Digital Alarm Clock

Cole Costa

Summer 2023

Behavior Description

The digital alarm clock designed in this experiment utilizes a Nucleo-L476RG board [1-2], a DS3231 Real Time Clock Module [3], a buzzer, a 4x4 hard-key keypad, and a UART terminal. The digital alarm clock is organized into a finite state machine consisting of 7 states. The clock features 4 different commands input from the keypad: set the current time/date, set an alarm, display a list of commands, and display the current alarm. Upon connecting to power, the clock and alarm are not set, therefore the initial time and alarm set display incorrect values. This means that upon disconnecting from power, the current time/date and alarms are reset. The initial state of the device is the GET_KEY state.

System Specifications

Specification	**Parameter**
Commands	A: Set an Alarm B: Set the Current Time C: Display list of commands D: Display Alarm
Number of Alarms	1
Alarm interrupts when	hours, seconds, minutes match
Time Format	12-hour format Hour : Minute : Second
Date Format	Month : Date : Year
RTC features	- Corrections for months with less than 31 days - Corrections for leap year
RTC Communication Protocol	I2C Fast Mode (400 kHz)
UART Terminal Baud Rate	115.2 kbps
System Clock Speed	24 MHz

System Schematic

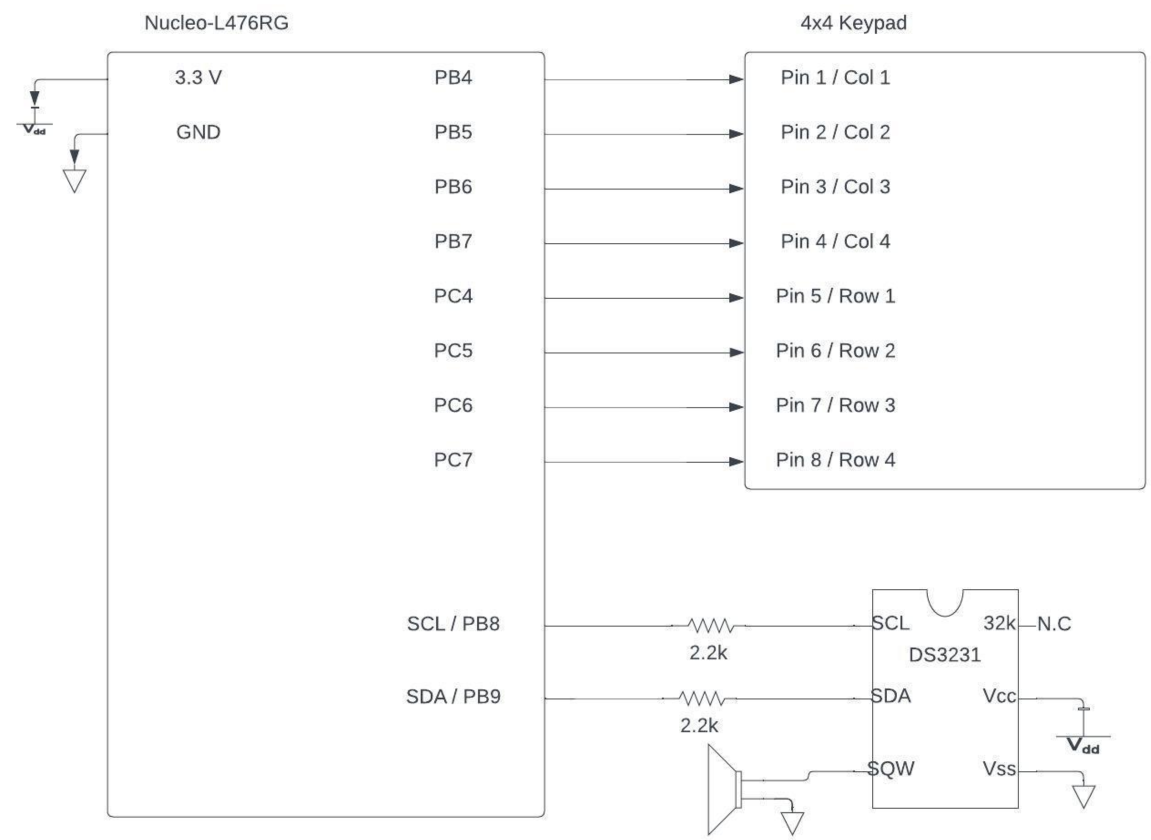

Figure 1:
Digital Alarm Clock Hardware Schematic

Terminal Screens of the Digital Alarm Clock

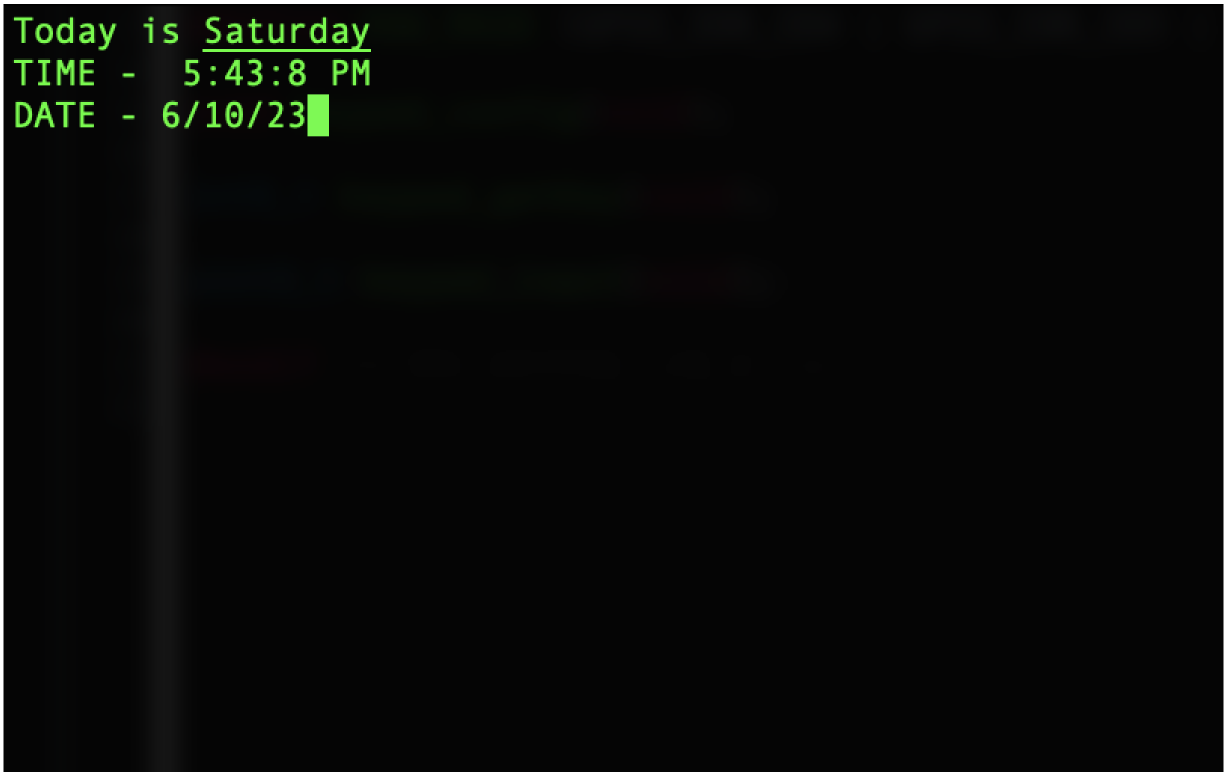

Figure 2: Terminal Clock Display

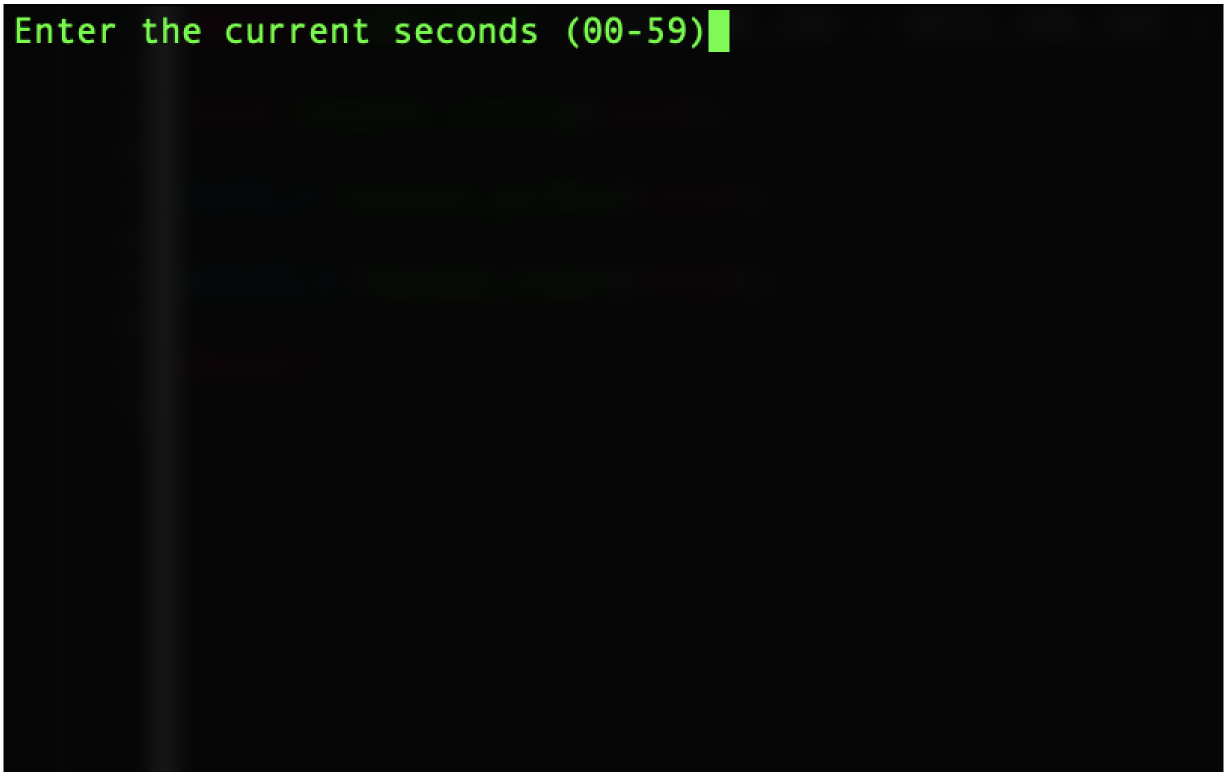

Figure 3: Terminal Set Time Display

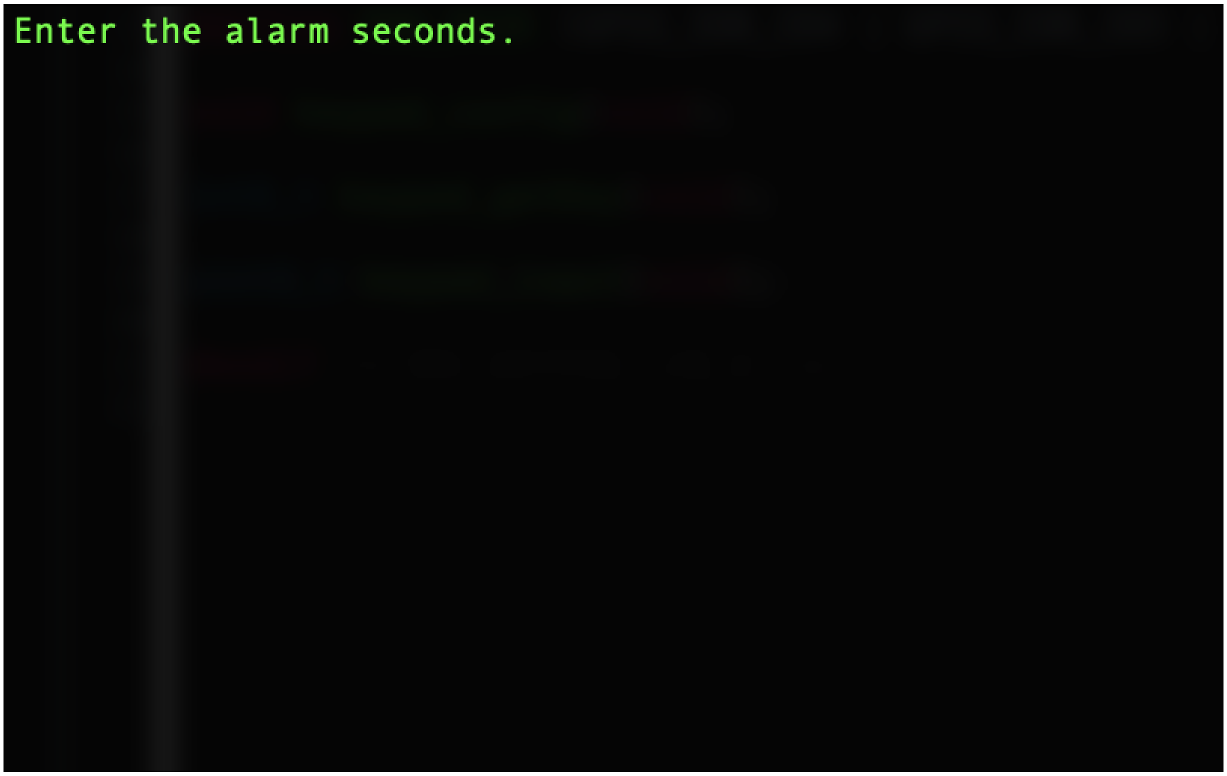

Figure 4: Terminal Set Alarm Display

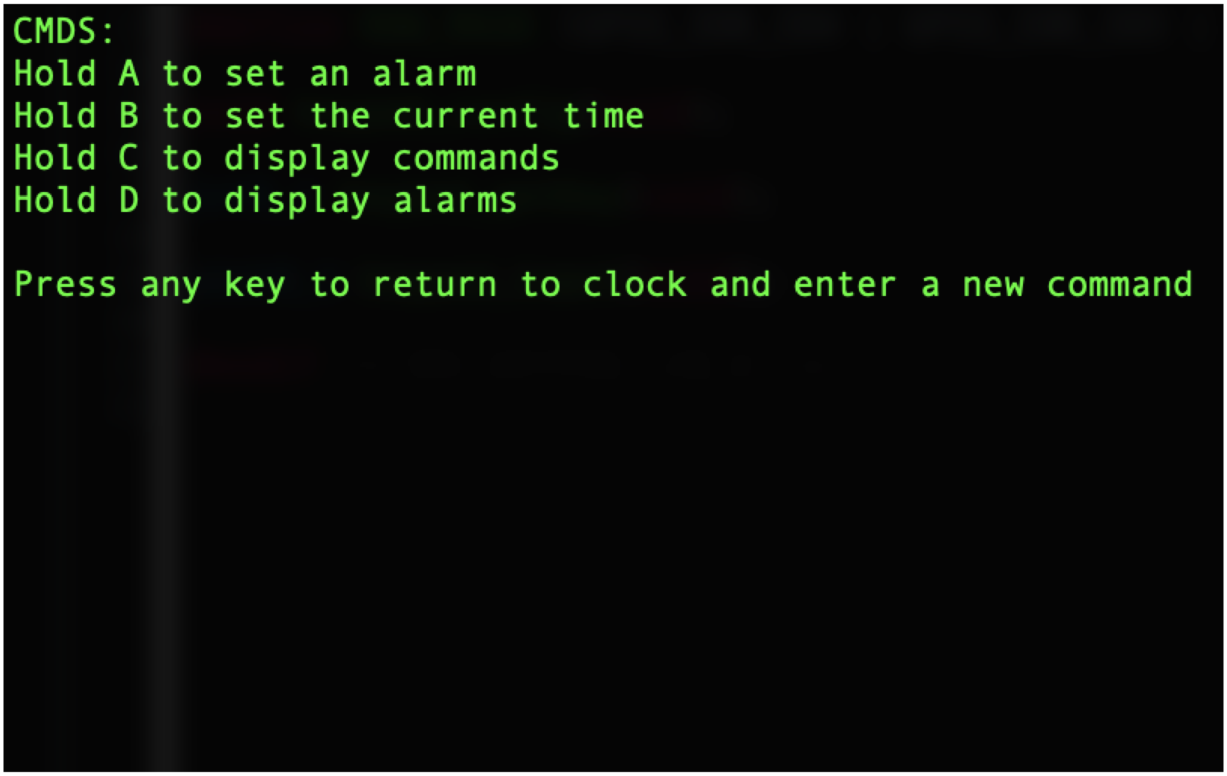

Figure 5: Terminal Commands Display

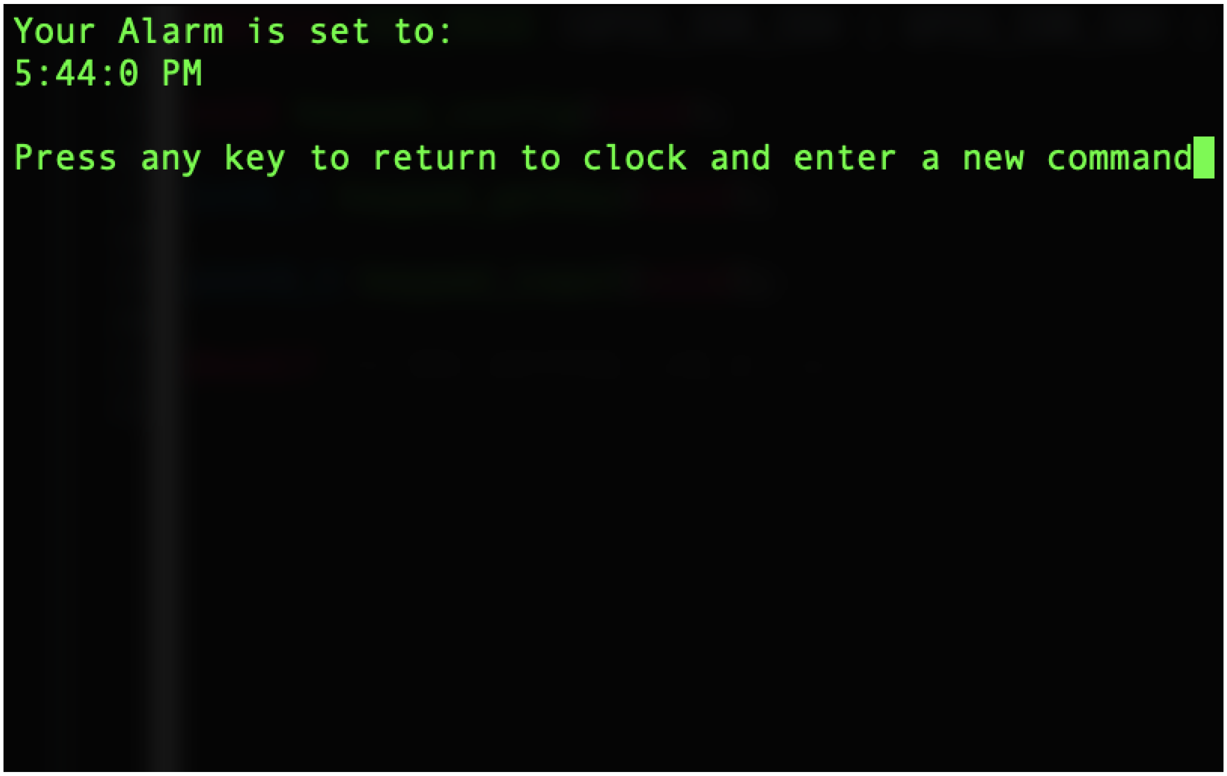

Figure 6: Terminal Alarm Display

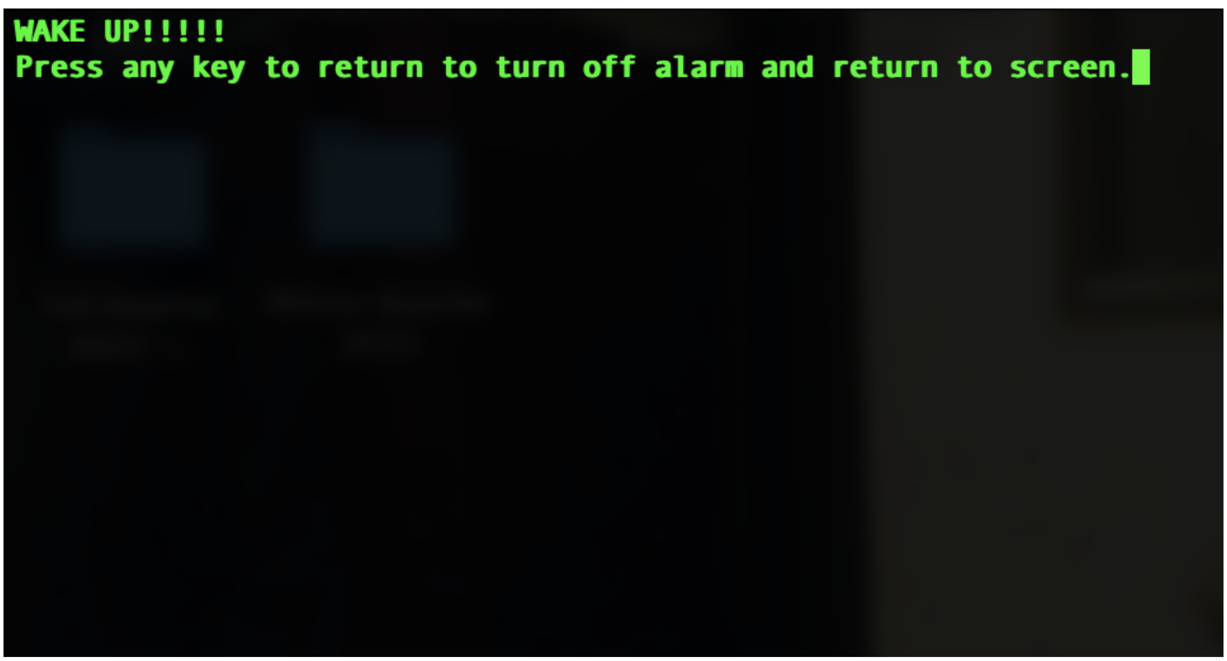

Figure 7: Terminal Alarm Interrupt Display

Software Architecture

The digital alarm clock program is guided by a 7 state FSM including GET_KEY, SET_TIME, SET_ALARM, DISPLAY_ALARM, DISPLAY_CMDS, DISPLAY_TIME, and ALARM_INT states. The digital alarm clock also uses three libraries that include functions to initialize and control the keypad, DS3231 RTC, and UART terminal actions.

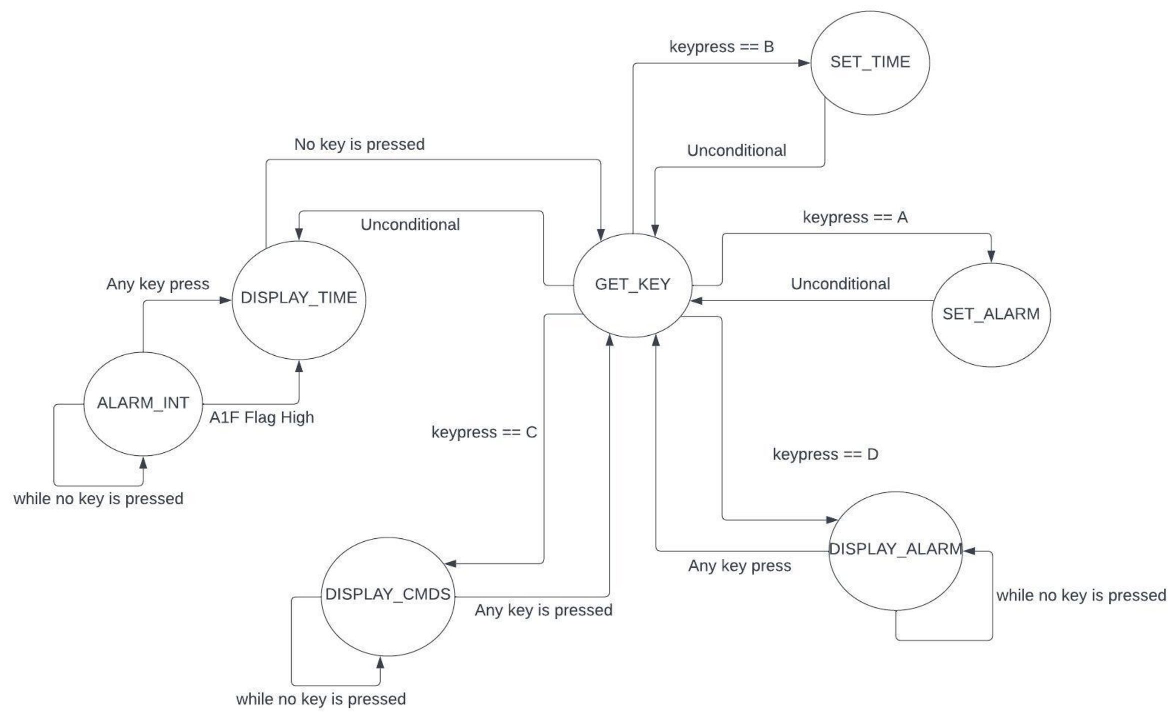

Figure 8: FSM

The digital alarm clock FSM consists of 7 states. The main state that guides the next program action is the GET_KEY state. This state reads the key pressed from the keypad (or not pressed) and determines the next state to enter. Perhaps the next most important state is the DISPLAY_TIME state. This is where the current times held in the registers of the DS3231 are read and printed to the UART terminal. This state requires no key to be pressed to enter. This state also reads the status register of the DS3231, and if the A1F flag is high, the clock enters the ALARM_INT state. This state prints an alarm screen to the terminal and the buzzer is activated by the DS3231. It also returns to GET_KEY upon any key press. If the "A" key is pressed on the keypad, the clock enters the SET_ALARM state. It is in this state where the program asks for user input from the keypad and writes to the alarm registers on the DS3231. If the "B" key is pressed on the keypad, the clock enters a SET_TIME state, where the program again asks for user input to write the current time to the DS3231. When the "C" key is pressed, the clock enters the DISPLAY_CMDS state where a set of commands is printed to the terminal. Upon a "D" key press, the clock enters the DISPLAY_ALARM state that prints the current alarm set. DISPLAY_CMDS and DISPLAY_ALARM screens remain on the terminal until a key is pressed and the clock returns to the GET_KEY state.

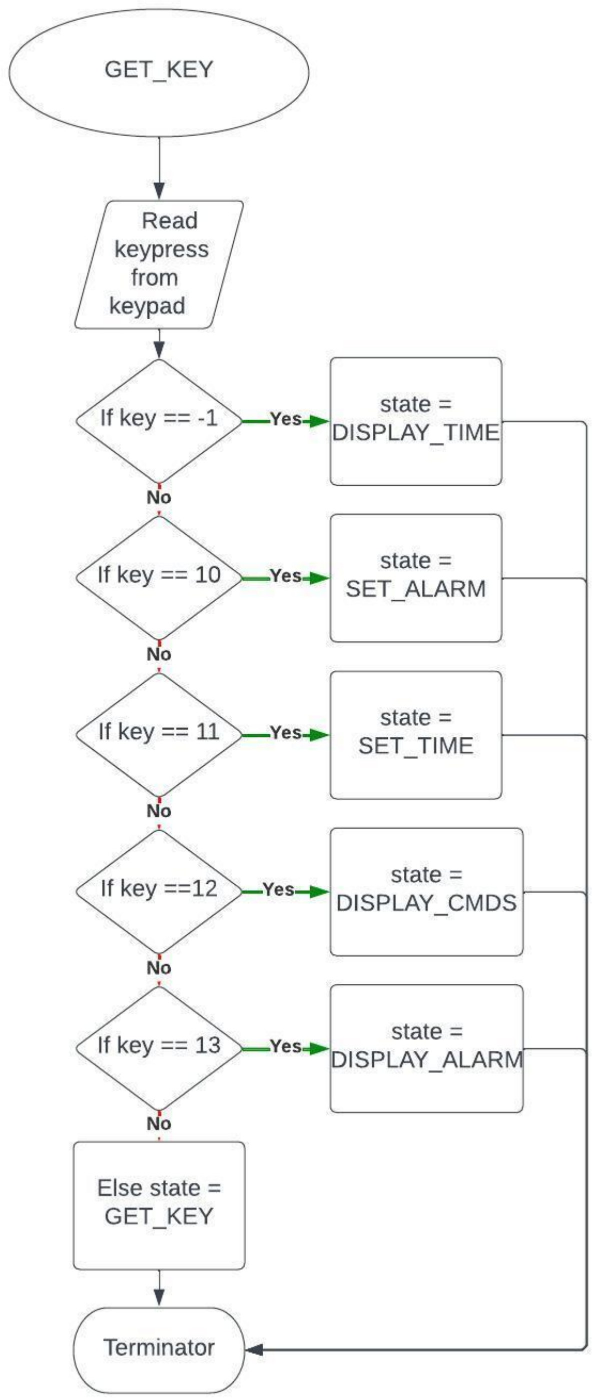

Figure 9: GET_KEY

The purpose of the GET_KEY state is to determine the next state to enter based on the key read from the keypad.

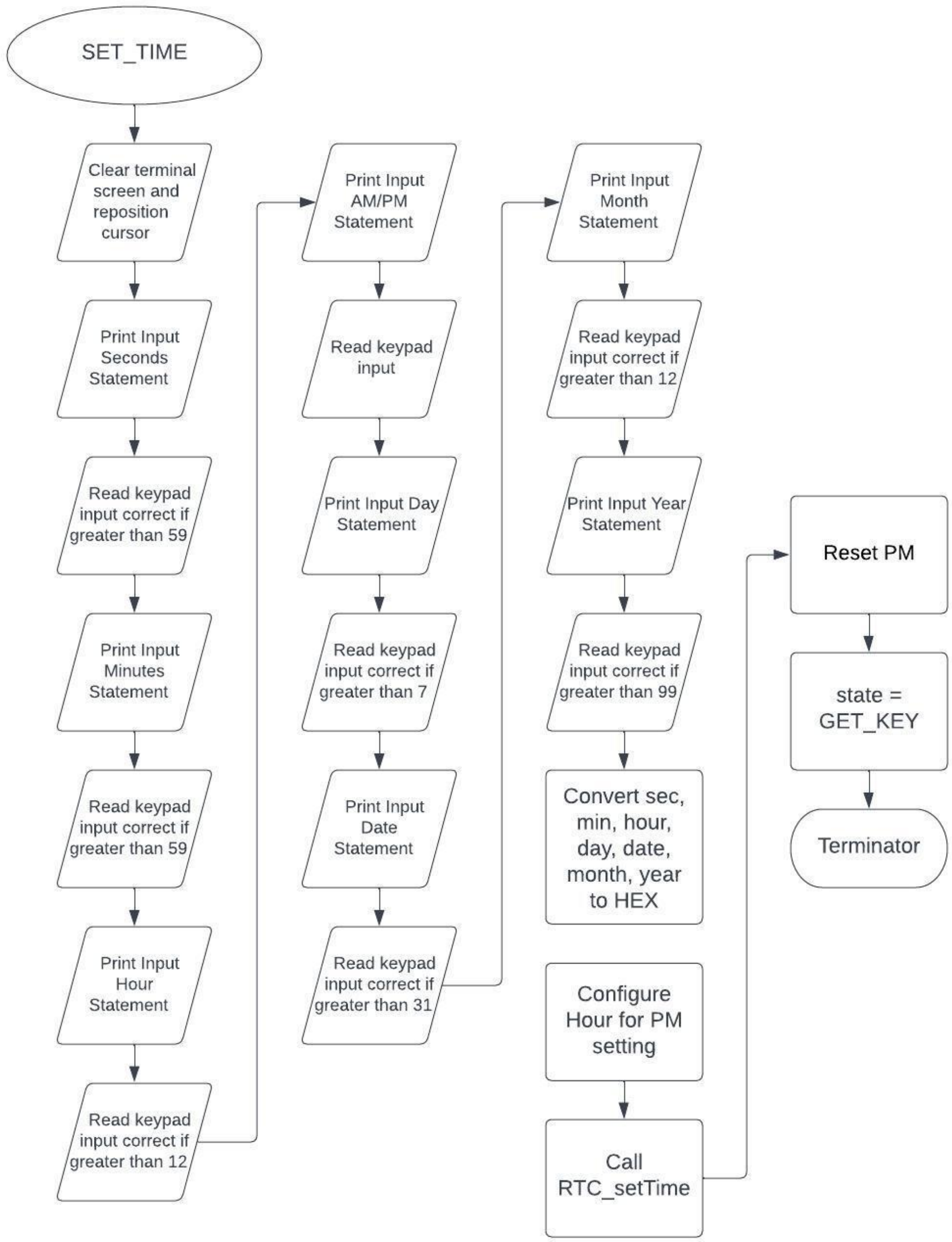

Figure 10: SET_TIME

The purpose of the SET_TIME state is to write to the seconds, minutes, hours, day, date, month, and year registers for the clock. The SET_TIME state does this by clearing the terminal then printing a series of instructions for user input. There is also an instruction to input AM or PM for the hour. Once all of the data is input from the keypad, it is converted to hexadecimal format, and the hour byte is configured with the aforementioned AM/PM bit. The RTC_setTIME function is then called and the data is written to the DS3231 clock registers.

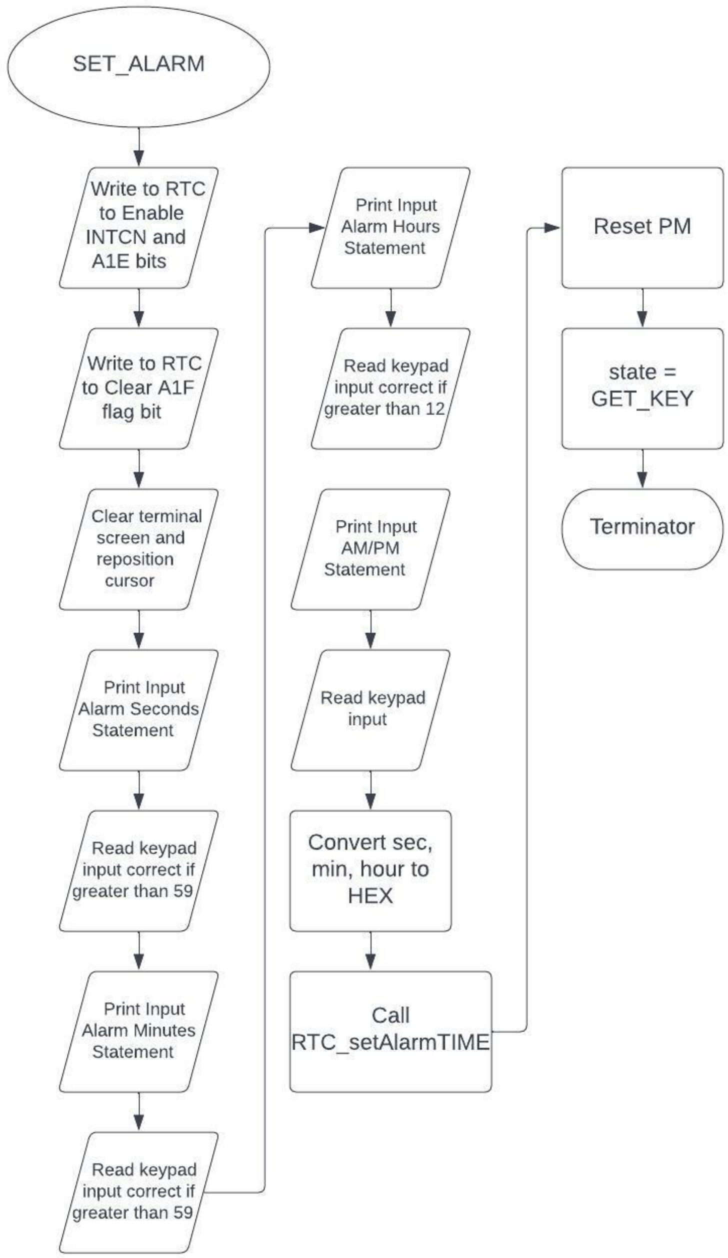

Figure 11: SET_ALARM

The purpose of the SET_ALARM state is to write to the alarm registers from keypad input. The SET_ALARM state does this in a similar manner to the SET_TIME state in that it prints a series of instructions, then waits for keypad input. The SET_ALARM state also converts the data to hexadecimal, but then calls the RTC_setAlarmTime function that writes the seconds, minutes, hours, and date alarm registers on the DS3231.

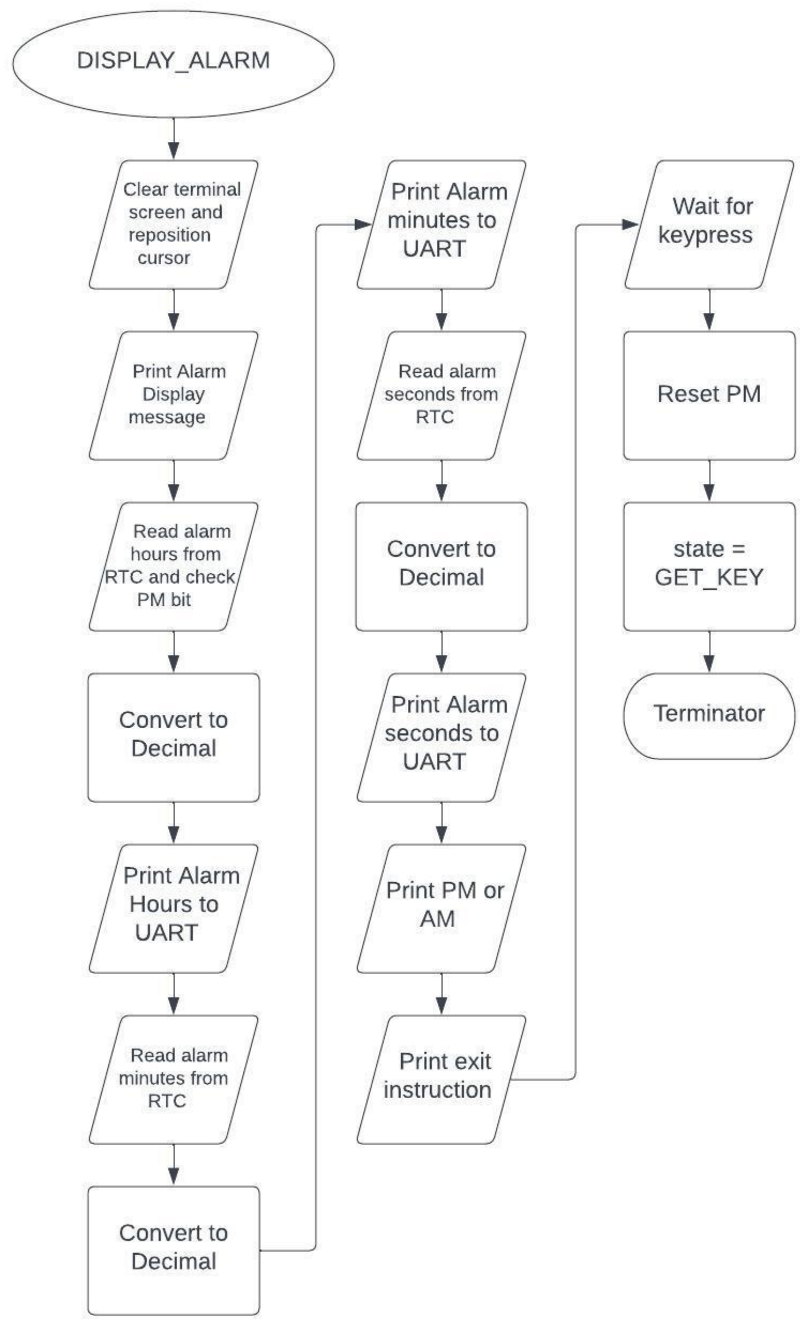

Figure 12: DISPLAY_ALARM

The purpose of the DISPLAY_ALARM state is to display the current alarm set on the terminal. The DISPLAY_ALARM state accomplishes this by reading from the alarm seconds, minutes, and hour registers, then printing to the terminal. To determine whether the hour is AM or PM, the program reads the fifth bit of the hour register, if it is high then it is PM and if not it is AM. This state then prints a user-friendly message, followed by the hours, minutes, and seconds separated by a colon, then instructions to exit.

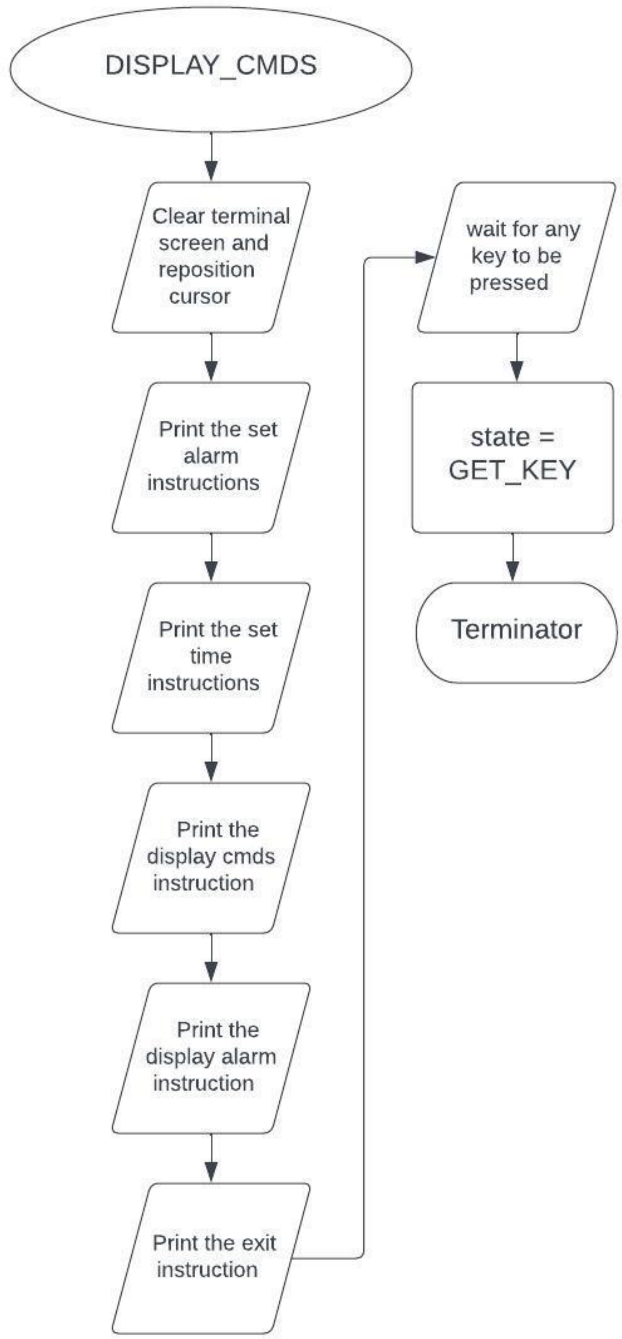

Figure 13: DISPLAY_CMDS
The purpose of the DISPLAY_CMDS state is to print a set of commands to the terminal. This is done by clearing the terminal, then printing a set of user-friendly instructions. To exit this screen, the user simply presses any key and the next state is GET_KEY.

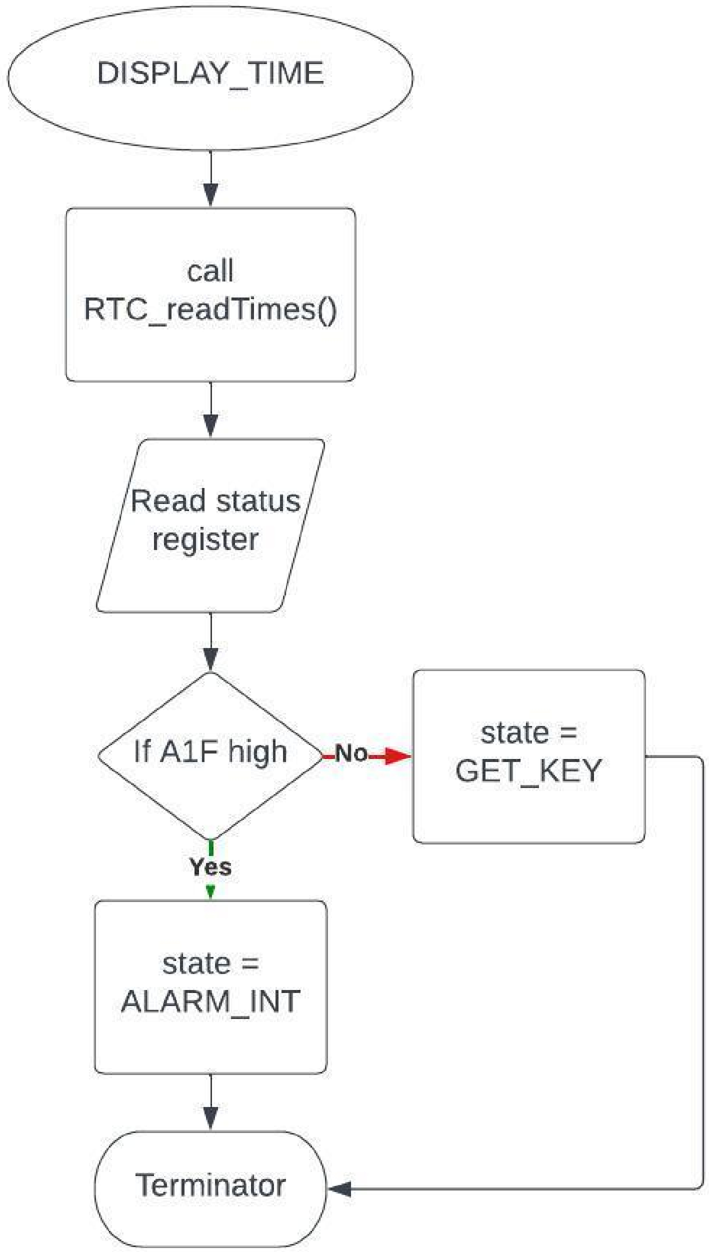

Figure 14: DISPLAY_TIME
The purpose of the DISPLAY_TIME state is to print the current date and time to the terminal via UART. This is completed by calling the RTC_readTimes function that reads the seconds, minutes, hours, day, date, month, and year registers from the DS3231 clock module. Once these values are read, they are printed to the terminal along with labels and a user-friendly message including the day of the week. It is also in this state that the status register is read, and if the first bit is high (A1F) that means the alarm is going off and the next state is ALARM_INT. Otherwise, the alarm has not gone off yet, and the next state is to GET_KEY.

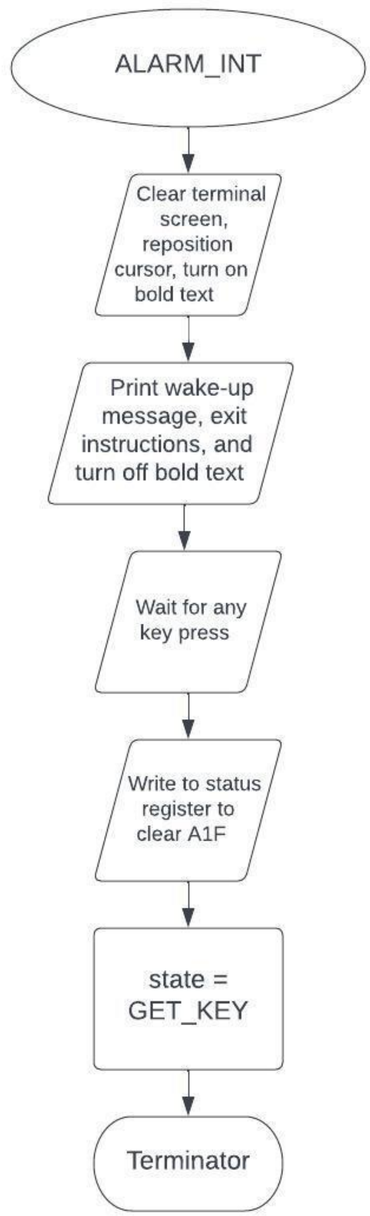

Figure 15: ALARM_INT

The purpose of the ALARM_INT screen is to print the alarm screen and clear the alarm flag in the status register. This is done by clearing the terminal screen, printing a user-friendly wakeup message and exit instructions. These messages are also printed in bold text. To exit this state the user simply presses any key and the status register is written to in order to clear the A1F bit and the next state is GET_KEY. It is in this state that the buzzer is activated, however this action is handled completely by the DS3231 clock module when the INTCN bit was written to high in the SET_ALARM state.

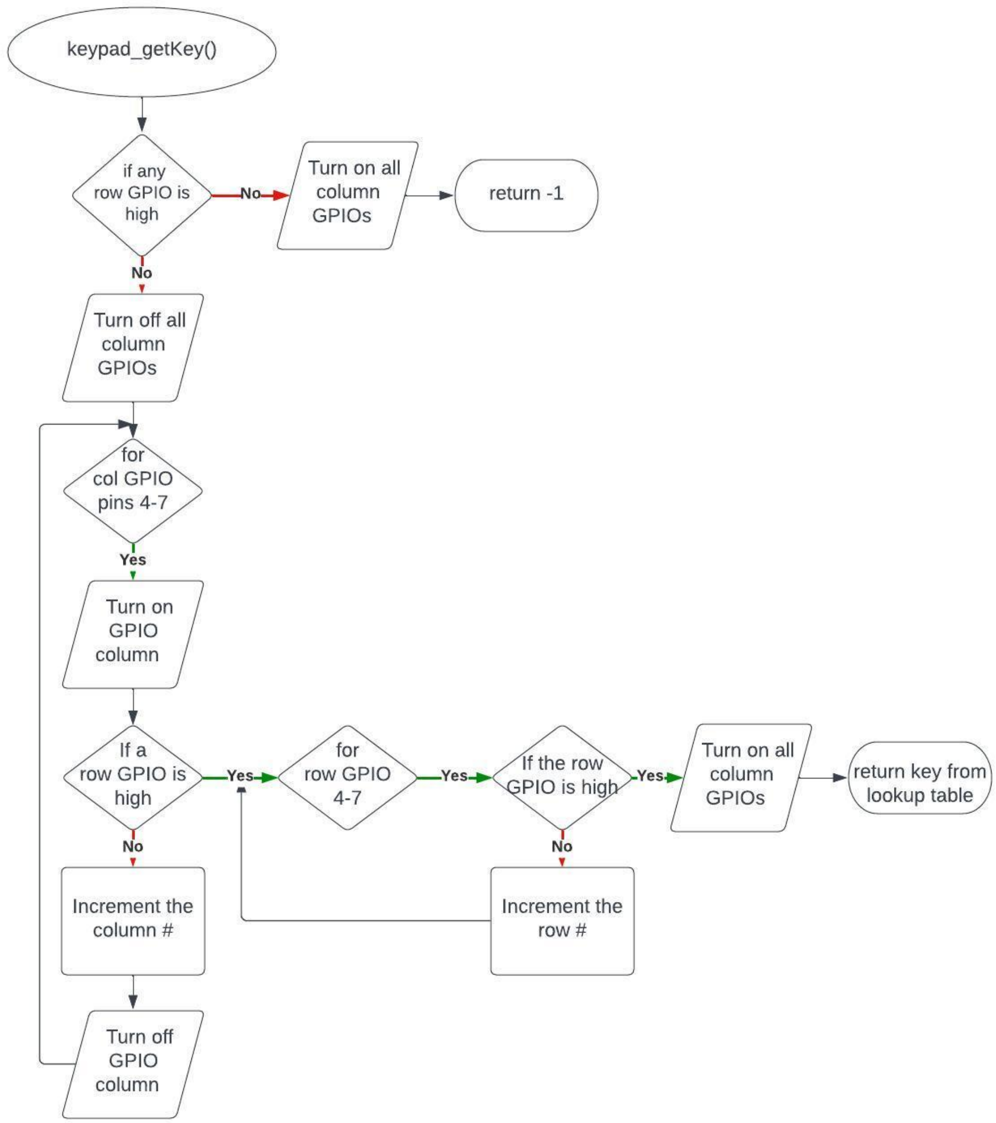

Figure 16: keypad_getKey

The purpose of the keypad_getKey() function is to read a keypress off of the keypad. This is done by first checking to see if any of the row GPIO pins are read to be logic high. If they aren't, the program turns on all column GPIO pins and immediately returns -1. Otherwise, the column GPIO pins are turned off and looped through. In each loop iteration, each column GPIO in is turned on, then checked to see if it turned on a row GPIO pin. If not, the column number is incremented, the columns GPIO is turned off, then a new loop iteration starts. Otherwise, the row GPIO pins are then for-looped. For each pin, it is checked to see if it is logic high, and if so all column GPIOs are turned on so another press can be read, then the key is returned from a lookup table using the row and column numbers as indices. If that row was not read at logic high, then the row number is incremented and a new loop iteration begins.

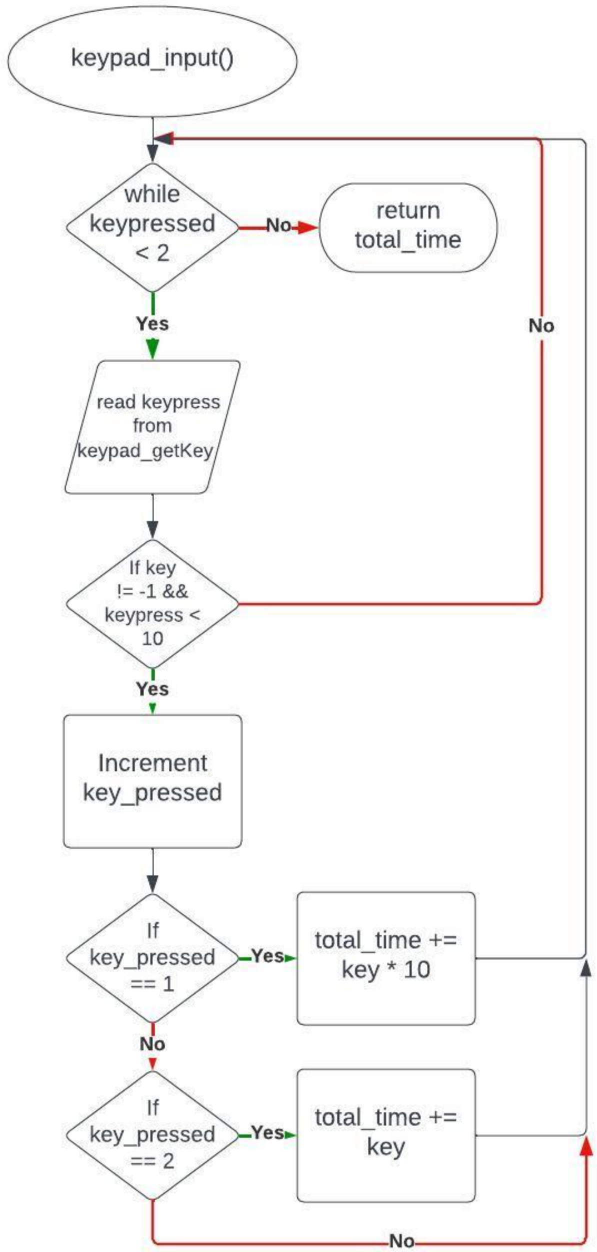

Figure 17: keypad_input

The purpose of the keypad_input function is to read user input from the keypad. This is done by keeping track of the number of keypresses that have already been registered. If two keypresses have been registered, then the time that was inputted is returned. On the first correctly input keypress, the total_time is incremented by the keypress * 10. This is because the first press represents the tens' place. The second keypress is done in a similar manner, but registers the ones' place.

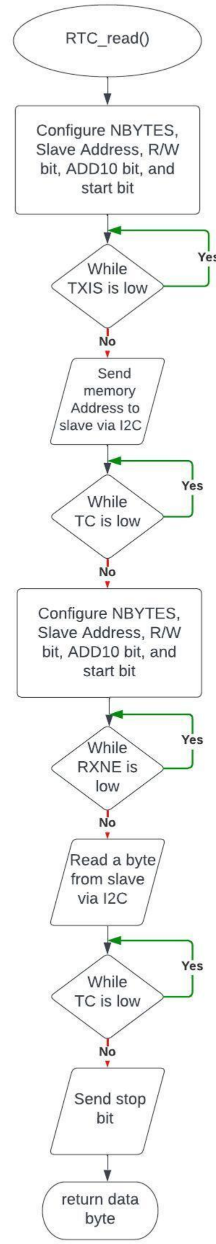

Figure 18: RTC_read

The purpose of the RTC_read function is to read a byte from the DS3231 via I2C. This is done first by configuring the NBYTES, SADD, R/W, ADD10, and start bits in the I2C CR2 register. Once this is done, the memory address to be read from is written to the device via I2C, then the data byte from that address can be read. Before the write can take place, the TXIS bit needs to be set by hardware, and the RXNE bit needs to be set by hardware before a read can take place. After each action, the program waits for the TC bit to be set, and finally sends a stop bit after the byte has been read.

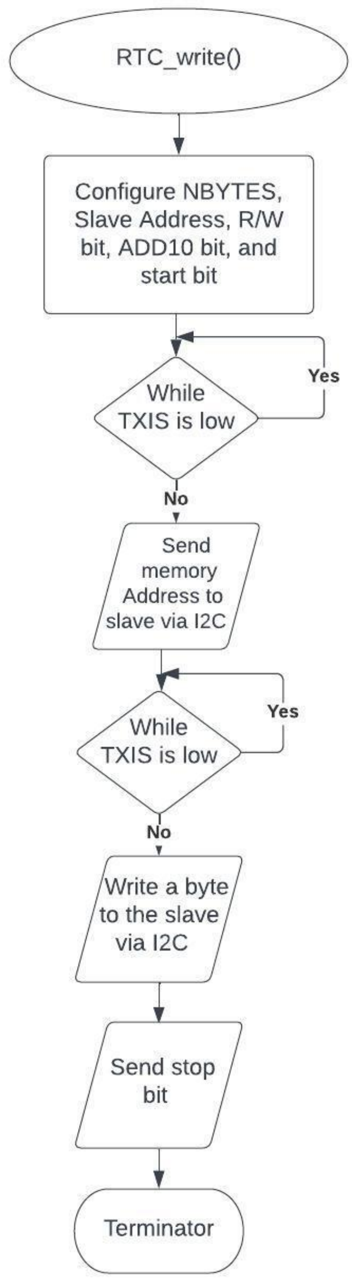

Figure 19: RTC_write

The purpose of the RTC_write function is to write a byte to the DS3231 via I2C. This is accomplished by first configuring the NBYTES, SADD, R/W, ADD10, and start bits in the I2C CR2 register. Once this is complete the program waits for the TXIS bit to go high, then sends the memory address to be written to the DS3231. Once this completes, the program waits for the TXIS bit to go high again, and once it does it writes the data to be written to the DS3231. Finally a stop bit is sent and the function terminates.

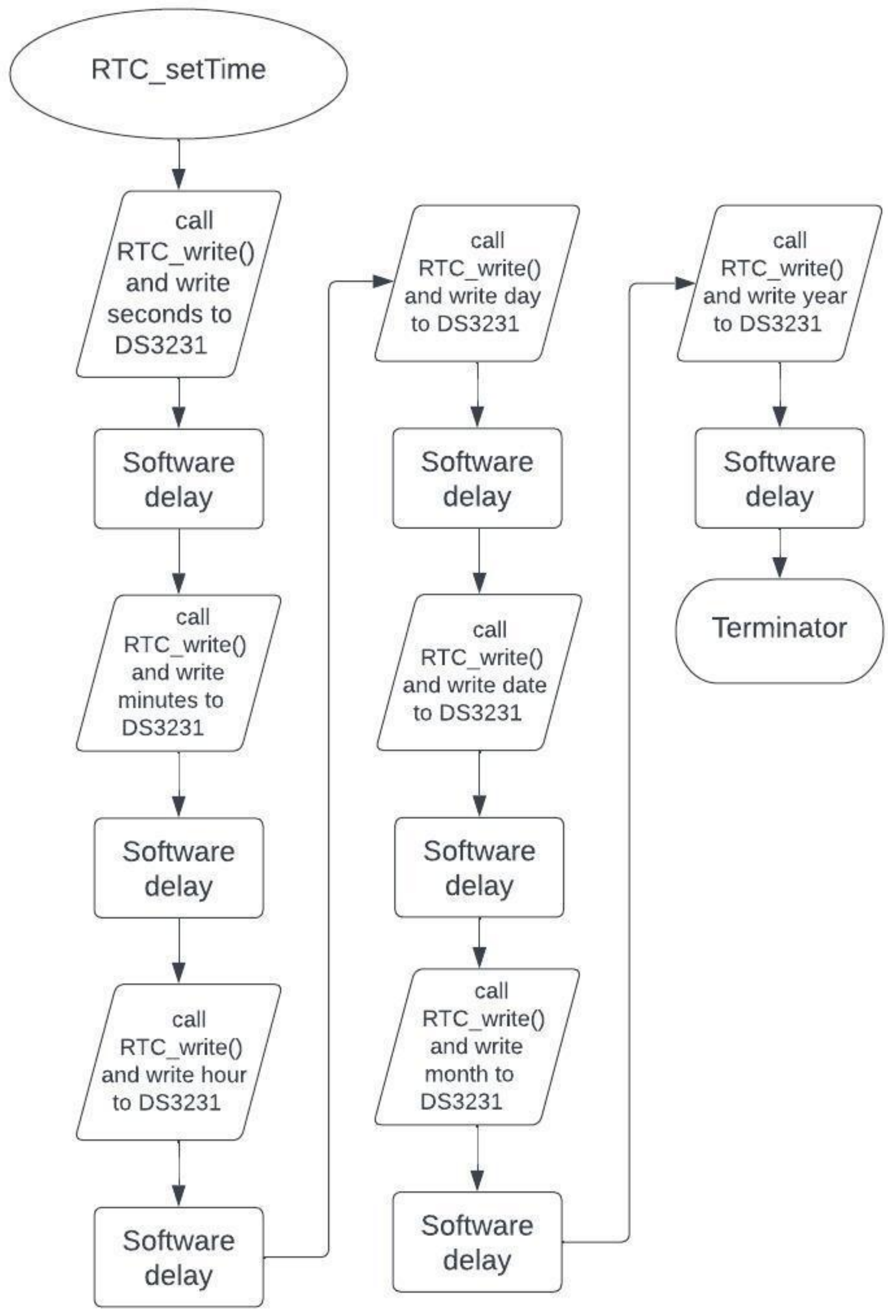

Figure 20: RTC_setTime

The purpose of this function is to write the input seconds, minutes, hour, day, date, month, and year to the current time registers on the DS3231. This is done by calling RTC_write() for each time variable, then adding a software delay for the I2C write operation.

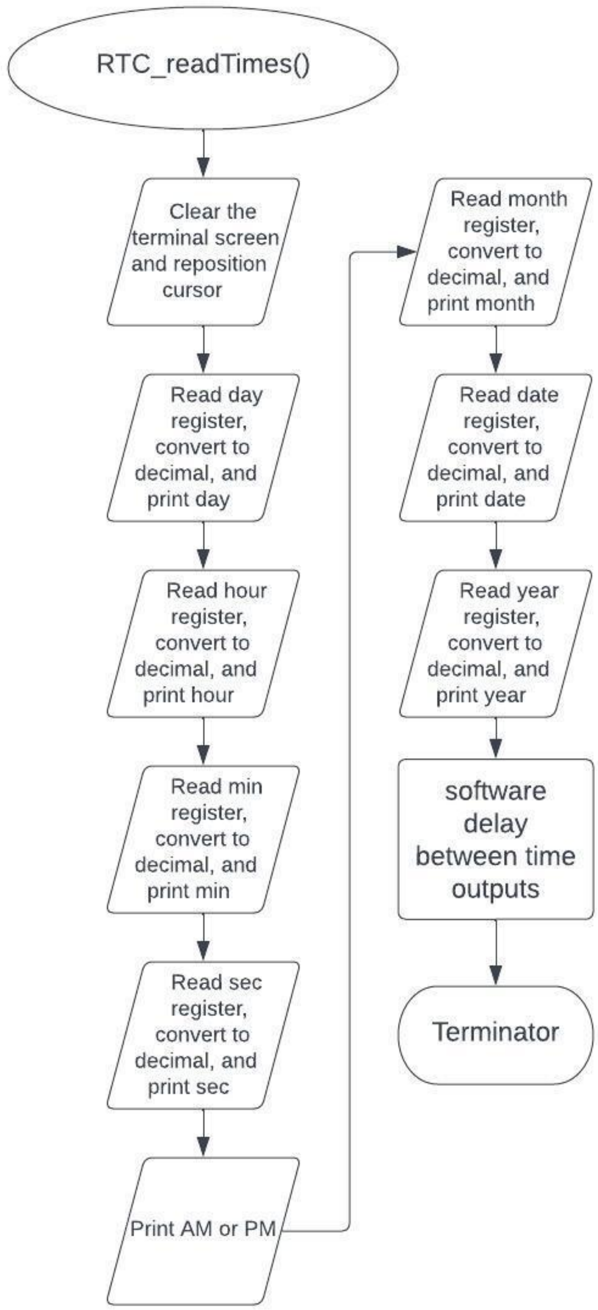

Figure 21: RTC_readTimes

The purpose of the RTC_readTimes function is to read the seconds, minutes, hours, day, date, month, and year registers on the DS3231 and print these values to the terminal. This is done by calling the RTC_read function for each time value, then converting the return value to decimal. Once converted the values are then printed to the terminal via UART along with a user-friendly message. The time is printed in hours:minutes:seconds format in 12-hour mode, and the date is printed in month/day/year format.

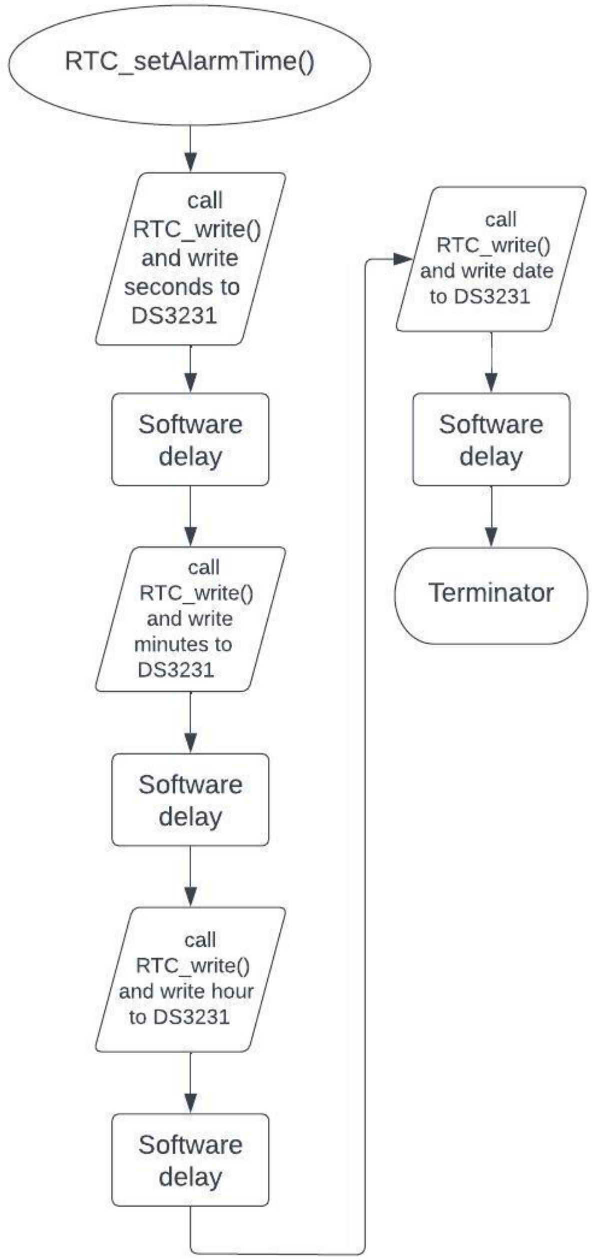

Figure 22: RTC_setAlarmTime

The purpose of this function is to write the input alarm seconds, minutes and hours to the DS3231 clock module via I2C. This is done by calling RTC_write for each time variable. The date write does not include a date variable, this is because the date is not used when setting the alarm off. The alarm is triggered when hours, minutes, and seconds match. However, the date register contains a configuration bit in order to add this alarm setting, so 0x81 is written to the alarm date register.

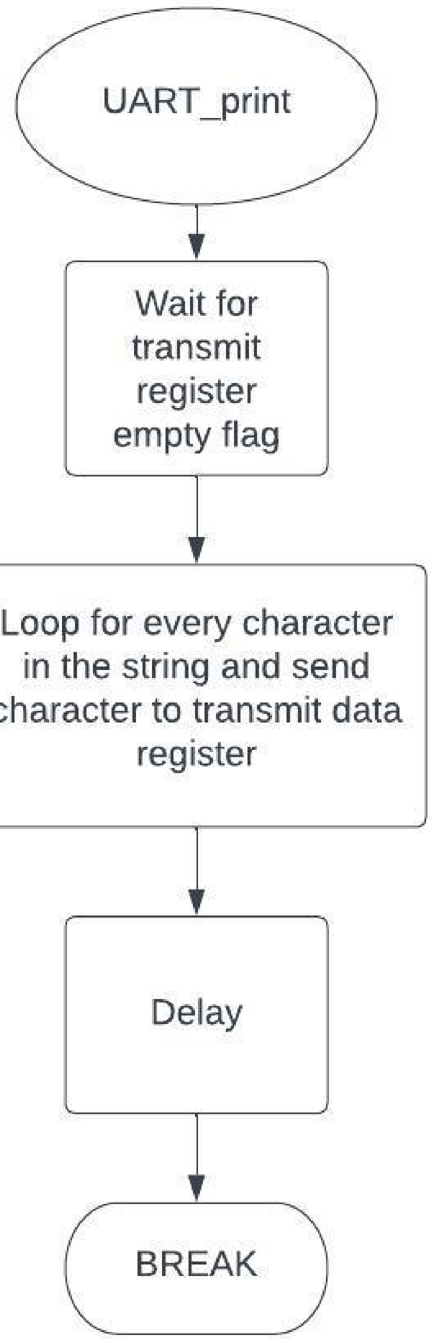

Figure 23: UART_print
This function starts by waiting for the transmit register empty flag to be set to high, then loops to send a character from an input string to the transmit data register. A delay is then used between sending characters.

Power Calculations

- Battery Life for the Digital Alarm Clock with only DS3231 RTC module:
    - Timekeeping Battery Current = 3.0 µA
    - Capacity of Energizer CR2032 Battery = 220 mAh
    - Battery Life = 220 mAh / 3.0 µA = **8.4 years**

- Battery Life for the Digital Alarm Clock using all Peripherals:
    - Average Current Consumption of GPIOA = 115.2 µA
    - Average Current Consumption of USART2 = 33.6 µA
    - Average Current Consumption of I2C1 Fast Mode = 31.2 µA
    - Average Active Supply Current of DS3231 RTC = 200 µA
    - Total Current Consumption = 380 µA
    - Total Power Consumption = (380 µA) * (3.3 V) = 1.2 mW
    - Battery Life = 220 mAh / 380 µA = **24.2 days**

User Manual

- Setting the Time
    1. To set the time on the digital alarm clock, make sure that the clock screen is displayed on the terminal screen. If not, press any key or complete your current action to return to the clock screen.
    2. Press the "B" key on the keypad in order to start the SET_TIME state.
    3. Upon pressing the "B" key, the terminal screen clears and an instruction to enter the current seconds (00-59) is printed to the terminal, as shown in Figure 24. Enter the current seconds on the keypad with 2 keypresses; be sure to follow the correct format shown in the parentheses.

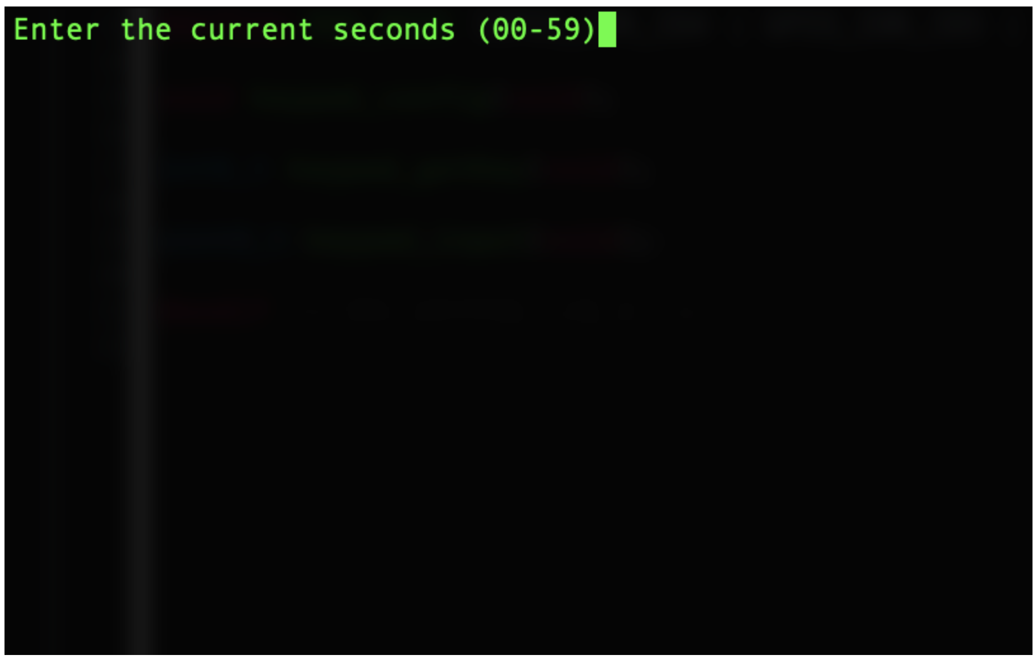

Figure 24: Set Clock Terminal Display

    4. After the seconds have been entered, repeat Step 3 for the current minutes, hour, AM/PM, day, date, month, and year.
    5. Upon completion of Step 4, the digital alarm clock returns to display the current time, which displays the time that was just set.

- <u>Setting the Alarm</u>
    1. To set the alarm on the digital alarm clock, make sure that the clock screen is displayed on the terminal screen. If not, press any key or complete your current action to return to the clock screen.
    2. Press the “A” key on the keypad in order to start the SET_ALARM state.
    3. Upon pressing the “A” key, the terminal screen clears and an instruction to enter the alarm seconds (00-59) is printed to the terminal, as shown in Figure 25. Enter the alarm seconds on the keypad with 2 keypresses; be sure to follow the correct format shown in the parentheses.

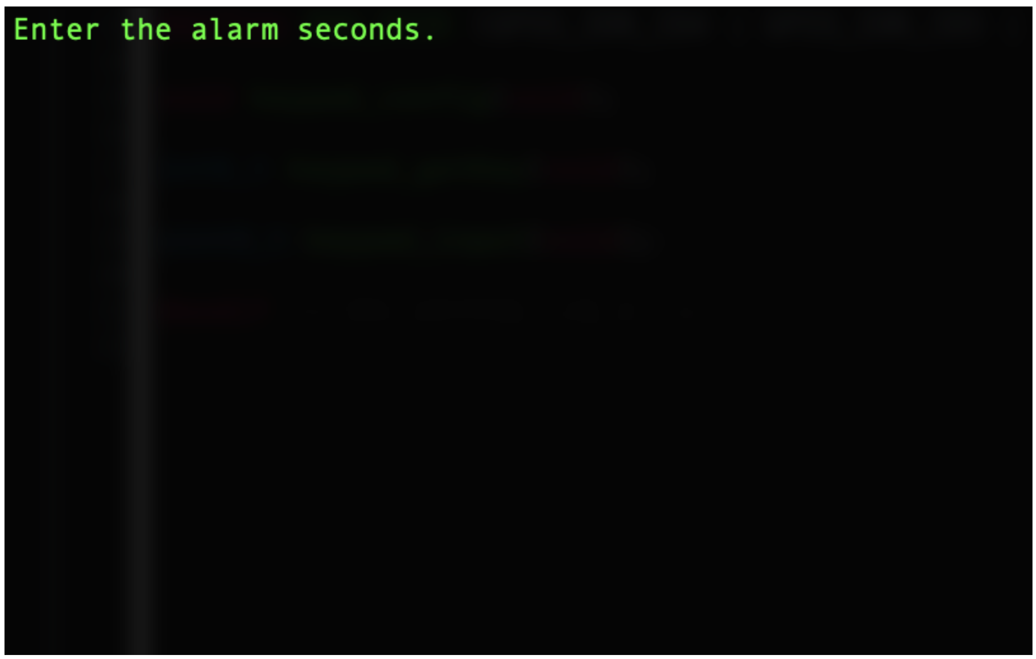

Figure 25: Set Alarm Terminal Display

    4. After the seconds have been entered, repeat Step 3 for the alarm minutes, hour, and AM/PM.
    5. Upon completion of Step 4, the digital alarm clock returns to display the current time.

- <u>Displaying the Set of Commands</u>
    1. To display the set of digital alarm clock commands, make sure that the clock screen is displayed on the terminal screen. If not, press any key or complete your current action to return to the clock screen.
    2. Press the “C” key on the keypad in order to start the DISPLAY_CMDS state.
    3. Upon pressing the “C” key, the terminal screen clears and a set of digital alarm clock commands are printed to the terminal screen, as shown in Figure 26.

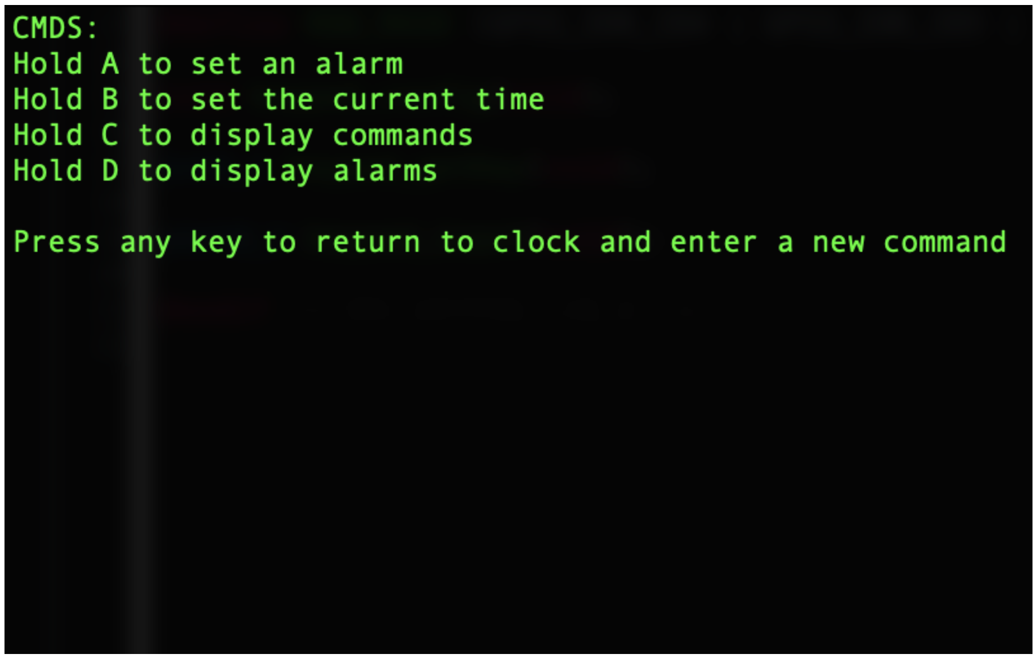

Figure 26: Commands Display

4. In order to return to the clock screen, simply follow the exit instructions and press any key.

- <u>Displaying the Current Alarm</u>
    1. To display the digital alarm clock alarm, make sure that the clock screen is displayed on the terminal screen. If not, press any key or complete your current action to return to the clock screen.
    2. Press the “D” key on the keypad in order to start the DISPLAY_ALARM state.
    3. Upon pressing the “D” key, the terminal screen clears and the current alarm set is printed to the terminal screen, as shown in Figure 27.

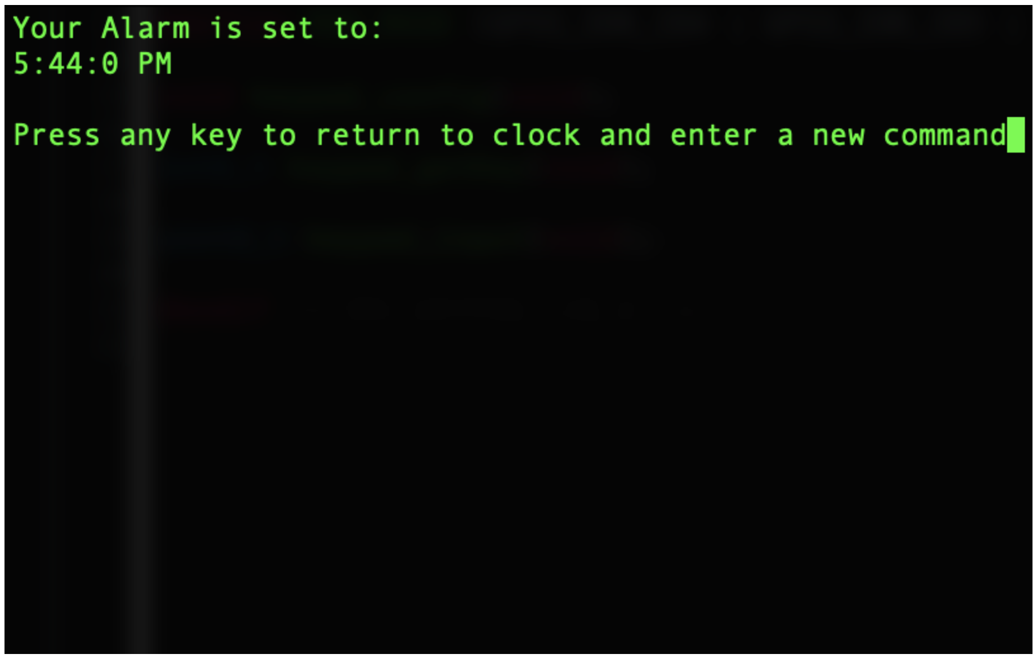

Figure 27: Set Alarm Terminal Display

    4. In order to return to the clock screen, simply follow the exit instructions and press any key.

References

1. **STM32-L47xxx Reference Manual [Accessed June 4, 2023]**
2. **STM32-L47xxx Datasheet [Accessed June 4, 2023]**
3. **Analog Devices - DS3231 Datasheet [Accessed June 4, 2023]**