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
www.hunteng.co.uk
www.hunt-dsp.com

Different ways of using the EM2 to connect systems.

v 1.0 P.Warnes 28-11-02

The PC9-EM2 is an inter-board connection module that can make a Virtual FIFO connection between boards, either in the same Host machine or separate machines/racks.

This document discusses the different ways to use that connection and demonstrates some of those using examples for which source code is supplied.

History

Rev 1.0	first written

Concepts

HEART provides a way to connect using software nodes on a board using “virtual FIFOs”. That is once you have made the connection it behaves exactly as if it were a single dedicated FIFO connected between the nodes.

Those connections are made using the Heartconf or Server/Loader tools.

The PC9-EM2 allows those virtual FIFOs to be extended between different boards, using a Gbit serial connection to provide a bi-directional connection that can handle up to 125Mbytes/sec in each direction.

These connections are also made using the Heartconf or Server/Loader tools.

The PC9-EM2 connection can also propagate the system Reset and HSB, making it possible to treat the whole system as if it were a single board.

In some circumstances however it can be desirable to boot the separate boards as separate systems, but then some care must be taken over the synchronisation of data. This document concentrates on that issue.

Getting Started

There are many examples and documents that discuss the use of HEART. There are also documents that describe the Server/Loader and Heartconf tools, along with the document that describes the syntax of the network file.

Once the connection is made, the nodes at each end use that connection exactly as if it were a HEART connection on the same board, i.e. FPGA modules can use the Hardware Interface Layer, C6000 modules use HERON-API and the Host machine used Host API.

The purpose of this example is to demonstrate that data synchronisation is an issue that must be carefully considered. This issue exists regardless of the type of nodes on either board, but for clarity uses a DSP node on each board. The methods used are just one illustration of how the problems can be solved, and your application may require a more complex solution, or perhaps needs no solution at all.

If your combination of modules is different then you need to find the correct solution for that.

The first part of our tutorial assumes two boards in the same PC.

The example is very simple – sending data to and from each board in the system, with the receiving node checking the data to see if it is the expected data.

The second example can be used on that same system with two boards in the same PC but can also be used with 2 boards in different host PCs.

It is the same example, but with some additional data synchronisation.

Running the first example

Follow this sequence:

1) Fit a C6000 module to slot 2 of the an HEPC9 that has it's red switch set to 0. Fit a C6000 module to slot 3 of the an HEPC9 that has it's red switch set to 1. Fit a PC9-EM2 module to each board.

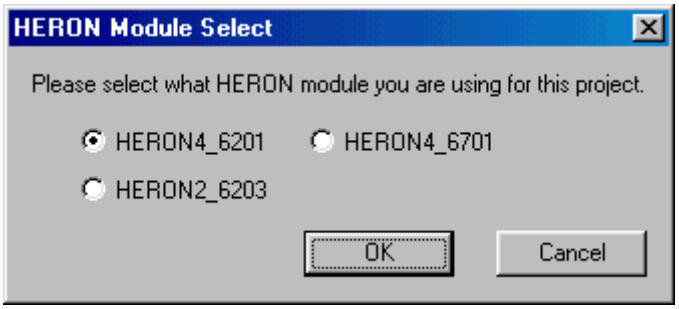
   Fit the boards to a PC and fit an interconnect cable between the PC9-EM2s. Fit the cable to Channel 0 of board 0 and channel 1 of board 1.

2) Power up the PC, and copy the files from the "2bds1net" directory of this example. There should be 2 C files "board0.c", "board1.c" and a network file "network".

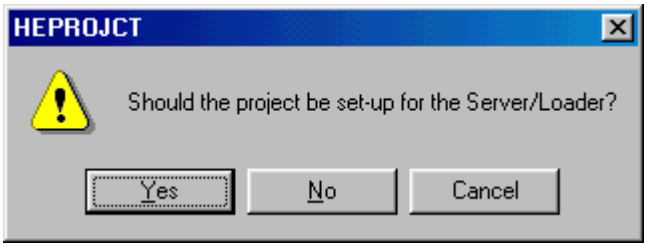
3) Make sure that Code Composer studio is configured for the number of C6000 modules fitted to the board you will run CCS on. You can run the Autoconfigure plug in to do this .

   Open Code Composer, and use the "Create New HERON-API project" plug in to create a Server/Loader project using board0.c as a template, for the module type you have, i.e

   When you see



   Select the type of C6000 module you have plugged onto board 0, and choose yes when you see



   Open the .cdb file in Code Composer and add another task. Set the function property of this task to be _readtask (the underbar is important!)

   Add two Semaphores, and rename then as "read0" and "write0"

   Now you can build the board0 project

4) Follow the same instructions as 3) but using a project made with board1 as a template and for the module type you have fitted to board0.

5) Look at the source file board1.c and notice that there are two functions "mantask" and "readtask" which will be run as separate tasks on the same processor. Because they are running as tasks the Semaphore model is used for HERON-API. Also there are some TSK_Sleep(1) calls which will force some scheduling to take place and will ensure that both tasks get run. Notice that the FIFO number that the DSP will use for the communication is set at the top of the file with a #define . You can change this, but must

remember to make the corresponding change to the network file if you do. The maintask sends blocks of data containing a count from 0 to the BLOCKSIZE which is in the beginning set to 500. You can change this but must also change the corresponding define in the other project, as the maintask of this processor sends to the readtask of the other processor and vice versa.

6) Quit Code Composer, and use Start→Programs→HUNT ENGINEERING → API JTAG reset to free the JTAG chain.

7) Edit the file “network” using notepad or a similar tool.

```
#-------------------------------------------------------------------------
# Board description
# For raw board handling, use:
# BD Board_type Board_address
# For HUNT ENGINEERING's Device Driver API use:
# BD API        Board_type      Board_Id        Device_Id
#-------------------------------------------------------------------------
# Using API
BD API HEP9A 0 0
BD API HEP9A 1 0
#-------------------------------------------------------------------------
# Nodes description
# ND  BD_nb  ND_NAME  ND_Type        HERON-ID  filename(s)
#-------------------------------------------------------------------------
  c6  0      HERON1    ROOT           00000002  board0.out
  c6  1      HERON2   Root           0x13  board1.out
# Host interface:
  pcif 0     host1   normal              0x05
  pcif 1     host2   normal              0x15
# HEPC9 inter-board modules, currently EM1-C, EM1 or EM2:
em2  0     em21   normal              0x06
em2  1     em22   normal              0x16

#-------------------------------------------------
# Number of the link connected to the host system
# HOSTLINK  PORT
#-------------------------------------------------
  TOHOST    0
  FROMHOST  0

#-------------------------------------------------------------------------
# The actual HEART programming statements. Used by both Server/Loader and
# HeartConf.
#       from:slot  fifo  to:slot   fifo   timeslots
#-------------------------------------------------------------------------

# Create a connection between host (fifo 0) and the 'C6x (fifo 0). It uses
# 1 timeslot. The precise timeslot is chosen by Server/Loader or HeartConf
 heart      host1  0      heron1   0       1
 heart      host2  0      heron2   0       1

# And create a connection back, from the 'C6x (fifo 0) to the host (fifo 0)
  heart      heron1  0      host1   0       1
  heart      heron2  0      host2   0       1

# now the connections to and from the inter board module
  heart      heron1  1      em21   0       1
  heart      em21  0      heron1   1       1
  heart      heron2  1      em22   1       1
  heart      em22  1      heron2   1       1
```

Notice that both boards are defined at the top, one as board 0 and the other as board1. These must be the same as the red switch settings on the boards.

Now notice that the C6000, host and EM2 nodes for *both* boards are defined in this file, with the board number it is on. Remember! The board number here is the position in the list of board in this file, not the board switch setting. This is so you can define a change in the board switch in just the board definition. In our case actually these are the same as we chose boards 0 and 1 and defined them in that order.

Finally notice that there are connections to and from the DSPs from the host node of their own boards. This is used for the STDIO connections of Server/Loader. Then notice also the connections to and from the PC9-EM2 modules. The FIFO numbers defined for the HERON module must match the #define FIFO number in the C source files. The FIFO numbers defined for the EM2 must match where you fitted the cable.

8) The plug in will have made a batch file w32.exe for you that contains the following:-

%HESL_DIR%\bin\win32\win32sl -I%HESL_DIR%\lib -rls%1 network

which calls the server/loader with the network file “network”. If you renamed that file you would need to change this batch file too.

Run that batch file and you should see that the processors have booted and are running as follows :-

```
Bd0: HERON module write task has started running
Bd0: HERON module read task has started running
Bd1: HERON module write task has started running
Bd1: HERON module read task has started running
board0 completed loop 100
board1 completed loop 100
board0 completed loop 200
board1 completed loop 200
board0 completed loop 300
board1 completed loop 300
board0 completed loop 400
board1 completed loop 400
board0 completed loop 500
board1 completed loop 500
board0 completed loop 600
```

Lessons of the first example

Because the server/loader has been used to boot both boards, the starting of execution is dealt with properly. Every time you run the program it will start with both processors synchronised.

Because the boards are placed in the same PC (because they are both being booted from the same PCI bus), there should be no problems with data unreliability between these two boards so the example is a good and safe implementation.

Running the second example

With the same hardware set up as the first example, follow this sequence:

1) Copy the files from the “2bds2nets” directory of this example. There should be 2 C files “board0.c”, “board1.c” and 2 network files “network0” and “network1” and also 2 batch files “w320.bat” and “w321.bat”.

   With Code Composer studio is configured as above, follow the same instructions to make and build the two projects using these new board1.c and board1.c files.

2) Follow the same instructions as 3) above but using a project made with board1 as a template and for the module type you have fitted to board0.

3) Look at the source file board0.c or board1.c and notice in addition to the loops of the first example, that there is a function called “sync”. This is called when the read task is first started, and also if a data error is detected. Simply it reads from the connection until the data value that we know only occurs at the end of a block is found. Then we know the next data item read will be the beginning of a new block.

4) Quit Code Composer, and use Start→Programs→HUNT ENGINEERING → API JTAG reset to free the JTAG chain.

5) Edit the file “network0” using notepad or a similar tool.

```
#-------------------------------------------------------------------------
# Board description
# For raw board handling, use:
# BD Board_type Board_address
# For HUNT ENGINEERING's Device Driver API use:
# BD API        Board_type      Board_Id        Device_Id
#-------------------------------------------------------------------------
# Using API
BD API HEP9A 0 0
#-------------------------------------------------------------------------
# Nodes description
# ND  BD_nb  ND_NAME  ND_Type        HERON-ID  filename(s)
#-------------------------------------------------------------------------
  c6  0      HERON1    ROOT           00000002  board0.out
# Host interface:
  pcif 0     host1   normal              0x05
# HEPC9 inter-board modules, currently EM1-C, EM1 or EM2:
em2  0     em21   normal              0x06

#-----------------------------------------------
# Number of the link connected to the host system
# HOSTLINK  PORT
#-----------------------------------------------
  TOHOST    0
  FROMHOST  0

#-------------------------------------------------------------------------
# The actual HEART programming statements. Used by both Server/Loader and
# HeartConf.
#       from:slot  fifo  to:slot   fifo   timeslots
#-------------------------------------------------------------------------

# Create a connection between host (fifo 0) and the 'C6x (fifo 0). It uses
# 1 timeslot. The precise timeslot is chosen by Server/Loader or HeartConf
  heart      host1  0      heron1   0       1

# And create a connection back, from the 'C6x (fifo 0) to the host (fifo 0)
```

```
  heart      heron1  0      host1   0       1

# now create the connections to the em2
  heart      heron1  1      em21    0       1
  heart      em21  0      heron1    1       1
```

Notice that only board 0 is defined at the top. This must be the same as the red switch settings on the board.

Now notice that the C6000, host and EM2 nodes for just board 0 are defined in this file, with the board number it is on. Remember! The board number here is the position in the list of board in this file, not the board switch setting. This is so you can define a change in the board switch in just the board definition. In our case actually these are the same as we chose boards 0 as the only board. .

The connections are also only the connections for this board. The FIFO numbers defined for the HERON module must match the #define FIFO number in the C source files. The FIFO numbers defined for the EM2 must match where you fitted the cable.

6) Now edit the file “network1” using notepad or a similar tool.

```
#-------------------------------------------------------------------------
# Board description
# For raw board handling, use:
# BD Board_type Board_address
# For HUNT ENGINEERING's Device Driver API use:
# BD API        Board_type      Board_Id        Device_Id
#-------------------------------------------------------------------------
# Using API

BD API HEP9A 1 0
#-------------------------------------------------------------------------
# Nodes description
# ND  BD_nb  ND_NAME  ND_Type        HERON-ID  filename(s)
#-------------------------------------------------------------------------
   c6  0      HERON2   Root           0x13  board1.out
# Host interface:
  pcif 0     host2   normal              0x15
# HEPC9 inter-board modules, currently EM1-C, EM1 or EM2:
em2  0     em22   normal              0x16

#-----------------------------------------------
# Number of the link connected to the host system
# HOSTLINK  PORT
#-----------------------------------------------
  TOHOST    0
  FROMHOST  0

#-------------------------------------------------------------------------
# The actual HEART programming statements. Used by both Server/Loader and
# HeartConf.
#       from:slot  fifo  to:slot   fifo   timeslots
#-------------------------------------------------------------------------

# Create a connection between host (fifo 0) and the 'C6x (fifo 0). It uses
# 1 timeslot. The precise timeslot is chosen by Server/Loader or HeartConf

 heart      host2  0      heron2   0       1

# And create a connection back, from the 'C6x (fifo 0) to the host (fifo 0)

  heart      heron2  0      host2   0       1
```

```
# now create the connections to the em2
  heart      heron2  1      em22   0       1
  heart      em22  0      heron2   1       1
```

Notice that only board 1 is defined at the top. This must be the same as the red switch settings on the board.

Now notice that the C6000, host and EM2 nodes for just this board are defined in this file, but as board 0. Remember! The board number here is the position in the list of board in this file, not the board switch setting. This is so you can define a change in the board switch in just the board definition. In our case we define only board1, so it is referenced as the "0"th board.

The connections are also only the connections for this board. The FIFO numbers defined for the HERON module must match the #define FIFO number in the C source files. The FIFO numbers defined for the EM2 must match where you fitted the cable.

7) The plug in will have made a batch file w32.exe for you that contains the following:-

%HESL_DIR%\bin\win32\win32sl -I%HESL_DIR%\lib -rls%1 network

which calls the server/loader with the network file "network".

In our case we need 2 batch files, one for each board. They are the batch files w320.bat and w3231.bat that we copied from the CD.

Run one of the batch files to start a board, and you should see :0

```
Bd0: HERON module write task has started running
Bd0: HERON module read task has started running
```

Depending on the board number you booted.
Now run the other batch file. After some synchronisation you should see that each board is looping something like

```
board0 completed loop 100
board0 completed loop 200
board0 completed loop 300
board0 completed loop 400
board0 completed loop 500
board0 completed loop 600
```

You can stop either end and re-boot it and you should see each end re-synchronise and start to loop again.

Lessons of the second example

Because the server/loader has been used to boot each board separately, there can be some data lost as the system is started. This can be recovered from if the application software is written to handle this case.

The same solution works well if one end is re-booted for some reason.

When the cable is routed between PCs this external cable can be "abused" by unplugging the cable or by routing it close to a source of extreme electrical (or magnetic noise). This can cause data loss, gain or corruption that can be handled by the application software in a similar way.