



**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 RTOS-32***

***Document Rev A***  
***Server/Loader SL/API (threads) Example Rev 4.11***  
***J.Thie 27-01-04***

## **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](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 [www.hunteng.co.uk](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](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

<b>THE THREADS EXAMPLE .....</b>	<b>4</b>
<b>COMPILING, LINKING AND RUNNING THE EXAMPLE .....</b>	<b>5</b>
COMPILING/LINKING THE EXAMPLE .....	5
RUNNING THE EXAMPLE .....	5
EX1 AND EX2.....	5
<b>MAKEFILE .....</b>	<b>7</b>
THE MAKEFILE.....	7
INCLUDE FILE .....	7
LIBRARIES .....	7
COMPILE PARAMETERS .....	8
INITIALISATION .....	8
<b>CONFIGURATION FILES .....</b>	<b>9</b>
THE THREADS.CFG CONFIGURATION FILE.....	9
THREADS.CFG: COMMANDLINE .....	9
THREADS.CFG: FLOPPY ACCESS.....	9
<b>TECHNICAL SUPPORT .....</b>	<b>10</b>

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

To compile and link the example, please use the 'makefile' that is present in this directory. This makefile is set-up to use a Microsoft 32-bit C/C++ compiler. You can only execute the makefile in a DOS-box prepared by On Time for any of the Microsoft C/C++ command line examples ('Visual C++ (Command Line) Demos'). Or, if you execute from a standard DOS-box, please execute the standard RTOS-32 'varsvc.bat' first.

To execute the 'makefile':

```
nmake
```

### Running the example

To run the example, prepare a floppy disk and insert it into the 'a:' drive. Then type:

```
bootdisk threads a:
```

Next, copy the \*.out files to the floppy disk as well:

```
copy network a:
copy test2.out a:          (if you have a HERON2 module in slot 1)
copy test4.out a:          (if you have a HERON4 module in slot 1)
```

(To help you start up faster, we have included 2 prepared out files, test2.out for use with a HERON2 and HEPC9, and test4.out for use with a HERON4 and HEPC9. Please change the 'network' file to suit the module type you have in slot 1 and the board type you use. But usually the \*.out file must first be created using Code Composer Studio. Please refer to the document in the lower (upper?) directory how to do this.) After completion, remove the floppy disk and insert it into the target machine's floppy disk. Reboot. The target machine should boot from disk. 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.

To compile/build the ex1 example, run: -

```
nmake -f makefile.ex1
```

and create an RTOS-32 floppy as follows: -

```
bootdisk ex1 a:
```

The ex1 example use the same \*.out and network files are the 'threads' example.

To compile/build the ex2 example, run: -

```
nmake -f makefile.ex2
```

and create an RTOS-32 floppy as follows: -

```
bootdisk ex2 a:
```

The ex2 example use the same \*.out and network files are the 'threads' example.

## The Makefile

What changes have been made to the original RTOS-32 example makefile? This section will explain what needs to be changed (or added) in a makefile to compile/link successfully the Hunt Engineering API programs

### Include file

All Hunt Engineering API programs must include 'heapi.h'. This file is located in the Hunt Engineering API&Tools installation directory. The installation program will have created an environment variable 'HEAPI\_DIR' that points to the installation directory. To have the makefile understand where 'heapi.h' lives, the following line must be in your makefile:

```
INCLUDE = $(RTTARGET)\include;$(HEAPI_DIR);$(INCLUDE)
```

The bold italic part is the part added by us.

Programs that want to use the Server/Loader library must include 'hesl.h'. This file is located in the 'hesl\inc' sub-directory of the API&Tools installation directory. There's an environment variable 'HESL\_DIR' that points to the 'hesl' directory within the API&Tools installation directory, created by the installation process. To make sure 'hesl.h' also gets found, use the following line in your makefile:

```
INCLUDE $(RTTARGET)\include;$(HEAPI_DIR);$(HESL_DIR)\inc;$(INCLUDE)
```

### Libraries

The Hunt Engineering Server/Loader library is delivered as a static library ('rtossl.lib'). This library uses functions from the hrn\_fpga library ('hrnfpga.lib') and the Hunt Engineering API ('rtosdrv.lib'). Therefore, to link any program that uses the Server/Loader library, you must also link with 'hrn\_fpga.lib' and 'rtossl.lib'.

The Server/Loader (library) uses RTOS's file support, and a Server/Loader program must be linked with RTTARGET-32 and RTFILES-32. The HUNT ENGINEERING API uses threads, and therefore the application must also be linked with RTKERNEL-32.

In the lines following your '.exe' declaration 'rtossl.lib' must be linked in first, then 'hrnfpga.lib' and then 'rtosdrv.lib', before all of the RTOS-32 libraries:

```
threads.exe: Init.c ..\threads.cpp
cl /MT /W3 /GX /Fm /Zi -D_RTOS32=1 -DCMDLINE=1 -DPC=1 -othreads.exe \
..\threads.cpp \
$(HESL_DIR)\lib\rtos32\rtossl.lib \
$(HEAPI_DIR)\hrn_fpga\lib\rtos32\hrnfpga.lib \
$(HEAPI_DIR)\rtos32\rtosdrv.lib \
rtk32.lib \
drvrt32.lib \
rtfiles.lib \
rtfsrtt.lib \
rtt32.lib \
rttheap.lib \
$(LNKOPT)
```

The bold italic part is the part added by us.

The necessary RTFILES-32 libraries are 'rtfiles.lib' and 'rtfsk32.lib'. Note that the latter is the RTKERNEL-32 version of the RTFILES-32 library.

The necessary RTKERNEL-32 libraries are 'rtk32.lib' (debug version) and 'drvrt32.lib', as the Hunt Engineering API uses multi-threading. (The HeRead and HeWrite will spawn separate threads to do the actual reading and writing. HeTestIo and HeWaitForIo 'test' the thread to see whether it has completed a transfer.)

The RTTARGET-32 library is 'rtt32.lib'. Library 'rttheap.lib' is optional. Please refer to the RTOS-32 manual (for example, ch.7 page 106 and 107).

## Compile Parameters

The Hunt Engineering API supports several different types of Operating Systems. To select RTOS-32 support, you need to #define a variable '\_RTOS32'. The easiest way to do this is in the makefile. Also, as the Hunt Engineering API is multi-threaded, you need to use the '/MT' option of the Microsoft C/C++ compiler. Example:

```
threads.exe: Init.c ..\threads.cpp
cl /MT /W3 /GX /Fm /Zi -D_RTOS32=1 -DCMDLINE=1 -DPC=1 -othreads.exe \
..\threads.cpp \
$(HESL_DIR)\lib\rtos32\rtossl.lib \
$(HEAPI_DIR)\hrn_fpga\lib\rtos32\hrnfpga.lib \
$(HEAPI_DIR)\rtos32\rtosdrv.lib \
rtk32.lib \
drvrt32.lib \
rtfiles.lib \
rtfsrtt.lib \
rtt32.lib \
rttheap.lib \
$(LNKOPT)
```

The bold italic part is the part added by us.

Before Server/Loader version 4.08, include files 'network.h', 'common.h' and 'ccif.h' were used to interface with the Server/Loader library. Server/Loader versions 4.08 and later use 'hesl.h' to interface. The legacy interface consisting of 'network.h', 'common.h' and 'ccif.h' is still supported, but for existing applications only. Use 'hesl.h' for new applications.

When using the legacy interface include file 'network.h', CMDLINE and PC needed to be defined to 'select' the appropriate portions from that include file. This is no longer necessary for applications using only 'hesl.h'. For completeness it is still shown in the example above, but for 'hesl.h' based applications it will make no difference, even though it is essential for 'legacy' applications.

## Initialisation

A file 'init.c' is included in the project. This is a 'standard' file from On Time, which they use for projects that use RTFILES-32. I have simply copied it into this example; it's needed when you use file support. On Time's comment in 'init.c':

```
/* Some standard initializations for RTFiles-32 programs.

This file is linked with most RTFiles-32 demo programs. It provides a
convenient place to configure RTTarget-32 and RTFiles-32.
*/
```

### The threads.cfg configuration file

What changes have been made to the original RTOS-32 example configuration file? This section will explain what needs to be changed (or added) in a configuration file to compile and link successfully the Hunt Engineering Server/Loader programs

#### threads.cfg: commandline

RTOS-32 programs have the possibility to carry a command line. This is done by specifying a command line in one of the configuration files (we just chose 'threads.cfg'). The 'threads' example needs to use a command line, as you may want to specify. The command line is further just as you would expect with a normal DOS or win32 program. The following line must be added to a configuration file, for the 'threads' example to work properly:

```
CommandLine "a:\threads.exe -v"
```

The directory ('a:\') is significant.

#### threads.cfg: floppy access

To access files on a floppy disk, not only do you need to link with RTFILES-32 libraries, you also need to allocate a DMA buffer for the floppy driver in your configuration file. We added the following line to the 'threads.cfg' configuration file:

```
Locate Nothing FloppyDMA HighMem 18k 32k ReadWrite
```

Please refer to the RTOS-32 manual (Part III, ch. 7, page 300) for more information.

Technical support for HUNT ENGINEERING products should first be obtained from the comprehensive Support section [www.hunteng.co.uk/support/index.htm](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 [www.hunteng.co.uk](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](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.