DSP II: ELEC 4523

LOG Module

Objectives

- Become familiar with LOG module and its use

Reading

- SPRU423 TMS320 DSP/BIOS Users Guide: Event Log Manager (LOG Module) (section)
- PowerPoint Slides from DSP/BIOS II Workshop: Module 3
- Code Composer Studio Online Help: LOG Module

Lab Module Prerequisites

None

Description

When running a real-time application it is desirable to not to interrupt the processing while attempting to monitor or debug code. Printing to the standard I/O is very time consuming on the processor. The LOG module helps reduce this time. Formatting of the display takes place on the host computer rather than on the DSP. Data is stored in real-time to a buffer on the DSP and then during idle time it is transferred to the host computer. On the host computer the print statements are formatted and displayed.

The main function used in the LOG module is `LOG_printf`. It works very similar to the C `printf` function. In order to print using the LOG module a LOG object must be defined. To create a new LOG object open the configuration file and right click on Instrumentation->LOG and select Insert LOG. This will create a LOG object called `LOG0`. The name of the object can be changed. On the properties window for the object the buffer length is in words and sets the length of the buffer. The buffer can be either a circular buffer, which contains the last messages written to the object, or fixed, which contains the first messages written to the buffer. A circular buffer allows messages to be sent continually. Each message writes 4 words of data. The format of the function call is

```
LOG_printf(LOG_Obj *log,String format,arg0,arg1)
```

where the `String format` is a usual C `printf` format string with the conversion characters given in Table 1. Note that only two arguments are allowed for conversion in the string format.

Table 1: Conversion characters for use in LOG_printf

Conversion Character	Description
%d	Signed integer
%x	Unsigned hexadecimal integer
%o	Unsigned octal integer
%s	Character string. This character can only be used with constant

	string pointers.
%r	Symbol from symbol table
%p	pointer

When using objects defined with the configuration tool the code must include definitions of the objects and the header file `log.h`. The configuration tool generates a header file that contains the definitions. For instance, the file `loglab.cdb` would generate the file `loglabcfg.h`. If an object called `LOG0` was created then the file would contain

```
extern far LOG_Obj LOG0;
```

Include the configuration header file in your code and you won't need to make the definitions yourself.

To view the message log select DSP/BIOS->Message Log and make sure the log object is selected.

Laboratory

- Create a new project called `loglab`.
- Create a new DSP/BIOS Configuration file and use the `C6xxx.cdb` template for use with the simulator.
- Save the file as `loglab.cdb` and include it in your project. Also include the `loglabcfg.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.
- Create a LOG object called `trace`. See the Description section on how to create a LOG object. Use the default length of 32 and the logtype circular.
- 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 `_logTSK`. This will specify the C function `logTSK` as the function it will call.
- Create a main.c file and include a `main` function that does nothing and a function `logTSK`. The main structure of the file is

```
#include <std.h>
#include <log.h>
#include <tsk.h>
#include "loglabcfg.h"

main()
{
}
```

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

- Write a loop in the logTSK function that loops 20 times and writes some text indicating that the loop number is being printed and print the loop number.
- Build and load your program.
- Open the message log view, DSP/BIOS->Message Log.
- Run the program and record the results.