Building Steem on Windows

This document explains how to build the Steem Emulator on the Windows platform using the Steem source trees JLG 32A and JLG 32B that I have released.

This Steem trees compiles directly on Windows platform. The UNIX code has not been updated and therefore probably does not compile directly (however all the UNIX code is still there).

Note that some of the information only apply to JLG32B tree

Source tree organization

The Steem source tree has been kept as in the original tree:

- The ***3rdparty*** directory contains all the sources for the third party libraries. Currently in this directory you will find the Pasti library (in Pasti directory) and the unrar library (in unrarlib directory). This directory is unchanged from original.
- The ***include*** directory contains all the sources files that are somewhat independent of Steem. This directory includes:
  - **asm** directory that contains Steem independent assembly source.
  - **x** directories that contains Unix/Linux sources
  - a directory called **not_used** that contains all the files currently not used in the Windows Steem build.
- The **steem** directory contains the following directories:
  - **asm** directory that contains Steem independent assembly source
  - **code** directory: contains all the Steem sources. This directory also contains
    - **x** directories that contains Unix/Linux sources
    - a **not_used** directory where all the code unused in the current Windows build has been moved.
  - The **doc** directory contains the Steem documentation. The subdirectory called **jlg** contains my added documentation.
  - The **lib** directory: Contains the unzipd32.dll as well as dinput8.lib and dxguid.lib (part of Microsoft DirectX SDK) that are required to build Steem;
  - The **patches,** and **shortcuts** directories contains some files used for release
  - The **rc** directory contains the Steem resources
- The Windows-build directory contains directories and files for building Steem on Windows for different compilers.
- The X-build directory contains file to build Steem on UNIX/LINUX
- The VC2010 This directory contains the files necessary to build Steem on Windows using Microsoft Visual Studio C++ 2010

In the root directory you will also find the Visual C++ 2010 solution files for the Steem project.

Things you need to build Steem using VC++ 2010

To build Steem with visual C++ 2010 you need to have the following installed:

- Visual C++ 2010: You can download the free Microsoft Visual C++ 2010 IDE from http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-cpp-express or use the professional version of VC++.
- The Steem program uses some DirectX libraries and therefore you need to have access to two of these libraries (*dinput8.li*b and *dxguid.lib*). Normally you will need to install the DirectX SDK or at least the required libraries. For example using the following SDK from Microsoft http://www.microsoft.com/download/en/details.aspx?id=6812. However the two required libraries are provided in the **steem/lib** directory of my release tree.
- The **Netwide Assembler**: nasm. Dowload the latest version from http://www.nasm.us/.

- If you want to generate Doxygen documentation (which I recommend) you will need to install it from: http://www.stack.nl/~dimitri/doxygen/ and if you want to generate graphs in the Doxygen documentation you will need the Graphviz package from http://www.graphviz.org/
- Optionally you can use a software revision package. Personally I use the free CS-RCS package from ComponentSoftware Inc http://www.componentsoftware.com/Products/freetool.htm for revision control.

Assembling the Assembly sources

The Steem sources contain some x86 assembly code that need to be compiled with **nasm**. You need to assemble with ***nasm*** the **asm_draw.asm**, **asm_osd_draw.asm**, and **asm_porio.asm** sources. To help you I have added in the **include/asm** directory a batch file, called **make_portio_obj.bat,** and in the **steem/asm** directory a batch file, called **make_draw_obj.bat**. These batch files can be used to create the 3 objects files required to build Steem: **asm_draw.obj**, **asm_osd.obj**, and **asm_portio.obj**. You will eventually need to adjust the set path command in these batch files so that the path points to the directory where you have installed nasm.

Compiling the C++ Sources with VC++

The JLG Steem source tree is provided with Visual C++ solution configuration files. Therefore you can use them to directly build Steem with VC++ 2010.

However for you information I detail the major adjustment you need to do when starting from scratch:

- Create a new solution called Steem (you may want to use the MFC template for that or use empty solution).
- In the solution explorer you will need at least 4 folders:
  - Header Files: In this folder you have to add all the .h files from the code directory as well as the unrarlib.h file from the unrar library and the pasti.h from the Pasti directory.
  - Object Files: In this folder you have to add the 3 .obj files generated with nasm in the asm directory (**asm_draw.obj**, **asm_osd.obj**, and **asm_portio.obj).** You should also add the **dinput8.lib** and **dxguid.lib** DirectX libraries.
  - Ressource Files: Add all the Windows resources files from the **rc** directory
  - Source Files: In this folder you have to add all the .cpp files from the **steem/code** and from the **include** directories as well as the unrarlib.c file from the unrar library.
- Now you need to set the property pages for the project (right click on the project)
  - In the General tab check that the
    - ❖ config type = application (.exe),
    - ❖ Use MFC = Use MFC in shared DLL
    - ❖ Character Set = Use Multi-byte char set
  - In C++ General tab
    - ❖ Additional include directory = ..\include;..\steem\code;..\3rdparty
    - ❖ Warning level = currently I use W3
  - In C++ Preprocessor Definitions you need to have = _VC_BUILD; WIN32
    if you want to build Steem with the integrated debugger add STEEM_DEBUGGER
    Be aware that this has nothing to do with the debug/release version of the program!
  - In C++ precompiled header specify that you are not using them
  - In C++ advanced. Check that **callinf** convention is set to __cdecl (/Gd)
  - In the Linker general tab you may want to add the in the additional Library Directories the location of the DirectX SDK lib
  - In the Linker input tab: additional dependencies =
    kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32.lib;advapi32.lib;shell32.lib;ole32.lib;oleaut32.lib;uuid.lib;odbc32.lib;odbccp32.lib;dinput8.lib;dxguid.lib;winmm.lib;ComCtl32.Lib;%(AdditionalDependencies)

Note the original Steem build was using **dinput.lib** but this lib is no more available in recent DirectX SDK. Using **dinput8.lib** instead seems to work fine.

All the other default options of the VC++ IDE are fine. However feel free to modify them if you know what you are doing.

Once you have correctly set all these properties you should be able to generate the Steem program.

Note that the only drawback of splitting all the sources compared to the original organization is that it takes more time to compile. To generate Steem from scratch on my machine it used to take 7 seconds (JLG32A) and now it takes about 30 seconds (JLG32B).

Major Macro Conditional defines

Some macro defines are used to specify the **compiler** used and the **platform**. I have not tested any of these macros. I have always compiled with _VC_BUILD and WIN32 defined

The following Build Options have been tested:

- STEEM_DEBUGGER Steem program will be built with the Steem Debugger
- NO_ASM_PORTIO no need for the portio asm code
- NO_ASM_DRAW no need for the asm drawing routines (will display slower)
- NO_ASM no need for any asm source
- ENABLE_LOGFILE - Turn logging on (even when debug build is off)
- NO_RAR_SUPPORT - Turn off that lovely RAR code
- ONEGAME (not working currently)
- ENABLE_LOGFILE turn logging on (independently of debug/release flag)

Generating Doxygen Documentation (JLG32B)

Unfortunately Steem contains almost no internal documentation to help developers!

I have therefore started to add some Doxygen information so we can get a minimum documentation.

If you have installed Doxygen using it to generate Steem documentation is easy:

- First in Windows you will need to associate the .dcf extension with the Doxygen GUI Frontend.
- Go to the **steem/doc/jlg** directory. Here you will find a file called steem.dcf (Doxygen Configuration File). Just double click on it to bring the Doxygen GUI. Everything has already been setup for you and therefore you just need to select the **run** tab and click the **run Doxygen.** This will create an html directory in the jlg directory that contains all the documentation in html.
- By default the generation of the graph is not enabled. If you have installed the **Graphviz** package you can turn the graphics on in the configuration file (sensible flags have already been preset).
- If you know what you are doing feel free to modify the Doxygen configuration file.