1 Introduction

This document provides a brief guide to setting up release 4.0.0 of the Harlequin Embedded SDK (HES) source-code distribution and using it to build the various components of the HES, including the Harlequin Embedded RIP.

What follows is an outline of the process for building the Harlequin Embedded RIP from this Harlequin Embedded SDK distribution:

- set up your computer for building the supplied source code
  - requires the listed set of tools to be installed
- copy files from the distribution onto your computer
  - unpack distribution zip file, preserving relative folder layout
- run the supplied setup script
  - unpack source code archive files
  - set up build environment
- run the supplied build script
  - build the Harlequin Embedded SDK components from source.

This distribution is supplied as three separate zip files:

`EmbeddedSDK-4.00-linux.zip`
For use on computers running Ubuntu (Linux)

`EmbeddedSDK-4.00-win32.zip`
For use on computers running MS Windows

`EmbeddedSDK-4.00-netbsd.zip`
For use on computers running NetBSD.

The remainder of this document covers each of these in turn.

Note: In those cases where these zip files have been encrypted, the appropriate password(s) are supplied separately.

For the Windows, Ubuntu and NetBSD procedures for setting up the build environment and running the build process see:

- "Ubuntu procedures" on page 8
- "NetBSD procedures" on page 10
- "Windows procedures" on page 11.

This document also provides instructions for at least installing the VS 2010 swaddin which is distributed in the `.zip` folder. For more information see:

- "Visual Studio Add In for Core RIP debugging" on page 14.

2 Setting up the build computer

The computer on which you will be using the distributions is referred to as the "build computer".

This section provides a suggested configuration for this computer, plus details of any third-party software development tools that need to be installed on it. See:

- "Build computer running Ubuntu" on page 2
- "Build computer running NetBSD" on page 2
- "Build computer running MS Windows" on page 3.

2.1 Build computer running Ubuntu

The following minimum configuration is recommended:

- Intel x86 processor, Xeon or later
- 1 GiB of memory
- 5 GiB of available disk space
- 32-bit Ubuntu 10.04 ("lucid").

The following third-party packages must be installed:

- gcc v4.5 (v4.5.3 recommended)
- g++ v4.5 (v4.5.3 recommended)
- cpp
- Perl 5 (v5.8.6 or later, v5.10 is recommended).

In addition, the "bash" shell and the "en_US" locale should be installed. If the following warning on Ubuntu is displayed:

```
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US)
```

you should perform the following two steps to rectify the issue:

```
$ sudo locale-gen en_US en_US.UTF-8
$ sudo dpkg-reconfigure locales
```

Note: Other Linux OS may use a different procedure.

This, in effect, builds a `en_US` locale capable of building the Harlequin Embedded RIP.

2.2 Build computer running NetBSD

The following minimum configuration is recommended:

- Intel x86 processor, Xeon or later
- 1 GiB of memory
- 5 GiB of available disk space
- NetBSD 6.0 i386.

The following third-party packages must be installed:

- Perl 5
- libexecinfo 1.1

- gmake 4.0.

The following line should be added to the shell startup file:

```
export PERL_BADLANG=0
```

2.3 Build computer running MS Windows

The following minimum configuration is recommended:

- Intel x86 processor, Xeon or later
- 3 GiB of memory
- 5 GiB of available disk space
- 32-bit or 64-bit Microsoft Windows 7 (Professional edition or higher).

Note: This version of the Harlequin Embedded SDK has not been qualified for Windows 8.

The following third-party product must be installed:

- Microsoft Visual Studio 2010, SP1 or later (only the C/C++ language components are required).

In addition:

- the Windows Command Extensions must not be disabled (they are enabled by default, so it is unlikely any action needs to be taken.)
- an English localization (UK or US) should be installed.

3 Unpacking the distribution

Copy the distribution zip archive onto the build computer, using the “linux” “netbsd” or “win32” archive as appropriate. Unpack this file into an empty directory using any convenient tool capable of unzipping standard zip files (e.g. unzip). On Windows you can also use Windows Explorer to do this (*).

Note: The name of the selected directory must not contain any space characters within its path.

The result should be a directory tree of the form:

```
chosen top-level folder
script file(s)
---  build-tools
     --- bin
     --- src
---  product-sources
     zip archive file(s)
---  third-party
     --- bin
```

(*) If the distribution file has been encrypted, the password for accessing its contents will have been supplied separately.

Note: As all of the build processes will be performed within this directory tree, please make sure that there is 5 GiB of space available on the disk being used.

4 Setting up the build environment

Before building the Harlequin Embedded RIP from source you need to establish the build environment. This is done by:

- running one of the supplied script files to unpack the source and create a list of build tools
- ensuring that the list of build tools correctly defines their names and locations.

The list of build tools comprises a set of environment variable definitions which are used within the build process to invoke the relevant tools—hence the importance of ensuring they are set correctly.

To avoid any conflict with existing environment variables all of these names start with `CV_`; these are defined in the toolmap file.

A “CV” variable defines where a product, application or library may be found on your machine.

The build process ultimately invokes a script called `hqmake` that requires the ability to find various products, applications, and libraries on your machine. This is achieved using “CV” variables (“CV” stands for “Configuration Variable”). Naturally, its value is likely to differ from one machine to another (because it depends, on each machine, where applications are installed). That is, “CV” variables describe the configuration of your machine to the `hqmake` build system.

“CV” variable names are carefully versioned so that they do not “outdate”. For example; if you were to build an application which required the use of gcc 2.9.2 and another application which required the use of gcc 4.5, because the “CV” variables are versioned, you can have both gcc 2.9.2 and gcc 4.5 installed on a machine and build both applications without modifying any environment variables. This works because you would have a variable named “CV_GCC_2_9_2” and another called “CV_GCC_4_5” rather than just a single variable called “CV_GCC” meaning that you do not need to constantly change environment variables which “define” where these applications are installed.

You may wish to make a copy or backup of the directory tree after completing this stage so that you can return to this ready-to-build state without having to repeat the process up to this point.

Note: The setup process generates some `.tmp` files which should be included in this copy.

For more information see:

- “Setting up the build environment—Ubuntu” on page 8
- “Setting up the build environment—NetBSD” on page 10
- “Setting up the build environment—Windows” on page 11.

5 Running the build process

Once the toolmap file contents have been checked (see previous section), the actual build process can be run using the `build` or `default-build` scripts.

The build process may be run multiple times without having to repeat the set-up process.

The `default-build` script is provided as a quick start example on how to use the `build` script. This `default-build` script does not take any parameters and simply calls the `build` script with a default set of parameters. The set of parameters used in the `default-build` script will produce a release Harlequin Embedded RIP executable which can be run.

The `build` script is provided as a wrapper around the real build tool `bt/hqmake` and does have required parameters. The command line parameters required are as follows:

```
build <type_of_build> <what_to_build> <-t target_system> <-f font_library>
<build_options>
```

For example, if you were to look in the `default-build.cmd` script you would see the following `build` command line for Windows:

```
build.cmd release depend ebdrip -t win_32-pentium -f ufst5 -va pdls=all
-va minimal=1 -va corefeatures=none -va ICUbuiltin=1 -va
cpp_compiler=msvc_10_0 -va ebd_pdfout=epdfouty -va rebuild=sw
```

If you were to look in the `default-build.sh` script you would see the following `build` command line for Ubuntu:

```
build.sh release depend ebdrip -t linux-pentium -f ufst5 -va pdls=all
-va minimal=1 -va corefeatures=none -va ICUbuiltin=1 -va
cpp_compiler=gcc_4_5_3 -va ebd_pdfout=epdfouty -va rebuild=sw
```

If you were to look in the `default-build.sh` script you would see the following `build` command line for NetBSD:

```
build.sh release depend ebdrip -t netbsd-386 -f ufst5 -va pdls=all
-va minimal=1 -va corefeatures=none -va ICUbuiltin=1 -va
cpp_compiler=gcc_4_5_3 -va ebd_pdfout=epdfouty -va rebuild=sw
```

The `build` parameters which are allowed are:

`type_of_build`

indicates how this build should be compiled and built one of: `release`, `debug`, `asserted`.

Default: `debug`

`what_to_build`

does not need to be specified for normal incremental builds. If you wish to remove any existing built files from the build directories, set this to:

`clean` to remove build files for the specified build type, or:

`clean all` to remove build files for all build types.

Default: `depend ebdrip`

`target_system`

the operating system and hardware on which the built HES will run. For example: `win_32-pentium`, `netbsd-386` or `linux-pentium`.

Default: N/A

`font_library`

The font library to be included in the build one of:

`none`, `ufst5`, `ufst7`.

Default: `ufst5`

Note: `none` can only be used for non PCL builds.

`build_options`

additional build configuration settings of the form:

`-va name=value`

For example: `-va pdls=all`

The additional build configuration settings are documented in "Build configuration options" on page 12.

For more information see:

- "Running the build process—Ubuntu" on page 8
- "Running the build process—NetBSD" on page 10
- "Running the build process—Windows" on page 11.

6 Ubuntu procedures

This is the procedure for setting up the build environment and running the build process.

6.1 Setting up the build environment—Ubuntu

To establish the build environment:

- open a Terminal window on the build computer
- use the cd command to move to the top-level folder for the unpacked distribution
- Enter ./setup-products.sh to run the script file in that folder.

This setup process unpacks the source archive file(s) into the sub folder linux/ebdrip and creates a linux/CV_variables folder containing various tools used by the build.

It also creates a linux/toolmap.sh file containing definitions for the tools to be used when building the Harlequin Embedded RIP. As some of these tools are not included with the distribution itself you should, at this point, check that all of the entries in this file refer to the correct locations on the system. In particular, the following will need to be set by editing the toolmap.sh file:

```
CV_GCC_4_5_3=??            ; export CV_GCC_4_5_3
CV_GCC_4_5_3_CPLUSPLUS=?? ; export CV_GCC_4_5_3_CPLUSPLUS
CV_GCC_4_5_3_CPP=??        ; export CV_GCC_4_5_3_CPP
CV_PERL_5_005_02=??        ; export CV_PERL_5_005_02
```

For example, if on your build computer the appropriate GCC tools have been installed into /usr/bin and the Perl interpreter into /usr/local/bin, you should edit the toolmap file to include:

```
CV_GCC_4_5_3=/usr/bin/gcc-4.5            ; export CV_GCC_4_5_3
CV_GCC_4_5_3_CPLUSPLUS=/usr/bin/g++-4.5 ; export CV_GCC_4_5_3_CPLUSPLUS
CV_GCC_4_5_3_CPP=/usr/bin/cpp-4.5        ; export CV_GCC_4_5_3_CPP
CV_PERL_5_005_02=/usr/local/bin/perl     ; export CV_PERL_5_005_02
```

Because the CV_ environment variables defined in the toolmap file always need to be set to build the product, it is common practice to source this file from your login shell script so that they are automatically set when you login.

For example, you could add the following line to your .bashrc file (if your login shell is bash):

```
. ~/SDK_framework/linux/toolmap.sh
```

Depending on what your login shell is, the file where you add this line will differ.

6.2 Running the build process—Ubuntu

To build the Harlequin Embedded RIP:

- open a Terminal window on the build machine
- use the cd command to move to the top-level folder for the unpacked distribution
- check the current locale by using the locale command:

  if it is not set to en_US, set it with the command: export LC_ALL=en_US

  If the following warning on Ubuntu is displayed:

```
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US)
```

you should perform the following two steps to rectify the issue:

```
$ sudo locale-gen en_US en_US.UTF-8
$ sudo dpkg-reconfigure locales
```

Note: Other Linux OS may use a different procedure.

This, in effect, builds a `en_US` locale capable of building the Harlequin Embedded RIP.

- enter `./default-build.sh` to run the build script file (add option if required).

The SDK buildkit uses the standard GNU tools for building, including `gmake`. This does require some extra setup work as follows:

- the `gmake` utility may not exist on the build computer as `gnu make` is often installed as `make` on Ubuntu systems; if this is the case, add a soft link on your Path that defines `gmake` as referring to `make`
- the compiler is referred to as "gcc" and the linker as "g++" in the generated `makefile`, so again soft links should be added on your Path in order to define these names to be the specific gcc and g++ versions that are used for the main Harlequin Embedded SDK build.

Note: If there are already files named "gcc" and "g++" in, for example, `/usr/bin`, define these new links in a separate directory such as `/usr/local/bin` and ensure this precedes `/usr/bin` in your `PATH` variable.

7 NetBSD procedures

This is the procedure for setting up the build environment and running the build process.

7.1 Setting up the build environment—NetBSD

To establish the build environment:

- open a Terminal window on the build computer
- use the `cd` command to move to the top-level folder for the unpacked distribution
- Enter `./setup-products.sh` to run the script file in that folder.

This setup process unpacks the source archive file(s) into the sub folder `netbsd/ebdrip` and creates a `netbsd/CV_variables` folder containing various tools used by the build.

It also creates a `netbsd/toolmap.sh` file containing definitions for the tools to be used when building the Harlequin Embedded RIP. As some of these tools are not included with the distribution itself you should, at this point, check that all of the entries in this file refer to the correct locations on the system. In particular, the following will need to be set by editing the `toolmap.sh` file:

```
CV_GCC_4_5_3=??            ; export CV_GCC_4_5_3
CV_GCC_4_5_3_CPLUSPLUS=?? ; export CV_GCC_4_5_3_CPLUSPLUS
CV_GCC_4_5_3_CPP=??       ; export CV_GCC_4_5_3_CPP
CV_PERL_5_005_02=??       ; export CV_PERL_5_005_02
```

On NetBSD 6.0, with perl installed in the default package directory, you should edit the toolmap file to include:

```
CV_GCC_4_5_3=/usr/bin/gcc           ; export CV_GCC_4_5_3
CV_GCC_4_5_3_CPLUSPLUS=/usr/bin/g++ ; export CV_GCC_4_5_3_CPLUSPLUS
CV_GCC_4_5_3_CPP=/usr/bin/cpp       ; export CV_GCC_4_5_3_CPP
CV_PERL_5_005_02=/usr/pkg/bin/perl  ; export CV_PERL_5_005_02
```

Because the `CV_` environment variables defined in the toolmap file always need to be set to build the product, it is common practice to source this file from your login shell script so that they are automatically set when you login.

For example, you could add the following line to your `.shrc` file:

```
. ~/SDK_framework/netbsd/toolmap.sh
```

You should also add the following line your `.shrc` file:

```
export PERL_BADLANG=0
```

7.2 Running the build process—NetBSD

To build the Harlequin Embedded RIP:

- open a Terminal window on the build machine
- use the `cd` command to move to the top-level folder for the unpacked distribution
- enter `./default-build.sh` to run the build script file (add option if required).

8 Windows procedures

This is the procedure for setting up the build environment and running the build process.

8.1 Setting up the build environment—Windows

To establish the build environment:

- open a VS 2010 Command Prompt window on the build computer.

  This can be done via the Windows Start menu. `Start\All Programs\Micosoft Visual Studio 2010\Visual Studio Tools\Visual Studio Command Prompt (2010)`. This opens a `cmd.exe` window with the PATH and other environment variables set correctly for the 32-bit MSVC compiler.

  It is important that you select the 32-bit command prompt and not one of the other two x64-bit command environments. This is because, at present, the Harlequin Embedded RIP can only be built as a 32-bit application on Windows.
- use the `cd` command to move to the top-level folder for the unpacked distribution
- enter `setup-products` to run the first of the two script files in that folder.

This setup process will unpack the source archive file(s) into the sub folder `win\ebdrip` and will create a `win\CV_variables` folder containing various tools used by the build.

It will also create a `win\toolmap.cmd` file containing definitions for the tools to be used when building the Harlequin Embedded RIP. As some of these tools are not included with the distribution itself you should, at this point, check that all of the entries in this file refer to the correct locations on the system. So, for example, a line:

```
set CV_MSVCNT_10_0=C:\Program Files\Microsoft Visual Studio 10.0\VC
```

corresponds to the Microsoft Visual C compiler version 10 and this particular definition will only be correct if the Visual Studio 2010 product was installed in its default location (on a Win32 system).

If any of these definitions are incorrect then edit the `toolmap.cmd` file to correct them.

8.2 Running the build process—Windows

To build the Harlequin Embedded RIP:

- open a VS 2010 Command Prompt window on the build computer (as described above).
- use the `cd` command to move to the top-level folder for the unpacked distribution
- Enter `default-build` to run the build script file.

9 Build configuration options

To aid understanding of the build configuration options it is worth providing a general outline of how the Harlequin Embedded RIP build system works in regard to the provided `default-build` and `build` scripts.

The `default-build` script calls the `build` script. These are both convenience scripts provided to get the SDK up and running quickly. Understanding how these convenience scripts work should be relatively simple by reading them. In essence, they construct command line parameters for the `bt/hqmake` build script and then invoke this `bt/hqmake` build script with the constructed parameters.

The `bt/hqmake` build script is the real Harlequin Embedded RIP build system. It is a small wrapper cross platform perl program which in turn invokes the JAM build system (`jam.exe`).

If you prefer you can invoke the `bt/hqmake` command line directly and entirely ignore the `default-build` and `build` scripts, as long as you are in the correct directory, the command line parameters are correct and the `CV_` environment variables have been set in the current shell (For more information see "Running the build process" on page 6).

The JAM build system in essence loads its JAM base from `bt/jambits` and the various component `makefile.jam` files which exist in the source distribution.

Hence you will see `makefile.jam` files in most component directories in the location `make/makefile.jam`. A description of the JAM build system is beyond the scope of this document.

The `bt/hqmake` build command takes parameters in the following form:

```
bt/hqmake  <BuildOptions>  <Variants>  <BuildTargets>
```

For example, invoking the `default-build` script ultimately invokes the command:

```
bt/hqmake +r -a -d -g -t linux-pentium  -va thirdparty -va customised=EBDEVAL
-va security=le -va dll=nd -va rt_libs=multi_threaded -va pdls=all -va
minimal=1 -va corefeatures=none -va ICUbuiltin=1 -va
cpp_compiler=gcc_4_5_3 -va ebd_pdfout=epdfouty -va rebuild=sw depend ebdrip
```

The `build` script therefore shares some of the `<Variants>` and `<Targets>` as `bt/hqmake`.

To explain in detail (lines broken up for easier annotation):

Note: So that you do not need to set them when using the convenience scripts, `build.cmd` and `build.sh` set the following variants for you:

```
-va thirdparty -va customised=EBDEVAL -va security=le -va dll=nd -va
rt_libs=multi_threaded) .

[1] bt/hqmake +r -a -d
[2] -target linux-pentium
[3] -va thirdparty
[4] -va cpp_compiler=gcc_4_5_3
    -va rt_libs=multi_threaded
    -va minimal=1  -va corefeatures=trapping
[5] -va pdls=all
    -va customised=EBDEVAL -va dll=nd -va security=le
    -va ebd_pdfout=epdfouty
[6] -va ebd_ufst5=eufst5y -va ebd_ufst5_fontset=pclps3
[7] -va ICUbuiltin=1-va metrics=no -va rebuild=sw
[8] depend ebdrip distrib
```

Notes:

1. These single-letter options turn general build details on or off (+ or -).

`+r` => release build, fully optimized, no debug data.

`+a` => asserts enabled, otherwise fully optimized, limited debug data.

`+d` => debug build (asserts are not enabled unless +a is also specified).

Note: The `build.cmd` and `build.sh` convenience scripts set the correct combination of these when you set `type_of_build` to one of: `release`, `rel`, `r`, `debug`, `dbg`, `d`, `asserted`, `assert`, `a`.

2. The target computer type for this build, in this case any Intel x86 compatible hardware (pentium class or later) running any linux-compatible operating system; the equivalent for a Windows build would be: `-target win_32-pentium`, and for a NetBSD build it would be: `-target netbsd-386`.

3. This option must always be specified.

Note: The `build.cmd` and `build.sh` convenience scripts set this for you.

4. Compiler to be used.

Note that this value must match the name of the CV variable in the toolmap script file, so for a Windows build this would be: `-va cpp_compiler=msvc_10_0`.

5. List which PDLs will be processed by the RIP from: `ps`, `pcl`, `pdf`, `xps`).

6. Use UFST font library.

Note: The `build.cmd` and `build.sh` convenience scripts set this for you when you set the `font_library` parameter to one of: `none`, `ufst5` or `ufst7`.

7. Do not use Font Fusion library.

8. What to actually build (that is, `<BuildTargets>`):

`depend`	Regenerate `makefile` dependencies.
`ebdrip`	Build Harlequin Embedded RIP (this must always be present).
`clean`	Remove any built object files.

Note: The `build.cmd` and `build.sh` convenience scripts have slightly different options to specify these targets. The build script default is `depend ebdrip`. For more information see "Running the build process" on page 6.

10 Visual Studio Add In for Core RIP debugging

There is a Visual Studio plugin called "swaddin" which when enabled will display the contents of OBJECT, and several other complex core RIP data types in the watch pane, in a human-readable form. You will find "swaddin" useful when debugging the internals of the Harlequin Embedded RIP although it is not necessary to use it.

10.1 How to install

Although the below covers VC 6 to VC 10, future versions of Visual Studio are likely to follow a similar pattern for installation. Note that on a Win64 machine, Visual Studio is likely to be installed in `Program Files (x86)` rather than `Program Files`. This is because the Visual Studio IDE is in fact a 32-bit application.

To install, copy the release plugin from:

```
SDK_framework\swaddin-testsrc\testsrc\swaddin\VS_2010_VC10_bin\Release\swaddin.dll
```

to:

Step 1:

VC6	`C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\`
VC7	`C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\`
VC8	`C:\Program Files\Microsoft Visual Studio 8\Common7\IDE`
VC9	`C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE`
VC10	`C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE`

Step 2:

Find the file `autoexp.dat`, and add the configuration to it:

VC6 Add text to the end of the file:

```
C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\autoexp.dat
```

VC7 Add text to the end of the file:

```
C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\autoexp.dat
```

VC8 Add text to the end of the [AutoExpand] section of:

```
C:\Program Files\Microsoft Visual Studio 8\Common7\Packages\Debugger\autoexp.dat
```

VC9 Add text to the end of the [AutoExpand] section of:

```
C:\Program Files\Microsoft Visual Studio 9.0\Common7\Packages\Debugger\autoexp.dat
```

VC10 Add text to the end of the [AutoExpand] section of:

```
C:\Program Files (x86)\Microsoft Visual Studio
10.0\Common7\Packages\Debugger\autoexp.dat
```

The configuration text to add is:

```
OBJECT            = $ADDIN(swaddin.dll,AddIn_OBJECT)
NAMECACHE         = $ADDIN(swaddin.dll,AddIn_NAME)
LISTOBJECT        = $ADDIN(swaddin.dll,AddIn_LISTOBJECT)
OMATRIX           = $ADDIN(swaddin.dll,AddIn_OMATRIX)
CLINK             = $ADDIN(swaddin.dll,AddIn_CLINK)
GS_CHAINinfo      = $ADDIN(swaddin.dll,AddIn_GS_CHAINinfo)
GS_CRDinfo        = $ADDIN(swaddin.dll,AddIn_GS_CRDinfo)
charcontext_t     = $ADDIN(swaddin.dll,AddIn_charcontext_t)
LINELIST          = $ADDIN(swaddin.dll,AddIn_LINELIST)
sw_datum          = $ADDIN(swaddin.dll,AddIn_sw_datum)
STACK             = $ADDIN(swaddin.dll,AddIn_STACK)
FILELIST          = $ADDIN(swaddin.dll,AddIn_FILELIST)
xmlGIStr          = $ADDIN(swaddin.dll,AddIn_xmlGIStr)
xmlGAttributes    = $ADDIN(swaddin.dll,AddIn_xmlGAttributes)
FONTinfo          = $ADDIN(swaddin.dll,AddIn_FONTinfo)
color_entry_t     = $ADDIN(swaddin.dll,AddIn_color_entry_t)
dl_color_t        = $ADDIN(swaddin.dll,AddIn_dl_color_t)
p_ncolor_t_opaque = $ADDIN(swaddin.dll,AddIn_p_ncolor_t_opaque)
```

10.2 How to use

When looking at supported types in the watch, locals, or auto windows, the value column should display a summary of the object.

10.3 How to build

Should you find yourself needing to extend or change swaddin, open or drag the file:

```
SDK_framework\swaddin-
testsrc\testsrc\swaddin\VS_2010_VC10_bin\Release\swaddin.sln
```

onto MSVC and build a Win32 Release build (this is because Visual Studio is in fact a 32-bit application, even when building and debugging 64-bit applications).

It should compile without problems except for some warnings which can be ignored. For instance:

```
.\swaddin.cpp(455) : warning C4996: 'sprintf' was declared deprecated
C:\Program Files\Microsoft Visual Studio 8\VC\include\stdio.h(345) : see
declaration of 'sprintf'

Message: 'This function or variable may be unsafe. Consider using sprintf_s
instead. To disable deprecation, use _CRT_SECURE_NO_DEPRECATE. See online
help for details.'
```

The resulting DLL will be around 90 KiB.

10.4 Beyond VC 10

Within Global Graphics, Visual Studio 2012 (VC 11) was never used. The decision was made to jump straight to VS 2013 (VC 12).

From VS 2013 (VC 12) Global Graphics primarily switched from using swaddin to using the Natvis framework. This has the advantage of also working with 64-bit architectures. The natvis code is held within the file `SDK_framework\swaddin-testsrc\testsrc\natvis\hqn.natvis`. Documentation on how to install this is held within the file itself. Having said this, swaddin should continue to work in VS 2013 and beyond.

For Natvis documentation see:

http://msdn.microsoft.com/en-us/library/jj620914.aspx

11 Copyright

Harlequin® Embedded SDK—v4.0.0

June 2014

ReadMe

Document issue: 110

========================

The information in this publication is provided for information only and is subject to change without notice. Global Graphics Software Ltd and its affiliates assume no responsibility or liability for any loss or damage that may arise from the use of any information in this publication. The software described in this book is furnished under license and may only be used or copied in accordance with the terms of that license.

Protected by U.S Patents 5,862,253; 6,343,145; 6,330,072; 6,483,524; 6,380,951; 6,755,498; 6,624,908; 6,809,839; 6,755,498; 6,624,908; 6,809,839; 6,996,284; 7,298,526; 7,359,530.

Other U.S. Patents Pending

Protected by European Patents 0 803 160; 0 772 934

Portions Type 1 font renderer contains licensed third party software

The ECI and FOGRA ICC color profiles supplied with this Harlequin RIP are distributed with the kind permission of the ECI (European Color Initiative) and FOGRA respectively, and of Heidelberger Druckmaschinen AG (HEIDELBERG).

The IFRA ICC profiles supplied with this Harlequin RIP are distributed with the kind permission of IFRA and of Gretag-Macbeth.

Harlequin is a trademark of Global Graphics Software Ltd, which may be registered in certain juristrictions. Harlequin ColorPro, Harlequin Dispersed Screening (HDS), Harlequin Precision Screening (HPS), and TrapPro are all trademarks of Global Graphics Software Ltd.

TrueType is a registered trademark of Apple Computer, Inc.

PANTONE® Colors displayed herein may not match PANTONE-identified standards. Consult current PANTONE Color Publications for accurate color. PANTONE® and other Pantone trademarks are the property of Pantone LLC. © Pantone LLC, 2014.

Microsoft, Win32, Windows, Windows NT, Windows Server, Windows Vista, and WinFX are either registered trademarks or trademarks of the Microsoft Corporation in the United States and/or other countries.

Portions include software licensed under the following terms:

Expat - XML parser library

ICU - IBM library providing Unicode and Globalization support

---

Libpng - PNG (Portable Network Graphics) image library

code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the rouines from the library being used are not cryptographic related :-).

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

---

zlib - general purpose compression library

Copyright © 1995-2013 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

Little cms - color management engine

Little cms Copyright © 1998-2004 Marti Maria

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

pthreads-win32 - a POSIX threads library for Microsoft Windows

This file is Copyrighted

The contact addresses for pthreads-win32 is as follows:

Web:http://sources.redhat.com/pthreads-win32

Email: Ross Johnson

Please use: Firstname.Lastname@homemail.com.au

Pthreads-win32 copyrights and exception files

The current list of contributors is contained n the file CONTRIBUTORS included with the source code distribution. The current list of CONTRIBUTORS can also be seen at the following WWW location:

http://sources.redhat.com/pthreads-win32/contributors.html

Contact Email: Ross Johnson

Please use: Firstname.Lastname@homemail.com.au

These files are not covered under one of the Copyrights listed above:

COPYING

COPYING.LIB

tests/rwlock7.c

This file, COPYING, is distributed under the Copyright found at the top of this file. It is important to note that you may distribute verbatim copies of this file but you may not modify this file.

The file COPYING.LIB, which contains a copy of the version 2.1 GNU Lesser General Public License, is itself copyrighted by the Free Software Foundation, Inc. Please note that the Free Software Foundation, Inc. does NOT have a copyright over Pthreads-win32, only the COPYING.LIB that is supplied with pthreads-win32.

The file tests/rwlock7.c is derived from code written by Dave Butenhof for his book 'Programming With POSIX® Threads'. The original code was obtained by free download from his website http://home.earthlink.net/~anneart/family/Threads/source.html and did not contain a copyright or author notice. It is assumed to be freely distributable.

In all cases one may use and distribute these exception files freely. And because one may freely distribute the LGPL covered files, the entire pthreads-win32 source may be freely used and distributed.

General Copyleft and License info

For general information on Copylefts, see:

http://www.gnu.org/copyleft/

For information on GNU Lesser General Public Licenses, see:

http://www.gnu.org/copyleft/lesser.html

http://www.gnu.org/copyleft/lesser.txt

---

sRGB Profile Licensing Agreement:

To anyone who acknowledges that the files "sRGB_IEC61966-2-1_noBPC.icc" and "sRGB_IEC61966-2-1_withBPC.icc" are provided "AS IS" WITH NO EXPRESS OR IMPLIED WARRANTY, permission to use, copy and distribute these file for any purpose is hereby granted without fee, provided that the files are not changed including the HP copyright notice tag, and that the name of Hewlett-Packard Company shall not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Hewlett-Packard Company makes no representations about the suitability of this software for any purpose.

Universal Font Scaling Technology (UFST) Version 7.2

Copyright © 1989 - 2006. All rights reserved, by Monotype Imaging Inc., Woburn, MA.

The software described in this document is furnished under a license agreement with Monotype Imaging Inc., and may be used and copied only in accordance with the terms of such license and with the inclusion of the above copyright notice. This software or any other copies thereof may not be provided or otherwise made available to any other person except as allowed under license. No title to and ownership of this software is hereby transferred.

The information contained in this document is subject to change without notice. Monotype Imaging Inc. MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Monotype Imaging Inc. shall not be liable for errors contained herein for incidental or consequential damages in connection with the furnishings, performance, or use of this material.

This document contains proprietary information protected by copyright. All rights are reserved.

No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Monotype Imaging Inc.

The Intellifont rasterizer is technology covered by US Patent number 4,675,830, issued to Monotype Imaging Inc. The TrueType rasterizer, licensed from Apple Computer, Inc., is optimized and delivered by Monotype Imaging Inc. The MicroType rasterizer is technology covered by US Patents 5,734,388, 5,754,187 with other US and Foreign patents pending. The Type 1 rasterizer, licensed from Pipeline Associates, Inc., is optimized and delivered by Monotype Imaging Inc.

UFST® and Universal Font Scaling Technology® are trademarks of Monotype Imaging Inc., registered in the United States Patent and Trademark Office, and may be registered in certain jurisdictions.

Monotype® is a trademark of Monotype Imaging Inc., registered in the United States Patent and Trademark Office, and may be registered in certain jurisdictions. MicroType™ is a trademark of Monotype Imaging Inc.

Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

The Independent JPEG Group's JPEG software

This software is copyright © 1991-2014, Thomas G. Lane, Guido Vollbeding.

All Rights Reserved except as specified below.

Permission is hereby granted to use, copy, modify, and distribute this software (or portions thereof) for any purpose, without fee, subject to these conditions:

(1) If any part of the source code for this software is distributed, then this README file must be included, with this copyright and no-warranty notice. unaltered; and any additions, deletions, or changes to the original files must be clearly indicated in accompanying documentation.

(2) If only executable code is distributed, then the accompanying documentation must state that “this software is based in part on the work of the Independent JPEG Group”.

(3) Permission for use of this software is granted only if the user accepts full responsibility for any undesirable consequences; the authors accept NO LIABILITY for damages of any kind.

The authors make NO WARRANTY or representation, either express or implied, with respect to this software, its quality, accuracy, merchantability, or fitness for a particular purpose. This software is provided "AS IS", and you, its user, assume the entire risk as to its quality and accuracy.