FSM Design Patterns

Lab 6, ESDM

Objective

Using the Stateflow tool in Simulink to model to implement simple design requirements which are very often encountered in practice.

Theoretical aspects

TBD. See the Lectures.

In this lab you will implement:

- Signal debouncing
  - One-sided
  - Two-sided
- Minimum Timer

Exercises

1. Design a FSM in Stateflow with one input `UserCommand` and one output `MotorCommand` for the following requirements:
   1. The motor shall be started (`MotorCommand = TRUE`) as soon as the input `UserCommand` becomes TRUE
   2. The motor shall be stopped (`MotorCommand = FALSE`) when the input `UserCommand` is FALSE for a duration of at least `CP_DbounceOffTime` (default value = 1 second).
2. Test your design: put appropriate inputs and observe the output signals.
3. Design a FSM in Stateflow with one input `Voltage` and one output `OvervoltageError` for the following requirements:

1. The error flag `OvervoltageError` shall be set when input `Voltage` exceeds `CP_MaxVoltage` for at least `CP_DebounceOnTime`
2. The error flag `OvervoltageError` shall be cleared when input `Voltage` is below `CP_MaxVoltage` for at least `CP_DebounceOffTime`

4. Test your design: put appropriate inputs and observe the output signals.

5. How would you add **hysteresis** to the previous block?

6. Design a FSM in Stateflow with one input `UserCommand` and one output `ActivateHighBeam` for the following requirements:

    1. The High Beam shall be started (`ActivateHighBeam = TRUE`) as soon as the input `UserCommand` becomes TRUE, if they were stopped for a duration of at least `CP_MinimumOffDelay` until the current moment.
    2. The High Beam shall be stopped (`MotorCommand = FALSE`) as soon as the input `UserCommand` is FALSE
    3. When the High Beam is stopped, no activation is allowed for at least `CP_MinimumOffDelay` afterwards.

7. Test your design: put appropriate inputs and observe the output signals.

8. Design a FSM in Stateflow with two inputs `MotorOn` and `LatchReached` and one output `LiftgateClosed`, for the following requirements:

    1. The liftgate shall be considered open (`LiftgateClosed = FALSE`) always when `MotorOn = TRUE`.
    2. The liftgate shall be considered closed (`LiftgateClosed = TRUE`) when `MotorOn = FALSE`, if the input `LatchReached` becomes `TRUE` within `CP_MaxLatchDelay` after `MotorOn` has become FALSE.
    3. If the input `LatchReached` becomes `TRUE`, but the motor was not started anytime within `CP_MaxLatchDelay` prior to this moment, it shall be ignored and the liftgate shall be considered open.

9. Test your design: put appropriate inputs and observe the output signals.