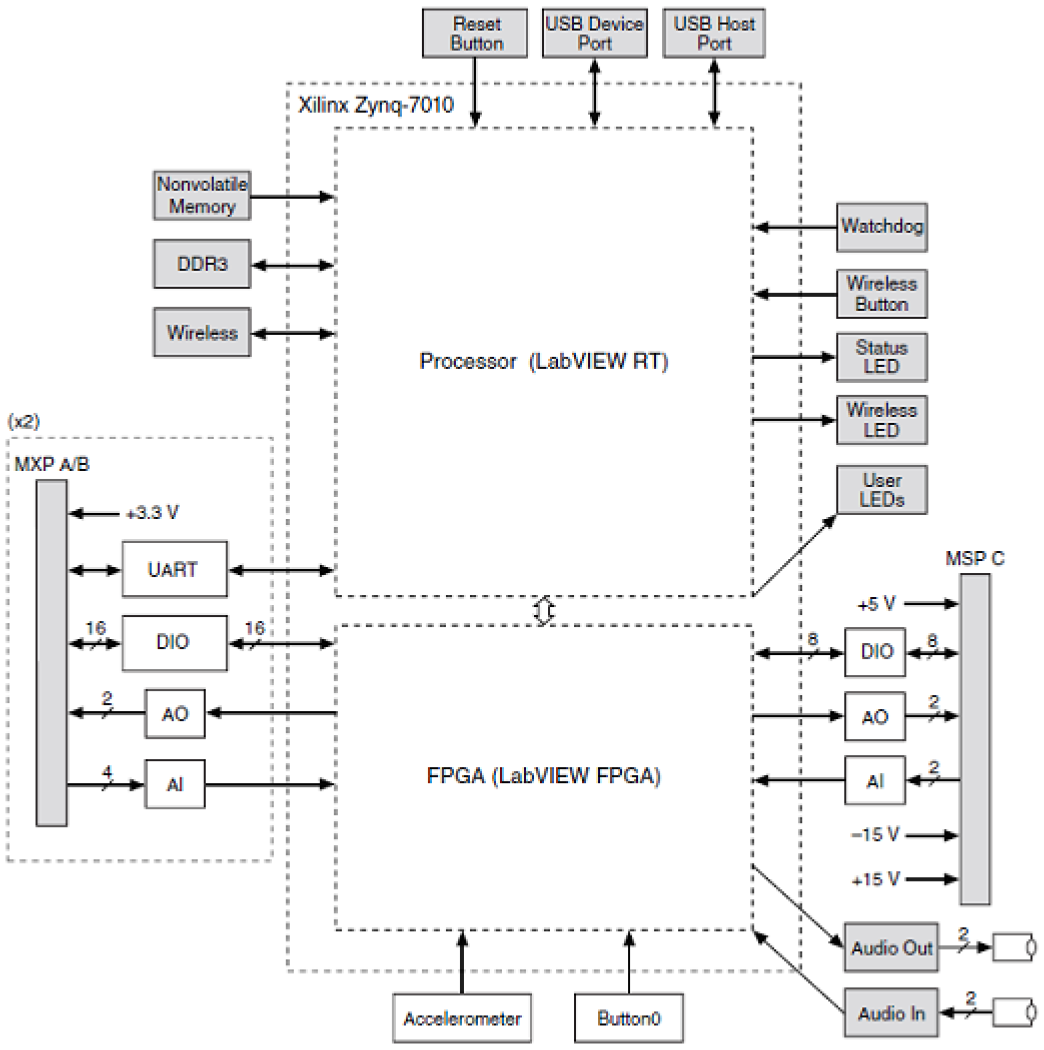
Mectron/Sfwr Eng 4aa4 - Lab 4

Design of real time tasks for signal generation

Introduction:

As mentioned earlier in lab 3, NI myRIO integrates the software programability of a processor with the hardwere programability of an FPGA. So far in the labs we have used the processor only, however most real time systems need to react with the environment, which requires an interface between the processor and external devices. NI myRIO provides shuch an interface using its FPGA. Refer to the following diagram taken from MyRio.pdf:



Support provided for a number of peripherials is described in the document named 'myrio_personality_v3.0.pdf'. This document also describes the naming conventions for naming different control registers associated with peripherials and how to access and control them. The interface implemented on the FPGA of myRIO is described in a set of files in the folder 'C Support for myRIO' which was imported as a part of the example projects into your workspace in a previous lab. A bitfile that implements hardware has already been installed on myRIO boxes in the lab. Files such as 'MyRio.h, NiFpga.h' and corresponding C files have details of the API functions. You can also get details of the API at http://zone.ni.com/reference/en-XX/help/372928G-01/TOC2.htm. Note that when a template project is created its main.c file has the minimum code to use myRIO interface. Particularly it has #include "MyRio.h" and the first function call within the main function is 'MyRio_Open()', which opens the myRIO NiFpga session. You must use this function before using all the other functions. After you finish using this function, the NI myRIO target is ready to be used. You must close this session with 'MyRio_Close()' after using all other function. Looking at the example projects can provide more information about the use of various functions in the API.

Goals:

- Learn how to create periodic real time tasks for controlling certain devices
- Create a periodic task to generate a signal to blink one of the LEDs on myRIO.
- Create a threaded task to generate a signal to blink the LEDs on myRIO.
- Create a timer thread to generate a signal to blink the LEDs on myRIO.

Note: Before you go for your lab sessions, please read the following documents at your own convenient time, in addition to the class notes:

- C Support for myRIO User Guide6.0.pdf. Most of the software components mentioned in this document have already been installed and configured on lab computers.
- Getting Started with C Development Tool(Eclipse).pdf
- Document at this link: http://www.drdobbs.com/soft-real-time-programming-with-linux/184402031. Or, a PDF version can be found in folder of "ref" for Lab 3 .
- Document at this link: https://hpc-tutorials.llnl.gov/posix/pthreads api/
- **Please familiar with the "Create–>Build–>SSH connect–>Run", four basic steps in Lab 3.**

Activities in this lab:
Note: Please use a separate folder for each of the following activities in the lab:
Part 1: [40]
Blinking an LED on myRIO using a periodic real time task:

- NI myRIO has four user LEDs named LED0 to LED3. The goal is to create a real time periodic task that alternately writes a 0 or a 1 to the bit that represents an LED.

- Import the **C Support for myRIO1900 v6.0**(.zip file) as we did in lab 3. Find the **'myRio Template'** project, then rename it to a suitable name.

- Double click on the project name in the workspace and use the triangle to the left of the project name to open the folder. There is a 'main.c' file in addition to other folders/files.

- Double click ‘main.c’ which opens in an editor. Notice that the template contains basic code to open the myRIO FPGA session. You will use any other functions that may be required after the session is opened.
- Before you start adding code to main.c in this project, open the example project named ‘myRIO Example - DIO’.
- Note that the project includes two files named ‘DIO.h’ and ‘DIO.c’ which contain functions for reading from or writing to a DIO port. You must copy these two files into your project. Also add ‘#include “DIO.h” ’in the file main.c of your project.
- Make changes in the main.c file of your project to create a periodic real time task as done in **lab3 Part 3**.
- For each period, the task is required to write a 1 to the bit that corresponds to an LED in a control register for the LEDs, if the previous value was 0 and a 0 if the previous value was 1.
- Find out the name of the control register that you want to use from myRIO personality, e.g. the name given to LED register in the myRIO personality is ‘DO.LED3:0’. While using register names in C code punctuation marks are eliminated. So the name for LED register will be ‘DOLED30’ when used in C code. Note that this is an eight bit register. You can use the API function ‘NiFpga_WriteU8(myrio_session, led,val)’ to write a ‘val’ into ’led’ where led is a variable that has the name of LED register saved in it.
- In order to make blinking of LED visible to human eye, choose the time period of the task to be half second or more.
- Build your project and run on myRIO. Here we choose the **LED0** blink.
- Show the output to one of your TAs!

Part 2 [30]
Blinking the LEDs on myRIO using a threaded periodic real time task:

- Import the **C Support for myRIO1900 v6.0**(.zip file) as we did in lab 3. Find the **’myRio Template’** project, then rename it to a suitable name. Do not forget to copy ‘DIO.h’ and ‘DIO.c’ files from the sample project to your new project.
- Code the ’main.c’ file of the new project such that a prieodic task is created as a thread. Refer to **Lab3 Part 4** if necessary.(**If you cannot finish the mutli-thread to control the LEDs by turn, then just use one thread to control the LEDs by turn.**)
- Ensure that the **LEDs blink by turn** for this part.

- Compile and run your project and show the output to one of the TAs.

Part 3 [30]
Introduction to threaded interrupt handlers and Use a timer IRQ to blink the LEDs:

- The PREEMPT-RT Linux has all the interrupt handlers running as regular kernel threads; this way, a high priority thread can avoid being preempted by the interrupt handler thread by setting its priority to be higher than the interrupt handler thread priority.
- An example project named 'myRIO Example - TimerIRQ provides and example of a threaded IRQ for a timer.
- Open the example project, read its 'main.c' file and try to understand how the code works.
- Compile the project and run on myRIO. The Time IRQ occurs only once after Timer IrqConfigure().
- The timerIRQ only printed the IRQ number and its count. Now, you are required to use the timerIRQ to blink the user LEDs by turn on myRIO.
- This requires that the IRQ should keep on repeating with a set interval or time period. This can be done by calling the functions: Irq UnregisterTimerIrq() and Irq RegisterTimerIrq() from inside the Timer Irq Thread().
- Import the **C Support for myRIO1900 v6.0**(.zip file) as we did in lab 3. Find the **'myRio Template'** project, then rename it to a suitable name. Do not forget to copy 'TimerIRQ.h' and 'TimerIRQ.c' files from the sample project to your new project.
- In 'main.c' file of your project, replace the code that prints the IRQ number and its count in the example project with suitable code to cause blinking of all LEDs(LED0 to LED3) by turn in MyRIO.
- Call the functions to un-register the IRQ request made in the main() func-tion and then register a fresh IRQ with a timeout value equal to the desired period.
- Complie and run your project on myRIO.
- Show results to one of the TAs.