DSP II: ELEC 4523

CLK Module and System Clock

Objectives

- Become familiar with CLK module and system clock

Reading

- SPRU423 TMS320 DSP/BIOS Users Guide: Timers, Interrupts, and the System Clock (section)
- PowerPoint Slides from DSP/BIOS II Workshop: Module 3
- Code Composer Studio Online Help: CLK Module

Lab Module Prerequisites

None

Description

DSP/BIOS provides two methods of tracking time, the high and low resolution clock managed by the CLK module and the system clock. For this handout the default configuration for the CLK module and system clock will be used. In this case the system clock and the low resolution clock will be the same.

CLK Module

The CLK module is driven by a timer on the ʼC6000, Timer 0 in the default settings. Figure 1 shows a diagram of how the low and high resolution clocks work. The timer 0 is set up with a period that will give the desired rate for the low resolution clock. When the counter is equal to the period an interrupt occurs and the low resolution clock is incremented. To get the high resolution clock the low resolution clock is multiplied by the period and the current counter value is added. In the default settings the period is set to 1 millisecond (1000 microseconds). The timer counter is incremented every 4 instruction cycles and thus the number of instruction cycles is the high resolution clock times 4. The CPU clock rate in the simulator is set to 133 MHz as default. This is the same clock rate as the ʼC6701 EVM. The CPU clock rate sets the instruction cycle. Therefore, the timer increments its counter every 4/133,000,000 seconds.

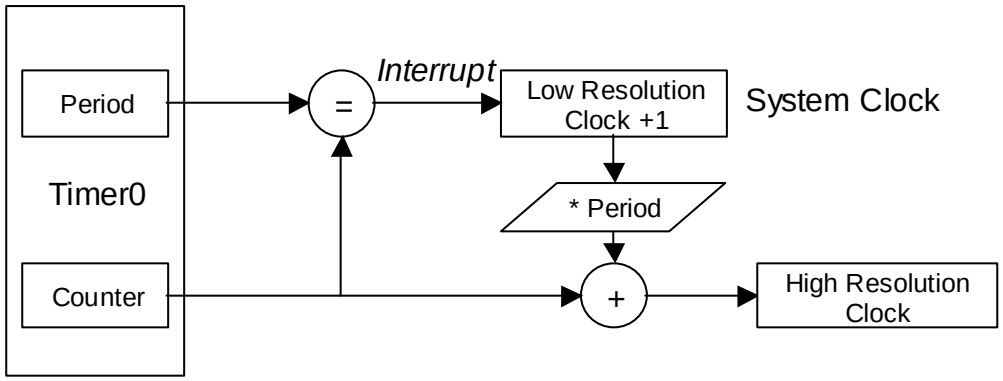

Figure 1: Low and high resolution clock calculations, taken from PowerPoint Slides *DSP/BIOS II Workshop: Module 3*

There are four main functions for accessing the clock information:

- `CLK_countspms()` - Timer counts per millisecond
- `CLK_gethtime()` - Get high resolution time
- `CLK_getltime()` - Get low resolution time
- `CLK_getprd()` - Get period register value

The first three functions return values of type `LgUns` and the last one returns a type Uns. None of them have any parameters.

System Clock

Many functions in DSP/BIOS have a timeout associated with them. The system clock is used to determine the timeout. The system clock is configured by the periodic function monitor module (PRD module) and is driven by whatever object or function calls `PRD_tick`. In the default settings the `PRD_tick` function is called by the CLK object `PRD_clock`. When the timer expires the hardware interrupt `CLK_F_isr` is run which then increments the low resolution clock and calls the functions for all the CLK objects. Since `PRD_clock` is a CLK object it gets called which in turn calls the function `PRD_tick`. This increments the system clock. The `PRD_tick` function then determines if the SWI `PRD_swi` should run. If it should then it posts an SWI for `PRD_swi`. See Figure 2 for the sequence of events. More information on the PRD module can be found in another lab assignment.

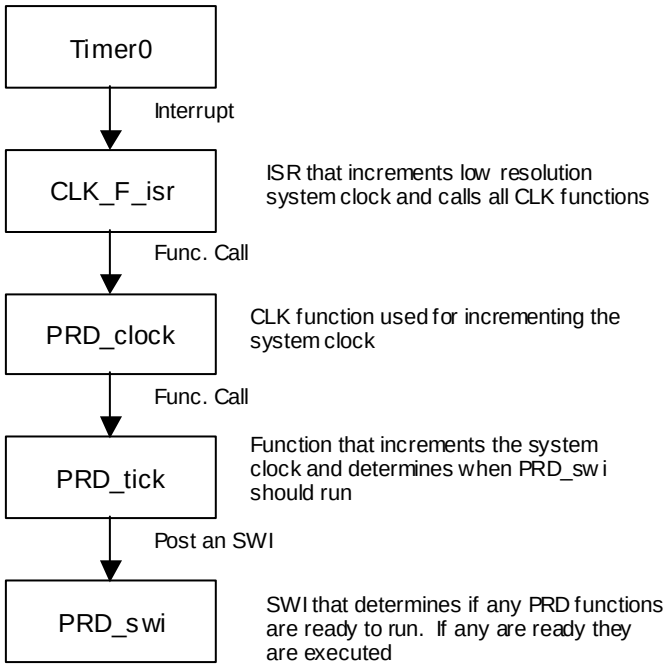

Figure 2: Sequence of events for the system clock and PRD module

Laboratory

- Create a new project called `clklab`.
- Create a new DSP/BIOS Configuration file and use the `C6xxx.cdb` template for use with the simulator.
- Save the file as `clklab.cdb` and add it to your project. Also add the `clklabcfg.cmd` file to your project.
- *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 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.
- Add a task to the configuration. This is part of the TSK module but you don't need to know about the module to be able to do this lab. Add a task by right clicking on Scheduling->TSK and selecting Insert TSK. A new task should be added called TSK0.
- Specify the function the task will run by right clicking on TSK0 and bringing up its properties. On the Function tab next to Task Function type in the function name `_testClock`. This will specify the C function `testClock` as the function it will call.
- Create a main.c file and include a `main` function that does nothing and a function `testClock`. Include this file in your project. The main structure of the file is

```
#include <std.h>
```

```
#include <clk.h>
#include <stdio.h>

main()
{
}

testClock()
{
/* your code here */
}
```

- In the testClock function add code that does the following:
  - Use the C `printf` function to print the number of counts per millisecond to the standard I/O.
  - Write a `for` loop that loops 100 times. Print the number of high resolution ticks it takes to run the loop if inside the loop you multiply the loop counter by a constant and put it in a variable.
  - Write a `for` loop that loops 1000 times. Print the number of high resolution ticks it takes to run the loop if inside the loop you multiply the loop counter by a constant and put it in a variable.
- Make sure that optimization is turned off. If optimization is on then the optimizer may not do any processing in the loops since the results are not used. Select Project->Build Options... and next to Opt Level select None.
- After building and loading the project a Stdout window should open up in the same window as the Build window.
- Run the program and record the results.
- Verify that the counts per ms are what you expected.