# İNDİKATÖRLER

## İÇİNDEKİLER

# İNDİKATÖRLER

## Ağırlıklı Kapanış (WC)

Ağırlıklı Kapanış (Weighted Close); yüksek, düşük ve iki katına çıkarılmış kapanış fiyatının ortalamasıdır. Weighted Close, kapanış fiyatına daha yüksek ağırlık verilerek yüksek, düşük ve kapanış fiyatlarının ortalaması olarak hesaplanır:

Ağırlıklı Kapanış = ((Kapanış Fiyatı * 2) + Yüksek Fiyat + Düşük Fiyat) / 4

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.WeightedClose(candles); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# Alligator İndikatörü

Alligator İndikatörü, Bill Williams tarafından geliştirilmiştir ve hareketli ortalamalar ile fraktal geometriyi birleştirir. Gösterge, analistlerin piyasanın bir trendi olup olmadığını belirlemelerine yardımcı oluyor. Mavi çizgi (Alligator Çenesi), kırmızı çizgi (Alligator Dişleri) ve yeşil çizgi (Alligator Dudakları) olmak üzere 3 çizgiden oluşur. Her biri, farklı geriye yönelik aralıklar ve kullanıcı tarafından ayarlanabilecek farklı ofsetlere sahiptir. Alligator Göstergesinin kullanımı kolaydır, çizgilerin birbirine ne kadar yakın ya da uzakta olduğu, bir timsahın ağzını açıp kapatmasına benzetilerek işlem kurguları kurulabilir. Bu gösterge, diğer analiz teknikleriyle birlikte kullanılabilir.

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.
public string Symbol="ASELS";
public string Period="1";
public string SonYon="";
public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var indikator=Engine.Alligator(candles); // 3 çizgisi mevcuttur.
   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)

{
}
```

# AROON

## AROON İndikatörü Nedir?

AROON indikatörü fiyatları analiz ederek bir trendin var olup olmadığını bulmayı amaçlayan bir indikatördür. AROON indikatörü belirlenen periyot kadar önce oluşmuş bir tepe fiyatı ile aynı vadedeki dip fiyatı arasında geçen zamanı baz alarak oluşturulur. AROON indikatörünün "UP" ve "DOWN" olmak üzere iki çizgisi bulunur. AROON UP hesaplanan tepeden itibaren geçen zamanı, AROON DOWN ise hesaplanan dipten itibaren geçen zamanı belirtir.

## AROON Nasıl Kullanılır?

AROON indikatörünün referans değerleri 0 ile 100 olarak belirlenmiştir. AROON UP göstergesinin 100 seviyesine yaklaşması veya değmesi durumu ile AROON DOWN göstergesinin 0-30 bandında seyretmesi yukarı trendin güçlü olduğunu göstermektedir. Aynı şekilde AROON DOWN göstergesinin 100 seviyesine yaklaşması veya değmesi durumu ile AROON UP göstergesinin 0-30 bandında seyretmesi aşağı trendin güçlü olduğunu göstermektedir. Başka bir kullanım şeklinde ise bu iki çizginin kesişimleri yorumlanıp sinyal üretilir. Eğer UP çizgisi DOWN çizgisini kesip yukarı yönlü giderse "Alım" , aksi durumda ise "Satım" sinyali dikkate alınır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametresini tanımlıyoruz.
        int Parametre=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
        //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.AROON(candles,Parametre);
                var indicatorUP=indicator[0];
                var indicatorDOWN=indicator[1];
        //Eğer indikatörün UP çizgisi DOWN çizgisini aşağıdan yukarı keserse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.Intersect(indicatorUP,indicatorDOWN, "up") && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün UP çizgisi DOWN çizgisini yukarıdan aşağı keserse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.Intersect(indicatorUP,indicatorDOWN, "down") && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Aroon Osilatörü

Aroon Osilatörü, mevcut bir trendin gücünü ve devam etme olasılığını ölçmek için Aroon Göstergesinin (Aroon Up ve Aroon Down) özelliklerini kullanan trend takip eden bir göstergedir . Sıfırın üstündeki okumalar bir yükseliş trendi olduğunu gösterirken, sıfırın altındaki okumalar bir düşüş trendi olduğunu gösterir . Yatırımcılar potansiyel eğilim değişikliklerini göstermek için sıfır hat geçişlerini izler. Ayrıca, güçlü fiyat hareketlerini bildirmek için 50'nin üzerinde veya -50'nin altında büyük hamleler izliyorlar.

## Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

//İndikatörün parametresini tanımlıyoruz.
        int Parametre=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
        //SubscribePrice("Sembol") metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles("Sembol","Periyot") metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,"1");

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.AroonOscillator(candles,Parametre);

                //Eğer indikatörün bir önceki değeri -50' den küçükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)<-50 && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

                //Eğer indikatörün bir önceki değeri 50'den büyükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)>50 && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Birikim Salınım Endeksi

Birikim Salınım Endeksi İndikatörü, birbiri ardına gelen Salınım Endeksi değerlerinin değişimi göz önüne alınarak oluşturulmuş bir indikatördür. Salınım Endeksi'ne kıyasla daha uzun dönemli bir görünüm elde etmeyi amaçlar. Var olan trendin yönü hakkında ipuçları içeren Birikim Salınım Endeksi, uzun dönemli bir yükselen trend sırasında pozitif bir eğime, alçalan bir trend sırasında ise negatif eğime sahip olacaktır. Piyasanın kararsız olduğu yatay trendsizlik döneminde ise Birikim Salınım Endeksi de yatayda kalacaktır. Birikim Salınım Endeksi ile çalışırken gösterge üzerinde trend çizimlerinden yararlanmak mümkündür. Fiyatlarda olduğu gibi Birikim Salınım Endeksi'nin tepe ve diplerinden atılacak trendlerle birlikte trend kırılımları ve trend dönüş noktaları da yakalanabilir. Genellikle vadeli piyasalar için kullanılsa da hisse senetleri için de kullanılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametresinin tanımlaması
        int Parametre=3;

//Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
//SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

// Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.AccumulationSwingIndex(candles,Parametre);

//Eğer indikatörün bir önceki değeri, iki önceki değerinden küçük eşitse ve SonYon BUY 'a eşit değilse Alış Yap.
                if(Engine.PreviousValue(indicator,1)<=Engine.PreviousValue(indicator,2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
//Eğer indikatörün bir önceki değeri, iki önceki değerinden büyük eşitse ve SonYon SELL 'e eşit değilse Satış Yap.
                else if(Engine.PreviousValue(indicator,1)>=Engine.PreviousValue(indicator,2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Birikim/Dağıtım (A/D) İndikatörü

Birikim / dağıtım, bir hisse senedinin birikip birikmediğini veya dağıtıldığını değerlendirmek için hacim ve fiyat kullanan kümülatif bir göstergedir. Birikim / dağıtım ölçüsü, hisse senedi fiyatı ile hacim akışı arasındaki farklılıkları belirlemeye çalışır. Bunu yaparken kapanış fiyatının günün en yükseği ve en düşüğüne olan uzaklığının hacimle olan ilişkisini inceler. Bu, bir eğilimin ne kadar güçlü olduğuna dair fikir verir. Fiyat yükseliyor ancak gösterge düşüyorsa, bu, alım veya birikme hacminin fiyat artışını desteklemek için yeterli olmayabileceğini ve fiyat düşüşünün yaklaşabileceğini gösterir.

Fiyatlardaki değişimin ne kadar yüksek işlem hacmiyle gerçekleşirse o kadar güçlü olacağından hareketle bir trendin var olup olmadığını ve var olan trendin sürüp sürmeyeceğini araştırır. Bu yüzden göstergedeki düşüşün ardından tekrar yükselmeye başlaması "al" sinyali, yükselişin ardından tekrar düşüşe geçmesi ise "sat" sinyali olarak kabul edilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.AccumulationDistribution(candles);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden küçük eşitse ve SonYon BUY 'a eşit değilse Alış Yap.
                if(Engine.PreviousValue(indicator,1)<=Engine.PreviousValue(indicator,2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
//Eğer indikatörün bir önceki değeri, iki önceki değerinden büyük eşitse ve SonYon SELL 'e eşit değilse Satış Yap.
                else if(Engine.PreviousValue(indicator,1)>=Engine.PreviousValue(indicator,2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Bollinger Bantları

Finansal piyasalarda alım satım seviyelerini bulmaya yardımcı olması amacıyla ortaya çıkan gösterge, ilk olarak John Bollinger tarafından 1980 yılında piyasa değişimlerini daha doğru ölçebilmek için tasarlanmıştır.

Bollinger Bantları, piyasadaki değişimleri daha iyi ölçümlemek ve daha hızlı aksiyon alabilmek için kullanılmaktadır. Bollinger Bantları,

-Trend başlangıcının belirlenmesi

-Trendin yönü

-Dip ve tepe seviyeleri

-Fiyat hedefleri

-Fiyatlardaki sıkılaşma

-Fiyatların volatilitesi

gibi değişkenleri ölçümlemek için kullanılabilmektedir.

Bollinger Bantları alt, orta ve üst olmak üzere üç çizgiden oluşur. Orta bant 20 periyotluk basit hareketli ortalamayı baz alarak hareket eder. Alt ve üst bantlar 20 periyotluk hareketli ortalamanın aşağı ve yukarı yönde 2 standart sapma değeri kadar kaydırılması ile oluşur. Üst ve alt bantların amacı ortalama kapanış fiyatlarının en yüksek ve en düşük noktalarının belirlenmesidir. Alt ve üst bant destek/direnç seviyeleri olarak yorumlanmaktadır.

John Bollinger 20 periyotluk hareketli ortalamayı önermektedir. Ancak farklı ürün ve piyasalarda bu parametre daha değişken kullanılabilir. Genel kabul gören kullanımda ortalama değiştikçe sapma oranı da buna paralel değiştirilmeli görüşü hakimdir.

Bollinger Bantlarının yorumlanması :

-Bollinger çizgileri daraldıysa piyasadaki hareketin kararsız ve yatay olduğu anlamına gelir. Bu daralmanın ardından aşağı ya da yukarı yönlü bir trend olacağı varsayılmaktadır.

-Fiyatların alt ve üst bandı geçmesi o yönde trendin devam edeceğine yorumlanır.

-Fiyatların üst veya alt bandın dışında çıkması durumunda kısa sürede tekrar bu banda gireceğine yorumlanır.

-Eğer fiyat alt banttan uzaklaşır ve 20 periyotluk hareketli ortalamanın üzerine çıkarsa yükseliş eğilimi göstereceği varsayılır.

Alım satım kararları verirken tek bir değişkene bakmak beklenilmeyen sonuçlar doğurabilir.

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.BollingerBands(candles, 20, 2); // 3 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Bollinger Bantları Genişliği (BBW)

Bollinger Bantları Genişliği (BBW), standart Bollinger Bantları göstergesinden türetilen bir teknik analiz göstergesidir. Bollinger Bantları, bir menkul kıymetin fiyatına göre çizilen üç satırlık bir bant oluşturan bir volatilite göstergesidir. Orta çizgi genellikle 20 günlük Basit Hareketli Ortalamadır. Üst ve alt bantlar tipik olarak Basit Hareketli Ortalamanın (orta hat) üstünde ve altında 2 standart sapmadır. Bollinger Bantları Genişliği, üst ve alt bantlar arasındaki genişliği nicel olarak ölçmenin bir yoludur. BBW, bazı durumlarda alım satım sinyallerini tanımlamak için kullanılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
 public string SonYon="";
public string Sembol="VAKBN";
public  string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre=20;
   int Parametre2=2;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
          {

    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.BollingerWidth(candles,Parametre,Parametre2);

                //Eğer indikatörün önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap

                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                     }

            //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap

                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                     }
        }
```

## Chaikin Osilatörü (CHO)

Chaikin Osilatörü, özünde, bir göstergenin göstergesidir. Chaikin Osilatörü Birikim / Dağıtım (AD) indikatörünü alır ve çizgiye değişen uzunlukta iki Üstel Hareketli Ortalama uygular. Chaikin Osilatörünün değeri daha sonra, ADL'nin daha uzun EMA terimi, ADL'nin daha kısa EMA teriminden çıkarılarak elde edilir. Sonuçta bu, ADL'nin momentumunu, pozitif ve negatif değerler arasında dalgalanan bir çizgi çizerek ölçmenin bir yolu olarak kullanılır. Momentumdaki değişikliklerin farkında olmak, bir yatırımcı veya teknik analistin trend değişikliklerini tahmin etmesine yardımcı olabilir, çünkü momentumdaki değişiklikler genellikle trenddeki değişikliklerden önce gelir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;


        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ChaikinOscillator(candles);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Chaikin Para Akışı (CMF)

Chaikin Para Akışı (CMF), belirli bir süre boyunca Para Akışı Hacmini ölçmek için kullanılan teknik bir analiz göstergesidir. Para Akışı Hacmi (Marc Chaikin tarafından da yaratılan bir kavram), kıymetin alım ve satım baskısını tek bir dönem boyunca ölçmek için kullanılan bir metriktir. CMF daha sonra kullanıcı tarafından belirlenen bir yeniden inceleme dönemi boyunca Para Akışı Hacmini toplar. Herhangi bir yeniden inceleme süresi kullanılabilir, ancak en popüler ayarlar 20 veya 21 gündür. Chaikin Para Akışı'nın Değeri 1 ile -1 arasında dalgalanır. CMF, alış ve satış baskısındaki değişiklikleri daha fazla ölçmenin bir yolu olarak kullanılabilir ve gelecekteki değişiklikleri ve dolayısıyla yatırım fırsatlarını tahmin etmeye yardımcı olabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;


        //İndikatörün parametresinin tanımlaması
        int Parametre=21;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ChaikinMoneyFlow(candles,Parametre);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Chaikin Volatilitesi (CHV) İndikatörü

Chaikin Volatilitesi (CHV), Marc Chaikin tarafından geliştirilen bir göstergedir. Belirlenebilecek belirli bir zaman diliminde fiyatın yükseklerine ve düşüklerine bakarak volatiliteyi ölçer. Kesin hesaplama, bir Üstel Hareketli Ortalama (EMA) kullanır, ancak genel olarak, yüksek ve düşükler arasındaki aralık ne kadar geniş olursa, o kadar yüksek fiyat dalgalanması olur. Gösterge, olası bir trend değişimini (eğer volatile daha uzun süre azalırsa) tahmin etmek veya riski değerlendirmek için (eğer volatilite aniden artarsa, endişeli yatırımcılara işaret vererek) çeşitli şekillerde kullanılabilir. Göstergenin yorumlanmasında en yaygın yol, göstergenin yükselmesinin senette alım yapıldığı, düşmesinin ise senetten çıkıldığı şeklinde yorumlanmasıdır. CHV genellikle diğer sinyaller ve analiz teknikleriyle birlikte kullanılır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametresinin tanımlaması
        int Parametre1=10;
        int Parametre2=10;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ChaikinVolatility(candles,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Chande Momentum Osilatörü (CMO)

## Chande Momentum Osilatörü (CMO) Nedir?

CMO, genel olarak aşırı alış-satış bölgelerini belirleyen bir indikatördür. CMO açılımı "Chande Momentum Oscilator"'dür. Belirli bir vade arasındaki fiyat değişimlerini hesaplayarak trendin yönünü ve gücünü ölçüp hesaplanan değerleri bir aralık arasında gösterir.

## Chande Momentum Osilatörü (CMO) Nasıl Kullanılır?

CMO'nun referans değerleri genel olarak +50 ve -50 olarak belirlenmiştir. +50 seviyesi aşırı alımın olduğunu ve kısa bir vadede paritenin düşme eğilimine girebileceği, -50 seviyesi ise aşırı satımın olduğunu ve kısa bir vadede paritenin yükselme eğilimine girebileceği hakkında fikir verir. Başka bir kullanım şeklinde ise CMO'nun üzerine atılan bir hareketli ortalama ile gerçekleşen kesişimlerine de bakılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametrelerinin tanımlanması
        int Parametre=20;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ChandeMomentum(closed,Parametre);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Commodity Channel Endeksi (CCI)

## Commodity Channel Endeksi (CCI) İndikatörü Nedir?

CCI indikatörü fiyatların ortalamadan ne ölçüde saptığı bulup bu değerleri belli bir aralıkta tutan bir aşırı alım-satım indikatörüdür.

D. Lambert tarafından geliştirilen CCI indikatörünün açılımı "Commodity Channel Index"'tir. Fiyatların ortalamadan sapma değerlerini kullanan bu indikatör genellikle trendin zayıf olduğu zamanlarda tercih edilir.

## Commodity Channel Endeksi (CCI) Nasıl Kullanılır?

Genel olarak -100 ve +100 olarak referans seviyeleri kullanılmaktadır. Eğer CCI'ın değeri -100'ü aşağıdan yukarıya doğru keserse "Alım", 100 seviyesini yukarıdan aşağıya doğru keserse ise "Satım" sinyalinin oluştuğu çıkarımı yapılabilir. Eğer istenirse fiyatlar ile CCI'ın değerleri arasında uyumsuzluk yöntemi de uygulanabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametresinin tanımlaması
        int Parametre=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.CommodityChannelIndex(candles,Parametre);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Commodity Selection Endeksi (CSI)

Commodity Selection Endeksi , momentumu belirlemek için eğilim gücünü ve oynaklığını dikkate alır. Özellikle fiyatta büyük hareketler ve potansiyel olarak büyük bir yatırım getirisi olan emtia vadeli işlem sözleşmelerinin kısa vadeli ticareti için özel olarak geliştirilmiştir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        int Parametre2=100;
        int Parametre3=2500;
        int Parametre4=25;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
var indicator =  Engine.CommoditySelectionIndex(candles,Parametre1,Parametre2,Parametre3,Parametre4);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# DEMA

Çift Üstel Hareketli Ortalama (DEMA), gecikmeyi azaltmak ve yanıt vermeyi artırmak amacıyla Patrick Mulloy tarafından geliştirildi. Bu hızlı hareket eden ortalama, yatırımcıların trend dönüşlerini hızlı bir şekilde bulmalarını ve yeni oluşan trendlere daha iyi girmeleri sağlar. Gösterge esasen Üstel Hareketli Ortalama(EMA) 'yı temel alır, ancak fiyatı daha yakından takip eder. Hesaplaması ve kullanımı Hull Hareketli Ortalama (HMA)'ya benzerdir. Analistlerin hakim trendi belirlemesine yardımcı olur ve genellikle diğer sinyaller ve analiz teknikleri ile birlikte kullanılır.

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.DEMA(C, 7, 13); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# Denge İşlem Hacmi (OBV)

## Denge İşlem Hacmi (OBV) İndikatörü Nedir?

OBV indikatörü fiyatlardaki değişimin yönüyle işlem hacmini ilişkilendiren bir indikatördür.

OBV indikatörünün açılımı "On Balance Volume"'dur. Kapanış fiyatının bir önceki kapanış fiyatının üzerinde olmasıyla belirtilen zaman periyotunda yaratılan işlem hacmi pozitif olarak işleme dahil edilirken, kapanış fiyatının bir önceki kapanış fiyatının altında kaldığı zamanda ise hacim negatif olarak hesaba katılmaktadır.

## Denge İşlem Hacmi (OBV) İndikatörü Nasıl Kullanılır?

Eğer kullanılan finansal araçta talep artmaya başlarsa bu süreç OBV'nin değerini artırmaya başlayacaktır. İşleme girilmek istenen yönün OBV'nin yönü ile teyit edilmesi verilen kararın daha sağlıklı olmasına yardımcı olabilir. Bunun yanında, OBV indikatörü ile fiyat arasındaki uyumsuzluk da işlem yönü tayin etmede kullanılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.OnBalanceVolume(candles);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Dikey Yatay Filtre (VHF)

Dikey Yatay Filtre (VHF), fiyatlardaki hareketlerin bir trend içerip içermediğini araştıran bir göstergedir. Bir trend var mıdır, varsa ne kadar güçlüdür gibi sorulara cevap aramaktadır. Son günün en yüksek ve en düşük değerleri farkının, kapanış fiyatı ile önceki günün kapanış fiyatı arasındaki farka bölünmesi ile bulunur.

0 ile 1 değerleri arasında değişen bu göstergenin yorumlanmasında VHF'nin yükseldiği bölgelerde trendin güçlendiği, alçaldığı bölgelerde ise zayıfladığı kabul edilir. VHF'nin yükselişi ise trendin yönünden bağımsızdır. Sadece aşağı veya yukarı, var olan trendin güçlendiğine işaret etmektedir. VHF'nin çok yüksek veya çok düşük değerlere ulaşması da diğer göstergelerden biraz daha farklı değerlendirilmelidir. Yüksek değerlere ulaşan bir VHF, var olan trendin bir süre sonra zayıflayarak yataya gireceğine işaret ederken çok düşük değerlere ulaşan bir VHF de yeni bir trendin oluşmasına hazır olunması gerektiğine işaret etmektedir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=28;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.VerticalHorizontalFilter(closed,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Directional İndikatörü (DI + -)

Directional İndikatörü'nde pozitif ve negatif olmak üzere 2 ayrı çizgi vardır. Bu değer, bir önceki güne göre hareketin yönü yukarı ise pozitif, aşağı ise negatif olacaktır. +DM ve -DM değerlerini bulduktan sonra ise +DM değerinden DI+, -DM değerinden de DI- türetilir.

Yorumlama açısından genel kullanım DI+ göstergesinin DI- yi yukarı doğru keserek üzerine çıkmasıyla "al" sinyalinin üretildiği, DI- eğrisinin DI+ eğrisini yukarı doğru keserek üzerine çıkmasıyla ise "sat" sinyalinin üretildiği şeklindedir. Ancak trendin varlığında oldukça etkili alım satım noktaları veren DI+DI- trendin olmadığı dönemlerde çok sık kesişerek hatalı sinyaller de üretebilmektedir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametresini tanımlıyoruz.
        int Parametre=14;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice("Sembol") metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
 //GetCandles("Sembol","Periyot") metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.DirectionalIndicator(candles,Parametre);
                var indicatorArtı=indicator[0];
                var indicatorEksi=indicator[1];

        //Eğer indikatörün UP çizgisi DOWN çizgisini aşağıdan yukarı keserse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.Intersect(indicatorArtı,indicatorEksi, "up") && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün UP çizgisi DOWN çizgisini yukarıdan aşağı keserse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.Intersect(indicatorArtı,indicatorEksi, "down") && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Directional Movement Endeksi (DMI)

Veri terminallerinde +Dİ –Dİ, dm, isimleriyle yer alan Directional Movement Endeksi, J.Welles Wilder'in geliştirdiği bir indikatördür. Trend göstergeleri arasında yer almaktadır.

Trendin yönü hakkında fikir verir. Grafik üzerinde 2 çizgi şeklinde gösterilir. +Dİ alıcıları –Di satıcıları gösterir diyebiliriz. +Di fiyatın yukarı yönlü hareketini gösterirken, -Di aşağı yönlü fiyat hareketlerini göstermektedir.

En temel kullanım şekli ile +Dİ'nin –Dİ'yi yukarı kırması "alım", +Dİ'nin –Dİ'yi aşağı kırması "satım" sinyalidir.

## Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametrelerinin tanımlanması
        int Parametre=14;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.DirectionalMovement(candles,Parametre);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Ease Of Movement İndikatörü

Bu göstergenin amacı düşük bir işlem hacmi ile kolayca hareketlenen senetleri bulmakla birlikte asıl amacı fiyatların işlem hacmine olan duyarlılığını ölçmektir.

Gösterge hesaplanırken elde edilmesi gereken ilk değer " orta nokta hareketi " bugünün yükseği ve düşüğü toplamının ikiye bölümünden, dünün yüksek düşük farkının ikiye bölümünün çıkarılmasıyla elde edilir. İkinci değer olan " kutu oranı" ise bugün ki hacmin bugünün yüksek ve düşük farkına bölünmesiyle bulunur. Bulunan ilk değerin ikinciye bölümü de göstergenin değerini verir.

Göstergenin yorumunda, yüksek gösterge değeri düşük işlem hacminin bile önemli miktarda bir yükselişe yol açtığı bölgeleri, düşük gösterge değeri ise düşük işlem hacminin bile önemli miktarda düşüşe yol açtığı bölgeleri ifade eder. Fiyat hareketliliğinin düşük olduğu ya da yüksek hacimle gerçekleştiği bölgelerde ise gösterge değeri sıfır seviyesi civarındadır. Diğer bir kullanımda ise 0 seviyesinin aşağıdan yukarı kesilmesi alım, tersi durumda ise satım noktası olarak kabul görmektedir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametrelerinin tanımlanması
        int Parametre=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
        //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.EaseOfMovement(candles,Parametre);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Ehler’s Distant Coefficient Filtresi

Ehler’s Distant Coefficient Filtresi, sabit olmayan verileri filtreleyen doğrusal olmayan bir filtredir. Filtre, medyan fiyat verilerinin ağırlıklı ortalamasını bulma tekniği olarak düşünülebilir: hesaplama mekanizması, medyan fiyatlara ve son fiyat değişikliklerine göre barlara ağırlıklar atar.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.EhlersDistCoefFilter(candles); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# Ehler's Filtresi

Ehlers Filtresi (EF), John Ehlers tarafından yazılmıştır. EF, değerini hesaplamak için cari fiyatları, önceki fiyatları (momentum uzunluğuna göre belirlenir) ve bunların bir zaman aralığı içindeki farklarını kullanır.

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.EhlersFilter(candles); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Elliot Dalga Osilatörü

Elliot Dalga Oscillatörü, 5 ve 34 günlük basit ortalamalar arasındaki farktır. Fiyat hareketlerinin dalgalar şeklinde ilerlediğini varsayar. Yeni zirve, bir önceki zirvenin üzerinde oldukça yükselişin devam edeceğini, altında kaldığında ise trendin düşüşe döneceğini varsayar.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=5;
        int Parametre2=34;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ElliotWaveOscillator(candles,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# En Düşük Düşük Değer (LLV)

## En Düşük Düşük Değer (LLV) İndikatörü Nedir?

LLV, belirli bir sembolün listelendiğinden bu yana tüm zamanların en düşük seviyesine ulaştığı anlamına gelir ve sembolün o belirli zamanında talebinin olmadığını gösterir. n dönemler üzerinden en düşük düşük fiyatı gösterir.

En Düşük Düşük Değer = n Dönemlik Minimum (Düşük)

## En Düşük Düşük Değer (LLV) İndikatörü Nasıl Kullanılır?

LLV indikatörü HHV indikatörü ile kullanımında basit fakat etkili bir teknik gözlem sonucu doğurabilmektedir. İkisi, farklı indikatörler tarafından gönderilen sinyalleri onaylayan veya çürüten, kendi başlarına veya destekleyici bir araç olarak kullanılabilir. Potansiyel geri dönüş noktalarına ipucu verebilirler, yatırımcının bir trendin tersine dönme olasılığını ve ani bir fiyat hareketinin beğenilebilirliğini tahmin etmesine yardımcı olabilirler.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.LLV(C,1); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# En Yüksek Yüksek Değer (HHV) İndikatörü

## En Yüksek Yüksek Değer (HHV) İndikatörü Nedir?

HHV, hisse senedinin listelendiğinden bu yana tüm zamanların en yüksek seviyesine ulaştığı anlamına gelir ve o hisse senedinin o belirli zamanında büyük talep olduğunu gösterir. En Yüksek Yüksek Değer anlamına gelen indikatör, n dönemler üzerinden en yüksek yüksek fiyatı gösterir.

En Yüksek Yüksek Değer = n Dönemlik Maksimum (Yüksek)

## En Yüksek Yüksek Değer (HHV) İndikatörü Nasıl Kullanılır?

HHV indikatörü LLV indikatörü ile kullanımında basit fakat etkili bir teknik gözlem sonucu doğurabilmektedir. İkisi, farklı indikatörler tarafından gönderilen sinyalleri onaylayan veya çürüten, kendi başlarına veya destekleyici bir araç olarak kullanılabilir. Potansiyel geri dönüş noktalarına ipucu verebilirler, yatırımcının bir trendin tersine dönme olasılığını ve ani bir fiyat hareketinin beğenilebilirliğini tahmin etmesine yardımcı olabilirler.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.HHV(C,1); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Envelope

Envelope iki tane hareketli ortalamadan oluşur. Bir hareketli ortalama yukarı, diğeri ise aşağı kaydırılır.

Envelope, bir piyasanın normal işlem görme bandının alt ve üst sınırlarını belirler. Üst banda ulaşıldığında satış, alt banda ulaşıldığında ise alış sinyali üretilir. Bantların aşağı ve yukarı kaydırma oranları hareketliliğine göre değişir. Piyasa ne kadar hareketli ise kaydırma oranlarının o kadar yüksek olması gerekir.

Fiyatlar bantlara değdikten sonra daha gerçekçi seviyelere gelme eğilimidedir. Bollinger Bantları ile aşağı yukarı aynı şekilde yorumlanabilir.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.Envelope(C, 25, 5); // 2 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# Fibonacci Bantları

Fibonacci Bantları göstergesi Leonardo Fibonacci tarafından geliştirilmiş bir takım sayı dizisine dayanmaktadır.

Dizideki sayınların kendinden önceki sayıya bölünmesiyle altın orana yaklaşılması ve altın oranın da hayatımızdaki objelerin içinde yer alması bu sayıları önemli ve gizemli kılmıştır. Fibonacci dizisinde yer alan altın oran, eski Mısırlılar tarafından bulunmuştur ve Yunanlılar da, Mısırlılar gibi bu sayıyı mimaride kullanmışlardır. Altın oran, bütünü oluşturan parçalar arasındaki geometrik orandır.

Fibonacci dizisi Finans sektöründe finansal alacakların değeri tahminlemede kullanılır. Teknik analiz uygulamalarında kullanılan oranlar genel olarak 1.618 ve 1.232 dir.

Fibonacci disizi bir ürünün hareketinin trendini belirlememize yardımcı olacak analizler sunmaktadır.

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.FibonacciBands(candles, 1); // 3 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Fiyat Değişim Oranı (ROC)

Fiyat Değişim Oranı (ROC), mevcut fiyatın belirli bir zamanda önceki fiyata göre değişimini yüzde(%) cinsinden ölçen momentuma dayalı bir teknik göstergedir. ROC göstergesi, fiyat değişiklikleri yukarı yöndeyse, gösterge yukarı doğru pozitif bölgeye ve fiyat değişiklikleri aşağı yönlü ise negatif bölgeye hareket edecek şekilde sıfıra karşı çizilir.

Gösterge; sapmaları, aşırı alım ve aşırı satım koşullarını ve merkez hattı geçişlerini belirlemek için kullanılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=12;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ROCPercent(closed,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Fiyat Değişim Puanı (ROC)

Fiyat Değişim Puanı, mevcut fiyatın belirli bir zamanda önceki fiyata göre değişimini puan cinsinden ölçen momentuma dayalı bir teknik göstergedir. ROC göstergesi, fiyat değişiklikleri yukarı yöndeyse, gösterge yukarı doğru pozitif bölgeye ve fiyat değişiklikleri aşağı yönlü ise negatif bölgeye hareket edecek şekilde sıfıra karşı çizilir.

Gösterge; sapmaları , aşırı alım ve aşırı satım koşullarını ve merkez hattı geçişlerini belirlemek için kullanılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=12;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ROCPoints(closed,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Fiyat Hacim Trendi (PVT)

Fiyat Hacim Trend göstergesi (PVT), para akışını ölçmek için kullanılan momentum temelli bir göstergedir. PVT, hacim birikimi olması bakımından başka bir teknik analiz aracı olan Denge İşlem Hacmi (OBV) ile benzerdir. OBV, yükselen ya da düşen bir gün olup olmadığına bağlı olarak toplam günlük hacmi ekler veya çıkarırken PVT yalnızca günlük hacmin bir bölümünü ekler veya çıkarır. PVT toplamına eklenen veya çıkartılan hacim miktarı, önceki günün kapanışına kıyasla mevcut günün fiyatının yükselme veya düşme miktarına bağlıdır. Fiyat Hacim Trendi (PVT) öncelikle trendleri teyit etmek için ve ayrıca farklılıklar nedeniyle olası işlem sinyallerini bulmak için kullanılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.PriceVolumeTrend(candles);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Fiyat Osilatörü Puanı

Fiyat Osilatörü Puanı göstergesi tıpkı MACD göstergesinde olduğu gibi biri uzun diğeri kısa iki hareketli ortalamanın karşılaştırılması suretiyle fiyatların yönünü belirlemeye çalışmaktadır. MACD'de genel olarak kullanılan 26 ve 12 periyotlarından farklı olarak istenilen 2 periyodun karşılaştırılmasında kullanılabilir. Yorumlamalarda daha kısa vadeli hareketli ortalamanın daha uzun vadeli hareketli ortalamadan uzaklaştığı yani "0" değerinin altında ya da üzerinde olduğu durumlar sırasıyla aşırı alım ve aşırı satım bölgeleri olarak değerlendirilebilir. Ayrıca bu kırılmalar alım ve satım sinyali olarak da yorumlanabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=10;
        int Parametre2=30;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

// Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
var indicator = Engine.PriceOscillatorPoints(closed,MovingAverageMethods.Exponential,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Fiyat Osilatörü Puanı Histogram

Fiyat Osilatörü Puanı göstergesine ek olarak hareketli ortalama eklenerek tekrar hesaplanmasıyla oluşur. Yorumlanması ise aynı şekildedir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=10;
        int Parametre2=30;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

        // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
var indicator = Engine.PriceOscillatorPoints(closed,MovingAverageMethods.Exponential,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Fiyat Osilatörü Yüzdesi (PPO)

Fiyat Osilatörü Yüzdesi göstergesi tıpkı MACD göstergesinde olduğu gibi biri uzun diğeri kısa iki hareketli ortalamanın karşılaştırılması suretiyle fiyatların yönünü belirlemeye çalışmaktadır. Yorumlamalarda daha kısa vadeli hareketli ortalamanın daha uzun vadeli hareketli ortalamadan uzaklaştığı yani "0" değerinin altında ya da üstünde olduğu durumlar sırasıyla aşırı alım ve aşırı satım bölgeleri olarak değerlendirilebilir. Ayrıca bu kırılmalar alım ve satım sinyali olarak da yorumlanabilir. Kullanılan bir diğer yöntem ise yine MACD de olduğu gibi Price Oscillator'ü kendi hareketli ortalaması ile beraber kullanarak al sat sinyallerini üretmektir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=10;
        int Parametre2=30;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.PriceOscillatorPercent(closed,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Fiyat Osilatörü Yüzdesi Histogram

Varlık performansını ve oynaklığı karşılaştırmak, fiyatların tersine dönmesine yol açabilecek farklılıkları tespit etmek, ve trend yönünü onaylamaya yardımcı olmak için kullanılır. PPO, hareketli ortalama yakınsama sapması (MACD) göstergesiyle aynıdır; ancak PPO, iki EMA arasındaki yüzde farkını ölçerken, MACD mutlak farkı ölçer. Merkez çizgisinin aşağıdan yukarıya doğru bir hareketi yükseliş olarak ve merkez çizgisinin yukarıdan aşağıya doğru bir hareketi düşüş olarak yorumlanır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması

public string SonYon="";
public string Sembol="VAKBN";
public int lot=1;
public string Periyot="1";

//İndikatörün parametrelerinin tanımlanması
public int Parametre1=10;
public int Parametre2=30;
public int AvrPeriod=9;
//Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
public void Load()
{
   //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
   SubscribePrice(Sembol);
}
//Fiyat değişikliklerinde girilen fonksiyon
public void PriceChanged(Tick t)
{
   //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
   var candles = GetCandles(Sembol,Periyot);
   //Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
   var closed=Engine.GetPriceList(candles,PriceFields.Close);
   // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
   var indicator =
Engine.PriceOscillatorPercentHistogram(closed,MovingAverageMethods.Exponential,AvrPeriod,Parametre1,Parametre
2);
   //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
   if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY")
   {
      SonYon="BUY";
      SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
   }
   //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
   else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL")
   {
      SonYon="SELL";
      SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
   }
}
```

## Öngörü Osilatörü

Öngörü Osilatörü, belirli bir zaman dilimi içerisinde fiyat hareketlerini lineer regresyon kullanarak inceleyip gelecek bardaki fiyat hareketini tahmin etmeye çalışan bir osilatördür. -10 ve 10 seviyelerini baz alan Öngörü Osilatörü; -10 seviyesinin altına indiğinde aşırı alım, 10 seviyesinin yukarısına çıktığında aşırı alım olarak kabul edilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre=5;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ForecastOscillator(closed,Parametre);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }


        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap

                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){

                  SonYon="SELL";

                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);

                }

        }
```

## FxSniper

FX Sniper, trend takip eden ve al/sat sinyalleri üreten bir göstergedir.

Yeşil renkli sinyal çizgisi, trendin yukarı yönlü olduğunu gösterir ve al sinyali verir. Kırmızı renkli sinyal çizgisi, trendin aşağı yönlü olduğunu gösterir ve sat sinyali verir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=2;
        int Parametre2=10;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.FxSniper(closed,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Göreceli Güç Endeksi (RSI)

## Göreceli Güç Endeksi (RSI) Nedir?

Göreceli Güç Endeksi(RSI), belirlenen zaman periyodundan itibaren günlerin bir öncekilerle kıyaslanarak uyumsuzluk ve aşırı alım-satım bölgeleri bulmayı amaçlayan bir indikatördür. RSI indikatörünün açılımı "Relative Strength Index"'tir. RSI indikatörü en çok kullanılan göstergelerden biridir ve istatistiksel olarak sinyallerinin diğer göstergelere göre daha başarılı olduğu hesaplanmıştır.

## Göreceli Güç Endeksi (RSI) Nasıl Kullanılır?

RSI göstergesinin kullanımı yaygın olarak 2'ye ayrılmaktadır. İlk olarak aşırı alım-satım noktalarını tespit etmek, ikinci olarak ise uyumsuzluk belirlenmesidir. Aşırı alım-satım odaklı kullanıldığında genel kabul görmüş 30 ve 70 seviyeleri tercih edilir. RSI değeri 30 seviyesinin altına indiğinde aşırı satımı, 70 değerinin üzerine çıktığında ise aşırı alımın gerçekleştiği çıkarımı yapılabilir. RSI'ın bu seviyelerdeki değerleri trendin kısa-orta vadeli bir sürede sonlanabileceğine işaret eder.RSI'ın yorumlanışında kullanılan ikinci yöntem ise uyumsuzluklardır. Eğer trend yukarı doğru yükseliyor ve RSI değerlerinde yeni oluşan dip ve tepelerin bir önceki dip ve tepelerden daha yüksekte oluşması beklenirken, aşağı devam eden bir trendde RSI değerlerinde dip ve tepelerin bir önceki dip vetepelerden daha aşağıda oluşması beklenir. Bu koşulların gerçekleşmediği durumlar uyumsuzluk olarak kabul edilir ve genellikle RSI'ın son tepeleri veya son diplerinin oluşturduğu eğim yönünde işleme girilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.RSI(closed,Parametre1);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Göreceli Momentum Endeksi (RMI)

## Göreceli Momentum Endeksi (RMI) Nedir?

Göreceli Momentum Endeksi, aşırı alım ve satım indikatörleri ile benzer özellikler sergiler. Trendin güçlü olduğu piyasalarda uzun süre aşırı alım ve satım seviyelerinde kalacaktır. Göreceli Momentum Endeksi, RSI'a dayandığından, aynı yorum yöntemlerinin çoğu kullanılabilir. Aslında, bunların çoğunda RMI, RSI'dan çok daha nettir.

## Göreceli Momentum Endeksi (RMI) Nasıl Kullanılır?

Göreceli Momentum Endeksi'nde de RSI da olduğu gibi genellikle 30 ve 70 referans değerleri kullanılmaktadır. 30'dan küçük Göreceli Momentum Endeksi değerleri aşırı satım, 70'den büyük değerler ise aşırı alım bölgesi olarak görülür. Diğer aşırı alım satım göstergelerinde olduğu gibi bu bölgelerde dolaşan RMI değerleri bize trendin yakın bir gelecekte sonlanabileceği sinyallerini vermekte olsa da bu indikatör RSI'a göre yumuşak yapısından dolayı RSI kadar kesin bir trend dönüşü anlamına gelmeyecektir. Sinyal için referans değerlerinin kırılımını beklemek daha uygundur. Göreceli Momentum Endeksi'ni yorumlarken kullanılan bir diğer yöntemse uyumsuzluklardır. Fiyat hareketleri ile uyumlu hareket eden bir Göreceli Momentum Endeksi beklenmektedir. Fiyatlardaki hareketi aynı yönde takip etmeyen RMI uyumsuzluğa ve bir trend değişimine işaret etmektedir.

### Kodlama Editörü Örneği

```
public string SonYon="";
public string Sembol="VAKBN";
public string Periyot="1";
int lot=1;
        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=20;
        int Parametre2=5;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);
// Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.RelativeMomentumIndex(closed,Parametre1,Parametre2);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Göreceli Volatilite Endeksi (RVI)

## Göreceli Volatilite Endeksi (RVI) Nedir?

Donald Dorsey'in Göreceli Volatilite Endeksi (RVI), oynaklığın yönünü ölçen doğrulayıcı bir göstergedir.

## Göreceli Volatilite Endeksi (RVI) Nasıl Kullanılır?

30 ila 20'nin altındaki RVI değerleri aşırı satış olarak kabul edilir ve “alım” sinyali üretmiş olur. 70 ila 80'in üzerindeki RVI değerleri aşırı satın alınmış olarak kabul edilir ve bu nedenle “satım” sinyali üretmiş olur.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        int Parametre2=10;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.RelativeVolatilityIndex(candles,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Hacim Osilatörü Puanı

Hacim Osilatörü Puanı**,** hacimdeki değişikliği iki hareketli ortalamayı puansal değerler üzerinden ölçer.

Biri düşük periyotlu biri de daha yüksek periyotlu hareketli hacim ortalamaları kullanır. Düşük periyotlu olan hareketli ortalama hacimdeki değişikliklere daha duyarlıdır. Daha yüksek periyotlu olan hareketli ortalama ise hacim değişikliklerine daha az karşılık verir. Bu iki hareketli ortalama daha sonra birbirinden çıkarılır. Hacimdeki değişim hızı zamanla alçalacak ve akacaktır. Diğer bir deyişle, hacim, uzun süre boyunca katlanarak artamaz veya azalamaz. Buna göre, gösterge zamanla döngüsel veya salınımlı olacaktır.

PVO ile ilgili olarak, pozitif bir değer, hacmin değişim oranındaki son eğilimin pozitif olduğunu göstermektedir. Negatif bir değer, hacimdeki değişim oranının düştüğü anlamına gelir. Bu, düşen hacmin, fiyatta bir durgunluğa veya tersine dönmeye neden olma etkisine sahip olabileceğini gösterebilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=12;
        int Parametre2=26;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
                //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir
listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator =
Engine.VolumeOscillatorPoints(candles,MovingAverageMethods.Exponential,Parametre1,Parametre2);

                //Eğer indikatörün önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
                //Eğer indikatörün önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }

        }
```

## Hacim Osilatörü Puanı Histogram

Hacim Osilatörü Puanı Histogram, hareketli ortalama eklenerek bir süzgeçten geçirilip tekrar düzenlenmesidir. Yorumlanması aynıdır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int AveragePeriyot=9;
        int Parametre1=12;
        int Parametre2=26;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator =
Engine.VolumeOscillatorPointsHistogram(candles,MovingAverageMethods.Exponential,AveragePeriyot,Parametre1,Para
metre2);

                //Eğer indikatörün önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

                //Eğer indikatörün önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Hacim Osilatörü Yüzdesi (PVO)

Hacim Osilatörü Yüzdesi (PVO), hacimdeki değişikliği iki hareketli ortalamayı yüzdesel değerler üzerinden ölçer.Biri düşük periyotlu biri de daha yüksek periyotlu hareketli hacim ortalamaları kullanır. Düşük periyotlu olan hareketli ortalama hacimdeki değişikliklere daha duyarlıdır. Daha yüksek periyotlu olan hareketli ortalama ise hacim değişikliklerine daha az karşılık verir. Bu iki hareketli ortalama daha sonra birbirinden çıkarılır.

Hacimdeki değişim hızı zamanla alçalacak ve akacaktır. Diğer bir deyişle, hacim, uzun süre boyunca katlanarak artamaz veya azalamaz. Buna göre, gösterge zamanla döngüsel veya salınımlı olacaktır.

Pozitif bir PVO değeri hacmin değişim oranındaki son eğilimin pozitif olduğunu göstermektedir. Negatif bir değer, hacimdeki değişim oranının düştüğü anlamına gelir. Bu, düşen hacmin, fiyatta bir durgunluğa veya tersine dönmeye neden olma etkisine sahip olabileceğini gösterebilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=12;
        int Parametre2=26;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator =
Engine.VolumeOscillatorPercent(candles,MovingAverageMethods.Exponential,Parametre1,Parametre2);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Hacim Osilatörü Yüzdesi (PVO) Histogram

Hacim Osilatörü Yüzdesi'ne (PVO) hareketli ortalama eklenerek bir süzgeçten geçirilip tekrar düzenlenmesidir. Yorumlanması aynıdır.

## Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int AveragePeriyot=9;
        int Parametre1=12;
        int Parametre2=26;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator =
Engine.VolumeOscillatorPercentHistogram(candles,MovingAverageMethods.Exponential,AveragePeriyot,Parametre1,Par
ametre2);

                //Eğer indikatörün önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

                //Eğer indikatörün önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Hareketli Ortalama (MA)

## Hareketli Ortalama (MA) Nedir?

Hareketli Ortalama (MA) indikatörü Dünya'nın en çok kullanılan bir trend takip indikatörüdür.MA indikatörünün birçok hesaplama türü bulunur. "Basit","Üssel","Ağırlıklı" ortalama bunlardan bazılarıdır. Basit hesaplamalar yapmasına rağmen son derece kullanışlı bir indikatördür.

1. Basit Hareketli Ortalama: Bir finansal ürünün, belirlenen dönem içerisindeki fiyat hareketlerinin ortalaması alınarak oluşturulan hareketli ortalamadır. Basit hareketli ortalama kapanış fiyatlarını dikkate almaktadır. Örneğin; 5 günlük basit hareketli ortalama 5 günlük kapanış fiyatlarının toplanarak 5'e bölünmesi ile elde edilmektedir.
2. Ağırlıklı Hareketli Ortalama: Bir finansal ürünün, belirlenen dönem içerisindeki fiyat hareketlerinin belirlenen ağırlıklara göre ortalaması alınarak hesaplanan hareketli ortalamadır.
3. Üssel Hareketli Ortalama: Bir finansal ürünün, belirlenen dönem içerisindeki fiyat hareketlerinin ortalaması alınarak, yakın dönemdeki fiyat hareketlerine daha fazla ağırlık verilerek hesaplanan hareketli ortalamadır. Ağırlıklandırma yapıldığından dolayı üssel hareketli ortalama daha az gecikmeli bir hareketli ortalama olarak sayılmaktadır.

## Hareketli Ortalama (MA) Nasıl Kullanılır?

MA indikatörü kullanılan periyota göre çok değişkenlik göstermesine rağmen çoğu zaman fiyat ile benzer hareket etmektedir. Kullanımında birçok varyasyon denenebilmektedir. Örneğin 20 periyotluk bir MA ile 5 periyotluk bir MA'nın kesişiminden, 10 periyotluk bir MA'nın fiyat ile kesişiminden veya iki adet MA'nın birbirlerden olan uzaklıkları belli bir değeri geçtiğinde sinyal üretilebilir.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.
public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}
public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.MovingAverage(C, MovingAverageMethods.Simple, 14); // 1 çizgisi mevcuttur.
   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Hızlı Stokastik

Stokastik göstergenin her fiyat hareketine hemen tepki veren kararsız yapısı nedeniyle analistler yine bundan üretilen Hızlı Stokastik'i kullanmayı tercih ederler.

2 ayrı çizgi oluşturmaktadır. Bu indikatör için referans çizgileri 20 ve 80'dir. Hızlı Stokastikte %K eğrisinin %D eğrisini yukarı doğru kesmesi "al" sinyali olarak kabul edilirken, %K eğrisinin %D eğsini yukarıdan aşağıya kesişi ise "sat" sinyalini oluşturulmaktadır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametresini tanımlıyoruz.
        int Parametre1=5;
        int Parametre2=3;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.StochasticFast(candles,Parametre1,Parametre2);
                var indicator1=indicator[0];
                var indicator2=indicator[1];

        //Eğer indikatörün birinci çizgisi ikinci çizgisini aşağıdan yukarı keserse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.Intersect(indicator1,indicator2, "up") && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün birinci çizgisi ikinci çizgisini yukarıdan aşağı keserse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.Intersect(indicator1,indicator2, "down") && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## HLC3

HLC, yüksek(high), düşük(low) ve kapanış(close) anlamına gelir ve ortalamayı bölmek için 3'e böler.

Bazı yatırımcılar, bu ortalamanın bir grafikteki fiyatının gerçek değerini daha fazla temsil ettiğini düşünürler. Bu, genellikle basit, ağırlıklı, üssel gibi hareketli ortalamaları hesaplarken kapanış fiyatı yerine kullanılır.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.HLC3(candles); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# Ichimoku İndikatörü

İlk olarak 1930'lu yıllarda Japon analistler tarafından geliştirilmiş ve 1960'lı yıllarda yaygınlaşmıştır. Bu indikatörün amacı diğer indikatörleri kullanmaya gerek kalmadan, piyasada trendin yönünü tespit etmek ve işlem fikri üretmek amacı ile kullanılır.

İndikatör 5 çizgiden oluşmaktadır. Çizgilerden dördü en yüksek ve en düşük rakamların ortalamasının alınmasıyla hesaplanır. 26 ve 9 hareketli ortalamalar ile kullanılması genel kabul görmüştür.

Tenkan Sen ve Kjun Sen seviyeleri destek ve direnç noktalarını gösterir. Aynı zamanda bu seviyelerin kesişmesi işlem fikri olarak kullanılmaktadır.

Bulut analizinin temelini Tenkansen ve Kijun Sen' in uzaklaşması yakınlaşması ve kesişimi oluşturur.

1 - Tenkan Sen: 9 periyotluk hareketli ortalama
2 – Kijun Sen: 26 periyotluk hareketli ortalama
3 – Senkou Span A: Tenkan Sen ve Kijun Sen toplanarak 2' ye bölünür ve 26 periyot ileri kaydırılır.
4 – Senkou Span B: 52 periyot içindeki en yüksek ve en düşük değerler toplanarak ikiye bölünür ve çıkar değer 26 periyot ileri kaydırılır.
5 – Chinkou Span: Kapanış fiyatının 26 periyot geriye kaydırılması ile elde edilir. Eğer bulunan değer 26 periyot önceki kapanışının üzerindeyse boğa, altında ise ayı piyasası olarak yorumlanır.

Destek ve direnç seviyelerini tespit ederken 9 günlük ortalama fiyata daha yakın olduğu için ilk destek seviyesi olarak kullanılır. 26 günlük ortalama ise 2. Destek olarak kabul edilir.

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.
public string Symbol="ASELS";
public string Period="1";
public string SonYon="";
public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var indikator=Engine.Ichimoku(candles); // 5 çizgisi mevcuttur.
   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Gün İçi Momentum İndikatörü (IMI)

Tushar Chande tarafından geliştirilmiş olan indikatör, son fiyat hareketliliğini ve momentumu baz alır. Momentumun farklı bir yöntemle hesaplanması ile oluşur. 30 ve 70 referans seviyelerini kullanır. 30'un altı aşırı satım bölgesidir. 70'in üstü ise aşırı alım bölgesidir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.IntradayMomentumIndex(candles,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Kairi

## Kairi İndikatörü Nedir?

Kairi, momentum temelli bir osilatör olarak kullanılmaktadır. RSI ile benzerliği ve bazı piyasalarda RSI yerine kullanılması ile popülerlik kazanmıştır. Momentum göstergeleri, piyasa fiyatlarındaki değişim oranını ölçer ve fiyatlar yükseldiğinde artarken, fiyatlar düştüğünde ise azalacaktır.

Kairi = (Kapanış Fiyatı – (X günlük ortalama Fiyat))/ (X günlük ortalama fiyat)*100

Sıfır çizgisi etrafında değer alan Kairi için, fiyatın ilgili ortalamadan daha yüksek değer aldığı durumlarda pozitif değer, tersinde ise negatif değer oluşacaktır.

## Kairi İndikatörü Nasıl Kullanılır?

Kullanımda, göstergenin dip ya da tepe seviyelerine geldikten sonra RSI ve kendi ortalaması ile karşılaştırılarak ya da trend çizgileri çizerek, dönüş hareketlerini tahmin etmede yardımcı olması beklenir. Aşırı alım durumunda bir Kairi değerinin, Kısa vadeli ortalamasını ve trend çizgisini aşağı geçmesi halinde ilgili varlığın fiyatlarının yeniden ortalamaya dönüş sinyali ve dolayısıyla düşmesi beklenebilir. Tek başına değil, ancak momentum temelli olarak kullanımı, sistemlerde iyi bir yardımcı araç vasfı vermektedir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametrelerinin tanımlanması
        int Parametre=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
        //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.Kairi(closed,Parametre);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Keltner Kanalı (KC)

Keltner Kanalı (KC) indikatörü, Bollinger Bantları ve Hareketli Ortalama zarflarına benzer bantlı bir göstergedir. Bunlar orta çizginin üzerinde ve altındaki birer zarftan oluşur. Orta çizgi, kullanıcı tanımlı bir zaman aralığında hesaplanan fiyatın bir hareketli ortalamasıdır. Genellikle basit hareketli ortalama veya üstel hareketli ortalama kullanılır. Üst ve alt zarflar (kullanıcı tanımlı), orta çizgiden bir mesafede olarak ayarlanır. Bu günlük yüksek / düşük aralığının katları veya daha yaygın olarak Ortalama Gerçek Aralık'ın bir katı olabilir.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.KeltnerChannel(candles, 10); // 2 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Klinger Osilatörü

Klinger Osilatörü, Stephen Klinger tarafından uzun vadeli para akışı eğilimini belirlemek için geliştirilmiştir ve kısa vadeli dalgalanmaları tespit etmek için fazla hassas kalmaktadır. Gösterge, menkul kıymetlerden akan hacmi menkul kıymetin fiyat hareketleriyle karşılaştırır ve ardından sonucu bir osilatöre dönüştürür. Klinger Osilatörü, fiyattan daha fazlasına dayanan iki hareketli ortalama arasındaki farkı gösterir. Tüccarlar, potansiyel fiyat dönüşlerini işaret etmek için göstergede farklılaşmayı izler. Diğer osilatörler gibi, ek ticaret sinyalleri sağlamak için bir sinyal hattı eklenebilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre=13;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.KlingerOscillator(candles,Parametre);

//Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

//Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Lineer Regresyon

## Lineer Regresyon İndikatörü Nedir?

Lineer Regresyon indikatörü kullanılan dönem içerisindeki fiyat hareketlerinin oluşturduğu trende bağlı olarak kendini güncelleyen bir indikatördür.

Hesaplanışı matematikteki "En küçük kareler" metodununa dayanmaktadır. Önceki fiyatları dikkate alarak bir sonraki fiyatın hesaplanan değerini göstermektedir.

## Lineer Regresyon İndikatörü Nasıl Kullanılır?

İndikatörün değerinin, fiyatın altında veya üstünde seyretmesine bakılarak trend yönü belli bir ölçüde tahmin edilebilir. Kullanımı açısından hareketli ortalamaya benzemektedir. Uzmanlar düşük periyot kullanan bir hareketli ortalamayla Linear Regresyon indikatörünün değerlerinin kesişimine göre sinyal üretmektedir. Eğer Lineer Regresyon değeri hareketli ortalamadan yüksekse "Satım" sinyali veya trendin "Aşağı" devam edeceği, düşük olduğu durumda ise "Alım" sinyali veya trendin "Yukarı" yönlü devam edeceği yorumu yapılabilir.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.LinearRegression(C, 14, 0); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# Lineer Regresyon Eğimi

## Lineer Regresyon Eğimi Nedir?

Lineer (Doğrusal) Regresyon Eğimi trendin genel yönünü tahmin etmek amacıyla oluşturulmuş bir indikatördür.

Lineer Regresyon Eğimi -1 ve 1 arasında bir sonuç vermektedir. Lineer Regresyon indikatörü ile fiyat arasındaki farka bağlı olarak eğimi oluşturan bir indikatördür.

## Lineer Regresyon Eğimi Nasıl Kullanılır?

İndikatörün değerinin pozitif yönde(0'ın üzerinde) yükselmeye başlaması "Alım" sinyali olarak yorumlanabilir. Aynı şekilde negatif yönde(0'ın altında) düşmeye başlaması "Satım" sinyali olarak değerlendirilebilir.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.LinearRegressionSlope(C, 14); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# MACD

## MACD İndikatörü Nedir?

MACD indikatörü kısa ve orta vadeli iki hareketli ortalamanın birbirinden çıkarılması sonucu elde edilen trend bazlı bir indikatördür.

MACD indikatörünün açılımı "Moving Average Convergence Divergence"'tır. Kullanılan iki hareketli ortalamanın ilişkisine bakılarak sinyal üretir.

## MACD İndikatörü Nasıl Kullanılır?

MACD indikatöründe yaygın olarak genel kabul görmüş 26-12-9 parametreleri kullanılır. Bu parametrelerin anlamları 26 barlık hareketli ortalama ile 12 barlık hareketli ortalamanın birbirinden çıkarılması sonucu elde edilen verinin 9 barlık hareketli ortalamasının alınmasıdır. Eğer 26 ve 12 periyodunu kullanıp oluşmuş yeni hareketli ortalamanın sonucu 9 barlık hareketli ortalamadan büyükse "Alım" sinyali olarak, küçükse "Satım" sinyali olarak yorum yapılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=12;
        int Parametre2=26;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.MACD(closed,Parametre1,Parametre2);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## MACD Histogram

MACD çizgisi ile MACD indikatörünün sinyal çizgisinin arasındaki mesafenin histograma dönüştürülmüş halidir. Kullanım kolaylığı olması amacıyla geliştirilmiştir. MACD indikatörü ile temel yapısı aynıdır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=12;
        int Parametre2=26;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.MacdHistogram(closed,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Kitle Endeksi

Tushar Cande ve Donald Dorsey tarafından geliştirilen Kitle Endeksi, fiyatların gün içi en yüksek ve en düşük seviyelerinin arasındaki değişim miktarlarının üssel hareketli ortalamaları alınarak hesaplanır. Gün içindeki değişimin daralma ve genişlemesini takip etmektedir. Trend yönünden çok trend dönüşlerinin sinyalini veren bir gösterge olduğundan başka bir hareketli ortalama indikatörü ile kullanımı ile analiz edilebilir.

## Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre=25;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.MassIndex(candles,Parametre);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Momentum İndikatörü

Momentum indikatörü, basit olarak ilgili varlığın belirlenen dönem boyunca yüzde değişimini gösterir. Ve bu yüzde değişimleri hareketli olarak izlediği için, 100 seviyesinin etrafında dalgalanan bir çizgi verecektir. Momentum değerinin 10 periyotta 105 değerini alması fiyatların son 10 periyotta %5 yükseldiğini, 95 değerini alması ise yüzde 5 gerilediğini gösterecektir.

Momentum değerinin 100 referans değerinin üzerine çıkması, Alım sinyali ve 100 değerinin altına inmesi ise fiyatların x periyot önceki değerin altına düşmesi nedeniyle satım sinyali olarak değerlendirilebilir.

Kullanımda, kendi ortalaması ile kesişimi ve fiyat ile uyumsuzlukları alım-satım sinyallerinde daha iyi fikir verecektir. Ancak bu gösterge belirli bir periyotta sadece fiyat hareketini özetlediğinden tek başına sistemlerde kullanılması yerine ana indikatöre yardımcı olarak kullanılması daha uygundur.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre=12;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
        //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.Momentum(closed,Parametre);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Müthiş Osilatör (AO)

Awesome (Müthiş) Osilatör piyasa momentumunu ölçmek için kullanılan bir göstergedir. AO, 34 periyot ve 5 periyot basit hareketli ortalamaların farkını hesaplar. Kullanılan basit hareketli ortalamalar kapanış fiyatı kullanılarak değil, her bir çubuğun orta noktaları alınarak medyan fiyat üzerinden hesaplanır. AO, genellikle eğilimleri teyit etmek veya olası ters dönüşleri tahmin etmek için kullanılır.

**AO = SMA (Medyan Fiyat, 34) - SMA (Medyan Fiyat, 5)**

## Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametresinin tanımlaması
        int Parametre=5;
        int Parametre2=34;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.AwesomeOscillator(candles,Parametre,Parametre2);
//Eğer indikatörün bir önceki değeri, iki önceki değerinden küçük eşitse ve SonYon BUY 'a eşit değilse Alış Yap.
                if(Engine.PreviousValue(indicator,1)<=Engine.PreviousValue(indicator,2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY, lot,PriceTypes.Market);
                }

//Eğer indikatörün bir önceki değeri, iki önceki değerinden büyük eşitse ve SonYon SELL 'e eşit değilse Satış Yap.
                else if(Engine.PreviousValue(indicator,1)>=Engine.PreviousValue(indicator,2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Negatif Hacim Endeksi (NVI)

Negatif Hacim Endeksi (NVI), Paul Dysart tarafından geliştirilmiş ve tüccarlar tarafından akıllı parayı takip ederek boğa ve ayı piyasalarını tanımlamak için kullanılmaktadır. Hangi fiyat hareketlerinin bu paradan kaynaklandığını tespit için hacim verileri kullanılır. Göstergenin arkasındaki varsayım, akıllı para, fiyatı oynatmak için daha az hacim gerektirir. Fiyat düşük hacimde yükseldiğinde ve gösterge yükselir ve fiyat düşük hacimde düştüğünde gösterge de düşer. Hacim yüksek olduğunda, kalabalığın etkin olması nedeniyle değişmeden kalır. Gösterge, kalabalık aktifken fiyat hareketlerini belirlemeyi amaçlayan Pozitif Hacim Endeksinin (PVI) tam tersidir. En iyi diğer analiz teknikleriyle birlikte kullanılır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.NegativeVolumeIndex(candles);

//Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# OHLC4

Belirli sembol ve periyot ile toplama dönemi ve fiyat türü için:
(Açılış + Yüksek + Düşük + Kapanış) / 4 değerini gösterir.

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.OHLC4(candles); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# Ortalama Gerçek Aralık (ATR)

## Ortalama Gerçek Aralık (ATR) Nedir?

J. Welles Wilder tarafından geliştirilen ATR indikatörü fiyatların hareketliliğini ölçmek amacıyla oluşturulmuştur. Belirlenen zaman aralığındaki her barın en yüksek ve en düşük değerlerinin farklarının ortalaması alınarak hesaplanır. Genel kabul olarak periyod 14 olarak ele alınır.

İndikatörün adı Average True Range kelimelerinin baş harflerinden gelmektedir. ATR fiyatların gideceği yön ile ilgili bir çıkarım yapmaz sadece fiyatların ne kadar hızlı ve yavaş değiştiği ile ilgili bilgi verir. Eğer ATR'nin değeri yükselmeye başlarsa fiyat hareketliliğinde artmış olduğu anlaşılır. Eğer değerleri düşmeye başlarsa fiyatların sıkışma alanında olduğuna yönelik bir yorum yapılabilir.

## Ortalama Gerçek Aralık (ATR)  Nasıl Kullanılır?

Kullanımında yaygın olarak, indikatör değerinin artmaya başladığı dönemlerde, kullanılan trend bazlı indikatörlerin sinyalleri dikkate alınabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

//İndikatörün parametresini tanımlanması
        int Parametre=14;

//Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
//SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
            SubscribePrice(Sembol);
        }
//Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
            var candles = GetCandles(Sembol,Periyot);

// Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
            var indicator = Engine.ATR(candles,Parametre);

//Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
            if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
              SonYon="BUY";
              SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
            }
//Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
            else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
              SonYon="SELL";
              SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
            }
        }
```

## Ortalama Günlük Aralık (ADR)

Ortalama Günlük Aralık, menkul kıymetin günlük aralığının ortalamasıdır. Adından da anlaşılacağı gibi, ortalama günlük aralık göstergesi, bir süre boyunca fiyatların ortalama günlük aralığını gösterir. Bu gösterge ADX ile de benzerlik göstermektedir. ADR'de kullanılan parametre 14'tür.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

//İndikatörün parametresinin tanımlaması
        int Parametre=14;

//Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
//SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
            SubscribePrice(Sembol);
        }

//Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
            var candles = GetCandles(Sembol,Periyot);

// Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
            var indicator = Engine.AverageDirectionalRating(candles,Parametre);

//Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
            if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
              SonYon="BUY";
              SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
            }

//Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
            else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
              SonYon="SELL";
              SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
            }
        }
```

# Ortalama Yön Hareketi Endeksi (ADX)

## Ortalama Yön Hareketi Endeksi (ADX) Nedir?

ADX indikatörü oluşan ya da oluşmakta olan trendin gücünü ölçmeye yarayan bir indikatördür. Birbiri ardına gelen barların en düşük ve en yüksek değerlerinin karşılaştırılmasıyla elde edilen +DM ve –DM göstergelerinin değişimlerinin belirli bir zaman diliminde incelenmesiyle elde edilir (Genel kabul periyodun 14 olarak kullanılması yönündedir.)

ADX indikatörünün açılımı "Average Directional Movement Index"'tir. İndikatörün değerleri 0 ile 100 arasında salınım yapar. Değerlerin 0 seviyesine yaklaşması piyasada belirgin bir trend olmadığına, 100 seviyesine yaklaşması ise trendin belli bir yöne daha kararlı gittiğine yönelik yorumlanabilir.

## Ortalama Yön Hareketi Endeksi (ADX) Nasıl Kullanılır?

ADX indikatörü genel olarak ana stratejiye bir yardımcı olarak çalışır. Ana stratejiden oluşan sinyalleri belli bir ölçüde filtrelemek için kullanılabilir. Eğer oluşan sinyali ADX indikatörü destekliyorsa işleme girilebilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametresini tanımlıyoruz.
        int Parametre=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
        //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
 //GetCandles("Sembol","Periyot") metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

// Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ADX(candles,Parametre);

//Eğer indikatörün bir önceki değeri 20' den büyükse  ve SonYon BUY 'a eşit değilse Alış Yap.
                if(Engine.PreviousValue(indicator,1)>20 && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

                //Eğer indikatörün bir önceki değeri 20 den küçükse ve SonYon SELL 'e eşit değilse Satış Yap.
                else if(Engine.PreviousValue(indicator,1)<20 && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Para Akışı Endeksi (MFI)

## Para Akışı Endeksi (MFI) Nedir?

MFI, bir hisse senedine girmekte, ya da hisse senedinden çıkmakta olan paranın gücünü ölçen bir göstergedir. Hacim verisine ihtiyaç duyar.

MFI adının açılımı Money Flow Index'tir. Pozitif para akışları toplamı ile negatif para akışlarının toplamı ile aşağıdaki şekilde hesaplanır.
MFI = 100- (100/(1 + Toplam Para Oranı))

## Para Akışı Endeksi (MFI) Nasıl Kullanılır?

Genel olarak İndikatörün değerini analiz ederken aşırı alım-satım bölgelerine bakılır. Varsayılan olarak 20(aşırı satım) ve 80(aşırı alım) değerleri kullanılmaktadır. Eğer indikatörün değeri 20 değerini aşağıdan yukarı keserse "Alım", 80 değerini yukarıdan aşağıya keserse ise "Satım" sinyali oluşumu olarak yorumlanabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametrelerinin tanımlanması
        int Parametre=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.MoneyFlowIndex(candles,Parametre);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Parabolic SAR

## Parabolic SAR Nedir?

Parabolic SAR indikatörü J.Welles Wilder tarafından geliştirilmiştir. Pozisyon değişim noktalarını belirlemek için oluşturulmuştur. Ne zaman alım-satım yapılacağı ile ilgili bilgi verir.

İndikatörün adı Stop And Reverse kelimelerinin baş harflerinden gelmektedir. Çift yönlü piyasalarda genellikle trendi oluşmuş dönemlerde kullanılmak üzere geliştirilmiş Parabolic SAR indikatörü, fiyat grafiğinin üzerinde kesikli noktalardan oluşmaktadır.

## Parabolic SAR Nasıl Kullanılır?

Kullanımında yaygın olarak, indikatör değerinin, fiyatın altında seyretmesinde "Alım", üstünde seyretmesinde ise "Satım" sinyali olarak yorumlanır.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.SAR(candles, 0.02, 0.2); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## PHPL 01

Grafikler üzerine saatlik, günlük, haftalık, aylık gibi dönemlere en yüksek / en düşük fiyat seviyelerini atmak için kullanılır. PH01 yüksek çizgisini, PL01 düşük çizgisini getirir.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.PHPL01(candles); // 3 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Pivot Bantları

Pivot Bantları, hareketli bir ortalama ile benzerlik göstermektedir ancak aynı zamanda bir Bollinger Bantları gibi piyasa oynaklığına göre ayarlama yeteneğine de sahiptir. Bandının genişliği, piyasa daha değişken hale geldikçe kademeli olarak artar.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.PivotBand(candles); // 3 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Polarize Fraktal Verimlilik (PFE) İndikatörü

Polarize Fraktal Verimlilik (PFE), Hans Hannula tarafından kullanıcı tanımlı geliştirilen bir süre boyunca fiyat verimliliğini belirlemek için kullanılan teknik bir indikatördür. Bu indikatör, merkez çizgisi 0 olmak üzere -100 ile +100 arasında dalgalanır. Sıfırdan büyük PFE'ye sahip menkul kıymetlerin yükseliş eğilimi gösterdiği kabul edilirken, sıfırdan düşük bir okuma eğilimin düştüğünü gösterir. Polarize Fraktal Verimlilik'in en önemli özelliği, bir güvenlik fiyatının ne kadar verimli hareket ettiğini belirlemede fraktal geometri kullanmasıdır.

Polarize Fraktal Verimlilik (PFE) indikatörü, trendin gücünü sıfır çizgisine göre konumu ile ölçer. Genel bir kural olarak, PFE değeri sıfırdan ne kadar uzaksa, verilen eğilim o kadar güçlü ve verimli olur. Sıfır çizgisi etrafında dalgalanan bir PFE değeri, menkul kıymet arz ve talebinin dengede olduğunu ve fiyatın yana doğru işlem görebileceğini gösterebilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        int Parametre2=3;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.PolarizedFractalEfficiency(closed,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Pozitif Hacim Endeksi (PVI)

Pozitif Hacim Endeksi (PVI), işlem hacminin bir önceki güne göre arttığı günlerde gerçekleşenlerle ilgilenmektedir. Fiyatların ve hacmin yükseldiği günlerde piyasa hakkında yeterli bilgi, deneyim ve mali güce sahip olan yatırımcıların bu fiyatları değerlendirdiği ve buralardan satıma geçtikleri varsayımından hareket etmektedir. Bilgisiz ve deneyimsiz küçük yatırımcı açısındansa durum fiyatların ve hacmin düştüğü seviyelerden satıma geçmek olarak özetlenebilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.PositiveVolumeIndex(candles);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Projeksiyon Bant Genişliği

Projeksiyon Bant Genişliği, Projeksiyon Bantları'na dayanır ve bantlardan orta noktaya olan oranı gösterir. Düşük sayılarda bantlar daralırken, yüksek sayılar bantların genişlediğini gösterir. Bu nedenle, bant genişliği oynaklığın bir ölçüsü olarak kullanılabilir. Dar bantlar, dar menzil ve düşük volatiliteye eşittir, geniş bantlar ise geniş bir aralığı ve yüksek volatiliteyi gösterir. Gösterge ayrıca, bant genişliğinin ne zaman aşırı alınıp satıldığını gösteren gölgeli renkli grafiklerle birlikte gelir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ProjectionBandwidth(candles,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Projeksiyon Bantları

Projeksiyon Bantları, minimum ve maksimum öngörülen sınırları gösteren iki banttan oluşur. Bantlar, belirli bir süre boyunca minimum ve maksimum fiyatlar kullanılarak türetilir ve bunları doğrusal bir regresyon çizgisine paralel olarak ileriye doğru yansıtır. Bantlar, fiyatlar sınırlarına ulaştığında yönlü fiyat tersine dönüş sinyali olarak yorumlanır.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.ProjectionBands(candles, 14); // 2 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Projeksiyon Osilatörü

Projeksiyon Osilatörü çalışması, mevcut fiyat ile zaman içindeki minimum ve maksimum fiyatları arasındaki ilişkiyi gösterir. Stochastik Osilatörden farklı olarak, burada minimum ve maksimum fiyatlar, fiyatın regresyon çizgisinin eğimine göre yukarı veya aşağı ayarlanır.

Projeksiyon Osilatörü, geçerli fiyatın Projeksiyon Bantları arasında nerede olduğunu gösterir. 0 değeri, fiyatların alt banda dokunduğunu, 50 değeri mevcut fiyatın tam olarak iki bandın ortasında olduğunu, 100 değeri ise fiyatların üst banda dokunduğunu gösterir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        int Parametre2=1;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ProjectionOscillator(candles,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## QStick İndikatörü

Qstick indikatörü, mum çubuğu grafiklerinde eğilimleri sayısal olarak belirlemek için Tushar Chande tarafından geliştirilen bir teknik analiz göstergesidir. Açılış ve kapanış fiyatları arasındaki farkın 'n' periyotlu hareketli ortalaması alınarak hesaplanır. Sıfırdan büyük bir Qstick değeri, son 'n' günün çoğunun arttığı anlamına gelir ve bu, satın alma baskısının arttığını gösterir. Özetle, ölçü bir menkul kıymetin üssel hareketli ortalaması (veya EMA), açılış fiyatı, kapanış fiyatı ve bunların farkının yanı sıra bu değerler basit hareketli ortalamalar (veya SMA) için bir yaklaşıklık sağlar.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=8;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.QStick(candles,MovingAverageMethods.Simple,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Quantitative Qualitative Estimation (QQE) İndikatörü

RSI göstergesi üzerinde, çeşitli ortalamalar uygulayarak elde edilen bir göstergedir. Hızlı ve Yavaş olmak üzere iki çizgi getirir.

Gösterge değeri, 50 seviyesinin altında olduğunda “Alım” sinyali olarak yorumlanırken 50 seviyesinin yukarısında olduğunda “Satım” sinyali olarak yorumlanır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        int Parametre2=5;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.QQE(closed,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Range İndikatörü

ATR indikatörü gibi fiyatların hareketliliğini ölçmek amacıyla oluşturulmuştur. Belirlenen zaman aralığındaki her barın en yüksek ve en düşük değerlerinin farklarının ortalaması alınarak hesaplanır. Genel kabul olarak periyodlar 10 ve 3 olarak ele alınır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=10;
        int Parametre2=3;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.RangeIndicator(candles,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }


        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap

                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){

                  SonYon="SELL";

                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);

                }

        }
```

# RAVI İndikatörü

RAVI (Range Action Verification Index) göstergesi, iki Ağırlıklı hareketli ortalama göstergesinin (VMA) birleşimidir. Biri 7 dönem hareketli ortalama ve diğeri 65 dönem hareketli ortalamadır. Biri kısa süreli VMA ve ikincisi uzun süreli VMA'dır.

İki VMA değeri arasındaki fark, bir RAVI gösterge histogram çubuğu olarak gösterilecektir. İki VMA değeri arasındaki daha fazla boşluk, daha yüksek histogram çubukları oluşturur.

## Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=7;
        int Parametre2=65;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.RAVI(closed,MovingAverageMethods.Simple,Parametre1,Parametre2);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Renko Barları

Renko Barları, zamandan bağımsızdırlar ve bütünüyle fiyatı yansıtırlar. Çubuk grafikler veya çizgi grafikler daha yaygın olarak kullanılsa da, bunların tümü ortaktır çünkü fiyat belirli bir zaman diliminde alınır. Belirli bir süre boyunca açılış fiyatı, yüksek ve düşük ve kapanış fiyatını gösterir.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var indikator=Engine.RenkoBars(candles,1); // Barları Getirir mevcuttur.
   var C=Engine.GetPriceList(indikator, PriceFields.Close);
   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(C) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(C).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(C) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(C).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# RSI Denvelope

Aşırı alım ve satım bölgelerini bulmak adına yardımcı bir göstergedir.

## Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        int Parametre2=2;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.RSIDenvelope(candles,Parametre1,Parametre2);
                var indicator1=indicator[0]; //Birinci Çizgisi
                var indicator2=indicator[1]; //İkinci Çizgisi
                var indicator3=indicator[2]; //Üçüncü Çizgisi

//Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator1, 1)>Engine.PreviousValue(indicator2, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator2,1)<Engine.PreviousValue(indicator3, 2) &&
SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Rsquared İndikatörü

Bu göstergenin amacı düşük bir işlem hacmi ile kolayca hareketlenen senetleri bulmakla birlikte asıl amacı fiyatların işlem hacmine olan duyarlılığını ölçmektir.

Değer ne kadar 1' e yakınsa o derece endekse hareket ediyor anlamına gelmektedir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.Rsquared(closed,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Standart Hata Bantları

Geçmişe dönük x periyodun her biri için kendisinden önceki x periyodun ortalamasından farkının karesi ayrı ayrı hesaplanıp bu değerlerin ortalamasının karekökü alınarak hesaplanır.

Uygulandığı enstrümanın hareketliliğini ölçer.

Sıfıra yaklaştıkça hareketlilik azalıp, 1' e yaklaştıkça hareketlilik yükselir.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.StandardErrorBands(C, 14, 2); // 2 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Standart Hata İndikatörü

"Standart hata" terimi , ortalama veya medyan gibi çeşitli numune istatistiklerinin standart sapmasına atıfta bulunmak için kullanılır . Örneğin, "ortalamanın standart hatası", bir popülasyondan alınan numune araçlarının dağılımının standart sapmasına karşılık gelir. Standart hata ne kadar küçükse, örnek toplam popülasyonu o kadar temsil eder.

Standart hata ile standart sapma arasındaki ilişki, belirli bir numune boyutu için, standart hata, numune büyüklüğünün karekökünе bölünen standart sapmaya eşit olacak şekildedir. Standart hata da örneklem büyüklüğüyle ters orantılıdır; örneklem boyutu ne kadar büyükse, standart hata o kadar küçük olur çünkü istatistik gerçek değere yaklaşacaktır.

Standart hata, tanımlayıcı istatistiklerin bir parçası olarak kabul edilir. Bir veri kümesindeki ortalamanın standart sapmasını temsil eder. Bu, rastgele değişkenler için bir varyasyon ölçüsü olarak hizmet eder, dağılım için bir ölçüm sağlar. Yayılma ne kadar küçükse veri kümesi o kadar doğru olur.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
        //Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.StandardError(closed,Parametre1);

                //Eğer indikatörün önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

                //Eğer indikatörün önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Standart Sapma

Fiyat ortalamasının standart sapmasını belirli matematiksel formüllerle gösteren bir indikatördür. Yani fiyatın gelecekte ne kadar değişken olabileceğini tahmin etmek için bir varlığın son fiyat hareketlerinin büyüklüğünü ölçen bir göstergedir denilebilir.

Volatilitenin artacağına ya da azalacağına karar vermenize yardımcı olabilir. Volatilite, fiyat seviyelerindeki değişimin ne kadar yüksek aralıklarla gerçekleştiğini, finansal yatırım aracının fiyat hareketleri arasındaki farkın büyüklüğünü ölçer. Çok yüksek bir standart sapma okuması, büyük bir fiyat değişikliğinin meydana geldiğini, ancak volatilitedeki bir azalmanın yakında gerçekleşebileceğini gösterir. Çok düşük bir standart sapma ise karşıtlığı gösterir. Standart Deviation indikatörü genellikle tek başına kullanılmamaktadır, yani aslında tek başına kullanılmaktan çok başka bir indikatörün bir bileşeni olarak kullanılır. Örneğin, Bollinger Bantları, bir hareketli ortalamaya hisse senedinin standart sapması eklenerek hesaplanır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=5;
        int Parametre2=1;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
        //Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.StandardDeviation(closed,Parametre1,Parametre2);
                //Eğer indikatörün önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
                //Eğer indikatörün önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Stokastik Momentum Endeksi

William Blau tarafından geliştirilen Stokastik Momentum kapanış fiyatının, fiyatın belirlenen zaman dilimi içindeki günlük düşük-yüksek aralığının orta noktasına olan uzaklığını kullanır. Bu osilatör -100 ve 100 arasında değişirek eşit periyodlu Stokastik Osilatör'den biraz daha az değişken bir yapı sergiler. Bir çift çizgi ile gösterilen bu göstergede, %K eğrisinin %D eğrisini yukarı kırması "Alım" sinyali iken %K eğrisinin %D eğrisini aşağı kırması "Satım" sinyali olarak kabul edilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametresini tanımlıyoruz.
        int Parametre1=5;
        int Parametre2=3;
        int Parametre3=3;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.StochasticMomentumIndex(candles,Parametre1,Parametre2,Parametre3);
                var indicator1=indicator[0];
                var indicator2=indicator[1];

        //Eğer indikatörün birinci çizgisi ikinci çizgisini aşağıdan yukarı keserse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.Intersect(indicator1,indicator2, "up") && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün birinci çizgisi ikinci çizgisini yukarıdan aşağı keserse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.Intersect(indicator1,indicator2, "down") && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Stokastik Osilatörü

## Stokastik Osilatörü Nedir?

Stokastik Osilatörü kapanış fiyatını verilen periyot içindeki fiyatlar ile karşılaştıran bir göstergedir.

Stokastik Osilatörü genel olarak aşırı alım-satım göstergesi olarak kullanılır. Varsayılan seviyeleri 20 ve 80 olarak belirlenmiştir.

## Stokastik Osilatörü Nasıl Kullanılır?

Yaygın kullanımda ise Stokastik Osilatörü üzerine 5 periyotluk üssel hareketli ortalama eklenir. Eğer Stochastic Osilatörü değeri 20 seviyesinin altında iken hareketli ortalamanın değeri Stokastik Osilatörü değerinden büyükse "Alım", eğer STOCH göstergesinin değeri 80 seviyesinin üstünde iken hareketli ortalamanın değeri STOCH'un değerinden küçükse "Satım" sinyali olarak yorumlanabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";
        //İndikatörün parametresini tanımlıyoruz.
        int Parametre1=5;
        int Parametre2=3;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.StochasticOscillator(candles,Parametre1,Parametre2);
                var indicator1=indicator[0];
                var indicator2=indicator[1];

                //Eğer indikatörün birinci çizgisi 20 noktasından küçükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator1, 1)<20 && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

                //Eğer indikatörün birinci çizgisi 20 noktasından büyükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator1,1)>20 && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Stokastik RSI

## Stokastik RSI İndikatörü Nedir?

STOCHRSI indikatörü, RSI'a benzer olarak aşırı alım-satım bölgelerini belirlemek amacıyla oluşturulmuş bir göstergedir.

STOCHRSI indikatörünün açılımı Stochastic Relative Strength Index'tir. RSI indikatörünün en yüksek ve en düşük seviyelerinin zaman içinde değişiminin hesaplanmasıyla oluşturulmuş bir indikatördür. RSI indikatörünün gürültüsünü belli bir oranda azaltmak amacıyla oluşturulmuştur.

## Stokastik RSI İndikatörü Nasıl Kullanılır?

İndikatörün değerinin yaygın olarak kullanılan 20 ve 80 aşırı alım-satım noktalarını kesmesi sinyal olarak kabul edilir. Eğer indikatörün değeri 20 seviyesine aşağıdan yukarı keserse "Alım", 80 değerini ise yukarıdan aşağı kesmesi "Satım" sinyali olarak değerlendirilebilir. 20-80 seviyeleri yaygın olarak 30-70 olarak da kullanılabilmektedir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametresini tanımlıyoruz.
        int Parametre=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
        //Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatları listesinin closed değişkenine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.StochasticRSI(closed,Parametre);

                //Eğer indikatör 20 noktasını aşağıdan yukarı keserse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.Intersect(indicator,20, "up") && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
                //Eğer indikatör 0 noktasını yukarıdan aşağı keserse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.Intersect(indicator,0, "down") && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Salınım Endeksi

Son bardaki açılış değeri, kapanış değeri, en yüksek ve en düşük değerleri önceki barın açılış ve kapanış değerleriyle karşılaştırarak senedin salınımı hesaplar. Sıfırın üstü “ALIM” sinyali iken sıfırın altı “Satım” sinyalini işaret eder.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=3;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.SwingIndex(candles,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Talep Endeksi

Talep Endeksi, talep durumunu hesaplar. Sıfırın üstü değerler alım yönünü, sıfırın altı değerler satış yönünü işaret eder. Tepe ve diplerden dönüş noktaları, trendin ters yöne dönüyor olduğunun göstergesidir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
        //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.DemandIndex(candles);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# TEMA

Patrick Mulloy tarafından yazılan Üçlü Üstel Hareketli Ortalama (TEMA), geleneksel üstel hareketli ortalamaya göre daha az gecikmeli bir hareketli ortalama sunar.

'T' zaman serisinin Üçlü Üstel Hareketli Ortalaması (TEMA):

EMA1 = EMA (t, periyot)

EMA2 = EMA (EMA1, periyot)

EMA3 = EMA (EMA2, periyot)

TEMA = 3 * EMA1 - 3 * EMA2 + EMA3

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.Tema(C, 21, 0); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Time Series Forecast (TSF) İndikatörü

Time Series Forecast indikatörü Lineer Regresyon metodu kullanarak hesaplanır. Lineer regresyon bir istatistik aracı olarak geçmiş değerleri karşılaştırarak gelecek fiyat değerlerini tahmin içindir. Bu amaçla trendlerin yukarıya veya aşağıya doğru meyillerini tanımlar ve bu sonuçları geleceğe taşır. Örneğin, fiyatlar yukarı doğru hareket ederken, TSF fiyatın yukarı meylini o anki fiyatla karşılaştırarak bu hesaplamayı geleceğe taşır.

Time Series Forecast, fiyatlar indikatörün altına düştüğünde trendi aşağı yönlü, indikatörün üstüne çıktığında ise yukarı yönlü kabul etmektedir. TSF indikatörü eğer yönde ve eğimde bir değişiklik yoksa devam eden trendi tanımlar.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.TimeSeriesForecast(C, 14, 0); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# Tipik Fiyat İndikatörü

Tipik Fiyat İndikatörü, her günün fiyatının ortalamasını gösterir.
Günün ortalama fiyatının basit, tek çizgi ile grafiğe yansır. Bazı yatırımcılar hareketli ortalama penetrasyon sistemleri oluştururken kapanış fiyatı yerine Tipik Fiyat kullanır. Tipik Fiyat, Para Akışı Endeksi'nin yapı taşını oluşturan bir indikatördür.

Tipik Fiyat, yüksek, düşük ve kapanış fiyatlarının toplanması ve ardından üçe bölünmesiyle hesaplanır.

Tipik Fiyat = ( Yüksek Fiyat + Düşük Fiyat+ Kapanış Fiyatı) / 3

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu

   var indikator=Engine.TypicalPrice(candles); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

# TOMA (MOST)

## TOMA (MOST) İndikatörü Nedir?

TOMA (Trailing Oscillator of Moving Average), üstel hareketli ortalama ve izleyen stop-loss kullanılarak oluşturulan bir indikatördür. TOMA indikatörü klasik olarak stop loss yapmanızı sağlayan göstergelerin aksine fiyatların yükselmesi durumunda alım sinyali de üretmektedir. TOMA indikatöründe stoploss sadece o anlık fiyattan ziyade sinyallerini üç günlük üssel ortalamayı kullanarak üretmektedir. Bu sayede gün içini aşırı fiyat hareketlerinin üretebileceği yanlış sinyallerin de önüne geçmeyi amaçlamaktadır.

## TOMA (MOST) İndikatörü Nasıl Kullanılır?

TOMA indikatöründe, TOMA eğrisinin fiyatların üstüne çıkması stoploss olarak değerlendirir. Fiyatların, TOMA eğrisinin üstüne çıkması ve TOMA eğrisinin çanak gibi göründüğü durum ise alış yapılması gerektiği ya da varsa açığa satış işlemini kapatmanız gerektiği sinyalini üretmektedir.

TOMA, fiyatları temsil eden EMA yani hareketli ortalamanın üzerine çıkarsa satım yapılması gerektiğini, altına inerse de alım yapılması gerektiğini gösterir. Tek fark, MOST İndikatörü aynı yönde olan hareketli ortalamanın hareketinde aradaki stop loss mesafesini koruyarak ortalamayı izlemektedir. Hareketli ortalama, TOMA'yı kesip altına giderse MOST düz bir çizgi halini alıp yatay hareket etmektedir.
TOMA eğrisindeki iki değişkenden birincisi TOMA'nın ilişki içinde olduğu hareketli ortalama periyodudur. Fiyatlar için kullanmış olduğunuz yumuşatma oranı (yani hareketli ortalamanın süresi) ne ise TOMA'nın periyodu da o olacaktır.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";
public void Load()
{
   SubscribePrice(Symbol);
}
public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.Toma(C, 3, 2); // 2 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
```

# TOMA (MOST) Puan

TOMA Puan, TOMA indikatörü gibi hareketli ortalama ve izleyen stop-loss kullanılarak oluşturulan bir indikatördür. TOMA indikatöründen farklı olarak, stoploss üç günlük üssel hareketli ortalamaya göre değil; basit hareketli ortalamayı kullanarak üretmektedir. Bu sayede gün içini aşırı fiyat hareketlerinin üretebileceği yanlış sinyallerin de önüne geçmeyi amaçlamaktadır. Kullanım mantığı TOMA indikatörü ile aynıdır.

## Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.TomaPuan(C, 3, 2); // 2 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator[0]) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator[0]) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator[0]).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Trend Skoru

Son 20 günlük verileri karşılaştırarak trend yönünü ve gücünü ölçmeyi hedefler. -10 ve +10 arasında hareket eder. 10'a yaklaştıkça, yukarı doğru güçlü bir trendi, -10'a yaklaştıkça aşağı doğru güçlü bir trendi işaret eder.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=10;


        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

//Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.TrendScore(closed,Parametre1);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# Trendsiz Fiyat Osilatörü (DPO)

Trendsiz Fiyat Osilatörü göstergesi (DPO) fiyatlardan trendi çıkartmak için kullanılır. Bu, kısa vadeli döngüleri tanımlamak ve izole etmek için yapılır. DPO, tipik olarak en güncel fiyatlarla uyumlu değil. Mevcut trendin kaldırılmasına yardımcı olacak şekilde sola (geçmişe) kaydırılır. Geçmişe kaydırılmış olduğundan, DPO bir momentum osilatörü olarak kabul edilmez. Geçmişin fiyatlarını basit hareketli ortalamalara karşı, yalnızca bir döngünün yüksek / düşük aralığını ve tipik süresini göstermenin bir yolu olarak ölçer.

## Kodlama Editörü Örneği

```
        //Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre=20;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
    //GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

    //Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
                var closed=Engine.GetPriceList(candles,PriceFields.Close);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.DetrendedPriceOscillator(closed,Parametre);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# TRIX

## TRIX İndikatörü Nedir?

TRIX indikatörü, 3 derece hareketli ortalama kullanan bir hareketli ortalama versiyonudur.

TRIX indikatörü hesaplanırken kapanış fiyatının hareketli ortalamasını alır. Bu aşamadan sonra hesaplanan hareketli ortalamanın da hareketli ortalamasını alarak, klasik bir hareketli ortalamaya göre fiyat değişimlerinden daha düşük bir ölçüde etkilenen bir indikatördür.

## TRIX Nasıl Kullanılır?

TRIX indikatörünü kullanırken 0 referans seviyesine kesişim durumuna ve TRIX indikatörünün üzerine atılan başka bir hareketli ortalama ile olan kesişimlerine bakılabilir. İlk yöntemde TRIX indikatörünün 0 seviyesini yukarıdan aşağıya doğru kestiğinde "Satım" , aşağıdan yukarıya kestiğinde ise "Alım" sinyalinin oluşumu yorumu yapılabilir. İkinci yöntemde ise TRIX ve hareketli ortalamadan, daha hızlı olanın yavaş olanı aşağıdan yukarıya doğru kesmesi "Alım", aksi durumda ise "Satım" sinyali yorumu yapılabilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=12;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
            //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
            SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
            var candles = GetCandles(Sembol,Periyot);
        //Engine.GetPriceList(candles,PriceFields.Close) Kapanış fiyatlarının listesini closed değişkeninine tanımlanması
            var closed=Engine.GetPriceList(candles,PriceFields.Close);
            // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
            var indicator = Engine.TRIX(closed,Parametre1);
            //Eğer indikatörün önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
            if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
              SonYon="BUY";
              SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
            }
            //Eğer indikatörün önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
            else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
              SonYon="SELL";
              SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
            }
        }
```

# Ultimate Osilatörü

## Ultimate Osilatörü Nedir?

ULTIMATEOSC indikatörü aşırı alım-satım noktalarını belirlemeye yarayan bir indikatördür.İndikatörün değerleri 0 ile 100 arasında salınım yapar. RSI indikatörünün bir benzeridir.

## Ultimate Osilatörü Nasıl Kullanılır?

3 adet parametre alan ULTIMATEOSC indikatörünün genel olarak kullanılan parametreleri 7,14 ve 28'dir. Bu parametreler indikatörün kısa-orta-uzun vade anlayışına göre belirlenmiştir. ULTIMATEOSC indikatörü bu periyotları kullanarak hesaplamalarını yapar. Genellikle aşırı alım-satım noktalarına göre işlem yapılması uygun görülmüştür.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=7;
        int Parametre2=14;
        int Parametre3=28;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);
                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.ULTIMATEOSC(candles,Parametre1,Parametre2,Parametre3);
        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                   SonYon="BUY";
                   SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }
        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                   SonYon="SELL";
                   SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

# WILLIAMSR

## WILLIAMSR İndikatörü Nedir?

WILLIAMSR indikatörü fiyatların birbirleriyle aralarında olan ilişkiye bakarak aşırı alım-satım bölgeleri ve uyumsuzluk tahmin etmeye çalışan bir indikatördür.

L.Williams tarafından geliştirilmiş WILLIAMSR indikatörü belirlenen zaman periyodunun içinde yer alan fiyatları belirlenen periyottaki en yüksek ve en düşük değerlere olan uzaklıklarına göre değerlendiren bir indikatördür.

## WILLIAMSR Nasıl Kullanılır?

Genel kabul görmüş -80 ve -20 referans seviyeleri kullanılmaktadır. WILLIAMSR göstergesi sadece -100 ve 0 aralığında seyreder. İndikatörün değerinin -80 seviyesinin altına inmesi, aşırı-satım yapıldığının ve yakın bir zamanda "Alım" yapılabileceğine, -20 seviyesinde ise aşırı-alım yapıldığını ve yakın bir zamanda "Satım" yapılabileceğine yönelik bir çıkarım yapılabilir. Diğer aşırı alım-satım göstergeleri gibi fiyat ile indikatör değerleri arasındaki uyumsuzluk analiz edilebilir.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=14;
        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }
        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.WILLIAMSR(candles,Parametre1);

                //Eğer indikatör -50 noktasını aşağıdan yukarı keserse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.Intersect(indicator,-50, "up") && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

                //Eğer indikatör -50 noktasını yukarıdan aşağı keserse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.Intersect(indicator,-50, "down") && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Williams Birikim/Dağılım (Williams AD)

Williams Birikim/Dağılım (AD), pozitif "birikimli" fiyat hareketlerinin ve negatif "dağıtıcı" fiyat hareketlerinin birikmiş toplamıdır. "Birikim" terimi, alıcılar tarafından kontrol edilen piyasayı ifade etmek için kullanılır, "dağıtım" ise piyasanın satıcılar tarafından kontrol edildiğini belirtir.

Fiyatlar yeni bir minimum seviyeye ulaştığında ve Williams AD bu seviyeye ulaşamıyor ise, güvenlik birikiminin gerçekleştiği anlamına gelir. Bu durum “Alım” sinyalini oluşturur. Fiyat yeni bir maksimum seviyeye ulaştığında ve Williams AD bu seviyeye ulaşamıyor ise, güvenlik dağıtımının gerçekleştiği anlamına gelir. Bu durum “Satım” sinyalini oluşturur.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
                //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
                SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
                var candles = GetCandles(Sembol,Periyot);

                // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
                var indicator = Engine.WilliamsAccDist(candles);

        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
                if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
                  SonYon="BUY";
                  SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
                }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
                else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
                  SonYon="SELL";
                  SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
                }
        }
```

## Yavaş Stokastik

Hızlı Stokastik verilerini hesaplar. 2 ayrı çizgi oluşturmaktadır. Bu indikatör için referans çizgileri 20 ve 80'dir. Yavaş Stokastik'te %K eğrisinin %D eğrisini yukarı doğru kesmesi "al" sinyali olarak kabul edilirken %K eğrisinin %D eğsini yukarıdan aşağıya kesişi ise "sat" sinyalini oluşturulmaktadır.

### Kodlama Editörü Örneği

```
//Strateji Girdilerinin(Gloabal Değişkenler) tanımlaması
        public string SonYon="";
        public string Sembol="VAKBN";
        int lot=1;
        public string Periyot="1";

        //İndikatörün parametrelerinin tanımlanması
        int Parametre1=5;
        int Parametre2=3;

        //Strateji çalışmaya başladığında ilk olarak Load fonksiyonunu çalıştırır.
        public void Load()
        {
            //SubscribePrice(Sembol) metodu ile o sembol'e abone olur ve fiyat dinlemeye başlar.
            SubscribePrice(Sembol);
        }

        //Fiyat değişikliklerinde girilen fonksiyon
        public void PriceChanged(Tick t)
        {
//GetCandles(Sembol,Periyot) metodu ile sembolun o periyottaki açılış,kapanış vs. fiyatlarını tutan bir listeyi çeker.
            var candles = GetCandles(Sembol,Periyot);

            // Aşağıdaki kod satırı ile indikatörümüzü tanımlıyoruz.
            var indicator = Engine.StochasticSlow(candles,Parametre1,Parametre2);


        //Eğer indikatörün bir önceki değeri, iki önceki değerinden büyükse ve SonYon BUY'a eşit değilse Alış Yap
            if(Engine.PreviousValue(indicator, 1)>Engine.PreviousValue(indicator, 2) && SonYon!="BUY"){
              SonYon="BUY";
              SendOrder(Sembol,Directions.BUY,lot,PriceTypes.Market);
            }

        //Eğer indikatörün bir önceki değeri,iki önceki değerinden küçükse ve SonYon SELL'e eşit değilse Satış Yap
            else if(Engine.PreviousValue(indicator,1)<Engine.PreviousValue(indicator, 2) && SonYon!="SELL"){
              SonYon="SELL";
              SendOrder(Sembol,Directions.SELL,lot,PriceTypes.Market);
            }
        }
```

## Zig Zag Puanı

Zig Zag Puanı indikatörü, Zig Zag Percent indikatörü gibi nispeten küçük fiyat hareketlerini filtrelemek için kullanılabilen bir indikatördür. İndikatör yalnızca, göreli bir fiyat hareketi belirlenen sapmadan daha büyükse bir çizgi çizer, böylece piyasa gürültüsünü kaldırarak ve yan hareketleri göz ardı ederek grafiği netleştirir. Otomatik sinyaller üretmez. Dönüş parametresi genellikle 1 olarak kullanılır.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.ZigZagPoints(C, 1); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```

## Zig Zag Yüzdesi

Zig Zag Yüzdesi İndikatörü, nispeten küçük fiyat hareketlerini filtrelemek için kullanılabilen bir indikatördür. İndikatör yalnızca, göreli bir fiyat hareketi belirlenen sapmadan daha büyükse bir çizgi çizer, böylece piyasa gürültüsünü kaldırarak ve yan hareketleri göz ardı ederek grafiği netleştirir. Otomatik sinyaller üretmez. Dönüş parametresi genellikle 5 olarak kullanılır.

### Kodlama Editörü Örneği

```
//Çalıştırmak istediğiniz stratejiyi bu alanda kodlayabilirsiniz.

public string Symbol="ASELS";
public string Period="1";
public string SonYon="";

public void Load()
{
   SubscribePrice(Symbol);
}

public void PriceChanged(Tick t)
{
   var candles=GetCandles(Symbol, Period); // Bar Getirme Fonksiyonu
   var C=Engine.GetPriceList(candles, PriceFields.Close); //Fiyat Listesini Getirme Fonksiyonu
   var indikator=Engine.ZigZagPercent(C, 5); // 1 çizgisi mevcuttur.

   //Alış Koşulu (Tamamen Örnek Amaçlıdır.)
   if(t.Price<Engine.LastValue(indikator) && SonYon==""){
      SonYon="A";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.BUY, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
   //Satış Koşulu (Tamamen Örnek Amaçlıdır.)
   else if(t.Price>Engine.LastValue(indikator) && SonYon=="A"){
      SonYon="";
      SendMessage(MessageTypes.Log, Engine.LastValue(indikator).ToString()); //Log Yazdırma Fonksiyonu
      SendOrder(Symbol, Directions.SELL, 1, PriceTypes.Market); //Emir Gönderme Fonksiyonu
   }
}
public void OrderStatusChanged(Order o)
{
}
```