**Comparing Non-Real Time and Real Time Systems**

**Description:**

The objective for this project was to be able to compare the benefit of a Real Time System (RT) to a Non-Real Time System (NRT). Simply stated, there are two main programs. One is written in kernel space as a real time task and the other is written in user space, NRT. When run, they both take in data from a sine wave. Load is added to both in controlled increments to better gauge the value of using a real time system over non-real time. The results should give a better understanding and appreciation for real time embedded systems. It shows real time benefits, such as less data lost, and the ability to visually see the frequency of RT and NRT with and without loading.

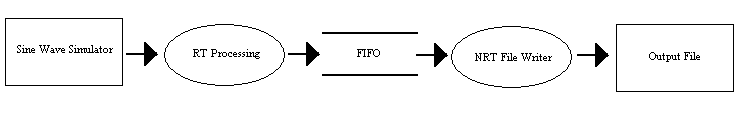
**Implementation:**

The project is split into three major components, the simulation of the sine wave, the RT system, and the NRT system.

Arduino Sine wave

A program was written using C++ on an arduino to simulate a sine wave. In a while loop it calculates the sine wave from 0-360 degrees in increments of one degree. This is then scaled to be represented from 0 to 255. This allows for the information to be sent out on 8 pins to the board. The b port on the TS-7250 board was used. This was because it was a familiar port one had seen in the lab. The advantage to this method is that one did not have to use a function generator. This would require getting permission to bring it into to the lab etc.

Real-Time Task



**Figure 1: Real-Time Task Model**

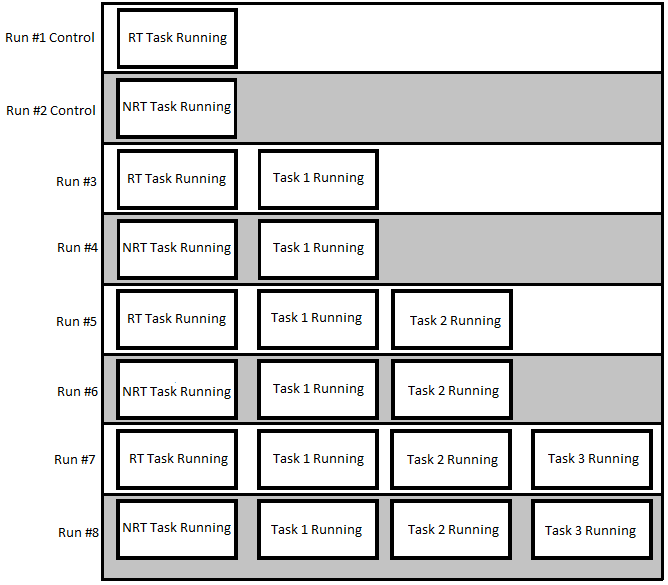
The real time task in kernel space is used to read data from Port B, whose period is 1µs. This period is as fast as I could manage. The information is then sent to a buffer in user space, via a fifo to be written to a text file. This text file allows for easier access to the data, and easier extraction to a graphing program. The data is graphed using Excel. This is simple and convenient.

Non-Real Time Task

In user space, the program takes in the data from port b and writes it to a file to be compared with the real time information. There was a delay placed in this programs while loop to even the initial amount of data without load with the real time task. This delay was added such that the RT and NRT plots could easily be compared. It is graphed in Excel as well.

Both

Both programs were run separately with varying amounts of load. The RT was run with no load first, as was the NRT. Then, load was added by adding another kernel module, that simply wastes time. Both RT and this new kernel were run together. Then, the NRT was run with the same new kernel. Instead of installing multiple kernel modules to run at the same time, as previously thought of. One simply increased the amount of time this kernel would waste. An rt\_busy\_sleep was run with increments of 1000=1extra task, 2000=2extra tasks, 3000=3extra tasks, to simulate the varying amounts of load. These were recorded and graphed for comparison.

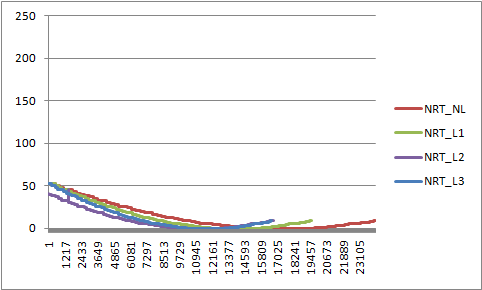


**Figure 2: Testing Procedure Model**

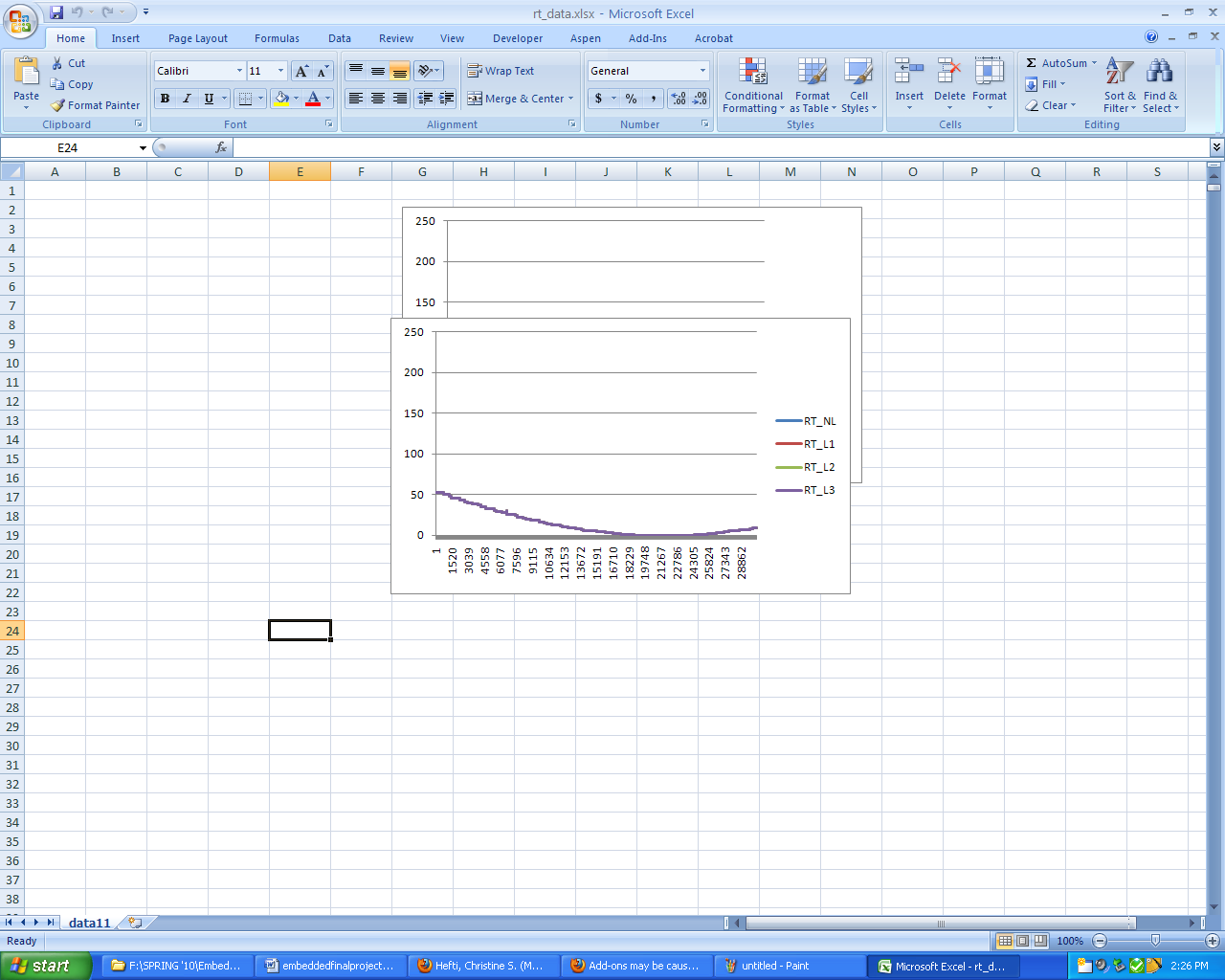
**Results/ Conclusions:**

The results were satisfactory. As can be seen in the graphs below, the real time tasks data tended not to change. The non real time task, however tends to make the period smaller as more load is added to the system. The plots below are simply a listing of all the data received instead of plotting the data against time. Therefore, when less data is collected, over the same interval of data, the frequency declines due to fewer data points. The board takes in the information several times before it gets the new value. If the speeds matched, then it would be expected that the graphs would turn out jagged when load was added to the non-real time task. This shows that the Real-Time system has better performance, more accurate graphs, and is good for using when time and load are factors. Non-real time performed alright for no tasks running and is bad if time dependent and multiple programs are running at the same time.

The plots below show the value of the sine wave that was scaled to be between 0 and 255 or 1 byte of data, the Y axis, and the X axis represents the number of data points taken, not time.



**Figure 3: Non Real Time Data**



**Figure 4: Real Time Data**

The biggest problem encountered was the incompatibility of the arduino used to simulate the sine wave and the TS-7250 board. It took a long time to try and match the delay times of the arduino with the program times so that the data could be easily interpreted. They are still off, but close enough to see the difference between RT and NRT. Also, one tried to use UART on the board first, instead of the b port. This failed, probably due to set up or the clock frequency being incorrect.

**Code:**

Code is attached

RT code: FinalProjectb & FinalProjectmain

NRT code: FinalProjectb2

Arduino: send\_sin\_bit

Load code: FinalProject