Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

**COURSE:** CEG4166/CSI4141 Real Time Systems Design
**PROFESSOR:** Gilbert Arbez
**SEMESTER:** Practice Exam
**DATE:** **Practice**
**TIME:** **Practice**

# PRACTICE FINAL EXAMINATION

Solution

**Name and Student Number:** _____/_____

There are 3 parts in this examination.

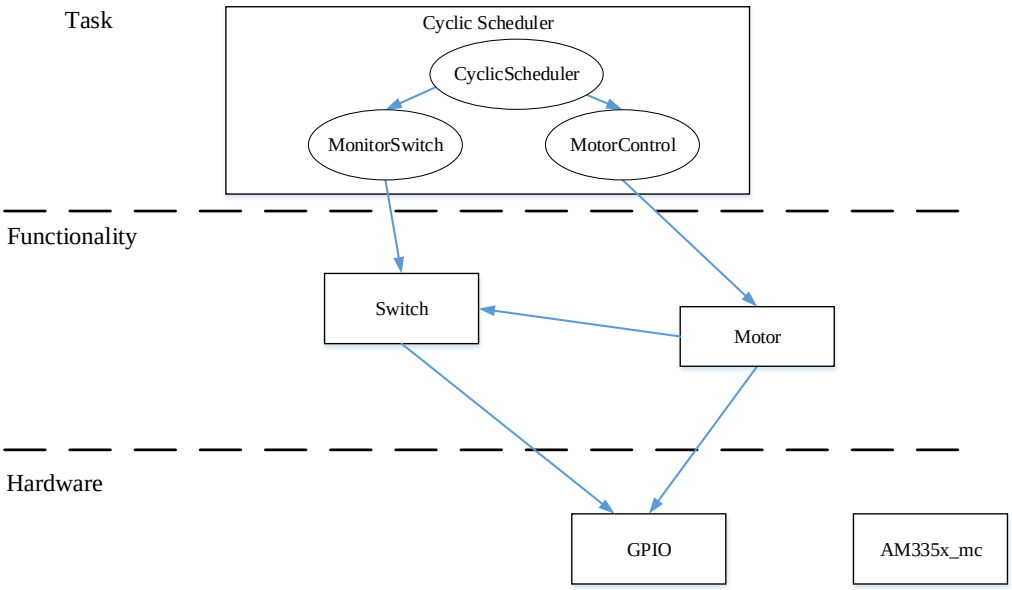
| | | | |
|---|---|---|---|
| Question 1 | | | |
| Question 2 | | | |
| Question 3 | | | |
| Total | | | |

Answer questions in the provided examination booklets.

This is an open book exam.

Total number of pages: 4

## Question 1 – Software Modules




Software Modules:

- CyclicScheduler Module
  - Implements a cyclic scheduler for running two tasks.
  - MonitorSwitch task monitors the switch (including debouncing).
  - MotorControl turns the motor ON or OFF according to the state of the switch.
- Switch module: contains all the functionality for monitoring the switch including debouncing the switch during its transition from on to off and from off to on. The module also tracks the state of the switch (either ON or OFF).
- Motor Module: Simple module that turns the motor on and off. It consists of wrapper functions that make the appropriate calls to the GPIO module.
- GPIO Module: Provides the functionality that configures and controls the GPIO bits connected to the switch and motor control hardware.

Given Modules:

- A335x_mc Module: for enabling the microcontroller peripherals, in particular the GPIO peripheral.

Tasks

| Task | Thread Type | Invoked by | Implemented in Module | Other related Modules |
|---|---|---|---|---|
| MonitorSwitch Task | Clock | CyclicScheduler | CyclicScheduler | Switch, GPIO |
| MotorControl | Clock | CyclicScheduler | CyclicScheduler | Motor, Switch, GPIO |

Tasks

1) MonitorSwitch: This task consists monitoring the switch position, including deboucing the switch. The Switch module contains the logic for debouncing the switch and determining its position.
2) MotorControl: This is a simple task that turns on and off the switch according to the value returned by a call to the Switch module to determine the position of the switch.

## Question 2 – Cyclic Scheduler Design

Background/Algorithm

- Quick introduction to cyclic scheduler (not provided here).

Design of Cyclic Scheduler

- Two tasks are required:
  - Task1 (T1) – MonitorSwitch: Given that the debouncing time of the switch is 50 ms, the period of this task is set to 50 ms. This value will simplify the FSM implemented in the Switch module for debouncing he switch. Execution time will be on the order of 100's of microseconds, thus 200 microseconds would be a conservative value for maximum time of execution.
  - Task 2 (T2) – MotorControl: Turns the motor on/off according to the position of the switch. Considering that it will take a multiple of 50 ms time periods for task T1 to update the switch position, the period of this tasks should be >100ms (at least 2 state transitions are required to debounce the switch). The value of the period will reflect response time, so 500 ms (1/2 second) is a reasonable maximum response time. Execution time will be on the same order (and shorter) than task T1.

| Task | $e_i$ | $p_i$ |
|---|---|---|
| MonitorSwitch (T1) | 200 micro-sec. | 50 ms |
| MotorControl (T2) | 200 micro-sec. | 100 ms < $p_2$ <500 ms |

- Major and Minor Cycles.
  - The period for T2 can be selected so that it is a multiple of the period for T1. This will simplify considerably the design of the cyclic scheduler.
  - The MotorControl task need only be executed as fast as the switch changes value. We have assumed that two state transitions are required to debounce the switch, and thus we set the period for T2 to 100 ms. The task table above becomes:

| Task | $e_i$ | $p_i$ |
|---|---|---|
| MonitorSwitch (T1) | 200 micro-sec. | 50 ms |
| MotorControl (T2) | 200 micro-sec. | 100 ms |

  - Thus the major cycle time selected will be 100 ms.
  - The minor cycle time chosen will be 25 ms. It meets our three constraints for the cyclic scheduler
    - > max $e_i$ , that is > 200 micro-seconds
    - Is a factor of the major cycle time.
    - Third constraint
      - Task T1: 2f – gcd(50, 25) = 50 – 25 = 25 < 50
      - Task T2: 2f – gcd(100, 25) = 50 – 24 = 25 < 100
  - The following shows a schedule with a minor cycle of 25 ms and a major cycle of 100 ms.

| | 50 ms | | 50 ms | |
|---|---|---|---|---|
| | **T1** | | T1 | **T2** |
| Frame Number | 0 | 1 | 2 | 3 |

Design of software module

- The following data structures are used for implementing the cyclic scheduler
  - Given the symmetry of the schedule, the cyclicScheduler function can be implemented with a simply static integer variable, frameNum as follows:
    - frameNum modulo 2 = 0: run T1
    - frameNum = 3: run T2
- MonitorSwitch Task Design
  - Implemented in the function "monitorSwitch()". Calls the debounceSwitch() function in the Switch module that implements an FSM for debouncing the switch.
- MonitorSwitch Task Design
  - Simply calls the function motorOn() when getSwitchState() (from the Switch module) returns ON and motorOff() when the getSwtichState() function returns OFF. The motorOn/motorOff functions are provided in the Motor module.

## Question 3 – Software Module: Switch Module

This module provides the functionality to support the MonitorSwitch task. It contains the logic for debouncing the switch and maintaining its state. The following is a summary of the principle API functions and global variables important to achieve this functionality.

Global variables:

switchState: an integer variable set to ON when the switch is in the on position and OFF when in the off position.

API functions:

debounceSwitch: function that debounces the switch using the isBitHigh() function from the GPIO module to determine the state of the line. The FSM shall be implemented using a switch statement. The FSM changes the value of the switchState variable.

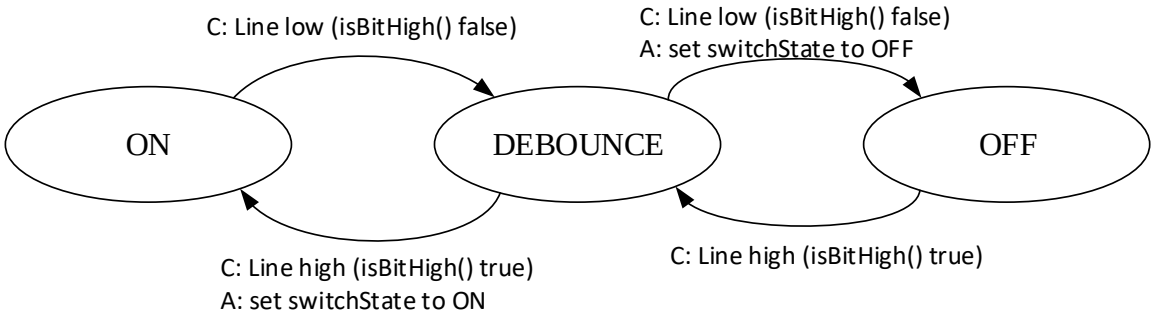
getSwitchState: gives the state of the switch by returning the value of the variable "switchState".

### Debouncing the switch:

The function "debounceSwitch" shall provide this functionality and should be executed every 50 ms (see details of Cyclic Scheduler design) which corresponds to the debounce time of the switch. Thus an FSM can be developed without any timing concerns, since 50 ms delay is present between each state transition. Three states are required, ON, OFF and DEBOUNCE. The third state DEBOUNCE indicates that a change in switch signal has occurred. If only noise is detected, then the FSM returns to its previous state (remains in OFF or in ON state), otherwise, it will move to the new state (from ON to OFF or OFF to ON).

The GPIO module function isBitHigh(SWITCH_BIT) is used to determine if the input line from the switch is high (function returns true) or low (function returns false).

The following FSM diagram




Note that the DEBOUNCE state is used for debouncing from either ON to OFF (turning the switch OFF) state or OFF to ON state (turning the switch ON).