```
//----
#property indicator_separate_window
#property indicator_buffers 15
#property indicator_color1 Blue
#property indicator_color2 FireBrick
#property indicator_color3 Blue
#property indicator_color4 Violet
//---- input parameters
extern string separator1 = "*** OSMA Settings ***";
extern int fastEMA = 12;
extern int slowEMA = 26;
extern int signal = 9;
extern string separator2 = "*** Indicator Settings ***";
extern bool drawDivergenceLines = true;
extern bool displayAlert = true;
extern string Symbol1 = "";
extern string Symbol2 = "";
//---- buffers
double Value=0,Value1=0,Value2=0,Fish=0,Fish1=0,Fish2=0;
int buy=0,sell=0;
double buffer1[1000];
double buffer2[1000];
double buffer3[1000];
double MA1buffer[1000];
double MA2buffer[1000];
//----
extern int period=10;
extern int price=0; // 0 or other = (H+L)/2
extern bool Mode_Fast= False;
extern bool Signals= False;
extern int MA1period=9, MA2period=45;
extern string TypeHelp="SMA- 0, EMA - 1, SMMA - 2, LWMA- 3";
extern string TypeHelp2="John Hyden settings TypeMA1=0, TypeMA2=3";
extern int TypeMA1=0;
extern int TypeMA2=3;
string str1;
int SELLMA[];
int BUYMA[];
double upOsMA[];
double downOsMA[];
double bullishDivergence[];
double bearishDivergence[];
double bullishDiverg[];
double bearishDiverg[];
bool isBuy, isSell;
bool isBuySet, isSellSet;
double OsMA[];
double upMA[];
double downMA[];
double bullishDivergenceMA[];
double bearishDivergenceMA[];
double bullishDivergenceMADeviation[1000];
double bearishDivergenceMADeviation[1000];
```

```
double LPeak, LTrough;
double LPeak2, LTrough2;
double MA[];
double MA2[1000];
double MA1H1_1, MA1H1_2, MA1H1_3;
double SLShort,SLLong, strTargetLong,strTargetShort,C0,C1,C2;
string Target,ask,SL,bid , strSell,strBuy;
string strDirection[];
static string isBuyExist = "false" ;
static string isSellExist ="false" ;
int BarSell = -1;
int BarBuy = -1;
string str;
//----
static datetime lastAlertTime;
//+------------------------------------------------------------------+
//| Custom indicator initialization function |
//+------------------------------------------------------------------+
int init()
{
SetIndexStyle(0,DRAW_NONE);
SetIndexBuffer(0,buffer1);
SetIndexStyle(1,DRAW_NONE);
SetIndexBuffer(1,buffer2);
SetIndexStyle(2,DRAW_NONE);
SetIndexLabel(2,"line");
SetIndexBuffer(2,buffer3);
SetIndexStyle(3, DRAW_HISTOGRAM, STYLE_SOLID, 2);
SetIndexLabel(3,"MA1 "+MA1period);
SetIndexStyle(4, DRAW_HISTOGRAM, STYLE_SOLID, 2);
SetIndexLabel(4,"MA2 "+MA2period);
SetIndexBuffer(3,MA1buffer);
SetIndexBuffer(4,MA2buffer);
SetIndexStyle(5, DRAW_NONE);
SetIndexStyle(6, DRAW_NONE);
SetIndexStyle(7, DRAW_NONE);
SetIndexStyle(8, DRAW_NONE);
SetIndexStyle(9, DRAW_NONE);
SetIndexStyle(10, DRAW_NONE);
SetIndexBuffer(6, downMA);
SetIndexBuffer(7, MA);
SetIndexBuffer(8, OsMA);
SetIndexBuffer(9, SELLMA);
SetIndexBuffer(10, BUYMA);
ObjectCreate("Symbol1",OBJ_LABEL,0,0,0,0,0);
ObjectCreate("Symbol2",OBJ_LABEL,0,0,0,0,0);
//----
return(0);
}
//+------------------------------------------------------------------+
//| Custom indicator deinitialization function |
//+------------------------------------------------------------------+
```

```
int deinit()
{f
or(int i = ObjectsTotal() - 1; i >= 0; i--)
{
string label = ObjectName(i);
if(StringSubstr(label, 0, 14) != "DivergenceLine")
continue;
ObjectDelete(label);
}
ObjectDelete("Symbol1");
ObjectDelete("Symbol2");
return(0);
}
//+------------------------------------------------------------------+
//| Custom indicator iteration function |
//+------------------------------------------------------------------+
int start()
{
string Symbol_1 = " ";
string Symbol_2 = " ";
ObjectSetText("Symbol1",Symbol_1,"","Arial Black",Lime);
ObjectSet("Symbol1",OBJPROP_XDISTANCE,3);
ObjectSet("Symbol1",OBJPROP_YDISTANCE,30);
ObjectSet("Symbol1",OBJPROP_COLOR,Red);
ObjectSet("Symbol1",OBJPROP_CORNER,"111");
ObjectSetText("Symbol2",Symbol_2,"6","Arial Black",Lime);
ObjectSet("Symbol2",OBJPROP_XDISTANCE,3);
ObjectSet("Symbol2",OBJPROP_YDISTANCE,50);
ObjectSet("Symbol2",OBJPROP_COLOR,Red);
ObjectSet("Symbol2",OBJPROP_CORNER,"111");
helper();
int countedBars = IndicatorCounted();
if(countedBars < 0)
countedBars = 0;
CalculateIndicator(countedBars);
return(0);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
void CalculateIndicator(int countedBars)
{
for(int i = Bars - countedBars; i >= 1; i--)
{
CalculateMA(i);
CatchBullishDivergenceMA(i + 2);
CatchBearishDivergenceMA(i + 2);
CalculateOsMA(i);
CatchBullishDivergence(i + 2);
CatchBearishDivergence(i + 2);
}
for( i = Bars - countedBars; i >= 1; i--)
{
```

```
setSignals(i);
}
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
int helper()
{i
nt counted_bars=IndicatorCounted();
int i;
int barras;
double _price;
double tmp;
double MinL=0;
double MaxH=0;
double Threshold=1.2;
//----
if(counted_bars>0) counted_bars--;
//barras = Bars;?
barras=Bars-counted_bars;
if (Mode_Fast)
barras=100;
i=0;
while(i<barras)
{M
axH=High[Highest(NULL,0,MODE_HIGH,period,i)];
MinL=Low[Lowest(NULL,0,MODE_LOW,period,i)];
//----
switch(price)
{c
ase 1: _price=Open[i]; break;
case 2: _price=Close[i]; break;
case 3: _price=High[i]; break;
case 4: _price=Low[i]; break;
case 5: _price=(High[i]+Low[i]+Close[i])/3; break;
case 6: _price=(Open[i]+High[i]+Low[i]+Close[i])/4; break;
case 7: _price=(Open[i]+Close[i])/2; break;
default: _price=(High[i]+Low[i])/2; break;
}V
alue=0.33*2*((_price-MinL)/(MaxH-MinL)-0.5) + 0.67*Value1;
Value=MathMin(MathMax(Value,-0.999),0.999);
Fish=0.5*MathLog((1+Value)/(1-Value))+0.5*Fish1;
buffer1[i]= 0;
buffer2[i]= 0;
//----
if((Fish<0) && (Fish1>0))
{i
f (Signals)
{O
bjectCreate("EXIT: "+DoubleToStr(i,0),OBJ_TEXT,0,Time[i],_price);
ObjectSetText("EXIT: "+DoubleToStr(i,0),"EXIT AT "+DoubleToStr(_price,4),7,"Arial",White);
}b
uy=0;
```

```
}i
f ((Fish>0) && (Fish1<0))
{i
f (Signals)
{O
bjectCreate("EXIT: "+DoubleToStr(i,0),OBJ_TEXT,0,Time[i],_price);
ObjectSetText("EXIT: "+DoubleToStr(i,0),"EXIT AT "+DoubleToStr(_price,4),7,"Arial",White);
}s
ell=0;
}i
f (Fish>=0)
{b
uffer1[i]=Fish;
buffer3[i]= Fish;
}e
lse
{b
uffer2[i]=Fish;
buffer3[i]= Fish;
}t
mp=i;
if ((Fish<-Threshold) &&
(Fish>Fish1) &&
(Fish1<=Fish2))
{
sell=1;
}i
f ((Fish>Threshold) &&
(Fish<Fish1) &&
(Fish1>=Fish2))
{
buy=1;
}V
alue1=Value;
Fish2=Fish1;
Fish1=Fish;
i++;
}f
or(i=0; i<barras; i++)
MA1buffer[i]=iMAOnArray(buffer3,Bars,MA1period,0,TypeMA1,i);
for(i=0; i<barras; i++)
MA2buffer[i]=iMAOnArray(MA1buffer,Bars,MA2period,0,TypeMA2,i);
//----
return(0);
}
void CalculateOsMA(int i)
{
OsMA[i] = buffer3[i];
//----
if(OsMA[i] > 0)
{
upOsMA[i] = OsMA[i];
downOsMA[i] = 0;
```

```
}
else
if(OsMA[i] < 0)
{
downOsMA[i] = OsMA[i];
upOsMA[i] = 0;
}
else
{
upOsMA[i] = 0;
downOsMA[i] = 0;
}
}
void CalculateMA(int i)
{
MA[i] = MA1buffer[i];
MA2[i] = MA2buffer[i];
//----
if(MA[i] > 0)
{
upMA[i] = MA2[i];
downMA[i] = 0;
}
else
if(MA[i] < 0)
{
downMA[i] =MA2[i];
upMA[i] = 0;
}
else
{
downMA[i] = 0;
upMA[i] = 0;
}
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
void CatchBullishDivergence(int shift)
{
if(IsIndicatorTrough(shift) == false)
return;
int currentTrough = shift;
int lastTrough = GetIndicatorLastPeakMA(shift);
bullishDivergence[currentTrough] = OsMA[currentTrough];
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
void CatchBearishDivergence(int shift)
{
if(IsIndicatorPeak(shift) == false)
return;
```

```
int currentPeak = shift;
int lastPeak = GetIndicatorLastTroughMA(shift);
bearishDivergence[currentPeak] = OsMA[currentPeak];
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
bool IsIndicatorPeak(int shift)
{
if(OsMA[shift] > 0 && OsMA[shift] > OsMA[shift+1] && OsMA[shift] > OsMA[shift-1])
{
for(int i = shift + 1; i < Bars; i++)
{
if(OsMA[i] < 0)
return(true);
if(OsMA[i] > OsMA[shift])
break;
}
}
return(false);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
bool IsIndicatorTrough(int shift)
{
if(OsMA[shift] < 0 && OsMA[shift] < OsMA[shift+1] && OsMA[shift] < OsMA[shift-1])
{
for(int i = shift + 1; i < Bars; i++)
{
if(OsMA[i] > 0)
return(true);
if(OsMA[i] < OsMA[shift])
break;
}
}
return(false);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
int GetIndicatorLastPeak(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(OsMA[i] >= OsMA[i+1] && OsMA[i] > OsMA[i+1] &&
OsMA[i] >= OsMA[i-1] && OsMA[i] > OsMA[i-1])
return(i);
}
return(-1);
}
//+------------------------------------------------------------------+
//| |
```

```
//+------------------------------------------------------------------+
int GetIndicatorLastTrough(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(OsMA[i] <= OsMA[i+1] && OsMA[i] < OsMA[i+1] &&
OsMA[i] <= OsMA[i-1] && OsMA[i] < OsMA[i-1])
return(i);
}
return(-1);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
void DrawPriceTrendLine(datetime x1, datetime x2, double y1,
double y2, color lineColor, double style)
{
string label = "DivergenceLine2.1# " + DoubleToStr(x1, 0);
//ObjectDelete(label);
ObjectCreate(label, OBJ_ARROW, 0, x2, y1, x1, y2, 0, 0);
ObjectSet(label,OBJPROP_COLOR,lineColor);
ObjectSet(label,OBJPROP_SCALE,500);
ObjectSet(label,OBJPROP_RAY, true);
ObjectSet(label,OBJPROP_WIDTH,2);
if(lineColor == Violet)
ObjectSet(label,OBJPROP_ARROWCODE,234);
if(lineColor == Blue)
ObjectSet(label,OBJPROP_ARROWCODE,233);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
void DrawIndicatorTrendLine(datetime x1, datetime x2, double y1,
double y2, color lineColor, double style)
{
int indicatorWindow = WindowFind("TestSignalsM15_V1_MQL4");/*(" + fastEMA +
"," + slowEMA + "," + signal + ")");
Alert("indicatorWindow = ",indicatorWindow);*/
if(indicatorWindow < 0 )
return;
string label = "DivergenceLine2.1$# " + DoubleToStr(x1, 0);
ObjectDelete(label);
}
//+------------------------------------------------------------------+
//Return DrawLine Indicator value
double GetDrawIndicatorValue(int y1,int y2,double z1, double z2)
{
int zz1,zz2;
int val = (y1 - y2) + 1;
zz2 = MathAbs(z2*10000);
zz1 = MathAbs(z1*10000);
int z = zz1 - zz2 ;
return (MathAbs(z / val));
```

```
}
double GetDrawIndicatorValueLine(int shiftDev, int lastMA, int Deviation)
{
double z = MA[lastMA]* 10000;
double val = ( z + (shiftDev * Deviation))/ 10000 ;
return (val);
}
void setSignals(int shift)
{
if (shift <= 1000)
{
int lastPeakMA = GetIndicatorLastPeakMA(shift)-1;
int lastTroughMA = GetIndicatorLastTroughMA(shift)-1;
int lastPeakMAPOS = GetIndicatorLastPeakMAPOS(shift)-1;
int lastTroughMAPOS = GetIndicatorLastTroughMAPOS(shift)-1;
int lastPeakMA2 = GetIndicatorLastPeakMA2(shift)-1;
int lastTroughMA2 = GetIndicatorLastTroughMA2(shift)-1;
int lastPeakMA2POS = GetIndicatorLastPeakMA2POS(shift)-1;
int lastTroughMA2POS = GetIndicatorLastTroughMA2POS(shift)-1;
}}
//MA
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
void CatchBullishDivergenceMA(int shift)
{
if(IsIndicatorTroughMA(shift) == false)
return;
int currentTrough = shift;
int lastTrough = GetIndicatorLastTrough(shift);
bullishDivergenceMA[currentTrough-1] = MA[currentTrough];
if(lastTrough > currentTrough-1)
{
DrawIndicatorTrendLine(Time[lastTrough],Time[currentTrough-1],OsMA[lastTrough
], MA[currentTrough-1], Blue, STYLE_SOLID);
bullishDivergenceMADeviation[currentTrough-1] = GetDrawIndicatorValue(
lastTrough, currentTrough,OsMA[lastTrough],MA[currentTrough-1]);
}
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
void CatchBearishDivergenceMA(int shift)
{
if(IsIndicatorPeakMA(shift) == false)
return;
int currentPeak = shift;
int lastPeak = GetIndicatorLastPeak(shift);
bearishDivergenceMA[currentPeak-1] = MA[currentPeak];
if(lastPeak > currentPeak-1)
{D
rawIndicatorTrendLine(Time[lastPeak],Time[currentPeak-1], OsMA[lastPeak],MA[
currentPeak-1], Violet, STYLE_SOLID);
```

```
bearishDivergenceMADeviation[currentPeak-1] = GetDrawIndicatorValue(lastPeak,
currentPeak,OsMA[lastPeak],MA[currentPeak-1]);
}
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
bool IsIndicatorPeakMA(int shift)
{
if(MA[shift] > 0 && MA[shift] > MA[shift+1] && MA[shift] > MA[shift-1])
{
for(int i = shift + 1; i < Bars; i++)
{
if(MA[i] < 0)
return(true);
if(MA[i] > MA[shift])
break;
}
}
return(false);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
bool IsIndicatorTroughMA(int shift)
{
if(MA[shift] < 0 && MA[shift] < MA[shift+1] && MA[shift] < MA[shift-1])
{
for(int i = shift + 1; i < Bars; i++)
{
if(MA[i] > 0)
return(true);
if(MA[i] < MA[shift])
break;
}
}
return(false);
}
int GetIndicatorLastMultiplePeakMA(int shift)
{
int Res = 0;
for(int i = shift + 2; MA[i] > 0; i++)
{
if(MA[i+1] > MA[i])
{
if(Res == 0)
Res = i+1;
if(Res != 0 && MA[i+1] > MA[Res])
Res = i+1;
}
}
return(Res);
}
```

```
int GetIndicatorLastMultTroughMA(int shift)
{
int Res = 0;
for(int i = shift + 2; MA[i] < 0; i++)
{
if(MA[i+1] < MA[i])
{
if(Res == 0)
Res = i+1;
if(Res != 0 && MA[i+1] < MA[Res])
Res = i+1;
}
}
return(Res);
}
int GetIndicatorLastMultiplePeakMA2(int shift)
{
int Res = 0;
for(int i = shift + 2; MA2[i] > 0; i++)
{
if(MA2[i+1] > MA2[i])
{
if(Res == 0)
Res = i+1;
if(Res != 0 && MA2[i+1] > MA2[Res])
Res = i+1;
}
}
return(Res);
}
int GetIndicatorLastMultTroughMA2(int shift)
{
int Res = 0;
for(int i = shift + 2; MA2[i] < 0; i++)
{
if(MA2[i+1] < MA2[i])
{
if(Res == 0)
Res = i+1;
if(Res != 0 && MA2[i+1] < MA2[Res])
Res = i+1;
}
}
return(Res);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
int GetIndicatorLastPeakMA(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(MA[i] > MA[i+1] && MA[i] > MA[i+1] && MA[i] > 0 &&
```

```
MA[i] > MA[i-1] && MA[i] > MA[i-1])
return(i);
}
return(-1);
}
int GetIndicatorLastPeakMAPOS(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(MA[i] > MA[i+1] && MA[i] > MA[i+1] &&
MA[i] > MA[i-1] && MA[i] > MA[i-1])
return(i);
}
return(-1);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
int GetIndicatorLastTroughMA(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(MA[i] < MA[i+1] && MA[i] < MA[i+1] && MA[i] < 0 &&
MA[i] < MA[i-1] && MA[i] < MA[i-1])
return(i);
}
return(-1);
}
int GetIndicatorLastTroughMAPOS(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(MA[i] < MA[i+1] && MA[i] < MA[i+1] &&
MA[i] < MA[i-1] && MA[i] < MA[i-1])
return(i);
}
return(-1);
}
//------------------------------------------------------------------
//------------------------------------------------------------------
int GetIndicatorLastPeakMA2(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(MA2[i] > MA2[i+1] && MA2[i] > MA2[i+1] && MA2[i] > 0 &&
MA2[i] > MA2[i-1] && MA2[i] > MA2[i-1])
return(i);
}
return(-1);
}
int GetIndicatorLastPeakMA2POS(int shift)
{
for(int i = shift + 2; i < Bars; i++)
```

```
{
if(MA2[i] > MA2[i+1] && MA2[i] > MA2[i+1] &&
MA2[i] > MA2[i-1] && MA2[i] > MA2[i-1])
return(i);
}
return(-1);
}
//+------------------------------------------------------------------+
//| |
//+------------------------------------------------------------------+
int GetIndicatorLastTroughMA2(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(MA2[i] < MA2[i+1] && MA2[i] < MA2[i+1] && MA2[i] < 0 &&
MA2[i] < MA2[i-1] && MA2[i] < MA2[i-1])
return(i);
}
return(-1);
}
int GetIndicatorLastTroughMA2POS(int shift)
{
for(int i = shift + 2; i < Bars; i++)
{
if(MA2[i] < MA2[i+1] && MA2[i] < MA2[i+1] &&
MA2[i] < MA2[i-1] && MA2[i] < MA2[i-1])
return(i);
}
return(-1);
}
//------------------------------------------------------------------
//------------------------------------------------------------------
```