DSP II: ELEC 4523

STS Module

Objectives

- Become familiar with STS module and its use

Reading

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

Lab Module Prerequisites

May want to do the CLK module before the STS module but it is not necessary.

Description

The STS module manages statistics objects. Statistics are stored based on the information given the object. Information stored by the STS object consists of

- **Count.** The number of values on the target in an application-supplied data series
- **Total**. The arithmetic sum of the individual data values on the target in this series
- **Maximum.** The largest value already encountered on the target in this series
- **Average.** Using the count and total, the Statistics View Analysis Tool calculates the average on the host

Information is stored on the target in 32-bit words and on the host in 64-bit words. The accumulated data can also be filtered before it is diplayed on the host. Figure 1 shows the variable accumulation on the target and host. The host requests data from the target and then clears the accumulated data on the target. Data is accumulated on the host in the 64-bit words. The filter (A·total +B)/C or (A·total +B)/(C·count) can be applied to the data before it is displayed. When specifying the STS object the values of A, B and C are specified. The default values are A=1, B=0 and C=1.

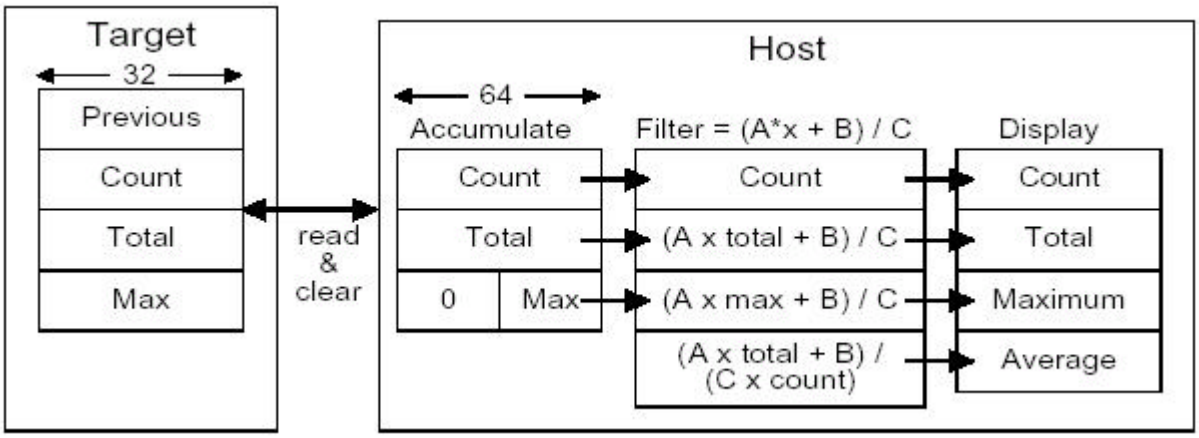

Figure 1: Host/Target variable accumulation, taken from *SPRU423 TMS320 DSP/BIOS Users Guide*

The host reads and clears the target data with interrupts disabled. This allows tasks on the target to updated STS data reliably.

Functions in the STS module are

- `STS_add` - Update statistics using provided value
- `STS_delta` - Update statistics using difference between provided value and setpoint
- `STS_reset` - Reset values stored in STS object
- `STS_set` - Save a setpoint value

To track information about a varying value use the `STS_add` function. The definition is

```
void STS_add(STS_Handle sts, LgInt value);
```

The STS_add function passes a value to the STS object which increments the count and updates the total and maximum values. To track the minimum value of a variable pass the negative value to the STS object.

To time events or monitor incremental differences in a value use the `STS_set` and `STS_delta` functions. Their definitions are

```
void STS_set(STS_Handle sts, LgInt value);
void STS_delta(STS_Handle sts, LgInt value);
```

The `STS_set` function puts the value in the "previous" storage location as seen in Figure 1. Then when `STS_delta` function is called the "previous" value is subtracted from the new value and used to update the accumulator. This can be handy when wanting to track how long a function takes to run or an event to occur. Use the CLK module function `CLK_gethtime` to get the current high resolution system time and pass that value to the functions. This would look like

```
STS_set(&stsObj, CLK_gethtime());
/* function that you want to time goes here */
STS_delta(&stsObj, CLK_gethtime());
```

To create an STS object open the configuration file and right click on Instrumentation->STS and select Insert STS. After the object is inserted you can change its' name and properties.

To view the statistics when running an application open the statistics view by selecting DSP/BIOS->Statistics View. This view will show statistics on the STS objects you have created as well as statistics on HWI, PIP, PRD, TSK and SWI modules. To select statistics gathering on these modules make sure they are selected in the DSP/BIOS->RTA Control Panel. Also, to select the objects you want displayed right click on the Statistics View and select Properties Page. Select in the menu all the objects you want displayed.

Laboratory

Part 1

- Create a new project called `stslab`.
- Create a new DSP/BIOS Configuration file and use the `C6xxx.cdb` template for use with the simulator.
- Save the file as `stslab.cdb` and include it in your project. Also include the `stslabcfg.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 three STS objects called `period`, `timing` and `avg`. See the Description section on how to create an STS object. See below on how to set up the properties of these objects.
- Add a periodic task to the configuration. This is part of the PRD module but you don't need to know about the module to be able to do this lab. Add a periodic task by right clicking on Scheduling->PRD and selecting Insert PRD. A new periodic task should be added called PRD0.
- Specify the function the periodic task will run by right clicking on PRD0 and bringing up its properties. Next to Function type in the function name `_stsPRD`. This will specify the C function `stsPRD` as the function it will call. Also set the period to 2 ticks. This will cause the function `stsPRD` to be executed every 2 milliseconds.
- Create a main.c file and include a `main` function that does nothing and a function `stsPRD`. Include this file in your project. The main structure of the file is

```
#include <std.h>
#include <sts.h>
#include <prd.h>
#include <clk.h>
#include "stslabcfg.h"

main()
{

}

stsPRD()
```

```
{
/* your code here */
}
```

- In the function `stsPRD` do the following:
  - Add a loop that increments its count variable from 0 to 99.
  - Inside this loop use the `avg` STS object to get the average of the count variable.
- Build and load your project.
- Open the statistics view, DSP/BIOS-> Statistics View and make sure the objects you want are selected in the properties page.
- Run the program and record the results.

Part 2

- In the function `stsPRD` add the following:
  - Determine the amount of time it takes the loop that was added in part 1 to run. Use the function call `CLK_gethtime` to get the high resolution clock time, the `STS_set` and `STS_delta` functions and the `timing` STS object.
- Display the `timing` statistics in milliseconds. Since the high resolution timer is in 4 instruction cycles and the clock rate is 133 MHz the time in milliseconds is calculated by (total · 4 · 1000ms/s)/(133,000,000 Hz · count). Use this to set up the filtering values of A, B and C for the `timing` STS object. Open the objects properties in the configuration file and change A, B and C.
- Build and load your project.
- Run the program and record the results.

Part 3

- In this part you will be determining how often the `stsPRD` function is called. This should be 2 ms.
- To time how often the `stsPRD` is called use the `STS_delta` function at the beginning of the `stsPRD` function and `STS_set` right after it. The first time the `STS_delta` function is called the "previous" value is initialized to zero (default) so the delta will just be how long it has taken to call the function since the CLK clock started. For a periodic function this should be the value of the period. The `STS_set` function then sets the previous value so that when the `stsPRD` function is called again the time between the two functions calls is determined.
- In the function `stsPRD` add the following:
  - At the beginning of the function add the STS calls to determine the period of the periodic function. Use the function call `CLK_gethtime` to get the high resolution clock time, `STS_delta` followed by `STS_set` and the `period` STS object.
- Display the `period` statistics in milliseconds. Since the high resolution timer is in 4 instruction cycles and the clock rate is 133 MHz the time in milliseconds is calculated by (total · 4 · 1000ms/s)/(133,000,000 Hz · count). Use this to set up the filtering values of A, B and C for the `period` STS object. Open the objects properties in the configuration file and change A, B and C.
- Build and load your project.

- Run the program and record the results.