Compiling & Downloading DSPL Programs From C

After a DSPL program has been written, the source code text file is fed to a DSPL compiler, which generates a binary .LOD file. This .LOD file is then fed to the download_dspl() routine in MX4NT.DLL, which downloads the binary code to an Octavia. The start_dspl() routine in MX4NT.DLL is used to begin execution of the DSPL program on an Octavia.

RUNNING THE DSPL COMPILER

DSPCG's DSPL compiler is a stand-alone executable named either DSPLC32.EXE or DSPLCO32.EXE, depending on the version. The DSPL source code text file is fed to the compiler as a command-line argument. A second command line argument, "/dsplc_licensed", is included as well, as license verification. So, if the user has a DSPL text file named MY_PROG.HLL, this file can be compiled by running the following at a Windows command prompt:

```
dsplco32 my_prog.hll /dsplc_licensed
```

The DSPL compiler will generate the following files:

DSPLCOK.TMP:
: A blank file whose existence indicates that compilation was completed. The existence of this file does not mean that there were no errors, and it does not indicate that the resulting .LOD file will work. This is only a flag to indicate that the compiler has finished.

DSPLCERR.TMP:
: A blank file which is only created if the compiler encounters errors in the DSPL program. If this file exists when the compiler is started, and no errors are encountered in the source code, the compiler will automatically erase this file.

WINDOWS.TXT:
: A text file which contains the compiler's output text messages. Any errors in the source code will be shown in this file. If this file shows no errors and no warnings, the DSPL file was compiled successfully and the LOD file is ready for download.

MY_PROG.LST:

A text file which contains the DSPL program's source code, the compiler's binary output, and flags indicating the location and type of errors encountered, if any. This is created as a debugging aid.

MY_PROG.LOD:

The binary data which can be downloaded to an Octavia via DSPCG's DLL.

The DSPL compiler can be run from within a Windows application like any other external application. For example, Windows' Shell() function can be used, with the command line shown in the example above as its argument.

DOWNLOADING TO OCTAVIA

Once a .LOD file has been generated, it is downloaded using the download_dspl() function in MX4NT.DLL. Function download_dspl() takes a single argument, a text string which is the complete name of the .LOD file, including any necessary path information. In C, the code looks like this:

```
download_dspl("my_prog.lod");
```

This function returns zero on success. Nonzero return values are error codes, as shown in the header file MX4NT.H.

To start the Octavia executing the DSPL program, call start_dspl(), which takes no arguments. This function also returns zero on success. This function does not block during DSPL execution -- it only triggers DSPL execution and then returns to the caller.

Here're the Visual Basic function and an example program that let you use the DSPL *Compiler* and *downloader* from C.

SHELL is that function:

1. Shell Function

Used in Visual Studio

Runs an executable program and returns an integer containing the program's process ID if it is still running.

```
Public Function Shell( _
   ByVal Pathname As String, _
   Optional ByVal Style As AppWinStyle =
AppWinStyle.MinimizedFocus, _
   Optional ByVal Wait As Boolean = False, _
   Optional ByVal Timeout As Integer = -1 _
) As Integer
```

Parameters

Pathname

Required. **String**. Name of the program to execute, together with any required arguments and command-line switches. *Pathname* can also include the drive and the directory path or folder.

Style

Optional. **AppWinStyle**. A value chosen from the **AppWinStyle** enumeration corresponding to the style of the window in which the program is to be run. If *Style* is omitted, **Shell** uses **AppWinStyle.MinimizedFocus**, which starts the program minimized and with focus.

The *Style* argument can have one of the following settings:

Enumeration value	Description
AppWinStyle.Hide	The window is hidden and focus is given to the hidden window.
AppWinStyle.NormalFocus	The window is given focus and displayed in its most recent size and position.
AppWinStyle.MinimizedFocus	The window is given focus and displayed as an icon.
AppWinStyle.MaximizedFocus	The window is given focus and displayed using the entire screen.
AppWinStyle.NormalNoFocus	The window is set to its most recent size and position. The currently active window remains in focus.
AppWinStyle.MinimizedNoFocus	The window is displayed as an icon. The currently active window remains in focus.

Wait

Optional. **Boolean**. A value indicating whether the **Shell** function should wait for completion of the program. If *Wait* is omitted, **Shell** uses **False**.

Timeout
Optional. **Integer**. The number of milliseconds to wait for completion if *Wait* is **True**. If *Timeout* is omitted, **Shell** uses -1, which means there is no timeout and **Shell** does not return until the program completes. Therefore, if you omit *Timeout* or set it to -1, it is possible that **Shell** might never return control to your program.

2. Example: Perform Real-time DSPL Compilation & Download

```
    sCompiler = sMainWindowPath & "\dsplc32.exe out.hll
/DSPLC_Licensed"
    Shell sCompiler, 0

    sCompiler = sMainWindowPath & "\dsplcok.tmp"
    While ((UCase(Dir(sCompiler))) <> "DSPLCOK.TMP")
        ' Yield to other forms
        DoEvents
    Wend

 sCompiler = "out.lod"
    download_dspl (sCompiler)

' Now start dspl program.

    start_dspl


 reverse_lock = 0
```