Below is an example **indicator** script for Quantower (v1.141.19) in C# that demonstrates how to:

1. Assign a hotkey (A key) to capture the current crosshair **Y-value** (price) on the chart.
2. Store that price in a **global static** variable named pricedump.

**Note:**

* For simplicity, this script uses an **indicator** that you load onto the chart.
* Quantower does not provide a “global variable” mechanism in the sense of sharing data across all scripts by default. However, marking a static field within the script effectively makes it accessible anywhere else in your code that references this class.
* This code is specifically structured for Quantower’s standard C# scripting environment as of version **1.141.19**.
* In order for hotkeys to work, make sure the chart (with the loaded indicator) is in focus when pressing the key.

**Step-by-Step**

1. **Open Visual Studio 2022** (or your preferred C# IDE).
2. Create a new **Class Library** project (if you haven’t already) for your Quantower custom scripts.
3. Add references to the necessary Quantower .dll files (usually found in your Quantower installation folder).
4. Replace the contents of your .cs file with the following code (or add this class to your existing project).
5. Compile the project into a .dll and place it into the Quantower “Custom” or “Addons” folder (depending on your setup).
6. Restart Quantower (or refresh scripts), and apply the indicator to any chart.

**Indicator Script**

using System;

using System.Windows.Forms;

using TradingPlatform.BusinessLayer;

namespace MyQuantowerExtensions

{

/// <summary>

/// Indicator that, when loaded on a chart, allows the user to press the 'A' key

/// to capture the current crosshair price (Y-value) and store it in a static global variable.

/// </summary>

public class CrosshairPriceDumper : Indicator

{

// Global static variable accessible from outside this class if needed.

public static double pricedump = 0.0;

public CrosshairPriceDumper()

{

// Name and description as they appear in Quantower

this.Name = "Crosshair Price Dumper";

this.Description = "Press 'A' to dump crosshair's price (Y-value) into a global static variable.";

}

/// <summary>

/// Called when a key is pressed on the chart.

/// </summary>

/// <param name="e">ChartKeyEventArgs containing the pressed key info.</param>

protected override void OnKeyDown(ChartKeyEventArgs e)

{

base.OnKeyDown(e);

// We check if the user pressed the 'A' key.

if (e.KeyEventArgs.KeyCode == Keys.A)

{

// Attempt to get the Crosshair tool from the current chart

var crosshair = this.CurrentChart?.GetChartTool<ChartToolCrosshair>();

if (crosshair != null)

{

// Extract the current crosshair price

double crosshairPrice = crosshair.CrosshairData.Price;

// Store it in our static global variable

pricedump = crosshairPrice;

// Optionally, log it in the Quantower Output / Logs

this.Log($"Crosshair price dumped: {crosshairPrice}", IndicatorLogLevel.Information);

}

else

{

this.Log("Crosshair tool not found on the chart.", IndicatorLogLevel.Warning);

}

}

}

protected override void OnInit()

{

base.OnInit();

// Called once when the indicator is loaded

}

protected override void OnUpdate(UpdateArgs args)

{

base.OnUpdate(args);

// Called on each chart update/tick if needed

}

}

}

**How it Works**

1. **OnKeyDown**
   * Quantower calls OnKeyDown whenever you press a key while the chart (with this indicator) is focused.
   * We check if the key is A (e.KeyEventArgs.KeyCode == Keys.A).
2. **Grabbing the Crosshair Price**
   * We retrieve the crosshair tool from the chart via this.CurrentChart?.GetChartTool<ChartToolCrosshair>().
   * If it exists, we read its current price from crosshair.CrosshairData.Price.
3. **Storing the Value**
   * We store the crosshair’s Y-value in the static variable pricedump, which you can access from anywhere by referencing CrosshairPriceDumper.pricedump.
4. **Logging (Optional)**
   * The this.Log(...) call sends a message to the Quantower log pane. Not required, but useful for verification.

**Usage**

1. **Load the Indicator on the Chart**: Right-click the chart, choose “Indicators”, and apply **Crosshair Price Dumper**.
2. **Hover the Crosshair** over any price/time on the chart.
3. **Press the A Key** to capture the crosshair price.
4. Inspect logs or the value of CrosshairPriceDumper.pricedump in your code to see the stored price.

You now have a hotkey-based solution that sets a static/global price variable whenever you press A.

<https://api.quantower.com/docs/TradingPlatform.BusinessLayer.Chart.html>