Control Flow Diagram

Key

Task
(Priority: 1 is Lowest, 10 is Highest)
ISR
Callback
Queue
Global Variable (Mutex)
Global Variable(No Mutex)
User Interface

Keyboard ISR

Reads user inputs from PS2 keyboard on interrupt

Ps2KeyValueQ

Maintenance Mode ISR (ButtonISR)

Write →

Maintenance Mode Enabled

Read

Task1:Keyboard Manager
PRIORITY: 4

- Receive key input value
- Update Threshold

Executes every 50ms when idle and 200ms during user input
Reads from Ps2KeyValueQ

Task2:Load Manager
PRIORITY: 2

- Relay Controller

- Detects tripping condition based on frequency value or ROC

- Starts/restarts timer for stability observation

- Sheds/Reconnects load every 500 ms based on based on stability requirements

- Maintenance mode for debugging

- Check switches for user request to turn ON/OFF loads. (In Normal and Maintenance) and to disable relay control in Load Managing State

- Sets/Clears Green/Red LED based on user request and relay operation

- Updates running frequency and roc data

- Computes reaction time stats for first load shed

Executes periodically 10ms
Reads from freqRocDataQ

Task 3: VGA Monitor
PRIORITY: 1

- Read Frequency,RoC Data
- Read Reaction Time Data
- Display Data in Graph
- Display Network Stability

Executes whenever possible

Write

Read → **Freq Threshold RoC Threshold** ← Read

Read   Read

Read   Read

Write → **Freq Data[50] RoC Data[50]**

Write → **Time Reaction Data time[5],min,max,avg,total**

freqRocDataQ

Read/Write

timer_exp flag

Write

Freq Analyser ISR

- Calculates frequency from count and clock of FPGA
- Generates a timestamp from the FreeRTOS scheduler
- Puts values in a compound data structure (struct)

Writes to frequencyQ

frequencyQ →

Task4: Frequency Updater
PRIORITY: 5

- Calculates ROC based on timestamp values and frequency
- Puts ROC, frequency and timestamp in a compound data structure (struct)

Executes periodically 20ms
Reads from frequencyQ and writes to freqRocDataQ

FreeRTOS Timer Callback
PRIORITY: 10

Set timer_exp flag on expiry

External Overview of System

Switches
Load Manager Task

Buttons
Load Manager Task

Keyboard
Keyboard Manager Task

Relay System

VGA Monitor
VGA Monitor Task

NIOS II Console
Eclipse IDE

LCD
Keyboard Manager Task

LED
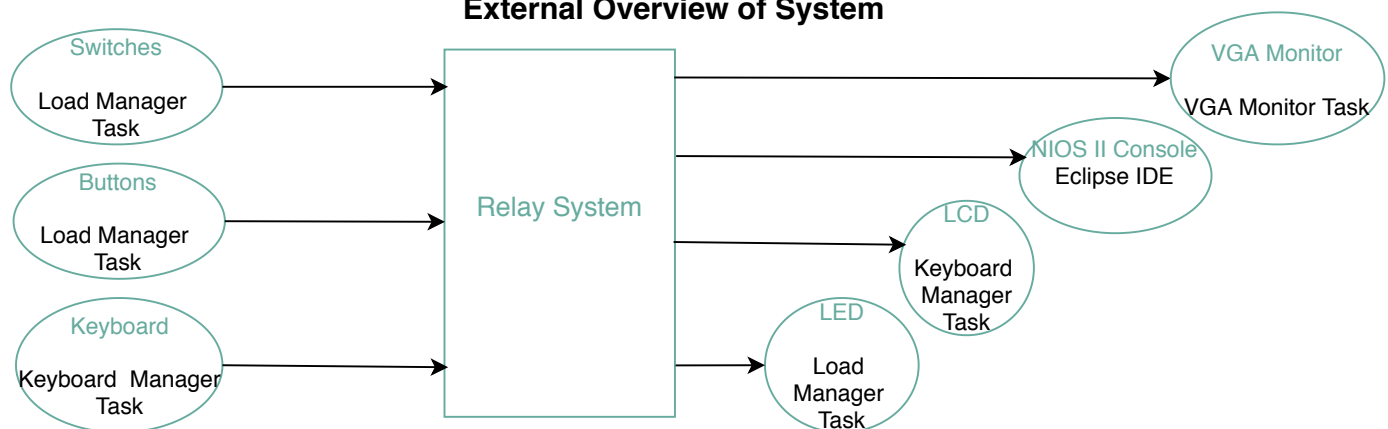Load Manager Task

Design Decisions:

TOTAL HOURS : Approx 50 Hours (Including Dev,Code Review,Documentation & Testing)

Tasks
Priority from lowest to highest: {VGA Monitor,Load Manager,Keyboard Manager ,Frequency Updater, Timer Callback Function}

VGA Monitor is the lowest priority and does not have a fixed period to run (it runs whenever possible) as the task is not crucial to relay functionality.
Load Manager is the FSM of the system. The reason **Keyboard Manager** and **Frequency Updater** are higher priority than the FSM is to make the system more Reactive to inputs from the Keyboard and the ADC. Frequency Updater runs at 50HZ as that is the expected rate at which the ADC generates a value (recorded in **Freq Analyser ISR**). **Load Manager** runs at 100HZ (double) to make sure it handles all new frequency values. There are added delays in **Keyboard Manager** to make sure the user does not provide inputs too quickly (as it can affect controller functionality).

Using Periperals and Components
Keyboard ISR, Freq Analyser ISR and Button ISR are used to access the Keyboard,ADC and Buttons respectively. A Free RTOS Timer is used which executes a Callback Function on expiry. The priority of the Callback function is set to the highest so that any Task cannot interrupt it. Switches are checked manually in **Load Manager** using a local function.**Load Manager** also updates the Red and Green LEDs. **VGA Monitor** writes to the VGA Monitor.

Inter Task Communication
Queue is used for:
- Synchronisation between ISR and Task.
  Ps2KeyValueQ : **Keyboard ISR** => **Keyboard Manager Task**
  frequencyQ : **Freq Analyser ISR** => **Frequency Updater Task**.
- Synchronisation between Tasks
  freqRocDataQ: **Frequency Updater Task** => **Load Manager Task**.

Use of compound types(struct) for messages in Queue:
frequencyQ : Stores Frequencies from the ADC and Timestamp in Compound Type Message.
freqRocDataQ : Stores Frequencies from the ADC, RoC and Timestamp in Compound Type Message.

Global Variables:
FreqData[50] and RocData[50]: Updated by **Load Manager Task(Producer)**. 5 most recent values are read by the **VGA Monitor Task.** There is no mutex protection as the consumer task does not change the state of the program and only displays a value. This is also the case with Time Reaction Data, which are the parameters for the reaction time of the first load shedding.
Freq,RoC Thresholds: Updated by **Keyboard Manager Task (Mutex-Protected)** and read by **Load Manager Task(Mutex-Protected)** and **VGA Monitor Task(Non-Mutex).** A mutex is used in this case.Load manager task is changing the state of the program based on the threshold input so must be mutex guarded read while VGA is not so no need to guard. Additionally, using a mutex protected read for **VGA Monitor Task** means that **Keyboard Manager Task** & **Load Manager Task** are blocked, which are higher priority. This can lead to Priority Inversion.
Timer Flag is updated by both **Load Manager and Timer Callback Function.** However, this case no mutex is required as Mutual Exclusion is guaranteed.**Load Manager** only resets Timer Flag when the FreeRTOS Timer is switched off, hence it will never execute the **Timer Callback Function** at the same time.
Maintenance Mode Enabled is updated by **Button ISR** and read by **Load Manager Task.** Mutex is not being used as the variable is updated inside the ISR (actions inside ISR are not interrupted), therefore Mutual Exclusion is ensured.

ADDITIONAL FEATURE: LCD Prompt
Use of LCD Interface to interact with user when updating thresholds
The User can update Freq/RoC Thresholds from numpad on their PS2 keyboard. The LCD screen displays instructions/prompts to the User. The VGA screen updates the threshold values if typed correctly else a user has to start from the beginning.
Freq example input: 30(30.0 HZ) ; RoC example input: 200 (20.0 Hz/Sec).