Contents

[Introduction 2](#_Toc160442659)

[Code Structure 3](#_Toc160442660)

[Lcd.cpp 3](#_Toc160442661)

[Class Constructor: 4](#_Toc160442662)

[Initialization Function (`void LCD::init()`): 4](#_Toc160442663)

[Command and Data Transmission Functions 4](#_Toc160442664)

[Additional Display Functions: 4](#_Toc160442665)

[RTC.cpp 4](#_Toc160442666)

[Class Structure 5](#_Toc160442667)

[Initialization Function (`DS1307\_HAL::begin()`) 5](#_Toc160442668)

[Time Initialization Function (`DS1307\_HAL::initializeTime()`) 5](#_Toc160442669)

[Time Reading Function (`DS1307\_HAL::readTime()`) 5](#_Toc160442670)

[Utility Functions (`DS1307\_HAL::decToBcd()` and `DS1307\_HAL::bcdToDec()`) 5](#_Toc160442671)

[Main code 5](#_Toc160442672)

[Explanation 7](#_Toc160442673)

[Outputs 7](#_Toc160442674)

[Conclusion 8](#_Toc160442675)

[References 9](#_Toc160442676)

# Introduction

HAL is basically a layer between software and hardware that makes code user-friendly and portable. In the paper INTRODUCTION TO HARDWARE ABSTRACTION LAYERS FOR SOC by Sungjoo Yoo and Ahmed A. Jerraya it is said that “we define HAL as all the software that is directly dependent on the underlying HW” [1]

Using a HAL let us implement functionality without affecting or modifying the higher level system. This project includes the integration and interaction of key hardware components, the DS1307 Real-Time Clock (RTC) and a 16x2 LCD display, with the ATMega328p microcontroller. All these components work together to efficiently manage time, communicates and displays. Utilizing Hardware Abstraction Layer (HAL) principles, the code establishes a well-structured interface with the ATMega328p, allowing for direct communication and control of the DS1307 RTC through I2C protocol.

The RTC here is used as time keeping module that updates time every second with minimal power consumption. The 16x2 LCD is used to display the time using 4bit control mode, facilitates a user-friendly display mechanism, presenting real-time information fetched from the RTC. Employing a register-level approach for communication with both the RTC and LCD, the project emphasizes efficient utilization of the ATMega328p's resources, ensuring optimal performance. The code initializes the RTC with a specific date and time, using the HAL to abstract away low-level details and provide a direct interface for the ATMega328p to interact with the RTC. The LCD initialization sequence and subsequent display operations similarly employ the HAL structure, containing important details and ensuring modularity for future code upgrades. The choice of 4-bit mode for the LCD communication optimizes pin utilization on the ATMega328p, an important consideration in resource-constrained environments.

This project integrates hardware components, software design principles, and microcontroller capabilities, presenting an efficient use of HAL to show the interaction between the ATMega328p, DS1307 RTC, and 16x2 LCD. Through this system, the project demonstrates an efficient use of HAL to integrate real-time hardware using custom designed interfaces.

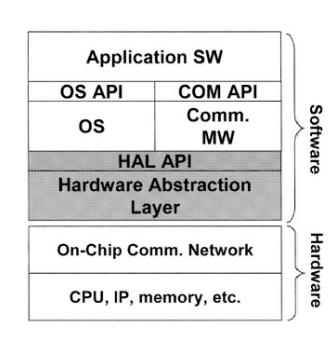


Figure 1 Hardware Abstraction Layer [1]

# Code Structure

Figure 2 Structure of HAL

The HAL structure is defined as follows,

We have developed .cpp and .h files for each module used in the hardware abstraction layer of the project. These two files maintain the modularity of the code.

* **.cpp files:** Definitions of functions used in code.
* **.h files:** Declaration of functions used in code.

# Lcd.cpp

This particular file contains the code that implements a C++ class for interfacing with a 16x2 LCD display. This class, denoted as `LCD` contains functionalities for initializing the LCD, sending commands and data, and performing common display operations. The code follows a hardware abstraction layer (HAL) design, providing an abstraction for the underlying hardware, specifically the ATMega328p microcontroller in an Arduino environment.

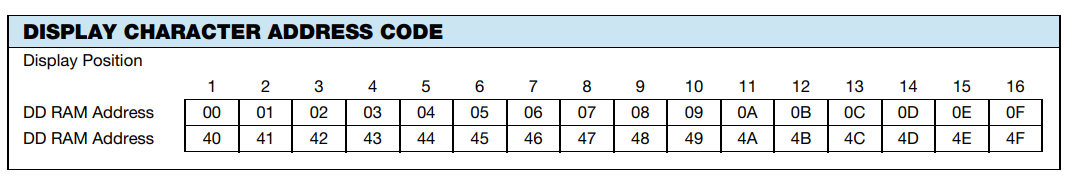


Figure 3 LCD Character Map [5]

## Class Constructor:

The constructor initializes the LCD object by assigning provided pin values for RS (Register Select), RW (Read/Write), E (Enable), and data pins (D4-D7). These pins are crucial for communication between the microcontroller and the LCD.

## Initialization Function (`void LCD::init()`):

This function sets the pin modes for RS, RW, E, and data pins. It then executes an initialization sequence (0x33, 0x32, 0x28, 0x0C, 0x01) to configure the LCD in 4-bit mode. A short delay is given to allow the LCD to stabilize.

Command and Data Transmission Functions

These functions control the communication with the LCD. `lcdCommand` sets RS low to indicate a command, while `lcdData` sets RS high for data transmission. Both functions call the `send` method, which manipulates the data pins to transmit 4-bit information, and then invokes `pulseEnable` to generate the necessary pulses for the LCD.

## Additional Display Functions:

* `void LCD::clear()`: Sends the command to clear the LCD screen.
* `void LCD::setCursor(int col, int row)`: Positions the cursor at the specified column and row.
* `void LCD::lcdPrint(String str)`: Iterates through each character in the input string, using `lcdData` to display the characters on the LCD.

In the code, delay functions are added to ensure proper timing, a common practice in embedded systems programming. This class contains the basics of LCD communication, providing a clean and modular interface for developers to incorporate LCD functionalities into their projects while abstracting away low-level hardware details. The modular design and adherence to HAL principles make this code easily reusable and adaptable for diverse applications.

# RTC.cpp

The provided code presents a Hardware Abstraction Layer (HAL) for interfacing with the DS1307 Real-Time Clock (RTC) through the I2C protocol in the context of an Arduino environment. This HAL, within the `DS1307\_HAL` class, facilitates the initialization and retrieval of time information from the DS1307 RTC with a focus on clarity, modularity, and ease of integration.

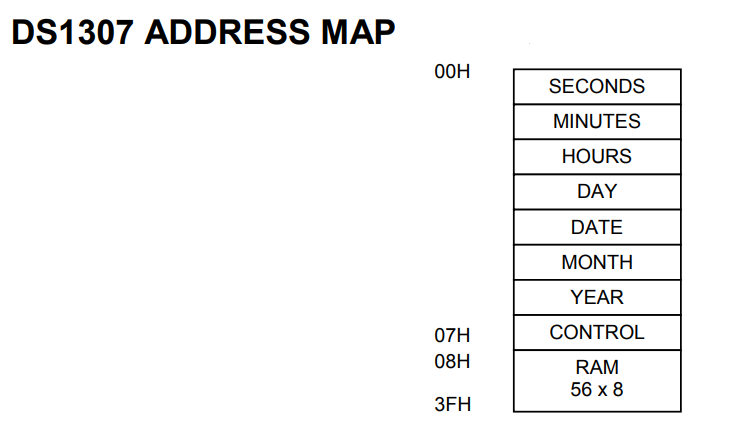


Figure 4 RTC Register Mapping [6]

## Class Structure

The `DS1307\_HAL` class is designed to abstract the intricacies of I2C communication with the DS1307 RTC, promoting modularity and code reuse. It incorporates functions for initiating communication (`begin()`), initializing the RTC with a specific date and time (`initializeTime()`), and reading the current time from the RTC (`readTime()`). The class employs utility functions (`decToBcd()` and `bcdToDec()`) for converting between decimal and binary-coded decimal (BCD) representations, a common requirement in RTC communication.

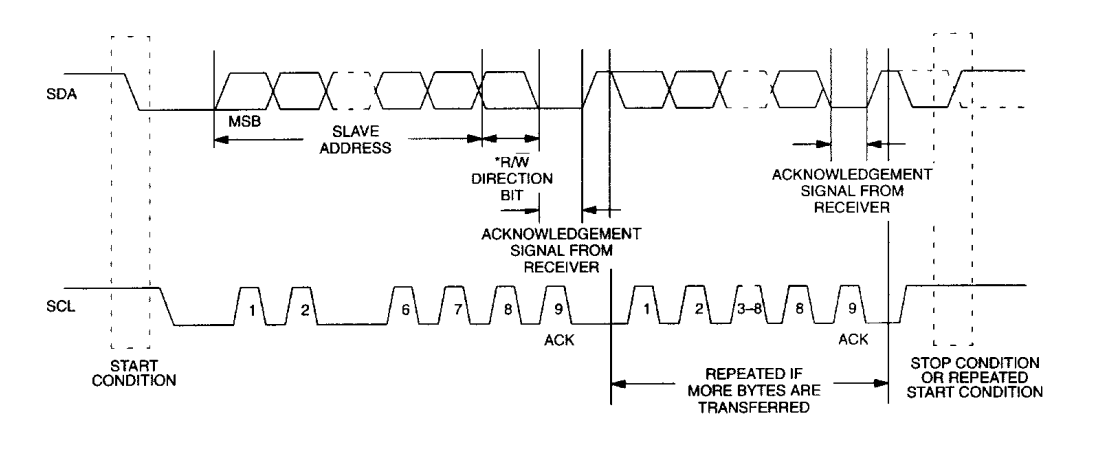


Figure 5 Data Transmission on RTC DS1307

Initialization Function (`DS1307\_HAL::begin()`)

The `begin()` function initializes the Wire library for I2C communication, establishing the groundwork for subsequent interactions with the DS1307 RTC.

## Time Initialization Function (`DS1307\_HAL::initializeTime()`)

The `initializeTime()` function sets the DS1307 RTC to a specified date and time. It begins the I2C transmission at the DS1307 address and writes the appropriate values to the corresponding registers, considering the specific bit masking needed for clock halt (CH) and 24-hour mode.

## Time Reading Function (`DS1307\_HAL::readTime()`)

The `readTime()` function retrieves the current time from the DS1307 RTC. It initiates an I2C transmission to the DS1307 address, requests the necessary 7 bytes representing seconds, minutes, hours, day, date, month, and year, and then decodes the received binary-coded decimal (BCD) values into integer representations.

## Utility Functions (`DS1307\_HAL::decToBcd()` and `DS1307\_HAL::bcdToDec()`)

These utility functions assist in converting between decimal and binary-coded decimal representations, ensuring compatibility with the DS1307 RTC's register format.

In summary, the `DS1307\_HAL` class provides a clear and modular interface for Arduino developers to interact with the DS1307 RTC, abstracting away low-level I2C details and offering functionalities for time initialization and retrieval. This HAL design promotes code readability, reusability, and ease of integration into diverse projects requiring real-time clock capabilities.

# Main code

The code for the interface of RTC and LCD with Arduino module to display the current time is provided and explained below.

#include <Arduino.h>

#include "lcd.h"

#include "rtc.h"

// LCD pins

const int RS = 8;

const int RW = 9;

const int E = 10;

const int D4 = 11;

const int D5 = 12;

const int D6 = 13;

const int D7 = 14;

// Create instances of LCD and RTC

LCD lcd(RS, RW, E, D4, D5, D6, D7);

DS1307\_HAL rtc;

void setup() {

Serial.begin(9600);

// Initialize RTC and LCD

rtc.begin();

lcd.init();

// Set the initial time (adjust as needed)

rtc.initializeTime(2024, 3, 4, 12, 0, 0);

}

void loop() {

// Variables to store time components

int year, month, day, hour, minute, second;

// Read the current time from RTC

rtc.readTime(year, month, day, hour, minute, second);

// Display the current time on the LCD

lcd.clear();

lcd.setCursor(0, 0);

lcd.lcdPrint("Current time:");

lcd.setCursor(0, 1);

lcd.lcdPrint(String(year) + "-" + String(month) + "-" + String(day) + " " + String(hour) + ":" + String(minute) + ":" + String(second));

// Delay for 1 second before updating the display again

delay(1000);

}

## Explanation

* Include Libraries: The code includes necessary libraries for Arduino, LCD, and RTC.
* Pin Definitions: Defines pins for connecting the LCD and initializes instances of the LCD and RTC.
* Setup Function: Initializes the serial communication, RTC, and LCD. It also sets the initial time on the RTC.
* Loop Function: Repeatedly reads the current time from the RTC, clears the LCD, and displays the current time. It then adds a delay of 1 second before updating the display again.
* Note: This code assumes the existence of custom libraries for the LCD (lcd.h and lcd.cpp) and the RTC (rtc.h and rtc.cpp). The actual functionality of these libraries is not provided in the code snippet.

# Outputs

After the compiling code successfully, the next step is to check the results. In this regard, circuit is designed in proteus and the HEX file is uploaded. The designed project runs perfectly i.e. according to our requirements. When the simulation starts, the time can be set using header file of RTC than it updates the time accordingly.

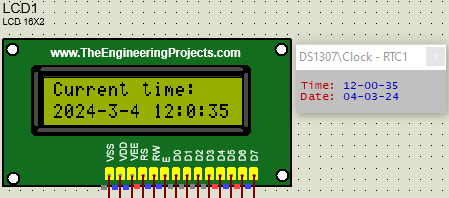


Figure 6 Tested Outputs

# Conclusion

In conclusion, this project clearly states the use of Hardware Abstraction Layer (HAL) that combines a real-time clock (RTC) module and a liquid crystal display (LCD) to create a functional timekeeping system. The code establishes seamless communication between the Arduino microcontroller, RTC, and LCD, laying the groundwork for an intuitive and visually engaging interface. The Hardware Abstraction Layer (HAL) implementation streamlines the integration process, protecting the programmer from the intense hardware details and allowing for a more straightforward and modular approach to code development. The setup function, intelligently designed, initializes the system by configuring the RTC and LCD, and sets the initial time, providing a foundation for accurate timekeeping. The loop function continuously reads the current time from the RTC, updates the information on the LCD screen, and introduces a one-second delay, creating a visually appealing and dynamically changing display. The use of object-oriented principles, evident in the instantiation of classes for both the LCD and RTC, contributes to a more organized and efficient codebase.

# References

1. Yoo, S., Jerraya, A.A. (2003). Introduction to Hardware Abstraction Layers for SoC. In: Jerraya, A.A., Yoo, S., Verkest, D., Wehn, N. (eds) Embedded Software for SoC. Springer, Boston, MA. https://doi.org/10.1007/0-306-48709-8\_14
2. R. A. Dinda, S. Sadrina, and M. Mursyidin, “The High Accurate Automatic School Bell Controller Based On Arduino Uno DS1307 I2C Real-time Clock”, *JTMMX*, vol. 4, no. 1, pp. 17-26, Jul. 2023.
3. Perenc I, Jaworski T, Duch P. Teaching programming using dedicated Arduino Educational Board. *Comput Appl Eng Educ*. 2019; 27: 943–954. <https://doi.org/10.1002/cae.22134>
4. Atmel (now Microchip Technology), "Atmel-7810-Automotive-Microcontrollers-ATmega328P\_Datasheet," <https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf>, accessed March 4, 2024.
5. Vishay Intertechnology, Inc., "LCD016N002BCFHE/T Datasheet," <https://www.vishay.com/docs/37484/lcd016n002bcfhet.pdf>, accessed March 4, 2024.
6. SparkFun Electronics, "DS1307 Real Time Clock Breakout Board," <https://www.sparkfun.com/datasheets/Components/DS1307.pdf>, accessed March 4, 2024.