC_Top** – This is a fixed code where XX represents an abbreviated description of the pavement layer (See Figure 28). This allows a corridor surface for each layer to be created easily by selecting the relevant link code

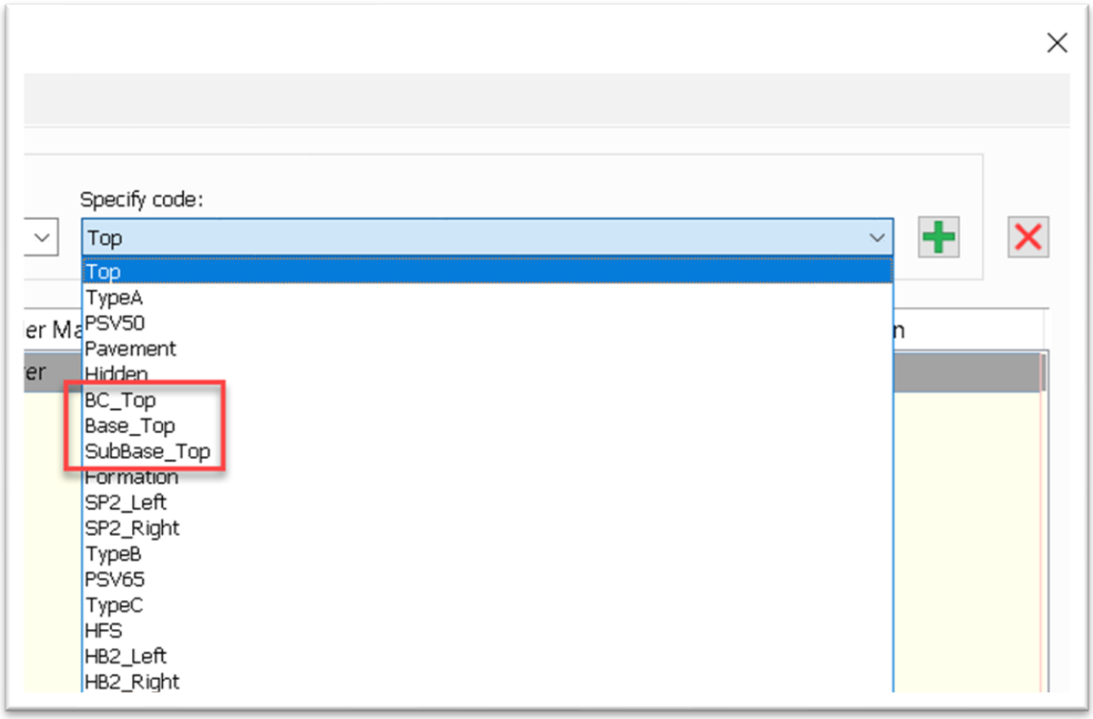


Figure 28 - Intermediate Link Code Names

- **Pavement type e.g. Type A** – In the same way as the top layer, this code could be used with the code set style to apply a *material area fill style* to hatch the pavement in different colours based on the pavement type parameter set in the subassembly. This would allow models to be produced hatching the different pavement types for each layer of the construction, not just the top layer.

Bottom Layer

The bottom layer of the pavement requires codes on both the top link and the bottom link. The top link of the layer is the same as the intermediate layers. The bottom link has a fixed value “Formation” to allow the corridor formation surface to be created.

Shape Code

The shape code is built up following the same principle as the kerb so that it forms a layer name for exactly the same reasoning.

The classification is picked up from four parameters, depending on the pavement layer allowing different material classifications for each layer.

Top Tips for Subassemblies

1. Insert a sequence at the top of your subassembly flowchart to house all of your codes for points, links and shapes
2. Use enumeration lists. Consistency is key for code sets to work, allowing users to manually enter codes doesn't give that consistency. Even if you only have one item in that list now, enumeration lists allow for rapid updates
3. Define variables in the "Code Set" sequence (See item 1) for all "moving parts" of the point, link and shape codes you want to automate. Insert these variables on the point, links and shape. This makes managing codes much easier as you don't need to hunt through the flowchart looking for the correct point, link or shape to modify the code
4. Learn how to use IF statements
5. Learn some key subassembly "data management" coding techniques such as
    a. VariableName.value – Use this to insert a value that's selected from a parameter
    b. Combining values from multiple sources with/without text strings
    c. How to apply multiple codes to a point, link or shape

Learning Objective 3 - How to modify your code set styles to speed up drawing output and annotation

Code set styles are one of the most underutilised automation opportunities in Civil3D. Couple these with corridor datashortcuts and using different presentation models, corridors can be stylised to be displayed for many different purposes while maintaining a dynamic link to the design corridor.

The key aim of these processes is to improve speed, quality and consistency of outputs. Therefore, a clear CAD Standard and Model / Drawing Composition Standard (See Figure 9) is required.

Style Development

This section is going to focus on 8 code set styles

- AU_Pres_Linework – Basic linework background used in most drawings
- AU_Pres_Visualisation – Basic hatching to improve the display of the corridors often used in consultation drawings
- AU_Pres_Kerbs – Kerbs stylised to annotate kerb types
- AU_Pres_Pavements – Pavements hatched based on pavement type
- AU_Pres_PavementsPSV - Pavements hatched based on pavement PSV (surface friction value)
- AU_Pres_Solids – Pavements styles set to improved output of corridor solids
- AU_Pres_SettingOut – Corridor linework set to display only the featurelines needed for setting out purposes
- AU_Pres_GIS – Corridor linework minimised to simplify export to GIS instead of manual editing in CAD or FME

In order develop the above Code Sets to then the styles within the groups highlighted in Figure 29 will need to be developed.

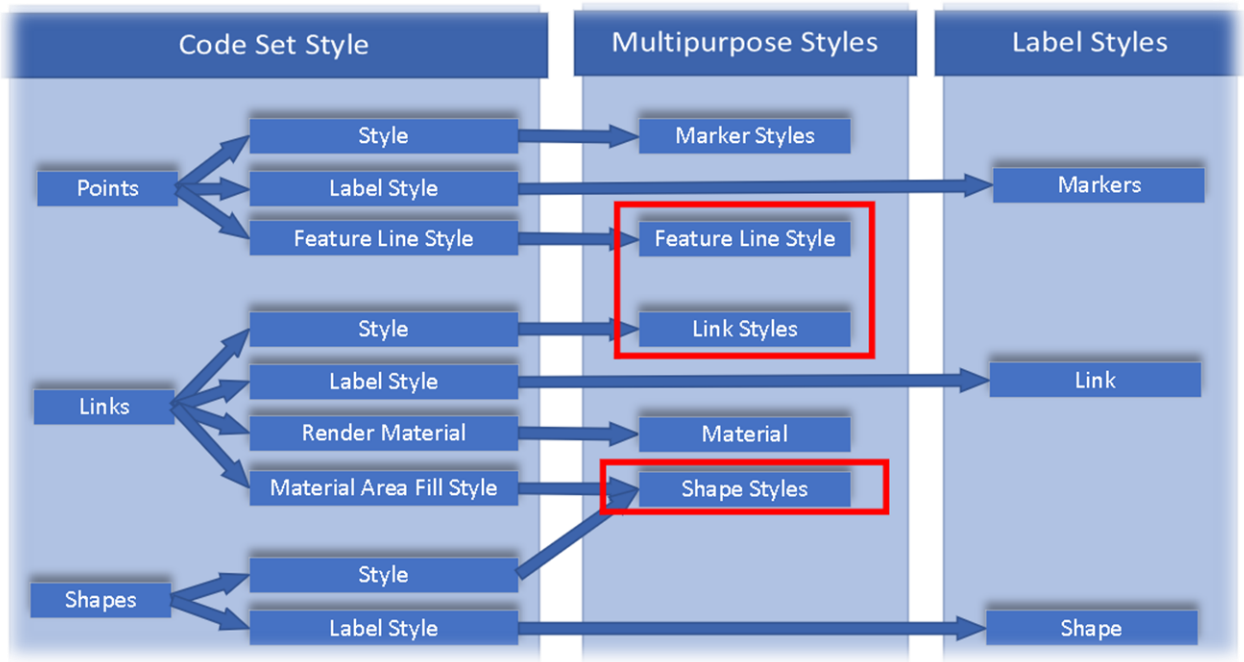


Figure 29 – Point, Link and Shape Style Relationships

Creating Code Set Styles – The Basics

Code Set Styles are found in Toolspace / Settings. Your template will probably have existing code sets in place so it's tempting to start with one of these and modify it. However, because the subassemblies now have custom point, link and shape codes it's better to start with the "New" option (As shown in Figure 30) so none of the existing codes are cluttering the code set with unnecessary codes.

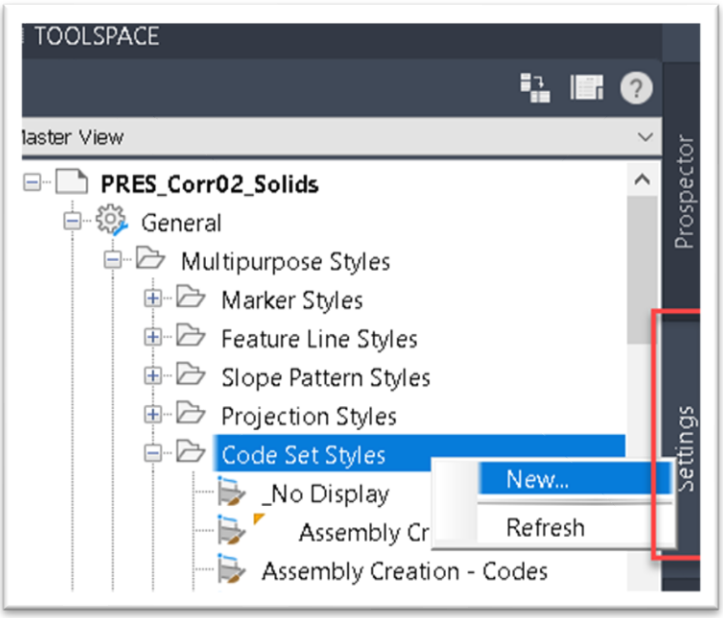


Figure 30 - Create Noe Code Set

Once the new code set has been given a name the next step is to import all the codes. This requires a methodical process to ensure all combinations of the codes are imported. There are two ways to do this.

Method 1

Under each of the headings point, link or shape right click on the heading choose add

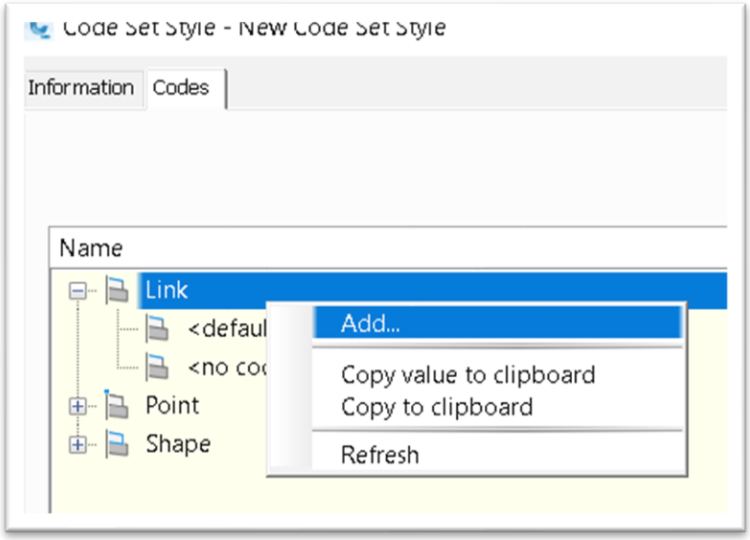


Figure 31 - Manually Adding Codes

The next prompt will be to chose a style for code. Once selected a new code will be created called “New Code” which can then be renamed. You can then use copy / paste from a list of all the codes developed during subassembly planning.

Method 2

Requires the subassemblies needed, to be inserted in to the .dwg (or selecting a corridor) where the code set is being created. A methodical approach is needed to open the subassembly properties and work through each of the subassembly parameters that control point, link or shape codes then use the “Import Codes” button for each parameter iteration to import the codes.

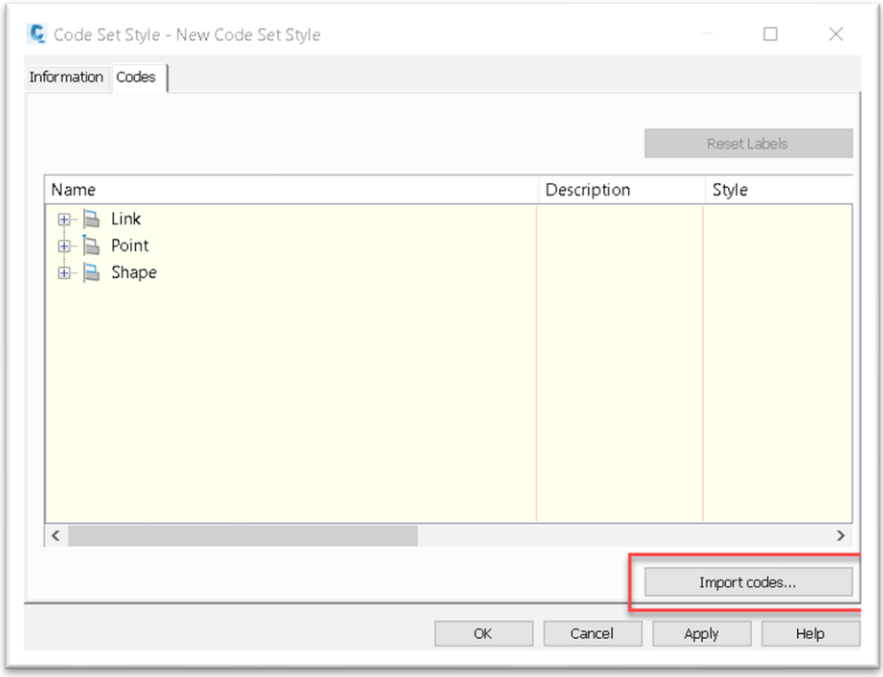


Figure 32 - Importing Codes

Both methods require a degree of work and a methodical approach.

Once all of the point, link and shape codes have been loaded, set all styles and labels to No Display or None then save as _No Display. All of the code set styles outlined below can then be created by copying and renaming the _No Display style that has all the codes already loaded. It just requires the styles to be applied to the correct code entries.

AU_Pres_Linework

This is a simple style to show the corridor linework that can be used on its own or layered up with other models using the other code set styles to provide project context. This is a core skill you should already be familiar with.

Name	Description	Style	Label Style	Render Material	Material Area Fill Style	Feature Line Style
Link						
Point						
<default>		_No Display	<none>			_No Display
<no codes>		_No Display	<none>			_No Display
BC_Top		_No Display	<none>			_No Display
BOK_BN3_Left		_No Display	<none>			KERB - Back
BOK_BN3_Right		_No Display	<none>			KERB - Back
BOK_HB2_Left		_No Display	<none>			KERB - Back
BOK_HB2_Right		_No Display	<none>			KERB - Back
BOK_SP2_Left		_No Display	<none>			KERB - Back
BOK_SP2_Right		_No Display	<none>			KERB - Back
BOK_TK_Left		_No Display	<none>			KERB - Back
BOK_TK_Right		_No Display	<none>			KERB - Back
Base_Top		_No Display	<none>			_No Display
CentralReserve_Left		_No Display	<none>			_No Display
CentralReserve_Right		_No Display	<none>			_No Display
Channel_Left		_No Display	<none>			CWAY - Channel
Channel_Right		_No Display	<none>			CWAY - Channel

Figure 33 - AU_Pres_Linework Code Set Style

AU_Pres_Visualisation

A second simple style that applies “realistic” colour hatches to the corridors, primarily used for public consultation type drawings. Uses “Pavement” and various kerb link codes to apply suitably coloured hatches.

Name	Description	Style	Label Style	Render Material	Material Area Fill Style
Link					
<default>		_No Display	<none>	<none>	<none>
<no codes>		_No Display	<none>	<none>	<none>
BC_Top		_No Display	<none>	<none>	<none>
BN3_Left		_No Display	<none>	<none>	Colours - Dark Grey ...
BN3_Right		_No Display	<none>	<none>	Colours - Dark Grey ...
Base_Top		_No Display	<none>	<none>	<none>
Formation		_No Display	<none>	<none>	<none>
HB2_Left		_No Display	<none>	<none>	Colours - Dark Grey ...
HB2_Right		_No Display	<none>	<none>	Colours - Dark Grey ...
HFS		_No Display	<none>	<none>	<none>
Hidden		_No Display	<none>	<none>	<none>
PSV50		_No Display	<none>	<none>	<none>
PSV53		_No Display	<none>	<none>	<none>
PSV58		_No Display	<none>	<none>	<none>
PSV60		_No Display	<none>	<none>	<none>
PSV63		_No Display	<none>	<none>	<none>
PSV65		_No Display	<none>	<none>	<none>
PSV68		_No Display	<none>	<none>	<none>
Pavement		_No Display	<none>	<none>	Colours - Light Grey
SP2_Left		_No Display	<none>	<none>	Colours - Dark Grey ...
SP2_Right		_No Display	<none>	<none>	Colours - Dark Grey ...
SubBase_Top		_No Display	<none>	<none>	<none>
Top		_No Display	<none>	<none>	<none>
TypeA		_No Display	<none>	<none>	<none>
TypeB		_No Display	<none>	<none>	<none>
TypeC		_No Display	<none>	<none>	<none>
TypeD		_No Display	<none>	<none>	<none>
Point					

Figure 34 - AU_Pres_Visualisation Code Set Style

Note – Future improvement to the kerb subassembly. Add "*Kerb*" as an additional link code to the links shown in Figure 11, in a similar way that the pavement subassembly adds "*Pavement*" as an additional link to the top link of all pavement types regardless of what pavement type is set. This would allow the number of entries in the material fill style column to be reduced to two, "*Kerb and "*Pavement*".

AU_Pres_Kerbs

This is the first code set style where the automation intelligence built into the subassemblies begins to be leveraged. Each kerb type has a different feature line style assigned to the point codes. In this example the feature line styles use line styles with letters representative of the kerb type, except the transition kerbs which use a plain line due to the short lengths of these lines resulting in the text within the linestyle not displaying. A sample of the output can be seen in Figure 2.

Name	Description	Style	Label Style	F	M	Feature Line Style
Link						
Point						
<default>		_No Display	<none>			_No Display
<no codes>		_No Display	<none>			_No Display
BC_Top		_No Display	<none>			_No Display
BOK_BN3_Left		_No Display	<none>			KERB - BN
BOK_BN3_Right		_No Display	<none>			KERB - BN
BOK_HB2_Left		_No Display	<none>			KERB - HB2
BOK_HB2_Right		_No Display	<none>			KERB - HB2
BOK_SP2_Left		_No Display	<none>			KERB - SP
BOK_SP2_Right		_No Display	<none>			KERB - SP
BOK_TK_Left		_No Display	<none>			KERB - Transition
BOK_TK_Right		_No Display	<none>			KERB - Transition

Figure 35 - AU_Pres_Kerbs Code Set Style

AU_Pres_Pavements

This code set style hatches the pavement based on the pavement type set in the subassembly. For each of the types assign a material area fill style. In this example, coloured solid hatch is used, enabling AU_Pres_PavementsPSV to use patterned hatch. This then allows two presentation models to be created, the corridor referenced and these two styles applied. The models can then be overlayed in a drawing to show both type and PSV details.

Name	Description	Style	Label Style	Render Mat...	Material Area Fill Style	Fe
Link						
<default>		_No Display	<none>	<none>	<none>	
<no codes>		_No Display	<none>	<none>	<none>	
SubBase_Top		_No Display	<none>	<none>	<none>	
Top		_No Display	<none>	<none>	<none>	
TypeA		_No Display	<none>	<none>	CWAY - Pavement Type A	
TypeB		_No Display	<none>	<none>	CWAY - Pavement Type B	
TypeC		_No Display	<none>	<none>	CWAY - Pavement Type C	
TypeD		_No Display	<none>	<none>	CWAY - Pavement Type D	
Point						

Figure 36 - AU_Pres_Pavements Code Set Style

AU_Pres_PavementsPSV

This code set style hatches the pavement PSV based on the PSV value set in the subassembly, using the link codes of HFS and PSV*nn* to apply different Material Area Fill Styles that are created using patterned hatch.

Name	Description	Style	Label Style	Render Mat...	Material Area Fill Style
Link					
<default>		_No Display	<none>	<none>	<none>
<no codes>		_No Display	<none>	<none>	<none>
BC_Top		_No Display	<none>	<none>	<none>
BN3_Left		_No Display	<none>	<none>	<none>
BN3_Right		_No Display	<none>	<none>	<none>
Base_Top		_No Display	<none>	<none>	<none>
Formation		_No Display	<none>	<none>	<none>
HB2_Left		_No Display	<none>	<none>	<none>
HB2_Right		_No Display	<none>	<none>	<none>
HFS		_No Display	<none>	<none>	CWAY - Pavement HFS
Hidden		_No Display	<none>	<none>	<none>
PSV50		_No Display	<none>	<none>	CWAY - Pavement PSV50
PSV53		_No Display	<none>	<none>	CWAY - Pavement PSV53
PSV58		_No Display	<none>	<none>	CWAY - Pavement PSV58
PSV60		_No Display	<none>	<none>	CWAY - Pavement PSV60
PSV63		_No Display	<none>	<none>	CWAY - Pavement PSV63
PSV65		_No Display	<none>	<none>	CWAY - Pavement PSV65
PSV68		_No Display	<none>	<none>	CWAY - Pavement PSV68

Figure 37 - AU_Pres_PavementsPSV Code Set Style

AU_Pres_Solids

This code set is created for solids extraction / export and is critically important as it plays a part in asset data, co-ordination, construction scheduling and quantification processes.

This is a complicated code style to build as there are lots of possible combinations that need to be picked up when importing codes to the code set. The more granular the need for any of the use cases above, the more combinations that will be generated. Setting the styles in the code set also sets the colours of the 3D solids during the extraction process therefore, it impacts how they display in Civil3D and Navisworks. So, the more consideration this is given at this point the easier it is to use once the solids are extracted.

Name	Description	Style
Link		
Point		
Shape		
<default>		_No Display
<no codes>		_No Display
CH-Pr1531-E_-Pavement_SubBaseCourseTypeA		Colours - Light Brown (23)
CH-Pr1531-E_-Pavement_SubBaseCourseTypeB		Colours - Light Brown (23)
CH-Pr1531-E_-Pavement_SubBaseCourseTypeC		Colours - Light Brown (23)
CH-Pr1531-E_-Pavement_SubBaseCourseTypeD		Colours - Light Brown (23)
CH-Pr2593-E_-KerbBN3		Colours - Dark Grey (252)
CH-Pr2593-E_-KerbHB2		Colours - Dark Grey (252)
CH-Pr2593-E_-KerbSP2		Colours - Dark Grey (252)
CH-Pr2593-E_-KerbTransition		Colours - Dark Grey (252)
CH-Pr3531-E_-Pavement_BaseCourseTypeA		Colours - Blue Grey
CH-Pr3531-E_-Pavement_BaseCourseTypeB		Colours - Blue Grey
CH-Pr3531-E_-Pavement_BaseCourseTypeC		Colours - Blue Grey
CH-Pr3531-E_-Pavement_BaseCourseTypeD		Colours - Blue Grey
CH-Pr3531-E_-Pavement_BinderCourseTypeA		Colours - Light Green
CH-Pr3531-E_-Pavement_BinderCourseTypeB		Colours - Light Green
CH-Pr3531-E_-Pavement_BinderCourseTypeC		Colours - Light Green
CH-Pr3531-E_-Pavement_BinderCourseTypeD		Colours - Light Green
CH-Pr3531-E_-Pavement_SurfCourse_TypeA_HFS		CWAY - Pavement Type A
CH-Pr3531-E_-Pavement_SurfCourse_TypeA_PSV50		CWAY - Pavement Type A
CH-Pr3531-E_-Pavement_SurfCourse_TypeA_PSV53		CWAY - Pavement Type A
CH-Pr3531-E_-Pavement_SurfCourse_TypeA_PSV58		CWAY - Pavement Type A
CH-Pr3531-E_-Pavement_SurfCourse_TypeA_PSV60		CWAY - Pavement Type A
CH-Pr3531-E_-Pavement_SurfCourse_TypeA_PSV63		CWAY - Pavement Type A
CH-Pr3531-E_-Pavement_SurfCourse_TypeA_PSV65		CWAY - Pavement Type A
CH-Pr3531-E_-Pavement_SurfCourse_TypeA_PSV68		CWAY - Pavement Type A

Figure 38 - AU_Pres_Solids Code Set Style

AU_Pres_SettingOut & AU_Pres_GIS

These two code set styles are similar to AU_Pres_Linework. The key aim for both styles is to only display the linework that is needed for the specific purpose rather than displaying all the linework and using CAD export and delete or a tool like FME to extract just the lines needed. The advantage of this methodology is that the level of rework required for any design changes is significantly reduced.

For the “Setting Out” code set style the key is to talk with the construction partner, as the setting out strings required may depend on the Engineer, setting out methodology and even the software / equipment that is being used to do the setting out.

For the GIS code set style again it’s about what information is going to be displayed in GIS. For example, in most code set styles we don’t normally draw a feature line along the crown of the road as this is normally represented by the alignment. However, if using the ArcGIS Connector for Civil3D, alignments won’t export directly. Therefore, the simplest method is to use the code set style to plot a feature line that is more easily extracted.

How to use these new code set styles

Figure 39 shows the model breakdown structure needed to maximise the effectiveness of the automation opportunities created. Since the ultimate aim is to make the drawing content as dynamic to the design model as possible, numerous presentation models are needed, allowing the corridor to data shortcuted into these presentation models and stylised using the different code set styles to deliver the different outputs.

In this example the focus has been a carriageway object and a kerb object. This has been on purpose to ensure link between the subassemblies created and the code set styles can be more easily understood. Once these two object types are understood the same principles can be applied to many other objects, some examples as follows:-

- The principles from the kerb object can be applied to fencing, vehicle barriers, linear drainage channels or any other narrow linear features.
- The principles from the carriageway object can be applied to verges, footways, footpaths, earthworks, drainages channel, swales or any other “area based” corridor objects.

Model Setup Principles

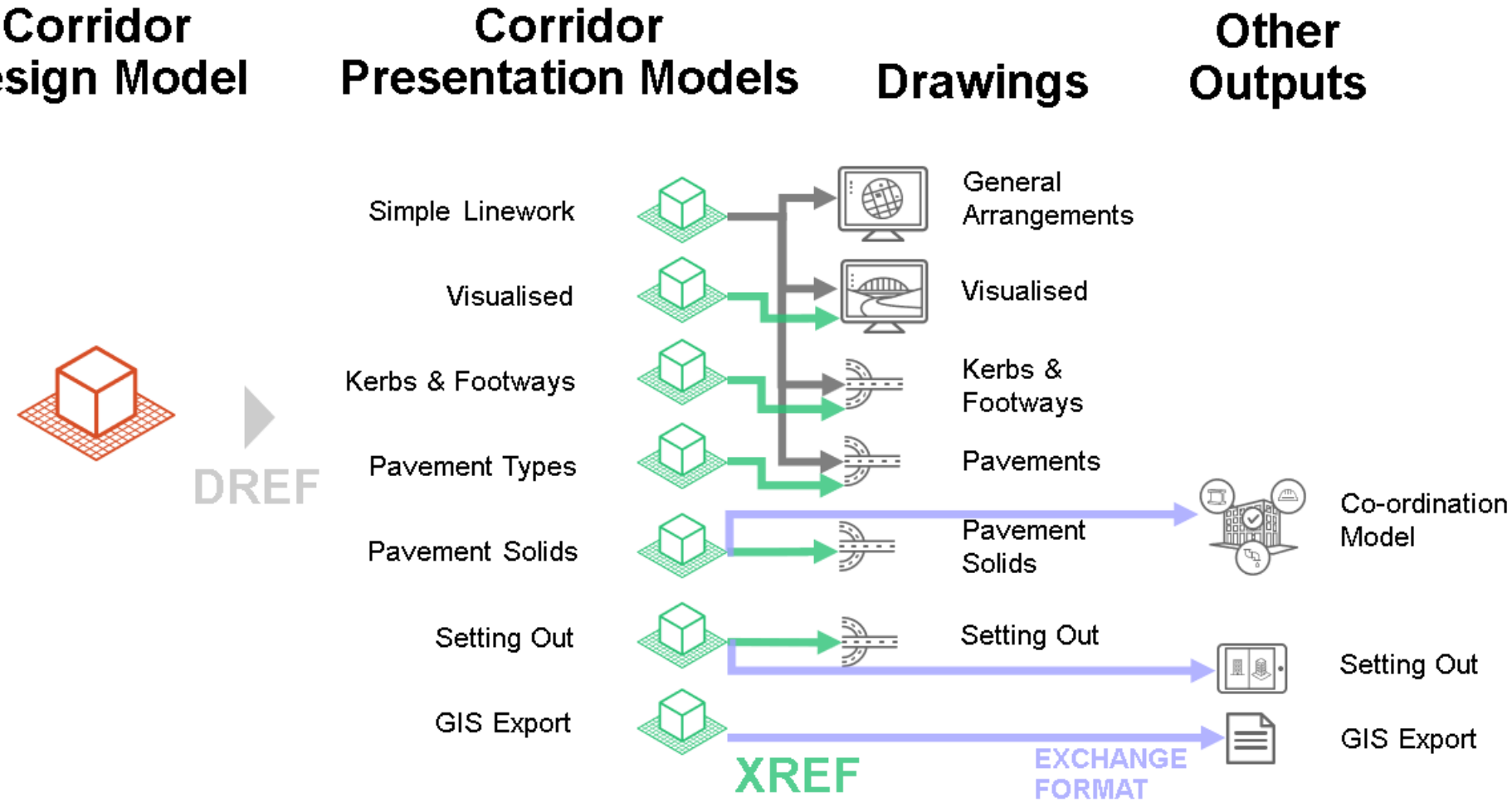


Figure 39 - Model Setup Requirements

Learning Objective 4 - How to use subassemblies and code sets to drive data deep into your quantification process

Quantification is not normally the realm of the Civil Engineer other than working with earthworks balance. However, with the increased use of model based delivery we are getting more frequent requests from our construction partners to provide quantification and in some cases we are asked to carry some / all of the risk from quantification of objects extracted from our models.

In order to maximise the benefits of using model object quantification the Work Breakdown Structure (WBS) or Method of Measurement needs to be defined. This need to be considered at the start of the project as it may influence some of the codification embedded in the subassemblies and model breakdown structure.

This is not a replacement for specialist software. The aim of these workflows is to be able to extract key quantities and show how using point, link and shape codes along with code set styles can make it easier and faster to do as well as provide more granular detail.

There are two possible routes that will be looked at as part of this learning objective: Civil3D QTO and Navisworks Quantification.

Civil3D QTO

For Civil3D we will look at how using the point, link and shape codes generated in the other learning objectives can extract the following

- Lengths of different kerbs using the kerb type point code styles
- Area of high friction surfacing using the PSV link codes
- Volumes of the pavement layers using the extracted corridor solids that are assigned type based layer naming automatically during extraction

The first step is to create a pay item file. This is a simple table saved as a CSV file, comprising pay item reference, description and unts. For more guidance watch Getting Started with Civil 3D Quantity Take-Off (QTO) - Civil Immersion (typepad.com)

	A	B	C
1	Pay Item	Item Description-USC	UNIT_E
2	1100-001	Precast concrete kerb type HB2	M1
3	1100-002	Precast concrete kerb type SP2	M1
4	1100-003	Precast concrete kerb type BN3	M1
5	1100-004	Precast concrete kerb transition	M1
6	912-001	Surface course Type A	M3
7	912-002	Surface course Type B	M3
8	912-003	Surface course Type C	M3
9	912-004	Surface course Type D	M3
10	905-001	Binder course Type A	M3
11	905-002	Binder course Type B	M3
12	905-003	Binder course Type C	M3
13	905-004	Binder course Type D	M3
14	904-001	Base course Type A	M3
15	904-002	Base course Type B	M3
16	904-003	Base course Type C	M3
17	904-004	Base course Type D	M3
18	924-001	High Friction Surfacing	M2
19	803-001	Type 1 Granular Sub-base	M3

Figure 40 - Simple Pay Item List

Step 2 is to load the Pay Item File (Sample file - AU Demo Pay Items.csv provided).

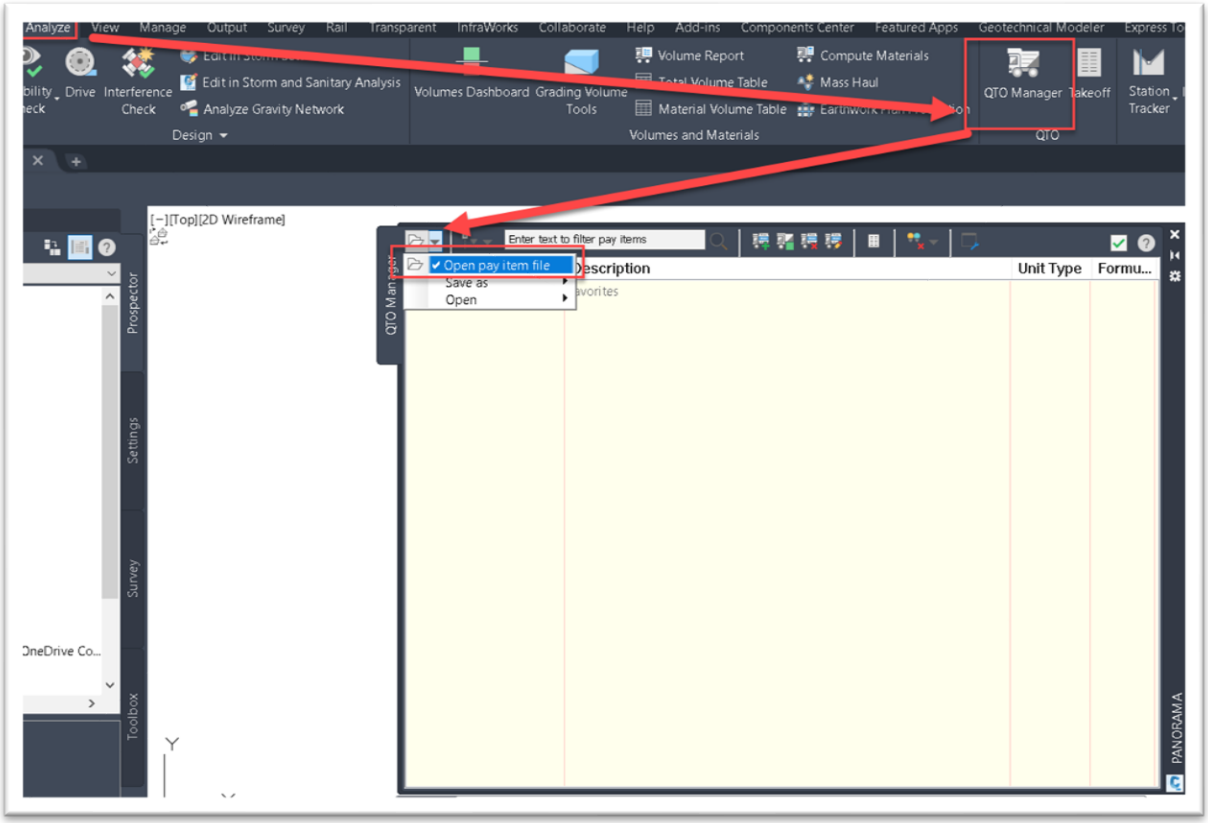


Figure 41 - Loading Pay Item File

Step 3 is to create a code set style which starts with similar steps to above.

- Start with the No Display Code Set Style previously created and create a copy.
- Assign feature lines styles to the same back of kerb point codes as done in AU_Pres_Kerbs.
- Assign Material Fill Styles to link codes for area based assets to be taken off e.g. High Friction Surfacing (HFS) as done in AU_Pres_PavementsPSV.
- Assign shape styles to the shape codes exactly the same as done in AU_Pres_Solids. This is important so extracted solids are easy to individually identify and assign the correct pay items to.
- Extract the corridor solids using “<Codes>” as the Layer Name Template
- Assign the pay items to the point codes as shown in Figure 42.

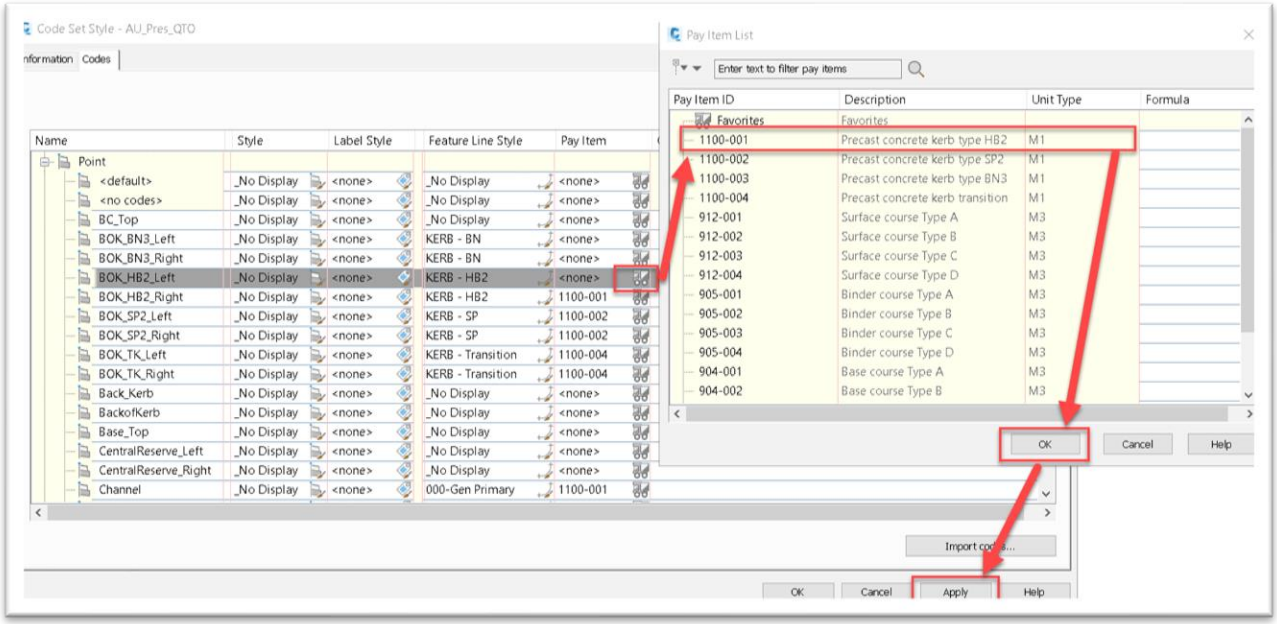


Figure 42 - Assigning Pay Items to Point Codes

- Repeat the pay item assignment with the Link Code for the high friction surfacing (HFS)

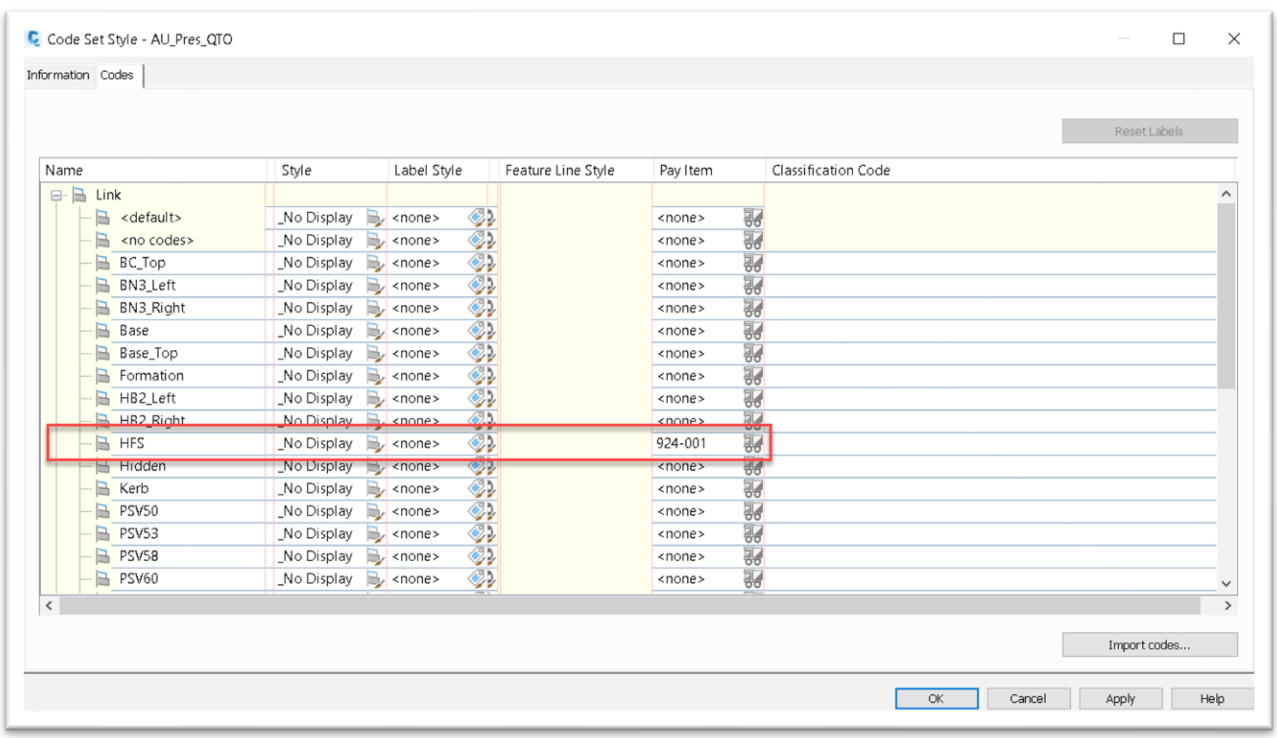


Figure 43 - Assigning Pay Items to Link Codes

Step 4 – Extracting corridor solids

- To obtain pavement volumes, the corridor needs to be extracted to solids and then pay items assigned to the solids. When extracting solids use “<Codes>” as the layer template, the codification done in the subassembly and code set style to ensure that the solids will be placed on the correct layers.

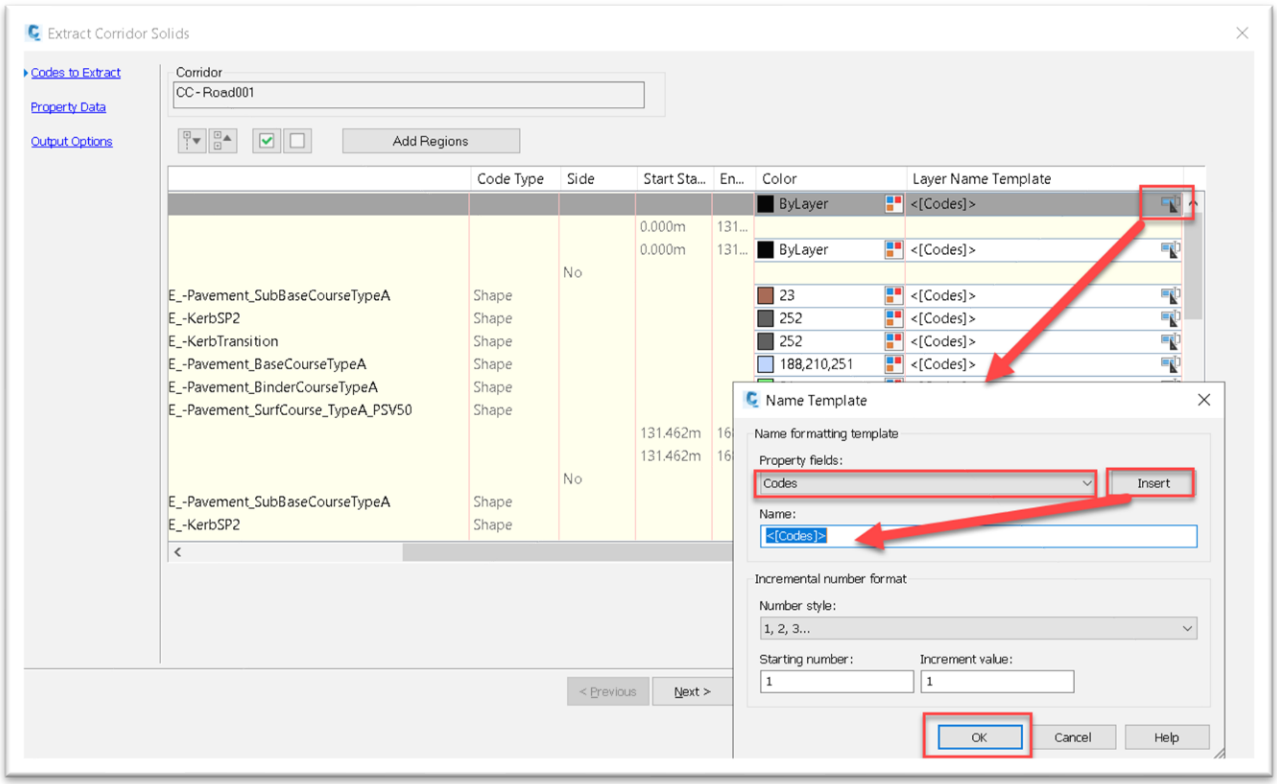


Figure 44 - Assigning <Codes> to solids extraction Layer Name Template

Step 5 – Assigning pay items to solids

- Use the layer manager to isolate the individual groups of the same solids.
- Select the solids and assign a pay item as shown in Figure 45.

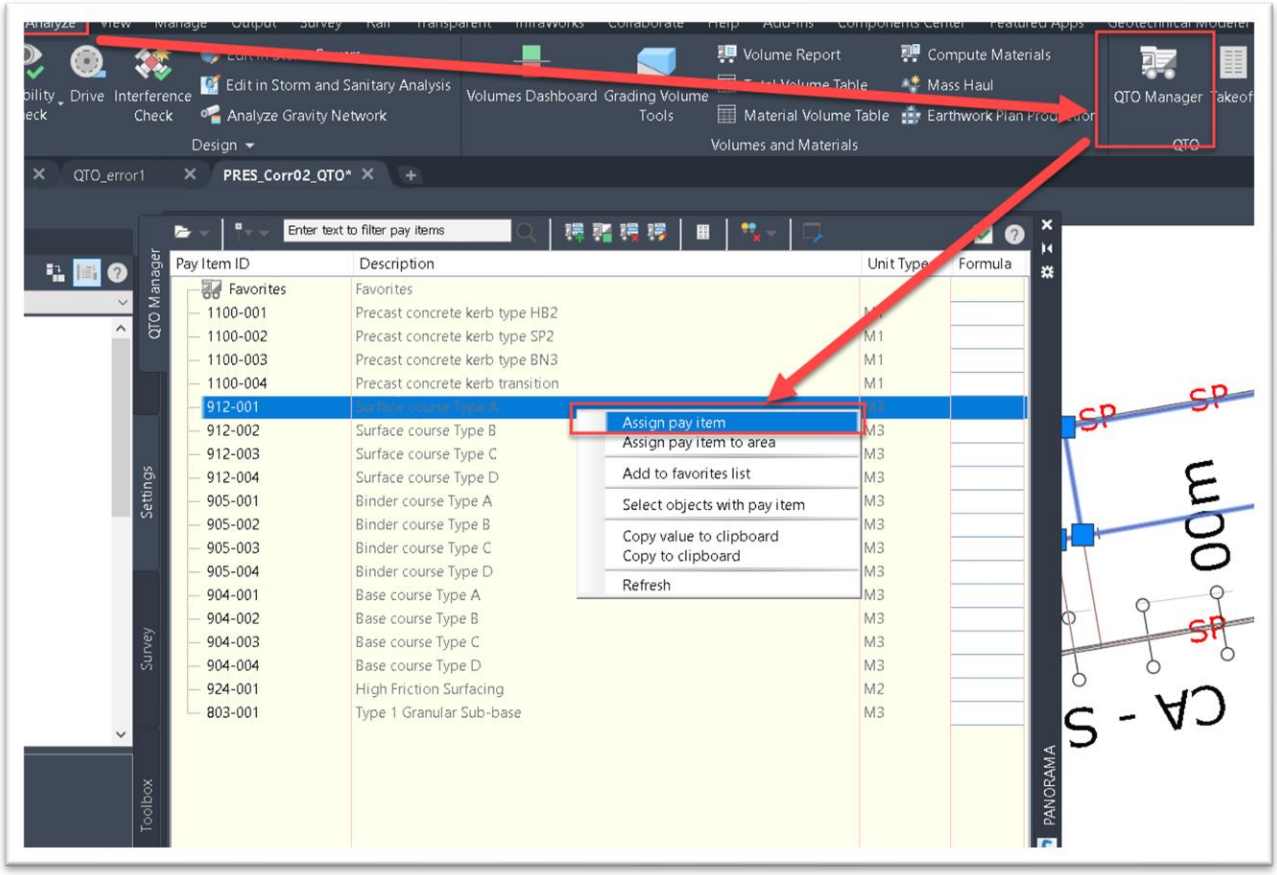


Figure 45 – Assign Pay Items to Corridor Solids

- Repeat with all solids.

Step 6 – Running the take off

- Follow the steps shown in Figure 46.

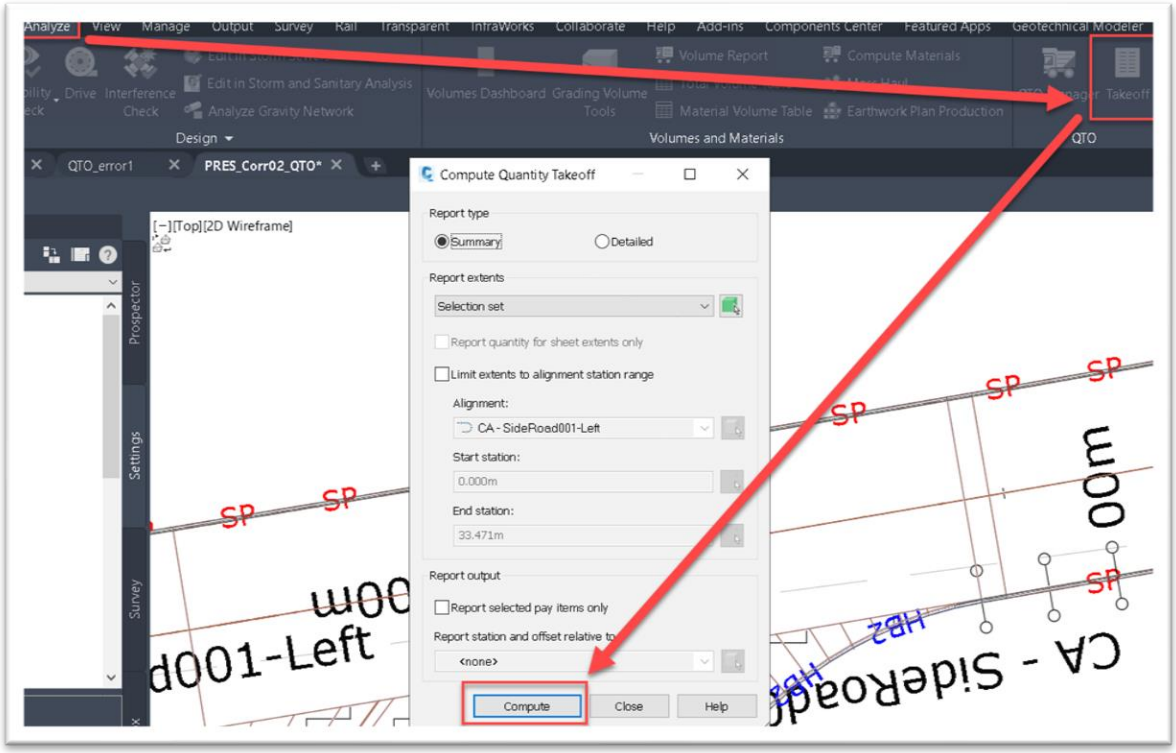


Figure 46 - Running the takeoff

- Choose the output format.

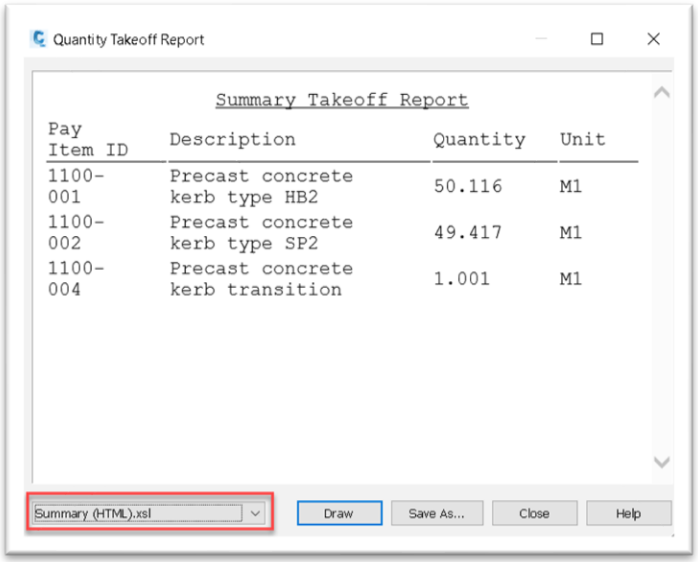


Figure 47 - Takeoff output format

Navisworks Quantification

This technique is developed around the best methodology I have found to build civil co-ordination models in Navisworks. During the design phase we expect our designers to perform clash avoidance and for this we would reference the Civil3D elements directly e.g., corridors, surface etc. For our co-ordination models referencing Civil3D elements doesn't give the granularity of components, nor do they provide great visuals that enhance the communication of design intent. Therefore, extracted solids are our preferred method to build the co-ordination models but this gives the added advantage of being able to leverage the Quantification module to access corridor volumes.

Before working in Navisworks there are a number of prerequisites that need to be undertaken in Civil3D

- Create a presentation model and datashortcut in the corridor(s)
- Stylise the corridor using the AU_Pres_Solids cose set style
- Create 2 layer states, a working state with all layers on and Navisworks Export with only the extracted solids layers on. This ensures you have a clean model and minimal object layers in the Navisworks Selection Tree.

  Now working with Navisworks the Civil3D solids model needs to be appended before the quantification can be set up as shown in Figure 48, Figure 49, Figure 50 and Figure 51

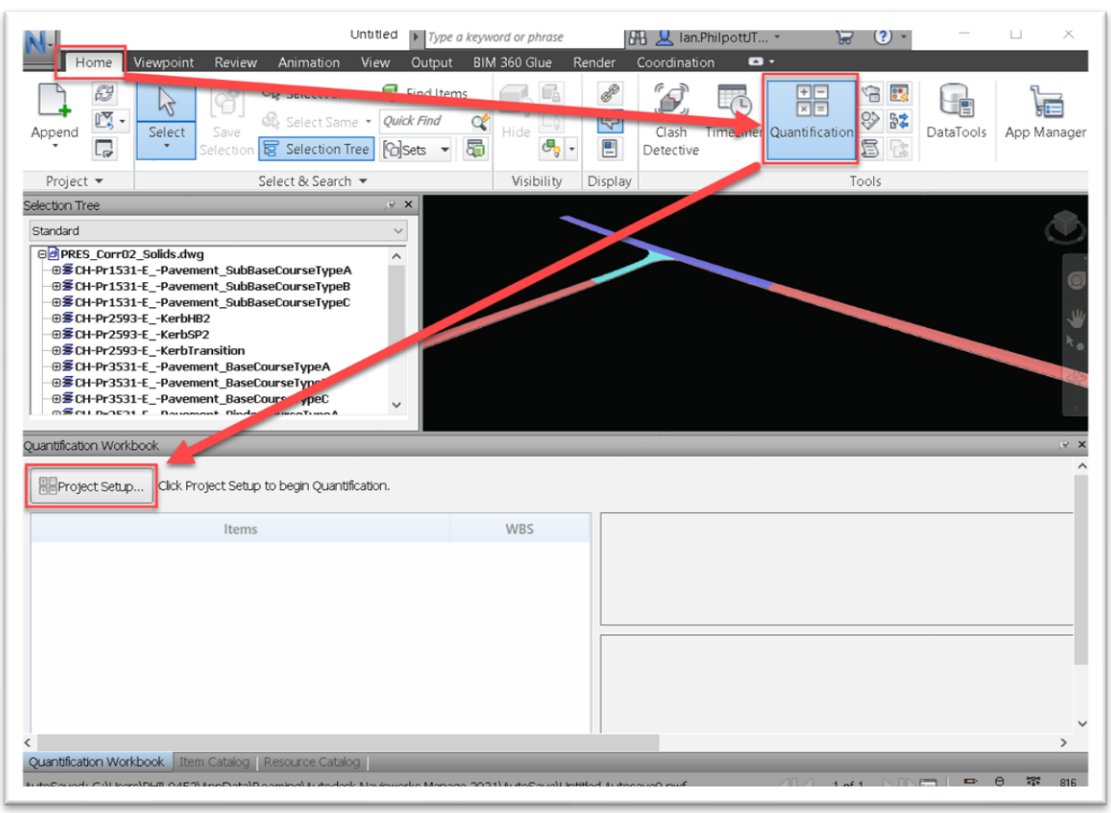


Figure 48 - Navisworks Quantification Setup

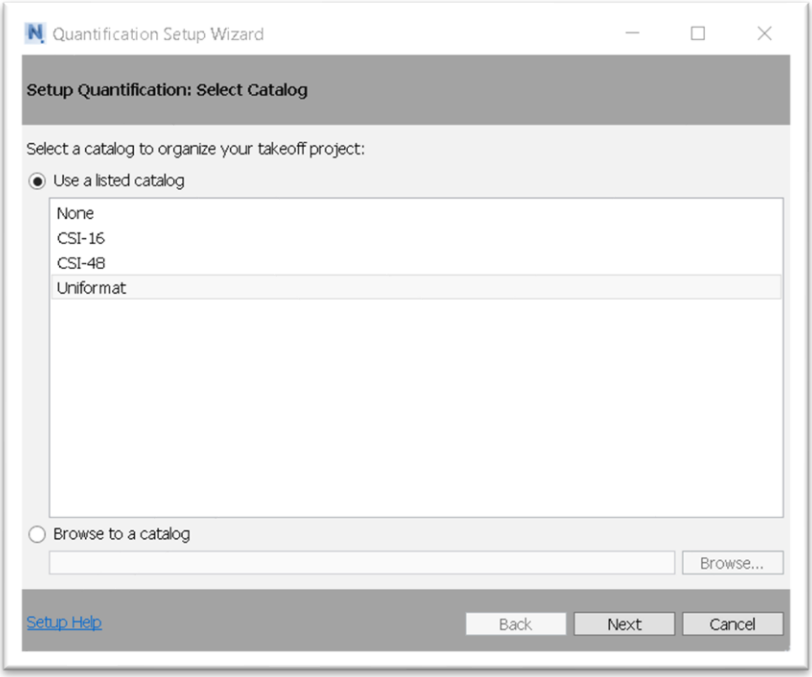


Figure 49 - Choose the Quantification Catalog

At the beginning of this learning objective, we talked about the Work Breakdown Structure (WBS). This catalog is where the WBS is specified. For more information on creating custom catalogs refer to Help | Create custom catalog data | Autodesk. In this example we will use the Uniformat catalog.

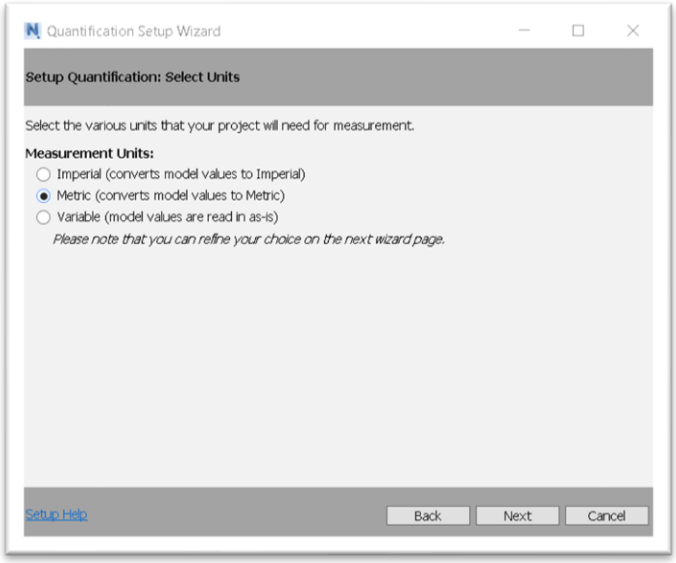


Figure 50 - Quantification Unit Selection

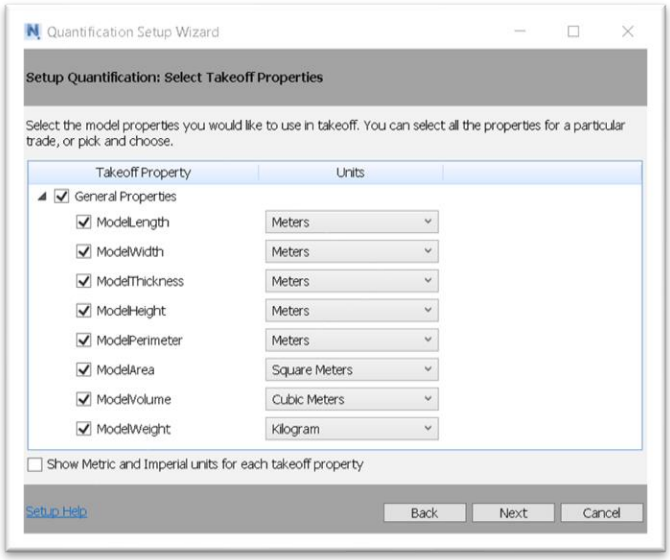


Figure 51 - Quantification Unit Mappting

Next step is to assign the property mapping to ensure that values from the solids are equated to the fields in the quantification module. See Figure 52,.adding the three properties shown

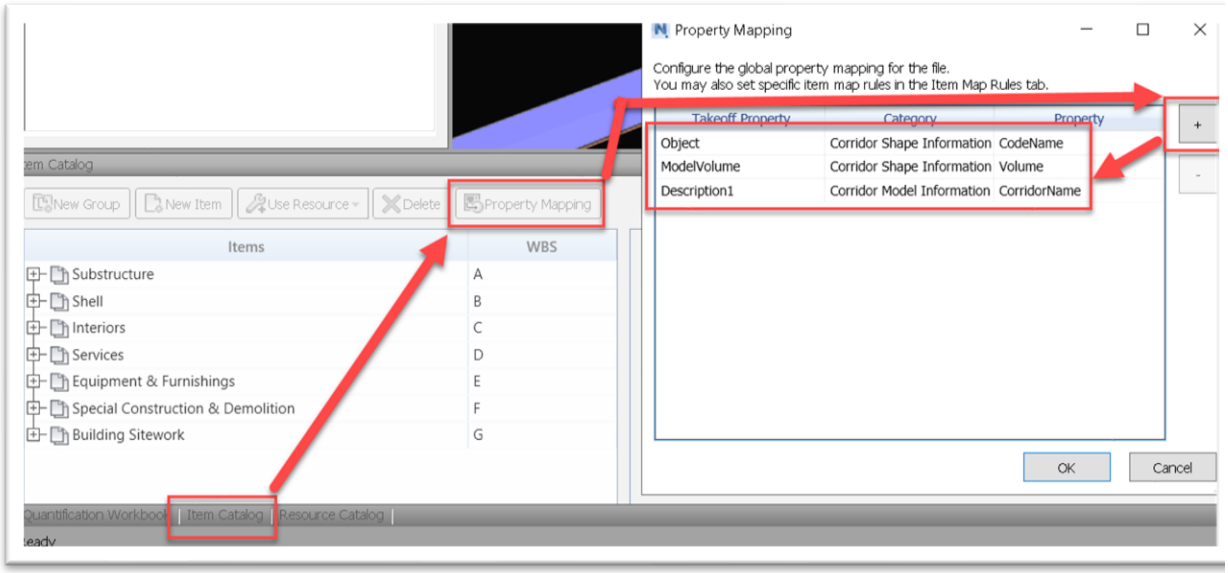


Figure 52 - Assigning Property Mapping

The final step is to add the objects to the quantification, which is a simple drag and drop. First expand the WBS and locate the relevant item then either drag the whole model or individual layer into the correct category (See Figure 53).

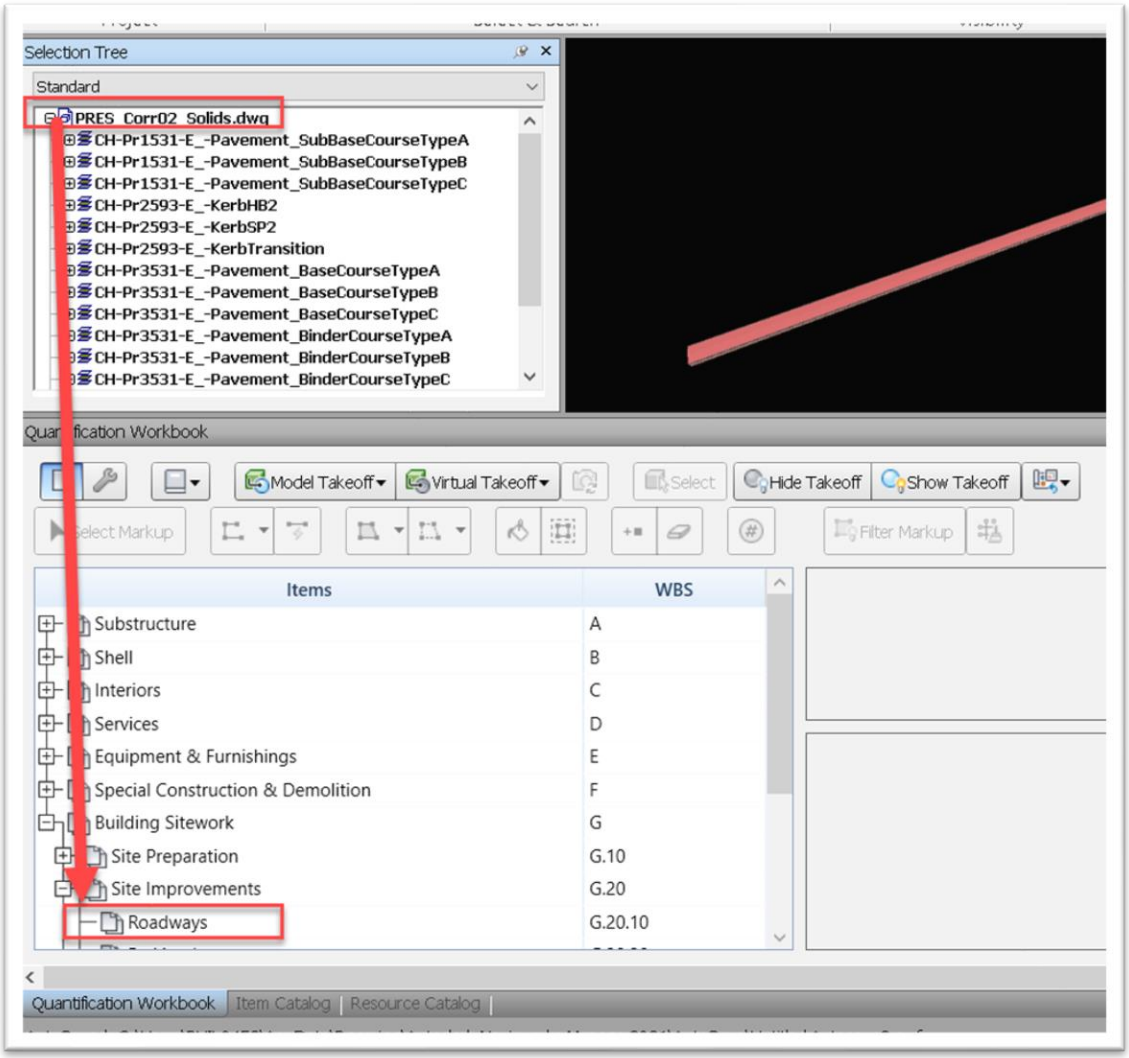


Figure 53 - Adding Elements to the Quantification

As a result of the effort put in developing to subassembly codes and creating code sets that provide a very granular level of breakdown of the solids but automatically putting them on layers that are descriptive of contents, the quantification can provide a granular breakdown of the objects and volume (See Figure 54).

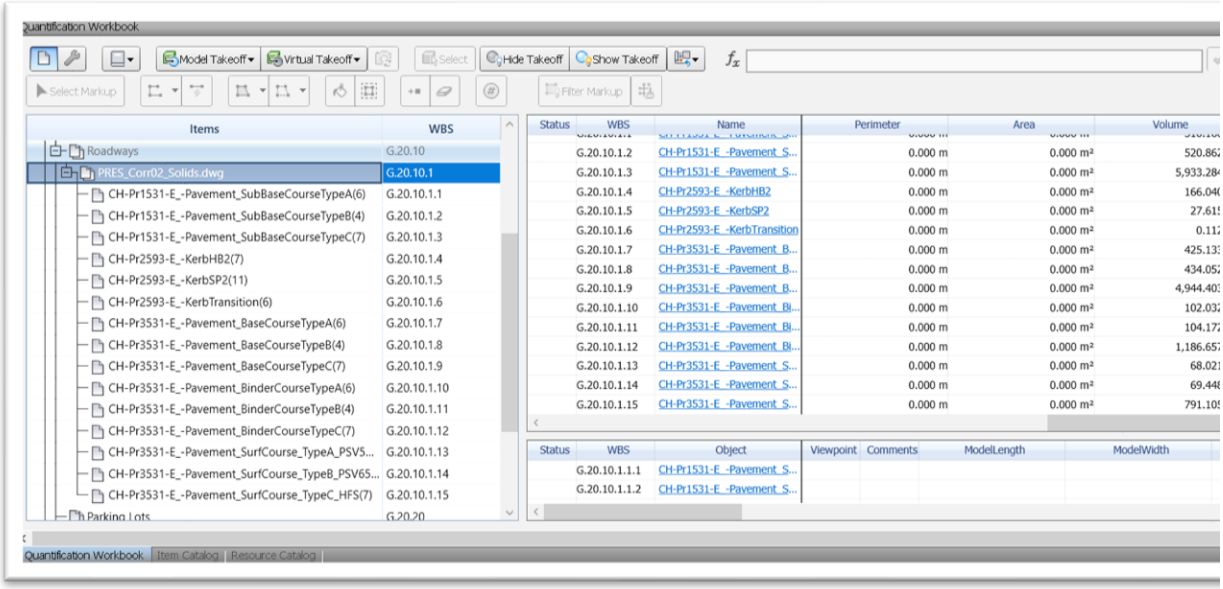


Figure 54 - Extracted Quantities

References

The internet is full of learning material on subassembly composer, Civil3D code sets and the other techniques used in this class. Below is listed a number of the resources I have found useful over the years developing these techniques.

Product Help

Civil3D Subassembly Help

Subassembly Composer Help
Subassembly Composer Tool Reference

Autodesk University Classes

Reverse Engineering with Subassembly Composer for AutoCAD® Civil 3D – Kati Merier (AU2012)
Autodesk® Civil 3D® 2013: An Introduction to Autodesk® Quantity Takeoff Using Subassembly Composer – Ravi Vaish (AU2012)
AutoCAD Civil 3D and Subassembly Composer-Real-World-Practice Tips and Tricks – Jowenn Lua (AU2017)
Custom AutoCAD Civil 3D Subassemblies—Why Would I Need that?? – Matthew Dalton (AU2019)

Civil Immersion Blog

Civil Immersion Blog
Exploring Civil 3D Code Set Styles
Creating a Code Set Style to Display the Plottable Geometry of a Civil 3D Corridor Model
Hatching Civil 3D Corridor Models using a Code Set Style
Visualizing Corridor Models by Assigning Render Materials to a Civil 3D Code Set Style
Getting Started with Civil 3D Quantity Take-Off (QTO) - Civil Immersion (typepad.com)

Navisworks Quantification

Help | Create custom catalog data | Autodesk