HUNT ENGINEERING
Chestnut Court, Burton Row,
Brent Knoll, Somerset, TA9 4BP, UK
Tel: (+44) (0)1278 760188,
Fax: (+44) (0)1278 760199,
Email: sales@hunteng.co.uk
http://www.hunteng.co.uk
http://www.hunt-dsp.com

HUNT ENGINEERING

HeartConf

HEART connection configuration tool

USER MANUAL

Software Version 4.11
Document Rev C
P.Warnes 28/05/04

COPYRIGHT

WARRANTIES LIABILITY and INDEMNITIES

HUNT ENGINEERING warrants the hardware to be free from defects in the material and workmanship for 12 months from the date of purchase. Product returned under the terms of the warranty must be returned carriage paid to the main offices of HUNT ENGINEERING situated at BRENT KNOLL Somerset UK, the product will be repaired or replaced at the discretion of HUNT ENGINEERING.

Exclusions - If HUNT ENGINEERING decides that there is any evidence of electrical or mechanical abuse to the hardware, then the customer shall have no recourse to HUNT ENGINEERING or its agents. In such circumstances HUNT ENGINEERING may at its discretion offer to repair the hardware and charge for that repair.

Limitations of Liability - HUNT ENGINEERING makes no warranty as to the fitness of the product for any particular purpose. In no event shall HUNT ENGINEERING'S liability related to the product exceed the purchase fee actually paid by you for the product. Neither HUNT ENGINEERING nor its suppliers shall in any event be liable for any indirect, consequential or financial damages caused by the delivery, use or performance of this product.

Because some states do not allow the exclusion or limitation of incidental or consequential damages or limitation on how long an implied warranty lasts, the above limitations may not apply to you.

TECHNICAL SUPPORT

Technical support for HUNT ENGINEERING products should first be obtained from the comprehensive Support section http://www.hunteng.co.uk/support/index.htm on the HUNT ENGINEERING web site. This includes FAQs, latest product, software and documentation updates etc. Or contact your local supplier - if you are unsure of details please refer to http://www.hunteng.co.uk/ for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing mailto:support@hunteng.co.uk, calling the direct support telephone number +44 (0)1278 760775, or by calling the general number +44 (0)1278 760188 and choosing the technical support option.

TABLE OF CONTENTS

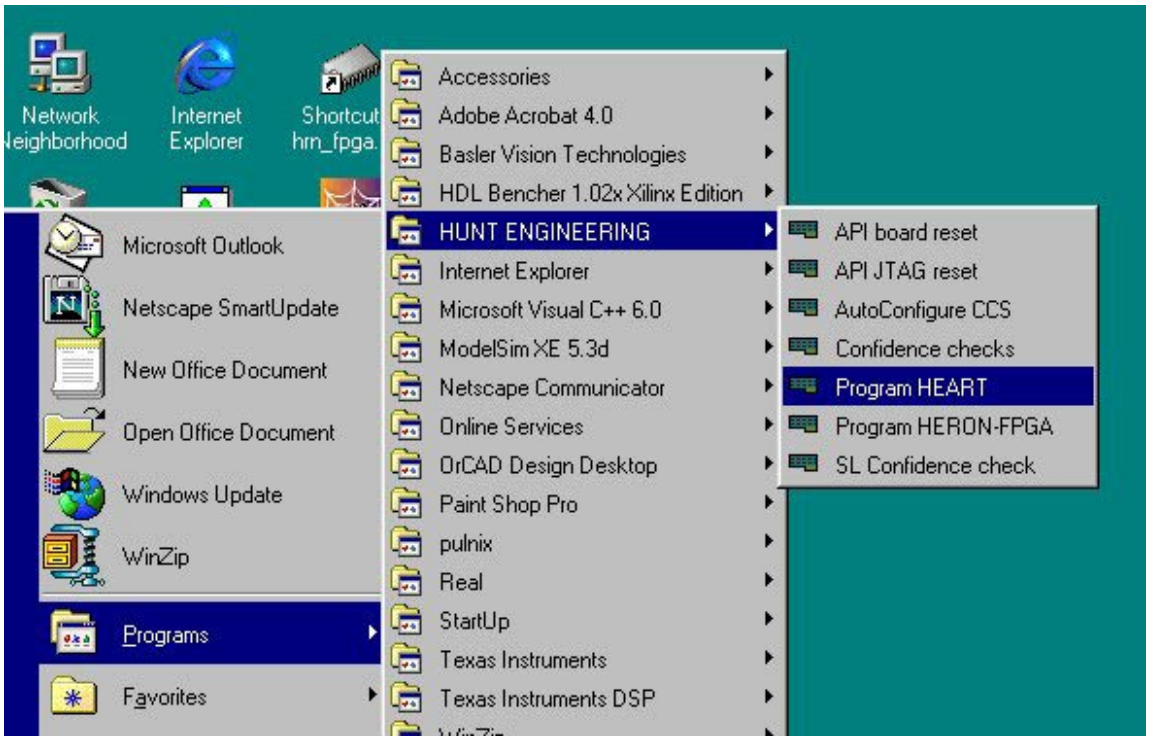
Module carriers, that offer the HEART communications system, use software configurable "virtual FIFO" connections between HERON modules. HEART is an underlying technology that is documented for interested readers in the technology document "Technical description of HEART". However most users do not need to know anything about the underlying technology, but simply how to use it.

The Heartconf utility is provided by HUNT ENGINEERING to make the programming of those "virtual FIFO" connections simple. It uses a common text based file format developed by HUNT ENGINEERING for use by system configuration tools.

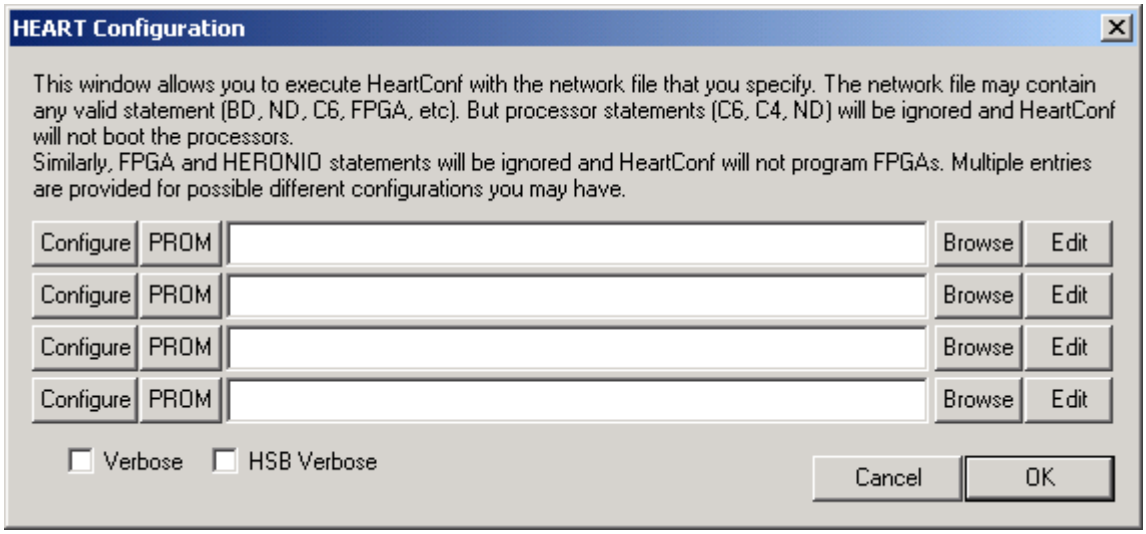
By describing the connections that are required in your system in that text (network) file you can describe to Heartconf how to make the connections. Heartconf itself runs a mapping algorithm to determine first if the connectivity you described is possible, and secondly to determine the correct programming sequence to make those connections for you.

Heartconf is shipped as an executable that is installed on your windows development system during the HUNT ENGINEERING Host API installation. It resides in the %HEAPI_DIR%\heartconf directory, where %HEAPI_DIR% is the location you chose during the installation (default is c:\heapi).

For the windows operating system there is also a windows front end for this program that can be found by accessing programs→ HUNT ENGINEERING → Program HEART



Which when selected gives you: -



There you can select a network file that describes your needs, and use the configure button

to configure the system from that file. Actually this is simply a windows front end that calls heartconf.exe for you with the correct parameters.

The network file can contain the information to configure as many boards as you have in your PC – you do not need to make a separate file for each.

For other supported operating systems like Linux or RTOS32 etc heartconf.exe is also supplied as part of that installation, but the windows front-end programs are not. Here you must provide the correct parameters yourself.

If you simply run heartconf with no parameters it gives you a usage as follows :-

C:\heapi\heartconf>heartconf

HeartConf [PC/32bits] v4.07 (11-2002) HUNT ENGINEERING

Usage is

"C:\HEAPI\HEARTC~1\HEARTC~1.EXE [options] <network_description_file>"

Options:	-r	Reset the target processors.
	-z	Don't erase existing HEART connections.
	-v	Verbose.
	-bx	Where x=0/1/2/3. Creates a VHDL or C file, so that a DSP or FPGA can program HEART by executing the information within that file.

So you need to provide the name of a valid network description file (see next section) and choose if you first reset the system (normally yes), if you undo any existing HEART connections (normally yes – otherwise pre-existing connections could interfere with your connections) and if you want to view verbose messages or not (normally not).

However in a windows environment it is simpler to use the provided windows front ends as shown above.

There are situations, where you don't want or cannot employ HeartConf but you still need to program HEART to create fifo connections. It is possible to ask HeartConf to create a file that, when executed, will program HEART.

When you execute HeartConf with the –b0 option, a C file will be written for use with a host program which, when executed, will program HEART according to the network file you used to produce the file. When you use the –b1 option, a VHDL file fit for use with a Spartan device will be generated. When you use the –b2 option, a VHDL file fit for use with a Virtex II device will be generated. And when you use the –b3 option a C file fit for use with Code Composer Studio and HERON-API will be generated.

The generated VHDL files are really only initialising a buffer (in block ram) and you need some HIL VHDL to actually execute the parameters stored in the buffer.

The C code generated is in the shape of a function.

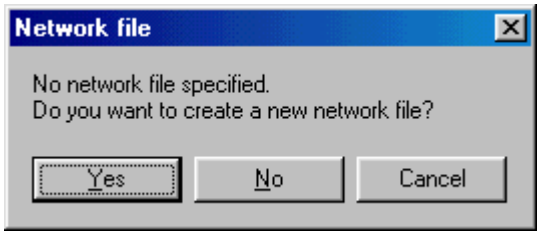
You can also use the Windows front-end explained in the previous Chapter. Select network file and press 'PROM' button.

The network file is a standard file that is an editable text file. It is also used by utilities like the HUNT ENGINEERING Server/Loader.

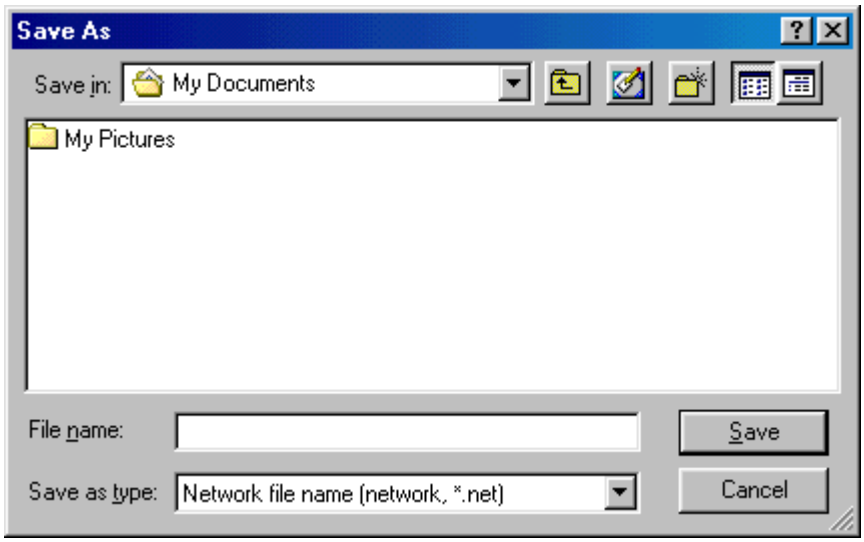
Because it is a standard the syntax is defined in a separate Network File Syntax document that can be found on the HUNT ENGINEERING CD and web site under user manuals.

A useful feature of the windows front end is the edit button

If you want to edit an existing network file that is listed in the tool, you can use the edit button to open that file for editing. If the file description is left empty and that button pressed you will be asked: -

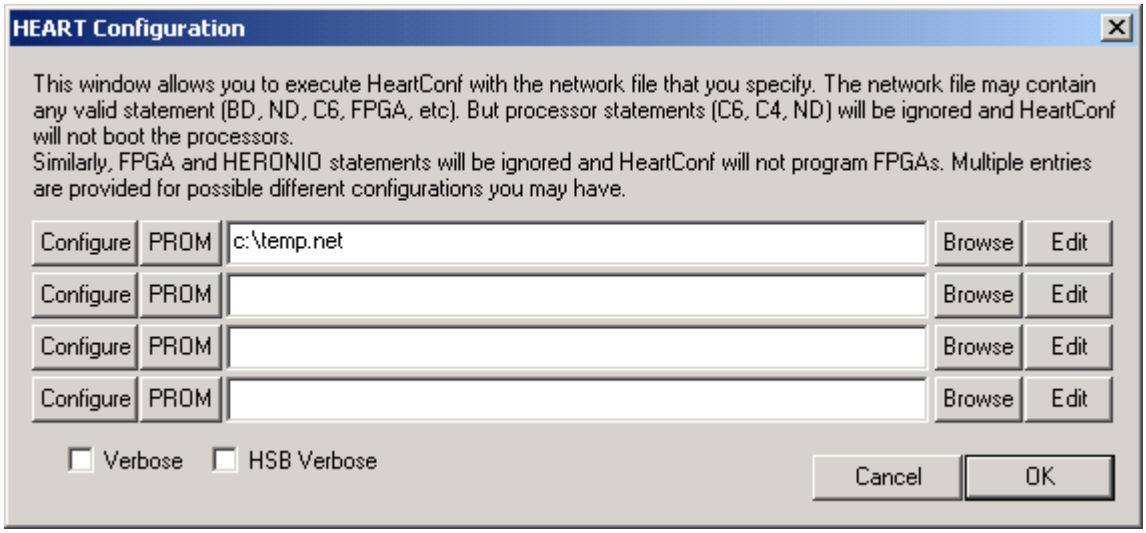


If you answer yes, then you are asked for a filename: -



Select your preferred name and a template network file will be created for you. In the template almost everything is commented out, so you must edit this file to be your correct description and remember to save it.

The filename you have chosen is also filled in for you in the tool: -



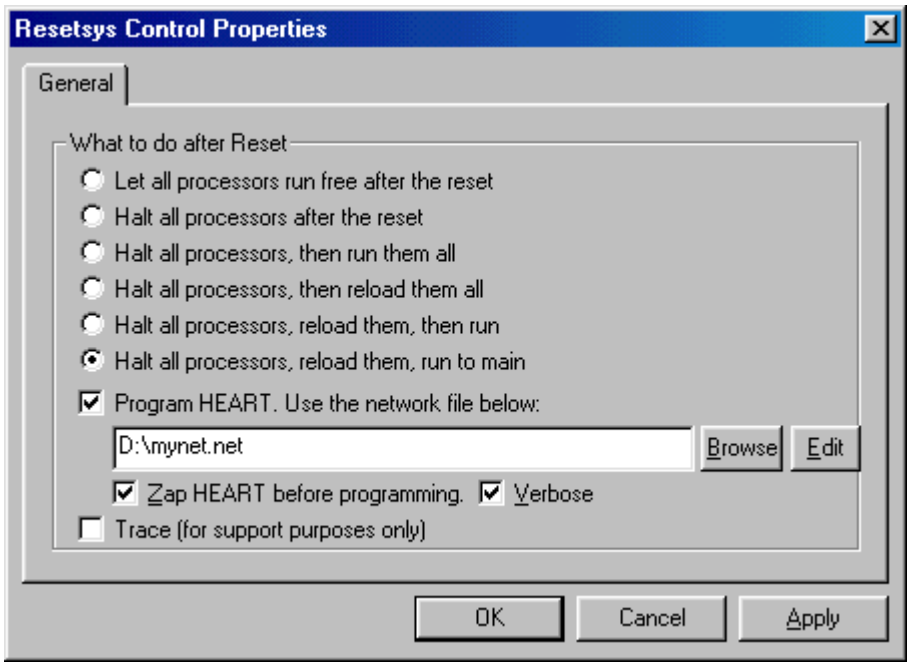
The filenames that you fill in here are remembered for you so that they are still filled in the next time you use the tool.

Because the HEART network is "cleared" by a hardware reset, the connections must be programmed after each hardware reset.

So you always need to use heartconf following a system hardware reset.

If you are developing an FPGA only system, then you will use the "Configure HEART" from the HUNT ENGINEERING tools, after configuring the FPGAs and making the reset.

If you are developing a C6000 application then you will be using Code Composer Studio. To load and test your application you will use the HUNT ENGINEERING reset plug in, which allows you to reset the system and load the processors. There is an option in this tool to also configure HEART using heartconf. In that case click on the background of the Reset plug in (for information about that see its separate user manual). You will see: -



Where you can select program HEART and the network file that you want to use.

If you are developing or using a Server/Loader project (see separate manuals) then the same network file is used for that. Using the Server/Loader plug in for Code Composer or the other versions of the Server/Loader (command line or library) the Server/Loader will load HEART itself using the information in the network file, and it is not necessary to separately use Heartconf.

When you develop a system that contains both FPGA and C6000 you can choose which method to use according to the part you are developing at the time.

HeartConf and Linux

With Linux, HeartConf is available as a command-line utility. It has the same options and features as the standard Windows version as explained in the 'how do I use it' section.

HeartConf and VxWorks

With VxWorks, HeartConf is available as a command-line utility. It has the same options and features as the standard Windows version as explained in the 'how do I use it' section.

HeartConf is part of the 'vxwsl.o' file, and to use it, you would have to have 'heapi.o' and 'hrnfpga.o' loaded first: -

```
ld<heapi.o
ld<hrnfpga.o
ld<vxwsl.o
```

HeartConf can then be called, for example, as follows: -

```
sp HeartConf,"-v network"
```

where the options are identical to those explained in the 'how do I use it' section.

HeartConf and RTOS-32

With RTOS-32, HeartConf is available as a command-line utility. It has the same options and features as the standard Windows version as explained in the 'how do I use it' section.

HeartConf for RTOS-32 is located in heartconf\rtos32 in your API&Tools installation directory. Both an executable 'HeartConf.exe' and 'HeartConf.rtb' are provided in this directory, as well as configuration files. Use the RTB file to create a floppy disk: -

```
bootdisk heartconf a:
```

and copy the network file (and accompanying *.out and *.hcb or *.rbt files) onto the floppy, or onto the harddisk of the target machine. You can simply run 'heartconf' off the floppy-disk, and it will read the network file and load *.out files and bit-streams as required.

The CommandLine, as used in 'heartconf.cfg', assumes a network file on a:\, as follows:

```
Commandline "a:\heartconf.exe –v a:\network"
```

The configuration files are there, so you can make changes here as required.

HeartConf is also 'embedded' in the Server/Loader library. If you have purchased the Server/Loader you can also call HeartConf and Server/Loader with library calls from within your host C or C++ application. Please refer to the Server/Loader documentation for more details.

Overview

It is possible to add HeartConf functionality to your PC program. HeartConf is available as a library. For win32 systems, an include file (hc.h) plus lib file (hc.lib for Microsoft C/C++, hcbl.lib for Borland C/C++) are in your API&Tools installation directory.

The 'hc.h' interface should only be used when no SDP (Software Developer's Pack) has been installed. When the SDP is installed, please use 'hesl.h' instead of 'hc.h'. Both include files define the same class, 'hesl', but the class defined in 'hesl.h' has more functions defined, and it includes the functions defined in 'hc.h'. In the Server/Loader manual it is further explained how to use the 'hesl.h' include file and how to use its interface.

However, applications built with 'hc.h' will still work when the SDP is installed. It is when you wish to use both HeartConf and Server/Loader functions that you should use 'hesl.h' and not 'hc.h'. Both include files define an 'hesl' class, and would otherwise clash.

HeartConf functions

The library form of HeartConf comes in 4 'shapes': -

```
int  heartconf(int   argc   , char *argv[] );
int  heartconf(char *options, char *network);
int  heartconf(HE_HANDLE *uDevice, int n, int   argc   , char *argv[] );
int  heartconf(HE_HANDLE *uDevice, int n, char *options, char *network);
```

The top `heartconf` entry has two parameters: an `argc` parameter that is the number of `argv` arguments, and the `argv` parameter itself, which is an array of character string pointers. This is identical to the arguments used by `main()` in a console program. The possible arguments in 'argv' are the same ones as defined in the "How Do I Use It" chapter (page 7): "-r" (reset), "-v" (verbose), "-z" (no zap) and "-b0/1/2/3".

Example.

```
int main(int argc, char *argv)
{
      hesl sl;
      int r = sl.heartconf(argc, argv);
      …
}
```

The second `heartconf` entry allows all options to be specified in one string, and the network file in another string. For example: -

```
int r = sl.heartconf("-r", "network");
```

In some applications you may already have 1 or more handles open to devices (such as a fifo or HSB). As HeartConf may also access those same devices, you would have to close all devices before calling HeartConf. An alternative is offered by the handle functions. Here, you can simply pass all open handles as a parameter (array). Before opening any device, HeartConf will first check if that device is listed in the open handle list provided by you. If so, the handle you provided will be used. If not, then HeartConf will try to open that

device. Note that HeartConf may use multiple handles, for example, both Fifo A and HSB. In general it depends on what is defined in the network file what devices it needs to open.

Example: -

```
int main(int argc, char *argv)
{
      hesl sl; HE_DWORD Status; int r;
      HE_HANDLE hDevice[2]={NULL,NULL};
      Status = HeOpen("hep9a", 0, HSB, &hDevice[0]);
      if (Status!=HE_OK)
      {
            printf("Open error %x\n", Status); return 0;
      }
      r = sl.heartconf(hDevice, 1, argc, argv);
      …
}
```

The return value for each of the 4 `heartconf` functions is 0 upon success. Upon error a value other than 0 is returned. With the 'getlasterr' function you can retrieve a description of the error encountered by HeartConf.

Example: -

```
int main(int argc, char *argv[])
{
      int   r;
      char *errstr;
      hesl  sl;
      r = sl.heartconf("-rv", argv[1]);
      if (r)
      {
            errstr = sl.getlasterr();
            if ((errstr==NULL)||(errstr[0]==0))
            {
                  printf("SL reports error %d.\n", r);
            }
            else
            {
                  printf(errstr);
            }
      }
      return r;
}
```

Parsing the network file

With the `heartconf` functions, there's no need for a separate parse network function. But in some cases you may only want or need to parse the network file in order to extract some information. That's when this function can be used.

Example: -

```
int main(int argc, char *argv)
{
      hesl sl;
      r = sl.parse_network_file("network");
      if (r)
```

```
        {
                errstr = sl.getlasterr();
                if ((errstr==NULL)||(errstr[0]==0))
                {
                        printf("SL reports error %d.\n", r);
                } else {
                        printf(errstr);
                }
        }
        …
}
```

Windows (and other non-console) programs

When using the verbose ("-v") option, progress information is printed to stdout. There is also the option to 'redirect' verbose print messages. HeartConf allows you to replace the functions it uses to print by another function, a function that you have written yourself.

To replace the verbose print option, you can use: -

```
void set_user_vprint(USER_VBSFUNC fie);
```

where the function prototype is defined as follows: -

```
typedef void (*USER_VBSFUNC)(char *str);
```

Example: -

```
void myprint(char *str)
{
      print_to_window(somehWnd, str);
}
int main(int argc, char *arvg[]
{
   hesl sl; int r;
   sl.set_user_vprint(myprint);
   r = sl.loader("-rlv", "network");
}
```

The `print_to_window` function would be a function you defined yourself writing text to a window in your Windows application.

Error handling

For all functions, the return value is 0 upon success. A return value of larger than 1 means some error was encountered. A return value of 1 is used to indicate failure. Failure doesn't mean error, for example, if you ask if a board is remote, replying 'no' means the board isn't remote, not that an error has occurred.

If an error has occurred, use the 'getlasterr' function to retrieve a descriptive string of the error. The function should always return a non-NULL, non-zero-length string, but to be sure you would want to verify that before attempting to display the string.

Version Information

The 'hesl' class has the option of retrieving version information. This may be helpful for

detecting features or functions introduced in later versions of HeartConf. Example:

```
int    major, minor;
char  *verstr=NULL;
hesl   h;

h.version(&major, &minor, &verstr);
if (verstr!=NULL)
{
      printf("HeartConf version %s.\n", verstr);
}
else
{
      printf("SL version %d.%d.\n", major, minor);
}
```

How to build

In the API&Tools installation directory (typically c:\heapi if you installed into the default directory) there is an include file 'hc.h'. This file uses some definitions from the API, so usually you would need to include both files as follows: -

```
#include "heapi.h"
#include "hc.h"
```

Note that if you have also installed the SDP (Software Developer's Pack) you should use the 'hesl.h' include file instead of 'hc.h'. The 'hesl.h' interface defines the same 'hesl' class but with a larger set of functions, which also include all functions defined in 'hc.h'. Please refer to the Server/Loader manual to learn how to use the 'hesl.h' interface. Include: -

```
#include "heapi.h"
#include "hesl.h"
```

The HeartConf library is multi-threaded, and you must set the proper switches in your project. For example, in Microsoft C/C++ you go to Project → Options → C/C++ → Code Generation, set field "use run-time library" to "multi-threaded".

The "heapi.h" file has definitions that may be different between different platforms, or even between different compilers. Environment/Compiler variables are used to select the proper definitions.

If you build for a WIN32 platform with a 32-bit Microsoft Visual C/C++ compiler then no specific environment/compiler options need to be set.

For VxWorks, _VXWORKS must be defined and be 1.

For LINUX, _LINUX must be defined and be 1.

For RTOS-32, _RTOS32 must be defined and be 1.

Linking with Libraries

With a 32-bit Microsoft Visual C/C++ compiler you should link with the `hc.lib` library. To run your application, make sure that the `win32sl.dll` file is located in the system directory of your operating system (Windows 95/98: c:\windows\system, and in Windows NT/W2K: c:\winnt\system32). The API&Tools installation should have done this for you. When you have the SDP (Software Developer's Pack) installed, please link with

`win32sl.lib` instead.

With a 32-bit Borland C/C++ compiler you should link with the `win32slbl.lib` library. To run your application, make sure that the `win32slbl.dll` file is located in the system directory of your operating system (Windows 95/98: c:\windows\system, and in Windows NT/W2K: c:\winnt\system32). The API&Tools installation should have done this for you. When you have the SDP (Software Developer's Pack) installed, please link with `win32slbl.lib` instead.

With LINUX you should link with `liblinuxsl.so` which should have been deposited in /usr/local/lib by the installation script.

With VxWorks there's no need to link with a library if the 'vxwsl.o' has already been loaded separately onto the system. If this is not the case, either link with 'vxwsl.o' or 'vxwsllib.o'. The former, 'vxwsl.o', contains the Server/Loader executable, HeartConf as well as the Server/Loader library.

With RTOS-32, you should link with 'rtossl.lib', which should be located in the hesl\lib\rtos32 sub-directory of your API&Tools installation.

In your API&Tools installation directory (c:\heapi if you used the default installation directory) there's a 'heartconf' sub-directory. In here is the 'heartconf.cpp' file. This shows how the main form of the HeartConf library can be used.

Technical support for HUNT ENGINEERING products should first be obtained from the comprehensive Support section http://www.hunteng.co.uk/support/index.htm on the HUNT ENGINEERING web site. This includes FAQs, latest product, software and documentation updates etc. Or contact your local supplier - if you are unsure of details please refer to http://www.hunteng.co.uk for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing support@hunteng.co.uk, calling the direct support telephone number +44 (0)1278 760775, or by calling the general number +44 (0)1278 760188 and choosing the technical support option.