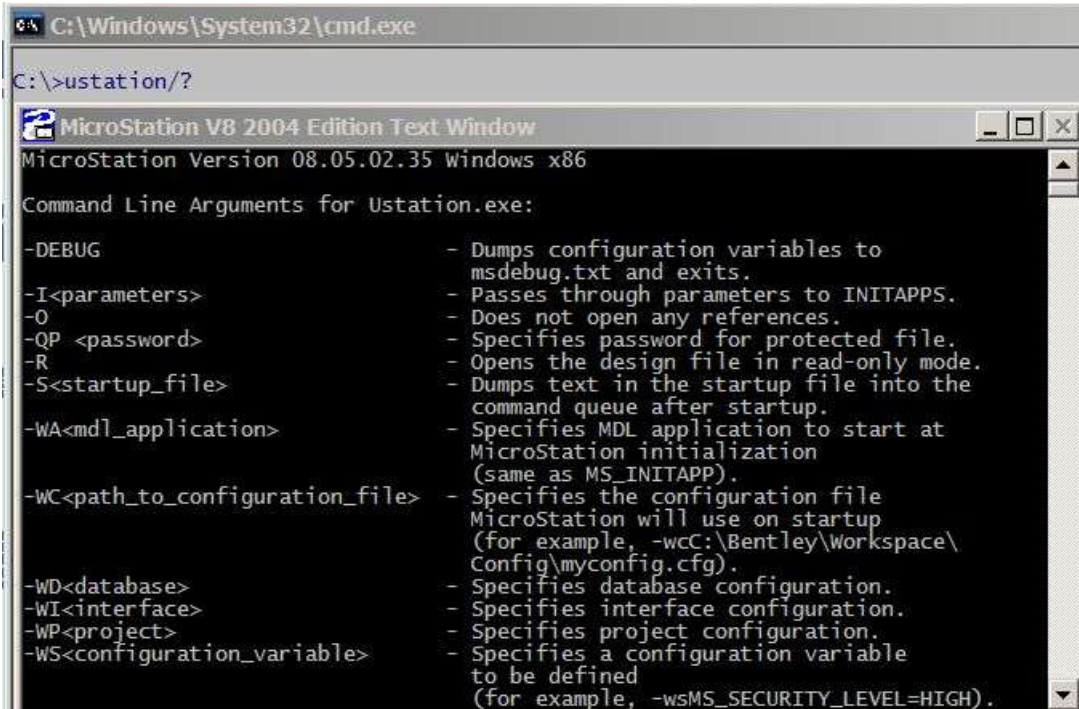


A question that pops up now and again on the Bentley discussion groups concerns Windows shortcuts and their use with MicroStation. MicroStation is a highly configurable product, and can be started in various ways to invoke one of many different configurations. There are two parts to understanding shortcuts ...

- MicroStation's command-line switches
- MicroStation configuration files

### Command-Line Switches

MicroStation has a number of command-line switches and options. Bentley Systems has published a technical note (<http://selectservices.bentley.com/technotes/technotes/7102.htm>) that describes them. You can see the options at any time by opening a Windows command prompt and entering `ustation.exe /?`. The result will look something like this screenshot ...



```
C:\Windows\System32\cmd.exe
C:\>ustation/?

MicroStation V8 2004 Edition Text Window
MicroStation Version 08.05.02.35 Windows x86

Command Line Arguments for Ustation.exe:

-DEBUG                - Dumps configuration variables to
                        msdebug.txt and exits.
-I<parameters>        - Passes through parameters to INITAPPS.
-O                    - Does not open any references.
-QP <password>        - Specifies password for protected file.
-R                    - Opens the design file in read-only mode.
-S<startup_file>      - Dumps text in the startup file into the
                        command queue after startup.
-WA<mdl_application> - Specifies MDL application to start at
                        MicroStation initialization
                        (same as MS_INITAPP).
-WC<path_to_configuration_file> - Specifies the configuration file
                        MicroStation will use on startup
                        (for example, -wC:\Bentley\Workspace\
                        Config\myconfig.cfg).
-WD<database>         - Specifies database configuration.
-WI<interface>        - Specifies interface configuration.
-WP<project>          - Specifies project configuration.
-WS<configuration_variable> - Specifies a configuration variable
                        to be defined
                        (for example, -wsMS_SECURITY_LEVEL=HIGH).
```

Why are command-line switches important? Because you can use them in a Windows shortcut. The image below is the shortcut on my desktop for MicroStation GeoGraphics.



When I right-click the icon, Windows pops a menu. I selected Properties to see the Properties Dialog for this shortcut.



The command-line use to start a Windows application is in the Target box. The command to start GeoGraphics is too long to display in the screen-shot, but you can copy and paste it into a text editor to see it in full. Here's the command line for my GeoGraphics shortcut (this is all on one line in the shortcut, but line-wrapped here to fit the PDF document) ...

```
"C: \Program Files\Bentley\Program\MicroStation\ustation.exe"
-wuMasterMapSQL
-wc"C: \Program Files\Bentley\Program\GeoGraphics\config\gglocal.cfg"
blank.dgn
```

Let's analyse each component of this command-line ...

1. "C: \Program Files\Bentley\Program\MicroStation\ustation.exe" starts the MicroStation executable. It's wrapped in double quotes because the path contains spaces
2. -wuMasterMapSQL uses the -wu switch to specify a user workspace named MasterMapSQL. The user workspace is defined in a configuration file C: \Program Files\Bentley\Workspace\Users\MasterMapSQL.ucf
3. -wc"C: \Program Files\Bentley\Program\GeoGraphics\config\gglocal.cfg" uses the -wc switch to specify an application configuration file. It's wrapped in double quotes because the path contains spaces. This is a file gglocal.cfg installed by GeoGraphics in its own directory tree
4. blank.dgn specifies a design file to open

Your shortcut doesn't need to be as complex as this. The simplest shortcut starts MicroStation and specifies a design file to be edited ...

```
"C: \Program Files\Bentley\Program\MicroStation\ustation.exe"  
file-name.dgn
```

You can start MicroStation with a named user configuration file `MyConfig.ucf` using the `-wu` switch ...

```
"C: \Program Files\Bentley\Program\MicroStation\ustation.exe"  
-wuMyConfig file-name.dgn
```

You can start MicroStation with a named project configuration file `MyProject.pcf` using the `-wp` switch ...

```
"C: \Program Files\Bentley\Program\MicroStation\ustation.exe"  
-wpMyProject file-name.dgn
```

You can start MicroStation with both a named user configuration file and a named project configuration file `MyProject.pcf` using both the `-wu` and `-wp` switches ...

```
"C: \Program Files\Bentley\Program\MicroStation\ustation.exe"  
-wuMyConfig -wpMyProject file-name.dgn
```

### Configuration Files

MicroStation configuration files are used to specify operational settings, data files, and folders, and multiple file locations. Many configuration files are stored in a sub-folder of `C: \Program Files\Bentley\Workspace`. As a CAD administrator, you will be interested in the contents of `C: \Program Files\Bentley\Workspace\Projects`. As a CAD user, you will be more interested in the contents of `C: \Program Files\Bentley\Workspace\Users`.

Configuration files are plain-text files. You can edit them with a simple editor such as Windows Notepad. You may prefer a more sophisticated editor such as [TextPad](http://www.textpad.com) (<http://www.textpad.com>), which, among other things, can open several text files simultaneously. Configuration files always use a UNIX-style forward slash (/) as a directory separator.

As described above, you instruct MicroStation to start in a given configuration using a command-line switch. The `-wu` switch instructs MicroStation to read the specified user configuration file. The `-wp` switch instructs MicroStation to read the specified project configuration file. You can use both switches to tell MicroStation to read both files.

The `-M` switch tells MicroStation to open the named model in a file.

The contents of a configuration file include a number of statements that assign a value to a configuration variable. For example, the variable `MS_DEF` determines the folder where MicroStation looks to find design files for editing. You can assign it to a Windows folder with this statement in a configuration file ...

```
MS_DEF = C: /Projects/Project01/dgn/
```

Configuration files always use a UNIX-style forward slash (/) as a directory separator. The backslash (\) is sometimes misinterpreted, so try to stick to the forward-slash to avoid problems.

It is common practise to assign multiple paths to a variable. For example, you may have several folders that contain files for a project. You might define the reference file search paths in `MS_RFDIR` like this ...

```
MS_RFDIR = C: /Projects/Project01/refs/  
MS_RFDIR > C: /Projects/Project01/borders/
```

Where the assignment operator `>` means to concatenate the paths, so MicroStation sees this as the final value of `MS_RFDIR` ...

```
C: /Projects/Project01/refs/; C: /Projects/Project01/borders/
```

One variable can be defined in terms of another variable. In the previous two examples, I assumed that a common root `C: /Projects/Project01/` contains sub-folders for design files and reference files. I can make the configuration file clearer if I define the project root in one variable, and re-use it for subsequent assignments ...

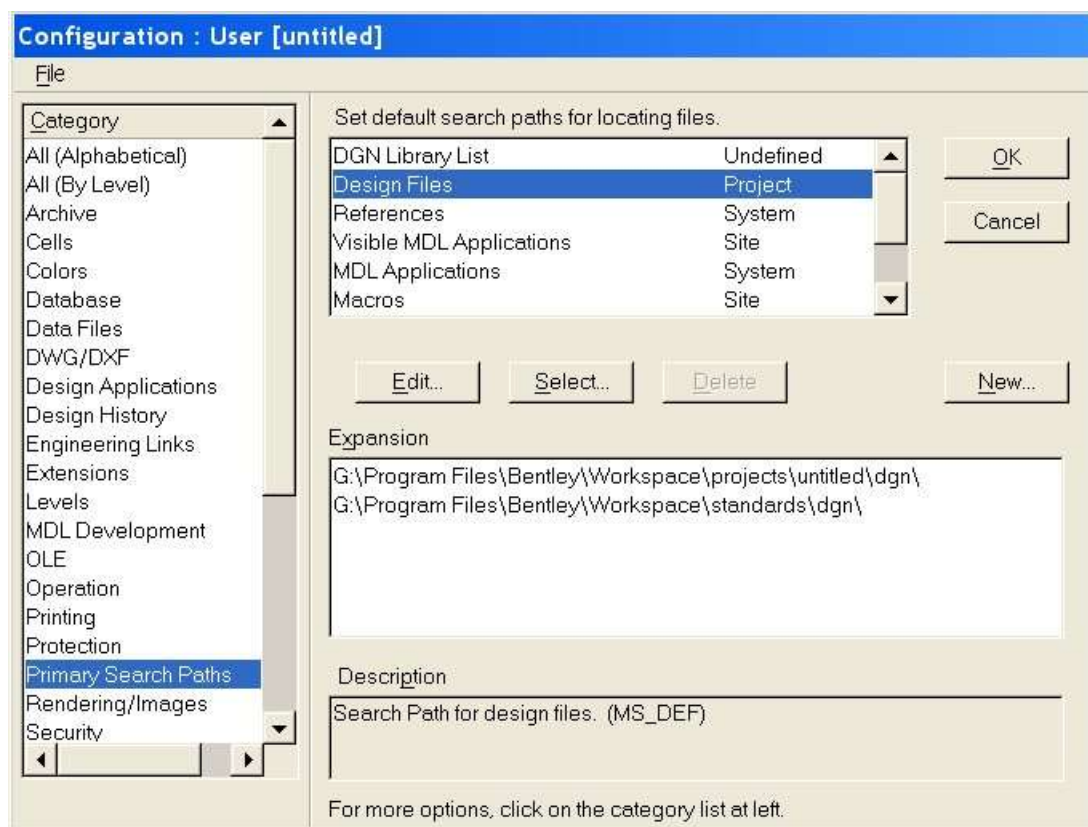
```
PROJECT_ROOT      = C: /Projects/Project01/
MS_DEF            = $(PROJECT_ROOT)dgn/
MS_RFDIR         = $(PROJECT_ROOT)refs/
MS_RFDIR         > $(PROJECT_ROOT)borders/
```

Here, the syntax `$(PROJECT_ROOT)` tells MicroStation to expand the variable before assigning it. The result is the same as before, but it's easier to read. An additional benefit is that, if you decided to relocate the project folders, you simply change the value of `PROJECT_ROOT` to change `MS_DEF` and `MS_RFDIR`.

Often, a multi-user setup stores files on a server computer, so you want to be able to specify a network location for design files and references. Your Windows system administrator may have mapped server folders to local drive letters, in which case you can continue to use the syntax above. Alternatively, you can refer to remote folders using the Universal Naming Convention (UNC) ...

```
MS_RFDIR = //remote-computer-name/projects/
```

You can see the current value of configuration variables in MicroStation's Configuration dialog, opened from the Workspace|Configuration menu ...



You can also start MicroStation in its debug mode to trace configuration variables as they are read and evaluated from the configuration files. Use the `-debug` switch to create an `msdebug.txt` file ...

```
ustation.exe -debug
```

You can include other switches to specify configuration files to be included ...

```
ustation.exe -wuMyConfiguration -wpOurProject -debug
```

The resulting `msdebug.txt` file is too large to be included here. Open `msdebug.txt` with a text editor, and search on a configuration variable to see how its value is obtained while MicroStation evaluates various configuration files.

Another way to see configuration variable values is to use some key-in commands provided for this very purpose ...

```
mdl load cfgvars printCfgVarResource
```

The line above is case-sensitive. This will write two text files to MicroStation directory (`cfgvars.txt` and `cfglong.txt`). These files contain a list of configuration variables and a short description of each.

Configuration variables that begin with an underscore (`_`) are normally not displayed in MicroStation's workspace configuration dialog. You can enable the display of all configuration variables by setting `_USTN_DISPLAYALLCFGVARS = 1` in a configuration file.