

Low Cost EPROM Programmer

Tareq Hasan Khan, Nayeem Ahmed Ninad[†], Md. Hasanul Kabir[‡]

Student, ECE Department, Concordia University, Montreal, Quebec, Canada.

[†] Lecturer, EEE Department, Islamic University of Technology, Gazipur, Dhaka, Bangladesh.

[‡] Student, Department of Computer Engineering, Kyung Hee University, Kyunggi-doo, South Korea.

Tareq_992403@yahoo.com, ninad@iut-dhaka.edu, hasanul@gmail.com

Abstract

A microprocessor-based system consists of Micro-processor, ROM, RAM, Ports and other peripheral devices. Microprocessor fetches instruction and code from ROM and executes according to the instruction. So, to create a microprocessor-based system, we must write the machine code and data in the ROM. EPROMs (Erasable Programmable Read Only Memory) are widely used as ROM. To feed data in EPROM, EPROM Programmers are used. But due to high cost, EPROM Programmers are difficult to manage. This paper shows a way to construct a Low Cost PC Based EPROM Programmer.

Keywords: Data grabbing, Fetching instruction, Handshaking, Hardware Signaling, Machine code.

I. INTRODUCTION

An EPROM Programmer is a device used to feed data and code in an EPROM [1,6]. It is an undeniable and significant requirement for microprocessor-based project. Although EPROMs are highly available, EPROM programmers are extremely rare. Even though it is obtainable through ordering from outside the country, it is still extremely expensive. So we decided to develop an EPROM programmer ourselves and at some point, we managed to come up with a working EPROM programmer prototype. This significantly cut down our expenses since the amount of money that we spent for our EPROM programming is almost nine times cheaper than those available for ordering.

II. CONSTRUCTION OF EPROM PROGRAMMER

The Machine Code of the Program must be downloaded in the EPROM for a microprocessor-based system to work. So we need an EPROM Programmer, which will download our machine code in the EPROM from the PC via PC Ports.

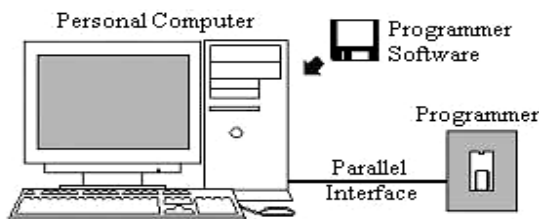


Fig 1: PC based EPROM Programmer

A. Hardware Part

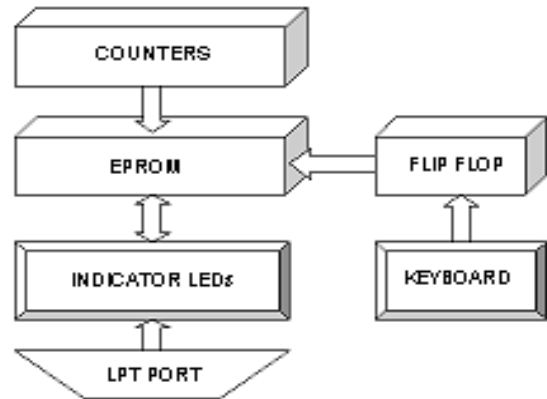


Fig 2: Block Diagram of EPROM Programmer

In the Hardware part, the address pins of the EPROM are connected with binary counters and counters get their clock pulse from a timer circuit. The 8 data pins of the EPROM are connected with the LPT port output pins. There are also 8 indicator LEDs to see the content of the EPROM. To manage the EPROM Read, Write, Chip Enable (CE), Output Enable (OE) and other signaling, Flip-flops with a small keyboard is used. Using this keyboard we can select different modes of operation.

B. Software Part

The software portion is written in C. This Program grabs the hex data from the file "eprom.dat" and sends the data to the LPT port in binary form.

B.1 Format of eprom.dat file

```
<8 bit Hex code>
<8 bit Hex code>
.
.
<8 bit Hex code>
xx
```

B.2 Example

```
3F
5E
E2
77
xx
```

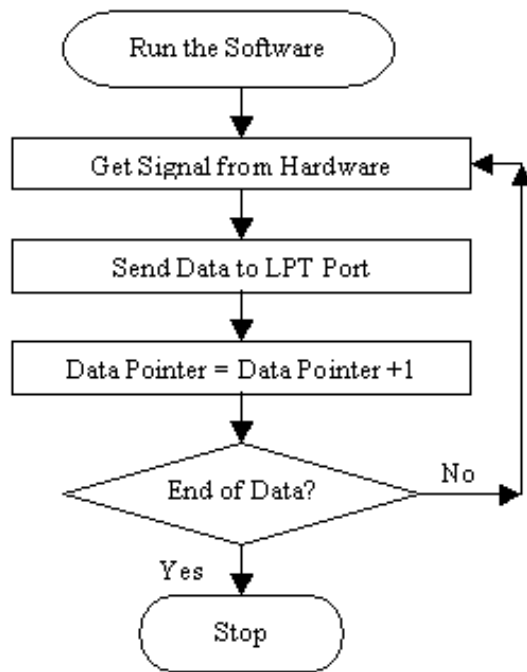


Fig 3: Flowchart of the Software

When we run the software it waits until it gets a signal from the hardware. When it gets the signal from the hardware, that is when the programmer is “Turned on”, the software starts grabbing data from the “eprom.dat” file and send it to the LPT port. In the next cycle again the hardware counter increases by 1, software gets signal from the hardware and sends the next data from the “eprom.dat” file to LPT port. This process goes on until it reaches the end of the “eprom.dat” file.

III. MODES OF OPERATION

A. Read Mode

Our EPROM Programmer is able to read the data of any EPROM (27Cxxx series). The Data are shown in binary by glowing the Data Indicator LEDs.

B. Programming Mode

To download data in the EPROM, we chose this mode. In this mode our Hardware interacts with the software portion via the LPT port of the PC. At first the data, which is to be downloaded in the EPROM, is saved in the “eprom.dat” file in hex format. Then we must run the software. When the software is ready it will display message to “Turn on The EPROM Programmer Now...”. When it is turned on, a two-way communication (handshaking) [5,6] gets established between the EPROM Programmer and the Software. The Software sends data from the “eprom.dat” file to the LPT port and the hardware grabs it and writes the data in the

proper address of the EPROM. When programming is completed it shows the finishing message “Done!!! Disable The EPROM Now...” Then we can disable the EPROM.

C. Programming From the Address 00000H Mode

In this mode data is downloaded in the EPROM from its starting address 00000H.

D. Programming From the Address FFFF0H Mode

In this mode data starts downloading form the address FFFF0H of the EPROM. This address is important because the 8086 [5,6] microprocessor starts fetching instruction from the address FFFF0H. So to write the boot up program (Monitor program) we switch to this mode. It helps EPROM programming much easier

IV. FLEXIBILITY

In this design we can program EPROM of any size. The more the size, we have to add more counters with the address pins of the EPROM.

If we use N bit counter then the number of N bit counters say n, required to program an EPROM of size M x 8 is

$$n = \text{ceil} ((\log_2 M)/N) \quad (1)$$

where,

ceil(x) function returns
the smallest integer not < x

The programming voltage of EPROM known as Vpp varies from EPROM to EPROM. So we can make it adjustable by making a voltage divider circuit using variable resistor.

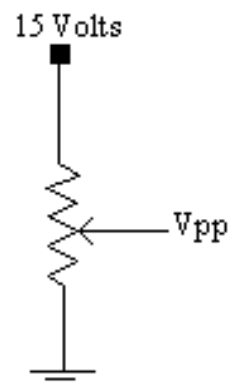


Fig 4: Adjustable Vpp Using Voltage Divider Rule

V. CONCLUSION

At the end we can say that our designed Low Cost EPROM Programmer can really help the developers, researchers and mostly our young students to prepare their microprocessor-based hardware. We also hope to bring this programmer in our local market and can sell it within a very cheap price.

REFERENCES

- [1] Malvino & Brown, *Digital Computer Electronics*, Tata McGraw-Hill, 1995, ISBN 0-07-462235-8.
- [2] Barry B. Brey, *The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Processor, Pentium 2, Pentium 3 and Pentium 4 – Architecture, Programming, and Interfacing*, Prentice-Hall, 2002.
- [3] Ronald J. Tocci, *Digital System, Principles and Applications*, Prentice Hall of India, Sixth Edition, 1996.
- [4] Ramesh S. Goankar, *Microprocessor, Architecture, Programming and Applications with the 8085/8080A*, Wiley Eastern Limited, 1993.
- [5] Douglas V. Hall, *Microprocessors and Interfacing: Programming and Hardware*, McGraw-Hill, 1992.
- [6] R. Chandra and M. Rafiquzzaman, *Modern Computer Architecture*, Galgotia Publications Pvt. Ltd., 2000.