Техническое задание по созданию программы для Meta Trader 4

**Камикадзе ver. 5.0**

# Введение

Данный советник разработан с целью получения положительного торгового результата в случае резкого кратковременного движения рынка против тренда или в боковом движении (флете). Кроме того, настройки советника позволяют достаточно эффективно использовать его и при затухании тренда, которое характеризуется резким импульсом по направлению тренда перед его коррекцией.

# Требования к написанию программного кода

При написании программы необходимо соблюдать следующие условия:

1. Использование в коде переменных синтаксиса из технического задания. В случае необходимости создать новые переменные – давать им краткое описание в коде и вносить соответствующие изменения в ТЗ;
2. Сохранять структуру пользовательских переменных для логической настройки программы;
3. Структура программного кода должна быть схожей с блок-схемой, прилагаемой к ТЗ. В начале каждого блока необходимо указывать содержимое в виде комментария, аналогичное операторам из блок-схемы, которая может быть изменена по соглашению сторон – перед началом работы подумать об оптимизации;
4. Программный код должен позволять вносить в себя изменения в рамках одного блока блок-схемы без существенных изменений во всем коде – программы, а очередность фильтров не должна быть сложно изменяемой;
5. Обеспечить (дописать ТЗ и реализовать в алгоритме программы) работу советника при его включении/отключении с уже выставленными ордерами, а так же при потери связи с сервером или ином сбое;
6. Если отключение проверок при работе на реальном счете позволяет существенно снизить время на работу с ордерами в тестере, то необходимо реализовать переключатель для работы в тестере или на реальном счете;
7. Допущенные логические ошибки в ТЗ подлежат исправлению по соглашению сторон, верным ТЗ считается исправленный вариант.

# Требования к тестированию готового решения, сдаче работы и исправлению ошибок

1. Любые выявленные существенные ошибки в коде после принятия работ должны быть устранены, согласно техническому заданию ТЗ в течение 14 календарных дней.
2. Тестирования промежуточной версии советника не может превышать 3 (трёх) календарных дней.
3. Срок исправления ошибки в коде в период разработки программы не может превышать 3 (трёх) календарных дней.
4. Если по не зависимым от стороны причинам одна из сторон не может исполнить оговоренные сроки, то другая сторона должна быть уведомлена о дате, когда свои обязательства сторона сможет исполнить.
5. Если одна из сторон не исполнила свои обязательства в течение 30 дней, то другая сторона может отказаться от дальнейших исполнений своих обязательств. Программист – забрать деньги, а заказчик – отменить заказ.

# Алгоритм по блок-схеме:

**1. Блок "Start"**

Начало выполнения программы.

Чтение настроек пользователя:

extern string text1 = "1. Настройка координат и стратегии";

extern string text2 = "1.1 Настройка средней скользящей и отступа для верхнего конверта";

//maH – среднее скользящее:

extern int TFH=0;//тайм фрейм H

extern int pMAH=128;//период H средней

extern int priceMAH=0;//цена H средней

extern int typeMAH=0;//метод H средней

extern int shiftMAH=0;//сдвиг H средней

//priceSell – цена, по которой выставляется отложенный ордер на продажу по magicA;

extern int pipsXH=180; // величина пунктов для расчета расстояния от maH, на котором выставляется отложенный ордер на продажу

extern string text3 = "1.2 Настройка средней скользящей и отступа для нижнего конверта";

//maL – среднее скользящее:

extern int TFL=0;//тайм фрейм L

extern int pMAL=128;//период L средней

extern int priceMAL=0;//цена L средней

extern int typeMAL=0;//метод L средней

extern int shiftMAL=0;//сдвиг L средней

//priceBuy – цена, по которой выставляется отложенный ордер на покупку по magicA;

extern int pipsXL=180; // величина пунктов для расчета расстояния от maL, на котором выставляется отложенный ордер на покупку

extern string text4 = "1.3 Настройка средней скользящей для определения цикла и цели";

//maT – среднее скользящее:

extern int TFT=0;//тайм фрейм T

extern int pMAT=128;//период T средней

extern int priceMAT=0;//цена T средней

extern int typeMAT=0;//метод T средней

extern int shiftMAT=0;//сдвиг T средней

extern string text5 = "1.4 Настройка блока maMirror":

// Настройка блока «maMirror»

//maMirrorZ (настройки по iMA)

extern int tf\_maMirrorZ=0;//таймфрейм средней

extern int p\_maMirrorZ=128;//период средней

extern int price\_maMirrorZ=0;//цена средней

extern int type\_maMirrorZ=0;//метод средней

extern int shift\_maMirrorZ=0;//сдвиг средней

extern int pipsXHmaMz=300;//Расстояние в пунктах от maMirror

extern int pipsXLmaMz=300;// Расстояние в пунктах от maMirror

extern string text6 = "1.5 Выбор и настройка типа используемых ордеров покупка/продажа и размера лота для magicA ";

//magicA – идентификационное ("магическое") число для основных ордеров;

extern int magicA=1;

//

extern int OpenSell=1; // если значение в настройках советника равно 1, то ордера на продажу открываются, если 0, то не открываются.

extern int OpenBuy=1; // если значение в настройках советника равно 1, то ордера на покупку открываются, если 0, то не открываются.

//

extern int MaxOrder=0;//Проверка условия на блокировку открытия максимального количества ордеров, если 0, то не используется

//

extern int lotM=0;//вариант изменения размера лота при повторном открытии отложенного ордера до достижения maT (до начала нового цикла)

extern double lot=0.1;//размер начального лота

extern int RoundLot=2;//Знаков лота

extern string text7 = "1.6 Выбор метода и настройка тейк профита";

extern int metodTP=0 // метод расчета тейк профита

extern int pointTP=0 // расстояния для выставления тейк профита в пунктах

extern int TPerror=100 // расстояния для выставления тейк профита в пунктах в случае ошибки

extern string text8 = "1.7 Выбор метода и настройка стоп лосса";

extern int metodSL=0 // метод расчета стоп лосса

extern int pointSL=0 // расстояния для выставления стоп лосса в пунктах

extern int SLerror=100 // расстояние для выставление стоп лосса в пунктах в случае ошибки

extern int Levl\_Zerro=0;//0-не используем,!=0-модернизируем стоп лосс до максимально заданного значения, отсчет идет от цены открытия ордера, отрицательное значение значит, что стоп лосс переводиться в положительное значение (перекрывают в плюс безубыток).

extern int InsideMA=0;// При выходе maMirrorZ за пределы канала из MA+-Pips стоп лосс не выставляется, фильтры не срабатывают – при значении 1, а при значении 0 – данное условие игнорируется.

extern int Bezubitok=0;// метод перевода в безубыток

extern string text9 = "2 Тактика и фильтры";

extern string text10 = "2.1 Настройка цикла";

extern int cCycle=0; // 0 – не закрывать открытые ордера при новом цикле, 1 – закрывать открытые ордера при новом цикле.

extern int CycleBarPeriod=0;// – максимальное количество баров, на протяжение которых может действовать Cycle=1 (Разрешения на выставление отложенных ордеров), если CycleBarPeriod=0, то постоянно до появления нового цикла или паузы (Cycle=2).

extern int CycleBarPause=0;// – максимальное количество баров, на протяжение которых может действовать Cycle=2 (Блокировка разрешения на выставление отложенных ордеров), если CycleBarPause=0, то постоянно до появления нового цикла.

extern string text11 = "2.2 Фильтры на отложенные и открытые ордера";

extern string text12 = "2.2.1 Осцилятор";

extern string text13 = "2.2.1.1 Выбор осцилятора: 1 – Procent\_3MA ; 2 – RSI ";

extern int oscillator=0;// - выбираем какой осциллятор будет использоваться, 0 – не использовать, 1 – «Procent\_3MA», 2 – RSI.

extern string text14 = "2.2.1.2 Настройка фильтра, работающего по осциллятору: 0 – не использовать; 1 – удаляем все отложенные ордера, 2 – удаляем все отложенные и открытые ордера, если выходят из диапазона";

// Настройка фильтра, работающего по осциллятору.

extern int BlockDiap=0;//0-не используем,1-удаляем все отложенные ордера и больше не выставляем до выход из диапазона

//2-удаляем все отложенные ордера и больше не выставляем до выход из диапазона, и закрываем открытые ордера

extern double SDiapАА=100.0;//не торгуем продажу - начало

extern double SDiapАZ=100.0;//не торгуем продажу - конец

extern double SDiapBА=100.0;//не торгуем продажу - начало

extern double SDiapBZ=100.0; //не торгуем продажу - конец

extern double BDiapАА=0.0;//не торгуем покупку - начало

extern double BDiapАZ=0.0;//не торгуем покупку - конец

extern double BDiapBА=0.0;//не торгуем покупку - начало

extern double BDiapBZ=0.0;//не торгуем покупку - конец

//Конец настройки осциллятора "Procent\_3MA"

extern string text15 = "2.2.1.3 Настройки осциллятора Procent\_3MA";

//Настройки осциллятора "Procent\_3MA"

extern int \_pMAH=68;//период H средней

extern int \_shiftMAH=0;//сдвиг H средней

extern int \_typeMAH=0;//метод H средней

extern int \_priceMAH=0;//цена H средней

extern int \_tfMAH=0;//таймфрейм H средней

extern int \_pMAL=14;//период L средней

extern int \_shiftMAL=0;//сдвиг L средней

extern int \_typeMAL=0;//метод L средней

extern int \_priceMAL=0;//цена L средней

extern int \_tfMAL=0;//таймфрейм L средней

extern int \_pMAT=3;//период T средней

extern int \_shiftMAT=0;//сдвиг T средней

extern int \_typeMAT=0;//метод T средней

extern int \_priceMAT=0;//цена T средней

extern int \_tfMAT=0;//таймфрейм T средней

extern int \_pipsXH=50;//величина пунктов от ma для расчета коэффициента, на который будет скорректирован показатель maH

extern int \_pipsXL=50;//величина пунктов от ma для расчета коэффициента, на который будет скорректирован показатель maL

extern int \_pp=2;//точка точности - сколько знаков после запятой учитывать при расчете индикатора

extern string text16 = "2.2.1.3 Настройки осциллятора RSI ";

//Настройки осциллятора "RSI"

extern int periodRSI=14; // Период RSI

extern int priceRSI=1; // Тип цен, по которым рассчитывается RSI

extern string text17 = "2.2.2 Проверка условия на блокировку открытия отложенных ордеров, при условии если тренды с верхних тайм фремов синхронизированы (-1) или если тренды рассинхронизированы (1), 0 – не используется";

extern int fmaF=0;// фильтр для блокировки открытия ордеров в противотренде (1) и в тренде (-1) с верхних тайм фреймов

//maF – среднее скользящее:

extern int TFF=0;//тайм фрейм F

extern int pMAF=128;//период F средней

extern int priceMAF=0;//цена F средней

extern int typeMAF=0;//метод F средней

extern int shiftMAF=0;//сдвиг F средней

extern string text18 = "2.2.3 Проверка условия на закрытие отложенного ордера по времени в течении которого отложенный ордер был выставлен: 0 – не используется ";

extern int PreOrderLife; // – количество баров, в течении которых не будет закрыт выставленный отложенный ордер, 0 – не используется ;

extern string text19 = "2.2.4 Проверка условия на закрытие отложенного(открытого) ордера по времени в течении которого открытый ордер был в рынке: 0 – не используется";

extern int OrderLife=0;//Проверка условия на закрытие отложенного(открытого) ордера по времени в течении которого открытый ордер был в рынке

extern int foOrderLife=0;//Если 1 – закрывать и блокировать выставление отложенных ордеров, 0 – не закрывать и не блокировать выставление отложенных ордеров

extern string text20 = "2.2.5 Проверка условия на закрытие отложенного(открытого) ордера по признаку сробатывания стоп лосса на открытом ордере: 0 – не используется";

extern int blockSL=0;//Проверка условия на закрытие отложенного(открытого) ордера по признаку сробатывания стоп лосса на открытом ордере

extern int foblockSL=0;// Если 1 – закрывать и блокировать выставление отложенных ордеров, 0 – не закрывать и не блокировать выставление отложенных ордеров

extern string text21 = "2.2.6 Проверка условия на закрытие отложенного ордера по признаку пересечения цены и maMirrorZ: 0 – не используется , 1 – Блокирует отложенные ордера, 2 – Блокирует отложенные и удаляет открытые ордера";

extern int open\_maMirrorZ=0;// Проверка условия на закрытие отложенного ордера по признаку пересечения цены и maMirrorZ (Если 0 – не используется / 1 – Блокирует отложенные ордера/ 2 – Блокирует отложенные и удаляет открытые ордера)

extern string text22 = "2.2.7 Проверка условия на блокировку открытия отложенных ордеров на заданное количество баров, после выставления отложенного ордера: 0 – не используется ";

extern int OpenBar=0;//Проверка условия на блокировку открытия отложенных ордеров на заданное количество баров, после выставления отложенного ордера

extern string text23 = "2.2.8 Проверка условия на блокировку открытия отложенных ордеров, при условии если заданное количество раз цена не пересекает priceBuy/priceSell

: 0 – не используется ";

extern int barX=3;//Проверка условия на блокировку открытия отложенных ордеров, при условии если заданное количество раз цена не пересекает priceBuy/priceSell

extern string text24 = "2.2.9 Проверка условия на блокировку открытия отложенных ордеров - дает разрешение на открытие отложенного ордера после пропуска открытия указанное количество раз, если значение в настройках советника равно 0, то сигнал на открытие ордера после фильтрации не пропускается, если >0, то пропускается столько раз, сколько указанно в значении переменной: 0 – не используется ";

extern int fPropusk=0;//Проверка условия на блокировку открытия отложенных ордеров - дает разрешение на открытие отложенного ордера после пропуска открытия указанное количество раз, если значение в настройках советника равно 0, то сигнал на открытие ордера после фильтрации не пропускается, если >0, то пропускается столько раз, сколько указанно в значении переменной.

extern string text25 = "3. Выбор и настройка типа используемых ордеров покупка/продажа и размера лота для magicB ";

//magicB – идентификационное ("магическое") число для страховочных ордеров.

extern int magicB=2; // если 0, то не открываются.

//

extern int UPpips=500; // расстояние выставления ордера на покупку от начала цикла в пунктах

extern int DOWNpips=500; // расстояние выставления ордера на продажу от начала цикла в пунктах

extern int Ktrend=1;// коэффициент для определения размера лота

//

Int Суcle=-1 // значение цикла при запуске советника – необходима для корректного расчёта maMirrorZ , и возможно для продолжение работы с уже открытыми ордерами.

//Объявляем все переменные фильтров и счетчиков и приравниваем их к нулю

extern int StopO=0;

extern int oCycleS=0;

extern int oCycleB=0;

extern int oBlockDiapS=0;

extern int oBlockDiapB=0;

extern int oPreOrderLifeS=0;

extern int oPreOrderLifeB=0;

extern int oOrderLifeS=0;

extern int oOrderLifeB=0;

extern int oBlockSLS=0;

extern int oBlockSLB=0;

extern int oOpen\_maMirrorZS=0;

extern int oOpen\_maMirrorZB=0;

extern int StopF=0;

extern int fCycleS=0;

extern int fCycleB=0;

extern int fBlockDiapS=0;

extern int fBlockDiapB=0;

extern int fPreOrderLifeS=0;

extern int fPreOrderLifeB=0;

extern int fOrderLifeS=0;

extern int fOrderLifeB=0;

extern int fMaxOrderS=0;

extern int fMaxOrderB=0;

extern int fBarXS=0;

extern int fBarXB=0;

extern int fBlockSLS=0;

extern int fBlockSLB=0;

extern int fOpen\_maMirrorZS=0;

extern int fOpen\_maMirrorZB=0;

extern int fPropuskB=0;

extern int fPropuskS=0;

extern int fmaFS=0;

extern int fmaFB=0;

extern int omaFS=0;

extern int omaFB=0;

extern int NOpenBarS(0)=-1;

extern int NOpenBarB(0)=-1;

extern int fOpenBarS=0;

extern int fOpenBarB=0;

extern int StopC=0;

extern int cCycleS=0;

extern int cCycleB=0;

extern int cBlockDiapS=0;

extern int cBlockDiapB=0;

extern int cOrderLifeS=0;

extern int cOrderLifeB=0;

extern int cOpen\_maMirrorZS=0;

extern int cOpen\_maMirrorZB=0;

extern int CycleBarT=0;

extern int CycleBarP=0;

extern int fPropusk=0;

extern int OpenOrderS=0;

extern int OpenOrderB=0;

extern int lotVS=0;

extern int lotVB=0;

extern int nBarXs=0;

extern int nBarXb=0;

extern int LastTPs=0;

extern int LastTPb=0;

extern int LastSLs=0;

extern int LastSLb=0;

extern int StopO=0;

extern int StopF=0;

extern int StopC=0;

//

**2.1 Блок «Задержка»**

Фактическая задержка перед появлением нового бара.

Если постоянный опрос советником информации о появлении нового бара не загружает ЦПУ, то используем как абстракцию, если же загружает существенно, то делаем опрос о новом баре раз в 5 секунд

http://www.mql5.com/ru/articles/159

**2.2 Блок «Новый бар (Yes/No)»**

Проверка на открытие нового бара на текущем тайм фрейме.

Если бар открыт, то переходим к следующему блоку.

Если бар не открыт, то возвращаемся к блоку № 2.1

**4.1 Блок "Определение цикла работы советника и типа ордеров"**

Данный блок предназначен для определения состояния (цикла), в котором должен работать советник и с какими типами ордеров он будет работать (на покупку или продажу).

Cycle – проверка на наличие нового цикла:

Cycle=-1 – цикл при запуске советника

Cycle=0 – новый цикл – необходимо обнулить накопленные данные

Cycle=1 – старый цикл – продолжаем расчет

Cycle=2 – блокировка действий по выставлению ордеров в цикле

CycleOld – массив, в который сохраняется информация о цикле на прошлом баре (2 значения), необходим для расчета maMirrorZ.

CyclePrice – цена, при которой начался новой цикл (произошло пересечение ценой maT).

*В коде советника все скользящие средние вызываются оператором iMA.*

maT – среднее скользящее, настройки, которой задает пользователь;

maTm – среднее скользящее по настройкам значения maT, за исключением сдвига – он +1(maT(1)) – для использования данных maT прошлого бара на текущем баре;

Проверка новый или старый цикл:

Если maTm<High(1) и maTm>Low(1), тогда Cycle =0 и CycleOld(0)=0 и CyclePrice=maTm

иначе Cycle=1 и CycleOld(0)=1

Если Cycle=0, тогда: // обнуляем все переменные фильтров, массивов и счетчиков

StopO=0;

oCycleS=0;

oCycleB=0;

oBlockDiapS=0;

oBlockDiapB=0;

oPreOrderLifeS=0;

oPreOrderLifeB=0;

oOrderLifeS=0;

oOrderLifeB=0;

oBlockSLS=0;

oBlockSLB=0;

oOpen\_maMirrorZS=0;

oOpen\_maMirrorZB=0;

StopF=0;

fCycleS=0;

fCycleB=0;

fBlockDiapS=0;

fBlockDiapB=0;

fPreOrderLifeS=0;

fPreOrderLifeB=0;

fOrderLifeS=0;

fOrderLifeB=0;

fMaxOrderS=0;

fMaxOrderB=0;

fBarXS=0;

fBarXB=0;

fBlockSLS=0;

fBlockSLB=0;

fOpen\_maMirrorZS=0;

fOpen\_maMirrorZB=0;

fPropuskB=0;

fPropuskS=0;

fmaFS=0;

fmaFB=0;

omaFS=0;

omaFB=0;

NOpenBarS(0)=-1;

NOpenBarB(0)=-1;

fOpenBarS=0;

fOpenBarB=0;

StopC=0;

cCycleS=0;

cCycleB=0;

cBlockDiapS=0;

cBlockDiapB=0;

cOrderLifeS=0;

cOrderLifeB=0;

cOpen\_maMirrorZS=0;

cOpen\_maMirrorZB=0;

CycleBarT=0;

CycleBarP=0;

fPropusk=0;

OpenOrderS=0;

OpenOrderB=0;

lotVS=0;

lotVB=0;

nBarXs=0;

nBarXb=0;

LastTPs=0;

LastTPb=0;

LastSLs=0;

LastSLb=0;

Если Cycle=1, тогда: // обнуляем результат расчета флагов

StopO=0;

StopF=0;

StopC=0;

TipOrder – ордера на продажу, если равен 1, и ордера на покупку, если равен 2;

Если Cycle=1, то проверяем Low(1)>maTm(1), если неравенство верно, то TipOrder=1, если нет, то TipOrder=2

CycleBarPeriod – максимальное количество баров, на протяжение которых может действовать Cycle=1 (Разрешения на выставление отложенных ордеров), если CycleBarPeriod=0, то постоянно до появления нового цикла или паузы (Cycle=2).

CycleBarP – переменная для расчета оставшегося количества баров до окончания цикла.

Если CycleBarPeriod!=0, тогда

Если CycleBarPeriod>CycleBarP, то CycleBarP+1

Иначе Cycle=2;

CycleBarPause – максимальное количество баров, на протяжение которых может действовать Cycle=2, если CycleBarPause=0 – то до нового цикла.

CycleBarT – переменная для расчета оставшегося количества баров до начала нового цикла.

//Значение фильтров, которые блокируют открытие ордеров до конца цикла.

Если Cycle!=0, и одно из равенств верно:

Cycle=2;

fMaxOrderS=1;

fMaxOrderB=1;

oPreOrderLifeS=1;

oPreOrderLifeB=1;

cOrderLifeS=1 и foOrderLife=1;

cOrderLifeB=1 и foOrderLife=1;

fOpen\_maMirrorZS=1;

fOpen\_maMirrorZB=1;

oOpen\_maMirrorZS=1;

oOpen\_maMirrorZB=1;

cOpen\_maMirrorZS=1;

cOpen\_maMirrorZS=1;

fBlockSLS=1 и foblockSL=1;

fBlockSLB=1 и foblockSL=1;

oBlockSLS=1 и foblockSL=1;

oBlockSLB=1 и foblockSL=1;

то Cycle=2 иначе Cycle=1

Если Cycle=2 и CycleBarPause!=0 и CycleBarT< CycleBarPause , то CycleBarT= CycleBarT+1 и Cycle=2

иначе Cycle=1

Если Cycle=2 и CycleBarPause=0, то Cycle=2

**3. Блок "Открыт страховочный ордер (Yes/No)"**

Проверяем на открытие страховочного ордера по magicB.

Если страховочный ордер открыт, то возвращаемся к блоку № 2.1

Если страховочный ордер не открыт, то переходим к следующему блоку.

**4.2 Блок "Блок на использование ордеров "**

OpenSell – если значение в настройках советника равно 1, то ордера на продажу открываются, если 0, то не открываются.

OpenBuy – если значение в настройках советника равно 1, то ордера на покупку открываются, если 0, то не открываются.

ЛОГИКА:

Для ордеров на продажу:

Если TipOrder=1 и OpenSell=1, то TipOrder=1

Иначе TipOrder=0

Для ордеров на покупку:

Если TipOrder=2 и OpenBuy=1, то TipOrder=2

Иначе TipOrder=0

**4.3 Блок "Определение условий необходимых для выставления отложенного ордера"**

В данном блоке производим расчет координат (цифровых значений цены, необходимых для дальнейшей работы советника), а так же часть данных для дальнейшего использования:

В коде советника все скользящие средние вызываются оператором iMA.

//

maH – среднее скользящее, настройки, которой задает пользователь;

maL – среднее скользящее, настройки, которой задает пользователь;

maT – среднее скользящее, настройки, которой задает пользователь;

//

priceSell – цена, по которой выставляется отложенный ордер на продажу:

priceSell =maH+pipsXH\*Point

//

priceBuy – цена, по которой выставляется отложенный ордер на покупку:

priceBuy =maL-pipsXL\*Point

//

Если BlockDiap!=0 и oscillator=1 тогда, рассчитываем (получаем данные) осциллятора "Procent\_3MA" на прошлом баре(1), используется функция iCustom

Если BlockDiap!=0 и oscillator=2 тогда, рассчитываем (получаем данные) осциллятора "RSI" на прошлом баре(1), используется функция вызова встроенного осцилятора.

BloclDiapS – показывает, попадает ли осциллятор в диапазон (0-нет/1-да) для продажи.

1. Ордер на продажу при значении осциллятора в диапазоне от SDiapAA до SDiapAZ

2. Ордер на продажу при значении осциллятора в диапазоне от SDiapBA до SDiapBZ

BlockDiapB – показывает, попадает ли осциллятор в диапазон (0-нет/1-да) для покупки.

3. Ордер на покупку при значении осциллятора в диапазоне от BDiapAA до BDiapAZ

4. Ордер на покупку при значении осциллятора в диапазоне от BDiapBA до BDiapBZ

5. Значение предела диапазона может быть со знаком "-".

6. Значение предела диапазона может быть равно не целому числу.

В коде обеспечивается возможность работы по пользовательскому осциллятору с одной сигнальной линией при диапазоне значений осциллятора - отдельно для покупки и продажи, по умолчанию используем осциллятор"Procent\_3MA"

//

maMirrorZ – среднее скользящее, настройки, которой задает пользователь;

Если open\_maMirrorZ!=0 и (или) metodSL=2, то производим расчет maMirrorZ по формуле:

Для продажи:

Если Cycle=1 и CycleOld(1)=0 и Low(1)>maTm(1) то,

maMirrorZs(0)= maMirrorZ(1)+ pipsXHmaMz – первая точка,

запоминается число MINmaMirrorZ= maMirrorZs(0) – минимальное значение индикатора за цикл, иначе продолжаем расчет maMirrorZs.

последующие точки расчета (Cycle=1):

Если TipOrder=1 и CycleOld(1)=1

Если maMirrorZ(1) > maMirrorZ(2) тогда Dsma= maMirrorZ(1) - maMirrorZ(2), иначе Dsma=0

Если maMirrorZs(1)-(Dsma)<MINmaMirrorZ

то maMirrorZs(0)=maMirrorZs(1)-(Dsma) и MINmaMirrorZ=maMirrorZs(1)-(Dsma)

иначе maMirrorZs(0)=MINmaMirrorZ

Если Cycle=-1 то определяем точку пересечения (Cycle=0) и делаем ретроспективный перерасчет (алгоритм от программиста).

Для покупки:

Если Cycle=1 и CycleOld(1)=0 и High(1)<maTm(1) то,

maMirrorZb(0)= maMirrorZ(0)-pipsXLmaMz – первая точка,

запоминается число = MAXmaMirrorZ= maMirrorZb(0) – максимальное значение индикатора за цикл, иначе продолжаем расчет maMirrorZb.

последующие точки расчета (Cycle=1):

Если TipOrder=2 и CycleOld(1)=1

Если maMirrorZ(1)< maMirrorZ(2) тогда Dsma= maMirrorZ(1) - maMirrorZ(2), иначе Dsma=0

Если maMirrorZb(1)-(Dsma)>MAXmaMirrorZ

то maMirrorZb(0)= maMirrorZb(1)-( Dsma) и MAXmaMirrorZ= maMirrorZb(1)-( Dsma)

иначе maMirrorZb(0)= MAXmaMirrorZ

Если Cycle=-1 то определяем точку пересечения (Cycle=0) и делаем ретроспективный перерасчет (алгоритм от программиста).

//

nBarX(b/s) – для расчета barX необходимо создать счетчик, который будет считать по следующему алгоритму:

Для продажи:

Если priceSell<High(1) и priceSell>Low(1), то nBarXs=0

Иначе nBarXs=nBarXs+1

Для покупки:

Если priceBuy<High(1) и priceBuy>Low(1), то nBarXb=0

Иначе nBarXb=nBarXb+1

//

maF – среднее скользящее, настройки, которой задает пользователь;

**5. Блок "Есть отложенные ордера (Yes/No)"**

Проверяем на наличие выставленных отложенных ордеров по magicA.

Если есть отложенный ордер, то переходим к блоку № 5.1.1.

Если нет отложенных ордеров, то переходим к блоку № 5.2.1.

**5.1.1 Блок "Проверка условия на закрытие отложенного ордера "oCycle""**

oCycle(B/S) – переменная для ускорения обработки данных (1 – сигнал на закрытие/ 0 – пропуск). Т.е. при новом цикле отложенные ордера закрываются.

StopO – переменная сигнализирующая о закрытии (StopO =1) отложенного ордера на текущем баре.

ЛОГИКА:

Для ордеров на продажу:

Если Cycle=0 , то oCycleS=1 и StopO=1

Иначе oCycleS=0 и StopO=0

Для ордеров на покупку:

Если Cycle=0 , то oCycleB=1 и StopO=1

Иначе oCycleB=0 и StopO=0

**5.1.2 Блок "Проверка условия на закрытие отложенного ордера "oBlockDiap""**

Данный блок использует уже рассчитанные данные пользовательского индикатора.

oBlockDiap (B/S) – возвращает результат фильтрации данных (1 – положительный и 0 – отрицательный)

ЛОГИКА:

Если BlockDiap!=0 и StopO!=1

Для ордеров на продажу:

Если TipOrder=1

Если BlockDiapS=1 , то oBlockDiapS=1 и StopO=1

Иначе oBlockDiapS=0 и StopO=0

Для ордеров на покупку:

Если TipOrder=2

Если BlockDiapB=1 , то oBlockDiapB=1 и StopO=1

Иначе oBlockDiapB=0 и StopO=0

**5.1.3 Блок "Проверка условия на закрытие отложенного ордера "oPreOrderLife""**

Определяем, сколько баров осталось до закрытия выставленного отложенного ордера.

oPreOrderLife (B/S) – возвращает результат фильтрации данных (1 – положительный и 0 – отрицательный)

Nbar – количество баров прошедших с момента выставленного отложенного ордера.

Узнаем дату выставления отложенного ордера, текущую дату бара и таймфрейм выбранный для расчета maT, зная эти данные, определяем, сколько баров было сформировано за этот период времени, при этом бар, на котором был выставлен отложенный ордер, равен единице (как вариант, возможно и альтернативное решение).

ЛОГИКА:

Если PreOrderLife!=0 и StopO!=1

Для ордеров на продажу:

Если TipOrder=1

Если Nbar=PreOrderLife, то oPreOrderLifeS=1 и StopO=1

Иначе oPreOrderLifeS=0 и StopO=0

Для ордеров на покупку:

Если TipOrder=2

Если Nbar=PreOrderLife, то oPreOrderLifeB=1 и StopO=1

Иначе oPreOrderLifeB=0 и StopO=0

**5.1.4 Блок "Проверка условия на закрытие отложенного ордера "oOrderLife""**

oOrderLife (B/S) – возвращает результат фильтрации данных (1 – положительный и 0 – отрицательный). Сигнал фактически построен на результатах обработки прошлого бара, при закрытии открытого ордера по сигналу cOrderLife(B/S).

ЛОГИКА:

Если OrderLife!=0 и foOrderLife=1 и StopO!=1

Для ордеров на продажу:

Если TipOrder=1

Если cOrderLifeS!=0, то oOrderLifeS=1 и StopO=1

Иначе oOrderLifeS=0 и StopO=0

Для ордеров на покупку:

Если TipOrder=2

Если cOrderLifeB!=0, то oOrderLifeB=1 и StopO=1

Иначе oOrderLifeB=0 и StopO=0

**5.1.5 Блок "Проверка условия на закрытие отложенного ордера "oBlockSL""**

Если blockSL=1, то происходит проверка, было ли на прошлом баре закрытие открытого ордера на покупку или продажу по стоп лоссу, если было, то oBlockSLB (или oBlockSLS) =1, иначе 0. (алгоритм реализации готов рассмотреть)

OrderCloseLastBar(B/S) – переменная, которая возвращает положительный (1) или отрицательный (0) результат проверки закрытия открытого ордера на прошлом баре.

ЛОГИКА:

Если blockSL!=0 и foblockSL=1 и StopO!=1

Для ордеров на продажу:

Если TipOrder=1

Если OrderCloseLastBarS!=0, то oBlockSLS=1 и StopO=1

Иначе oBlockSLS=0 и StopO=0

Для ордеров на покупку:

Если TipOrder=2

Если OrderCloseLastBarS!=0, то oBlockSLB=1 и StopO=1

Иначе oBlockSLB=0 и StopO=0

# 5.1.6 Блок "Проверка условия на закрытие отложенного ордера "omaMirrorZ""

oOpen\_maMirrorZ (B/S) – переменная определяющая результат расчета блока.

ЛОГИКА:

Если open\_maMirrorZ!=0 и StopO!=1

Для ордеров на продажу:

Если TipOrder=1

Если Open(0)>maMirrorZs, то oOpen\_maMirrorZS=1 и StopO=1

Иначе oOpen\_maMirrorZS=0 и StopO=0

Если InsideMA=1

Если maMirrorZs>maH+pipsXH , то StopO=0

Иначе StopO= StopO

Для ордеров на покупку:

Если TipOrder=2

Если Open(0)<maMirrorZb, то oOpen\_maMirrorZB=1 и StopO=1

Иначе oOpen\_maMirrorZB=0 и StopO=0

Если InsideMA=1

Если maMirrorZb<maL-pipsXL , то StopO=0

Иначе StopO= StopO

# 5.1.7 Блок "Фильтр omaF"

omaF(B/S) – возвращает результат проверки на открытие ордеров по тренду с верхних тайм фреймов.

ЛОГИКА:

Если maF!=0 и StopO!=1

Для ордеров на продажу:

Если TipOrder=1

Если fmaF=-1 и maF<priceSell, то omaFS=1 и StopO=1

Иначе omaFS=0 и StopO=0

Если fmaF=1 и maF>priceSell, то omaFS=1 и StopO=1

Иначе omaFS=0 и StopO=0

Для ордеров на покупку:

Если TipOrder=2

Если fmaF=-1 и maF>priceBuy, то omaFB=1 и StopO=1

Иначе omaFB=0 и StopO=0

Если fmaF=1 и maF<priceBuy, то omaFB=1 и StopO=1

Иначе omaFB=0 и StopO=0

**5.1.8 Блок "Есть одно условие по результатам проверки (Yes/No)"**

Если следующих равенство верно:

StopO=1 или Cycle=2

то, переходим к пункту № 5.1.7.1

если нет, то переходим к пункту № 5.1.5.2

# 5.1.8.1 Блок "Закрытие отложенных ордеров"

Закрывается обработанный отложенный ордер, и переходим к блоку № 6.

# 5.1.8.2 Блок "Модификация отложенного ордера"

Ордер модифицируется по координатам для выставления нового отложенного ордера.

ЛОГИКА:

Если StopO!=1

Для ордеров на продажу:

Если TipOrder=1

Цена продажи: priceSell =maH+pipsXH\*Point

Тейк профит: 0

Стоп лосс: 0

Для ордеров на покупку:

Если TipOrder=2

Цена продажи: priceBuy =maL-pipsXL\*Point

Тейк профит: 0

Стоп лосс: 0

и переходим к блоку № 6.1

**5.2.1 Блок "Фильтр fCycle"**

fCycle(B/S) – переменная для ускорения обработки данных (1 – сигнал на блокировку выставления отложенных ордеров/ 0 – пропуск). Т.е. при новом цикле отложенные ордера не выставляются.

StopF – переменная сигнализирующая о блокировке выставления отложенного ордера (StopF =1) на текущем баре.

ЛОГИКА:

Для ордеров на продажу:

Если Cycle=0 , то fCycleS=1 и StopF=1

Иначе fCycleS=0 и StopF=0

Для ордеров на покупку:

Если Cycle=0 , то fCycleB=1 и StopF=1

Иначе fCycleB=0 и StopF=0

**5.2.2 Блок "Фильтр fBlockDiap"**

Данный блок использует уже рассчитанные данные пользовательского индикатора.

fBlockDiap (B/S) – возвращает результат фильтрации данных (1 – положительный и 0 – отрицательный). Показывает, попадает ли осциллятор в предварительно заданный в настройках советника диапазон.

ЛОГИКА:

Если BlockDiap!=0 и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если BlockDiapS=1 , то fBlockDiapS=1 и StopF=1

Иначе fBlockDiapS=0 и StopF=0

Для ордеров на покупку:

Если TipOrder=2

Если BlockDiapB=1 , то fBlockDiapB=1 и StopF=1

Иначе fBlockDiapB=0 и StopF=0

**5.2.3 Блок "Фильтр fPreOrderLife"**

fPreOrderLife (B/S) – блокировка выставления отложенных ордеров, если превышено значение времени жизни выставленных отложенных ордеров, которые за заданный период времени в барах небыли открыты. Для расчета берутся данные о закрытие отложенных ордеров по данному признаку.

ЛОГИКА:

Если PreOrderLife!=0 и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если oPreOrderLifeS=1, то fPreOrderLifeS=1 и StopF=1

Иначе fPreOrderLifeS=0 и StopF=0

Для ордеров на покупку:

Если TipOrder=2

Если oPreOrderLifeB=1 , то fPreOrderLifeB=1 и StopF=1

Иначе fPreOrderLifeB=0 и StopF=0

**5.2.4 Блок "Фильтр fOrderLife"**

fOrderLife (B/S) – возвращает результат фильтрации данных (1 – положительный и 0 – отрицательный). Сигнал фактически построен на результатах обработки прошлого бара, при закрытии открытого ордера по сигналу cOrderLife(B/S).

ЛОГИКА:

Если OrderLife!=0 и foOrderLife и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если cOrderLifeS!=0, то fOrderLifeS=1 и StopF=1

Иначе fOrderLifeS=0 и StopF=0

Для ордеров на покупку:

Если TipOrder=2

Если cOrderLifeB!=0, то fOrderLifeB=1 и StopF=1

Иначе fOrderLifeB=0 и StopF=0

**5.2.5 Блок "Фильтр fMaxOrder"**

fMaxOrder (B/S) – возвращает результат фильтрации данных (1 – положительный и 0 – отрицательный). Фактически ограничивает выставление отложенных ордеров, если превышено их количество за цикл. Для определения количества выставленных отложенных ордеров используется блок «Счетчик выставленных ордеров».

ЛОГИКА:

Если MaxOrder!=0 и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если OpenOrderS(1)=>MaxOrder, то fMaxOrderS=1 и StopF=1

Иначе fMaxOrderS=0 и StopF=0

Для ордеров на покупку:

Если TipOrder=2

Если OpenOrderB(1)=>MaxOrder, то fMaxOrderB=1 и StopF=1

Иначе fMaxOrderB=0 и StopF=0

**5.2.6 Блок "Фильтр fBarX"**

fbarX(b/s) – проверка условия, при котором цена не пересекает заданное переменной barX число баров подряд:

ЛОГИКА:

Если BarX!=0 и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если nBarXs=>BarX, то fBarXS=1 и StopF=1

Иначе fBarXS=0 и StopF=0

Для ордеров на покупку:

Если TipOrder=2

Если nBarXb=>BarX, то fBarXB=1 и StopF=1

Иначе fBarXB=0 и StopF=0

# 5.2.7 Блок "Фильтр fBlockSL"

fblockSL(B/S) – возвращает результат проверки, было ли на прошлом баре закрытие открытого ордера на покупку или продажу по стоп лоссу, если было, то fBlockSLB (или fBlockSLS) =1, иначе 0. (алгоритм реализации готов рассмотреть)

OrderCloseLastBar(B/S) – переменная, которая возвращает положительный (1) или отрицательный (0) результат проверки закрытие открытого ордера на прошлом баре.

ЛОГИКА:

Если blockSL!=0 и foblockSL=1 и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если OrderCloseLastBarS!=0, то fBlockSLS=1 и StopF=1

Иначе fBlockSLS=0 и StopF=0

Для ордеров на покупку:

Если TipOrder=2

Если OrderCloseLastBarS!=0, то fBlockSLB=1 и StopF=1

Иначе fBlockSLB=0 и StopF=0

# 5.2.8 Блок "Фильтр fmaF"

fmaF(B/S) – возвращает результат проверки на открытие ордеров по тренду с верхних тайм фреймов.

ЛОГИКА:

Если maF!=0 и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если fmaF=-1 и maF<priceSell, то fmaFS=1 и StopF=1

Иначе fmaFS=0 и StopF=0

Если fmaF=1 и maF>priceSell, то fmaFS=1 и StopF=1

Иначе fmaFS=0 и StopF=0

Для ордеров на покупку:

Если TipOrder=2

Если fmaF=-1 и maF>priceBuy, то fmaFB=1 и StopF=1

Иначе fmaFB=0 и StopF=0

Если fmaF=1и maF<priceBuy, то fmaFB=1 и StopF=1

Иначе fmaFB=0 и StopF=0

**5.2.9 Блок "Фильтр fmaMirrorZ"**

fOpen\_maMirrorZ (B/S) – переменная определяющая результат расчета блока.

Блокировка открытия отложенного ордера, если нельзя будет выставить стоп лосс по maMirrorZ.

ЛОГИКА:

Если open\_maMirrorZ!=0 и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если Open(0)>maMirrorZs, то fOpen\_maMirrorZS=1 и StopF=1

Иначе fOpen\_maMirrorZS=0 и StopF=0

Если InsideMA=1

Если maMirrorZs>maH+pipsXH , то StopF=0

Иначе StopF=StopF

Для ордеров на покупку:

Если TipOrder=2

Если Open(0)<maMirrorZb, то fOpen\_maMirrorZB=1 и StopF=1

Иначе fOpen\_maMirrorZB=0 и StopF=0

Если InsideMA=1

Если maMirrorZb<maL-pipsXL , то StopF=0

Иначе StopF=StopF

# 5.2.10 Блок "Фильтр fOpenBar"

fOpenBar(B/S) – возвращает результат проверки на открытие ордеров. Разрешение дается, если после прошлого разрешения прошло заданное количество баров по тренду с верхних тайм фреймов.

NOpenBar(B/S) – счетчик, считающий бары прошедшие с разрешения на открытие отложенного ордера. По прошествии заданное количество баров, разрешение на открытие отложенного ордера выдается.

Первый ордер должен выставляться, а счетчик фактически начинает работать после разрешения на выставление первого ордера.

ЛОГИКА:

Если OpenBar!=0 и StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если NOpenBarS(0)=-1, то NOpenBarS(0)=0 и fOpenBarS=0 и StopF=0

Если NOpenBarS(0)=>OpenBar , то NOpenBarS(0)=0 и fOpenBarS=0 и StopF=0

Иначе NOpenBarS(0)=NOpenBarS(1)+1 и fOpenBarS=1 и StopF=1

Для ордеров на покупку:

Если TipOrder=2

Если NOpenBarB(0)=-1, то NOpenBarB(0)=0 и fOpenBarB=0 и StopF=0

Если NOpenBarB(0)=>OpenBar , то NOpenBarB(0)=0 и fOpenBarB=0 и StopF=0

Иначе NOpenBarB(0)=NOpenBarB(1)+1 и fOpenBarB=1 и StopF=1

Если OpenBar!=0 и StopF=1

Для ордеров на продажу:

Если TipOrder=1

то NOpenBarS(0)=NOpenBarS(1)+1 и fOpenBarS=1 и StopF=1

Для ордеров на покупку:

Если TipOrder=2

то NOpenBarB(0)=NOpenBarB(1)+1 и fOpenBarB=1 и StopF=1

**5.2.10 Блок "Фильтр fPropusk"**

Смысл данного фильтра в пропуске сигнала на выставление ордера заданное количество раз, в случае если остальные фильтры пройдены успешно (необходимо понять, какие ордера по счету наиболее доходны)

fPropusk (B/S) – переменная определяющая результат расчета блока (1 – да /0 – нет)

ЛОГИКА:

Если fPropusk!=0 или StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Если fPropusk>0 и High(1)>priceSell и Low(1)<priceSell, то fPropusk=fPropusk-1 и fPropuskB=1 и StopF=1

Иначе fPropuskB=0 и StopF=0

Для ордеров на покупку:

Если TipOrder=2

Если fPropusk>0 и High(1)>priceBuy и Low(1)<priceBuy, то fPropusk=fPropusk-1 и fPropuskS=1 и StopF=1

Иначе fPropuskS=0 и StopF=0

**5.2.9.1 Блок "Фильтрация пройдена (Yes/No)"**

Если следующее равенство верно:

StopF=1 или Cycle=2

то, переходим к пункту № 6.1

если нет, то переходим к пункту № 5.2.9.1

# 5.2.9.2 Блок "Расчет размера лота"

lotM=0;//вариант изменения размера лота при повторном открытии отложенного ордера до достижения maT (до начала нового цикла)

lotV(B/S) – объём лота, который будет выставлен в отложенном ордере.

Советник позволяет использовать следующие варианты увеличения лота в отложенных ордерах:

0 - Постоянный лот "lot"

1 - 1+1 lot+lot+...

2 - 1+2+3+4+...

3 - 1 + 1 = 2

1 + 2 = 3

2 + 3 = 5

3 + 5 = 8

5 + 8 = 13

8 + 13 = 21

13 + 21 = 34

21 + 34 = 55

34 + 55 = 89

4 – происходит расчет лота в зависимости от просадки (требуется описание)

ЛОГИКА:

Если open\_maMirrorZ!=0 или StopF!=1

Для ордеров на продажу:

Если lotM =0, тогда lotVS=lot

Если lotM =1, тогда lotVS=lot+ lot(n)

Если lotM =2, тогда lotVS=lot+ lot(n)+ lot(n-1)

Если lotM =3, тогда lotVS=(формулу точно не знаю, но выше написал результат)

OpenOrderS – укажет какой по счету открывается ордер за цикл (-1)

Для ордеров на покупку:

Если lotM =0, тогда lotVB=lot

Если lotM =1, тогда lotVB=lot+ lot(n)

Если lotM =2, тогда lotVB=lot+ lot(n)+ lot(n-1)

Если lotM =3, тогда lotVB=(формулу точно не знаю, но выше написал результат)

OpenOrderB – укажет какой по счету открывается ордер за цикл (-1)

Необходимо помочь описать логику в формулах.

# 5.2.9.3 Блок "Выставление отложенного ордера"

Выставляется отложенный ордер.

ЛОГИКА:

Если StopF!=1

Для ордеров на продажу:

Если TipOrder=1

Цена продажи: priceSell =maH+pipsXH\*Point

Тейк профит: 0

Стоп лосс: 0

Размер лота: lotVS

Время жизни ордера: (нужна переменная/формула)

Для ордеров на покупку:

Если TipOrder=2

Цена продажи: priceBuy =maL-pipsXL\*Point

Тейк профит: 0

Стоп лосс: 0

Размер лота: lotVB

Время жизни ордера: (нужна переменная/формула)

и переходим к блоку № 5.2.9.4

# 5.2.9.4 Блок "Счетчик выставленных ордеров"

OpenOrder (B/S) – счетчик выставленных отложенных ордеров за цикл.

ЛОГИКА:

Если StopF!=1

Для ордеров на продажу:

Если TipOrder=1

OpenOrderS(0)=OpenOrderS(1)+1

Для ордеров на покупку:

Если TipOrder=2

OpenOrderB(0)=OpenOrderB(1)+1

# 6.1 Блок "Количество открытых ордеров"

Запрашивается сколько в настоящий момент открытых ордеров (по MagicB) – нужно для понятия, сколько ордеров нужно будет обработать в цикле. Алгоритм реализации на усмотрение программиста.

# 6.2 Блок "Проверка на наличие открытого ордера, но необработанного (Yes/No)"

Если все открытые ордера обработаны (Счетчик, из блока № 6.1 = 0), то переходим к блоку № 11

Если нет, то запрашиваем информацию об ордере:

OpenPrice – цена по которой ордер был открыт.

minTP – минимальное значение в пунктах тейк профита (запрашивается у ДЦ)

minSL – минимальное значение в пунктах стоп лосса (запрашивается у ДЦ)

LastTPb – тейк профит на прошлом баре для ордеров на покупку

LastTPs – тейк профит на прошлом баре для ордеров на продажу

LastSLb – стоп лосс на прошлом баре для ордеров на покупку

LastSLs – стоп лосс на прошлом баре для ордеров на продажу

Price – текущая цена инструмента

Point – количество знаков после запятой в инструменте

Если ордер на продажу, тогда TipOrder=1

Если ордер на покупку, тогда TipOrder=2

**6.2.1 Блок "Проверка условия на закрытие открытого ордера "cCycle""**

cCycle(B/S) – переменная для ускорения обработки данных (1 – сигнал на закрытие открытых ордеров/ 0 – пропуск). Т.е. при новом цикле открытые ордера удаляются.

StopC – переменная сигнализирующая о закрытии открытого ордера (StopC=1) на текущем баре.

ЛОГИКА:

Для ордеров на продажу:

Если Cycle=0 и cCycle!=0, то cCycleS=1 и StopC=1

Иначе cCycleS=0 и StopC=0

Для ордеров на покупку:

Если Cycle=0 и cCycle!=0, то cCycleB=1 и StopC=1

Иначе cCycleB=0 и StopC=0

**6.2.2 Блок "Проверка условия на закрытие открытого ордера "cBlockDiap""**

Данный блок использует уже рассчитанные данные пользовательского индикатора.

cBlockDiap (B/S) – возвращает результат фильтрации данных (1 – положительный и 0 – отрицательный). Показывает, попадает ли осциллятор в предварительно заданный в настройках советника диапазон.

ЛОГИКА:

Если BlockDiap=2 и StopC!=1

Для ордеров на продажу:

Если TipOrder=1

Если BlockDiapS=1 , то cBlockDiapS=1 и StopC=1

Иначе cBlockDiapS=0 и StopC=0

Для ордеров на покупку:

Если TipOrder=2

Если BlockDiapB=1 , то cBlockDiapB=1 и StopC=1

Иначе cBlockDiapB=0 и StopC=0

**6.2.3 Блок "Проверка условия на закрытие открытого ордера "cOrderLife""**

cOrderLife (B/S) – возвращает результат фильтрации данных (1 – положительный и 0 – отрицательный). Фактически служит для ограничения времени жизни открытого ордера.

ЛОГИКА:

Если OrderLife!=0 и StopC!=1

Для ордеров на продажу:

Если TipOrder=1

Если (нужен алгоритм – возможно счетчик или запрос), то cOrderLifeS=1 и StopC=1

Иначе cOrderLifeS=0 и StopC=0

Для ордеров на покупку:

Если TipOrder=2

Если (нужен алгоритм – возможно счетчик или запрос), то cOrderLifeB=1 и StopC=1

Иначе cOrderLifeB=0 и StopC=0

**6.2.4 Блок "Проверка условия на закрытие открытого ордера "cmaMirrorZ""**

cOpen\_maMirrorZ (B/S) – переменная определяющая результат расчета блока.

ЛОГИКА:

Если open\_maMirrorZ=2 и StopC!=1

Для ордеров на продажу:

Если TipOrder=1

Если Open(0)>maMirrorZs, то cOpen\_maMirrorZS=1 и StopC=1

Если Open(0)>Levl\_Zerro, то cOpen\_maMirrorZS=1 и StopC=1

Иначе cOpen\_maMirrorZS=0 и StopC=0

Если InsideMA=1

Если maMirrorZs>maH+pipsXH , то StopC=0

Иначе StopC= StopC

Для ордеров на покупку:

Если TipOrder=2

Если Open(0)<Levl\_Zerro, то cOpen\_maMirrorZB=1 и StopC=1

Иначе cOpen\_maMirrorZB=0 и StopC=0

Если InsideMA=1

Если maMirrorZb<maL-pipsXL , то StopC=0

Иначе StopC= StopC

# 6.3 Блок "Есть одно условие по результатам проверки (Yes/No)"

Если следующее равенство верно:

StopC=1

то, переходим к пункту № 6.3

если нет, то переходим к пункту № 8

# 6.4 Блок "Закрытие открытого ордера"

Закрывается обработанный открытый ордер, и переходим к блоку № 6.5

# 6.5 Блок «Метка «Ордер обработан»»

Счетчик, уменьшающий значение из блока № 6.1 – необходим для выхода из цикла.

Делается метка, что осталось меньше не обработанных ордеров – необходимо для выхода из цикла. Алгоритм на усмотрения программиста.

# 7. Блок "Выбор способа расчета тейк профита"

Определяем, по какому методу будет рассчитываться тейк профит:

- Не модифицировать тейк профит (metodTP=0)

- Фиксированный в пунктах (metodTP=1)

- Значение по maT (metodTP=2)

# 7.1 Блок "Расчет по maT"

minTP – минимальное значение в пунктах тейк профита (запрашивается у ДЦ)

TP (b/s) – значение тейк профита

LastTPb – тейк профит на прошлом баре для ордеров на покупку

LastTPs – тейк профит на прошлом баре для ордеров на продажу

Price – текущая цена инструмента

Point – количество знаков после запятой в инструменте

pointTP – расстояния в пунктах для выставления тейк профита

Тейк профит рассчитывается по формуле:

ЛОГИКА:

Если metodTP=1

Для ордеров на продажу:

Если TipOrder=1

Если LastTPs=0, то LastTPs=Price-TPerror\*Point

Иначе если Price>maT-pointTP то TPs=Price-TPerror\*Point

Иначе если Price-minTP<maT-pointTP то TPs=LastTPs

Иначе TPs=maT-pointTP и LastTPs=TPs

Для ордеров на покупку:

Если TipOrder=2

Если LastTPb=0, то LastTPb=Price+TPerror\*Point

Иначе если Price<maT+pointTP, то TPs= Price+TPerror\*Point

Иначе если Price+minTP>maT+pointTP, то TPs=LastTPb

Иначе TPb=maT+pointTP и LastTPb=TPb

# 7.2 Блок "Расчет по фиксированному тейк профиту"

TP(b/s) – значение тейк профита

OpenPrice – цена по которой ордер был открыт.

Тейк профит рассчитывается по формуле:

ЛОГИКА:

Если metodTP=2

Для ордеров на продажу:

Если TipOrder=1

TPb= OpenPrice-pointTP\*Point

Для ордеров на покупку:

Если TipOrder=2

TPs= OpenPrice+pointTP\*Point

# 8. Блок "Выбор способа расчета стоп лосса"

Определяем, по какому методу будет рассчитываться стоп лосс:

- Не модифицировать стоп лосс (metodSL=0)

- Фиксированный в пунктах от текущей цены (metodSL=1)

- Значение по maMirrorZ (metodSL=2)

# 8.1 Блок "Фиксированный – от текущей цены "

TP (b/s) – значение стоп лосса

LastSLb – стоп лосс на прошлом баре для ордеров на покупку

LastSLs – стоп лосс на прошлом баре для ордеров на продажу

ЛОГИКА:

Если metodSL=1

Для ордеров на продажу:

Если TipOrder=1

Если LastSLs>price+pointSL\*Point , то SLs=price+pointSL\*Point

LastSLs= price+pointSL\*Point

Иначе SLs=LastSLs

Для ордеров на покупку:

Если TipOrder=2

Если LastSLb<price-pointSL\*Point , то SLb=price-pointSL\*Point

LastSLb= price-pointSL\*Point

Иначе SLb=LastSLb

# 8.2 Блок "Не модифицировать стоп лосс"

OpenSL – стоп лосс при ордера, указанный ранее в отложенном ордере.

SL(b/s) – стоп лосс

ЛОГИКА:

Если metodSL=0

Для ордеров на продажу:

Если TipOrder=1

То SLs= OpenPrice+pointSL\*Point

Для ордеров на покупку:

Если TipOrder=2

То SLb= OpenPrice-pointSL\*Point

Или просто не модифицировать стоп лосс.

# 8.3 Блок "По maMirrorZ"

ЛОГИКА:

Если metodSL=2

Для ордеров на продажу:

Если TipOrder=1

Если maMirrorZs<OpenPrice, то SLs= SLerror\*Point

Если LastSLs>maMirrorZs , то SLs=maMirrorZs

LastSLs=maMirrorZs

Иначе SLs= LastSLs

Если InsideMA=1

Если maMirrorZs>maH+pipsXH , SLs=0

Иначе SLs=SLs

Если maMirrorZs<OpenPrice, то SLs= SLerror\*Point

Для ордеров на покупку:

Если TipOrder=2

Если LastSLb<maMirrorZb , то SLb= maMirrorZb

LastSLb= maMirrorZb

Иначе SLb=LastSLb

Если InsideMA=1

Если maMirrorZb<maL-pipsXL , то SLb=0

Иначе SLb=SLb

Если maMirrorZs>OpenPrice, то SLb= SLerror\*Point

# 8.3.1 Блок "Ограничение Levl\_Zerro"

Ограничение по «зажатию» стоп лосса близко к цене.

OpenPrice – цена по которой ордер был открыт.

ЛОГИКА:

Если Levl\_Zerro!=0

Для ордеров на продажу:

Если TipOrder=1

Если OpenPrice + Levl\_Zerro\*Point=>SLs, то SLs=Levl\_Zerro\*Point

Иначе SLs=SLs

Для ордеров на покупку:

Если TipOrder=2

Если OpenPrice-Levl\_Zerro\*Point<=SLb, то SLb=Levl\_Zerro\*Point

Иначе SLb=SLb

# 9. Блок "Расчет условия для перевода стоп лосса "в безубыток""

Bezubitok – переменная, определяющая какой метод выставления стоп лосса будет использоваться.

Варианты:

1. Без перевода в безубыток – 0;
2. От цены открытия ордера – цена открытия +- дельта цены открытия бара;
3. Если цена за заданное количество баров не пересекала цену открытия ордера;
4. Если до maT осталось заданное расстояние

# 9.1 Блок "Перевод в безубыток в пунктах от цены открытия"

Переводит стоп лосс в безубыток при прохождении цены на прошлом баре заданное количество пунктов от цены открытия ордера.

ЛОГИКА:

Если Bezubitok=1

Для ордеров на продажу:

Если TipOrder=1

Если Low(1)<=OpenPrice-pipsXT и SLs>OpenPrice-pipsXT, то SLs=OpenPrice-pipsXT

Иначе SLs=SLs

Для ордеров на покупку:

Если TipOrder=2

Если High(1)>=OpenPrice+pipsXT и SLb<OpenPrice+pipsXT, то SLb=OpenPrice+pipsXT

Иначе SLb=SLb

# 9.2 Блок "Перевод в безубыток если цена не пересекала цену открытия"

Переводит стоп лосс в безубыток, если Hig(X)/Low(X) не пересекал заданное количество раз цену открытия ордера (OpenPrice).

Алгоритм от программиста.

# 9.3 Блок "Перевод в безубыток по maT"

Переводит стоп лосс в безубыток, если до maT осталось заданное количество пунктов.

ЛОГИКА:

Если Bezubitok=3

Для ордеров на продажу:

Если TipOrder=1

Если Low(1)>=maTm+pipsXT и SLs>maTm+PipsXT, то SLs=OpenPrice

Иначе SLs=SLs

Для ордеров на покупку:

Если TipOrder=2

Если High(1)<=maTm-pipsXT и SLb<maTm-PipsXT, то SLb=OpenPrice

Иначе SLb=SLb

# 10. Блок "Модификация открытого ордера"

Модифицируем открытый ордер.

ЛОГИКА:

Если StopC!=1

Для ордеров на продажу:

Если TipOrder=1

Цена продажи: priceSell =maH+pipsXH\*Point

Тейк профит: TPs

Стоп лосс: SLs

Размер лота: lotVB

Время жизни ордера: (нужна переменная/формула)

Для ордеров на покупку:

Если TipOrder=2

Цена продажи: priceBuy =maL-pipsXL\*Point

Тейк профит: TPb

Стоп лосс: SLb

Размер лота: lotVB

Время жизни ордера: (нужна переменная/формула)

и переходим к блоку № 6.2

# 11. Блок "Использовать страховочный ордер (Yes/No)""

Если magicB!=0 , то будет использован «страховочный» ордер.

# 12. Блок "Расчет страховочного ордера""

Производится расчет страховочного ордера (алгоритм в разработке).

ЛОГИКА:

Если magicB!=0

Для ордеров на покупку:

Если TipOrder=1

то priceBuyB=CyclePrice+UPpips\*Point, TPstraxB=maT

Для ордеров на продажу:

Если TipOrder=2

то priceSellB=CyclePrice-DOWNpips\*Point, TPstraxS=maT

# 13. Блок "Выставление страховочного ордера""

Выставляем отложенный «страховочный» ордер.

ЛОГИКА:

Если magicB!=0

Для ордеров на покупку:

Если TipOrder=1

Цена покупки: priceBuyB

Тейк профит: TPstraxS

Стоп лосс: 0

Размер лота: lotVS\* Ktrend\*ОКРУГЛЕНИЕ

Время жизни ордера: (нужна переменная/формула)

Для ордеров на продажу:

Если TipOrder=2

Цена продажи: priceSellB

Тейк профит: TPstraxB

Стоп лосс: 0

Размер лота: lotVB\* Ktrend\*ОКРУГЛЕНИЕ

Время жизни ордера: (нужна переменная/формула)

и переходим к блоку № 2.1

СУПЕР ПАТИ! Дома у Феде, ПЕРВОГО ЧИСЛА. Приходите все, отмечаем первый день лето! РОВНО В СЕМЬ ВЕЧЕРА, встречаемся у арки! Все включено, приходите будет круто!!!