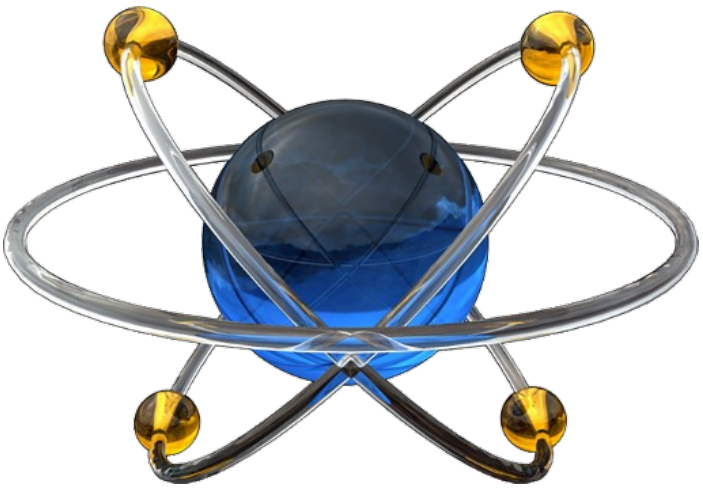
INSTITUTE of TECHNOLOGY CARLOW

Institiúid Teicneolaíochta Cheatharlach



# PROTEUS

## Computer Aided Design

**Dr Diarmuid O'Briain**

Version 2.0

**Adapted and updated from original works by:**

Keith Smyth
Frank Fennelly

# Table of Contents

# Table of Figures

*This page is intentionally blank*

# 1 Getting started with Proteus design Suite 8.2

The objectives of this exercise are to:

1. Introduce Proteus design Suite 8.2.
2. Use the New Project Wizard to create a new schematic.
3. Introduce the Proteus ISIS user interface, menu structures etc.
4. Guide through the process of editing an existing schematic in order to familiarise with the basic editing features of Proteus ISIS.

---

**1) Starting Proteus ISIS.** To start the Intelligent Schematic Input System (ISIS), click on the *Start* button and select *Proteus 8 Professional*, and select *File > New Project*. The Project Wizard will now guide through the process. Select a suitable name for the project and a path to where the project files will be stored, select *next*. To start, create a schematic diagram by checking the *Create a schematic from the selected template* option, select *Landscape A4*, select *next*. At this stage do not produce a PCB layout so select *next* again *(Do not create a PCB layout)*. An embedded controller is not required for this project so select *next* again *(no firmware project)*. The summary window confirms that a schematic diagram is all that is being produced and it displays the path to the saved files. Select *Finish* and Proteus ISIS schematic capture opens.

**2) User interface & screen layout**: The Proteus ISIS screen contains the following areas - the *Editing Window, the Overview Window, the object selector, the toolbars and the pull down menus* as shown in Figure 1:

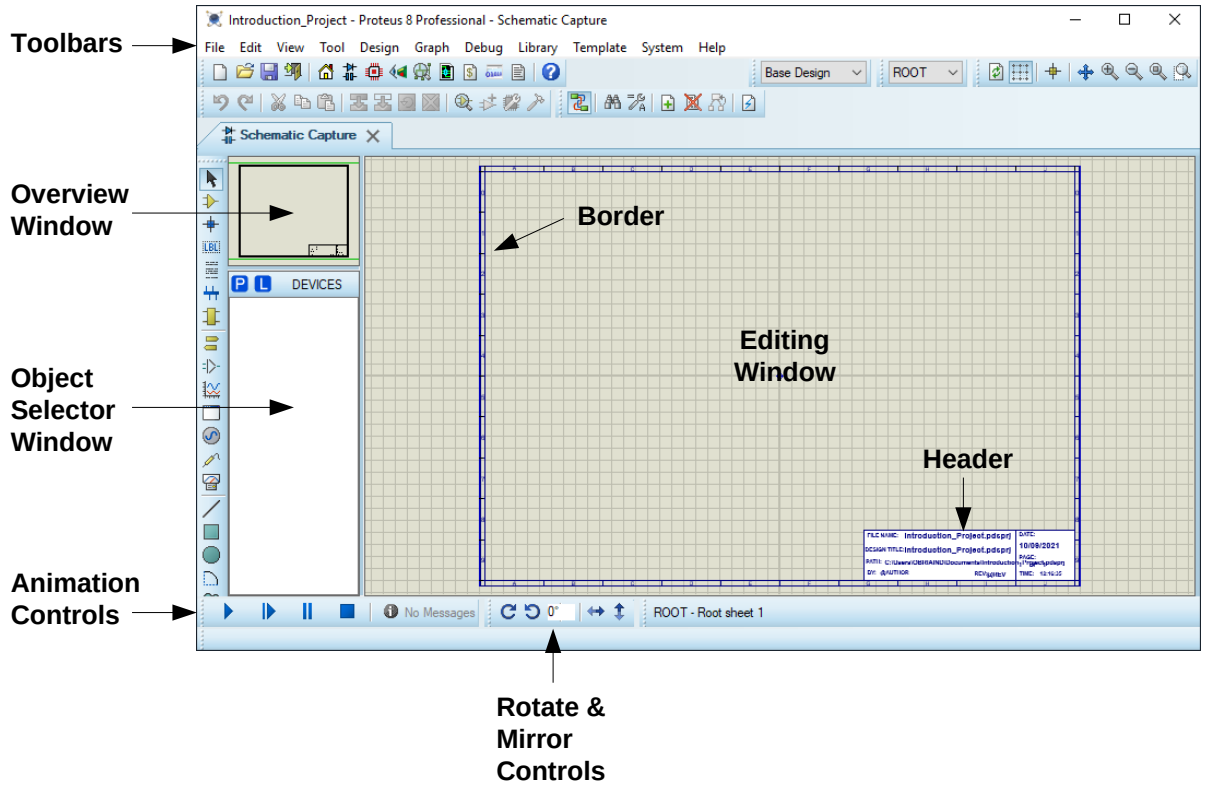



*Figure 1: User interface and Screen Layout*

The largest area of the screen is called the *Editing Window*, and it acts as a window through which the drawing can be viewed and modified - this is where component symbols are placed and connected (wired-up). The smaller area at the top left-hand side of the screen is called the *Overview Window*. In normal use the *Overview Window*, as its name suggests, displays an overview of the entire drawing - the blue box indicates the outer edges of the entire sheet and the green box the area of the sheet currently displayed in the *Editing Window*. However, the *Overview Window* also has another function: when a new object, such as a component symbol, is selected from the *Object Selector* the *Overview Window* is used to preview the symbol for that object - this feature is useful for setting the orientation of the component before it is placed.

**3) Pull down menus:** The Menu Bar is located across the top of the screen

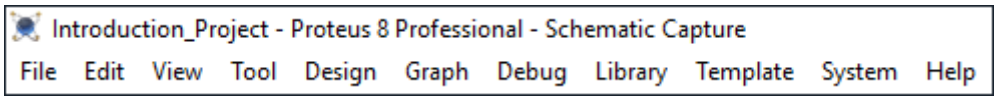



*Figure 2: Pull-down Menu*

Pull down menus are activated by pointing at the menu name and clicking the left mouse button. The required option is selected by clicking the left mouse button on the menu item required. This is the same process used in all Microsoft Windows programs. The only difference here is that the menus and options are different.

Some options have associated shortcut keys which can be used instead of the menus - these are displayed at the right of the menu option.

Two of the most useful shortcuts which should be remembered are:

| | |
|---|---|
| **Ctrl Z**: | Used to undo the last action completed.<br>ISIS allows multiple uses of the undo feature. |
| **R**: | Used to redraw the screen. |

Options which enable or disable some program feature show that feature's status with a small tick symbol to the left of the option. Presence of the tick symbol implies that the option is selected.

**4) The Toolbars:** The Toolbars consists a number of Buttons or Icons which are organised in rows across the screen. They are located just below the pull down menus. If pressed they act as a convenient way of using frequently used features. The icons are divided into groups according to function. The icons located along the top of the screen (by default) provide alternative access to the menu commands, as follows:

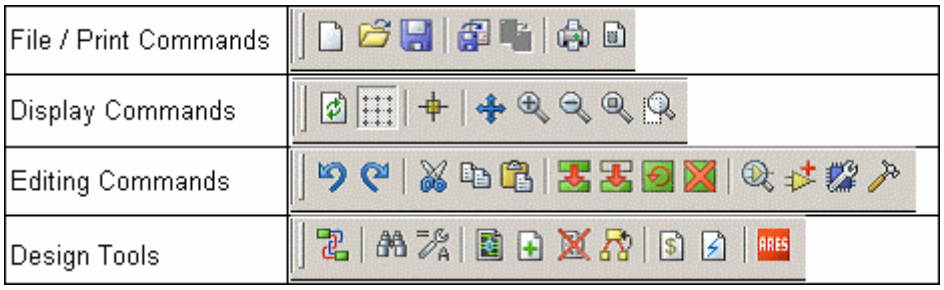



*Figure 3: Toolbars*

The icons in the mode select toolbar determine what mode the editor is in i.e. what is going to happen in the edit window. These functions are not duplicated on the menus.


*Figure 4: Mode select toolbars*

**5) Getting help:** Select *Help* from the drop down menu or press the function key *F1*.

**6) Panning & Zooming:**
The area of the drawing displayed in the Editing Window can be adjusted in a number of ways.

By **Zooming** the scale of the drawing displayed ca be adjusted in the *Editing Window*. This can be achieved in one of three ways. 1) By use of the function keys *F6* & *F7* to zoom in and out 2) selection of the corresponding commands in the *View Menu* or 3) by rolling the roller on a the mouse. *F8* is the Zoom all function – pressing this button causes all of the schematic to be displayed in the editing window.

**Panning** is the process of moving to different areas of the schematic. To simply move the displayed area so that a particular component is at the centre the *Editing Window*, position the mouse pointer over the desired component or area of the *Editing Window* and press the *F5* key.
Should it be required to move the *Editing Window* to a completely different part of the drawing, the quickest method to achieve this is to simply point at the centre of the new area on the *Overview Window* and click the left mouse button.
It is also possible to pan by holding the shift key down and cursor to the direction of the required view.
At the bottom right of the screen is the co-ordinate display, which displays the co-ordinates of the mouse pointer when appropriate. These co-ordinates are in 1 thou units (0.001") (25µm) and the origin is in the centre of the drawing which is indicated by a blue circle with a cross through it.

**7) Tagging and moving an object**: Tagging is the process where by a component that may need to be moved, have its properties changed, deleted etc. is identified.
To tag an object move the cursor over the symbol and right or left click. A component that has been tagged then turns red. A drop down menu appears offering a list of editing options. To move a tagged object, move the cursor over the object and press the left mouse button, continue to hold it down and drag the object into the position required.

**8) To delete unwanted components:** It is easy to add too many copies of a component to a page. If it is required to delete components press Ctrl 'Z' if the component has just been added. Otherwise to remove a component, point at the component and right click it twice. The first right click tags the object. Clicking right on a tagged object deletes it.

*Figure 5: Rotation and Mirror tools*

**9) Rotation & Mirror Images**: To rotate an object, tag it, and then click on the left and or right Rotation icon which appear in the pull down menu as illustrated in Figure 5. The illustration also shows the mirror controls that can be used in a similar way to reflect the last object tagged.

## *1.1 Exercise #1: Introduction to Proteus*

This exercise will foster familiarity with and to use the features described in this section:

1 Load the project '*sample*' from the CAD folder on the student common drive

2 Use the *F6* & *F7* keys to zoom in & out. Use the *F8* key. Then zoom in until the images are large enough that the message can be read which is written in very small text at the centre of the editing area.

What is written there?_____ [This is the midpoint]

3 Pan to all four corners of the schematic. What is the number of the components in the

top left corner:_____ [1] the bottom left corner_____ [0]

top right corner_____ [1]

4 Tag *R1* – experiment with moving it around.

5 Remove the components *R1*, *C8* and *C9*.

6 Replace *D1* with a short circuit. [Component mode, click on point A then B]

7 Move and rotate *J1* & *J2* as required to get the bottom left corner of the schematic to look like Figure 6.

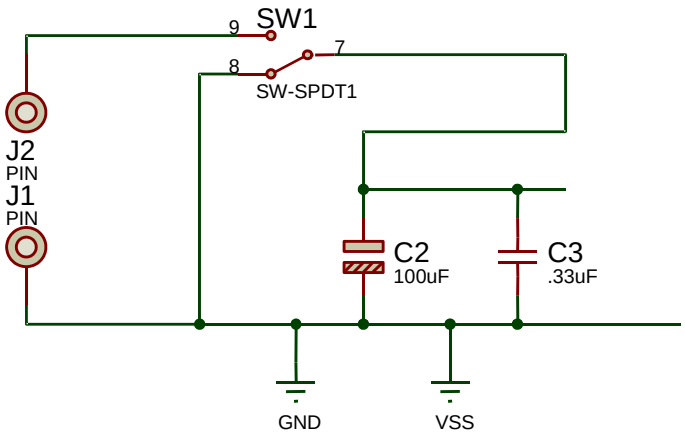



*Figure 6: Exercise 1 sub-section of the 'sample'*

8 Create a folder called CAD on the local home drive. Save this modified file with an appropriate name to this CAD folder.

9 Importing a legacy Proteus ISIS schematic. Select *Import Legacy*, browse to the file to be imported, if necessary browse to the ARES file/s. Select a new file name if applicable and browse to a path where this new project is to be stored.

Import the schematic *Dice* from the CAD folder on the common drive. Save this file on the home drive. Change the value of *R1* and *R2* to *4k* and *6k* respectively. Save the changes.

**Student notes:**

# 2 First PROTEUS 8 Schematic

## *2.1 Exercise #2.1: First schematic*

The objectives of this exercise are to:

1. Create a simple schematic of a circuit using Proteus.
2. Place a border and a title block on an imported legacy project.
3. Simulate the circuit. In particular use virtual instruments to measure the voltage and current at various points in the circuit.
4. State the relationships between voltages and currents in series and parallel resistive circuits.

---

**1) Getting Started:** As described in the introduction open a *New Project*. The Wizard will present an option to select an appropriate file name and path to save the file.

**2) Selecting the components which will be used in the design:**
ISIS refers to components as devices. The first schematic to draw is a simple circuit which consists of two resistors and a battery connected in series. This schematic to be draw is illustrated in Figure 7.

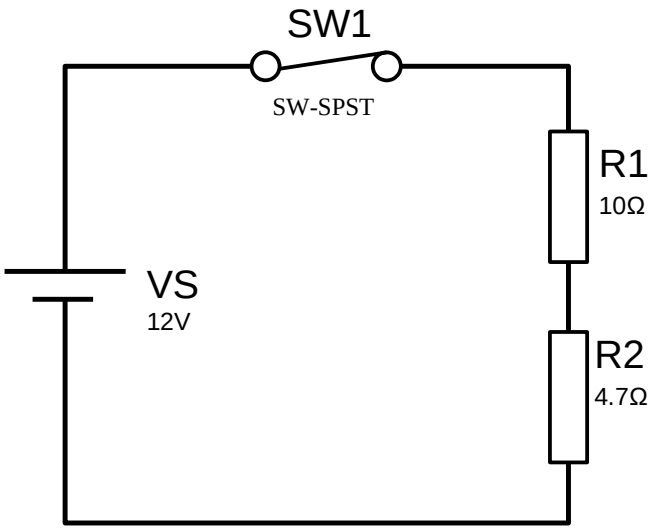



*Figure 7: Series circuit*

First select the components required from the library of components available. Below the Toolbar is the Component Selector which is used to select devices, symbols and other library objects. To add components to the design start by pressing the component button in the "*Main Mode*" toolbar.

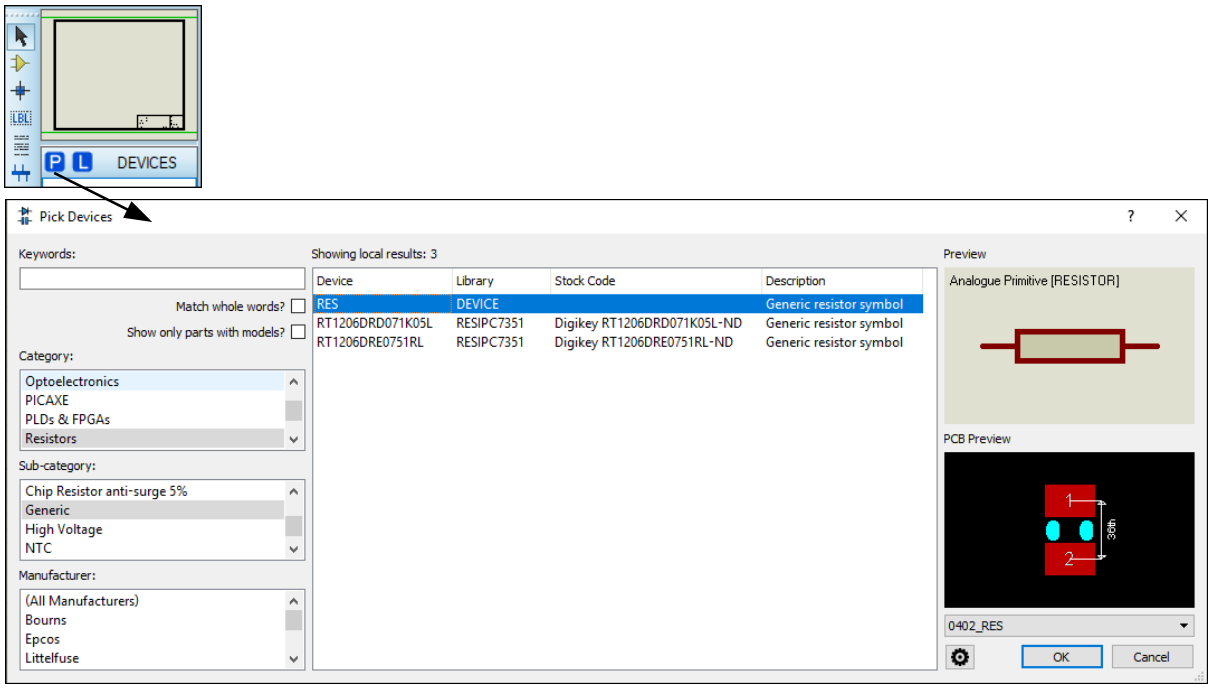



*Figure 8: Select a component*

To select a device, point at the *P* button at the top left of the *Component Selector* Window and clicking the left mouse button as illustrated in Figure 8. This causes the *Pick Devices* dialogue box to appear which allows for the selection devices from the various device libraries.

The category of device stored in a library is usually reflected in the library name e.g. Diodes, Capacitors etc. For example if a resistor is required, obtain by selecting the *resistor* category and then the *generic* sub-category. Then, double-click on the Generic resistor listed – this will add the resistor symbol to the list of devices available for use.

Sometimes it may not be obvious where a device will be found. In such cases enter the name of the device being searched for in the *Keyword* box. If there is a device in the library with a name that matches the keyword it will be displayed. To close the *Pick Devices* dialogue box, select *OK*.

When the dialogue box is closed the *RES* device appears in the *Devices Selector* window. The device last picked is highlighted ready for placement.

### 3) Placing components on the blank page:

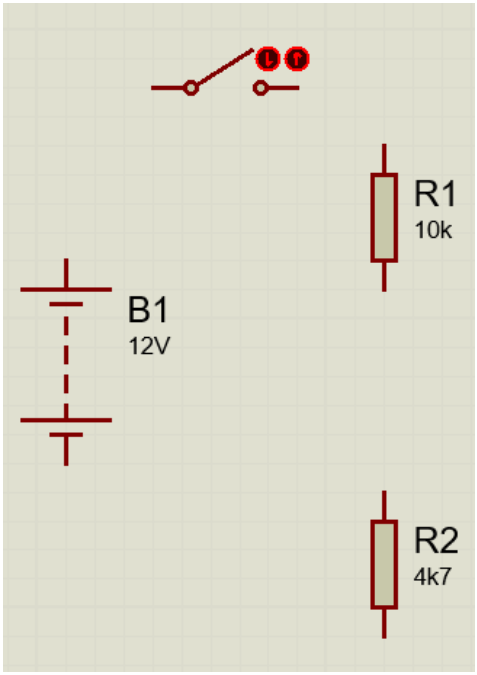



*Figure 9: Placing components on blank page*

Components to be added are selected by clicking on it in the list shown in the device selector.

They are then placed in the editing window. Prior to placing a device in the *Edit Window* the orientation the device will have when it is placed is shown in the *Overview Window*.

It may be necessary to change the orientation. This can be achieved using the rotate and mirror icons. As the Rotation and/or Mirror icons are used, the device is redrawn in the *Overview Window* to preview the new orientation.

For this design select the generic resistor device and then move the mouse pointer into the middle of the *Editing Window*. Click the left mouse button. An outline of the resistor will appear which can be moved around by dragging the mouse. When the mouse button is clicked again, the component will be placed on the schematic. Two copies or instances of the RES device are required. Repeat this process for the other resistor. Then select the BATTERY device.

**Note**. Do not worry about interconnecting the components for now.

***Virtual instruments:*** Proteus ISIS provides a large number of *Virtual instruments*. These are symbols which can be placed in a schematic and simulate the behaviour of a real instrument such as a Voltmeter or an Oscilloscope. For this schematic select the DC voltmeter. Each click on the screen creates a new instance of the voltmeter. Repeat the process for the DC ammeter.

**4) The Edit Window & the Grid:** At the bottom right of the screen is the co-ordinate display, which reads out the co-ordinates of the mouse pointer when appropriate. These co-ordinates are in 1 thou units and the origin is in the centre of the drawing. 1 thou is 1/1000 of an inch (25µm).

A grid is normally displayed in the *Editing Window*. The grid helps in lining up components and wires. It can be toggled on and off using the Grid command on the *View Menu*. It can be toggled on & off by using the shortcut key *G*. The line spacing usually reflects the current snap setting. When the pointer is moved within the *Editing Window*, the increments of the co-ordinate display are in fixed steps - initially 100th. This is called snapping and enables the positioning of components and other objects on a neat grid. The resolution of the snap may be selected with the Snap commands on the *View Menu*.

**5) Rotating & Repositioning components**: It is unlikely when adding components to have gotten the components oriented and positioned satisfactorily at this first attempt, so it may be necessary to move things around.

***Tagging items*:** In Proteus ISIS, objects are selected for further editing by 'tagging' them. Try pointing at the Battery and clicking right. This tags the object, causing it to be highlighted. Now, still keeping the pointer over it, hold the left button down and drag it around. When the component is in the desired position release the left mouse button. Objects may be rotated or a mirror image obtained using the buttons in the bottom left corner of the screen as seen in the previous exercise.

**6) Inter connecting components:**
The next step is to interconnecting the components. Start by pointing at the tip of the upper end of *R1* and clicking left. Proteus ISIS senses that the pointer is aimed at a component pin and deduces that a wire should be connected from it. To signify this, it displays a green line which goes from the pin to the pointer. Now point at the tip of negative terminal of the *DC ammeter* and left click again.
ISIS takes this as the other end for the wire and invokes the Wire Auto Router (WAR) to choose a route for the wire. Now do the same thing for each end of *R2*, following the diagram. Try tagging objects and moving them around whilst observing how the WAR re-routes the wires accordingly.

If the route that the Wire Auto Router has chosen is not what is wanted, delete it and manually route by left click on the first pin, clicking left at each point along the required route where corners are required, and then finish by clicking left on the second pin.

**7) Annotating the components:** Annotation is the process of giving a name and a value to the components. Proteus ISIS automatically gives the components a name when they are initially placed in the edit window. It may be desirable to change the default name or default value given to the component. Proteus ISIS offers a number of possible ways to annotate the components in a design. Initially employ the *Manual Annotation*. Any object can be annotated either by tagging the device (right clicking the mouse) and then selecting edit properties. A dialogue box as shown below then appears. Enter the relevant properties such as Reference number, Value etc.

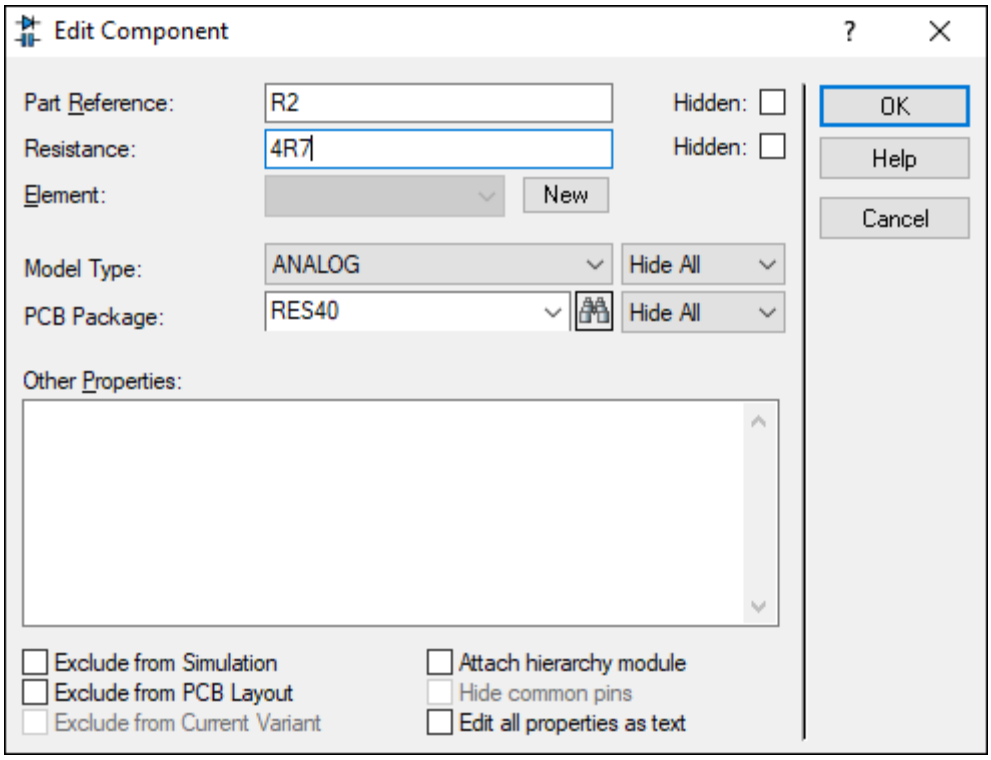



*Figure 10: Change the properties of a component*

To change the value of *R2* to *4R7* as illustrated in Figure 10. In this case the *R* represents a decimal point. This technique is used to avoid confusion occurring when a drawing is copied several times and the decimal point may not be seen. Change the value of *R1* to *10R* and Change the Battery voltage to 12V.

**8) The Border and Title block:** If a legacy project is imported from an older version of Proteus the sheet/s may or may not have a border and header. If a sheet is added to this project it will NOT have a border and header.

Select the box buttons in the 2D Graphics toolbar. Then select the Marker style from the options in the graphic style object selector box. The blue line which is normally around the drawing does not print and acts only as a guide to locate the page edge.

Place the tip of the cursor on the top left corner of the blue box. Press the left mouse button and drag across the screen to draw a black box which overlaps the blue box.

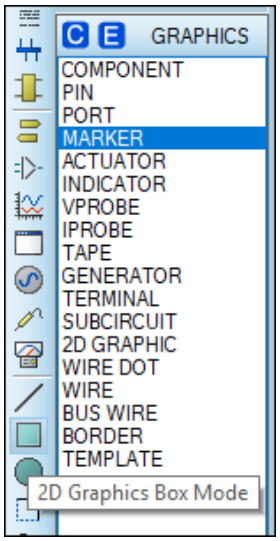



*Figure 11: 2D Graphics Box Mode - Marker*

***Title Block***: To add the *Title block*, select 2D graphics and the symbol mode in the toolbox. Use the mouse to press the *P* (pick) button. From the pick symbols dialogue box, select the *Header* object by double clicking on it. Now place the header in the same way as the components were placed.

**Note**: The text in the header file cannot be edited directly. It is edited using the *Edit design properties* menu options which are available under the *Design* pull down menu.

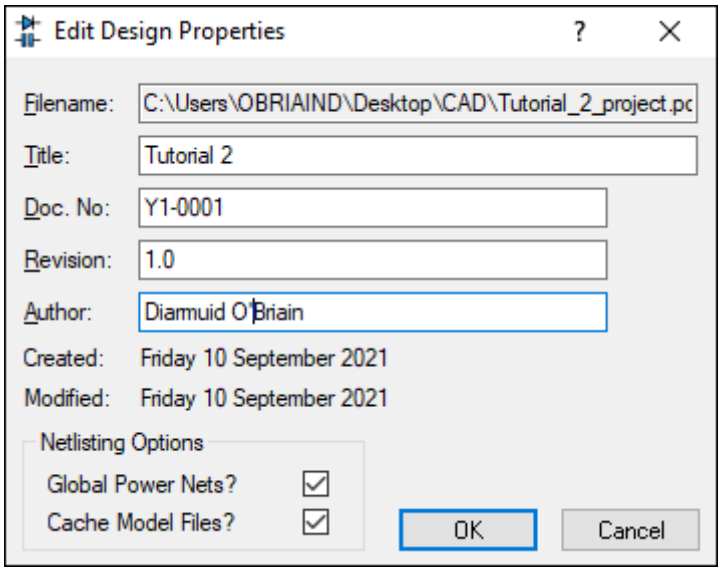



*Figure 12: Edit Design Properties*

Press *R* to Redraw the schematic diagram, the Header will adjust as illustrated in Figure 13.


*Figure 13: Header after R for Redraw is pressed*

**8) Simulation:** To simulate the circuit press the play button

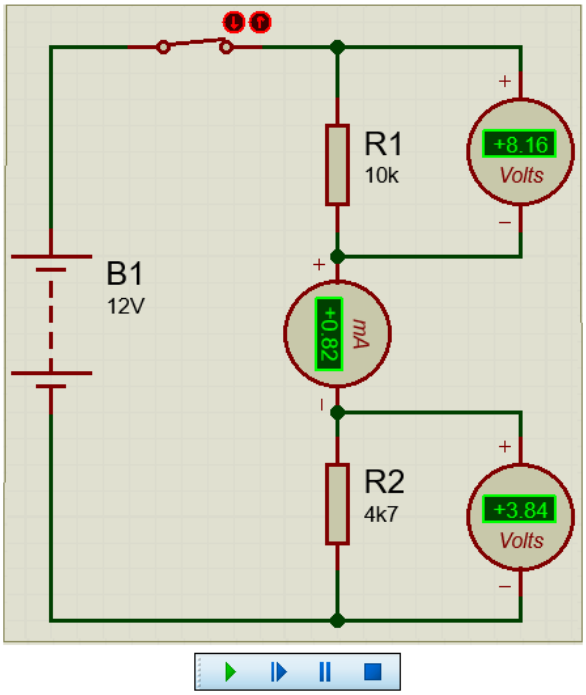



*Figure 14: Running a simulation*

If all connections are made correctly the currents and voltages in the system should be displayed by the virtual instruments as illustrated in Figure 14.

**9) Avoiding Problems:** Two of the most common problems encountered when starting Proteus ISIS are:

- Graphic lines do not interconnect components and do not feed any interconnection data to ARES, the Printed Circuit Board (PCB) layout program.
- Wires are not properly terminated at the component terminals. If a component is tagged and dragged, all wires should follow the component (rubber banding).

**Note**: Also, as with all programs, it is important to ***save work frequently***.

## *2.2 Exercise #2.2: Build a schematic diagram*

With reference to the schematic diagram in Figure 7.

A1) Calculate all the voltage and current measurements expected.
A2) Record the actual voltages and current indicated by the virtual instruments – do they agree?
A3) Change the battery voltage to 10 Volts – and again calculate and record the voltages and currents
A4) What electrical principles can be deduced from the readings?

B1) Stop the simulator. Change both R1 and R2 to 4.7 KΩ (4K7)
B2) Calculate the voltages and current measurements expected.
B3) Simulate the circuit. What voltages and current are indicated by the virtual instruments? Hint a change to the range of the ammeters may be required.
B4) What electrical principle can be deduced from the readings?

C1) Import the 'Legacy project' file from the CAD folder on the common drive.
C2) Add a new Root sheet, on this new sheet add a border and a header.
C3) Place a suitable Title, Revision and Author in the header.
C4) On this new sheet create the diagram shown in Figure (c) below. Calculate the voltage across each resistor and the current through each resistor. Simulate the circuit – use virtual instruments to measure each current and voltage

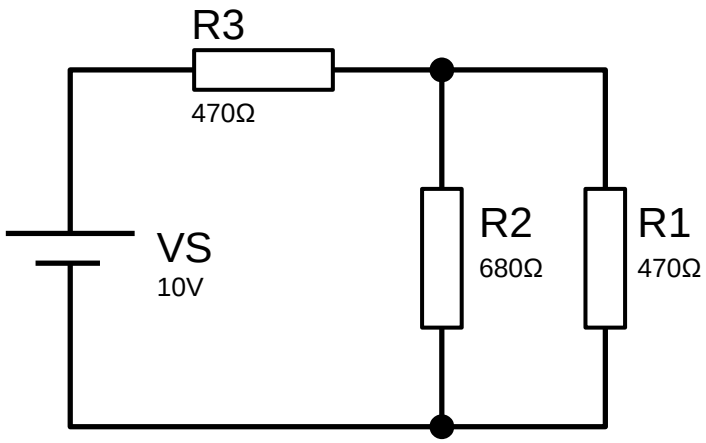



*Figure 15: Series-Parallel circuit*

With reference to the schematic diagram in Figure 15 (Series Parallel Circuit).

D1) Calculate all the voltage and current measurements expected.
D2) Record the actual voltages and current indicated by the virtual instruments – do they agree?
D3) Change the battery voltage to 10 Volts – and again calculate and record the voltages and currents
D4) What electrical principles can be deduced from the readings?

# 3 LED & Switch animation

## *3.1 Exercise #3.1: Working with LEDs*

**a.** Create a simple schematic which uses the following components.
- **i.** Switch - SPST
- **ii.** Coloured Light emitting diodes
- **iii.** DC Ammeter.
- **iii.** DC Voltmeter.

**b.** Simulate the circuit. In particular use virtual instruments to measure the voltage and current at various points in the circuit.
**d.** Determine the limiting resistor value.
**e.** Calculate the current through a diode.

---

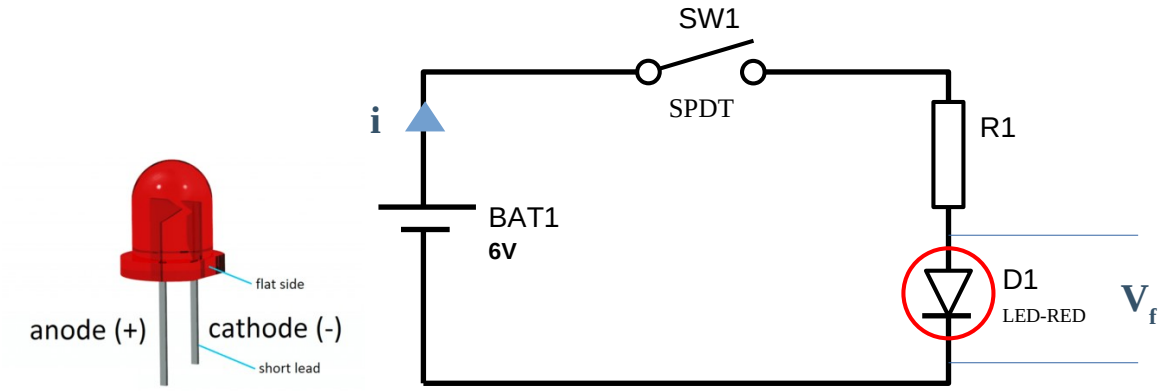



*Figure 16: LED limiting resistor*

Typical Diodes have the following specifications:

**LED Specification**
Forward voltage drop: **1.8-2.2VDC**
Breakdown voltage drop: **4VDC**
Full drive current: **10mA**
Suggested forwarding current: **16-18mA**
Max current: **20mA**
Luminous Intensity: **150-200 mcd**

It is essential therefore that the voltage across the diode never exceeds the breakdown voltage drop or the current through the diode exceed the maximum current. To prevent this a limiting resistor is placed in series with the diode. Consider Figure 16, the diode model is “Analogue”, it is specified as having a Forwarding Voltage of 2.2V and a breakdown voltage of 4V.

As this is a series circuit and if 2.2V is dropped across the diode through which it is expected that 10mA will be required to drive it, the resistance of the diode in operation can be calculated as:

$$R_{D1} = \frac{2.2}{0.01} = 220\Omega$$

The current passing through R1 is also 10 mA when the diode is in operation and the voltage drop is 6 – 2.2 = 3.8V, therefore its resistance value must be:

$$R1 = \frac{3.8}{0.01} = 380\,\Omega$$

Alternatively the value of the limiting resistor could be calculated directly as:

$$R = \frac{V_S - V_f}{i} = \frac{6 - 2.2}{0.01} = 380\,\Omega$$

where: ***Vs*** = supply voltage, ***Vf*** = LED forward voltage drop, and ***i*** = LED forward current

**1) Getting Started:** Build the circuit in Figure 16. Add a DC Voltmeter across the Resistor, R1 and the LED, D1. Place a DC Ammeter (set to mA) in circuit between the Resistor R1 and Diode, D1 and confirm the values calculated.

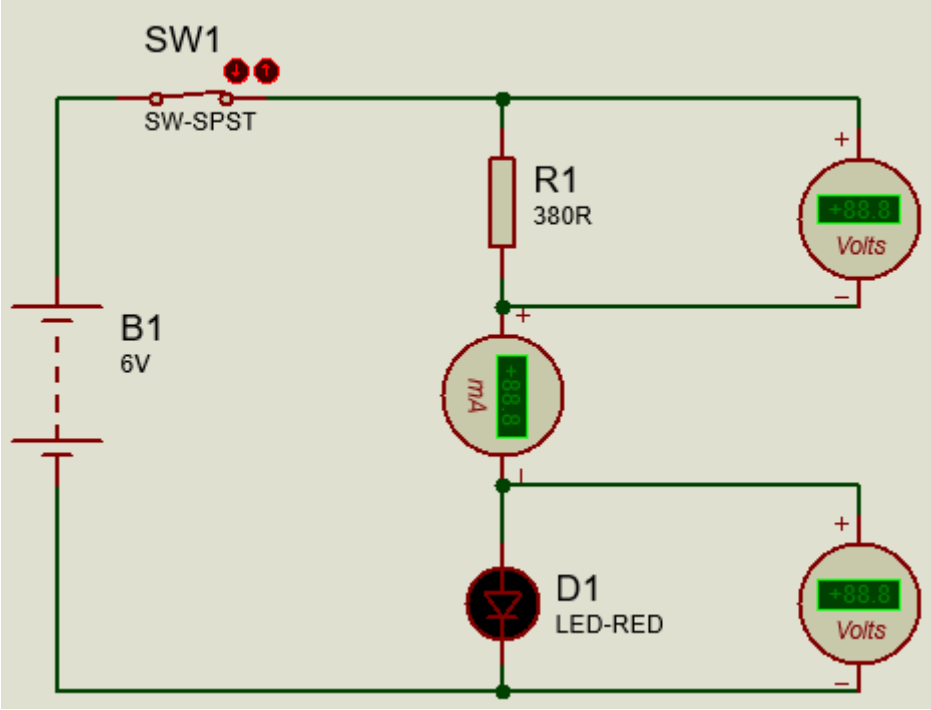



*Figure 17: LED and limiting resistor calculations*

**2) Confirm values:** Write down the values and confirm:

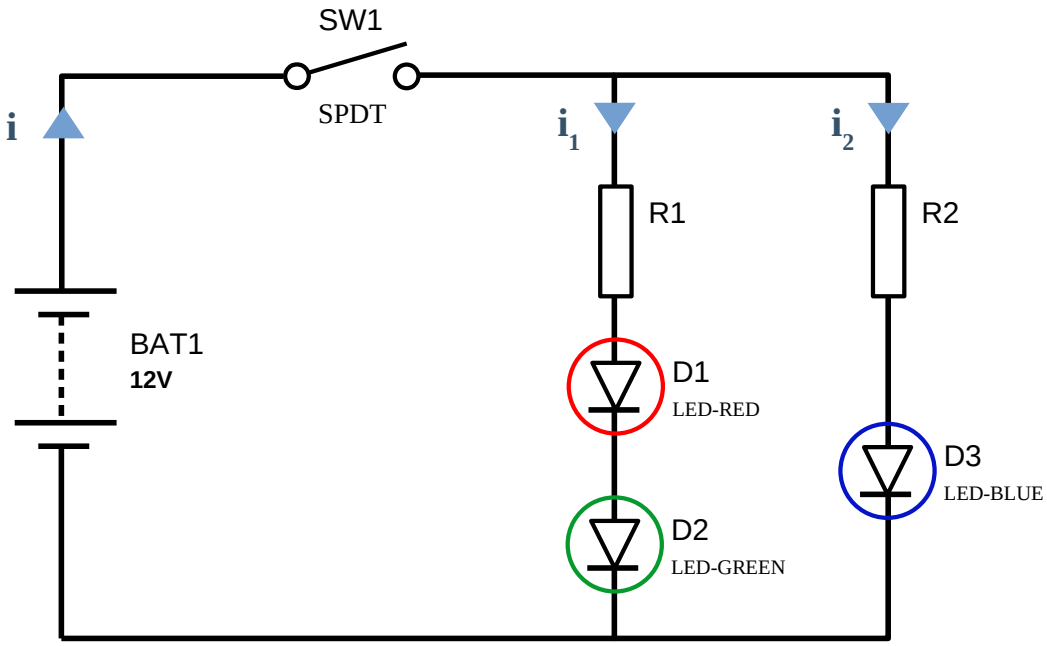



*Figure 18: Series parallel circuit*

Consider the circuit in Figure 18, calculated the values of resistors, R1 and R2 assuming the three LEDs, D1, D2 and D3 have the following specifications:

**LED Specification**
Mode: **Analogue**
Forward voltage drop: **1.8-2.2VDC**
Breakdown voltage drop: **4VDC**
Full drive current: **10mA**
Suggested forwarding current: **16-18mA**
Max current: **20mA**
Luminous Intensity: **150-200 mcd**

**D1** – Red, **D2** – Green and **D3** – Blue

**3) Build and test:** Build the circuit but add DC Ammeters (mA) to monitor the current in each leg, $\mathbf{i_1}$ and $\mathbf{i_2}$. Did the values match the calculations?

## *3.2 Exercise #3.2: LED & Switch animation*

The objectives of this exercise are to:

**a.** Create a simple schematic which uses the following components.
  **i.** Switches - SPST & SPDT
  **ii.** Coloured Light emitting diodes
  **iii.** A pulse generator.
**b.** Use the Block copy command.
**c.** Simulate the circuit. In particular use virtual instruments to measure the voltage and current at various points in the circuit
**d.** Determine whether an LED is forward or reverse biased
**e.** Calculate the current through a diode.

---

**1) Getting Started:** Add a new project as demonstrated in section 1. The Proteus ISIS editor will then load and run.

**2) Selecting the components which will be used in the design:**

**Note**: Refer to step 2 of tutorial 2 regarding selecting devices (components).

Select the following devices for use in the design:

- LEDs Green, Red, and Blue,
- LED (generic),
- Battery,
- Resistor (generic),
- SPST switch,
- SPDT switch

**3) Placing components, Terminals, instruments & generator on the page:**

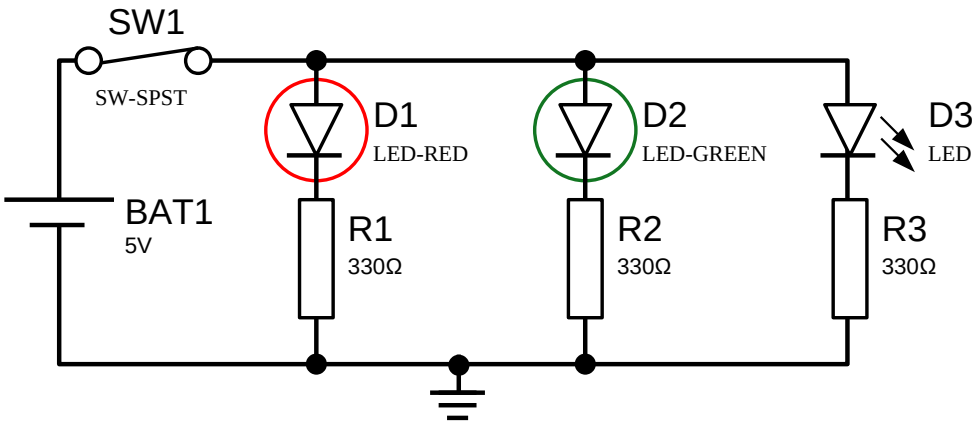



*Figure 19: LEDs and Switches #1*

**Note:** Do not attempt to draw the full circuit.

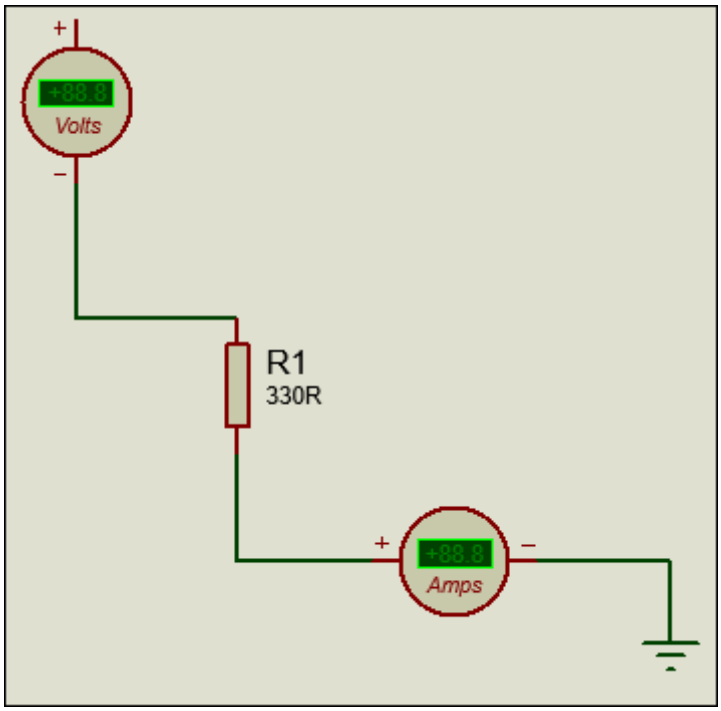



*Figure 20: First part of Circuit #1*

Draw the portion of the diagram in Figure 20.

Use the Terminal mode to access the GROUND.

A lot of time can be saved by identifying sections of the diagram that can be drawn once and then copied many times. In this case select the portion of the circuit shown above.

Select the block copy command icon which will allow for the creation of extra copies of this portion of the circuit. Make three copies. **Note** that all points which have a ground terminal attached are connected together. Using this technique the schematic is less cluttered. Add the light emitting diodes to the top half of the diagram and change the battery voltage to 5V.

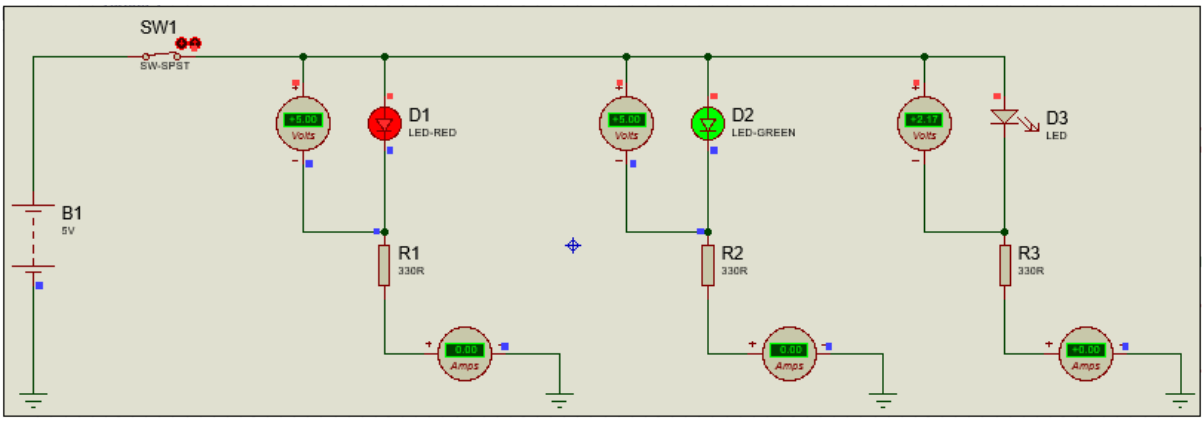



*Figure 21: LEDs and Switches #1 - Complete*

### Circuit 2

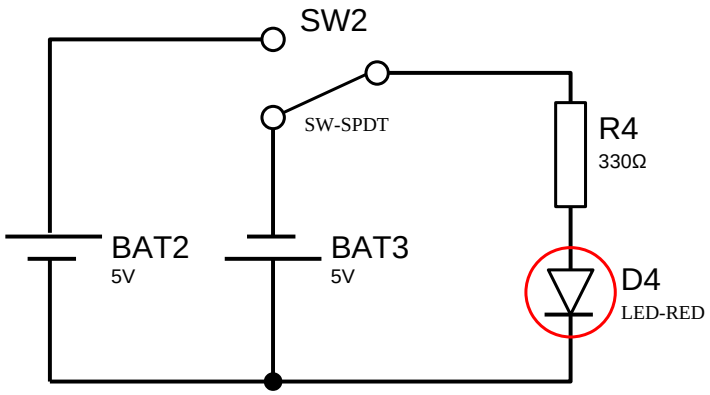



*Figure 22: LEDs and Switches #2*

Build the circuit in Figure 22. Place the Single Pole Double Throw (SPDT) switch and the batteries in the bottom left hand side. Note the polarity of the batteries.

### Circuit 3

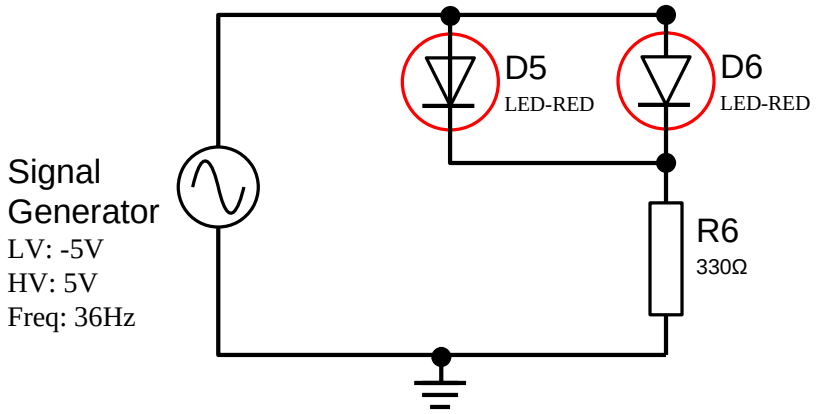



*Figure 23: LEDs and Switches #3*

Finally add the section of the circuit shown in the bottom right hand side of the diagram. This section of the diagram uses a pulse generator for the simulation.

To add the pulse generator select the generator symbol from the mode menu 

Right click on the pulse generator symbol to access the properties as illustrated in Figure 24.

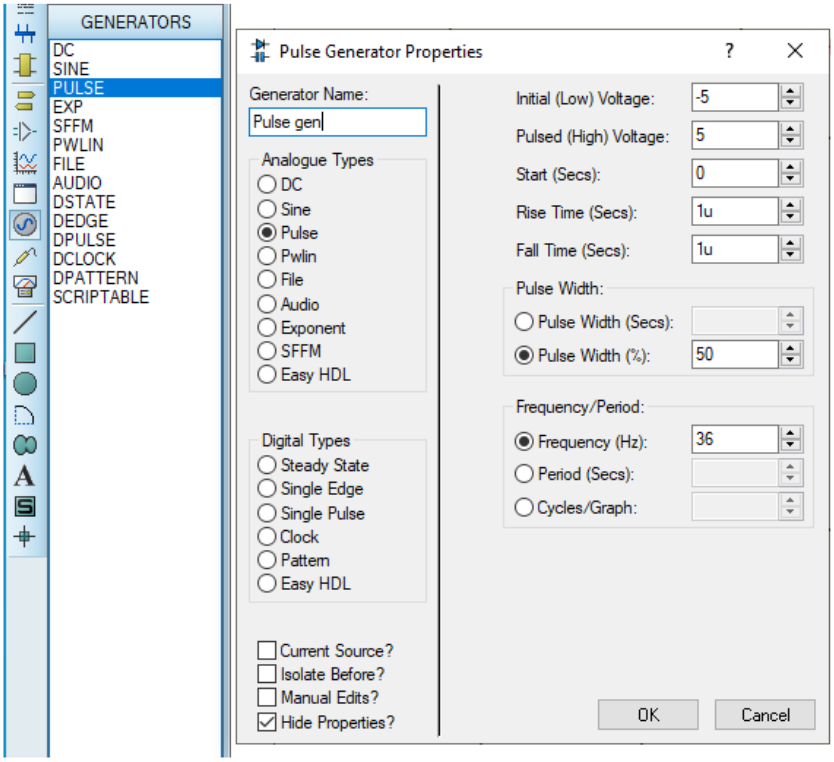



*Figure 24: Pulse Generator properties*

**4) The Header block:**

Place suitable content into the header block. (See tutorial 2)

**Note**: It is important to ***save work frequently***. Save work as **xxxx v.1** on the home drive.

## *3.3 Assignments: Record all results*

**Circuit 1**

1. What voltages and current measurements can be expected using the instruments shown?
2. Simulate the circuit.
3. Record the actual voltages and current indicated by the virtual instruments – do they agree?
4. Replace D3 with a blue LED and modify the circuit so that D1, D2 & D3 can be turned on & off separately. Save this new file as **xxxx v.2**

**Circuit 2**

1. What can be expected to happen to LED D4 when the switch SW2 is in the Up position?
2. What can be expected to happen to LED D4 when the switch SW2 is in the Down position?
3. Simulate the circuit. Does the circuit operate as expected? Explain the operation.

**Circuit 3**

1. Under what conditions will the green LED D5 be turned on?
2. Under what conditions will the red LED D6 be turned on?
3. What will happen if the initial (low) voltage is changed on the pulse generator to 0v? Why?
4. What property on the pulse generator is changed to make the diodes blink faster?

**General**

1. Use the circuit to investigate the difference between a SPST latched switch and a SPST momentary switch.
2. Replace the 5v battery with a 5v power terminal. Experiment with the value of the voltage.

# 4 Digital to Analogue Conversion

## *4.1 Operational Amplifiers (Op-Amp)*

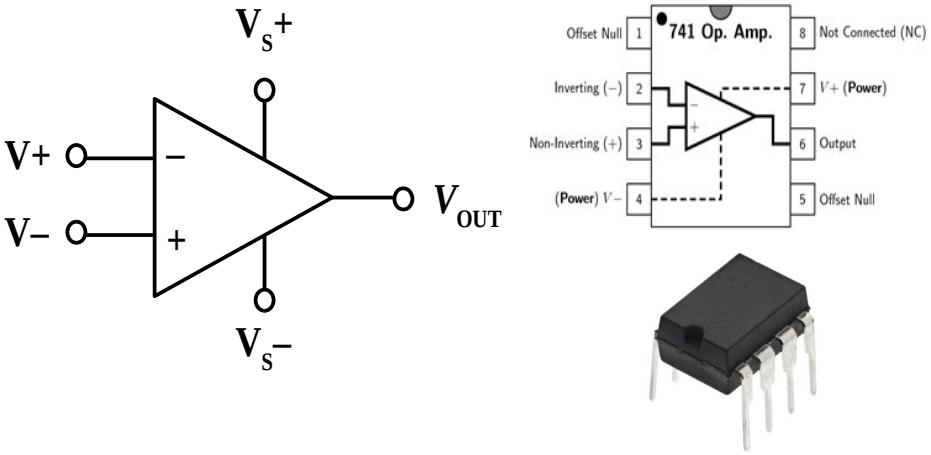



*Figure 25: Operational Amplifier (op-amp)*

**V+**: non-inverting input
**V−**: inverting input
$\mathbf{V_{OUT}}$: output
$\mathbf{V_S+}$: positive power supply
$\mathbf{V_S−}$: negative power supply

An operational amplifier (op-amp), illustrated in Figure 25, is a Direct Current (DC)-coupled high-gain electronic voltage amplifier with a differential input and, usually, a single-ended output. An op-amp produces an output potential (relative to circuit ground) that is typically 100,000 times larger than the potential difference between its input terminals. Op-amps have their origins in analogue computers, where they were used to perform mathematical operations in linear, non-linear, and frequency-dependent circuits.

Op-amps have enormous open-loop gain $\mathbf{A_{OL}}$. Open-loop gain is the gain of the op-amp Integrated Circuit (IC) itself with no feedback. That gain is too big to be useful, so it is lowered with negative feedback. The gain with feedback is the closed-loop gain $\mathbf{A_{CL}}$.

Characteristics of Op-Amps:

1 Very high gain
2 Very high input impedance
3 Very low output impedance

Below are schematics for the two basic feedback circuits: the inverting amplifier, as illustrated in Figure 26, and the non-inverting amplifier, as illustrated in Figure 27. The gain equation for each circuit is included. Notice that the gain equations do not include frequency as a variable

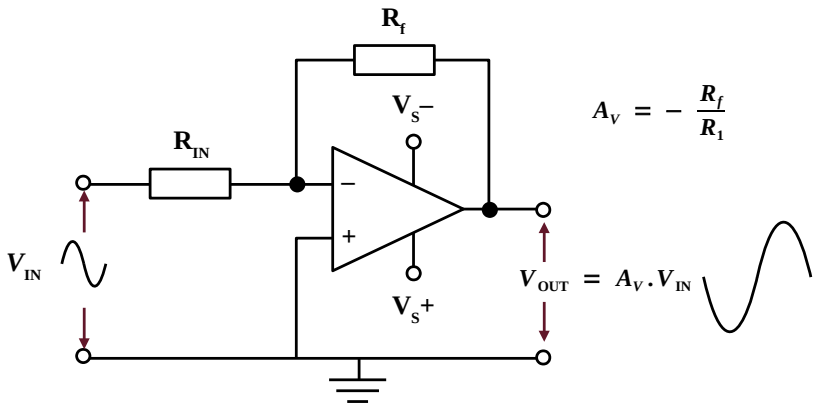



*Figure 26: Inverting op-amp*

**Key Differences Between Inverting and Non-Inverting Amplifier**

- The key factor of differentiation between inverting and non-inverting amplifier is the basis of the phase relationship existing between input and output.
- In the case of the inverting amplifier, the output is out of phase with respect to the input, whereas for the non-inverting amplifier, both input and output are in phase.
- The input signal in the inverting amplifier is applied at the negative terminal of the op-amp, whereas the input in the case of a non-inverting amplifier is provided at the positive terminal.
- The gain provided by the inverting amplifier is the ratio of the resistances, whereas the gain of the non-inverting amplifier is the summation of 1 and the ratio of the resistances.

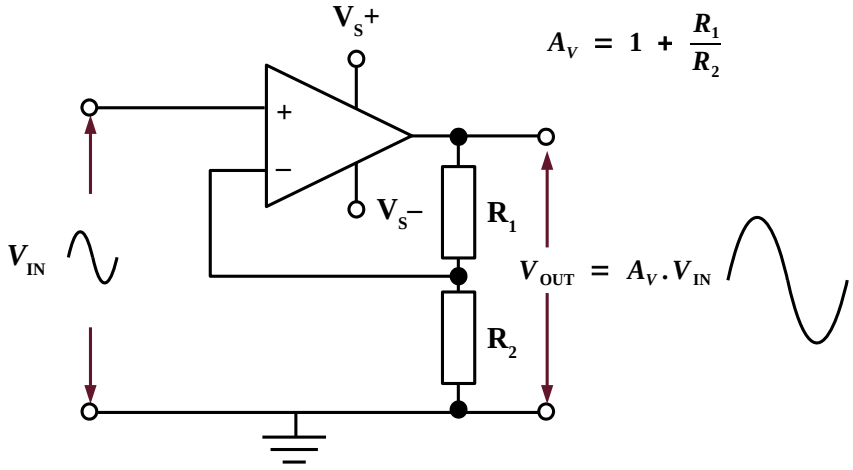



*Figure 27: Non-inverting op-amp*

Inverting Op-Amps are used mainly for high frequency applications where high input impedance is not a requirement. This is because, the rate of change in the output voltage caused by a step change on the input, called the slew rate, of the inverting Op-Amp is high, compared to non-inverting configuration. This configuration is ideal for summing amplifiers.

Non-inverting op-amps are used where high input impedance is a requirement, a typical application is as a buffer amplifier.

**Bode plot**
(Gain vs Frequency response for an amplifier)

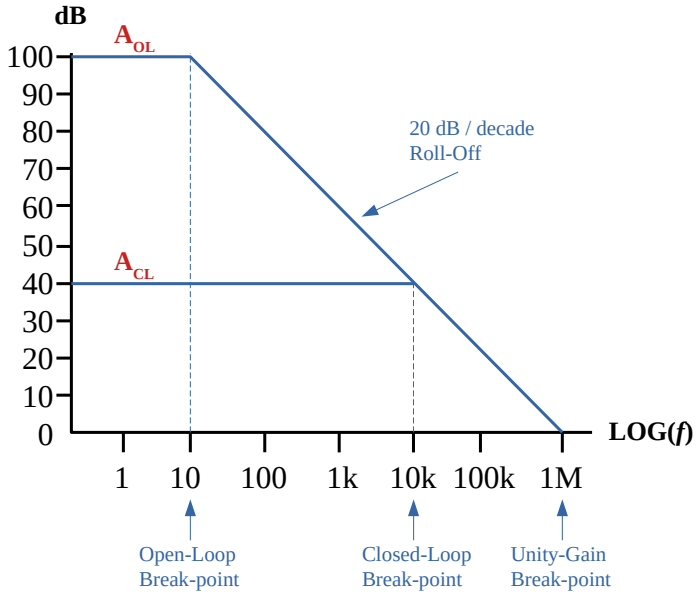



*Figure 28: Bode Plot for a typical Op-Amp (741)*

A Bode plot is a graph of the frequency response of a system. It is a combination of a Bode magnitude plot, expressing the magnitude (usually in decibels) of the frequency response, and a Bode phase plot, expressing the phase shift.

From the bode plot in Figure 28 it appears that the closed-loop gain doesn't change with frequency until the line for $A_{CL}$ meets the line for $A_{OL}$ on the amplifier's Bode plot. A Bode plot is a graph that shows how the gain of an amplifier "rolls off" as signal frequency increases. Many op-amps, such as the 741 (part of your component kit), roll off at 20 dB per decade. (A decade is when the frequency changes by a factor of 10). The open-loop gain of an op-amp starts rolling off at a relatively low frequency, maybe 10 Hertz. But they have so much $A_{OL}$ that it doesn't get to 1 (0 dB) until the frequency reaches Mega-Hertz (MHz).

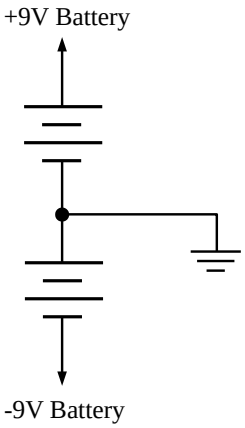



*Figure 29: Bipolar power supply*

## *4.2 Digital Converters*

Connecting digital circuitry to sensor devices is simple if the sensor devices are inherently digital themselves. Switches, relays, and encoders are easily interfaced with gate circuits due to the on/off nature of their signals. However, when analogue devices are involved, interfacing becomes much more complex. What is needed is a way to electronically translate analogue signals into digital (binary) quantities, and visa-versa. An *analogue-to-digital converter* (**ADC**), performs the former task while a *digital-to-analogue converter*, or **DAC**, performs the latter.

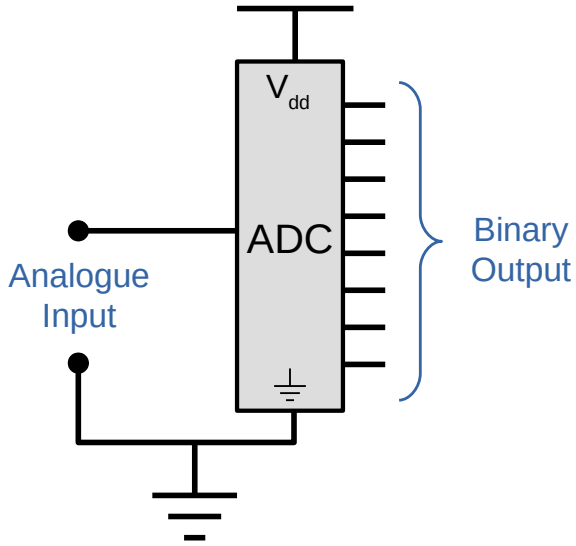



*Figure 30: Analogue to Digital Converter (ADC)*

An ADC inputs an analogue electrical signal such as voltage or current and outputs a binary number. In block diagram form, it can be represented illustrated in Figure 30.

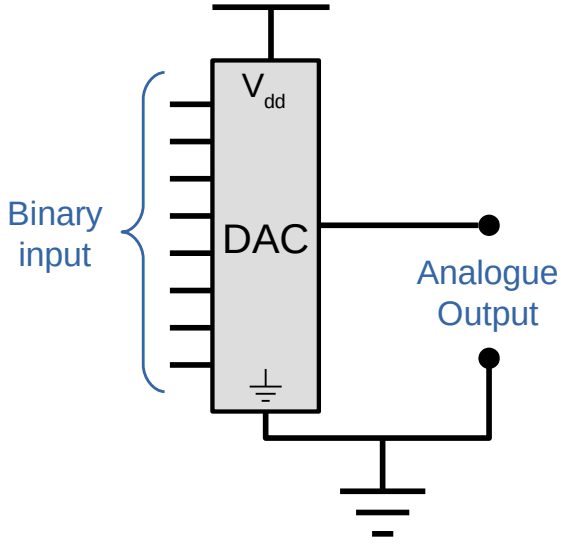



*Figure 31: Digital to Analogue Converter (DAC)*

A DAC, on the other hand, illustrated in Figure 31, inputs a binary number and outputs an analogue voltage or current signal.

Together, as illustrated in Figure 32, they are often used in digital systems to provide complete interface with analogue sensors and output devices for control systems such as those used in automotive engine controls:

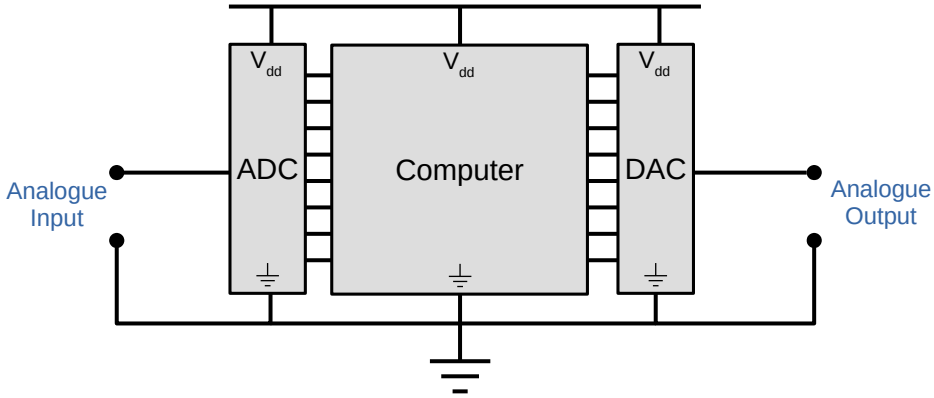



*Figure 32: Digital control system with analogue I/O*

## *4.3 The R/2nR DAC*

This DAC circuit, otherwise known as the *binary-weighted-input* DAC, is a variation on the inverting summer op-amp circuit. Recall that the classic inverting circuit is an operational amplifier using negative feedback for controlled gain, with several voltage inputs and one voltage output. The output voltage is the inverted (opposite polarity) sum of all input voltages:

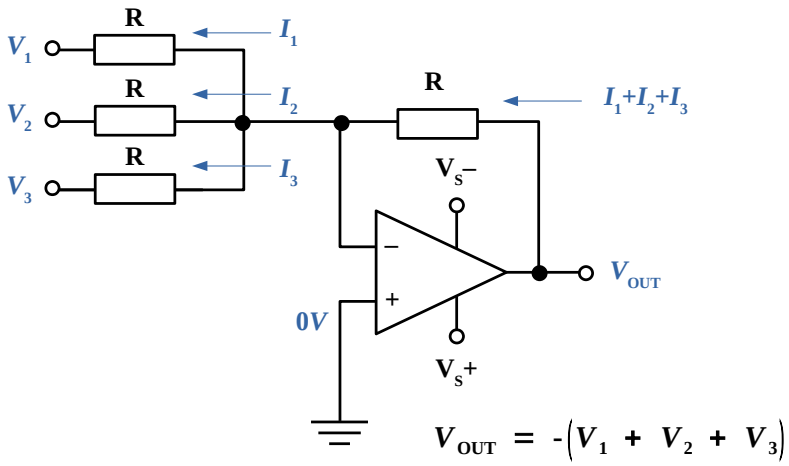



*Figure 33: Inverting Summer*

As illustrated in Figure 33, for a simple inverting summer circuit, all resistors must be of equal value. If any of the input resistors were different, the input voltages would have different degrees of effect on the output, and the output voltage would not be a true sum. Consider; however, intentionally setting the input resistors at different values. Suppose the input resistor values were set at multiples of two: R, 2R, and 4R, instead of all the same value R as illustrated in Figure 34.

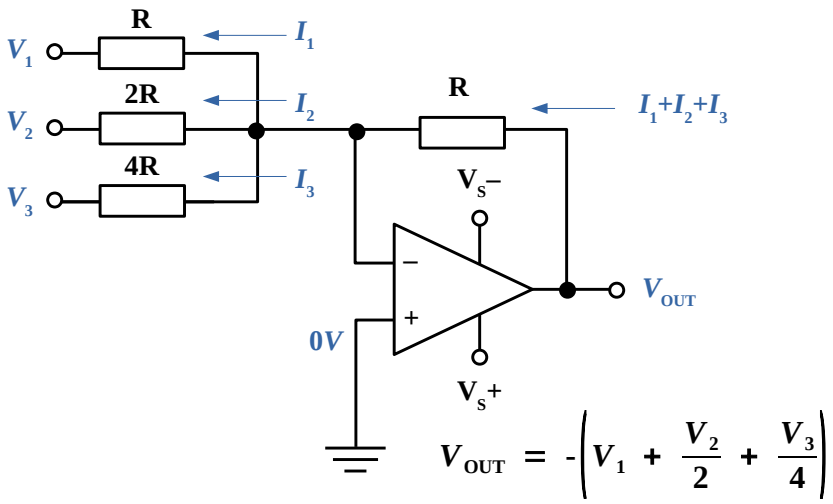



*Figure 34: Inverting Summer with different input values*

Starting from $V_1$ and going through $V_3$, this would give each input voltage exactly half the effect on the output as the voltage before it. In other words, input voltage $V_1$ has a 1:1 effect on the output voltage (gain of 1), while input voltage $V_2$ has half that much effect on the output (a gain of 1/2), and $V_3$ half of that (a gain of 1/4). These ratios are not arbitrarily chosen: they are the same ratios corresponding to place weights in the binary numeration system. If the inputs of this circuit are driven with digital gates so that each input is either 0 volts or full supply voltage, the output voltage will be an analogue representation of the binary value of these three bits.

Charting the output voltages for all eight combinations of binary bits (000 through 111) input to this circuit, the following progression of voltages, displayed in the chart in Figure 35, are output.

| Binary I/P | O/P voltage |
|---|---|
| 000 | 0.00 V |
| 001 | -1.25 V |
| 010 | -2.50 V |
| 011 | -3.75 V |
| 100 | -5.00 V |
| 101 | -6.25 V |
| 110 | -7.50 V |
| 111 | -8.75 V |

*Figure 35: Binary logic to voltages*

Note that with each step in the binary count sequence, results a 1.25 volt change in the output. This circuit is very easy to simulate using Proteus.

If it is necessary to expand the resolution of this DAC (add more bits to the input), all is needed is to add more input resistors, holding to the same power-of-two sequence of values as illustrated in Figure 36.

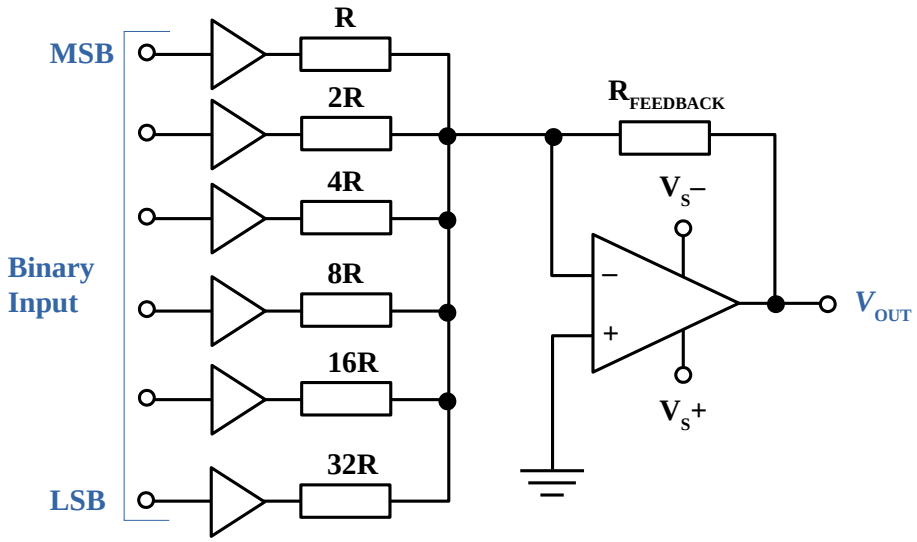



*Figure 36: 6-bit binary-weighted DAC*

It should be noted that all logic gates must output exactly the same voltages when in the "high" state. If one gate is outputting +5.02 volts for a "high" while another is outputting only +4.86 volts, the analogue output of the DAC will be adversely affected. Likewise, all "low" voltage levels should be identical between gates, ideally 0.00 volts exactly. It is recommended that CMOS output gates are used, and that input/feedback resistor values are chosen so as to minimise the amount of current each gate has to source or sink.

**The R/2R DAC**

An alternative to the binary-weighted-input DAC is the so-called R/2R DAC, which uses fewer unique resistor values. A disadvantage of the former DAC design was its requirement of several different precise input resistor values: one unique value per binary input bit. Manufacture may be simplified if there are fewer different resistor values to purchase, stock, and sort prior to assembly.

Of course, the last DAC circuit could be modified to use a single input resistance value, by connecting multiple resistors together in series as illustrated in Figure 37.

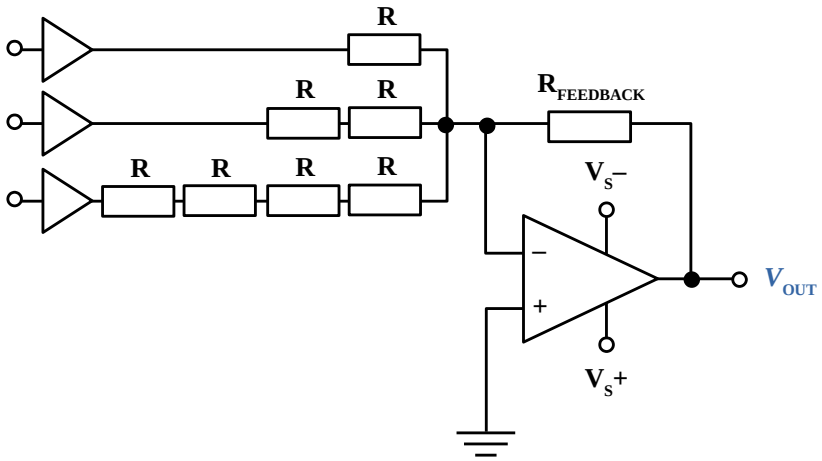



*Figure 37: 3-bit binary-weighted DAC*

Unfortunately, this approach merely substitutes one type of complexity for another: volume of components over diversity of component values. There is, however, a more efficient design methodology.

By constructing a different kind of resistor network on the input of the summing circuit, the same kind of binary weighting can be achieved with only two kinds of resistor values, and with only a modest increase in resistor count. This "ladder" network looks like the illustration in Figure 38.

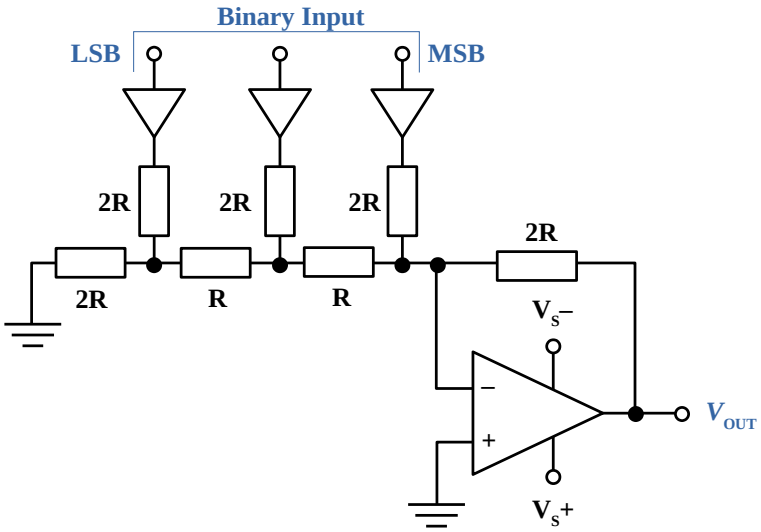



*Figure 38: "Ladder" resistor network*

Mathematically analysing this ladder network is more complex than for the previous circuit, where each input resistor provided an easily-calculated gain for that bit. For those interested in pursuing the intricacies of this circuit further, opt to use Thevenin's theorem for each binary input (remember to consider the effects of the *virtual ground*),

## 4.4 *Exercise #4: Analogue to Digital converter*

The objectives of this exercise are to:

1. Create a simple schematic which uses the following components.
   a) Switches - SPDT
   b) Op-Amp (741)
   c) Power and ground terminals.
2. Describe the main characteristics of an Op-Amp
3. Calculate the gain of an Inverting amplifier
4. Calculate the gain of an non-Inverting amplifier
5. Describe the basic operation of a Digital to Analogue Converter
6. Describe the basic operation of a Analogue to Digital Converter
7. Verify the correct operation of the **R/2R DAC** circuit given.

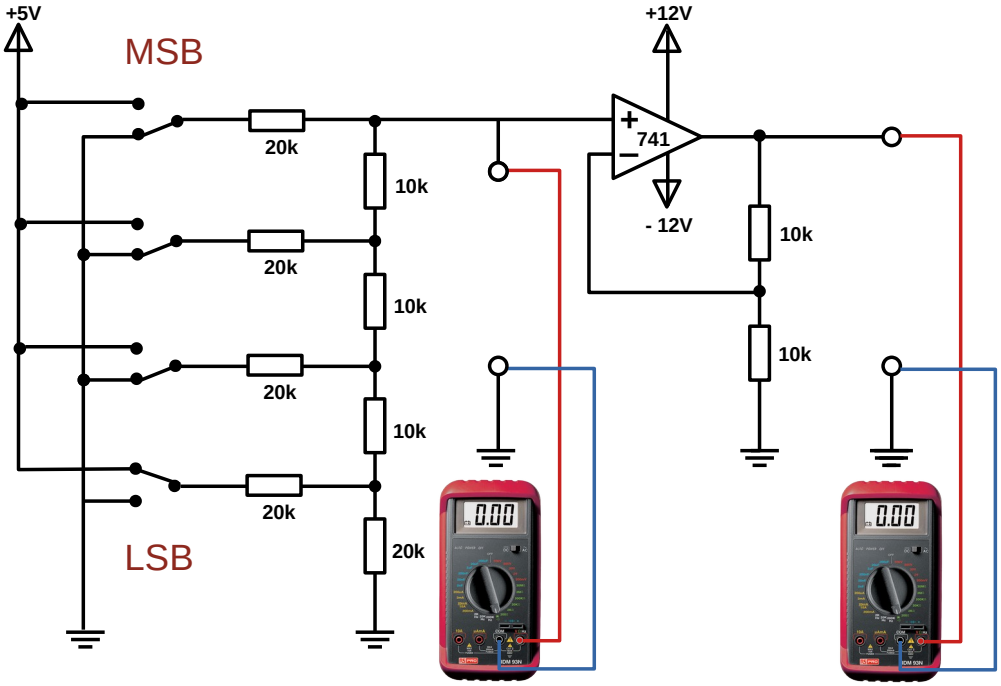



*Figure 39: Exercise: DAC*

1 **Getting Started:** As in the previous exercise - to start the Proteus ISIS program, click on the Start button in windows and select the All Programs, Proteus 8 Professional, add a new project. The Proteus ISIS editor will then load and run.

2 Select the components which will be used in the design:

3 Place components, Terminals and instruments on the page:

4 Compile a table of Binary input versus Analogue output for the circuit and verify circuit operation.

As there are two voltage rails, 12V to supply the op-amp and 5V for the switches, it is necessary to configure these for Proteus. Add three Voltage rails.

Select: **Design** >> **Configure Power Rails …**

Name: **12V** Voltage: [**Change to 12**]
Name: **-12V** Voltage: [**Change to -12**]
Name: 5**V** Voltage: [**Change to 5**]

Click OK.

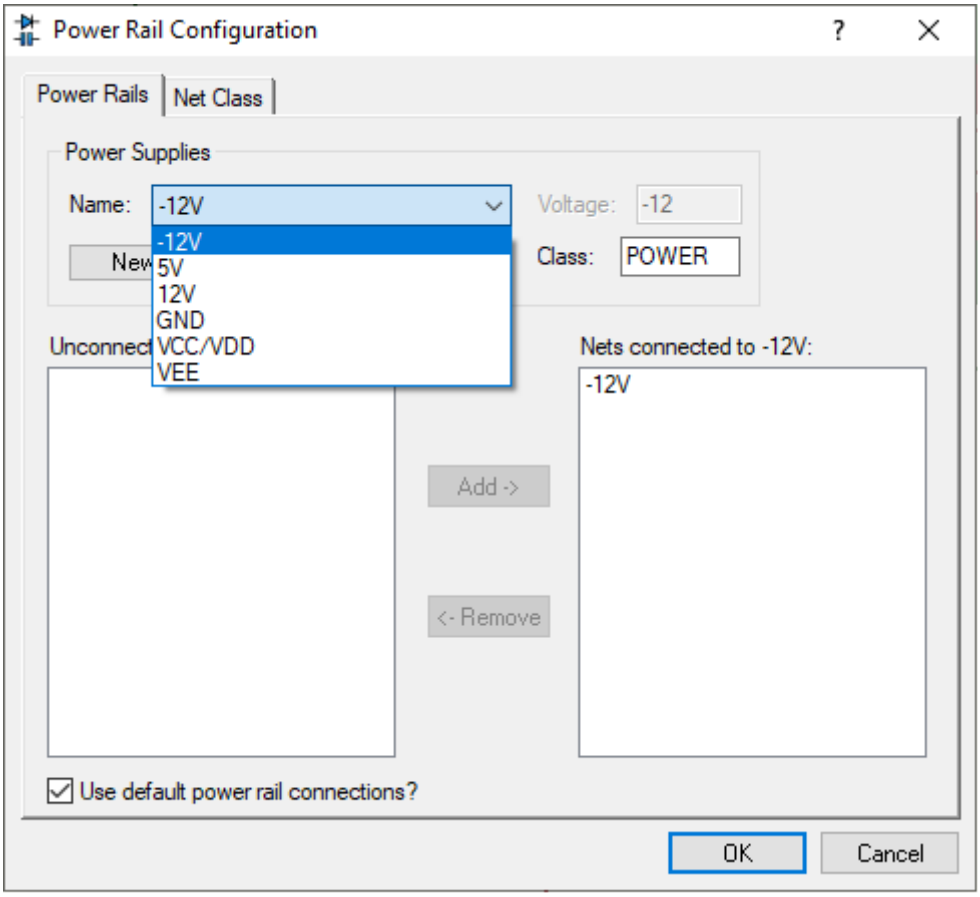



*Figure 40: Power Rail Configuration*

**Note**: It is important to ***save work frequently***.

Confirm output matches the Digital to Analogue table.

*Table 1: DAC conversion table*

| Base 10 | Base 16 | Base 2 | Voltage |
|---|---|---|---|
| 0 | 0 | 0000 | 0V |
| 1 | 1 | 0001 | 0.63V |
| 2 | 2 | 0010 | 1.25V |
| 3 | 3 | 0011 | 1.88V |
| 4 | 4 | 0100 | 2.5V |
| 5 | 5 | 0101 | 3.13V |
| 6 | 6 | 0110 | 3.75V |
| 7 | 7 | 0111 | 4.38V |
| 8 | 8 | 1000 | 5V |
| 9 | 9 | 1001 | 5.63V |
| 10 | A | 1010 | 6.25V |
| 11 | B | 1011 | 6.88V |
| 12 | C | 1100 | 7.5V |
| 13 | D | 1101 | 8.13V |
| 14 | E | 1110 | 8.75V |
| 15 | F | 1111 | 9.38V |

# 5 Logic Gates

## *5.1 An introduction to Logic Gate families*

Digital Integrated Circuit (IC) devices are often classified according to the semiconductor technology used in their manufacture. The logic family to which a device belongs is largely instrumental in determining its operational characteristics (such as power consumption, speed, and immunity to noise).

The two basic logic families are Complementary Metal Oxide Semiconductor **(CMOS)** and Transistor-Transistor-Logic **(TTL)**. Each of these families is then further sub-divided.

| | |
|---|---|
| TTL | **74 series** |
| CMOS | **4000 series** |

**TTL (74 series) - general characteristics**
There are several sub categories of the 74 series of logic ICs numbered from 74xx00 onwards with letters (xx) in the middle of the number to indicate the type of circuitry, eg 74LS00 and 74HC00. The original family (now obsolete) had no letters, eg 7400.

74 series TTL operate with a 5-volt power supply between Voltage Common Collector ($V_{CC}$) pin and ground (GND). TTL input signals are considered "low" when between 0V and 0.8V with respect to GND and "high" when between 2V and the $V_{CC}$ (5V).

The **74LS** (Low-power Schottky) family (like the original) uses TTL (Transistor-Transistor Logic) circuitry which is fast but requires more power than later families. The 74 series is often still called the 'TTL series' even though the latest ICs do not use TTL!

The **74HC** family has High-speed CMOS circuitry, combining the speed of TTL with the very low power consumption of the 4000 series. They are CMOS ICs with the same pin arrangements as the older 74LS family.
**Note**: 74HC inputs cannot be reliably driven by 74LS outputs because the voltage ranges used for logic 0 are not quite compatible, use 74HCT instead.

The **74HCT** family is a special version of 74HC with 74LS TTL compatible inputs so 74HCT can be safely mixed with 74LS in the same system. In fact 74HCT can be used as low-power direct replacements for the older 74LS ICs in most circuits.

**Note:** The minor disadvantage of 74HCT is a lower immunity to noise, but this is unlikely to be a problem in most situations.

**Field-Effect Transistor**

Field Effect Transistor (FET) employs the concept that charge on a nearby object can attract charges within a semiconductor channel. It operates by using an electric field effect. There are two types P-channel and N-channel. For the purpose of description consider the N-channel type. The gate of the FET controls the flow of carriers, in this case electrons, flowing from the Source (S) to Drain (D). This is implemented by controlling the size and shape of the conductive channel as increasing the voltage on the Gate (G) can either deplete or enhance the number of charge carriers available in the channel.

As it is only the electric field that controls the current flowing in the channel, the device is said to be voltage operated and it has a high input impedance, usually many megaohms (MΩ). This can be a distinct advantage over the bipolar transistor that is current operated and has a much lower input impedance.

FET circuits consume much lower levels of power than ICs using bipolar transistor technology.

A Metal Oxide Semiconductor FET (MOSFET) is a particular type of FET based on the operation of the capacitor. It uses an insulated layer between the Gate (G) and the channel which is made of a layer of metal set down on the silicon oxide which in turn is on the silicon channel. Typically this is formed from a layer of oxide of the semiconductor as illustrated in Figure 41.

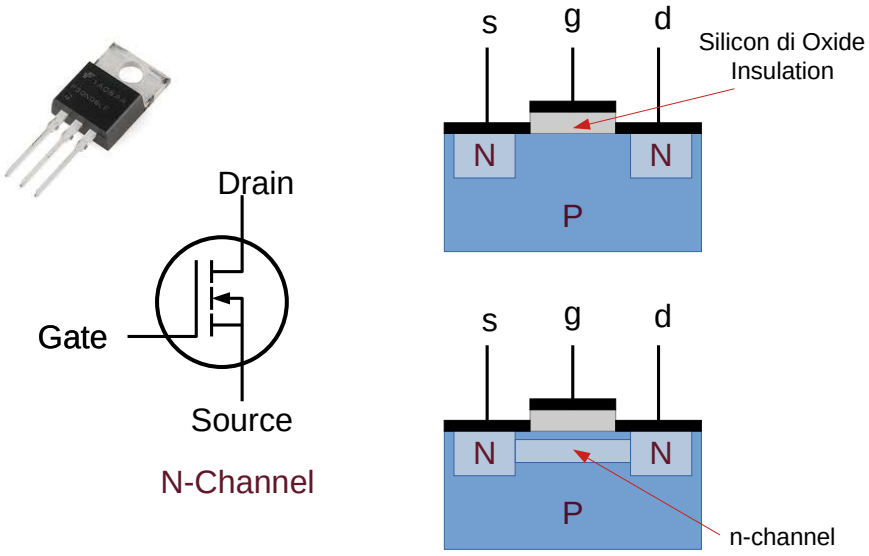



*Figure 41: N-Channel MOSFET*

**Complementary Metal–Oxide–Semiconductor (CMOS)**

CMOS is a type of MOSFET fabrication process that uses complementary and symmetrical pairs of p-channel and n-channel MOSFETs for logic functions. The technology is used for constructing ICs, including microprocessors, microcontrollers, memory chips, and other digital logic circuits. CMOS technology is also used for analogue circuits such as image sensors, data converters, RF circuits, and highly integrated transceivers for many types of communication.

## CMOS (4000 series) - general characteristics

CMOS digital ICs are now the preferred choice over TTL for most applications. There are some good reasons for this.

### Advantages of CMOS

The power consumption of CMOS is much lower than most versions of TTL, making CMOS ideal for battery-powered circuits. Since CMOS consumes less power, CMOS circuits are less likely to produce power supply transients when switching between states.

CMOS can operate over a wide supply voltage range, typically +3 to +15V; however, the operating speed of CMOS drops as supply voltage drops. The best supply voltage for CMOS is usually +5 or +9V.

CMOS inputs have a high impedance and are easy to drive, meaning CMOS inputs draw little current and don't "load down" devices providing input signals to them. CMOS outputs have superior fan-out capabilities compared to TTL; in some cases, CMOS can drive five times as many other inputs as a comparable TTL device. CMOS devices can tolerate variations in the power supply voltage much better than TTL devices.

### Disadvantages of CMOS

The major disadvantage of CMOS devices are their susceptibility to damage due to static electricity. Special care must be taken when handling and storing CMOS devices.

Another problem with CMOS is its lack of speed compared to TTL. While newer versions of CMOS have closed this gap significantly, the fastest versions of TTL can still beat the fastest versions of CMOS.

### Some points to note

To help differentiate between TTL and CMOS supply voltages, the supply voltage for CMOS is referred to as $V_{dd}$ and the ground connection is sometimes referred to as $V_{ss}$. The supply voltage for TTL is referred to as $V_{cc}$ and the ground connection is referred to as ground.

The pin arrangements of the two types of IC are quite different. A 74 series NAND gate IC cannot be replaced by simply plugging a 4000 series NAND gate into the same socket. The diagrams, in Figure 42, show the pin layout for NAND gate ICs in each of the two series.

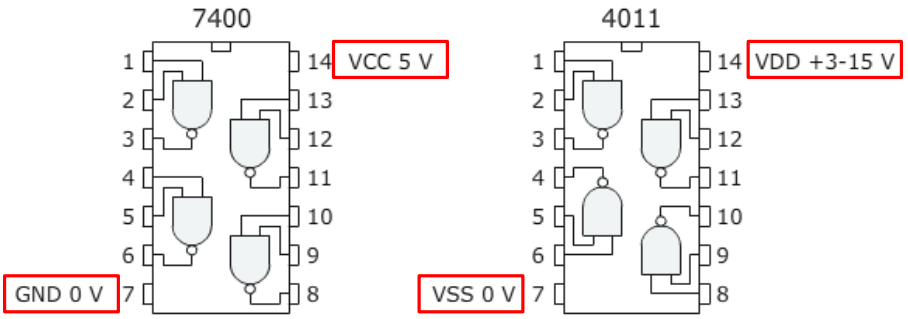



*Figure 42: IC pin arrangements*

The CMOS circuitry used in the **74HC** and **74HCT** series ICs means that they are static sensitive. Touching a pin while you are charged with static electricity (from your clothes for example) may damage the IC. Most of these ICs are quite tolerant to static electricity yet all the anti-static safety precautions as used with CMOS devices should be adhered to. ICs should be left in their protective packaging until they need to be used.

For most new projects the 74HC family is the best choice. Remember the 74LS and 74HCT sub categories require a 5V supply so they are not convenient for battery operation.

| **Property** | **4000 Series** | **74 Series 74HC** | **74 Series 74HCT** | **74 Series 74LS** |
|---|---|---|---|---|
| **Technology** | CMOS | High-speed CMOS | High-speed CMOS TTL compatible | TTL Low-power Schottky |
| **Power Supply** | 3 to 15V | 2 to 6V | 5V ±0.5V | 5V ±0.25V |
| **Inputs** | Very high impedance. Unused inputs must be connected to $+V_{dd}$ or 0V. Inputs cannot be reliably driven by 74LS outputs unless a 'pull-up' resistor is used (see below). | | Very high impedance. Unused inputs must be connected to +Vs or 0V. Compatible with 74LS (TTL) outputs. | 'Float' high to logic 1 if unconnected. |
| **Outputs** | Can sink and source about 5mA (10mA with 9V supply), enough to light an LED. To switch larger currents use a transistor. | Can sink and source about 20mA, enough to light an LED. To switch larger currents use a transistor. | | Can sink up to 16mA (enough to light an LED), but source only about 2mA. To switch larger currents use a transistor. |
| **Fan-out** | One output can drive up to 50 CMOS, 74HC or 74HCT inputs, but only one 74LS input. | One output can drive up to 50 CMOS, 74HC or 74HCT inputs, but only 10 74LS inputs. | | One output can drive up to 10 74LS inputs or 50 74HCT inputs. |
| **Maximum Frequency** | about 1MHz | about 25MHz | about 25MHz | about 35MHz |
| **Power consumption** of the IC itself | A few µW. | A few µW. | A few µW. | A few mW. |

*Figure 43: Summary of important properties of the most popular logic families*

## *5.2 Logic Gate Symbols*

Figure 44 shows the American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE) and Military Specification (MIL), knowm as ANSI/IEEE 91, the International Electrotechnical Commission (IEC), IEC 60617 and the British System (BS) BS 3939 symbols for the eight major logic gates: NOT, AND, NAND, OR, NOR, XOR, XNOR and the buffer.

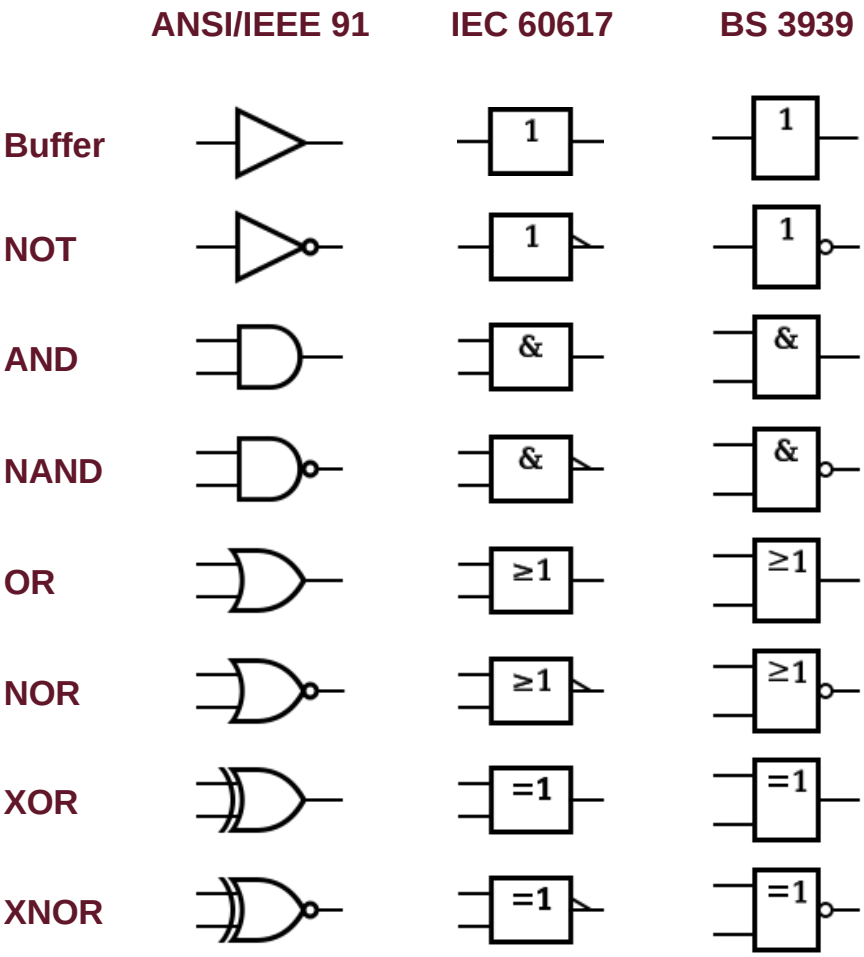



*Figure 44: Logic Gates*

## *5.3 Exercise #5: Logic Gates*

The objectives of this exercise are to:

1. Simulate digital circuits using Proteus.
2. Identify logic gates when any of the three commonly used logic symbols are used. (ANSI, IEC and British Standard symbols).
3. Down load a data sheet to view the specifications of an IC.
4. Draw the truth table for popular logic gates.
5. Draw a schematic to represent a Boolean logic equation.
6. Describe what is meant by an ‘open collector’ output.

---

# Assignments:

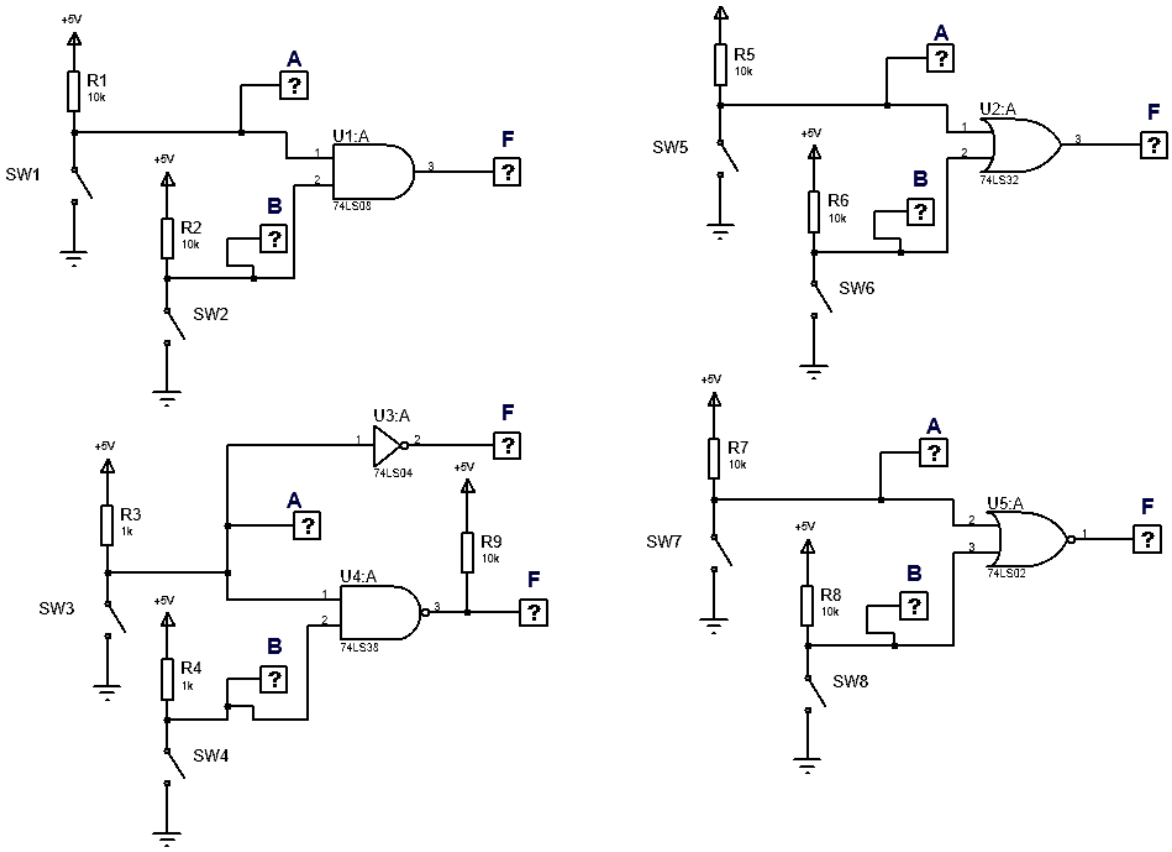



*Figure 45: Exercise: Logic Gates*

1 Identify each of the logic gates shown in Figure 45.

2 Use Proteus to draw the schematic in Figure 45.

Components used:

- 74LS08, 74LS32, 74LS04, 74LS02, 74LS38 - the “TTL 74LS series” library
- The big LOGICPROBE from the debug library
- RES - Generic resistor
- SW-SPST – single pole single throw switch.

Terminals used:

- Power & ground

The font size for the logic inputs and outputs should be 0.2 inch (5mm) high – set using the properties.

3 Simulate the circuit and write down the truth table **for each gate**.

| **A** | **B** | **F** |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

If the truth table for a particular gate is NOT as expected, download the IC’s data sheet, and verify if the gate is being used correctly.

4 Verify if the SPST switch can be replaced with a SPST push button switch?

5 Use Proteus to draw a schematic to represent the following Boolean logic equation. Label each input and output on the diagram.

$$F = A.\overline{B} + \overline{A}.B$$

6 Simulate the circuit and verify the truth table.

7 Design a circuit to show that $\overline{A.B} \neq \overline{A}.\overline{B}$

Note: To obtain the overbar (inversion) on an input / output or wire label – place a $ before the letter when editing the text.

# 6 Digital Logic

## *6.1 Half Adder*

Consider the addition of single bits.
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

These are the least possible single-bit combinations. But the result for 1+1 is 10. Though this problem can be solved with the help of an EXOR Gate, the sum result must be re-written as a 2-bit output.

Thus the above equations can be written as:
0 + 0 = 00
0 + 1 = 01
1 + 0 = 01
1 + 1 = 10

Here the output '1'of '10' becomes the carry-out. The result is shown in a truth-table in Figure 46. 'SUM' is the normal output and 'CARRY' is the carry-out.

| INPUTS | | OUTPUTS | |
|---|---|---|---|
| **A** | **B** | **SUM** | **CARRY** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

*Figure 46: Two bit truth table*

From the equation it is clear that this 1-bit adder can be easily implemented with the help of XOR Gate for the output 'SUM' and an AND Gate for the carry. Take a look at the implementation in Figure 47.

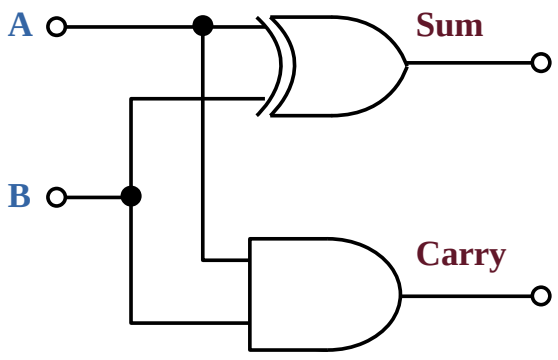



*Figure 47: Half Adder*

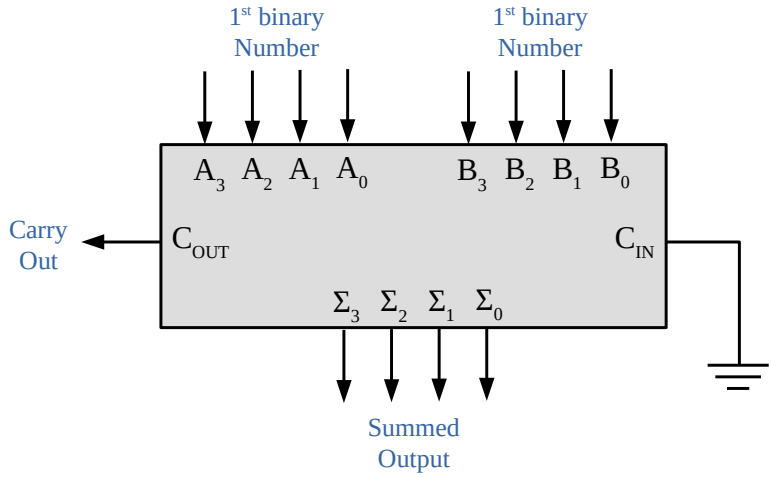



*Figure 48: Block Diagram of a Single Digit Full Adder*

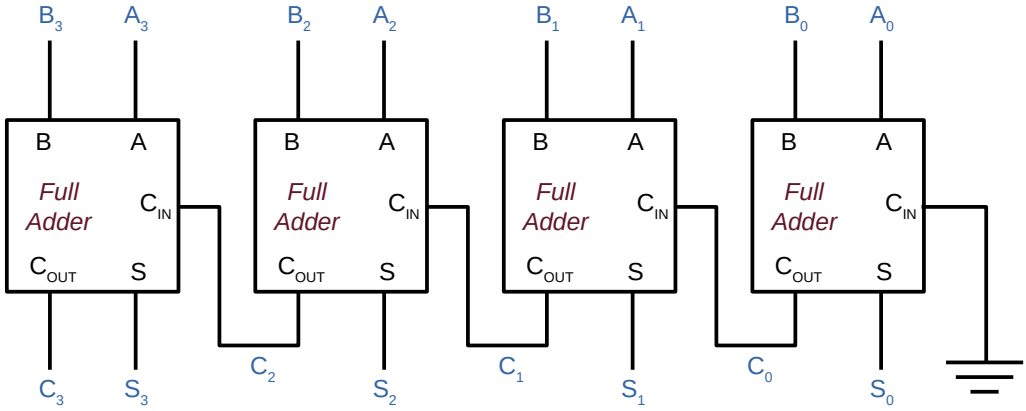



*Figure 49: 4-Bit (digit) Full Adder*

The 4008 is a 4-bit binary full adder, illustrated in Figure 50, with two 4-bit data inputs (A0 to A3, B0 to B3), a carry input ($C_{IN}$), four sum outputs (S0 to S3), and a carry output ($C_{OUT}$). The outputs give the sum of the two 4-bit binary numbers.

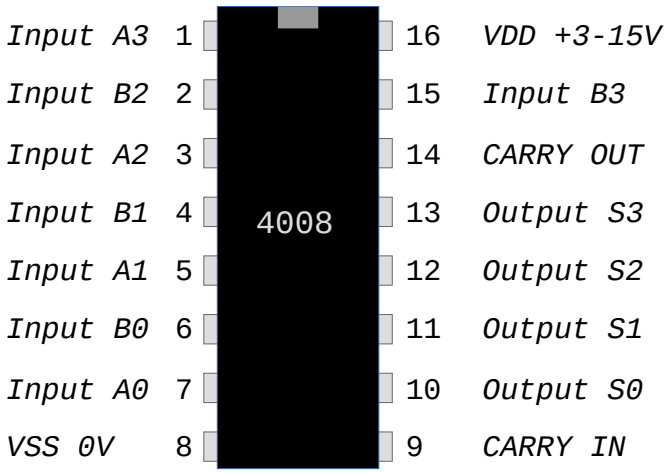
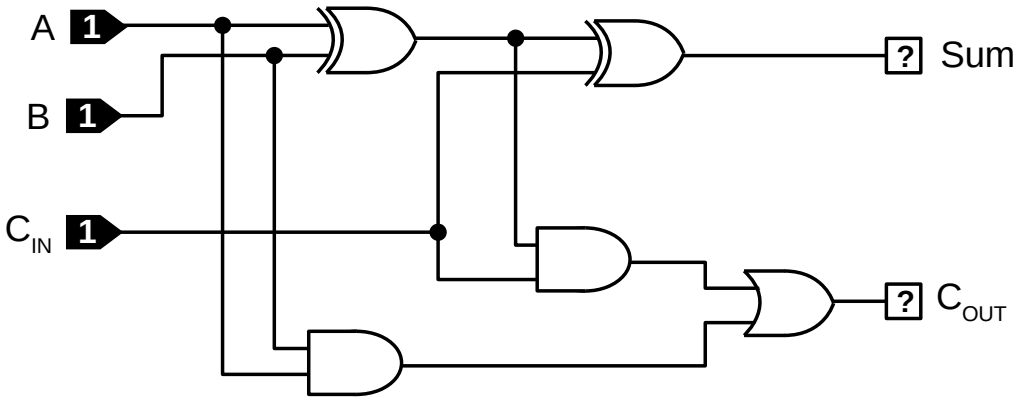



*Figure 50: 4008 4-bit binary full-adder*




AND 74LS08
OR 74LS32
XOR 74LS86

*Figure 51: Full adder*

## *6.2 Exercise #6: Digital Logic - Full adder, Two bit adder*

The objectives of this exercise are to:

1. Create a schematic diagram of a digital circuit which uses the following components.
   a) Logic gates
   b) Logicprobes
   c) Logicstate
2. Describe the operation of the full adder and the ripple carry adder.
3. Use the Block copy command.
4. Simulate the circuit and obtain a truth table.

---

# Components:

- All the Logic gates can be found in the ***"TTL74LS series"*** library
- The Logicprobe & Logicstate symbols can be found in the ***"Debugging tools"*** library

**Note:** Use the 74LSXX gates (e.g. 74LS86) or the design may not simulate.

# Assignments:

1. Draw a schematic diagram for a full adder.
   a) Simulate the full adder.
   b) Use the Logicstate inputs to apply logic values to the input.
   c) Label the inputs and outputs appropriately
   d) Record the logic levels at the outputs as observed using the Logicprobes.
   e) Draw up a truth table for the Full adder.
   f) Write down the Boolean logic equations that describe the operation of a full adder.

2. Use the "Block Copy" command make a copy of the full adder to form a schematic of a two bit adder as shown in Figure 52.
   a) Simulate the Two bit adder.
   b) Use the Logicstate inputs to apply logic values to the inputs.
   c) Record the logic levels at the outputs as observed using the Logicprobes.
   d) Test the Two bit adder with several two bit numbers. The two bit numbers are A ( A1,A0 ) and B (B1,B0).

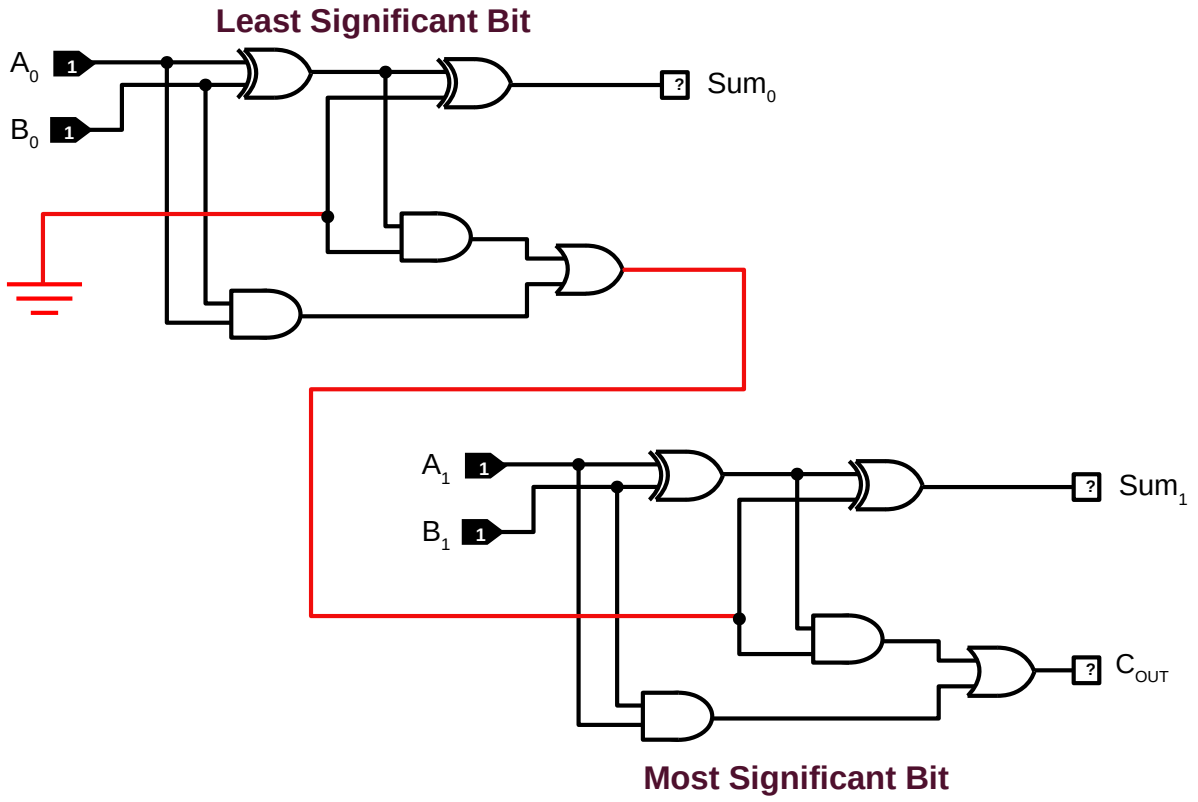



*Figure 52: Two bit adder*

**Note**: It is important to ***save work frequently***.

# 7 Digital logic, flip-flops

## *7.1 SR Flip-flop*

The Set-Reset (SR) flip-flop, also known as a SR Latch, is a basic sequential logic circuit. This flip-flop is essentially a one-bit memory bistable device that has two inputs, one to SET the device (***S*** = 1), and one which will RESET the device (***R*** = 1, therefore ***Q*** = 0, $\bar{Q}$ = 1). The ***R*** input resets the flip-flop back to its original state with the output ***Q*** that will be either at a logic level 1 or logic 0 depending upon this ***S***/***R*** condition.

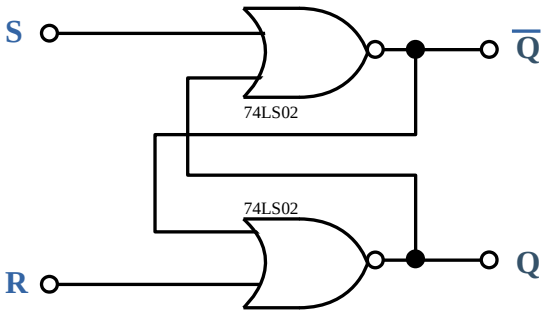



*Figure 53: Cross-coupled NOR gates*

The two outputs ***Q*** and $\bar{Q}$ are complementary to each other, if one is logic 0 the other is logic 1 and vice versa. Consider the different possibilities of inputs and their corresponding outputs.

Situation 1: ***R*** = 0 and ***S*** = 0
In the first case, the inputs of both the NOR gates are logic 0. As neither of them are dominating inputs, they have no effect on the output. So, the output retains their previous states i.e., there is no change in the output. This condition is called as **Hold Condition**.

Situation 2: ***R*** = 0 and ***S*** = 1
In this case, the ***S*** input is 1, which means the output of the top NOR gate will become 0. As a result, both the inputs of bottom NOR gate become 0 and hence the output of the bottom NOR gate and thus the value of Q is 1. As 1 at input ***S*** makes the output switch to one of its stable states and sets it to 1.

Situation 3: ***R*** = 1 and ***S*** = 0
In this case, the ***R*** input is 1, which means the output of the bottom NOR gate will become 0 i.e., ***Q*** is 0. As a result, both the inputs of the top NOR gate become 0 and hence the output of the top NOR gate is 1. As 1 at input R makes the output switch to one of its stable states and resets it to 0, the R input is known as RESET input.

Case 4: ***R*** = 1 and ***S*** = 1
This input condition is forbidden as it forces outputs of both NOR Gates to become 0, which is a violation of complementary outputs. Even if this input condition is applied, if the next inputs become ***R*** = 0 and ***S*** = 0 (hold condition), then it causes a ‘race condition’ between the NOR Gates, which causes an unstable or unpredictable state at the output. Hence, the input condition ***R*** = ***S*** = 1 is simply not used.

## 7.2 *Gated (Clocked) SR flip-flop*

It is sometimes desirable in sequential logic circuits to have a SR flip-flop that only changes state when certain conditions are met regardless of the condition of either ***S*** or the ***R*** inputs. By connecting a 2-input AND gate in series with each input terminal of the SR Flip-flop a Clocked SR Flip-flop (Gated SR Flip-flop) can be created. This extra conditional input is called an Enable (EN) input. The addition of this input means that the output at **Q** only changes state when it is logic level 1, and can therefore be used as a clock (CLK) input making it level-sensitive as illustrated in Figure 54.

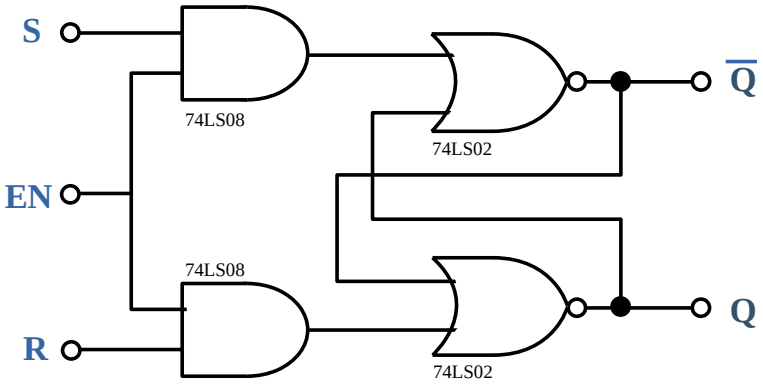



*Figure 54: Gated SR flip-flop*

## 7.3 *D-latched flip-flop*

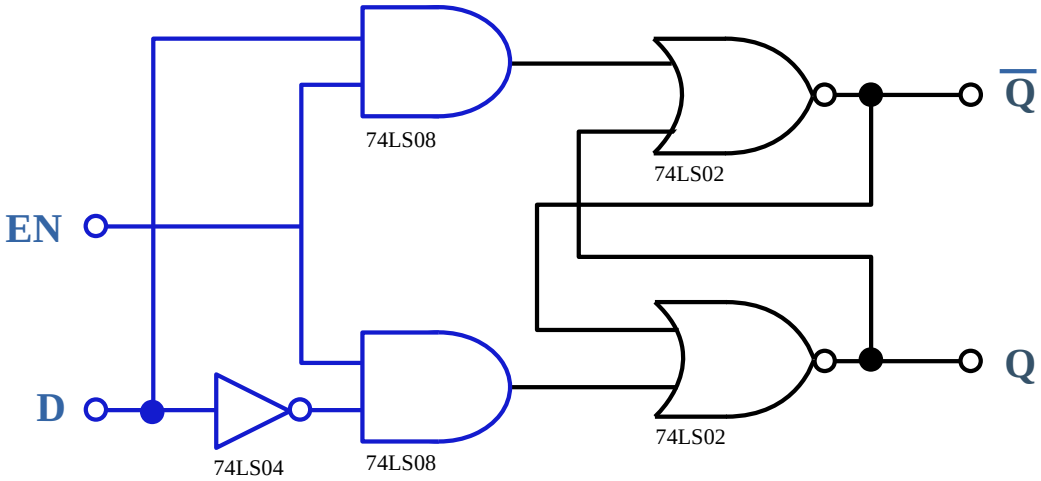



*Figure 55: Gated D-latch Flip-flop*

The D-latched flip-flop, illustrated in Figure 55, may be considered as a one-input synchronous SR latch. The design exploits the fact that, in the two active input combinations (01 and 10) of a gated SR latch, **R** is the complement of **S**. The low state of the enable signal produces the inactive "11" combination.

## *7.4 Exercise #7.1: D-latch flip-flop*

The objectives of this exercise are to:

1. Draw the D-latch flip-flop as illustrated in Figure 55. Select appropriate devices to generate a complete truth table for the device. Hint (use a logic toggle for the EN). Set either of the inputs to a 1 before starting the simulation. On the diagram make the text in the labels: 0.17 inch (4.3mm) in height, blue in and use bold effect.
2. Use logicprobes
3. Use logicstates
4. Use logictoggles

5. Describe the operation of the clocked SR flip- flop.

6. Simulate the circuits and obtain truth tables.

---

# Components:

- The logic gates can be found in the ***"TTL74LS series"*** library.
- The Logicprobe, Logictoggle & Logicstate symbols can be found in "***Debugging tools"***.

**Note:** Use the 74LSXX gates or the design may not simulate.

# Assignment

Build the D-latch flip-flop and complete a truth table for it.

## *7.5 JK flip-flop*

From the previous examples it can be seen that the basic gated SR flip-flop suffers from a major problem, the **S** = 1 and **R** = 1 condition, or **S** = **R** = 1, must always be avoided. To overcome this problem the JK flip-flop was developed.

The JK flip-Flop is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit. The sequential operation of the JK flip-flop is exactly the same as for the previous SR flip-flop with the same **S** and **R** inputs. The difference this time is that the JK flip-flop has no invalid or forbidden input states of the SR Latch (when **S** and **R** are both 1).

The **JK flip-flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs **S** and **R** are equal to logic level 1. Due to this additional clocked input, a JK flip-flop has four possible input combinations:

- High, logic 1,
- Low, logic 0,
- No change
- Toggle.

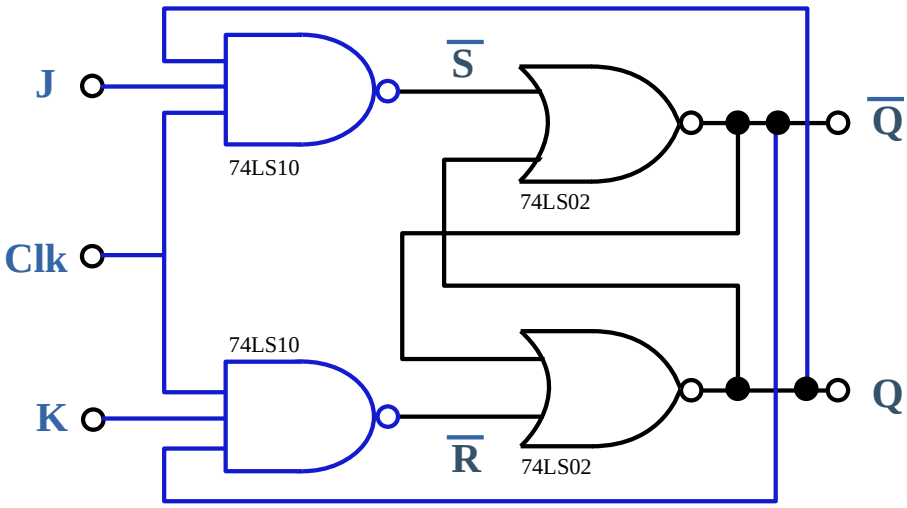



*Figure 56: JK flip-flop*

In summary

- **J** = **K** = 1 = Toggle
- **J** = 1, **K** = 0 = SET
- **J** = 0, **K** = 1 = RESET

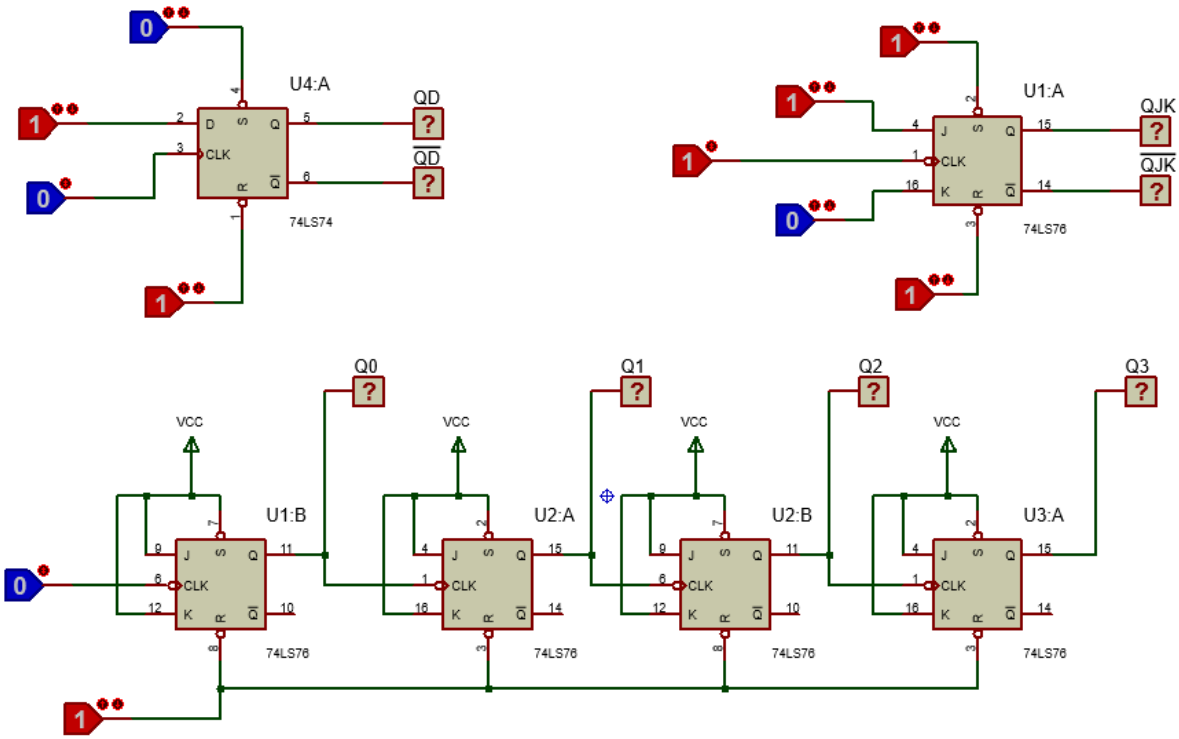



*Figure 57: Flip-flop and counter*

## *7.6 Exercise #7.2: JK flip-flops*

The objectives of this exercise are to:

1. Draw the JK flip-flop illustrated in Figure 57. Select appropriate devices to generate a complete truth table for the device.
   a) Logicprobes
   b) Logicstate
   c) Logictoggle
2. Describe the operation of the JK flip flop and a ripple counter.
3. Describe the difference between logicstate and logictoggle inputs.
4. Use the Block copy command.
5. Simulate the circuits and obtain a transition table.

---

# Components:

- The logic gates can be found in the ***"TTL74LS series"*** library
- The Logicprobe, Logictoggle & Logicstate symbols can be found in "***Debugging tools"***.

**Note:** Use the 74LSXX gates (e.g. 74LS86) or the design may not simulate.

# Assignments:

1. Draw the D Flip-flop and the JK flip-flop as shown in the top half of the diagram overleaf.

2. Simulate both flip flops
   a) Use the Logicstate inputs to apply logic values to the inputs.
   b) Use the Logictoggle input to generate a single clock pulse.
   c) Record the logic levels at the outputs as observed using the Logicprobes.
   d) Draw up a transition table for the flip-flops.

3. Draw the 4 bit ripple counter as shown in the bottom half of the diagram overleaf
   a) Simulate the ripple counter.
   b) Record the logic levels at the outputs as observed using the Logicprobes.

**Note**: It is important to ***save work frequently***.

# 8 Analogue Signals

## 8.1 *Instruments introduction*

Much like in physical networks, Proteus has a set of virtual instruments available for use. The instruments can be found by clicking the virtual instrument tab on the left hand side of the screen. 

To demonstrate these create a new Proteus project and draw the instruments as shown below.

***Save the schematic***.

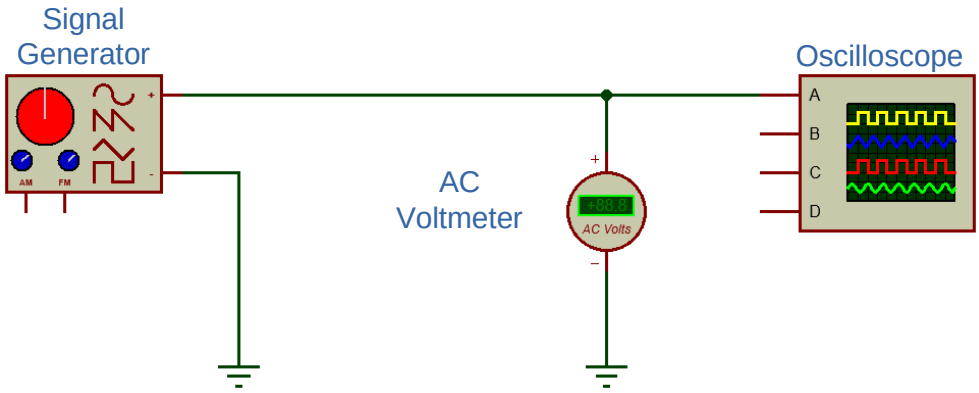



*Figure 58: Proteus project*

## 8.2 *Signal Generator*

When the play button, to simulate the circuit, is pressed a Signal Generator appears. The front panel of the Signal Generator, is illustrated in Figure 59.

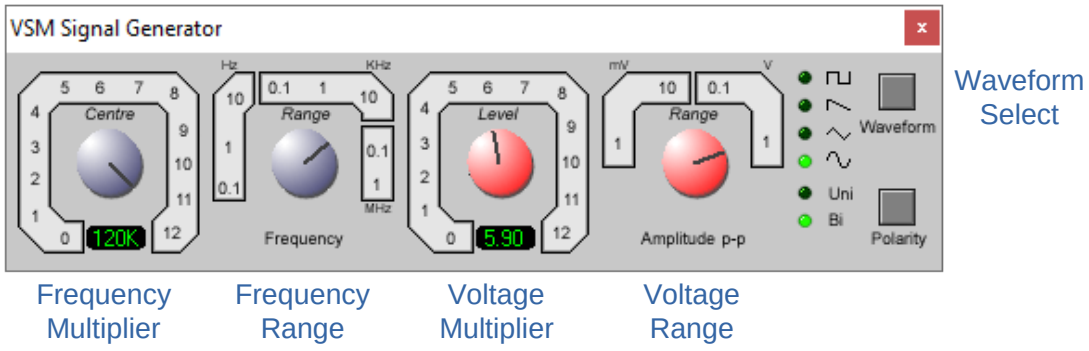



*Figure 59: Signal Generator*

With the mouse it is easy to change the settings on the signal generator, for demonstration purposes, adjust to the settings shown in Figure 59. This will generate a 6V peak to peak sinusoidal waveform which as a frequency of 120 kHz.

## *8.3 Oscilloscope*

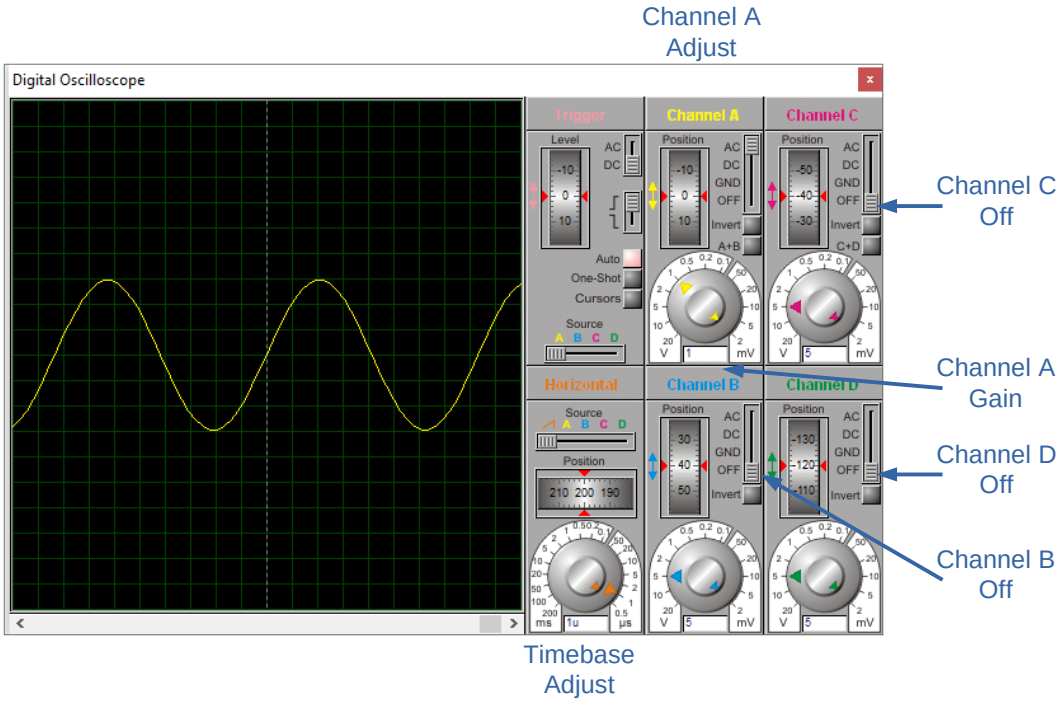


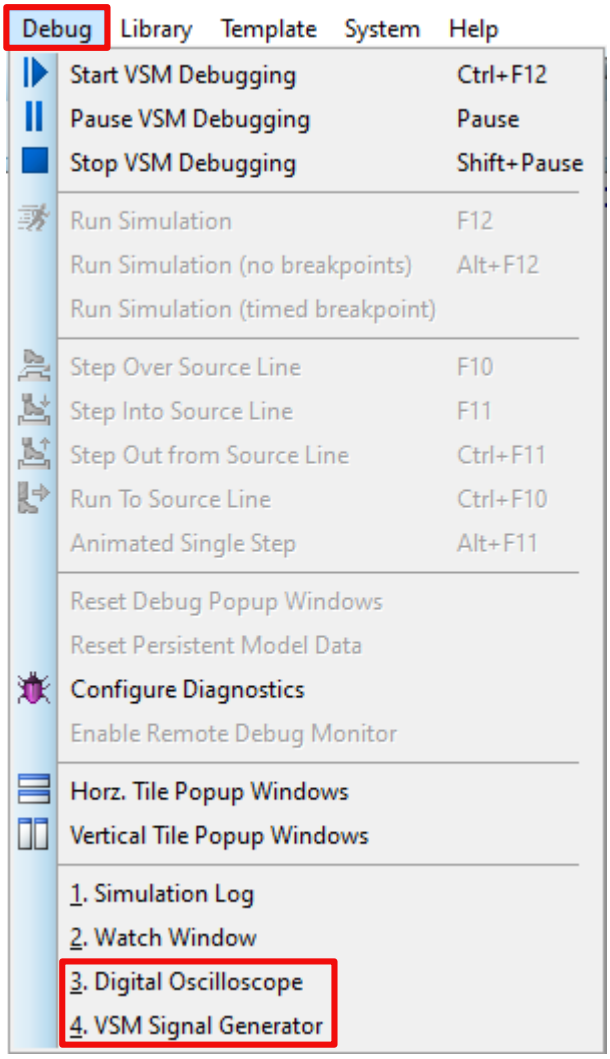

*Figure 60: Oscilloscope*

To read and visualise the generated signal an oscilloscope, as illustrated in Figure 60, will appear on the screen. Using the mouse the oscilloscope settings can be changed to match the generated signal. Adjusting as follows:

a) Use the mouse to turn off channels B, C & D.
b) Adjust the gain of Channel A to 1 volt per division.
c) Adjust the Horizontal time base to 1 µSec per division.
d) Position the A trace in the centre of the screen – by scrolling the Ch A position dial.

With the oscilloscope measure the signals generated by the Signal Generator:

1) Measure the amplitude of the waveform displayed on the oscilloscope. Record the voltage measured by the AC voltmeter. Explain why the voltages measured are different?
2) Measure the frequency of the waveform displayed on the oscilloscope. Is it as expected?
3) Experiment with the Signal generator. Change the type of waveform. Change the amplitude and frequency of the waveform.
4) Stop the simulation.




**Note**: If the Signal Generator or the Oscilloscope are closed during a simulation, they can be found under the debug menu and brought back.

## *8.4 Exercise #8: Analogue Signals*

The objectives of this exercise are to:

1. Create a schematic diagram of an analogue circuit which uses the following components and instruments.
   a) Oscilloscope
   b) Signal generator
   c) Operational amplifier

2. Simulate circuits which include an operational amplifier.
3. Describe the basic operation of an inverting and non-inverting op-amp circuit using the virtual instruments described above.

---

# Components:

- The op-amp used in this exercise is the UA741 and can be found in the "***Operational amplifier***" category of components.

# Procedure: Simulation of Op-amp circuits

1. Add the inverting and non-inverting op-amp circuits.
   a) Label the Inverting and non-inverting amplifiers.
   b) From the component values shown calculate the gain of both amplifiers

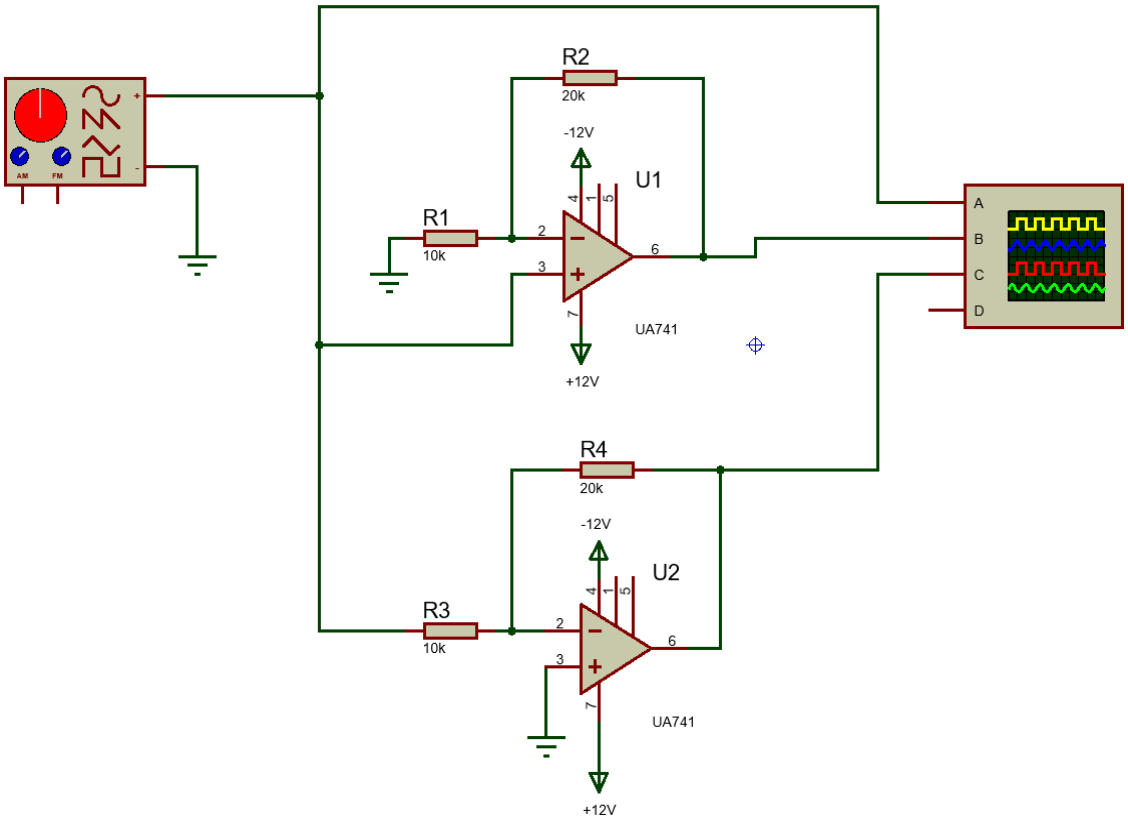



*Figure 61: Op-amp circuit*

2. Simulate the circuit.
   a) Adjust the signal generator so that a 12 kHz signal with an amplitude of 6V peak to peak is generated.
   b) Adjust the gain of Channels B & C to 1 volt per division.
   c) Adjust the Horizontal time base to 10 µSec per division.
   d) Position the B & C trace in the centre of the screen – by scrolling the position dials.
   e) Use the oscilloscope to measure the gain of the inverting and non-inverting amplifiers.
   f) Do the measured results agree with the calculated results?
   g) Adjust the signal generator so that the amplitude of the signal is 12V peak to peak, explain what happens. Adjustment of the A, B & C channel gain is necessary.

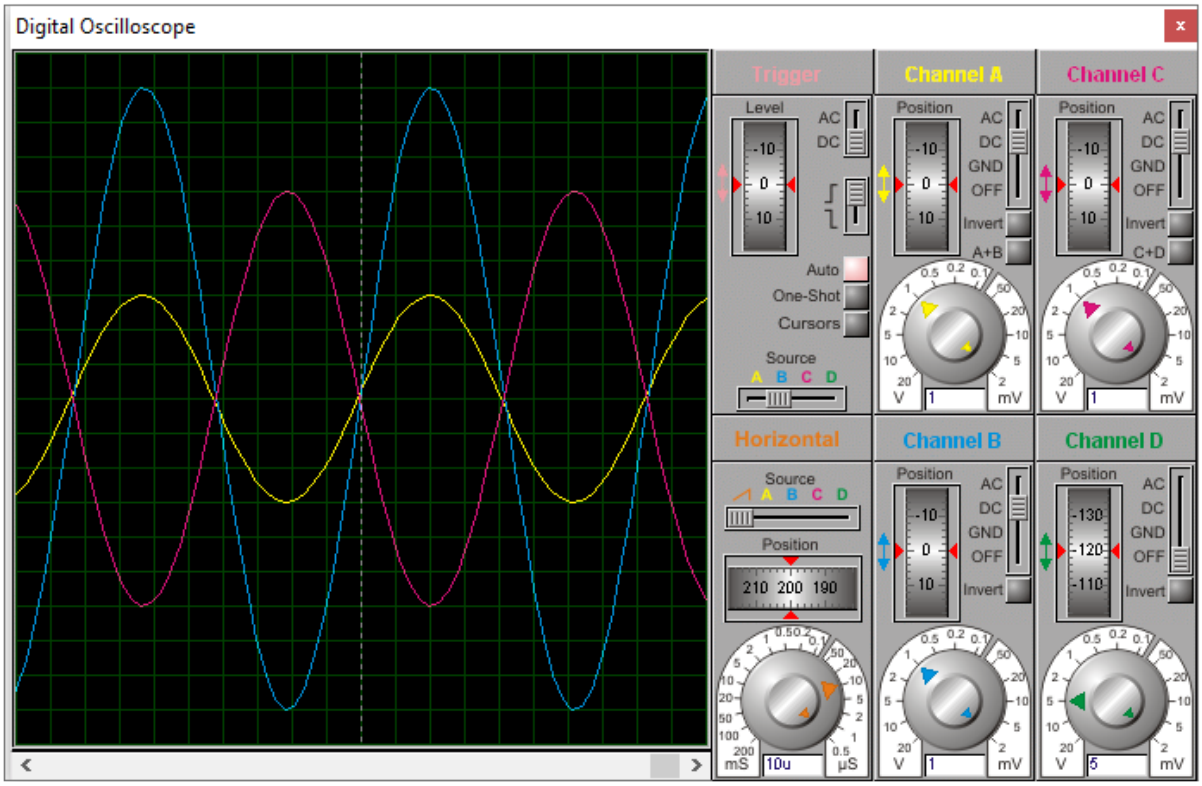



*Figure 62: Monitoring op-amp circuit response to signal*

# 9 Graphs

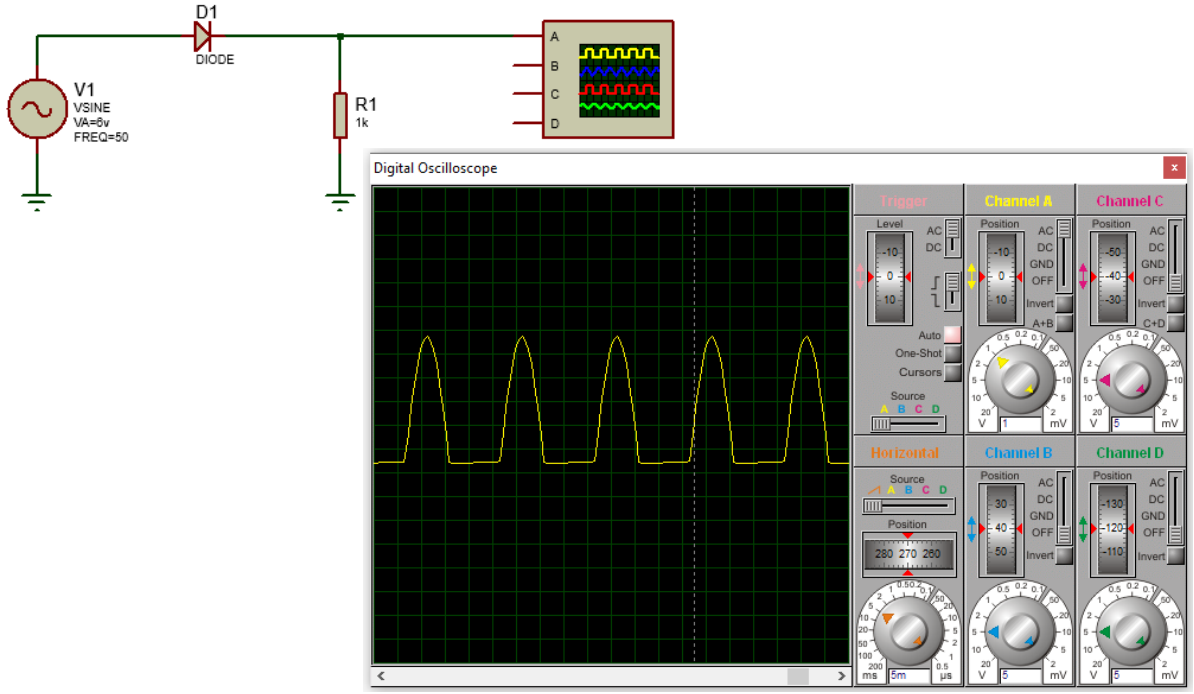



*Figure 63: Half-wave Rectifier circuit with Oscilloscope*

## 9.1 *Analogue introduction*

Consider the half-wave rectifier circuit in Figure 63. For each cycle of the sine wave the negative values are blocked by the diode resulting in the output demonstrated in the oscilloscope view.

## 9.2 *Graphs introduction*

Another useful tool in Proteus is the probe and from the probe analogue analysis can be performed and output as graphs.

From the *Probe Mode*  add a VOLTAGE probe to the schematic as illustrated in Figure 64.

From the *Graph Mode*  add an ANALOGUE Graph, also illustrated in Figure 64.

## 9.3 *Linking Probe to the Graph*

Right-clicking on the Analogue Graph and using the selection ***Add Traces***, provides a pane to link the probe to the graph. In the example Probe **P1: ROOT_R1(1)** are linked. It is worth noting that up to four probes can be linked to the same graph.

## 9.4 *Simulate Graph*

Again, right-clicking on the Analogue Graph and using the selection ***Simulate Graph*** to trigger the graph.

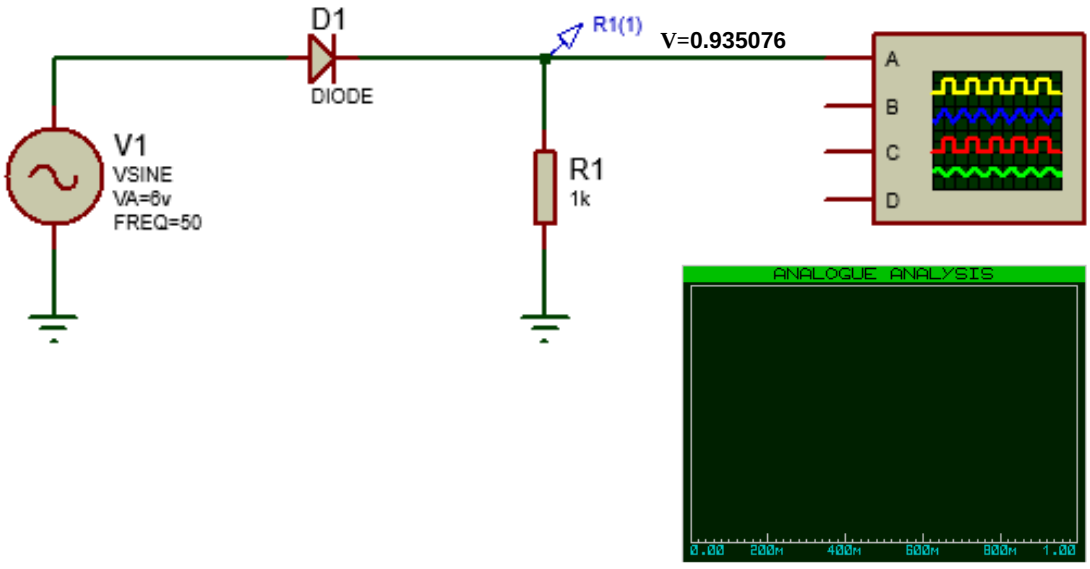



*Figure 64: Add Probe and Analogue Analysis*

## 9.5 *Graph granularity*

One problem that may occur at this stage is a flooded graph. This is down to the granularity of the X-axis. As illustrated in Figure 65 this can be addressed by Editing the properties of the Analogue Analysis graph and changing the ***Stop Time*** value. In the example given the value is changed to 100 milli-seconds (100ms).

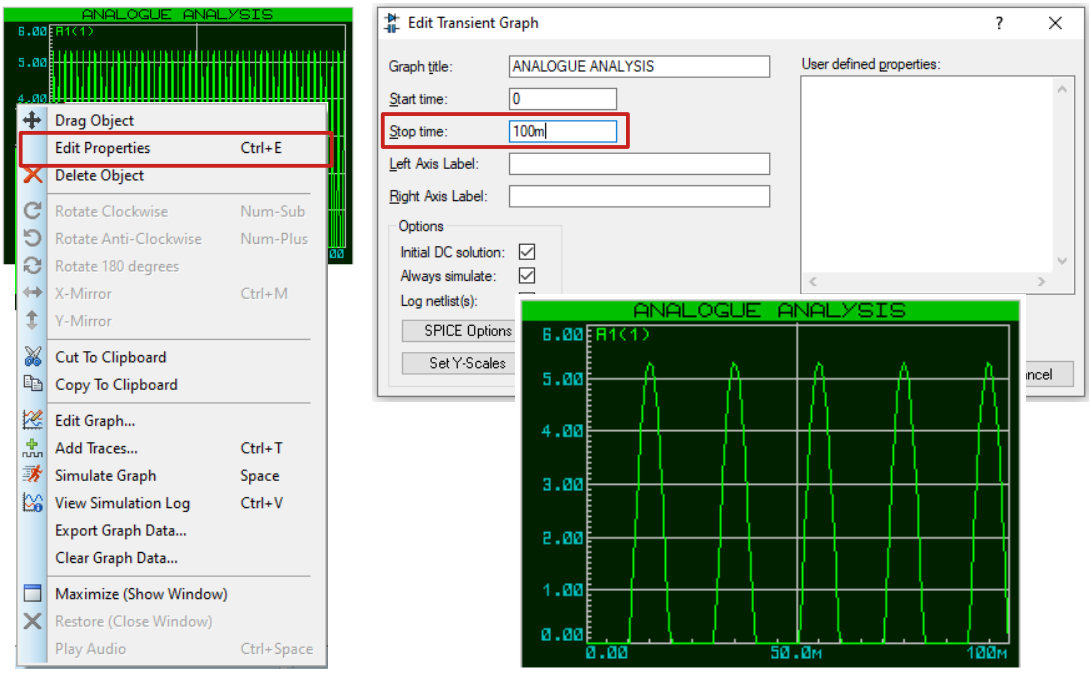



*Figure 65: Graph Granularity*

***Save the schematic***.

## *9.6 Export data to a spreadsheet*

It is often useful to extract the data from the simulation for additional analysis in a spreadsheet for example. Right-clicking on the Analogue Analysis graph and selecting ***Export Graph Data…*** presents the opportunity to save the data to the computer as a **.DAT** file. This file is a Comma Separated Value (CSV) delimited text file that is easily imported into a spreadsheet. From their the data can be analysed as demonstrated in Figure 66.

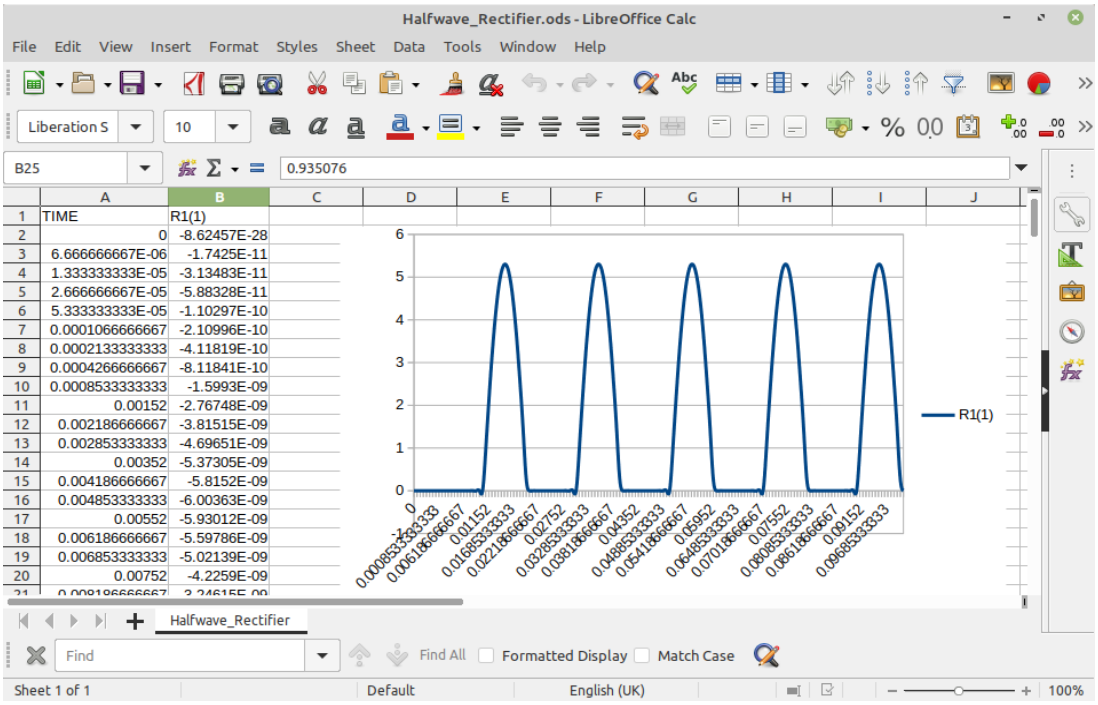



*Figure 66: .DAT file imported into a LibreOffice Calc spreadsheet*

## *9.7 Exercise #9.1: Analogue Graphs*

The objectives of this exercise are to:

1. Design and use a master sheet.
2. Create a schematic diagram of an analogue circuit which uses the following components and instruments:
   a) Sine wave AC voltage source.
   b) Half wave rectifier.
   c) Bridge rectifier.
   d) Load resistor.
   e) Smoothing capacitor.
3. Have a basic understanding of how to use the virtual instruments described above.
4. Describe the basic operation of a half and full wave rectifier.
5. Describe the effect of changing the value of the smoothing capacitor.

---

# Components:

- All the components can be found in the device library including the sine wave AC voltage source.

**Procedure: Part A**: this section focuses on familiarity with the setup of a Master Sheet.

1. Download a suitable Electronic Engineering image from the Web and save it to the home drive.
2. Create a new Proteus project.
3. Under ‘Template’, select ‘Goto Master Sheet’.
4. Modify the Header block:
   a) Delete the path.
   b) Delete the file name.
   c) Move the design title.
   d) Set the Authors name to ‘Your Name’.
5. Under ‘File’, select ‘Import Image’. Select the image downloaded from the Web. Place the image on the worksheet and resize it.
6. Under ‘Template’, select ‘Save design as Template’, select a suitable name for the new template. If the new template is saved in a directory other than the default, Proteus will need to be given the path to this directory. This is achieved by selecting ‘System’ then ‘System settings’ and adding a new path to the template folders.
7. Now select ‘New Project’. Choose a suitable Name and Path. Select ‘Next’. Select ‘Create a schematic from the selected template’. Select the new template created.
8. Continue as before and the master sheet will open.

## **Procedure: Part B**: this section simulates a half-wave rectifier.

1. Create the half-wave rectifier circuit shown in Figure 67.
   a) Set the voltage to $6v_P$ and the frequency to 50Hz.
   b) Set the load to 1KΩ.
   c) Select an analogue graph and graph the half wave rectified voltage across the load.
   d) Are the shape, frequency and amplitude as expected?
   e) Press the spacebar key to redraw the graph at any time.
   f) Now add the capacitor and graph the voltage across the load.
   g) Change C to 10µF. Graph the voltage across the load. Explain the shape of the graph?
   h) Describe any TWO methods of reducing the amount of Ripple on the output of a Half Wave Rectifier.
   i) Verify results with an Oscilloscope.

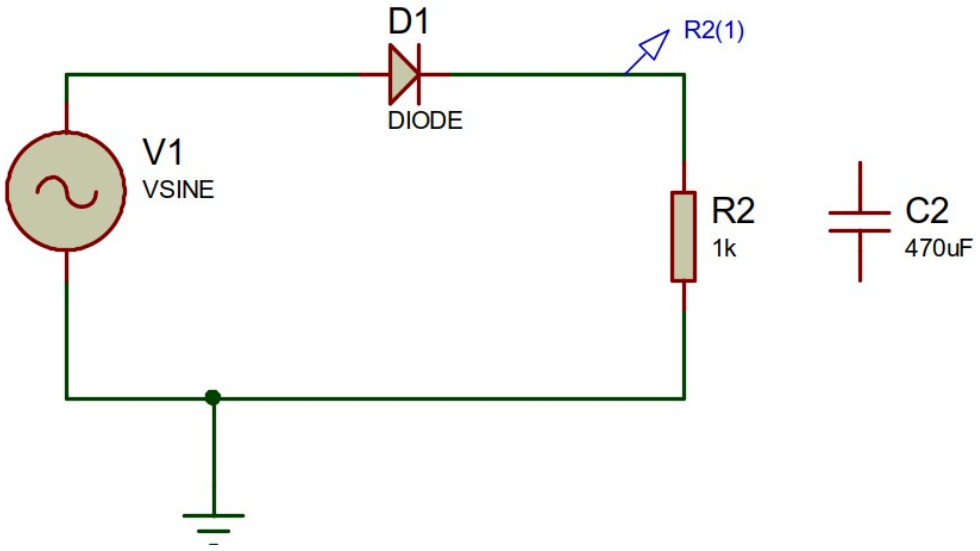



*Figure 67: Half-wave Rectifier*

**Procedure: Part C**: this section simulates a full-wave rectifier.

2. Create the full-wave rectifier circuit shown in Figure 68.
   a) Set the voltage to $6v_P$ and the frequency to 50Hz.
   b) Set the load to 1KΩ.
   c) Select an analogue graph and graph the full wave rectified voltage across the load.
   d) Are the shape, frequency and amplitude as expected?
   e) Now add the capacitor and graph the voltage across the load.
   f) Change C to 10µF. Graph the voltage across the load. Explain the shape of the graph?
   g) Describe any TWO methods of reducing the amount of Ripple on the output of a Full Wave Rectifier.
   h) Verify results with an Oscilloscope.

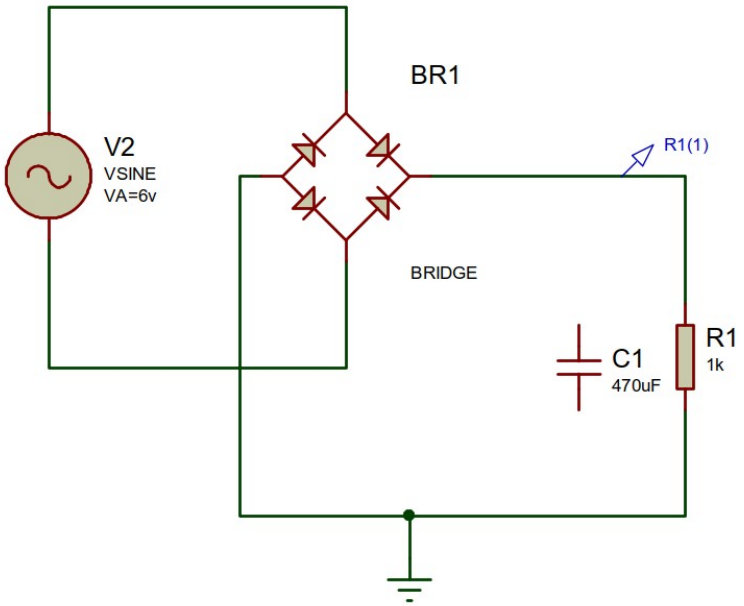



*Figure 68: Full-wave Rectifier*

## *9.8 Digital introduction*

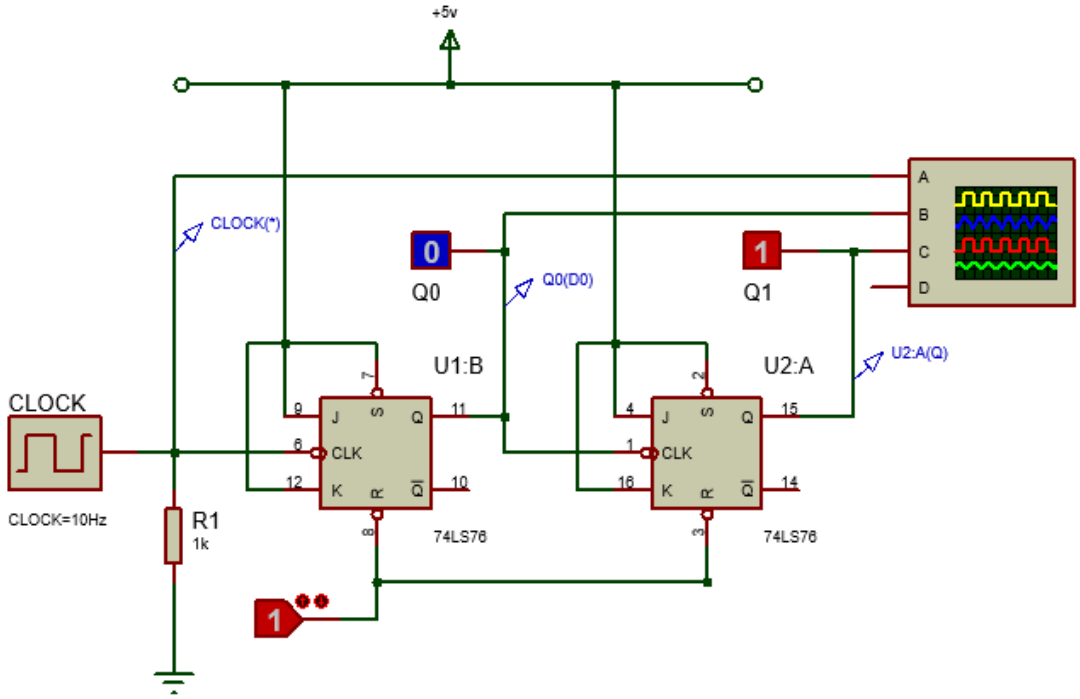



*Figure 69: Clocked, dual JK flip-flop*

Consider the clocked, dual JK flip-flop circuit in Figure 69. Monitoring this in the oscilloscope is not that helpful, particularly when observing signal edges to identify what triggers what. Adding digital probes to the clock, as illustrated in Figure 70, Q0 and Q1 and aligning outputs in a Digital Analysis graph can prove useful indeed.

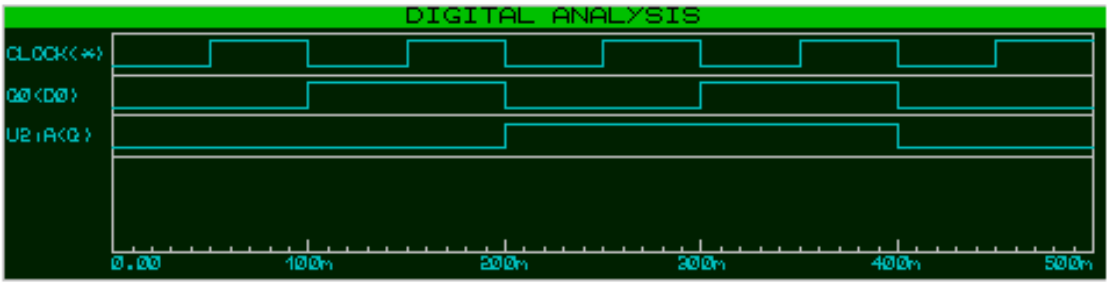



*Figure 70: Clocked, dual flip-flop digital analysis*

## 9.9 *Digital Analysis*

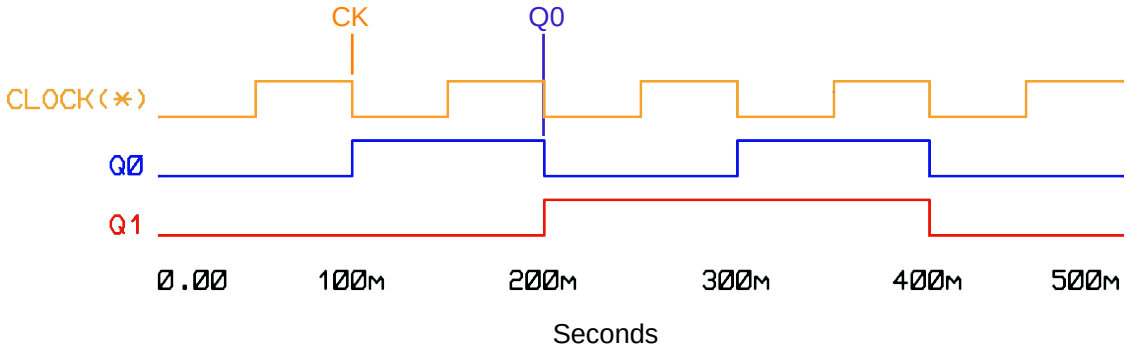



*Figure 71: Digital analysis*

Referring to Figure 71, Consider the impact of each transition of the CLOCK from logic 1 to logic 0. Each of these cause a toggle of the next stage, Q0. Now consider each transition of Q0 from logic 1 to logic 0. Each of these cause a toggle of the next stage, Q1. In summary, each signal in this circuit is the result of a chance in state of the previous signal and the state change that has effect is the transition from HIGH to LOW in each case.

## *9.10 Exercise #9.2: Digital Graphs*

The objectives of this exercise are to:

6. Design and use a master sheet.
7. Create a schematic diagram of an analogue circuit which uses the following components and instruments:
   a) Square wave clock voltage source.
   b) JK flip-flop.
   c) Oscilloscope.
   d) Login state.
   e) Logic probe.
   f) Digital probe.
8. Have a basic understanding of how to use the virtual instruments described above.
9. Describe the basic operation of a JK flip-flop counter.

---

# Components:

- All the components can be found in the device library including the CLOCK voltage source.

# Procedure:

1. Create a new Proteus project.
2. Create the counter with four JK flip-flops as shown in Figure 57.

- Replace the CLK LOGIC STATE with a 20Hz CLOCK source.
- Connect Q0, Q1, Q2 and Q3 to channels A, B, C and D on an Oscilloscope.
- Verify results with an Oscilloscope.
- Are the shape, frequency and amplitude as expected?

3. Add voltage probes to the CLOCK output and to Q0, Q1, Q2 and Q3.
   - Create a Digital Analysis graph and drag the probes in order to the graph.
   - Press the spacebar or right-click and select ***Simulate Graph*** to update.
   - Is this what you expected to see, if so explain why.

This page is intentionally blank