Windows Mapping Modes and Tools

Suppose the following sequence of events:

1. BeginPaint returns a default HDC (map mode = MM_TEXT)
2. Select Pens, Brushes, and Fonts into the device context
3. Change the mapping mode using SetMapMode
4. Change the window & viewport extents

Question: Have the pens, brushes, fonts, etc already in the device context been updated by Windows automatically, since they were specified in "logical units" when they are created and I have changed the "logical unit" mapping? Are they still the same size,
(in device units) that they were before? Do I have to create new pens, brushes, and fonts in order to have the size updated? Do all three have to be recreated, or just some?

Answer: Pen widths are in logical units. When you change mapping modes, there is the possibility that a logical unit could cover more that one pixel. Windows does not change the width of the pen when the mapping mode is changed. The pen width is still the same logical units.

The change to a different mapping mode can cause the pen to expand or contract depending on the mapping mode, window extents and viewport extents. Fonts will act the same way as pens. A font will be stretched or compressed with respect to the mapping mode, window extents and viewport extents. The size of a brush is 8 pixels by 8 pixels. This will not change among mapping modes.

All in all, some items might have to be recreated and some might not. It mainly depends on how the logical units map on to the new map mode. Windows does not adjust the logical sizes of the pen ,font, or brush.

Question: The Windows SDK documents the calcultations used by Windows to transform Logical coordinates to device coordinates. Is this calculation performed all the time such that no matter what the mapping mode or extents, an output sequence of events to a device always has the same overhead/time, or is MM_TEXT more efficient because it maps directly to the internal system used by Windows and thus the transformation calculations are not done?

Answer: With respect to your question of mapping mode overhead:

For ALL mapping modes, Windows will translate logical coordinates to device coordinates using the following formulas:
xViewport = (xWindow - xWinOrg)*(xViewExt/xWinExt) + xViewOrg
yViewPort = (yWindow - yWinOrg)*(yViewExt/yWinExt) + yViewOrg
where (xWindow, yWindow) is a logical point to be translated and (xViewPort, yViewPort) is the translated point in device coordinates. If the device coordinates are client-area coordinates or window coordinates then Windows must translate these device coordinates to screen coordinates before drawing an object.

For the MM_TEXT mapping mode, the default origins and extents are:

Window origin	(0, 0)
Viewport Orgin	(0, 0)
Window extent	(1, 1)
Viewport extent	(1, 1)

The ratio of the viewport extent to the window extent is 1. No scaling will be performed between logical and device coordinates.

So the overhead is generally going to be the same since all mapping modes will use this translation. A good source on mapping modes is "Programming Windows 3.00" by
Charles Petzold.

End.