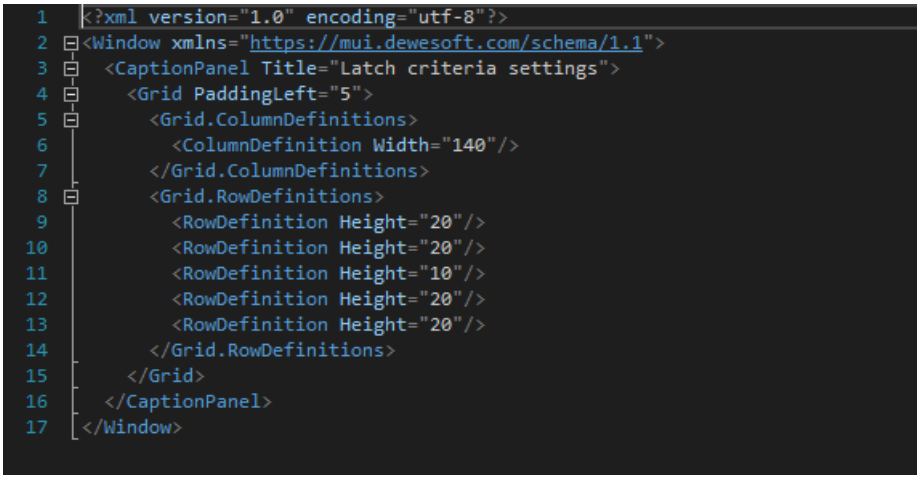
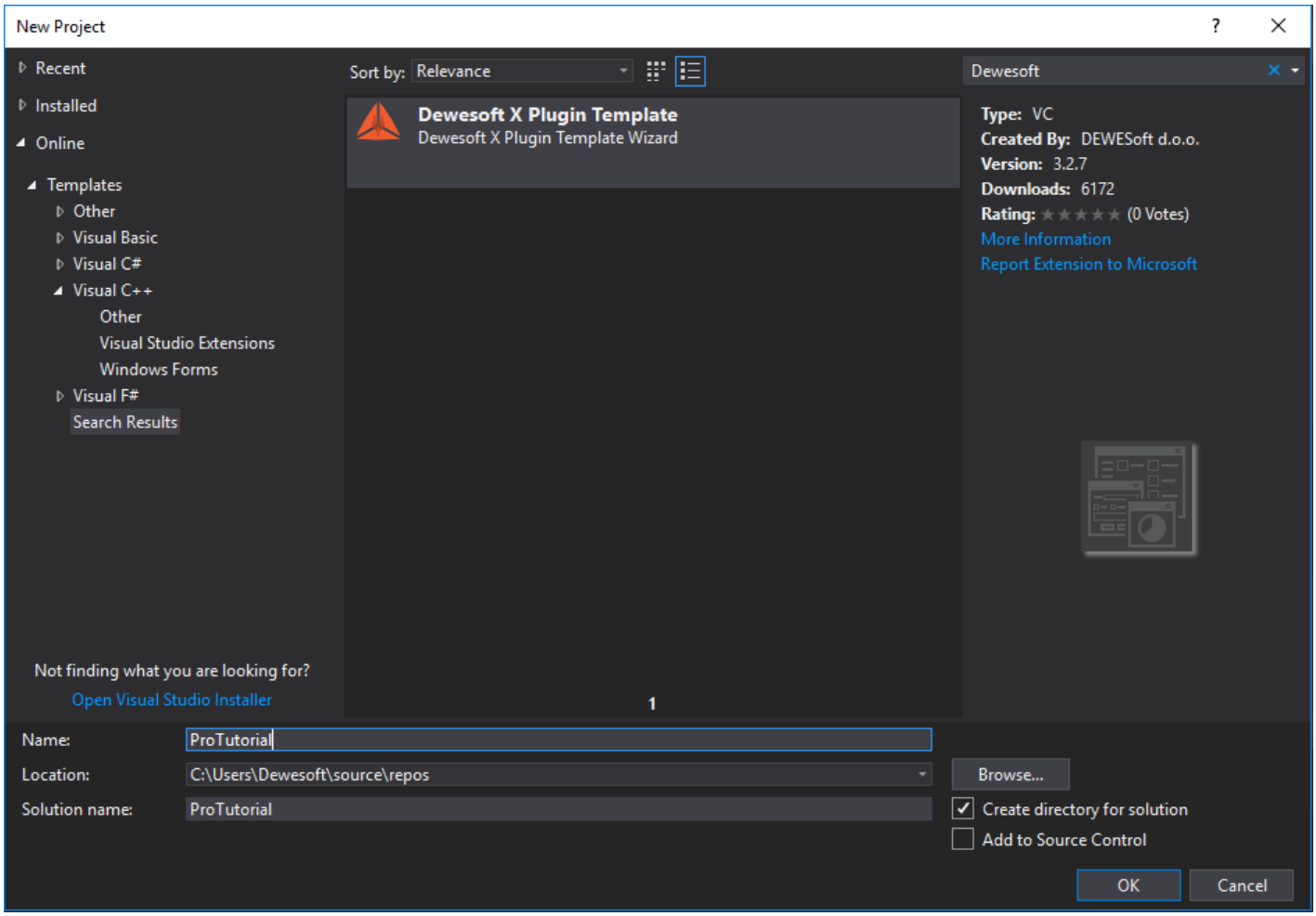
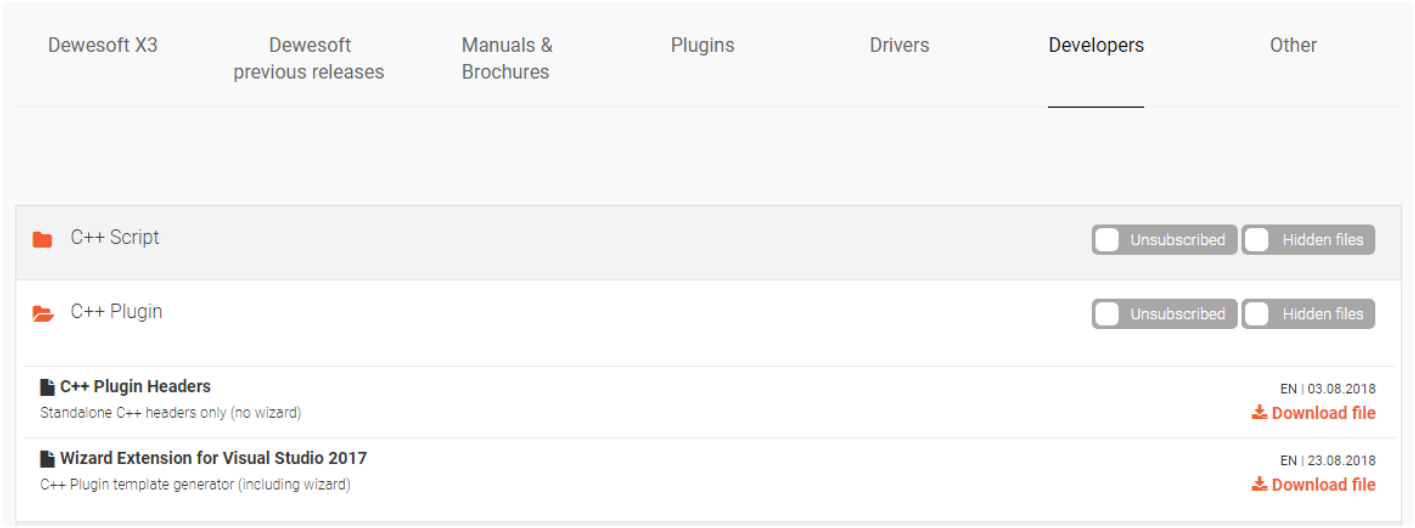
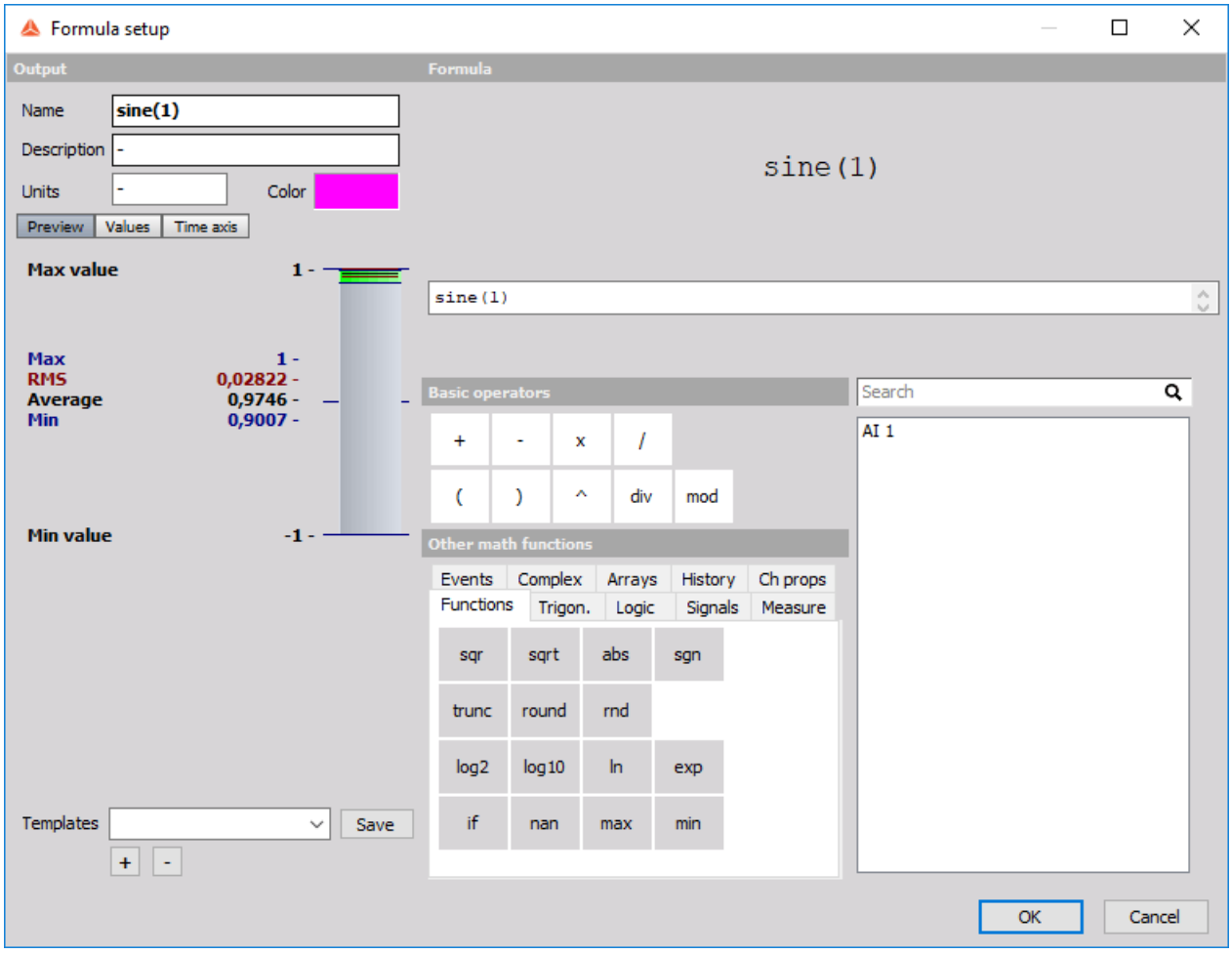


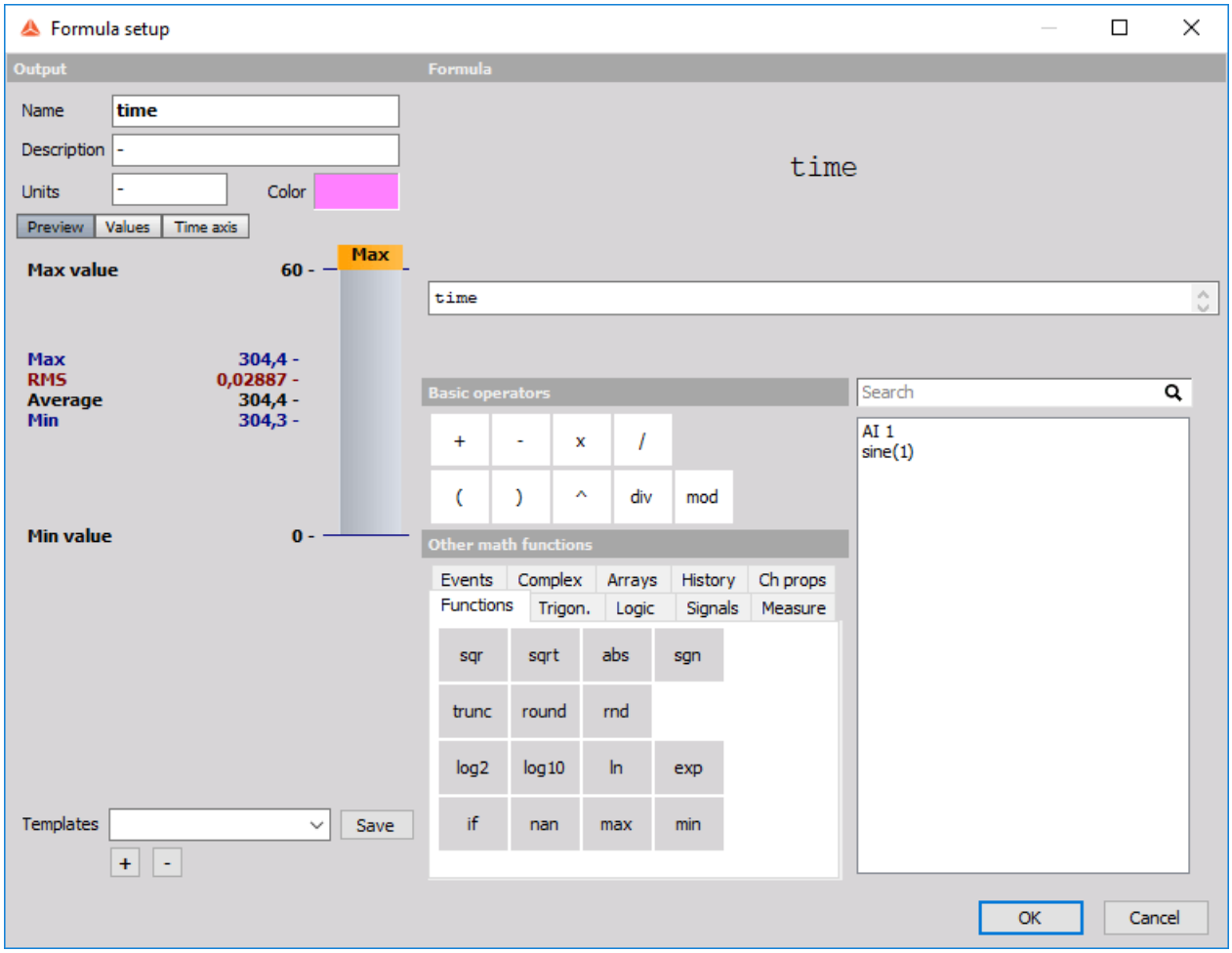
```xml
<?xml version="1.0" encoding="utf-8"?>
<Window xmlns="https://mui.dewesoft.com/schema/1.1">
  <CaptionPanel Title="Latch criteria settings">
    <Grid PaddingLeft="5">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="140"/>
      </Grid.ColumnDefinitions>
      <Grid.RowDefinitions>
        <RowDefinition Height="20"/>
        <RowDefinition Height="20"/>
        <RowDefinition Height="10"/>
        <RowDefinition Height="20"/>
        <RowDefinition Height="20"/>
      </Grid.RowDefinitions>
    </Grid>
  </CaptionPanel>
</Window>
```

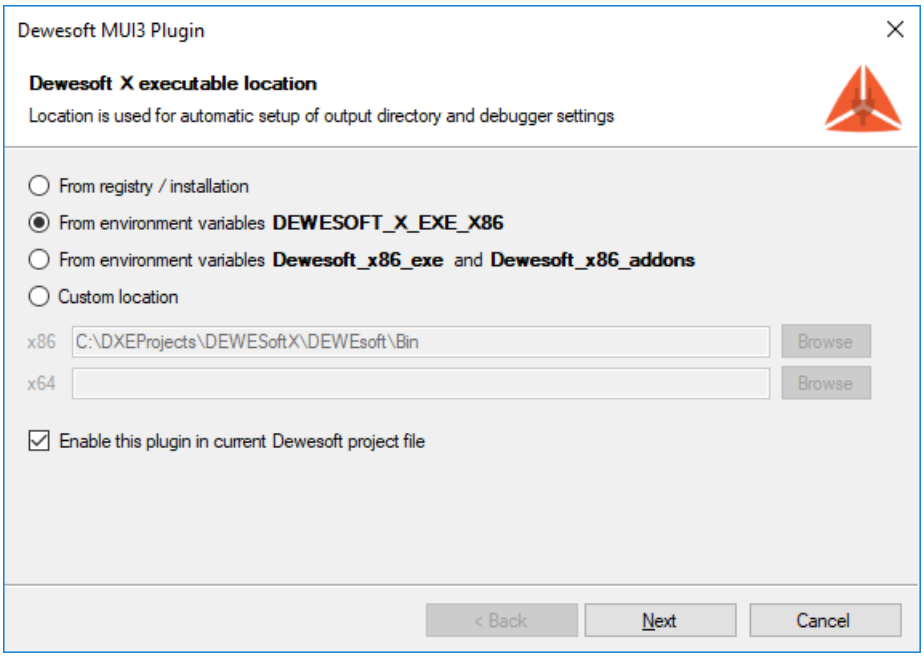
Table of Contents

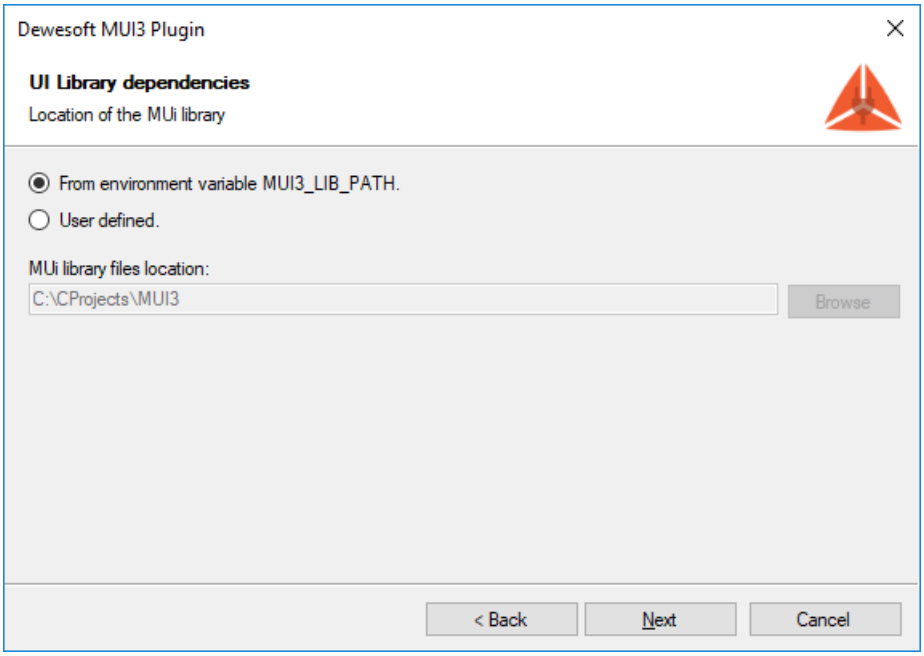

New Project
Recent
Installed
Online
Templates
Other
Visual Basic
Visual C#
Visual C++
Other
Visual Studio Extensions
Windows Forms
Visual F#
Search Results
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Relevance
Dewesoft X Plugin Template
Dewesoft X Plugin Template Wizard
1
Dewesoft
Type: VC
Created By: DEWESoft d.o.o.
Version: 3.2.7
Downloads: 6172
Rating: (0 Votes)
More Information
Report Extension to Microsoft
Name: ProTutorial
Location: C:\Users\Dewesoft\source\repos
Solution name: ProTutorial
Browse...
Create directory for solution
Add to Source Control
OK
Cancel

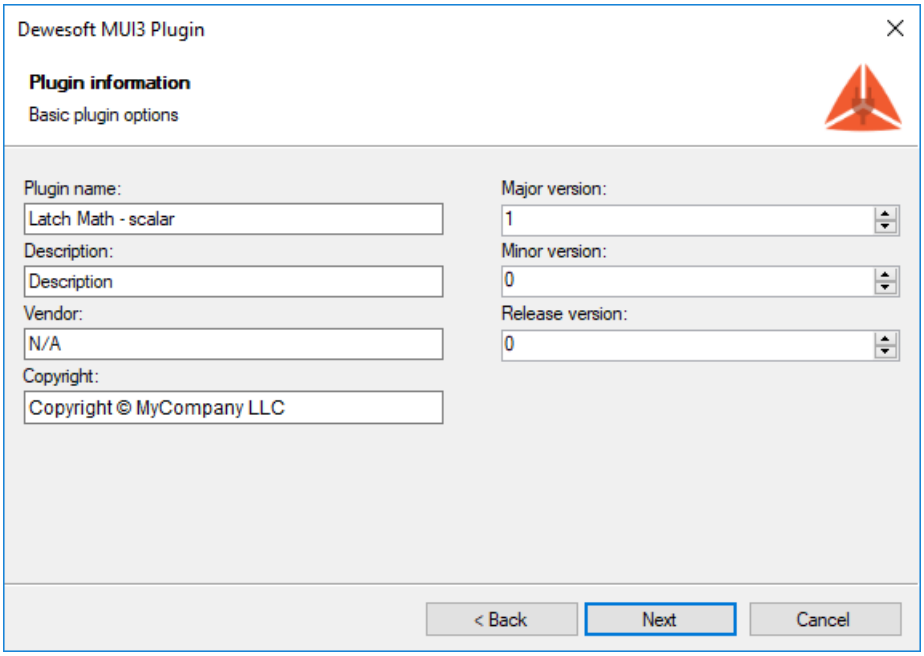

Dewesoft X3
Dewesoft previous releases
Manuals & Brochures
Plugins
Drivers
Developers
Other
C++ Script
Unsubscribed
Hidden files
C++ Plugin
Unsubscribed
Hidden files
C++ Plugin Headers
Standalone C++ headers only (no wizard)
EN | 03.08.2018
Download file
Wizard Extension for Visual Studio 2017
C++ Plugin template generator (including wizard)
EN | 23.08.2018
Download file


Formula setup
Output
Formula
Name
sine(1)
Description
-
Units
-
Color
Preview
Values
Time axis
Max value
1 -
Max
1 -
RMS
0,02822 -
Average
0,9746 -
Min
0,9007 -
Min value
-1 -
sine(1)
sine(1)
Basic operators
+
-
x
/
(
)
^
div
mod
Other math functions
Events
Complex
Arrays
History
Ch props
Functions
Trigon.
Logic
Signals
Measure
sqr
sqrt
abs
sgn
trunc
round
rnd
log2
log10
ln
exp
if
nan
max
min
Search
AI 1
Templates
Save
+
-
OK
Cancel

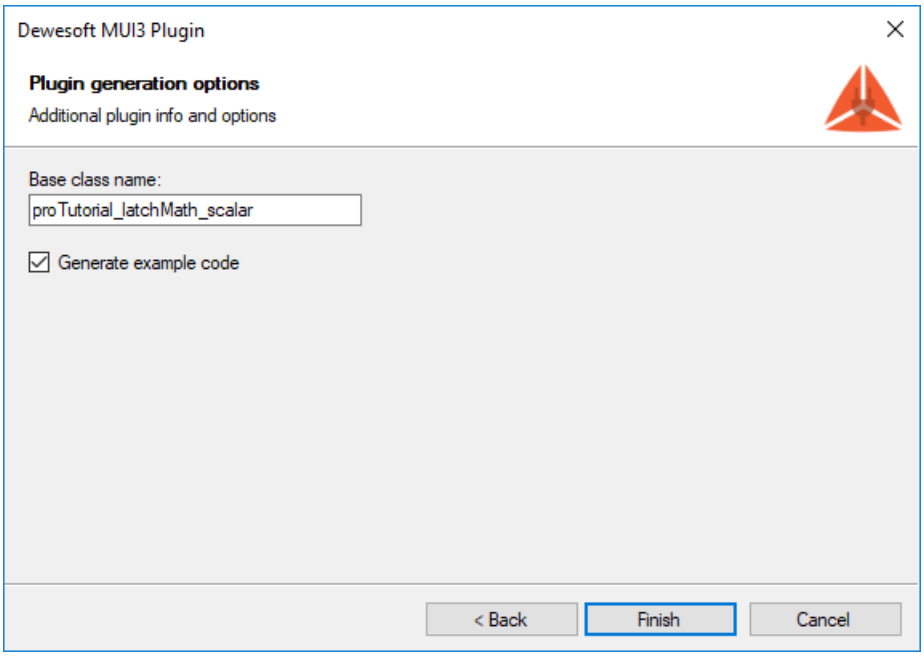

Formula setup
Output
Formula
Name
time
Description
-
Units
-
Color
Preview
Values
Time axis
Max value
60 -
Max
Max
304,4 -
RMS
0,02887 -
Average
304,4 -
Min
304,3 -
Min value
0 -
time
time
Basic operators
+
-
x
/
(
)
^
div
mod
Other math functions
Events
Complex
Arrays
History
Ch props
Functions
Trigon.
Logic
Signals
Measure
sqr
sqrt
abs
sgn
trunc
round
rnd
log2
log10
ln
exp
if
nan
max
min
Search
AI 1
sine(1)
Templates
Save
+
-
OK
Cancel

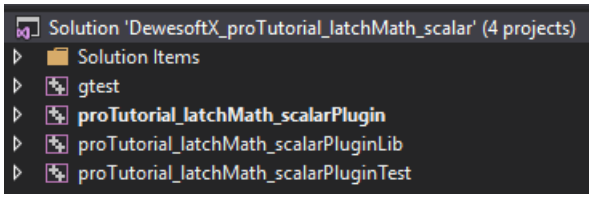

Dewesoft MUI3 Plugin
Dewesoft X executable location
Location is used for automatic setup of output directory and debugger settings
From registry / installation
From environment variables DEWESOFT_X_EXE_X86
From environment variables Dewesoft_x86_exe and Dewesoft_x86_addons
Custom location
x86
C:\DXEProjects\DEWESoftX\DEWEsoft\Bin
Browse
x64
Browse
Enable this plugin in current Dewesoft project file
< Back
Next
Cancel


Dewesoft MUI3 Plugin
UI Library dependencies
Location of the MUi library
From environment variable MUI3_LIB_PATH.
User defined.
MUi library files location:
C:\CProjects\MUI3
Browse
< Back
Next
Cancel

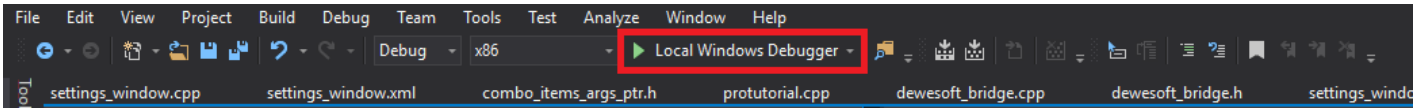

Dewesoft MUI3 Plugin
Plugin information
Basic plugin options
Plugin name:
Latch Math - scalar
Description:
Description
Vendor:
N/A
Copyright:
Copyright © MyCompany LLC
Major version:
1
Minor version:
0
Release version:
0
< Back
Next
Cancel

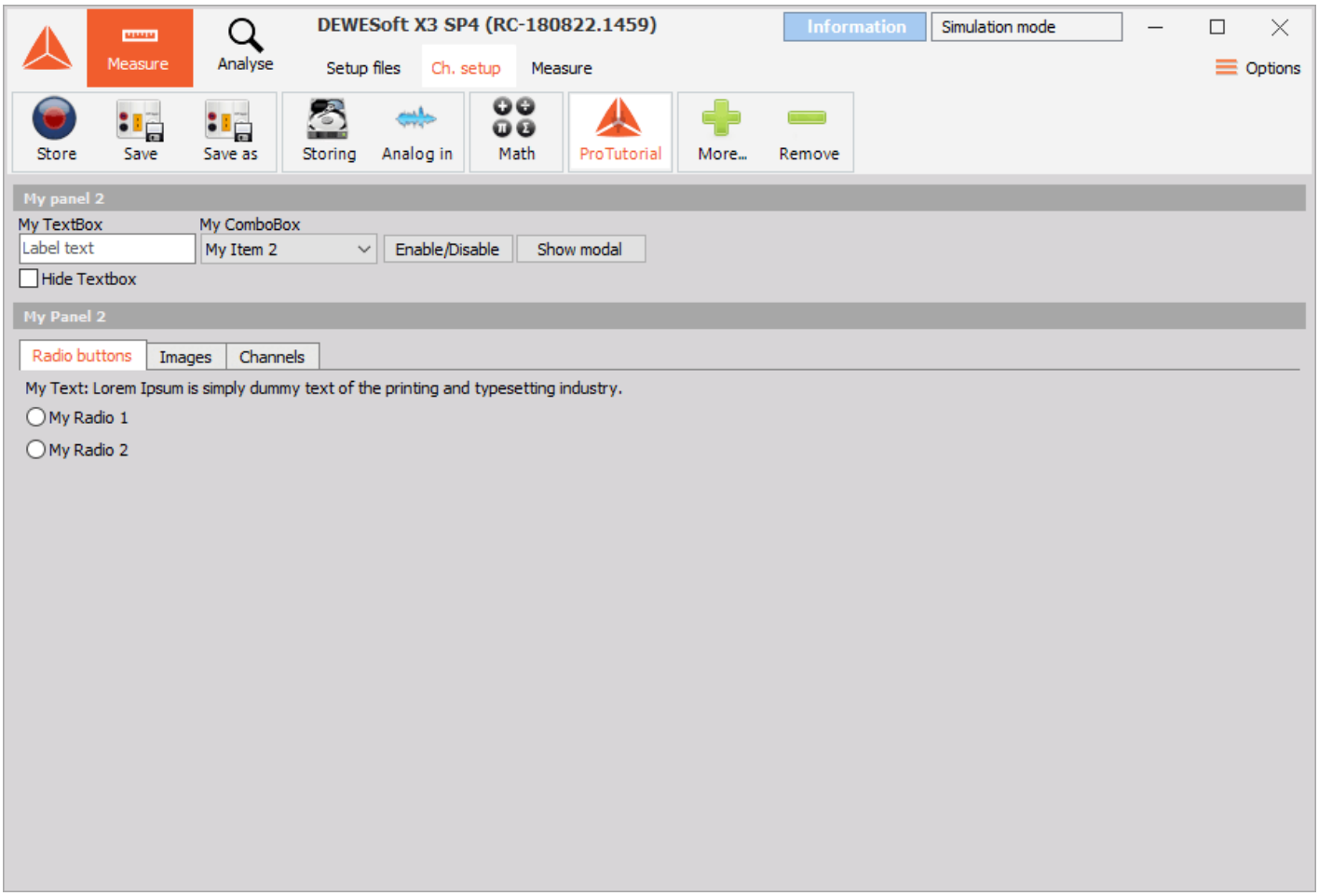
-
-
-
-
-
-
-

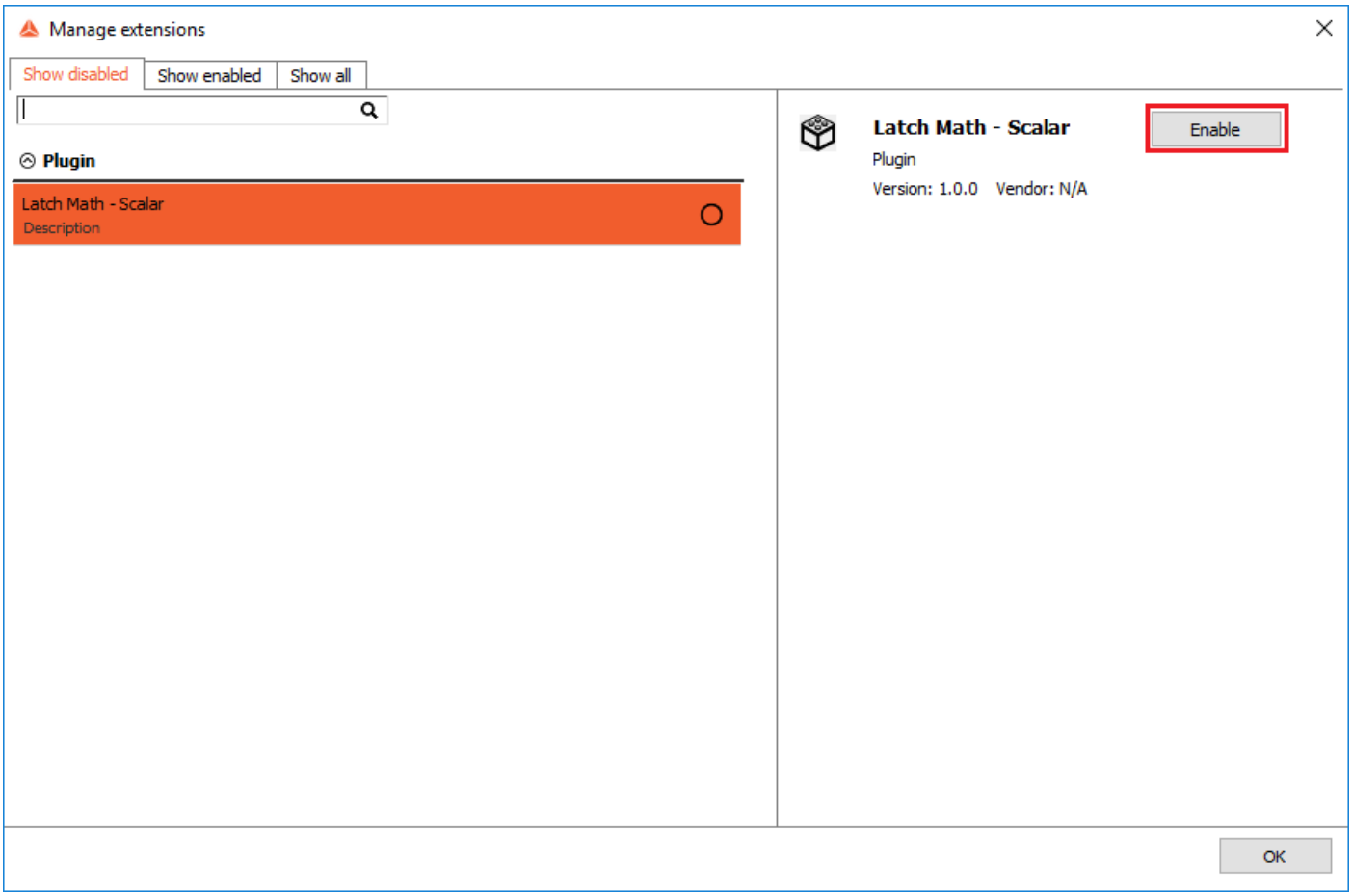

Dewesoft MUI3 Plugin
Plugin generation options
Additional plugin info and options
Base class name:
proTutorial_latchMath_scalar
Generate example code
< Back
Finish
Cancel

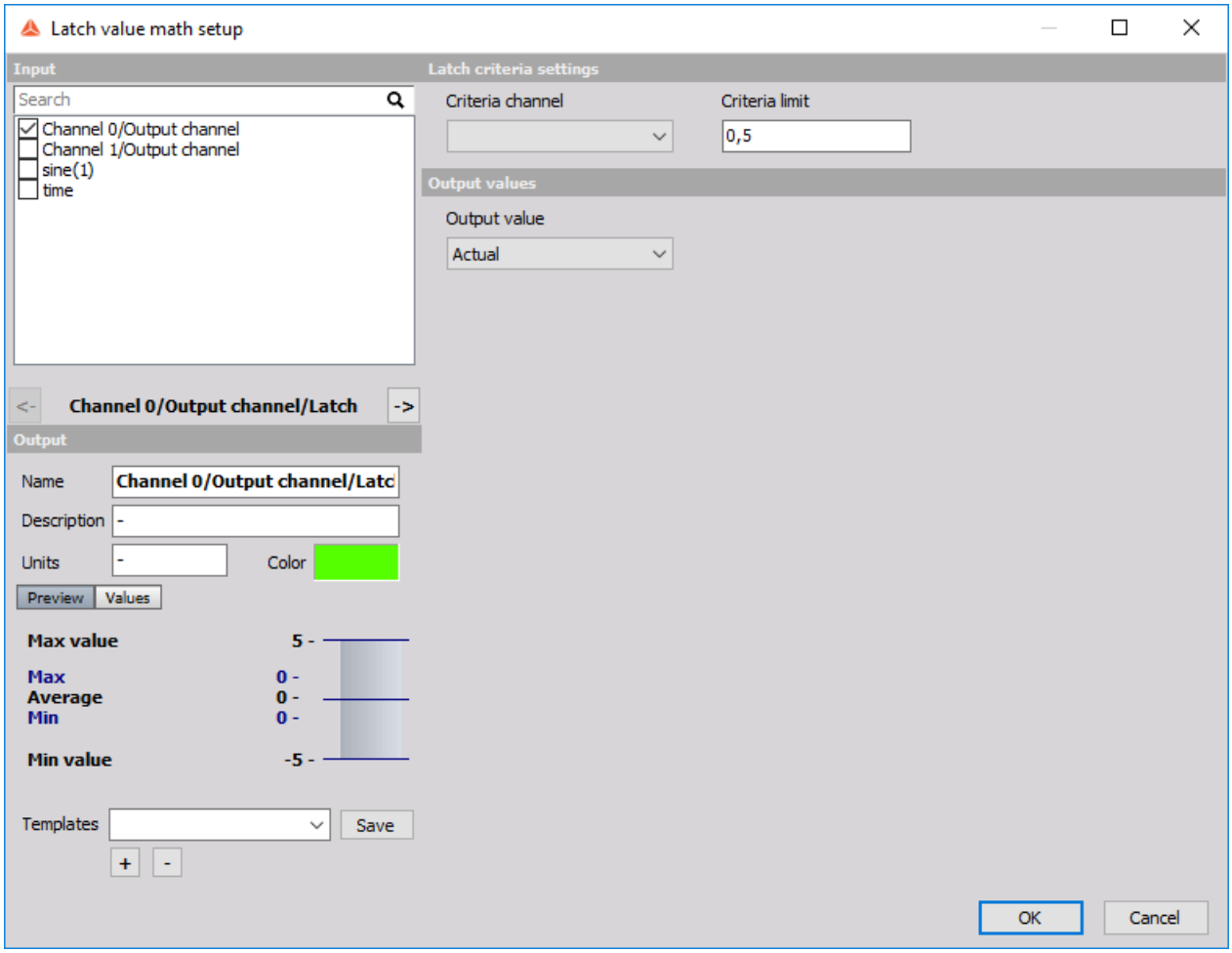

Solution 'DewesoftX_proTutorial_latchMath_scalar' (4 projects)
Solution Items
gtest
proTutorial_latchMath_scalarPlugin
proTutorial_latchMath_scalarPluginLib
proTutorial_latchMath_scalarPluginTest

-
-
-
-

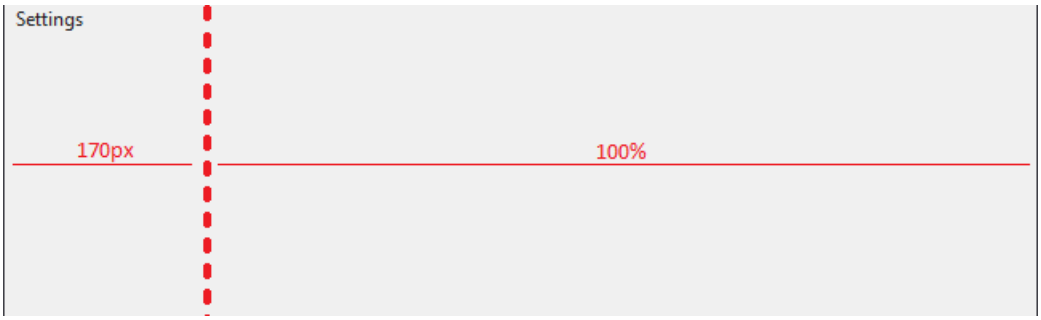

File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug
x86
Local Windows Debugger
settings_window.cpp
settings_window.xml
combo_items_args_ptr.h
protutorial.cpp
dewesoft_bridge.cpp
dewesoft_bridge.h


Measure
Analyse
DEWESoft X3 SP4 (RC-180822.1459)
Information
Simulation mode
Setup files
Ch. setup
Measure
Options
Store
Save
Save as
Storing
Analog in
Math
ProTutorial
More...
Remove
My panel 2
My TextBox
Label text
My ComboBox
My Item 2
Enable/Disable
Show modal
Hide Textbox
My Panel 2
Radio buttons
Images
Channels
My Text: Lorem Ipsum is simply dummy text of the printing and typesetting industry.
My Radio 1
My Radio 2


Manage extensions
Show disabled
Show enabled
Show all
Plugin
Latch Math - Scalar
Description
Latch Math - Scalar
Plugin
Version: 1.0.0 Vendor: N/A
Enable
OK


Latch value math setup
Input
Search
Channel 0/Output channel
Channel 1/Output channel
sine(1)
time
<-
Channel 0/Output channel/Latch
->
Output
Name
Channel 0/Output channel/Latc
Description
-
Units
-
Color
Preview
Values
Max value
5 -
Max
0 -
Average
0 -
Min
0 -
Min value
-5 -
Templates
Save
+
-
Latch criteria settings
Criteria channel
Criteria limit
0,5
Output values
Output value
Actual
OK
Cancel

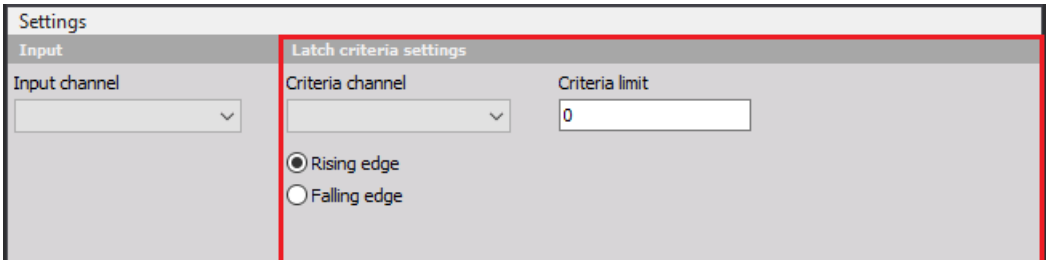
```
<?xml version="1.0" encoding="utf-8"?>
<Window xmlns="https://mui.dewesoft.com/schema/1.1">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width ="170"/>
            <ColumnDefinition Width ="100%"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="100%"/>
        </Grid.RowDefinitions>
```

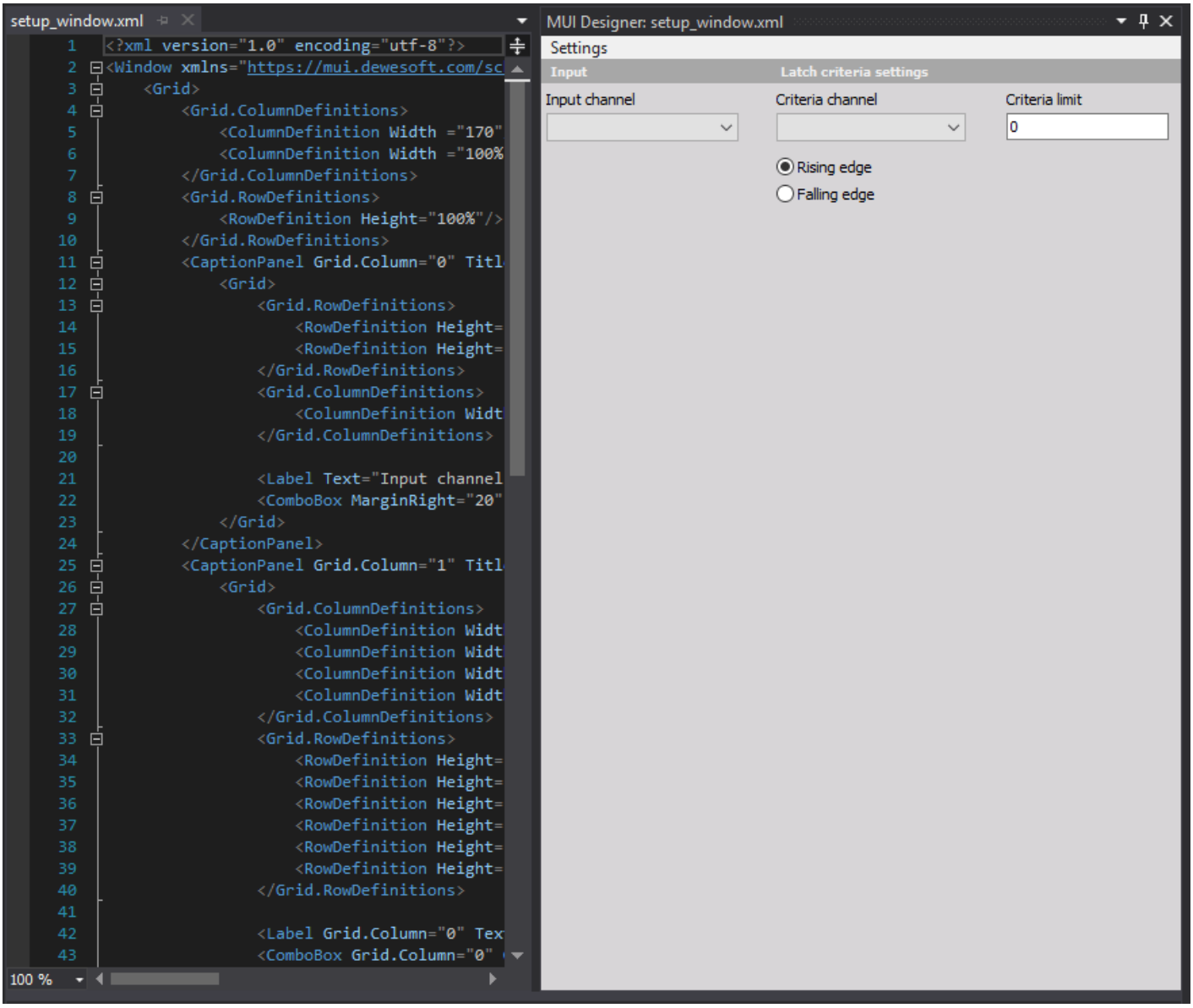


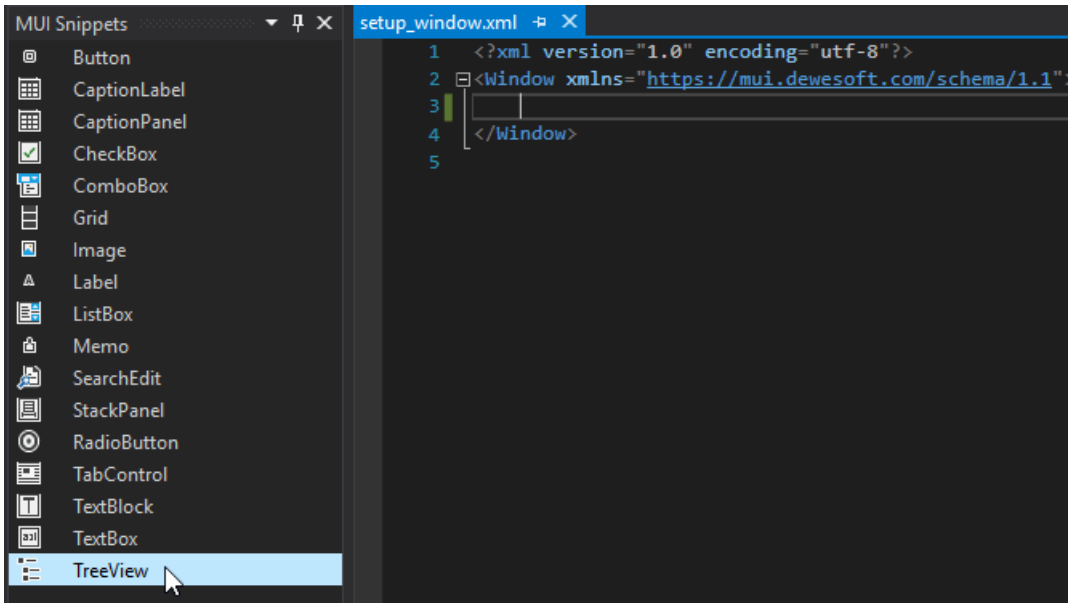
```
<CaptionPanel Grid.Column="0" Title="Input">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="20"/>
            <RowDefinition Height="100%"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100%"/>
        </Grid.ColumnDefinitions>

        <Label Text="Input channel" Grid.Row="0" />
        <ComboBox MarginRight="20" Grid.Row="1" Name="inputChannelName" />
    </Grid>
</CaptionPanel>
```

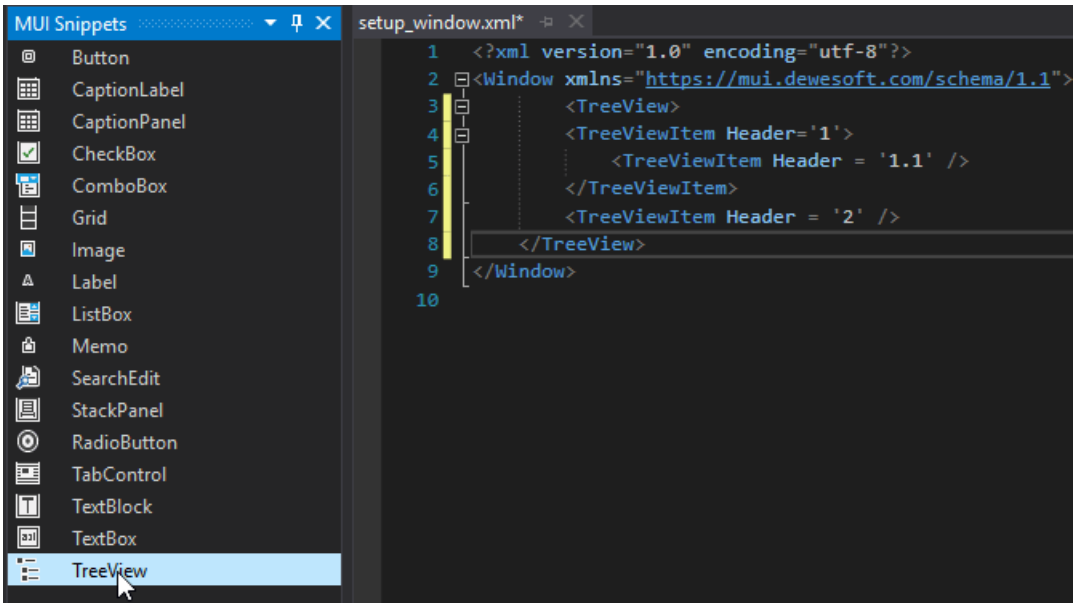
`Grid.Column="0"`

```
        <CaptionPanel Grid.Column="1" Title="Latch criteria settings">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="140"/>
                    <ColumnDefinition Width="30"/>
                    <ColumnDefinition Width="120"/>
                    <ColumnDefinition Width="100%"/>
                </Grid.ColumnDefinitions>
                <Grid.RowDefinitions>
                    <RowDefinition Height="20"/>
                    <RowDefinition Height="20"/>
                    <RowDefinition Height="10"/>
                    <RowDefinition Height="20"/>
                    <RowDefinition Height="20"/>
                    <RowDefinition Height="100%"/>
                </Grid.RowDefinitions>

                <Label Grid.Column="0" Text="Criteria channel"/>
                <ComboBox Grid.Column="0" Grid.Row="1" Name="criteriaChannelName" />
                <Label Grid.Column="2" Text="Criteria limit" />
                <TextBox Grid.Column="2" Grid.Row="1" Name="latchCriteria" Text="0"></TextBox>
                <RadioButton Grid.Column="0" Grid.Row="3" Name="risingEdge" Label="Rising edge"
IsChecked="True" />
                <RadioButton Grid.Column="0" Grid.Row="4" Name="fallingEdge" Label="Falling edge" />
            </Grid>
        </CaptionPanel>
    </Grid>
</Window>
```




setup_window.xml
<?xml version="1.0" encoding="utf-8"?>
<Window xmlns="https://mui.dewesoft.com/sc
<Grid>
<Grid.ColumnDefinitions>
<ColumnDefinition Width ="170"
<ColumnDefinition Width ="100%
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="100%"/>
</Grid.RowDefinitions>
<CaptionPanel Grid.Column="0" Titl
<Grid>
<Grid.RowDefinitions>
<RowDefinition Height=
<RowDefinition Height=
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
<ColumnDefinition Widt
</Grid.ColumnDefinitions>
<Label Text="Input channel
<ComboBox MarginRight="20"
</Grid>
</CaptionPanel>
<CaptionPanel Grid.Column="1" Titl
<Grid>
<Grid.ColumnDefinitions>
<ColumnDefinition Widt
<ColumnDefinition Widt
<ColumnDefinition Widt
<ColumnDefinition Widt
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height=
<RowDefinition Height=
<RowDefinition Height=
<RowDefinition Height=
<RowDefinition Height=
<RowDefinition Height=
</Grid.RowDefinitions>
<Label Grid.Column="0" Tex
<ComboBox Grid.Column="0"
100 %
MUI Designer: setup_window.xml
Settings
Input
Latch criteria settings
Input channel
Criteria channel
Criteria limit
0
Rising edge
Falling edge


MUI Snippets
Button
CaptionLabel
CaptionPanel
CheckBox
ComboBox
Grid
Image
Label
ListBox
Memo
SearchEdit
StackPanel
RadioButton
TabControl
TextBlock
TextBox
TreeView
setup_window.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <Window xmlns="https://mui.dewesoft.com/schema/1.1"
3
4 </Window>
5


MUI Snippets
Button
CaptionLabel
CaptionPanel
CheckBox
ComboBox
Grid
Image
Label
ListBox
Memo
SearchEdit
StackPanel
RadioButton
TabControl
TextBlock
TextBox
TreeView
setup_window.xml*
1 <?xml version="1.0" encoding="utf-8"?>
2 <Window xmlns="https://mui.dewesoft.com/schema/1.1">
3 <TreeView>
4 <TreeViewItem Header='1'>
5 <TreeViewItem Header = '1.1' />
6 </TreeViewItem>
7 <TreeViewItem Header = '2' />
8 </TreeView>
9 </Window>
10

```h
#pragma once
#include <mui/ds_window.h>
#include <mui/controls.h>
#include <mui/layout.h>

class DewesoftBridge;

class SetupWindow : public Dewesoft::MUI::DSWindow
{
public:
    SetupWindow(Dewesoft::MUI::WindowPtr ui, DewesoftBridge& bridge);

private:
    DewesoftBridge& bridge;
};
```

```cpp
#include "StdAfx.h"
#include "setup_window.h"
#include "dewesoft_bridge.h"
#include <thread>
#include <chrono>
#include <regex>

using namespace Dewesoft::MUI;
using namespace Dewesoft::RT::Core;

SetupWindow::SetupWindow(WindowPtr ui, DewesoftBridge& bridge)
    : DSWindow(ui, "ui/setup_window.xml")
    , bridge(bridge)
{
}
```

```
private:
    DewesoftBridge& bridge;

    Dewesoft::MUI::ComboBox criteriaChannelCBox;
    Dewesoft::MUI::ComboBox inputChannelCBox;
    Dewesoft::MUI::TextBox latchCriteriaTBox;
    Dewesoft::MUI::RadioButton risingEdgeRButton;
    Dewesoft::MUI::RadioButton fallingEdgeRButton;
```

_____

`Connect`

```
SetupWindow::SetupWindow(WindowPtr ui, DewesoftBridge& bridge)
    : DSWindow(ui, "ui/setup_window.xml")
    , bridge(bridge)
{
    criteriaChannelCBox = ComboBox::Connect(ui, "criteriaChannelName");
    inputChannelCBox = ComboBox::Connect(ui, "inputChannelName");
    latchCriteriaTBox = TextBox::Connect(ui, "latchCriteria");
    risingEdgeRButton = RadioButton::Connect(ui, "risingEdge");
    fallingEdgeRButton = RadioButton::Connect(ui, "fallingEdge");
}
```

`app`

`getSyncChannels()` `app`

```
// dewesoft_bridge.h
// this method needs to be public
std::vector<IChannelPtr> getSyncChannels();

// dewesoft_bridge.cpp
std::vector<IChannelPtr> DewesoftBridge::getSyncChannels()
{
    std::vector<IChannelPtr> allSyncChannels;

    app->Data->BuildChannelList();
    IChannelListPtr channels = app->Data->UsedChannels;

    for (int i = 0; i < channels->Count; i++)
        if (!channels->GetItem(i)->Async)
            allSyncChannels.push_back(channels->GetItem(i));

    return allSyncChannels;
}
```

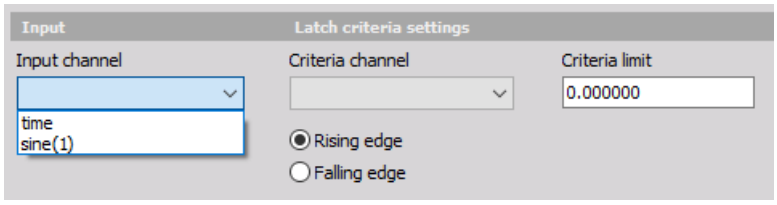
`getSyncChannels()`

`criteriaChannelCBox` `inputChannelCBox`

`void addChannelsToCBox();`

```
// setup_window.h
void addChannelsToCBox();
```

```cpp
// setup_window.cpp
SetupWindow::SetupWindow(WindowPtr ui, DewesoftBridge& bridge)
    : DSWindow(ui, "ui/setup_window.xml")
    , bridge(bridge)
{
    // member initialization
    // ...
    addChannelsToCBox();
}

void SetupWindow::addChannelsToCBox()
{
    criteriaChannelCBox.clear();
    inputChannelCBox.clear();
    std::vector<IChannelPtr> allChannels = bridge.getSyncChannels();

    for (int i = 0; i < allChannels.size(); i++)
    {
        std::string channelName = allChannels[i]->Name;
        criteriaChannelCBox.addItem(channelName);
        inputChannelCBox.addItem(channelName);
    }
}
```



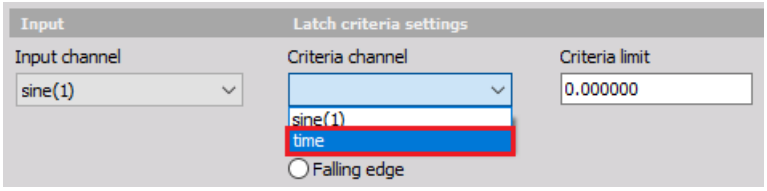
`void(ComponentType& sender, EventArgsType& args)`

```
private
    // ...
    void onCriteriaChannelChanged(Dewesoft::MUI::ComboBox& cBox, Dewesoft::MUI::EventArgs& args);
    void onInputChannelChanged(Dewesoft::MUI::ComboBox& cBox, Dewesoft::MUI::EventArgs& args);
    void onEditTextChanged(Dewesoft::MUI::TextBox& editBox, Dewesoft::MUI::EventArgs& args);
    void onRadioGroupChanged(Dewesoft::MUI::RadioButton& radioGroup, Dewesoft::MUI::EventArgs&
args);
}
```

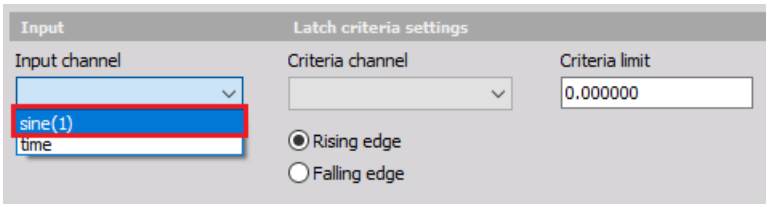
`+=`

`-=`

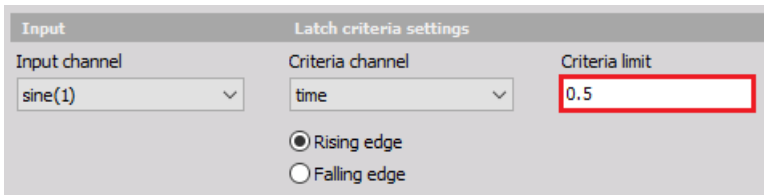
```
SetupWindow::SetupWindow(WindowPtr ui, DewesoftBridge& bridge)
    : DSWindow(ui, "ui/setup_window.xml")
    , bridge(bridge)
{
    // member initialisation
    // adding channel names to ComboBoxes
    // ...
    criteriaChannelCBox.OnChange += event(&SetupWindow::onCriteriaChannelChanged);
    inputChannelCBox.OnChange += event(&SetupWindow::onInputChannelChanged);
    latchCriteriaTBox.OnTextChanged += event(&SetupWindow::onEditTextChanged);
    risingEdgeRButton.OnCheckedChanged += event(&SetupWindow::onRadioGroupChanged);
    fallingEdgeRButton.OnCheckedChanged += event(&SetupWindow::onRadioGroupChanged);
}
```



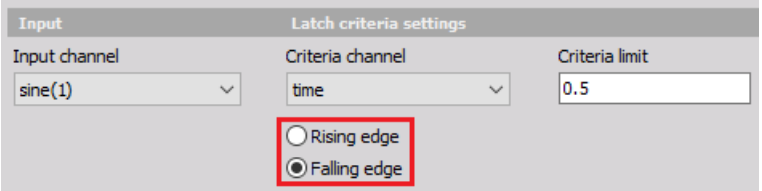

```
void SetupWindow::onCriteriaChannelChanged(Dewesoft::MUI::ComboBox& cBox,
Dewesoft::MUI::EventArgs& args)
{
    bridge.setCriteriaChannelName(cBox.getSelectedItem());
}
```




```
void SetupWindow::onInputChannelChanged(Dewesoft::MUI::ComboBox& cBox,
Dewesoft::MUI::EventArgs& args)
{
    bridge.setInputChannelName(cBox.getSelectedItem());
}
```




```
void SetupWindow::onEditTextChanged(Dewesoft::MUI::TextBox& editBox, Dewesoft::MUI::EventArgs&
args)
{
    bridge.setCriteriaLimit(stof(editBox.getText().toStdString()));
}
```




```
void SetupWindow::onRadioGroupChanged(RadioButton& radioGroup, EventArgs& args)
{
   if (SetupWindow::fallingEdgeRButton.getIsChecked())
      bridge.setEdgeType(FallingEdge);
   else
      bridge.setEdgeType(RisingEdge);
}
```

```
enum edgeTypes
{
    RisingEdge = 0,
    FallingEdge = 1
};

class DewesoftBridge
{
public:
    // ...

    /* Declarations of methods we added*/
    std::vector<IChannelPtr> getSyncChannels();

    void setCriteriaChannelName(std::string name);
    std::string getCriteriaChannelName();

    void setInputChannelName(std::string name);
    std::string getInputChannelName();

    void setCriteriaChannel(std::string channelName);
    IChannelPtr getCriteriaChannel();

    void setInputChannel(std::string channelName);
    IChannelPtr getInputChannel();

    void setCriteriaLimit(float threshold);
    float getCriteriaLimit();

    void setEdgeType(edgeTypes type);
    edgeTypes getEdgeType();

private:
    // ...

    /* Declarations of methods and variables we added */
    std::string inputChannelName = "";
    std::string criteriaChannelName = "";
    IChannelPtr inputChannel;
    IChannelPtr criteriaChannel;

    float criteriaLimit = 0;
    edgeTypes edgeType = RisingEdge;

    int64_t lastPosChecked;
};
```

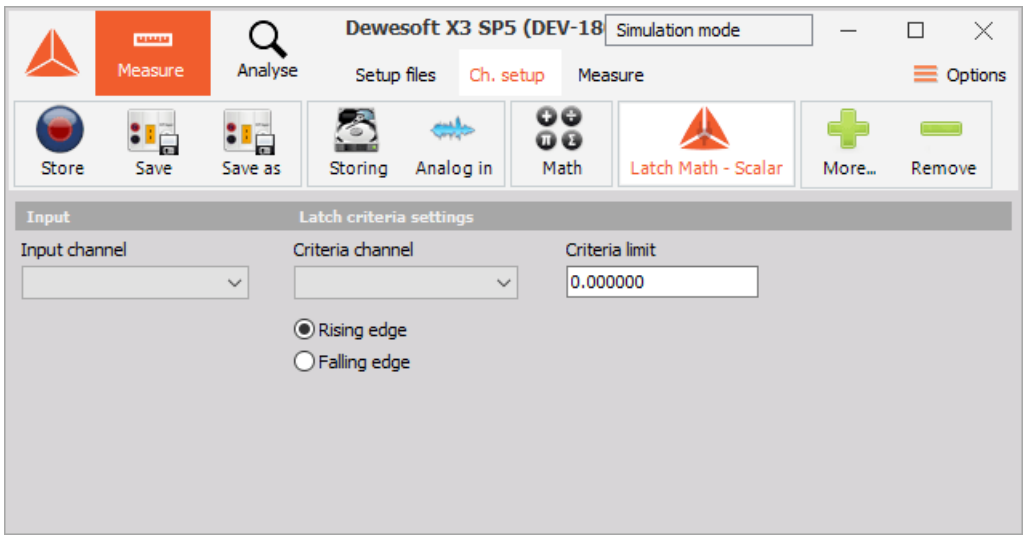
```
void DewesoftBridge::setCriteriaChannelName(std::string name)
{
    this->criteriaChannelName = name;
}
std::string DewesoftBridge::getCriteriaChannelName()
{
    return this->criteriaChannelName;
}

void DewesoftBridge::setInputChannelName(std::string name)
{
    this->inputChannelName = name;
}
std::string DewesoftBridge::getInputChannelName()
{
    return this->inputChannelName;
}

void DewesoftBridge::setCriteriaChannel(std::string channelName)
{
    criteriaChannel = app->Data->FindChannel(channelName.c_str());
}
IChannelPtr DewesoftBridge::getCriteriaChannel()
{
    return this->criteriaChannel;
}

void DewesoftBridge::setInputChannel(std::string channelName)
{
    this->inputChannel = app->Data->FindChannel(channelName.c_str());
}
IChannelPtr DewesoftBridge::getInputChannel()
{
    return this->inputChannel;
}

void DewesoftBridge::setCriteriaLimit(float threshold)
{
    this->criteriaLimit = threshold;
}
float DewesoftBridge::getCriteriaLimit()
{
    return this->criteriaLimit;
}

void DewesoftBridge::setEdgeType(edgeTypes type)
{
    this->edgeType = type;
}
edgeTypes DewesoftBridge::getEdgeType()
{
    return this->edgeType;
}
```

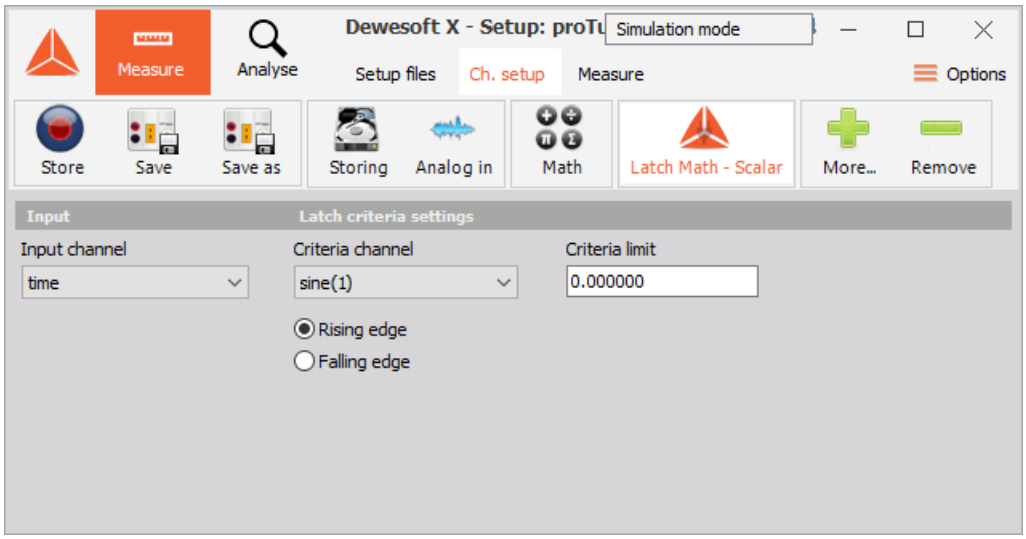
`void DewesoftBridge::mountChannels()`

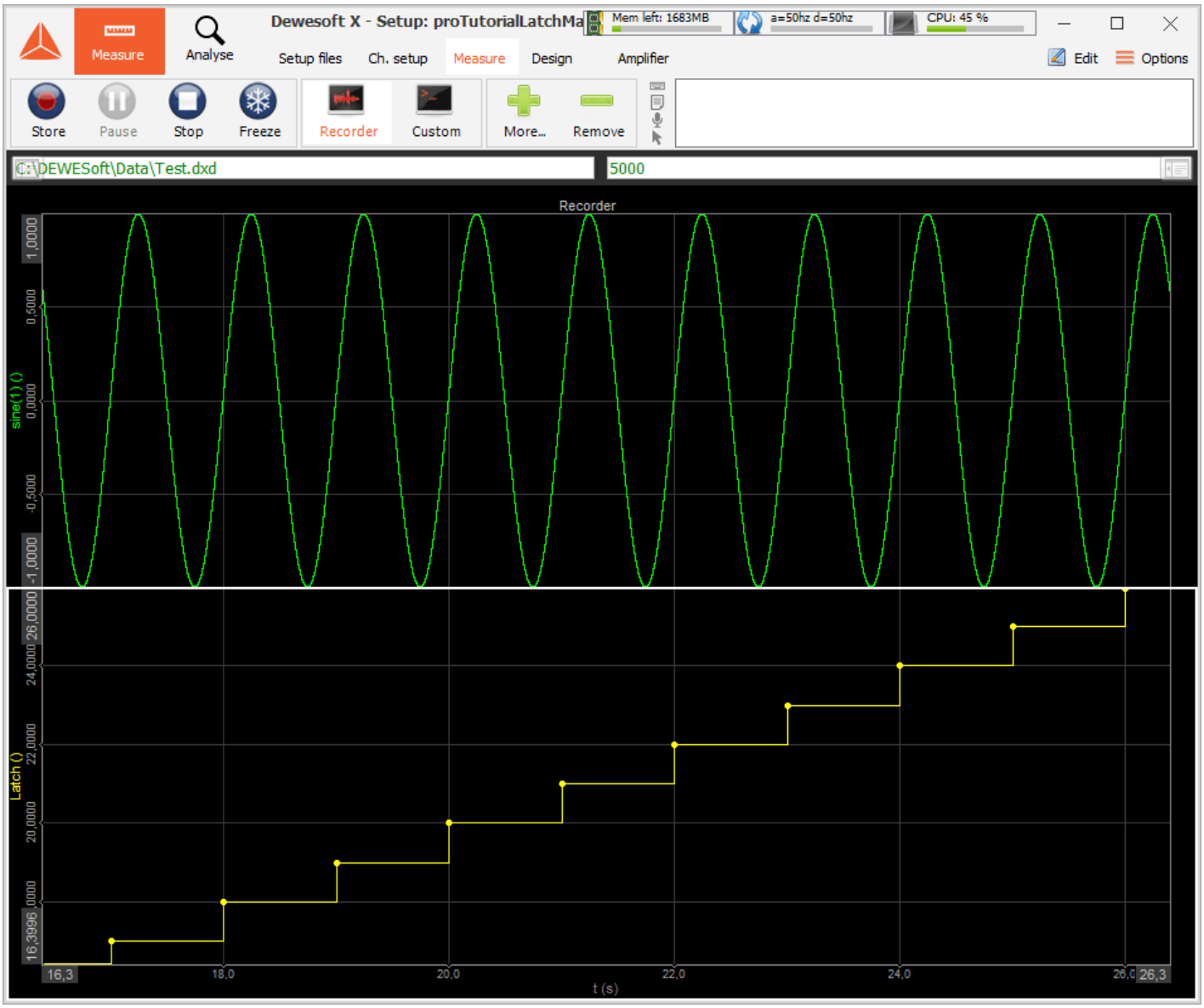
```
void DewesoftBridge::mountChannels()
{
    const std::vector<int> chIndexVector = {1};

    outputChannel = pluginGroup->MountChannelEx(pluginGuid, long(chIndexVector.size()),
fromStdVec(chIndexVector));

    outputChannel->SetDataType(long(ChannelDataType::Single));
    outputChannel->Name = "Latch";
    outputChannel->Unit_ = "";
    outputChannel->SetAsync(true);
    outputChannel->ExpectedAsyncRate = app->Data->GetSampleRateEx();
    outputChannel->Used = true;
}
```

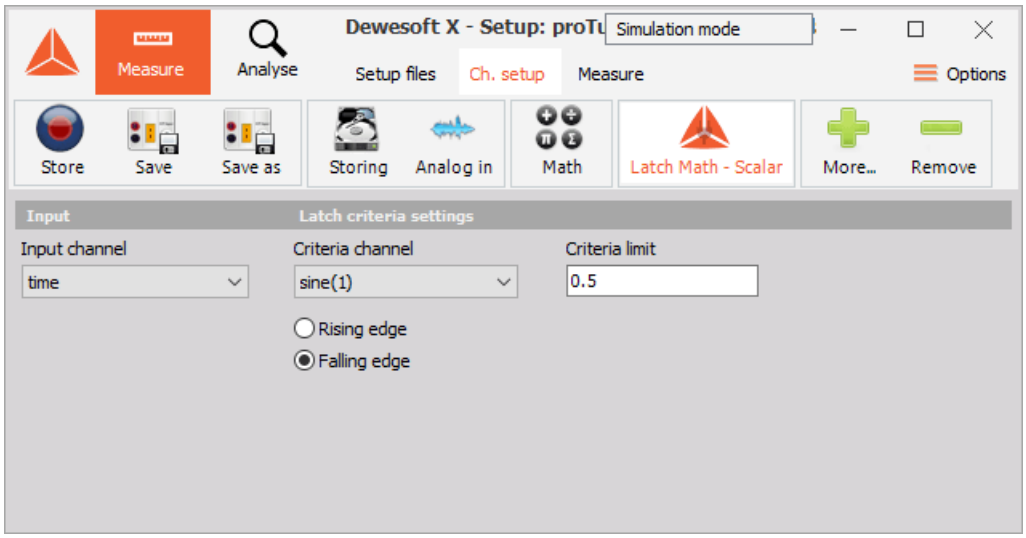
```
STDMETHODIMP DewesoftBridge::onStartData()
{
    setInputChannel(inputChannelName.c_str());
    setCriteriaChannel(criteriaChannelName.c_str());
    lastPosChecked = 0;

    return S_OK;
}
```

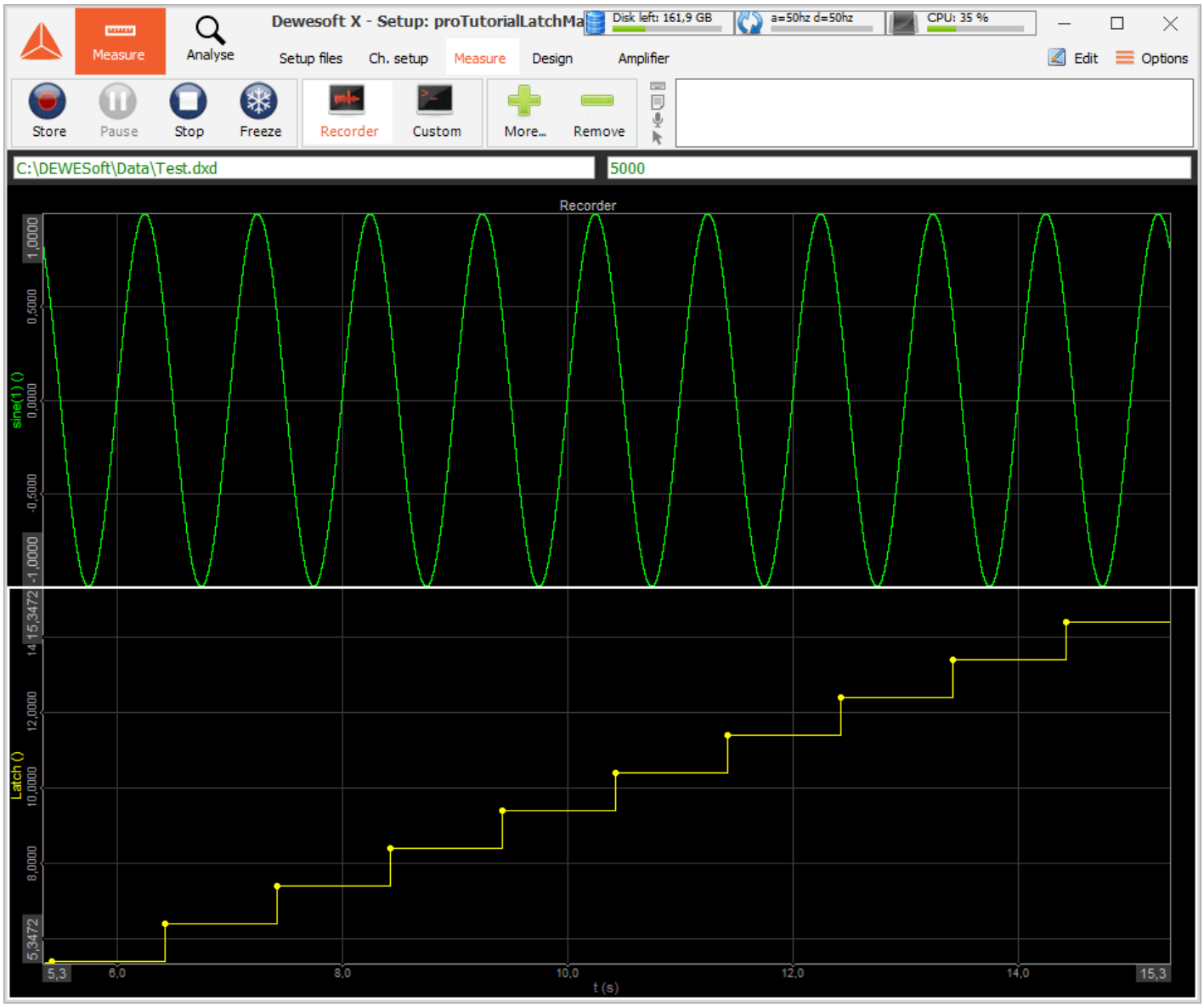
```
STDMETHODIMP DewesoftBridge::onGetData()
{
    if (!inputChannel || !criteriaChannel)
        return S_OK;

    // calculate the size of blocks in both channels
    int blockSizeCriteriaChannel =
        (criteriaChannel->DBPos - (lastPosChecked % criteriaChannel->DBBufSize) + criteriaChannel->DBBufSize) % criteriaChannel->DBBufSize;
    int blockSizeInputChannel =
        (inputChannel->DBPos - (lastPosChecked % inputChannel->DBBufSize) + inputChannel->DBBufSize) % inputChannel->DBBufSize;

    int minBlockSize = min(blockSizeCriteriaChannel, blockSizeInputChannel);

    for (int i = 0; i < minBlockSize - 1; i++)
    {
        float currentSampleCriteriaChannel = criteriaChannel->DBValues[lastPosChecked % criteriaChannel->DBBufSize];
        float nextSampleCriteriaChannel = criteriaChannel->DBValues[(lastPosChecked + 1) % criteriaChannel->DBBufSize];

        // check if the two samples from Criteria channel are on different sides of Latch criteria
        bool crossedRisingEdgeCriteria = currentSampleCriteriaChannel <= criteriaLimit && nextSampleCriteriaChannel >= criteriaLimit;
        bool crossedFallingEdgeCriteria = currentSampleCriteriaChannel >= criteriaLimit && nextSampleCriteriaChannel <= criteriaLimit;

        // if user set the type of edge to rising edge and the Criteria channel crossed it
        // or user set the type of edge to falling edge and the Criteria channel crossed it
        if ((crossedFallingEdgeCriteria && edgeType == FallingEdge) || (crossedRisingEdgeCriteria && edgeType == RisingEdge))
        {
            // add the value of the next sample from Input channel to the Latched channel
            float value = inputChannel->DBValues[(lastPosChecked + 1) % inputChannel->DBBufSize];
            outputChannel->AddAsyncSingleSample(value, (lastPosChecked + 1) / app->Data->SampleRateEx);
        }
        lastPosChecked++;
    }
    return S_OK;
}
```


Measure
Analyse
Dewesoft X3 SP5 (DEV-18
Simulation mode
Setup files
Ch. setup
Measure
Options
Store
Save
Save as
Storing
Analog in
Math
Latch Math - Scalar
More...
Remove
Input
Latch criteria settings
Input channel
Criteria channel
Criteria limit
0.000000
Rising edge
Falling edge


Measure
Analyse
Dewesoft X - Setup: proTu
Simulation mode
Setup files
Ch. setup
Measure
Options
Store
Save
Save as
Storing
Analog in
Math
Latch Math - Scalar
More...
Remove
Input
Latch criteria settings
Input channel
time
Criteria channel
sine(1)
Criteria limit
0.000000
Rising edge
Falling edge

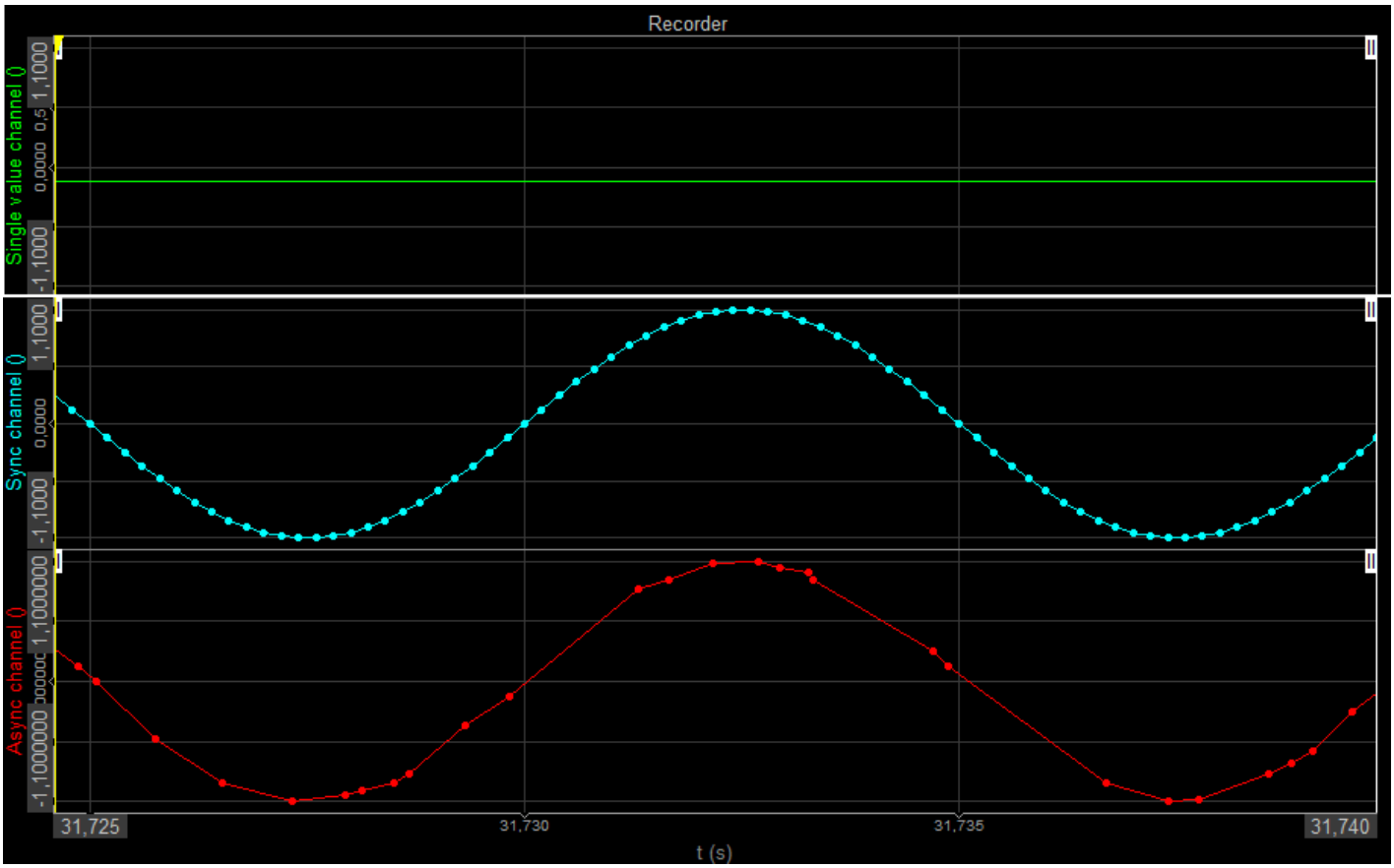

Dewesoft X - Setup: proTutorialLatchMa
Mem left: 1683MB
a=50hz d=50hz
CPU: 45 %
Measure
Analyse
Setup files
Ch. setup
Measure
Design
Amplifier
Edit
Options
Store
Pause
Stop
Freeze
Recorder
Custom
More...
Remove
C:\DEWESoft\Data\Test.dxd
5000
Recorder
sine(1) ()
1,0000
0,5000
0,0000
-0,5000
-1,0000
Latch ()
26,0000
24,0000
22,0000
20,0000
16,3996,0000
16,3
18,0
20,0
22,0
24,0
26,0
26,3
t (s)

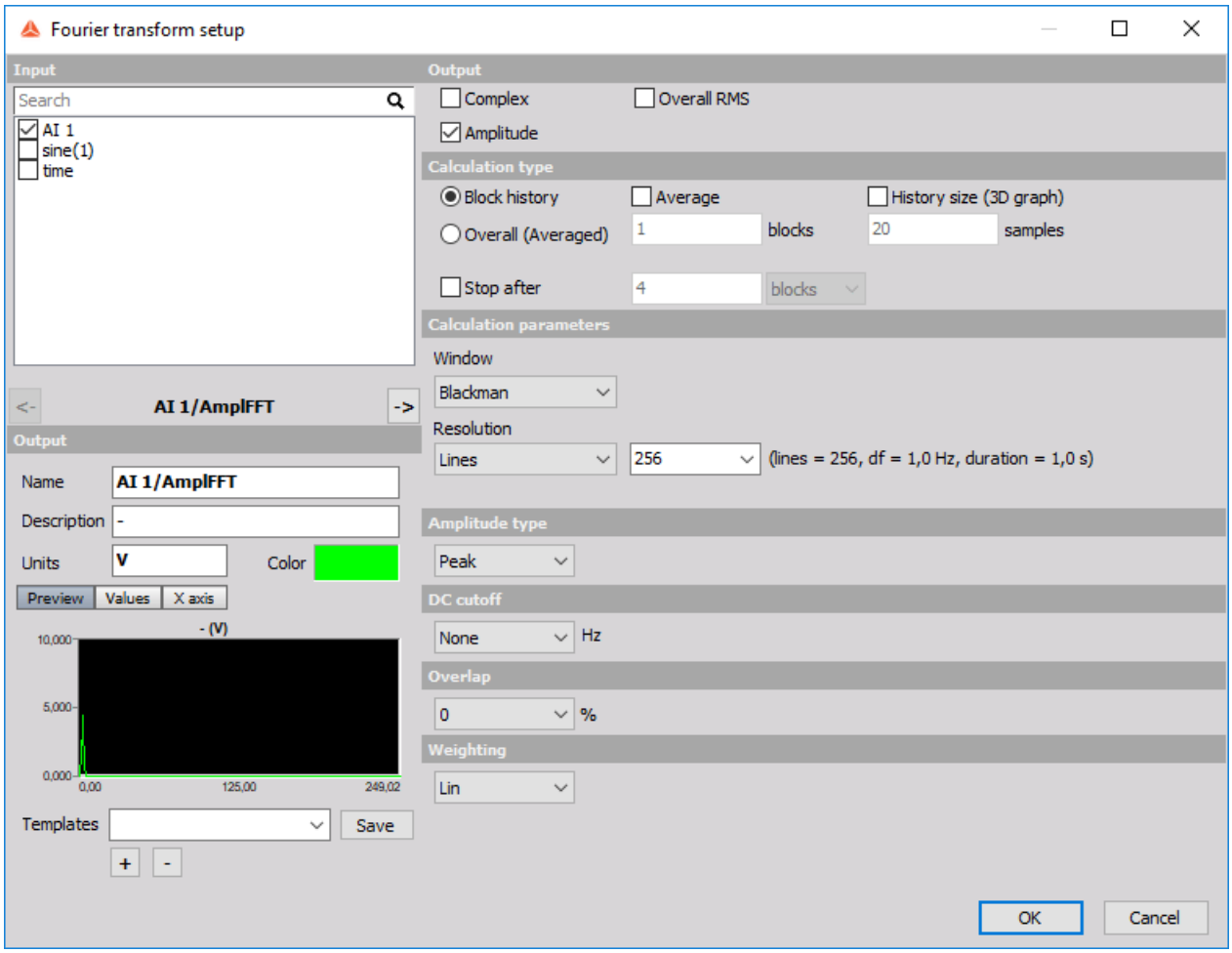

Dewesoft X - Setup: proTu
Simulation mode
Measure
Analyse
Setup files
Ch. setup
Measure
Options
Store
Save
Save as
Storing
Analog in
Math
Latch Math - Scalar
More...
Remove
Input
Latch criteria settings
Input channel
time
Criteria channel
sine(1)
Criteria limit
0.5
Rising edge
Falling edge


Measure
Analyse
Dewesoft X - Setup: proTutorialLatchMa
Disk left: 161,9 GB
a=50hz d=50hz
CPU: 35 %
Setup files
Ch. setup
Measure
Design
Amplifier
Edit
Options
Store
Pause
Stop
Freeze
Recorder
Custom
More...
Remove
C:\DEWESoft\Data\Test.dxd
5000
Recorder
sine(1) ()
1,0000
0,5000
0,0000
-0,5000
-1,0000
Latch ()
15,3472
12,0000
10,0000
8,0000
5,3472
5,3
6,0
8,0
10,0
12,0
14,0
15,3
t (s)

-
-

-
-
-


Recorder
Single value channel ()
1,1000
0,5
0,0000
-1,1000
Sync channel ()
1,1000
0,0000
-1,1000
Async channel ()
1,1000000
0,0000
-1,1000000
31,725
31,730
31,735
31,740
t (s)


Fourier transform setup
Input
Search
AI 1
sine(1)
time
<-
AI 1/AmplFFT
->
Output
Name
AI 1/AmplFFT
Description
-
Units
V
Color
Preview
Values
X axis
- (V)
10,000
5,000
0,000
0,00
125,00
249,02
Templates
Save
+
-
Output
Complex
Overall RMS
Amplitude
Calculation type
Block history
Average
History size (3D graph)
Overall (Averaged)
1
blocks
20
samples
Stop after
4
blocks
Calculation parameters
Window
Blackman
Resolution
Lines
256
(lines = 256, df = 1,0 Hz, duration = 1,0 s)
Amplitude type
Peak
DC cutoff
None
Hz
Overlap
0
%
Weighting
Lin
OK
Cancel

```
class proTutorial_LatchMath_vector
{
public
   bool checkCrossedEdgeCriteria(float currentSampleCriteriaChannel, float nextSampleCriteriaChannel,
float criteriaLimit, int edgeType);
};
```

```
bool proTutorial_LatchMath_vector::checkCrossedEdgeCriteria(float currentSampleCriteriaChannel,
                                                            float nextSampleCriteriaChannel,
                                                            float criteriaLimit,
                                                            int edgeType)
{
   bool crossedRisingEdgeCriteria = currentSampleCriteriaChannel <= criteriaLimit &&
nextSampleCriteriaChannel >= criteriaLimit;
   bool crossedFallingEdgeCriteria = currentSampleCriteriaChannel >= criteriaLimit &&
nextSampleCriteriaChannel <= criteriaLimit;

   if ((crossedFallingEdgeCriteria && edgeType == 1) || (crossedRisingEdgeCriteria && edgeType == 0))
      return true;
   else
      return false;
}
```

`DewesoftBridge::onGetData()`

```
STDMETHODIMP DewesoftBridge::onGetData()
{
   if (!inputChannel || !criteriaChannel)
      return S_OK;

   int blockSizeCriteriaChannel =
      (criteriaChannel->DBPos - (lastPosChecked % criteriaChannel->DBBufSize) + criteriaChannel-
>DBBufSize) % criteriaChannel->DBBufSize;
   int blockSizeInputChannel =
      (inputChannel->DBPos - (lastPosChecked % inputChannel->DBBufSize) + inputChannel-
>DBBufSize) % inputChannel->DBBufSize;

   int minBlockSize = min(blockSizeCriteriaChannel, blockSizeInputChannel);

   for (int i = 0; i < minBlockSize - 1; i++)
   {
      float currentSampleCriteriaChannel = criteriaChannel->DBValues[lastPosChecked %
criteriaChannel->DBBufSize];
      float nextSampleCriteriaChannel = criteriaChannel->DBValues[(lastPosChecked + 1) %
criteriaChannel->DBBufSize];

      bool crossedEdgeCriteria =
protutorial_latchmath_vector.checkCrossedEdgeCriteria(currentSampleCriteriaChannel,
                                                      nextSampleCriteriaChannel,
                                                      criteriaLimit,
                                                      edgeType);
      if (crossedEdgeCriteria)
      {
         float value = inputChannel->DBValues[(lastPosChecked + 1) % inputChannel->DBBufSize];
         outputChannel->AddAsyncSingleSample(value, (lastPosChecked + 1) / app->Data-
>SampleRateEx);
      }
      lastPosChecked++;
   }
   return S_OK;
}
```

```
std::vector<IChannelPtr> DewesoftBridge::getAsyncChannels()
{
    std::vector<IChannelPtr> allChannels;

    app->Data->BuildChannelList();
    IChannelListPtr channels = app->Data->UsedChannels;

    for (int i = 0; i < channels->Count; i++)
        if (channels->GetItem(i) != outputChannel && channels->GetItem(i)->Async)
            allChannels.push_back(channels->GetItem(i));

    return allChannels;
}
```

`addChannelsToCBox()` `inputChannelCBox` `criteriaChannelCBox`

```
void SetupWindow::addChannelsToCBox()
{
    criteriaChannelCBox.clear();
    inputChannelCBox.clear();

    std::vector<IChannelPtr> allChannels = bridge.getAsyncChannels();
    for (int i = 0; i < allChannels.size(); i++)
    {
        std::string channelName = allChannels[i]->Name;
        if (allChannels[i]->ArrayChannel)
            inputChannelCBox.addItem(channelName);
        else
            criteriaChannelCBox.addItem(channelName);
    }
}
```

WriteInteger

-
-
-
-

```
STDMETHODIMP DewesoftBridge::onSaveSetup(IDwXMLDocumentPtr doc, IDwXMLNodePtr node, bool
dataFile)
{
   if (!doc)
      return S_OK;

   doc->WriteSingle(node, "criteriaLimit_vector", getCriteriaLimit(), 0);
   doc->WriteString(node, "inputChannelName_vector", getInputChannelName().c_str(), "");
   doc->WriteString(node, "criteriaChannelName_vector", getCriteriaChannelName().c_str(), "");
   doc->WriteInteger(node, "edgeType_vector", getEdgeType(), 0);

   return S_OK;
}
```

```
STDMETHODIMP DewesoftBridge::onLoadSetup(IDwXMLDocumentPtr doc, IDwXMLNodePtr node, bool
dataFile)
{
   if (doc)
   {
      float criteriaValue = 0;
      doc->ReadSingle(node, "criteriaLimit_vector", &criteriaValue, 0);
      setCriteriaLimit(criteriaValue);

      _bstr_t inputChannelNameVectorBSTR = L"s";
      _bstr_t criteriaChannelNameVectorBSTR = L"s";

      doc->ReadString(node, "inputChannelName_vector", &inputChannelNameVectorBSTR.GetBSTR(),
"");
      inputChannelName = inputChannelNameVectorBSTR;

      doc->ReadString(node, "criteriaChannelName_vector",
&criteriaChannelNameVectorBSTR.GetBSTR(), "");
      criteriaChannelName = criteriaChannelNameVectorBSTR;

      long edgeTypeVectorIndex;
      doc->ReadInteger(node, "edgeType_vector", &edgeTypeVectorIndex, 0);
      edgeType = edgeTypeVectorIndex == 0 ? RisingEdge : FallingEdge;
   }

   mountChannels();

   return S_OK;
}
```

```
//setup_window.h
void setSavedValues();
```

```
// setup_window.cpp

void SetupWindow::setSavedValues()
{
    inputChannelCBox.setSelectedIndex(inputChannelCBox.getIndexOf(bridge.getInputChannelName()));

criteriaChannelCBox.setSelectedIndex(criteriaChannelCBox.getIndexOf(bridge.getCriteriaChannelName()));

    latchCriteriaTBox.setText(std::to_string(bridge.getCriteriaLimit()));

    edgeTypes savedEdgeType = bridge.getEdgeType();
    if (savedEdgeType == RisingEdge)
        risingEdgeRButton->setIsChecked(risingEdgeRButton.getIsEnabled());
    else
        fallingEdgeRButton->setIsChecked(fallingEdgeRButton.getIsEnabled());
}
```

```
void DewesoftBridge::onSetupEnter()
{
    setupWindow->addChannelsToCBox();
    setupWindow->setSavedValues();
}
```

DimCount

DimCount

```
void DewesoftBridge::mountChannels()
{
    const std::vector<int> chIndexVector = {1};
    outputChannel = pluginGroup->MountChannelEx(pluginGuid, long(chIndexVector.size()),
fromStdVec(chIndexVector));

    outputChannel->SetDataType(long(ChannelDataType::Single));
    outputChannel->Name = "Latch";
    outputChannel->Unit_ = "";
    outputChannel->SetAsync(true);
    outputChannel->ExpectedAsyncRate = 5;
    outputChannel->Used = true;
    outputChannel->ArrayChannel = true;

    outputChannel->ArrayInfo->DimCount = 1;
    outputChannel->ArrayInfo->DimSizes[0] = 1;
    outputChannel->ArrayInfo->Init();
}
```

ExpectedAsyncRate

DewesoftBridge::mountChannels()

ExpectedAsyncRate

outputChannel->ExpectedAsyncRate inputChannel->ExpectedAsyncRate

ExpectedAsyncRate ExpectedAsyncRate

`ExpectedAsyncRate`

`ArrayInfo->ArraySize`

`DewesoftBridge::onEstablishConnections()`

```
STDMETHODIMP DewesoftBridge::onEstablishConnections()
{
   setInputChannel(inputChannelName.c_str());
   setCriteriaChannel(criteriaChannelName.c_str());

   return S_OK;
}
```

`evPreInitiate`

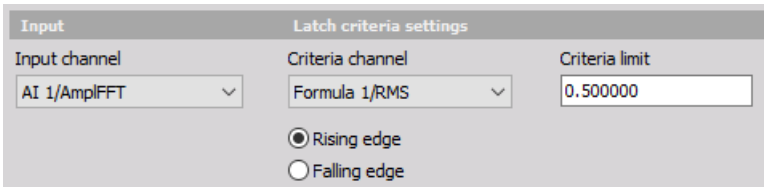
```
// plugin_impl.h
//IPlugin4
STDMETHODIMP eventBeforeReserveMemory();

// plugin_impl.cpp
STDMETHODIMP Plugin::raw_OnEvent(enum EventIDs eventID, VARIANT inParam, VARIANT* outParam)
{
   // ...
   switch (eventID)
   {
      // ...
      case evPreInitiate:
         returnValue = eventBeforeReserveMemory();
         break;
      // ...
   return returnValue;
}

STDMETHODIMP Plugin::eventBeforeReserveMemory()
{
   return bridge.onBeforeReserveMemory();
}
```

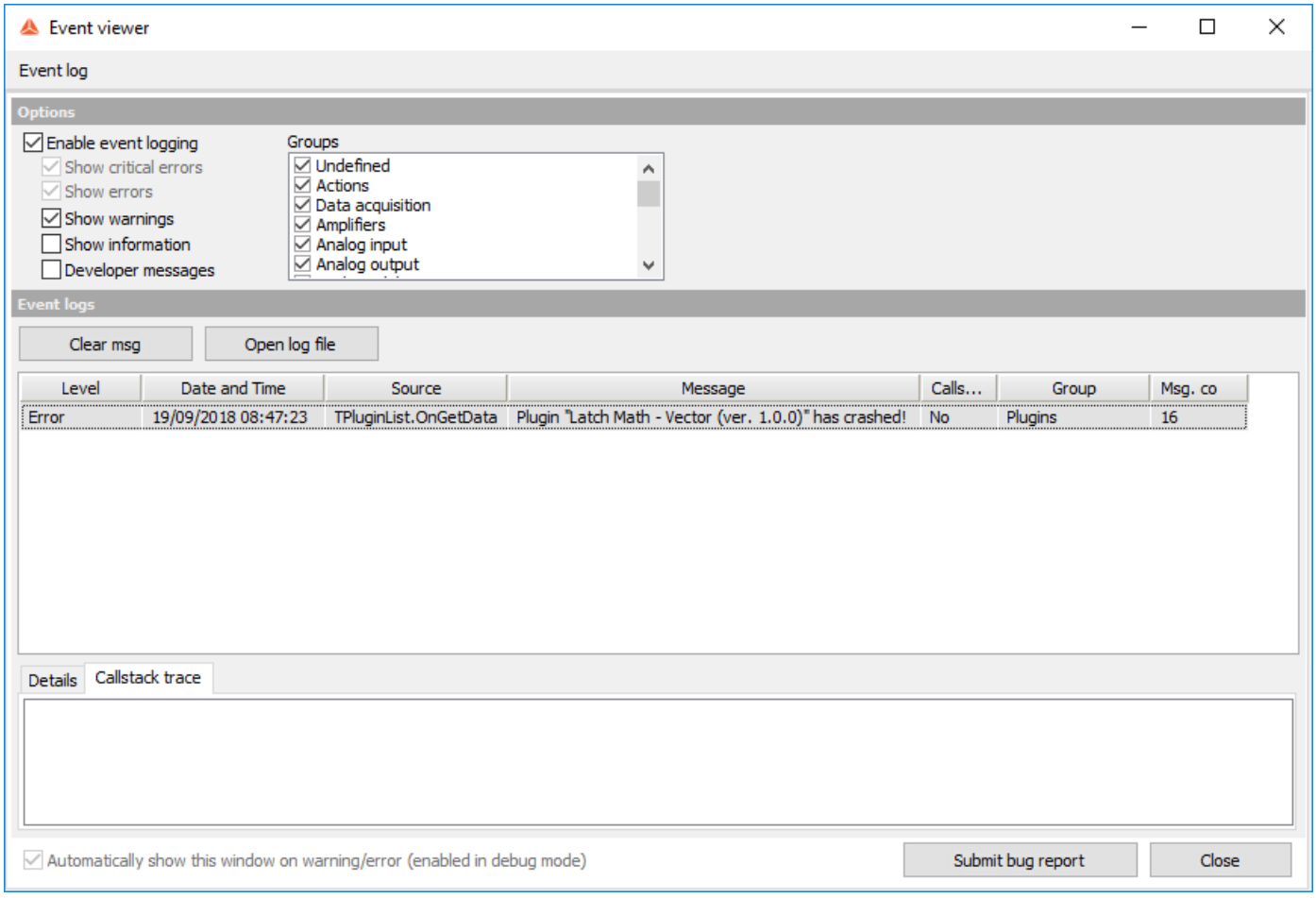
```
// dewesoft_bridge.h
STDMETHODIMP onBeforeReserveMemory();

// dewesoft_bridge.cpp
STDMETHODIMP DewesoftBridge::onBeforeReserveMemory()
{
    if (!inputChannel)
        return S_OK;

    outputChannel->ArrayInfo->DimCount = 1;
    outputChannel->ArrayInfo->DimSizes[0] = inputChannel->ArraySize;
    outputChannel->ArrayInfo->Init();
    outputChannel->ArrayInfo->AxisDef[0]->Name = inputChannel->ArrayInfo->AxisDef[0]->Name;
    outputChannel->ArrayInfo->AxisDef[0]->_Unit = inputChannel->ArrayInfo->AxisDef[0]->_Unit;

    outputChannel->ArrayInfo->AxisDef[0]->AxisType = atFloatLinearFunc;
    outputChannel->ArrayInfo->AxisDef[0]->StartValue = inputChannel->ArrayInfo->AxisDef[0]-
>StartValue;
    outputChannel->ArrayInfo->AxisDef[0]->StepValue = inputChannel->ArrayInfo->AxisDef[0]-
>StepValue;

    return S_OK;
}
```

`onGetData()`

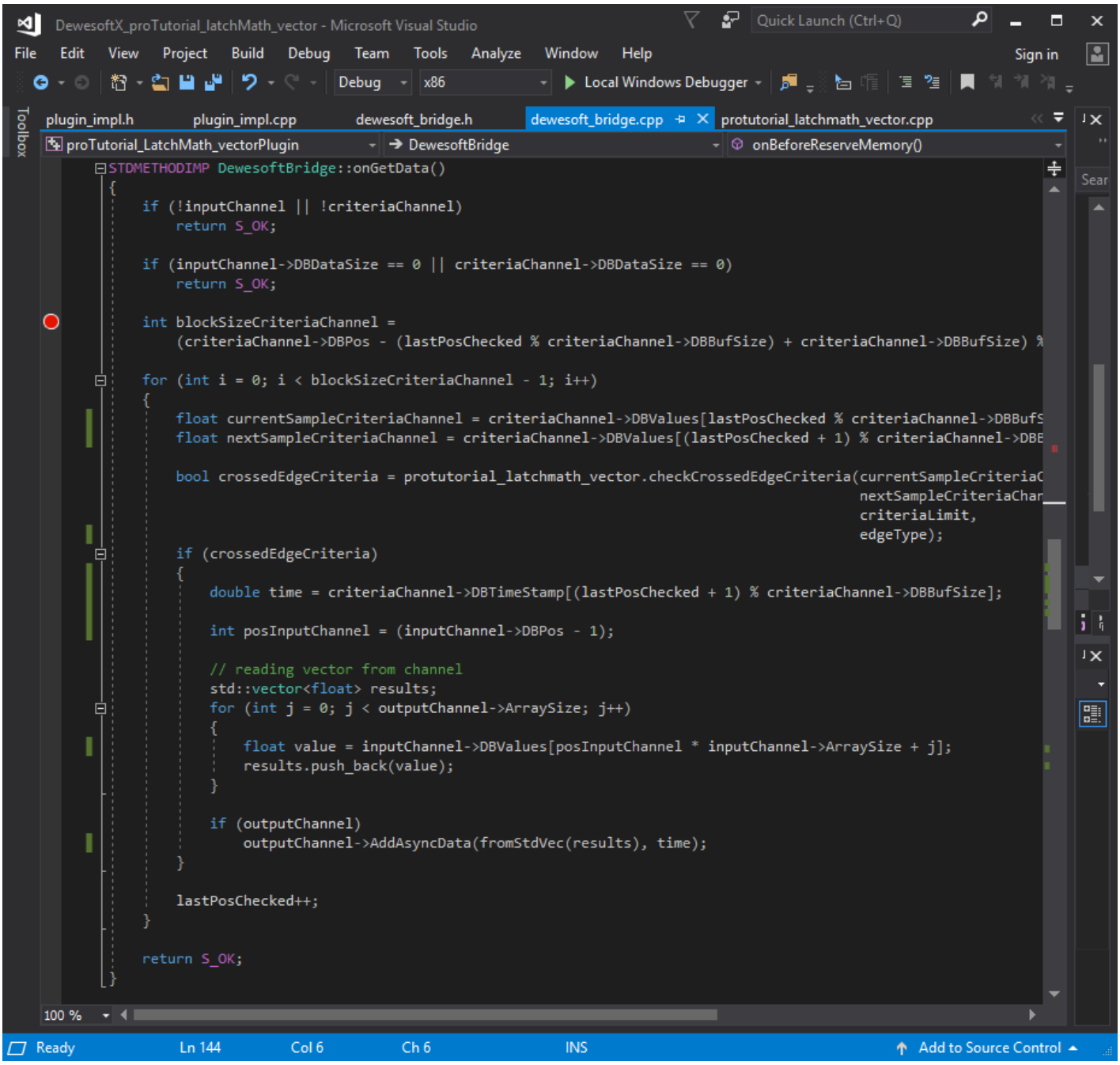
`AddAsyncData(...)`

```
STDMETHODIMP DewesoftBridge::onGetData()
{
    if (!inputChannel || !criteriaChannel)
        return S_OK;

    if (inputChannel->DBDataSize == 0 || criteriaChannel->DBDataSize == 0)
        return S_OK;

    int blockSizeCriteriaChannel =
        (criteriaChannel->DBPos - (lastPosChecked % criteriaChannel->DBBufSize) + criteriaChannel-
>DBBufSize) % criteriaChannel->DBBufSize;

    for (int i = 0; i < blockSizeCriteriaChannel - 1; i++)
    {
        float currentSampleCriteriaChannel = criteriaChannel->DBValues[lastPosChecked %
criteriaChannel->DBBufSize];
        float nextSampleCriteriaChannel = criteriaChannel->DBValues[(lastPosChecked + 1) %
criteriaChannel->DBBufSize];

        bool crossedEdgeCriteria =
protutorial_latchmath_vector.checkCrossedEdgeCriteria(currentSampleCriteriaChannel,
                                                      nextSampleCriteriaChannel,
                                                      criteriaLimit,
                                                      edgeType);
        if (crossedEdgeCriteria)
        {
            double time = criteriaChannel->DBTimeStamp[(lastPosChecked + 1) % criteriaChannel->DBBufSize

            int posInputChannel = (inputChannel->DBPos - 1);

            // reading vector from channel
            std::vector<float> results;
            for (int j = 0; j < outputChannel->ArraySize; j++)
            {
                float value = inputChannel->DBValues[posInputChannel * inputChannel->ArraySize + j];
                results.push_back(value);
            }

            if (outputChannel)
                outputChannel->AddAsyncData(fromStdVec(results), time);
        }
        lastPosChecked++;
    }
    return S_OK;
}
```


Input
Latch criteria settings
Input channel
AI 1/AmplFFT
Criteria channel
Formula 1/RMS
Criteria limit
0.500000
Rising edge
Falling edge

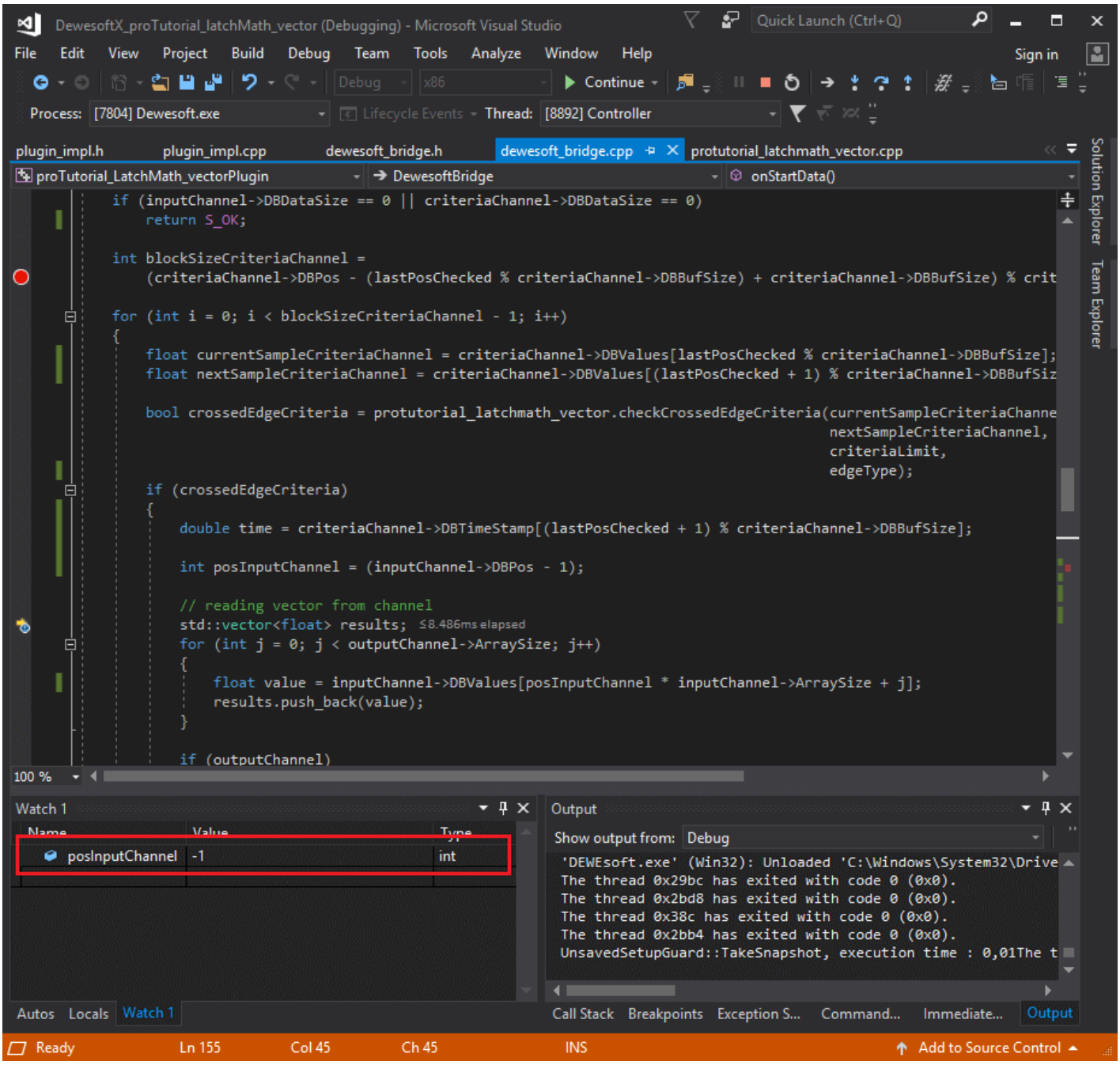

Event viewer
Event log
Options
Enable event logging
Show critical errors
Show errors
Show warnings
Show information
Developer messages
Groups
Undefined
Actions
Data acquisition
Amplifiers
Analog input
Analog output
Event logs
Clear msg
Open log file
Level
Date and Time
Source
Message
Calls...
Group
Msg. co
Error
19/09/2018 08:47:23
TPluginList.OnGetData
Plugin "Latch Math - Vector (ver. 1.0.0)" has crashed!
No
Plugins
16
Details
Callstack trace
Automatically show this window on warning/error (enabled in debug mode)
Submit bug report
Close

`OnGetData()`



`posInputChannel`

`inputChannel->DBValues[...]`

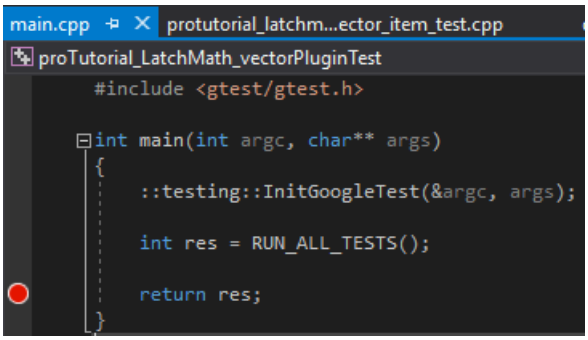


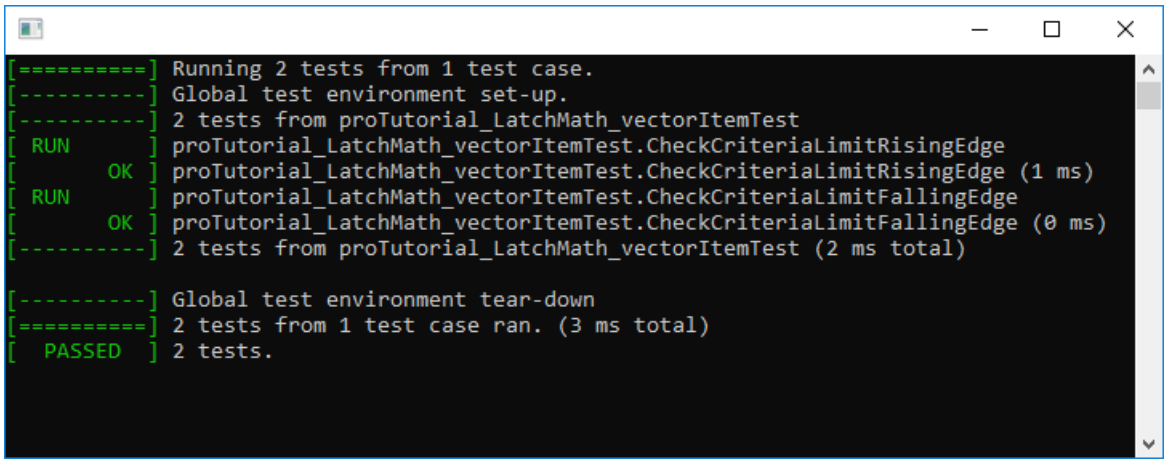

```
int posInputChannel = inputChannel->DBPos != 0 ? inputChannel->DBPos - 1 : inputChannel->DBBufSize - 1;
```

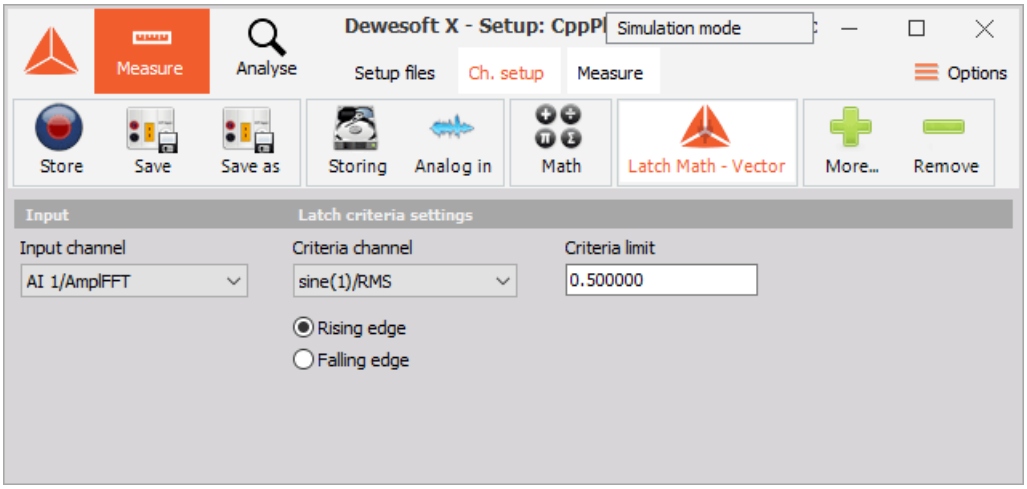
checkCrossedEdgeCriteria(...)

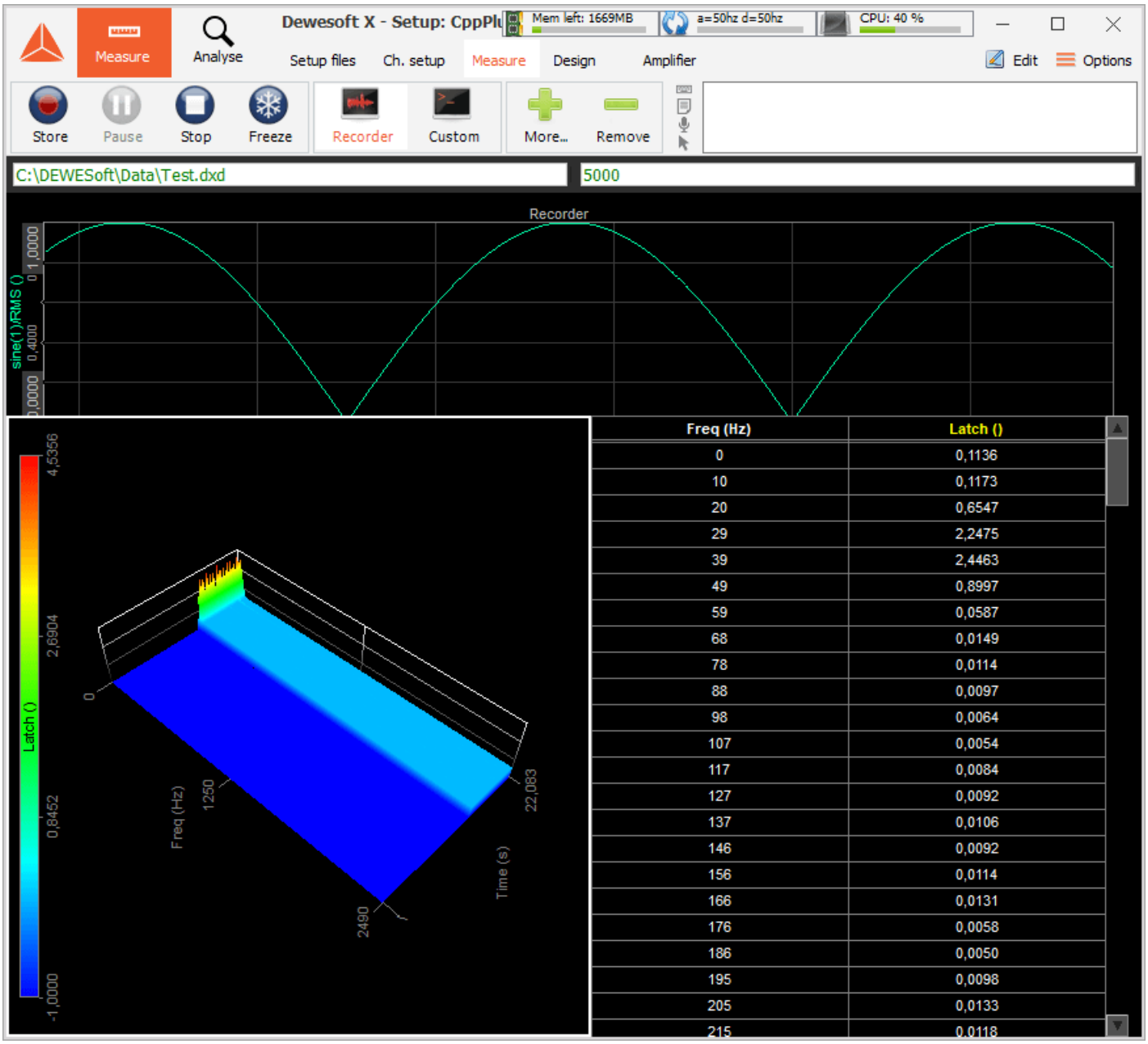
```
TEST_F(proTutorial_LatchMath_vectorItemTest, CheckCriteriaLimitRisingEdge)
{
    proTutorial_LatchMath_vector protutorial_latchmath_vector;

    float currentSampleCriteriaChannel = 0.49;
    float nextSampleCriteriaChannel = 0.51;
    float criteriaLimit = 0.5;
    int edgeType = 0; // RisingEdge

    bool crossedEdgeCriteria =
protutorial_latchmath_vector.checkCrossedEdgeCriteria(currentSampleCriteriaChannel,
                                                      nextSampleCriteriaChannel,
                                                      criteriaLimit,
                                                      edgeType);
    ASSERT_TRUE(crossedEdgeCriteria);
}

TEST_F(proTutorial_LatchMath_vectorItemTest, CheckCriteriaLimitFallingEdge)
{
    proTutorial_LatchMath_vector protutorial_latchmath_vector;

    float currentSampleCriteriaChannel = -0.12;
    float nextSampleCriteriaChannel = 0.01;
    float criteriaLimit = 0;
    int edgeType = 1; // FallingEdge

    bool crossedEdgeCriteria =
protutorial_latchmath_vector.checkCrossedEdgeCriteria(currentSampleCriteriaChannel,
                                                      nextSampleCriteriaChannel,
                                                      criteriaLimit,
                                                      edgeType);
    ASSERT_FALSE(crossedEdgeCriteria);
}
```

```
return res;
```

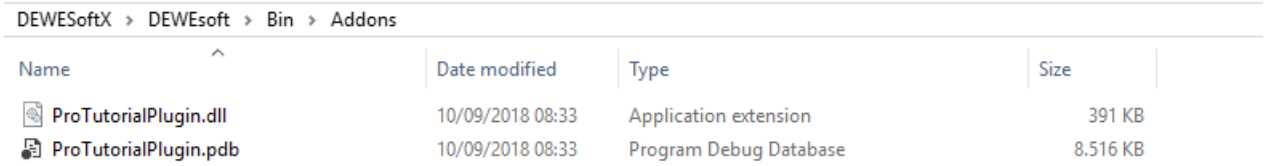



[==========] Running 2 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 2 tests from proTutorial_LatchMath_vectorItemTest
[RUN   ] proTutorial_LatchMath_vectorItemTest.CheckCriteriaLimitRisingEdge
[OK] proTutorial_LatchMath_vectorItemTest.CheckCriteriaLimitRisingEdge (1 ms)
[RUN   ] proTutorial_LatchMath_vectorItemTest.CheckCriteriaLimitFallingEdge
[OK] proTutorial_LatchMath_vectorItemTest.CheckCriteriaLimitFallingEdge (0 ms)
[----------] 2 tests from proTutorial_LatchMath_vectorItemTest (2 ms total)
[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (3 ms total)
[PASSED] 2 tests.


Measure
Analyse
Dewesoft X - Setup: CppP
Simulation mode
Setup files
Ch. setup
Measure
Options
Store
Save
Save as
Storing
Analog in
Math
Latch Math - Vector
More...
Remove
Input
Latch criteria settings
Input channel
AI 1/AmplFFT
Criteria channel
sine(1)/RMS
Criteria limit
0.500000
Rising edge
Falling edge



Freq (Hz)	Latch ()
0	0,1136
10	0,1173
20	0,6547
29	2,2475
39	2,4463
49	0,8997
59	0,0587
68	0,0149
78	0,0114
88	0,0097
98	0,0064
107	0,0054
117	0,0084
127	0,0092
137	0,0106
146	0,0092
156	0,0114
166	0,0131
176	0,0058
186	0,0050
195	0,0098
205	0,0133
215	0,0118


DEWESoftX › DEWEsoft › Bin › Addons
Name
Date modified
Type
Size
ProTutorialPlugin.dll
10/09/2018 08:33
Application extension
391 KB
ProTutorialPlugin.pdb
10/09/2018 08:33
Program Debug Database
8.516 KB