MAKERSPACE 5.0 : DIGITAL CLOCK

INTRODUCTION

This Project involves using an ATMega32 Microcontroller, simulated on Proteus software and coded using ATMEL Studio in Embedded C.

We got a rough idea that the project will be divided into 3 main components. RTC, Microcontroller, and LCD. We will have to interface each of these components. Each component will have its separate code and procedure. We later realized that all this will be available in the Official ATMEL Datasheet.

Communication Protocols

Communication protocols are the procedures using which IC's communicate with each other. They are predefined procedures which are given/programmed by the manufacturer, which the programmer has to keep in mind while using the protocol. There are some basic protocols like , UART, USART and I2C.

We'll be using I2C for our project. Source File from where we read I2C Protocol in depth is https://www.electronicshub.org/basics-i2c-communication/

LED Blinking and Basic Circuit Design.

To get a feel for Proteus we were asked to Simulate Basic Logic gates. Then we started the LED Blinking Project. We used whole Youtube Tutorial to Learn Embedded C Programming that helped us until end of project.

Link : https://youtube.com/playlist?list=PLE72E4CFE73BD1DE1

CODE :

```
#define F_CPU 8000000UL

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
```

```
DDRD=0b00000011;
while (1)
{
{
PORTD=0b0000001;
_delay_ms(10);
PORTD=0b0000000;
_delay_ms(10);
}
{
PORTD=0b00000010;
_delay_ms(100);
PORTD=0b00000000;
_delay_ms(100);

}
}
}
```

I2C Protocol:

- I2C communication is the short form for inter-integrated circuits.
- This is a type of synchronous serial communication protocol. It means that data bits are transferred one by one at regular intervals of time set by a reference clock line.
- Only two common bus lines (wires) are required to control any device/IC on the I2C network.
- Uses 7-bit addressing system to target a specific device/IC on the I2C bus.

- I2C Bus (Interface wires) consists of just two wires and are named as Serial Clock Line (SCL) and Serial Data Line (SDA). The data to be transferred is sent through the SDA wire and is synchronized with the clock signal from SCL.
- Any device/IC on the I2C network can drive SDA and SCL low, but they cannot drive them high. So, a pull up resistor is used for each bus line, to keep them high (at positive voltage) by default.

Source File from where we read I2C Protocol in depth is
https://www.electronicshub.org/basics-i2c-communication/

Master and Slave Devices:

- The devices connected to the I2C bus are categorized as either masters or slaves. At any instant of time only a single master stays active on the I2C bus. It controls the SCL clock line and decides what operation is to be done on the SDA data line.
- In our project Master is microcontroller and other devices (RTC and LCD) act as slaves.
- All the devices that respond to instructions from this master device are slaves. For differentiating between multiple slave devices connected to the same I2C bus, each slave device is physically assigned a permanent 7-bit address.

Data Transfer Protocol:

- The following protocol (set of rules) is followed by master device and slave devices for the transfer of data between them.
- Data is transferred between the master device and slave devices through a single SDA data line, via patterned sequences of 0's and 1's (bits).
- Data is transferred between master and slave by the following steps:
- Start Condition: Whenever a master device/IC decides to start a transaction, it switches the SDA line from high voltage level to a low voltage level before the SCL line switches from high to low.
- Address Block: It comprises of 7 bits and are filled with the address of slave device to/from which the master device needs send/receive data. All the slave devices on the I2C bus compare these address bits with their address.
- Read/Write Bit: This bit specifies the direction of data transfer. If the master device/IC need to send data to a slave device, this bit is set to '0'. If the master IC needs to receive data from the slave device, it is set to '1'.
- ACN/NACN Bit: It stands for Acknowledged/Not-Acknowledged bit. If the physical address of any slave device coincides with the address broadcasted by the master device, the value of this bit is set to '0' by the slave device. Otherwise, it remains at logic '1'.

- Data Block: It comprises of 8 bits and they are set by the sender, with the data bits it needs to transfer to the receiver. This block is followed by an ACK/NACK bit and is set to '0' by the receiver if it successfully receives data. Otherwise, it stays at logic '1'. This combination of data block followed by ACK/NACK bit is repeated until the data is completely transferred.
- Stop Condition: After required data blocks are transferred through the SDA line, the master device switches the SDA line from low voltage level to high voltage level before the SCL line switches from high to low.

RTC: A real-time clock (RTC) is a computer clock, usually in the form of an integrated circuit that is solely built for keeping time. Naturally, it counts hours, minutes, seconds, months, days and even years. RTCs can be found running in personal computers, embedded systems and servers, and are present in any electronic device that may require accurate time keeping.

In our Project, We will be using DS1307 Module of RTC. The main advantage of using RTCs is that even If we turn off the incoming Power from Micro-Controller Unit, It will keep updating Time in Data register of RTC. So, When we restart it, It will show us the real time again. In the circuit of RTC we also add oscillator Capacitor. That helps in frequency balancing and reducing the Noise due to load unbalancing or environmental factors such as temperature, pressure etc. If the Crystal Capacitor is absent then it will increase the clock's working speed. While coding we initiate the I2C Protocol within RTC Module in Master Transmission Mode and then change it to Master Receiving mode for receiving an Ack or NAck bit. Then we make attempts to display our data.

Link to DS1307 DataSheet : https://datasheets.maximintegrated.com/en/ds/DS1307.pdf

Registers : Referring through Atmega 32 Datasheet we read in detail functions of various Registers. This Registers were implemented in our code while initiating I2C Protocol in our Micro-Controller Unit. The short Description of registers goes as below :

1.) TWBR (BitRate Register)

Basically it is a frequency divider which generates the SCL Clock Frequency in Master Modes.

2.) TWCR (Control Register)

It is used to enable Two Wire Interaction by initiating Master Access on applying the START Condition. We have TWINT (Interrupt Flag), TWEA (Enable Acknowledgement), TWSTA(Start Condition), TWSTO (Stop Condition) registers alloted to pin bits as required.

3.) TWSR (Status Register)

There are various Status Codes in Hexademicals for Master Transmission and Reciever both modes like ($08 for Starting I2C Communication , $18 signifying Ack has been received. Etc)

So we add Prescaler bits and mask them using Shift operators and set those values within Status Register.

4.) TWDR (Data Register)

While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave.

Reference : AtMega32 DataSheet. Official ATMEL Datasheet

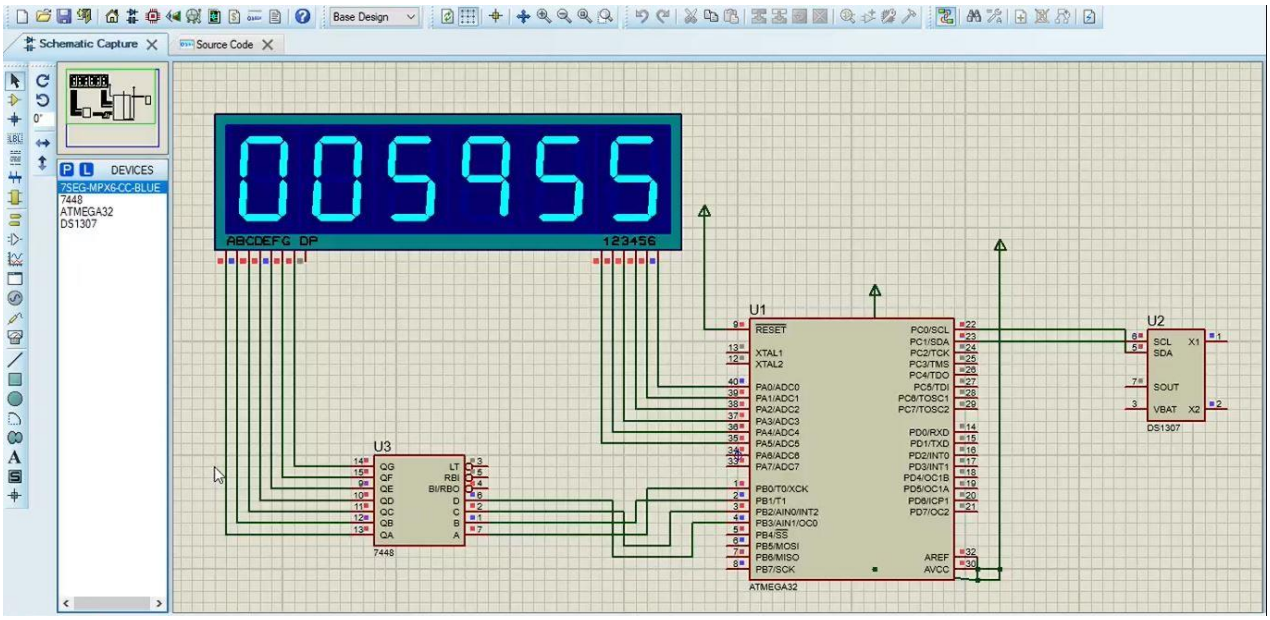
LCD :- We firstly thought, we may use LCD in our project to display time from RTC. But while coding, We encountered some issues where we were supposed to convert the BCD(Binary Coded Data) to Decimal. So, On our LCD Screen we were getting few garbage value and It didn't worked for our project. We than made attempts to switch over to 7 Segment LED Display with 6 panels. We used Decoder to convert BCD data to Decimal.

7 Segment LED & Decoder : It is known as 7 segment LED because, it consists of 7 Partitions arranged horizontally and vertically collectively. We can use combinometrics and make it to display numbers 0 to 9 and Alphabets A to G. It works in two Modes, Common Cathode and Common Anode Mode. We set Logic as high and low and correspondingly the Circuits will choose whether it is in forward biasing or reverse biasing and thus respective segments will lit up to form the character. Now here we attach a Decoder in between Micro-controller and LED Display. The main purpose of Decoder is to convert the BCD(Binary Coded Decimal) Data coming from RTC to be decoded into normal decimal format so we can show time on our screen in readable format.

7 Segment LED Display : https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html

Decoder : https://www.electronics-tutorials.ws/combination/comb_6.html

Final Circuit Diagram :



Github Repository Link : https://github.com/Harshil-Jani/Makerspace-Digital-Clock/