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

Using the C6201/C6701 timer.

Rev 2.0 P.Warnes 14-7-00 (changed to reflect CCS 1.2 and additional HUNT CCS plug ins)

The C6201/C6701 processors have two 32 bit timers. They can be configured to count processor clocks, or external events. They can drive an external pin of the device and or generate interrupts for the processor.

DSP/BIOS uses one of the timers to provide periodic functions.

*****

Uncommitted Module Interconnect (UMI) pins of the HERON module are intended for application specific use. There are four identical UMI pins on the HERON module, which are simply bussed together between the module slots of a Module Carrier card. This means that there are four signals that can be used between modules, for any application specific purpose. They are not buffered at all so that they have not in-built direction, but they are pulled up to +5v on the module carrier cards using a 10K resistor. This resistor ensures that unconnected lines are kept at a high level. They are also brought to a connector of the Module Carrier card to enable connection to external devices, or other HERON module carrier cards.

HERON Processor modules are designed to allow a timer output (such as timer0 out on the HERON1) of the Processor to drive any one of the UMI pins. This is demonstrated by this example.

This example uses DSP/BIOS to demonstrate the use of a timer to drive one of the UMI pins.

History

Example revision 2.0 made for CCS V1.2

Example software

The example that we supply consists of a C files for the DSP called timer.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 timer to drive a UMI out pin. Although the HERON-API makes the use of the UMI pins hardware independent, the timer used in the example is a C6201/C6701 timer number 0. Different C6000 processors might have different addresses, or functions for timer 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 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→New Heron-API project. This will guide you through setting up the project, and as long as you choose the name “main” for the project -will incorporate the main.c file. Then all you need to do is to open the .cdb file and insert the TSK0 and set it to me _maintask. You then need to insert the other source file into the project using Project→add files to project→timero.c

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. Under its properties set its function to be "_maintask".

This example uses Timer0 to drive the UMI pin, and normally that is used by the DSP/BIOS CLK and PRD functions. Change the properties of the "clock manager" to use Timer1.

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 files (main.c and timero.c) 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 democfg.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.

Select –o3 optimisation from the compiler optimisation menu.

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 simply uses the HERON-API to select which of the UMI pins is to be driven by the timer, and sets the bits in the timer0 control register to drive the TOUT0 pin of the processor, and to have a 50% mark-space ratio.

The timer period is then repeatedly changed, and can be observed on the UMI pin selected using an oscilloscope.

Emulator halt!

When the C6000 timers are configured to count processor clock/4, as in the DSP/BIOS standard configuration, they only count while the processor is not halted by the emulator.

This means that the timer will not free run while the processor is halted by a breakpoint or “halt” option of Code Composer.

It is also necessary that the program has some instructions to execute. A simple loop in the program or the presence of some SWI or HWI function can be enough for this to be the case.

Changing the timer period

As demonstrated by the example the timer period can be updated during program execution by writing the timer0 period register. It is up to the application writer to make sure that the effect this has on other parts of the application such as periodic functions is not a problem to the overall application.

It is also a good idea to re-start the timer after setting its period, as shown in the example by setting the GO bit in the timer0 control register. Failure to do this can cause the timer to stall under certain circumstances.