APT360 1.0.2 32-bit Windows Build

The ZIP file contains:

* `apt360.exe`	(release build with VS2008 a/k/a VC9)
* `callpost.bat`	(batch file gateway for invoking postprocessor)
* `mlpost.exe`	(example lathe postprocessor executable)
* `mmpost.exe`	(example mill postprocessor executable)
* `win-apt-doc.pdf`	(this file)

Note the EXE and BAT files were saved into the ZIP file with fully qualified paths.

You should not need any MFC*.DLLs, ATL*.DLLs or MSVCRT*.DLL to run any of the executables.

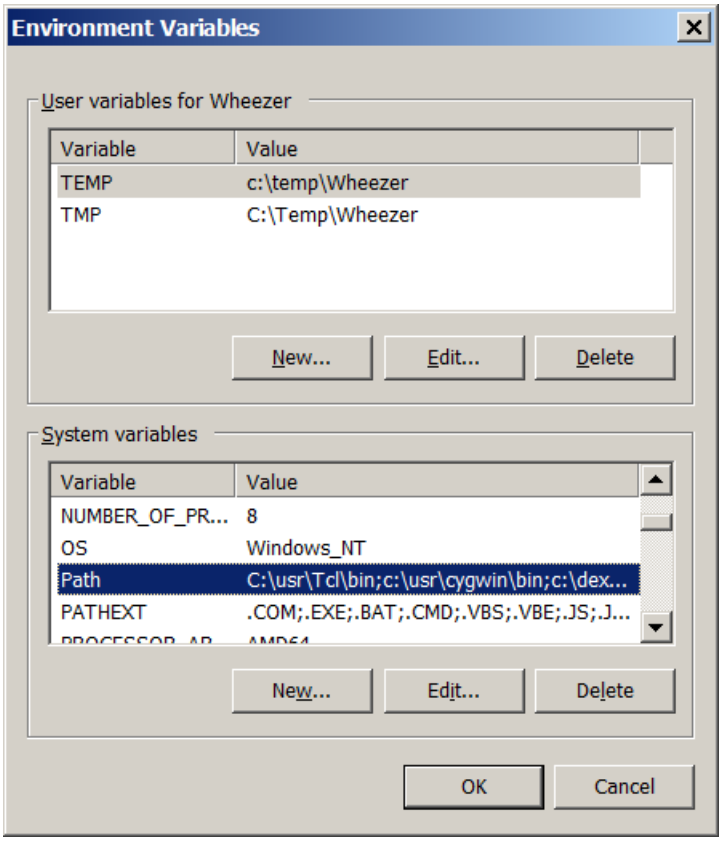
Functionally, this build should sync with the Linux build with two exceptions:

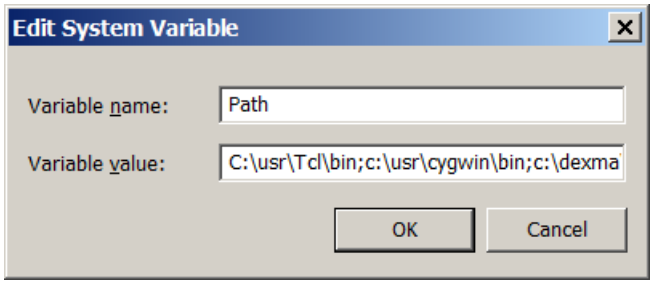
- This version implements an extension: `FEDRAT/ddd.dd,{IPM|IPR}`
- Not built with the GSL library (affects TABCYL).

Installation:

* If you don't have one already, make a folder: `C:\usr\bin`
* Copy(1) contents of ZIP file to `C:\usr\bin`
* If not present already, add `C:\usr\bin` to your **system** `Path` environment variable (2)

(1) - Windows Explorer provides native support for ZIP files.
(2) – Start → Control Panel → System → Advanced system settings → Advanced (tab) → Environment Variables... (button at bottom) → select path in the System variables listbox then click the Edit... button





The EXE files in this ZIP file should not have any DLL dependencies.

Execution:

`apt360.exe` should work just like the Linux version, receiving the APT source input via `stdin` and sending textual output to `stdout` and `stderr`.

All output files (ex. `cl.tap`) are written to the current working directory.

Usage:

* Open command line window (ex. Start → Run... → **`cmd.exe`**)
* Change the current working directory to the folder where the APT source file is located, ex:

```
C:\> cd \src\apt\examples <enter>
```

* Run an APT program(3):

```
C:\src\apt\examples> apt360 < spool2.apt > spool2.lst <enter>
```

(3) - If you do not add **`C:\usr\bin`** *to your system* **`PATH`** *environment variable then you'll have to prefix* `apt360` *with the path name to execute it.*

Postprocessing:

The two postprocessors included with the ZIP file are not warranted as production level code and you use them at your own risk! There is a Python based postprocessor project as part of the APTOS project that has source available that you can experiment with if you want to write your own post.

At this time I am not sure if / when I'll make the source code for the two included posts available though I welcome feedback on them. Though they are fairly generic, the MLPOST is targeted to support a Haas GT-20, MMPOST is targeted to support a Haas SuperMiniMill.

The `apt360.exe` invokes the postprocessor by creating a process that invokes `callpost.bat`, so you need to have `callpost.bat` in your path.

Given the APT source contains:

```
     MACHIN / MMPOST, 1
```

Then `callpost.bat` would then receive:

```
%1 = <current_working_directory>\cl.tap
%2 = MMPOST
%3 = 1
```

The first argument passed will be the fully qualified local path name to the CLFILE, ex:

```
C:\src\apt\examples\cl.tap
```

Building the Windows 32-bit Version

If you do not have VS2008 (or newer) installed on your computer, you can find a free version at Microsoft: Visual C++ 2010 Express.

There is one solution file: apt360.sln
The solution references two project files: libf2c.vcproj and apt360.vcproj

The solution is set up to build libf2c first then apt360.

This version should open the VS2008 based apt360.sln file (it will likely ask you if you want to convert the project to VS2010 to which you can answer 'yes') and build the source into an executable.

Once you have a version of Microsoft Visual Studio installed, you can also build direct from the command line if you wish using:

```
c:\src\aptos\apt360\> msdev apt360.sln <enter>
```

Or you can go through the steps in the IDE to generate an NMAKE compatible makefile from the project files (SLN and VCPROJ) which can the be used to build the executable from the command line using Microsoft's nmake tool. NOTE: You generally need to run vcvars32.bat from the command line first to so the additional environment variables are set for the specific command shell window you are using.

If you prefer to use gcc make, the NMAKE makefile will not be 100% compatible with gcc make, but hopefully you can either adjust the make file manually – or – just generate a viable makefile on your own.