COMPSYS 723 Assignment 1 – Frequency Relay

I. User Interface

a. DE2 Board

- Toggle switches 4 to 0: the wall switches for loads 4 to 0 respectively.
- Red LEDs 4 to 0: the current status of connectivity for loads 4 to 0 respectively.
- Green LEDs 4 to 0: the current shed status for loads 4 to 0 respectively.
- Button 3: toggles *Maintenance Mode*.
- Red LED 17: whether *Maintenance Mode* is on or off.
- Button 2: toggles *Precise Increment* - for setting thresholds.

b. Keyboard

- UP/DOWN: Increment/decrement *frequency threshold* by 0.5 or 0.1 Hz.
- LEFT/RIGHT: Increment/decrement *rate of change* (RoC) *threshold* by 1 or 0.2 Hz/s.

Amount dependents on *Precise Increment* flag.

c. VGA Display

- Graph of Frequency and RoC - updated right to left.
- Total run time of the system.
- Frequency and RoC thresholds.
- Reaction times - updated right to left (round table).
- Maximum, minimum, average reaction times.

II. ISRs

a. Frequency Relay

Triggered when a new peak data has been received, calculate instantaneous frequency and push to *frequency data* queue. It also records the current tick stamp and pushes that to *time stamp* queue.

b. Button ISR

Reads the edge capture register, toggle *maintenance mode* or *precise increment* flag then clear edge capture register.

c. Keyboard ISR

Retrieve PS2 keyboard data, decode and push relevant data (ASCII_MAKE_CODE) to *keyboard data* queue.

d. Timer ISR

Set *timer expired* flag, indicating that 500 ms has expired.

III. Tasks

a. Frequency Update Task

Priority: 5 Run condition: when frequency data queue is not empty.

Retrieve the head of the *frequency data* queue and *time stamp* queue, calculate RoC and store them into the *frequency* and *time stamp* arrays at *freq index,* which gets incremented.

b. Keyboard Update Task

Priority: 4 Run condition: when keyboard data queue is not empty.

Get key press and increment/decrement instantaneous frequency and rate of change (RoC) thresholds.

Pops the *keyboard data* queue, checks if it is one of the arrow keys and it is press down instead of release by toggling a *key pressed* flag every time an arrow key has been popped. If it is a press down it will take the *threshold semaphore* protecting the thresholds, increment or decrement by a certain amount then give the semaphore back. The amount that is incremented or decremented depends on *precise increment* flag, toggled by button ISR.

A minor problem with this approach is that if two or more arrows keys are pressed, the thresholds change may update twice or not at all. This can be simply fixed by having individual key pressed flag for each arrow key.

This task has higher priority than Load Manager Task as the manager requires the threshold values to determine whether the system is stable.

c. Load Manager Task

Priority: 3 Run condition: Task Notification from Frequency Relay ISR (new data)

Core functionality of this Frequency Relay, controlled by a FSM (Fig. 2). When a new frequency data has been received, it checks the current state, apply state output and decide on the next state.

If unstable condition is met, it will push a request to shed to *load requests* queue and go to UNSTABLE state, where it starts a 500 ms timer, if the timer expires, it will send another shed request and restart timer. If the system becomes stable before timer expires, it will go to STABLE state and restart the timer.

If the timer expires in the STABLE state then it will send a reconnect request and restart the timer, or if the system becomes unstable, it will go back to the UNSTABLE state. This repeats until it is stable for long enough that all the disconnected loads have been reconnected.

d. Load Control Task

Priority: 2 Run condition: periodic, 100 ms.

Read toggle switches, if *maintenance mode* is on, set load LEDs according to toggle switches. Only the toggle switches 4 to 0 are used. If the Load Manager is in IDLE state, i.e. stable, it clears the *managed LEDs* (green) and set load LEDs according to toggle switches.

When the Load Manager is active (not IDLE), Load Controller will attempt to pop the *load request* queue and disconnect the lowest priority load or reconnect highest priority load. If it's a Disconnect request, it will also get the current system tick count, calculate the reaction time and update min/max/avg reaction times.

Regardless of the Load Manager's state, if any toggle switches have been turned off, the respective load will be immediately turned off, but will not reset the 500 ms timer that Load Manager uses for stable/unstable state transitions.

Then it will check whether the toggle switch values against the current loads using xor, and all connected flag will be set/cleared.

e. VGA Task

Priority: 1 Run condition: periodic, 33 ms (~30 Hz).

Reads the frequency and RoC arrays, plot them onto graphs. Also prints the total run time, current frequency and RoC thresholds, along with 5 most recent reaction times and min/max/avg reaction times.

This task has the lowest priority, and the reading of frequency data arrays are not mutex protected, as it was decided to not be a problem if the user saw one old data point for one frame.

IV. Shared Variables

A total of sixteen global variables are included in this design and are considered as four different types; user-interface, internal system control, frequency related data and reaction time information.

Mutual exclusion protection is required for tasks that may perform incorrectly if operating with incorrect values of shared variables. All shared variables have a single task writing to them, therefore only binary semaphores are used for complete protection. For a more lightweight and robust design, only binary semaphore protection has been allocated if absolutely required.

Shared variables containing information about system control, internally or through user interface such as *maintenance mode*, *precise increment* or the *all connected* flag do not need semaphore protection. This data is used to check conditions for operation and will simply be considered the next time the relevant task is executed, an update during use of the data will not affect functionality.

Shared variables containing frequency related data require semaphore protection. This data is critical in determining system functionality. To ensure that every frequency data packet is processed correctly, a semaphore is used to protect the data from being overwritten.

Shared variables containing reaction time information do not require semaphore protection. It does not contribute to system functionality and will still retain complete functionality regardless of whether it operates on data that was just recently modified.

V. Other Design Decisions

Queues were used to communicate data between some ISRs-task and task-task. They are thread-safe and makes sure data is not missed, unlike storing in global variables.

Task Notify was used to unblock the Load Manager as supposed to a binary semaphore because it is more efficient and faster.

Binary semaphores were used instead of mutex due to tasks only Giving the semaphore if it had taken it, which really has the same functionality as mutex.

VI. Limitations/Future Development

The task tick counter is 32 bits in milliseconds;

$$\frac{2^{32}}{1000 \times 60 \times 60 \times 24} = 49\ days$$

As of now, the hour counter will not overflow after 23, and will restart to 0 after 49 days. This does not impact the core functionality of the Frequency Delay whatsoever, but it can be fixed in future development.

Reaction times to shed a load are typically 1 to 22 ms. Other than the actual decision logic and writing to the load register (red LEDs), there are two things adding additional delay:

1. Load Manager Task runs on the FSM model – state transition is triggered on new frequency from ISR by a Task Notification. The reaction time could be delayed up to one period of the input frequency (~20 ms at ~50 Hz) when timer expires after Load Manager task has ran, and the delay gets worse when the frequency drops.
2. Periodicity of Load Control Task (up to 10 ms) – the task that actually sheds the load.

Those additional delays can be cut significantly if Manager Task becomes periodic, say 10 ms period, which would reduce the reaction time by up to 10 ms. This change would also make the system more time predictable due to this periodicity, as it no longer relying on an external event to trigger.