*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*

# *Hrn_fpga*

## *FPGA configuration tool using HSB*

# *USER MANUAL*

***Software Version 2.4***
***Document Rev E***
***P.Warnes 02/09/04***

## 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

HERON modules that have a user configurable FPGA, can accept their configuration information (bit stream) over the Hunt Engineering Serial Bus (HSB). Alternatively they can sometimes be configured from a non-volatile PROM fitted to the module.

When using the HSB as the configuration method, the Hrn_fpga.exe tool is provided to interpret the Xilinx format bit stream file (Raw Bit stream Text format .rbt) and to send the correct messages over the HSB via the Host API.

In version 2.0 there is another file type introduced. This file is a Hunt Compressed Bitstream file (.hcb). This is a run-length encoded version of the .rbt file. It was introduced so that the bitstream files shipped on the HUNT CD or via the web site are much smaller. Typically the .hcb is less than 10% of the size of the .rbt. Actually the size of a.rbt is constant for a particular FPGA type, but the size of the .hcb varies according to the content.

Hrn_fpga.exe uses the –c option to generate a .hcb from a .rbt file, and the –r option to re-generate the .rbt from a .hcb file.

Actually re-generating it is not normally necessary as Hrn_fpga.exe will read a .hcb and load it directly to an FPGA module in the same way that a .rbt file is loaded.

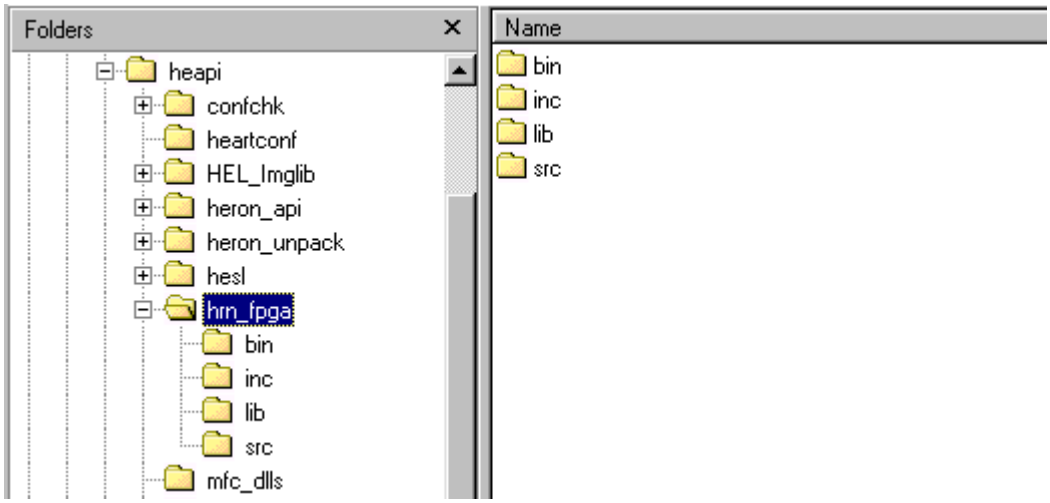
There is a second use of this run-length encoding – as FPGAs become larger, the time to load their program becomes longer. The limit on the speed is the Heron Serial Bus. As version 2.0 of Hrn_fpga.exe is released, some newly built modules will accept the compressed data format over HSB. This enables modules to be loaded faster. The modules will identify themselves to Hrn_fpga.exe as supporting this feature or not, and the faster method will be used if possible. However it is still possible to use either file type with either download method.

For Windows operating systems a GUI front end is also provided to assist you in entering and remembering the correct command line options.

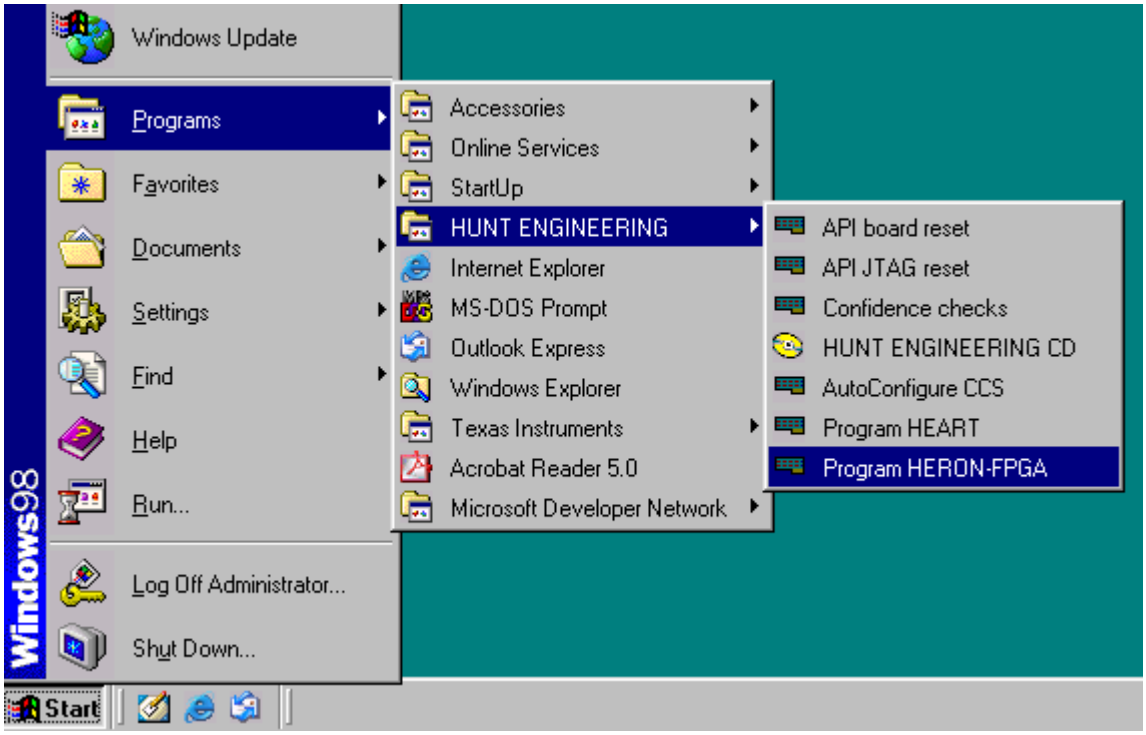
This makes configuring the FPGAs with your design or one of the standard IPs (Intellectual Properties) from the HUNT ENGINEERING CD a simple mouse click operation.

Hrn_fpga is shipped as both an executable and a library that is installed on your windows development system during the HUNT ENGINEERING Host API installation. The executable resides in the %HEAPI_DIR%\utils directory, where %HEAPI_DIR% is the location you chose during the installation (default is c:\heapi).

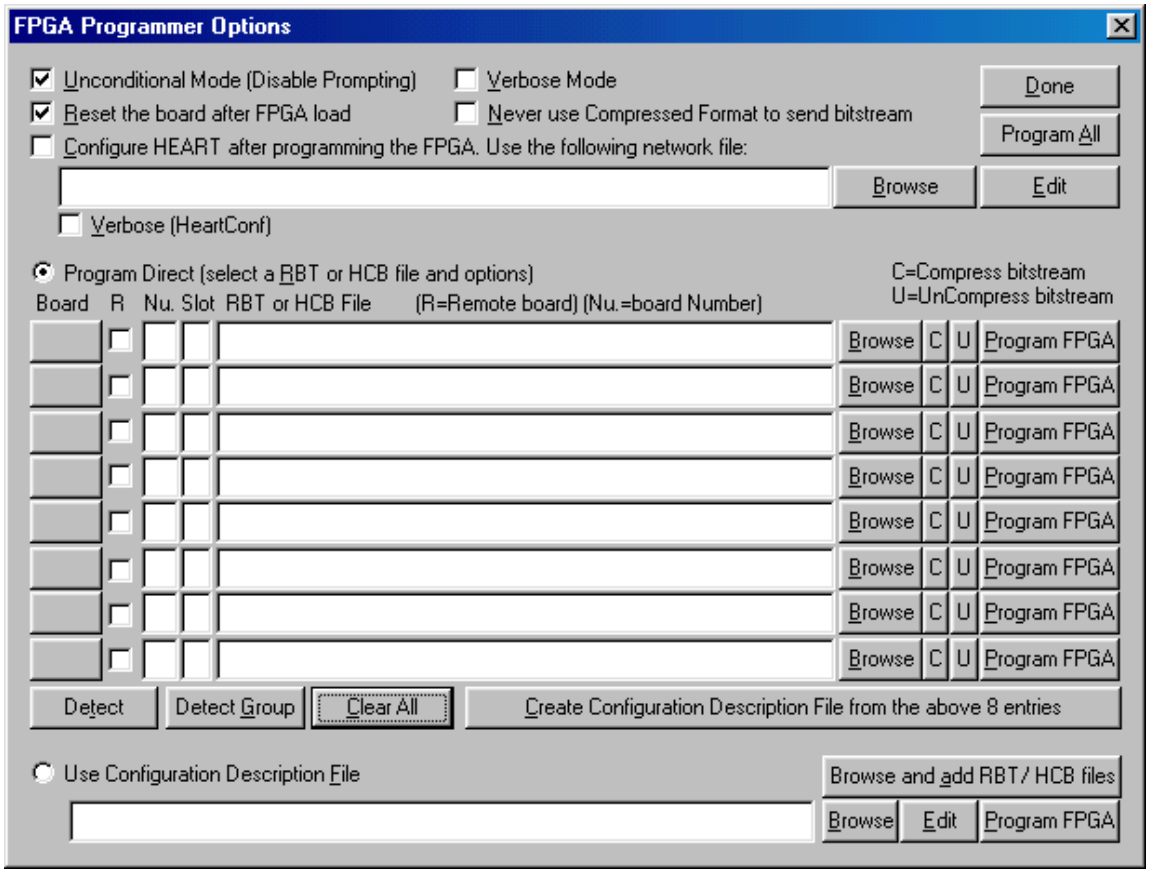
The full hrn_fpga installation is located in %HEAPI_DIR%\hrn_fpga. The library is located in %HEAPI_DIR%\hrn_fpga\lib. The hrn_fpga.h include file, holding the function prototypes and declarations, is located in %HEAPI_DIR%\hrn_fpga\inc. Directory %HEAPI_DIR%\hrn_fpga\src holds the sources of the hrn_fpga library.




For the windows operating system there is also a windows front end that can be found by accessing Programs→ HUNT ENGINEERING → Program HERON_FPGA.



Selecting Programs→ HUNT ENGINEERING → Program HERON_FPGA gives you: -



## Using hrn_fpga from a command-line

If you run hrn_fpga without parameters it searches your system for FPGA modules and asks you if you want to program them i.e.

```
c:\heapi\utils>hrn_fpga
Searching for HERON-FPGA and HERON-IO modules

A HERON-IO2 has been found in slot 3 of board 0
The FPGA on this module is a Virtex-2 device with 1000K gates
The FPGA speed grade is -4
Is this the module you want to load?
```

If you choose yes it prompts you to type in the path and name of the bit stream file to use

```
Enter the name of the bitstream RBT file:
```

You can also specify these things as command line options, and hence can use a batch file to perform the whole process without having to retype the data each time.

Using the '-h' option: -

```
c:\heapi\utils>hrn_fpga -h
```

Gets you the usage: -

```
HRN_FPGA   Heron-FPGA and Heron-IO Configuration Utility version 2.2 11/9/03
                (c) HUNT ENGINEERING

Usage:  HRN_FPGA [options] [configuration description file]
Options:
-f<File>    Specify a bitstream file (.rbt or .hcb)
-c          Generate Hunt Compressed Bitstream (.hcb) file from .rbt file
-r          Regenerate .rbt file from Hunt Compressed Bitstream (.hcb) file
-u          Unconditional mode - disable prompting
-v          Verbose mode
-s<no>      Specify Slot to configure.
-t<no>      Specify Board No of Slot (-t/-s together form heronid).
            If not specified, slot is assumed to be on board specified with -b.
-d<device>  Specify a [board type] to use e.g. hep8a, hep9a etc.
-b<no>      Specify Board No of board that accesses HSB.
-n          Never send to module in compressed format.

 If no options are specified, environmental variable HEAPIHSB is used
to determine the board type and number. If no HEAPIHSB exists, HEAPIJTAG
is used, if neither exists then the user is prompted. Options specified
on the command line override the environmental settings.
```

If you are using the command line option to load an FPGA you would use a syntax like: -

```
hrn_fpga -fd:\fpga\io2v2\example1\2v1000fg456.rbt -s2
```

to load a module in slot 2 with that bit-stream. If you don't supply the slot number you will be offered detected FPGA modules to choose from.

### Compressing files

If you want to compress your own .rbt files, use the following syntax: -

```
hrn_fpga -fd:\fpga\io2v2\example1\2v1000fg456 -c
```

Which will take the .rbt file and generate a .hcb file of the same name.

### Uncompressing files

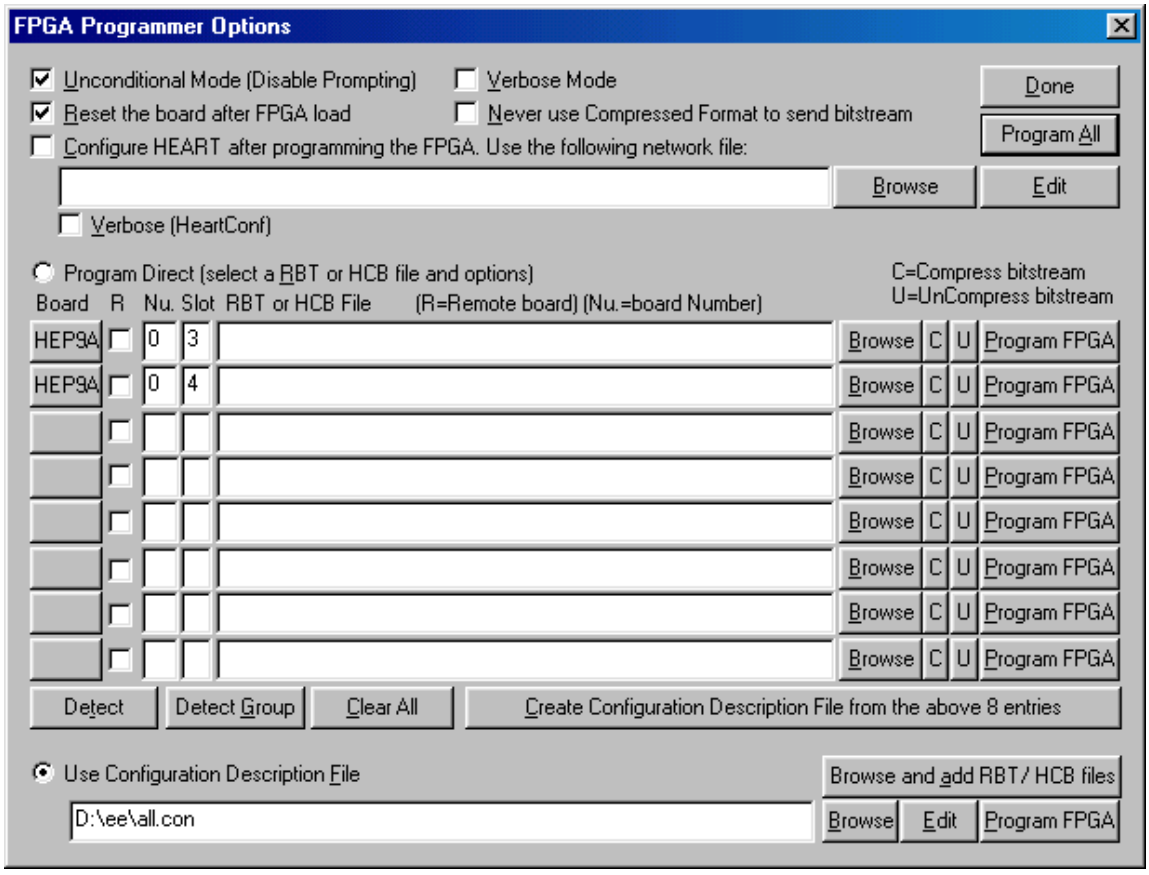
You can uncompress files into .rbt files again, but there is not normally a need to do so. Hrn_fpga.exe will accept .hcb files directly. If however there is a circumstance where you want a .rbt file, you can use :-

```
hrn_fpga -fd:\fpga\io2v2\example1\2v1000fg456 -r
```

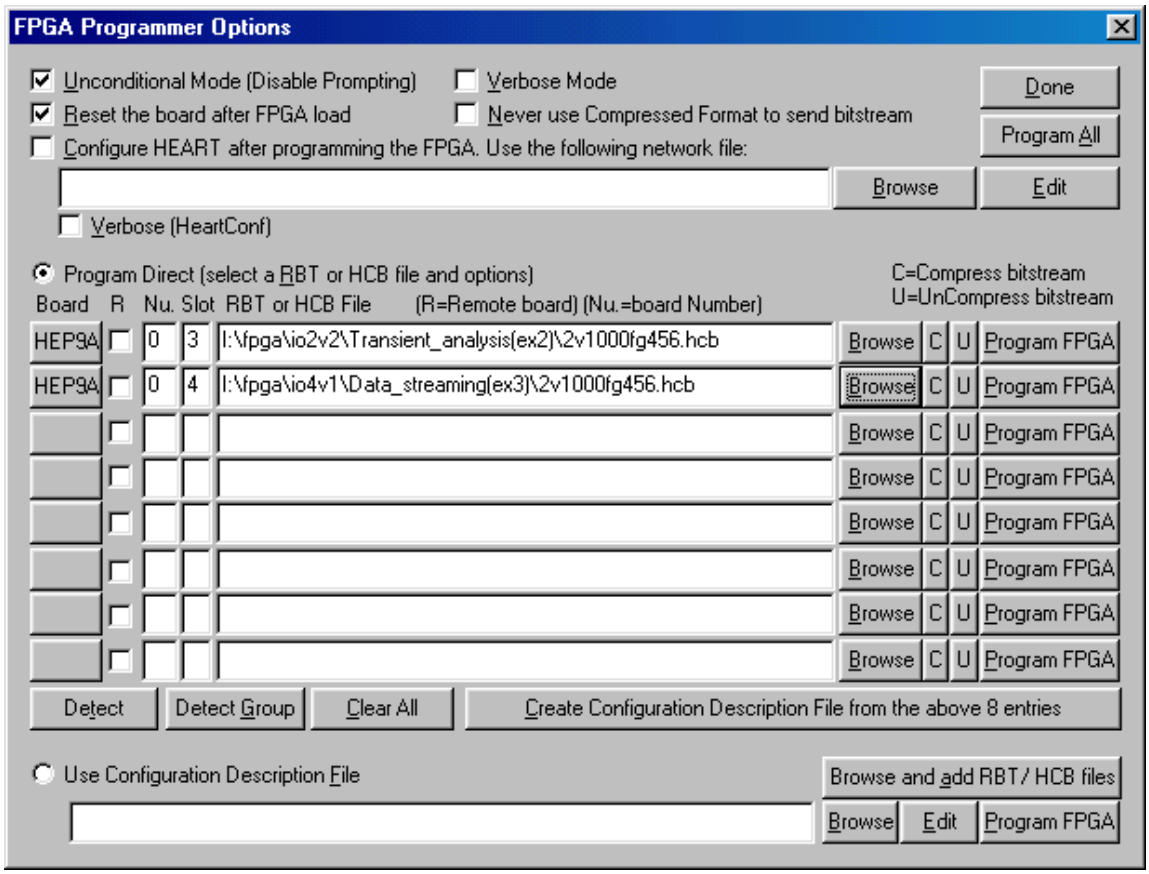
Which will take the .hcb file and generate a .rbt file of the same name.

## Using the Windows GUI

Windows users however will normally use the windows front-end program supplied. Here you can use the 'Detect' or 'Detect Group' button to identify all FPGA modules in your system at that time. The board and slot information is filled in: -



Then you can type in the path and bit-stream file name, or use the browse button to navigate to the correct file: -




Now the Program FPGA button can be used to program a single FPGA, or the Program All button used to program all FPGAs in your list. The 'Board', 'Nu' and 'Slot' fields indicate what board and slot the detected module is on. You can manually change the fields.

## Compressing files

You can compress a bit-stream (*.rbt) using the C button.

## Uncompressing files

And you can un-compress a bit-stream (*.hcb) by using the U button.

## Remote access

It is also possible to program HERON-FPGA or HERON-IO modules located on remote boards. A remote board is a board connected to another board via Inter-Board Connectors, such as EM2 or EM1 modules. A remote board could be embedded, or be located in another PC, or be in the same PC. When ticking the 'R' box you tell the windows front-end that you want to program the FPGA via another board.

To remotely program an FPGA, you must have selected or created a network file via Tools → Init Remote. This network file should define inter-board connections between boards. When the 'R' box is ticked, the Windows front-end will access boards defined as 'remote' in the network file via another board. Boards that are not defined as 'remote' and boards that are not defined at all in the network file are accessed directly, i.e., ticking the 'R' box will have no effect for such boards.

As a last remark, please note that modules on a truely 'remote' board (e.g. an embedded board or a board located in another PC) cannot be detected by 'Detect' or 'Detect Group'.

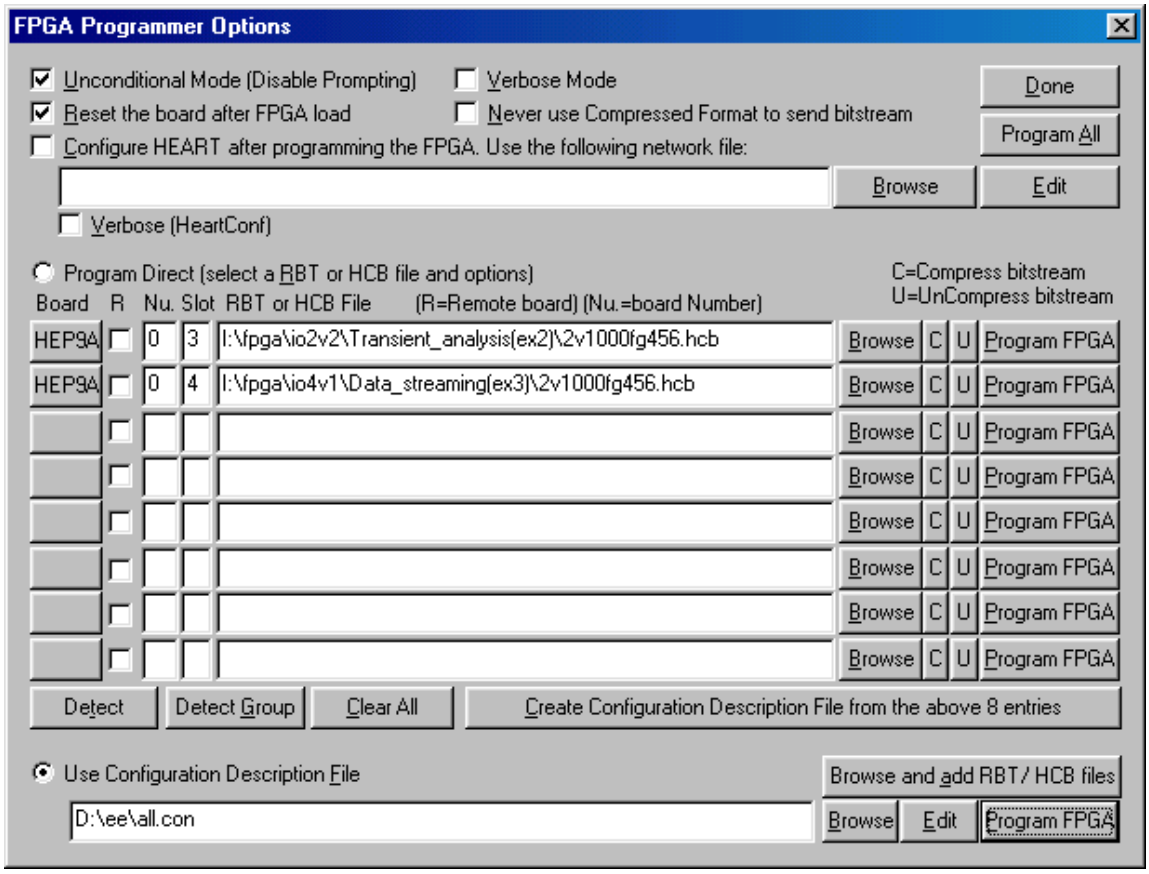
## Configuration description file

If you have more than 8 modules it is possible to use a configuration file. You can create a configuration file using the "Create Configuration Description File from the above 8 entries" where for the above example you will see: -

```
I:\fpga\io2v2\Transient_analysis(ex2)\2v1000fg456.rbt HEP9A 0 3
I:\fpga\io4v1\Data_streaming(ex3)\2v1000fg456.hcb HEP9A 0 4
```

i.e. the full path of the bit stream file followed by the board type as defined for the Host API i.e. HEPC9 = HEP9A, then the board number and slot number for that module.

When you have a configuration file you can use it by selecting the Use Configuration Description file and filling in the path of the file i.e




Then you can use the program FPGA button next to that line to configure all devices that appear in that file.

The configuration description file can also be used under non windows operating systems, but you need to create it yourself, or use this tool under windows to create it for you.

As configuration is taking place there is a count of the % completed to show you the progress.

At the end of configuration there will be a message like :-

```
% Done: 100
```

```
The bitstream has been downloaded
Configuration was successful
Press return to continue
```

If you have chosen the "Unconditional mode" then this message will immediately disappear unless there is an error.

### Reset the board after FPGA load

With HEPC9 boards it is sometimes necessary to reset the board for the bit-stream to start running properly. Our example bit-streams flash an LED on the HERON-FPGA or HERON-IO module to show that they have started properly. If not, a system reset will usually make it (re-) start properly.

### Configure HEART after programming the FPGA

After a reset is done on a board, all HEART connections on that board will disappear. With this feature you can instruct the Windows front-end to (re-) program HEART. The (re-) programming will occur after a single entry Program FPGA, but also after a Program All and a Program FPGA with a configuration description file. You need to select or create a network file in the edit box below the 'Use the following network file:' field. Tick the 'verbose' tick box if you need to do some network file troubleshooting.

### Never use Compressed Format to send bitstream

After the introduction of *.hcb bit-streams (HCB=Hunt Compressed Bit-stream), newly designed HERON-FPGA and HERON-IO modules may have an option that allows *.hcb files to be programmed directly over HSB into the FPGA. With older modules, an *.hcb file would get un-compressed first, and then be sent over HSB to the FPGA. However, there may be unforeseen situations where the sending of *.hcb data in uncompressed format doesn't work. In such situations you can try programming the FPGA with this box ticked. An *.hcb bit-stream will then always get un-compressed first before sending data over HSB to the FPGA.

## Using the hrn_fpga library

### Command-line version

One implementation of hrn_fpga is the 'hrn_fpga_main' function. This accepts command-line parameters as used in a standard console program's 'main' function. It is not necessary to use command-line parameters from the 'main' routine; you can also create your own 'argc' and 'argv' parameters. The function declaration is as follows: -

```
int hrn_fpga_main(int argc, char *argv[]);
```

Example:

```
int main(int argc, char *argv[])
{
      int     r;
      char *s;
      hrn_fpga hf;
```

```
        r = hf.hrn_fpga_main(argc, argv);
        if (r)
        {
              s = hf.hrn_fpga_getlasterr();
              if ((s!=NULL)&&(s[0]!=0)) printf("%s\n", s);
              else printf("Error %d.\n", r);
              return -1;
        }
}
```

The 'hrn_fpga_main' function returns 0 upon success, and not 0 upon error. If the return value indicates an error, the return value is itself an error value. The 'hrn_fpga_getlasterr' will return a descriptive string of the error encountered. Although it should always return a non-NULL pointer, a defensive programmer would sensibly check for the return value to be not NULL and not 0 in length.

### Command-line version with handle

There are cases when you already have a HSB handle open on a board, a board on which you now want to program an FPGA. You can close the handle, then call 'hrn_fpga_main', then re-open the handle. However, with another variant of 'hrn_fpga_main' you can use an open handle and ask 'hrn_fpga_main' to use that. The handle relevant to 'hrn_fpga_main' is the HSB device handle. If you supply a non-HSB handle, 'hrn_fpga_main' will open the HSB handle for the specified board, ignoring the handle you supplied. Similarly, if the board you specify via the 'argv' parameters doesn't match the board of the HSB handle you supplied, the proper HSB handle is opened and your handle is ignored. The function declaration is as follows: -

```
int hrn_fpga_main(HE_HANDLE hDev, int argc, char *argv[]);
```

Example:

```
int main(int argc, char *argv[])
{
      int       r;
      char     *s;
      hrn_fpga  hf;
      HE_DWORD  Status;
      HE_HANDLE hDevice=NULL;
      Status = HeOpen("hep9a", 0, HSB, &hDevice);
      if (Status!=HE_OK)
      {
            printf("Open error %x\n", Status);
            return 0;
      }
      r = hf.hrn_fpga_main(hDevice, argc, argv);
      if (r)
      {
            s = hf.hrn_fpga_getlasterr();
            if ((s!=NULL)&&(s[0]!=0)) printf("%s\n", s);
            else printf("Error %d.\n", r);
            return -1;
      }
}
```

The 'hrn_fpga_main' function returns 0 upon success, and not 0 upon error. If the return value indicates an error, the return value is itself an error value. The 'hrn_fpga_getlasterr' will return a descriptive string of the error encountered. Although it should always return a non-NULL pointer, a defensive programmer would sensibly check for the return value to be not NULL and not 0 in length.

### Generic version with handle

A more generic version of hrn_fpga is 'hrn_fpga_handle'. Rather than using arguments as used in a console program's 'main' function, it uses normal parameters: -

```
int hrn_fpga_handle(HE_HANDLE hDev, int heronid, char *bitstream,
                                    int action , int   options);
```

The `hDev` parameter should be a valid and open handle of a board's HSB device. The heronid specifies the board number and slot where the target module is on. Bits 0..3 of the heronid identify the slot number (1=slot 1, 2=slot 2, 3=slot 3, 4=slot 4). Bits 4..7 identify the board switch of the carrier board. The `bitstream` parameter is the full path and name of the bit-stream you wish to program.

The `action` parameter specifies what action needs to taken. In the include file 'hrn_fpha.h' (at time of writing) three possibilities are defined:

```
#define HRN_FPGA_LOAD                   0
#define HRN_FPGA_COMPRESS               1
#define HRN_FPGA_EXPAND                 2
```

The first action is to load the FPGA, the second is to compress the bit-stream (*.rbt), the third is to un-compress the bit-stream (*.hcb). You cannot combine actions; you must select one only.

The options parameter specifies some optional switches you may choose to use. In the include file 'hrn_fpga.h' (at time of writing) the following options are defined:

```
#define HRN_FPGA_VERBOSE                1
#define HRN_FPGA_PROMPT                 2
#define HRN_FPGA_NEVERCOMPRESS          4
```

The first option will ask hrn_fpga to show some more information; the second option will ask hrn_fpga to prompt at certain stages of execution; and the third option asks hrn_fpga to first un-compress HCB files before downloading bit-stream data over HSB to program an FPGA. Some modules support downloading of compressed data. When for any reason this doesn't work, this option allows you to try downloading the bit-stream un-compressed.

Example:

```
#include "heapi.h"
#include "hrn_fpga.h"

int main(int argc, char *argv[])
{
      int   r;          char *s;
      hrn_fpga hf;      HE_DWORD Status;
      char *rbtfile;    HE_HANDLE hDevice=NULL;


      rbtfile="i:\\fpga\\io2v2\\Transient_analysis(ex2)\\2v1000fg456.hcb";

      Status = HeOpen("hep9a", 1, HSB, &hDevice);
```

```
        if (Status!=HE_OK)
        {
              printf("Open error %x\n", Status);
              return -1;
        }
        r = hf.hrn_fpga_handle(hDevice, 0x13, rbtfile, HRN_FPGA_LOAD, 0);
        if (r)
        {
              s = hf.hrn_fpga_getlasterr();
              if ((s!=NULL)&&(s[0]!=0)) printf("%s\n", s);
              else printf("Error %d.\n", r);
              return r;
        }
        return 0;
}
```

In this example, we try to program the HERON-IO2V2 example 2 bit-stream onto a module in slot 3 of an HEPC9 with board-switch 1.

The 'hrn_fpga_handle' function returns 0 upon success, and not 0 upon error. If the return value indicates an error, the return value is itself an error value. The 'hrn_fpga_getlasterr' will return a descriptive string of the error encountered. Although it should always return a non-NULL pointer, a defensive programmer would sensibly check for the return value to be not NULL and not 0 in length.

## Generic version with board specifiers

A variation on the generic version with handle is a function that instead of taking a handle as parameter takes a board specification as parameter: -

```
int hrn_fpga_board(char *devstr, int sw, int heronid, char *bstream,
                                                int action , int   options);
```

Parameter `devstr` specifies a board-type (e.g. "hep9a", "hep8a", etc). Parameter `sw` specifies that board's board switch. The heronid specifies the board number and slot where the target module is on. Bits 0..3 of the heronid identify the slot number (1=slot 1, 2=slot 2, 3=slot 3, 4=slot 4). Bits 4..7 identify the board switch of the carrier board. The `bstream` parameter is the full path and name of the bit-stream you wish to program.

The `action` parameter specifies what action needs to taken. In the include file 'hrn_fpha.h' (at time of writing) three possibilities are defined:

```
#define HRN_FPGA_LOAD                   0
#define HRN_FPGA_COMPRESS               1
#define HRN_FPGA_EXPAND                 2
```

The first action is to load the FPGA, the second is to compress the bit-stream (*.rbt), the third is to un-compress the bit-stream (*.hcb). You cannot combine actions; you must select one only.

The options parameter specifies some optional switches you may choose to use. In the include file 'hrn_fpga.h' (at time of writing) the following options are defined:

```
#define HRN_FPGA_VERBOSE                1
#define HRN_FPGA_PROMPT                 2
#define HRN_FPGA_NEVERCOMPRESS          4
```

The first option will ask hrn_fpga to show some more information; the second option will ask hrn_fpga to prompt at certain stages of execution; and the third option asks hrn_fpga to

first un-compress HCB files before downloading bit-stream data over HSB to program an FPGA. Some modules support downloading of compressed data. When for any reason this doesn't work, this option allows you to try downloading the bit-stream un-compressed.

Example:

```
#include "heapi.h"
#include "hrn_fpga.h"

int main(int argc, char *argv[])
{
      int   r;          char *s;
      hrn_fpga hf;
      char *rbtfile;    HE_HANDLE hDevice=NULL;


      rbtfile="i:\\fpga\\io2v2\\Transient_analysis(ex2)\\2v1000fg456.hcb";

      r = hf.hrn_fpga_board("hep9a", 1, 0x13, rbtfile, HRN_FPGA_LOAD, 0);
      if (r)
      {
            s = hf.hrn_fpga_getlasterr();
            if ((s!=NULL)&&(s[0]!=0)) printf("%s\n", s);
            else printf("Error %d.\n", r);
            return r;
      }
      return 0;
}
```

In this example, we try to program the HERON-IO2V2 example 2 bit-stream onto a module in slot 3 of an HEPC9 with board-switch 1.

The 'hrn_fpga_handle' function returns 0 upon success, and not 0 upon error. If the return value indicates an error, the return value is itself an error value. The 'hrn_fpga_getlasterr' will return a descriptive string of the error encountered. Although it should always return a non-NULL pointer, a defensive programmer would sensibly check for the return value to be not NULL and not 0 in length.

## Windows (and other non-console) programs

All 4 forms of using hrn_fpga library can be used in console and non-console programs. By default, hrn_fpga will print progress messages to the standard output (stdout). For console programs this is fine, but for non-console programs (e.g. a Windows application) you may want to redirect print messages to somewhere else, like some application window. In that case, use the 'hrn_fpga_set_userprint' to replace the print function with your own: -

```
void  hrn_fpga_set_userprint(HRN_FPGA_USER_PRINT fie);
```

The function you must define yourself is of type `HRN_FPGA_USER_PRINT`, which is defined in hrn_fpga.h as follows: -

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

So the print function you need to create is a void function with one parameter, a pointer to a string. The string will be input by hrn_fpga.

For the generic functions, if prompting isn't used, the above is the only issue with non-console programs. But when prompting is used you may want to redirect other functions as well. There are three other functions (at time of writing) that you may redirect to use your

own function. The functions are for prompting, asking the user for a string, and asking the user to reply 'yes' or 'no'. To replace these functions, you can use: -

```
int hrn_fpga_set_userfie(HRN_FPGA_USER_FIE fie, int which);
```

The `which` parameter decides what function you want to replace / redirect. In hrn_fpga.h the following values for `which` are defined: -

```
#define HRN_FPGA_PROMPTFIE                  0
#define HRN_FPGA_ASKSTR                     1
#define HRN_FPGA_ASKYN                      2
```

The function prototype for each of the 3 functions is identical, although not in all cases all parameters are actually used: -

```
typedef int (*HRN_FPGA_USER_FIE)(char *str, void *arg);
```

We will now discuss each of the 3 functions that you can redirect in more detail.

### Prompting

Prompting can be selected in all of the 4 hrn_fpga functions. When prompting is enabled, hrn_fpga may ask the user things that aren't clear. For example, it may ask for a bit-stream, if none was specified, or it may ask the user whether to use a certain module if no slot was specified.

Enabling prompting will also cause 'hrn_fpga' to prompt before doing certain actions. A string will be printed, and then hrn_fpga waits for a key-press. Places where prompting occurs are before programming a bit-stream, after successful programming the FPGA, and in the help screen to separate successive help pages. When prompting is not selected / used there's no need to redirect the hrn_fpga prompt function.

The standard implementation by hrn_fpga is simply a printf followed by a key press wait:

```
printf("%s\n", str);
getchar();
```

For example, a possibly suitable Windows function could use MessageBox: -

```
int myprompt(char *str, void *arg)
{
      MessageBox(NULL, str, "header", MB_OK); return 0;
}
```

The return value isn't used, so you may return any value. The arg parameter is always passed NULL by hrn_fpga.

### Ask string

The command-line functions may ask the user for a board-type, board-number or bit-stream if prompting is enabled. A board-type will be asked if no board-type was specified using the –d option, and the board-type could not successfully be extracted from environment variables HEAPI_HSB or HEAPIJTAG. Similarly, a board-number will be asked if no board-number was specified using the –b option, and the board-number could not successfully be extracted from environment variables HEAPI_HSB or HEAPIJTAG. A bit-stream will be asked if none was specified using the –f option. When prompting is not selected / used there's no need to redirect the hrn_fpga ask string function.

The standard implementation by hrn_fpga is simply 2 printf's followed by a gets: -

```
printf("%s\n", str);      // show what is asked for
printf(">");
gets(answer);
```

There isn't a short example showing how to redirect the hrn_fpga ask string function, but, for example, you could think of using an edit box within a small window: -

```
int myaskstring(char *str, void *answer)
{
      // create a window with an edit box,
      // write the text contained in 'str' in the window,
      // and write the user data back into 'answer'.
}
```

The return value isn't used, so you may return any value. The answer parameter is always passed as a character pointer to a buffer defined in hrn_fpga. The buffer size is 32 when asking for board-type, and 256 when asking for board number and bit-stream.

### Ask yes/no

The command-line functions may ask the user for confirmation when selecting a *.rbt or a *.hcb bit-stream, or for confirming that the bit-stream should be programmed into a choice of detected modules. That is, if prompting is enabled. When prompting is not selected there's no need to redirect the hrn_fpga ask yes/no function.

The standard implementation by hrn_fpga is simply a printf's followed by a gets: -

```
char tmp[256];
printf("%s\n", str);            // yes/no question
gets(tmp);
return tmp[0];
```

For example, a possibly suitable Windows function could use MessageBox: -

```
int myaskyn(char *str, void *arg)
{
      int r;
      r = MessageBox(NULL, str, "header", MB_YESNO);
      if (r==IDYES) return 'y';
      return 0;
}
```

The return value must be 'y' or 'Y' to confirm the question, or any other value to reply 'no'. The arg parameter is always passed NULL by hrn_fpga.

## Help screen

The hrn_fpga library has a built-in help screen function. As versions get updated this help screen will get updated as well. The function prototype is in hrn_fpga.h: -

```
void hrn_fpga_help(void);
```

Although its main use would be in console programs, it could also be used in Windows and other non-console programs. All text is redirected if you replaced / redirected the print function and the prompt function. (I.e. used 'hrn_fpga_set_userprint' and 'hrn_fpga_set_ userfie' with `HRN_FPGA_PROMPTFIE` to replace hrn_fpga's functions.)

### Version information

Successive hrn_fpga versions may add functionality, such as new functions, new interfaces, more options or more actions. To allow your program to be able to differentiate between versions, you can ask hrn_fpga for its version number. The function prototype is, as usual, in hrn_fpga.h, as follows:-

```
void hrn_fpga_version(int *major, int *minor, char **vstr);
```

The first two parameters give a numerical interpretation of the version. The third parameter gives a character string representation of the version, typically including a date. One would expect a certain matching between the first two parameters and the third.

### Remote access

It is also possible to program HERON-FPGA or HERON-IO modules located on remote boards. A remote board is a board connected to another board via Inter-Board Connectors, such as EM2 or EM1 modules. A remote board could be embedded, or be located in another PC, or be in the same PC.

To remotely program an FPGA, you must have selected or created a network file via Tools → Init Remote. This network file should define inter-board connections between boards.

With the generic form, the 'heronid' parameter lets you define the target slot's identifier. The handle or board specifiers select the board used to access HSB. For the command-line forms, use the '-s' and –'t' options. The '-s' option selects a slot, the '-t' option selects a board number (of the board on which the slot is located). Together the information of '-s' and '-t' is used to create a heronid. In case there's no '-t' option given, hrn_fpga assumes that the slot is on the same board as that given by the '-b' option.

## File types used

You can specify a .rbt or a .hcb file, or simply no extension to the filename. If the file exists that you specify it will be used. If you don't specify an extension it will use whichever exists, unless they both exist when it will use the .rbt file.

If you do specify an extension and the file does not exist, if you have not selected the unconditional mode (-u option) , Hrn_fpga will look for the other type and offer it to you if it exists.

Whenever you power up your system the FPGA will be "empty".

If you have the "PROM Boot" jumper fitted, then the FPGA will be configured soon after using the contents of the PROM.

If you do not have the PROM Boot jumper fitted then you need to supply the configuration using hrn_fpga.exe.

System Reset DOES NOT clear the FPGA contents so unless you need a different configuration to be loaded, there is no need to reload the FPGAs after a reset.

## Hrn_Fpga and Linux

With Linux, hrn_fpga has the same options, features and functionality as explained in all of the sections above. However, there is no GUI interface to hrn_fpga when using Linux.

## Hrn_Fpga and VxWorks

With VxWorks, hrn_fpga has the same options, features and functionality as explained in all of the sections above. However, there is no GUI interface to hrn_fpga when using VxWorks.

Hrn_fpga is part of the 'hrnfpga.o' file. This file holds both the 'executable' and the library form of hrn_fpga. To use it, you would have to have 'heapi.o' loaded first: -

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

With VxWorks, because of the hassle with user input, the utility must be used with all options used (except verbose, which remains optional). To run:

```
sp HrnFpga,"-fbitstream.rbt -dhep9a -b0 -s2 -u -v"
```

This will try to load a bit-stream called 'bitstream' (-f*bitstream.rbt*) onto an FPGA/HERON-IO module located on a HEPC9 (-d*hep9a*), with board switch set to 0 (-b*0*), in slot 2 (-s*2*). The mode is unconditional (-u) and here the optional verbose option is used (-v).

The VxWorks utility ('hrnfpga.o') is located on the CD in \software\api\vxworks, and has also been installed in your installation directory if you used the HUNT ENGINEERING CD setup program. Assuming the default installation directory, the file would be in c:\heapi\vxworks.

## Hrn_Fpga and RTOS-32

With RTOS-32, hrn_fpga has the same options, features and functionality as explained in the sections above. However, there's no GUI interface to hrn_fpga when using RTOS-32.

The executable form of hrn_fpga for RTOS-32 is located in hrn_fpga\bin\rtos32 in your API&Tools installation directory. Both an executable 'hrnfpga.exe' and 'hrnfpga.rtb' are provided in this directory. Use the RTB file to create a floppy disk: -

```
bootdisk hrnfpga a:
```

and copy the *.hcb files that you want to program onto the floppy, or onto the harddisk of the target machine. You can simply run 'hrnfpga' off the floppy-disk, and it will ask you for the board type and number, slot number and what file to use to program the FPGA with.

Alternatively, use a command-line in the 'hrnfpga.cfg' file, for example as follows: -

```
Commandline "hrnfpga.exe -fbitstream.rbt -dhep9a -b0 -s2 -u -v"
```

This will try to load a bit-stream called 'bitstream' (-f*bitstream.rbt*) onto an FPGA/HERON-IO module located on a HEPC9 (-d*hep9a*), with board switch set to 0 (-b*0*), in slot 2 (-s*2*). The mode is unconditional (-u) and here the optional verbose option is used (-v).

**Error 9000**. Slot %d is not a valid slot (must be 1,2,3 or 4).

Via the '-s' parameter you have given an invalid slot number. There are only 4 slots on boards such as the HEPC8 or HEPC9, and they are numbered 1 to 4.

**Error 9001**. Board %d not a valid board (must be 0..15).

Via the '-b' parameter you have given an invalid board number. The red switch on boards such as the HEPC8 and HEPC9 can only be set to 0 to 15.

**Error 9002**. Unknown option %s.

You have used an option that is unknown to 'hrn_fpga'.

**Error 9003**. A con file was defined twice or more.

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. This error indicates that two or more arguments without '-' are detected, but 'hrn_fpga' doesn't know which argument is meant to be the *.con file.

**Error 9004**. %s does not have a .con extension. If you meant to provide a bitstream use the -f option.

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. This error indicates that the argument doesn't have a '.con' extension. A proper *.con file should have a '.con' extension so that 'hrn_fpga' knows it isn't a mistake and you supply a proper, correct configuration file.

**Error 9005**. Target board %d not a valid board (must be 0..15).

Via the '-t' parameter you have given an invalid board number. The red switch on boards such as the HEPC8 and HEPC9 can only be set to 0 to 15.

**Error 9020**. Failed to find/open file "%s".

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. This error indicates that the *.con file doesn't exist or cannot be opened. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9021**. Cannot find/open %s.rbt or %s.hcb file.

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. This error indicates that a file (no extension) listed in the *.con file doesn't exist or cannot be opened. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9022**. Cannot find/open %s file.

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. This error indicates that a *.rbt file listed in the *.con file doesn't exist or cannot be opened. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9023**. Cannot find/open %s file.

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. This error indicates that a *.hcb file listed in the *.con file doesn't exist or cannot be opened. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9024**. Incorrect file extension in %s (should be .rbt or .hcb or none).

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. Files listed in a *.con file must have an extension '.rbt' or '.hcb', or have no extension at all.

**Error 9025**. Invalid board number %d (should be 0..15).

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. An entry in the *.con file uses an invalid board number. The red switch on boards such as the HEPC8 and HEPC9 can only be set to 0 to 15.

**Error 9026**. Invalid slot number %d (should be 1,2,3 or 4).

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. An entry in the *.con file uses an invalid slot number. There are only 4 slots, numbered 1 to 4, on boards such as the HEPC8 or HEPC9.

**Error 9027**. No HERON-FPGA or HERON-IO modules were configured.

If you pass an argument to 'hrn_fpga' without a '-' in front of it, the argument is interpreted as a *.con file. A *.con file is a configuration file that holds a list of *.rbt or *.hcb files that you want to program, together with board type (name), board switch and slot number. This error is more like a warning, telling you that it has processed all lines in the *.con file, but found no entries.

**Error 9040**. Cannot find/open %s.rbt file.

The file that you specified with the '-f' function doesn't exist or cannot be opened. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9041**. Cannot find/open %s.rbt or %s.hcb file.

The file that you specified with the '-f' function doesn't exist or cannot be opened.

The 'hrn_fpga' function tried with both '.rbt' and '.hcb' appended to the filename. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9042**. Cannot find/open %s.hcb file.

The file that you specified with the '-f' function doesn't exist or cannot be opened. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9043**. Sorry, you cannot expand a .rbt file.

You use option '-r' to indicate you want to expand a bit stream, but you specify an already expanded bit stream (*.rbt file). You cannot expand more than once.

**Error 9044**. Cannot find/open %s file.

The '.rbt' file that you specified with the '-f' function doesn't exist or cannot be opened. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9045**. Cannot find/open %s file.

The '.rbt' file that you specified with the '-f' function doesn't exist or cannot be opened. The 'hrn_fpga' function also looked for an '.hcb' file of the same name, but such a file could not be opened either. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9046**. Cannot find/open %s file.

The '.rbt' file that you specified with the '-f' function doesn't exist or cannot be opened. The 'hrn_fpga' function also looked for an '.hcb' file of the same name, but when asked, the user replied the '.hcb' file should not be used. Verify the '.rbt' file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9047**. Sorry, cannot compress a .hcb file.

You use option '-c' to indicate you want to compress a bit stream, but you specify an already compressed bit stream (*.hcb file). You cannot compress more than once.

**Error 9048**. Cannot find/open %s file.

The '.hcb' file that you specified with the '-f' function doesn't exist or cannot be opened. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9049**. Cannot find/open %s file.

The '.hcb file that you specified with the '-f' function doesn't exist or cannot be opened. The 'hrn_fpga' function also looked for an '.rbt' file of the same name, but such a file could not be opened either. Verify the file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9050**. Cannot find/open %s file.

The '.hcb' file that you specified with the '-f' function doesn't exist or cannot be opened. The 'hrn_fpga' function also looked for an '.rbt' file of the same name, but when asked, the user replied the '.rbt' file should not be used. Verify the '.hcb' file's existence, make sure it's spelled correctly, and that the file's security settings allow

opening by the current user.

**Error 9051**. Incorrect file extension in %s (should be *.rbt or *.hcb, or none).

The file that you specified with the '-f' function must have an extension '.rbt' or '.hcb', or have no extension at all.

**Error 9052**. Error 9052. No RBT, HCB, or BIT file specified (-f option).

Please use the '-f' option to select the bit stream that should be programmed.

**Error 9053**. Sorry, compressing a .bit file is not supported.

Converting a *.bit file to a *.hcb file is not supported. Please don't use the –c option with *.bit files.

**Error 9054**. Sorry, expanding a .bit file is not supported.

Converting a *.bit file to a *.rbt file is not supported. Please don't use the –r option with *.bit files.

**Error 9055**. Cannot find/open %s file.

The '.bit' file that you specified with the '-f' function doesn't exist or cannot be opened. The 'hrn_fpga' function also looked for an '.rbt' and '.hcb' file of the same name, or if prompted, the user replied the '.rbt' or '.hcb' file should not be used. Verify the '.bit' file's existence, make sure it's spelled correctly, and that the file's security settings allow opening by the current user.

**Error 9060**. Cannot send HSB message to slot 0x%x (via "%s %d") (%s).

The 'hrn_fpga' function is unable to send a MODULE_TYPE_QUERY message to the target module. The MODULE_TYPE_QUERY message asks a module what type it is. But the API function used, `HeHSBSendMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9061**. Cannot receive HSB message, board "%s %d" (%s).

The 'hrn_fpga' function is has successfully sent a MODULE_TYPE_QUERY message to the target module. The MODULE_TYPE_QUERY message asks a module what type it is. It now waits for a reply, but the target module doesn't answer. The error returned by API function used, `HeHSBReceiveMessageEx`, is shown at the end of the error message. Please refer to the API manual for more information on the API error reported.

**Error 9062**. Expected module type query reply (0x%x), received 0x%x, board "%s %d"

The 'hrn_fpga' function is has successfully sent a MODULE_TYPE_QUERY message to the target module. The MODULE_TYPE_QUERY message asks a module what type it is. It has also successfully received a reply from the target module. However, the 'hrn_fpga' function expected a MODULE_TYPE_REPLY message type but a different message type was found in the reply.

**Error 9070**: You cannot compress a compressed (.hcb) file.

You use option '-c' to indicate you want to compress a bit stream, but you specify an already compressed bit stream (*.hcb file). You cannot compress more than once.

**Error 9071**: Internal error.

The software finds that you didn't select an *.rbt file nor a *.hcb file, even though the user interface should have forced a choice between them. Possibly the error may occur with memory (heap or stack) problems.

**Error 9072**: You cannot expand a .rbt file.

You use option '-r' to indicate you want to expand a bit stream, but you specify an already expanded bit stream (*.rbt file). You cannot expand more than once.

**Error 9073**: Internal error.

The software finds that you didn't select an *.rbt file nor a *.hcb file, even though the user interface should have forced a choice between them. Possibly the error may occur with memory (heap or stack) problems.

**Error 9074**. Please define HEAPIHSB and HEAPIJTAG, or select a board type.

When environment variables HEAPIHSB or HEAPIJTAG are defined, 'hrn_fpga' will use those to retrieve a board type and board switch, if they are not given by the '-d' and '-b' options. This error is returned if the board type is not given, but no environment variable is defined, and it is thus impossible for 'hrn_fpga' to retrieve a board type. Either specify a board type using the '-d' parameter or define one of the environment variables. The API installation should have created one or both of the environment variables, though.

**Error 9075**. Please define HEAPIHSB and HEAPIJTAG, or select a board number.

When environment variables HEAPIHSB or HEAPIJTAG are defined, 'hrn_fpga' will use those to retrieve a board type and board switch, if they are not given by the '-d' and '-b' options. This error is returned if the board switch is not given, but no environment variable is defined, and it is thus impossible for 'hrn_fpga' to retrieve a board number (switch). Either specify a board number using the '-b' parameter or define one of the environment variables. The API installation should have created one or both of the environment variables, though.

**Error 9076**. Invalid board number %d (should be 0..15).

Via the '-b' parameter you have given an invalid board number. The red switch on boards such as the HEPC8 and HEPC9 can only be set to 0 to 15.

**Error 9077**. Invalid slot number %d (should be 1,2,3 or 4).

Via the '-s' parameter you have given an invalid slot number. There are only 4 slots on boards such as the HEPC8 or HEPC9, and they are numbered 1 to 4.

**Error 9090**. Cannot send HSB message to slot 0x%x, board "%s %d" (%s).

The 'hrn_fpga' function is unable to send a COMPRESSED_QUERY message to the target HERON-FPGA. The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format. The string at the end is the description if the API error code returned by API function HeHSBSendMessage.

**Error 9091**. Cannot receive HSB message to slot 0x%x, board "%s %d" (%s).

The 'hrn_fpga' function has successfully sent a COMPRESSED_QUERY message to the target HERON-FPGA module, but it has not received a reply from the target module. The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format. The string at the end is the description if the API error code returned by API function HeHSBReceiveMessage.

**Error 9092**. Error in reply to "Do you support compressed loading?". (Got 0x%d, expected 0x%x). Board "%s %d".

The 'hrn_fpga' function has successfully sent a COMPRESSED_QUERY message to the target HERON-FPGA module, and it has successfully received a reply from the target module. But the message type of the return message is not COMPRESSED_REPLY as expected. The 'got %d' part shows the message type that was actually received, instead of the expected COMPRESSED_REPLY (==12). The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format.

**Error 9093**. Cannot send HSB message to slot 0x%x, board "%s %d" (%s).

The 'hrn_fpga' function is unable to send a COMPRESSED_QUERY message to the target HERON-IO. The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format. The string at the end is the description if the API error code returned by API function HeHSBSendMessage.

**Error 9094**. Cannot receive HSB message to slot 0x%x, board "%s %d" (%s).

The 'hrn_fpga' function has successfully sent a COMPRESSED_QUERY message to the target HERON-IO module, but it has not received a reply from the target module. The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format. The string at the end is the description if the API error code returned by API function HeHSBReceiveMessage.

**Error 9095**. Error in reply to "Do you support compressed loading?". (Got 0x%d, expected 0x%x). Board "%s %d".

The 'hrn_fpga' function has successfully sent a COMPRESSED_QUERY message to the target HERON-IO module, and it has successfully received a reply from the target module. But the message type of the return message is not COMPRESSED_REPLY as expected. The 'got %d' part shows the message type that was actually received, instead of the expected COMPRESSED_REPLY (==12). The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format.

**Error 9096**. No HERON-FPGA or HERON-IO modules were configured.

This is a warning, rather than an error. The 'hrn_fpga' function has searched the board for modules with a user FPGA (such as HERON-FPGA and HERON-IO modules), but none were found; or the modules that were found were not the target module, when prompting the user.

**Error 9100**. Failed to read header section of RBT file %s.

The 'hrn_fpga' function was unable to read the first line of the *.rbt file.

**Error 9101**. Invalid first line in RBT file %s.

To verify that the file read is really a *.rbt file, the 'hrn_fpga' function checks if the first line of the *.rbt file matches "Xilinx ASCII Bitstream\n". If the first line of the file is different, this error is returned, indicating that it is most likely not a *.rbt file.

**Error 9102**. Failed to read header section of RBT file %s.

The 'hrn_fpga' function was unable to read line 2, 3, 4 or 5 of the '*.rbt file.

**Error 9103**. Bitstream %s is not for a %s device.

The target module that you have specified (on board '-d' with switch '-b', slot '-s') has a different FPGA device type or module type than is specified in the *.rbt file header. Errors like this will happen if you try to load, for example, a HERON-IO5 bit stream onto, for example, a HERON-FPGA7.

**Error 9104**. Failed to read header section of RBT file %s.

The 'hrn_fpga' function was unable to read line 6 or 7 of the *.rbt file.

**Error 9105**. Cannot start HSB message, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' message is trying to start sending data to the target module, but the API function used, `HeHSBStartSendMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9106**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 2 bytes (compressed bit stream select) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9107**. Incorrect line length (%d), bit_count = %d, in RBT file %s.

Every line in a *.rbt file that relates to bits should be 32 digits ('0' or '1') long. But when converting the *.rbt file to a *.hcb file, the 'hrn_fpga' function found a line longer than 32 digits ('0' or '1' digits). Verify that the *.rbt file isn't corrupted.

**Error 9108**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 1 byte (compressed count, from a *.rbt file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9109**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 1 byte (compressed value, from a *.rbt file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9110**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send last byte (compressed count, from a *.rbt file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9111**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send last byte (compressed value, from a *.rbt file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9112**. Incorrect line length (%d), bit_count = %d, in RBT file %s.

Every line in a *.rbt file that relates to bits should be 32 digits ('0' or '1') long. But

when converting the *.rbt file to a *.hcb file, the 'hrn_fpga' function found a line longer than 32 digits ('0' or '1' digits). Verify that the *.rbt file isn't corrupted.

**Error 9113**. Incorrect number of bits (%d, %d) in RBT file %s.

The 'hrn_fpga' function has counted the number of bits ('0' or '1' digits) in the *.rbt file. But this number is not equal to the number of bits as declared by the *.rbt file in line 7.

**Error 9113**. Configuration successfull, but found incorrect number of bits (%d, %d) in RBT file %s.

The 'hrn_fpga' function has counted the number of bits ('0' or '1' digits) in the *.rbt file. But this number is not equal to the number of bits as declared by the *.rbt file in line 7. However, the user FPGA responds that it has been successfully configured.

**Error 9114**. Cannot end HSB message, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to end the HSB message, used to program the user FPGA, but the API function used, `HeHSBEndOfSendMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9115**. Cannot start receive HSB message (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.rbt bit stream bits, but when trying to start reading a response from the user FPGA, the API function used, `HeHSBStartReceiveMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9116**. Cannot receive HSB message data (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.rbt bit stream bits, but when trying to read a response from the user FPGA, the API function used, `HeHSBReceiveMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9117**. Cannot end HSB message (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.rbt bit stream bits, but when trying to end reading a response from the user FPGA, the API function used, `HeHSBEndOfReceiveMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9118**. Configuration was unsuccessful (%d).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.rbt bit stream bits. And it has successfully received a reply from the user FPGA. However, the user FPGA replies that the configuration was not successful. The cause is sometimes (but certainly not always) an incorrect design. At the end of the error message the user FPGA's reply is shown (integer %d).

**Error 9119**. Configuration unsuccessfull. Incorrect number of bits (%d, %d) in RBT file %s.

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.rbt bit stream bits. And it has successfully received a reply from

the user FPGA. The user FPGA replies that the configuration was not successful. The 'hrn_fpga' function has counted the number of bits ('0' or '1' digits) in the *.rbt file. It found that this number is not equal to the number of bits as declared by the *.rbt file in line 7.

**Error 9120**. Failed to read header section of RBT file %s.

The 'hrn_fpga' function was unable to read line 7 of the '*.rbt file.

**Error 9121**. Failed to find bit size of RBT %s.

A *.rbt file has a line (7) that details how many bits there are defined/used in the file. This line has not been detected in the *.rbt file that you specified with the '-f' option.

**Error 9140**. Cannot send HSB message data, board "%s %d" (%s).

The 'hrn_fpga' function is trying to send a line of data (32 bits, from a *.rbt file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9150**. Failed to read header section of HCB file %s.

The 'hrn_fpga' function was unable to read the first 8 bytes of the *.hcb file.

**Error 9151**. Invalid magic numbers in HCB file %s.

A proper *.hcb file has 8 bytes at the start of the file that form a 'magic number'. When reading a *.hcb file, the 'hrn_fpga' function tests for the magic number. If the first 8 bytes don't match the 'magic number', the file isn't a *.hcb file or is corrupted.

**Error 9152**. Failed to read device type from HCB file %s.

Tthe 'hrn_fpga' function was unable to read the device type string, located after the first 8 bytes, of the *.hcb file.

**Error 9153**. Bitstream %s is not for a %s device.

The target module that you have specified (on board '-d' with switch '-b', slot '-s') has a different FPGA device type or module type than is specified in the *.hcb file header. Errors like this will happen if, for example, you try to load a HERON-IO2 bit stream onto, for example, a HERON-FPGA3.

**Error 9154**. Failed to read timestamp from HCB file %s.

The 'hrn_fpga' function was unable to read line 6, part of the header, of the *.hcb file.

**Error 9155**. Failed to read number of bytes from HCB file %s.

The 'hrn_fpga' function was unable to read 4 bytes, part of the header, denoting the number of bits, of the *.hcb file.

**Error 9156**. Cannot start HSB message, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' message is trying to start sending data to the target module, but the API function used, `HeHSBStartSendMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9157**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 2 bytes (compressed bit stream select) to the

target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9158**. Failed to read of HCB file %s.

The 'hrn_fpga' function was unable to read a byte (compressed count) of the hcb file.

**Error 9159**. End of HCB file %s reached unexpectedly.

The 'hrn_fpga' function detects the end of the file while it expects to read more data.

**Error 9160**. Failed to read of HCB file %s.

The 'hrn_fpga' function was unable to read a byte (compressed value) of the hcb file.

**Error 9161**. End of HCB file %s reached unexpectedly.

The 'hrn_fpga' function detects the end of the file while it expects to read more data.

**Error 9162**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 1 byte (compressed count, from a *.hcb file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9163**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 1 byte (compressed value, from a *.hcb file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9164**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 32 bytes (expanded data from a *.hcb file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9165**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send last byte (compressed count, from a *.hcb file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9166**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send last byte (compressed value, from a *.hcb file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9167**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send last 32 bytes (expanded data from a *.hcb file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9168**. Cannot end HSB message, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to end the HSB message, used to program the user FPGA, but the API function used, `HeHSBEndOfSendMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9169**. Cannot start receive HSB message (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.hcb bit stream bits, but when trying to start reading a response from the user FPGA, the API function used, `HeHSBStartReceiveMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9170**. Cannot receive HSB message data (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.hcb bit stream bits, but when trying to read a response from the user FPGA, the API function used, `HeHSBReceiveMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9171**. Cannot end receive HSB message (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.hcb bit stream bits, but when trying to end reading a response from the user FPGA, the API function used, `HeHSBEndOfReceiveMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9172**. Configuration was unsuccessful %d.

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.hcb bit stream bits. And it has successfully received a reply from the user FPGA. However, the user FPGA replies that the configuration was not successful. The cause is sometimes (but certainly not always) an incorrect design. At the end of the error message the user FPGA's reply is shown (integer %d).

**Error 9190**. Cannot do HeGetBoardInfo, board "%s %d" (%s).

The 'hrn_fpga' function tries to make sure that a module is present in the slot on the board specified by the user. It uses API function `HeGetBoardInfo` for that. However, this function returns an API error, shown at the end of the error message. Please refer to the API manual for more information on the API error reported. Typically you have specified the wrong board (type) or board switch.

**Error 9200**: Invalid board number %d (should be 0..15).

Via the '-b' parameter you have given an invalid board number. The red switch on boards such as the HEPC8 and HEPC9 can only be set to 0 to 15.

**Error 9201**. Cannot open HSB device on board "%s %d" (%s).

The 'hrn_fpga' function tried to open HSB on the target board, but was unsuccessfull. It used API function `HeOpenS`, and the error reported by `HeOpenS` is shown at the end of the error message. Please refer to the API manual for more information on the API error reported.

**Error 9202**. Cannot initialise ReadIoStatus, board "%s %d" (%s).

The 'hrn_fpga' function tried to initialise a read status object, but was unsuccessfull. It used API function `HeInitIoStatus`, and the error reported by `HeInitIoStatus` is shown at the end of the error message. Please refer to the API manual for more information on the API error reported.

**Error 9203**. Cannot initialise WriteIoStatus, board "%s %d" (%s).

The 'hrn_fpga' function tried to initialise a write status object, but was unsuccessfull. It used API function `HeInitIoStatus;` the error reported by `HeInitIoStatus` is shown at the end of the error message. Please refer to the API manual for more information on the API error reported.

**Error 9215**. No function type %d defined. Use type 0,1 or 2.

In the `hrn_fpga_set_userfie` function you have used an invalid number for the `which` parameter. Valid values are `HRN_FPGA_PROMPTFIE`, `HRN_FPGA_ASKSTR` and `HRN_FPGA_ASKYN`. These values are defined in 'hrn_fpga.h'.

**Error 9220**. There is no module in slot %d of board %s %d.

The 'hrn_fpga' function cannot detect a module in the slot specified by the '-s' option on the board specified by the '-d' (board type) and '-b' (board number) options.

**Error 9221**. The module in slot %d of board %s %d is a processor module.

The 'hrn_fpga' function cannot detect a module with a user programmable FPGA (such as a HERON-FPGA or HERON-IO) in the slot specified by the '-s' option on the board specified by the '-d' (board type) and '-b' (board number) options.

**Error 9222**. The module in slot %d of board %s %d does not have a serial bus."

The 'hrn_fpga' function cannot detect a module with a user programmable FPGA (such as a HERON-FPGA or HERON-IO) in the slot specified by the '-s' option on the board specified by the '-d' (board type) and '-b' (board number) options.

**Error 9223**. Cannot send HSB message to slot 0x%x (via "%s %d") (%s).

The 'hrn_fpga' function is unable to send a COMPRESSED_QUERY message to the target module. The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format. The string at the end is the description if the API error code returned by API function HeHSBSendMessage.

**Error 9224**. Cannot receive HSB message (via board "%s %d") (%s).

The 'hrn_fpga' function has successfully sent a COMPRESSED_QUERY message to the target module, but it has not received a reply from the target module. The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format. The string at the end is the description if the API error code returned by API function HeHSBReceiveMessage.

**Error 9225**. Error in reply to "do you support compressed loading?", got %d, board %s %d

The 'hrn_fpga' function has successfully sent a COMPRESSED_QUERY message to the target module, and it has successfully received a reply from the target module. But the message type of the return message is not COMPRESSED_REPLY as expected. The 'got %d' part shows the message type that was actually received, instead of the expected COMPRESSED_REPLY (==12). The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format.

**Error 9226**. Cannot send HSB message to slot 0x%x (via "%s %d") (%s).

The 'hrn_fpga' function is unable to send a COMPRESSED_QUERY message to the target module. The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format. The string at the end is the description if the API error code returned by API function HeHSBSendMessage.

**Error 9227**. Cannot receive HSB message (via "%s %d") (%s).

The 'hrn_fpga' function has successfully sent a COMPRESSED_QUERY message to the target module, but it has not received a reply from the target module. The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format. The string at the end is the description if the API error code returned by API function HeHSBReceiveMessage.

**Error 9228**. Error in reply to "do you support compressed loading?", got %d, board %s %d

The 'hrn_fpga' function has successfully sent a COMPRESSED_QUERY message to the target module, and it has successfully received a reply from the target module. But the message type of the return message is not COMPRESSED_REPLY as expected. The 'got %d' part shows the message type that was actually received, instead of the expected COMPRESSED_REPLY (==12). The COMPRESSED_QUERY message asks a module if it supports sending bit stream data in compressed format.

**Error 9240**. Failed to read header section of RBT file %s.

The 'hrn_fpga' function was unable to read the first line of the *.rbt file.

**Error 9241**. Invalid first line in RBT file %s.

To verify that the file read is really a *.rbt file, the 'hrn_fpga' function checks if the first line of the *.rbt file matches "Xilinx ASCII Bitstream\n". If the first line of the file is different, this error is returned, indicating that it is most likely not a *.rbt file.

**Error 9242**. Cannot open/create HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function will try to create an *.hcb file that has the same name (but a different extension) as the *.rbt file specified with the '-f' option. This error indicates that an error occurred trying to open/create the *.hcb file. Verify if perhaps the *.hcb file already exists, and if so, verify that its security settings allow write access by the current user.

**Error 9243**. Cannot write header to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to write 4 bytes (magic number, part 1) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9244**. Cannot write header to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to write 4 bytes (magic number, part 2) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9245**. Failed to read header section of RBT file %s.

The 'hrn_fpga' function was unable to read line 2, 3, 4 or 5 of the '*.rbt file.

**Error 9246**. Cannot write header to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to

write a line (line 5 of the *.rbt file) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9247**. Failed to read header section of RBT file %s.

The 'hrn_fpga' function was unable to read line 6 of the '*.rbt file.

**Error 9248**. Cannot write header to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to write a line (line 6 of the *.rbt file) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9249**. Failed to read header section of RBT file %s.

The 'hrn_fpga' function was unable to read line 7 of the '*.rbt file.

**Error 9250**. Cannot write header to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to write a line (line 7 of the *.rbt file) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9251**. Incorrect line length. Bit_count = %d. In RBT file %s.

Every line in a *.rbt file that relates to bits should be 32 digits ('0' or '1') long. But when converting the *.rbt file to a *.hcb file, the 'hrn_fpga' function found a line longer than 32 digits ('0' or '1' digits). Verify that the *.rbt file isn't corrupted.

**Error 9252**. Cannot write HCB file data to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to write a byte (count of compressed data) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9253**. Cannot write HCB file data to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to write a byte (compressed data value) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9254**. Cannot write HCB file data to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to write a last byte (count of compressed data) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9255**. Cannot write HCB file data to HCB file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function was unable to write a last byte (compressed data value) to the *.hcb file. Verify that the disk isn't full and that the section of the disk that the *.hcb file gets written to isn't corrupted.

**Error 9256**. Incorrect number of bits in RBT file %s.

When compressing a *.rbt file to a *.hcb file, the 'hrn_fpga' function has counted the number of bits ('0' or '1' digits) in the *.rbt file. But this number is not equal to the number of bits as declared by the *.rbt file in line 7.

**Error 9257**. Failed to find bit size of RBT file %s.

A *.rbt file has a line (7) that details how many bits there are defined/used in the file.

This line has not been detected in the *.rbt file that you specified with the '-f' option.

**Error 9270**. Failed to read header section of HCB file %s.

The 'hrn_fpga' function was unable to read the first 8 bytes of the *.hcb file.

**Error 9271**. Invalid magic numbers in HCB file %s.

A proper *.hcb file has 8 bytes at the start of the file that form a 'magic number'. When reading a *.hcb file, the 'hrn_fpga' function tests for the magic number. If the first 8 bytes don't match the 'magic number', the file isn't a *.hcb file or is corrupted.

**Error 9272**. Failed to read device type from HCB file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to read the device type string, located after the first 8 bytes, of the *.hcb file.

**Error 9273**. Cannot open/create RTB file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function will try to create an *.rbt file that has the same name (but a different extension) as the *.hcb file specified with the '-f' option. This error indicates that an error occurred trying to open/create the *.rbt file. Verify if perhaps the *.rbt file already exists, and if so, verify that its security settings allow write access by the current user.

**Error 9274**. Failed to write header section of RBT file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to write the first line ("Xilinx ASCII Bitstream\n") to the *.rbt file.

**Error 9275**. Failed to write header section of RBT file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to write lines 2, 3 or 4 to the *.rbt file.

**Error 9276**. Failed to write header section of RBT file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to write line 5 (device type information) to the *.rbt file.

**Error 9277**. Failed to read HCB file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to read line 6, part of the header, of the *.hcb file.

**Error 9278**. Failed to read HCB file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to read 4 bytes, part of the header, denoting the number of bits, of the *.hcb file.

**Error 9279**. Failed to write header section of RBT file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to write line 6 to the *.rbt file.

**Error 9280**. Failed to read HCB file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to read a byte (compressed count) of the *.hcb file.

**Error 9281**. End of HCB file %s reached unexpectedly.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function detects the end of the file while it expects to read more data.

**Error 9282**. Failed to read HCB file %s."

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to read a byte (compressed value) of the *.hcb file.

**Error 9283**. End of hcb file %s reached unexpectedly.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function detects the end of the file while it expects to read more data.

**Error 9284**. Incorrect number of bytes in HCB file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function has counted the number of bits in the *.hcb file. But this number is not equal to the number of bits as declared by the *.hcb file its header.

**Error 9295**. Failed to write to RBT file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to write digit '1' to the *.rbt file.

**Error 9296**. Failed to write to RBT file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to write digit '0' to the *.rbt file.

**Error 9297**. Failed to write to RBT file %s.

When expanding a *.hcb file to a *.rbt file, the 'hrn_fpga' function was unable to write the closing end-of-line characters ('\n') to the *.rbt file.

**Error 9300**: hrn_fpga_handle: The device handle is NULL.

Parameter `hDev` is NULL in the call to `hrn_fpga_handle`.

**Error 9301**: hrn_fpga_handle: Cannot retrieve information from device handle (API error 0x%x).

The 'hrn_fpga_handle' function uses API function `HeGetDeviceInfo` to retrieve the board name and board switch for a `uDevice` entry, but `HeGetDeviceInfo` failed. Use the API error to find what the problem is. Usually the `hDev` entry is NULL or is not yet initialised.

**Error 9302**: hrn_fpga_handle: The 'rbtfile' parameter is NULL.

Parameter `rbtfile` in the call to 'hrn_fpga_handle' is NULL. It should point to a string with the full pathname of the bit-stream you want to load.

**Error 9303**: hrn_fpga_handle: Unknown action %d.

In the call to 'hrn_fpga_handle' you use an invalid value for parameter action. Valid values are `HRN_FPGA_LOAD`, `HRN_FPGA_COMPRESS` and `HRN_FPGA_EXPAND`. These values are defined in 'hrn_fpga.h'.

**Error 9307**: hrn_fpga: Cannot retrieve information from device handle (API error 0x%x).

The hrn_fpga function uses API function `HeGetDeviceInfo` to retrieve the board name and board switch for a `uDevice` entry, but `HeGetDeviceInfo` failed. Use the API error to find what the problem is. Usually a `uDevice` entry is NULL or is not yet initialised.

**Error 9310**: hrn_fpga_handle: The 'devstr' parameter is NULL.

Parameter `devstr` in the call to 'hrn_fpga_board' is NULL. It should point to a string with a board name (of the board on which the -FPGA or -IO module is).

**Error 9311**: hrn_fpga_handle: The 'rbtfile' parameter is NULL.

Parameter `rbtfile` in the call to 'hrn_fpga_board' is NULL. It should point to a string with the full pathname of the bit-stream you want to load.

**Error 9312**: hrn_fpga_handle: Unknown action %d.

In the call to 'hrn_fpga_board' you use an invalid value for parameter action. Valid values are `HRN_FPGA_LOAD`, `HRN_FPGA_COMPRESS` and `HRN_FPGA_EXPAND`. These values are defined in 'hrn_fpga.h'.

**Error 9320**. Failed to read header section of BIT file %s (magic header length).

The 'hrn_fpga' function was unable to read the first 2 bytes of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9321**. Expected length of 9 in header section of BIT file %s (found %d).

The first 2 bytes of the *.bit file were expected to be an integer with the value of 9. But a different value was found. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9322**. Failed to read header section of BIT file %s (magic string).

The 'hrn_fpga' function was unable to read bytes 2 to 10 of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9323**. Header of file %s indicates it is not a BIT file.

The magic number (8 bytes) in the header does not match the magic number for a *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9324**. Failed to read header section of BIT file %s ('a' char length).

The 'hrn_fpga' function was unable to read bytes 11 and 12 of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9325**. Expected length of 1 in header section of BIT file %s (found %d).

Byte 11 and 12 of the *.bit file were expected to be an integer with the value of 1. But a different value was found. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9326**. Failed to read header section of BIT file %s ('a' character).

The 'hrn_fpga' function was unable to read byte 13 of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9327**. Expected to find 'a' in header of BIT file %s, but found %c.

Byte 13 of the *.bit file was expected to be a character with the value of 'a'. But a different value was found. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9328**. Failed to read header section of BIT file %s (ncd string length).

The 'hrn_fpga' function was unable to read bytes 14 and 15 of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9329**. Unable to allocate memory while reading header section of BIT file %s.

To read a variable length field, 'hrn_fpga' tries to allocate some memory using 'malloc'. The error indicates that the memory allocation operation failed.

**Error 9330**. Failed to read header section of BIT file %s (ncd string).

The 'hrn_fpga' function was unable to read n bytes off byte 16 of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9331**. Failed to read header section of BIT file %s ('b' character).

The 'hrn_fpga' function was unable to read a one-byte field in the header of the file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9332**. Expected to find 'b' in header of BIT file %s, but found %c.

The value of a field of the header in the bit file was expected to be a character with the value of 'b'. But a different value was found. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9333**. Failed to read header section of BIT file %s (device string length).

The 'hrn_fpga' function was unable to read a variable length field (null-terminated string) in the header of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9334**. Unable to allocate memory while reading header section of BIT file %s.

To read a variable length field, 'hrn_fpga' tries to allocate some memory using 'malloc'. The error indicates that the memory allocation operation failed.

**Error 9335**. Failed to read header section of BIT file %s (device string).

The 'hrn_fpga' function was unable to read a variable length field (null-terminated string) in the header of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9336**. Bitstream %s is not for a %s device.

The bitstream was created for an FPGA device that is different from the FPGA device that was detected on the target module.

**Error 9337**. Failed to read header section of BIT file %s ('c' character).

The 'hrn_fpga' function was unable to read a one-byte field in the header of the file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9338**. Expected to find 'c' in header of BIT file %s, but found %c.

The value of a field of the header in the bit file was expected to be a character with the value of 'c'. But a different value was found. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9339**. Failed to read header section of BIT file %s (date string length).

The 'hrn_fpga' function was unable to read a variable length field (null-terminated string) in the header of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9340**. Unable to allocate memory while reading header section of BIT file %s.

To read a variable length field, 'hrn_fpga' tries to allocate some memory using 'malloc'. The error indicates that the memory allocation operation failed.

**Error 9341**. Failed to read header section of BIT file %s (date string).

The 'hrn_fpga' function was unable to read a variable length field (null-terminated string) in the header of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9342**. Failed to read header section of BIT file %s ('d' character).

The 'hrn_fpga' function was unable to read a one-byte field in the header of the file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9343**. Expected to find 'd' in header of BIT file %s, but found %c.

The value of a field of the header in the bit file was expected to be a character with the value of 'd'. But a different value was found. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9344**. Failed to read header section of BIT file %s (time string length).

The 'hrn_fpga' function was unable to read a one-byte field in the header of the file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9345**. Unable to allocate memory while reading header section of BIT file %s.

To read a variable length field, 'hrn_fpga' tries to allocate some memory using 'malloc'. The error indicates that the memory allocation operation failed.

**Error 9346**. Failed to read header section of BIT file %s (time string).

The 'hrn_fpga' function was unable to read a variable length field (null-terminated string) in the header of the *.bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9347**. Failed to read header section of BIT file %s ('e' character).

The 'hrn_fpga' function was unable to read a one-byte field in the header of the file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9348**. Expected to find 'e' in header of BIT file %s, but found %c.

The value of a field of the header in the bit file was expected to be a character with the value of 'e'. But a different value was found. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9349**. Failed to read header section of BIT file %s (number of bits).

The 'hrn_fpga' function was unable to read a four-byte field in the header of the file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9350**. Cannot start HSB message, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' message is trying to start sending data to the target module, but the API function used, `HeHSBStartSendMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9351**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 2 bytes (compressed bit stream select) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9352**. Failed to read BIT file %s (after %d bits read, expected %d).

The 'hrn_fpga' function was unable to read a byte (compressed count) of the bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9353**. Incorrect word length (%d), bit_count = %d, in BIT file %s.

Expected to read 4 bytes, but 'fread' returned 3 or fewer. Typically this indicates that the end-of-file has been reached while expecting to read more data. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9354**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 1 byte (compressed count, from a *.bit file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9355**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send 1 byte (compressed value, from a *.bit file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9356**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send last byte (compressed count, from a *.bit file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9357**. Cannot send HSB message data, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to send last byte (compressed value, from a *.bit file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9358**. Failed to read BIT file %s (after %d bits read, expected %d).

The 'hrn_fpga' function was unable to read a byte (compressed count) of the bit file. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9359**. Incorrect word length (%d), bit_count = %d, in BIT file %s.

Expected to read 4 bytes, but 'fread' returned 3 or fewer. Typically this indicates that the end-of-file has been reached while expecting to read more data. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9360**. Incorrect number of bits (%d, %d) in BIT file %s.

The 'hrn_fpga' function has counted the number of bits ('0' or '1' digits) in the *.bit file. It found that this number is not equal to the number of bits as declared by the file in the header. The file may have been corrupted or truncated, or is not a *.bit file.

**Error 9361**. Cannot end HSB message, slot 0x%x (via '%s %d') (%s).

The 'hrn_fpga' function is trying to end the HSB message, used to program the user FPGA, but the API function used, `HeHSBEndOfSendMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more

information on the API error reported.

**Error 9362**. Cannot start receive HSB message (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.bit bitstream bits, but when trying to start reading a response from the user FPGA, the API function used, `HeHSBStartReceiveMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9363**. Cannot receive HSB message data (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.bit bitstream bits, but when trying to read a response from the user FPGA, the API function used, `HeHSBReceiveMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9364**. Cannot end HSB message (via '%s %d') (%s).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.bit bitstream bits, but when trying to end reading a response from the user FPGA, the API function used, `HeHSBEndOfReceiveMessageEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

**Error 9365**. Configuration successfull, but found incorrect number of bits (%d, %d) in BIT file %s.

The 'hrn_fpga' function has counted the number of bits ('0' or '1' digits) in the *.bit file. But this number is not equal to the number of bits as declared by the *.bit file in the header. But the user FPGA responds that it has been successfully configured.

**Error 9366**. Configuration was unsuccessful (%d).

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.bit bitstream bits. And it has successfully received a reply from the user FPGA. However, the user FPGA replies that the configuration was not successful. The cause is sometimes (but certainly not always) an incorrect design. At the end of the error message the user FPGA's reply is shown (integer %d).

**Error 9367**. Configuration unsuccessfull. Incorrect number of bits (%d, %d) in BIT file %s.

The 'hrn_fpga' function has successfully sent a long HSB message to program the user FPGA with the *.bit bitstream bits. And it has successfully received a reply from the user FPGA. The user FPGA replies that the configuration was not successful. The 'hrn_fpga' function has counted the number of bits ('0' or '1' digits) in the *.bit file. It found that this number is not equal to the number of bits as declared by the *.bit file in the header.

**Error 9390**. Cannot send HSB message data, board \"%s %d\" (%s).

The 'hrn_fpga' function is trying to send 4 bytes (from a *.bit file) to the target module, but the API function used, `HeHSBSendMessageDataEx`, reports an error (shown at the end of the error message). Please refer to the API manual for more information on the API error reported.

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.