



**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>



## The “sl\_api” Server/Loader example.

Rev 3.3 JT 15/10/03 (changed for SL 4.08)

The “sl\_api” example shows how the Server/Loader could be used together with a host API program.

The “exe” and “batch” programs first boot a system using the Server/Loader. The Server/Loader then stops, and a host API program then takes over, and communicates with DSP processors booted by the Server/Loader. With the “exe” example, the Server/Loader Library (“win32sl.dll”) is used to boot the system, and then host API code is run. With the “batch” program, the Server/Loader executable is used to boot the system, which then quits, and then a host API program is run.

The “threads” and “parallel” programs first boot a system using the Server/Loader, but also continue to serve the booted system. With the “threads” example, the Server/Loader Library (“win32sl.dll”) is used to boot, reset and serve the system, and in a separate thread host API code is run. With the “parallel” example, the Server/Loader executable is used to boot the system, which keeps running, and a host API program is run in a separate DOS-box.

There are 2 general approaches:

1. Use the Server/Loader to load (and perhaps serve) the network. Then start as a second program the user program, which opens the API and communicates with the program loaded onto the DSPs by the Server/Loader. This user program must NOT reset the board! This method is used in the sample in sub-directory “batch” and “parallel”.
2. Use the Server/Loader library to load (and perhaps serve) the network of DSPs, and use the API to communicate with the loaded program. In this case, both Server/Loader and API are used within one executable file. This method is used in the sample in sub-directory “exe” and “threads”.

### History

Example revision 2.0 made for HERON-API V2.3

Example revision 3.0 made for CCS V1.2

Example revision 3.1 made for Server/Loader V3.2

Example revision 3.2 made for Server/Loader V3.3

Example revision 3.3 made for Server/Loader V4.07

## **Example software**

The example that we supply is a C file called test.c in each 'sl\_api' example. It needs to be built using Code Composer Studio and uses the HERON-API software that has been installed on your PC when you did the "install drivers and tools" from your CD.

## **Hardware setup**

The example shows the communication between a HERON module and the host. It is also using the FIFO connection to the Host machine to boot over. This means that the first HERON module must be connected to the host, via a HERON FIFO.

The demo as shipped is for an HEPC9 with a module in HERON slot 1. The Host connection is then FIFO #0 and is configured using the appropriate HEART statements in the network file. The "exe" and "batch" examples have no example network file, but one will be generated when you create a new project using HUNT ENGINEERING's "Create New HERON\_API Project" plug-in.

Note that the "threads" and "parallel" examples can only run with HEART boards. This is because 2 node – host connections are needed. For example, the HEPC8 only has 1 node – host connection, and this connection exists for HEPC8 slot 1 only.

If you are running the demo on a different hardware configuration, you will need to change the #defines in the DSP source code to reflect the connections that you have, and also the network file that describes the connections to the Server/Loader.

## **DSP/BIOS**

DSP/BIOS is the multi-threading environment provided as part of the Code Composer development Environment. It also provided 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.

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

## **HERON\_API**

HERON\_API is the hardware independence layer that we provide to access HERON FIFOs and other features of the HERON modules. It allows the DMA engines of the processor to be used when transferring to and from the FIFOS without knowledge of the FIFO hardware, or the DMA engines.

## **Starting**

We assume that a user of this example has previously installed Code Composer and followed the confidence checks. They should also be familiar with using Code Composer.