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.demon.co.uk
URL: http://www.hunteng.co.uk

The HERON Serial Bus Example

Rev 1.0 R.Williams 26-03-01

The HERON Serial Bus example demonstrates how to send and receive messages using the HERON Serial Bus (HSB). The example does this with two separate programs. The first example program is written to use the Host API, and accordingly, is run on the host processor. The second example program is written to use the HERON-API, and needs to be run on a HERON processor module.

The example performs a simple repetitive loop. For each pass through the loop, the host sends a message to the HERON processor module, and then the HERON processor module sends a message back to the host program.

History

Example revision 1.0 first written for HERON-API V3.1 and CCS 1.2

Using the Host Program

Devices

The Host API works with a concept called 'devices'. A carrier board has 1 or more devices. For example, the HEPC8 has a FIFO connecting the PCI interface to the first module on the board. This FIFO is one device ('FIFOA'). The HEPC8 also has a serial bus interface. This is another device ('HSB'). Finally, there is a JTAG interface, used (for example) by Code Composer Studio, called 'JTAG'.

Different carrier boards may have different devices. For example, some boards may have more than 1 FIFO, and may support a device 'FIFOB'. As another example, some carrier boards may have no serial bus interface. Typically there is always at least a 'FIFOA' device and a 'JTAG' device, but this is not a rule and you must not assume that a certain device exists on all carrier boards.

For this example, the host program is only concerned with the use of the HSB device.

Open / Close

Before you can access a device, you must claim the device. Different devices must be claimed separately. The 'HeOpen' and 'HeOpen1' calls are generic device open functions. The difference is that 'HeOpen1' expects a character string to identify the device you want to open, and 'HeOpen1' expects an integer identifier.

A third open function is 'HeOpenS'. This function is used to 'hide' operating system quirks. For example, in an operating system like VxWorks, the open function needs to know for ISA boards what interrupt and address to use. This information can be transferred using 'HeOpenS', which takes an array of 32-bit specifiers as extra input parameter. Therefore, you would not usually need this function.

The HERON Serial Bus (HSB)

The serial bus is a relatively slow bus in comparison to the fast FIFO interface. The serial bus is typically used to retrieve module information or to configure devices (examples: HERON-FPGA module configuration and HEPC9 FIFO connections). In this example HSB is used to send and receive messages between the host program and the DSP.

The host API provides two easy to use functions, `HeHSBSendMessage` and `HeHSBReceiveMessage`. Once the HSB device has been correctly opened for use, the example program uses these two functions to send and receive messages with the DSP program.

Compiling and Linking the Host Program

The source code for the host part of the HSB example is contained in the C file '`host_hsb.c`'. This source code is set up to use the Host API.

Include File

Source files that use Host API functions must include the header file '`heapi.h`'. There are no other HUNT ENGINEERING header files that are required.

When setting up the project to build the host example, please ensure the additional include directories include the 'inc' directory that is found below the HUNT ENGINEERING installation directory.

To include the header file, the following line must appear in the source code.

```
#include <heapi.h>
```

Library

Projects that use the Host API should be linked with the Host API library. This has a different name, depending on the operating system and/or compiler used.

For example, when using a win32 Microsoft Visual C/C++ compiler creating a Windows executable, link with 'hendrv.lib'. When using the win32 Borland compiler, link with 'hebdrv.lib'. There are no other libraries that need to be linked.

Environment Variable

The Host API installation program will generate an environment variable '`HEAPI_DIR`' that points to the directory in which you installed the API.

Using the DSP Program

The DSP example program can be run on any HERON processor module placed in any slot of the carrier board that you are using. The example program for the DSP uses the HERON-API to communicate over HSB.

The HERON-API is the hardware independence layer that we provide to access HERON FIFOs and other features of the HERON modules, such as the HERON Serial Bus. By using the HERON-API, the user is able to access devices such as HSB without needing to understand the underlying hardware. The HERON-API also makes the code portable from one processor module type to the next.

The example that we supply for the DSP is a C file called '`heron_hsb.c`'. This example program uses the HERON-API to manage sending and receiving messages over HSB. The program also uses DSP/BIOS.

DSP/BIOS

DSP/BIOS is the multi-threading environment provided as part of the Code Composer Studio integrated development environment. It also provides services for configuring processor features such as hardware interrupts and timers.

As it is included in Code Composer Studio, along with the Compile tools for the 'C6000, all users of HERON hardware will be able to use it.

The DSP example is configured and built using Code Composer Studio and DSP/BIOS.

Setting up the DSP Example

The DSP example is a HERON-API project that can be set up using the 'create project' Code Composer Studio plug-in. To do this, in Code Composer Studio, choose `Tools -> HUNT ENGINEERING -> Create new HERON-API Project`. This will guide you through setting up the project and as long as you choose the name '`heron_hsb`' for the project it will incorporate the '`heron_hsb.c`' source file.

HERON-API and HSB

When using the HERON-API to send and receive messages using HSB, the HSB device for that processor module must first be claimed with a call to the function '`HeronHsbOpen`'. When the HSB device has been successfully opened, the program is then free to send and receive messages.

The HERON-API provides two easy to use functions, `HeronHsbSendMessage` and `HeronHsbReceiveMessage` in order to do this.

The HSB functions are synchronous in operation, in that the function will not return until the message has been sent or received.

HSB Messages

To use the HERON Serial Bus, you need to adhere to a protocol. That is, both the sending end and the receiving end must agree to the format of each message transmitted.

There is an implicit message format that is used by the HSB functions provided by HUNT ENGINEERING. As such, any use of the HSB functions must build upon this basic format.

All HSB messages must start with an address byte. This is used to identify the intended destination for the message. This address is formed from information such as the board ID and the slot ID of the intended message recipient.

For the HERON-API, the HSB functions automatically create the address byte using the '`board`' and '`slot`' function arguments.

For the Host API, the HSB functions that are provided automatically create the address byte using the internally stored board ID and the '`slot`' function argument. The board ID that is stored is taken from the board number that was used when the host HSB device was opened.

Following the address byte two data bytes are transmitted. The first of these bytes is used to define the '`message type`', and the second byte defines the '`address`' of the device initiating the message.

The message type byte can be used to define what to do with the optional data bytes that follow, and whether a reply is required. The second 'address' byte can be used to form the new address if message is to be sent in reply to the original message.

With these bytes sent, any number of optional data bytes may follow.

The Message Type

For many applications the message type is an important value that defines the exact operation that is to take place when using HSB. A simple example is the message type of 1. This value is used by HUNT ENGINEERING to indicate a Module Type Query.

Messages transmitted with this message type will typically be used to find out information about the type of module that is located in a particular slot of a carrier board. The Module Type Query message is a simple message that has no optional data bytes, just the address, the message type (set to 1), and the address to reply to.

Other examples are the transmission of a serial bitstream to a FPGA located on a HERON-FPGA module. This message (message type 3) will always containing as many optional data bytes as there are bytes in the bitstream for the FPGA.

In the case of this example the message type value is not important and has been set to arbitrary values, `0xaa` for the message from the host program to the DSP, and `0xbb` for the message from the DSP to the host.

If the sender and receiver processes are both *your* code, no pre-defined message type is needed, essentially you may define your own. It is recommended however that you avoid the message types already used by HUNT ENGINEERING software to prevent confusion.

HUNT ENGINEERING defines message numbers starting from one upwards, so using message types greater than 128 will avoid confusion. The message protocol used by HUNT ENGINEERING is defined in a separate document.