The University of Queensland
School of Information Technology and Electrical Engineering
Semester 2, 2020

CSSE2010 / CSSE7201

AVR PROGRAMMING TASK

Due: **12noon Saturday 31 October 2020**
Weighting: **20% (30 marks)**

Objective

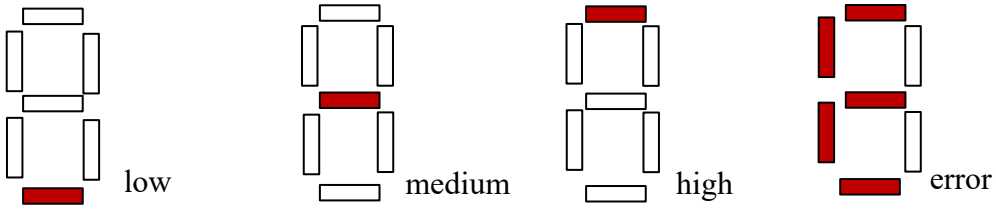
As part of the assessment for this course, you are required to undertake an AVR programming task which will test you against some of the practical learning objectives of the course. In particular, the programming task will test your understanding of the practical material covered during the learning labs of this course. Specifically, you are asked to implement a system to mimic the operation of a washing machine controller as described below, which uses various aspects of Atmega324a microcontroller. The basic building blocks required to implement the given system are already covered in exercises in learning labs 10-17 (for which answers have been provided as well) and you are required to use these building blocks to implement and test the given system for the required functionality. You can use any of the lab codes as a starting point to build upon, hence no code is provided to you.

Task Description

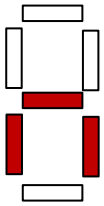
You are required to implement and test a system which mimics the operation of a simple washing machine controller using the Atmega324a microcontroller and the I/O board. Inputs and outputs of the system and the functionality are described below.

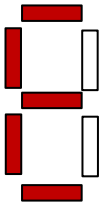
- The washing machine supports two modes of operations, each containing "wash", "rinse", and "spin" cycles:
  - Normal mode: wash→rinse→spin
  - Extended mode: wash→rinse→rinse→spin
- For both wash and rinse cycles, the system supports three water levels as "low", "medium", and "high"
- Water level is selected by the switches S0 and S1 on the IO board as
  - S0=0, S1=0 → water level "low"
  - S0=0, S1=1 → water level "medium"
  - S0=1, S1=0 → water level "high"
  - S0=1, S1=1 → this option is not considered and treated as "error"

  and should be displayed on the **right digit** of the seven-segment display (SSD) as,



- The operational mode (normal/extended) is set by the switch S2 (S2=0 for normal and S2=1 for extended) on the IO board and should be displayed on the **left SSD** digit as


normal


extended

- Both the mode and water level should be first set by the switches S0-S2 and the system starts its operation upon **press and release** of button B0 on the IO board. Prior to pressing and releasing of button B0, the system in its 'initial' state in which all the LEDs are OFF and the two SSDs displays indicate the operational mode and water level. If invalid water level is selected (i.e. S0=1, S1=1) the system should not start (i.e. remains in initial state) and indicate error **on the right SSD** by displaying "E". Once the system starts (upon the press and release of B0), the individual cycles within the selected mode are mimicked using a blinking pattern on LEDs L0, L1, L2, L3 together with PWM-driven intensity variation on LED L7 as described below.

Cycle	Blinking pattern on LEDs L0-L3	PWM controlled intensity on LED L7
Wash	One-directional running pattern from L0 to L3 (i.e. from right to left) for 3 seconds followed by all four LEDs on for 3 seconds	Low-intensity (10% PWM duty cycle)
Rinse	One-directional running pattern from L3 to L0 (i.e. from left to right) for 3 seconds followed by all four LEDs blinking ON and OFF at a reasonable and visible rate of your choice for 3 seconds	Medium-intensity (50% PWM duty cycle)
Spin	Bi-directional running pattern from L3 to L0 and then from L0 to L3 (i.e. from left to right and right to left) for 3 seconds followed by all four LEDs blinking ON and OFF at twice the rate in rinse cycle for 3 seconds (the blinking ON and OFF should be visible)	High-intensity (90% PWM duty cycle)

- After the cycles are completed for the particular mode selected, the system goes to the 'finished' state which is indicated by all LEDs L0-L3 and L7 are OFF and digit "0" displayed on both SSDs. During the cycle operation, for the PWM-driven output L7, steady visual output should be seen without flicker. Also, there should not be any ghosting effect on SSD outputs.
- While on the finished state, the system can be restarted by pressing and releasing button B0 again, and this will start a new operation. During the operational state (i.e. while the wash, rinse, and spin cycles are in operation) the system **should not respond to button B0**.
- Button B1 is used as a reset input, to reset the system to its initial state at any time **upon press and release of B1**.
- You can make any other assumptions in addition to the basic operation outlined above, and such assumptions should be clearly mentioned in your submission (on the marking sheet).
- You must use the following pin assignments for the Atmega324a microcontroller so that the circuit connections are consistent as required for marking.
    - SSD segments A-G: PA0-PA6 (i.e. PA0→Seg A, PA1→Seg B and so on)
    - SSD digit select (CC): PA7
    - Switches S0, S1, S2: PD0→ S0, PD1→S1, PD4→S2
    - Buttons B0, B1: PD2→ B0, PD3→B1 (allowing interrupt-driven inputs, if needed)
    - LEDs L0-L3: PB0→L0, PB1→L1, PB2→L2, PB3→L3
    - LED L7: PB4→L7 (allowing PWM output)

  In case you've encountered any hardware defects that prevents you from using the above pin assignments, you can use a different pin assignment, and this must be clearly stated in your submission.
- You may select either polling or interrupts to interface with the external inputs

Submission

You need to electronically submit one zip file (i.e. a zipped folder) containing the following items by **12pm October 31st, 2020**, via the course Blackboard site.

- **All** of the C source files (.c and .h) necessary to build the project
- Your final .hex file[1] (suitable for downloading to the ATmega324A AVR microcontroller program memory); and
- A PDF of the completed marking sheet provided to you

Do not submit .rar or other archive formats – the single file you submit must be a zip format file. All files must be at the top level within the zip file – do not use folders/directories or other zip/rar files inside the zip file. If you make more than one submission, each submission must be complete – the single zip file must contain the marking sheet and the hex file and all source files needed to build your work. We will only mark your last submission and we will consider your submission time (for late penalty purposes) to be the time of submission of your last submission.

NOTE for external mode students residing overseas and haven't received a kit: If you are an external mode student and haven't received a kit, you should still attempt the task by coding and testing the individual sub-systems of the main system using the Atmel Studio Simulator. You should be able to compile your code with no errors and warnings and then inspect individual register bits and memory content within the Atmel Studio Simulator to ensure the configurations you intend to make are actually in place. Further, you must indicate in your marking sheet that you have not received a kit and the marking process will take this into account and will mark you based on your code and simulation outputs only. This exception is only applicable to overseas external mode students who haven't received a kit due to difficulties associated with postal services etc.

Incomplete or Invalid Code

If your submission does not compile and/or link in Atmel Studio 7, then the marker will make reasonable attempts to get your code to compile and link by fixing a small number of simple syntax errors and/or commenting out code which does not compile. **A penalty of between 10% and 50% of your mark will apply depending on the number of corrections required.** If it is not possible for the marker to get your submission to compile and/or link by these methods, then you will receive 0 for the task (and will have to resubmit if you wish to have a chance of passing the course). A minimum 10% penalty will apply, even if only one character needs to be fixed.

Compilation Warnings

If there are compilation warnings when building your code (in Atmel Studio 7, with default compiler warning options) then a mark deduction will apply – **1 mark penalty per warning up to a maximum of 5 marks.** To check for warnings, rebuild ALL of your source code (choose "Rebuild Solution" from the "Build" menu in Atmel Studio) and check for warnings in the "Error List" tab.

Late submission will result in a penalty of 10% plus 10% per calendar day or part thereof, i.e. a submission less than one day late (i.e. submitted by 12noon November 1st 2020) will be penalised 20%, less than two days late 30% and so on. (The penalty is a percentage of the mark you earn (after any of the other penalties described above), not of the total available marks.) Requests for extensions should be made via the process described in the course profile (before the due date) and be accompanied by appropriate documentary evidence of extenuating circumstances (e.g. personal statement or medical certificate). The application of any late penalty will be based on your latest submission time.

Notification of Results

Students will be notified of their results via Blackboard's "My Grades" when marking is complete.

[1] The .hex file can be found in the "Debug" folder within your Atmel Studio project.

The University of Queensland – School of Information Technology and Electrical Engineering
Semester 2, 2020 – CSSE2010 / CSSE7201 AVR Programming Task Marking Sheet
(A separate PDF version of this form is also available on Blackboard and must be included in your submission)

Student Number								Family Name	Given Names

Complete the table below if you use a different pin assignment than what is specified in the task due to any hardware faults in your kit.

Port	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2	Pin 1	Pin 0
A								
B								
C								
D								

Are you an external mode student who hasn't received a kit? YES/NO

State any assumptions you make

Marking Rubric

Item	Comment (anything you want the marker to know)	Marks (to be completed by the marker)
Correctly implemented operational modes with SSD indication		/4
Correctly implemented water level with SSD indication		/4
Correctly implemented cycles with LED blinking patterns		/12
Correctly implemented PWM output		/4
Overall system functionality		/3
Clarity and modularity of code		/3

Total marks (out of 30, without any penalties):
Total marks (with any penalties as applicable including late submissions):
Marker's initials: