MQL4 COURSE

By Coders' guru
www.forex-tsd.com

-18-

Working with templates

We have created our first indicator, expert advisor and script in the previous lessons.
Every time we decide to create our program whatever the kind of the program (indicator, expert or script) we have to write it from scratch.
This is a useful thing for MQL4 beginners, but it's time consuming for advanced programmers.
That's why MQL4 has the feature of creating ***templates based programs***.

Today we will create an indicator based on MACD template shipped with MetaTrader.
So, let's template!

What are the MQL4 templates?

The MQL4 templates are text files contain instructions (commands) for MetaEditor to generate a new program based on these instructions. MetaEditor will read these instructions and use them to generate the appropriated lines of code.

We use the templates to duplicate the frequently used codes and save our coding time.
By using the templates you can create indicators, experts and scripts based on your favorite indicators, experts and scripts and save the time of writing the code from scratch.

Today we are going to duplicate the MACD indicator and we will not add to it anything that's because we are not in the indicator creating lesson but we are in the template lessons.

Inside look:

We are going to give the template file a look before digging into our creation of our first template based indicator.

Note: The template files are stored in the **experts\templates** path.
And have the "**.mqt**" file extension.

Here's the content of the "**MACD.mqt**" template which we will use it today creating our template based indicator:

Note: We have indented some of content and colored them to make it clearer but you can give the original file an inside look.

```
<expert>
      type=INDICATOR_ADVISOR
      separate_window=1
      used_buffers=2

      <param>
            type=int
            name=FastEMA
            value=12
      </param>

      <param>
            type=int
            name=SlowEMA
            value=26
      </param>

      <param>
            type=int
            name=SignalSMA
            value=9
      </param>

      <ind>
            color=Silver
            type=DRAW_HISTOGRAM
      </ind>

      <ind>
             color=Red
      </ind>
</expert>

#header#
#property copyright "#copyright#"
#property link      "#link#"

#indicator_properties#
#extern_variables#
#mapping_buffers#

//---- indicator buffers
double ExtSilverBuffer[];
double ExtRedBuffer[];

//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int init()
  {
   #buffers_used#;
//---- drawing settings
   #indicators_init#
//----
   SetIndexDrawBegin(1,SignalSMA);
   IndicatorDigits(5);
//---- indicator buffers mapping
   SetIndexBuffer(0, ExtSilverBuffer);
   SetIndexBuffer(1, ExtRedBuffer);
//---- name for DataWindow and indicator subwindow label
   IndicatorShortName("MACD("+FastEMA+","+SlowEMA+","+SignalSMA+")");
//---- initialization done
   return(0);
  }
//+------------------------------------------------------------------+
//| Moving Averages Convergence/Divergence                           |
//+------------------------------------------------------------------+
int start()
  {
   int limit;
   int counted_bars=IndicatorCounted();
//---- check for possible errors
   if(counted_bars<0) return(-1);
//---- last counted bar will be recounted
   if(counted_bars>0) counted_bars--;
   limit=Bars-counted_bars;
//---- macd counted in the 1-st buffer
   for(int i=0; i<limit; i++)
      ExtSilverBuffer[i]=iMA(NULL,0,FastEMA,0,MODE_EMA,PRICE_CLOSE,i)-
iMA(NULL,0,SlowEMA,0,MODE_EMA,PRICE_CLOSE,i);
//---- signal line counted in the 2-nd buffer
   for(i=0; i<limit; i++)

ExtRedBuffer[i]=iMAOnArray(ExtSilverBuffer,Bars,SignalSMA,0,MODE_SMA,i)
;
//---- done
   return(0);
  }
//+------------------------------------------------------------------+
```

If you give the above code a look you can notice that the most of the code is a normal MQL4 code with two new kinds of different code.

The first kind of code (Ex: <**expert**> and <**param**>) looks like the HTML tags; these are the ***template tags***.
And the second kind of code (Ex: **#header#** and **#link#**) looks like the MQL4 directives; these are the ***template commands***:

MQL4 template tags:

The template file starts with **tags** very like the Html and XML tags.
There is a start tag and a closing tag. The closing tag uses a slash after the opening bracket. For example <**expert**> is the start tag and <**/expert**> is the closing tag.
The text between the brackets is called an element. For example:
type=int
name=FastEMA
value=12

There are three tags in MQL4 template:

expert tag:

This is main tag and the other two tags are belonging to it.
In this tag we write the type of program, the indicator chart window, the number of buffer used, the bottom border for the chart and the top border for the chart of the indicator.

These are the elements used in the **expert** tag:

type: The type of program. Possible values are EXPERT_ADVISOR, INDICATOR_ADVISOR, SCRIPT_ADVISOR.
separate_window: The chart window type.
used_buffers: The number of used buffers.
ind_minimum: The fixed bottom border of indicator window.
ind_maximum: The fixed top border of indicator window.

param tag:

We use this tag to create external variables. For every variable we use a **param** tag.

These are the elements used in the **param** tag:

type: The data type of the variable.
name: The variable name.
value: The default value of the variable.

ind tag:

We use this tag to set the parameters of every indicator (drawing line) we use in our program.

These are the elements used in the **ind** tag:

type: The indicator drawing shape style. Possible values are DRAW_LINE, DRAW_HISTOGRAM, DRAW_SECTION, DRAW_ARROW.
color: The indicator color.
arrow: The character for displaying the DRAW_ARROW indicator. "Wingdings" font is used for displaying the characters.

MQL4 template commands:

They are lines of code starts and ends with # symbol. And these lines of code will be replaced with the corresponded lines with code in the generation process.

Note: MetaEditor reads these lines and replace them in the place they found and generate the MQ4 file.

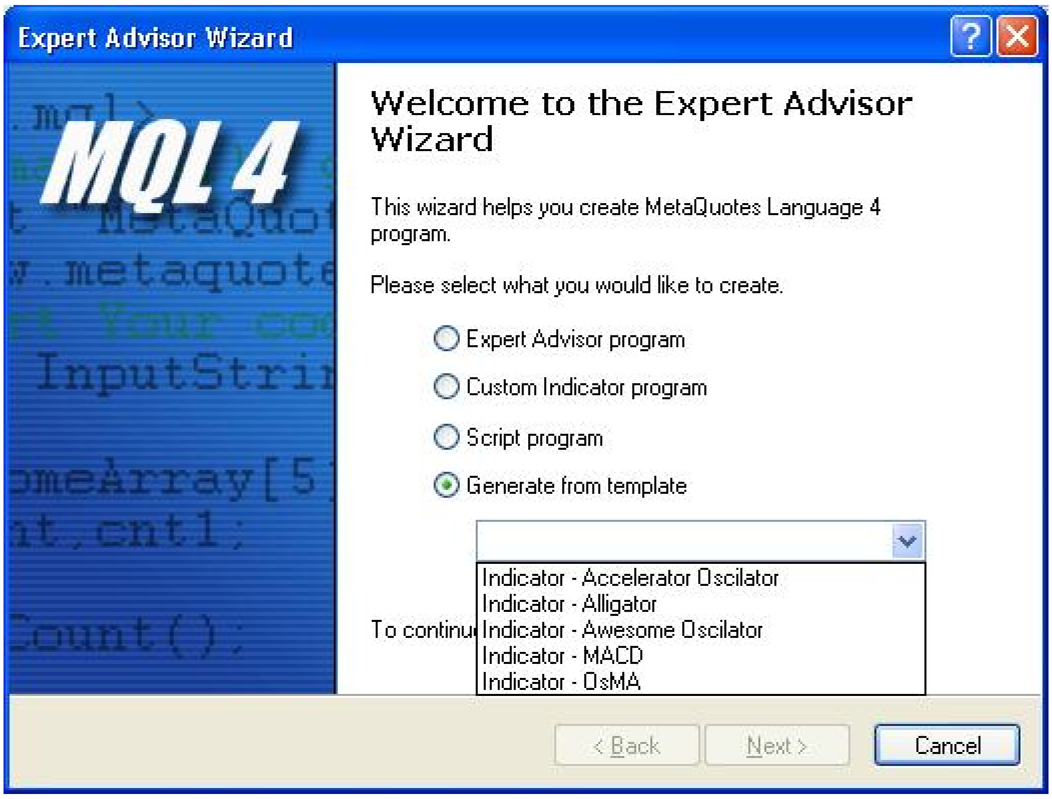
These are the template commands:

#header#: This line will be replaced with the program header block (comments with file name, author name and the company name).
#copyright#: This line will be replaced with your company name.
#link#: This line will be replaced with your website link.
#indicator_properties#: This line will be replaced with the properties of the indicator.

#extern_variables#: This line will be replaced with external variables used in your program with their types and default values.
#buffers_used#: This line will be replaced with the number of buffer used if any.
#indicators_init#: This line will be replaced with the initialization of the indicators (using the function ***SetIndexStyle***).

Creating our MACD template based indicator.

Now, let's create our template based indicator by hitting our ***File*** menu and choose ***New*** or clicking CTRL+N. That will bring the New Program wizard (Figure 1).
This time we will choose "Create from template" option and from the drop list we will choose "Indicator – MACD".


Expert Advisor Wizard

Welcome to the Expert Advisor Wizard

This wizard helps you create MetaQuotes Language 4 program.

Please select what you would like to create.

- Expert Advisor program
- Custom Indicator program
- Script program
- [✓ Generate from template]

Indicator - Accelerator Oscilator
Indicator - Alligator
Indicator - Awesome Oscilator
Indicator - MACD
Indicator - OsMA

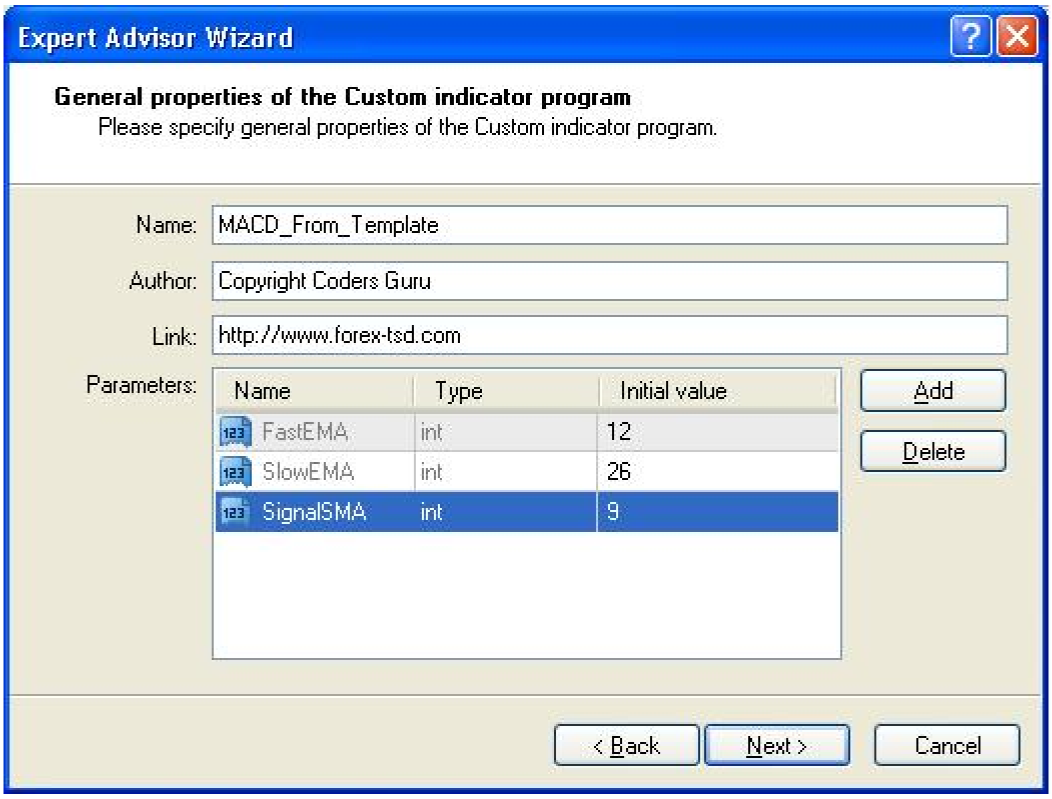
To continu

Figure 1 – New program wizard

Then we will hit ***Next*** button which brings the second step wizard (Figure 2).

MetaEditor has filled the fields with ***Author*** name and the ***Link*** (reading them from the windows registry) and left the ***Name*** field blank which we typed in it "MACD_From_Template"

In the ***Parameters*** list the MetaEditor has red our template file and filled the list with the external variables and its default values.



Expert Advisor Wizard

General properties of the Custom indicator program

Please specify general properties of the Custom indicator program.

Name: MACD_From_Template

Author: Copyright Coders Guru

Link: http://www.forex-tsd.com

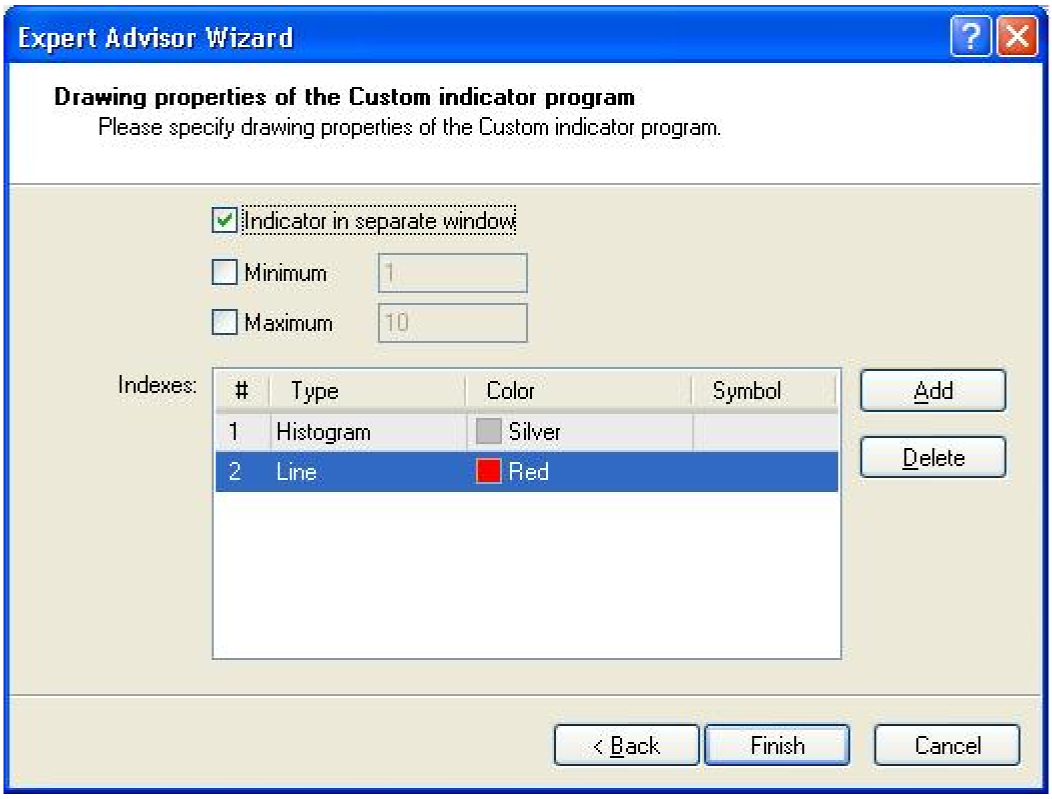
Parameters:

Name	Type	Initial value
FastEMA	int	12
SlowEMA	int	26
SignalSMA	int	9

Figure 2 – Second step wizard

Hitting ***Next*** button will bring the third step wizard (figure 3) which contains the indicator parameters.
As you guessed MetaEditor has red our template file and generated the values of the indicator properties.

Now click the finish button and you will see the magic of the templates.



Drawing properties of the Custom indicator program

Please specify drawing properties of the Custom indicator program.

[✓] Indicator in separate window

[ ] Minimum 1

[ ] Maximum 10

Indexes:

#	Type	Color	Symbol
1	Histogram	⬜ Silver	
2	Line	🟥 Red	

Figure 3 – Third step wizard

What have we got?

The wizards have generated a new indicator for us based on MACD template. Code ready to compile, ready to run:

```
//+------------------------------------------------------------------+
//|                                            MACD_From_Template.mq4 |
//|                                             Copyright Coders Guru |
//|                                         http://www.forex-tsd.com |
//+------------------------------------------------------------------+
#property copyright "Copyright Coders Guru"
#property link      "http://www.forex-tsd.com"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_color1 Silver
#property indicator_color2 Red
//---- input parameters
extern int       FastEMA=12;
extern int       SlowEMA=26;
extern int       SignalSMA=9;
//---- buffers
//---- indicator buffers
double ExtSilverBuffer[];
double ExtRedBuffer[];

//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int init()
```

```
  {
//---- drawing settings
   SetIndexStyle(0,DRAW_HISTOGRAM);
   SetIndexStyle(1,DRAW_LINE);
//----
   SetIndexDrawBegin(1,SignalSMA);
   IndicatorDigits(5);
//---- indicator buffers mapping
   SetIndexBuffer(0, ExtSilverBuffer);
   SetIndexBuffer(1, ExtRedBuffer);
//---- name for DataWindow and indicator subwindow label
   IndicatorShortName("MACD("+FastEMA+","+SlowEMA+","+SignalSMA+")");
//---- initialization done
   return(0);
  }
//+------------------------------------------------------------------+
//| Moving Averages Convergence/Divergence                           |
//+------------------------------------------------------------------+
int start()
  {
   int limit;
   int counted_bars=IndicatorCounted();
//---- check for possible errors
   if(counted_bars<0) return(-1);
//---- last counted bar will be recounted
   if(counted_bars>0) counted_bars--;
   limit=Bars-counted_bars;
//---- macd counted in the 1-st buffer
   for(int i=0; i<limit; i++)
      ExtSilverBuffer[i]=iMA(NULL,0,FastEMA,0,MODE_EMA,PRICE_CLOSE,i)-
iMA(NULL,0,SlowEMA,0,MODE_EMA,PRICE_CLOSE,i);
//---- signal line counted in the 2-nd buffer
   for(i=0; i<limit; i++)

ExtRedBuffer[i]=iMAOnArray(ExtSilverBuffer,Bars,SignalSMA,0,MODE_SMA,i)
;
//---- done
   return(0);
  }
//+------------------------------------------------------------------+
```

Compile the code and enjoy your new MACD indicator.

I welcome very much your questions and suggestions.

Coders' Guru
10-01-2006