

Руководство пользователя по языку программирования MGX-C



Содержание

| | |
|---|-----------|
| Часть I О языке | 3 |
| Часть II Раздел init | 3 |
| 1 Описание индикатора | 3 |
| Описание источника | 6 |
| Описание выходных данных | 6 |
| Имя индикатора | 7 |
| Текстовое описание | 8 |
| Графическое представление | 9 |
| Вертикальный масштаб | 10 |
| Вертикальная шкала | 11 |
| Текущее значение | 11 |
| 2 Изменяемые параметры | 12 |
| Часть III Раздел make_dot | 14 |
| 1 Получение изменяемых параметров | 14 |
| 2 Изменение параметров родительского индикатора | 15 |
| 3 Получение исходных данных | 15 |
| 4 фиксация результата | 18 |
| Часть IV Дополнительные возможности | 18 |
| Часть V Предустановленные индикаторы | 19 |
| 1 Индикатор Bars | 19 |
| 2 Индикатор Tick count | 19 |
| 3 Индикатор Acceleration/Deceleration | 19 |
| 4 Индикатор Accumulation/Distribution | 20 |
| 5 Индикатор Alligator | 20 |
| 6 Индикатор Average True Range | 21 |
| 7 Индикатор Awesome Oscillator | 21 |
| 8 Индикатор Bollinger Bands | 22 |
| 9 Индикатор Commodity Channel Index | 22 |
| 10 Индикатор DeMarker | 23 |
| 11 Индикатор Elder-Rays | 23 |
| 12 Индикатор Envelopes | 23 |
| 13 Индикатор Force Index | 24 |
| 14 Индикатор Gator Oscillator | 24 |
| 15 Индикатор Momentum | 25 |

| | | |
|---------------------------------------|---|-----------|
| 16 | Индикатор Money Flow Index | 25 |
| 17 | Индикатор Moving Average | 25 |
| 18 | Индикатор Moving Average Convergence/Divergence | 26 |
| 19 | Индикатор Moving Average of Oscillator | 26 |
| 20 | Индикатор Relative Strength Index | 27 |
| 21 | Индикатор Rate of Change | 27 |
| 22 | Индикатор Stochastic Oscillator | 28 |
| 23 | Индикатор Williams' Percent Range | 28 |
| Часть VI Графические примитивы | | 28 |
| 1 | Свечки | 29 |
| 2 | Бары | 30 |
| 3 | Линии | 31 |
| 4 | Гистограммы | 31 |
| Предметный указатель | | 33 |

1 О языке

Уважаемый пользователь!

Коллектив разработчиков представляет язык программирования MGX-C. Язык предназначен для создания индикаторов для программы технического анализа MoneyGraphX.

Основой языка MGX-C является язык C/C++. Синтаксис и семантика языка полностью соответствуют стандарту C/C++ и могут быть изучены по любой книге, описывающей эти языки.

Специфика языка MGX-C заключается лишь в том, что в него входят дополнительные библиотеки, функции которых позволяют получать исходные данные для индикатора, оперировать с ними и передавать результат программе, строящей индикатор. Кроме того, правильно написанный индикатор должен содержать два обязательных раздела (две функции). Они предназначены для описания свойств и атрибутов индикатора и вычисления значений индикатора в каждой точке. Более подробно об этом можно прочитать в соответствующих главах настоящего руководства: раздел `init` и раздел `make_dot`.

дополнительные возможности.

Indicator Editor поставляется почти два десятка индикаторов, присутствующих в программе MoneyGraphX. С кодом этих индикаторов можно ознакомиться, использовать их как примеры, изменять по своему усмотрению.

2 Раздел `init`

Раздел `init` предназначен для описания свойств и атрибутов индикатора. К ним относятся: имя индикатора (полное и сокращенное), текстовое описание индикатора, описание графического представления индикатора при его отрисовке в окне программы MoneyGraphX и т. д. Также в этом разделе можно ввести собственные свойства индикатора. Эти свойства, будучи отображены в диалоге свойств индикатора, дадут пользователю возможность настроить индикатор соответствующим образом.

Раздел `init` выполняется один раз при создании экземпляра индикатора в программе MoneyGraphX.

Следует отметить, что в раздел `init` может быть помещен и любой другой код, который будет исполнен при исполнении раздела. Однако делать это следует очень осторожно, отдавая себе отчет в том, зачем это сделано.

2.1 Описание индикатора

Индикатор - это результат математических расчетов на основе каких-либо данных. Данными могут служить цены, объемы, а также значения других индикаторов. Но в конечном счете первичными данными всех расчетов все равно являются цены. Цены меняются во времени, поэтому в истории цен каждая конкретная цена имеет свое время (находится в определенной точке временной оси). На этих же точках временной оси будут находиться и значения индикаторов.

Таким образом история цен образует числовую последовательность, выстроенную по времени. Результаты математических расчетов, основанные на этих данных, также образуют числовую последовательность. Индикатор, являющийся продуктом таких расчетов, может содержать одну или несколько последовательностей. На их основе могут быть рассчитаны числовых последовательности других ("дочерних") индикаторов. Индикатор-источник будет в этом случае "родительским" индикатором.

При построении индикатора на экране по горизонтали откладывается ось времени, по

вертикали - ось значений. Каждый элемент числовых последовательностей индикатора однозначно определяет в этих координатах точку. Точки индикатора отображаются различными графическими примитивами:

- ломаной линией, соединяющей точки каждой последовательности
- гистограммой, свечой или баром, которые строятся по точкам из одного и того же временного отрезка.

Пример:



На рисунке показаны два индикатора: Bars (история цен) и Envelopes.

Индикатор Bars имеет 4 числовых последовательности: Open (цены открытия), High (максимальные цены интервала), Low (минимальные цены интервала), Close (цены закрытия). Их значения можно представить в виде таблицы:

| Время | 9:38 | 9:39 | 9:40 | 9:41 | 9:42 | 9:43 | 9:44 | 9:45 |
|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| Open | 2.2846 | 2.2849 | 2.2846 | 2.2844 | 2.2846 | 2.2846 | 2.2847 | 2.2845 |
| High | 2.2850 | 2.2849 | 2.2846 | 2.2846 | 2.2848 | 2.2848 | 2.2847 | 2.2847 |
| Low | 2.2846 | 2.2846 | 2.2843 | 2.2841 | 2.2845 | 2.2846 | 2.2845 | 2.2845 |
| Close | 2.2850 | 2.2847 | 2.2845 | 2.2845 | 2.2848 | 2.2846 | 2.2846 | 2.2846 |

Значения числовых последовательностей отображены в виде свечек. Первая свечка построена по значениям имеющим время 9:38 (первый столбец таблицы). Вторая свечка построена по значениям имеющим время 9:39 (второй столбец таблицы). И так далее...

Индикатор Envelopes имеет 2 числовых последовательности: Upper band (верхняя линия) и Lower band (нижняя линия). Их значения можно представить в виде таблицы:

| Время | 9:38 | 9:39 | 9:40 | 9:41 | 9:42 | 9:43 | 9:44 | 9:45 |
|------------|---------|---------|---------|---------|---------|---------|---------|---------|
| Upper band | 2.28540 | 2.28538 | 2.28532 | 2.28530 | 2.28528 | 2.28524 | 2.28522 | 2.28521 |
| Lower band | 2.28445 | 2.28444 | 2.28443 | 2.28440 | 2.28438 | 2.28437 | 2.28435 | 2.28430 |

Значения числовых последовательностей отображены в виде линий. Верхняя линия (синяя) построена по числовой последовательности Upper band (первая строка таблицы). Нижняя линия (красная) построена по числовой последовательности Lower band (вторая строка

таблицы).

2.1.1 Описание источника

Источником данных для индикатора всегда служат числовые последовательности других индикаторов. В простейшем случае такими индикаторами являются значения цен и объемов. Источник данных задается функцией `set_description`, в первом параметре которой указано ключевое слово `ADD_INPUT`.

```
void set_description(ADD_INPUT, const char* alias, const char* indicator_name,
const char* line_name);
```

Параметры

alias

Параметр *alias* задает имя, по которому в дальнейшем можно будет обратиться к числовой последовательности индикатора.

indicator_name

Параметр задает индикатор, числовая последовательность которого будет использоваться как источник данных. Строка должна содержать имя файла индикатора с расширением.

line_name

Параметр задает имя числовой последовательности, которая будет использоваться как источник данных. Имена последовательностей индикаторов можно посмотреть в разделе Предустановленные индикаторы.

Возвращаемое значение

Функция не возвращает значения.

Замечания

Вызов функции `set_description` с ключевым словом `ADD_INPUT` осуществляется столько раз, сколько требуется источников данных.

Пример

```
set_description(ADD_INPUT, "src close", "bars.dll", "Close");
set_description(ADD_INPUT, "src open", "bars.dll", "Open");
set_description(ADD_INPUT, "src high", "bars.dll", "High");
```

Требования к версии

MoneyGraphX 2.0

2.1.2 Описание выходных данных

Выходные данные индикатора образуют последовательность точек в координатах время-значение. Каждый индикатор может содержать сколько угодно таких последовательностей. Например у индикатора *Moving Average* одна единственная последовательность, отображаемая линией, у индикатора *Чарты* (ценовой график) таких последовательностей четыре - цены открытия, максимальные цены, минимальные цены, цены закрытия и эти последовательности могут отображаться свечами барами или линиями. Чтобы индикатор мог заполнять числовые последовательности они должны быть введены (созданы) с помощью функции `set_description`, в первом параметре которой указано ключевое слово `ADD_OUTPUT`.

```
void set_description(ADD_OUTPUT, const char* alias);
```

Параметры

alias

Параметр *alias* задает имя выходной последовательности. По нему в дальнейшем можно будет обратиться к выходной последовательности для записи посчитанных значений, а так же из другого индикатора, если данный индикатор будет использоваться как источник данных (см. Описание источника).

Возвращаемое значение

Функция не возвращает значения.

Замечания

Вызов функции `set_description` с ключевым словом `ADD_OUTPUT` осуществляется столько раз, сколько требуется выходных последовательностей.

Пример

```
set_description(ADD_OUTPUT, "MA");
```

Требования к версии

MoneyGraphX 2.0

2.1.3 Имя индикатора

Язык программирования индикаторов позволяет задать индикатору два имени: сокращенное и полное. Оба имени используются в программе технического анализа для вывода названия индикатора. Для экономии места может использоваться сокращенное имя, в остальных случаях выводится полное.

Сокращенное имя задается с помощью функции `set_description` в первом параметре которой указано ключевое слово `SNAME`.

Полное имя задается с помощью функции `set_description`, в первом параметре которой указано ключевое слово `NAME`.

```
void set_description(SNAME, const char* short_name);
```

Параметры

short_name

Параметр определяет сокращенное имя индикатора.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
set_description(SNAME, "MA");
```

Требования к версии

MoneyGraphX 2.0

```
void set_description(NAME, const char* full_name);
```

Параметры

full_name

Параметр определяет полное имя индикатора..

Возвращаемое значение

Функция не возвращает значения.

Пример

```
set_description(NAME, "Moving Average");
```

Требования к версии

MoneyGraphX 2.0

2.1.4 Текстовое описание

При создании индикатора есть возможность задать ему текстовое описание. Описание носит информативный характер. Обычно оно выводится в окнах, содержащих список доступных индикаторов (окно построения нового индикатора и т. п.). Текстовое описание задается с помощью функции `set_description`, в первом параметре которой указано ключевое слово `DESCRIPTION`.

```
void set_description(DESCRIPTION, const char* text);
```

Параметры

text

Параметр должен содержать текст с описанием.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
set_description(DESCRIPTION, "Индикатор простое скользящее среднее");
```

Требования к версии

MoneyGraphX 2.0

Также имеется возможность задать индикатору номер его версии. Это текстовая строка носящая исключительно информативный характер и предназначенная для упрощения отслеживания версий индикатора в процессе его разработки. Задается версия с помощью функции `set_description` в первом параметре которой указано ключевое слово `VERSION`.

```
void set_description(VERSION, const char* version);
```

Параметры

version

Параметр должен содержать текст с версией индикатора.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
set_description(VERSION, "Ver 1.0");
```

Требования к версии

MoneyGraphX 2.0

2.1.5 Графическое представление

Как правило, любой индикатор представляется на экране некоторой графической схемой. Под графической схемой (графическим представлением) подразумевается набор графических примитивов, с помощью которых числовая последовательность, полученная при вычислении индикатора, отображается на экране (образуя график). Графические примитивы входят в программу MoneyGraphX в виде подключаемых модулей (плагинов). Набор доступных модулей и их подробное описание можно посмотреть в разделе Графические примитивы.

Для задания графической схемы индикатора используется функция `set_description`, в первом параметре которой указано ключевое слово `VIEW`.

```
void set_description(VIEW, const char* scheme_name, const char* plugin_name, int point_count, [const char* point_name]...);
```

Параметры

scheme_name

Параметр задает имя графической схем, в которую мы ходим включить графический примитив.

plugin_name

Задаёт имя файла модуля (плагина) с графического примитива. Имя задается без расширения! Список доступных модулей и их подробное описание можно посмотреть в разделе Графические примитивы.

point_count

Задаёт количество выходных числовых последовательностей, по которым будет строится примитив. Например, для линии - это одна единственная последовательность, для свечки - это четыре последовательности: цены открытия, максимальные цены, минимальные цены, цены закрытия (см. Описание выходных данных).

point_name

Задаёт имена одной или нескольких выходных числовых последовательностей, которые будут отображаться данным графическим примитивом. Имена должны быть из набора имен, заданных в функции `set_description` с ключевым словом `ADD_OUTPUT` (см. Описание выходных данных). Количество имен должно строго соответствовать параметру `point_count`.

Возвращаемое значение

Функция не возвращает значения.

Замечания

Для добавления в графическую схему нескольких примитивов вызов функции `set_description` с ключевым словом `VIEW` и именем этой схемы осуществляется столько раз, сколько примитивов требуется включить в схему.

Для каждого индикатора может быть задано несколько различных графических схем (например, история цен может быть представлена свечками, барами или линиями). Для этого следует вызвать функцию `set_description` с ключевым словом `VIEW` и именем задаваемой схемы.

Пример

```
// индикатор "Bars"  
set_description(VIEW, "Свечки", "Candle", 4, "Open", "High", "Low", "Close");  
set_description(VIEW, "Бары", "fbar", 4, "Open", "High", "Low", "Close");
```

```
set_description(VIEW, "Линия по закрытию цен", "FMLine", 1, "Close");  
// индикатор "MACD"  
set_description(VIEW, "Line/Histogram", "FMLine", 1, "Signal");  
set_description(VIEW, "Line/Histogram", "FHist", 2, "MACD", "Zero");
```

Требования к версии

MoneyGraphX 2.0

2.1.6 Вертикальный масштаб

При отрисовке графика индикатора на экране, он автоматически масштабируется вдоль вертикальной оси. Способ масштабирования задается с помощью функции `set_description`, в первом параметре которой указано ключевое слово `SCALE..`

```
void set_description(SCALE, int scale_type);
```

Параметры

scale_type

Параметр задает способ масштабирования индикатора вдоль вертикальной шкалы. Он может принимать одно из следующих значений:

`SCALE_AUTO` - график индикатора растягивается вдоль вертикальной шкалы от минимального значения до максимального. Минимальное и максимальное значение ищется на временном отрезке отображаемом в данный момент на экране.

`SCALE_FIX` - график индикатора растягивается вдоль вертикальной шкалы от значения, заданного параметром `MIN_VALUE`, до значения, заданного параметром `MAX_VALUE`.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
set_description(SCALE, SCALE_FIX);
```

Требования к версии

MoneyGraphX 2.0

Если параметр `SCALE` задан как `SCALE_FIX`, то необходимо задать параметры `MIN_VALUE` и `MAX_VALUE`. Они задаются с помощью функции `set_description` в первом параметре которой указано ключевое слово `MIN_VALUE` (`MAX_VALUE`).

```
void set_description(MIN_VALUE, double min_value);  
void set_description(MAX_VALUE, double max_value);
```

Параметры

min_value (max_value)

Параметр задает нижнее значение диапазона отображения.

Замечание: даже если число не содержит дробной части, она должна присутствовать в записи числа!

Например: **100** - неправильно, **100.0** - правильно.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
// для индикатора RSI
set_description(MIN_VALUE, 0.0);
set_description(MAX_VALUE, 100.0);
```

Требования к версии

MoneyGraphX 2.0

2.1.7 Вертикальная шкала

Любой индикатор может быть построен в одной из нескольких вертикальных шкал отличающихся единицами измерения. На данный момент доступны две шкалы: ценовая шкала и числовая шкала. Принадлежность индикатора к той или иной шкале определяет его вертикальный масштаб относительно других индикаторов, расположенных в том же окне. Шкала задается с помощью функции `set_description`, в первом параметре которой указано ключевое слово `UNITS..`

```
void set_description(UNITS, int units);
```

Параметры

units

Параметр задает тип вертикальной шкалы. Допустимые значения:

`UNIT_ABS_NUMBER` - произвольная числовая шкала. Индикатор, построенный в этой шкале, масштабируется в соответствии со значением параметра `SCALE` независимо от других индикаторов, построенных в том же окне.

`UNIT_MONEY` - ценовая шкала. Индикатор, построенный в этой шкале, масштабируется с учетом масштаба других индикаторов, построенных в этом же окне и этой же шкале. Так, например, строится индикатор Moving average на графике цен.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
// для индикатора Moving average
set_description(UNITS, UNIT_MONEY);
```

Требования к версии

MoneyGraphX 2.0

2.1.8 Текущее значение

При выводе графика индикатора бывает удобно видеть текущее значение индикатора в виде горизонтальной линии на графике. Чтобы включить эту возможность в индикаторе, следует вызвать функцию `set_description`, в первом параметре которой указано ключевое слово `LAST_POINT_LINE`. В дальнейшем отображение этой линии можно будет включить/отключить через диалог свойств графического окна. Если показывать эту линию не требуется совсем, то задавать этот параметр не надо.

```
void set_description(LAST_POINT_LINE, const char* point_name);
```

Параметры

point_name

Имя выходной последовательности последнее (текущее) значение которой требуется отображать на графике горизонтальной линией.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
// индикатор Bars
set_description(LAST_POINT_LINE, "Close");
```

Требования к версии

MoneyGraphX 2.0

2.2 Изменяемые параметры

Помимо стандартных параметров индикатора, задающих основные атрибуты и свойства, предусмотрена возможность задавать собственные параметры. Значения этих параметров могут быть использованы при вычислении значений индикатора. Так же они могут быть выведены в диалог свойств индикатора и, таким образом, позволяют писать индикаторы свойства которых могут быть настроены пользователем. Создания собственных параметров служит функция `init_param`.

```
void init_param(const char* name, int visible, int type, int def_value, int
min_value, int max_value, int value_step);
void init_param(const char* name, int visible, int type, double def_value,
double min_value, double max_value, double value_step);
void init_param(const char* name, int visible, int type, int string_count,
[const char* string]...);
```

Параметры

name

Задает имя вводимого параметра. По этому имени можно будет обращаться к параметру в коде индикатора. Именно это имя будет выводиться в диалоге свойств индикатора в качестве имени настраиваемого параметра.

visible

Определяет, должен ли параметр выводиться в диалог свойств индикатора.
Допустимые значения:

PT_VISIBLE - параметр отображается в диалоге свойств индикатора и его значение может быть изменено пользователем.

PT_INVISIBLE - параметр не отображается в диалоге свойств индикатора и его значение может быть изменено только в коде самого индикатора.

type

Задает тип параметра. Допустимые значения:

PT_INT - задает целочисленный тип параметра. Соответствует типу `int` языка C/C++.

PT_FLOAT - задает параметр типа число с плавающей точкой. Соответствует типу `double` языка C/C++.

PT_STRING - задает строковый параметр. Соответствует типу `char*` языка C/C++.

В зависимости от типа создаваемого параметра используется тот или иной прототип функции `init_param`.

Для типа PT_INT:

```
void init_param(const char* name, int visible, int type, int def_value, int
min_value, int max_value, int value_step);
```

def_value

Задаёт начальное значение создаваемого параметра.

min_value

Задаёт минимально возможное значение создаваемого параметра.
Пользователь не сможет задать значение меньше, чем указано здесь.

max_value

Задаёт максимально возможное значение создаваемого параметра.
Пользователь не сможет задать значение больше, чем указано здесь.

value_step

Задаёт шаг с которым может меняться значение создаваемого параметра. Если шаг задан равным нулю, то значение параметр может меняться с любым шагом.

Для типа PT_FLOAT:

```
void init_param(const char* name, int visible, int type, double def_value,
double min_value, double max_value, double value_step);
```

def_value

Задаёт начальное значение создаваемого параметра.

min_value

Задаёт минимально возможное значение создаваемого параметра.
Пользователь не сможет задать значение меньше, чем указано здесь.

max_value

Задаёт максимально возможное значение создаваемого параметра.
Пользователь не сможет задать значение больше, чем указано здесь.

value_step

Задаёт шаг с которым может меняться значение создаваемого параметра. Если шаг задан равным нулю, то значение параметр может меняться с любым шагом.

Замечание: даже если число не содержит дробной части, она должна присутствовать в записи числа!

Например: **100** - неправильно, **100.0** - правильно.

Для типа PT_STRING:

```
void init_param(const char* name, int visible, int type, int string_count,
[const char* string]...);
```

string_count

Задаёт количество строковых значений одно из которых сможет принять создаваемый параметр.

string

Задаёт одну или несколько строк. Их количество должно строго равняться значению *string_count*. Значение создаваемого параметра в будущем может быть равно только какой-либо из них. Начальное значение параметра будет равно первой из заданных строк.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
// для индикатора Moving average
init_param("Уровень сглаживания", PT_VISIBLE, PT_INT, 7, 1, 100, 1);
init_param("MA Type", PT_VISIBLE, PT_STRING, 4, "Simple", "Exponential",
"Smoothed", "Linear Weighted");

// для индикатора Envelopes
init_param("Deviation %", PT_VISIBLE, PT_FLOAT, 0.1, 0.1, 100.0, 0.1);
```

Требования к версии

MoneyGraphX 2.0

3 Раздел `make_dot`

Раздел `make_dot` должен содержать код, реализующий собственно логику индикатора. Принцип работы этого раздела должен быть следующим. Программа MoneyGraphX будет вызывать на исполнение раздел `make_dot` для каждой точки временной оси в которой должно быть посчитано значение индикатора. Раздел `make_dot` должен содержать действия, которые, основываясь на источниках данных (история цен, значения других индикаторов и т. д.), позволят посчитать очередное значение создаваемого индикатора и передать его с помощью специальных функций программе MoneyGraphX.

Следует отметить, что в раздел `make_dot` может быть помещен и любой другой код, который будет исполнен при исполнении раздела. Однако делать это следует очень осторожно, отдавая себе отчет в том, зачем это сделано.

3.1 Получение изменяемых параметров

Перед собственно вычислением значения индикатора целесообразно получить текущие значения введенных в разделе `init` параметров (см Изменяемые параметры). Это делается вызовом функции `get_param`.

```
int get_param(const char* name);
double get_param(const char* name);
char* get_param(const char* name);
```

Параметры

name

Задаёт имя параметра, значение которого надо получить. Параметр с таким именем должен быть создан в разделе `init` (см Изменяемые параметры). Имя параметра должно задаваться с учетом регистра букв.

Возвращаемое значение

В зависимости от типа параметра будет вызвана одна из трех функций, которая и вернет в качестве значения текущее значение параметра с именем *name*.

Пример

```
// из индикатора Envelopes
int ma_period = get_param("MA Period");
char* ma_type = get_param("MA Type");
char* ma_by = get_param("MA by");
double deviation = get_param("Deviation %");
```

Требования к версии

MoneyGraphX 2.0

3.2 Изменение параметров родительского индикатора

Очень часто вычисление одного индикатора основано на значениях другого (родительского). При этом возникает необходимость настроить параметры родительского индикатора. Для этого используется функция `set_parent_param`.

```
set_parent_param(const char* source_name, const char* param_name, int value);
set_parent_param(const char* source_name, const char* param_name, double value);
set_parent_param(const char* source_name, const char* param_name, const char*
value);
```

Параметры

source_name

Имя источника данных. Источник с таким именем должен быть определен в разделе `init` (см. Описание источника).

param_name

Задаёт имя параметра родительского индикатора. Параметр с таким именем должен быть определен в родительском индикаторе. Для более подробной информации см. описание предустановленных индикаторов.

value

Значение, присваиваемое параметру. В зависимости от типа параметра будет вызвана одна из трех функций.

Возвращаемое значение

Функция не возвращает значения.

Пример

```
// из индикатора Envelopes

int ma_period = get_param("MA Period");
char* ma_type = get_param("MA Type");
char* ma_by = get_param("MA by");
double deviation = get_param("Deviation %");

set_parent_param("MA", "Period", ma_period);
set_parent_param("MA", "Type", ma_type);
set_parent_param("MA", "By", ma_by);
```

Требования к версии

MoneyGraphX 2.0

3.3 Получение исходных данных

Общая схема получения исходных данных следующая. Функцией `get` необходимо получить указатель на соответствующую точку в источнике данных (родительском индикаторе). Если нам необходимы еще и соседние к полученной точке, то мы получаем их с помощью функций `prev` и `next`, которые возвращают указатель соответственно на предыдущую и следующую точку по отношению к указанной. При получении данных может оказаться, что они отсутствуют в источнике. В этом случае указатель принимает специальное значение, проверить которое можно вызовом функции `eodata`. Для получения из указателя значения служит функция `value`. Также доступна функция `date_time`, позволяющая получить по указателю дату и время указанной точки.

```
data_ptr get(const char* source_name);
```

Параметры

source_name

Имя источника данных. Источник с таким именем должен быть определен в разделе `init` (см. Описание источника).

Возвращаемое значение

Функция возвращает указатель на текущую точку источника данных (родительского индикатора).

Требования к версии

MoneyGraphX 2.0

```
data_ptr prev(data_ptr ptr);
```

Параметры

ptr

Указатель на ранее полученную точку.

Возвращаемое значение

Функция возвращает указатель точку предыдущую по отношению к указанной.

Требования к версии

MoneyGraphX 2.0

```
data_ptr next(data_ptr ptr);
```

Параметры

ptr

Указатель на ранее полученную точку.

Возвращаемое значение

Функция возвращает указатель точку следующую за указанной.

Требования к версии

MoneyGraphX 2.0

```
int eodata(data_ptr ptr);
```

Параметры

ptr

Указатель на ранее полученную точку.

Возвращаемое значение

Функция возвращает 0 если указатель ссылается на полученные данные и значение отличное от нуля, если данные отсутствуют.

Требования к версии

MoneyGraphX 2.0

```
double value(data_ptr ptr);
```

Параметры

ptr

Указатель на ранее полученную точку.

Возвращаемое значение

Функция возвращает значение точки на которую ссылается указатель.

Замечания

Если указатель ссылается на отсутствующие данные, функция вернет 0. Чтобы этого не произошло следует проверять полученный указатель функцией `eodata`.

Требования к версии

MoneyGraphX 2.0

```
SYSTEMTIME date_time(data_ptr ptr);
```

Параметры

ptr

Указатель на ранее полученную точку.

Возвращаемое значение

Функция возвращает дату и время точки в формате `SYSTEMTIME`.

Замечания

Структура `SYSTEMTIME` имеет следующий вид:

```
typedef struct _SYSTEMTIME {
    WORD wYear;
    WORD wMonth;
    WORD wDayOfWeek;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
    WORD wMilliseconds; } SYSTEMTIME, *PSYSTEMTIME;
```

За более подробными сведениями по работе со временем следует обратиться к документации по Win32 API.

Требования к версии

MoneyGraphX 2.0

Пример

```
// из индикатора Moving average
//
int period = get_param("Уровень сглаживания");
//
double val = 0;
data_ptr data_ptr = get("src close");
//
for(int i=0; i<period; i++)
{
    if(eodata(data_ptr)) return;
    val += value(data_ptr);
    data_ptr = prev(data_ptr);
}
```

```
}
```

3.4 фиксация результата

После того, как индикатор вычислил значение своей точки это значение следует передать вызывающей программе (зафиксировать результат). Для этого служит функция `output`. Если значение индикатора посчитать не удалось, то функцию `output` вызвать не надо. Система устроена таким образом, что если для вычисляемой точки функция не будет вызвана, то эта точка будет отсутствовать в индикаторе (и, соответственно, на его графике).

```
double& output(const char* out_name);
```

Параметры

`out_name`

Имя выходной последовательности в которую добавляется очередная посчитанная точка.

Возвращаемое значение

Функция не возвращает значения.

Замечания

Функцию `output` следует вызывать для каждой числовой последовательности, заданной в описании индикатора (см. Описание выходных данных).

Пример

```
// из индикатора Moving average
//
int period = get_param("Уровень сглаживания");
//
double val = 0;
data_ptr data_ptr = get("src close");
//
for(int i=0; i<period; i++)
{
    if(eodata(data_ptr)) return;
    val += value(data_ptr);
    data_ptr = prev(data_ptr);
}
output("МА") = val / period;
```

Требования к версии

MoneyGraphX 2.0

4 Дополнительные возможности

Основой данного языка написания индикаторов является язык C/C++. Поэтому при написании индикатора программисту доступны практически все возможности этих языков. Кроме того, с программой поставляются стандартные библиотеки функций языка C (доступны при включении соответствующих заголовочных файлов: `stdio`, `stdlib` и т. д.), библиотеки STL, а также библиотека функций для работы с Win32 API (доступна при подключении файла `windows.h`, хотя ее возможности серьезно ограничены).

5 Предустановленные индикаторы

В программу входит около двух десятков индикаторов, которые могут быть использованы в качестве источника данных. Далее можно посмотреть подробное описание каждого из них.

5.1 Индикатор Bars

Имя файла

bars.dll

Выходные данные

Open - цены открытия

High - максимальные цены

Low - минимальные цены

Close - цены закрытия

Параметры

Отсутствуют.

Текущая версия

1.0

5.2 Индикатор Tick count

Имя файла

volume.dll

Выходные данные

Volume - количество тиков в интервале

Параметры

Отсутствуют.

Текущая версия

1.0

5.3 Индикатор Acceleration/Deceleration

Имя файла

ac.dll

Выходные данные

AC - значение индикатора ускорения/замедления

Параметры

| Имя параметра | Тип параметра | Значения |
|-----------------|---------------|-------------|
| Period | PT_INT | от 1 до 100 |
| Parent Period 1 | PT_INT | от 1 до 100 |
| Parent Period 2 | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.4 Индикатор Accumulation/Distribution

Имя файла

ad.dll

Выходные данные

AD - значение индикатора накопления/распределения

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| Period | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.5 Индикатор Alligator

Имя файла

alligator.dll

Выходные данные

Jaw - значение линии Jaw

Teeth - значение линии Teeth

Lips - значение линии Lips

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|--|
| Jaw Period | PT_INT | от 1 до 100 |
| Jaw Shift | PT_INT | от 1 до 100 |
| Teeth Period | PT_INT | от 1 до 100 |
| Teeth Shift | PT_INT | от 1 до 100 |
| Lips Period | PT_INT | от 1 до 100 |
| Lips Shift | PT_INT | от 1 до 100 |
| MA Type | PT_STRING | "Simple", "Exponential", "Smoothed", "Linear Weighted" |
| MA by | PT_STRING | "Close", "Open", "High", "Low", "Median price (HL/2)", "Typical price (HLC/3)", "Weighted close (HLCC/4)" |

Текущая версия

1.0

5.6 Индикатор Average True Range**Имя файла**

atr.dll

Выходные данные

ATR - значение индикатора Средний Истинный Диапазон

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| Period | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.7 Индикатор Awesome Oscillator**Имя файла**

ao.dll

Выходные данные

АО - значение индикатора Awesome Oscillator

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| Period1 | PT_INT | от 1 до 100 |
| Period2 | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.8 Индикатор Bollinger Bands

Имя файла

bb.dll

Выходные данные

TL - верхняя линия

ML - средняя линия

BL - нижняя линия

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|---|
| Period | PT_INT | от 1 до 100 |
| Deviations | PT_INT | от 1 до 100 |
| By | PT_STRING | "Close", "Open", "High", "Low", "Median price (HL/2)", "Typical price (HLC/3)", "Weighted close (HLCC/4)" |

Текущая версия

1.0

5.9 Индикатор Commodity Channel Index

Имя файла

cci.dll

Выходные данные

CCI - значение индикатора Индекс Товарного Канала

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|---|
| Period | PT_INT | от 1 до 100 |
| By | PT_STRING | "Typical price (HLC/3)", "Close", "Open", "High", "Low", "Median price (HL/2)", "Weighted close (HLCC/4)" |

Текущая версия

1.0

5.10 Индикатор DeMarker

Имя файла

dem.dll

Выходные данные

DEM - значение индикатора DeMarker

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| Period | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.11 Индикатор Elder-Rays

Имя файла

elder_rays.dll

Выходные данные

Bulls - значение линии "сила быков"

Bears - значение линии "сила медведей"

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| EMA Period | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.12 Индикатор Envelopes

Имя файла

envelopes.dll

Выходные данные

Upper band - значения верхней огибающей линии

Lower band - значения нижней огибающей линии

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|---|
| MA Period | PT_INT | от 1 до 100 |
| MA Type | PT_STRING | "Simple", "Exponential", "Smoothed", "Linear Weighted" |
| MA by | PT_STRING | "Close", "Open", "High", "Low", "Median price (HL/2)", "Typical price (HLC/3)", "Weighted close (HLCC/4)" |
| Deviation % | PT_FLOAT | от 0.1 до 100 с шагом 0.1 |

Текущая версия

1.0

5.13 Индикатор Force Index

Имя файла

frc.dll

Выходные данные

FRC - значение индикатора Индекс Силы

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|---|
| MA Period | PT_INT | от 1 до 100 |
| MA Type | PT_STRING | "Simple", "Exponential", "Smoothed", "Linear Weighted" |
| MA by | PT_STRING | "Close", "Open", "High", "Low", "Median price (HL/2)", "Typical price (HLC/3)", "Weighted close (HLCC/4)" |

Текущая версия

1.0

5.14 Индикатор Gator Oscillator

Имя файла

gator.dll

Выходные данные

Delta1 - значения верхней гистограммы

Delta2 - значения нижней гистограммы

Параметры

Отсутствуют.

Текущая версия

1.0

5.15 Индикатор Momentum

Имя файла

momentum.dll

Выходные данные

Momentum - значение индикатора Momentum

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| Period | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.16 Индикатор Money Flow Index

Имя файла

mfi.dll

Выходные данные

MFI - значение индикатора Индекс Денежных Поток

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| Period | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.17 Индикатор Moving Average

Имя файла

MA.dll

Выходные данные

MA - значение индикатора Скользящее среднее

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|---|
| Period | PT_INT | от 1 до 30 |
| Type | PT_STRING | "Simple", "Exponential", "Smoothed", "Linear Weighted" |
| By | PT_STRING | "Close", "Open", "High", "Low", "Median price (HL/2)", "Typical price (HLC/3)", "Weighted close (HLCC/4)" |

Текущая версия

1.0

5.18 Индикатор Moving Average Convergence/Divergence

Имя файла

macd.dll

Выходные данные

MACD - значение индикатора MACD (Схождение/Расхождение Скользящих Средних)

Signal - значение сигнальной линии индикатора MACD

Параметры

| Имя параметра | Тип параметра | Значения |
|-----------------|---------------|---|
| Fast EMA period | PT_INT | от 1 до 100 |
| Slow EMA period | PT_INT | от 1 до 100 |
| MACD SMA period | PT_INT | от 1 до 100 |
| By | PT_STRING | "Close", "Open", "High", "Low", "Median price (HL/2)", "Typical price (HLC/3)", "Weighted close (HLCC/4)" |

Текущая версия

1.0

5.19 Индикатор Moving Average of Oscillator

Имя файла

osma.dll

Выходные данные

OsMA - значение индикатора OsMA (Скользящая Средняя Осциллятора)

Параметры

| Имя параметра | Тип параметра | Значения |
|-----------------|---------------|---|
| Fast EMA period | PT_INT | от 1 до 100 |
| Slow EMA period | PT_INT | от 1 до 100 |
| MACD SMA period | PT_INT | от 1 до 100 |
| By | PT_STRING | "Close", "Open", "High", "Low", "Median price (HL/2)", "Typical price (HLC/3)", "Weighted close (HLCC/4)" |

Текущая версия

1.0

5.20 Индикатор Relative Strength Index

Имя файла

RSI.dll

Выходные данные

RSI - значение индикатора RSI (Индекс относительной силы)

Buy - значение линии уровня перекупленности

Sell - значение линии уровня перепроданности

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-----------------------|
| Period | PT_INT | от 1 до 100 |
| Buy level | PT_FLOAT | от 1 до 100 с шагом 1 |
| Sell level | PT_FLOAT | от 1 до 100 с шагом 1 |

Текущая версия

1.0

5.21 Индикатор Rate of Change

Имя файла

roc.dll

Выходные данные

RoC - значение индикатора Rate of Change

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| Period | PT_INT | от 1 до 100 |

Текущая версия

1.0

5.22 Индикатор Stochastic Oscillator

Имя файла

stochastic.dll

Выходные данные

%K - значение линии %K

%D - значение линии %D

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------------------------|
| %K period | PT_INT | от 1 до 100 |
| %D period | PT_INT | от 1 до 100 |
| Slowing | PT_INT | от 1 до 100 |
| Price field | PT_STRING | "Low / High", "Close / Close" |

Текущая версия

1.0

5.23 Индикатор Williams' Percent Range

Имя файла

r.dll

Выходные данные

%R - значение индикатора %R (Процентный Диапазон Вильямса)

Параметры

| Имя параметра | Тип параметра | Значения |
|---------------|---------------|-------------|
| Period | PT_INT | от 1 до 100 |

Текущая версия

1.0

6 Графические примитивы

Как правило, любой индикатор представляется на экране некоторой графической схемой. Под графической схемой (графическим представлением) подразумевается набор графических примитивов, с помощью которых числовая последовательность, полученная при вычислении индикатора, отображается на экране (образуя график). Графические примитивы входят в программу MoneyGraphX в виде подключаемых модулей (плагинов). Описание доступных примитивов можно посмотреть в соответствующих разделах:

Свечи

Бары

Линии

Гистограммы

6.1 Свечи



Графический примитив "свечи". Используется чаще всего при выводе истории котировок.

Название модуля

Candle

Число точек (числовых последовательностей), отображаемых примитивом

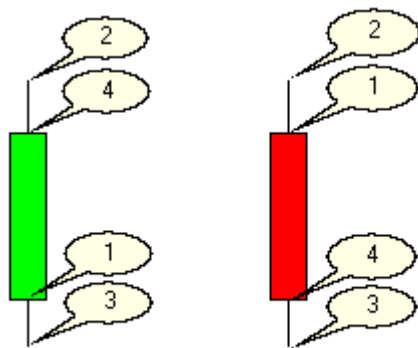
4

Порядок отображения точек

Точка №2 всегда задает верхний конец фитиля

Точка №3 всегда задает нижний конец фитиля

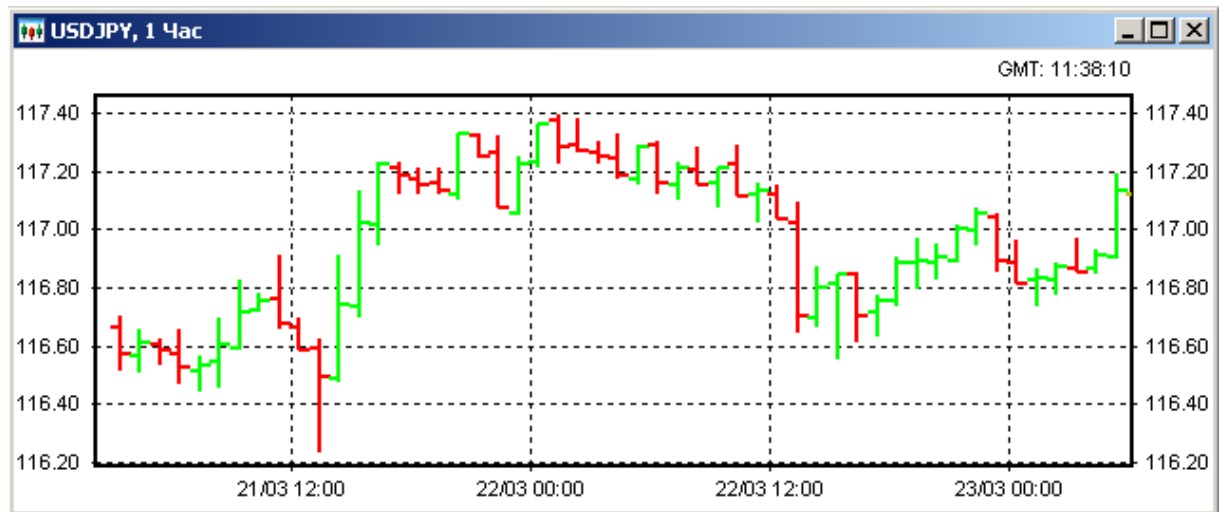
Из точек №1 и №4 точка с большим значением отображается вверху, точка с меньшим - внизу.



Пример использования

```
// из индикатора Bars
set_description(VIEW, "Свечи", "Candle", 4, "Open", "High", "Low", "Close");
```

6.2 Бары



Графический примитив "бары". Используется чаще всего при выводе истории котировок.

Название модуля

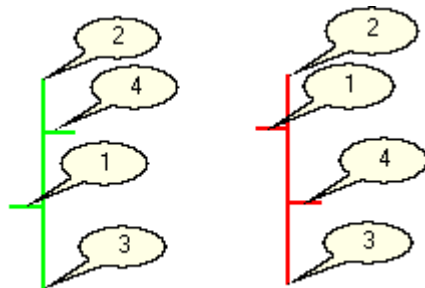
fbar

Число точек (числовых последовательностей), отображаемых примитивом

4

Порядок отображения точек

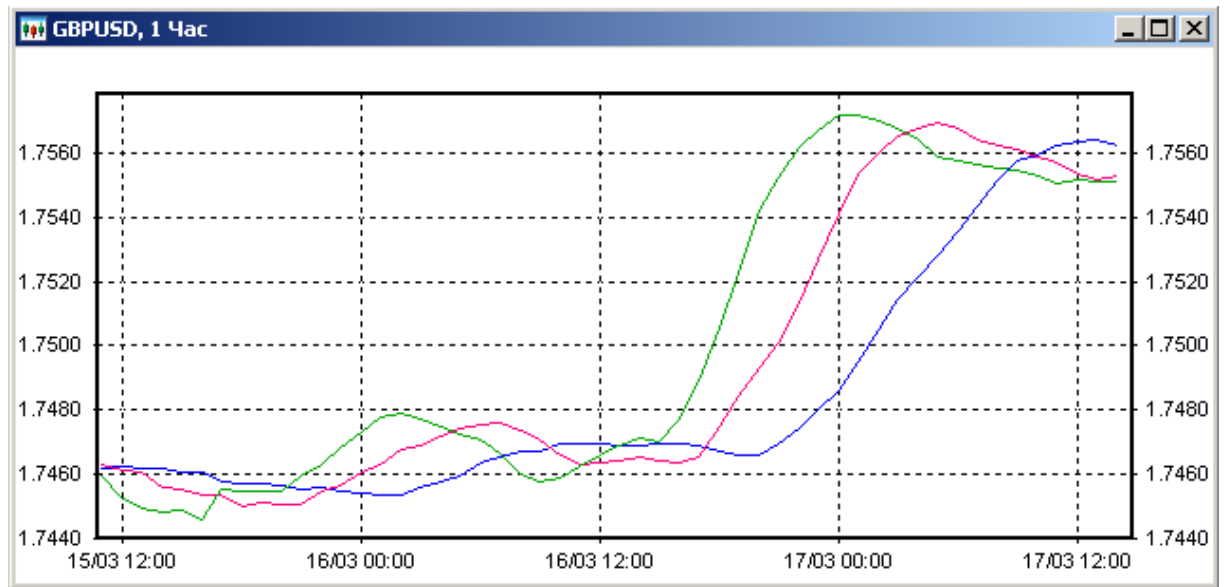
- Точка №2 задает верхний конец бара
- Точка №3 задает нижний конец бара
- Точка №1 задает левую засечку
- Точка №4 задает правую засечку



Пример использования

```
// из индикатора Bars
set_description(VIEW, "Бары", "fbar", 4, "Open", "High", "Low", "Close");
```

6.3 Линии



Графический примитив "линия".

Название модуля

FMLine

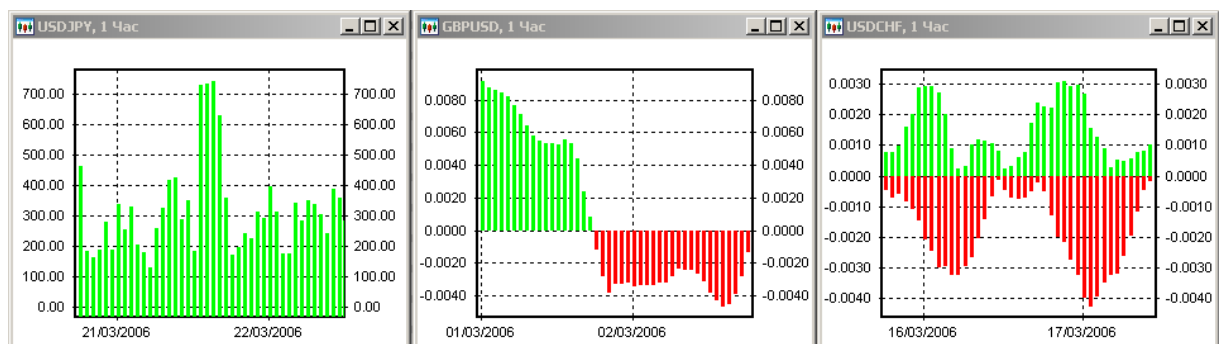
Число точек (числовых последовательностей), отображаемых примитивом

1

Пример использования

```
// из индикатора Alligator
set_description(VIEW, "Line", "FMLine", 1, "Jaw");
set_description(VIEW, "Line", "FMLine", 1, "Teeth");
set_description(VIEW, "Line", "FMLine", 1, "Lips");
```

6.4 Гистограммы



Графический примитив "гистограмма".

Название модуля

FHist

Число точек (числовых последовательностей), отображаемых примитивом

1, 2 или 3

Порядок отображения точек

Если задана только одна точка, то гистограмма отображается от нижней части окна до значения индикатора.

Если заданы 2 точки, то гистограмма строится от первой до второй.

Если заданы 3 точки, то строится двойная гистограмма: от второй точки до первой и от второй точки до третьей.

Пример использования

```
// из индикатора Tick count
set_description(VIEW, "Гистограмма", "FHist", 1, "Volume");

// из индикатора Awesome Oscillator
set_description(VIEW, "Histogram", "FHist", 2, "Zero", "AO");

// из индикатора Gator Oscillator
set_description(VIEW, "Histogram", "FHist", 3, "Delta1", "Zero", "Delta2");
```

Предметный указатель

- % -

%D 28
%K 28
%R 28

- A -

AC 19
Acceleration/Deceleration 19
Accumulation/Distribution 20
AD 20
ADD_INPUT 6
ADD_OUTPUT 6
Alligator 20
AO 21
ATR 21
Average True Range 21
Awesome Oscillator 21

- B -

Bars 19
BB 22
Bears 23
BL 22
Bollinger Bands 22
Bulls 23

- C -

Candle 29
CCI 22
Close 19
Commodity Channel Index 22

- D -

date_time 15
Delta1 24
Delta2 24
DEM 23
DeMarker 23
DESCRIPTION 8

- E -

Elder-Rays 23
Envelopes 23

- F -

fbar 30
FHist 31
FMLine 31
Force Index 24
FRC 24

- G -

Gator Oscillator 24
get 15
get_param 14

- H -

High 19

- I -

init 3
init_param 12

- J -

Jaw 20

- L -

LAST_POINT_LINE 11

Lips 20
Low 19
Lower band 23

- M -

MA 25
MACD 26
make_dot 14
MAX_VALUE 10
MFI 25
MIN_VALUE 10
ML 22
Momentum 25
Money Flow Index 25
Moving Average 25
Moving Average Convergence/Divergence 26
Moving Average of Oscillator 26

- N -

NAME 7
next 15

- O -

Open 19
OsMA 26
output 18

- P -

prev 15

- R -

Rate of Change 27
Relative Strength Index 27
RoC 27
RSI 27

- S -

SCALE 10
SCALE_AUTO 10
SCALE_FIX 10

set_description 6, 7, 8, 9, 10, 11
ADD_INPUT 6
ADD_OUTPUT 6
DESCRIPTION 8
LAST_POINT_LINE 11
MAX_VALUE 10
MIN_VALUE 10
NAME 7
SCALE 10
SNAME 7
UNITS 11
VERSION 8
VIEW 9

set_parent_param 15
Signal 26
SNAME 7
Stochastic Oscillator 28

- T -

Teeth 20
Tick count 19
TL 22

- U -

UNIT_ABS_NUMBER 11
UNIT_MONEY 11
UNITS 11
Upper band 23

- V -

value 15
VERSION 8
VIEW 9
Volume 19

- W -

Williams' Percent Range 28