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

API for RTOS-32

Installation and User Manual

Document Rev B
API software Rev 1.9.7
J.Thie 24-05-04

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 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.

Revision History

API ver 1.7
First API version to support RTOS-32.

API ver 1.9.8

RTOS32 support updated to full 1.9.8
VxWorks support updated to full 1.9.8
Linux support updated to full 1.9.7

TABLE OF CONTENTS

To develop RTOS-32 programs, the RTOS-32 manual recommends using a development machine (running Windows) and a target machine (running RTOS-32). A Hunt Engineering board would be present in the target machine. On the Windows machine you would then develop RTOS-32 programs; these programs are then downloaded over RS-232 to the target machine or run off a floppy disk.

To develop DSP programs for your HERON modules, you need to have Code Composer Studio installed. This program is not only a debugger, but also an IDE (Integrated Development Environment), and includes a real-time kernel (DSP/BIOS). To use Code Composer Studio and develop DSP programs, you need to have a Hunt Engineering board inserted in the same machine (where Code Composer is installed).

Combining both of these demands, one possibility is to insert the Hunt Engineering board in the development machine, and use this combination to develop the DSP code. Once this is ready, move the Hunt Engineering board over to the target machine, and develop the RTOS-32 code. Another possibility is to have a dual-booting target machine (dual booting between RTOS-32 and Windows). In this case too, the DSP code development and the RTOS-32 code development will proceed seperately. A third possibility is to connect the JTAG of the Hunt Engineering board (in the target PC) to a JTAG interface card (in the development PC). However, not all Hunt Engineering boards support this setup. For example, the HEPC8 has no JTAG in (or out) connectors, and you cannot use an external JTAG interface board. Finally, a fourth option is to have only one machine, dual booting between Windows and RTOS-32.

In the first case (Windows development machine and RTOS-32 target machine), you should do a Windows + RTOS-32 install on the development machine. Make sure that the carrier board is in the development machine while installing.

In the second case (Windows development machine and dual-booting target machine), you should do a Windows install on the target machine, and an RTOS-32 install on the development machine. Make sure that the carrier board is in the target machine during installation.

In the third case (Windows development machine and RTOS-32 target machine connected via JTAG), you should do a Windows + RTOS-32 install on the development machine. Make sure that the carrier board is in the development machine during installation.

In the fourth case (dual-booting Windows and RTOS-32 single machine), you should do a Windows + RTOS-32 install. Make sure that the carrier board is in the PC while installing.

General

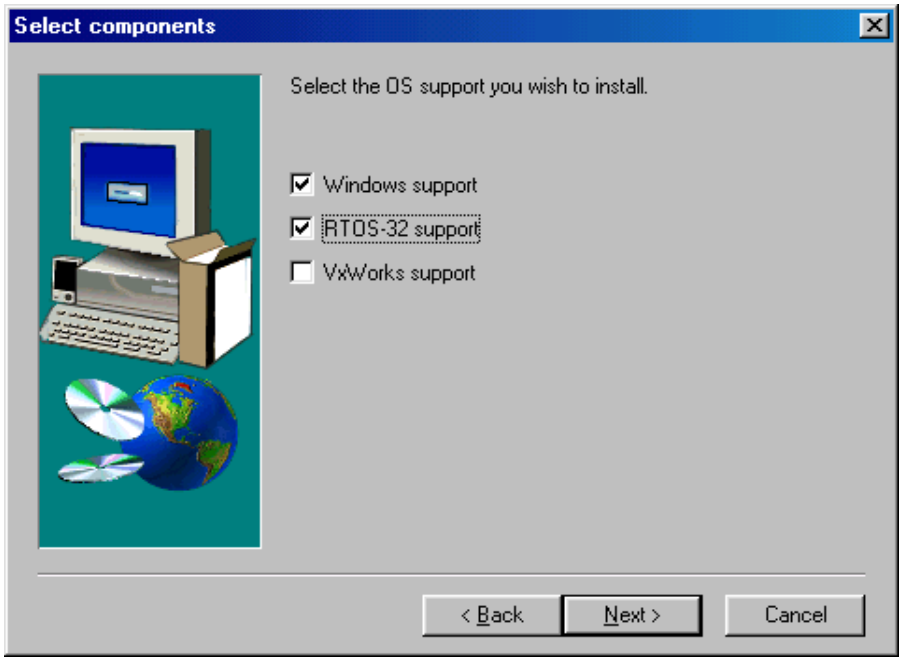
RTOS-32 installation is integrated into the standard HUNT ENGINEERING Windows installation program. As per the previous section, you must decide whether to install for RTOS-32 only, Windows only, or RTOS-32 + Windows.

The HUNT ENGINEERING CD install program will install and confidence check the API for Windows 95/98/ME, NT, W2K and DOS installations. If, for any reason, you experience problems with installation or confidence checking use this section for a step-by-step installation check.

Windows 2000 / XP Installation

The API supports Windows 2000, installing WDM drivers. This has been tested with Windows 2000 and Windows XP and the API works fine on both. To install:

1. Log in as ***Administrator*** onto your Windows 2000 or XP machine.
2. Insert the HUNT ENGINEERING CD.
3. A new window should now appear. Select '***Install Drivers & Tools***'. If the window doesn't appear, use Windows Explorer and browse to your CD drive. Then double-click on 'DSP_CD'.
4. Next, select '***Install or Upgrade***'.
5. The 'InstallShield' setup program will now start.
6. After two screens ('destination folder', 'board type') you should see:



Make sure that you select the proper components, as explained in the 'Installation Guide'. Ie for cases 1, 2 and 4 select both 'Windows support' and 'RTOS-32 support'.

7. Follow the directions given by setup.

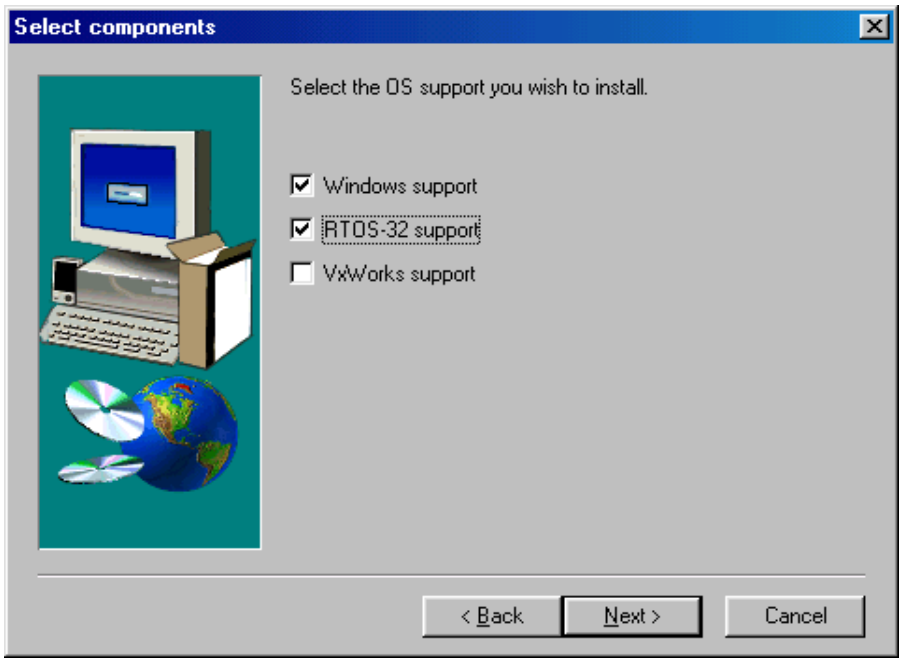
The setup program usually installs the drivers straight away, without rebooting. After the drivers are installed, the setup program will confidence check the installation. The HUNT ENGINEERING board must be inserted in this machine or else the confidence checks will fail.

If you upgrade from an earlier API version, first un-install the old API installation. Then reboot the machine. Then follow the instructions above.

Windows NT Installation

The API supports Windows NT V4.0. It has been tested with Service Pack 3 and 4, and without a Service pack; but it should work on all varations. To install:

1. Log in as ***Administrator*** onto your NT machine.
2. Insert the HUNT ENGINEERING CD.
3. A new window should now appear. Select '***Install Drivers & Tools***'. If the window doesn't appear, use Windows Explorer and browse to your CD drive. Then double-click on 'DSP_CD'.
4. Next, select '***Install or Upgrade***'.
5. The 'InstallShield' setup program will now start.
6. After two screens ('destination folder', 'board type') you should see:



   Make sure that you select the proper components, as explained in the 'Installation Guide'. Ie for cases 1, 2 and 4 select both 'Windows support' and 'RTOS-32 support'.

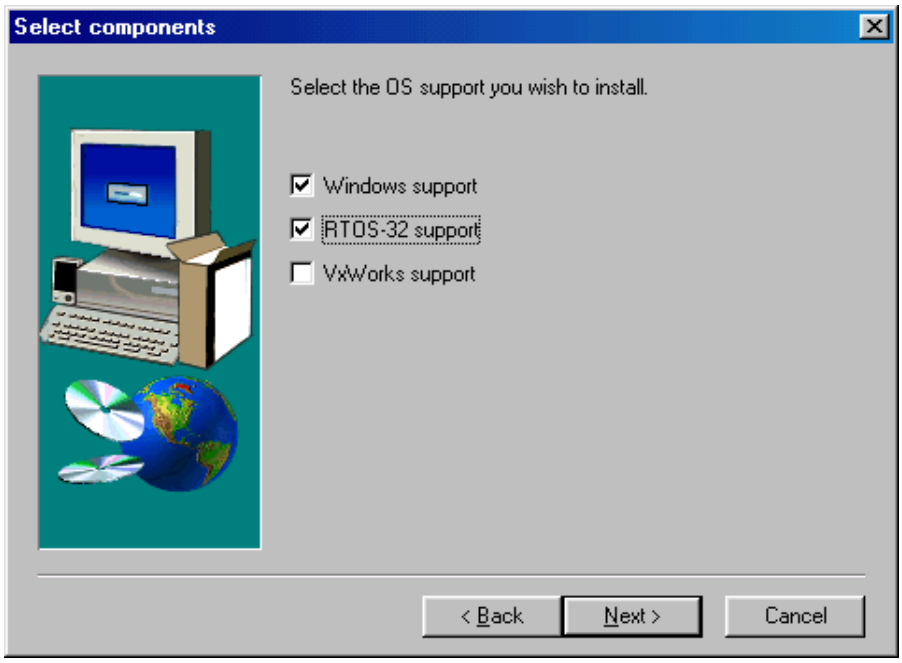
7. Follow the directions given by setup.

The setup program will try to reboot the PC once. After the reboot, the setup program will confidence check the installation. The HUNT ENGINEERING board must be inserted in this machine or else the confidence checks will fail.

If you upgrade from an earlier API version, first un-install the old API installation. Then reboot the machine. Then follow the instructions above.

Windows 95/98/ME Installation

1. Insert the HUNT ENGINEERING CD.
2. A new window should now appear. Select '***Install Drivers & Tools***'. If the window doesn't appear, use Windows Explorer and browse to your CD drive. Then double-click on 'DSP_CD'.
3. Next, select '***Install or Upgrade***'.
4. The 'InstallShield' setup program will now start.
5. After two screens ('destination folder', 'board type') you should see:



   Make sure that you select the proper components, as explained in 'Installation Guide'. Ie for cases 1, 2 and 4 select both 'Windows support' and 'RTOS-32 support'.

6. Follow the directions given by setup.

The setup program will try to reboot the PC once. After the reboot, the setup program will confidence check the installation. The HUNT ENGINEERING board must be inserted in this machine or else the confidence checks will fail.

If you upgrade from an earlier API version, first un-install the old API installation. Then reboot the machine. Then follow the instructions above.

DOS Installation

There is no HUNT ENGINEERING API support for DOS with RTOS-32.

RTOS32 Manual Install

There are perhaps cases where you wish to install the RTOS-32 files manually. Or perhaps you just want to understand what and where the RTOS-32 related files are.

1. Create an installation directory, for example somewhere on your harddisk.

2. Copy 'rtosdrv.lib' from the CD (\software\api\rtos32) into the installation directory.
3. Copy the four include files and 'hecodes.c' from the CD (\software\api\heapi) into the installation directory.
4. Optionally, you may want to copy the example directory tree from the CD (\software\examples\host_api_examples\c6x) into the installation directory.
5. Define an environment variable 'HEAPI_DIR' and set it to point to your installation directory. For example, edit autoexec.bat and add a line "set heapi_dir=c:\heapi". Then reboot the machine.

Uninstall

1. Insert the HUNT ENGINEERING CD.
2. A new window should now appear. Select '***Install Drivers & Tools***'. If the window doesn't appear, use Windows Explorer and browse to your CD drive. Then double-click on 'DSP_CD'.
3. Next, select '***UnInstall'***. Follow the instructions.

Alternatively, you can uninstall via Windows. Go to "Start → Settings → Control Panel", then double click on "Add/Remove Programs". Select "HUNT ENGINEERING API" and click the "Add/Remove…" button. Follow the instructions.

If you want to re-install the API, **first reboot**!

RTOS-32 Confidence Check

To confidence check the RTOS-32 part of the installation please have a look at the examples. They are on the HUNT ENGINEERING CD, in the \software\examples\host_api_ examples directory. Copy the examples onto your harddisk to a convenient location. If you installed the HUNT ENGINEERING API with library and examples, you will already have the examples directory in c:\heapi\etc. (Assuming you installed into c:\heapi).

The examples double as a confidence check as well. Each of the examples has an RTOS-32 sub-directory where RTOS-32 specific makefiles etc. are located. They assume you use a floppy disk to run the examples on the RTOS-32 target machine. PDF documents in the example directories will explain further how to compile, link and run the examples.

Windows Confidence Check Program

If you selected 'Windows Support', then the setup program will automatically have done a number of confidence checks. You can do the confidence checks manually also. The Windows Confidence check program is installed automatically on any Windows version. It is started up as follows:

Start → Programs → HUNT ENGINEERING → Confidence checks

Please review the HUNT ENGINEERING's API&Tools Windows installation manual for more information on the Confidence Check program.

General (for all Windows versions)

For troubleshooting, please review the Troubleshooting section in the API for Windows Installation and User manual. The following suggestions are just a summary from what is described more fully in the Troubleshooting section in the API for Windows Installation and User manual.

Different PCI slots may have different resources assigned to them. One thing to try is to move the HERON carrier board to other, unused, PCI slots. If that doesn't work, you can try swapping the HERON carrier board with other PCI boards.

The HEPC9 board uses MasterMode, which is like dma, but with the dma engines on board of the HEPC9. Sometimes a motherboard doesn't support MasterMode properly, and you can try if the Confidence Tests succeed if MasterMode is disabled.

In the PC BIOS you **may** have an ECSD setting ("Electronic Configuration Setup Data"). If it exists, it must be cleared. Clearing this field gives you one (1) chance to install a new board. After 1 reboot this ECSD reverts to being set. So, each time you want to install or re-install or re-arrange PCI boards, you need to clear this ECSD switch again.

Usually you can access BIOS settings by pressing DEL or F2 very soon after a reboot, far before the operating system starts to run. You would see a message at the bottom of the screen, saying something like 'press DEL to go to setup'. Please refer to your motherboard's manual if necessary.

With an HEPC8 board, verify the following: -

1. The boot jumpers of the module in slot 1 must be set to boot from FIFO 0.
2. The board switch (the red one) must be set to 0.
3. Make sure that there is a HERON processor module in slot 1.

Also ensure that the HERON module in slot 1 is plugged in properly. To be sure, take the HERON carrier board out of the PC; then remove the slot 1 module, and re-insert. Now re-insert the carrier board in your PC and try again.

When to use the API

The HUNT ENGINEERING API offers a platform independent, target independent way of accessing a HUNT ENGINEERING board. You can write your own programs that use the API to boot DSP programs onto a board. However, the Server/Loader may provide all functionality you need, and you may not have to develop API applications.

The Server/Loader is a HUNT ENGINEERING tool that can boot a network of DSP processors. The Server/Loader uses a 'network' file that describes what processors must be booted with what file. A 'network' file is a simple and short ASCII file. ASCII files can be created with, for example, 'edit' on MS-DOS, 'notepad' on Windows, or 'vi' on Linux. In cases where you want the Server/Loader to only boot the network, and process messages from your DSP application yourself, you can use the Server/Loader library. Please refer to the Server/Loader manual for more information.

Include file

All API applications must '`#include <heapi.h>`' in all source files that call API functions. This include file is located in the root of the API installation directory. The install program will have created an environment variable 'HEAPI_DIR' that points to your installation directory. Therefore, to make sure that the compiler knows where to find 'heapi.h', please add this path to the makefile's include path. For example, for Microsoft Visual C/C++:

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

The bold italic part is what needs to be added to a 'standard' RTOS-32 makefile for Microsoft Visual C/C++.

Compile parameters

The HUNT ENGINEERING API covers several different operating systems. In your application you have to select for what operating system you want to compile. To select an RTOS-32 application, set parameter '_RTOS32' to 1. For example, you could do this in the makefile, on the compiler line. For example, for Microsoft Visual C/C++:

```
cl /MT /Fm /Zi -D_RTOS32=1 -oreads.exe \
```

This defines the _RTOS32 parameter to be one. Also, the HUNT ENGINEERING API library for RTOS-32 is a multi-tasking library. Therefore, the compiler has to be notified to use multi-tasking support. For the Microsoft Visual C/C++ compiler, this means that the '/MT' parameter must be used.

Libraries

The HUNT ENGINEERING API library for RTOS-32 is a static library that uses multi-tasking and file support ('rtosdrv.lib'). That means that an RTOS-32 API application must be linked with the multi-tasking RTFILES-32 libraries ('rtfiles.lib' and 'rtfsk32.lib'), with the RTKERNEL-32 libraries ('rtk32.lib' and 'drvrt32.lib'), and with the RTTARGET-32 libraries ('rtt32.lib' and optionally 'rttheap.lib'). For example, in the Microsoft Visual C/C++

makefile you would have:

```
xxxx.exe: { C/C++ files }
   cl /MT /Fm /Zi -D_RTOS32=1 \
      { C/C++ files }
      $(HEAPI_DIR)\rtos32\rtosdrv.lib \
      rtfiles.lib \
      rtfsk32.lib \
      rtk32.lib \
      drvrt32.lib \
      rtt32.lib \
      rttheap.lib \
      $(LNKOPT)
```

Initialisation

RTOS-32 example programs for RTFILES-32 usually include a file 'init.c'. You may want to use the same file for your own project(s). The file is also used in all HUNT ENGINEERING API example programs. At the beginning of the 'init.c' file, the following comment:

```
/* 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.
*/
```

Also, the RTKERNEL-32 library needs initialisation(s). The standard initialisation as used in RTOS-32 programs is:

```
// Initialize the kernel and keyboard driver
RTKernelInit(5);
KBInit();
```

The same code has been used in the HUNT ENGINEERING API examples.

CFG file

The HEPC9 and other HEART boards (not HEPC8) use MasterMode by default. MasterMode is like dma, except that the dma engines are on the board (HEPC9) itself. With RTOS32, MasterMode uses fixed buffers with a fixed physical address, which must be defined in your CFG file. Below the default entries in a CFG are shown: -

```
Locate Nothing FifoABuffer HighMem 128k 32k ReadWrite
Locate Nothing FifoBBuffer HighMem 128k 32k ReadWrite
Locate Nothing FifoCBuffer HighMem 128k 32k ReadWrite
Locate Nothing FifoDBuffer HighMem 128k 32k ReadWrite
Locate Nothing FifoEBuffer HighMem 128k 32k ReadWrite
Locate Nothing FifoFBuffer HighMem 128k 32k ReadWrite
```

Obviously, FifoABuffer is for Fifo A, FifoBBuffer for Fifo B, and so on. The buffers are only used if you use the associated fifo and if you use MasterMode with that fifo. The minimum size of the buffer has to be 16 kB. The maximum size may be anything, but the maximum that can be used is 128 kB, and the API only uses the buffer in multiples of 4 kB.

Virtual or Physical Memory

RTOS-32 has the option of using virtual or physical memory. The HUNT ENGINEERING API can be used with either option.

PCI cards such as the HEPC9 and HEPC8 use a memory-mapped interface. The HeOpen() function will automatically try to find the HUNT ENGINEERING board you specified, and map the interface memory area into the system (using 'RTFindPhysMem', 'RTReserveVirtual-Address' and 'RTMapMem'). There is no need for you to make any provisions in your configuration files for a HUNT ENGINEERING board.

Pre-emptive or Co-operative Multi Tasking

The HUNT ENGINEERING API can be used with pre-emptive multi-tasking, but also with co-operative multi-tasking. However, you have to be much more careful when using co-operative multi-tasking.

The API uses threads to execute read and write requests. For example, the HeRead function will merely 'initiate' a read transfer, by asking a read thread to perform the transfer. The HeRead then returns, with return value 'He_IoInProgress'. Similarly, the HeWrite function only 'initiates' a write transfer. You must use HeTestIo to test if the transfer has completed, or HeWaitForIo to wait for the transfer to complete.

The read and write threads run at a high priority (51). Interrupts are used to inform the read and write threads that any data can be read or written. The interrupt routine uses semaphores to implement this. If there's no data to read or write, the read and write threads sit idle waiting for the interrupt semaphore.

With pre-emptive multi-tasking, as soon as an interrupt comes in, the interrupt semaphore is signalled; if the read or write thread has the highest runnable priority, they will immediately start to run. If not, they will start to run after higher priority runnable tasks have changed state to waiting or have completed.

With co-operative multi-tasking, the interrupt semaphore is signalled as well. However, the reads and write threads will have no chance to start running until the current thread makes a kernel call that may trigger a task switch. Once the kernel call is made, the task with the highest priority will be made to run, just like the pre-emptive case.

Why is this important? Well, consider a low priority task that wants to test if a read or write transfer has completed:

```
while (status!=HE_OK) status = HeTestIo(pIoStatus);
```

With co-operative multi-tasking this inifinite loop would prevent any other task from running. Therefore, the HeTestIo function is programmed in such a way, that a task switch can occur. (The RTKScheduler function is called within HeTestIo.) Thus, you need to be aware that HeTestIo and also HeWaitForIo may trigger task switches. In the case of the HeWaitForIo function it is the RTKWait function that may possibly cause a task switch.

With pre-emptive multi-tasking you have no such worries. A task switch may occur at any time and infinite loops as shown above pose no problems.

Read and Write Threads

The HeRead function will not perform a read transfer itself, but instead will ask a read thread to perform the read transfer. Similarly, the HeWrite function will not perform a write transfer itself, but instead will ask a write thread to perform the write transfer.

A read or write thread is associated with a HE_IOSTATUS, not with a HE_HANDLE. When the HE_IOSTATUS is used for the first time (ie the first HeRead or HeWrite using that HE_IOSTATUS), the associated thread is actually started up.

You cannot issue more than 1 HeRead at a time, and you cannot issue more than 1 HeWrite at a time, with the same fifo. (But you can issue 1 HeRead and 1 HeWrite at a time on the same fifo, or 1 HeRead on e.g. FifoA and 1 HeRead on e.g. FifoC at the same time.) After a HeRead or HeWrite has been executed, you need to confirm the end of the transfer (using HeTestIo or HeWaitForIo) before you can issue the next HeRead or HeWrite.

Read and write threads run at a high priority. The priority for fifo threads is 51; the priority for HSB threads is 50. In case you want to change this, please use any or all of 14 environment variables as specified in the next section.

Environment variables

The HUNT ENGINEERING API for RTOS-32 recognizes 14 environment variables that can be used to set a read or write thread's priority.

Variable HE_RDPRI_HSB sets the HSB read thread priority (default: 50).

Variable HE_RDPRI_FIFOA sets the FIFOA read thread priority (default: 51).

Variable HE_RDPRI_FIFOB sets the FIFOB read thread priority (default: 51).

Variable HE_RDPRI_FIFOC sets the FIFOC read thread priority (default: 51).

Variable HE_RDPRI_FIFOD sets the FIFOD read thread priority (default: 51).

Variable HE_RDPRI_FIFOE sets the FIFOE read thread priority (default: 51).

Variable HE_RDPRI_FIFOF sets the FIFOF read thread priority (default: 51).

Variable HE_WRPRI_HSB sets the HSB write thread priority (default: 50).

Variable HE_WRPRI_FIFOA sets the FIFOA write thread priority (default: 51).

Variable HE_WRPRI_FIFOB sets the FIFOB write thread priority (default: 51).

Variable HE_WRPRI_FIFOC sets the FIFOC write thread priority (default: 51).

Variable HE_WRPRI_FIFOD sets the FIFOD write thread priority (default: 51).

Variable HE_WRPRI_FIFOE sets the FIFOE write thread priority (default: 51).

Variable HE_WRPRI_FIFOF sets the FIFOF write thread priority (default: 51).

You can use environment variables with RTOS-32 via a configuration file. For example, in the 'reads' example, you could use the 'reads.cfg' configuration file, and set the FIFOA read thread priority to 60 in this file:

```
set HE_RDPRI_FIFOA=60
```

Please also refer to the RTOS-32 manual, Part I, Chapter 3, page 34.

MasterMode and Interrupts

Some HUNT ENGINEERING boards, such as the HEPC9, support MasterMode. This feature is easiest described as on-board DMA. Boards such as the HEPC9 have their own DMA engines, which are used and accessed by the HEPC9 drivers. Not all motherboards / chipsets work perfectly with the MasterMode feature, and the RTOS32 software allows you to switch it off – ie tell the drivers to not use MasterMode. Switching off MasterMode is done in your application program when opening a FIFO. Use the HeOpenS function, which takes a list of switches as option. One of the switches then specifies to not use MasterMode. A small example: --

```
HE_DWORD switches[2];
switches[0]=HE_Switch_NoPCIMasterMode;
switches[1]=HE_Switch_Last;
Status = HeOpenS("hep9a", 0, "FifoA", &hDevice, switches);
if (Status != HE_OK) return error(HeGetLastOsError(hDevice));
```

Similarly, you can tell the driver not to use interrupts. By default the driver uses interrupts, but with some motherboard / chipsets interrupts don't work properly sometimes. In such cases you can try to switch off interrupts. As with MasterMode, telling the driver not to use interrupts is done via the HeOpenS function, this time with a different switch.

```
HE_DWORD switches[2];
switches[0]=HE_Switch_NoInterrupts;
switches[1]=HE_Switch_Last;
Status = HeOpenS("hep9a", 0, "FifoA", &hDevice, switches);
if (Status != HE_OK) return error(HeGetLastOsError(hDevice));
```

The switches array can be any length, but the last valid array element must have the value '`HE_Switch_Last`'. The parser in the HeOpenS function will process switches until it reads the '`HE_Switch_Last`' value. To switch both MasterMode and interrupts off, you can combine the above into one switches array: --

```
HE_DWORD switches[3];
switches[0]=HE_Switch_NoInterrupts;
switches[1]=HE_Switch_NoPCIMasterMode;
switches[2]=HE_Switch_Last;
Status = HeOpenS("hep9a", 0, "FifoA", &hDevice, switches);
if (Status != HE_OK) return error(HeGetLastOsError(hDevice));
```

The order of the switches is not important. There are other switches than '`HE_Switch_NoPCIMasterMode`' and '`HE_Switch_NoInterrupts`', and they are defined in 'heapi.h'. For RTOS32, however, only the 2 switches mentioned here are relevant.

The default of the API is to use MasterMode and and to use interrupts. In almost all cases this is the best option and gives highest performance. It is only when you have trouble using MasterMode and/or interrupts that you may need to try switching these off.

The HUNT ENGINEERING CD has a number of examples that show how you could use the API. The examples are located on the CD in: \software\examples\host_api_examples\ c6x\examples. There will be PDF files in the example directories that explain how the examples work. Each example directory has a number of sub-directories; each sub-directory dedicated to a certain operating system. In sub-directory 'rtos32' you will find a PDF file that explains how to compile and build the example for RTOS-32, using the Microsoft Visual C/C++ compiler.

The API Reference Manual can be found in the \manuals directory on the HUNT ENGINEERING CD. The Reference Manual explains in detail how to use the API, shows all possible return values, contains an overview of all functions in the API library plus a per-function explantion, and so on. Essentially, the platform and carrier board independent API interface is explained in this manual.

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.