EPROM PROGRAMMER

Submitted by: Group 73

Group Members
Kartik Jain
Malika Chhibber
Kunal Seernani
Yadav Soham B
Kushagra Motiani
Sarvesh Shinde

Date: 15/4/2021

Contents

User Requirements & Technical Specifications

Design a microprocessor-based EPROM Programmer to program 2732.

1. While programming, a 25V (write voltage) should be applied to the VPP, and 5V at the OE pin of EPROM.
2. All locations should be empty initially (i.e. each location is FFh), in order to ensure that the EPROM is programmable.
3. If a location is non-empty, the LED labelled “not empty” must glow.
4. A set of seven segment displays must be used to display the location (which is to be programmed).
5. A keypad containing all hex digits (0-F), backspace and enter must be used by the user to input data.
6. 8-bit data should be provided on the data lines of EPROM.
7. The address provided to the EPROM must be stable for 55 ms.
8. A 50ms active high pulse must be applied to the CE input when address and data are stable.
9. The data entered in the location address should be echoed back to the user to ensure whether the programming was successful.
10. If the programming was unsuccessful at any given location, the LED labelled at fail must glow.
11. The above process must be repeated for all locations. If all the locations are successfully programmed, an LED labelled “programming completed” must glow.

Assumptions & Justifications

1. The EPROM is programmed sequentially starting from 0000h.
2. Only one key can be pressed at a time.
3. The first digit that the user enters should be taken as the upper nibble, the second digit is taken in the lower nibble.
4. After the first keypress(at a particular location), if the user presses enter then it is ignored.
5. Only backspace and enter are considered as key presses after the user has entered two digits at a given location. All other keys are ignored.
6. The EPROM will be programmed after the user enters two hexadecimal digits and presses enter.
7. If the user presses backspace without entering any hexadecimal digit then it is ignored.
8. When the user presses backspace only the last entered digit is deleted and the location is made empty(fh).
9. The address of the location to be programmed must be stable for 55ms in order for it to not be considered as an accidental press, thus we need to use a software delay to ensure the same.

Components used with justification wherever required

1. **8284** - Clock Generator - input to 8086 (Reset, Clk, Ready)
2. **8086** - Microprocessor
3. **8255**- 3 Nos.

   8255(1) - It is interfaced with the EPROM, LEDs and 3*6 Keypad.

   8255(2) - It is interfaced with the EPROM. It is connected to address lines, data lines, chip enable of the EPROM and the relay switch.

   8255(3) - It is interfaced with the 6 7 segment displays.

4. **2732 –** EPROM (Manual Attached) - to be programmed in the design
5. **Relay** - EPROM requires a 25V on the Vpp pin when it is programmed and 5V on the Vpp pin when in reading mode. The Vpp pin is connected to a DC source of 25V or 5V through the Relay.
6. **Common anode LEDs –** 3 Nos.(3 different colors) **–** to indicate 3 states
   - not empty - yellow
   - failed programming - red
   - successful programming - green
7. **Custom 3x6 hex keypad with enter and backspace key** - Total 18 keys (0 to F and enter and backspace)
8. **Common Anode 7 Segment Display –** 6 Nos - 4 for the 16-bit address and 2 for the 8-bit data entered by the used.
9. **14495 – Hexadecimal to Common Anode 7 - Segment converter** (Manual Attached) **–** As values to be displayed are hexadecimal.
10. **2716 – 2K * 8 ROM Chip** - 4 Nos (2 for even bank, 2 for odd bank) .Need to store data in 2 locations. 1 ROM is required at reset address which is at FFFF0H and another at 00000H - where the IVT is stored. Since we have not used any interrupts in this design, the first ROM is added only as per convention.
11. **6116 – 2k * 8 RAM Chip** - 2 nos. Smallest RAM chip available is 2 K and we need to have an odd and an even bank. We need RAM for stack and temporary storage of data.
12. **LS 138 – 3:8 decoders** - 2 nos - 1 decoder for memory selection ROMs and RAM and 1 decoder for I/O selection i.e. between the 3 8255s
13. **LS 373** - **Octal Latch**
14. **LS 245** - **Octal Tri-State Buffer**
15. **LS 244** - **Octal Tri-State Transceiver**
16. **Not Gate**
17. **Or Gate**

Address Map

Memory Map

ROM1 – 00000_H – $0FFF_H$

RAM 1– 02000_H – $02FFF_H$

ROM2 – $FF000_H$ – $FFFFF_H$

I/O Map

8255 (1) – 50 – 56_H

8255 (2) – 60 – 66_H

8259 (3) – 70 – 76_H

Design

Attached as pdf

Flowchart

Attached as a pdf

Variations in Proteus Implementation with Justification

1. We have used the internal clock of 8086 as 8284 is not available in proteus.
2. We have used 2732 chips instead of 2716 chips as 2716 chips are not available in proteus.
3. An additional 8255 (Hexadecimal to Common Anode 7 - Segment converter) was used in proteus because of the absence of IC14495. The address of the 4th 8255 was 80h to 86h.
4. A separate code for converting hexadecimal digits to seven segments is written in the asm file as the IC 14495 is not available in proteus.
5. Relay unit wasn't used in the proteus simulation since it can't be altered in simulation.

Firmware

Implemented using emu8086 attached.

List of Attachments

1. Hardware Design - design.pdf
2. Flowchart - flowchart.pdf
3. Proteus File - emprom_grp73_prot.dsn
4. EMU8086 8086 ASM File - eprom_grp73_code.asm (Proteus Simulation)
5. Binary file after compiling in emu - eprom_grp73_emu.bin
6. Relevant Manuals
    a. M2732A (UV EPROM)
    b. IDT6116 (RAM)
    c. 14495 (Hexadecimal to 7 segment LED)
    d. SN74LS138 (3:8 decoder)
    e. SN74LS244 (Octal buffers with 3 state outputs)
    f. SN74LS245DW (Octal Bus Transceivers With 3-State Outputs)
    g. SN74LS373DW (Octal transparent D-type latches with 3-state Outputs)