Lab7

Negin Kheirmand

9831023

This week's lab is about working with EEPROM and making a washing machine's brain along the way.

IC & Modules I used:

- Arduino Nano
- EEPROM (ATMEL721P 24C02B PU27 D)
- 4 LEDs
- Keypad (used only the first 2 rows of my 4 row keypad)
- LCD (character display)
- some jumper wires & resistors(4 22Oh resistors)

Codes for the Lab:

1. first Part: sketch_Test_EEPROM (this is only a sketch to test the EEPROM and see if it works properly)
2. second part: sketch_setDefault_memory this sketch sets the default values of the memory
3. third part: sketch_WashingMachine this is the actual code for the washing machine

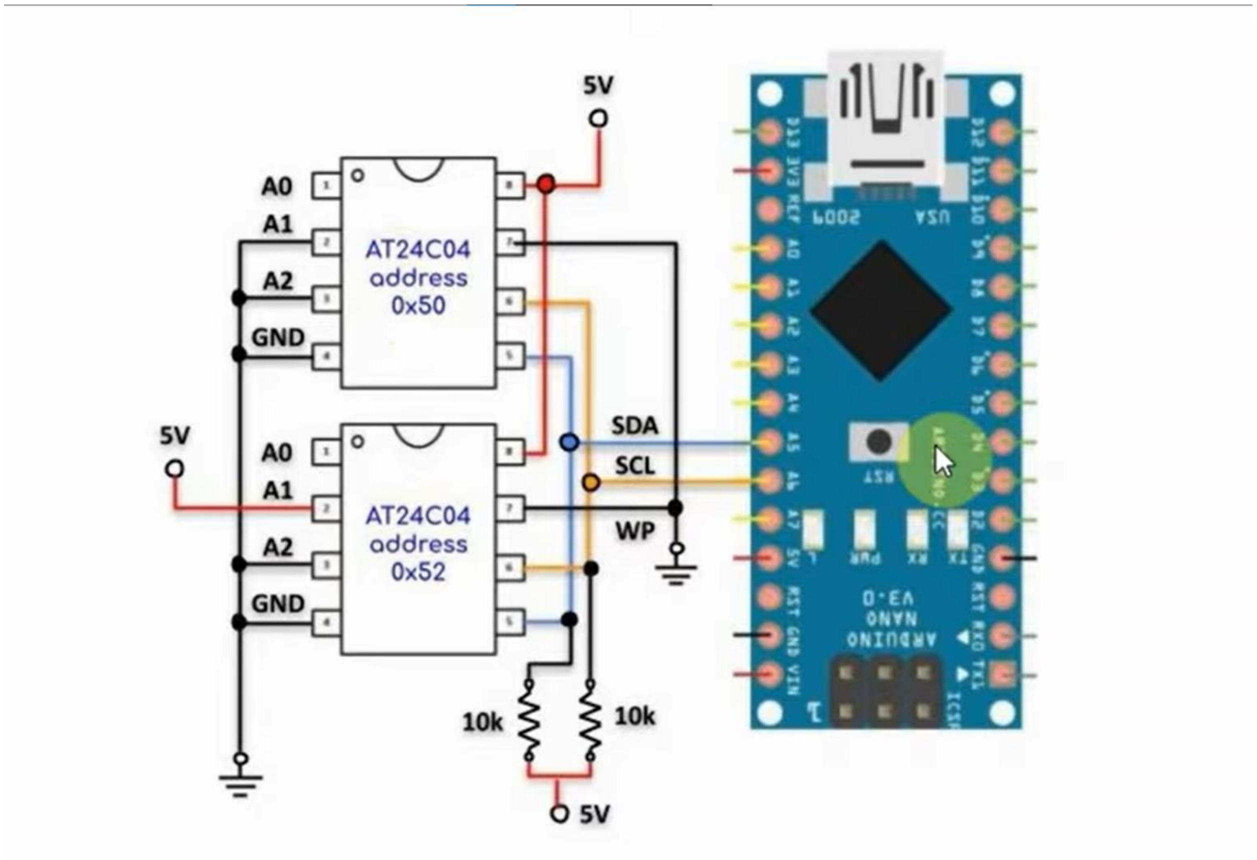
What I learnt in this experiments:

- **What EEPROM are and how they work**
    - **sometimes they don't work properly (idk know if this was because of my EEPROM or something I did wrong –probably the latter-)**
    - **writing data into them is limited**
- **you cant connect digitally working modules to the A6 and A7 pins of the Arduino Nano since they are analog input only**

Questions:

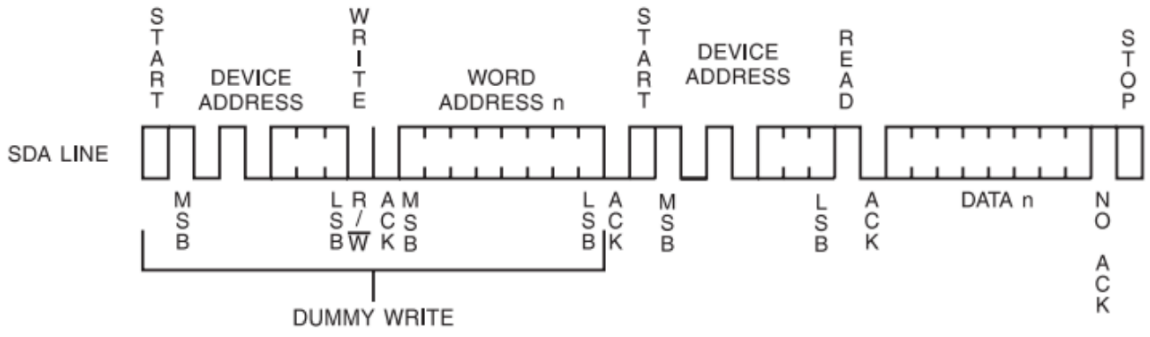
- use of EEPROMS:
    - They are used to store data (most of the parameters required for initial configuration) even when the device is turned off, as we saw in this experiment.
- Flash vs EEPROM:
    - Flash is just one type of EEPROM.
    - Flash uses NAND-type memory, while EEPROM uses NOR type.
    - Flash is block-wise erasable, while EEPROM is byte-wise erasable.
    - Flash is constantly rewritten, while other EEPROMs are seldom rewritten.
    - Flash is used when large amounts are needed, while EEPROM is used when only small amounts are needed.
    - In conclusion: they are both one thing. When flash came on the market, EEPROMs could both read and write information. The only difference was speed and quantity. EEPROMs had a moderate speed in reading and writing and did this byte by byte, but flash did the same thing with chunk / block of data, which was larger in size, and of course writing it was much faster than EEPROM. But in removing or deleting data, it was much slower than EEPROM so FLASH memory is much closer in function to an EEPROM (Electrically Erasable Programmable Read-Only Memory), with the key difference being that FLASH can be easily erased and rewritten without having to go through the electrical contortions that EEPROMs require. FLASH can also (usually) withstand a much higher number of write cycles before it starts to deteriorate.

- EEPROM vs ram:
    - EPROMs are preprogrammed modules which can be erased and reloaded by a special prommer device. They replace ROM modules but are more flexible for the vendor. EPROMs are like modern flash memories but cannot be reprogrammed from the HP71 itself.
    - RAM is just memory for programs, data and files.
    - In conclusion RAM (Random Access Memory) is typically volatile, meaning that if the power goes away so does the memory's contents. The big advantage with RAM is that the access time for read and write is far faster than any EPROM or EEPROM.
- Why to use EEPROM instead of ram in this lab? Because RAMs data is volatile meaning after turning off the custom configuration would have been deleted
- Why to use EEPROM instead of flash in this lab? Since the EEPROM is better editable in comparison to flash it would be a better option since the user can later change and edit the custom configuration
- What should have been done if the flash memory was used (for editing/writing) is:
    - 1. Read a block/chunk of data
    - 2. Change the required part of it
    - 3. Write the block again into the memory (as you can see we are dealing with a lot more data than needed, all this is time consuming when compared to EEPROM)
- If the EEPROM has 4KB of storage and we have 2 address ports how much data could be stored?
    - 2^2=4 so we can have 4 external EEPROM modules and for each of them we can store up to 4KB of data, in conclusion a total of 4*4=16KB data could be stored externally.
- Show the schematic of a circuit composed out of an Arduino and 2 external EEPROMS:

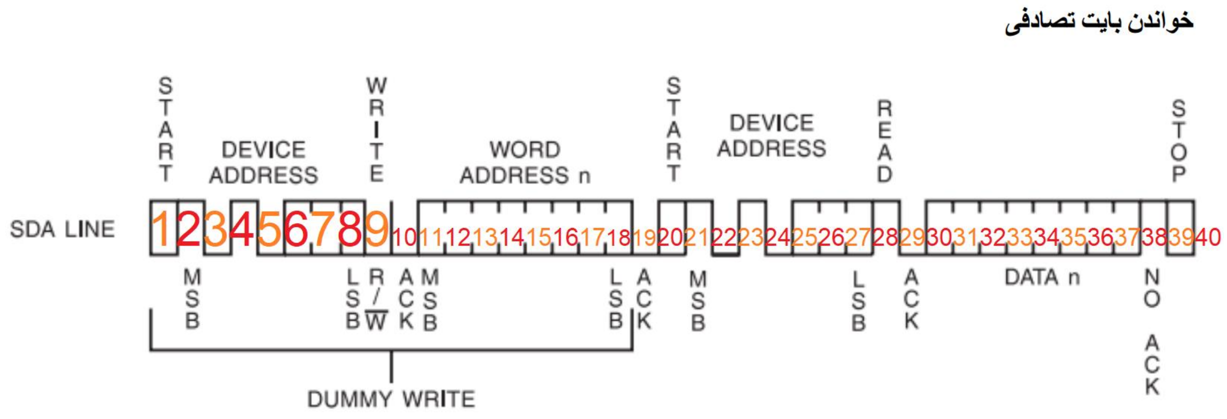


    - Since its specified that the write protection (WP) is inactive the WP pin should be connected to GND.
    - The addresses for each of the EEPROMs(0x50 and 0x52) are created because of the way the A1 and A2 signals of them are connected to the GND and VCC.
- The similarity between the diagram for the communication between this module and Arduino and SPI (TWI)
    - They are really similar:

- In SPI the master first sends an START BIT and then the slave address and then the form of communication (read/write), after that the data (can be 5 or 9 bits –with or without parity bit-) and at the end the STOP BIT.
- In the communication above we can see that the only thing different is the address of the word (meaning where the word received should be saved) and other than that they are completely equal.



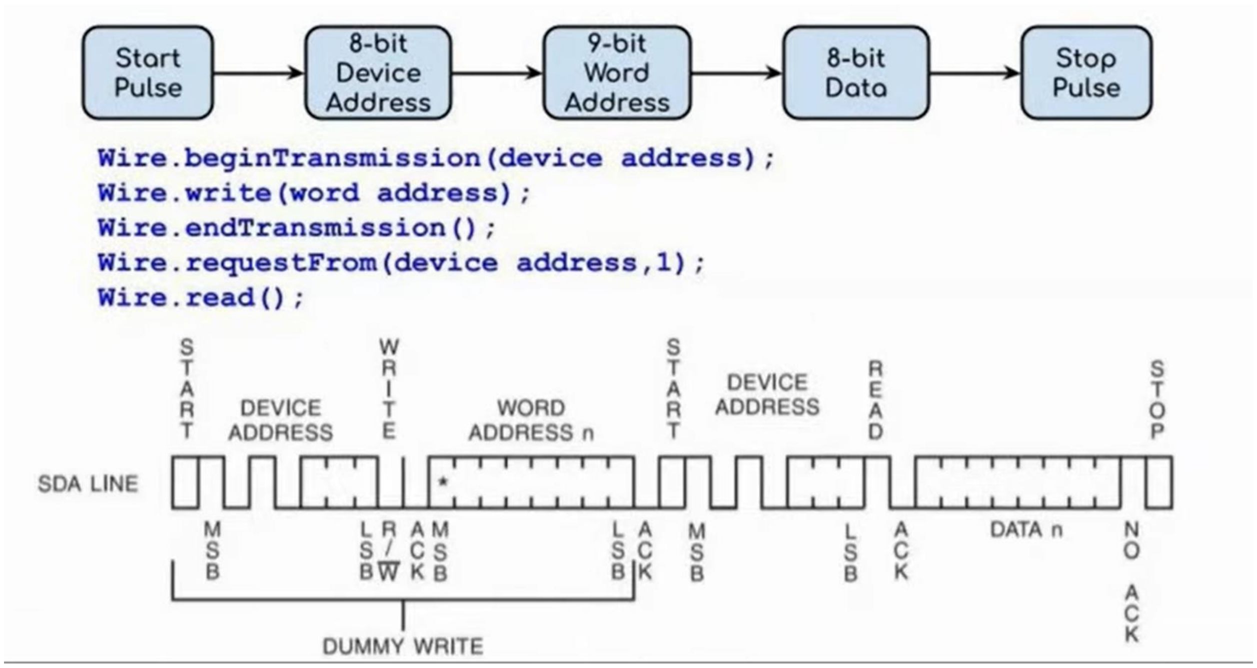
- Where does the frequency of the clock come from:
    - From the master serial clock pin and for the master that value comes from the Power management module
- If a clock with a frequency of 10KHz is used then the period would be 0.0001s and since for each package of data is needed (roughly) 40 bits, then that means a package can be sent every 0.004 seconds meaning it has the frequency of 250Hz.



- How to Read and write data using the functions specified in the lab documentation(these are the same functions I used in my code so you can reference the readI2CByte and the writeI2CByte code also):

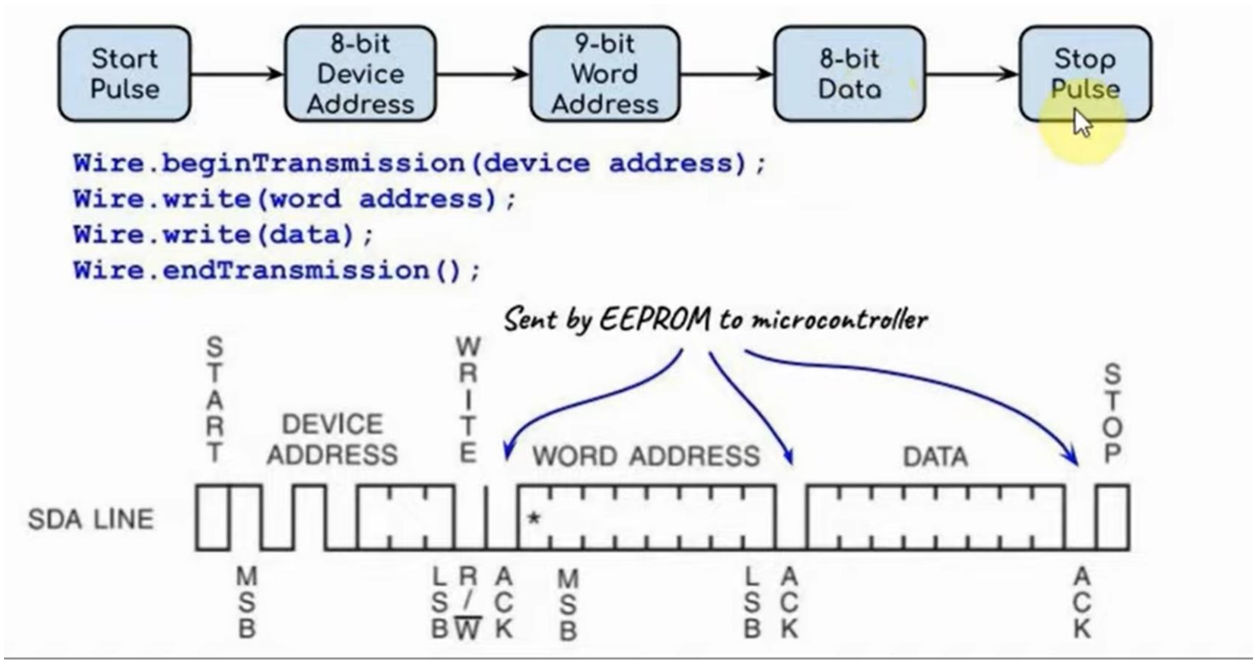
```
void writeI2CByte(byte data_addr, byte data){
  Wire.beginTransmission(ADDR);
  delay(10);
  Wire.write(data_addr);
  delay(10);
  Wire.write(data);
  delay(10);
  Wire.endTransmission();
}

byte readI2CByte(byte data_addr){
  byte data = NULL;
  Wire.beginTransmission(ADDR);
  delay(10);
  Wire.write(data_addr);
  delay(10);
  Wire.endTransmission();
  delay(10);
  Wire.requestFrom(ADDR, 1); //retrieve 1 returned byte
  delay(10);
  if(Wire.available()){
    data = Wire.read();
    delay(10);
  }
  return data;
}
```

read:



```
Wire.beginTransmission(device address);
Wire.write(word address);
Wire.endTransmission();
Wire.requestFrom(device address,1);
Wire.read();
```

write:



```
Wire.beginTransmission(device address);
Wire.write(word address);
Wire.write(data);
Wire.endTransmission();
```