Logical Reality

Terminus based Training Manual
for Digital Electronics

Hasib Rahman

Edition One

For Terminus 1.1A

Terminus
UART based
Testbench generator

Arbiter
Electrotech
innovation . unlimited

Logical Reality:

A Terminus based Training Manual For Digital Electronics

Author: Hasib Rahman

1st Edition 2021 (Debut Edition)

Language: English

Intended Terminus version: Terminus V1.1A (Hardware V-1.2
Firmware V-2.3)

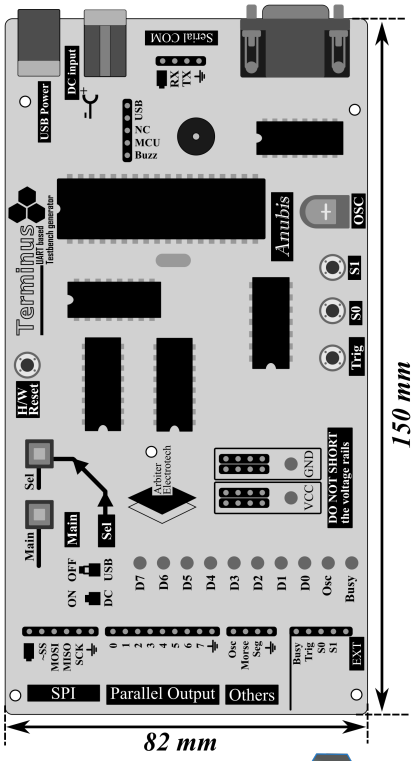
Cover & Illustration: Team AE

All images used in this literature are either extracted from source that are in public domain or were generated by Team AE
All illustrations and photographs (of electrical components) were produced and modified to suit the purpose by Team AE and are intellectual property of Arbiter Electrotech

This entire literature is to be considered as intellectual property of Arbiter Electrotech.

Table Of Content

Pinout & Brief Descrption



1. DC Barrel Jack
Provide Power (5V DC) to the Terminus
NOT RECOMMENDED

2. USB Port
Provide Power (5V DC) to the Terminus
RECOMMENDED

3. Function Header
Header to activate the USB power and Enable/ disable Buzzer for Morse Code Generator

4. Reset Button
Hardware Reset the whole system

5. Power Select Switch
Select power source, select between USB or External DC

6. Power Switch
Main power ON/OFF (push) switch
Lock - Power ON
Release - Power OFF

7. Logic Level LED
LED to show status of Parallel Output. LED will be lit for signal HIGH and off for signal LOW. Note that these LED cannot replace a proper Logic Probe

8. Output Pins
Signal Output pins for all Terminus Functions

SPI - Pins for the SPI Bus Generator

Parallel Output - 8-bit wide output port for functions requiring 8-bit data output

Others - Output of the Oscillator, Morse Code and Segment Selection of Seven Segment Display Driver

9. Interactive I/O Pins
Pins meant for syncing/controlling the using external signals/systems

Busy - Shows if theTerminus is Busy. Busy is shown by HIGH output

Trig - Trigger the Terminus externally

S0 , S1 - Multi purpose function inputs

10. Notification LED
LED showing logic status of Oscillator or Busy Output

Busy - Led is lit only if the system is Busy

Osc - LED is lit if the oscillator signal is HIGH

11. Power Rail
Power Bus to power external test Circuits 5V 400mA max

12. Function Buttons
Buttons connected to the EXT pins for convenience

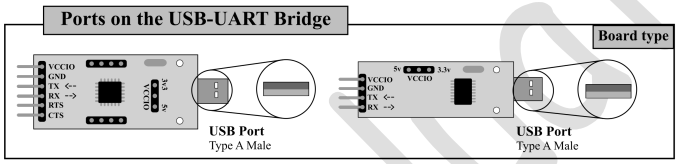
13. Oscillator Adjust
Knob to adjust parametrs of Oscillator in VCF mode

14. DB-9 Port
Used to connect Terminus to the Computer via a Serial-UART Bridge

15. Serial Communication
Used to connect Terminus to the Computer via a Serial-UART Bridge and provide power (5V DC)

Terminus
UART based
Testbench generator

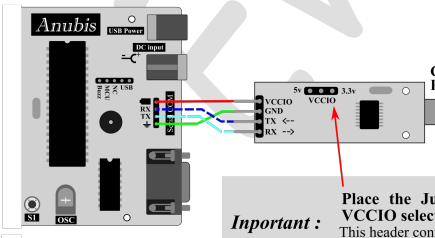
Connecting Terminus to the PC via Serial COM



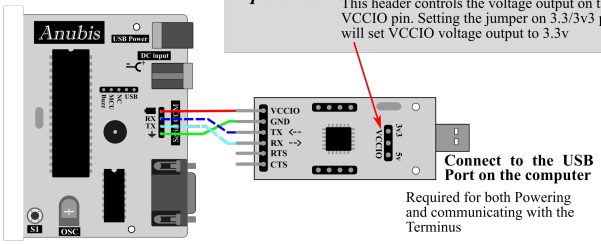
Pinout :
VCCIO - VCC (+5v Power)
GND - Common Ground return
TX - Transmit (signal going out of the Computer to the Terminus)
RX - Receive (signal going out of the Terminus to the computer)

Info :  A Jumper (shown on the left) is a small plastic covered metal link, used to short two consecutive header pins

Always Connect
Tip : Tx ⟶ Rx
Rx ⟵ Tx

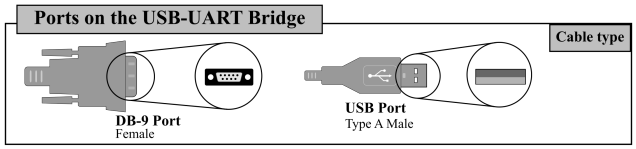


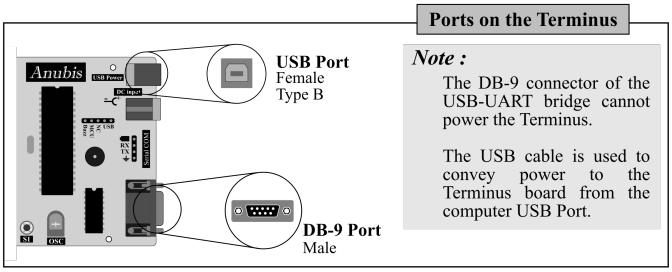
Important :
Place the Jumper on the 5v pin on VCCIO selection header
This header controls the voltage output on the VCCIO pin. Setting the jumper on 3.3/3v3 pin will set VCCIO voltage output to 3.3v

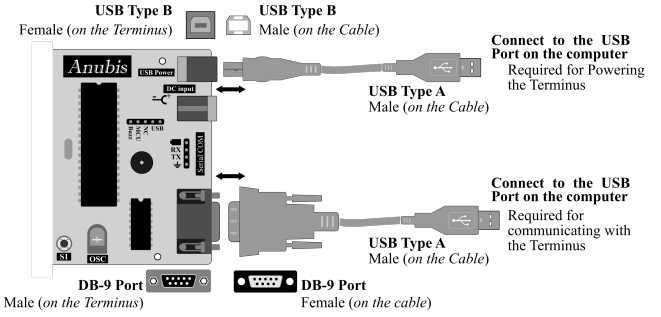


Note :
Only 1 USB Port is required on the computer
Same USB Port is used to provide Power and establish communication between the Terminus and the computer via the RS-232-UART Bridge

Connecting Terminus to the PC via RS-232 Serial Port

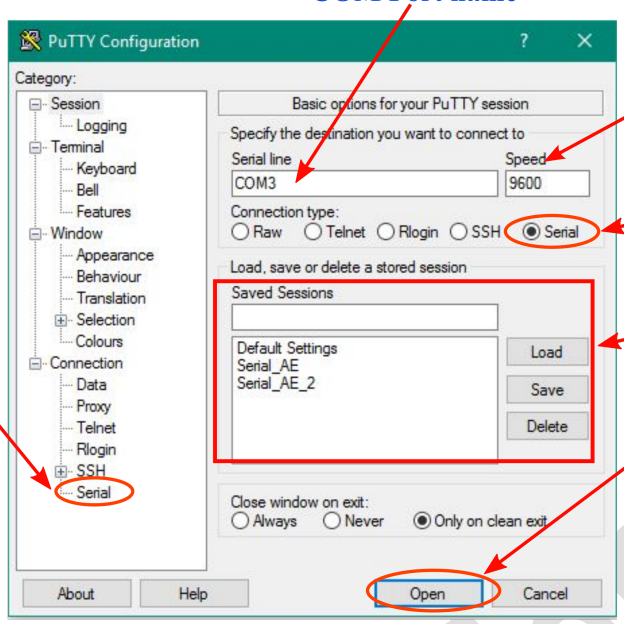






Note :
2 USB Ports are required on the computer
One USB Port will provide power to the Terminus via the USB cable
Another is used establish communication between the Terminus and the computer via the RS-232-UART Bridge

Configuring PuTTY



Baud Rate

Select "Serial" connection type

Settings can be saved for easy startup in the future

Select "Serial" option

Open Terminal after setup is complete

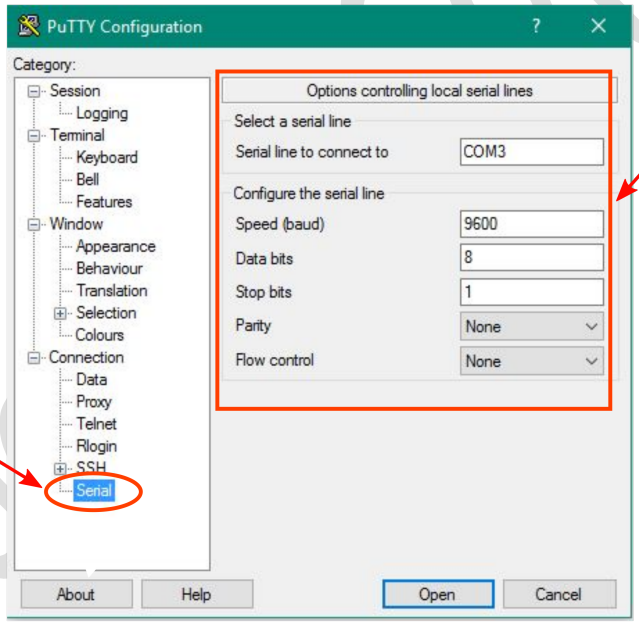
Steps:

1. Select the Serial option
2. Configure the serial communication
3. Select OK

COM Port Configuration :

Baud Rate - 9600
Data Bits - 8
Stop Bit - 1

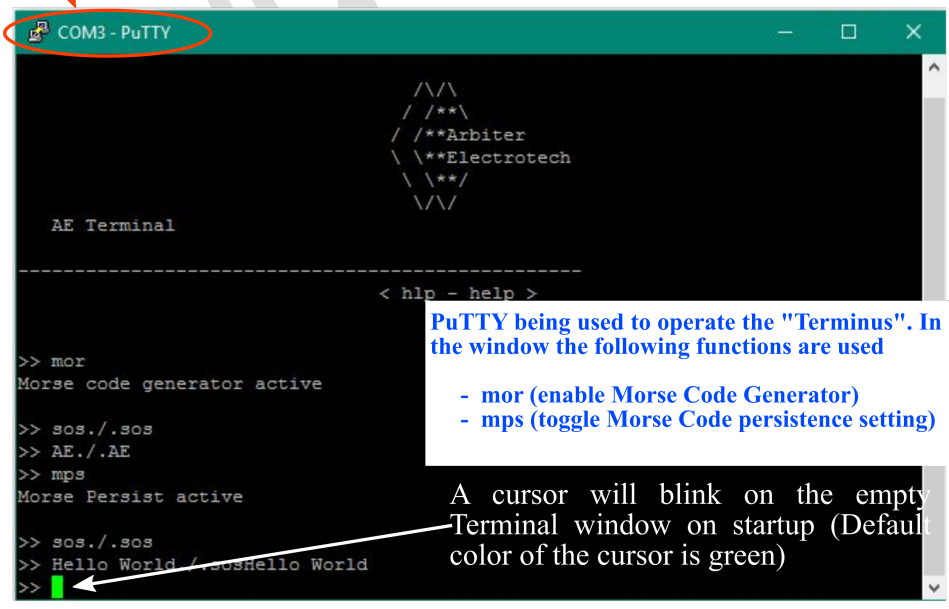
Parity - None
Flow Control - None



Configure the Serial Port by modifying these values

Select "Serial" option

Name of the currently connected Serial Device is shown in the Title bar



PuTTY being used to operate the "Terminus". In the window the following functions are used

- mor (enable Morse Code Generator)
- mps (toggle Morse Code persistence setting)

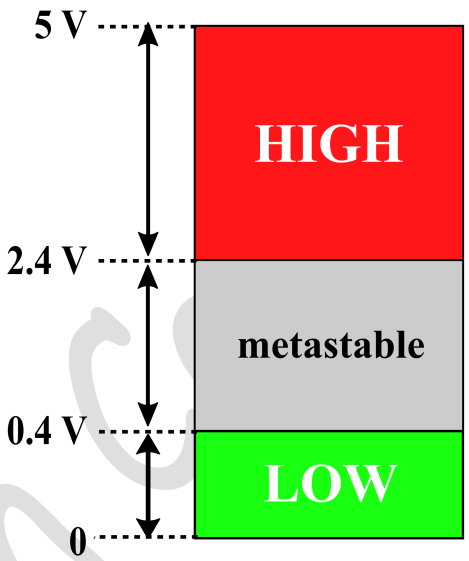
A cursor will blink on the empty Terminal window on startup (Default color of the cursor is green)

Terminus UART based Testbench generator

TTL of Transistor Transistor Logic is a very widely used and popular member of the digital logic family. In TTL standard, any voltage that is greater than or equal to 2.4v is a digital HIGH signal. Any voltage that is less than or equal to 0.4v is a digital LOW signal. Voltages between the 0.4v and 2.4v are not used in digital electronics and is known as metastable. These voltage levels are same for both input and output.

TTL Logic Level

- Voltage level for TTL standard
  - HIGH (Logic 1) :: 2.4v to 5v
  - LOW (Logic 0) :: 0.0v to 0.4v
  - Metastable :: 0.5v to 2.3v
- Any signal with voltage level within the metastable state is rejected by digital electronic components. In fact, metastable state is as same as the digital input connected to a high impedance source or kept in an open circuit condition.



Note that the voltage range for HIGH is 2.4v to 5v. 5v is the maximum voltage that can be applied to the gate input using TTL logic standard. Voltages greater than 5v will permanently damage the logic gate and the digital circuit. Although there are logic family members that can take voltages up to 20v, the modern trend is to minimize the voltage levels for both signal and power. Devices with lower voltage levels consumes less power than their higher voltage counterpart.

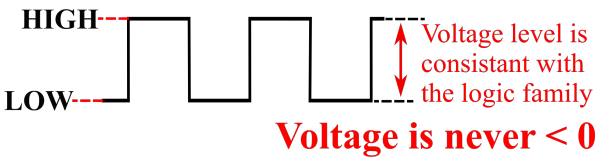
Both the input and output of any digital component is digital as well. As mentioned earlier, both input and output of digital components are expressed in terms of HIGH or LOW instead of voltage level. The voltage level of HIGH and LOW are referred to as “Digital State” or just “State”. So when anyone say that certain digital component input is LOW and output is HIGH, what they mean is that the digital component’s input state is LOW and output state is HIGH.

As explained before, digital electronics always show the result in term of “Positive result” or “Negative result” i.e. Yes or No. A positive result is shown by HIGH state. A negative result is shown by LOW state. So in a sense, digital systems are like a pass-fail system. The system takes in input(s), process the input(s) using the predefined rules and if the input passes the rules, the output is HIGH or positive, showing a pass. Else the output is LOW or negative, showing a fail. In digital systems, there is no maybe or in-between. Note that, all digital electronics components or devices has an inherent predefined rules that it uses to process the inputs and provide the proper output.

To determine if the output of any digital circuit is HIGH or LOW, we can use a simple LED (Light Emitting Diode). As LED work only when the applied voltage is greater than its forward voltage, we can use it to determine if the signal is HIGH or LOW. Typical forward voltage of LED is 1.2v – 1.8v and TTL output

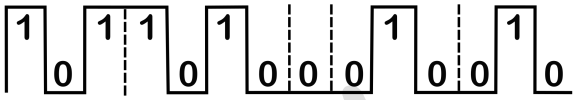
Digital Signals Intro

- Digital signals has only two state, HIGH & LOW
- All digital signal will only toggle between these two states (1 & 0)
- Digital signals are always Square or rectangular in shape
- All digital signals has sharp corners
- Digital signals with rounded corners considered to be of lower quality or noisy
- Digital data is transmitted and stored in a series of HIGH & LOW i.e. 1 & 0
- Some attributes associated with digital signal is Rising or Leading Edge and Falling or Trailing edge
- Transition of signal from 0 to 1 or 1 to 0 is called an Edge

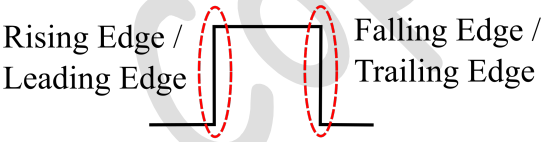


Voltage is never < 0

Digital data shown in waveform



Data - 10110100010010



Signal going from LOW to HIGH is called a **" Rising / Leading Edge "**

Signal going from HIGH to LOW is called a **" Falling / Trailing Edge "**

Terminus UART based Testbench generator

♫ • Logic Gates

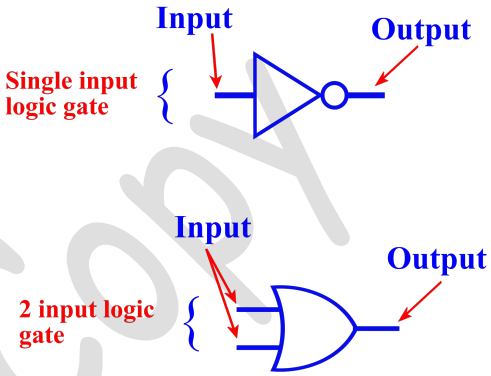
Logic gates are the fundamental building blocks of digital electronics. According to dictionary, “Gate” means “a barrier that can allow or stop anything from going in or out of a specific area”. And “Logic” means “reasoning based on certain set of rules”. Following these literal meanings, Logic Gates are supposed to be barriers that are operated (opened or closed) following some specific set of rules. In reality, that is exactly what happens. Logic gates are electronic devices that opens and closes based on given set of rules. Signals go in to the logic gates through the input. If the signals can satisfy the predefined set of rules or condition, the signals can come out of the gate output. Sometimes signals are modified when they are in the logic IC/chip and the output signal is different to what went in.

The set of rules are predefined. There are seven different fundamental digital logic gates, each with its own set of predefined rules (the rules of fundamental logic gates cannot be changed). These logic gates can be combined to make any other logical gates and logic circuits that you can think of. By the way, logic circuits are a type of digital circuit that can perform logical operations based on the predefined rules. Logic gates are always shown in the schematic by specific symbols. Each gate has its own unique symbol. There are seven different types of logic gates. Of these 7 types, 5 are unique logic gates. The rest 2 are called “**Universal gates**”. Universal gates are also unique but they have special property. All universal gates can be used to make any of the 5 fundamental logic gates but not the other way around.

Logic gates always comes in IC form. You can never buy a single logic gate. Each logic gate IC contains several logic gates. Generally, one IC contain multiple gates of same type however, there are some IC that has multiple types of gates in one IC/chip (ICs are also referred to as chip). The gates are briefly described in the list below. These gates are very simple to begin with and so there is not much to describe anyway. Anyhow, all these gates take in input, process it based on predefined rule and give the result on output. The predefined rules are different for each gates. All input and output are described in terms of state.

Logic Gate pinout

- Every logic gate has a unique symbol to represent it. These symbols are called Schematic symbol
- Logic gates has one or more input and always only one output. However, Except NOT gate, every single logic gate has multiple input
- Most logic gates comes in many different versions containing 2, 4 or up to 8 inputs
- Logic gates of different logic families have different follow the respective digital logic standards however, the schematic symbol is same for all logic gates of the same name. E.g. AND gate of TTL and CMOS standard has same symbol.
- All logic gates are designed to operate within a maximum frequency range. Logic gate will give out false result if Input signal frequency is higher maximum limit
- Input voltage higher than the absolute maximum input voltage threshold will permanently damage the logic gate

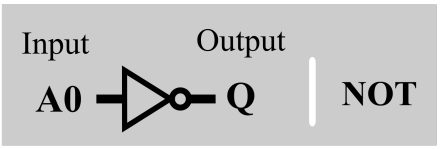


Truth Table

- A " Truth Table " is a table that shows the input-output combinations of any logic gate or digital component/circuit
- Truth table is a very effective way of showing the functionality of the digital component and is widely used by engineers and in industry

A digital HIGH is represented with binary 1
A digital LOW is represented with binary 0

In the example, a NOT gate is used to show the truth table



Input-Output Combination

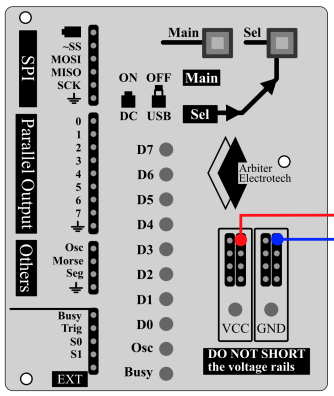
Inputs	Outputs
A0	Q
HIGH	LOW
LOW	HIGH

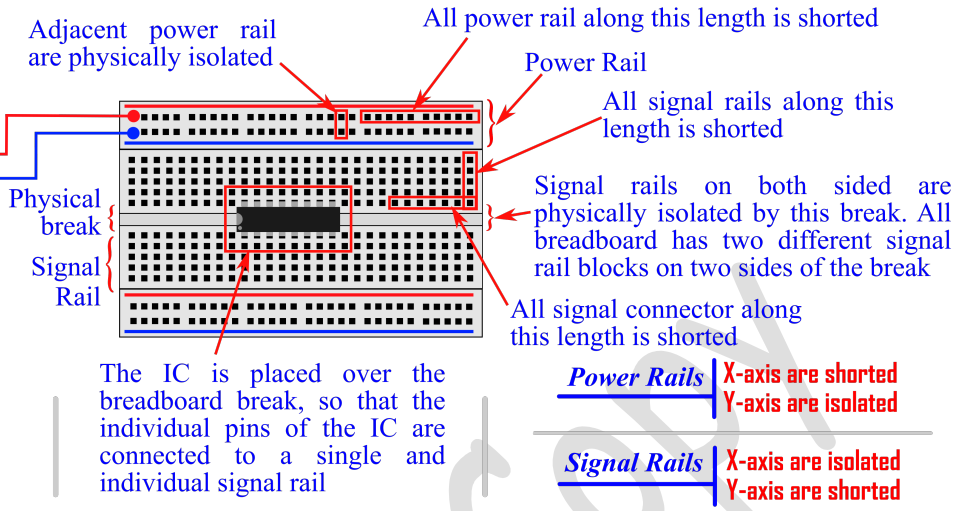
Truth Table

Inputs	Outputs
A0	Q
0	1
1	0

Terminus & Breadboard Intro

This section shows how an IC is placed on the breadboard and how Terminus power rails are connected to the breadboard power rails, along with the other fundamentals of breadboard.





Power Rails VCC is painted Red, GND is painted Blue

Power Rails X-axis are shorted, Y-axis are isolated

Signal Rails X-axis are isolated, Y-axis are shorted

Terminus Intro

Before jumping into the experiments, it would be appropriate to learn what Terminus is and how it is suppose to work. The details are unfolded as we advance through the literature and the experiments, but for now, a brief introduction

Terminus is a device that enables its user to generate a broad spectrum of user-definable signals for designing their own circuits or debugging/testing any other devices/ systems. For maximum efficiency, Terminus uses a Linux like Command Line Interface (CLI). A computer has to be used to access and modify all features and parameters of the Terminus respectively. The commands are given by the user from the terminal program running in the computer.

Terminus
UART based
Testbench generator

Powered by Arcon Interface

All control, configure and operating commands must come from a computer via a serial port. The serial port used here is based on RS-232 (built into the MCU). The host computer can connect either using the serial Port (older computers have serial port built in) or by using a USB-UART Bridge. Terminus uses third party open source Terminal program for computer. The terminal programs are available free of cost. However, the user can use any other terminal program as long as the program(s) are within the operating specifications and requirements of the Terminus.

Included Toools

- Oscillator
- Pattern Generator
- Manual Byte Generator
- Random Byte Generator
- Finite Pulse Generator
- Morse Code Generator
- Sweep Generator
- Servo Motor Test Signal Generator
- Stepper Motor Test Signal Generator
- DC Motor Test Signal Generator
- SPI Test Signal Generator
- Seven Segment Display Test Generator

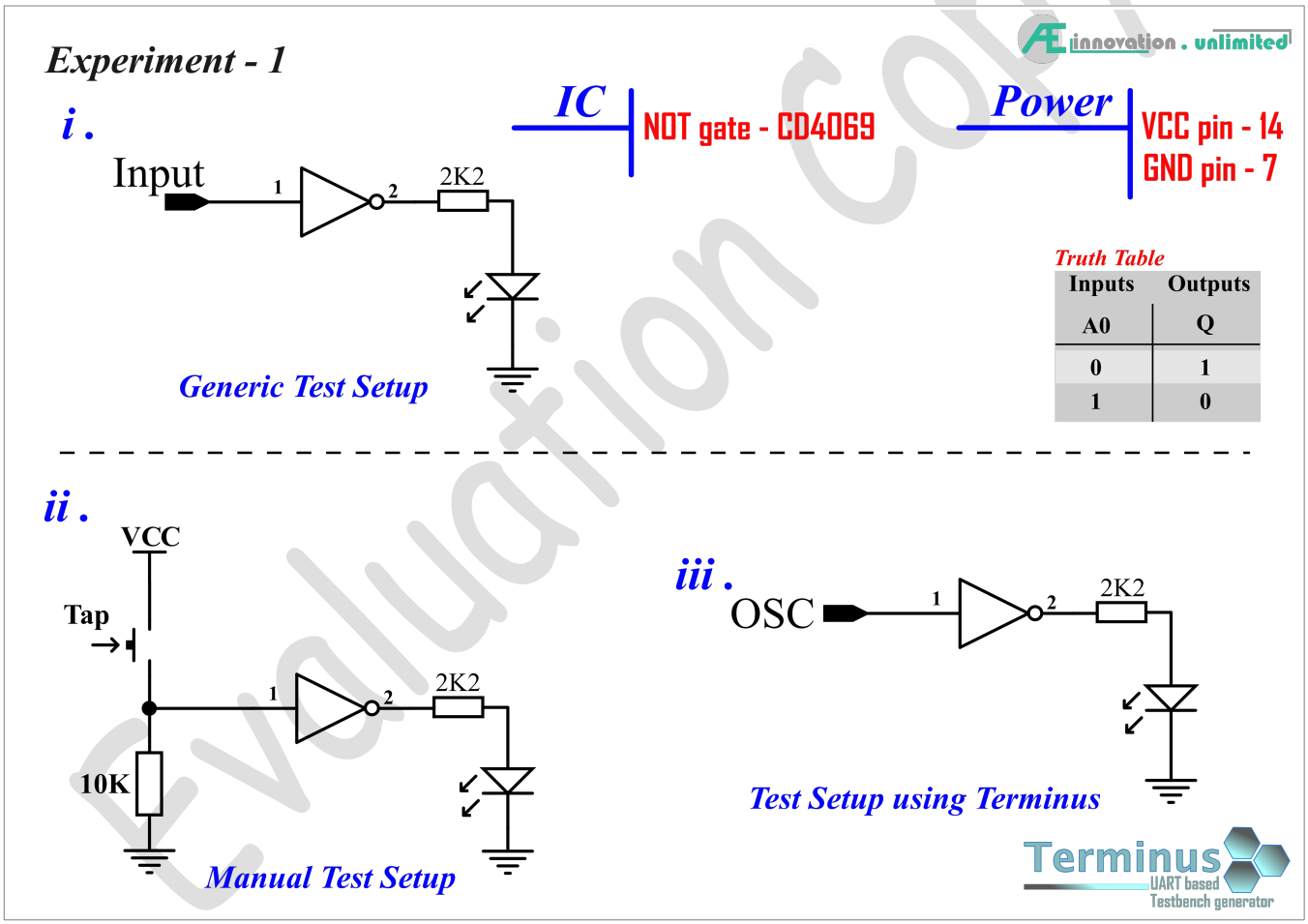
Arcon Interface

Arcon Interface is a propitiatory Command Line Interface (CLI), designed and coded by Arbiter Electrotech. Arcon Interface allows the host device to receive commands and transmit data to the connecting computer using RS-232 based Serial Port of the computer and the UART of the microcontroller. The CLI is embedded within the firmware and is an integral part of the product itself. The backbone of Terminus is the Arcon Interface. Every single command and notification is processed and generated by the Arcon Interface. Arcon Interface is optimized to be highly efficient and is used in several products developed by Arbiter Electrotech.

Experiment – 1

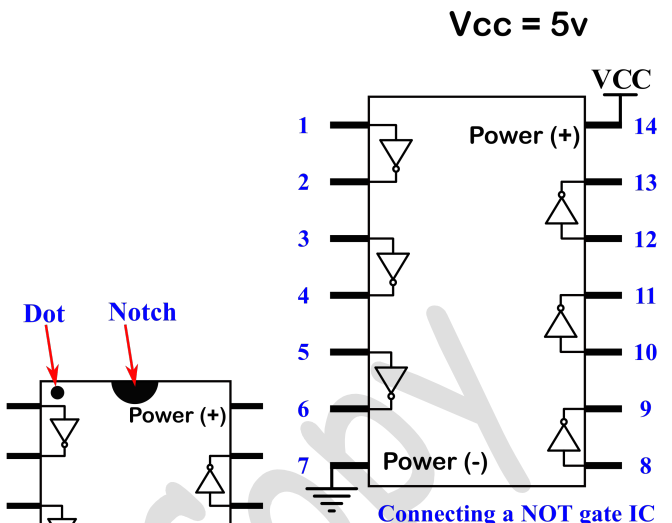
Experiment 1 is the first experiment of this literature. This experiment will demonstrate the basic working principle of digital logic. However, before starting experiments with the logic gates, it is important to know some basic and fundamental concepts relating to hardware. An engineer must know how to power the DUT/SUT, how to apply test signals on the DUT/SUT and the meaning of the output signals. For your information, DUT means “Device Under Test “and SUT means “System Under Test”.

We will start with the simplest logic gate, The NOT gate or Inverter.. NOT gates have one input and one output. The output is always inverse or 2's compliment of the input. Schematic of the circuit is shown below. Note that there are three circuits with only difference on the input. The circuit (i) shows how NOT gate is connected without any input circuit. The circuit (ii) shows the convenient method of testing by connecting a button. The circuit (iii) makes use of the Terminus.



Connecting an IC to Power Rail

- All logic gates are sold in IC form
- Each IC contains several single type of logic gates
- Logic gates must be powered before they can be used
- Powering up the IC will power up all the gates within the IC
- The voltage of the power rail must not exceed 5 volts
- To identify pin number of logic gates, physical markings are placed on the IC/chip
- First pin on the Left side of the Notch is pin 1, also first pin on the beside the dot is pin 1
- All IC have either the Notch or Dot or both
- Generally the two diagonal pins are assigned as power input e.g. pin 7 for GND and pin 14 for VCC but some manufacturers may not follow this convention



Connecting a NOT gate IC to Power rails

Basic physical layout of a NOT gate IC

Components:

a. **One** CD4069 IC (NOT gate)
b. **One** 10KΩ Resistor, **One** 2.2KΩ Resistor
c. **One** LED (any color)
d. **One** tap switch

Steps:

1) Set up the circuit No (ii) using schematic above. The schematic shows a single gate connected to GND via a resistor. From the NOT gate pinout seen earlier, it is seen that the input of the NOT gate is at Pin 1 and output is at Pin 2. So, the LED is connected to Pin 2 via the resistor and the input signal goes to the Pin 1 via the 10KΩ resistor.
2) Connect the Terminus power to the breadboard power rails
3) Currently, it is connected to GND via a 10K resistor, and therefore, the NOT gate input is LOW and the output LED is glowing.
4) Push the tap switch to feed the input side a digital HIGH by connecting it to VCC. Observe the LED. The LED should stop glowing.
5) Next, release the switch and feed the input side a digital LOW by connecting it to GND and observe the LED. The LED should be glowing.

Automated Test (Optional)

Terminus can be used to show the properties of the NOT gate. Using the oscillator function, the test would be fully automated. Follow the steps below.

Steps:

1. Connect the NOT gate (circuit iii) input to Terminus Oscillator output (OSC)
2. In the terminal, write the following command, one line at a time to start the oscillator

Note: *Brackets* [] *are used to show the command and the key press event*
[Enter] *means enter/return button on the keyboard needs to be pushed once*
The commands are to be typed in without the brackets (refer to user manual for details)

Command:

>> [osc] [Enter] – *Start the oscillator*
>> [1] [Enter] – *Set oscillator frequency to 1HZ*
>> [dty] [Enter] – *Set oscillator duty cycle*
>> [50] [Enter] – *Set oscillator duty cycle to 50%*
>> [;] [Enter] – *Run the oscillator*

Continue Oscillator until the experiment is complete

>> [osc] [Enter] – *Stop the oscillator when the experiment is complete*

To explain what happened in the experiment, we need to use the truth table (see the schematic). We are using a NOT gate. Now see the steps that we have just used in the experiment. Notice that, we started off the experiment with the input of the NOT gate connected to the GND rail. The voltage of ground rail (GND) is zero volts (0v). 0 volts meets the standard set for logic LOW. As the input of the NOT gate is logic LOW, the output is inverted i.e. output is HIGH. For logic LOW, there should be approximately 0 volts on the output of the gate and so the LED doesn't glow.

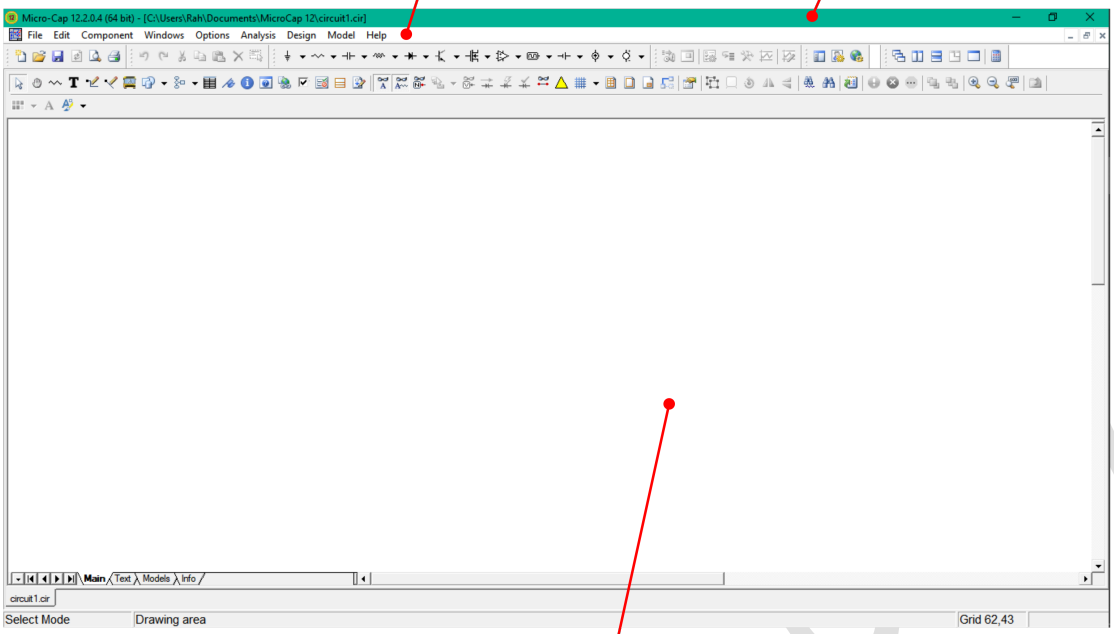
Next we connected the input of the NOT gate to positive voltage rail (VCC). The voltage of the ground voltage rail is approx. +5v. Voltage equal or greater than 2.4v meets the requirement for logic HIGH. Again, for input of logic HIGH, the output is inverted to logic LOW. For logic LOW, there is approx. 5 volts on the output of the gate and so the LED starts glowing.

Connecting a LED to the output of the logic gate is a great way to determine the output state. However, there is a drawback to this technique. If the digital signal is LOW, the LED stays OFF, and the LED is ON if the digital state is HIGH. But if the input is at metastable state, the LED maybe ON, which is false positive. A LED will light up as long as the input voltage is greater than its forward voltage. Therefore, LED lights up for both digital HIGH and metastable state. To determine the digital state properly, a Logic Probe is used (schematic is given on the DIY Projects chapter).

Micro-Cap Workspace

Menubar
Lists All available menu

Titlebar
Shows the name of the present project and saved directory

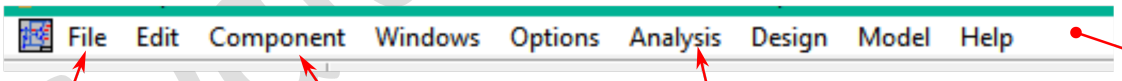


Toolbar
Shortcut to Tools generally housed in the menu

Drawing Area
The circuit is constructed and modified in this area and the circuit make here is simulated by the program

MicroCap Intro

- Micro-Cap is an extremely powerful electronics simulator however, we will be using only the basic features of this simulator
- Only three options of the menubar is important for us namely File, Components and Analysis



Menu Bar

Creating New projects, Saving the current projects or loading a saved project is controlled from here

Holds all the different components, signals and stimulus generators

Holds different available options for running the simulation

Toolbar (Compnents mode)
The components can be picked from here directly instead of picking from menubar

Toolbar (Mode Selection)

Select mode
Allows selection and modification of components, shapes wires and all objects on the drawing space

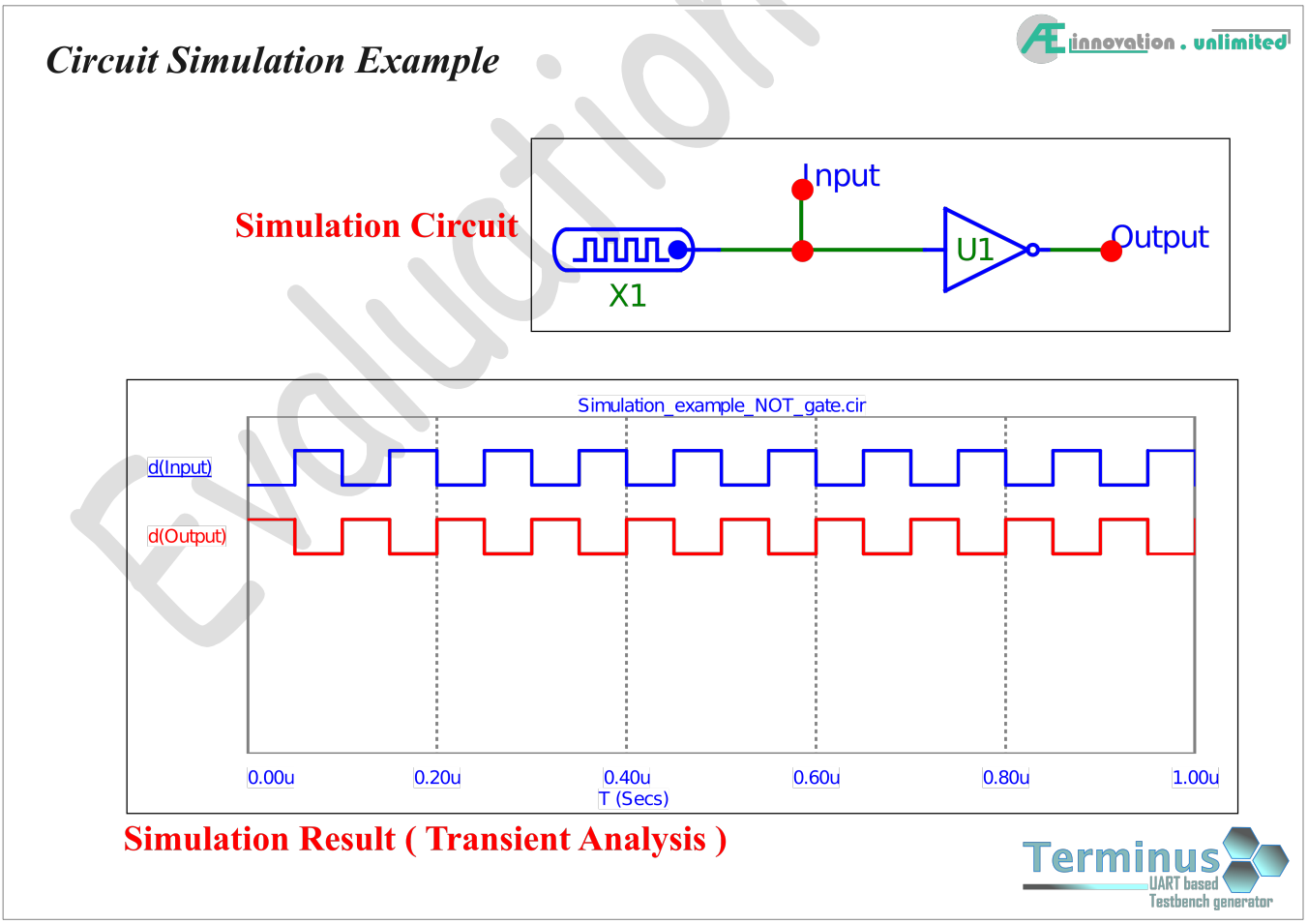
Wire mode
Allows user to place connecting wire on the circuit

Text mode
Allows user to place test object on the circuit

Simulation examples demonstrates how the Micro-Cap can be used to simulate circuits. Step by step instruction is provided. Reader only need to follow them in perfect order. Engineering or any other branch of science requires a lot of time and effort to be invested in experimentation. Its is important for an engineer to have the ability to learn by himself/herself. Learning own your own is a great way to master a craft. It should be noted that the objective of this literature is not to makes the reader a master of digital electronics. In fact no one can make anyone else a master at anything. It is the reader who has to invest time and effort to master digital electronics. This literature will guide the reader and introduce them to ideas and existing techniques. Therefore, only this chapter is provided with the instructions to use the simulator. Later chapters will not include step by step instructions. The name of the simulation file is given. In the file name of the parts, the location of the parts and the analysis type will be included. The readers have to open the file in Micro-Cap and either run the analysis or make the circuit themselves from the list of the components. It is expected that the reader will try and learn by themselves. However, the experiments involving physical components will have the step by step instruction.

Simulation Example – 1

For the very first simulation, we start with something very simple. We start by simulating a NOT gate. The simulation models and steps are shown below. Follow them perfectly and you will be presented with a waveform of the simulation. The intended circuit and the final result is shown below. The most interesting thing about these simulations is that power supply is not needed i.e. the simulation assumes the devices are powered.



For this simulation, detailed step by step instruction is provided. The simulation process requires manipulation of some parameters. The details of these parameters and the dialogue that holds them are shown after the step instructions.

Chapter – 5
Optimization of Digital Circuits

Optimization is a core part of design engineering. Optimization can make the system faster, more efficient and less power consuming. On the financial side, optimization can reduce the manufacturing cost, make the product smaller which reduces weight and lower manufacturing cost. Smaller and lighter products are also easier to store and are less expensive to transport.

• Karnaugh Map

Karnaugh Map or K map is a very simple optimization process for digital electronic circuits. K map is possible only for Boolean expressions. To get a k map for a specific digital system, first the Boolean expression for the system is generated. The Boolean expression is then standardized by expressing it as SOP or POS. The standard SOP or POS is then plotted in to the K map to get the optimum Boolean expression. Generally, the circuit gets significantly smaller and less complex after K map optimization. The circuit is optimized as K mapping can reduce the number of logic gates used in the design. It is possible to get K map from either the Boolean expression or directly from the truth table.

Karnaugh Map

- Karnaugh Map or K map is a method of optimizing the digital circuit
- K map can be used to minimize the number of variables from an SOP expression
- To use K map, the Boolean expression must be first expressed as a SOP
- K map can reduce the number of gates from the digital circuit
- Optimization using k map is possible only for circuits that are composed of only discrete logic gates
- K map is generally used for expressions that has maximum of 5 variables
- A number of variables increases, the table becomes larger and the process becomes more difficult and time consuming

A B \ C	0	1
00	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$
01	$\bar{A}B\bar{C}$	$\bar{A}BC$
11	$AB\bar{C}$	ABC
10	$A\bar{B}\bar{C}$	$A\bar{B}C$

Table for 3 variable K-map

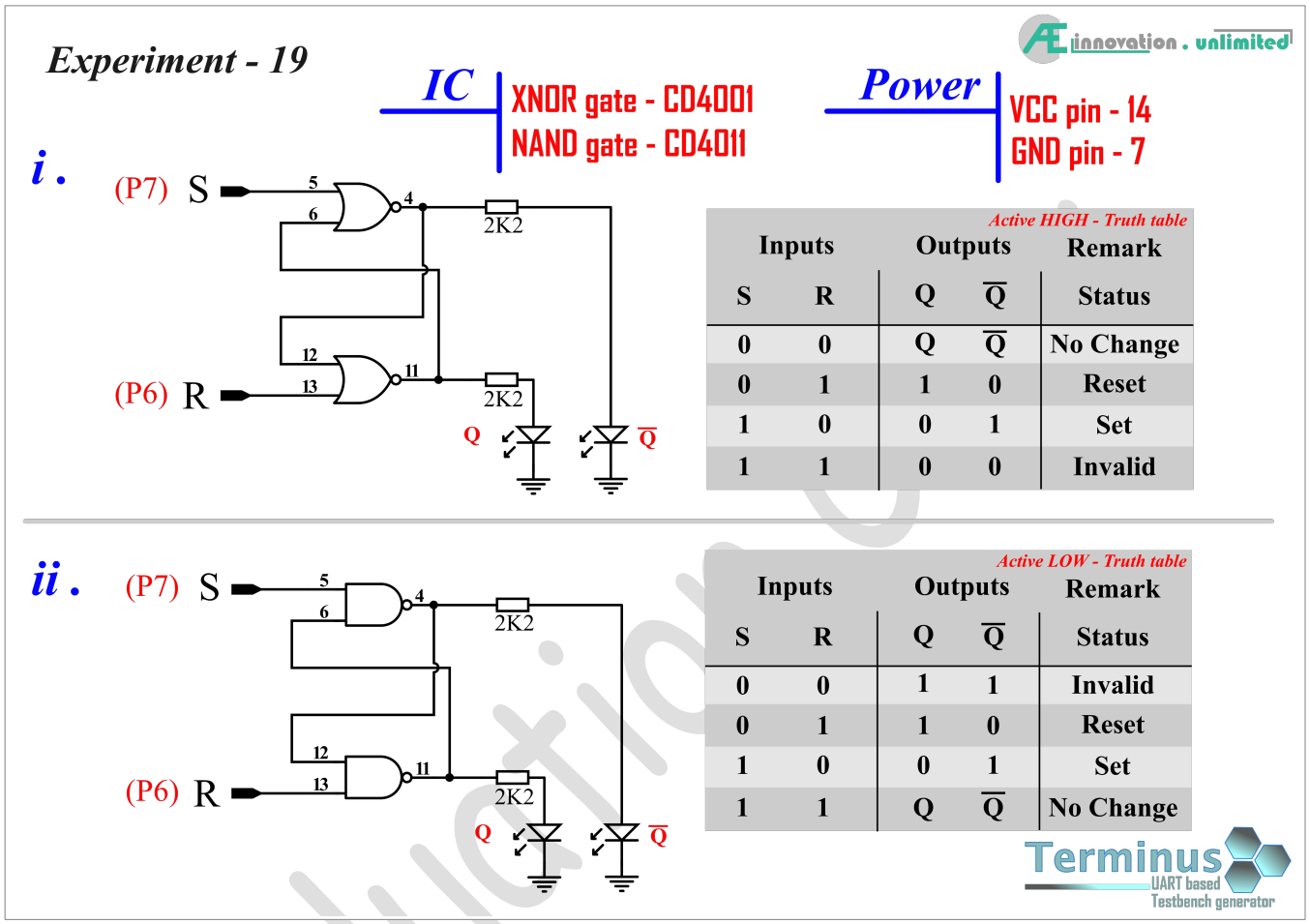
A B \ C D	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

Table for 4 variable K-map

Experiment – 19

Sim file: SR-latch.cir

The basic theory of SR-latch is demonstrated in this experiment. This experiment show all the functionality of a SR-latch along with its drawbacks. The pattern generator generates different stimuli to stimulate the SR inputs with different combination of states.



Components:

a. **One** CD4011 IC (NAND gate)
b. **One** CD4001 IC (NOR gate)
c. **Two** 2.2KΩ Resistor
d. **Two** LED (any color)

Steps:

1) Construct the circuit (***i***) as shown in the schematic
2) Connect the circuit with Terminus and Power ON
3) Start Pattern Generator in the Terminus and Load the program (see below in Automated Test)
4) Run the pattern generator and observe and note the output as different pattern are sweeping the circuit input
5) At the end of the experiment, compare the noted values with the theory and the truth table
6) Repeat the experiment with circuit (***ii***)

Automated Test – Gated Set Reset Latch

Terminus can be used to show the properties of the Gated SR Latch. Using the Pattern Generator function, the test would be fully automated.

Steps:

1. Connect the Gated Latch circuit to the Terminus Parallel output as shown in the schematic
2. In the terminal, write the following command

Note: *Brackets* [] *are used to show the command and the key press event*
[Enter] *means enter/return button on the keyboard needs to be pushed once*
The commands are to be typed in without the brackets (refer to user manual for details)

Command:

>> [pat] [Enter] – *Start the Pattern Generator*
>> [imp] [Enter] – *Set oscillator frequency to 1HZ*
Import pattern, Pattern name: **Gated_SR_latch_test.txt**
>> [sho] [Enter] – *Show the imported pattern*
>> [trg] [Enter] – *Set trigger to external (default is Auto)*
>> [trd] [Enter] – *Set trigger delay*
>> [1500] [Enter] –*Set trigger delay to 1500ms (1.5 seconds)*
>> [;] [Enter] – *Run the pattern generator*
>> [;] [Enter] – *Run the pattern generator, next pattern*

Continue Pattern step until Terminal says Completed

The experiment is also conducted by passing different combinations of states into the inputs of the Gated SR latch. You will notice that Gated SR Latch has all the disadvantages of a SR Latch. Gated SR Latch also may go to the invalid state when both S & R inputs are identical. Only advantage of a Gated SR latch is that it can be used to disable the latch temporarily when needed by setting the Gate input to LOW (0). The gate input can enable or disable the latch using the same AND gate based enable/disable circuit, explained earlier in the mux and demux experiments.

Automated Test – Frequency Divider

Terminus is used to generate the signals used for this experiment. The oscillator function is used to provide a controlled source of periodic pulse of TTL standard at a fixed duty cycle of 50%.

Steps:

1. Connect the Frequency Divider circuit to the Terminus Parallel output
2. In the terminal, write the following command

Note: *Brackets* [] *are used to show the command and the key press event*
[Enter] *means enter/return button on the keyboard needs to be pushed once*
The commands are to be typed in without the brackets (refer to user manual for details)

Command:

>> [osc] [Enter] – *Start the oscillator*
>> [1] [Enter] – *Set oscillator frequency to 1HZ*
>> [dty] [Enter] – *Set oscillator duty cycle*
>> [50] [Enter] – *Set oscillator duty cycle to 50%*
>> [;] [Enter] – *Run the oscillator*
Observe the Oscillator LED and the Frequency Divider output LED
>> [2] [Enter] – *Set oscillator frequency to 2HZ*
Change the frequency and compare the LED s
>> [5] [Enter] – *Set oscillator frequency to 5HZ*
>> [10] [Enter] – *Set oscillator frequency to 10HZ*

If the circuit are assembled correctly and everything goes as it is supposed to, you will see the frequency divided by 2 in the circuit ***(i)***. The frequency division can be observed using only your eyes.

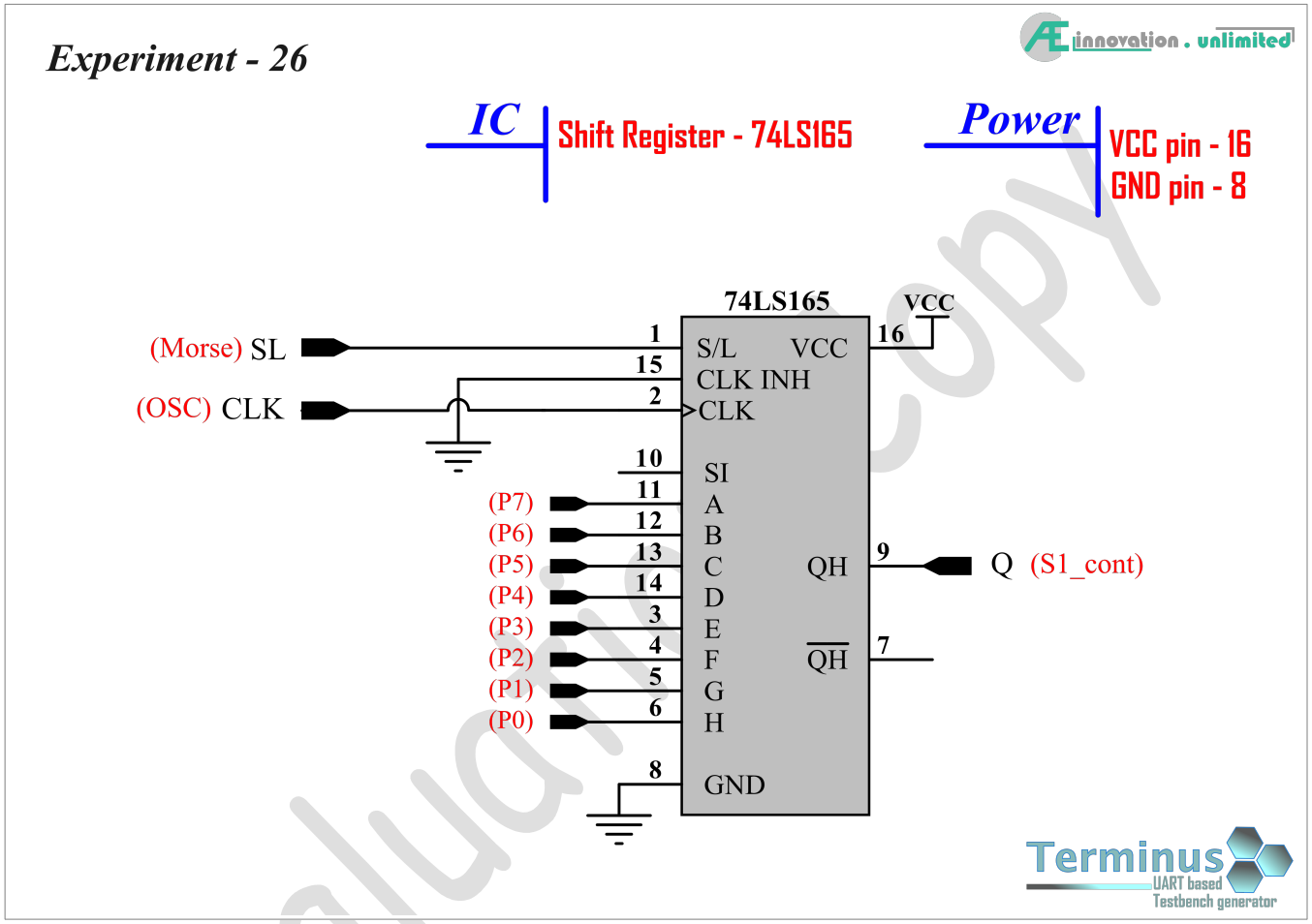
For both circuit, the frequency of the oscillator is first set to 1 HZ. In circuit (***ii.***), the frequency is divided by 2. That means, for every two blink of the oscillator LED, the circuit LED blink once showing that the circuit output signal has half the frequency of the oscillator. For circuit (***ii.***), the division is four times. The circuit output blink once for four blinks of the oscillator LED.

For higher frequencies, the LED of the circuit output will not blink as frequently as the oscillator LED. This shown the signal frequency coming out of the circuit is lower than the oscillator frequency.

You can repeat the experiment adding more stages. Frequency division is an important part of designing frequency counters. In the early days of electronics, frequency counter was used to make digital clocks.

Experiment – 26

The IC used in this experiment is a general purpose Parallel In Serial Out (PIPO) shift register, and is specifically designed to act as an input port. This experiment will make use of Terminus completely. Terminus has built-in functionality that allows us to demonstrate the working principle of the IC easily. The data input signals are applied on the IC inputs. Terminus will shift in the input data and show then on the terminal.



Components:

a. **One** 74LS165 IC (PISO)

Steps:

1) Construct the circuit as shown in the schematic
2) Connect the circuit with Terminus and Power ON (see below in Automated Test)

Automated Test – PISO 74LS165

Terminus can be used to show the properties of the Parallel In Serial Out shift register. The Byte Generator function can write, read and generate necessary signals for this experiment.

Steps:

1. Connect the 74LS165 circuit to the Terminus Parallel output as shown in the schematic
2. In the terminal, write the following command

Note: *Brackets* [] *are used to show the command and the key press event*
[Enter] *means enter/return button on the keyboard needs to be pushed once*
The commands are to be typed in without the brackets (refer to user manual for details)

Command:

>> [bin] [Enter] – *Start the Byte Generator*
>> [11001100;] [Enter] – *Set the 8 bit data*
>> [sci] [Enter] – *Serial input the data from the 74165*
Change the pattern and repeat
>> [10001101;] [Enter] – *Set the 8 bit data*
>> [sci] [Enter] – *Serial input the data from the 74165*
Change the pattern and repeat
>> [10001101;] [Enter] – *Set the 8 bit data*
>> [sci] [Enter] – *Serial input the data from the 74595*

The Byte Generator is also used in this experiment. This IC is literally the opposite of the device used in the last experiment. This IC takes in input from its parallel pins and propagates them serially.

The Byte Generator is used to place a user defined 8-bit data on the Terminus parallel port. The parallel port of Terminus is used to write an 8 bit data to the shift register parallel inputs. The **sci** command causes the Terminus to generate 8 clock pulses. With every clock pulse, the clock pulse shift out the 8 bit parallel data as a stream of 8 bit serial data. The **S1_cont** input pin of the Terminus captures the serial data and store it in its own memory. As the shift register is 8 bit, only 8 clock pulses are generated by the Terminus. Each clock pulse shifts out one bit of data into the Terminus serial input port (**S1_cont**). At the end of 8 clock pulses, the entire 8-bit data is stored in to the Terminus memory and is shown on the terminal display.