DigitalMeter Control Reference

Show numeric values as digital readouts.

Reference **Controls** **Charts** DigitalMeter

The DigitalMeter is a graphical component designed to display numerical values in a clear and customizable digital format within Displays/Draw. It supports multiple meters, which can be populated through DataTables or configured manually. This makes it suitable for visualizing values such as power consumption, temperature, order status, and other real-time or static data points.

Requirements
Configuration
Runtime Execution

Requirements

This component is Portable, which means it will run both on Windows WPF displays and on Web Pages on any platform.

Configuration

To configure the DigitalMeter, access the Charts section in the Components Panel, select DigitalMeter, and place it in the drawing area. To edit its settings, double-click the object.

DigitalMeter Settings	
Field	**Description**
DataTable	Associates a DataTable with the meter. The table must include a column named ***Title*** and one of the following: ***Value*** or ***ValueObject***. Optional columns include ***MinValue***, ***MaxValue***, ***MinValueObject***, ***MaxValueObject***, and ***MeterColor***.
Number of Bars	Defines the number of bars of the meter that will be present in each one.
Start Color	Defines the start color of the meter, indicating low values.
End Color	Defines the final color of the meter, indicating high values.

Runtime Execution

The DigitalMeter supports real-time data visualization through a linked DataTable. Each meter displayed by the component is generated from a row in the table, which may contain both static numeric values and dynamic tag bindings.

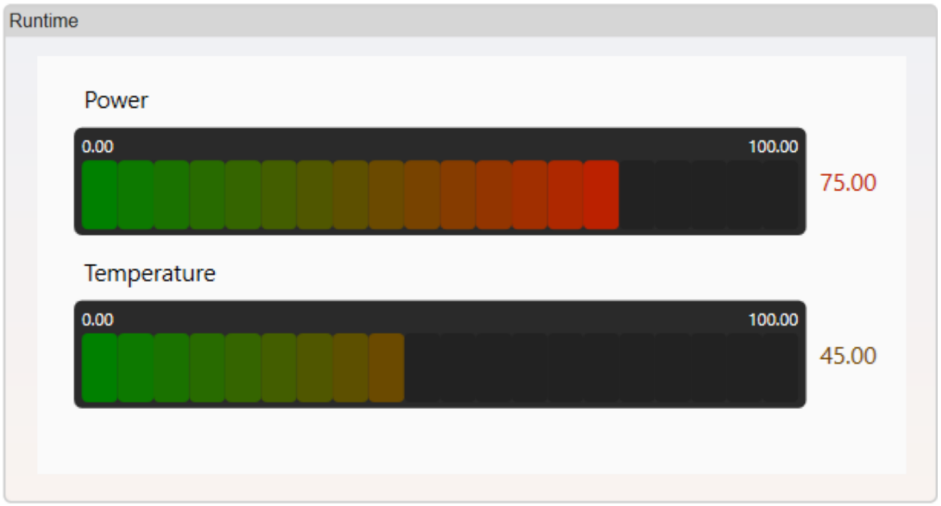
Column Behavior

- **Required Columns**:
  - *Title*: Label for the meter.
  - One of the following:
    - *Value*: A static numeric value.
    - *ValueObject*: A reference to a tag (object expression) that resolves at runtime.
- **Optional Columns**:
  - *MinValue* or *MinValueObject*: Sets the minimum value. Defaults to 0 if neither is provided.
  - *MaxValue* or *MaxValueObject*: Sets the maximum value. Defaults to 100 if neither is provided.
  - MeterColor: A hex color string to customize the meter's appearance (e.g., #*FFFFFF*).

Priority and Dynamic Binding

If a row contains both a direct value column (e.g., *Value*) and a corresponding object reference column (e.g., *ValueObject*), the object reference takes precedence. In this case, the component parses the string as a tag expression, resolves the runtime binding, and listens for tag value changes through events. This enables full dynamic behavior and real-time updates for each meter individually.

If the object reference is invalid or missing, the component gracefully falls back to the direct numeric value. If neither is present, a default value is used when applicable.



DataTable Fail-Safe

If the associated DataTable does not contain the required *Title* column or any valid value source (*Value* or *ValueObject*), the component does not render meters from the table. Instead, it loads default meters as a fallback, ensuring the chart continues to display data even if the linked DataTable is unavailable or misconfigured.

DataTable example

DigitalMeter DT

```
public void CreateDataTable() {
        DataTable table = new DataTable("Metrics");

    table.Columns.Add("Title", typeof(string));
    table.Columns.Add("Value", typeof(double));
    table.Columns.Add("MinValue", typeof(double));
    table.Columns.Add("MaxValue", typeof(double));
        table.Columns.Add("ValueObject", typeof(string));
        table.Columns.Add("MinValueObject", typeof(string));
        table.Columns.Add("MaxValueObject", typeof(string));
        table.Columns.Add("MeterColor", typeof(string));

    table.Rows.Add("Temperature", 22.5, 0, 35, null, null, null, "#52FC36");
        table.Rows.Add("Power", 50, null, null, null, null, null, "#5F7212"); // MinValue = 0 | MaxValue =
100
        table.Rows.Add("Level", null, null, null, @Tag.LevelValue.GetName(), @Tag.LevelMinValue.GetName(),
@Tag.LevelMaxValue.GetName(), "#FF4252");
        table.Rows.Add("Current", 50, 0, 300, @Tag.CurrentValue.GetName(), @Tag.CurrentMinValue.GetName(),
@Tag.CurrentMaxValue.GetName(), "#FFFFFF"); // Objects used in place of values

        @Tag.myDT.Value = table;
}
```

In this section...