Virtuoso component Standard Library working Details 1.0.0

Virtuoso Library Component

The Virtuoso Component Library is a comprehensive and versatile collection of pre-designed elements, modules, and assets that facilitate the streamlined development of complex projects. This library serves as a repository of ready-made building blocks, enabling designers, developers, and engineers to expedite their work and maintain a consistent design language throughout their projects.

The Virtuoso Component Library encompasses a wide array of components spanning various domains, such as user interface (UI) components, software modules, hardware modules, and more. These components are carefully crafted, following best practices and design guidelines, ensuring that they are not only visually appealing but also functionally robust.

Each component within the Virtuoso library comes with detailed documentation and usage instructions, allowing users to seamlessly integrate them into their projects. This documentation provides insights into the component's purpose, available customization options, input/output requirements, and any dependencies.

The library's components are designed with modularity and reusability in mind, fostering an environment where developers can efficiently assemble different pieces to create a cohesive whole. This approach not only accelerates the development process but also enhances collaboration among team members by establishing a common language and understanding of the components.

The Virtuoso Component Library also frequently undergoes updates and expansions, incorporating new components based on emerging technologies and design trends. This ensures that developers have access to contemporary solutions that align with the latest industry standards.

In conclusion, the Virtuoso Component Library stands as an invaluable resource for professionals across various disciplines, offering a collection of meticulously crafted components that streamline the development process, enhance project consistency, and contribute to the overall efficiency of complex projects.

ANALOG GAUGE

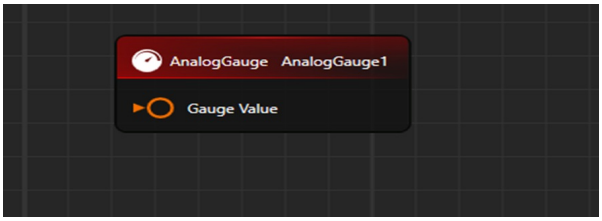
1. Analog Gauge

Newly implemented Feature in 1.x

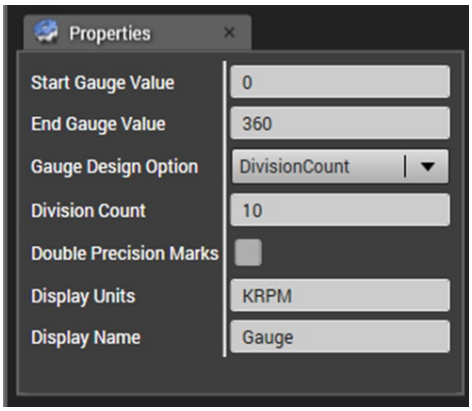
(A) Different type of Tick values scales

Case 1:-Default setup

(A)Default Node Style



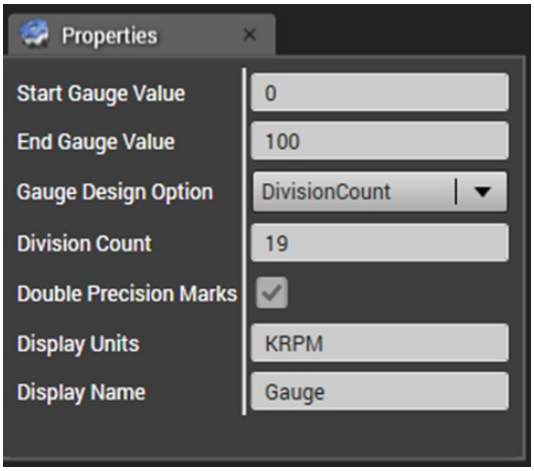
(B) Default Property window



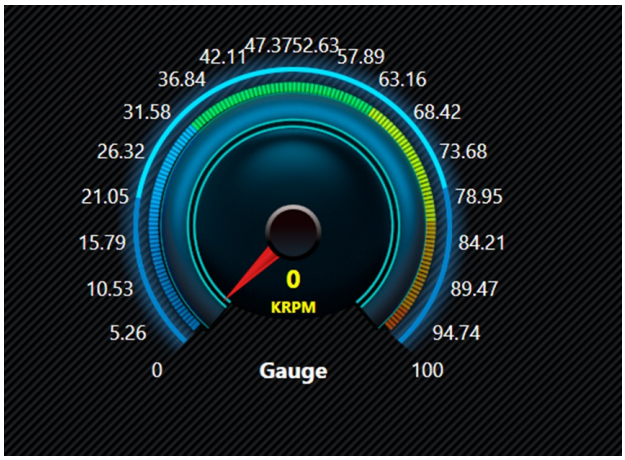
Initial Properties meaning are as follows:-

Start Gauge Value:-It is used to define the initial scale value
End gauge Value: - It is used to define the final scale value.
Gauge Design Option:-It is a dropdown that is used to set the scale in three different ways. These options are “List”, “Dictionary” and “Division Count”.
Double Precision Marks:- If it is checked then scale values will have decimal form otherwise Integer form.

(I) Default Property window

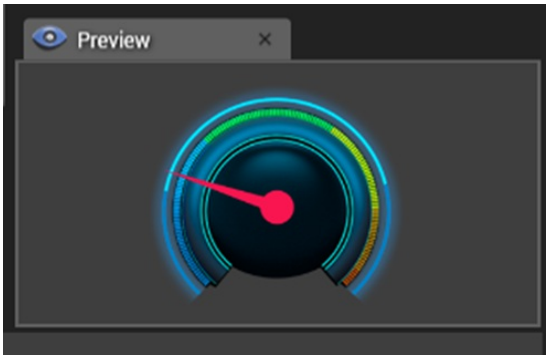


(II) View in Host

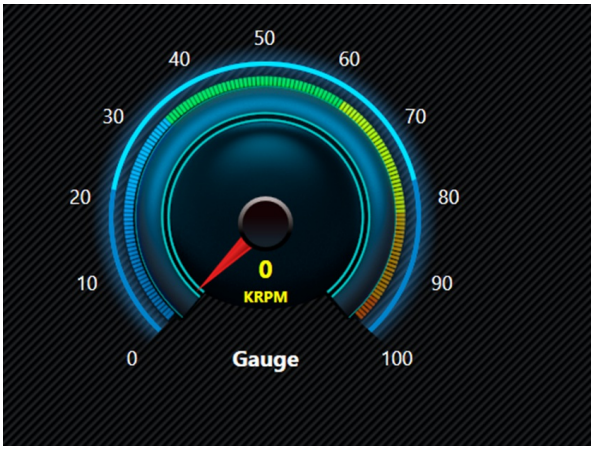


Display Unit:- It represent Unit of scale.
Display Name:- it can be used to define Name of the control.

(C) Preview Window



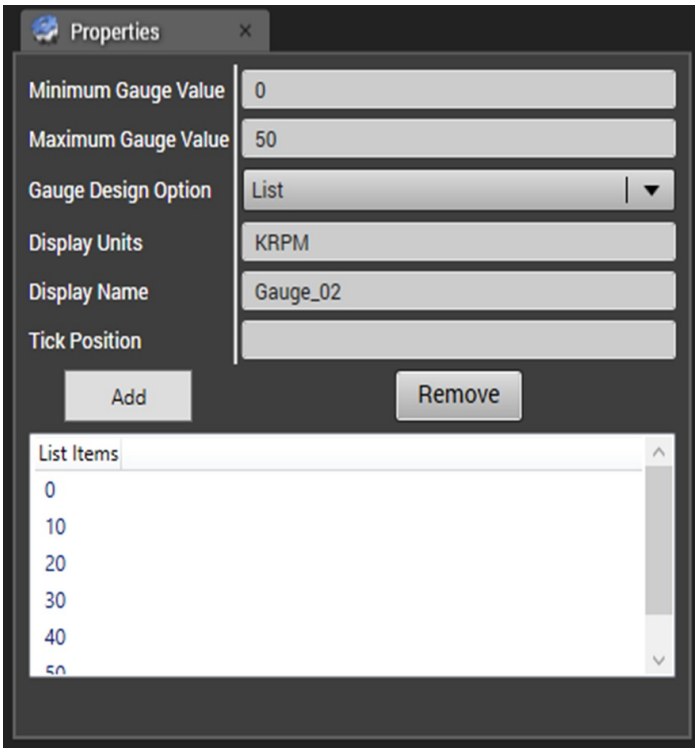
(D) Default View on HOST



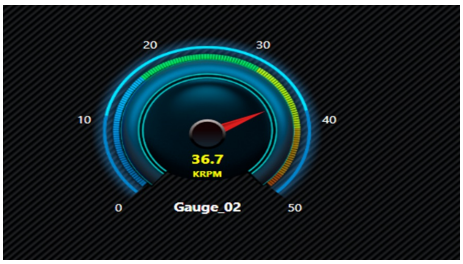
Case 2:-Changing "Gauge Design Option" from dropdown

Case (I) when selected option from "Gauge Design Option" dropdown is "List"

(A) Property Window

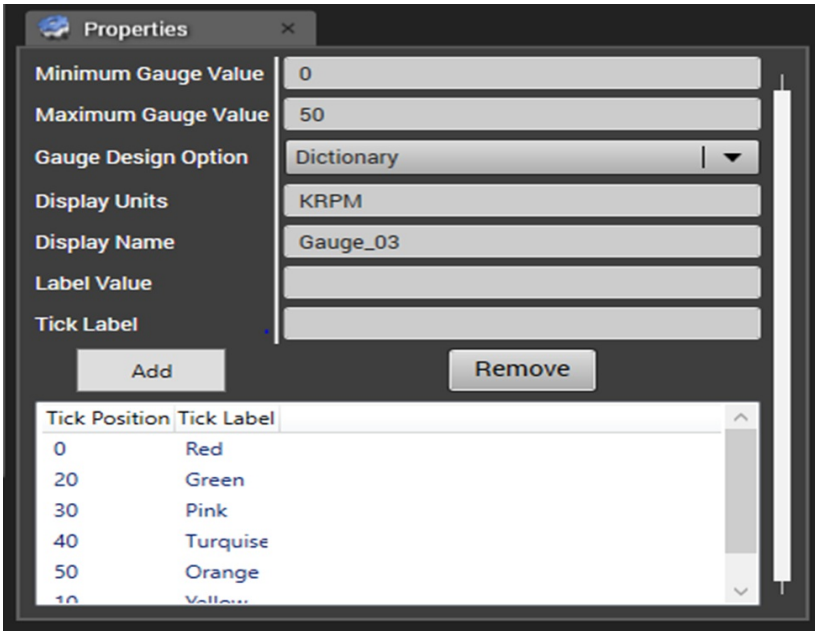


(B) View on HOST

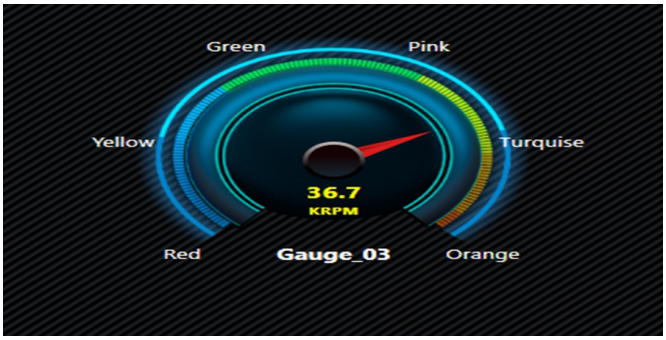


Case (II) when selected option from "Gauge Design Option" dropdown is "Dictionary"

(A) Property Window



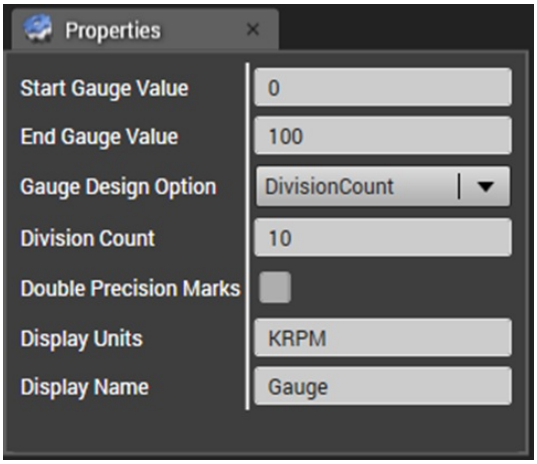
(B) View on HOST



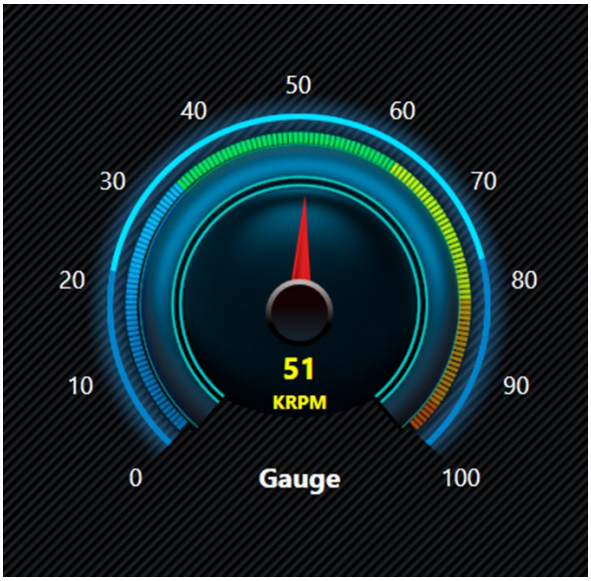
Case 3:-Different Tick values scale

(A) When Tick value scale like(0 To 100)

(1) Property window

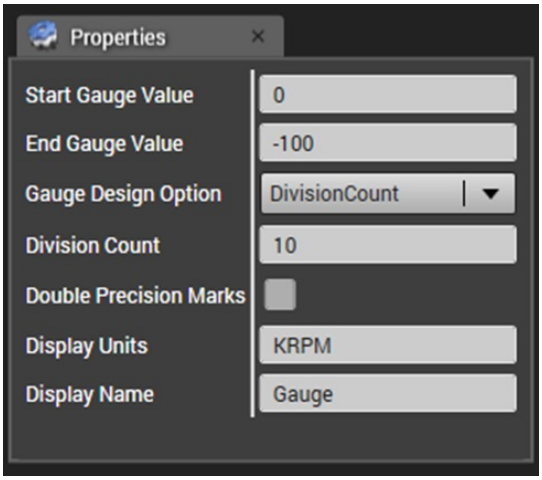


(2) View in HOST

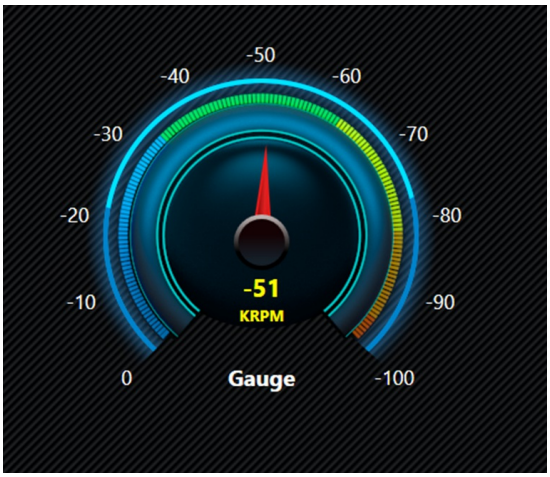


(B When Tick value scale like(0 To -100)

(1) Property window

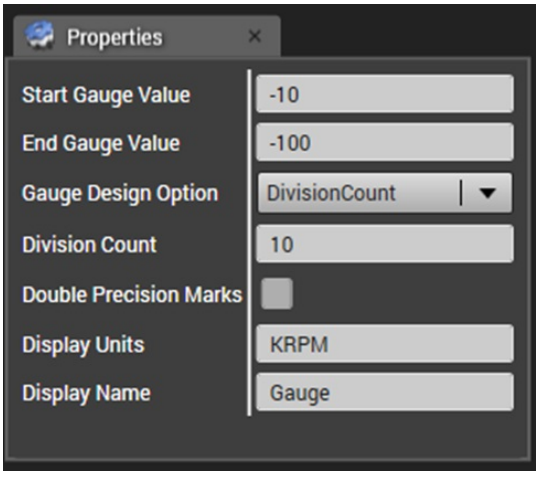


(2) View in HOST



(C) When Tick value scale like(-10 To -100)

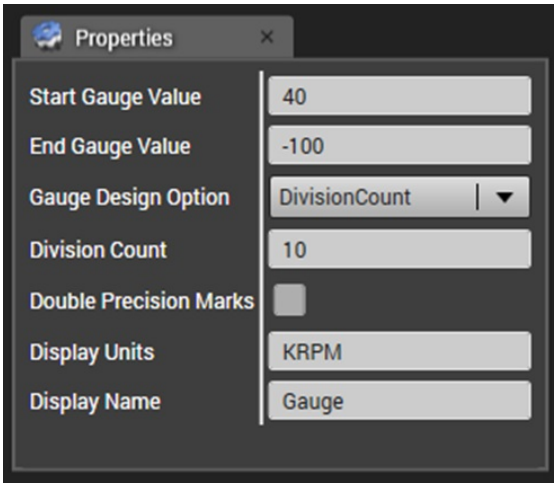
(1) Property window



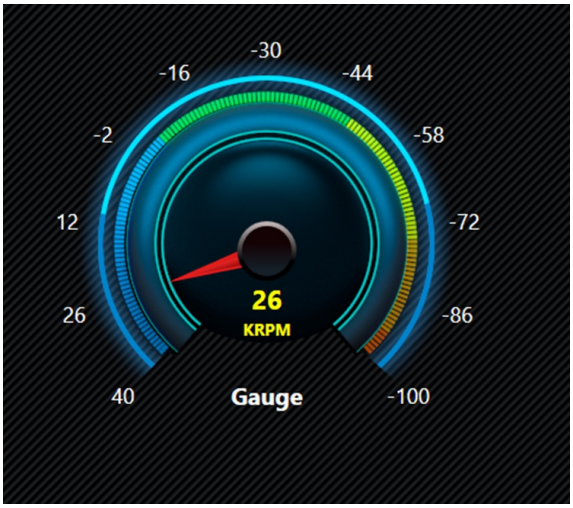
(2) View in HOST

(D) When Tick value scale like(40 To -100)

(1)Property window

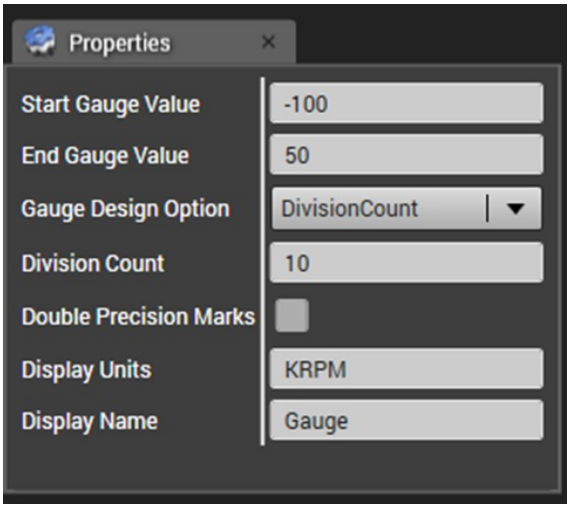


(2) View in HOST

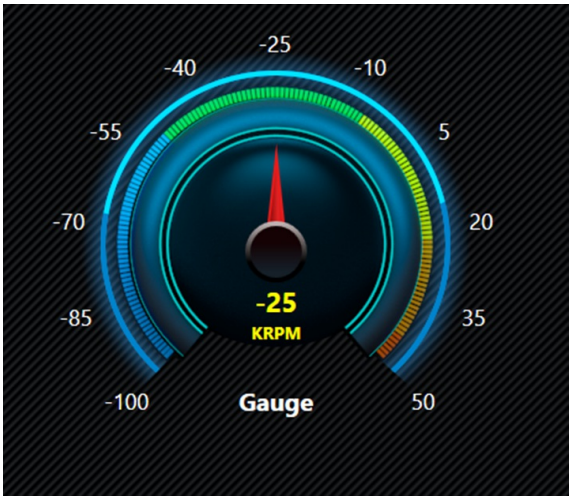


(E) When Tick value scale like(-100 To 50)

(1) Property window

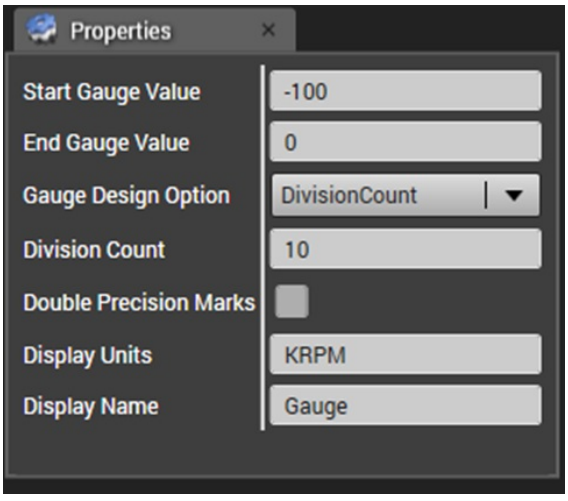


(2) View win HOST

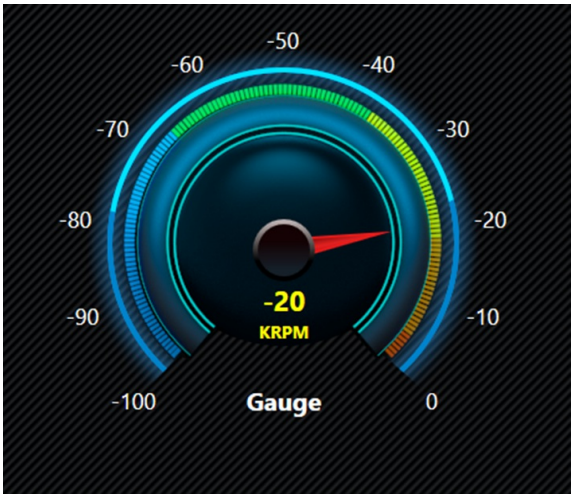


(F) When Tick value scale like(-100 To 0)

(1) Property window

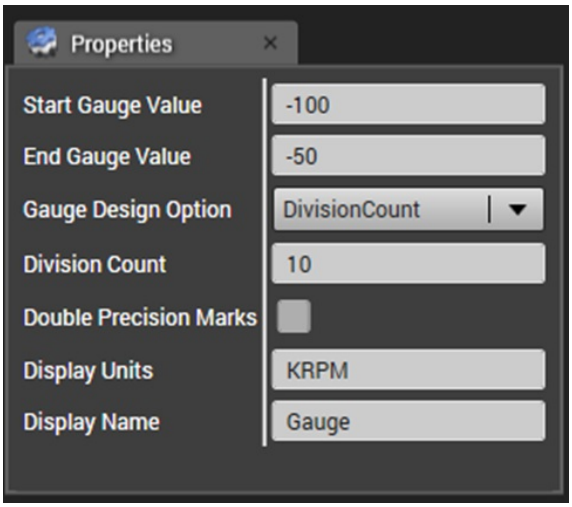


(2) View in HOST

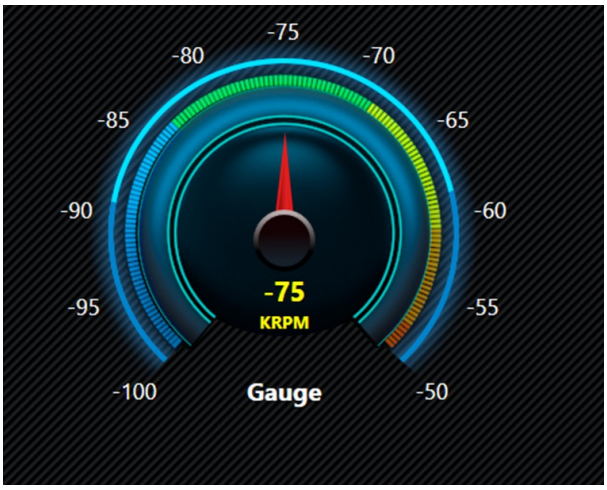


(G) When Tick value scale like(-100 To -50)

(1) Property window



(2) View in HOST



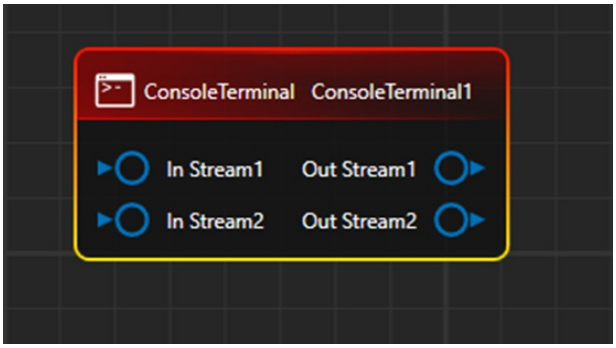
Note:-It also persist the state on HOST end. It will also maintain the last state of itself whether it is connected with any control or not. if in the initial stage, it is connected with any control and again if the same control is disconnected, still it will maintain its last value.

CONSOLE TERMINAL

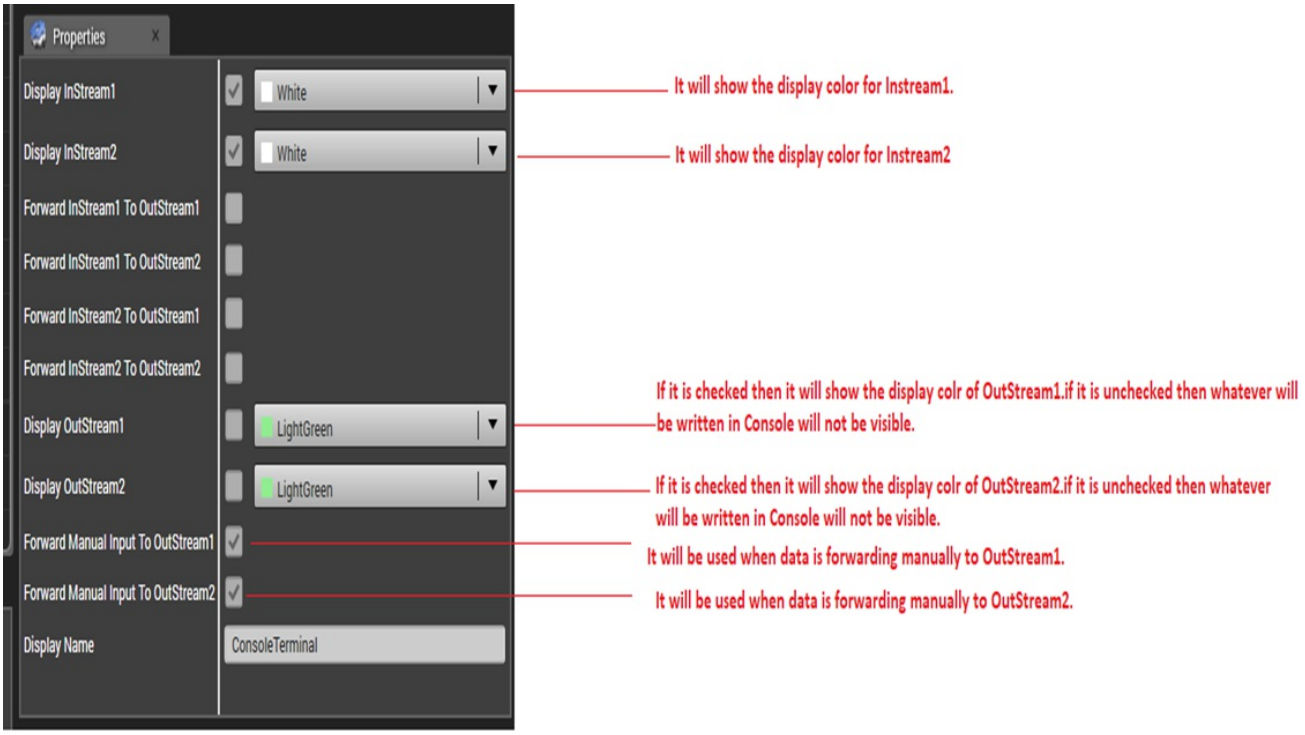
2. CONSOLE TERMINAL

Case 1:-Default setup

(A) Default Node Style

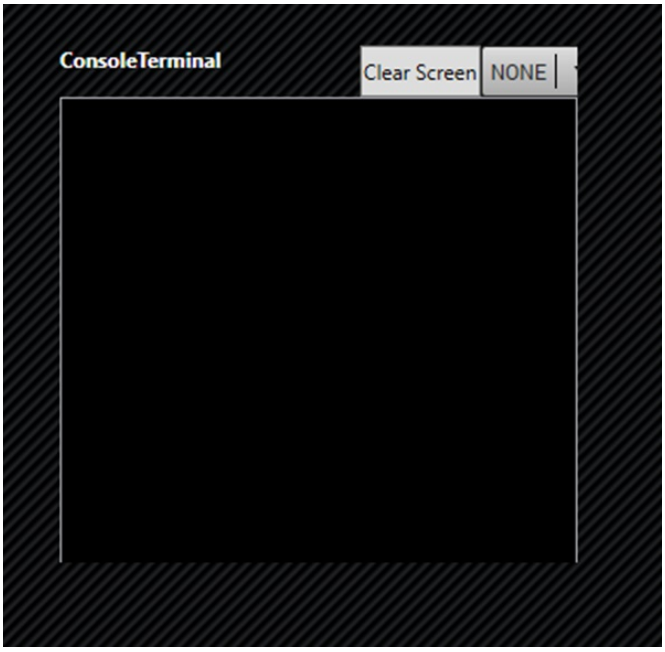


(B) Default Property window



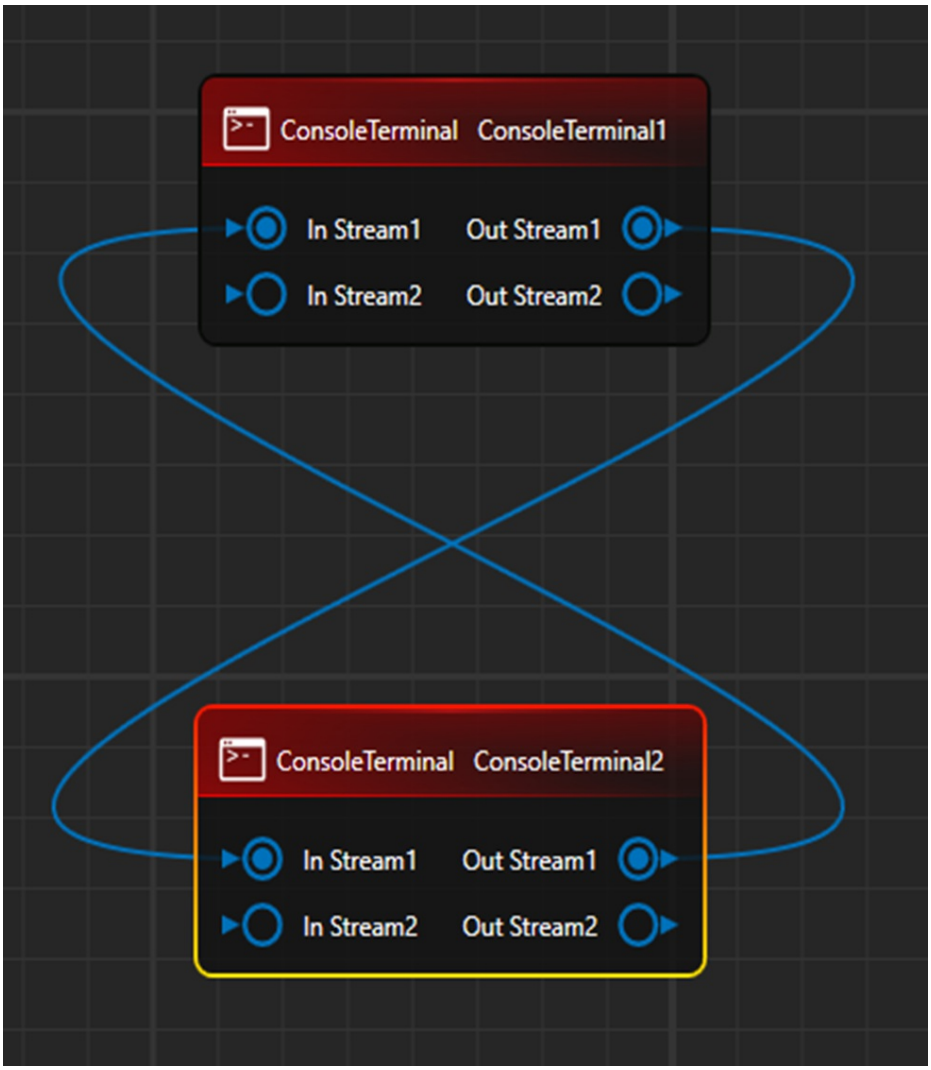
(C) Preview Window

(D) Default view in HOST



Case 2:-Console Terminal working process

(A) It will work when its input and out ports are connected with another console terminal input output ports.Ex-Default setup

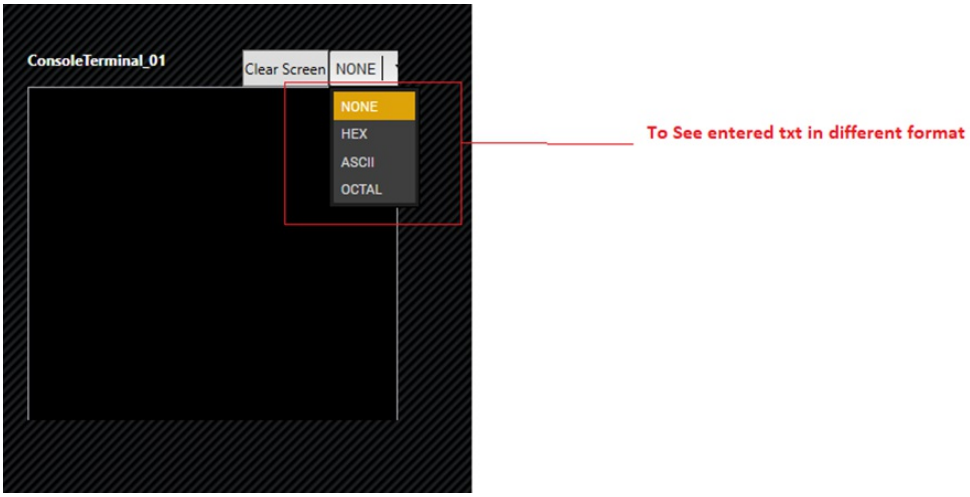


(B) View in HOST

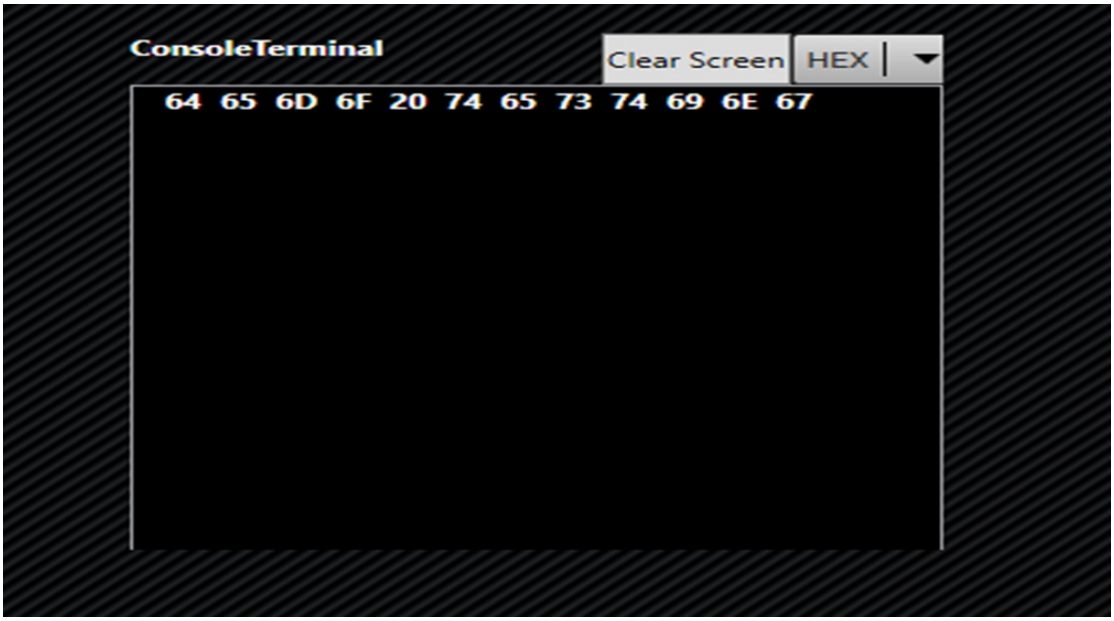
As according to the above connection if something is written in "Consoleterminal_01", it can be seen on "ConsoleTerminal_02" or Vice-Versa.

Feature to see values in different format

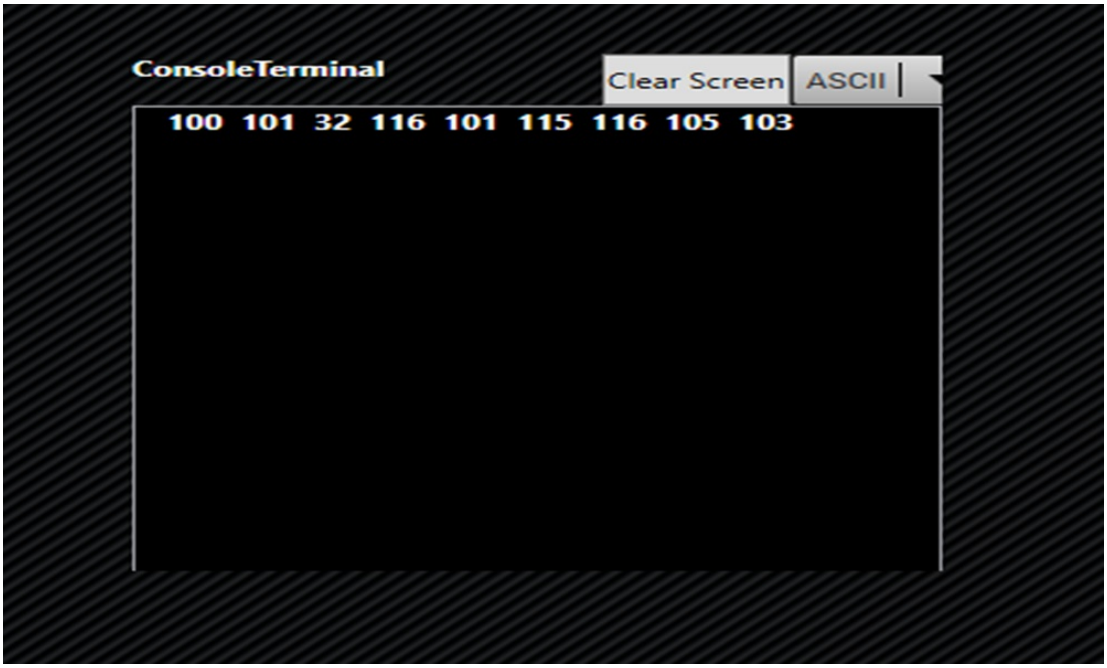
Default View



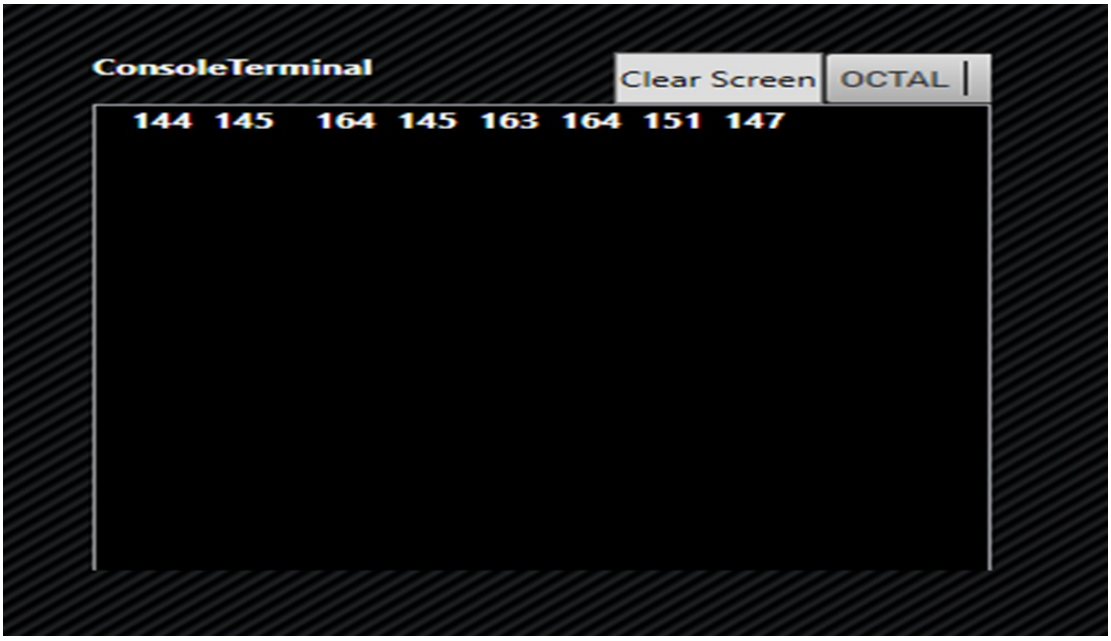
(A) In HEX



(B) In ASCII



(C) In OCTAL

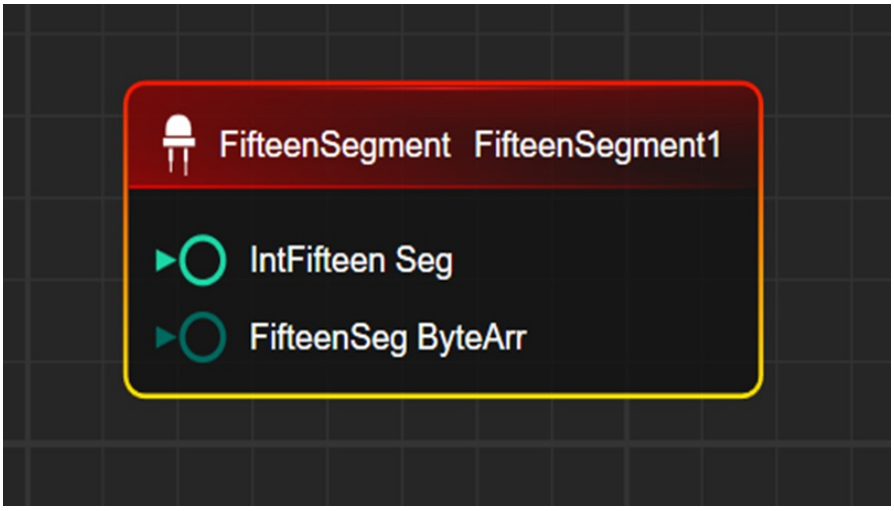


FIFTEEN SEGMENT CONTROL

3. FIFTEEN SEGMENT CONTROL

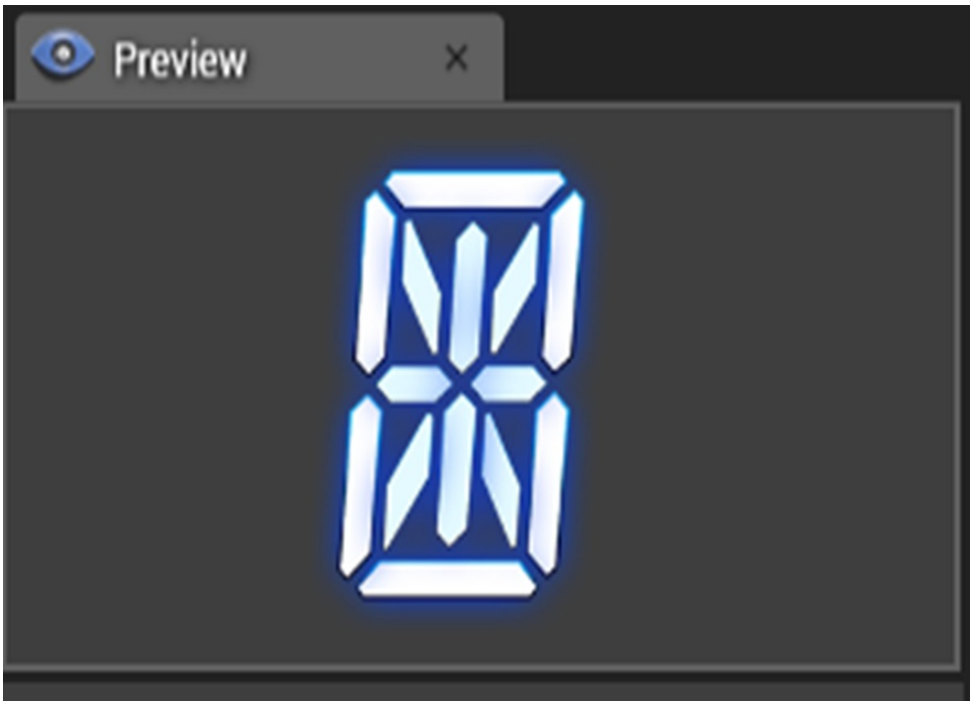
Case 1:-"when Display Input Config" is "ASCII from DisplayByte"

(A) Default Node Style



(B) Default property window

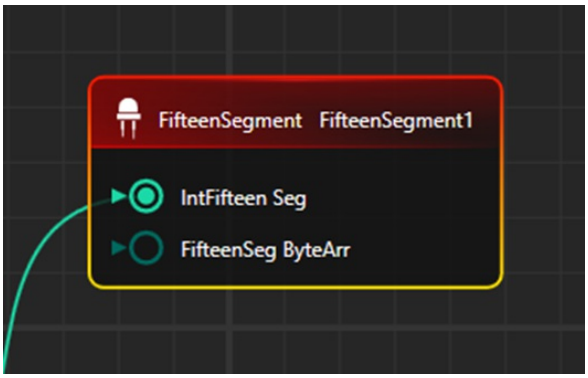
(C) Default Preview Window



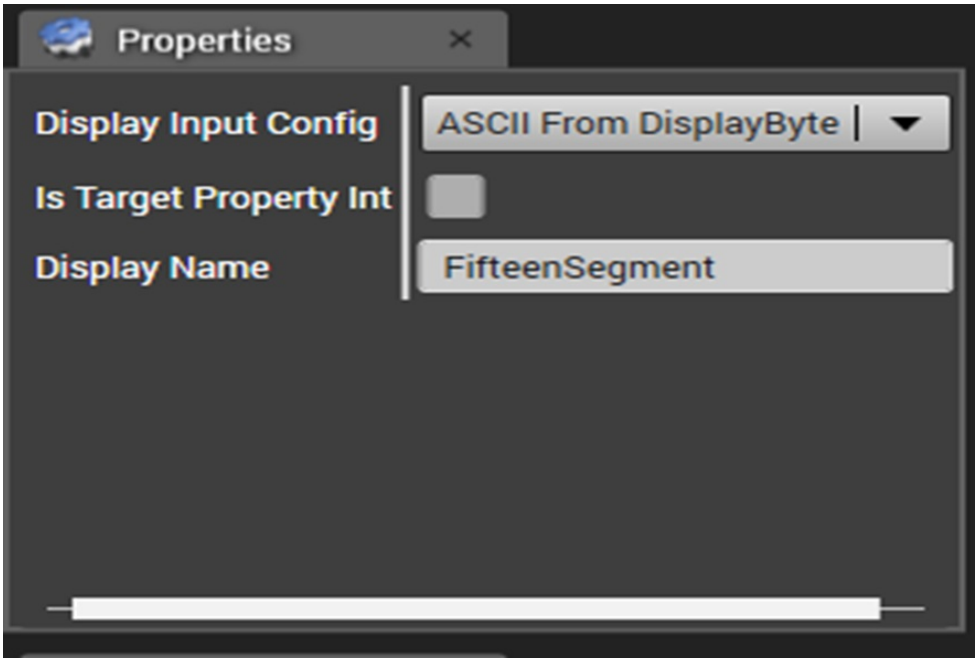
(D) Default View in HOST

Case 2:-When “Is Target Property Int” is not selected in property window, then sending value should be in Byte format.

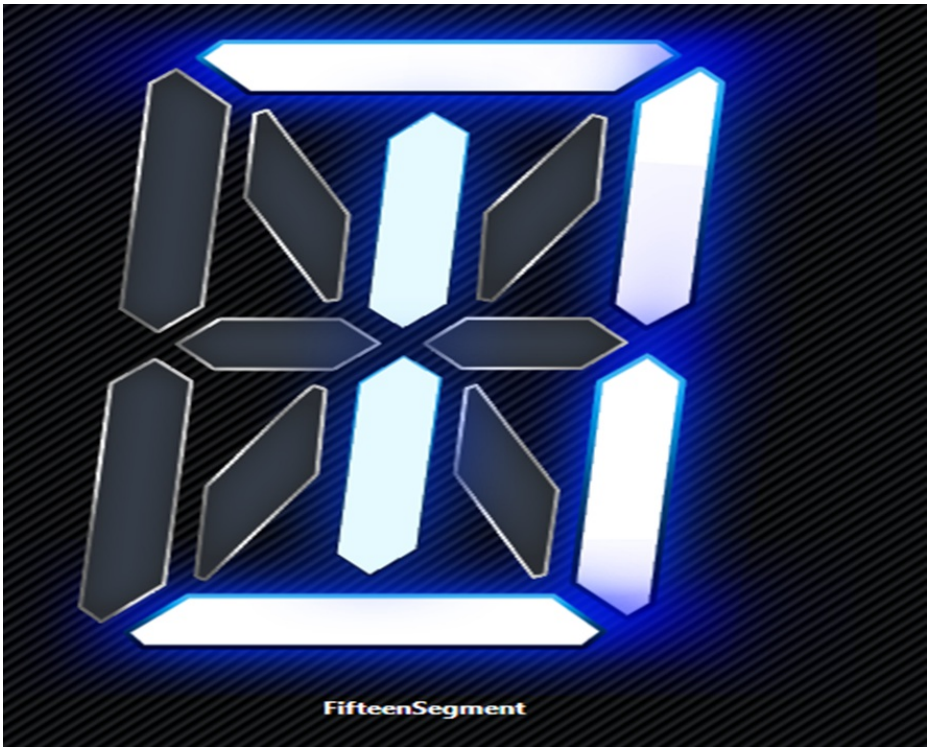
(A) Node Style

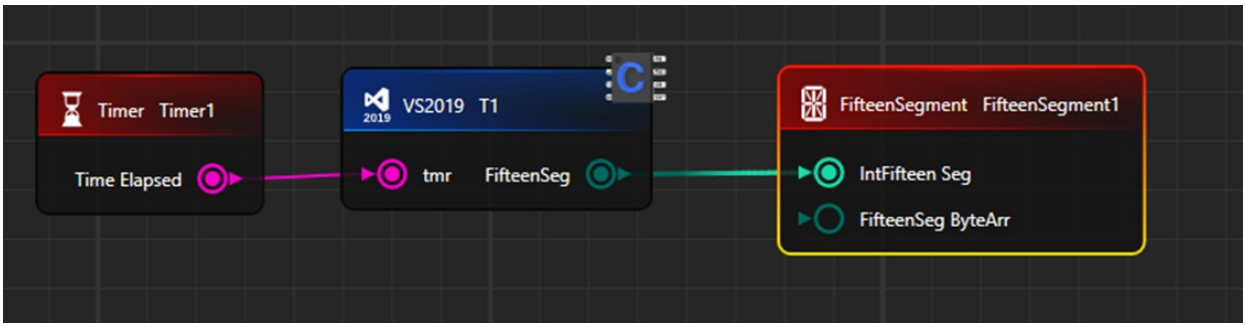


(B) Property Window



(C) View in HOST



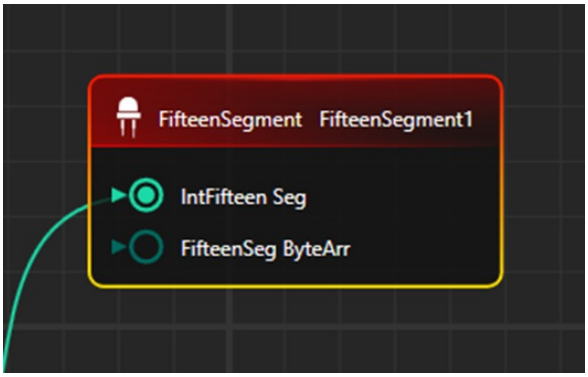


EX:-It will be defined in TMB

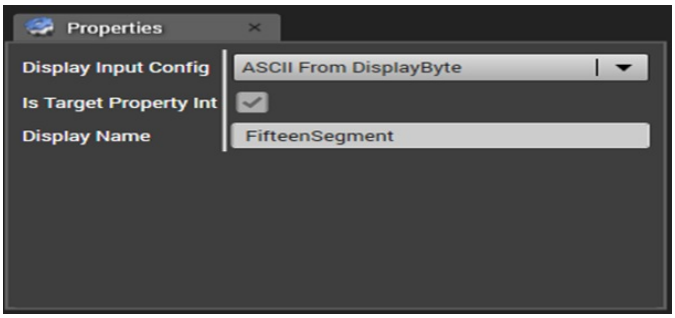
```
unsigned char FifteenSeg = 48;
void tmr()
{

    if (FifteenSeg >= 90)
    {
        FifteenSeg = 48;
    }
    FifteenSeg++;
}
```

Case 3:-When "Is Target Property Int" is selected in property window, then sending value should be Integer

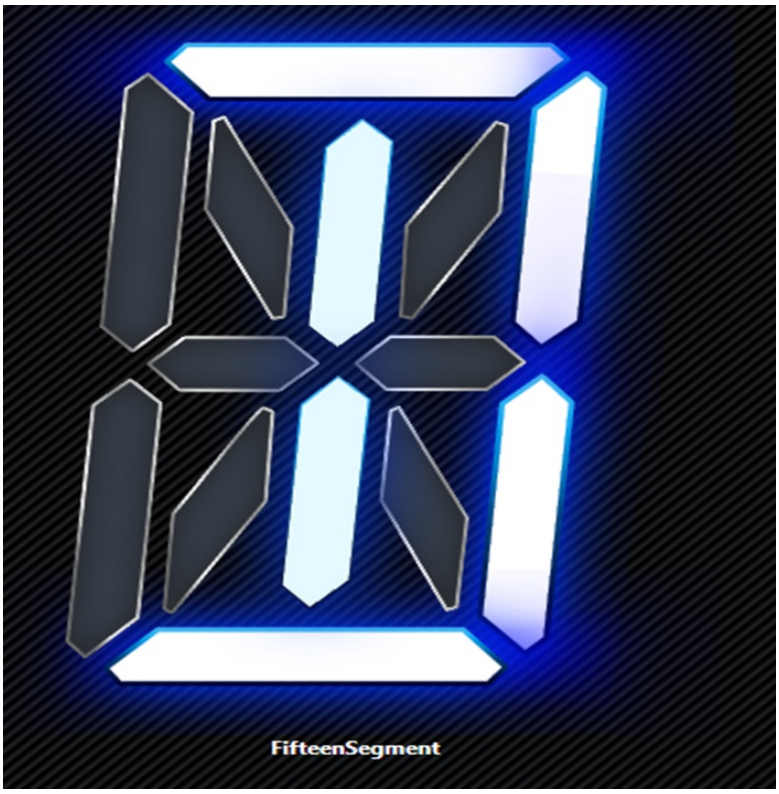
(A) Node Style



(B) Property window



(C) View In HOST

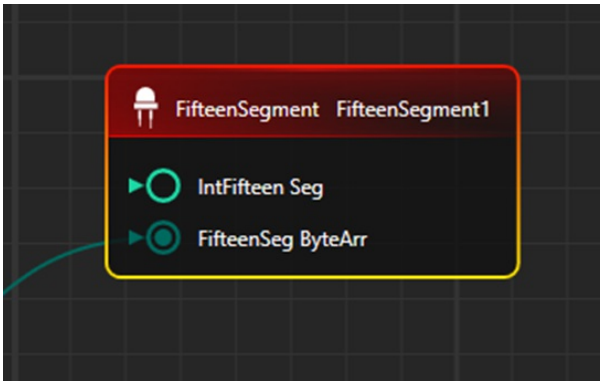


Ex-: It will be defined in TMB

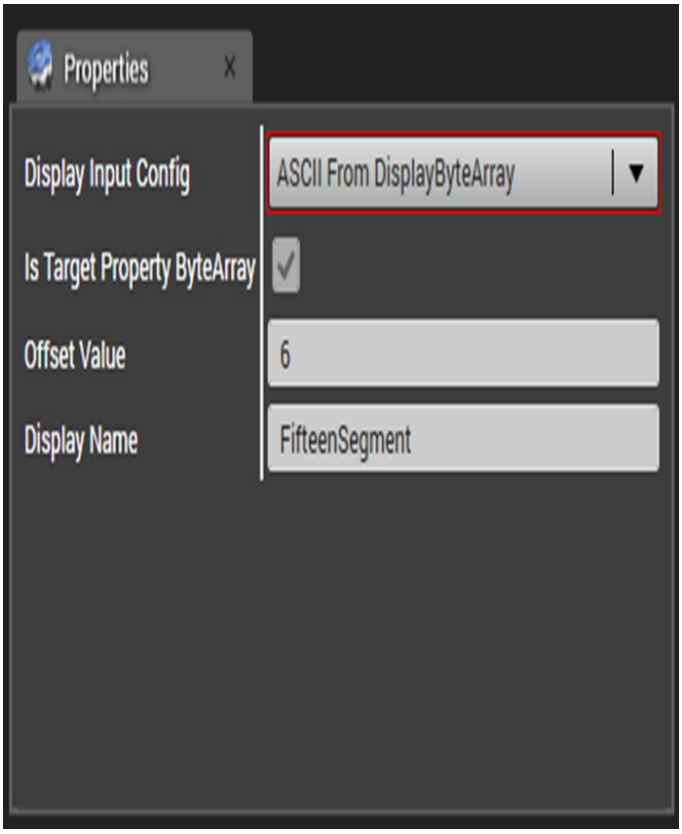
```
unsigned char FifteenSeg = 48;
void tmr()
{

    if (FifteenSeg >= 90)
    {
        FifteenSeg = 48;
    }
    FifteenSeg++;
}
```

Case 4:-"when Display Input Config" is "ASCII from DisplayByteArray"

(A) Node Style



(B) Property Window



Note:Need to provide array from "TMB"

```
unsigned char FifteenSegArr[] = {
      68,69,70,71,72,73,74,75

};
```

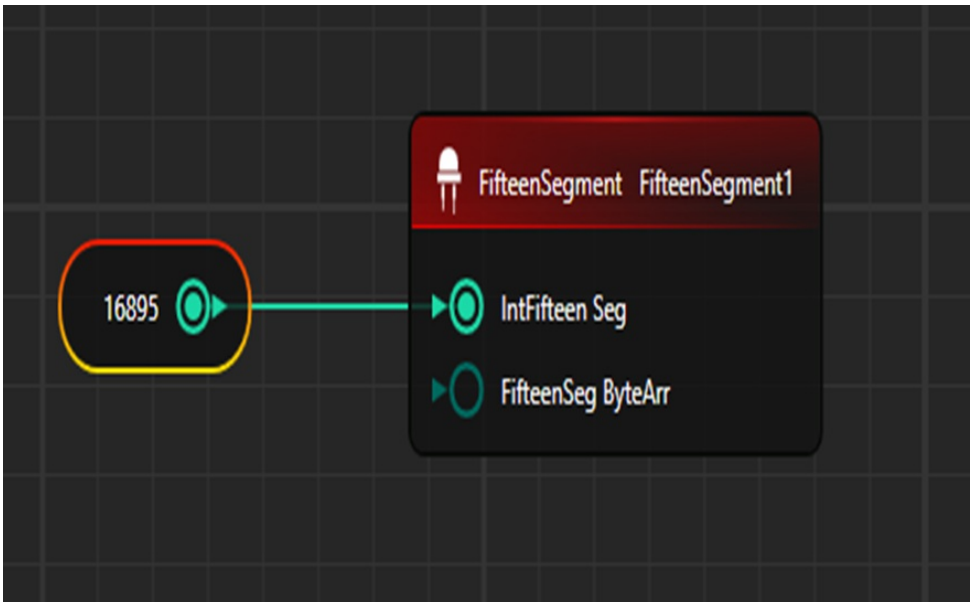
(A) View in HOST

Case 5:-"when Display Input Config" is "Raw from UnsignedShort"

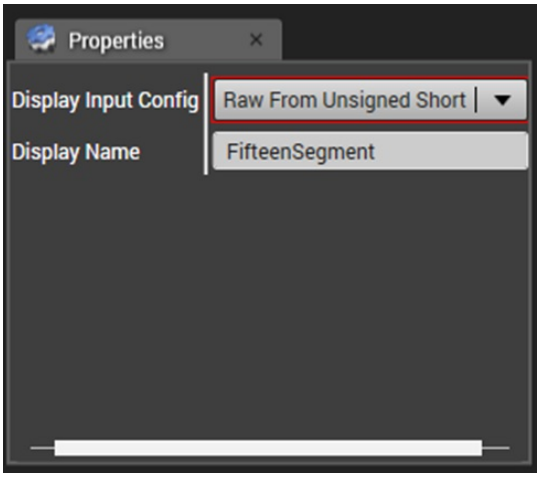
Value In binary: 100 0001 1111 1111

Value in DEC for binary value in "USHORT": 16,895

(A) Node Style



(B) Property Window



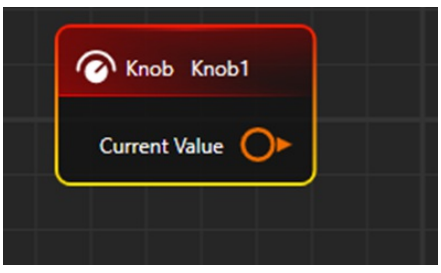
(C) View in HOST

KNOB CONTROL

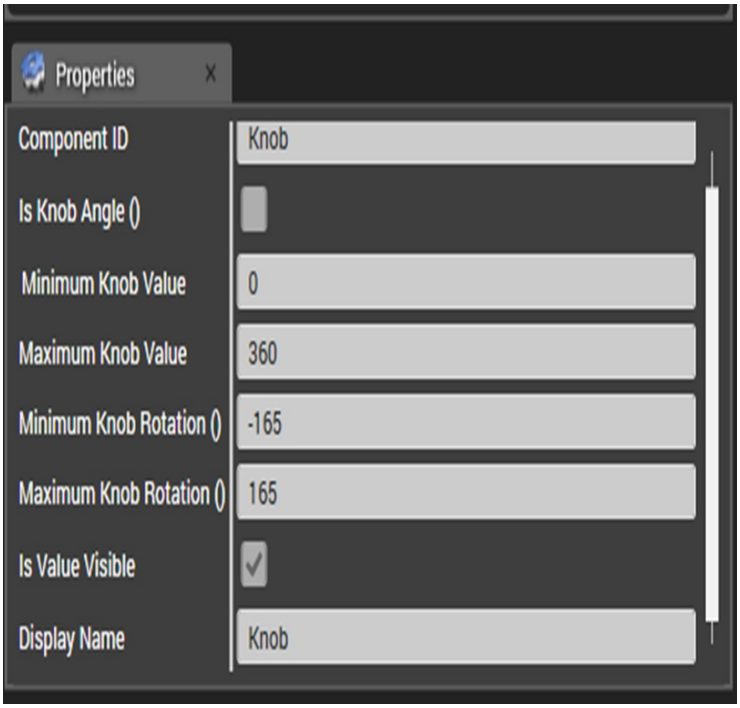
4. KNOB CONTROL

Case 1:Default Setup

(A) Default NodeStyle

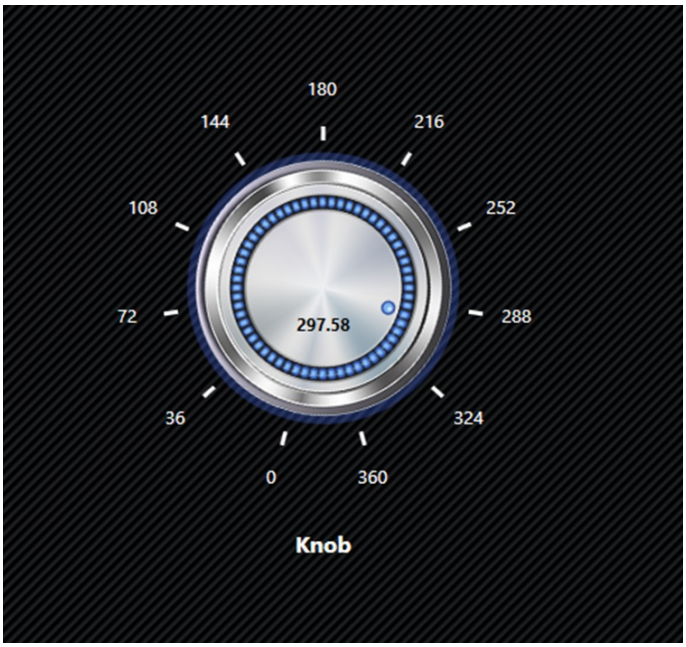


(B) Default Property window



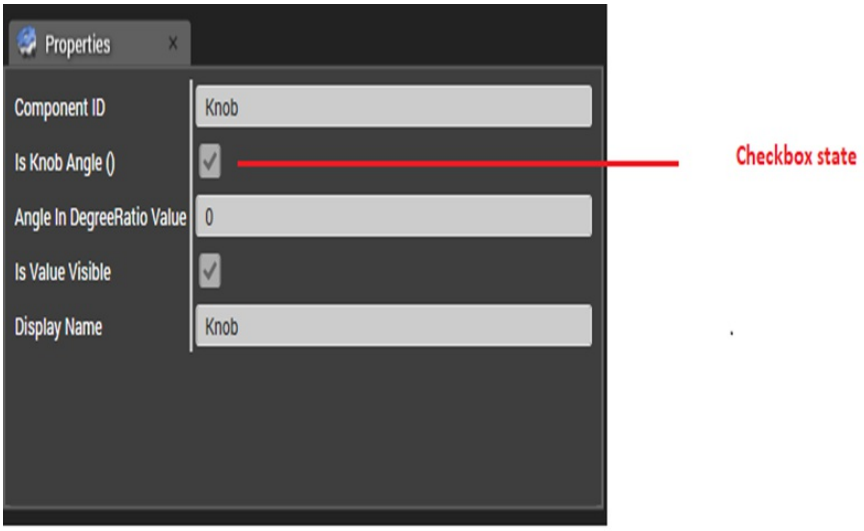
(C) Default Preview Window

(D) Default View in HOST

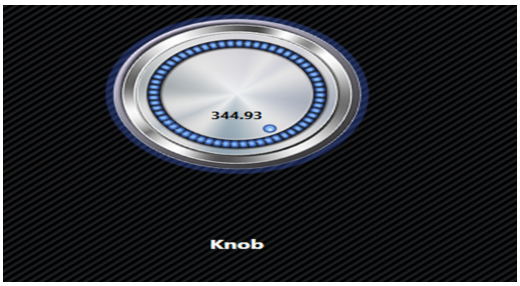


Case 2:-When "Is Knob Angle()" is checked in property window

(A) Property window



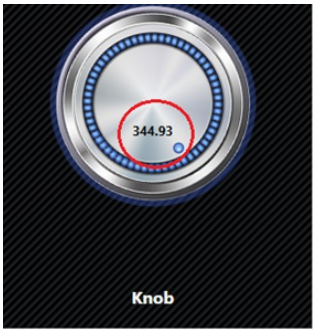
(B) View in HOST



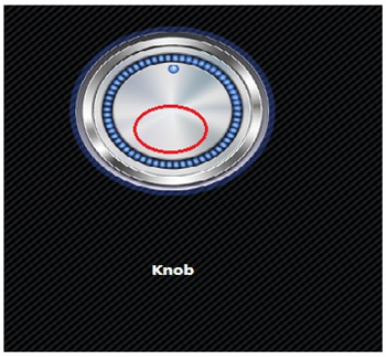
Case 3: when "Is Value Visible" is unchecked in property window.

(A) Property Window

(B) View in HOST



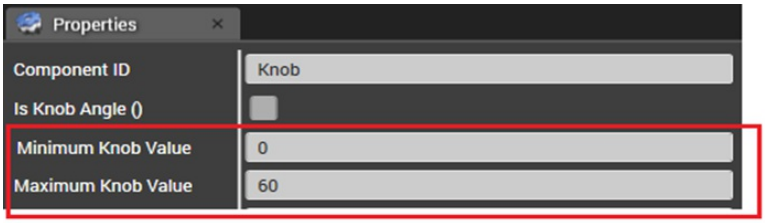
when "Is Value Visible"
CheckBox is True



when "Is Value Visible"
CheckBox is False

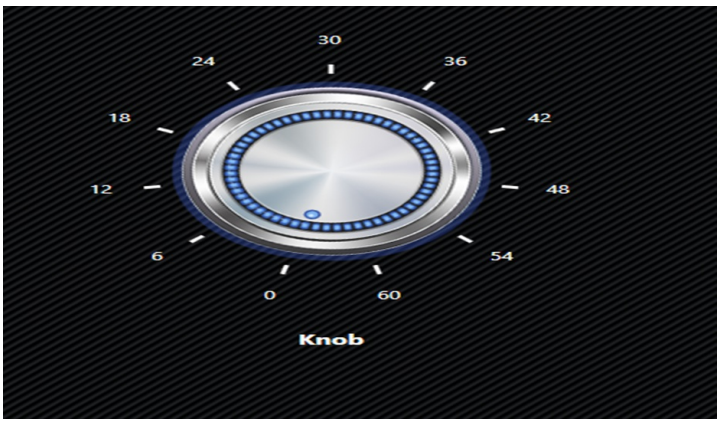
Case 4:-Changing values in property window for “Max Knob Value” and “Min Knob Value”

(A) Property Window



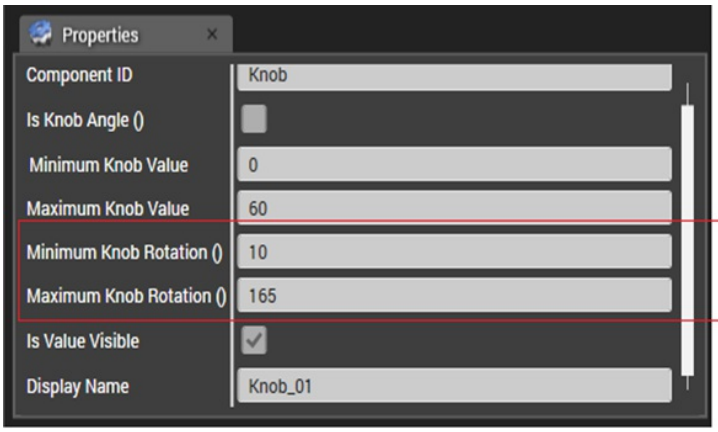
after making change in Knob values

(A) View in Host



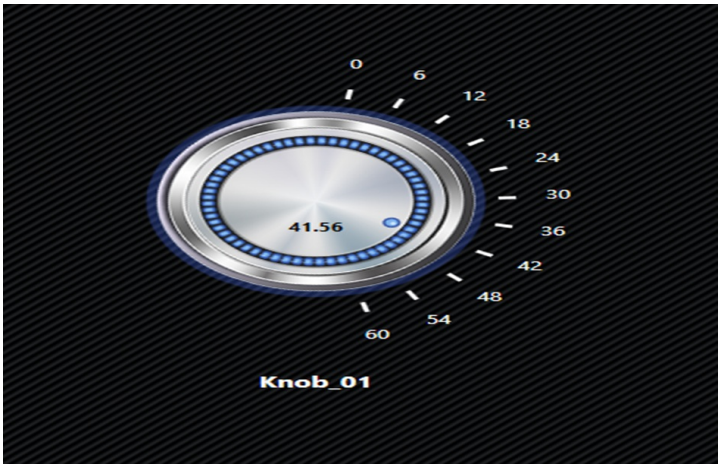
Case 5:-When “Minimum Knob Rotatation” and “Maximum Knob Rotation” is changed

(A) Property Window



Values are changed

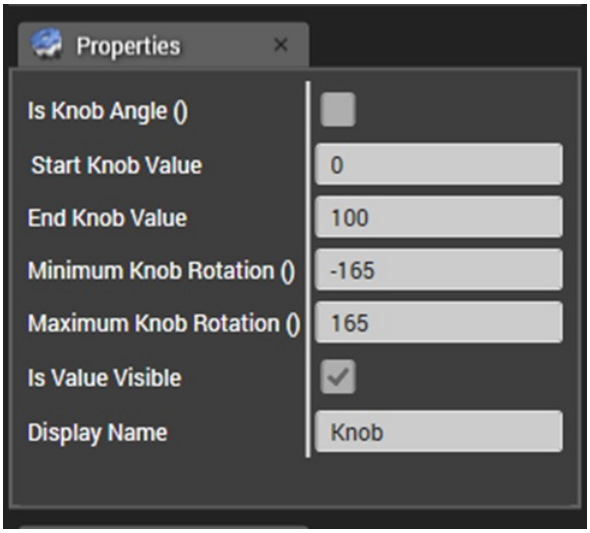
(A) View in HOST



Case 3:-Different Tick values scale

(A) When Tick value scale like (0 To 100)

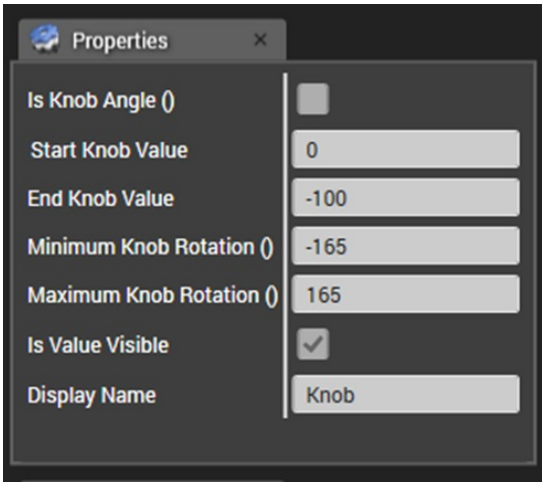
(1) Property window



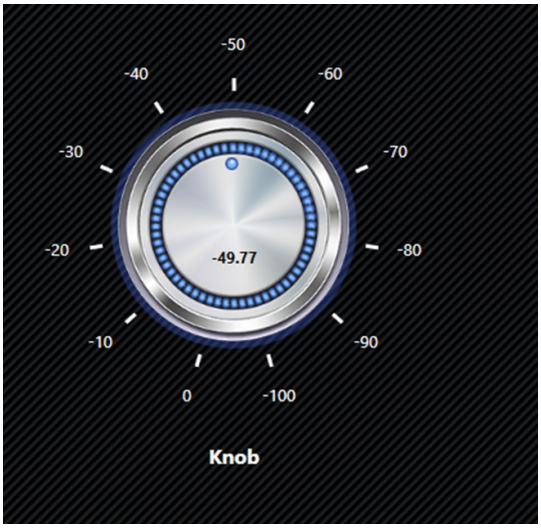
(2) View in HOST

(B) When Tick value scale like (0 To -100)

(1) Property window

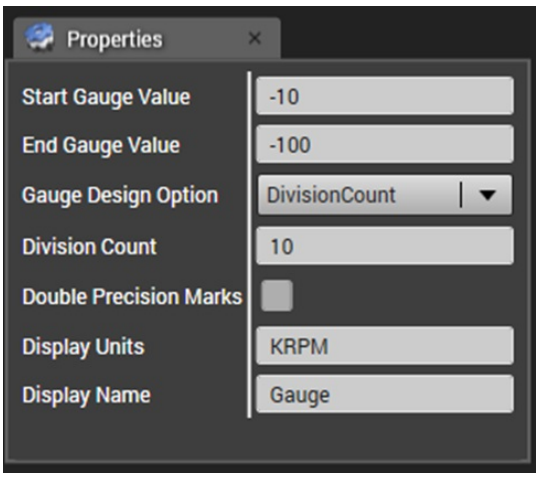


(2) View in HOST



(A) When Tick value scale like(-10 To -100)

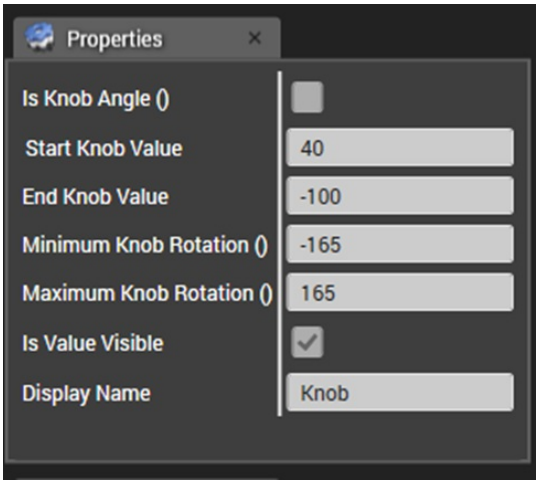
(1) Property window



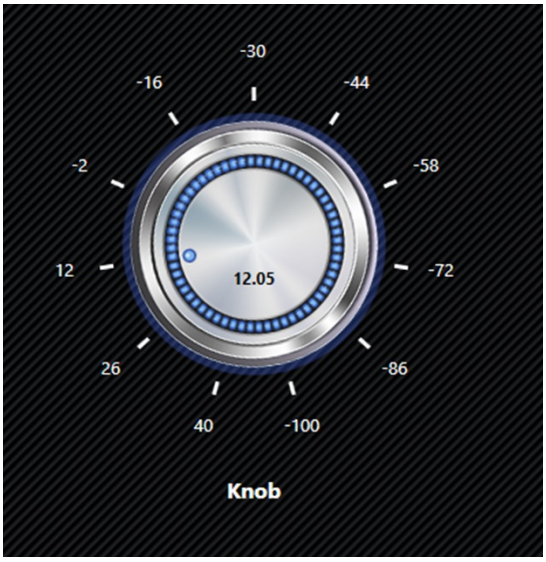
(2) View in HOST

(D) When Tick value scale like (40 To -100)

(1) Property window

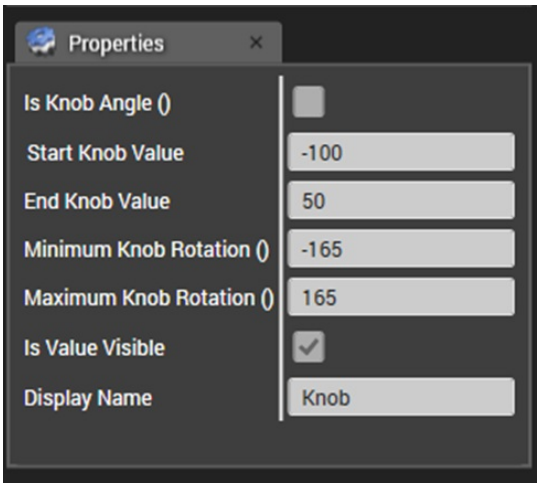


(2) View in HOST

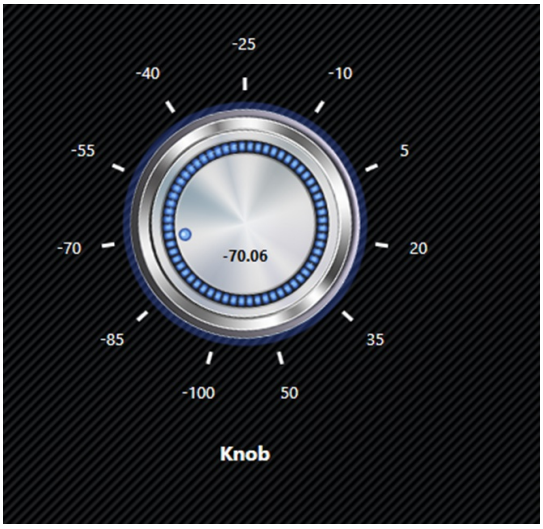


(E) When Tick value scale like(-100 To 50)

(1) Property window

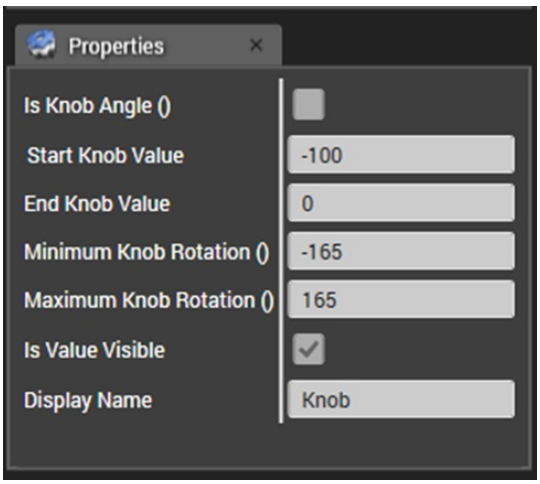


(2) View win HOST

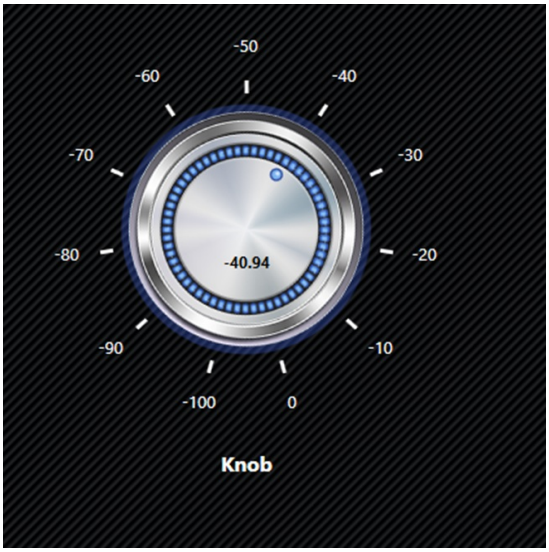


(F) When Tick value scale like(-100 To 0)

(1) Property window

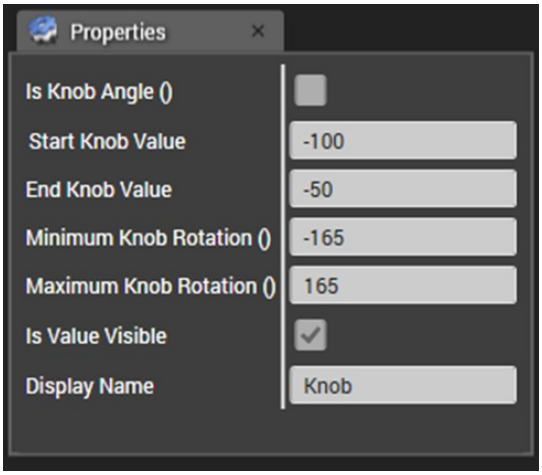


(2) View in HOST



(G) When Tick value scale like(-100 To -50)

(1) Property window



(2) View in HOST

Note: Value of this control is persist. It maintains the last state of control.

LED CONTROL

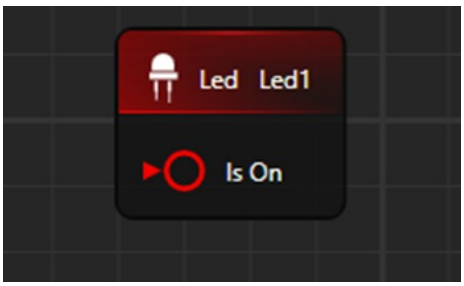
5. LED CONTROL

Newly implemented feature in 1.x

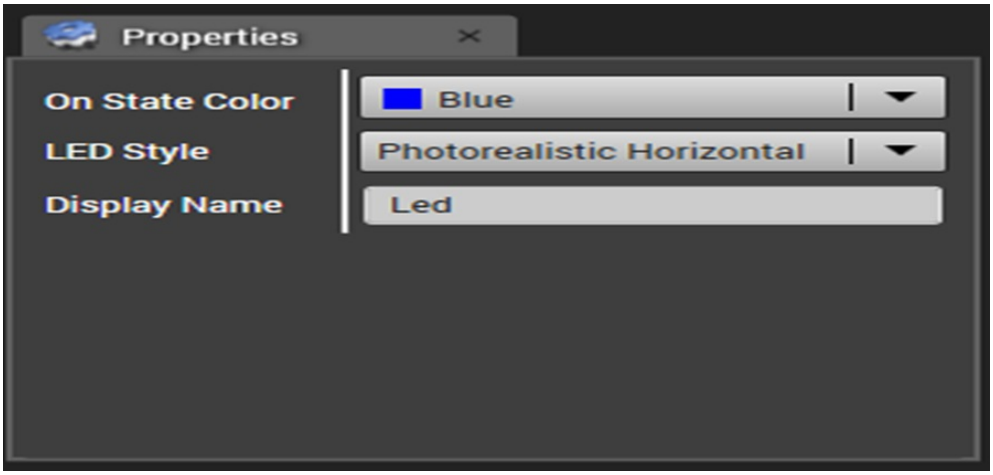
(A) View in HOST will be as according to the Led Style

Case 1:-Default Setup

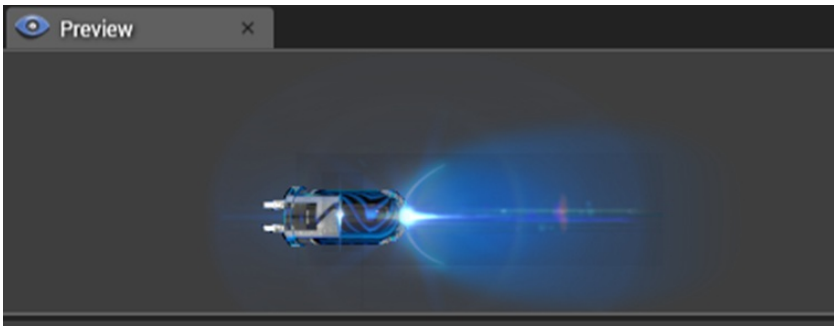
(A) Default Node Style



(B) Default Property Window



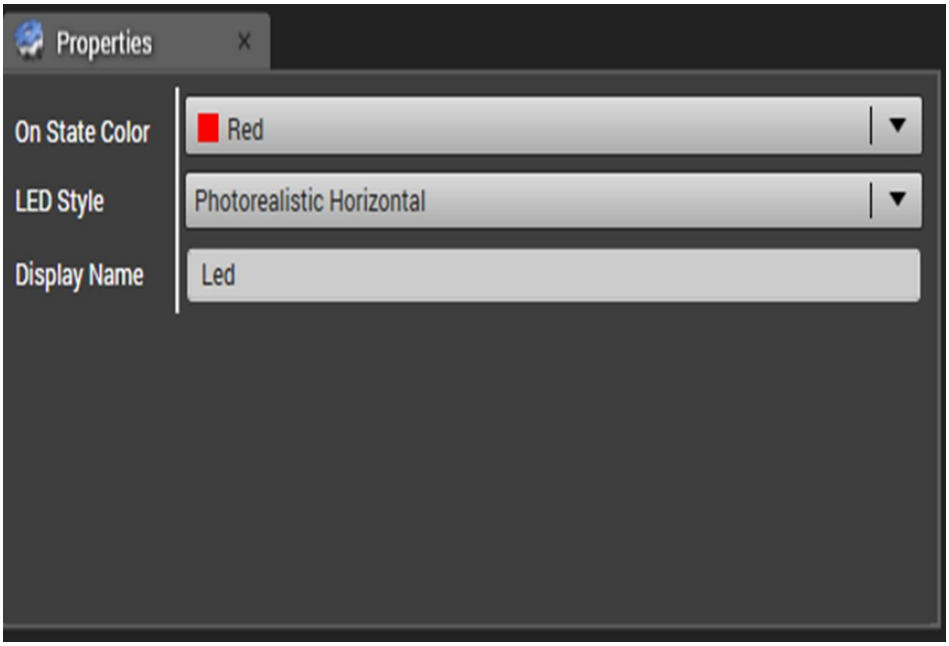
(C) Default Preview Window



(D) Default View in HOST

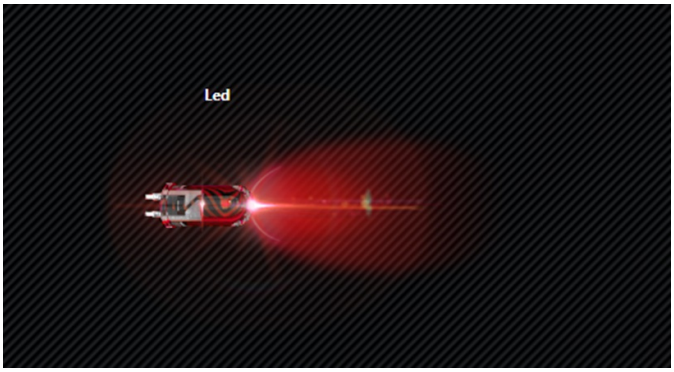
Case 2:-On changing "On State Color"

(A) Property window



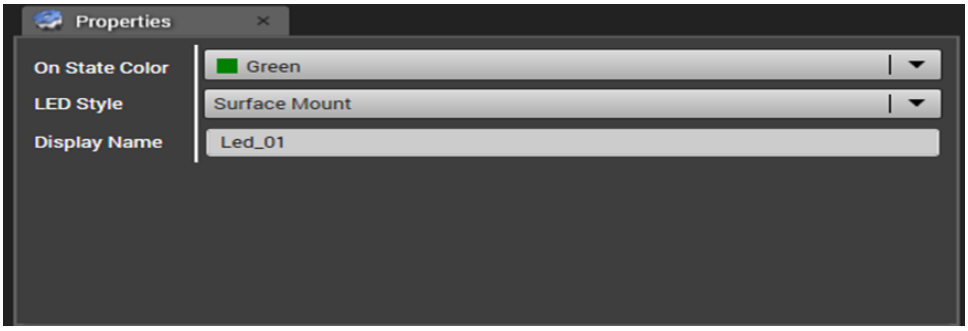
(B) Preview Window

(C) View In HOST

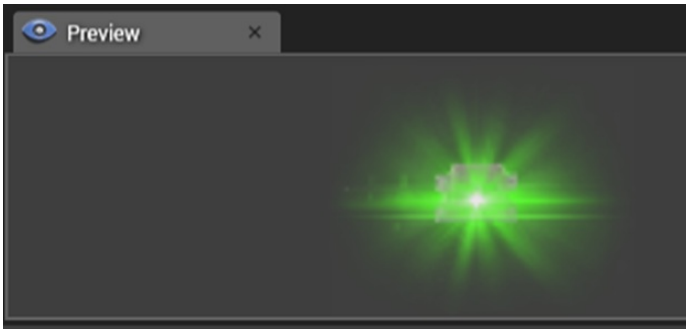


Case 3:-On changing “LED Style”

(A) Property window



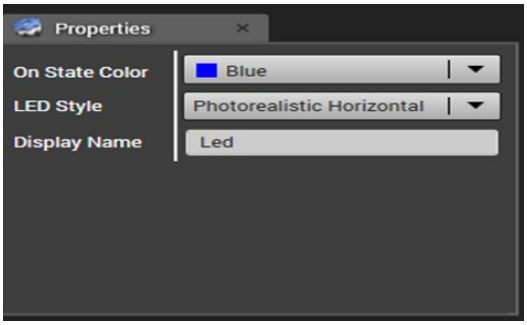
(B) Preview Window



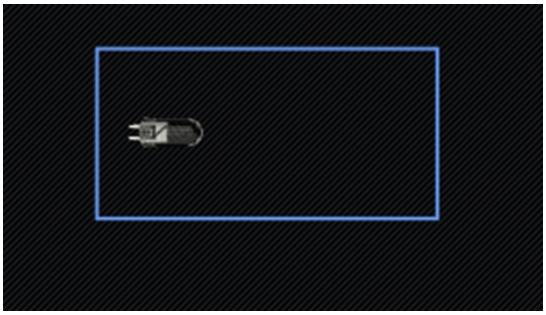
(C) View in HOST

Case 4:-On changing “LED Style” designer view in HOST

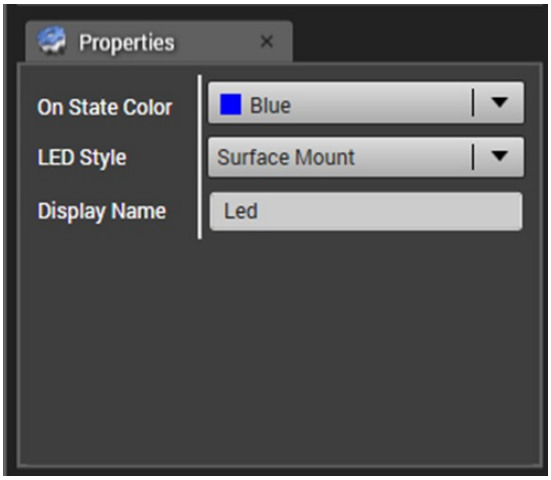
(A) Property window



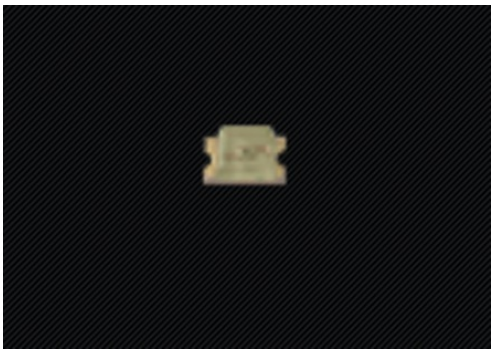
(B) View in HOST at Design time



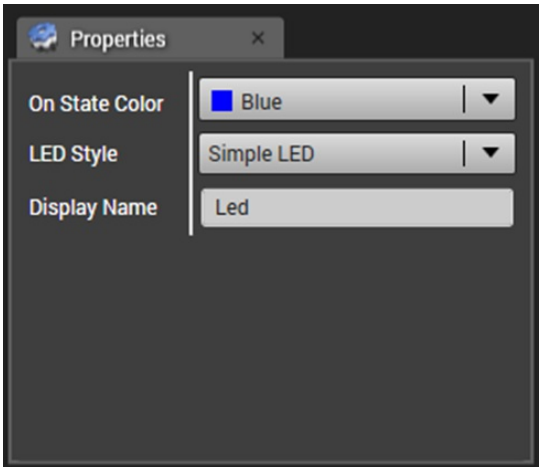
(C) Property window



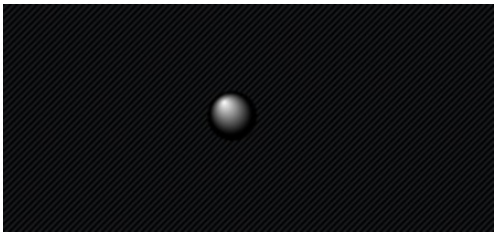
(D) View in HOST at design time



(E) Property window



(F) View in HOST at design time

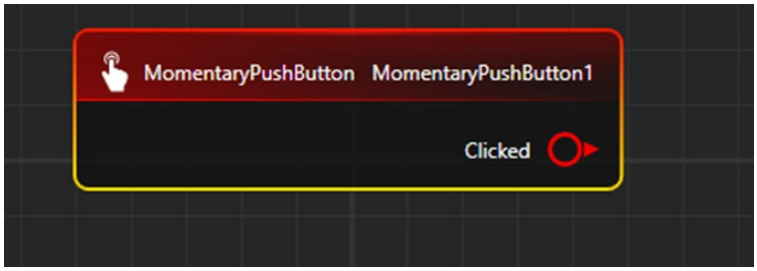


MOMENTARYPUSHBUTTON CONTROL

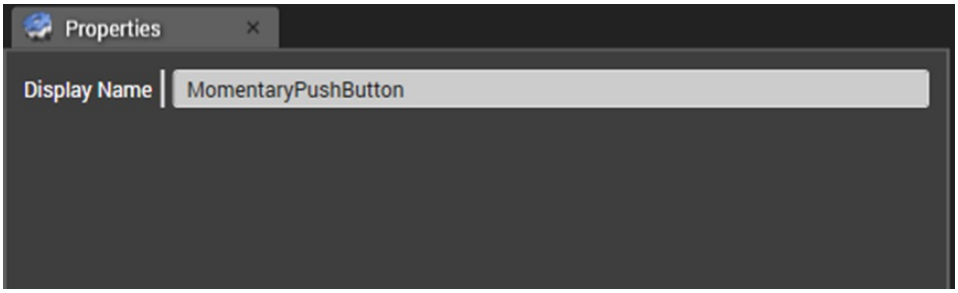
6. MOMENTARYPUSHBUTTON CONTROL

Case 1:-Default Setup

(A) Default Node Style

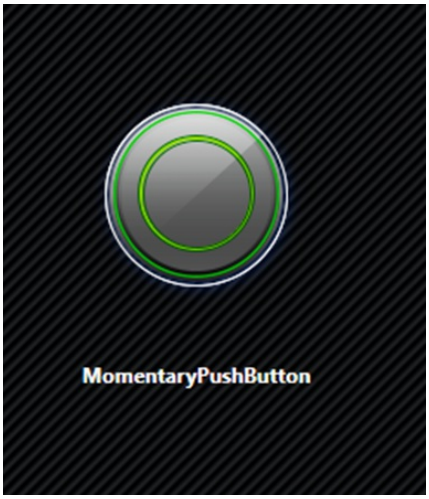


(B) Default Property Window



(C) Default Preview Window

(D) Default View In HOST



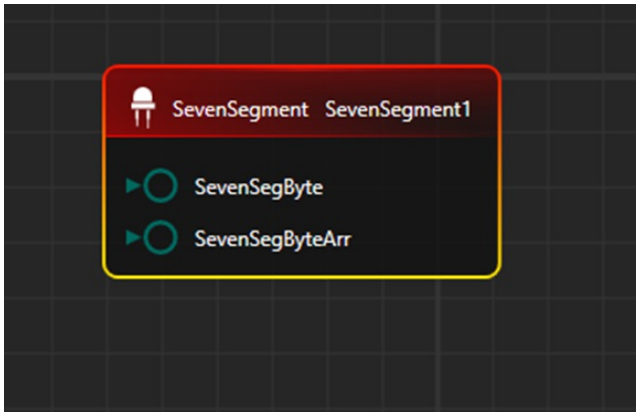
Note: It will either provide “True” or “False” value

SEVEN SEGMENT CONTROL

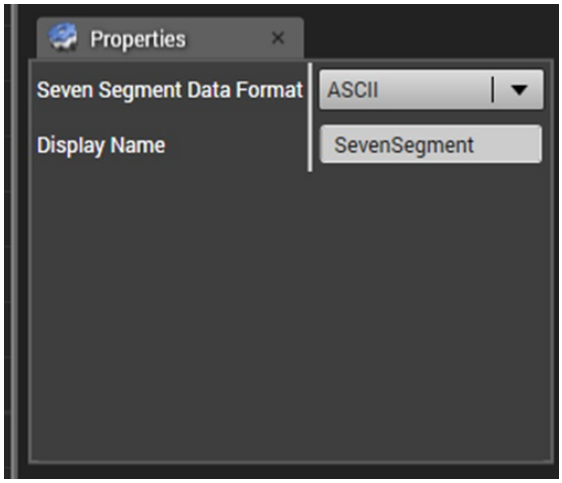
7. SEVEN SEGMENT CONTROL

Case 1:-Default Setup

(A) Default Node Style

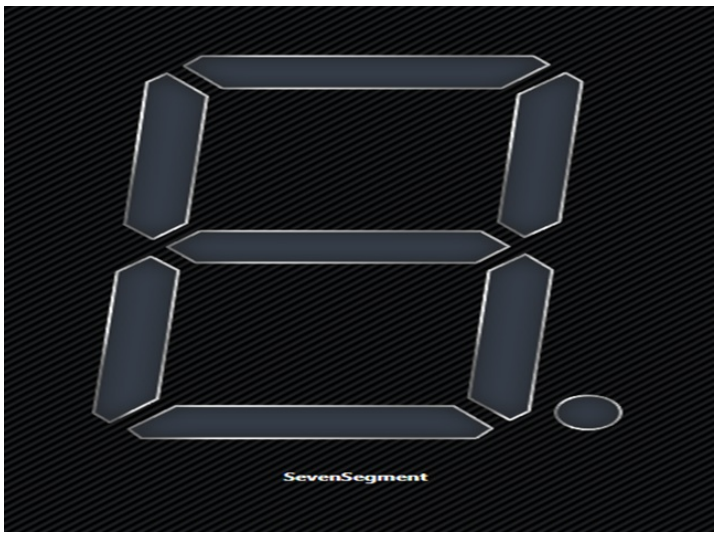


(A) Default Property Window



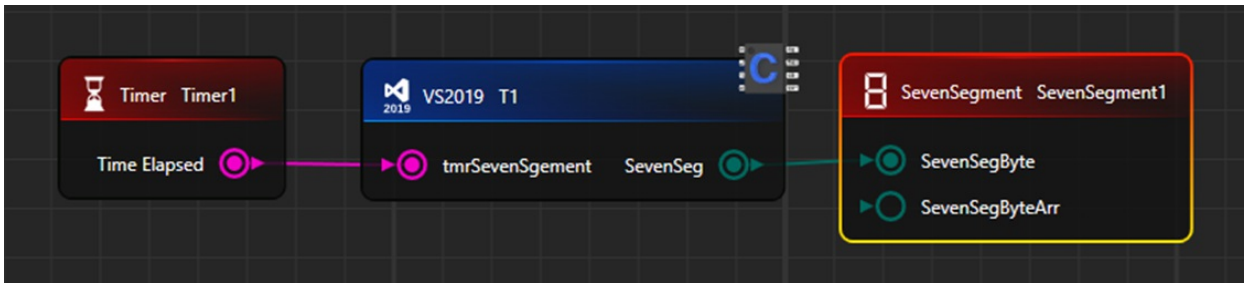
(C) Default Preview Window

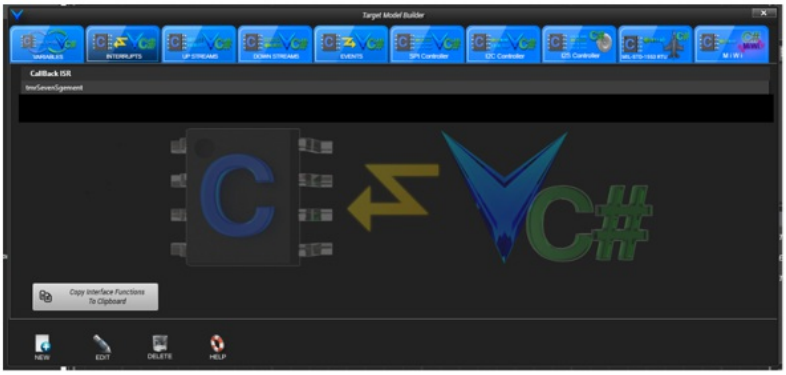
(D) Default view in HOST

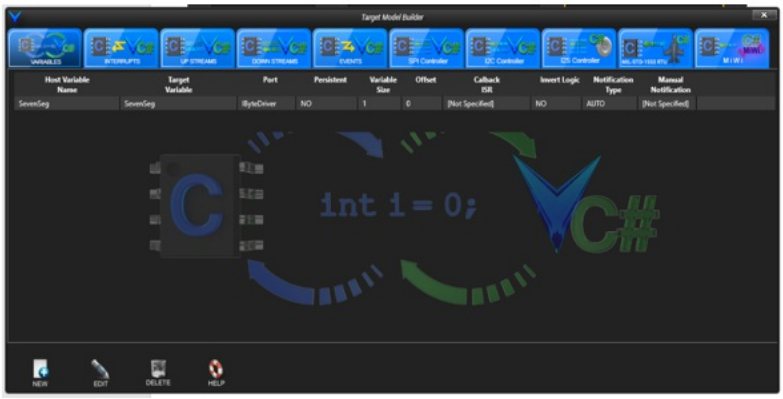


Case 2:- Result in Default Settings

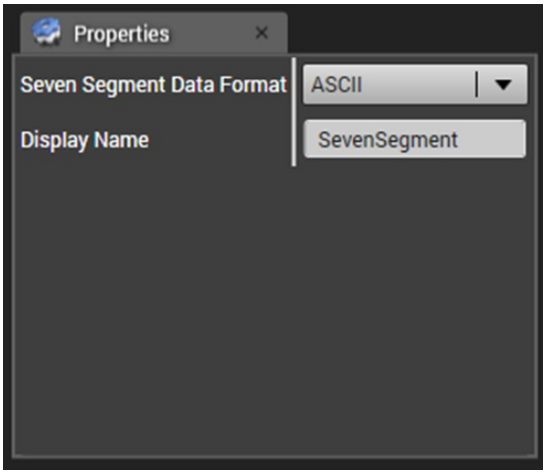
(A) Node Style







(B) Property Window



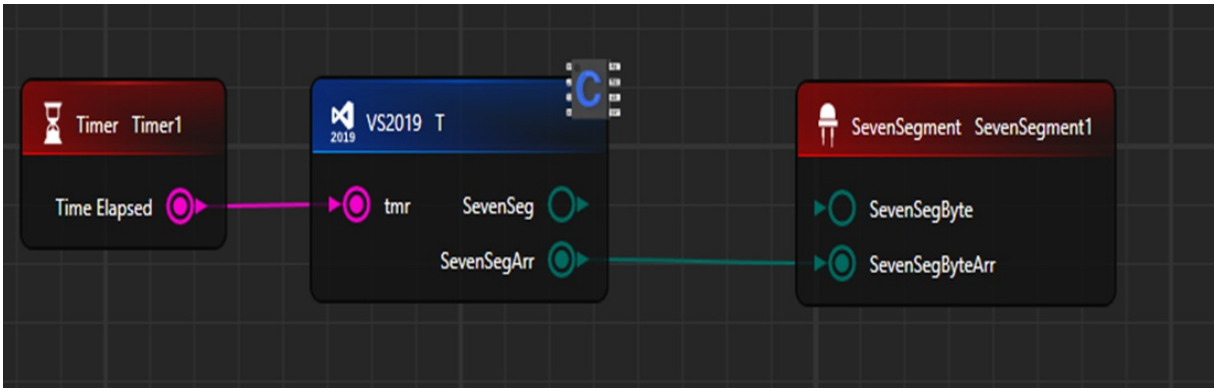
(C) View in HOST

Ex:-It will be defined in TMB

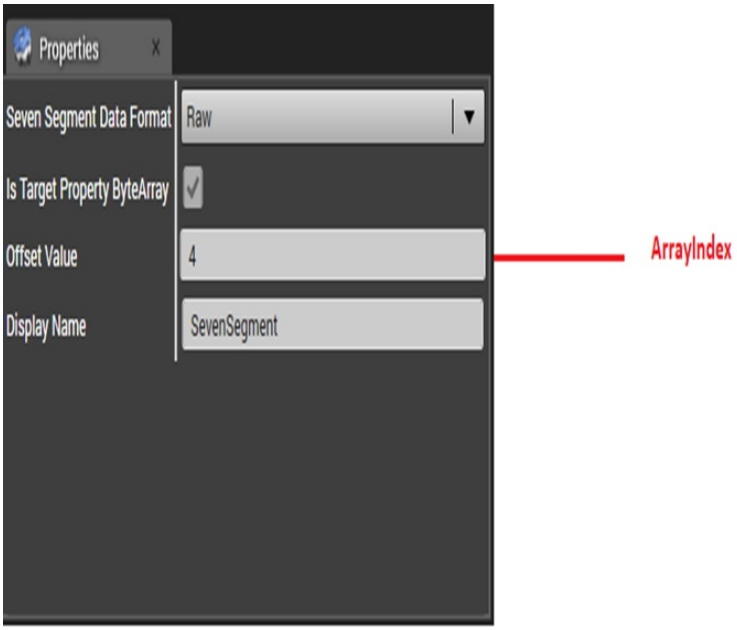
```
unsigned char SevenSeg = 48;
void tmrSevenSgement()
{
    if (SevenSeg >= 58)
    {
        SevenSeg = 48;
    }
    SevenSeg++;
}
```

Case 3:-when raw data is supplied in the form of "Array"

(A) Default Setting of Node



(A) Default Setting for Property Window



(C) View in HOST

Implemented Example:-It is implemented in TMB

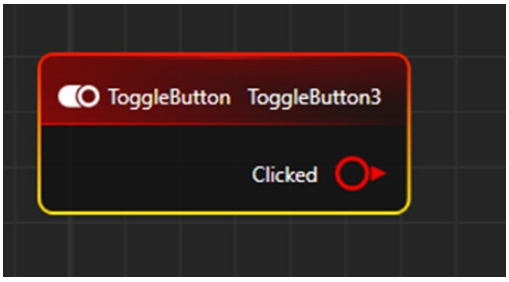
```
unsigned char SevenSegArr[] = {
    250,200,215,210,255
};
```

TOGGLE BUTTON CONTROL

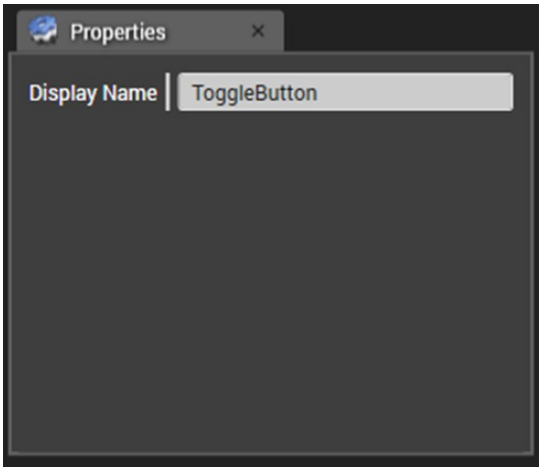
8. TOGGLE BUTTON CONTROL

Case 1:-Default Setup

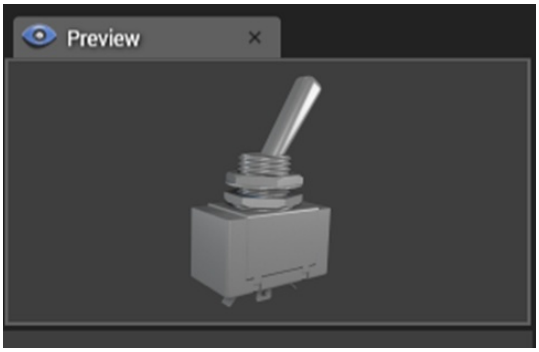
(A) Default Node Style



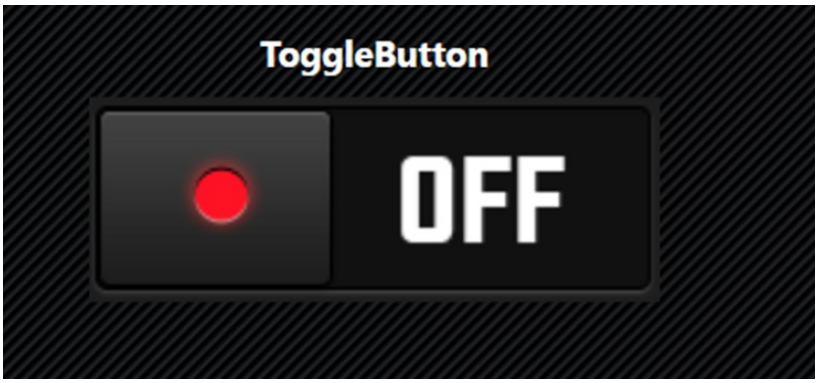
(B) Default Property Window



(C) Default Preview Window



(D) View in HOST



Note: It will either provide “True” or “False” value. It also persists the last state.

TIMER CONTROL

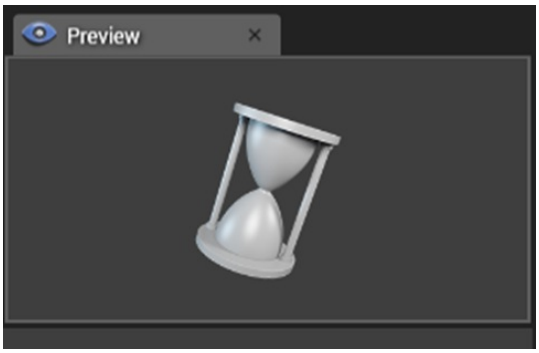
9. TIMER CONTROL

Case 1:-Default Setup

(A) Default Node Style

(B) Default Property window

(C) Default Preview Window



Note: It does not have any view.so it can't be seen in HOST. It is used to generate timer based event in TMB.