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

Using the C6201/C6701 DMA to copy from memory to memory.

Rev 1.0 P.Warnes 20-12-02

The C6201/C6701 processors have four DMA engines. They can be programmed by the user, but they are also used by the HUNT ENGINEERING HERON-API. HERON-API provides a resource management task that allows user to program DMAs themselves.

*****

This example shows how to claim and free DMAs and actually demonstrates a simple copy operation using the DMA.

History

Example revision 1.0 first written

Example software

The example that we supply consists of a C file for the DSP called dmacopy.c . It needs to be built using Code Composer Studio and it 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 use of a C6x01 DMA to copy memory buffers. Although the HERON-API provides hardware independence, the DMA used in the example is a C6x01 DMA, which is different from that of other members of the C6000 processor family. Different C6000 processors might have different addresses, or functions for DMA control registers, so you might have to alter this example to work with the processor type you actually have.

To reference the details of the timers for the processor your module has, please refer to the peripherals reference guide provided by TI but also available on the HUNT ENGINEERING CD under user manuals.

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.

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.

Configuring the example

HUNT ENGINEERING provide several Code Composer Plug-in tools that allow you to make your development faster. The first one is one that sets up Code Composer ready for your hardware, so you don’t need to configure device drivers etc and can be found from the Start→Programs→HUNT ENGINEERING→AutoConfigure CCS.

We assume that this is already set up, but this plug in also copies cdb files etc into the correct locations.

When you start with this example, simply copy the source files from the CD into a new directory. Then start Code Composer and choose Tools→HUNT ENGINEERING→Create new Heron-API project. This will guide you through setting the project up, and as long as you choose the name “dmacopy” for the project -will incorporate the dmacopy.c file.

You can now build the demo by choosing Project → re-build all. There should be no errors, or warnings.

Manually Setting up the Project

For your information (or if there is some problem) here is how to set up the project yourself:-

Make sure that you have copied all of the .cdb files from the directory %HEAPI_DIR%\heron_api\cmd into the directory C6000\bios\include under the directory where your Code Composer Studio installation is (usually c:\ti).

In Code Composer, select 'Project →new' and choose the path and name for your project.

Select 'File → New → DSP/BIOS Config' and choose the correct .cdb file for your hardware. This will have a name that uses your HERON module number and possibly an option that is available for that module.

In the DSP/BIOS config tool, right click on Global properties, and check that the CLKOUT property is set to the frequency of your processor module. This is used by DSP/BIOS to calculate the correct settings for the timer period.

This .cdb file has some items set up which are for HERON-API. DO NOT CHANGE THESE!

For this example you need to set up a task that is called TSK0.

Use 'File → Save' to save the cdb file to the project directory as the name you used to name the project.

Saving the .cdb file wll generate a .cmd file, but that file will not place the sections heronapi_code and heronapi_data. For this reason there is a .cmd file supplied by us, in the directory %HEAPI_DIR%\heron_api\cmd that will be called by your heron module number and have _bios.cmd at the end, i.e. heronx_bios.cmd. You need to copy this to your project directory.

Now add the source file to the project and the .cdb, and also the heronx_bios.cmd. Edit the .cmd file that you have inserted and change the .cmd file that it includes to replace the ***** by the name of your .cdb file. I.e. change *****cfg.cmd to be dmacopycfg.cmd or whatever you have called the project.

Add the HERON_API library "herons.lib" from the directory %HEAPI_DIR%\heron_api\lib to the project.

Go to Project Options and add %HEAPI_DIR%\heron_api\inc to the include path.

The default .cdb file will actually place all code into external memory, and switch on the program cache. This is a good general purpose setting, but might need ot be changed for your actual application.

You can now build the demo by choosing Project → re-build all. There should be no errors, or warnings.

The example

The example statically declares one array, which will thus be defined in internal memory, and uses malloc to declare another buffer in external memory.

The dma_start_copy function sets the dma in action, and we use a separate dma_wait_end function to poll for the DMA complete. We did it this way so that you could do some processing while the DMA is completing. In our case we do not have any to do.

Finally we check that the data has been copied correctly to prove the example.