FreeRTOS Lab Checkpoint

Prof. Li-Pin Chang
NYCU ESSLab

FreeRTOS Lab

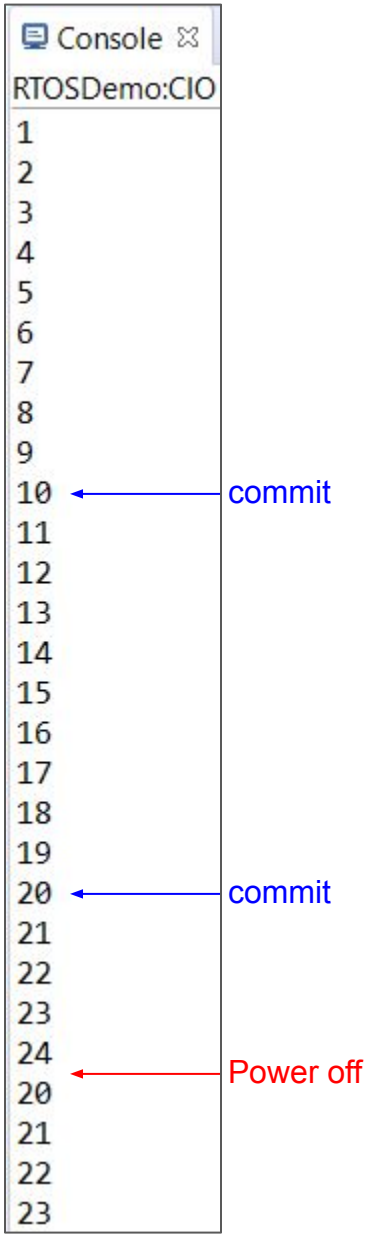
- This is Lab 5, Implement checkpoint for FreeRTOS
- Your program would be like:

```
main()
{
        prvSetupHardware() // re-initialize I/O devices
        restore();       // restore a checkpoint if power fail

        // add test task & enable FreeRTOS scheduler
        …
}

test_task()
{
        while(i++)
        {
                if(random_condition()) power_off();
                if(i %10 == 0) commit();
                print(i);
        }
}
```

output



Checkpoint Commit

- Reserve a backup space in FRAM
- Check memory mapping using .cmd and .map
- To commit a checkpoint, perform backup of the following (in order)
  - ucHeap
  - SRAM
  - CPU registers

Checkpoint Restoration

- If random_condition() is true, goto LPM4.5
- Push the reset button on the board

- Upon powering up, check if we are recovering from a power fail
  - No restoration on first start
- If so, restore the following (in order)
  - ucHeap
  - SRAM
  - CPU registers
- There should be an index pointing to which one of the two backup copies is valid

Remarks

- Use rand() to trigger power fails, but to avoid *stagnation*, power fail intervals cannot always be shorter than checkpoint commit intervals
- In you code, you must actually backup SRAM (.bss & .data), ucHeap, and CPU registers on checkpoint commit and actually restore these on checkpoint restoration
  - Failing to backup or restore any of these will receive a score penalty
- LPM4.5 is for the convenient of testing, but your program must also survive power source removal
  - This will also be tested in your demo