University of Auckland

Department of Electrical and Computer Engineering

COMPSYS 723 – Embedded Systems Design – Assignment 1 (15%)

Zoran Salcic, Johnny Yeh, Benjamin Tan

1. Introduction

Power systems are designed to operate at a specific frequency. For example, mains power in New Zealand is specified to operate at 50 Hz AC. However, the frequency of a system can vary according to the balance of supply and load in the power network. Over-frequency occurs when power supply is in excess. Conversely, under-frequency happens during overload situations. Off-nominal frequency operation can have devastating effects to the components within the network and can even cause a total disaster of the power system. Usually this balance is carefully monitored and controlled to ensure the system operates at optimal frequency or within the target range. However, this frequency can still change during disastrous situations such as generator failure. A frequency relay is a device designed to accurately detect abnormal power frequency and react quickly by tripping or reconnecting loads in an attempt to maintain a stable network frequency. Usually, frequency relays are attached to the zones of the power system where they can control the major loads. A completely different approach is to be able to accurately measure frequency and control the load at the lowest point of consumption, in this case, at the household level. Instead of shedding the load of a complete zone, we can selectively shed the loads in a multitude of households and make delivery of energy selective, yet more fair. A low-cost frequency relay (LCFR), which is the target of this assignment, is aimed at smaller individual consumers. It is capable of measuring the frequency in real-time and tripping or reconnecting the loads within the consumer scope, for example by controlling all individual sub-circuits in the home. Obviously, frequency relay in a single home does not contribute much to the frequency of the overall power network, but by deploying tens or even hundreds of thousands of these devices in home across New Zealand would make a significant difference. By their simultaneous action, these relays can be helping to keep the overall power system network stable.

2. Objective

The LCFR is a system that interfaces directly with the power network to measure current frequency and is able to switch on and off loads by using triacs (Triode for Alternating Current). It connects to the power network voltage and samples it to determine the frequency and rate of change of frequency in real time in order to make decisions on load tripping. A part of the device must use digital signal processing algorithms to prepare information for frequency and its rate of change calculation. This can be done by the use of these algorithms implemented in digital hardware or by using very fast and expensive processors. In our case we opted for hardware implemented unit for frequency calculation. The remaining tasks of the LCFR are performed using software running on an embedded processor. The processor we are using is the Intel (Altera) NIOS II-based SOPC implemented in A Cyclone FPGA on the DE2-115 development board. The aim of this assignment is the implementation of the application software of a frequency relay, targeted for use in normal households. The software will control load tripping according to the measured frequency of the power network. The application software written in C will be partitioned into concurrent tasks implemented on a real time operating system (in our case, FreeRTOS). In this assignment you are expected to utilise the multitasking abilities of the operating system, and to develop a thorough understanding of embedded system design issues such as concurrency, synchronisation, and use of memory-mapped devices.

3. Learning Outcomes

After completing this project, students will gain deep insight on the following:

- Task-based concurrency and synchronization: students will be able to design embedded systems consisting of multiple tasks. They will be able to identify required tasks and any associated synchronization issues and use suitable synchronization mechanisms provided by an RTOS.
- Embedded implementation and controlling memory mapped I/O peripherals and devices: students will learn how to implement a full embedded system using the SoC-based approach. Learning how to create programs which control memory mapped devices, such as VGA controllers or PS/2 controllers, is also an important goal of this assignment.
- Elements of HW/SW partitioning and co-design as a part of SoC-based approach will be explained, although no specific hardware will be designed for this assignment and will be given as already existing IP block.

4. Teaching Assistants

Feel free to contact the TAs via email for assistance in the first half of this course. To make our lives easier, please use 'C723-TA:' followed by a short description of your query in the subject line, and then a more detailed description in the email body. *For example "C723-TA: Can't program my FPGA".*

- Felix Marattukalam, fmar631@aucklanduni.ac.nz
- Mubashir Wani, mwan198@aucklanduni.ac.nz
- Hamish Lonergan, hlon009@aucklanduni.ac.nz

5. Labs and Consultations

There is a set of formal lab exercises (not assessed) introducing you to the features of FreeRTOS which will be useful in this assignment. Lab sessions for this half of the course are –10-12 pm on Thursdays in 405.522 (MDLS E&I 5). TAs will be available for consultation during the first hour and will stay through the second hour if required. **You will need to attend the first lab session on Thursday 19th March and those who do not complete will have an additional session on Thursday 26th (the same time).**

6. Groups

This assignment is to be undertaken as a group of 2. Please use canvas to create your own groups. The groups need to be confirmed by Sunday Week 2 (15th March). If you have not created a group by the date you will be assigned one. During the first lab the required hardware for the assignment will be issued to each group.

If any issues with regards to teamwork occur, you should notify the TAs as soon as possible, so that we can find a solution.

7. Assignment Deliverables and Deadlines

Week 5, Sunday (5 April 2020, 11:59 pm)

A. A paper design of your implementation completed as a group.
   - Please submit your paper designs to Canvas.
   - Only one submission per group is required.

Week 7, Wednesday (29 April 2020, 11:59 pm)

B. A description and discussion of your project and the results in a joint documentation file.
C. Assignment source code submitted in electronic form via Canvas.
   - This code will be tested by the assessors by executing your submission on a DE2-115 board.
   - **Delivery method for the joint documentation and implementation will be via Canvas**. If any error occurs on the submission day, submit a zipped file with the file name that contains your group to the lecturer (Prof Salcic).
D. Confidential peer review.
   - Submit the confidential peer review to Canvas.
   - Non-submission of the peer review results in a penalty of 10% of the total mark for the assignment.

Week 7, Thursday (30 May 2020, TBA-lab time and around)

E. Project demo and short interview of groups.

8. Assignment Components

This assignment is worth 15% of your final grade, which is broken down as follows:
2% - Joint paper design
5% - Joint project implementation
4% - Joint documentation
4% - Final demo and interview

9. Design Requirements

The main goal of this assignment is to implement the decision making software and user interface for the frequency relay using software that runs on a real time operating system. Your software will be written in C using NIOS II EDS (Altera) and execute on the DE2-115 development board. The operating system we will be using is FreeRTOS (http://www.freertos.org/). The operating system allows the creation of multi-tasking programs by providing functionalities such as task priority assignment and inter-task communication, memory management and use of real-time. Access the FreeRTOS website to see how to use these features. Guidance is provided in the lab session.

In the real world application, the source of the voltage signal comes from the power outlet. For the purpose of this assignment, the actual power signal is simulated with data stored in the on board memory (M9K) and read from the memory with the same frequency as the data is sampled using an analog to digital converter. This enables a testing scenario that may or may never happen in a real power network

Your submission will need to satisfy all the requirements in Section 10.

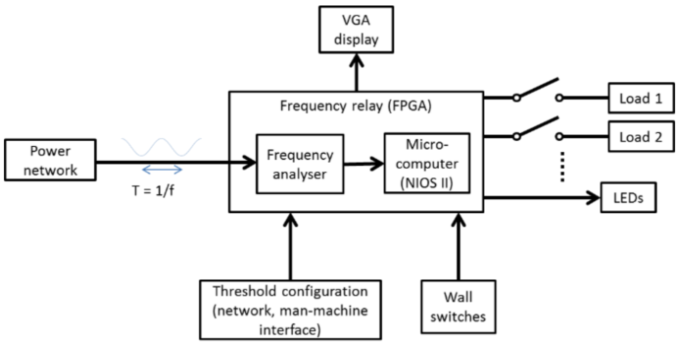

Figure 1. *Diagrammatic representation of the overall frequency relay system. The voltage is read from the power network and the frequency analyser performs signal processing to determine the period of the signal (T). The microcomputer converts the period to frequency to make load shedding/reconnecting decisions. Information of the frequency relay operation is displayed on a standard VGA display. The LEDs are used to display the current status of each load. Wall switches allow the user to manually turn on or off a load. The tripping thresholds of the frequency relay are configurable. The change of thresholds can be done through a man-machine interface such as a key pad, or over the Internet in case of devices connected to Internet.*

9.1. *Functional requirements of the frequency relay*

- The components of the overall frequency relay system are detailed in Figure 1.
- Implement the frequency relay with <u>at least five loads</u> in your system.
- Use red LEDs to represent the state of each load (whether each load is on or off).
- Each load has a priority corresponding to the LED that represents it. For example, the load represented by LEDR0 has the lowest priority, LEDR4 has the highest priority.
- Use slide switches to simulate wall switches. (Note: there is no interrupt associated with these switches). For example, sliding SW0 to the on position will cause LEDR0 to turn on (when the network is stable).

- There are two scenarios that indicate that the network is *unstable*. These will cause the frequency relay to **manage loads**, i.e. shed/turn off loads. The two conditions are *under-frequency* and *too high rate of change of frequency*.
- **Condition 1:** The frequency relay monitors the instantaneous frequency. When the instantaneous frequency drops below a set threshold (under-frequency conditions), the system should immediately start to manage loads (shedding).
- **Condition 2:** The frequency relay also monitors the rate of change (RoC) in frequency. When the rate of frequency change exceeds a configurable threshold, the system should immediately start to manage loads (shedding).
- The trip should trigger irrespective of the signedness of rate of change; i.e. only the absolute value of the rate of change is important.

- The first load shedding needs to happen <u>less than 200ms</u> after one of the above conditions is detected.
- The load that is shed first is the lowest priority load that is on. For example, if loads 2, 4, and 5 are on, load 2 is shed first, followed by load 4 etc.

- Once a load is shed, the relay should begin to monitor the network status and manage loads.
- During the load managing state, if the network remains unstable for a 500ms period, the next lowest priority load that is currently ON should be shed. This can happen until all loads are off.
- Conversely, if the network remains stable for a 500ms period, the highest priority load that has been shed should be reconnected. This can happen until all loads are reconnected.
- If at any time the network status changes from stable to unstable or vice versa before the 500ms period ends, reset the 500ms observation period at the time of change.

- The frequency relay exits the load managing state and returns to normal operation when all loads have been reconnected.
- While the frequency relay is managing loads, users can not manually turn on new loads, but can turn off currently on loads.
- Use green LEDs to represent whether the load is being switched off by the relay. For example, red LED 7 and green LED 7 correspond to the same load. If the relay sheds that load, the red LED should turn off, and the green one should turn on. Once the relay turns the load back on, the green LED should go off and the red LED goes back on.

- Implement a "maintenance" state which essentially disables the relay from managing the loads. In this state, only the switches can control whether a load is on or off. Use a push button to enter or exit this mode. In this mode, the green LEDs should remain off at all times.
- Display frequency relay information through VGA. It may look similar to Figure 2.

- You will also need to measure the time taken between detection of a deviation of frequency or rate of change of frequency from the given thresholds, to system output (i.e. the first load shed). This is to help verify that your system is meeting the real time requirements. Figure 3 provides a rough idea of what will be needed.
- A running record of the measurements should be maintained, and you should display the 5 most recent measurements, the minimum and maximum time taken, the average time taken, and the total time your system has been active.

- **You are required** to implement **at least three tasks** in addition to **at least two interrupt service routines (ISR)**.
    - This will require you to decompose the functionality of this system into smaller concurrent operations that interact each with the other.
    - As an example, you might consider a task to manage the VGA display, and an ISR to handle keyboard inputs.
    - Furthermore, you will be required to implement mutual exclusion for shared variables that can be modified by multiple tasks or interrupt service routines. For example, the status of a load may need to be protected against unpredictable concurrent access
    - You are also expected to use a variety of features provided by FreeRTOS (such as communication mechanisms), and to justify your use of these
- We suggest using the timer service in FreeRTOS to help with meeting timing requirements

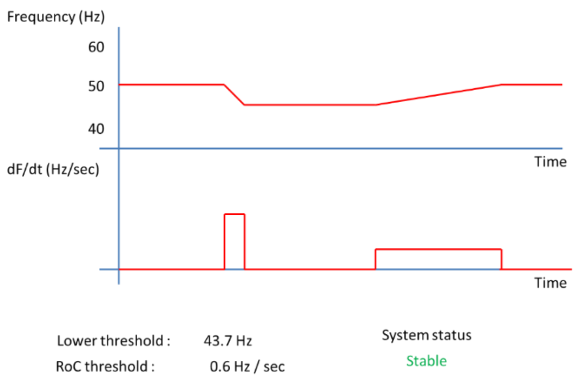

Figure 2*: An example of information display format*

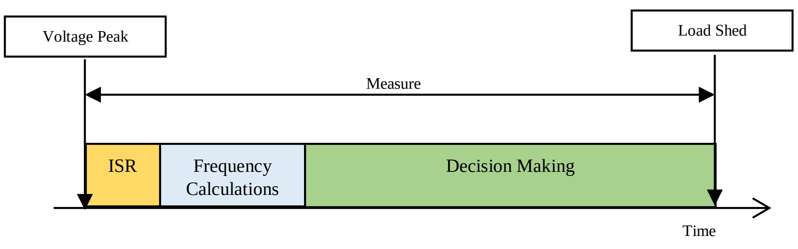

Figure 3*: A rough graphical representation of what you are instrumenting*

9.2. Paper design

Use a diagram and annotations to describe the following components clearly.

- All tasks/ISRs in your design and their functionality
- All communication/synchronization paths and what message is sent
- The conditions in which each task/ISR execute

The final implementation does not have to be the same as the initial paper design, but it is expected that you have already started implementation and have a clear idea on how to implement the required functionalities.

9.3. Final design documentation

Your final documentation should be in the form of a free-format document, in PDF format, which contains, as a minimum:

- Instructions for running your assignment on a DE2-115 board
- A description of your entire solution
  - All the tasks, what they do
  - Which tasks are periodic? Non-periodic? Why?
  - A final diagram of your design (highlighting any changes from your initial paper design)
- Justification of your design decisions
  - Why did you split functionality a certain way?
  - Why did you choose a specific mechanism for protecting shared variables?
  - Why did you choose a specific communication mechanism?
  - Why do your tasks have the priorities you’ve assigned?
  - Any issues with the application design
- Evidence that you're engaging with FreeRTOS
  - What features of FreeRTOS did you use?
  - How do your tasks interact?
  - What are potential problems with multitasking that you've overcome or prevented?
- Limitations/issues in your design
- An additional page indicating the way in which you partitioned work on the assignment, and the time taken for each part.

Your aim is to develop documentation so that a future student can understand your design, how to run your design, and know where/how to modify it if they need to.

10. Academic Integrity Notice

The University of Auckland will not tolerate cheating, or assisting others to cheat, and views cheating in coursework as a serious offence. The work that a student submits for grading must be the student's own work, reflecting his or her learning. Where work from other sources is used, it must be properly acknowledged and referenced. This requirement also applies to sources on the world-wide web.

11. Additional Information on frequency analyser

A version of frequency analyser was implemented in VHDL. The analyser reads the instantaneous voltage of the power network and provides information necessary to calculate the signal frequency. It is integrated with the NIOS II processor on DE2-115 development board. Effectively it functions as a hardware accelerator, offloading the signal processing algorithm from the main processor to the hardware component. Your application software can simply read the information provided by this hardware component to determine the frequency of the power network.

Frequency determination is achieved through a method called reference point detection. The reference points in this implementation are maximums of the power voltage wave. The symmetry about a maximum point is utilized in its identification, using a method very similar to the calculation of correlation. The correlation value is larger where it is more symmetrical about the data point being evaluated and reaches maximum at reference points. The elapsed time between two successive reference points is used to estimate the power signal frequency.

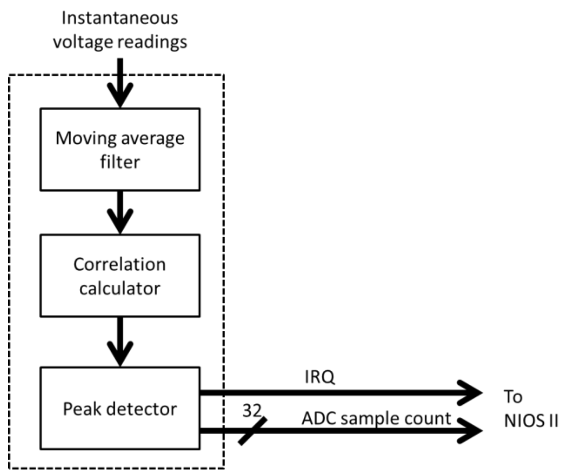

Figure 3. *Flow chart of the signal processing algorithm for reference point detection.*

The signal processing algorithm for reference point detection is summarized in Figure 3. The instantaneous power voltage is read using ADC. The raw data is filtered using a moving average filter to remove noise. Subsequently, the averaged values are used to calculate correlation. A peak detector identifies maximum correlation values and provides the time period between two peaks to the processor.

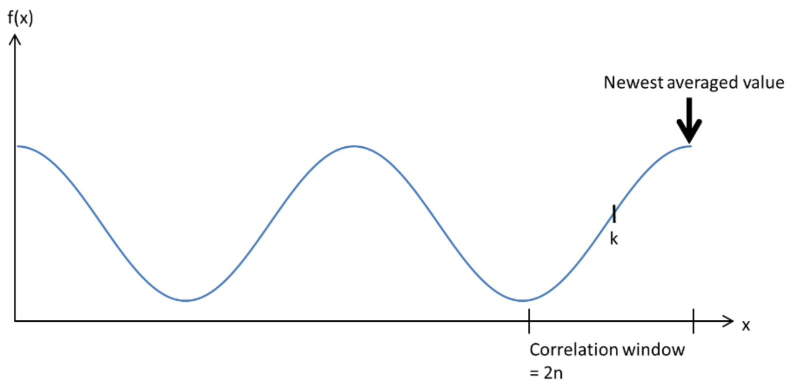

Figure 4. *Diagram showing the window of correlation calculation. The average filtered data is represented by f(x) with the independent variable x delineating each sample. Sample k is the sample around which symmetry is being evaluated. The correlation window size is 2n.*

Correlation is calculated when a new ADC reading (averaged) becomes available. If the averaged data of sample x is defined by f(x) (Figure 4), the correlation about the sample at x = k is calculated according to Equation 1.

$$\text{Correlation} = f(k).f(k-1) + f(k+1).f(k-2) + f(k+2).f(k-3)\ldots f(k+n-1).f(k-n) \tag{1}$$

In which 2n equals the number of samples used to calculate one correlation value (correlation window). The sample k is at (approximately) the centre of the correlation window.

The sampling rate of the frequency detector is configured as 16 kHz. Upon each available data, a new correlation value is calculated, and checked for whether a peak correlation value is arrived. When a peak in correlation has been detected, an interrupt request is raised to the NIOS II processor and the count of ADC samples elapsed between two newest peaks is read by the processor. Therefore, the actual frequency of the power signal can be deduced from Equation 2.

$$\text{Signal frequency} = \frac{\text{Sampling frequency}}{\text{Number of ADC samples}} \tag{2}$$

For example, if the number of ADC samples between two successive reference points is 324, the frequency of the sine wave is calculated as 16000 ÷ 324 = 49 Hz. The rate of change between two frequency readings can be calculated from Equation 3.

$$\text{Rate of change} = \frac{(\text{frequency}_{new} - \text{frequency}_{old}).\text{Sampling frequency}}{\text{Number of samples between the two frequency readings}} \tag{3}$$

The number of samples between the two frequency readings can be estimated with the average number of ADC samples of the two frequency readings.