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 “SEM model” HERON-API Example

Rev 1.0 R.Williams 25-9-00

The “SEM model” example is not really a program but more of a code segment to demonstrate how the HeronSemOpenFifo function can be used.

The DSP code uses HERON-API to manage the transfer of data over the HERON FIFOS.

The use of HERON-API means that the example is easily changed to use any HERON C6000 module. HERON-API uses DSP/BIOS internally so must be built using Code Composer Studio.

This document describes how to use the semaphore model to overlap tasks that receive and process data.

History

Example revision 1.0 first written for HERON-API V3.0 and CCS 1.2

Example Software

The example that we supply is a C file called example.c. It needs to be changed to reflect your actual needs, and then built using Code Composer Studio. It uses the HERON-API software that has been installed on your PC when you did the "install drivers and tools" from your CD.

DSP/BIOS

DSP/BIOS is the multi-threading environment provided as part of the Code Composer Studio 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.

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

HERON-API

The 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.

SEM Model

The I/O model used by HERON-API is an asynchronous one. That is the HeronRead and HeronWrite functions actually ***start*** the I/O, but it is left to the user to determine when the I/O is complete.

Sometimes it is convenient for the user program to be actively informed when the I/O is complete. The SEM model is one way of achieving this.

To use the SEM model, the user must supply a DSP/BIOS Semaphore Object (SEM_Obj). The SEM object is attached to a FIFO handle when the FIFO is opened with HeronSemOpenFifo. The SEM object is then posted by the HERON-API each time a read or write completes for that FIFO.

The user must use a call to the function SEM_pend to block a task on the completion of the I/O. When the semaphore becomes posted, the SEM_pend will become unblocked, allowing that task to continue.

Setting up the example

The example is a HERON-API project that can be set up using the create project plug in. 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 "example" for the project it will incorporate the example.c file.

When the project has been created, you need to open the .cdb file and insert two TSK objects and two SEM objects. The first task (TSK0) must be set to run the function _mainTask. The priority of _mainTask must be set to 2. The second task (TSK1) must be set to run the function _processTask. The priority of _processTask must be set to 1.

The first SEM object that you create must be named 'read_complete'. Its initial count must be set to 0. The second SEM object that you create must be named 'process'. Its initial count must also be set to 0.

The Example

The example program, “example.c” contains two tasks. The first task (_mainTask) first opens a FIFO for reading, using the semaphore model function, HeronSemOpenFifo. When the FIFO is opened with the HeronSemOpenFifo function, a semaphore object is attached to the FIFO handle. In this example, the SEM object ‘read_complete’ is attached to the FIFO handle.

The first task then performs a loop that repeatedly calls the function HeronRead. For each call to HeronRead, a read transfer is started from the FIFO, and when the transfer completes, a semaphore is posted using the ‘read_complete’ object.

In order for the first task to block until the read completes, there must be a call to the DSP/BIOS function SEM_pend. While the task is blocked, all other tasks are able to run. This includes the lower priority task ‘_processTask’.

The second task (_processTask) does simple processing on buffers that have been received by the first task. By blocking waiting for the completion of a FIFO read, the first task enables the second task to run, allowing it to process previously received data.

As each call to SEM_pend completes for the first task, a second semaphore is posted indicating to the second task that there is another buffer available for processing. With this done, the next HeronRead is started.