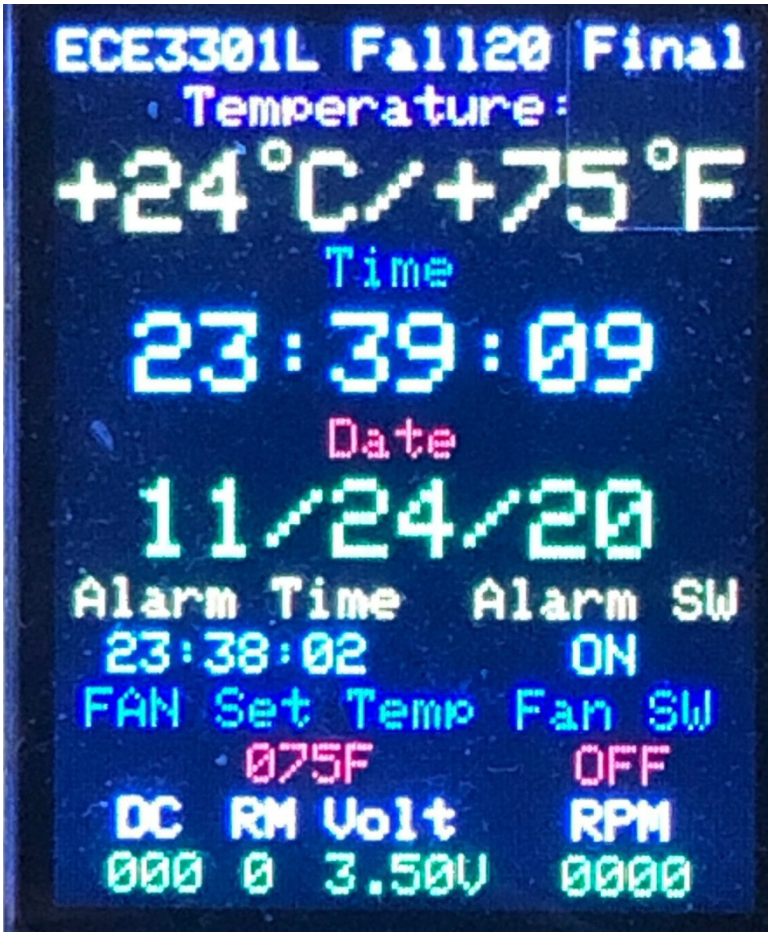
Final Project

The final project will integrate a design that will have the following functions:

- A TFT panel used as a main display.
- An ambient temperature in degree C and F is displayed on the screen
- A digital clock shows the time and date
- A fan support with full function control – On/Off and speed
- An indicator of the duty cycle for the fan control
- An alarm function is available to activate a multicolor LED when the alarm set time is reached
- Push-button switches and DIP switches are used to select three setup screens to change the setting for time/date, alarm time and programmable level of duty cycle of the fan
- Two DIP switches are used to enable the Alarm and Fan functions

The hardware schematics is provided on a separate pdf file.

Here is the screenshot of the main screen:

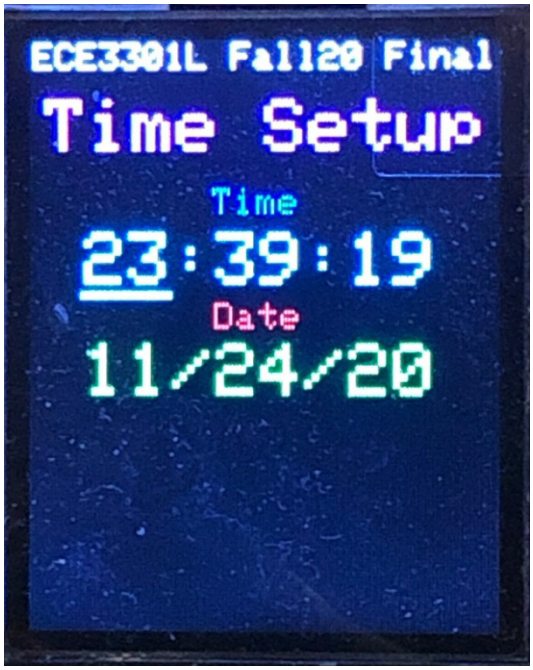


Here are the descriptions of the fields:

- ‘26C/78F’ : Actual Temperature display in degree C and degree F
- ’13:49:44’ : Time display
- ‘03/31/20’ : Date display
- ‘Alarm Time’ : Time set for the Alarm
- ‘Alarm Sw’ : Switch to turn On or Off the Alarm
- ‘Fan Set Temp’ : Temperature set to control the speed of the fan
- ON or OFF depending on the Alarm DIP switch
- ‘Fan Sw’ : Switch to turn ON or Off the Fan
- ‘DC’: Actual level of Duty Cycle
- ‘RM’ : RTC Match status used for Alarm
- ‘Volt’: Actual Readout of the voltage of the light sensor
- ‘RPM’: Actual RPM of the fan
-

Three setup screens are available to allow changes to the system variables:

1) Time Setup Screen:



2) Alarm Setup Screen:

3) Fan Temp Setup:

A) Setup Operations

Three ‘Setup’ switches are used to select one of three different setup screens. The switch ‘Setup_Mode’ should be in logic ‘0’ to force the system to be in normal mode. When it is set logic ‘1’, then the system is switching to setup mode. The other two switches ‘Setup_Sel0’ and ‘Setup_Sel1’ will indicate which setup mode to be in.

Setup_Sel1	Setup_Sel0	Setup Mode
0	0	Time Setup Mode
0	1	Alarm Setup Mode
1	x	Fan Temperature Setup

Time Setup Mode:

The screen will show the first line as the time and the second line as the date. The time line has three fields hour:minute:second while the date line shows the fields month/day/year. A total of 6 fields can be updated. The INT2 push-button is used to go from one field to the next and it does wrap around to the first field when it is at the last field. A cursor will show what field is being selected. Under a given field, the INT0

switch can be used to increase its value. Depending of the type of the field, the value in that field is limited to the highest value allowed by that field. The INT1 switch is used to decrease the value of the field.

When all the fields are properly updated, the Setup Mode switch can be switched back to the logic 0 to go back to the normal mode. By this action, the new time will be updated to the DS3231 RTC chip and it will take the new time/date.

Alarm Setup Mode:

This mode is similar to the Time setup mode but it only changes three time fields for the alarm function. It affects the three time registers of the alarm function of the DS3231. The three INT0, INT1, and INT2 push-button switches act the same as in the time setup mode.

Setup Fan Temperature Mode:

This mode is used to program the temperature level that will trigger the fan to be turned on when the ambient temperature is below that level. The difference between that level temperature and the ambient temperature will be multiplied by 2 and be used as the duty cycle to control the fan's speed. Any temperature above that level will force a 0 duty cycle.. Since there is only one field, only the two INT0 and INT1 push-button switches are used to increase/decrease the temperature level. The set temperature cannot go lower than 50F and cannot go higher than 110F. The button INT2 is ignored.

B) Standard Operations:

1) Fan Operation

   When not in Setup mode, the Fan Sw can be switched to the 'OFF' value by pressing the 'INT0' switch and to the 'ON' value by pushing on the 'INT1' switch.
   When 'OFF', the duty cycle of the fan must be programmed to the zero value and the 'FANEN' LED must be turned off. The 'DC' on the screen must show 000 while 'RPM' also indicates a 0000 value. When 'ON', the FANEN LED will be turned on and the ambient temperature is compared against the 'FAN Set Temp' value. If the ambient temperature is greater, then the fan should be treated as in the 'OFF' condition. If lower, take the difference between those two temperature and multiply it by 2 and use it as the duty cycle. Apply that duty cycle to the fan. Measure the RPM of the fan and show it on the display.

2) Alarm Operation

   The Alarm can be enabled or disabled by toggling the 'INT2' switch. The status of the Alarm operation can be seen under the 'Alarm SW' with the display of either 'OFF' or 'ON'. When enabled, the time set under the 'Alarm Time' will be programmed into the DS3231 chip and the Alarm interrupt operation must be enabled

in the DS3231 so that the signal ‘RTC_ALARM#’ will be set to the logic 0 when the alarm time matches with the actual time. When that event happens, the buzzer will be activated and the RGB LED will change to a different color every second from No color, RED, GREEN, YELLOW, BLUE, PURPLE, CYAN and WHITE. The value under ‘RM’ should show the value of 1 when the time matches. The buzzer and the RGB LED will stay active until the Alarm is turned off or when the light sensor is blocked momentary in order to clear the match. This later event will not disable the Alarm.

Guidance:

1) A base sample of the code is provided to the student to allow shorter development time for the project.

2) The provided code is actually broken down into several files separated into two groups: the ‘.c’ files and the ‘.h’ files. The ‘.c’ files are naturally the c code files while the ‘.h’ files are called the header files. These last ones usually contain the prototype functions of each corresponding ‘.c’ file. Since we will no longer have all the codes placed into a big and long file, we will now have separate sources files. Each source file is designed to contain codes that are specifically dedicated for certain tasks. By grouping all the functions that very related to each other, it would make the structure of the code very simple to follow. In addition, by breaking the long source code, it would be easier and faster to go to a specific function without scrolling the original long source code.

   Here is the list of those files:

   - Main.c : this is where the main program resides
   - Main_Screen.c : this is the file that will generate the main screen on the LCD
   - Interrupt.c : this is where the interrupt handler and initialization code reside
   - I2C_Soft.c : this is the original library file for the basic functions to handle the I2C protocol.
   - I2C_Support.c : this is all the functions needed to interface to the DS1621 and DS3231 ICs
   - TFT_ST7735.c : this is the original library file to support the LCD
   - Setup_Time.c : this is where all the functions to support the setup of the time
   - Setup_Alarm_Time.c : all the functions to support the setup of the alarm time are handled in this file
   - Setup_Fan_Temp.c : the functions to allow the setup of the temperature for the fan are in this file

3) It is important for each group to study of the organization of this project and to understand the way each file is linked to the other files via the use of header files and the use of 'extern'.

4) On the previous lab, it is noticed that from time to time the LCD does show that the second in the time field gets skipped. This is caused by the fact that in the 'get_RPM()' function a call to the 'Wait_Half_Sec()' function was used to measure the RPM. Waiting 500 msec can in the long run force a skip of a second when a polling for the change of the second. To fix this issue, we will use the Timer 0 in interrupt mode. We still use Timer 1 to count the number of Tach pulses from the fan. We will now program the Timer 0 to generate an interrupt every 500 msec. When the interrupt occurs, the content of the Timer 1 will be read and store in the variable 'tach_cnt'. The Timer 1 is then cleared to 0 to allow another count operation while the Timer 0 is reloaded with the count to wait again for another 500msec. The whole cycle is then repeated. The RPM can be simply calculated based on the value of 'tach_cnt' which is in fact equivalent to the RPS value.

   Do fix the content in the project to make sure that all the errors are fixed and run the new code. Observe that the 'SEC_LED' does blink for 500 msec ON and 500 msec OFF.

5) The main task of this project is to handle two functions: FAN and ALARM.

   The FAN function can be turned ON or OFF when pressed on INT0 and INT1. The 'Fan Sw' on the LCD screen will indicate the status. As indicated in the requirements of this function, implement the appropriate code to control the fan's based on the temperature differential between the actual temperature and the set fan temperature. Use the FAN temperature setup page to adjust the set temperature.

   The ALARM function will use the Alarm Setup Time page to allow the user to set a desired time for the alarm. When the Alarm switch is activated (by pushing on INT2 on the main page), the unit will program the RTC chip to compare between the actual time and the alarm time. When the time matches, the signal RTC_ALARM# will be set to 0. The buzzer will be activated and a RGB LED will be changing its color every second. To reset the alarm, either switches off the ALARM SW or shut off the light to the photo sensor.

   The routine 'Test_Alarm()' in main.c program is the place to implement this function. Follow the basic steps provided in that routine.

6) The full code for the Time_Setup() routine is provided as an example. The code will use the INT2 to move a cursor to six fields in the setup and the cursor will wrap around when it reaches either end of those fields. Then, the INT0 and INT1 switches are use to increment or to decrement the value in each field. Use this code to implement the other two setup routines.

7) The file ‘Main.h’ contains all the assignments for the pin signals. Modify them based on the schematics before testing the board.