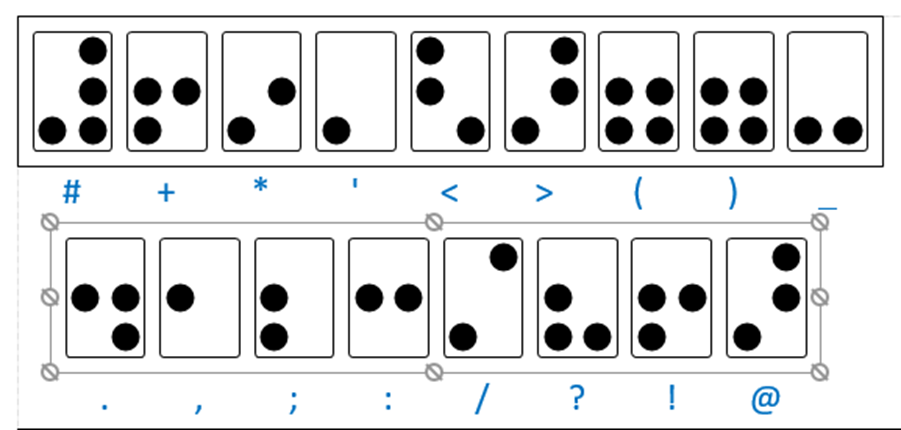
Before posting my Braille shape, I decided to embellish it first. The original shape was a single shape that had a Geometry section (Ellipse) for each of the dots (pips) and user cells User.Dot1-Dot6 contained the list of characters the shape represented. The Ellipses are independent of the shape and positioned by the X,Y of the Geometry section,. The NoFill cell in each ellipse Geometry controlled the visibility of each pip. Of course, this required negative logic, so I had to work out how to make the pip visible and then negate it.

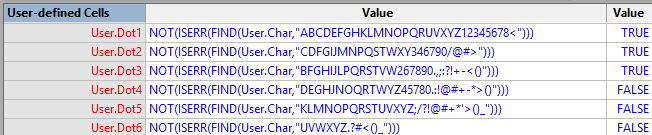
**Braille Cell**

The first enhancement I did was add the punctuation characters because they were single characters and a simple extension to the original set.



Punctuation

So, the User.Dotn cells now look like.



Dots

The other known characters would need more than a single character, so would require a different method. I also wanted to add a way to directly change the pips to make any pattern, which meant still another method. So three methods, that meant adding a Shape Data list that identifies the type of shape. The three types of patterns were identified by a Shape Data list called Type.

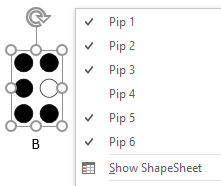
The original group is A-Z, the create your own is Pips and the other known characters is Special.

Prop.Type. has a Format of "A-Z;Pips;Special"

User.TypeIdx 1.0000 LOOKUP(Prop.Type,Prop.Type.Format)

**Pips**

The original way to control the pips to represent any character was to add six user cells, User.Pip1-Pip6 and use a right click menu to control the User cells. They are just Booleans.

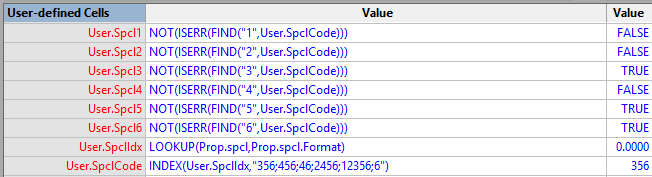
[](https://johnvisiomvp.files.wordpress.com/2022/02/pips.png)

Pips

For simplicity this was removed along with the right click menu.

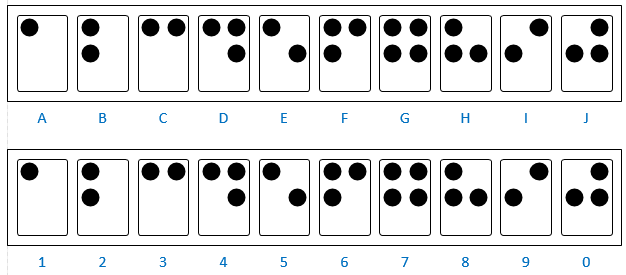
**Specials**

To handle the other shapes, I used a Shape Data List Prop.Spcl ("66; 99; ALPHA; NUM; AND; CAPITAL") with six user cells User.Spcl1-User.Spcl6, some more Booleans. There is also User.SpclCode that lists the pips that need to be enabled for each special character "356; 456; 46; 2456; 12356; 6"



Special

As you can see in the patterns below, some special codes are required.

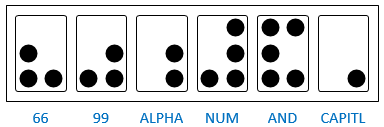


A-J 1-0

(those who developed Braille were more Fortran developers than C developers (numbers start with 1, not zero).)

As you can see, there needs to be a way to indicate what those characters mean so there are leading characters to indicate whether the string was Alpha, Numeric, or even if the string is capitalized.

The special shapes are:



Special

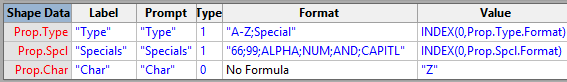
The 66 & 99 are the infamous smart double quotes.

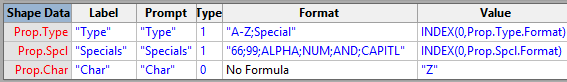
To simplify things, I decided to eliminate the Pips because if a new character is needed, it can be added as a Special by providing a name and a list of the pips that need to be on. But I found this cumbersome and if someone really needed a custom Pip, it would be far easier to add another item to the Specials collection. The process involved using a right click menu. Once added, I then had to expand the right click menu for the other patterns.

So, I removed it. That reduced the complexity of the shape by simplifying the Geometry sections and the need for a right click menu.

To add new shapes, update User.SpclCode and Prop.Spcl with the new name and pattern.

This is the Shape Data.



****

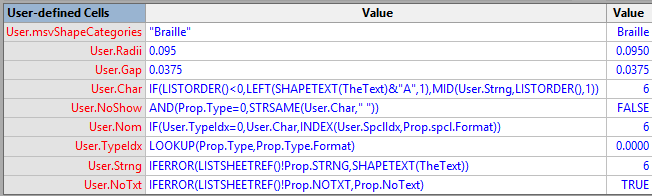
Shape Data

Prop.Type is either A-Z or Special.

Prop.Spcl is the list of Special types, User.SpclCode indicates what pips are on for tat code.

Prop.Char is the character to use when the shape is standalone and not in a container. In a container the character used is the one in the Container shape string at the same position.

These are the other User Defined Cells

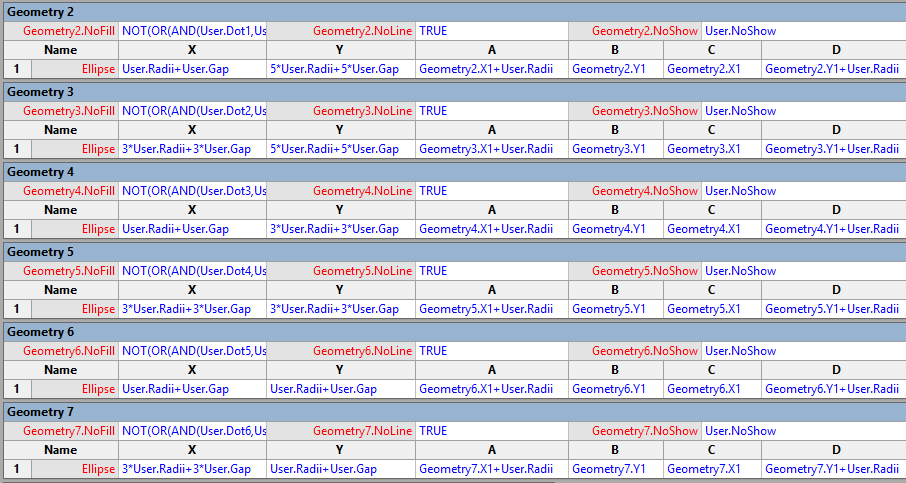


Other User Cells

Braille Cell contains two other cells. Rounding rounds the shape Shape Data of the Braille cell and HideText controls if the shapes text is visible. The visibility conditions are in User.NoTxt.

****

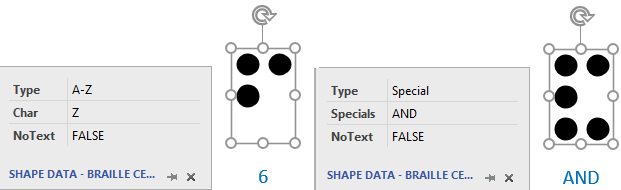
The Geometry section uses the method I described in a previous post that made the Ellipse row independent of the shape and located based on the X, Y values. Cells A, B, C and D are based on the X and Y cells.



Geometry

**Use**

To use the shape. Drop a shape and change the text of the shape.



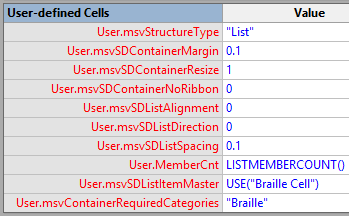
Shape Data

You can also change the Type to being Special.

**Braille Container**

Finally, I added a group shape that was a collection of Braille cells. Each subshape referenced a character in the group shape's text.

The Group/subshapes required a lot of work knitting the shapes together. Luckily Visio had a simpler method, Containers. The Container took care of the spacing of the Braille Cells and the Braille Cells had easy access to the Braille Containers Shape Data. With the Group/subgroup, I had to connect each shape to the Group shape by adding Sheet.5 where needed. I did use User cells as an interface rather placing the references to the group shape throughout the subshapes.



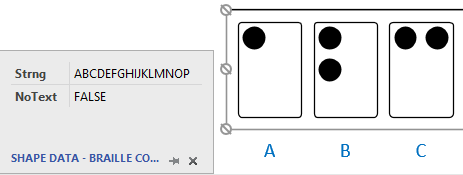
Container

**Use**

To use the shape, drag a Braille container shape to the page. On drop, seven Braille shapes will be added to the Braille container. The common type of Braille shape is A-Z and Special are unique. So, the Braille shape master should be A-Z. Once the container is dropped, the individual Braille shapes can be changed to a Special Shape. A-Z shapes inherit the character at the same position in the User Data Strng as it is in the container. If you are going to use a Special Shape remember to add a position saver character (other than a space) to the string.

Rather than create separate container shapes for each “word” put a space in the string, there will be a suitable gap for a missing Braille Cell.

The container is a bit finicky, trying to select the container, sometimes results in a new Braille Cell being accidently added.



Container Shape Data

**Summary**

I should have used VBA to create the shapes so that I had control of the order of the User cells and automate the stitching of the subshapes to the group shape. As shapes evolve, you are restricted to only add new items to the end of some sections. I compensate for the problem by adding extra User rows so I have some play with moving things around.

So, this has taken longer than planned. I had issues that kept me away from my Visio machine.

So, now to wait for David and Scott to point out what I missed. Considering I was very picky when I did the technical edit on their books. It is time for them to get pay back. 😉

**Samples**

These are the text strings used in the samples in the Visio drawing.

#+\*'<>()\_

.,;:/?!@

ABCDEFGHIJ

1234567890

