**Lab Experiment # 01**

Roll No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date of Submission: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Grade: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of Instructor: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **LAB PERFORMANCE INDICATOR** | SUBJECT KNOWLEDGE | DATA ANALYSIS AND INTERPRETATION | ABILITY TO CONDUCT EXPERIMENT | PRESENTATION | CALCULATION AND CODING | OBSERVATION/  RESULTS | SCORE |
|  |  |  |  |  |  |  |  |

**Familiarization to Module E16/ EV Microprocessor Based System and its usage**

**Objective:**

To become familiar with

* E16/ EV module, its components and features.
* Concepts of Address Bus & Data Bus.
* Address decoding.
* Data reading and writing on RAM memory.
* Data reading from E²PROM
* Data input/output (latch and bidirectional transceiver)
* Data transfer between digital and analog environments using A/D and D/A converters**.**

1. **Introduction:**

Before dealing directly with the study of a microprocessor system, it is useful to analyze the devices which constitute a system of this kind and their interfacing to the central unit to learn in detail the real structure which will be analyzed next. The system constitutes an effective introduction to the devices employed in microprocessor systems, starting from their theoretical and practical characteristics up to the control of the peripherals of common use in industrial applications such as A/D and D/A converters.

E16/EV shown in Fig.1 is an experiment board consisting of a screen-printed block diagram. The devices constituting a microprocessor system are mounted on the rear side of the front panel and can be interfaced by means of a BUS system. Appropriate operation of the module consists in the simulation of the most important functions of the microprocessor CPU.

The hardware components installed on the trainer are as follows

* RAM--------------2K
* EPROM------------2K
* Address decoder---2 to 4
* A/D & D/A 8-bit converter
* Latch
* I/O buffer
* Bi-directional transceiver
* Thumbwheel switches

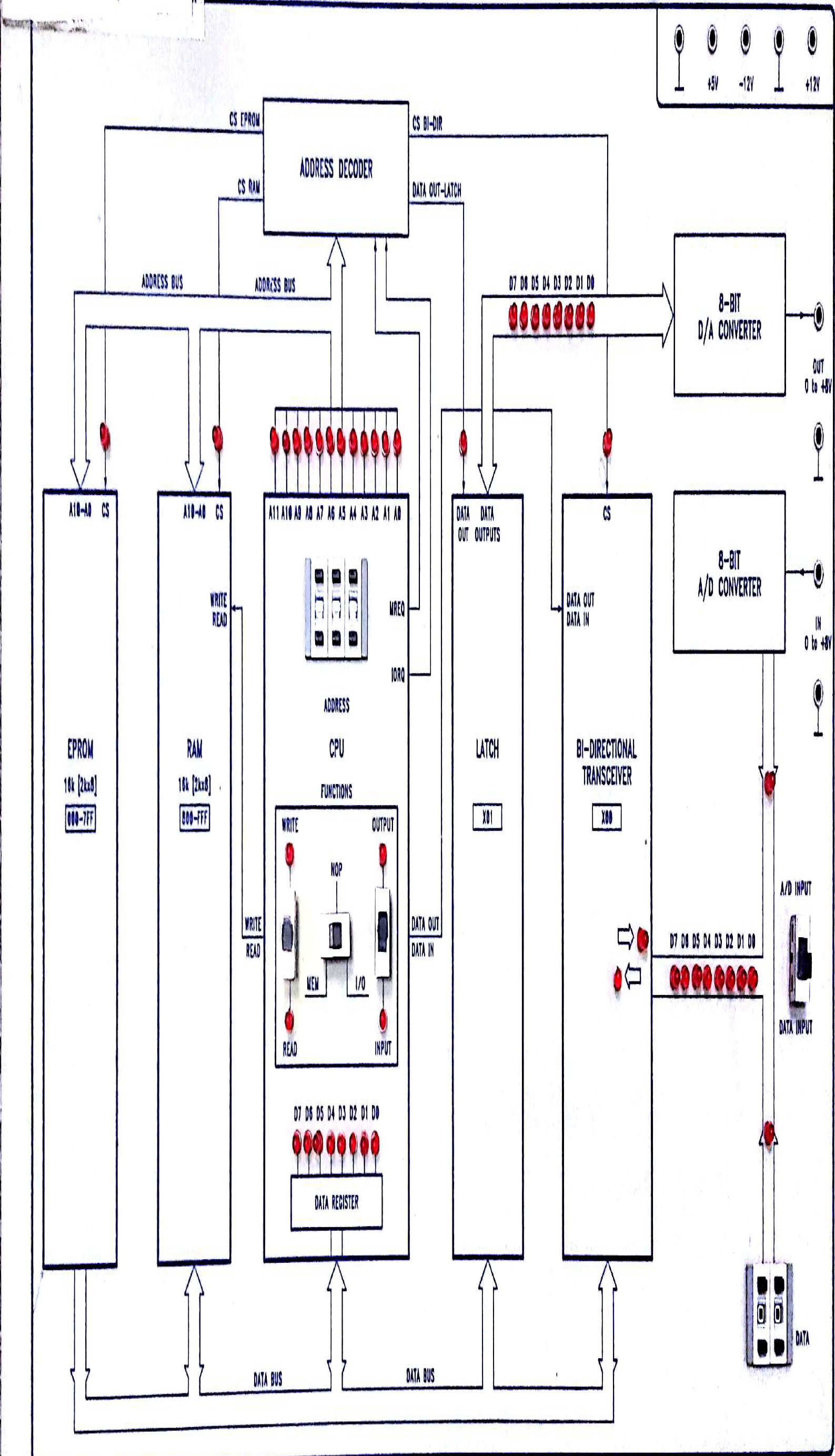
****

Fig.1 E16/EV Trainer layout

1. **Bus**

The bus is the physical medium along which the information is transferred between the blocks of a microprocessor system. There are three different types of bus; each bus carries out a specific function:

**Data Bus:** crossed by the data which are exchanged between the microprocessor and the other blocks of the computer.

**Address Bus:** Allowing the transfer of the information (addresses) for identifying a particular memory cell or an input/output device.

**Control Bus:** Allowing the transfer of the signals necessary for a correct and efficient control of the whole system.

1. **Address Decoder:**

When designing an address decoder, first of all state in which parts of the whole address field the various devices must be selected. For this this purpose, it is better to plot a map for the storage devices and another map for the input/ output devices (if the microprocessor realizes a distinction between memory and Input/ Output blocks) using the highest address lines.

There are two types of address decoding namely full decoding and partial decoding.

**Full Decoding:** Full decoding means that all the address lines are decoded and therefore all possible addresses can be accessed. Hence every device has a unique address.

**Partial Decoding:** When instead of all the address lines only indispensable lines are decoded, this method of decoding is known as partial decoding. The devices using this technique can be selected at multiple addresses.

The trainer has a single address decode both for storage and I/O devices. It consists of three integrated circuits 74LS139.

The memory decoding is carried out through a single 2 to 4 decoders; this is very simple because the memory chips are only two (1 ROM and 1 RAM), with equal storage capacity (16Kbit).

On the contrary, the decoding of the I/O device is rather complex because all the lines from A0 to A7 must be decoded: so, the adopted solution was the construction of a cascade network allowing the enabling of I/O devices when the lines from A1 to A7 are low. The last line states whether the communication occurs with the Buffer LATCH (only in output) or with the BIDIRECTIONAL TRANSCEIVER (both in input and output).

1. **RAM Access**

Figure 2 shows a typical sequence followed when writing and reading data to and from a RAM: tA is the access time.

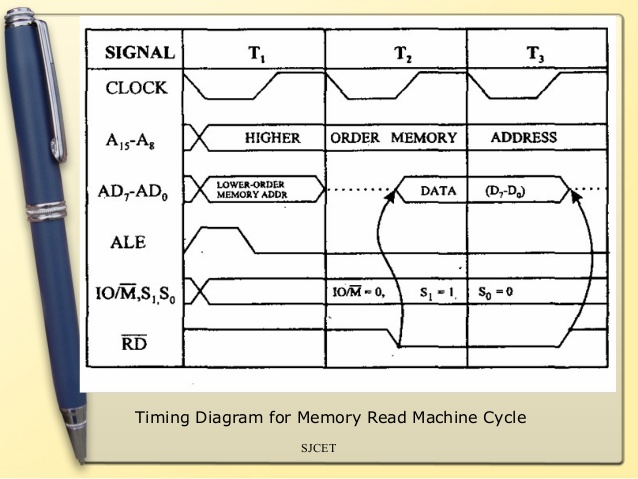
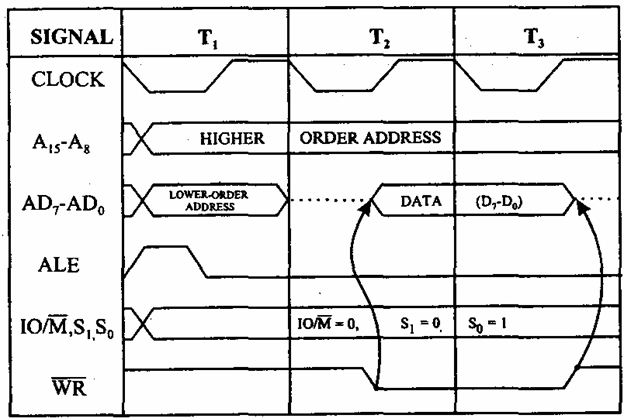


Fig.2 (a) Writing on RAM (b) Reading RAM

**Example 4.1:** Writing a data item into the RAM location $800

* Power on the module
* Set the address to 000H with pre-selectors.
* Set any data item with DATA pre selectors.
* Turn I/O function as an input and central switch to I/O and then to NOP.
* Set address $800.
* Set MEM switch to WRITE & central switch to MEM.
* Check the number on Address LEDs.
* Turn central switch to MEM and again to NOP.
* Check new data is stored in DATA REGISTER

**Example 4.2:** Reading the data item from the RAM location $800

* Power on the Module
* Following the procedure in Example 4.1 load a binary data item coming from external “DATA” pre-selector, into the CPU register. This data item must be different from that stored in the previous exercise.
* Set the central switch to NOP.
* Turn the switch MEM to READ
* Set the number $800 on the address pre-selectors.
* Check that the RAM is selected (CS signal)
* Read the data item displayed by LEDs D0-D7 in binary, and check that it corresponds exactly to the data item stored in Example 1.

1. **EPROM Access**

**Example 5.1: Reading Data from EPROM**

* Power on the module.
* Set central switch to NOP.
* Set the address to 000H with pre-selectors.
* Set the central switch to MEM.
* Check EEPROM is selected, (CS signal)
* Read data item in binary on LEDs D0-D7
* Set central switch to NOP.
* Increment address by 1 using address pre-selectors.
* Turn central switch to MEM and read data on DATA REGISTER

Table 1

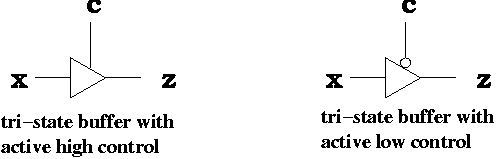
|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Contents** | **Address** | **Contents** |
| 0000 |  | 0008 |  |
| 0001 |  | 0009 |  |
| 0003 |  | 000A |  |
| 0004 |  | 000B |  |
| 0005 |  | 0010 |  |
| 0006 |  | 0011 |  |
| 0007 |  | 0012 |  |

**Tasks**

* Task.1: Fill Table.1 by reading the data starting from 0000H to 0012.
* Task.2: Can you determine which code conversion this table is performing?
* Task.3: How can you develop a Hex to Capital ASCII convertor using a lookup Table. Describe with the help of developing a new lookup table.

1. **Buffers & Latches**

**Tri State Buffer:** A tri-state buffer is a useful device that allows us to control when current passes through the device, and when it doesn't. Figure 3 shows two diagrams of tri state buffers.



(b)

(a)

Fig.3 Tri-state Buffer

A tri-state buffer has two inputs: a data input x and a control input c. The control input acts like a valve. When the control input is active, the output is the input. That is, it behaves just like a normal buffer. The "valve" is open. When the control input is not active, the output is "Z". The "valve" is open, and no electrical current flows through. Thus, even if x is 0 or 1, that value does not flow through.

This buffer can be used only for the read-out operation. The receiving device (either CPU or peripheral device) must acquire this information in that short period of time when it is available on the data bus.

In most cases, the data item sent by the central unit must be available for the peripheral device, for a rather long time with respect to the bit rate of the microprocessor without an excessive number of waiting states. This problem can be solved using LATCH BUFFERS which can hold the data item sent by the microprocessor (or by the peripheral device).

**Latch Buffers:** This is a unidirectional buffer which also includes 8 flip flops and can store the information arrived at the inputs Di, making it available at the output Yi through proper controls.

This function is carried out by the Output Control (OC) input through which what is stored in the internal flip flops can be available at the output Yi; this is obtained assigning a low logical value to OC and, in the opposite case, the output is brought to the high impedance sate.

The other control input is the Latch Enable G: it allows to store the input data. As shown in Figure 4 when this input is high, what is available at the inputs is also transfer to the output of the flip flops.

An integrated circuit SN74LS373 is included in the E16/EV Module for this purpose. Figure 4 shows the block diagram and the operating table of this IC.

In some cases, a peripheral device must either send data to the central unit and receives them from it. This problem was solved designing bidirectional devices where a proper input can determine the data direction.

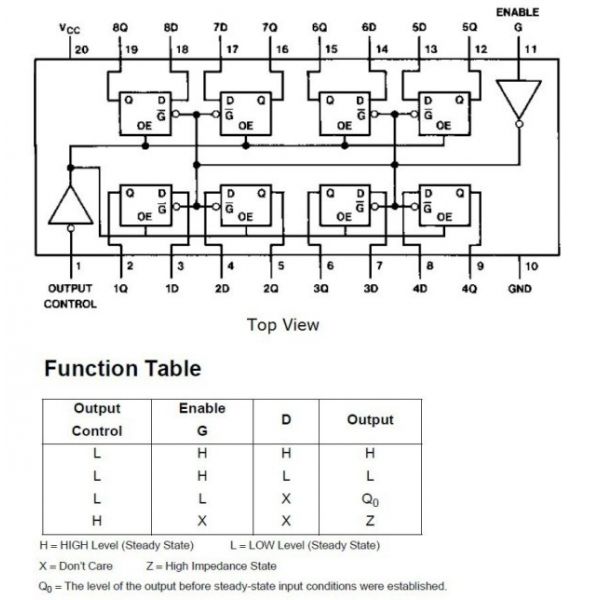
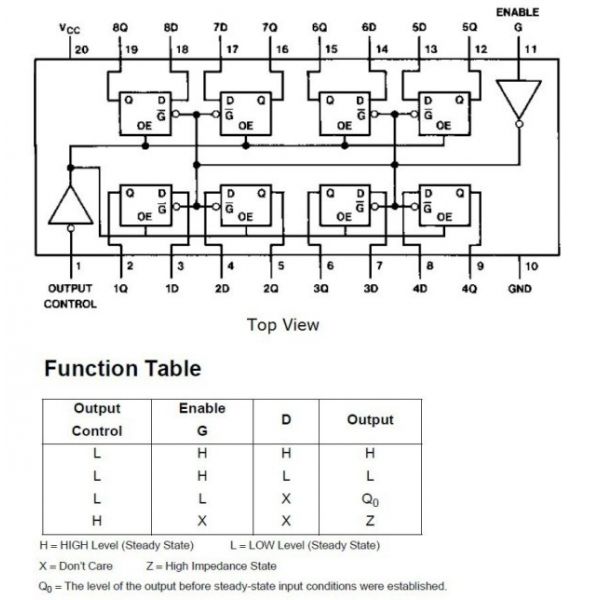


Fig.4 Block Diagram and operating Table of SN74LS373 Latch

**Bidirectional Buffers**: Bidirectional Buffer for 8-line bus allows the communication between two (internal and external) data buses, in both directions, using a minimum number of control elements.

The E16/EV Training Module uses medium scale integrated circuit SN74LS245. This device allows the data transfer from the bus A to the bus B, or from the bus B to bus A, according to the logic level of the “Direction Control” (DIR) input. The “Enable” (G) input can be used to disable the device setting to the high-impedance state, so that the buses are effectively isolated.

**Example 6.1**: Data Transfer through Latch Buffer

* Power on the Module.
* Set the central switch of the CPU to NOP position.
* Set the switch MEM to READ.
* Set the number $00F on the address preselectors.
* Check that EPROM is selected and that the data item stored in the address cell, is read.
* Position the central switch to NOP and check that the data item is stored in data register.
* Set the address X01 on the address preselector and turn the I/O switch to Output.
* Position the central switch on I/O and check that the LATCH BUFFER is selected (the LED DATA OUT is on); check that now the data item is sent along the output bus of the same latch, to the input of the A/D converter.
* Position the central switch again on NOP and check that the data item is kept stored in the output of the latch buffer.
* Repeat the operation with other data transfers from the EPROM to the output of the Latch Buffer.

**Example 6.2:** Input/ Output data transfer through bidirectional buffer.

* Power on the module
* Set the central switch of the CPU to NOP position.
* Set the switch MEM to READ.
* Set the number $003 on the address preselectors.
* Set the central switch to MEM.
* Check that EPROM is selected and that the data item stored in the address cell, is read.
* Position the central switch to NOP and check that the data item is stored in data register.
* Prearrange the address X00 on the address preselector and set the I/O switch to output.
* Set the central switch to I/O; then check that the BI-DIRECTIONAL TRANSCEIVER is selected and that the data item is sent along the buffer output bus (the output data bus is different from the input bus of the D/A converter).
* Set the central switch again to NOP and check that the data item is not held in the output of the bidirectional buffer because this is not a latch buffer.
* Repeat these operations for other data transfers from the EPROM to the output of the bidirectional buffer.

**Example 6.3**: Output/ Input data transfer through bidirectional buffer.

* Power on the module
* Set the central switch of the CPU to NOP position.
* Set the number $X00 on the address pre-selectors.
* Position the switch of input selection on DATA (position on rotary selector).
* Turn the central switch to I/O and observe the path of the data item set on the rotary DATA selector. The led of INPUT direction is on and the data item generated by the rotary selector is sent to the DATA REGISTER.
* Position the central switch again on NOP and check that the data item is stored in the data register. At this point, the readout from the peripheral device of the CPU has ended.
* Repeat these operations for other data transfers from the rotary selector to the DATA EGISTER of the CPU.

1. **A/D & D/A Conversion**

**Digital to Analog conversion**

Figure.5 shows the block diagram of the output of data in analog form, from EV16 and the steps to carry out this operation are illustrated in example 7.1

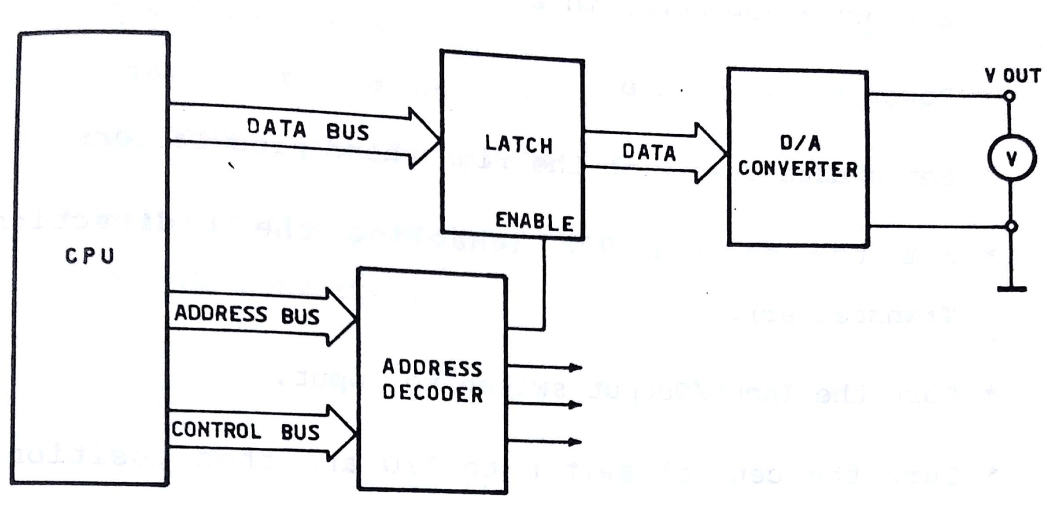
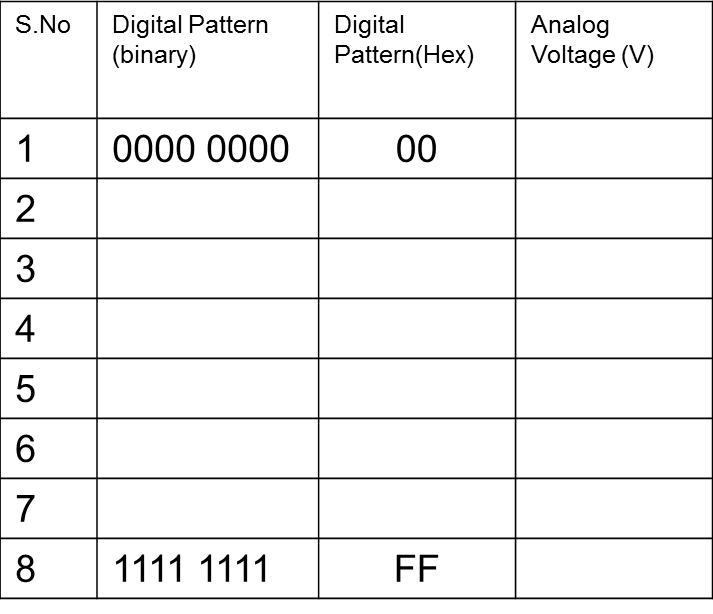


Fig.5 Block Diagram for D/A conversion in E16.

**Example 7.1** Output of Analog Data

* Fill the table
* Connect the digital MM to the op of converter (20 V D.C F.S)
* Power on the module.
* Set data item via DATA pre selectors.
* Set the address to 000H enabling bidirectional Transceiver.
* Turn I/O switch to INPUT.
* Turn central switch to IO and then to NOP.
* Check DATA REGISTER.
* Set address to 001 (enabling LATCH)
* Turn I/O switch to OUTPUT and turn MEM-IO switch to IO for a moment.
* Check that the data item has been transferred to the latch and check the corresponding voltage using Millimeter.
* N.B.: the D/A converter can convert digital data from 00 to FFH, within a voltage range going from 0 to 8 V D.C.
* Transcribe the data on table from 00H to FFH.
* Transcribe results on Graph (X-axis→ Hex value, Y-axis → Voltage)

Table.2



**Analog to Digital Conversion:**

**Example 7.2**: The input voltage is generated by an external regulated power supply which can vary the voltage from 0 to 8V.Analo signals can enter the Module according to the Block Diagram as shown in Figure.6

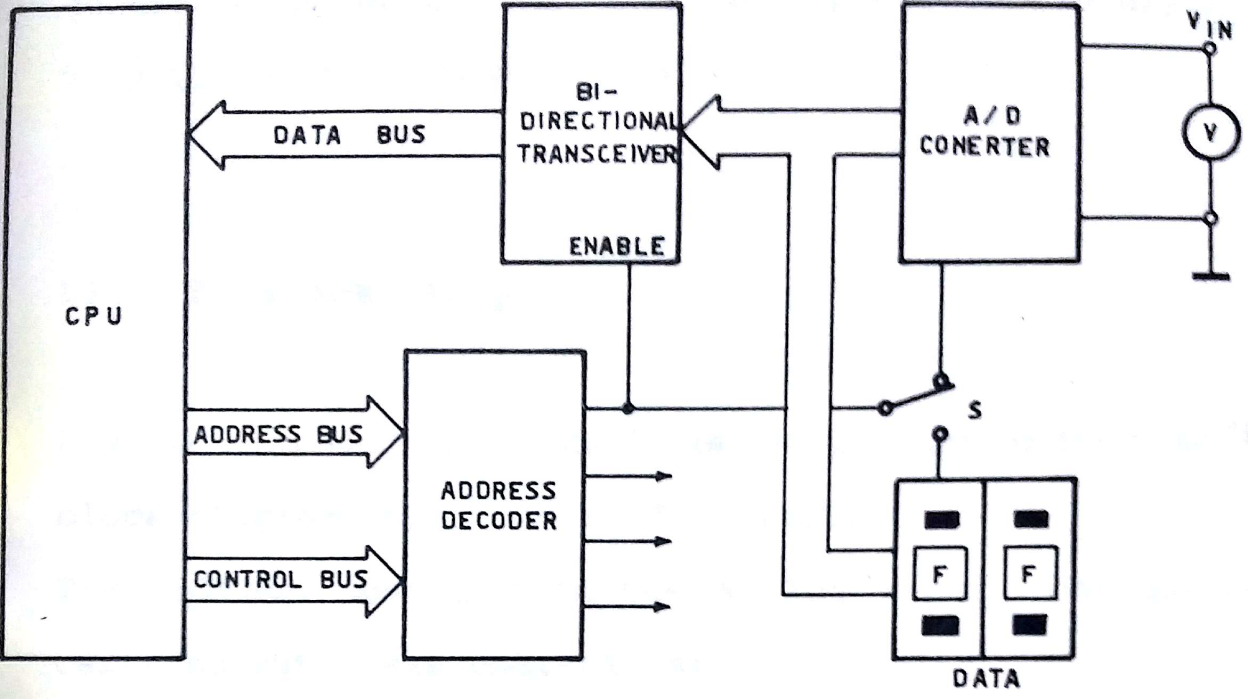


Fig.6 Block Diagram of Analog to Digital Convertor in E16

* Power on the module.
* Power on the analog input generator, without connecting i/p terminals.
* Set i/p voltage < 8V then switch on the generator.
* Turn I/O switch to INPUT and input selector to A/D converter.
* Turn MEM-I/O to I/O, and check that the digital data item delivered from the A/D converter corresponds to that of DATA REGISTER.
* Transcribe the input value and the obtained digital number on the Table.3
* Plot a graph with the sketch of voltage vs digital number.

Table.3

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Analog Voltage** | **Digital Pattern (binary)** | **Digital Pattern (Hex)** |
| 1 | 0V |  |  |
| 2 | 1V |  |  |
| 3 | 2V |  |  |
| 4 | 3V |  |  |
| 5 | 4V |  |  |
| 6 | 5V |  |  |
| 7 | 6V |  |  |
| 8 | 7V |  |  |

**Exercise Questions:**

1. Design the decoding circuitry for EV16 Trainer using address ranges for RAM, EPROM, input and Output as mentioned on the trainer.
2. Construct a binary to Gray code converter for the numbers from 1 to 8 and verify it using RAM. The binary number code should correspond to the number of memory location RAM as in Task.1 of section for EPROM Access.
3. What is the difference between Latch Buffer, Tristate Buffer and Bi-Directional Buffer?