DSP II: ELEC 4523

MBX Module

Objectives

- Become familiar with MBX object and their use with tasks to pass data.

Reading

- SPRU423 TMS320 DSP/BIOS Users Guide: Mailboxes (section)
- PowerPoint Slides from DSP/BIOS II Workshop: Module 8
- Code Composer Studio Online Help: MBX Module

Lab Module Prerequisites

Be sure to do the TSK moduls before the MBX module.

Description

Mailboxes in the MBX module are different from the SWI mailboxes. MBX objects are used to synchronize tasks and to pass data from one task to another. Passing of the data is done through copying of the data to a buffer in the MBX object. MBX objects are configured with a certain number of messages of a fixed size. In the configuration tool the length is the number of messages in the MBX and the size is the size of each message in MADUs, 8-bit bytes. Any number of tasks can read or write to a mailbox. Function calls for accessing messages in a mailbox are atomic so mutually exclusive access to the mailbox is maintained.

Syncronization is done through calls to `MBX_pend` and `MBX_post`. `MBX_pend` is used to get a message from a mailbox where the calling task will block if no messages are available. `MBX_post` is used to put a message in a mailbox where the calling task will block if no more room is available to put a message in the mailbox.

Laboratory

Part 1

- In this part you will be creating two TSKs were one will generate some data and send it to the other TSK in a mailbox. The transmitting TSK will generate 5 messages of data to send to the other TSK. The mailbox will be initialized with 3 buffers.
- Create a new project called `mbxlab`.
- Create a new DSP/BIOS Configuration file and use the `C6xxx.cdb` template for use with the simulator.
- Save the file as `mbxlab.cdb` and add it in your project. Also add the `mbxlabcfg.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 by right clicking on Instrumentation->LOG and selecting Insert LOG. Change the name to `trace`. Set its properties to have a length of 512 and be a circular buffer.
- Change the `LOG_system` object to have a length of 512.
- Create a MBX object by right clicking on Syncronization->MBX and selecting Insert MBX. The new mailbox will be called `MBX0`. Change the `MBX0` properties so that the message size is 4 and the message length is 3.
- Create two TSKs with the following properties
  - Name: `TSK0`, priority: 1, function: `_funTSK0`.
  - Name: `TSK1`, priority: 1, function: `_funTSK1`.
- Create a main.c file and include a `main` function that does nothing. Include this file in your project.
- In the main.c file make functions for your TSKs, `funTSK0` and `funTSK1`.
- In `funTSK0` add the following code:
  - Write to the `trace` LOG indicating that the `TSK0` is starting and ending.
  - A loop that will loop 5 times.
  - Inside the loop put the loop number in an `Int` message (4 MADUs) and post this to the mailbox `MBX0`.
  - Print to the `trace` LOG indicating which message is being written.
- In `funTSK1` add the following code:
  - Write to the `trace` LOG indicating that the `TSK1` is starting and ending.
  - A loop that will loop 5 times.
  - Inside the loop pend on the mailbox `MBX0` and get the message from the mailbox.
  - Print to the `trace` LOG indicating which message is being read.
- Build and load your project.
- Use the Execution Graph and the LOG manager to examine the processing. Open the RTA Control Plannel by selecting it under the DSP/BIOS menu. Uncheck the Enable SWI logging. This will cause the KNL_swi to not be logged since we are not interested in when it runs.
- Run the program and record the results.
- Describe in detail the processing that is occurring.

Part 2

- Change the priority of `TSK1` to 2.
- Build and load your project.
- Run the program and record the results.
- How does the execution change? Describe in detail.