



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

SL/API (threads) Server/Loader

Example

For VxWorks

Document Rev A
Server/Loader SL/API (threads) Example Rev 4.10
J.Thie 05-01-04

COPYRIGHT

This documentation and the product it is supplied with are Copyright HUNT ENGINEERING 1999. All rights reserved. HUNT ENGINEERING maintains a policy of continual product development and hence reserves the right to change product specification without prior warning.

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 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 www.hunteng.co.uk for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing support@hunteng.demon.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

THE THREADS EXAMPLE	4
COMPILING, LINKING AND RUNNING THE EXAMPLE	5
COMPILING/LINKING THE EXAMPLE	5
RUNNING THE EXAMPLE	5
EX1 AND EX2.....	6
COMMAND LINE.....	7
THE SERVER/LOADER COMMAND LINE.....	7
VXWORKS NETWORK FILE	8
THE VXWORKS NETWORK FILE	8
MAKEFILE	9
INCLUDE FILE	9
LIBRARIES	9
COMPILE PARAMETERS	9
LEGACY COMPILE PARAMETERS	10
TECHNICAL SUPPORT	11

The SL/API (threads) example shows how Server/Loader and the API program can be combined into 1 application. It also shows how Server/Loader and the API program can be run in parallel, using separate threads. One thread uses the Server/Loader library to boot and then serve the system. Another thread accesses the same system (but uses a different host fifo) and implements its own communications with the system. The system in this example is only one HERON module on one board, but the example can equally well be used with multiple HERON modules and multiple boards.

In this case, Server/Loader and API program are used in 1 combined application. It is also possible to run Server/Loader and API program as separate, parallel, programs. This is shown in the SL/API (parallel) example. The example is located in the '..\parallel' directory.

(This example will **not** work with TIM-40 carrier boards such as the HEPC2E, HEPC3, HEPC4 or HECPCI1. It will also **not** work with the HEPC6, a one 'C6x processor board.)

Compiling, linking and running the example

Compiling/Linking the Example

The Server/Loader is delivered as a 'vxwsl.o' file. This file contains both the Server/Loader library ('hesl' interface), the Server/Loader executable ('vxwsl') and HeartConf. The file is located in the 'vxworks' directory of your HUNT ENGINEERING API installation (default 'c:\heapi').

The 3 components in 'vxwsl.o' are also available separately as 'main.o' (Server/Loader executable) in the 'hesl\bin' sub-directory of your Server/Loader installation, 'vxwsl.lib.o' (Server/Loader library) in the 'hesl\lib' sub-directory, and 'heartconf.o' (HeartConf) in the 'heartconf\vxworks' sub-directory.

An environment variable 'HESL_DIR' points to the 'hesl' installation sub-directory. 'HESL_DIR' has been created and initialised by the HUNT CD installation program. Include files are located in the 'inc' directory of 'HESL_DIR'.

To build your own 'mysl.o', you can use the 'make.sl.bat' batch file (which uses a Makefile), also present in the example directory: In a DOS-box, in the example directory, type:

```
make.sl
```

However, the Makefile assumes HEAPI_DIR to be 'c:\heapi' and HESL_DIR to be 'c:\heapi\hesl'. If this is not the case, you must edit the Makefile and change HEAPI_DIR and HESL_DIR to your installation directory.

Running the example

The threads application needs file access to the network file and the DSP executables (the *.out files) and the 'stdio.c' file. Copy the network file, the *.out files, and 'stdio.c' onto a floppy disk, or copy them onto a hard disk if you have a VxWorks boot image with support for that. To be able to run the example successfully you must have included the dosfs module in your VxWorks BOOT ROM configuration. Set your default path to the location of the *.out files using the VxWorks system command `ioDefPathSet("location")`, where `location` is the VxWorks style path to the *.out file.

Make sure you have loaded the API, hrn_fpga, vxwsl, and threads.

```
ld<heapi.o
ld<hrn_fpga.o
ld<vxwsl.o
ld<threads.o
```

The reason for loading hrn_fpga.o as well is that the Server/Loader library supports loading of FPGA bit-streams. But the implementation of the Server/Loader library uses hrn_fpga to do the actual loading. Given that hrn_fpga is also a stand-alone utility, we have chosen to supply the Server/Loader library and hrn_fpga as two separate items.

Next, for a HEPC9, run the example as follows:

```
sp threads, "-v"
```

The example assumes a HERON4 in slot 1. If you don't use a HERON4 module, but a different HERON module such as HERON1, you will have to change the *.out file used in the network file. Some standard *.out files are supplied: test4.out (HERON4 on HEPC9)

and test2.out (HERON2 on HEPC9). For any other configuration, create a new project and build a new *.out file.

You should see something that ends like:

```
...
Write word. This will make the config light flash.
Message received was abcdef
End.
```

Ex1 and Ex2

Ex1 and ex2 are slight variations of the threads example. In the standard example, 'threads.cpp', we use hesl's loader, server and semaphore flag functions: first we create a semaphore and boot/load the system. Then we create a thread in which an API program is set to run – first waiting for the semaphore to be set. Once the thread is up and running, we run hesl's server function. The server function starts up all server threads (one thread for each fifo for each node that asked to be served). Once that is done, the server signals the semaphore, and this signals the API thread that it can start to run.

The ex1 example is slightly different, in ex1.cpp no semaphore is used; otherwise ex1 is identical to the standard threads example. Using no semaphore is all right as long as you guarantee that the API thread/program doesn't start communicating with a node until all nodes have been properly booted. Using hesl's loader and server functions, and starting the API thread after a successful loader execution guarantees that this is the case.

The ex2 example shows a case where you must use a semaphore. In ex2.cpp hesl's serverloader function is used, not the separate loader and server functions. In this case, the semaphore is the only way we have to know when it's safe to start communicating with nodes in the system.

The Server/Loader command line

The Server/Loader uses a command line so that a user can specify the name of a network file and a number of parameters. The most common parameters are `-r`, (reset), `-l` (load), `-s` (serve) and `-v` (verbose), but there are others as well (please have a look at the Server/Loader manual). The VxWorks Server/Loader has a default command line of:

```
sp vxwsl, "-rlsv networkfile"
```

With this command line the Server/Loader will expect to find a network description file on the drive specified by `ioDefPathSet`. (In addition, it will then expect to find the `*.out` files and bit-streams as specified in the network file on the same drive.) By default, this will reset the system, boot all processors, and then serve standard I/O requests (`printf`, `frwrite`, etc) coming from the first processor in the system. The verbose option will cause booting information to be show on the screen.

The VxWorks network file

Note that a VxWorks network file uses a few extra parameters in board definitions.

The usual way to define a board, for example a HEPC9, you would write:

```
BD API HEP9A 0 0
```

But for VxWorks you need to add three parameters:

```
BD API HEP9A 0 0 on on 12
```

The three extra parameters need to be there for any board type, whether 'hep9a', 'hep8a', 'hep3b', 'hep2e' or any other. The first extra parameter is the master mode switch, "on" in this example. The second extra parameter is interrupts, "on" in this example. The third extra parameter is the IRQ, "12" in this example. The extra third parameter is not used with PCI boards, such as the HEPC9, HEPC8 and HEPC3. But the syntax requires there's a value anyway.

Note that with an HEPC8, master mode is not supported, so you would define: -

```
BD API HEP8A 0 0 off on 12
```

Apart from this, a VxWorks network file is identical to the 'standard' network file.

Include file

Programs using the Server/Loader library must know where the 'heapi.h' and 'hesl.h' include files are located. In the Makefile we define two variables:

```
HEAPI_DIR      = c:/heapi
HESL_DIR       = c:/heapi/hesl
```

The HEAPI_DIR and HESL_DIR variables are then used in the options list to make sure the compiler finds the include files in the appropriate directory:

```
ARGS = (options) -I$(HEAPI_DIR) -I$(HESL_DIR)/inc (options)
```

You have to make sure that HEAPI_DIR is equal to your HUNT ENGINEERING API&Tools installation directory. Edit the makefile to do so, if your installation directory is not 'c:\heapi'. HESL_DIR is the 'hesl' sub-directory of HEAPI_DIR. The installation program will have created environment variables HEAPI_DIR and HESL_DIR pointing at the installation directory.

Libraries

There are several ways to use programs using the Server/Loader library. First, you can simply load the HUNT ENGINEERING API ('heapi.o'), hrn_fpga ('hrnfpga.o') and the Server/Loader bundle ('vxwsl.o'), and then load the 'mysl.o' program. When doing this, there's no need to link 'mysl.o' with any library. This method is used with the supplied example Makefile.

Alternatively, you could link 'mysl.o' with the Server/Loader bundle 'vxwsl.o'. For example, you could change the Makefile linker entry to: -

```
mysl : $(objects)
      $(ld) -o mysl.o -r $(objects) $(HEAPI_DIR)/vxwsl.o
```

The 'vxwsl.o' file also contains the Server/Loader executable 'stub' and the HeartConf 'stub'. If you don't want this with your executable, you can link with just the Server/Loader library, as follows: -

```
mysl : $(objects)
      $(ld) -o mysl.o -r $(objects) $(HESL_DIR)/lib/vxworks/vxwsl.lib.o
```

When explicitly linked in the above way, to run the resulting executable, there's no longer a need to first load the 'vxwsl.o' library. Naturally, you can extend this and also link hrn_fpga and the API into a single executable. However, in this example we have chosen to use and load the different parts ('heapi.o', 'hrnfpga.o', 'vxwsl.o' and 'mysl.o') separately.

Compile Parameters

The Hunt Engineering Server/Loader support several different types of Operating System. The 'heapi.h' and 'hesl.h' include files have a compiler option that need to be selected properly for a particular Operating System. For VxWorks, compiler option _VXWORKS need to be set to 1.

```
ARGS = (options) -D_VXWORKS
```

Legacy Compile Parameters

The legacy interface of the Server/Loader library uses 'network.h', 'common.h', and 'ccif.h' (which is only actually used on Windows systems). The 'network.h' file is different based on two environment variables CMDLINE and PC. The actual choice of CMDLINE and PC depends on the Operating System of choice. For VxWorks, both need to be set to 1.

In addition, the `_VXWORKS` variable needs to be set in order to select the correct Operating System. Thus, for VxWorks, we use compiler options `_VXWORKS`, `CMDLINE` and `PC`, and they need to be set to 1: -

```
ARGS = (options) -D_VXWORKS -DCMDLINE -DPC
```

The legacy interface ('network.h') should only be used for existing applications that use 'network.h', 'common.h' and/or 'ccif.h'. If you create a new application that will use the Server/Loader library, you are strongly recommended to use the 'hesl.h' interface.

1. Technical support for HUNT ENGINEERING products should first be obtained from the comprehensive Support section 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 www.hunteng.co.uk for the list of current re-sellers.
2. 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.