2011 Lab 3 and Lab. 4

Please check the lecture notes for more details

The idea behind Lab. 3 is to use your code from Lab. 1 and Lab. 2 to produce a working embedded system handling a number of different operations

- Lab. 3 starting position. All the tasks should occur "simultaneously' – meaning they should not block each other.
    - Construct a TTCOS operating system project
    - Play-back an input audio signal (will happen automatically if you include ProcessData.cpp
    - while flashing LED 6 at a rate close to your heart beat and
    - while setting the other LEDs in response to the switch positions.

- Lab. 3 – Develop an audio bio-feedback device
    - Operation 1. Construct a mock temperature device and test it using EmbeddedUnit testing framework
    - Operation 2. Use the mock temperature device to demonstrate bio-feedback control of an audio signal
    - Operation 3. Use SW3 to control the rate at which LED5 flashes –(20 lines TTCOS Task)
    - Operation 4. Use PF11 to read the signal from a TMP03 thermal sensor (3 TTCOS tasks, average of 20 lines for each task) – and then use the true temperature settings for the bio-feedback control of an audio signal
    - Operation 5. Use a Blackfin timer to measure the signal from a TMP03 thermal sensor (1 new TTCOS task which reuses the TTCOS task from operation 4.

  Provide demo and documented code – no other write-up for Lab. 3

- Lab. 4 Prelab -- Operation 6. Using switches on the logic lab to sent initialization and data signals to a LCD screen to display a message and a (faked) temperature value. This will happen during the Friday tutorial period (Section 1) and lecture period (Section 2) after the midterm November 19th – about 30 minutes work and counts for 40% of the mark for Lab. 4 – make sure you are ready for the demo. There is a simulator available
- Lab. 4 Operation 7. Use the Blackfin SPI interface to sent initialization and data signals to a LCD screen to display a message and the temperature value sent by the TMP03 thermal sensor. The temperature is updated every time SW1 is pressed. Otherwise the LCD screen is to display a variety of Xmas messages. (3 TTCOS tasks, average of around 20 lines for each task.)

Suggestion to solve these problems.

Approach 1) Quickly read the following notes and the associated lecture notes to get an idea of what to do, and then go away and solve the problem yourself with your own design

Approach 2) Read each task, and complete the task as detailed. Make sure that you think through the design.

REMEMBER – Get one person in the group writing the task and testing it, while the second person in the group, starts writing the next task and has it ready for testing.

Unless you share the work, and get a lot done before you come into the lab, you will never get it done in time

The tasksare typically around 10 lines of code – some being as small as 1 line of C++ code, others being around 30 lines of code

Lab. 3 Step 1.) Start position. Demonstrate using TTCOS the following tasks -- this is the end state for Lab. 2.

A) **FlashLED 6( )** – this LED flashes at a rate roughly equivalent to your heart rate – based on a 2 state task

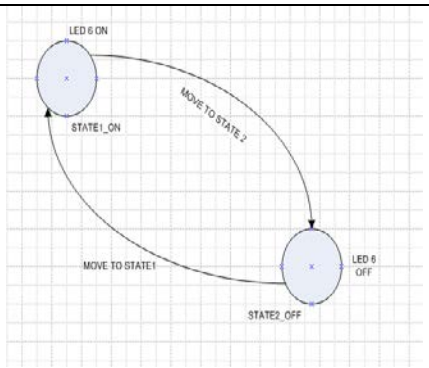
B) **DisplayLED12345( )** – This task picks up the value from a global variable *unsigned char LED12345* and display the value on LEDs 1,2, 3, 4 and 5 without interfering with the operation of FlashLED6( ) – this single stage task runs every 1 / 30 second (1 line of code) (Running this task this frequently enables later tasks to more work easily)

C) **ReadSwitches( )** – This task picks the values of the GPIO switches and stores them in a global variable *unsigned short int switchValues* – this single stage task runs every 1 second (1 line of code)

D) **DisplaySwitches( )** – This task picks up the values in *unsigned short int switchValues* and copies the value to *unsigned char LED12345* – this single stage task runs every 1 second (2 lines of code)

The delays of the tasks should be adjusted so that *ReadSwitches( ) runs before DisplaySwitches( )*

NOTE: For the later tasks, you may need to delete or modify these tasks from Lab. 3.



Lab. 3 Step 2.) Demonstration of a mock (fake) temperature device

Hook up a sound source (iPod) and head-phones to the Blackfin.

Down load the executable *Lab3_AudioDemoOct2010.dxe* and run it. This code makes use of a *MockTemperatureDevice( )* to provide a fake room temperature reading. Each time this function is called, it increases the global variable 'volatile int roomTemperature' by 1 C until the temperature becomes 35 C, and then decreases the temperature by 1 C until the temperature become 20 C. This corresponds to somebody continually warming a temperature sensor and then cooling it.

If you listen to the music, you will hear the orchestra being moved from the left side of the room to the right side of the room as the (mock) temperature increases, and back to the left as the (mock) temperature decreases. (You will have to take ENCM515 to find out how this can be coded.

Lab. 3 Step 3.) Build a mock (fake) temperature device

Down load the object file *Lab3_ProcessData.doj* into the *ENCM511 / Lab3*

Generate the source file *MockTemperatureDevice.cpp* in *ENCM511 / Lab3* with a stub for the function *void MockTemperatureDevice(void)* and a statement *volatile int roomTemperature;* ***WARNING THIS STATEMENT MUST BE CHANGED to extern volatile int roomTemperature FOR Step 4***

Build a test project in *ENCM511 / Lab3Tests* and add links to the files *Lab3_ProcessData.doj* and *MockTemperatureDevice.cpp.* Write tests to show that *MockTemperatureDevice( )* (when working) satisfies the requirement -- *Each time this function is called, it increases the global variable 'volatile int roomTemperature' by 1 C until the temperature becomes 35 C, and then decreases the temperature by 1 C until the temperature become 20 C. This corresponds to somebody continually warming a temperature sensor and then cooling it.* **Provide the tests and code as part of your lab 3 report.**

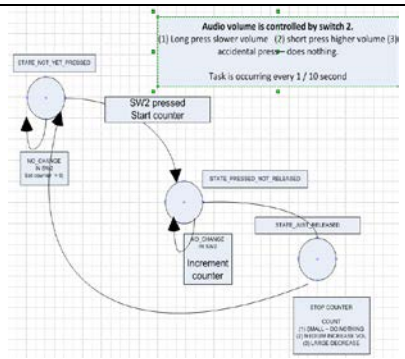
Lab. 3 Step. 4.) Using a mock (fake) temperature device

Build a standard TTCOS project in Lab 3. Add the working code for *MockTemperatureDevice.cpp* and replace the Lab 2 version of *ProcessData( )* with *Lab3_ProcessData.doj.* Add *MockTemperatureDevice( )* as a TTCOS task running around every 1/ 3 second. ***CHANGE TASK 3 STATEMENT MUST BE CHANGEDfrom volatile int roomTemperature to extern volatile int roomTemperature FOR Step 4***

Build and run – if your code from step 3 is correct – then you will hear the audio sound moving from left to right and back as in **Lab. 3 Step 2.)**

Lab. 3 Step. 5) FlashRateLED5_SW3( ) is a new (multi-stage) task that runs every 1 /10 second and examines the setting of SW3 in the global variable *unsigned short int switchValues*.

- If SW3 is held high and released after a long period of time, decrease the flash rate on LED6 slightly (30%). A couple of long presses of SW3 will be needed to make the flash rate change a lot.
- If SW3 is held high and released after a short period of time, increase (faster) the flash rate on LED6 slightly (30%). A couple of long presses of SW3 will be needed to make the flash rate change a lot.
- If SW3 is held high and released after a very short period of time, leave the flash rate of LED6 unchanged.
- Check that the rest of the system still works if you do a short press and release SW3 10 times (why could this be a problem)



Lab. 3 Step 6. Stop the task **DisplaySwitches( )** which copied the values in *unsigned short int switchValues* into *unsigned char LED12345* and replace it by a new task ***DisplayTemperatureAbove20C ( )*** which runs every second which take the value in the global variable int *roomTemperature,* subtracts 20 from the value. If SW1 (in *switchValues)* is pressed, then place the result (temperatureDifference) into *unsigned char LED12345* where the temperature difference will be automatically displayed by the task **DisplayLED12345( ). Remove the temperature display when SW1 released** (4 lines of code). Note that negative temperature differences (below 20) will alias to high temperature differences (the number 19 appears as the number 35 (underflow), the number 18 appears as the number 34 –Be prepared to explain why on the final exam.

If you now run the code with the mock temperature device, you should see the LED values increase and decrease as the music moves from left to right and back

Lab. 3 Task 7. CalculateTemperature( ) is a task that runs every 1 second and takes the values from two global variables *unsigned long int timeT1_HIGH* and *unsigned long int timeT2_LOW* and uses the formula in the TMP03 reference manual to calculate a temperature which is placed into the global variable *volatile int roomTemperature* where it will be displayed in LEDs 1, 2, 3 and 4 by the task ***DisplayTemperatureAbove24C ( ).***

Check in your lecture notes for typical values to be placed in *timeT1_HIGH* and *timeT2_LOW* to get temperature reading of 25C with an accuracy of around 1%. (1 line of code). Do a number of tests for different temperature readings by changing T1 and T2

IN LAB REPORT – PROVIDE THE CODE AND SCREEN DUMP FOR TESTS THAT VALIDATES THIS CALCULATION ROUTINE

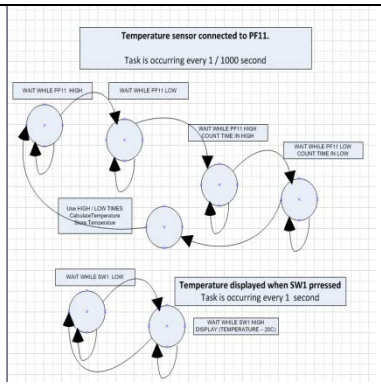
Lab. 3 Task 8. Hook up the TMP03 thermal sensor to +5V and ground and check that you see the pulse coming up of the sensor. The pulse is about 1/33 s duration**. Don't waste very much time verifying that T1 (time high) or T2 (time low) changes with temperature, the change is very small.** This is a simple to test to check that the thermal sensor works and has not been destroyed by previous 511 students. WARNING connecting up the 5V and ground the wrong way will destroy the sensor.

Lab. 3 Task 9. Hook up the ground from the thermal sensor to the ground of the Blackfin extension board and the output of the thermal sensor to PF11. The ground is very important.

Lab. 3 Task 10. Determine_T1_T2( ) is a task that runs every second.

- Remove the mock temperature device from the TTCOS project
- This **Determine_T1_T2( )** task launches a high speed task **MeasureTimes( )** that runs every 1/ 10 ms to determine how long the output of the TMP03 sensor stays high (answer placed in the global variable *unsigned long int timeT1_HIGH)* and how long it stays low (answer placed in the global variable *unsigned long int timeT2_LOW).* (29 lines of code, mostly cut-and-paste)
- The task *Destroy_MeasureTimesTask( )* is also launched which destroys the MeasureTimes( ) task after about 1/5 second.
- Very similar to **FlashRateLED6_SW3( )** I suggested in the class notes that the high speed task **MeasureTimes( )** has 4 states – wait while high (discard value), wait while low (discard value), measure while high, measure while low. If you want more states – use them
- Note that if you don't get enough accuracy in the measurement of T1_High and T2_Low then the temperature value will not be stable and will change a lot.

The temperature values will automatically be displayed in LEDs 1, 2, 3, 4 and 5 whenever switch 1 is pressed -- temperature 30C appearing as 10



Lab. 3 Task 11. Replace task **DetermineTemperature( )** with a new task **DetermineTemperatureUsingTimer( )** which replaces the high speed task **MeasureTimes( )** that runs every 1 ms with a low speed task that uses the Blackfin General Purpose Timer 2 in capture mode (Chapter 15) to determine the period and width of the TMP03 thermal sensor signal. (10 %) **Demonstrate when complete.** *Hand in code showing you are using multi-stage tasks that do not block each other.* – 11 lines of code

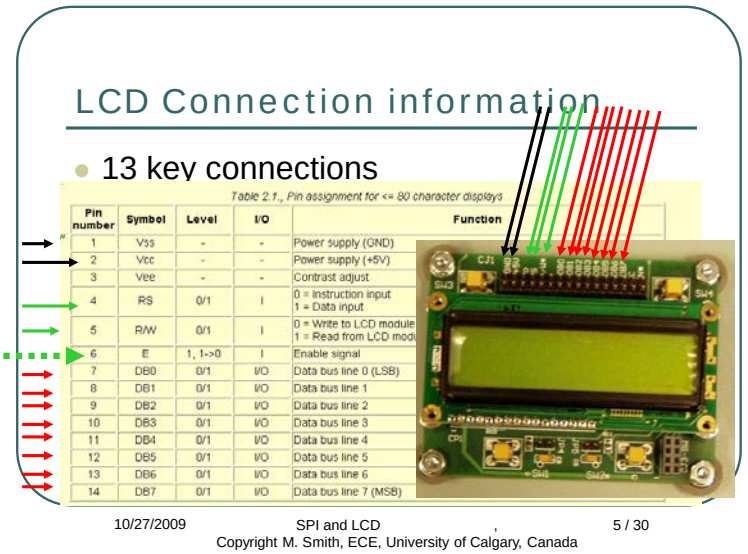
Lab. 4 Prelab. – Manual control of LCD screen. This will happen during the Friday lecture period after the midterm November 19th – about 30 minutes work and counts for 40% of the mark for Lab. 4 Last year, students had a lot of fun using switches to send messages directly to the LCD screen. I'll tell you how to do this.

Test the LCD screen using control signals sent from the switches on the logic laboratory.
Get TA to sign off that this test works. Each person in the lab. group must wire this themselves. That means after the demonstration by the first lab. partner, all the wires must be removed

NAME _____ MARKED BY _____ 40% of LAB. 4

These are new LCD screens -- check that they function as expected. MSB means most significant bit

- **Power down logic Lab**.
- **Hook up Lab switches 7 to 0 (MSB to the right) to LCD input lines 0 to 7** (MSB to the left **-- no the other left**). (SW0 to LCD0 etc). Set all switches low
- **Connect switch 8 to E line on LCD.** Set switch high
  - This is the LCD accept data / instruction bit
  - Must go High / Low / High
- **Connect switch 9 to R/W* line on LCD.** Set switch low to make sure LCD only accepts values written to it from the Blackfin
- **Connect switch 10 to RS line on LCD.** Switch low as we are starting to transmit instruction to LCD
- **Add +5V and GND lines to LCD**
- **Power up logic lab.**



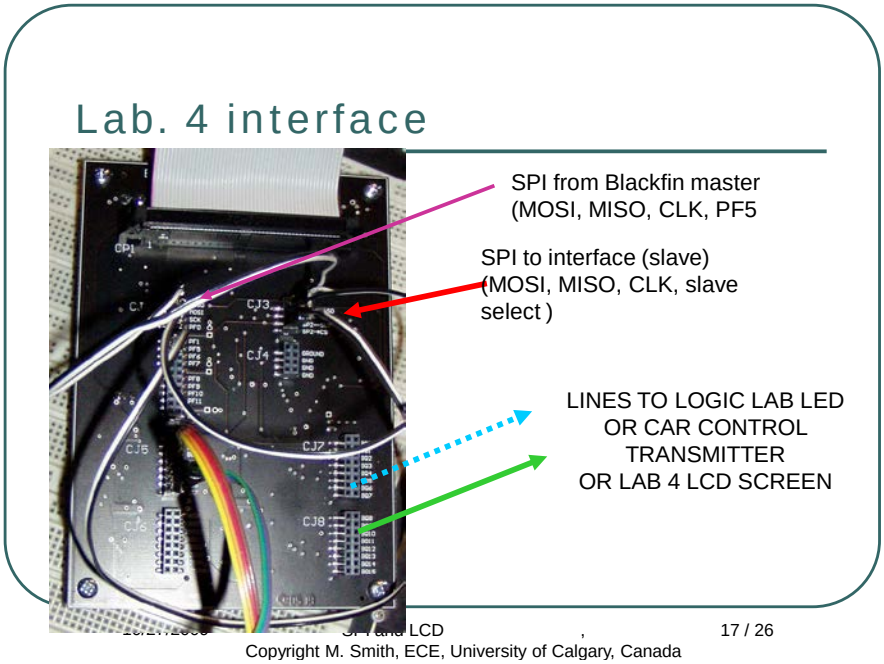
Set the following signals to the LCD
HERE IS THE LINK TO THE SIMULATOR

Initialize the LCD – send commands to the LCD screen	
Set logic lab switch 8 to high. Command request if SW10 is low	
Set logic lab switch 10 to value 0, Set logic lab switches 7 to 0 to value 0x30 Sets interface data length (DL)	Toggle logic lab switch 8 low then high to cause LCD to accept request
Set logic lab switch 10 to value 0, Set logic lab switches 7 to 0 to value 0x30 Sets interface data length (DL)	Toggle logic lab switch 8 low then high to cause LCD to accept request
Set logic lab switch 10 to value 0, Set logic lab switches 7 to 0 to value 0x3C Sets interface data length (DL) number of display lines (N) and character font (F) DON'T DO THIS INSTRUCTION ON THE SIMULATOR	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 0, Set logic lab switches 7 to 0 to value 0x0F Display on, number of display lines, cursor on, cursor blink	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 0, Set logic lab switches 7 to 0 to value 0x01 Clear Screen and put cursor to start of first line	Toggle logic lab switch 8 low then high
Send message – send data to the LCD screen	LCD data request if SW10 is high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x20 – ascii space	Toggle logic lab switch 8 low then high to cause LCD to accept request
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x20	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x35	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x31	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x31	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x20	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x20	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x52	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x55	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x4C	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x53	Toggle logic lab switch 8 low then high
Set logic lab switch 10 to value 1, Set logic lab switches 7 to 0 to value 0x20	Toggle logic lab switch 8 low then high
Now work out the code to add your initials to the screen	

Lab. 4 -- Task 1 -- Testing the SPI and LCD connections

1) Hook up the SPI wires on the Blackfin Remote Extender Board interface as discussed in class - **DON'T CONNECT TO THE LCD AT THIS TIME (BY HAPPEN-CHANCE ONE OF THE LCD CONNECTIONS IS LABELLED CJ2)**

CJ2 connector	**CJ3 connector**
MOSI	SP2-MOSI
MISO	SP2-MISO
SCK	SP2-SCK
PF5	SP2-CS*
Make sure that you do the following	
CJ4 GND to Logic Lab ground	



2) Connect the logic lab lights to Blackfin Remote Extender Board interface. CJ7 to lights 0 to 7 (match the pin numbers) and CJ8 to lights 8 to 15 (match the pin numbers)

3) Download and run this VDSP executable Lab4LightsTest.dxe (get from web page), you will see the lights flashing from right to left if all the wires are hooked up correctly. As discussed in the earlier laboratories, you can download an executable from inside VDSP using "control-L".

4) If the flashing lights don't work, check the wiring. It needs to be correct for the next tasks.

NOTE -- WE HAVE FOUND SOME STATIONS FAILING THIS TEST AND OTHERS PASSING IT EVEN WHEN THE WIRING IS CORRECT.

IT FAILS WHEN ONLY ONE LIGHT IS FLASHING ON AND OFF

This is a timing issue associated with stray capacitance on the board. You can actually make the code pass / fail by attaching scope probes to the LCD MOSI line or getting your hand close to the line.

You can do the following to make the code more reliable if it is failing the test.

- Download the code and start it running
- Use the menu option DEBUG | HALT to stop the running code
- Use the menu option REGISTER | EXTERNAL PERIPHERALS | SPI CONTROLLER to bring up the registers for the SPI controller
- The SPI_CNTL register will display 5901 -- high light the 5901 (actually hex 5901) with the mouse and right click
- Select the command EDIT and type in the value 5101 -- this changes the clock timing as discussed in class

- Use the menu option DEBUG | RUN to start the running code -- you should now see the test operate properly

WARNING -- IT IS VERY IMPORTANT THAT YOU TRY THIS TEST -- WE HAVE FOUND THAT A NUMBER OF STATIONS HAVE INTERNAL FUSES THAT HAVE BEEN BLOWN -- SO THE SPI WILL NOT WORK