```
//+------------------------------------------------------------------+
//|                                      StdDev_AverageAndBands.mq4 |
//|                                                        M Wilson |
//|                                    https://www.algotrader.blog |
//+------------------------------------------------------------------+
#property copyright "M Wilson"
#property link      "https://www.algotrader.blog"
#property version   "1.00"
#property strict
#property indicator_chart_window

///--- indicator settings
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_color1  DodgerBlue
#property indicator_color2  Red
#property indicator_color3  Orange
#property indicator_color4  Orange
//--- input parameter
input int I_STDDevPeriod=24;                    // StdDev Period
input int I_STDDevAveragePeriod=100;
// Period where STDDev Average is measured.
input double I_BandMultiplier=1.0;
// Multiplier for the StdDev of StdDev Bands
//--- buffers
double ExtSTDDevBuffer[];
double ExtSTDDevAverageBuffer[];
double ExtSTDDevAverageUpperBandBuffer[];
double ExtSTDDevAverageLowerBandBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit(void)
  {
   string short_name, short_name_av,short_name_up,short_name_down;
//--- 1 additional buffer used for counting.
   IndicatorBuffers(4);
   IndicatorDigits(Digits);
//--- indicator line
   SetIndexStyle(0,DRAW_LINE);
   SetIndexBuffer(0,ExtSTDDevBuffer);
   SetIndexStyle(1,DRAW_LINE);
   SetIndexBuffer(1,ExtSTDDevAverageBuffer);
   SetIndexStyle(2,DRAW_LINE);
   SetIndexBuffer(2,ExtSTDDevAverageUpperBandBuffer);
   SetIndexStyle(3,DRAW_LINE);
   SetIndexBuffer(3,ExtSTDDevAverageLowerBandBuffer);
//--- name for DataWindow and indicator subwindow label
   short_name="STDDev_"+IntegerToString(I_STDDevPeriod);
   short_name_av="STDDev_AV_"+IntegerToString(I_STDDevPeriod);
   short_name_up="STDDev_AV_UPPER_"+IntegerToString(I_STDDevPeriod);
   short_name_down="STDDev_AV_LOWER_"+IntegerToString(I_STDDevPeriod);
   IndicatorShortName(short_name);
   SetIndexLabel(0,short_name);
   SetIndexLabel(1,short_name_av);
   SetIndexLabel(2,short_name_up);
   SetIndexLabel(3,short_name_down);
//--- check for input parameter
   if(I_STDDevPeriod<=0)
     {
      Print("Wrong input parameter STDDev Period=",I_STDDevPeriod);
      return(INIT_FAILED);
     }
   if(I_STDDevAveragePeriod<=1)
   {
      Print("Wrong input parameter Average Period=",I_STDDevAveragePeriod);
      return(INIT_FAILED);
   }
   if(I_BandMultiplier<=0)
   {
      Print("Wrong input parameter Multiplier=",I_BandMultiplier);
```

```
      return(INIT_FAILED);
   }
//---
   SetIndexDrawBegin(0,I_STDDevPeriod);
   SetIndexDrawBegin(1,I_STDDevAveragePeriod+I_STDDevPeriod);
   SetIndexDrawBegin(2,I_STDDevAveragePeriod+I_STDDevPeriod);
   SetIndexDrawBegin(3,I_STDDevAveragePeriod+I_STDDevPeriod);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Average True Range                                               |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- check for bars count and input parameter
   if(rates_total<=I_STDDevPeriod+I_STDDevAveragePeriod || I_STDDevPeriod<=0)
      return(0);
//--- counting from 0 to rates_total
   ArraySetAsSeries(ExtSTDDevBuffer,True);
   ArraySetAsSeries(ExtSTDDevAverageBuffer,True);
   ArraySetAsSeries(ExtSTDDevAverageUpperBandBuffer,True);
   ArraySetAsSeries(ExtSTDDevAverageLowerBandBuffer,True);
//--- Populate STDDev Buffers
   for(int i=0;i<rates_total;i++)
   {
      ExtSTDDevBuffer[i]=iStdDev(Symbol(),0,I_STDDevPeriod,0,MODE_SMA,PRICE_CLOSE,i);
   }
//--- Populate Av Buffer
   for(int i=0;i<rates_total;i++)
   {
      if(i>rates_total-I_STDDevPeriod-I_STDDevAveragePeriod)
      {
         ExtSTDDevAverageBuffer[i]=0.0;
      }
      else
      {
         double dblAv=0.0;
         for(int k=0;k<I_STDDevAveragePeriod;k++)
         {
            dblAv+=ExtSTDDevBuffer[i+k];
         }
         ExtSTDDevAverageBuffer[i]=dblAv/I_STDDevAveragePeriod;
      }
   }
//--- Populate the Upper and Lower Bands
   for(int i=0;i<rates_total;i++)
   {
      if(i>rates_total-I_STDDevPeriod-I_STDDevAveragePeriod)
      {
         ExtSTDDevAverageUpperBandBuffer[i]=0.0;
         ExtSTDDevAverageLowerBandBuffer[i]=0.0;
      }
      else
      {
         double dblSTDev=0.0;
         for(int k=0;k<I_STDDevAveragePeriod;k++)
         {
            dblSTDev+=(ExtSTDDevBuffer[i+k]-ExtSTDDevAverageBuffer[i+k])*(
ExtSTDDevBuffer[i+k]-ExtSTDDevAverageBuffer[i+k]);
         }
         dblSTDev=sqrt(dblSTDev/(I_STDDevAveragePeriod-1));
```

```
         ExtSTDDevAverageUpperBandBuffer[i]=ExtSTDDevAverageBuffer[i]+I_BandMultiplier*
dblSTDev;
         ExtSTDDevAverageLowerBandBuffer[i]=ExtSTDDevAverageBuffer[i]-I_BandMultiplier*
dblSTDev;
       }
    }
//--- return value of prev_calculated for next call
   return(rates_total);
  }
//+------------------------------------------------------------------+
```