Unimal 2.0

Application note 2

Stretching Unimal

Documentation revision 2.00

Techniques:
Extending capabilities using temporary files

MacroExpressions
http://www.macroexpressions.com

An example and a technique of extending Unimal functionality

Imagine you have a need to render a numeric value using various formats. E.g. you want to be able to write something like

```
#MP Expand render("%d", 0x400)
#MP Expand render("%08d", 0x400)
#MP Expand render("%08x", 0x400)
```

And you expect to see the output

```
1024
00001024
00000400
```

Can Unimal do this? At first, it doesn't look that way, because formats must be specified literally and cannot be passed as arguments.

But here is an idea: can't we create a Unimal statement, put it in a file and execute it from there? This can be done indeed (please, see Samples\AppNotes\2\render.u):

```
#MPMacro render ;(format, number)
#MP Export Push
#MP Setstr cmd = {uJoin, "#mp", #1#, "#2#"}
#MP Export (0) "temp"
#mp%scmd
#MP Export Pop
#MP Include "temp"
#MPEndm
```

This is what's happening here.

First, we save (Export Push) the current output stream.

Second, we create a string variable, `cmd`, which contains the Unimal target language interface we want: the target language interface signature (#mp), the actual format (#1#, which is substituted for accordingly), and the formal argument to render as a numeric value (#2#). E.g., in the last macro invocation,

```
#MP Expand render("%08x", 0x400)
```

`cmd` will have the value "#mp%08x#2#"

Third, we switch the output stream to a temporary file; we call it temp. Then we simply render the string `cmd`; it goes to the file temp:

```
>type temp
#mp%08x#2#
```

Fourth, we restore (Export Pop) the previously saved output stream. This of course is necessary by itself, just to continue what we were doing. But it also has a useful side effect: the previous output stream (temp) is closed.

Now we simply include our temporary file which will do the actual rendering. The result of invoking the macro (see render.u) is as desired.