International Journal of Soft Computing 1(2) : 121-127, 2006

A Hardware-Software Co-Method for Low Cost EPROM Programmer

Kamruzzaman, S.M., Rashedur Rashid and Mahfuz-ur-Rahman
Department of Computer Science and Engineering, Manarat International University,
Dhaka, Bangladesh

Abstract: Nowadays Erasable Programmable Read Only Memory (EPROM) is widely used in various instruments and devices for storing experimental or ordinary data, assembly code etc. This memory chip has large application in any industry or institute. EPROM programmer is needed in various sections of microprocessor based control system such as city traffic light control, temperature control, DC motor control, stepper motor control, robot control, various toy etc. The EPROM chip is also needed in computer hardware like motherboard, processor, printer, scanner etc. It is used in all of the above applications to read or write programming code sequence or data or various forms of sound and music. Bearing this in mind, a parallel port EPROM programmer is developed that can be used to write data to and read data from EPROM. A software has been developed to read, write, verify and reset of EPROM's and controlling of the device. The developed hardware-software co-method for EPROM programmer is capable of programming those EPROM chips whose address buses are in the range of nine to sixteen bits and data buses are 8 bits. If the address buses and data buses of EPROM chips are not in the range mentioned above, then it is needed to set extra latches in the circuit and accordingly control words are generated. As the EPROM programmer is software based so it is user friendly and low cost compare to commercial one.

Key words: Low Costeprom, EPROM, Co-method

INTRODUCTION

An Erasable Programmable Read Only Memory (EPROM) programmer is a device used to feed data and code in an EPROM[1,2]. It is an undeniable and significant requirement for microprocessor-based project. Although EPROMs are highly available, EPROM programmers are extremely rare[3]. 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 then those available for ordering. Figure 1 shows the circuit diagram of EPROM Programmer.

EPROM is widely used in various instruments and devices for storing experimental or ordinary data, assembly code, boot loader etc[4,5]. This memory chip has large application in any industry or institute[6]. Bearing this in mind, a parallel port EPROM programmer that can be used to write data to and read data from EPROM. For that purpose, we have developed software in which a user can read or write data to EPROM by simply given command with keyboard using the EPROM programmer device that is connected to the CPU by parallel port.

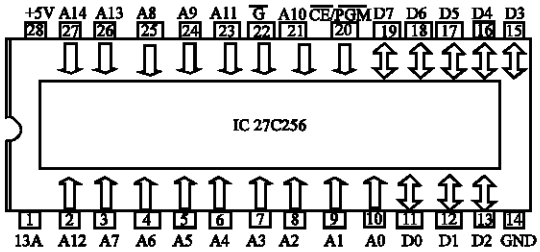

Fig. 1: Pin configuration of 27C256

In this study the EPROM chip 27C256 is used. Its address space is 2 bit and data bus is 8 bit. The read, write, verify and reset operation to this chip is not so difficult. The main learning features of EPROM are addressing, capacity, working level, pin description and programming techniques. Addressing refers the chip address capability, capacity is the data storing range and limits of data field, working levels are the fields of working mode, pin description indicates the chip pin layout explanations and programming techniques is it's working mode activation process.

GENERAL DESCRIPTION

The chip 27C256 is a high-speed 256K UV erasable and electrically re-programmable CMOS EPROM, ideally

Corresponding Author: Kamruzzaman, S.M., Department of Computer Science and Engineering, Manarat International University, Dhaka, Bangladesh

Table 1: Functional description of 27C256

Mode	$\overline{CE}$	$\overline{G}$	A0	A9	Vpp	Outputs
Read	L	L	X	X	Vcc	Data out
Output disable	X	H	X	X	Vcc	Hi-Z
Standby (TTL)	H	X	X	X	Vcc	Hi-Z
Standby (CMOS)	Vcc±0.3	X	X	X	Vcc	Hi-Z
Program	L	H	X	X	Vpp	Data in
Program verify	H	L	X	X	Vpp	Data out
Program inhibit	H	H	X	X	Vpp	Hi-Z
Auto SelectMan ufacture. Code	L	L	L	H	Vcc	01H
Device code	L	L	H	H	Vcc	10H

suited for applications where fast turnaround, pattern experimentation and low power consumption are important requirements. The 27C256 is designed to operate with a single 5V power supply with 10% tolerance[7]. The CMOS design allows the part to operate over Military Temperature Range. A new pattern can then be written electrically into the device by following the programming procedure. This EPROM is fabricated with National's proprietary, time proven CMOS double-poly silicon gate technology, which combines high performance and high density with low power consumption and excellent reliability.

The 27C256 EPROM : The 27C256 is packaged in a 28-pin dual-in-line package with transparent lid and a 32-pin windowed LCC. The transparent lid allows the user to expose the chip to ultraviolet light to erase the bit pattern[8]. Various features of 27C256 are described below: Addressing: In this chip 15-address lines are available, those capable of indicating 32 KB address space.

Capacity: The chip stores 32768 byte data as binary format of 8 bits each. Working level: We can perform read, standby, write, verify and program inhibit of data by this chip. Pin layout. Figure 1 shows the pin configuration of 27C256. Function description: Function descriptions of 27C256 are shown in Table 1. Programming technique: For write, we input a +ve pulse at pin no. 20, 13V at pin no. 1, high signal at pin no 22. For read, we input a low signal at pin no. 20, 5V at pin no. 1, low signal at pin no. 22. For standby, we input a low signal at pin no. 20, 5V at pin no. 1, high or low signal at pin no. 22. For program inhibit, we input a high signal at pin no. 20, 13V at pin no. 1, high signal at pin no. 22.

Overview of the parallel port: Parallel port transfer parallel data asynchronously. During asynchronous data transfers, we must find some means of regulating the flow of information between source and destination. The most general technique, known as handshaking, consists of a separate set of synchronizing signals used to co-ordinate the data transfers. These handshaking signals can be completely separate from the data and use separate signal pathways or can be included along with the data. Here all handshaking signals will be in parallel with the data.

Newer parallel port's are standardized under the IEEE 1284 standard first released in 1994[9]. This standard defines 5 modes of operation, which are as follows:

- Compatibility Mode
- Nibble Mode
- Byte Mode
- EPP Mode (Enhanced Parallel Port)
- ECP Mode (Extended Capabilities Mode)

The aim of this work was to design new drivers and devices, which were compatible with each other and also backwards compatible with the Standard Parallel Port (SPP). Compatibility, Nibble and Byte modes use just the standard hardware available on the original Parallel Port cards while EPP and ECP modes require additional hardware, which can run at faster speeds, while still being downwards compatible with the standard parallel port[10]. Compatibility mode or Centronics Mode as it is commonly known, can only send data in the forward direction at a typical speed of 50 Kbytes per second but can be as high as 150 + Kbytes a second. In order to receive data, we must change the mode to either Nibble or Byte mode. Nibble mode can input a nibble (4 bits) in the reverse direction. Byte mode uses the parallel's bi-directional feature (found only on some cards) to input a byte (8 bits) of data in the reverse direction. Extended and enhanced parallel ports use additional hardware to generate and manage handshaking. To output a byte to a printer (or anything in that matter) using compatibility mode, the software must capable to:

- Write the byte to the data port
- Check to see is the printer is busy. If the printer is busy, it will not accept any data, thus any data, which is written, will be lost.
- Take the Strobe (pin 1) low. This tells the printer that there is the correct data on the data lines (pins 2-9)
- Put the strobe high again after waiting approximately 5 microseconds after putting the strobe low (step 3).

This limits the speed at which the port can run at. The EPP and ECP ports get around this by letting the hardware check to see if the printer is busy and generate a strobe and/or appropriate handshaking. This means only one I/O instruction need to be performed, thus increasing the speed. These ports can output at around 1-2 megabytes per second. The ECP port also has the advantage of using DMA channels and FIFO buffers, thus data can be shifted around without using I/O instructions.

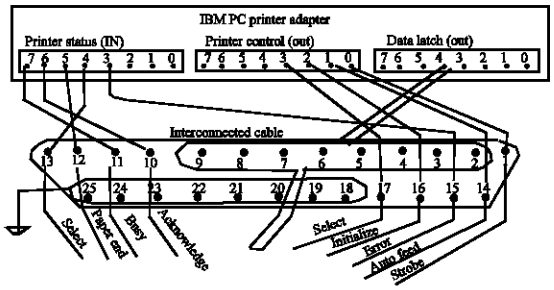

Fig. 2: The parallel port

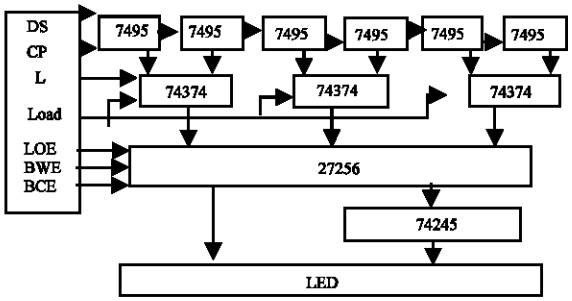

Fig. 3: Block diagram for serial shifting method

Port configuration: Physically, the parallel port is a 25-pin D-type connector on the parallel port adapter. Figure 2, the parallel port is actually a complex data channel built around a number of conventional I/O port/registers known as the data-latch, printer status and printer control registers. All three port/registers are accessed with simple IN and OUT instructions.

MAJOR PROCESS SPECIFICATIONS

Write operation: To perform write operation we need two bus cycles. In the first bus cycle the address (where user want to write) will give in the address input pin and give 40H in the data input pin, which is fixed for this IC 27C256. Basically by writing this, the command register takes the IC in write mode internally. In the next bus cycle, give the same address and data to write. Our program will take the address and data as hexadecimal input and convert it to binary and give to the input pin in a correct sequence. The signals in the write operation will must follow sequence shown in Fig. 3. From the diagram we see, in the WRITE operation first all the active low LOE, BCE and BWE signals should be high.

Then in the first bus cycle give the address and data (40H) in the latch but the latch output will disable the data output. Hence, only the address will be in the address input pin of the IC27256 and not the data. After some delay we have activated the BCE by giving low signal in this pin. Again after some delay we have activated the BWE by giving low signal in this pin. And finally activated the data by giving high signal in the latch output pin and the first bus cycle will be completed. In the second bus cycle, follow the same sequence but in this case data be the actual data to write in place of 40H. In this way the write operation will be performed.

Read operation: To perform read operation we need two bus cycles. In the first bus cycle we have to write the command register. Here we have to give FFH in the data pins and no address in the address pins. This will take the IC in read mode automatically. Now if we perform read by giving address in the address pin and disable the latch output signal, then we will get data in the data pin. Our program will take the address from where to read as hexadecimal input and convert it to binary and give to the input pin in a correct sequence. The signals in the read operation will must follow sequence shown in Fig. 3. So to read, we will first call the write program and give 00H as address and FFH as data. When the write will complete then we will read. The latch output will disable the data output. Hence the data from the IC 27256 will be seen in the LEAD. Our program will take the address as hexadecimal input and convert it to binary and give to the input pin in a correct sequence.

HARDWIRE OPERATIONS

The following subsections describe the steps required for read, write and reset operations of EPROM (Fig. 4).

Reading data from EPROM: In the control register pin no. 2 (1Q) connected to the EPROM's 22 no. pin ($\overline{G}$), pin no. 5 (2Q) connected to the EPROM's 20 no. pin (/$\overline{CE}$ $\overline{PGM}$), pin no. 6 (3Q) connected to the data latch's 1 no. pin ($\overline{ENO}$), pin no. 9 (4Q) connected to the MUX's 9 no. pin (A2), pin no. 12 (5Q) connected to the MUX's 10 no. pin (A1), pin no. 15 (6Q) connected to the MUX's 11 no. pin (A0). For the read operation EPROM's 20 no. pin and 22 no. pin both requires low signal[11]. The steps followed for reading data from EPROM are given below:

Step 1: Lower order 8 bit address passes to latch no. 1 input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 2: Clock to latch no. 1. For that purpose we pass 5 (110) to the decoder address pins (A2, A1, A0). The pin no. 9 (Q6) of decoder is activated. So lower order 8 bit address passes from latch no. 1 input pins to output pins (Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7).

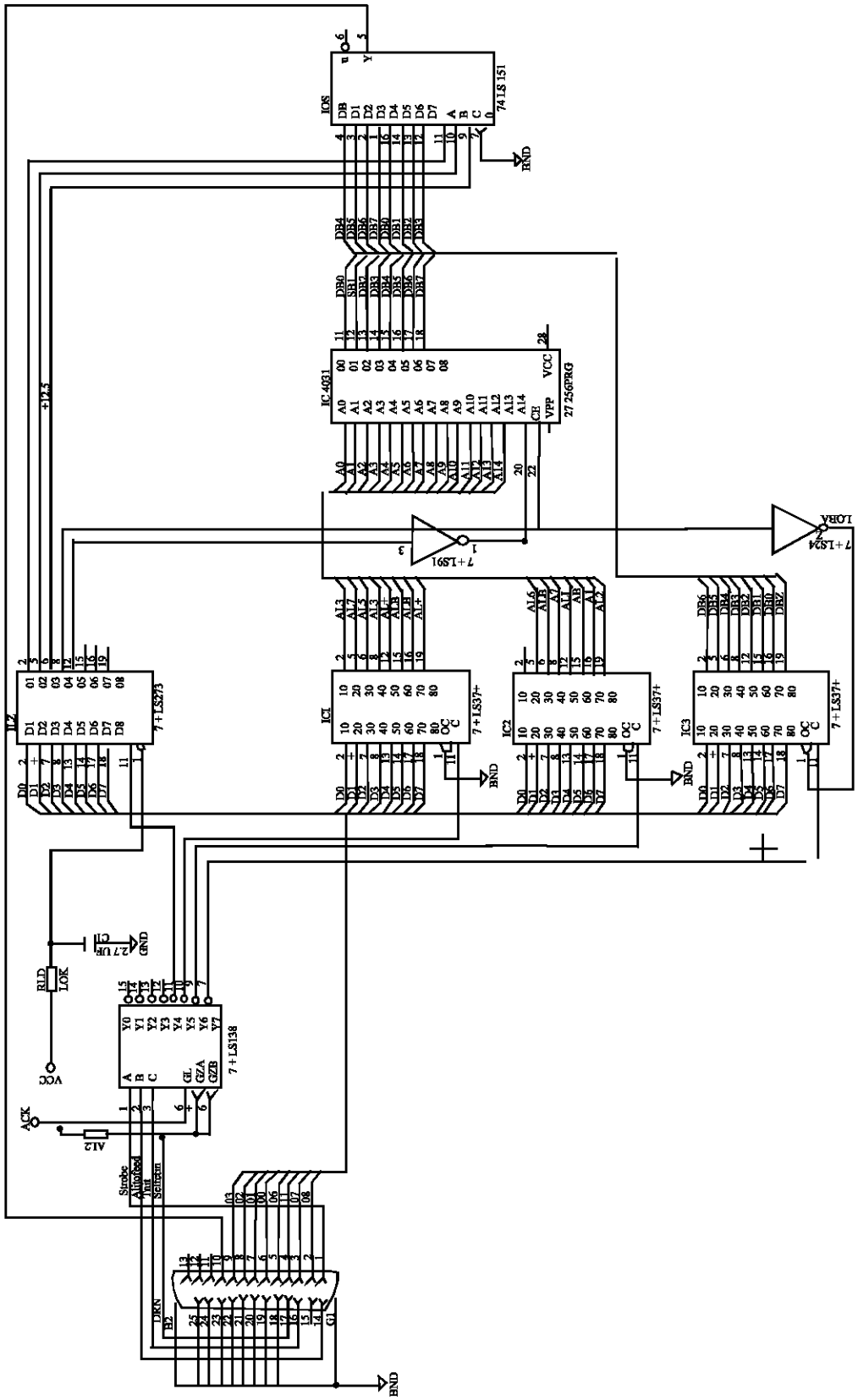

Fig. 4: Circuit diagram of the EPROM programmer

Step 3: Higher order 8 bit address passes to latch no. 2 input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 4: Clock to latch no. 2. For that purpose we pass 7 (111) to the decoder address pins (A2, A1, A0). The pin no. 11 (Q4) of decoder is activated. So lower order 8 bit address passes from latch no. 1 input pins to output pins (Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7).

Step 5: Data 00000000 passes to latch no. 3 (data latch) input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 6: Clock to latch no. 3. For that purpose we pass 4 (100) to the decoder address pins (A2, A1, A0). The pin no. 7 (Q7) of decoder is activated. So the data 00000000 passes from latch no. 3 input pins to output pins (Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7).

Step 7: Pass 4 (00000100) to control registers input pins (1D, 2D, 3D, 4D, 5D, 6D, 7D, 8D).

Step 8: Clock to control registers. For that purpose we pass 1 (001) to the decoder address pins (A2, A1, A0).

Step 9: EPROM is disabled.

- For the disable operation EPROM's 20 no. pin requires active low signal and 22 no. requires active high signal. Passes 5 (00000101) to control register input pins (1D, 2D, 3D, 4D, 5D, 6D, 7D, 8D).
- Clock to control registers. For that purpose we pass 1 (001) to the decoder address pins (A2, A1, A0). The pin no. 13 (Q2) of decoder is activated. So the 8 bit control word passes from control register input pins to output pins (1Q, 2Q, 3Q, 4Q, 5Q, 6Q, 7Q, 8Q). Now EPROM is disabled.

Writing data to EPROM: In the control register pin no. 2 (1Q) connected to the EPROM's 22 no. pin ($\overline{G}$), pin no. 5 (2Q) connected to the EPROM's 20 no. pin ($\overline{CE}$ / $\overline{PGM}$), pin no. 6 (3Q) connected to the data latch's 1 no. pin ($\overline{ENO}$), pin no. 9 (4Q) connected to the MUX's 9 no. pin (A2), pin no. 12 (5Q) connected to the MUX's 10 no. pin (A1), pin no. 15 (6Q) connected to the MUX's 11 no. pin (A0). For the write operation, firstly EPROM's 20 no. pin and 22 no. pin both requires high signal and then 20 no. pin requires low signal and 22 no. pin requires high signal. The steps followed for writing data to EPROM are given below:

Step 1: At first we disable the data latch and the EPROM is in program inhibit mode. (For disable data latch, its pin no. 1 (ENO) requires high signal and for the program inhibit mode to EPROM, its 20 no. pin and 22 no. pin both requires high signal.) For that operation we requires, as follows

- Passes 7 (00000111) to control register input pins (1D, 2D, 3D, 4D, 5D, 6D, 7D, 8D).
- Clock to control registers. For that purpose we pass 1 (001) to the decoder address pins (A2, A1, A0).

Step 2: Lower order 8 bit address passes to latch no. 1 input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 3: Clock to latch no. 1. For that purpose we pass 5 (110) to the decoder address pins (A2, A1, A0). The pin no. 9 (Q6) of decoder is activated. So lower order 8 bit address passes from latch no. 1 input pins to output pins (Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7).

Step 4: Higher order 8 bit address passes to latch no. 2 input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 5: Clock to latch no. 2. For that purpose we pass 7 (111) to the decoder address pins (A2, A1, A0). The pin no. 11 (Q4) of decoder is activated. So lower order 8 bit address passes from latch no. 1 input pins to output pins (Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7).

Step 6: The data for write to the EPROM passes to latch no. 3 (data latch) input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 7: Clock to latch no. 3. For that purpose we pass 4 (100) to the decoder address pins (A2, A1, A0). The pin no. 7 (Q7) of decoder is activated.

Step 8: Now, we requires data latch to be enabled.

- Passes 3(00000011) to control register input pins (1D, 2D, 3D, 4D, 5D, 6D, 7D, 8D).
- Clock to control registers. For that purpose we pass 1 (001) to the decoder address pins (A2, A1, A0).

Step 9: Now, we require chip enable signal to EPROM.

- Passes 1(00000001) to control register input pins (1D, 2D, 3D, 4D, 5D, 6D, 7D, 8D).
- Clock to control registers. For that purpose we pass 1 (001) to the decoder address pins (A2, A1, A0).

Step 10: At last EPROM is in program inhibit mode.

- Passes 3 (00000011) to control register input pins (1D, 2D, 3D, 4D, 5D, 6D, 7D, 8D).
- Clock to control registers. For that purpose we pass 1 (001) to the decoder address pins (A2, A1, A0).

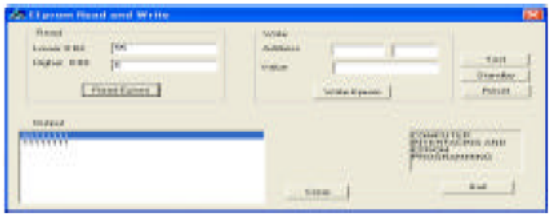

Fig. 5: Read operation screen

Step 11: We save the data in the permanent file that is writing to the EPROM.

Reset the EPROM : In the control register pin no. 2 (1Q) connected to the EPROM's 22 no. pin ($\overline{G}$), pin no. 5 (2Q) connected to the EPROM's 20 no. pin ($\overline{CE}$ / $\overline{PGM}$), pin no. 6 (3Q) connected to the data latch's 1 no. pin ($\overline{ENO}$), pin no. 9 (4Q) connected to the MUX's 9 no. pin (A2), pin no. 12 (5Q) connected to the MUX's 10 no. pin (A1), pin no. 15 (6Q) connected to the MUX's 11 no. pin (A0). For the reset operation, 20 no. pin requires low signal and 22 no. pin requires high signal. The steps followed for reset operation of EPROM are given below:

Step 1: Lower order 8 bit address 00000000 (0) passes to latch no. 1 input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 2: Clock to latch no. 1. For that purpose we pass 5 (110) to the decoder address pins (A2, A1, A0).

Step 3: Higher order 8 bit address 0000000 (0) passes to latch no. 2 input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 4: Clock to latch no. 2. For that purpose we pass 7 (111) to the decoder address pins (A2, A1, A0). The pin no. 11 (Q4) of decoder is activated. So lower order 8 bit address passes from latch no. 1 input pins to output pins (Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7).

Step 5: Data 00000000 (0) passes to latch no. 3 (data latch) input pins (D0, D1, D2, D3, D4, D5, D6, D7).

Step 6: Clock to latch no. 3. For that purpose we pass 4 (100) to the decoder address pins (A2, A1, A0). The pin no. 7 (Q7) of decoder is activated. So the data 00000000 passes from latch no. 3 input pins to output pins (Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7).

Step 7: Now, we require data latch disable and the EPROM is disabled.

- Passes 5(00000101) to control register input pins (1D, 2D, 3D, 4D, 5D, 6D, 7D, 8D).

Fig. 6: Write operation screen

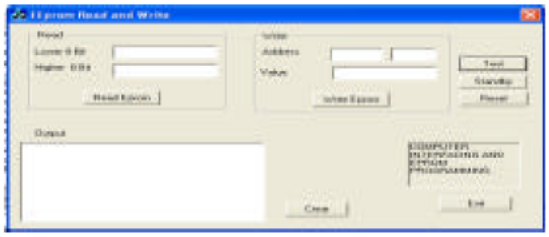

Fig. 7: Reset operation screen

- Clock to control registers. For that purpose we pass 1 (001) to the decoder address pins (A2, A1, A0). Now, the data latch is disabled and 20 no. pin of EPROM get low signal and 22 no. pin of EPROM get high signal. The EPROM is also reset.

SOFTWARE OPERATIONS

The following subsections describe the software operation of read, write and reset for EPROM. The software development flowchart is shown in Fig. 8.

Read operation: For read operation, we have to input starting address and number of data. Then press the read button. Here for each data read, need 20 seconds because we add delay at the code to trace each step (Fig. 5).

Write operation: For write operation, we have to input starting address, number of data and the data at a time. Then press the write button. Here for each data write, need 18 seconds because we add delay at the code to trace each step (Fig. 6).

Reset operation: For reset operation, press the reset button. In the reset operation, the circuit needs 40 seconds because we add delay at the code to trace each step (Fig. 7 and 8).

CONCLUSION

A device has been developed to read, write, verify and reset of EPROM's and software has been developed for controlling the device. The EPROM programmer is needed in various sections of microprocessor based control system such as city traffic light control, temperature control, DC motor control, stepper motor control, robot control, various toy etc. The EPROM chip

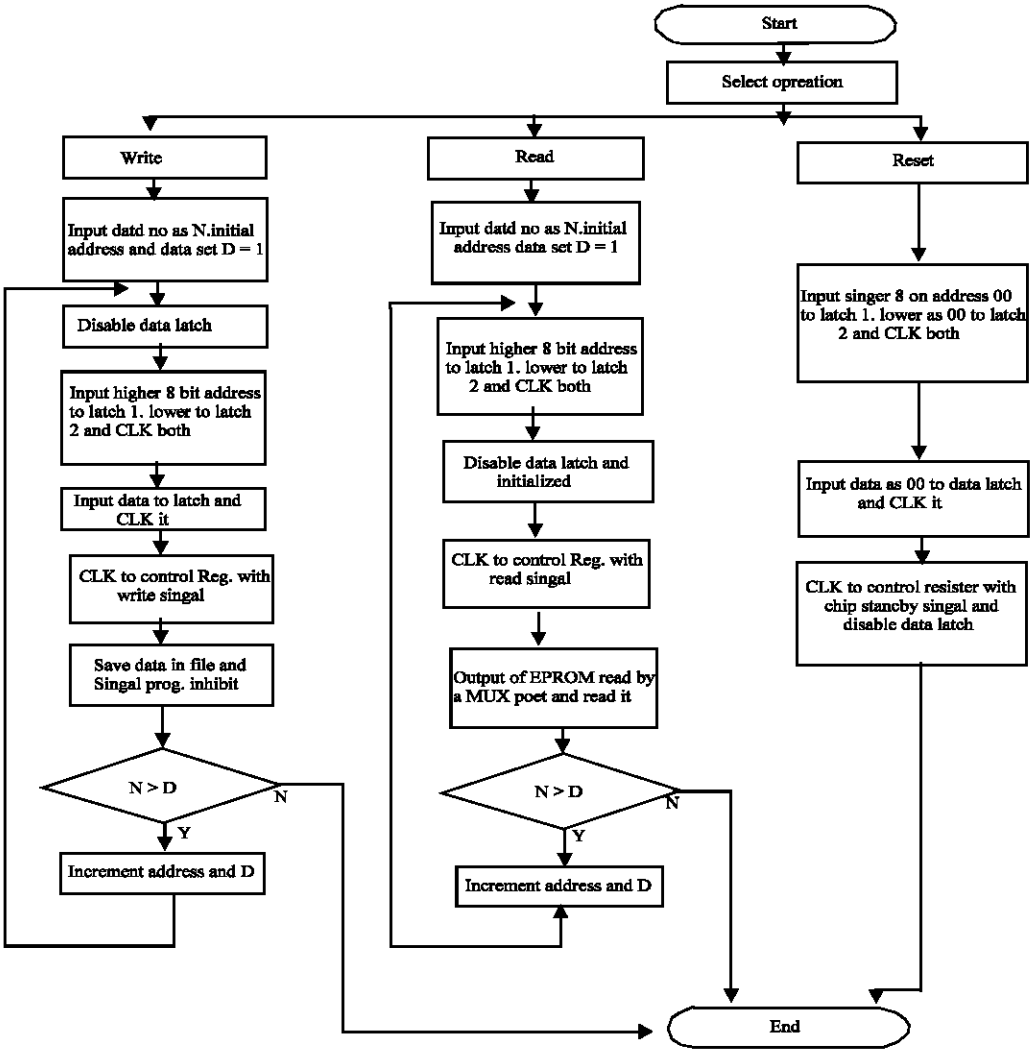

Fig. 8: Software flowchart for EPROM programmer

is also needed in computer hardware component like motherboard, processor, printer, scanner etc. At the end we can say that our developed hardware-software co-method for 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. Douglas, V., Hall, Microprocessors and interfacing 2nd Edition, Tata-McGraw-Hill Publishing Company Limited.
2. http://stephanecarrez.free.fr
3. Roy, W., Goody Intel Microprocessors Hardware, Software and Applications, 2nd Edn., McGraw-Hill International Editions.
4. Malvino and Brown, 1995. Digital Computer Electronics,Tata McGraw-Hill, , ISBN 0-07-462235-8.
5. Barry, B. and Brey, 2002. The Intel Microprocessors: Architecture, Programming and Interfacing, Prentice-Hall.
6. Gaonker, Microprocessor Architecture Programming and Application 3rd Edition, Tata-McGraw-Hill Publishing Company Limited.
7. Up to date Memory IC, Data and comparisons tables, 8X350...882048 Integrated memory circuits, BPB publications.
8. Data and comparisons tables, 7400...7450729 Integrated circuits, BPB publications.
9. http://neutrino.phys.wasington.edu/~berns/archive /LogicICs/74LS273.pdf
10. http://www.futurlec.com/74LS/74LS273pr.shtml
11. www.fairchildsemi.com