International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 9, September - 2013

The Design and Implementation of a Universal EPROM Programmer

Nathan David
Department of Electronic Engineering,
Faculty of Engineering, University of Nigeria, Nsukka.

Nnamani, Uchenna H
Department of Electronic Engineering,
Faculty of Engineering, University of Nigeria, Nsukka.

Odoeze, Jideofor A.H
Department of Electrical and Electronic Engineering
Federal Science and Technical College, Yaba

ABSTRACT
Since the discovery of transistors and consequently the digital integrated circuits used in digital system, memory devices have been used extensively and unavoidably in most digital applications. The embedded and non-embedded system such as the microprocessor and microcontroller systems respectively are one of those systems that make extensive use of memory devices both the volatile and non-volatile memories for temporal data or variable storage and permanent instruction storage respectively. The non-volatile memory is used in the state of art CD and DVD playback sets and computer systems as a repository for the low-level system information such as the BIOS. A more reliable and efficient digital circuit miniaturization such as the sequential circuit is best implemented using the non-volatile memories particularly the EPROM which has been programmed to accomplish such sequential functions. It is in light of this that we have taken up the challenge to Design and Implement a Universal EPROM Programmer that complies with the Joint Electron Device Engineering Council (JEDEC) Specification. It is microprocessor based with a user-friendly GUI that is supported by any Microsoft Windows Operating System platform. It incorporates a software driver with Chip Identification Algorithm (CIA) that has the capability of identifying any specific EPROM and automatically selects the optimal programming algorithm for that particular EPROM. The software interacts with the external hardware via the computer parallel port using the bi-directional protocol of the Line Printer (LPT) interface. The hardware and the software is designed to support a large memory addressing capability which is about 65kb more than most available non-volatile memory capacity in the market today.

1.0 INTRODUCTION
EPROM is widely used in various instruments and devices for storing experimental or ordinary data, assembly code and boot loader. Its relevance in microprocessor – based projects can never be over emphasized. This memory chip has large applications in any industry or institute. An EPROM integrated circuit has no usefulness until data or assembly code has been programmed or written in it. A device programmer or specifically EPROM programmer is a device used to feed data and code into an EPROM. The device programmer has the capability of performing all the EPROM programming operation such as the program reading operation, program or write operation etc. Although device programmer for particular EPROM are available but universal EPROM programmer are usually rare thus the relevance of this work.Gibilisco [1] defined memory as the section of a digital computer that records and holds data until it is necessary. In computer systems, ideally memory device is group into different categories using the following parameter which is commonly referred to as *Memory Hierarchy*.
These parameters include Speed, Size, Power Consumption, Volatility and Cost.

REGISTERS
At the very top of this memory hierarchy is registers (also sometimes referred to as processor registers), which consist of an extremely small amount of very fast (i.e., much faster than the main memory) memory cells that are built directly into the CPU. Their purpose is to speed up the execution of the CPU, and thus of programs, by providing quick access to commonly used values, generally those in the midst of a calculation. Registers are usually implemented using array of SRAM (i.e., Static RAM) cells.

CACHE
Below the registers in the memory hierarchy is the cache memory, it is a small amount of high-speed memory, usually with a memory cycle time comparable to the time required by the CPU to fetch one instruction. Whose purpose is to reduce the mismatch in speeds between the CPU and RAM because the modern processors have clock rates of several gigahertz (billions of cycles per second), whereas RAM chips have access times measured in megahertz (millions of cycles per second).
The amount of information which is replaces at one time in the cache is called theline sizefor the cache. This is normally the width of the data bus between the cache memory and the main memory.

PRIMARY MEMORY
Primary memory, previously known as storage, is the only one directly accessible to the CPU. The CPU continuously reads instructions stored there and executes them as required. Any data actively operated on is also stored there in uniform manner. Primary memory is directly or indirectly connected to the CPU via a memory bus, today sometimes referred to as a front side bus. It is actually comprised of two buses, an address bus and a data bus.
The Cpushack [2] based on these core characteristics the following are the different classification of Primary memory.
Random Access Memory (RAM)
    Static Random Access Memory (SRAM) Dynamic Random Access Memory (DRAM)
Read Only Memory (ROM)

Mask Read Only Memory (MROM)
Programmable Read-Only Memory (PROM)
One Time Programmable Read-Only Memory (OTPROM)
Erasable Programmable Read-Only Memory (EPROM
UV-Erasable Programmable Rom (UV-EPROM)
Non-Volatile UV-Erasable Programmable Rom (NVROM)

Hybrid Memory
Non-Volatile Ram (NVRAM)
Electrically Erasable Programmable Rom (EEPROM)
Flash Memory

1.5 SECONDARY MEMORY

Secondary memory or storage in popular usage differs from primary storage in that it is not directly accessible by the CPU. Secondary storage does not lose the data when the device is powered down - therefore it is non-volatile. Some other examples of secondary storage technologies are flash memory (e.g. USB sticks or keys), floppy disks, magnetic tape, paper tape, punch cards, standalone RAM disks, and Zip drives.

1.6 TERTIARY MEMORY

Tertiary Memory or Tertiary Storage provides a third level of storage. Typically it involves a robotic mechanism which will mountand unmount removable mass storage media into a storage device according to the system's demands; this data is often copied to secondary storage before use. It is primarily used for archival of rarely accessed information since it is much slower than secondary storage

2.0 JOINT ELECTRONIC DEVICE ENGINEERING COUNCIL (JEDEC).

JEDEC Standard [3] Joint Electronic Device Engineering Council (JEDEC) standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interoperability, interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for use by those other than JEDEC members, whether the standard is to be used either domestically or internationally. JEDEC standards and publications are adopted without regard to whether or not their adoption may involve patents or articles, materials, or processes. This project also adopted some of the regulation of this standardization organization such as EPROM programming electrical signal (Timing Diagram), object files (Intel Hex File), EPROM pin-outs etc; since the EPROM to be programmed by the Device Programmer cuts across all EPROM from different manufacture which is within the required size.

2.1THE PURPOSE AND SCOPE OF THE PROJECT

The early hard–wired digital systems had no future for personal use because of its size and heat dissipation as a result of wiring all the logic function (AND, OR, addition, multiplication etc.) performed by the digital system. The advent of memory devices has created a world of different in digital logic circuit design. For instants modern daysmicroprocessors implement only a hard–wired adder and NOR circuit while all other arithmetic and logic functions are stored as data in the memory such as EPROM creating a more dynamic (programmable) and compacted systems.

The Device Programmer is made up of the following basic units; this is also illustrated pictorial in figure 2:

- ☐ The Graphic User Interface
- ☐ The Microprocessor Parallel Port Interface
- ☐ The Decoder Clocking Unit
- ☐ The Address Register Unit
- ☐ The Data Latching Register Unit
- ☐ The Programming Control Latch Registers Unit
- ☐ The Programmer Identification Register Unit
- ☐ The Voltage Level Selection Units

The Graphic User Interface was implemented using Visual Basic Languageto keep the user in track with the events taking place in the programmer such as the Chip Write, Chip Read, Chip Verify controls. The GUI uses a very simple programming control structure such that with very minimal computer knowledge any individual can use it effectively.

The Microprocessor Parallel Port Interface is a computer peripheral device interface used to connect devices to the microprocessors bus. The use of Parallel port interface is primarily due to it easy of programmability and speed which makes it optimal for real-time communication application.

The Decoder Clocking Unit employs the use of digital Integrated circuit decoder and some logic gate to generate pulsing signal for the various part of the system. This clocking unit synchronizes the operation of the Programmer with the microprocessor data transfer speed.

The Address Register Unit uses the address multiplexing technique to hold the address of the memory location in the EPROM to be programmed or read using the D-type registers.

The Data Latching Register Unit also uses this aforementioned multiplexing technique to hold the data to be written into the memory location of the EPROM. It employs the use of D-type register and a transparent latch circuits. This unit is responsible of transferring data from the EPROM to the microprocessor bus line and from the microprocessor bus line into the EPROM memory cells

The Programming Control Latch registers Unit uses transparent latch interrogated circuit. It controls the selection or the different programming control signal such as the Chip Enable ($\overline{CE}$), Output enable ($\overline{OE}$) and Programming Voltages ($\overline{PGM}$) also the selection of the different programming voltage levels form the Voltage Level Selection Units

The Voltage Level Selection Units employs the use of simple discrete component to implement the generation of different DC voltage levels. This Unit, for a proper interconnection with the other part of the circuit employs the digital technique of tri–state technology in its operation

The Programmer Identification Register Unit uses a transparent latch integrated circuit to establish proper and defined connectivity between the microprocessor and EPROM programmer circuit. It uses the bi–directional parallel port interfacing technique to identify the version and the model of the programmer.

3.0 MICROPROCESSOR DATA TRANSFER

The microprocessor system communicates with the internal memory, the I/O devices e.g. hard disk, keyboard etc. and to or from outside world (peripherals) via a medium called microprocessor interfaces. This microprocessor data transfer is basically accomplished in two pre – defined data format namely the parallel data transfer and serial data transfer. The interface that implements these aforementioned data transfer in a personal computer system is traditionally called the computer ports or simply ports.

3.1 PARALLEL PORT PROGRAMMING

All programming languages allow programmers to access parallel port using some library functions in some version of Window Operating System such as DOS and non–NT Windows. For example, Borland C is has "Inportb" and "Outportb" functions to read or write I/O mapped peripheral. Visual Basic does not have any simple functions or support to access parallel port directly, but it is possible to add such capabilities to your Visual Basic application by writing a Dynamic Link Library in Visual C++ and calling its exported functions from Visual Basic. Logix4u [4]this is the case in this project work using inpout32.dll. The visual basic implementation of this project using Visual basic is shown in figure 1.

3.2 OBJECT FILE

In computer science, an object file is an organized collection of named objects, and typically these objects are sequences of computer instructions in a machine code format, which may be directly executed by a computer's CPU or embedded systems. Object files are typically produced by a compilers and assemblers as a result of processing a source code file. Object files contain compact code, and are often called "binaries". The most used object file format for embedded systems today are Intel Hex File, MOS Technology Hex File, Motorola Hex File and Binary File

3.3 INTEL HEX FILE FORMAT

Intel HEX is a file format for conveying binary information for applications like programming microcontrollers, EPROMs, and other kinds of chips. It is one of the oldest file formats available for this purpose, having been in use since the 1970s. The Intel HEX file is an ASCII text file with lines of text that follow the Intel HEX file format. Each line in an Intel HEX file contains one HEX record. These records are made up of hexadecimal numbers that represent machine language code and/or constant data. Most EPROM programmers or emulators use Intel HEX files.

3.4 INTEL HEX FILE RECORD FORMAT

This format is made up of six fields that are arranged in the following format assuming each latter represent a digit in hexadecimal.

```
:NNAAAARRHHHHHHHHHHHHHHHHHHHHHHHHCC
```

Each group of letters corresponds to a different field, and each letter represents a single hexadecimal digit. Each field is composed of at least two hexadecimal digits-which make up a byte-as described below:

- **Start Byte**(**:**) is always character **":".** The colon starts every Intel HEX record.
- **Byte Count** (**NN**) takes one byte (two hex digits) indicating a number of bytes in a data field of the line. Usually there are 16 or 32 bytes of data in each line.
- **Address** (**AAAA**) takes two bytes (16 bits – four hex digits). Address shows the beginning of memory position where data should starts. 16 bits gives a limit of 64kilobytes.
- **Record Type** (**RR**) takes one byte (two hex digits). It defines the type of data field.
- **Data Byte** (**HH**) is a sequence of N bytes (2*N hex digits);
- **Checksum** (**CC**) is one byte (two hex digits). It is a last visible byte in a line. Uffenbeck [5]it is calculated by adding together the hex-encoded bytes the Byte count, both bytes in Address, the Record type, each Data byte and the Checksum), taking only the LSB, and either subtracting the byte from 0x100 or inverting the byte (XORing 0xFF) and adding one (1). The result must then be ANDed again with 255 since both 0x100-0 and (0x00 XOR 0xFF) +1 equal 0x100

4.0 HARDWARE DESIGN AND IMPLEMENTATION

The hardware circuit design has been divided into different units or sub-section as listed above for ease of understanding but figure 3 shows the integration version of the whole sub – sections of units.

Control Signals	Hex
LOAD_LOW_ADD	&H0
LOAD_HIGH_ADD	&H1
LOAD_DATA_BYTE	&H2
ENABLE_RD_WR_CONTROL	&H3
PROGRAMMER_ID	&H4
RD_WR_ENABLE_MODE	&H5
UNUSED_PIN	&H6
NORMAL_CTRL_STATE	&H7

Table 4.1 Hardware Control Code.

4.1 ADDRESSING SUB – SYSTEM

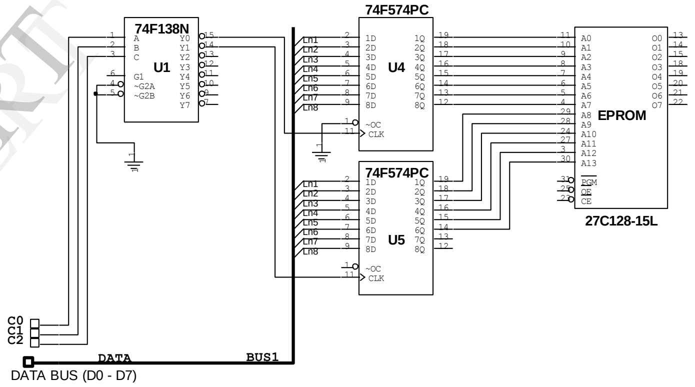


Fig 4.1 Addressing Sub – Section

The addressing of zero [0kb] to 65523[64kb] space is achieved via the use of the 74F138N a 3-8 Decoder, 74F574PC Register IC and the control or driver software algorithm that controls the address loading operation. The figure 4.1 is the hardware sub section of addressing unit.

The circuit operation is as follows; the software keeps the control line at NORMAL_CTRL_STATE i.e. by outputting **111** bits on the control line of the parallel port to line C0, C1, and C2 respectively as shown in the figure 4.1. This causes the Y7 output pin of the 74F138N a 3-8 Decoder to transit to logic low state. The software algorithm outputs the lower address through the data line of the parallel port to the DATA BUS (D0-07). This lower address is latched on the data latch of the parallel port. The software activates the LOAD_LOW_ADD signal by outputting **000** on the control line. This causes Y0 output line of the 74F138N 3-8 Decoder to transit from high to low logic state thereby causes a low to high transition at the clock input of the U4 Register IC since U4 is an edge triggered D-register IC. This transition on the clock input of the U4 IC causes every data bits that is on the data bus to be loaded in the U4 IC and made available on its output Q - lines.

Similarly, for the loading of the higher address at the U5 register; the software reset the controls back to NORMAL_CTRL_STATE by outputting **111** bitson the control lines which is latch by the control latch of the parallel port. The high address as gotten from the binary or Hex file is then loaded on the DATA BUS (D0-D7) through data latch of the parallel port. The software output **001** bits (i.e. LOAD_HIGH_ADD control bits) on the control line which causes the 3-8 Decoder Y1 output to transit from high logic state to low logic state, this will therefore cause the input clock signal to transit from low to high logic level making the data on the DATA BUS to be loaded in the U5 register and made available on its output Q - lines. This two 8 bit register address then combine together to form an addresses space of a maximum of 0xFFFF in Hexadecimal bit [64kbyte]

4.2 DATA MULTIPLEXING SUB – SYSTEM

During the Read Operation, when the appropriate address has been loaded in the address registers that is U4 and U5 by applying the appropriate control signal as described above. The Read Operation is achieved via the use of the Data Multiplex 74F157N

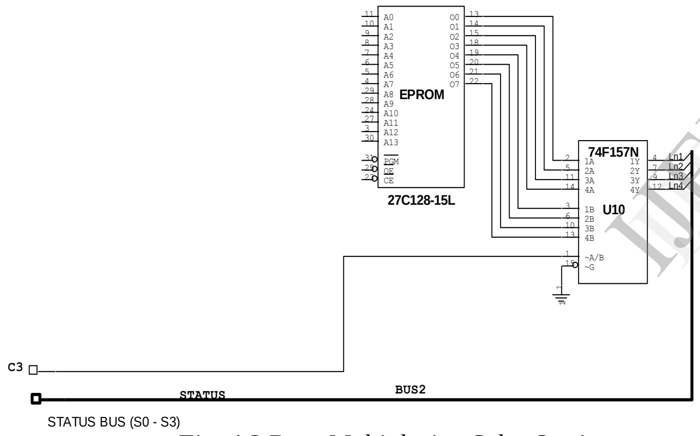


Fig. 4.2 Data Multiplexing Sub - Section

The 74F157N multiplex IC is called 4–bit–wide 2–1 Multiplexer with the A/B select input as indicated on U10 in figure 4.2 above.

If the 8–bit or a byte data is available on the output of the EPROM at line O0 to O7, this 8-bit or byte is read by the software algorithm in nibble mode or order i.e. 4-bit at a time. To avoid bus contention or data contamination by the U6 register IC, its output lines is place at a High impedance state [High-Z]. The software output zero logic (0) on the C3 line this makes available on the STATUS BUS the lower nibble of the data on the output of the EPROM at O0 – O3 via the 1A-4A of the multiplexer. The software reads this nibble and stores it in an integral variable.The software outputs a logic high (1) on the C3 line and just like before the high nibble is made available on the status bus line by connecting the 1B-4B input of the U10 Multiplex to STATUS BUS. The software reads and stores the value in integral variable. Then the low nibble and the high nibble are manipulated and merged to form its true 8-bit value as before on the output of the EPROM.

4.3 DATA PROGRAMMING SUB-SYSTEM

During the Data Write or Programming, the above section of the entire circuit is used to achieve the write or programming operation. The circuit include is 74F138N a 3–8 lines Decoder and the register IC 74F574.

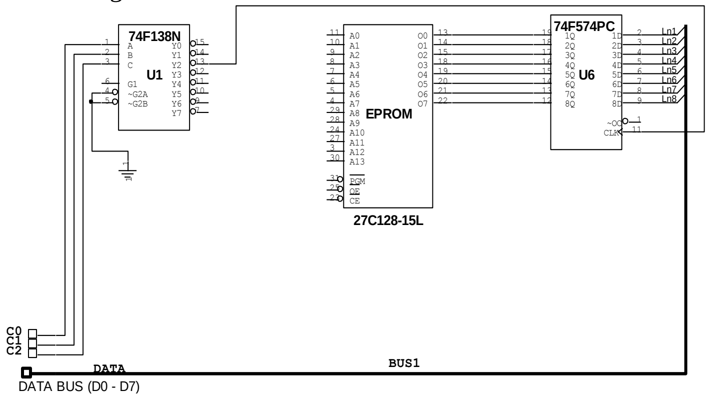


Fig 4.3 Data Programming Sub-System

During the Write Operation, the software reset the control lines to logic high state which is the NORMAL_CTRL_STATE state. Then the data to be programmed into the EPROM is loaded through the data latch of the parallel port to DATA BUS (O0 – O7). The software loads the control signal **010** bit(i.e. is the LOAD_DATA_BYTE control), this causes the Y2 output line of the 3 – 8 line Decoder to transit from high logic state to low logic state, therefore causing the Clock input (CLK) of U6 IC to transit from low to high logic state and then loading the data in the DATA BUS (O0 –O7) into the U6 register which is made available on its output Q – lines (output lines of U6 i.e. 1Q – 8Q).With the application of appropriate write control signal, the data on the U6 IC output lines is written or programmed into the selected or addressed cell units of the EPROM.

4.4 VOLTAGE LEVEL SELECTION UNIT

The voltage level selection unit is of two types used in this project (a) the voltage level variation unit used during EPROM Programming Operation and (b) voltage level variation unit used during Chip Identification Operation Mode.

4.4.1 AT EPROM PROGRAMMING OPERATION

The voltage level variation unit used During EPROM programming operation (particularly Read, Program and Verify Operation Mode) takes place at the V_{PP} control input line of the EPROM. This voltage variation at this input line is of three distinct and regulated steps. The steps are form 0.0V to 5.0V, form 0.0V to 13.0V and finally from 5.0V to 13.0V. This voltage variation takes place during programming operation such as the Read Mode, Programming or Write Mode and Verify Mode. The figure 4.5 below is the circuit that was used to achieve these three steps of voltage variations. The circuit is a combination of many discrete circuit elements to form a module. This module has the capability of generating all the aforementioned voltage variation states at its single output line bydigitally addressing the two input lines appropriately. The input line uses logic level voltages i.e. ideally 0.0 V and 5.0V (Zeros and Ones respectively) making the module to be compatible with digital circuitry. Due to discrete circuit element used the module is optimized for high transition speed (the time taken for the module to change from one voltage state to another voltage state after the application of the digital address at the two input line D_1 and D_2).

The circuit shown in circuit figure 4.5 below has two addressable inputs the J1 and the J2. These inputs accept digital signals or logic voltage levels i.e. ideally 0.00V and 5.00V. The voltage variation capability of the module is achieved via the use of zener voltage regulation i.e. the D1 and D2 which are 13.00V and 5.00V voltage zener diode respectively. Since the module has two digitally addressable inputs therefore there are four input states that can be obtained by the variation of the different input signals. A closer look at the different input states are as follows:

State 0; Here the J1 and J2 inputs are at a logic level zero i.e. connecting Key-A and Key-B to the ground line i.e. at 0.00V. Since there is Zero voltage at J1 input line, the base to Emitter voltage of transistor Q2 is approximately close to ground level therefore switching the transistor Q2 OFF.

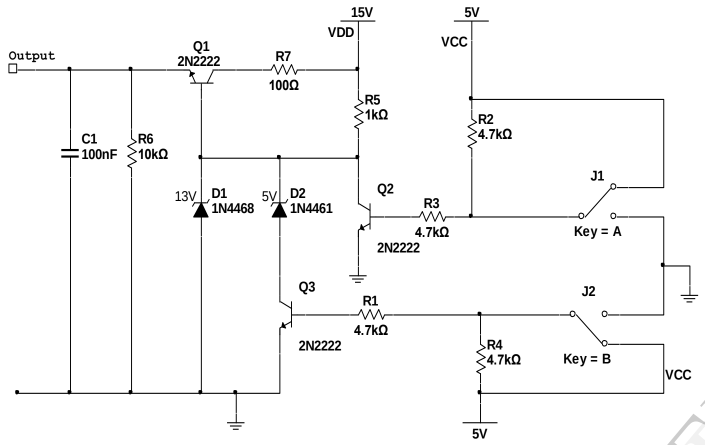


Fig. 4.5 Internal Circuitry Voltage Selection Module used At EPROM Programming Operation

The same thing happens to the transistor Q3 since the base to emitter voltage is less than 0.70V (silicon transistor switching voltage). This therefore switches the transistor Q3 OFF. Since Q2 and Q3 is switched off no current flows from the VDD through the Q2 transistor and no current flow through the zener diode D2 and transistor Q3. Instead the current flows from the VDD supply through R5 then through the D1 and to ground while some current divides through the base emitter junction of transistor Q1 therefore switching ON the transistor and then to the output. The output voltage is approximately equal to D1 zener voltage.

State 1; Here the J1 input is at logic level Zero while the J2 is at logic level One. This is accomplished by tapping Key-A to the ground line and Key-B to the VCC line. Just as before the transistor Q2 is at OFF state since base emitter voltage is less that 0.70V while transistor Q3 is at ON state since the base emitter voltage is approximately 5.00V now. The transistor Q3 will driver more current from VDD through the D2 zener diode then to ground while some part goes through the Q1 transistor then to the output. This will produce an output approximately equal to 5.00V.

State 2 and State 3; will produce Zero voltage at the output of the module. This is primarily because the transistor Q2 has been switched ON by tapping the Key-A to VCC line. Tapping the Key-A to VCC line switches on the transistor Q2; this causes the transistor to turn ON therefore driving all the current from VDD through R5 to ground. This invariably switches Q1 OFF therefore living 0.00V at the output of the module. The different state of operation is summarized in the table 4.2 below.

	J1	J2	Pin 9 (Output) Volt
State 0	0	0	13
State 1	0	1	5
State 2	1	0	0
State 3	1	1	0

Table 4.2 Voltage Selection Function Table

4.4.2 AT CHIP IDENTIFICATION

The chip identification algorithm states that the address line 9 (A9) should be placed at 12.0 ± 0.25V for EPROM Chip Identification Code to be made available at the output lines of the EPROM while other control line placed in their proper voltage state for the chip identification code to be read check section.

The voltage variation unit used during EPROM programming would have been the best option considering the fact that 12.0V state can be obtain by the module via digital addressing. The address pin A9 is serving a dual purpose first as the voltage line for Chip Identification and secondly as an address line. If the circuit is used, it will generate a phenomenon know as the *Bus Contention* in digital electronic. Paul Horowitz and Winfield Hill [6] it is the production of unpredictable signal on the bus lines as a result of jamming or combination of two or more signal from different output sources. This will either damage some part of the integrated circuit or generate a signal that is unpredictable on the bus line. Here it is the contamination of the voltage level of either the address line or the voltage variation unit since the output line and the address line (A9) for the EPROM are electrically connected. Therefore the circuit in figure 3.6 employs the tri-state (3-States) logic output feature to overcome this problem of bus contention. It is called tri-state because it allows three possible output states; high (logic 1), low (logic 0) and high – impedance (High – Z) state. The high impedance is a condition in which both the pull-up and the pull-down transistor are turned OFF so that the output terminal is at very high impedance to both ground and the power supply.

This circuit is used to generate the 12.00V required for the CIA EPROM operation. It has one special feature which is the generating of high-impedance (High-Z) state. This state is very important sine the point or pin on EPROM where it is connected is not only used for EPROM identification only but also used as an Address input pin (particularly A9) of the EPROM. High impedance state removes the problem of bus contention on this address input pin of the EPROM.

In the system whenever an EPROM is to be identified, the appropriate signal is sent to the input of this module selecting any the required voltage at the output but if the EPROM is operated in a mode where it is being addressed (i.e. in Read Mode, Program Mode etc) the output of the module most be at high impedance (High-Z) state for a better or desired performance. As before there are two input lines therefore it has a maximum of four input states when digitally addressed.

State 0; Here the B1 and B2 inputs is at a logic level zero i.e. connecting Key-A and Key-B to the ground line (0 line as indicated in the circuit diagram) at 0.00V. This will cause the two transistors Q1 (pull – up transistor) and Q2 (pull - down transistor) to be in the OFF state since the base to emitter voltage is less than 0.70V (silicon transistor switching voltage). Therefore the output line will be floating between VDD and Ground voltage signal, this state of operation creates a very high resistance in the output line. This generates a phenomenon known as the High impedance state. High impedance state is neither in logic 0 nor logic 1 although since it is a high resistive or impedance state the potential difference across it is expected to be the supply voltage level. It is a technique used to electrically disconnect a circuit from other interconnections.

State 1, State 3; the B2 inputs is at a logic level one i.e. connecting Key-A to the VCC supply line while the second input B1 is at a don't care state that is to say the state of input B1 do not affect the output of the module in this two states. This will cause the transistor Q2 to switch ON causing a very low resistance to be created between the collector and the emitter therefore completely sinking the output line to ground potential.

State 2; Here the B1 and B2 inputs is at a logic level one and zero respectively i.e. connecting Key-A to the VCC line and Key-B to the ground line. This connection will cause transistor Q1 to switch ON since the base to emitter voltage is greater than 0.70V therefore creating a negligible collector to emitter resistance and transistor Q2 will be in the OFF state since the base to emitter voltage is less than 0.70V. The transistor Q2 will create a very high base to emitter impedance (approximately 10MΩ) this will cause the current from Q1 transistor to flow into the load connected across output line then to the ground.

	B1	B2	Pin 5 Output (Volt)
State 0	0	0	High – Z
State 1	0	1	0
State 2	1	0	12
State 3	1	1	0

Table 4.3 Voltage Selection Function Table

5.0 CONLUSION

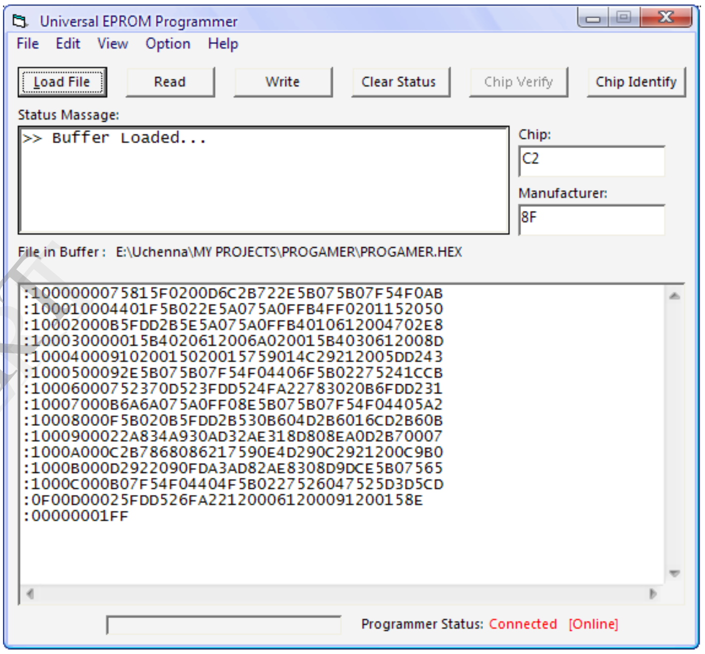
This Universal EPROM Programmer was designed to meet the basic requirement of present day available EPROM programmers. Each of the different hardwaresections where simulated by Multisim 10 simulator and physically implemented. Although it is not actually in the sense that not all available EPROM that it can program, but all the EPROM chip within 64kb of size. With this objective in mind we can conclude that this project met all its required requirements.

6.0 ACKNOWLEDGEMENT

This paper is a part of a larger research collaboration effort. The research implementation presented in the paper is part of the Design of a Universal EPROM Programmer project, completed and submitted as unpublished works at the Department of Electronic Engineering, University Of Nigeria, Nsukka, in February of 2010 by AgboChibuzo V, AsiegbuChiagozie P, NnamaniUchenna H and OssaiOnyemechi L, . The project was supervised by Nathan David.

REFERENCES

1. Gibilisco, Stan. *The Illustrated Dictionary of Electronics*.8th ed. New York: Mc-Graw Hill, 2001.
2. The Cpushack: CPU *History – EPROMs* [http://www.cpushack.net/EPROM.html]
3. JEDEC Solid State Technology Association 2004:*Marking, Symbols, and Labels for Identification of Lead (Pb) Free Assemblies, Components, and Devices.* [http://www.jedec.org/Catalog/catalog.cfm]
4. Logix4u: *How Inpout32.dll works* [http://www.logix4u.net/inpout32.htm]
5. Uffenbeck, John. *Microcomputers and microprocessors-The 8080, 8085, and Z-80 programming , interfacing, and Troubleshooting*, 3rd ed. India . Asoke K. Ghosh, 2000.
6 Paul Horowitz and Winfield Hill.*The Act of Electronics*.2nd ed. United Kingdom. Press Syndicate of the University of Cambridge. 2002



Status Massage:

```
>> Buffer Loaded...
```

Chip: C2

Manufacturer: 8F

File in Buffer : E:\Uchenna\MY PROJECTS\PROGAMER\PROGAMER.HEX

```
:100000007581 5F0200D6C2B722E5B075B07F54F0AB
:100010004401F5B022E5A075A0FFB4FF0201152050
:10002000B5FDD2B5E5A075A0FFB4010612004702E8
:100030000015B4020612006A020015B4030612008D
:1000400091020015020015759014C29212005DD243
:1000500092E5B075B07F54F04406F5B02275241CCB
:10006000752370D523FDD524FA22783020B6FDD231
:10007000B6A6A075A0FF08E5B075B07F54F04405A2
:10008000F5B020B5FDD2B530B604D2B6016CD2B60B
:1000900022A834A930AD32AE318D808EA0D2B70007
:1000A000C2B7868086217590E4D290C2921200C9B0
:1000B000D2922090FDA3AD82AE8308D9DCE5B07565
:1000C000B07F54F04404F5B0227526047525D3D5CD
:0F00D00025FDD526FA2212000612000912001 58E
:00000001FF
```

Programmer Status: Connected [Online]

Fig. 1: GUI Main Window

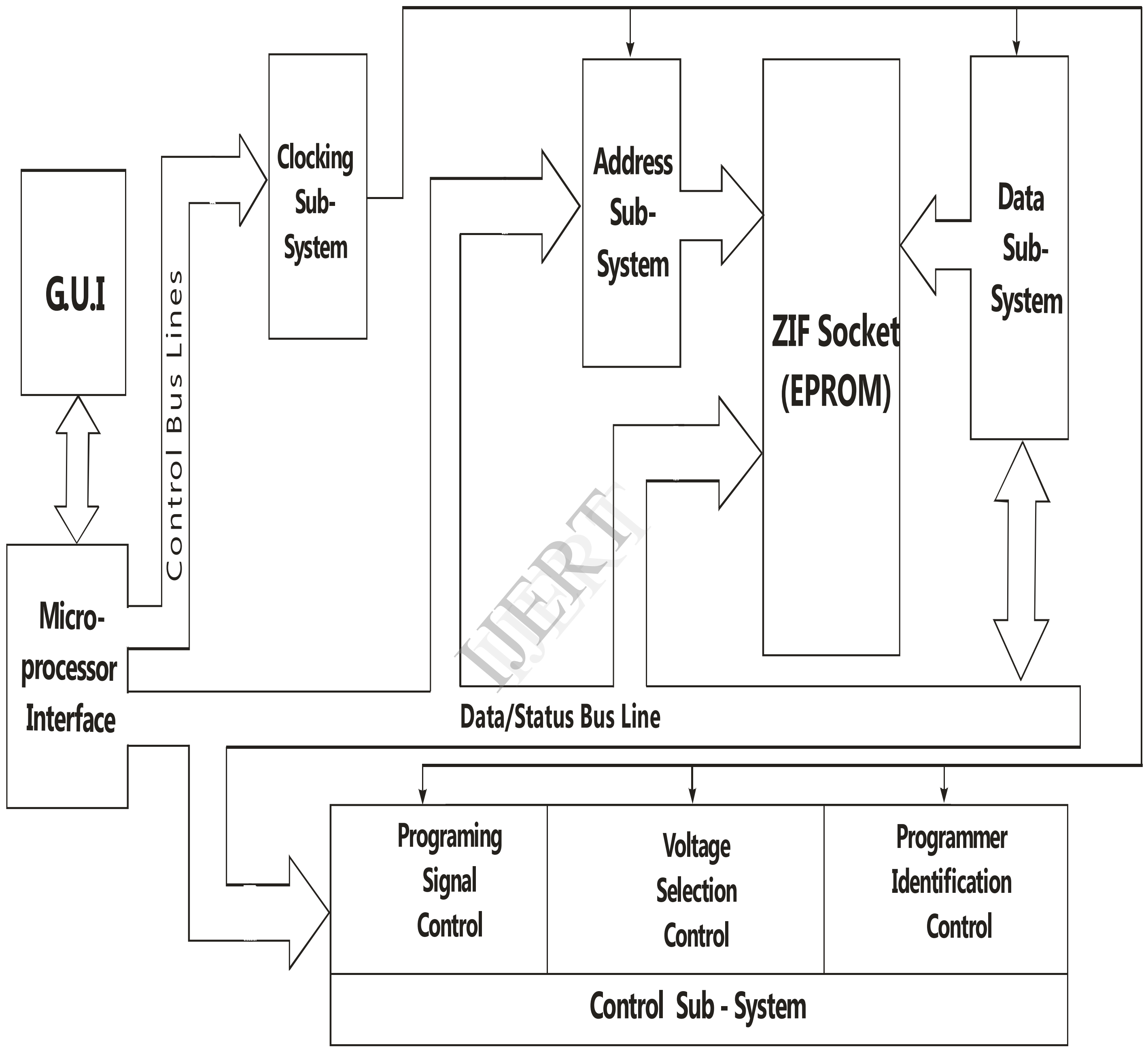
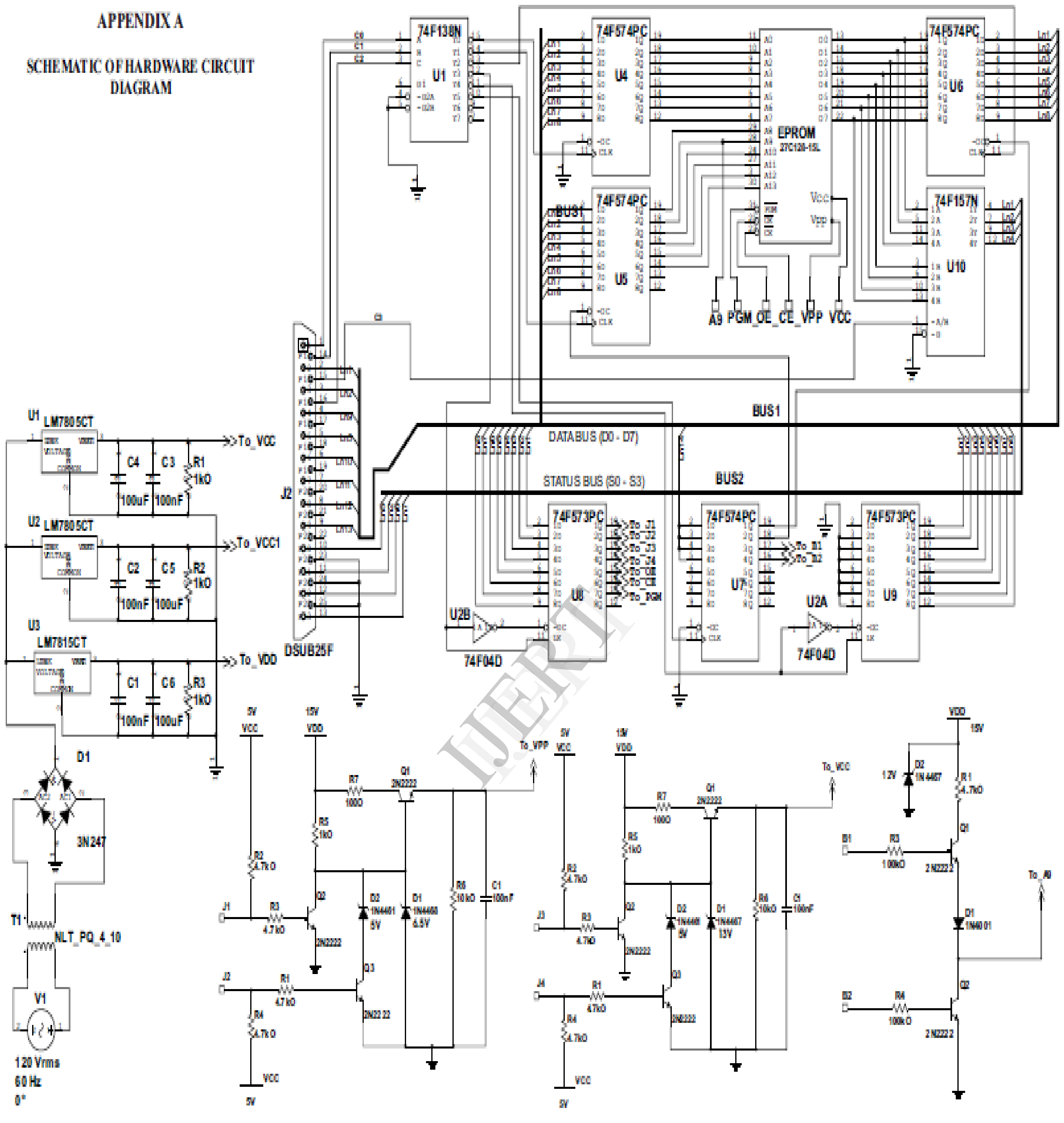


Fig. 2: Block Diagram of the System

APPENDIX A

SCHEMATIC OF HARDWARE CIRCUIT DIAGRAM



Note:
* All Arrows labeled To_XXX connect to the equivalent blocks labeled XXX.
* Power supply for the Digital ICs was omitted for clarity but is labeled VCC1.

* The O or kO attached to resistor value is Ohms or kilo Ohmes e.g. kO represent Kilo Ohms

101

Fig.3: Circuit Schematics