Windows Driver Programmatic Printing Sample Application Summary

Austin Neidert

10-24-2018

Purpose

The intention for this sample application is to provide code demonstrating printing to the Qualsoft Windows driver as well as modifying preferences used by the Qualsoft Windows driver. This project is targeted for software developers attempting to control a Microcom printer programmatically, while using the Qualsoft Windows driver as the communication interface.

Requirements

1. Windows Computer
2. Visual Studio
3. Qualsoft Windows Driver (Driver 2.0.0.79 for Sample Application 1.0.0.0)

Programming Guide

Note: An image must be loaded prior to other actions being performed.

1.0 Printing

Printing is done by the following few lines:

```
PrintDocument document = new PrintDocument();
document.PrintPage += printPageEvent;
document.Print();
```

In the above code, the document is created and then the printPageEvent is added as an event handler to the PrintPage event. This can be used to modify the graphics of the document within the event handler. In this case, the event handler prints the loaded image by the following line:

```
      e.Graphics.DrawImageUnscaledAndClipped(loadedImage, new Rectangle(0, 0,
loadedImage.Width, loadedImage.Height);
```

The code above should be called by the event handler prior to the Print function being called. The Print function will trigger the job to be sent to the Windows driver. At this point the job will appear in the spooler queue until it is actually printed.

2.0 Modifying Preferences

2.1 Print Dialog

One option to modify printing preferences is by using the print dialog. The print dialog can be opened and preferences can be modified within the dialog, itself. The dialog can be opened using the following code:

```
      PrintDialog printDialog = new PrintDialog();
      printDialog.ShowDialog();
```

After calling these two lines, the code will hang until the dialog is closed. Once the dialog is closed the code will progress. In order to print using the selected preferences, the printing code specified in section 1.0 should be called immediately following the print dialog code. This will trigger the printing of the loaded image using the modified settings.

2.2 DEVMODE struct

The preferences can be modified programmatically using the DEVMODE struct. The DEVMODE struct is a concatenation of the public half and the private half of the DEVMODE. The public half is not specific to a driver. It is a Microsoft standard struct and will not change. The private half will change based on the driver’s layout of the driver specific preferences in memory. This can be different for each version of the Windows driver. If a new Windows driver version introduces changes to its internal DEVMODE, the application’s private DEVMODE struct will need to be changed.

The DocumentProperties function has three main uses. First, it can be used to query for the total size of the DEVMODE struct. This is done with the following code:

```
      int dmSize = PrintDirect.DocumentProperties(IntPtr.Zero, lhPrinter,
null, IntPtr.Zero, IntPtr.Zero, 0);
```

This can be used to allocate the required memory used for the struct. Second, it can be used to populate a copy of the struct. This can be done with the following code:

```
      int dpResult = PrintDirect.DocumentProperties(IntPtr.Zero, lhPrinter,
null, outPtr, IntPtr.Zero, PrintDirect.DM_OUT_BUFFER);
      PrintDirect.DEVMODE devmode = new PrintDirect.DEVMODE();
      devmode = (PrintDirect.DEVMODE)Marshal.PtrToStructure(outPtr,
devmode.GetType());
```

Once the copy of the struct is populated, it can be modified. Third, DocumentProperties can be used to pass a modified DEVMODE struct back into a Windows driver to be used on the next printed document. This can be done with the following code:

```
      IntPtr inPtr = Marshal.AllocHGlobal(dmSize);
      Marshal.StructureToPtr(devmode, inPtr, false);
      dpResult = PrintDirect.DocumentProperties(IntPtr.Zero, lhPrinter, null,
outPtr, inPtr, PrintDirect.DM_IN_BUFFER);
```

After the desired properties are changed and passed back to the Windows driver, the printing code specified in section 1.0 should immediately be called. This will trigger the printing of the loaded image using the modified settings.