EEET2505- Introduction to Embedded Systems

Assignment – Digital Clock with AVR Microcontroller

Objectives

By the end of this assignment, you will be able to:

- Analyse requirements and go through a design process of an embedded system, both in software and hardware using AVR microcontroller, including:
  - General Purpose Input/output (GPIO)
  - Timers/Counters
  - Interrupts - Timer Interrupt and External Interrupt
- Integrate external pre-developed libraries to an embedded system.
- Get proficient in using an instrument such as Oscilloscope or Virtual Instrument to assist embedded system development.
- Demonstrate and discuss the design and operation of a complete system.

Expectations and Notes

- You are expected to build the supporting hardware onto the breadboards before writing and testing the code on that hardware. All the components for this Assignment will be given to each team.
- If you need to borrow other components and instruments, email me and cc the technician prior collecting. Note that Arduino Kit and Instruments are for borrowing within weekdays. You are responsible to arrange a time to borrow, return and keep these safely.

Requirements

Each team needs to design two versions of the clock as described below:

Version 1 - Basic Clock (60%)

In this version 1, you will design a basic clock having the following functions:

- The clock can count real time, and display hours and minutes in 24-hour format (seconds will not be shown).
- The time will be displayed on a 4-digit 7-segment LEDs via GPIOs of the AVR Controller.

- The clock has an alarm that goes off after some pre-set time. This pre-set time can be fixed in the embedded code (for example 2-5 minutes).

- The clock can be controlled and configured using external buttons.

To use the clock in this version, the process is as follows:

a) Before it runs, we have to set the initial time (this is the **Set mode**). We can either press a button to get to this mode or we can set this as our default mode right after reset or power-up. To set the time, we need:
   a. Buttons to choose different digits of the 7-segment for minute and hour settings.
   b. Other buttons to increase and decrease the value of digits.

b) To start count time, we need to press another button to change to **Time mode**. In this mode, the clock simply counts and display time starting from the Initial time on 7-segment.

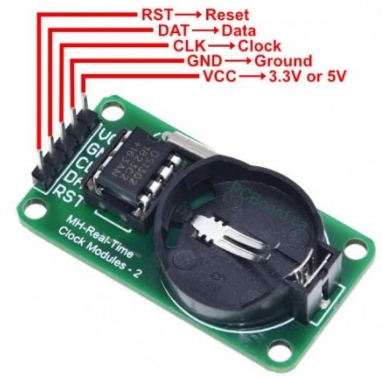
c) During the Time mode, as time elapses after the pre-set time, an LED/buzzer will blink/sound for 5 seconds to indicate this (you can choose the appropriate frequency for users to observe).

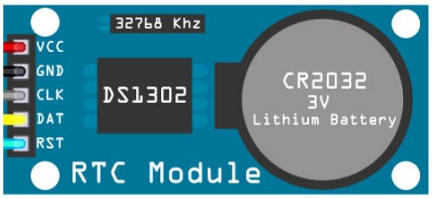
Other requirements of this version 1:

- The entire system is constructed properly on breadboard.
- Buttons should be debounced properly.
- You can use delay C functions (such as delay_ms) for other purposes however you must use Timer to keep the time for the clock.
- Timer for this clock should be precise – You should measure the time of this clock using Virtual Bench and maybe compare with another clock after some hours (or after a day) and see how they are compared.
- Pay attention to the pin assignment – since we got many peripherals now and your MCU has limited number of pins, you might need to plan your pin allocations wisely and come up with control strategy (for example for 7-segment).
- Optional – an indicator such as LED to tell what mode the clock is in.
- Optional – some button or mechanism to turn on and turn off Alarm when we want.

Version 2 - Clock with Time Keeper (40%)

In the version 1 earlier, the initial time is set by the user and then the clock will keep this time. But when the power is lost or the user resets, the initial time needs to be re-configured. In this version 2, you will extend the previous version to have the MCU to work with a Real Time Clock module, which include the DS1302 Time Keeper Chip (https://datasheets.maximintegrated.com/en/ds/DS1302.pdf). The module will ensure that you can keep track of the current time always (as long as the Timer Module Clock runs).





The Time Clock Module use a **simple 3-wire serial interface** (i.e. data are transferred in series in a bit steam). Your task in this version is communicate between your MCU board and the Module to:

- **Write** data from the MCU into the module – this is used to write the initial time to the chip (*unlike the version 1, you are supposed to do this once)*. After that this module will keep the time.
- **Read** data from the module – this is used to read the initial time from the module before you run the Clock.

To use the clock in this version, the process is:

- After the reset or power up, the Clock is in **the Idle mode**
- When a button is pressed, the Clock is change to **Set Time Keeper mode:**

For this mode, you will use the MCU to write the initial time to the DS1302 module. You need to study the interface of the module and then control the writing properly. You can decide what peripherals (i.e. buttons, how many of them) to assist the configuration.

- When another button is pressed, the Clock will read the time stored from the Time Module (**Reading Time and Run** mode). The Clock then start display time from the Initial time onwards.

Other requirements for this Version 2:

- We don't need the Alarm function.
- We will replace the 7-segment LED by a two-line LCD. The LCD will be used to display time as well as other information. You are allowed to use pre-developed C library to control LCD in this project.

- You need to **write your own control functions for write and read with the Timer module DSC1302**. You are not allowed to use external library to work with this module. You can use Virtual bench or OSC to support the development and troubleshooting processes.
- It's recommended that you will write separate functions in C to partition the design and troubleshoot the design.
- You can use other design techniques such as FSM or anything that you think are appropriate.

Submission

Each team is required to submit the following:

a. One report to discuss relevant assumptions or any information related to the design. Also, this is where you will be presenting your research and discussions for the above tasks. Don't forget the references. *A report template will be provided to assist you with this*
b. Folders to store Atmel studio projects. These projects should be ready to re-run (*I will verify your work when I mark)*.

Finally, you can upload all of these into a OneDrive folder, name it **EEET2505_Assignment_Team_Number_XXX**, and then share the access with me and your teammates.

The link to the folder MUST BE PASTED to the Canvas

Assessment

- Students will be marked base on their work, their report and demonstration with the lecturer.
- During the demonstration, students need to demonstrate their systems and prepared to answer some questions.
- Peer assessments and extra questions will be used to assess contributions of each student.

Marking Rubric is found on Canvas