

# Scripting in DetStudio

**Abstract**

---

Application note describes the examples of script use in the application.

Author: Zbyněk Říha  
Document: ap0023\_en\_01.pdf

**Appendix**

---

File contents: -

-	Not used

## Contents

	History of revisions .....	3
	Related documentation .....	3
<b>1</b>	<b>Script.....</b>	<b>4</b>
1.1	Script language .....	4
<b>2</b>	<b>Script use in examples .....</b>	<b>5</b>
2.1	Alarm screen display .....	5
2.2	Brightness and contrast setting .....	5
2.3	Displayed/ edited variable recalculation .....	6
2.4	Whole matrix variable editing by one control NumericEdit .....	7
2.5	Menu item use according to logged user .....	10
2.6	Optimalization through script .....	11
<b>3</b>	<b>Technical support .....</b>	<b>13</b>
<b>4</b>	<b>Warning.....</b>	<b>14</b>

**History of revisions**

---

Revision	Date	Changes
001	14. 9. 2009	New document.

**Related documentation**

---

1. DetStudio development environment Help  
file: DetStudioHelp.chm

# 1 Script

---

Script use is a way to influence displayed screen control properties, variable or alias values through screen editor. It is based on events that occurred in the control system (independently of process execution). Script use can also dynamically influence, e.g. the sequence of screens or an appearance of the individual controls.

It is possible to create large, but functionally limited applications without script (control type usually) only with DetStudio Basic controls (with limited feedback). Control positions, their visibility, language and other properties are predetermined; the sequence of screens cannot be dynamically changed. It is difficult to create failure states responses and their subsequent acknowledgment without script.

Much more effective and functionally richer applications can be created with correct script use than with Basic controls only. Generally speaking, the functionality that is not directly available in the DetStudio screen controls can be programmed by script use.

Controls on the screens are accessed as objects – by their name (same as the control type), properties and methods.

## 1.1 Script language

---

Script language used in the screens editor is ST-type (Structured text) and has following properties:

- ◆ commands are separated by a semicolon (";")
- ◆ more than one command can be located on a single line
- ◆ script can reference selected control properties on any screen and on any variable value
- ◆ declarations of any variables and constants are not supported
- ◆ only If command can be used for program branching.

It is recommended to use intellisense menu opened by Ctrl+J shortcut when writing code. Selection of individual properties, controls, etc., is possible from intellisense menu. List of displayed menu items depends on the place in script where the menu was called up. Writing a dot behind control name displays a list of all available methods and properties.

Script language can be used for user procedures designing in screens.

**More information about script language (including description of practical use) can be found in DetStudio development environment Help.**

## 2 Script use in examples

### 2.1 Alarm screen display

Following condition is required in the applications during terminal operation: In case that alarm occurs, alarm screen is displayed immediately, independently on the currently opened screen. This can be solved through a screen Global and its event OnRefresh().

Assuming, e.g. a variable “Err\_Alarm” is defined and its individual bits correspond to the different alarms in technology, the code for required screen displaying (e.g. named Alarms) is as follows.

```
event Global_OnRefresh()  
    If Err_Alarm > 0 Then //Determine, if alarm is present  
        Alarms.Show(); //If alarm is present, alarm screen opens  
    EndIf;  
end;
```



Alarm screen displaying depends on the reaction time of control system. This time is defined by the Global screen RefreshPeriod property value.

#### **Attention**

Once requested screen is already open, it does not open again. This means that trying to open the same screen twice does not execute either its event OnClose() or OnOpen().

### 2.2 Brightness and contrast setting

Brightness setting is supported only in selected types of control systems/terminals. Contrast can be set by trimmer on selected types of control systems/terminals. It can be set by script for other types of control systems/terminals.

Brightness and contrast settings can be programmed through the controls “NumericEdit”. User should enter brightness and contrast value in a range of 0 to 100 % directly. Another option (realised in this application note) is to use, e.g.  and  (**Up** a **Down**) keys for brightness or contrast setting and a control “NumericView” for displaying of set value.

Create a screen named Contrast. Insert the control “NumericView” (with format setting ###) and two controls “Key” according to following figure.

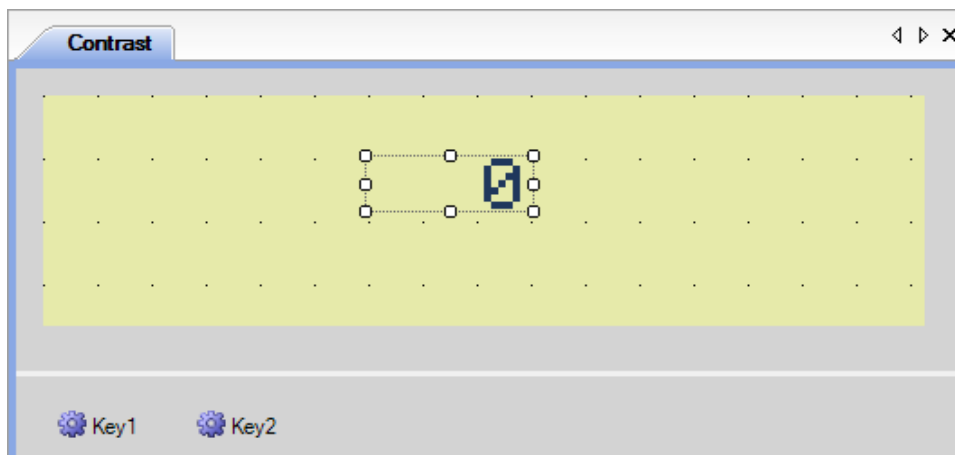




Fig. 1 – Contrast setting screen

Set the parameters `KeyCode` of the controls “Key” as  and  keys. Write the following script into event `OnKeyDown()` of control with assigned  key.


```
event Key1_OnKeyDown()  
    If Application.Contrast < 100 Then //Check, if the maximum contrast is not  
entered  
        Application.Contrast = Application.Contrast + 1; //Contrast increasing  
        NumericView1.Value = Application.Contrast; //Contrast displaying  
        NumericView1.Refresh(); //NumericView control refreshing  
    EndIf;  
end;
```

First, it is verified that maximum contrast value is not entered. If so, the next part of the script is not executed. If not, value 1 is added to the current value of the contrast setting. Then, changed value is saved into a control “NumericView1” that displays it on the screen.

### Attention

*If a parameter Value is supposed to be used in control, a parameter Variable (in window Properties) must be left as “(none)”. In case that some variable is set in parameter Variable, the control will behave according to this variable. The parameter Value has lower priority in script than the parameter Variable in window Properties.*

Finally, the control “NumericView1” is refreshed, so the value assigned as the parameter `Value` is displayed immediately on terminal/control system screen. If refresh function is not programmed, the value assignment takes affect after the screen parameter `RefreshPeriod` time (set in window Properties).

The procedure is similar for a control “Key2” with assigned  key in script. The contrast is decreased with this control.

```
event Key2_OnKeyDown()  
    If Application.Contrast > 0 Then  
        Application.Contrast = Application.Contrast - 1;  
        NumericView1.Value = Application.Contrast;  
        NumericView1.Refresh();  
    EndIf;  
end;
```

If system supports brightness setting, its change can be done with similar procedure. A property “`Application.Contrast`” is replaced with a property “`Application.Brightness`”.

## 2.3 Displayed/edited variable recalculation


---

The time information is set in a large number of modules in milliseconds (e.g. modules in DetStudio programming part). In case that long time periods are set (e.g. in hours); entering values in milliseconds can be very confusing for users. It is possible to program displaying and editing of the time periods directly in hours using script for easier use.

Displaying and editing of recalculated time value by the control “NumericEdit” is described in the following example.

When the requested screen for entering the time period (e.g. in hours) is opened, recalculation must be done immediately.

```
event Recalc_OnOpen()  
    Recalc.FocusFirstControl();  
    NumericEdit1.Value = Time / 3600000;  
end;
```

First line of the script (“ScreenName.FocusFirstControl();”) is generated automatically on each created screen by DetStudio. The control with the lowest value of a parameter `TabIndex` (see DetStudio development environment Help) is focused on currently open screen with previous script. User can start to edit this control immediately after screen displaying through  (Enter) key.

In next step the time value in milliseconds is recalculated to hours and saved to the control “NumericEdit1” (using the parameter `Value`). The screen displays now the time value in hours.

### **Attention**

*If the parameter `Value` is supposed to be used in control, the parameter `Variable` (in window Properties) must be left as “(none)”. In case that some variable is set in the parameter `Variable`, the control will behave according to this variable. The parameter `Value` has lower priority in script than the parameter `Variable` in window Properties.*

The event `OnEditComplete()` must be used to enter time value with the control “NumericEdit” in hours. Write the following script that recalculates hour-based time to milliseconds-based time into this event.

```
event NumericEdit1_OnEditComplete()  
    Time = NumericEdit1.Value * 3600000;  
end;
```

The actual value is displayed with previous scripts when screen is opened. Requested value setting by the control “NumericEdit1” is done as well. However, if someone change the value, e.g. from a visualization, this change will not take any affect until screen is closed and opened again. The event `OnRefresh()` is used to perform this value change (so the screen do not need to be closed and opened again). Write the same script as is written in an event `OnOpen()` into this event.

```
event Recalc_OnRefresh()  
    NumericEdit1.Value = Time / 3600000;  
end;
```

## **2.4 Whole matrix variable editing by one control NumericEdit**

In some cases, it is useful to edit all of the matrix cells on single screen with single editing control, e.g. for PID regulator parameters setting. Following steps are used for designing that.

Create new screen and insert the control “Label” with text “PID parameters” here. Each row of matrix variable used as PID module parameters has its own meaning. Different texts describing individual rows will be displayed for better clarification. Insert the control “CaseLabelView” on the screen, double click on it and define texts according to following figure.

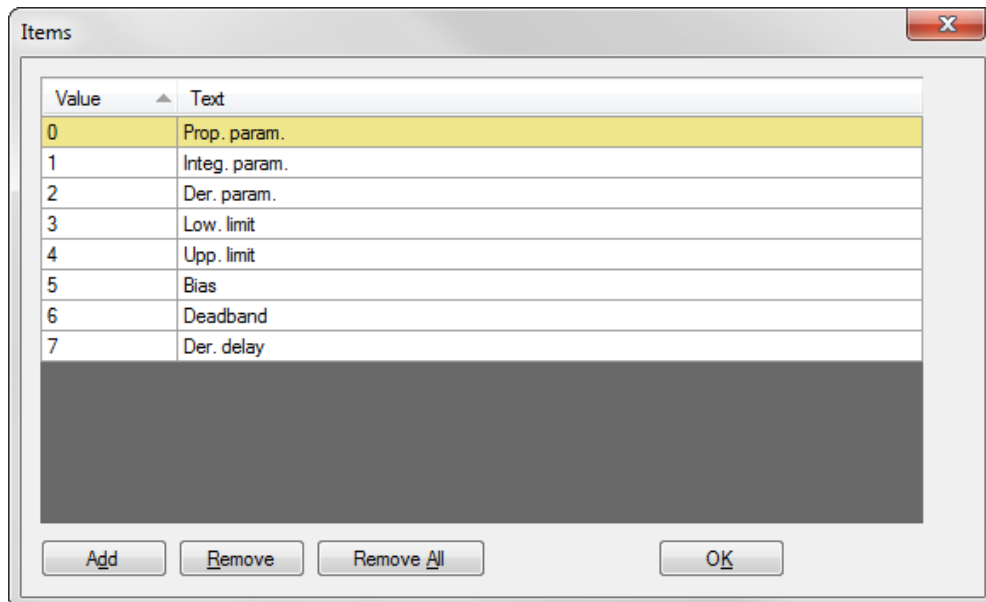




Fig. 2 – Control CaseLabelView setting

Next, insert the control “NumericEdit” on the screen. Double click on it and set editing possibility of zero cell of the PID\_params matrix (8 × 1 dimension matrix with PID module parameters).

Finally, insert two controls “Key” on the screen with the parameter `KeyCode` set as  and  (**Up** and **Down**) keys. Individual PID module parameters will be switched with these keys.

Actual screen design should be as follows.

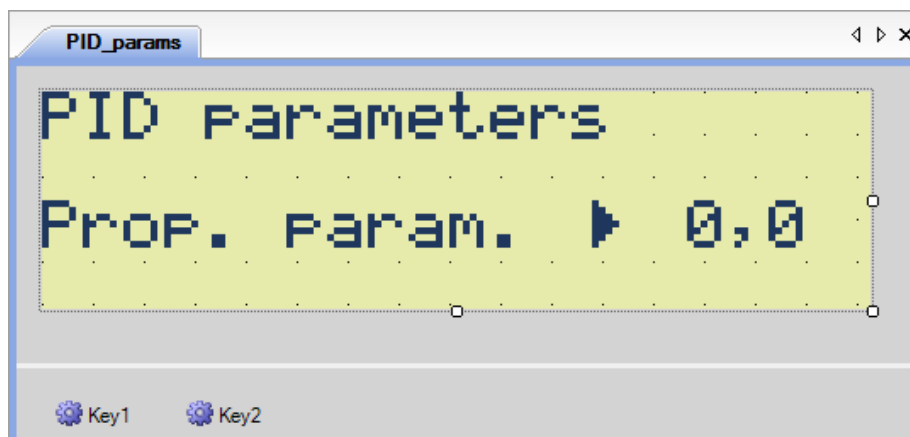





Fig. 3 – PID parameters setting screen design

The individual row switching in matrix is programmed for  or  key pressing in script. Text displayed in the control “CaseLabelView” will be changed together with this switching action.

First, correct text must be displayed in the control “CaseLabelView” control according to the current PID\_params matrix row. In this case the screen event `OnOpen()` will be used.

```
event PID_Params_OnOpen()
    PID_Params.FocusFirstControl();
    CaseLabelView1.Value = NumericEdit1.Row;
end;
```




First line of the script (`ScreenName.FocusFirstControl();`) is generated automatically on each created screen by DetStudio. Control with the lowest value of the parameter `TabIndex` (see DetStudio development environment Help) is focused on currently open screen with previous script. User can start to edit this control immediately after screen displaying through  (Enter) key.

The row number (displayed with the control “NumericEdit1”) is assigned to a control “CaseLabelView1” through the parameter `Value` in next line. This step leads to different matrix row editing and subsequently different text displaying.

### Attention


*If the parameter `Value` is supposed to be used in control, the parameter `Variable` (in window Properties) must be left as “(none)”. In case that some variable is set in the parameter `Variable`, the control will behave according to this variable. The parameter `Value` has lower priority in script than the parameter `Variable` in window Properties.*

The row switching (in “NumericEdit1” control) as well as the displayed text switching (in “CaseLabelView1” control) is done by pressing  key according to following script.

```
event Key2_OnKeyDown()  
    If NumericEdit1.Row < 3 Then //Check, if the value is within limit  
        NumericEdit1.Row = NumericEdit1.Row + 1; //Next row switching  
        CaseLabelView1.Value = NumericEdit1.Row; //Appropriate text selection  
        CaseLabelView1.Refresh(); //Control refresing  
    EndIf;  
end;
```

### Attention

*It is necessary to watch moving in matrix rows and columns to not cross over its dimensions. **In case that script code tries to display the non-existed matrix cell, control system will be restarted.***

First, the checking of row value (if it is within matrix dimension limits) must be done on pressing  key event. If limit is not exceeded, the control “NumericEdit1” allows moving to another row. Then, the value of currently displayed row is assigned as the control “CaseLabelView1” of the parameter `Value`. Requested text is displayed. Finally, the control “CaseLabelView1” is refreshed for row value assignment as the parameter `Value`. This action takes its affect on terminal/control system immediately. If refresh function is not programmed, the text change takes affect after the screen parameter `RefreshPeriod` time (set in window Properties).

Similarly  key pressing script is programmed.

```
event Key1_OnKeyDown()  
    If NumericEdit1.Row > 0 Then //Check, if the value is within limit  
        NumericEdit1.Row = NumericEdit1.Row - 1; //Preview row switching  
        CaseLabelView1.Value = NumericEdit1.Row; //Appropriate text selecting  
        CaseLabelView1.Refresh(); //Control refresing  
    EndIf;  
end;
```

### Note

*Similar procedure can be programmed for switching between matrix columns. A parameter `Col` can be used in script for this switching action.*

## 2.5 Menu item use according to logged user

In other cases, user can choose a particular menu item only after logging into control system as higher privileged user. Then, it is necessary to use script with a control "Menu" (from section General). The control "Menu" (unlike a control "MenuScreen" from section Basic) generates "the item was entered" event and therefore is not connected directly with screens. If it is required to recognize currently logged user on individual items, the control "Menu" takes place on screen with individual item settings (e.g. as shown below).

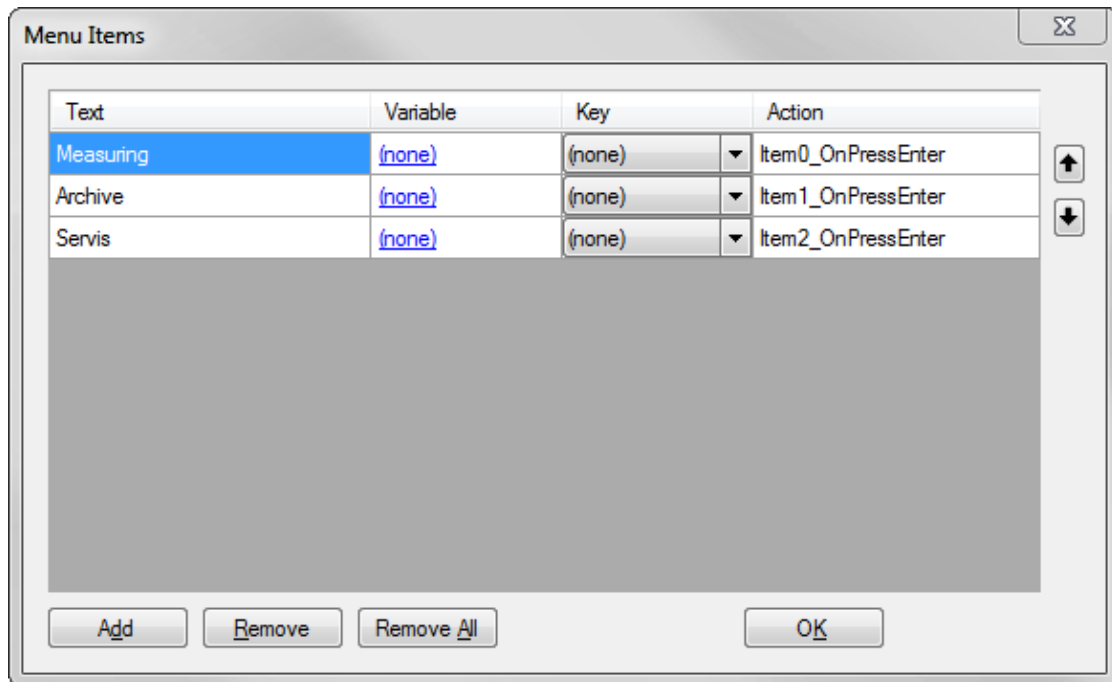


Fig. 4 – Item settings of controls Menu from section General

Three events are created in opened section DetStudio script after **OK** button confirming. Individual events correspond to set menu items. First two items (Measuring and Archive) can be used by user with any rights. Screen opening script is written directly into individual events.

```
event Menu1_Item0_OnPressEnter ()
    Measuring.Show();
end;
event Menu1_Item1_OnPressEnter ()
    Archive.Show();
end;
```

Only higher privileged user than Service can use last item. For our example, user is defined according to following figure.

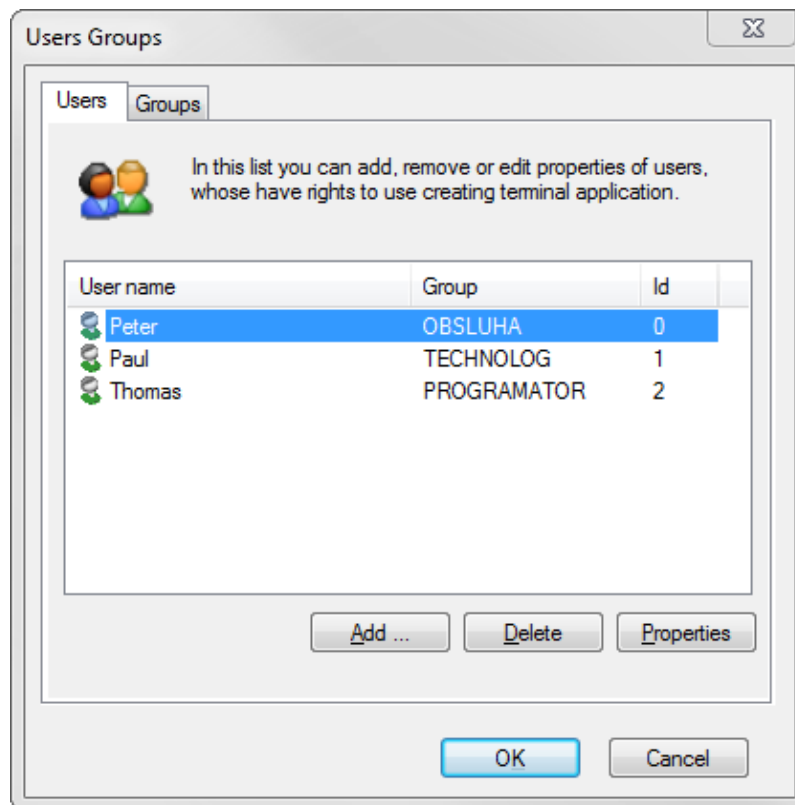


Fig. 5 – Users defined in the project

If Peter is logged, the Service item selection is without any affect. If Pavel or Thomas is logged, Servis screen appears. This functionality requires index of individual users (column “Id” in user definition) in combination with following script written into menu selection event of third item.

```

event Menu1_Item2_OnPressEnter ()
    If Application.ActualUser == 1 or Application.ActualUser == 2 Then
        Servis.Show();
    EndIf;
end;

```

### **Attention**

*If no one is logged into the control system, actually logged user index is 65535.*

## **2.6 Optimization through script**

The script use is also a big advantage for optimalization. It is common that one application may contain several heating branches and it is required to control these branches by terminal. Individual screens on terminal are identical for most heating branches, only different branches values are displayed. It is possible to use script for displaying/setting the group of parameters only with one screen. The condition for such parameter displaying is to store heating branches parameters in matrices.

The control “MenuScreen” (from the controls Basic) is used for selection of edited heating branch. For example, 3 branches are presented and it is required to display measured temperature, setpoint pipe flow temperature and set heating curve constant.

Create the screens “Menu” and “Branches”. Insert the control “MenuScreen” on the screen “Menu”. Double click on it and fill individual items as shown below.

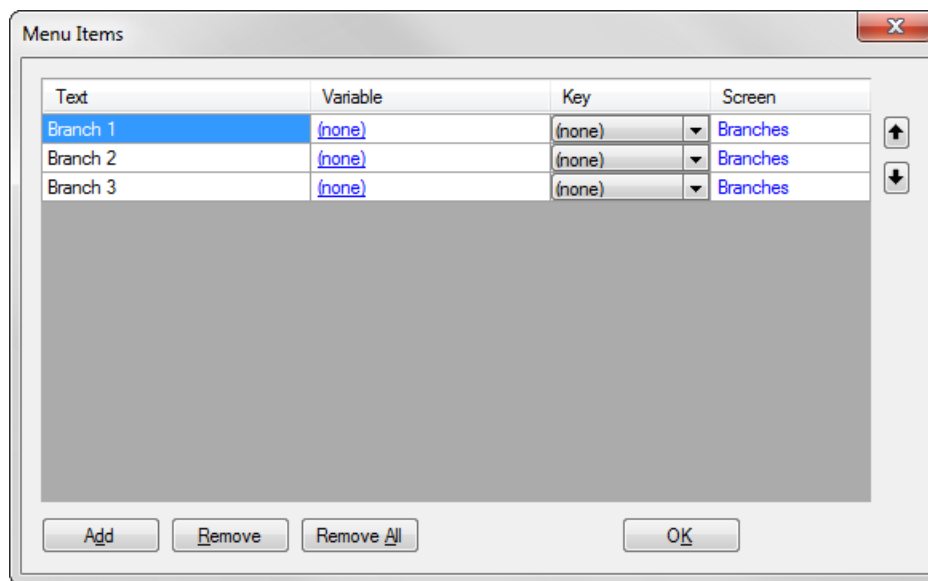


Fig. 6 – Labels of individual menu items

Insert two controls “NumericView” and one control “NumericEdit” on the screen “Branches” as shown below.

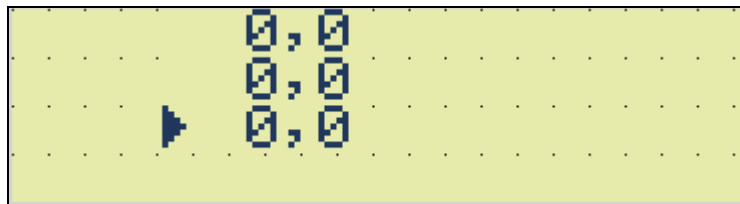


Fig. 7 – 3 heating branches – control screen

Three parameters of three heating branches are in  $3 \times 3$  dimension matrix. Each column represents one parameter and each row corresponds to one heating branch. Matrix cells, e.g. from the first row are assigned in controls.

Enter following script into the event OnOpen() on the screen “Branches”.

```
event Branches_OnOpen()
    Branches.FocusFirstControl();
    NumericEdit1.Row = Menu.MenuScreen1.SelectedIndex;
    NumericView1.Row = Menu.MenuScreen1.SelectedIndex;
    NumericView2.Row = Menu.MenuScreen1.SelectedIndex;
    Branches.Refresh();
end;
```

The matrix values are always assigned into controls when the screen “Branches” is opened. Value selection depends on the selected item from the control “MenuScreen” control on the screen “Menu”.

### 3 Technical support

---

All information concerning the script use in DetStudio will be provided by AMiT technical support. Technical support can be most preferably contacted via email at **support@amit.cz**.

## 4 Warning

---

AMiT spol. s r. o. does not provide any warranty concerning the contents of this publication and reserves the right to change the documentation without any obligation to inform about it

This document can be copied and redistributed under following conditions:

1. The whole text (all pages) must be copied without any changes.
2. All redistributed copies must retain the AMiT spol. s r. o. copyright notice and any other notices contained in the documentation.
3. This document must not be distributed for purpose of making profit.

The names of products and companies used herein can be trademarks or registered trademarks of their respective owners.