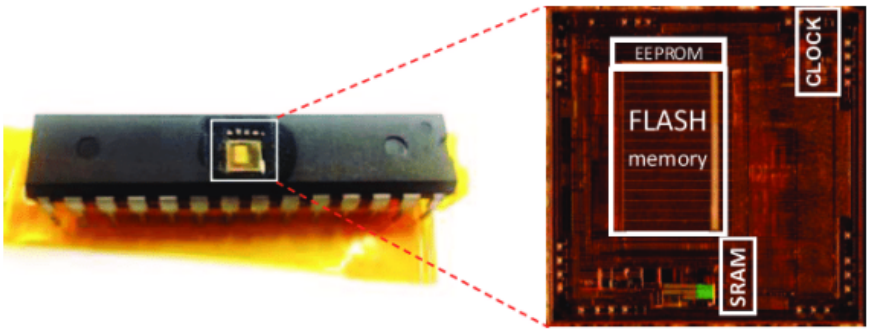
Marwan Ibrahim (msi9696)
Ahmed El Said (ae2200)

Lab 9: Reading and Writing Data from EEPROM

Introduction:
EEPROM is Electrically Erasable Programmable Read-Only Memory. It is non-volatile type of memory as it holds the data even when power is off. The main advantage of this memory is that controller can read, modify/write this memory in runtime application. So EEPROM can be used for storing sensor values, important parameters etc. with no fear of loss even in case of power failure.

The Atmega328p contains 1Kbytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. Normally EEPROM has limited life span. AVR ATmega328P EEPROM has endurance of 100,000 write/erase cycle. We need to take care about while writig EEPROM that not to put EEPROM write operation in continuous loop. Note that it has only limit for write/erase cycle, there is no limit for read operation. Access over EEPROM memory is made through three registers.

- **EEAR** (EEPROM Address register)
- **EEDR** (EEPROM Data register)
- **EECR** (EEPROM Control register)



Hardware:
1) ATMega328p (on Arduino UNO board)
2) Atmel ICE 3.0 programmer/debugger
3) Breadboard
4) Jumper wires
5) USB cables

Goals:
- Write and Retrieve data by writing a sample text into EEPROM memory of Atmega328p using UART interface.

Schematic & Pinout:

Lab 9: Reading and Writing Data from EEPROM

Marwan Ibrahim (msi9696)
Ahmed El Said (ae2200)

No special schematic was used beyond connecting the Arduino Board and the Atmel ICE programmer to the computer

Registers:

- EEAR: Holds the EEPROM address for read/write operations.
- EEDR: Stores the data to be written to, or read from, EEPROM.
- EECR: Controls the EEPROM read/write operations with flags for enabling operations and write protection.

PseudoCode:

- Initialize UART communication settings.
- Define a string "Hello World" to write to EEPROM.
- Write the string to EEPROM starting from address 0.
- Read back the string from EEPROM into a buffer.
- Send the string over UART to display on the terminal.

Code:

```
/*
 * main.c
 *
 * Created: 12/8/2023 9:32:24 PM
 *  Author: Marwan & Ahmed
 */

#define F_CPU 16000000UL  // Define the CPU frequency for delay calculations

#include <avr/io.h>       // Include AVR input/output header for IO port access
#include <avr/eeprom.h>   // Include AVR EEPROM header for EEPROM functionality

#define BAUD 9600         // Define baud rate for UART
#define BRC ((F_CPU/16/BAUD) - 1)  // Calculate baud rate register value

// Initialize UART for serial communication
void UART_init() {
        UBRR0H = (BRC >> 8);        // Set upper part of the baud rate
        UBRR0L = BRC;               // Set lower part of the baud rate

        UCSR0B = (1 << TXEN0) | (1 << RXEN0);  // Enable transmitter and receiver
        UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);  // Set frame: 8data, 1 stop bit
}

// Transmit a single character over UART
void UART_transmit(char data) {
        while (!(UCSR0A & (1 << UDRE0))); // Wait for empty transmit buffer
```

```
	UDR0 = data; // Put data into buffer, sends the data
}

// Send a string over UART
void UART_Send_String(char* str) {
	int i = 0;
	while (str[i]) {
		UART_transmit(str[i]); // Transmit each character of the string
		i++;
	}
}

// Write a byte to a specific EEPROM address
void EEPROM_Write(uint16_t uiAddress, uint8_t ucData) {
	while(EECR & (1<<EEPE)); // Wait for completion of previous write
	EEAR = uiAddress;        // Set up address register
	EEDR = ucData;           // Set up data register
	EECR |= (1<<EEMPE);      /// Enable EEPROM master write enable
	EECR |= (1<<EEPE);       // Start EEPROM write
}

// Read a byte from a specific EEPROM address
uint8_t EEPROM_Read(uint16_t uiAddress) {
	while(EECR & (1<<EEPE)); // Wait for completion of previous write
	EEAR = uiAddress;        // Set up address register
	EECR |= (1<<EERE);       // Start EEPROM read
	return EEDR;             // Return data from data register
}

// Write a string to EEPROM starting from a specific address
void EEPROM_Write_String(uint16_t uiAddress, const char* string) {
	// Write each character to EEPROM
	for (uint16_t i = 0; string[i] != '\0'; ++i) {
		EEPROM_Write(uiAddress + i, string[i]); // Write each character
	}
}

// Read a string from EEPROM starting from a specific address
void EEPROM_Read_String(uint16_t uiAddress, char* buffer, uint16_t length) {
	for (uint16_t i = 0; i < length; ++i) {
		buffer[i] = EEPROM_Read(uiAddress + i); // Read each character
	}
	buffer[length] = '\0'; // Null-terminate the string
}
```

Marwan Ibrahim (msi9696)
Ahmed El Said (ae2200)

```
int main(void) {
	UART_init(); // Initialize UART communication

	char data_to_write[] = "Hello World"; // Data to write to EEPROM
	char read_data[50]; // Buffer to store read string

	 // Write to EEPROM
	 EEPROM_Write_String(0, data_to_write);

	 // Read from EEPROM
	 EEPROM_Read_String(0, read_data, sizeof(data_to_write));

	 // Send read_data over UART
	 UART_Send_String(read_data);

	while (1) {
	}
	return 0;
}
```