DSP II: ELEC 4523

HWI Module

Objectives

- Become familiar with HWI module and its use

Reading

- SPRU423 TMS320 DSP/BIOS Users Guide: Hardware Interrupts (section)
- PowerPoint Slides from DSP/BIOS II Workshop: Module 4
- Code Composer Studio Online Help: HWI Module

Lab Module Prerequisites

None

Description

Hardware interrupts (HWI) are used for processing critical functions that usually have a hard deadline. Interrupts usually come from on chip devices or external devices and usually occur asynchronously. When an interrupt occurs it causes the processor to jump to an interrupt subroutine (ISR). HWI ISRs can be written in either assembly or C. When written in C the HWI dispatcher should be used. The dispatcher allows the ISR to use DSP/BIOS API calls that affect other DSP/BIOS objects. The dispatcher calls the `HWI_enter` and `HWI_exit` functions. In the configuration tool the dispatcher can be enabled for an HWI on the HWI's properties. HWIs have the highest priority.

HWIs run to completion. If an HWI is posted more than one time before the HWI completes the HWI will run only one time. For that reason the HWI should do as little processing as possible. Processing of data should be done in lower priority software interrupts or tasks.

HWIs can only be preempted by HWIs that have been enabled when the function `HWI_enter` is called. A bitmask is passed to the function when called. This bitmask can be set up in the configuration tool when the dispatcher is enabled for the HWI.

Laboratory

- In this lab we will set up one of the chips timers to cause an HWI periodically. Since there are two timers and one is used by DSP/BIOS we will use Timer1.
- Create a new project called `hwilab`.
- Create a new DSP/BIOS Configuration file and use the `C6xxx.cdb` template for use with the simulator.
- Save the file as `hwilab.cdb` and include it in your project. Also include the `hwilabcfg.cmd` file.
- If using the simulator then change the RTDX interface to Simulator by right clicking on Input/Output->RTDX and bringing up the properties. Change the RTDX mode to Simulator. If you do not do this then when you load your program you will see the error "RTDX

application does not match emulation protocol." If you are loading onto an EVM or DSK you shouldn't need to change this.

- Set up the timer to interrupt periodically.
  - First we must make a configuration for the timer by opening the configuration file and right clicking on the CSL - Chip Support Library->TIMER->Timer Configuration Manager and selecting Insert timerCfg. This will create a timer configuration called `timerCfg0`.
  - Open the properties of `timerCfg0`.
  - Click on the Clock Control tab and set the Input Clock Source to (CPU clock)/4.
  - Click on the Counter Control tab and set the period to 0x100. This will cause an interrupt every 0x100 system clocks. Click OK.
  - Right click on CSL - Chip Support Library->TIMER->Timer Resource Manager->Timer_Device1 and select Properties.
  - Check Open Timer Device, check Enable Pre-initialization and select `timerCfg0` next to Pre-initialize With. Click OK.
- Now set up the HWI
  - Right click on Scheduling->HWI->HWI_INT15 and select Properties. The interrupt source should already be set to Timer_1.
  - Set the function to `_timerHWI`. This will set up the HWI to call the function `timerHWI` when the interrupt occurs.
  - Click on the tab Dispatcher and check the Use Dispatcher box.
- Now create a LOG object by right clicking on Instrumentation->LOG and selecting Insert LOG. This will insert a LOG object called `LOG0`. Change its name to `trace`.
- Create a main.c file and include the following code. The timer interrupt must be set up in the main function before the interrupt will occur. The functions shown in the main function are part of the chip support library (CSL). The function timerHWI simply prints to the LOG object when it gets executed.

```
#include <std.h>
#include <hwi.h>
#include <log.h>
#include <csl.h>
#include <csl_irq.h>
#include <csl_timer.h>
#include "hwilabcfg.h"

static Uint32 TimerEventId;

main()
{
      /* Get the timer event ID */
      TimerEventId = TIMER_getEventId(hTimer1);

      /* Enable the event */
      IRQ_enable(TimerEventId);

      /* Start the timer */
      TIMER_start(hTimer1);
}
```

```
void timerHWI()
{
      /* print something to show the HWI is executing */
      LOG_printf(&trace,"HWI timer");

}
```

- Build and load your program.
- Open the message log view, DSP/BIOS->Message Log.
- Run the program and record the results.